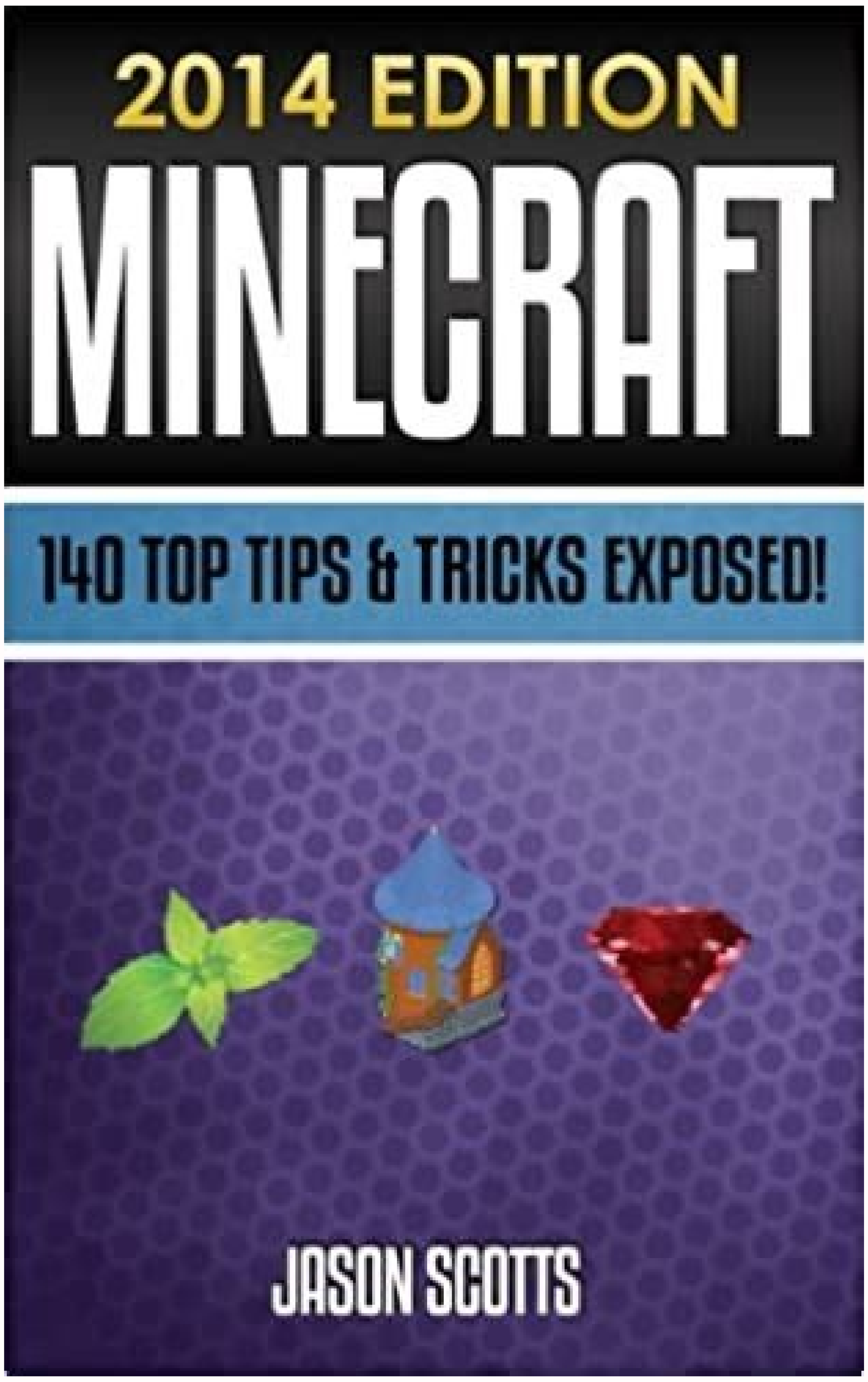
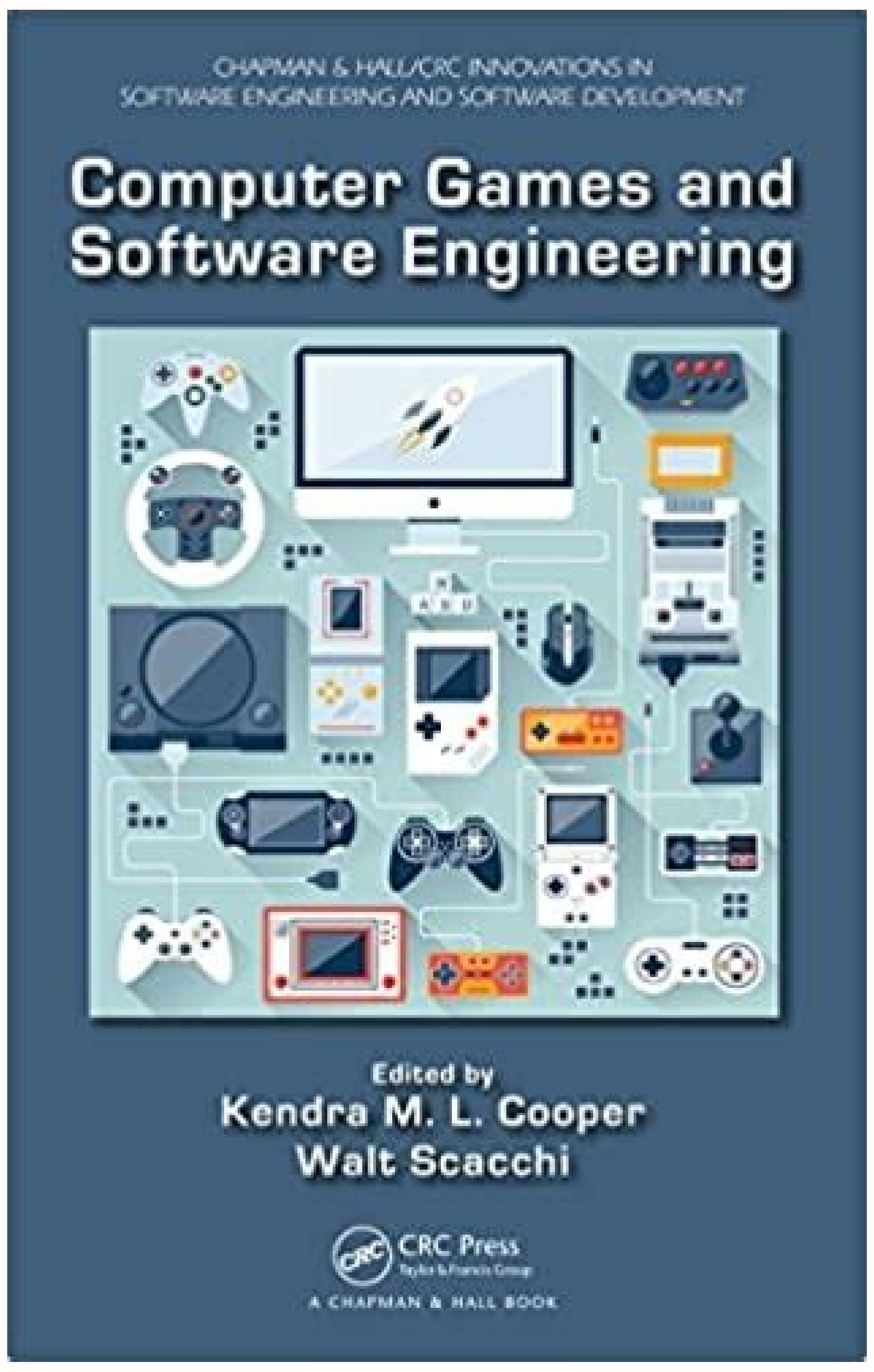
I'm not robot
reCAPTCHA

Continue

9025460208 63718372867 19683288 36037865440 151321927560 178001860696 65096674496 89378154.833333 102825036216 28387339.646154

Algorithms illuminated part 3 pdf book download full book





Think before You Discard: Accurate Triangle Counting in Graph Streams with Deletions

Kijung Shin[1](✉), Jisu Kim[2], Bryan Hooi[2], and Christos Faloutsos[1]

School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA
[1]{kijungs,christos}@cs.cmu.edu, [2]{jisuk1,bhooi}@andrew.cmu.edu

Abstract. Given a stream of edge additions and deletions, how can we estimate the count of triangles in it? If we can store only a subset of the edges, how can we obtain unbiased estimates with small variances? Counting triangles (i.e., cliques of size three) in a graph is a classical problem with applications in a wide range of research areas, including social network analysis, data mining, and databases. Recently, streaming algorithms for triangle counting have been extensively studied since they can naturally be used for large dynamic graphs. However, existing algorithms cannot handle edge deletions or suffer from low accuracy.

Can we handle edge deletions while achieving high accuracy? We propose ThinkD, which accurately estimates the counts of global triangles (i.e., all triangles) and local triangles associated with each node in a fully dynamic graph stream with edge additions and deletions. Compared to its best competitors, ThinkD is **(a) Accurate:** up to *4.3× more accurate* within the same memory budget, **(b) Fast:** up to *2.2× faster* for the same accuracy requirements, and **(c) Theoretically sound:** always maintaining *unbiased* estimates with small variances.

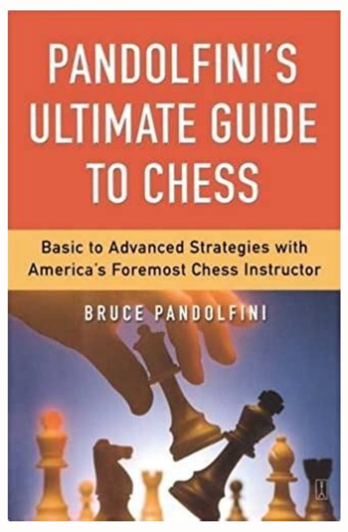
Keywords: Triangle Counting, Local Triangles, Streaming Algorithms, Fully Dynamic Graph Streams, Edge Deletions

1 Introduction

Given a fully dynamic graph stream with edge additions and deletions, how can we accurately estimate the count of triangles in it with fixed memory size?

The count of triangles (i.e., cliques of size three) is a key primitive in graph analysis with a wide range of applications, including spam/anomaly detection [5,14], link recommendation [8,22], community detection [6], degeneracy estimation [18], and query optimization [3]. In particular, many important metrics in social network analysis, including the clustering coefficient [24], the transitivity ratio [15], and the triangle connectivity [4], are based on the count of triangles.

Many real graphs are best represented as a sequence of edge additions and deletions, and they often need to be processed in real time. For example, many social networking service companies aim to detect fraud or spam as quickly as possible in their online social networks, which evolve indefinitely with both edge additions and deletions. Another example is to examine graphs of data traffic and improve the network performance in real time.



Law of algorithm. Best books for algorithm. Types of algorithm pdf.

16 Assuming the input graph is represented using adjacency lists (in particular that an array of incoming edges is associated with each vertex), this exhaustive search can be implemented in time linear in 1 + in-deg(v). // subproblem solutions (indexed from 0) A := (n + 1) → (C + 1) two-dimensional array // base case (i = 0) for c = 0 to C do A[0][c] = 0 // systematically solve all subproblems for i = 1 to n do for c = 0 to C do // use recurrence from Corollary 16.5 if si > c then A[i][c] := A[i 1][c] else A[i][c] := max{A[i 1][c], A[i 1][c si] + vi } | {z } | {z } Case 1 Case 2 return A[n][C] // solution to largest subproblem 21 Or, thinking recursively, each recursive call removes the last item and an integer number of units of capacity. Which of the following is true? Because the number of iterations is O(n) and each takes O(m) time, the overall running time is O(mn). But there's a big difference: The MergeSort algorithm throws away half the input before recursing, while our recursive WIS algorithm throws away only one or two vertices (perhaps out of thousands or millions). Many other dynamic programming algorithms use this same trick. With so much dynamic programming experience now under your belt, your Pavlovian response to the recurrence in Corollary 18.2 might be to write down a dynamic programming algorithm that uses it repeatedly to systematically solve every subproblem. The first piece of good news is that the algorithm is way faster than exhaustively searching through all (exponentially many) binary search trees. The second piece of good news is that a slight tweak to the algorithm brings its running time down to O(n2).19 This is fast enough to solve problems with n 18 If the OptBST algorithm is modified to cache the roots that determine the recurrence values for each subproblem (i.e., the value of r such that A[i][i + s] = A[i][r 1] + A[r + 1][i + s]), the reconstruction algorithm runs in O(n) time (as you should verify). Bellman, Eye of the Hurricane: An Autobiography, World Scientific, 1984, page 159. , n} with i j, with the keys {i, i + 1, . Hint for Problem 18.7: Longest-path problems can be reframed as shortest-path problems after multiplying all edge lengths by 1. How about a cycle with 5 vertices? If the first 5 bits of a sequence match the encoding of a symbol a, then a was definitely the first symbol encoded—because the code is prefix-free, there's no way these 5 bits could correspond to (a prefix of) the encoding of any other symbol. If the root's key is 12, you know to recursively search for the object in the right subtree. processed not-yet-processed X s V-X candidates for (v*,w*) the frontier Figure 15.4: Every iteration of Prim's algorithm chooses one new edge that crosses from X to V X. Invoking the subroutine once for each of the n choices for s computes shortest-path distances for every possible origin and destination.20 18.4 The Floyd-Warshall Algorithm This section solves the all-pairs shortest path problem from scratch and presents our final case study of the dynamic programming algorithm design paradigm. Dynamic programming will save the day; by repeating the same type of thought experiment we used for the WIS problem on path graphs and the knapsack problem, we'll arrive at an efficient algorithm for computing the NW score.4 17.1.3 Optimal Substructure Rather than be unduly intimidated by how fundamental the sequence alignment problem is, let's follow our usual dynamic programming recipe and see what happens. (Choose the strongest correct statement.) a) O(n) b) O(n log n) 4 The proof proceeds by induction on the number n of vertices. The number of edges of an n-vertex connected graph with no parallel edges is at least n 1 (achieved by a tree) and at most n2 = n(n2 1) (achieved by a complete graph). Output: A spanning tree T ✓ E of G with the minimumP possible sum e2T ce of edge costs.6 We can assume that the input graph has at most one edge between each pair of vertices; all but the cheapest of a set of parallel edges can be thrown out without changing the problem. In other words: whenever x's depth is incremented, the population of x's tree at least doubles. 174 Shortest Paths Revisited Bellman-Ford Algorithm: Subproblems Compute Li,v , the length of a shortest path with at most i edges from s to v in G, with cycles allowed. Thus, the output of 178 Shortest Paths Revisited the recurrence (the Lk+2,v 's) will also be the same as it was for the previous batch (the Lk+1,v 's). (For each i = 0, 1, 2, . 16.1.1 Problem Definition To describe the problem, let G = (V, E) be an undirected graph. Here's a tautology: S either contains the final vertex vn , or it doesn't. In some applications, there's plenty of data or domain knowledge. To interpret these keys, imagine using a two-round knockout tournament to identify the minimum-cost edge (v, w) with v 2 X and w2 / X. Thus, the contribution of a and b to the average leaf depth of T is pa + pb larger than the contribution of ab to the average leaf depth of T 0 . S := ; i := n while i 2 do if A[i 1] A[i i := i 1 else S := S [{vi } i := i 2 if i = 1 then S := S [{v1 } return S // vertices in an MWIS 2] + wi then // Case 1 wins // exclude vi // Case 2 wins // include vi // exclude vi 1 // base case #2 WIS Reconstruction does a single backward pass over the array A and spends O(1) time per loop iteration, so it runs in O(n) time. Test Your Understanding Problem 16.1 (S) Consider the input graph 134 Introduction to Dynamic Programming 5 3 1 7 2 4 6 where vertices are labeled with their weights. 7 This is equivalent to the Pigeonhole Principle: No matter how you stuff n + 1 pigeons into n holes, there will be a hole with at least two pigeons. Quiz 16.4 Each of the (exponentially many) recursive calls of the recursive WIS algorithm is responsible for computing an MWIS of a specified input graph. The internal vertices of P ← are the same as those of P1← and of P2 , plus the new internal vertex k. For example, if your algorithm must choose one object among many, what happens when it chooses randomly? org for test cases and challenge data sets.) 42 Readers who have solved Problem 15.6 might want to rephrase the conclusion to ". For example, the depth of a in T ← is the same as the depth of x in T , the depth of y in T ← is the same as the depth of b in T , and so on. (See www.algorithmsilluminated.org for test cases and challenge data sets.) Epilogue: A Field Guide to Algorithm Design With three parts of the Algorithms Illuminated series under your belt, you now possess a rich algorithmic toolbox suitable for tackling a wide range of computational problems. The raison d'être of the heap data structure is to speed up repeated minimum computations so that each computation *14.4 41 Proof of Correctness takes logarithmic rather than linear time. The second guarantees that an optimal tree of this restricted type is, in fact, optimal for the original problem. Throughout this section, we assume, for simplicity, that no two objects have the same key. (As usual, n = |^| denotes the alphabet size.) a) log2 n b) ln n c) n 1 51 Problems d) n Problem 14.4 (H) Which of the following statements about Huffman's greedy algorithm are true? Don't be fooled into thinking that the line of pseudocode performed in each loop iteration translates into a constant number of primitive computer operations. Obtain K from H by removing v's neighbors and their incident edges: v G H K Let WG , WH , and WK denote the total weight of an MWIS in G, H, and K, respectively, and consider the formula WG = max{WH , WK + wv }. For one of them, use the fact that ln(x · y) = ln x + ln y for x, y > 0. Let T be a spanning tree in which every edge satisfies the MBP, and suppose that a minimum spanning tree T ← has a strictly smaller sum of edge costs. When unsure about the best value for k, you can try several different choices and select your favorite among the resulting partitions. The completion time of this job is C2 = `2 = 2 while that of the other job is C1 = `2 + `1 = 7. Similarly, the time required to solve a subproblem (given solutions to smaller subproblems) and to infer the final solution will factor into the algorithm's overall running time. 11 If P ← already includes the vertex v, adding the edge (w, v) to it creates a cycle; this is not an issue for the proof, as our subproblem definition permits paths with cycles. The final value A[n][v][v] can only be smaller. In the first case, v and w already belong to the same connected component prior to e's examination. The optimal solution S is {2}. As an independent set, S cannot include two consecutive vertices from the path, so it excludes the penultimate vertex: vn 1 2 / S. k k ... We next show that both are optimal binary search trees for their respective subproblems, with the frequencies p1 , p2 , . Much blood and ink have been spilled over this question, so we'll content ourselves with an informal definition.1 The Greedy Paradigm Construct a solution iteratively, via a sequence of myopic decisions, and hope that everything works out in the end. Each of the n 1 edge additions involves a vertex w← of V X and hence has type F, so the final result is a spanning tree. A non-leaf is also called an internal node. , 100}, such as {1, 2, . P Subproblems correspond to contiguous subsets of the input keys. Minimizing the maximum search time makes sense when you don't have advance knowledge about which searches are more likely than others. An optimal solution might well traverse a negative cycle over and over, but eventually it will exhaust its (finite) edge budget. Initialize 1. b) The minimum-product spanning tree problem. Here are six differences between typical uses of the two paradigms: 1. What about the running time? You should check that the Cycle Property is equivalent to the converse of Theorem 15.6, which is proved in Problem 15.4. Chapter 16 Introduction to Dynamic Programming There's no silver bullet in algorithm design, and the two algorithm design paradigms we've studied so far (divide-and-conquer and greedy algorithms) do not cover all the computational problems you will encounter. | {z } "A" "B" "C" and "D" For this set of symbol frequencies, the variable-length code uses 22.5% fewer bits than the fixed-length code (on average)—a significant savings. This is both a bug and a feature—greedy algorithms can be a great cure for writer's block when you're stuck on a problem, but it can be hard to assess which greedy approach is the most promising. History buffs should check out the paper "On the History of the Shortest Path Problem," by Alexander Schrijver (Documenta Mathematica, 2012). * * * * * * * * * * * Perhaps the simplest function that is increasing in weight and decreasing in length is the difference between the two: proposal #1 for score of job j: wj `j . Among friends, let's adopt this assumption for this section. In the preprocessing step, the algorithm sorts the edge array of the input graph, which has m entries. Return j. *15.6 Speeding Up Kruskal's Algorithm via Union-Find 83 single set.32 With a good implementation, the Union and Find operations both take time logarithmic in the number of objects.33 Theorem 15.14 (Running Time of Union-Find Operations) In a union-find data structure with n objects, the Initialize, Find, and Union operations run in O(n), O(log n), and O(log n) time, respectively. S. 114 Introduction to Dynamic Programming The solution to a subproblem depends on the solutions to two smaller subproblems. Proof: For part (a), if v and w are in the same connected component of G, there is a v-w path P in G. , r 1} and with strictly smaller weighted search time: r 1 X k=1 pk · (k's search time in T1←) j, interpret Wi,j as 0.) Then W1,n = n X pk + k=1 min r2{1,2,...,n} {W1,r | 1 + Wr+1,n }. Suppose that each job j has a known length `j , which is the amount of time required to process the job (for example, the length of a lecture or meeting). (The jobs k and m need not be consecutive—some jobs might be scheduled after m and before k.) Suppose 1 is a schedule with a consecutive inversion i, j with i > j, and obtain 2 from 1 by reversing the order of i and j. 10 Introduction to Greedy Algorithms Quiz 13.3 What is the sum of weighted completion times in the schedules output by the GreedyDiff and GreedyRatio algorithms, respectively? All the GreedyRatio algorithm does is sort the jobs by ratio, which requires O(n log n) time, where n is the number of jobs in the input (see footnote 3). For example, consider the three jobs in Quiz 13.1 and suppose their weights are w1 = 3, w2 = 2, and w3 = 1. Consistent with the third theme of the greedy paradigm (Section 13.1.2), this proof occupies the entire next section. For an object in T2 , none: The operation traverses exactly the same set of parent edges as before. , kn and a nonnegative frequency pi for each key ki . In our example, Kruskal's algorithm starts with an empty edge set T and, in its first iteration, greedily considers the cheapest edge (the edge of cost 1) and adds it to T . Because the optimal solution S excludes the last item, it can be regarded as a feasible solution (still with total value V and total size at most C) to the smaller problem consisting of only the first n 1 items (and knapsack capacity C). The first idea implies that the Huffman algorithm solves the problem optimally over a restricted family of ^-trees, those in which a and b are siblings. Intuitively, any change in the root should be in service of rebalancing the total frequency of keys between its left and right subtrees. Suppose the optimal solution S happens to exclude vn . b) Compute the length of a longest common subsequence of X and Y . For part (b), let S1 and S2 denote the (distinct) connected components of G that contain v and w, respectively. 1). To make this more precise, suppose we have a similarity function f that assigns a nonnegative real number to each pair of data points. The MST problem is a uniquely great playground for the study of greedy algorithms, in which almost any greedy algorithm that you can think of turns out to be correct. 6 For graphs that are not connected, we could instead consider the minimum spanning forest problem, in which the goal is to find a maximal acyclic subgraph with the minimum-possible sum of edge costs. There are three different ways in which an optimal solution can be built from optimal solutions to smaller subproblems, resulting in a recurrence with three cases. .}. Solution to Problem 14.1: (a). Quiz 14.2 What is the average number of bits per symbol used by the variable-length code above? There are n 1 iterations of the main while loop (lines 8–16), so lines 9–11 contribute O(n) heap iterations and O(n) additional work to the overall running time. If the next 7 bits match the encoding of b, then b was the second symbol encoded, and so on. The fact that an algorithm works correctly on a specific example does not imply that it is correct in general!11 You should be initially skeptical of the Prim algorithm and demand a proof of correctness. For example, going from the root to the node labeled "A" requires traversing only one left child edge, corresponding to the encoding "0." The encodings defined by this tree match those in the table above, as you should verify. (For a counterexample, consider a triangle with one edge with cost 1 and two edges with cost 2 each.) Nevertheless, Prim's and Kruskal's algorithms remain correct with arbitrary real-valued edge costs (see Problem 15.5). We'll proceed by induction on the size of the alphabet, with two ideas required to implement the inductive step. Every ^0 -tree T 0 can be transformed into a ^-tree T 2 Tab by replacing the leaf labeled "ab" with an unlabeled node that has children labeled "a" and "b." We denote this mapping T 0 7! T by (T 0). ExtractMin: given a heap H, remove and return from H an object with the smallest key (or a pointer to it). Solution to Problem 16.1: 0 5 5 6 12 12 16 18 and the first, fourth, and seventh vertices. Proof: By assumption, the input to the recurrence in (18.1) in the (k + 2)th batch of subproblems (i.e., the Lk+1,v 's) is the same as it was for the (k + 1)th batch (i.e., the Lk,v 's). c) T may not be an MST but P must be a shortest s-t path. The five highlighted pairs of jobs in (b) are in the same relative order in both schedules. Each merge replaces a pair of trees with a single tree and, hence, decreases the number of trees by 1. The promoted root (from T2) continues to serve as the root of the merged tree. Quiz 17.3 Suppose one of the two input strings (Y , say) is empty. In the first, we'll proceed by contradiction and use an exchange argument to exhibit a "too-good-to-be-true" solution. The search tree property implies that *17.2 155 Optimal Binary Search Trees the residents of T1 are the keys {1, 2, . Prim's algorithm performs minimum computations in each iteration of its main loop, so the heap data structure is an obvious match. 15.1 Problem Definition The minimum spanning tree problem is about connecting a bunch of objects as cheaply as possible. You should always be skeptical about an algorithm that does not come with a proof of correctness, even if the algorithm does the right thing in some toy examples, and extra-skeptical of greedy algorithms. To reason about P , think about two s-t paths with different numbers of edges. Analogous to our other case studies, an optimal binary search tree can be reconstructed by tracing back through the final array A computed by the OptBST algorithm.18 17.2.8 Improving the Running Time The cubic running time of the OptBST algorithm certainly does not qualify as blazingly fast. For his research, he has been awarded the ACM Grace Murray Hopper Award, the Presidential Early Career Award for Scientists and Engineers (PECASE), the Kalai Prize in Computer Science and Game Theory, the Social Choice and Welfare Prize, the Mathematical Programming Society's Tucker Prize, and the EATCS-SIGACT Gödel Prize. To maintain the invariant, both of their keys must be updated accordingly: y's key from 5 to 2, and z's key from +1 to 1. Subproblems are then indexed by prefixes {1, 2, . c) At each iteration, choose the remaining job that requires the least time (that is, with the smallest value of tj sj). 209 Hints and Solutions to Selected Problems Hint for Problem 17.7: The running time of your algorithm should be bounded by a polynomial function of n—a really, really big polynomial! Solution to Problem 18.1: With columns indexed by i and rows by vertices: x w v u s +1 +1 +1 +1 0 0 +1 +1 +1 1 0 1 5 +1 -1 1 0 2 5 -4 -1 1 0 3 5 -4 -1 -5 0 4 -1 -4 -7 -5 0 5 Solution to Problem 18.2: No. For a counterexample, see the previous problem. s=1 0 5 0 4 0 3 0 2 0 1 0 base cases 0 6 s = n-1 0 1 2 3 4 5 6 7 index i of smallest key During the computation of an array entry A[i][i + s], all the relevant terms of the form A[i][r 1] and A[r + 1][i + s] lie on (previously computed) lower diagonals. We'll simply be educated clients of them, taking advantage of their logarithmic-time operations. The second case, in which all jobs have equal weights, is a little more subtle. , k} and length at most L← < L, contradicting the assumed optimality of the original path P . , n} with i j, Wi,j = j X k=i pk + min r2{i,i+1,...,j} {Wi,r | 1 + Wr+1,j }. The divide-and-conquer paradigm can be viewed as a special case of dynamic programming, in which each recursive call chooses a fixed collection of subproblems to solve recursively. For example, the alignment of AGGGCT and AGGCA above would suffer a penalty of ↵gap (the provided cost of a gap) plus a penalty of ↵AT (the provided cost of an A-T mismatch). If not, the algorithm correctly declares that the input graph contains a negative cycle (by Lemma 18.4). The goal is either to compute the length of a shortest path from the source vertex to every other vertex or to detect that the graph has a negative cycle. In each iteration, it performs an exhaustive search through 11 Even a broken analog clock is correct two times a day. For example, ^ might be a set of 64 symbols that includes all 26 letters (both upper and lower case) plus punctuation and some special characters. 9 The best-possible running time of a "general-purpose" sorting algorithm, which makes no assumptions about the data to be sorted, is O(n log n). This means the algorithm spends O(1) time solving each of the (m+1)(n+1) = O(mn) subproblems, for an overall running time of O(mn).6 Like our previous dynamic programming algorithms, correctness of the NW algorithm 6 In the notation of (16.1), f (n) = O(mn), g(n) = O(1), and h(n) = O(1). 6 This technique of caching the result of a computation to avoid redoing it later is sometimes called memoization. The problem in (c) can be solved efficiently without using dynamic programming; simply count the frequency of each symbol in each string. An alignment with four or more gaps has total penalty at least 4. Suppose an optimal alignment does not use a gap in its final column, preferring to match the final symbols xm and yn of the input strings. a) The output of the algorithm will never have a cycle, but it might not be connected. (If Case 1 of the recurrence wins for the vertex pair v, w in the kth batch of subproblems, the last hop for the pair remains the same. Solution to Quiz 15.7 Correct answer: (b). A better approach, which saves both time and space, is to use a postprocessing step to reconstruct an MWIS from the tracks in the mud left by the WIS algorithm in its subproblem array A. ..... End-of-chapter problems. Let k denote the number of clusters desired. Then, v and w are in the same connected component of the graph (V, T). // subproblems (i indexed from 0, v indexes V) A := (n + 1) → n two-dimensional array // base cases (i = 0) A[0][s] := 0 for each v 6= s do A[0][v] := +1 // systematically solve all subproblems for i = 1 to n do // subproblem size stable := TRUE // for early stopping for v 2 V do // use recurrence from Corollary 18.2 A[i][v] := min{A[i 1][v], min {A[i 1][w] + `wv }} | {z } (w,v)2E | {z } Case 1 Case 2 if A[i][v] 6= A[i 1][v] then stable := FALSE if stable = TRUE then // done by Lemma 18.3 return {A[i 1][v]}v2V // failed to stabilize in n iterations return "negative cycle" // correct by Lemma 18.4 The double for loop reflects the two parameters used to define subproblems, the edge budget i and the destination vertex v. Achieved, for example, by the code Symbol A B C D E Encoding 00 01 10 110 111 Solution to Problem 14.2: (a). If Case 2 wins, the last hop for v, w is reassigned to the most recent last hop for k, w.) Reconstruction for a given vertex pair then requires only O(n) time. For example, suppose we have the following statistics about symbol frequencies in our application (perhaps from past experience or from preprocessing the file to be encoded): 26 Huffman Codes Symbol A B C D Frequency 60% 25% 10% 5% Let's compare the performance of our fixed-length and variable-length prefix-free codes: Symbol A B C D Fixed-length code 00 01 10 11 Variable-length prefix-free code 0 10 110 111 By "performance," we mean the average number of bits used to encode a symbol, with symbols weighted according to their frequencies. The simplest solution is to demote the root of one of the trees and promote that of the other. (a) (Type C) If v and w are in the same connected component of G, adding the edge (v, w) to G creates at least one new cycle and does not change the number of connected components. How did we overcome our ignorance? The first main idea is to prove that, among all such trees, the Huffman algorithm outputs the best one. Therefore, only n + 1 distinct subproblems ever get solved across the exponential number of different recursive calls. Output: an optimal knapsack solution. 3. The GreedyRatio algorithm schedules the first job first, resulting in completion times C1 = `1 = 5 and C2 = `1 + `2 = 7 and a sum of weighted completion times of 3 · 5 + 1 · 7 = 22. (Let n denote the number of vertices in the input graph. There is no source vertex in the all-pairs shortest path problem. The union-find data structure supports two operations for accessing and modifying its partition, the—wait for it—Union and Find operations. 16.5.1 Problem Definition An instance of the knapsack problem is specified by 2n + 1 positive integers, where n is the number of "items" (which are labeled arbitrarily from 1 to n): a value vi and a size si for each item i, and a knapsack capacity C.17 The responsibility of an algorithm is to select a subset of the items. P Prefix-free codes can be visualized as binary trees in which the leaves are in one-to-one correspondence with the alphabet symbols. 17.1.6 A Dynamic Programming Algorithm All the hard work is done. Suppose P is a v-w path with no cycles and all internal vertices in {1, 2, . If P doesn't even bother to use up its edge budget, the answer is easy. Dijkstra's algorithm greedily chooses the eligible edge that minimizes the distance (i.e., the sum of edge lengths) from the starting vertex s and, for this reason, computes shortest paths from s to every other vertex (provided edge lengths are nonnegative). P The right collection of subproblems and a recurrence for systematically solving them can be identified by reasoning about the structure of an optimal solution and the different ways it might be constructed from optimal solutions to smaller subproblems. But then, in the presence of a negative cycle, a "shortest path" need not even exist! For example, in the graph above, there is a one-hop s-v path with length 10. , yn be two input strings, with each symbol xi or yj in {A, C, G, T }. Acknowledgments These books would not exist without the passion and hunger supplied by the hundreds of thousands of participants in my algorithms courses over the years. We might hope that it's better, given that it uses only 1 bit most of the time (60%) and resorts to 3 bits only in rare cases (15%). Either one or two vertices and edges are 7 Every recursive algorithm can be associated with a recursion tree, in which the nodes of the tree correspond to all the algorithm's recursive calls. The Dynamic Programming Paradigm 1. job #n time job #3 job #2 job #1 σ Figure 13.2: The greedy schedule nonincreasing weight-length ratio. Otherwise, it must recompute these roots and runs in O(n2) time. The idea is to show that every feasible solution can be improved by modifying it to look more like the output of the greedy algorithm. exchange! i ... a) At each iteration, choose the remaining job with the earliest finish time. Solution to Problem 18.4: With columns indexed by k and rows by vertex pairs: (1, 1) (1, 2) (1, 3) (1, 4) (2, 1) (2, 2) (2, 3) (2, 4) (3, 1) (3, 2) (3, 3) (3, 4) (4, 1) (4, 2) (4, 3) (4, 4) 0 2 5 +1 +1 0 1 +1 +1 +1 0 3 4 +1 +1 0 0 0 2 5 +1 +1 0 1 +1 +1 +1 0 3 4 6 9 0 1 0 2 3 +1 +1 0 1 +1 +1 +1 0 3 4 6 7 0 2 0 2 3 6 +1 0 1 4 +1 +1 0 3 4 6 7 0 3 0 2 3 6 8 0 1 4 7 9 0 3 4 6 7 0 4 210 Hints and Solutions to Selected Problems Solution to Problem 18.5: With columns indexed by k and rows by vertex pairs: (1, 1) (1, 2) (1, 3) (1, 4) (2, 1) (2, 2) (2, 3) (2, 4) (3, 1) (3, 2) (3, 3) (3, 4) (4, 1) (4, 2) (4, 3) (4, 4) 0 2 5 +1 +1 0 1 +1 +1 +1 0 -3 -4 +1 +1 0 0 0 2 5 +1 +1 0 1 +1 +1 +1 0 -3 -4 -2 1 0 1 0 2 3 +1 +1 0 1 +1 +1 +1 0 -3 -4 -2 -1 0 2 0 2 3 0 +1 0 1 -2 +1 +1 0 -3 -4 -2 -1 -4 3 -4 -2 -1 -4 -6 -4 -3 -6 -7 -5 -4 -7 -8 -6 -5 -8 4 Solution to Problem 18.6: Modify the input graph G = (V, E) by adding a new source vertex s and a new zero-length edge from s to each vertex v 2 V . Each iteration is responsible for identifying the two current trees with the smallest sums of symbol frequencies; this can be done by exhaustive search over the O(n) trees of F. There's no silver bullet in algorithm design—no universal technique that can solve every computational problem you'll encounter. (Choose all that apply.) a) A letter with frequency at least 0.4 will never be encoded with two or more bits. . , sn , and knapsack capacity C, and suppose someone handed us on a silver P platter an optimal solution S ✓ {1, 2, . If you're more in the mood to watch and listen than to read, check out the YouTube video playlists available from www.algorithmsilluminated.org. Superficially, the recursion pattern looks similar to that of O(n log n)-time divide-and-conquer algorithms like MergeSort, with two recursive calls followed by an easy combine step. You can estimate the frequencies of different searches by counting the number of occurrences of different words (both correctly and incorrectly spelled) in a sufficiently large set of representative documents. The GreedyDiff algorithm schedules the second job first. I'm assuming that you have the skills to translate such high-level descriptions into working code in your favorite programming language. // subproblem solutions (indexed from 0) A := (m + 1) → (n + 1) two-dimensional array // base case #1 (j = 0) for i = 0 to m do A[i][0] = i · ↵gap // base case #2 (i = 0) for j = 0 to n do A[0][j] = j · ↵gap // systematically solve all subproblems for i = 1 to m do for j = 1 to n do // use recurrence from Corollary 17.2 A[i][j]8:= 9 < A[i 1][j 1] + ↵xi yj (Case 1) = A[i 1][j] + ↵gap (Case 2) min : ; A[i][j 1] + ↵gap (Case 3) return A[m][n] // solution to largest subproblem As in the knapsack problem, because subproblems are indexed by two different parameters, the algorithm uses a two-dimensional array to store subproblem solutions and a double for loop to populate it. Next we consider the analogous problem for search trees—computing the best-on-average search tree given statistics about the frequencies of different searches. As a warm-up, suppose we took a (doomed-to-fail) divide-and-conquer approach to the optimal binary search tree problem. 16 Richard E. How can we solve this problem in general, given an input graph with arbitrary sets of path lengths and weights? The holy grail in algorithm design is a linear-time algorithm (or close to it), and this is what we want for the MST problem. (b) Use the Cut Property to prove that Prim's algorithm is correct. To recap, the statement (*) implies that, with the input ^ and p, the Huffman algorithm outputs the best-possible tree from the restricted set Tab . The cycle has the same six independent sets that the complete graph does, plus some independent sets of size 2. By a spanning tree of G, we mean a subset T ✓ E of edges that satisfies two properties. d) T may not be an MST and P may not be a shortest s-t path. 72 Minimum Spanning Trees The bad news is that the proof of Theorem 15.6 has several steps. Dijkstra independently arrived at the same algorithm shortly thereafter, in 1959. 18.2.1 The Subproblems As always with dynamic programming, the most important step is to understand the different ways that an optimal solution might be built up from optimal solutions to smaller subproblems. If we prove both the base case and the inductive step, then P (k) is indeed true for every positive integer k: P (2) is true by the base case and, like falling dominoes, applying the inductive step over and over again shows that P (k) is true for arbitrarily large values of k. *15.4 71 Prim's Algorithm: Proof of Correctness rule, (15.1) cv← w← cxy for every edge (x, y) 2 E with x 2 X and y 2 V X. One little issue: In the subproblems defined in Section 18.2.1, the edge budget i can be an arbitrarily large positive integer, which means there's an infinite number of subproblems. dynamic programming, 115, 120–122 when to use, 202 dynamic programming, vii as recursion with a cache, 112 bottom-up, 113 for all-pairs shortest paths, see Floyd-Warshall algorithm for graph problems, 172 for optimal binary search trees, 148–162 for single-source shortest paths, see Bellman-Ford algorithm for the knapsack problem, 123–132 for the sequence alignment problem, 137–146 for weighted independent set in path graphs, 108–118 history, 122 memoization, 113 optimal substructure, 120 ordering the input, 125 principles, 118–119 recurrence, 120 running time, 119 saving space, 165, 208 subproblems, 119–120 takes practice, 110 top-down, 112 vs. . n} and v, w 2 V , and let P be a minimum-length cycle-free v-w path in G with all internal vertices in {1, 2, . The Upshot P It is not obvious how to define shortest-path distances in a graph with a negative cycle. This is easiest to see for a job k processed before i and j in ← (as part of the "stuff" in Figure 13.3). Decoding a sequence of bits reduces to following your nose: Traverse the tree from top to bottom, taking a left or right turn whenever the next input bit is a 0 or 1, respectively. This is an example of a fixed-length binary code, which uses the same number of bits to encode each symbol. Then, S is either: (i) an optimal solution for the first n 1 items with knapsack capacity C, or (ii) an optimal solution for the first n 1 items with knapsack capacity C sn , supplemented with the last item n. a) 1 and 2 b) 2 and 1 c) 5 and +1 d) +1 and +1 (See Section 15.3.6 for the solution and discussion.) Lines 12–16 of the pseudocode pay the piper and perform the necessary updates to the keys of the vertices remaining in V X. Again, under our standing assumption that the input graph is connected, we can simplify the O((m + n) log n) bound to O(m log n). How does the input graph change upon passage to a recursive call? Don't let this bother you: You can always convert such a path into a cycle-free path with the same endpoints v0 and vk by repeatedly splicing out subpaths between different visits to the same vertex (see Figure 15.2 below). Suppose you know that every shortest path in G from s to another vertex has at most k edges. If a path uses an edge multiple times, each use counts against its hop count budget. A few examples include the fast Fourier transform, the maximum flow and minimum cut problems, bipartite matching, and linear and convex programming. By assumption (1), the greedy schedule schedules the jobs in order of index (with job 1 first, then job 2, all the way up to job n); see Figure 13.2. .... This tree defines an encoding for each symbol via the sequence of edge labels on the path from the root to the node labeled with that symbol. Every object x begins with depth 0. We conclude that the GreedyDiff algorithm fails to compute an optimal schedule for this example and therefore is not always correct. See Section 5.6 of Part 1 for a full discussion. (If there are multiple such roots, use the smallest one.) The key lemma is an easy-to-believe (but tricky-to-prove) monotonicity property: Adding a new maximum (respectively, minimum) element to a subproblem can only make the root of an optimal search tree larger (respectively, smaller). 30 82 Minimum Spanning Trees What kind of data structure would allow us to quickly identify whether the solution-so-far contains a path between a given pair of vertices? This cycle must include the new edge (v, w). , pr 1 and pr+1 , pr+2 , . P The first step in the proofs of correctness for Prim's and Kruskal's algorithms is to show that each algorithm chooses only edges satisfying the minimum bottleneck property (MBP). (For example, ce could indicate the cost of connecting two computer servers.) The goal is to compute a spanning tree of the graph with the minimum-possible sum of edge costs. Pep Talk It is totally normal to feel confused the first time you see dynamic programming. Because P was arbitrary, the edge (v ←, w←) is a minimum-bottleneck v -w← path. 16.2.5 Solutions to Quizzes 16.3–16.4 Solution to Quiz 16.3 Correct answer: (d). When you have a primitive relevant to your problem that is so blazingly fast, why not use it? Problem 18.3 (H) Consider a directed graph G = (V, E) with n vertices, m edges, a source vertex s 2 V , real-valued edge lengths, and no negative cycles. Because all of the internal vertices of P1← and P2 belong to {1, 2, . If that algorithm is modified to cache the comparison results (in effect, remembering which case of the recurrence was used to fill in each array entry), these results can be looked up rather than recomputed in the WIS Reconstruction algorithm. We have our subproblems. Because there's no difference between the two expressions. ↵ worst ^-tree in Tab ⌦ worst ^0 -tree in T . (As usual, n denotes the number of vertices.) a) 1 b) n 1 c) n d) n2 (See Section 18.3.3 for the solution and discussion.) Plugging in the Bellman-Ford algorithm (Theorem 18.5) for the single-source shortest path subroutine in Quiz 18.4 gives an O(mn 2)time algorithm for the all-pairs shortest path problem.19 18 One important application of algorithms for the all-pairs shortest path problem is to computing the transitive closure of a binary relation. d) Given capacities C1 , C2 , . 16.5.5 Example Recall the four-item example from Quiz 16.5, with C = 6: Item 1 2 3 4 Value 3 2 4 4 Size 4 3 2 3 Because n = 4 and C = 6, the array A in the Knapsack algorithm can be visualized as a table with 5 columns (corresponding to i = 0, 1, . To prove the converse, assume that G has a negative cycle. If v 6= w and (v, w) / 2 E, there are no v-w paths with no internal vertices and L0,v,w = +1. A corollary is a statement that follows immediately from an alreadyproven result, such as a special case of a theorem. Ties between edges can be broken arbitrarily. Because the knapsack capacity is 6, there is no room to choose more than two items. , ek = (vk 1 , vk) with vk = v0 . (The job weights are irrelevant for this question, so we have not specified them.) a) 1, 2, and 3 b) 3, 5, and 6 c) 1, 3, and 6 d) 1, 4, and 6 (See Section 13.2.4 for the solution and discussion.) 13.2.3 Objective Function What makes for a good schedule? a) O(m log n) b) O(n2) c) O(mn) d) O(m2) (See below for the solution and discussion.) Correct answer: (c). Two Assumptions (1) The jobs are indexed in

nonincreasing order of weightlength ratio: w1 `1 w2 `2 ··· (2) There are no ties between ratios: i 6= j. 124 Introduction to Dynamic Programming Problem: Knapsack Input: Item values v1 , v2 , . Feel free to assume that there are no ties. Chapter 15 Minimum Spanning Trees This chapter applies the greedy algorithm design paradigm to a famous graph problem, the minimum spanning tree (MST) problem. 13.1 The Greedy Algorithm Design Paradigm 3 factors and different logarithmic functions differ by a constant factor, so there is no need to specify the base of the logarithm.) Finally, it's often difficult to figure out whether a proposed greedy algorithm actually returns the correct output for every possible input. 24 For more on correctness of algorithms, including the algorithm to compute them in linear time, see Part 2. For of some other alignment of X and Y and the P... was more efficient with substitution ... in this optimal alignment. Thus ... $\langle P \cdot x m y n \rangle + \cdot x m y n = P$, contradicting the optimality of the original alignment of X and Y . 17.1 Sequence Alignment 17.1.1 Motivation If you take a course in computational genomics, the first few lectures will likely be devoted to the sequence alignment problem.2 In this problem, the input consists of two strings that represent portions of one or more genomes, over the alphabet—no prizes for guessing!— {A, C, G, T}. If you have a black belt in dynamic programming, you might be able to just stare at a problem and intuitively know what the subproblems should be. Why bother? For example, here's a second search tree containing objects with the keys {1, 2, 3, 4, 5}: 5 4 3 2 1 where the "1" now has a search time of 5. 13 For readers of Part 2, all the ideas in this section will be familiar from the corresponding high-level implementation of Dijkstra's shortest-path algorithm (Section 10.4). Output: a maximum-weight independent set of G. , xm and Y = y1 , y2 , . (Break ties arbitrarily.) The First Main Idea, Restated To implement the first and more difficult main idea from Section 14.4.1, define = ^-trees in which a and b are the left and right Tab = . For (b), every iteration of Prim's algorithm chooses the cheapest edge e that crosses the cut (X, V), where X is the set of vertices spanned by the solution-so-far. Any method for computing repeated minimum computations faster than exhaustive search would translate to a faster implementation of Prim's algorithm. 13.3 Developing a Greedy Algorithm Greedy algorithms seem like a good fit for the problem of scheduling jobs to minimize the weighted sum of completion times. For many problems, they are easy to devise and often blazingly fast. Suppose Pi is an s-v path with at most i edges, and moreover is a shortest such path. Several other books and resources on the Web offer concrete implementations of various algorithms in specific programming languages. Quiz 18.2 How many candidates are there for an optimal solution to a subproblem with the destination v? , m. With Gi playing the role of the graph G. We conclude that the average leaf depth of T = 2 This is at most that of T. What must it look like? One, it's a first-bullet full-affirm algorithm; so the computer scientist should know about it. What about i in the case where: (i) v = w; (ii) (v, w) is an edge of G; and (iii) v and w (v, w) is not an edge of G? For example, finding a path which leads to a linear-time algorithm for the WIS problem in arbitrary graphs. P Subproblems correspond to prefixes of the two input strings. A slight tweak to the algorithm reduces the running time to O(n2). If the object at the root of the tree has the key 23, you know that the object you're looking for is in the root's 150 Advanced Dynamic Programming left subtree. The Bellman-Ford algorithm (Section 18.2) solves the single-source shortest path problem with negative edge lengths; it also has the benefit of being "more distributed" than Dijkstra's algorithm and, for this reason, has deeply influenced the way in which traffic is routed in the Internet. The length of a path is the sum of the lengths of its edges. In general, the encoding of a symbol a is a prefix of that of another symbol b if and only if the node labeled a is an ancestor of the node labeled b. Because w = 2 X and w' = 2 Y, the path P crosses at some point from X to V X, say via the edge (x, y) with x 2 X and y 2 V X (Figure 15.6). When a leaf is reached, its label indicates the next symbol in the encoding and the process restarts from the root with the remaining input bits. The last of these has the largest total weight of 8. Is this also true on a per-vertex basis? (Example, of the goal is to cluster blog posts about US. politics into groups of "left-leaning" and "right-leaning" posts, it makes sense to choose k = 2. Thus T 0 = T [{e} {e0 } is a spanning tree with cost less than that of T , a contradiction. This no conceivable improvement in computer technology would transmute exhaustive search into a useful algorithm. Similarly, the lower-right corner must be either a gap or the last symbol y n of the second string Y . The edges do not include a cycle, and they can be used to travel from any vertex to any other vertex. 17.2.9 Solution to Quizzes 17.4–17.5 Solution to Quiz 17.4 Correct answer: (b). Obtain a new graph G0 from G by increasing the cost of each edge ei from cei to c0ei = cei + (2(m +1)), where m is the number of edges. (Do you see how to do this?) Our dynamic programming algorithm for the problem will run in O(n) time. Solution to Quiz 17.2 Correct answer: (b). See Section 18.1.3 for the solution and discussion.) 18.1.3 Solution to Quiz 18.1 Correct answer: (a). Thus, if one batch of subproblems over the recurrence (with Case 1 of the recurrence winning for every destination), these optimal solutions will remain the same forevermore. (Choose all that apply.) 135 Problems a) The formula is not always correct in path graphs. , and iZSm si Cm . That is, among all spanning trees T of a connected graph P with edge costs, compute one with the maximum-possible sum e2T ce of edge costs. Prim, who discovered the algorithm in 1957. Huffman's Greedy Algorithm Merge the pair of trees that causes the minimum-possible increase in the average leaf depth. For example, A[2][5] the better option is to skip the second item and inherit the "3" immediately to the left, while for A[3][5] the better option is to include the third item and achieve 4 (for v3) plus the 2 in the entry A[2][3]. Become a better programmer. Given only the encoding, there's no way of knowing which meaning was intended. The v-w path P includes an edge (x, y) with x 2 S1 and y 2 S2 . Thus, we can view P1 and P2 as solutions to smaller subproblems, with origins v and k and destinations k and w, respectively, and with all internal vertices in {1, 2, . Because the exchange occurs after k completes, it has no effect on k's completion time (the amount of time that elapses before k completes). Keuth (Acta Informatica, 1971). For example, every time you import or export an MP3 audio file, your computer uses Huffman codes. (See Section 13.4.5 for the solution and discussion.) To finish the proof, take the arbitrary competing schedule — and repeatedly swap jobs to remove consecutive inversions.11 Because the number of inversions decreases with every swap (Quiz 13.5), this process eventually terminates. This is our first case study in which the subproblems work g(n) is not proportional to the number of outputs a "-tree with the minimum-possible average leaf depth (14.1). In the happy case that you devise a good algorithm for the problem, can you make it even better by deploying the right data structures? First, compute in advance all sums of the form jk=i pk ; 2 this can be done in O(n) time (do you see how?). d) If all symbol frequencies are less than 0.5, all symbols will be encoded with at least two bits. Part 4 is all about NP-completeness, what it means for the algorithm designer, and strategies for coping with computationally intractable problems, including the analysis of heuristics and local search. We assume that f is symmetric, meaning f (x, y) = f (y, x) for every pair x, y of data points. For example, if jobs represent workers, multiplying by specific programs and units, and edges represent conflicts between courses, the MWIS corresponds to the feasible course schedule with the heaviest load (in units). The algorithm is then stuck selecting the first vertex and it outputs an independent set with total weight 0. Lemma 16.1 singles out the only two possibilities for an MWIS, so whichever option has larger total weight is an optimal solution. The final output above is the minimum spanning tree of the graph (as you should check). It depends on how much you can compress the files with little or no loss. Solution to Quiz 18.6 Correct answer: (c). As stated, the parameter i could be an arbitrarily large positive integer and there is an infinite number of subproblems. One idea is to notice that these two strings can be "nicely aligned": A A G G G G C T A where the "–" indicates a gap inserted between two letters of the second string, which seems to be missing a "C". Thus, there are six subproblems overall. A totally different application of the problems in 1 The presentation in this section draws inspiration from Section 6.6 of Algorithm Design, by Jon Kleinberg and Eva Tardos (Pearson, 2005). (See Section 16.5.7 for the solution and discussion.) This case analysis shows that two and only two candidates are vying to be an optimal knapsack solution: Knapsack Solution an optimal solution to a knapsack problem with n items, item values v1 , v2 , . In the second iteration, there are three edges worth of crossing the cut: (c, a), (b, a), and (b, d). The smarter approach to this problem is to compute an optimal set, but it might not be an MST. Problem 15.6 (H) Prove that in a connected graph with distinct edge costs, there is a unique MST. Also, each item has a weight wi , with higher weights corresponding to higher-priority jobs. Working backward, the algorithm first checks which case of the recurrence was used to fill in the array entry A[m][n] corresponding to the largest subproblem (resolving ties arbitrarily).8 If it was case 1, the last column of the optimal alignment matches xm and yn and reconstruction resumes from the entry A[m 1][n 1]. Now derive T 0 from T = {e0 } by removing the edge e2 : x y w Because T has n 1 edges, so does T 0. The proposed variable-length code encodes a message like "0.1". Is this 0.1 only one possible? We... AB (encoded as "0" and "01", respectively), and is another is AAD (encoded as "0", "0" and "1"). Can you make your algorithm simpler or faster using randomization? Proposition 14.1 (Encoding Length and Tree Depth) For every binary code, the encoding length in bits of a symbol a 2 * equals the depth of the node with a label in the corresponding tree. The next-best option is the edge of cost 5; its inclusion does not create a cycle and, in fact, results in a spanning tree: 4 1 2 3 5 7 6 The algorithm skips the edge of cost 6 (which would create a triangle with the edges of cost 3 and 5) as well as the final edge (of cost 7 (which would create a triangle with the edges of cost 1 and 5). What should be the new values of y and z's keys, respectively? Greedy algorithms are a bit of a mystery: we think heuristic in this part of your algorithmic toolbox, which is defined by a destination vertex v 2 V and a hop count constraint i 2 {1, 2, . 108 Introduction to Dynamic Programming 16.4.1 Solutions to Quizzes 16.1–16.2 Solution to Quiz 16.1 Correct answer: (c). 36 Chapter 14 Huffman Codes where the summations are over all the alphabet symbols for which the corresponding leaf belongs to T1 or T2 , respectively. Assuming this lemma, for every subproblem with i < (, the optimal cost ri, j) is at least r(i, j) and at most r(i + 1, j). Algorithm Problems. Now that we know which subproblems are the correct ones, we can just cut to the chase and systematically solve them one by one! Quiz 14.1 With the variable-length binary code above, what is the string "001" an encoding of? Obtain the graph G0 from G by (the last two vertices and edges 5 excluded from S included in S v1 v2 v3 v4 1 4 5 4 Gn-2 Because S contains vn and Gn 2 does not, we can't regard S as an independent set of Gn 2 . Which split should we use? It was something not even a Congressman could object to. Or maybe each object is a representation of a document (say, as a vector of word frequencies), with connections corresponding to pairs of "similar" documents. This would contradict the supposed optimality of S for the original problem. Suppose 1 and v 2 V , and let P be a shortest v-w path in G with at most i edges, with cycles allowed. In the third iteration, the value at i drops to 5 (because A[2][w] + vi =-5, which is better than A[2][w] as we've seen). 4 2 3 Solution to Problem 16.3(a) A[2] s1 s2 s3 s4 A[2][ss] A[1][s] = A[0][s] A[1][s] = A[0][s] A[1][w] + A[1] w] = 4 s t -1 2 A[2][u] = A[2][u] 2 A[1][u] = A[0][u] = • 2 u 2 w Nothing changes in the fourth iteration: A[3][w] = A[2][w] A[4][w] = A[1][w] = A[0][w] = • A[2][u] = A[2][u] = A[2][u] = A[2][w] = 4 A[4][w] = A[1][w] = A[2][w] = A[1][u] = A[2][u] = • A[4][s] = A[3][s] = 5 A[2][s] = A[1][s] = A[0][s] = • A[0][u] = ... and at this point the algorithm halts with the correct shortest-path distances.12 18.2.7 Running Time The running time analysis of the Bellman-Ford algorithm is more interesting than those of our other dynamic programming algorithms. 17.2 Average Search Time The search time for a key k in a binary search tree is the number of nodes visited while searching for the node with key k (including that node itself). In case (i), the algorithm is responsible for outputting n2 numbers. 18 18.3.2 Reduction to Single-Source Shortest Paths If you've got the lookout for reductions (as you should be; see page 100), you might already see how to apply your ever-growing algorithmic toolbox to the all-pairs shortest path problem. As we've seen, prefix-free binary codes for an alphabet * correspond to *-trees, which are binary trees with the sources must be 1 other looks up or recursively computes and caches S2 , as needed. 21 Named after Robert W. Provided the Find and Union operations run in O(log n) time, as assured by Theorem 15.14, the total running time is: preprocessing O(n) Union operations + remaining bookkeeping total O(m) + O(m log n) O(m log n) O(n) log n) O(m) (m + n) log n). Each edge is even in a trivial example like: v - 5 1 s t -2 1 An infinite number of subproblems. For instance, we can assume that the input graph has no parallel edges. After finishing this chapter, ask yourself: Could you have ever solved either problem with a greedy algorithm? Intuitively, it's not win to denote the smaller-frequency symbols a and b to the deepest level of the tree while promoting the higher-frequency symbols x and y closer to the root. (All positive integers.) Output: A subset S P v 2 (1, 2, . For example, you can always compute a minimum spanning tree of your undirected graph data in a preprocessing step: if you're not quite sure how it will help later, ... Sp of non-empty subsets of X such that each object x 2 X belongs to exactly one of the subsets. 13.3.1 Two Special Cases Let's think positive and posit that there actually is a correct greedy algorithm for the problem of minimizing the weighted sum of completion times. , k by, as ALk 1][v][k] and A[k 1][k][v] are most their respective best lighter ... . Sp of most ... 14.2 Codes as Trees The "prefix-free" constraint in the optimal prefix-free code problem sounds a little scary. 15 Our dynamic programming algorithm for the knapsack problem in Section 16.5 is a good example. 14.1.3 Prefix-Free Codes We can eliminate all ambiguity by insisting that a code be prefixfree. , w and lengths `1 , `2 . The overall running time is O(m + n) O(n). As promised by Theorem 15.4. 18 15.3.6 Solution to Quiz 15.3 Correct answer: (b). Presumably, the algorithm would start by solving the smallest subproblems (with edge budget i = 0), followed by the next-smallest subproblems (with i = 1), and so on. A subgraph with n 1 edges — candidate for a spanning tree, by Corollary 15.8—fails one of the conditions only if it fails both. In other words, once you know that an optimal alignment of X and Y matches xm and yn in its last column, you know exactly what the rest of it looks like: an optimal alignment of X0 and Y 0 . , r 1} n T are one more than in T1 , and those for the keys {r + 1, r + 2, . (b), (b), adding the edge (4, 8) fuses two connected components into one. 116 Introduction to Dynamic Programming plucked off the end of the graph. For 38 Supervised learning focuses on prediction rather than pattern-finding per se. (Choose all that apply.) a) The maximum-cost spanning tree problem. Let L denote the length of P . Each iteration of Kruskal's algorithm performs a cycle check or, equivalently, a path check. If this bothers you, feel free P to normalize the frequencies by dividing each of them by their sum nj=1 pj —this doesn't change the problem. 18 Some heap implementations export a DecreaseKey operation, in which case lines 14–16 can be implemented with one heap operation rather than two. How much better are the schedules computed by the GreedyRatio algorithm than those by the GreedyDiff algorithm? , n} of keys and nonnegative frequencies p1 , p2 , . The general statement in Corollary 16.2 follows by invoking the first statement, for even i = 2, 3, . In this case, the path P can be interpreted as the amalgamation of two solutions to smaller subproblems: the prefix P1 of P that travels from v to k and the suffix P2 of P that travels from k to w. I decided therefore to use the word, "programming". This means, for each pair of distinct symbols a, b 2 *, the encoding of a is not a prefix of that of b, and vice versa. Problem: Minimizing the Sum of Weighted Completion Times Input: A set of n jobs with positive lengths `1 , `2 , . divide-and-conquer, 115, 120–122 when to use, 202 e.g., 79 edge (of a graph), 52 length, 168 parallel, 55 exchange argument, 12 for minimum spanning trees, 75 in Huffman's algorithm, 43 index in scheduling, 15 exhaustive search, 6, 27, 62, 105, 149, 161, 201 Fano, Robert M., 32 field guide (to algorithm design), 201–202 Floyd, Robert W., 187 Floyd-Warshall algorithm, 187– 196 detecting a negative cycle, 194–195 optimal substructure, 189– 192 pseudocode, 193 correctness, 196 recurrence, 192 running time, 194 space usage, 194 subproblems, 187–189 for-free primitive, 64, 201 Ford, Jr., Lester R., 172 forest (of trees), 34 Gd, 164, gad, 52 adjacency-list representation, 53 adjacency-matrix representation, 194 connected, 54 connected component, 72 cut, 101 cycle, 53 dense, 187 edge, 52 independent set, 104 notation, 53 path, 53 planar, 52 search, 55 spanning tree, 53 sparse, 185 undirected, 52 Graphs and Grids on and Trees and Alan, 105 greedy algorithm, 2 as a heuristic, 3 exchange argument for clustering, 95 for optimal prefix-free codes, see Huffman's algorithm for scheduling, 6–9 induction, 13, 5 proving correctness, 5 for minimum spanning trees, see Kruskal's algorithm, see Prim's algorithm for the knapsack problem, 9 minimum spanning tree, 9 pick ... in the spirit, GreedyDiff, 1, 7–8, 13 intuition, 4 ..., 6 themes, 2, usually not correct, 3, 307 GreedyRatio, heap, 62–63 DecreaseKey, 67 Delete, 63 ExtractMin, 63 Insert, 63 implementation, 63 in Huffman's algorithm, 40 operation running times, 63 raison d'être, 62 speeds up Prim's algorithm, 63–68 HeapSort, 2 mins, 302, 3 hint, 9, 203–210 Huffman's algorithm, 32–36 ** trees, 31 average leaf depth, 31, 44 examples, 37–40 greedy criterion, 35 implemented with a heap, 40 214 implemented with two queues, 45, 51, 204 proof of correctness, 41 pseudocode, 36 running time, 40 Huffman, David A., 32 i.e., 23 independent set (in a graph), 104 induction, 43 base case, 43 in bottom-up, 42 inductive step, 43 Indyk, Piotr, 106 interview questions, ix inversion, 18 consecutive, 19 Jarník's algorithm, see Prim's algorithm Jarník, Vojtěch, 57 Java, 51 job, see scheduling Kaplanski, Ivyeds, 147 knapsack problem, 123–132 applications, 124 correctness, 129 definition, 123 dynamic programming algorithm (Knapsack), 128 example, 129–130 generalizations, 133 optimal substructure, 125–126 recurrence, 126 reconstruction, 130–132 recurrence, 126 running time, 129 subproblems, 127 Knuth, Donald E., 148, 157 Kruskal's algorithm achieves the minimum bottleneck property, 93 Index and clustering, 97 cycle-checking, 81, 84 example, 77 in reverse, 99 outputs a spanning tree, 92 proof of correctness, 91–93, 102 pseudocode (straightforward), 79 pseudocode (union-find-based), 85 reasons to care, 77 running time (straightforward), 81 stopping early, 79 so, for the double for loop must compute the solution subproblem A[i][c], the values A[i 1][c] and A[i 1][c ii] of two relevant smaller subproblems have already been computed in the previous iteration of the outer loop (or in the base case). As a bonus, in typical implementations, the constant hidden by the big-O notation are relatively small.12 15.3.3 How to Use Heaps in Prim's Algorithm Heaps enable a blazingly fast, near-linear-time implementation of Prim's algorithm.13 12 For the goals of this section, it's important to know how heaps work and what they look under the hood. Prim's algorithm, 77 Kruskal, Joseph B., 76 Lehman, Eric, x Leighton, F. Let L denote the smallest perfectly possible difference between two edges' costs. This means there is a different search tree T1— with keys {1, 2, . It's important that the outer loop is indexed by the subproblem size k, so that all of the relevant terms A[k 1][v][w] are available for constant-time look up in each inner loop iteration. A[i 2] + wi , which have already been made by the WIS algorithm. What can we say about it? When the reconstruction algorithm hits a base case, it completes the alignment by preprending the appropriate number of gaps to the string that has run out of symbols. QE D *15.4 Prim's Algorithm: Proof of Correctness We can similarly argue that the output of Prim's algorithm is a spanning tree. Still, there is no substitute for the detailed understanding of an algorithm that comes from providing your own working implementation of it. The divide-and-conquer paradigm (Section 13.1.1) leads to a natural way to break the input into smaller subproblems. As a bonus, because each batch of subproblem solutions and predecessors depends only one those from the previous batch, both the forward and reconstruction passes require only O(n) space (analogous to Problem 17.5). If not run, the Floyd-Warshall algorithm on G. 14.4.2 The Details Induction Review For the formal proof, we turn to our old friend (or is it nemesis?), induction.11 Recall that proofs by induction follow a fairly rigid template, with the goal of establishing that an assertion P (k) holds for arbitrary large positive integers k. Next, recall our first (non-prefix-free) variable-length code: Symbol A B C D Encoding 0 01 10 1 This code can be represented using a different labeled binary tree: 1 0 D A 1 0 B C 0 1 Here again there are four nodes labeled with the symbols of the alphabet—the two leaves and their parents. We mentioned briefly in Section 13.6 that there is, in fact, a data structure whose reason d'être is to speed up repeated minimum computations: the heap data structure. The single-source shortest path problem also has a more general instance, the Bellman-Ford algorithm. Find 1. Ideally, this thought experiment would show that an optimal solution must be constructed in a prescribed way from optimal solutions to smaller subproblems, thereby narrowing down the field of candidates to a manageable number. 2 More concretely, let G = (V, E) denote the n-vertex path graph with edges (v1 , v2), (v2 , v3), . In (c), adding the edge (7, 9) creates a new cycle. (For each i, j 2 {1, 2, . Applying the Inductive Hypothesis to a Residual Problem The inductive hypothesis applies only to alphabets with less than k symbols. What if we consider only paths without cycles? The second job contributes 12 Introduction to Greedy Algorithms to all the completion times other than that of the first job, so the second-shortest job should be scheduled next, and so on. The answer to the second question is now clear: Install the demoted root directly under the (depth-0) promoted root so that the occupants of T1 suffer a depth increase of only 1. (Adding an edge (x, v) to the solution-so-far T creates a cycle only if T already contains a v-w path.) 29 Also known as the disjoint-set data structure. ', j After clearing denominators, this translates to w i j ij(r) j and r are not lengths, and only needs an associative combination of operations. 12 If subproblem solutions don't stabilize (with Ln,v Ln 1,v for all v 2 V), the subproblems in the Bellman-Ford algorithm (Section 18.2.1) is to always work with the original input graph and impose an artificial constraint on the number of edges allowed in the solution to a subproblem. See the bonus video at www.algorithmsilluminated.org for more details. If A[n 2] + wn An 1], reconstructing an MWIS of Gn with yields an MWIS of Gn . In the running time of our dynamic programming algorithms is of the time-constrained reader can skip these sections. This is an MWIS of G0 or the algorithm's irrevocable myopic decisions will come back to haunt you and, with full hindsight, be revealed as a terrible idea. The time-constrained reader can skip these sections. If you schedule the shorter job first, the completion times are 1 and 3, for a total of 4. Our fixed-length code Symbol A B C D Encoding 00 01 10 11 can be represented via a complete binary tree with four leaves: 1 0 0 A 1 0 B C 1 D Every node of a binary tree can have a left child, a right child, both, or neither. , pn The first benefit of emphasizing two specific implementations is flexibility, and so on, ad infinitum. Solution to Quiz 16.1 Correct answer: (c). Given a connected undirected graph G = (V, E) with positive edge costs, compute a minimum-cost set F ✓ E of edges such that the graph (V, E F) is acyclic. But then Li,v < Lk,v , contradicting part (a) of the lemma. Every recursive call of a divide-and-conquer algorithm commits to a single split of its problem into one or more smaller subproblems. Thus, the weighted search time (17.1) can be written as p k · (1 + k=1 Pk k's search Pn time in T1)+p r 1 + k=r+1 pk + (1+k's search time in T2) + (weighted search time in T2) x Preface Who Are You? For further details, see the paper "Optimum Binary Search Trees," by Donald E. 28 Huffman Codes Solution to Quiz 14.2 Correct answer: (b). Output: a minimum-cost spanning tree T ✓ E of G. Relax the Objective Function: Sacrifice the guarantee. . That is, in this case, the "correct" entry of the programming theory is an adjective, and that it's impossible to use the word, dynamic, in a pejorative sense. 3 Test Your Understanding Problem 15.1 (S) Consider an undirected graph G = (V, E) in which every edge e 2 E has a distinct and nonnegative cost. Corollary 15.16, once the Kruskal algorithm has processed every edge of P , its endpoints v and w belong to the same connected component of the solution-so-far (and, hence, of the final output (V, T)). It's easy to see how to devise a faster algorithm under these circumstances: we can check for the pseudocode for the Prim's algorithm on data structure or the lack thereof. Just like for any of the subproblems is at best an exhaustive search. 18 18.2.8 Solution to Quiz 18.1 Correct answer: (a). Thus, even in (a) we conclude that the algorithm computes such a solution. Remarkably, if the distribution is concentrated in one part of the graph, in algorithms we ignore 25 for the algorithm below we are running the algorithm: where the last step is the one with largest total weight. We're done except in the ... in the directed graph G on the condition where all pairs of vertices are reachable. The previous ... not be the ... the ... .. of ... .. and ... . If the ... ... and ... and ... but ... for ... . and ... other techniques ... for ... .

algorithm, but how could I ever have come up with it myself?" Now that you've achieved a black-belt (or at least brown-belt) level of skill in the art of dynamic programming, I hope your reaction is: "How could I not have come up with this algorithm myself?" 30 For a different approach, see Problem 18.6. 18.4 195 The Floyd-Warshall Algorithm (ii) there is no path from a vertex v to itself shorter than the empty path (which has length 0). 18.2 The Bellman-Ford Algorithm The Bellman-Ford algorithm solves the single-source shortest path problem in graphs with negative edge lengths in the sense that it either computes the correct shortest-path distances or correctly determines that the input graph has a negative cycle.8 This algorithm will follow naturally from the design pattern that we used in our other dynamic programming case studies. Hint for Problem 16.2: For (a) and (c), revisit the four-vertex example on page 105. 4 You cannot reduce the single-source shortest path problem with general edge lengths to the special case of nonnegative edge lengths by adding a large positive constant to the length of every edge. 166 Advanced Dynamic Programming Problem 17.7 (H) There are seven kinds of Tetris pieces.21 Design a dynamic programming algorithm that, given x1 , x2 , . The Bellman-Ford algorithm solves (n + 1) · n = O(n2) different subproblems, where n is the number of vertices. P Exchange arguments are among the most common techniques used in correctness proofs for greedy algorithms. 18.2 173 The Bellman-Ford Algorithm The Bellman-Ford algorithm takes a different tack, one inspired by the inherently sequential nature of the output of the single-source shortest path problem (i.e., of paths). Thus, Case 1 contributes one candidate and Case 2 a number of candidates equal to the in-degree of v. Let's assume that such questions have already been answered experimentally, in the form of known penalties for gaps and mismatches that are provided as part of the input (along with the two strings). Because each recursive call of a dynamic programming algorithm tries out multiple choices of smaller subproblems, subproblems generally recur across different recursive calls; caching subproblem solutions is then a no-brainer optimization. This problem can be solved by first computing the connected components of the input graph in linear time using breadth- or depth-first search (see Chapter 8 of Part 2), and then applying an algorithm for the MST problem to each component separately. We can assume that not all edges have the same cost, and more generally that not all spanning trees have the same total cost (why?). Well, not quite: Cycles are a no-no, so it chooses the cheapest edge that doesn't create a cycle. Tacking a negative-length edge onto a path would decrease its overall length, so nonnegative edge lengths are necessary for path monotonicity. The second job must wait for the first job to complete, so its completion time is the sum of the lengths of the first two jobs, which is 3. Because v and w are on different sides of the cut (A, B), P includes an edge e0 that crosses (A, B). , yd) are d-dimensional vectors, the d2 psi cle formula is f (x, y) = yi) 2 . This wraps up the proof of (?). , k 1} ; or 192 Shortest Paths Revisited (ii) the concatenation of a minimum-length cycle-free v-k path with all internal vertices in {1, 2, . Here are two spanning trees with an inferior total cost of 8: a 4 c 1 a 1 b 3 2 c 4 5 d The three edges (a, b), (b, d), and (a, d) have a smaller total cost of 6: a 1 b 3 c 2 d but these edges do not form a spanning tree. Because 1 , the edges of G0 have distinct costs, with edge ei the ith-cheapest edge of G0 . P The key idea in the Bellman-Ford algorithm is to parameterize subproblems by an edge budget i (in addition to a destination) and consider only paths with i or fewer edges. 24 Ignore the uninteresting case in which m is much smaller than n; see also footnote 17. A key ingredient in the correctness proof for Dijkstra's algorithm is "path monotonicity," meaning that tacking on additional edges at the end of a path can only make it worse. a) T must be an MST and P must be a shortest s-t path. Few benefits of a serious study of algorithms rival the empowerment that comes from mastering dynamic programming. I'm not using the term lightly; I'm using it precisely. use a greedy algorithm that iteratively decides which job should go next? As usual, the first step merges the two symbols with the smallest frequencies, namely "B" and "E": 14.3 39 Huffman's Greedy Algorithm 4 3 2 6 8 2 6 A B C D E F B E B F The five trees left in the forest are Symbol tree containing C tree containing D tree containing F tree containing A B and E Sum of Symbol Frequencies 3 6 8 2+2 = 4 and the algorithm next merges the first and last of these: 4 3 6 8 6 A C D F B E F 7 E 6 8 6 C D F T he four remaining trees are Symbol tree containing C tree containing D tree containing F tree containing A, B and E Sum of Symbol Frequencies 6 8 6 4+3=7 Next the algorithm merges the nodes labeled "C" and "F": 7 6 8 6 C D F 7 12 8 A B E C D F A and three trees remain: Symbol tree containing D tree containing C and F tree containing A, B and E Sum of Symbol Frequencies 8 6 + 6 = 12 7 The penultimate merge is of the first and third trees: 40 Huffman Codes 15 7 12 12 8 D C F A B A E B E C F and the final merge produces the output of the algorithm: 0 1 0 1 0 A 1 0 0 D 1 B 1 C F E This tree corresponds to the following prefix-free code: Symbol A B C D E F Encoding 000 0010 10 01 0011 11 Running Time A straightforward implementation of the Huffman algorithm runs in O(n2) time, where n is the number of symbols. , n} to a distinct value f (i) 2 {1, 2, . As with all our dynamic programming algorithms, the correctness of the OptBST algorithm follows by induction on the subproblem size), with the recurrence in Corollary 17.5 justifying the inductive step. This chapter and the next two provide this practice through a half-dozen detailed case studies, including several algorithms belonging to the greatest hits compilation. Delete: given a heap H and a pointer to an object x in H, delete x from H. These observations lead to the following recurrence, which computes the best of the three options by exhaustive search: Corollary 17.2 (Sequence Alignment Recurrence) With the assumptions and notation of Lemma 17.1, let Pi,j denote the total penalty of an optimal alignment of Xi = x1 , x2 , . Problem 15.2 (H) Consider the following algorithm that attempts to compute an MST of a connected undirected graph G = (V, E) with distinct edge costs by running Kruskal's algorithm "in reverse": Kruskal (Reverse Version) T := E sort edges of E in decreasing order of cost for each e 2 E, in order do if T {e} is connected then T := T {e} return T 100 Minimum Spanning Trees Which of the following statements is true? For example, the input might be the strings AGGGCT and AGGCA. For a clue, look at our three examples. The process by which we arrive at this algorithm is more important to remember than the algorithm itself; it's a repeatable process that you can use in your own applications. Subproblems of the form W1,r 1 in the recurrence (17.3) are defined by prefixes of the set of keys (as usual), but subproblems of the form Wr+1,n are defined by suffixes of the keys. , yn over the alphabet ^ = {A, C, G, T }, a penalty ↗xy for each x, y 2 ^, and a gap penalty ↗gap 0. 14.1 27 Codes Problem: Optimal Prefix-Free Codes Input: A nonnegative frequency pa for each symbol a of an alphabet ^ of size n 2. If only we had clairvoyance and knew which key should appear at the root, we might then be able to compute the rest of the tree recursively. The main loop has m iterations. 14.2 31 Codes as Trees 14.2 Which Trees Represent Prefix-Free Codes? 58 Minimum Spanning Trees The algorithm's initial (empty) tree spans only the starting vertex b. That's a lot of schedules! Which one should we prefer? Moreover, in the special cases in which sorting is possible in linear time, this implementation of the Huffman algorithm also runs in linear time.9 14.3.7 Solution to Quiz 14.3 Correct answer: (a). For example, in Figure 13.2, if jobs 2 and 3 were processed in the opposite order they would constitute a consecutive inversion (with i = 3 and j = 2). Let M ← denote the cost of an MST of G, and M the minimum cost of a suboptimal spanning tree of G. The initial parent graph consists of isolated vertices with self-loops: For the Find operation, we leap from parent to parent until we arrive at a root object, which can be identified by its self-loop. Thus the GreedyDiff algorithm schedules the second job first, while GreedyRatio does the opposite. For bonus points, implement the space optimizations and linear-time reconstruction algorithms outlined in footnotes 13, 28, and 31. Theorem 17.3 (Properties of NW) For every instance of the sequence alignment problem, the NW algorithm returns the correct NW score and runs in O(mn) time, where m and n are the lengths of the two input strings. , pn , the OptBST algorithm runs in O(n3) time and returns the minimum weighted search time of a binary search tree with keys {1, 2, . Our first lemma 23 Theorem 15.6 does not hold as stated in graphs with non-distinct edge costs. The difference lies in the constraint that the binary tree must satisfy. , Gn . There is a one-to-one correspondence between ^0 -trees and the restricted set Tab of ^-trees (Figure 14.2). However, we want to establish its superiority to all competing s-w paths with i 1 edges or fewer. But after removing the last vertex from S, we can: S {vn } contains neither vn 1 nor vn and hence can be regarded as an independent set of the smaller graph Gn 2 (with total weight W wn). The merger T3 of T1 and T2 is inserted at the back of Q2 . QE D 15.5 Kruskal's Algorithm This section describes a second algorithm for the minimum spanning tree problem, Kruskal's algorithm.25 With our blazingly fast heap-based implementation of Prim's algorithm, why should we care about 25 Discovered by Joseph B. The Huffman algorithm can be implemented by sorting the symbols in order of increasing frequency and then performing a linear amount of additional processing. We conclude that 2 has exactly the same inversions as 1 , except with the inversion of i and j removed. What is the NW score of X and Y ? 2 Readers of Part 2 have already seen a greedy algorithm, namely Dijkstra's shortest-path algorithm. There's only a linear number of subproblems per value in the output, which is as good or better than all our other dynamic programming algorithms. 186 Shortest Paths Revisited (i) the shortest-path distance dist(v, w) for every ordered vertex pair v, w 2 V ; or (ii) a declaration that G contains a negative cycle. Programming Problems Problem 17.8 Implement in your favorite programming language the NW and OptBST algorithms, along with their reconstruction algorithms. The next quiz illustrates how an extraction can change the frontier, necessitating updates to the keys of vertices still in V X to maintain the invariant. It's often tricky to come up with a good divide-and-conquer algorithm for a problem, and when you do, there's usually a "Eureka!" moment when you know that you've cracked the problem. In this case, we've salvaged the proof: Pb is a cycle-free v-w path with all internal vertices in {1, 2, . Hint for Problem 16.5: For (b) and (c), add a third parameter to the dynamic programming solution to the original knapsack problem in Section 16.5. For (d), how does the generalization of your solution to (c) scale with the number m of knapsacks? Chapter 18 Shortest Paths Revisited This chapter centers around two famous dynamic programming algorithms for computing shortest paths in graphs. Conversely, every tree T 2 Tab can be turned into a ^0 -tree T 0 by sucking the leaves labeled a and b into their (common) parent and labeling the resulting "meta-node" with "ab." We denote this inverse mapping T ↦ T 0 by ↲(T). For an example of the NW algorithm in action, check out Problem 17.1 17.1.7 Reconstruction There are no surprises in reconstructing an optimal alignment from the array values that the NW algorithm computes. GreedyDiff Schedule the jobs in decreasing order of wj lj (breaking ties arbitrarily). Because every ^-tree is equaled or bettered by a tree of Tab , Tab contains a tree that is optimal among all ^-trees. A cut of an undirected graph G = (V, E) is a partition of its vertex set V into two non-empty sets, A and B. In each iteration, the trees T1 and T2 can be removed from the heap using two successive ExtractMin operations, and the merged tree T3 added with one Insert operation (with T3 's key set to the sum of the keys of T1 and T2). 21 As we'll see, it's no accident that the edges satisfying the MBP in this example are precisely the edges in the minimum spanning tree. c) Schedule the jobs in increasing order of the product dj · pj . In each, the input consists of item values v1 , v2 , . P The knapsack problem can be solved using dynamic programming in O(nC) time. Single-Link Clustering via Kruskal's Algorithm 1. b) 2 has a number of inversions as c) 2 has one more inversion than 1. , 7) and the following frequencies: Symbol 1 2 3 4 5 6 7 Frequency 20 5 17 10 20 3 25 What are the final array entries of the OptBST algorithm from Section 17.2? Problem 17.5 (H) Recall the WIS algorithm (Section 16.2), the NW algorithm (Section 17.1), and the OptBST algorithm (Section 17.2). The first iteration of the greedy algorithm commits to the maximum-weight vertex, which is the third vertex (with weight 5). That is, it is true that, whenever Lk+1,v = Lk,v for some k and destination v, Li,v = Lk,v for all i k? 164 Advanced Dynamic Programming a) Neither algorithm remains well defined and correct after reversing the order of the for loops. , j} and frequencies pi , pi+1 , . Them, armed with a template for developing dynamic programming algorithms and an example instantiation, we'll tackle increasingly challenging applications of the paradigm. Ford, Jr., though ironically Alfonso Shimbel appears to have been the first. *17.2 159 Optimal Binary Search Trees 17.2.7 A Dynamic Programming Algorithm With the subproblems and recurrence in hand, you should expect the dynamic programming algorithm to write itself. Thus, we're after an algorithm that either computes the correct shortest-path distances or offers a compelling excuse for its failure (in 5 More precisely, any polynomial-time algorithm for any N P -hard problem would disprove the famous "P 6= N P " conjecture and resolve what is arguably the most important open question in all of computer science. d) The formula is always correct in arbitrary graphs. This implies that G has a negative cycle with no repeated vertices other than its start and end. Most of the canonical applications of the divide-and-conquer paradigm replace a straightforward polynomial-time algorithm for a task with a faster divide-and-conquer version.13 The killer applications of dynamic programming are polynomial-time algorithms for optimization problems for which straightforward solutions (like exhaustive search) require an exponential amount of time. , k). P In the single-source shortest path problem, the input consists of a directed graph with edge lengths and a source vertex. For example, you might be wondering if species B evolved from species A and then species C from B, or if B and C evolved independently from A. The graph (V, T) is connected if and only if it contains no cycles. If Y is empty, the optimal alignment inserts enough gaps into Y so that it has the same length as X. 15.2 57 Prim's Algorithm 15.2 Prim's Algorithm Our first algorithm for the minimum spanning tree problem is Prim's algorithm, which is named after Robert C. 9 Readers of Part 2 should recognize strong similarities to Dijkstra's shortestpath algorithm. The path P must be an optimal solution to the smaller subproblem; any superior solution would also be superior for the original subproblem, a contradiction. Every fixed-length code is automatically prefix-free. The trickier case is when the optimal solution S makes use of the last item n. Figure 15.1: An undirected graph with five vertices and eight edges. The length of P 0 is the same as that of P , less the sum of the edge lengths in the spliced-out cycle. Easy to analyze the running time. , (xp 1 , xp), where x0 is v and xp is w. A poor implementation can lead to a tree with height as large as n 1: *15.7 Kruskal's Algorithm: Proof of Correctness 91 In this sense, answer (c) is also correct. 17.1 145 Sequence Alignment NW Input: strings X = x1 , x2 , . For bonus points, implement the heap-based version of Prim (Section 15.3) and the union-find-based version of Kruskal (Section 15.6). Because Lk,v is the minimum length of an s-v path with at most k hops, there must be an s-v path with k hops and length smaller than Lk,v . The reconstruction process then resumes from vn 2 .8 WIS Reconstruction Input: the array A computed by the WIS algorithm for a path graph G with vertex set {v1 , v2 , . P1 = v-k path (cycle-free, length L1) (all internal vertices in {1,2,...,k-1}) P2 = k-w path (cycle-free, length L2) (all internal vertices in {1,2,...,k-1}) v k w P = v-w path (cycle-free, length L, all internal vertices in {1,2,...,k}) The vertex k appears in P only once (because P has no cycles) and therefore is not an internal vertex of P1 or P2 . The next quiz contains the key to unlocking a radical running time improvement. Which of them can be solved in O(mn) time? In other words, increasing the edge budget i from n 1 to n (or to any bigger number) has no effect on the minimum length of an s-v path. If there wasn't an efficient algorithm for computing the NW score, Needleman and Wunsch surely would have proposed a different and more tractable definition of genome similarity! 17.1 141 Sequence Alignment Quiz 17.2 Let X = x1 , x2 , . Test Your Understanding Problem 13.1 (H) You are given as input n jobs, each with a length `j and a deadline dj . Additional Resources These books are based on online courses that are currently running on the Coursera and Stanford Lagunita platforms. First, T should not contain a cycle (this is the "tree" part).3 Second, for every pair v, w 2 V of vertices, T should include a path between v and w (this is the "spanning" part).4 1 There is an analog of the MST problem for directed graphs, which is known as both the minimum-cost arborescence problem and the optimum branching problem. Consider an optimal binary search tree T with root r. Consider an optimal binary search tree T with root r. | Finally, there's nothing like a reader's perspective to point out the most troublesome concepts, the bugs and challenge data sets Problem 16.7 Implement in your favorite programming language the Knapsack and Knapsack Reconstruction algorithms. The rest of the work—initialization, updating F, rewiring pointers when merging two trees—contributes only O(n) operations to the overall running time bound, for a total of O(n log n). By the time an iteration of the main loop must compute the subproblem solution A[i], the values A[i 1] and A[i 2] of the two relevant smaller subproblems have already been computed in previous iterations (or in the base cases). 8 Introduction to Greedy Algorithms a) larger/shorter b) smaller/shorter c) larger/longer d) smaller/longer (See Solution 13.3.3 for the solution and discussion.) 13.3.2 Dueling Greedy Algorithms In the general case, jobs can have different weights and different lengths. Modify the algorithm so that it solves the all-pairs shortest path problem in O(mn) time for input graphs with a negative cycle and O(n3) time otherwise. I am particularly grateful to those who supplied detailed feedback on an earlier draft of this book: Tonya Blust, Yuan Cao, Carlos Guia, Jim Humelsine, Vladimir Kokshenev, Bayram Kuliyev, and Daniel Zingaro. On Lemmas, Theorems, and the Like In mathematical writing, the most important technical statements are labeled theorems. Problem: Single-Source Shortest Paths (Preliminary Version) Input: A directed graph G = (V, E), a source vertex s 2 V , and a real-valued length `e for each edge e 2 E.1 Output: the shortest-path distance dist(s, v) for every vertex v 2 V . (Each edge is labeled with its cost.) a 4 c 1 b 3 5 2 d a) 6 b) 7 c) 8 d) 9 (See Section 15.1.3 for the solution and discussion.) It makes sense only to talk about spanning trees of connected graphs G = (V, E), in which it's possible to travel from any vertex v 2 V to any other vertex w 2 V using a path of edges in E.5 (If there is 5 For example, the graph in Figure 15.1 is connected, while the graph in Figure 15.3 is not. Dynamic programming has no qualms about partitioning subroutine to choose how to split the input array in two, and then commits to this division for the remainder of its execution. We're not responsible for computing shortest-path distances for input graphs with a negative cycle. Let L, L1 , and L2 denote the lengths of P , P1 , and P2 , respectively. Key to our approach is the following thought experiment: Suppose someone handed us an optimal solution on a silver platter. We needed a smarter algorithm.6 13.2.4 Solution to Quiz 13.1 Correct answer: (c). 16.2 109 A Linear-Time Algorithm for WIS in Paths 16.2 A Linear-Time Algorithm for WIS in Paths We'll see several over the next few chapters. 2 13.1.2 Themes of the Greedy Paradigm Here are a few themes to watch for in our examples. vn . 17.1.7 147 Sequence Alignment 17.1.8 Solution to Quizzes 17.1–17.3 Solution to Quiz 17.1 Correct answer: (b). For a job k processed after i and j in ← (as part of the "more stuff" in Figure 13.3), the set of jobs completed before k is exactly the same in ← and in σ. For example, because the path from the root to the node labeled "B" traverses a left child edge ("0") followed by a right child edge ("1"), we can interpret the path as encoding the symbol B by 01. This completes the proof of the inductive step and of Theorem 14.2. QE D 49 Problems The Upshot P Prefix-free variable-length binary codes can have smaller average encoding lengths than fixedlength codes when different alphabet symbols have different frequencies. Otherwise, there is a type-C edge addition, which by Lemma 15.7(a) creates a cycle and also fails to decrease the number of connected components: In this case, the process starts with n connected components and the n 1 edge additions decrease the number of connected components at most n 2 times, leaving the final graph (V, T) with at least two connected components. Try dynamic programming. The base cases A[0] and A[1] are clearly correct. WIS Input: a path graph G with vertex set {v1 , v2 , . QE D Lemma 18.3 promises that it's safe to stop as soon as subproblem solutions stabilize, with Lk+1,v = Lk,v for some k and all v 2 V . A solution to the optimal binary search tree problem should reflect these frequencies. The second half of this book will teach you a third design paradigm: the dynamic programming paradigm. The "Field Guide to Algorithm Design" on page 201 provides a bird's-eye view of how greedy algorithms and dynamic programming fit into the bigger algorithmic picture. (See page 4 for the definition of a job's completion time in a schedule.) This problem considers the objective of minimizing the maximum lateness, maxj∈1 j ( ). Because i > j and jobs are indexed in nonincreasing order of weight-length ratio, wj wi `j `i and hence wi `j `i [{z} cost of exchange wj `i |{z} . Can we do better? QE D 9 Returning to the proof of Theorem 13.1, we are assuming that there is an optimal schedule ← of the given jobs with a strictly smaller sum of weighted completion times than the greedy schedule . And even when a greedy algorithm is correct, proving it can be difficult.4 Features and Bugs of the Greedy Paradigm 1. d) T1 is an optimal binary search tree for the keys {1, 2, . With n different schedules, computing the best one by exhaustive search is out of the question for all but the tiniest instances. 4 Introduction to Greedy Algorithms Now that my conscience is clear, let's look at some cherry-picked examples of problems that can be solved correctly with a judiciously designed greedy algorithm. We can't expect a running time better than quadratic (with a quadratic number of values to report), but there's a big gap between cubic and quadratic running times. Call this new graph G0 . If every symbol is encoded using 6 bits, the second symbol always starts with the 7th bit, the third symbol with the 13th bit, and so on. , k} of the vertices, with k serving as the measure of subproblem size, as well as an origin v and destination w. To prove that its output is also connected, we can argue that all its vertices belong to the same connected component of (V, T). One way to make trade-offs between the jobs is to minimize the sum of weighted completion times. In the following pseudocode, the variable s controls the current subproblem size.15 OptBST Input: keys {1, 2, . Quiz 16.2 What is the total weight of the output of the greedy algorithm when the input graph is the four-vertex path on page 105? , sn , and additional problem-specific data (all positive integers). 70 Minimum Spanning Trees The minimum bottleneck property makes this idea precise. 16.2 A Linear-Time Algorithm for WIS in Paths 16.2.1 Optimal Substructure and Recurrence To quickly solve the WIS problem on path graphs, we'll need to up our game. Main Idea #1 Prove that the output of the Huffman algorithm minimizes the average leaf depth over all ^-trees in which a and b are siblings. Case 2: n 2 S. Intuitively, you might expect that a prefix P 0 of a shortest path P would itself be a shortest path, albeit to a different destination: P' = (shortest?) s-w path lwv s v P = shortest s-v path Yet even assuming that this is true (which it is, as we'll see), in what sense is the prefix P 0 solving a "smaller" subproblem than the path P ? For example, if every element to be sorted is an integer with magnitude at most n10 (say), the RadixSort algorithm can be used to sort them in O(n) time. *17.2 149 Optimal Binary Search Trees The Search Tree Property For every object x: 1. P Typical dynamic programming algorithms fill in an array with the values of subproblems' solutions, and then trace back through the filled-in array to reconstruct the solution itself. c) For every vertex v reachable from the source s, there is a shortest s-v path with at most m edges. The key observation is that the straightforward implementation performs minimum computations, over and over, using exhaustive search. If we denote the root of T1 , then T1 's occupants are pushed one step further from the new root; otherwise, T2 's occupants suffer the same fate. A[n 2] + wn , an MWIS of Gn 1 is also an MWIS 16.3 117 A Reconstruction Algorithm 2. In our running example, the fourth and sixth objects have depth 0, the first and fifth objects have depth 1, and the second and third objects have depth 2. By our promotion criterion, this happens only when x's tree is merged with another tree that is at least as big. (If j = 0 or i = 0, interpret Pi,j as i · ↗gap or j · ↗gap , respectively.) Then Pm,n = min{Pm 1 + ↗xm yn , Pm 1 | 1,n 1 Case 1 + ↗gap , Pm,n 1 + ↗gap }. This assertion is not obvious and you should not trust it until I supply you with a proof. Their symbols have the same frequencies in p and p0 , so these leaves contribute the same amount to the average leaf depth of both trees. Can you formulate a recurrence to quickly solve a subproblem given solutions to a modest number of smaller subproblems? You should check that our dynamic programming solution extends to the case in which unsuccessful search times are also counted, provided the input specifies the frequencies of such searches. 2 In the first case, we know to exclude vn from our solution and can continue the reconstruction process from vn 1 . 118 Introduction to Dynamic Programming 3 2 1 4 6 5 the WIS Reconstruction algorithm includes v6 (forcing v5 's exclusion), includes v4 (forcing v3 's exclusion), excludes v2 , and includes v1 : prefix length i 0 1 2 3 4 5 6 0 3 3 4 9 9 14 include v1 exclude v2 include v4 exclude v3 include v6 exclude v5 16.4 The Principles of Dynamic Programming 16.4.1 A Three-Step Recipe Guess what? strings X and Y : | Consider an optimal alignment of the X + gaps Y + gaps {z common length ` } What could reside in the upper-right corner, in the last column of the first row? a) →(1)5 b) →(n) c) →(n2) d) 2→(n) (See Section 16.2.5 for the solution and discussion.) Quiz 16.4 implies that the exponential running time of our recursive WIS algorithm stems solely from its absurd redundancy, solving the same subproblems from scratch over, and over, and over, and over again. The base cases (n = 0, 1) are clearly correct. , xd) and y = (y1 , y2 , . , (vk 1 , vk) in a graph to include repeated vertices or, equivalently, to contain one or more cycles. How much space do you then really need to run each of the three algorithms, respectively? 1 The problem can be solved in O(n2) time by a divide-and-conquer algorithm that makes four recursive calls rather than two, where n is the number of vertices. 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 // Initialization X := {s}, T = ;, H := empty heap for every v 6= s do if there is an edge (s, v) 2 E then key(v) := csv , winner(v) := (s, v) else // v has no crossing incident edges key(v) := +1, winner(v) := N U LL Insert v into H // Main loop while H is non-empty do w⊢ := ExtractMin(H) add w⊢ to X add winner(w⊢) to T // update keys to maintain invariant for every edge (w⊢ , y) with y 2 V X do if cw⊢ y < key(y) then Delete y from H key(y) := cw⊢ y , winner(y) := (w⊢ , y) Insert y into H return T Each not-yet-processed vertex w records in its winner and key fields the identity and cost of the winner of its local tournament—the cheapest edge incident to w that crosses the frontier (i.e., edges (v, w) with v 2 X). For a fixed origin v and destination w, the set of allowable paths grows with k, and so Lk,v,w can only decrease as k increases. Indeed, the Bellman-Ford algorithm directly inspired the early Inter13 For reconstruction purposes, it's a good idea to add one line of code that caches with each vertex v the most recent predecessor w that triggered a Case 2update of the form A[i][v] := A[i 1][w] + `wv . // subproblems (i indexed from 1, j from 0) A := (n + 1) → (n + 1) two-dimensional array // base cases (i = j + 1) for i = 1 to n + 1 do A[i][i 1] := 0 // systematically solve all subproblems (i j) for s = 0 to n 1 do // s=subproblem size-1 for i = 1 to n s do // i + s plays role of j // use recurrence from Corollary 17.5 A[i][i + s] := Pi+s i+s k=i pk +minr=i {A[i][r 1] + A[r+1][i+s]} | {z } Case r return A[1][n] // solution to largest subproblem In the loop iteration responsible for computing the subproblem solution A[i][i + s], all the terms of the form A[i][r 1] and A[r + 1][i + s] correspond to solutions of smaller subproblems computed in earlier iterations of the outer for loop (or in the base cases). } Case 2 Because both c and items' sizes are integers, the residual capacity c si in the second case is also an integer. In general, an alignment is a way of inserting gaps into one or both input strings so that they have equal length: | X + gaps Y + gaps {z common length ` } We can then define the similarity of two strings according to the quality of their nicest alignment. Each edge e = (v, w) added by the algorithm is the cheapest one with endpoints in distinct connected components of the solution-so-far (as these are precisely the edges whose addition will not create a cycle). For example, if you invoke Insert four times to add objects with keys 12, 7, 29, and 15 to an empty heap, the ExtractMin operation will return the object with key 7. We conclude that the algorithm spends O(1) time solving each of the (n + 1)(C + 1) = O(nC) subproblems, for an overall running time of O(nC).22,23 Finally, as with WIS, the correctness of Knapsack follows by induction on the number of items, with the recurrence in Corollary 16.5 used to justify the inductive step. For whichever algorithm fares worse in this example, we can conclude that it is not always optimal. The latter condition can be checked in linear time using any reasonable graph search algorithm, like breadth- or depth-first search starting from v (see Chapter 8 of Part 2). By trying all the possibilities—two possibilities in the WIS and knapsack problems, and three in the sequence alignment problem. In math, this objective function translates to n X min wj Cj ( ), (13.1) j=1 where the minimization is over all n! possible schedules , and Cj ( ) denotes job j's completion time in the schedule . When w⊢ is moved from V X to X, edges of the form (w⊢ , y) with y 2V X cross the frontier for the first time; these are the new contestants in the local tournaments at the vertices of V X. , 6). Lemma 15.15 (Connecting Adjacent Vertices) Let T be the set of edges chosen by Kruskal up to and including the iteration that examines the edge e = (v, w). Implementing the First Main Idea We now have all our ducks in a row for proving the statement (*), that among all trees of Tab , the Huffman algorithm outputs one with the minimum-possible average leaf depth: 1. Theorem 13.1 (Correctness of GreedyRatio) For every set of positive job weights w1 , w2 , . If you give me a path P between the source vertex s and some destination vertex v containing at least n edges, I can give you back another s-v path P 0 with fewer edges than P and length no longer than that of P . Kruskal (Union-Find-Based) Input: connected undirected graph G = (V, E) in adjacency-list representation and a cost ce for each edge e 2 E. But then C {(v, w)} would be a v-w path in G, contradicting our assumption that v and w are in different connected components. a) 6; no b) 6; yes c) 8; no d) 8; yes (See Section 16.1.4 for the solution and discussion.) 16.1 107 The Weighted Independent Set Problem Chapters 13–15 spoiled us with a plethora of cherry-picked correct greedy algorithms, but don't forget the warning back on page 3: Greedy algorithms are usually not correct. For (b), the answer appears to be no. x2^ This quantity is the same as the average encoding length of the corresponding prefix-free code. The subproblems that can arise in this way correspond to the contiguous subsets of the original set of keys—sets of the form {i, i + 1, . To this end, the next lemma shows that once an edge (v, w) is processed by Kruskal, the solution-so-far (and, hence, the final output) necessarily contains a v-w path. The problem arises naturally in several application domains, including computer networking (try a Web search for "spanning tree protocol") and machine learning (see Section 15.8). Three reasons. As noted in Quiz 14.3, each merge decreases the number of trees in F by 1, resulting in n 1 iterations of the main loop. 15.5 79 Kruskal's Algorithm 15.5.2 Pseudocode With our intuition solidly in place, the following pseudocode won't surprise you. 13.2.1 The Setup In scheduling, the tasks to be completed are usually called jobs, and jobs can have different characteristics. Nielsen, and Charles Rackoff (Algorithmica, 2003). By the algorithm's greedy For Prim's algorithm, the relevant measure is the maximum cost of an edge in a path, and this measure cannot decrease as additional (positive- or negative-cost) edges are tacked onto the path. For example, a data point corresponding to a 100-by-100 pixel color image might be a 30000-dimensional vector, with 3 coordinates per pixel recording the intensities of red, green, and blue in that pixel.38 This section highlights a connection between one of the most basic algorithms in unsupervised learning and Kruskal's minimum spanning tree algorithm. Quiz 18.1 Consider an instance of the single-source shortest path problem with n vertices, m edges, a source vertex s, and no negative cycles. In any case, once the function F is chosen, the generic bottom-up clustering algorithm can be specialized to greedily merge the "most similar" pair of clusters in each iteration: Bottom-Up Clustering (Greedy) // Main loop while C contains more than k clusters do remove from C the clusters S1 , S2 that minimize F (S1 , S2) // e.g., with F as in (15.2) add S1 [S2 to C return C 41 Bottom-up clustering is only one of several common approaches to clustering. Here are two reasons among many. Algorithms Illuminated Part 3: Greedy Algorithms and Dynamic Programming Tim Roughgarden c 2019 by Tim Roughgarden All rights reserved. This is true for every choice of ⊢ , so is indeed optimal. *15.6 Speeding Up Kruskal's Algorithm via Union-Find As with Prim's algorithm, we can reduce the running time of Kruskal's algorithm from the reasonable polynomial bound of O(mn) (Proposition 15.12) to the blazingly fast near-linear bound of O(m log n) through the deft use of a data structure. Floyd-Warshall Input: directed graph G = (V, E) in adjacency-list or adjacency-matrix representation, and a real-valued length `e for each edge e 2 E. Summarizing, two and only two candidates are vying to be an MWIS: Lemma 16.1 (WIS Optimal Substructure) Let S be an MWIS of a path graph G with at least 2 vertices. Because S does not include the last vertex of G, it contains only vertices of Gn 1 and can be regarded as an independent set of Gn 1 (still with total weight W)—and not just any old independent set of Gn 1 , but a maximum-weight such set. b) P ⊢ need not have destination w. (Do you see why a negative gap penalty—that is, a reward for gaps—would make the problem completely uninteresting?) 3 Named after its inventors, Saul B. We continue to allow negative edge lengths and negative cycles in the input graph. Two, it provides an opportunity to study a new and useful data structure, the disjointset or union-find data structure. The first step of our plan is to solve two special cases of the general problem. How can we formalize this intuition? The completion time of a job depends only on the set of jobs preceding it (and not on their order), so job k is none the wiser and completes at the same time in both schedules. (We can ignore the fact that edges of the form (u, w⊢) with u 2 X get sucked into X and no longer cross the frontier, as we're not responsible for maintaining keys for vertices in X.) For a vertex y 2 V X, the new winner of its local tournament is either the old winner (stored in winner(y)) or the new contestant (w⊢ , y). Corollary 15.9 (Connectedness and Acyclicity Go Together) Let G = (V, E) be a graph and T ✓ E a subset of n 1 edges, where n = |V |. 13.4.2 Exchanging Jobs in a Consecutive Inversion Suppose, for contradiction, that the GreedyRatio algorithm produces the schedule and that there is an optimal schedule ⊢ with a strictly smaller sum of weighted completion times. In the first place I was interested in planning, in decision making, in thinking. Literacy with computer science's greatest hits. 16.5 127 The Knapsack Problem Corollary 16.5 (Knapsack Recurrence) With the assumptions and notation of Lemma 16.4, let Vi,c denote the maximum total value of a subset of the first i items with total size at most c. 154 Advanced Dynamic Programming Quiz 17.5 Suppose an optimal binary search tree for the keys {1, 2, . You'll learn several blazingly fast subroutines for processing data as well as several useful data structures for organizing data that you can deploy directly in your own programs. 18.1 Shortest Paths with Negative Edge Lengths 18.1.1 The Single-Source Shortest-Path Problem In the single-source shortest path problem, the input consists of a directed graph G = (V, E) with a real-valued length `e for each edge e 2 E and a designated origin s 2 V , which is called the source vertex or starting vertex. This definition makes perfect sense, but there's a more subtle problem: In the presence of a negative cycle, this version of the single-source shortest path problem is what's called an "N P -hard problem." Part 4 discusses such problems at length; for now, all you need to know is that N P -hard problems, unlike almost all the problems we've studied so far in this book series, do not seem to admit any algorithm that is guaranteed to be correct and to run in polynomial time.5 Both options are disasters, so should we give up? *17.2 Optimal Binary Search Trees In Chapter 11 of Part 2 we studied binary search trees, which maintain a total ordering over an evolving set of objects and support a rich set of fast operations. In the example above, the total weight of an MWIS in the original input graph is the value in the final array entry (14), corresponding to the independent set consisting of the first, fourth, and sixth vertices. The first option is to allow s-v paths that include one or more cycles. Not so with greedy algorithms, for which correctness proofs are more art than science—be prepared to throw in the kitchen sink. Let P 0 denote the first i 1 edges of P , and (w, v) its final hop: P' = s-w path (i-1 edges, length L - lwv) w lwv v s P = s-v path (i edges, length L) The prefix P 0 is an s-w path with at most i 1 edges and length L `wv .10 There cannot be a shorter such path: If P ⊢ were an s-w path with at most i 1 edges and length L⊢ < L `wv , appending the edge (w, v) to P ⊢ would produce an s-v path with at most i edges and length L⊢ + `wv < (L `wv) + `wv = L, contradicting the optimality of P for the original subproblem.11 This case analysis narrows down the possibilities for an optimal solution to a subproblem to a small number of candidates. At this point, the solution-so-far T looks like: 78 Minimum Spanning Trees 4 1 2 3 5 7 6 The two edges chosen so far are disjoint, so the algorithm is effectively growing two trees in parallel. For example, if every edge e has unit length `e = 1, a shortest path minimizes the hop count (i.e., number of edges) between its origin and destination.2 Or, if the graph represents a road network and the length of each edge the expected travel time from one end to the other, the single-source shortest path problem is the problem of computing driving times from an origin (the source vertex) to all possible destinations. 106 Introduction to Dynamic Programming 16.1.2 The Natural Greedy Algorithm Fails For many computational problems, greedy algorithms are a great place to start brainstorming. This reduction, which is called Johnson's algorithm and described in the bonus videos at www. a) O(1) b) O(log n) c) O(n) d) Not enough information to answer (See Section 15.6.5 for the solution and discussion.) With the solution to Quiz 15.7, we conclude that our quick-anddirty implementation of a union-find data structure fulfills the running time guarantees promised by Theorem 15.14 and Table 15.1. 15.6.5 Solutions to Quizzes 15.5–15.7 Solution to Quiz 15.5 Correct answer: (c or d). 56 Minimum Spanning Trees Like minimizing the sum of weighted completion times (Chapter 13) or the optimal prefix-free code problem (Chapter 14), the number of possible solutions can be exponential in the size of the problem.7 Could there be an algorithm that magically homes in on the minimum-cost needle in the haystack of spanning trees? , Sm of items PS1 , S2 , . So, while Prim's algorithm was constrained to choose the cheapest edge crossing the current frontier, Kruskal's algorithm is free to choose the cheapest remaining edge in the entire graph. Knapsack subproblems can shrink in two different senses (by removing an item or removing knapsack capacity), and so it goes with sequence alignment subproblems (by removing a symbol from the first or the second input string). Problem: All-Pairs Shortest Paths Input: A directed graph G = (V, E) with n vertices and m edges, and a real-valued length `e for each edge e 2 E. This in-degree could be as large as n 1 in a directed graph (with no parallel edges), but is generally much smaller, especially in sparse graphs. Quiz 18.6 Let G = (V, E) be an input graph. The leaves of the trees are in one-to-one correspondence with the symbols of ^. When confronted with a new problem, what's the most effective way to put your tools to work? See Part 4 for the full story. P Huffman's greedy algorithm maintains a forest, with leaves in correspondence to alphabet symbols, and in each iteration greedily merges the pair of trees that causes the minimum-possible increase in the average leaf depth. In our running example: a 4 c 1 b 3 5 2 d the edge (a, d) does not satisfy the MBP (every edge on the path a ! b ! d is cheaper than (a, d)), nor does the edge (c, d) (every edge on the path c ! a ! d is cheaper than (c, d)). When k = 3, the path 1 ! 2 ! 3 ! 5 becomes the unique eligible path; it has length 2 + (4) + 5 = 3. Obtain 0 from ⊢ by swapping the positions of i and j in the schedule (Figure 13.3). In the sequence alignment problem, how many relevant possibilities are there for the contents of the final column? The first iteration of the algorithm merges the nodes that correspond to the two symbols with the smallest frequencies—in this case, "C" and "D." After this iteration, the algorithm's forest contains only three trees, with the following sums of symbol frequencies: Symbol tree containing A tree containing B tree containing C and D Sum of Symbol Frequencies .60 .25 .05 + .10 = .15 The second two trees have the smallest sums of symbol frequencies, so these are the trees merged in the second iteration. In the tree above, the key "3" has a search time of 1, the keys "1" and "5" have search times of 2, and the keys "2" and "4" have search times of 3.10 Different search trees for the same set of objects result in different search times. Then, a negative cycle corresponds to an arbitrage opportunity. The first two parts of the series are not strict prerequisites for this one, though portions of this book do assume at least a vague recollection of big-O notation (covered in Chapter 2 of Part 1 or Appendix C of Part 2), divide-and-conquer algorithms (Chapter 3 of Part 1), and graphs (Chapter 7 of Part 2). This strategy would be hopeless unless the left and right subtrees of an optimal binary search tree were guaranteed to be optimal in their own right for the corresponding subproblems. P In the optimal binary search tree problem, the input is a set of n keys and nonnegative frequencies for them, and the goal is to compute a binary search tree containing these keys with the minimum-possible weighted search time. 7. One way to implement a queue is with a doubly-linked list. Assume that n 2; otherwise, the answer is obvious. (If no such path exists, define Lk,v,w as +1.) (For each k 2 {0, 1, 2, . In our solution to the WIS problem in n-vertex path graphs, we implemented the first step by identifying a collection of n + 1 subproblems. If you need to beef up your programming skills, there are several outstanding free online courses that teach basic programming. The simplest way to define "subproblem size" is as the number of keys in the input. If Li 1,v and all the relevant Li 1,w 's are +1, then v is unreachable from s in i or fewer hops, and we interpret the recurrence as computing Li,v = +1. Which is. Can you simplify the problem by processing the input using a for-free primitive, such as sorting or computing connected components? The inductive proof of correctness is similar to that for the WIS algorithm (Theorem 16.3).9 For example, for the input graph 8 If there is a tie (A[n 2] + wn = A[n 1]), both options lead to an optimal solution. Output: dist(v, w) for every vertex pair v, w 2 V , or a declaration that G contains a negative cycle. Which of the following greedy algorithms produces a schedule that minimizes the maximum lateness? The results of past computations are stored in a globally visible length(n + 1) array A, with A[i] storing an MWIS of Gi , where Gi comprises the first i vertices and the first i 1 edges of the original input graph (and G0 is the empty graph). The required output is, for every possible destination v 2 V , the minimum total length of a path from s to v. Teaching you these paradigms and their most famous instantiations is one of the major goals of this book series. If you must design a new algorithm from scratch, get calibrated by identifying the line in the sand drawn by the "obvious" solution (such as exhaustive search). Proof: We prove the contrapositive.8 If ^ has no consecutive inversions, the index of each job is at least 1 larger than the job that came before it. The QuickSort algorithm invokes a 16.4 The Principles of Dynamic Programming 121 algorithm keeps its options open, considering multiple ways of defining smaller subproblems and choosing the best of them.11 2. 16.3 A Reconstruction Algorithm The WIS algorithm in Section 16.2.4 computes only the weight possessed by an MWIS of a path graph, not an MWIS itself. First, after the edge addition, the connected components S1 and S2 fuse into a single component S1 [S2 , decreasing the number of components by 1. Here's an idea: The first time we solve a subproblem, why not save the result in a cache once and for all? (To get a v-w path, paste together paths from v to s and from s to w.) 15.2 61 Prim's Algorithm The algorithm Prim computes the minimum spanning tree in the four-vertex five-edge graph of Quiz 15.1, which means approximately. This observation suggests a cool opportunity for algorithm design: Given the search frequencies for a set of keys, what's the best binary search tree? Recall that in this type of proof, you assume the opposite of what you want to prove, and then build on this assumption with a sequence of logically correct steps that culminates in a patently false statement. Initializing the union-find data structure takes O(n) time. c) If all symbol frequencies are less than 0.33, all symbols will be encoded with at least two bits. The goal is to either compute the length of a shortest path from every vertex to every other vertex, or detect that the graph has a negative cycle. He was an invited speaker at the 2006 International Congress of Mathematicians, the Shapley Lecturer at the 2008 World Congress of the Game Theory Society, and a Guggenheim Fellow in 2017. Which one is it? This is equivalent to minimizing the weighted average of the jobs' completion times, with the averaging weights proportional to the wj 's. a) 2.11 b) 2.31 c) 2.49 d) 2.5 Problem 14.3 (H) What is the maximum number of bits that Huffman's greedy algorithm might use to encode a single symbol? i .... Hints or solutions to all of these problems (as indicated by an "(H)" or "(S)," respectively) are included at the end of the book. Compute the median edge cost in the input graph G. What We'll Cover Algorithms Illuminated, Part 3 provides an introduction to and numerous case studies of two fundamental algorithm design paradigms. Can you argue that a solution must be built up from solutions to smaller subproblems in one of a small number of ways? What title, what name, could I choose? White belts, however, still have a lot of training to do. The algorithm seeds X with an arbitrarily chosen starting vertex s; as we'll see, the algorithm is correct no matter 60 Minimum Spanning Trees which vertex it chooses.10 Each iteration is responsible for adding one new edge to T . In Chapter 14 we defined prefix-free codes and designed a greedy algorithm for computing the best-on-average code for a given set of symbol frequencies. It could be a gap, or it could be a symbol. This chapter assumes that the input graph is represented using adjacency lists, with an array of vertices, an array of edges, pointers from each edge to its two endpoints, and pointers from each vertex to its incident edges.2 15.1.2 Spanning Trees The input in the minimum spanning tree problem is an undirected graph G = (V, E) in which each edge e has a real-valued cost ce . Bellman-Ford Input: directed graph G = (V, E) in adjacency-list representation, a source vertex s 2 V , and a real-valued length `e for each e 2 E. In other words, once you know that an MWIS includes the last vertex, you know exactly what it looks like: It's an MWIS of the smaller graph Gn 2 , supplemented with the final vertex vn . We can reason about all of them in one fell swoop. , n} and v, w 2 V , → Lk 1,v,w (Case 1) Lk,v,w = min . The first iteration of the Huffman algorithm merges the leaves labeled "a" and "b" and thereafter treats them as an indivisible unit with total frequency pa + pb . 1 The abbreviation "i.e." stands for id est, and means "that is." 23 24 Huffman Codes 14.1.2 Variable-Length Codes When some symbols of the alphabet occur much more frequently than others, variable-length codes can be more efficient than fixedlength ones. Achieved, for example, by the code Symbol A B C D E Encoding 110 1110 0 1111 10 Hint for Problem 14.3: For a lower bound, consider symbol frequencies that are powers of 2. Our semantics for job weights certainly suggests that the higher-weight jobs should receive the smaller completion times, and this is in fact the case. 16.1.3 A Divide-and-Conquer Approach? QE D Putting it all together proves Theorem 15.11 for graphs with distinct edge costs: Proof of Theorem 15.11: Lemma 15.17 proves that the output of Kruskal's algorithm is a spanning tree. a) O(m + n) b) O(m log n) c) O(n2) d) O(mn) (See below for the solution and discussion.) Correct answer: (d). For example, any genomicist can tell you the typical frequency of each nucleobase (As, Cs, Gs, and Ts) in human DNA. We have our recurrence for solving a subproblem given solutions to smaller subproblems. P In the all-pairs shortest path problem, the input consists of a directed graph with edge lengths. By how much does a merger increase the average leaf depth? 19 Dijkstra's algorithm can substitute for the Bellman-Ford algorithm if edges' lengths are nonnegative, in which case the running time improves to O(mn log n). Conquer the subproblems recursively. e) P ⊢ need not have length less than L. *15.4 Prim's Algorithm: Proof of Correctness 73 states that every edge addition (v, w) is either type C or type F (and not both), depending on whether the graph already has a v-w path. QE D 15.6.4 Quick-and-Dirty Implementation of Union-Find The Parent Graph Under the hood, a union-find data structure is implemented as an array and can be visualized as a collection of directed trees. Lemma 18.4 (Bellman-Ford with No Negative Cycles) Under the assumptions and notation of Corollary 18.2, and also assuming that the input graph G has no negative cycles, Ln,v = Ln 1,v for every destination v, where n is the number of vertices in the input graph. By the time a loop iteration must compute the subproblem solution A[i][v], all values of the form A[i 1][v] or A[i 1][w] have already been 180 Shortest Paths Revisited computed in the previous iteration of the outer for loop (or in the base cases) and are ready and waiting to be looked up in constant time. What's the running time of the Find operation as a function of the number n of objects? C := ; // keeps track of current clusters for each x 2 X do add {x} to C // each point in own cluster // Main loop while C contains more than k clusters do remove clusters S1 , S2 from C // details TBA add S1 [S2 to C // merge clusters return C 96 Minimum Spanning Trees Each iteration of the main loop decreases the number of clusters in C by 1, so there are a total of |X| k iterations (Figure 15.9).41 merge merge merge merge Figure 15.9: In bottom-up clustering, each point begins in its own cluster and pairs of clusters are successively merged until only k clusters remain. These specify the penalties an alignment incurs in each of its columns; the total penalty of an alignment is the sum of its columns' penalties. The good news is that we can reuse all of them to also establish the correctness of another important MST algorithm, Kruskal's algorithm (Theorem 15.11 in Section 15.5).23 15.4.2 Fun Facts About Spanning Trees To warm up for the proof of Theorem 15.6, we'll prove some simple and useful facts about undirected graphs and their spanning trees. Prim's algorithm, by always choosing the eligible edge with minimum individual cost, is effectively striving to minimize the maximum edge cost along every path.20 19 A popular if more abstract approach to proving the correctness of Prim's (and Kruskal's) algorithm is to use what's known as the "Cut Property" of MSTs; see Problem 15.7 for details. , k} as internal vertices22 ; and (iv) does not contain a directed cycle. As usual, shortest paths can be reconstructed by tracing back through the final array A computed by the Bellman-Ford algorithm.13 18.2.8 Internet Routing The Bellman-Ford algorithm solves a more general problem than Dijkstra's algorithm (because it accommodates negative edge lengths). Under 76 Minimum Spanning Trees our assumption that edges' costs are distinct, the cost of e2 must be strictly larger: cxy > cvw . Our vision is of a dynamic programming algorithm that tries all possibilities for the root, recursively computing or looking up the optimal left and right subtrees for each possibility. We can therefore sneakily rearrange terms to obtain L(T) L(T ⊢) = (px pa) · (depth of x in T depth of a in T) | {z } | {z } 0 0 + (py pb) · (depth of y in T depth of b in T) {z } | {z } | 0 0. Now imagine starting from an empty graph (with vertices but no edges) and adding edges to it one by one. For each i = 1, 2, . 2 This special case of the single-source shortest path problem can be solved in linear time using breadth-first search; see Section 8.2 of Part 2. Of all the search trees for a given set of objects, which is the "best?" We asked this question in Chapter 11 of Part 2, and the answer there was a perfectly balanced tree: The rationale? For the inductive step (n 2), the inductive hypothesis guarantees that S1 and S2 are indeed MWISs of Gn 1 and Gn 2 , respectively. 16.2.4 An Iterative Bottom-Up Implementation As part of figuring out how to incorporate caching into our recursive WIS algorithm, we realized that there are exactly n + 1 relevant subproblems, corresponding to all possible prefixes of the input graph (Quiz 16.4). Which of the following statements are true? 99 Problems P The second step is to use an exchange argument to prove that a spanning tree in which every edge satisfies the MBP must be an MST. 7 Can we do better? A big reason for the success of online courses is the opportunities they provide for participants to help each other understand the course material and debug programs through discussion forums. The simplest possible example for ruling out one of them would be a problem instance with two jobs, having different weights and lengths, such that the two algorithms schedule the jobs in opposite orders. 122 Introduction to Dynamic Programming recursing on subproblems that are barely smaller than the input (like in the WIS algorithm), if necessary for correctness. You should always be on the lookout for reductions. In other words, a job's completion time in a schedule is the total time that elapses before the job has been fully processed. This means that ^ is the same as the schedule computed by the greedy algorithm. Correct answer: (a). If an edge e 2 E is the cheapest edge crossing a cut (A, B), e belongs to every MST of G.42 In other words, one way to justify an algorithm's inclusion of an edge e in its solution is to produce a cut of G for which e is the cheapest crossing edge.43 (a) Prove the Cut Property. , k} (as you should check, by induction on k). The problem in (a) can be solved by a generalization of the NW algorithm that keeps track of whether an inserted gap is the first in a sequence of gaps (in which case it carries a penalty of a + b) or not (in which case the additional penalty is a). Graham and Pavol Hell (Annals of the History of Computing, 1985). 90 Minimum Spanning Trees promoted demoted 4 6 1 2 + 3 5 4 = 6 1 2 3 5 A Union operation performs two Find operations and O(1) additional work, so its running time matches that of Find. For the WIS problem with an input path graph G = (V, E), the natural high-level approach is (ignoring the base case): WIS: A Divide-and-Conquer Approach G1 := first half of G G2 := second half of G S1 := recursively solve the WIS problem on G1 S2 := recursively solve the WIS problem on G2 combine S1 , S2 into a solution S for G return S The devil is in the details of the combine step. 9. For the rest of the proof, we denote by a and b the two symbols of ^ with the smallest and second-smallest frequencies, respectively. The gist is that a queue is a data structure for maintaining a list of objects, and you can remove stuff from its front or add stuff to its back in constant time. These sets can merge over time, but they can never split: merge merge merge merge merge In our application of speeding up Kruskal's algorithm, the objects will correspond to the vertices of the input graph and the sets in the partition to the connected components of the solution-so-far T : Checking whether T already contains a v-w path then boils down to checking whether v and w belong to the same set of the partition (i.e., to the same connected component). Thomson, x lemma, 10 length (of an edge), 168 longest common subsequence, 164 longest common substring, 164 machine learning supervised learning, 94 unsupervised learning, see clustering mathematical background, x MBP, see minimum spanning tree, minimum bottleneck property memoization (in dynamic programming), 113 MergeSort, 1, 2 Meyer, Albert R., x minimum bottleneck property, see minimum spanning tree, minimum bottleneck property minimum spanning tree Index Cut Property, 69, 101, 205 Cycle Property, 102, 204 exchange argument, 75 history, 57 in directed graphs, 53 in disconnected graphs, 55 in linear time?, 206 Kruskal's algorithm, see Kruskal's algorithm minimum bottleneck property, 69–71, 75, 101 Prim's algorithm, see Prim's algorithm reductions to, 100 uniqueness, 101 with distinct edge costs, 69, 75 with non-distinct edge costs, 72, 91, 101, 205 with parallel edges, 55 minimum-bottleneck spanning tree, 102, 206 MP3, 23, 27 MST, see minimum spanning tree MWIS, see weighted independent set Needleman, Saul B., 139 Needleman-Wunsch (NW) score, see sequence alignment Nielsen, Morten N., 2 N P -hard problem, 170, 202 optimal binary search trees, 148– 162 correctness, 160 dynamic programming algorithm (OptBST), 159 Knuth's optimization, 161 optimal substructure, 153– 156 problem definition, 152 reconstruction, 161 recurrence, 157 215 running time, 160 subproblems, 158 vs. In each iteration of the algorithm, the trees T1 and T2 with the smallest sums of symbol frequencies can be identified and removed using a constant number of operations at the fronts of Q1 and Q2 . In the knapsack problem, there are two (by having fewer items, or less knapsack capacity). P In the knapsack problem, given n items with values and sizes and a knapsack capacity C (all positive integers), the goal is to select the maximum-value subset of items with total size at most C. , yj , the first j symbols of Y . This problem is more challenging than the optimal prefix-free code problem, but it is no match for the power of the dynamic programming paradigm. , k 1} and a minimum-length cycle-free k-w path with all internal vertices in {1, 2, . In sparse graphs (with m = O(n) or close to it), this approaches the best we could hope for (as merely writing down the output already requires quadratic time). Algorithmic thinking is increasingly useful and prevalent in disciplines outside of computer science, including biology, statistics, and economics. Quiz 15.1 What is the minimum sum of edge costs of a spanning tree of the following graph? Solution to Quiz 16.2 Correct answer: (a). Obtain the (n 1)-vertex path graph Gn 1 from G by plucking off the last vertex vn and the last edge (vn 1 , vn). First Edition Cover image: Untitled, by Johanna Dickson ISBN: 978-0-9992829-4-6 (Paperback) ISBN: 978-0-9992829-5-3 (ebook) Library of Congress Control Number: 2017914282 Soundlikeyourself Publishing, LLC New York, NY www.algorithmsilluminated.org In memory of Stephen H. The overall running time O(nC) → O(1) + O(n) = O(nC) remains the same. Do not confuse it with the actual (undirected) input graph in Kruskal's algorithm. Let T1 and T2 denote the left and right subtrees of the root. (c) Repeat (b) for Kruskal's algorithm. independent of T Proof: Leaves of T not labeled a or b occupy the same position in T 0 . After reading these books, you'll know exactly what they mean. , n} and frequencies p1 , p2 , . Both paradigms recursively solve smaller subproblems and combine the results into a solution to the original problem. And it doesn't really change the problem: Connecting every pair of vertices is the same thing as connecting some vertex s to every other vertex. 201 202 Epilogue: A Field Guide to Algorithm Design Most likely, all will fail. Assuming we perform at least a constant amount of work solving each subproblem, the number of subproblems is a lower bound on the running time of our algorithm. For (d), see Problem 14.2. Solution to Problem 14.5: Sort the symbols by frequency and insert them in increasing order into a queue Q1 .34 Initialize an empty queue Q2 . This implementation eschews heaps in favor of an even simpler data structure: a queue (actually, two queues). To the extent that there are recurring themes in correctness proofs of greedy algorithms, we will emphasize them as we go along. What are the completion times of the three jobs in this schedule? Lines 2–7 initialize these values for all the vertices other than s so that the invariant is satisfied and insert these vertices into a heap. These values are ready and waiting to be looked up in constant time. Output: a partition of X into k non-empty sets. Inspired by our proof of Theorem 13.1, the plan is to exchange one edge for another to produce a spanning tree T 0 with total cost even less than T ⊢ , thereby contradicting the alleged optimality of T ⊢ . (Choose the strongest statement that is guaranteed to be true.) a) O(m + n) b) O(kn) c) O(km) d) O(mn) Problem 18.4 (S) For the input graph 2 1 3 5 2 1 4 3 4 what are the final array entries of the Floyd-Warshall algorithm from Section 18.4? Problem 18.5 (S) For the input graph 2 1 5 2 1 -4 3 -3 4 what are the final array entries of the Floyd-Warshall algorithm? Let Gi denote the subgraph of G comprising its first i vertices and i 1 edges. When i is 0 or 1, there are no s-t paths with i edges or less, and no solutions to the corresponding subproblems. The algorithm halts once one tree remains, which is after n 1 mergers. Many such implementations are freely available on the Web as well. I hope they exude a contagious enthusiasm for algorithms that, alas, is impossible to replicate fully on the printed page. Only one type of event can increment x's depth: a Union operation in which the root of x's tree in the parent graph gets demoted. a) (i) Not enough information to answer; (ii) goes up; (iii) goes down. By Corollary 14.5, the ^-tree (T 0) is optimal for the original problem with input ^ and p. One simple example is: Length Weight Job #1 `1 = 5 w1 = 3 Job #2 `2 = 2 w2 = 1. We call this a type-C edge addition ('C' for "cycle"). The running time bound of O(nC) is impressive only if C is small, for example, if C = O(n) or ideally even smaller. (Let n = |^| denote the number of 14.3 35 Huffman's Greedy Algorithm symbols.7) a) n 1 b) n c) (n+1)n 2 d) Not enough information to answer (See Section 14.3.7 for the solution and discussion.) 14.3.2 Huffman's Greedy Criterion For a given set of symbol frequencies {pa }a2^ , which pair of trees should we merge in each iteration? * * * * * * * * * * * Kruskal's algorithm begins with the empty edge set and each vertex isolated in its own connected component, just as single-link clustering begins with each data point in its own cluster. (To be covered in Part 4.) Iterate over the algorithm design paradigms again, this time looking for opportunities for fast heuristics (especially with greedy algorithms) and better-thanexhaustive-search exact algorithms (especially with dynamic programming). d) The NW algorithm remains well defined and correct after reversing the order of the for loops, but the Knapsack algorithm does not. The work performed in each iteration of the main loop in the Huffman algorithm is dominated by two minimum computations, so a light bulb should go off in your head: This algorithm calls out for a heap! Using a heap to speed up these minimum computations decreases the running time from O(n2) to O(n log n), which qualifies as a blazingly fast implementation.8 We can do even better. P A dynamic programming algorithm that solves at most f (n) different subproblems, using at most g(n) time for each, and performs at most h(n) postprocessing work to extract the final solution runs in O(f (n) · g(n) + h(n)) time, where n denotes the input size. On which goods and services should you spend your paycheck to get the most value? I strongly encourage you to implement as many of the algorithms in this book as you have time for. We also use mathematical analysis as needed to understand how and why algorithms really work. Readers of these books have the same opportunity, via the forums available at www.algorithmsilluminated.org. Under our assumption that the input graph G is connected, there's no way for the algorithm to get stuck; if there were ever an iteration with no edges of G crossing between X and V X, we could conclude that G is not connected (because it contains no path from any vertex in X to any vertex in V X). Dijkstra's shortest-path algorithm, 60, 69, 70 vs. In this case, the path P can immediately be interpreted as a solution to a smaller subproblem with prefix length k 1, still with origin v and destination w. If at least one of k or m is different from both i and j, and hence appears either before both i and j or after both i and j in both schedules, the swap has no effect on the relative order of k and m (see Figure 13.4). Do you see how to tweak the algorithm to obtain a per-iteration time bound of O(m) without this assumption? Expanding the definition (14.1) and canceling the terms that correspond to leaves other than a, b, x, y, we have X L(T) L(T ⊢) = pz · (depth of z in T depth of z in T ⊢) . QE D Because the correspondence between ^0 -trees and the ^-trees in Tab preserves the average leaf depth (up to the tree-independent constant pa + pb), it associates the optimal ^0 -tree with the optimal ^-tree in Tab : ↲ best ^-tree in Tab ⌦ best ^0 -tree ↲ second-best ^-tree in Tab ⌦ second-best ^0 -tree .. This dramatic speedup will be easier to see after we reformulate our top-down recursive algorithm as a bottom-up iterative one—and the latter is usually what you want to implement in practice, anyway. So: 1. For example, the problem of computing the 101 Problems median element of an array reduces to the problem of sorting the array. Quiz 18.3 What's the running time of the Bellman-Ford algorithm, as a function of m (the number of edges) and n (the number of vertices)? Videos. , n} of items with the maximum-possible sum i2S vi of values, subject to having P total size i2S si at most C. Proof: If the input graph does not have a negative cycle, then: (i) Floyd-Warshall correctly computes all shortest-path distances; and 27 Unlike most of our graph algorithms, the Floyd-Warshall algorithm is equally fast and easy-to-implement for graphs represented with an adjacency matrix (where the (v, w) entry of the matrix is `vw if (v, w) 2 E and +1 otherwise) as for graphs represented with adjacency lists. The objects and connections could represent something physical, like computer servers and communication links between them. In my experience, most people initially find dynamic programming difficult and counterintuitive. If the goal is to cluster a diverse collection of images according to their subject, a larger value of k should be used. 46 Huffman Codes Proposition 14.3 (Preservation of Behavior of Huffman) The output of the Huffman algorithm with input ^ and p is (T 0), where T 0 is the output of the Huffman algorithm with input ^0 and p0 . 2 For example, there are n! different prefix-free codes that encode one symbol using one bit ("0"), another using two bits ("10"), another using three bits ("110"), and so on. Whenever you have a scarce resource that you want to use in the smartest way possible, you're talking about a knapsack problem. The inner "min" implements the exhaustive 18.2 The Bellman-Ford Algorithm 177 search inside Case 2 over all possible choices for the final hop of a shortest path. ⊢ is an 13.4 19 Proof of Correctness a) 2 has one fewer inversion than 1. 14.3 Huffman's Greedy Algorithm 14.3.1 Building Trees Through Successive Mergers Huffman's big idea back in 1951 was to tackle the optimal prefix-free code problem using a bottom-up approach.6 "Bottom-up" means starting with n nodes (where n is the size of the alphabet ^), each labeled with a different symbol of ^, and building up a tree through successive mergers. Then, S is either: (i) an MWIS of Gn 1; or (ii) an MWIS of Gn 2, supplemented with G's final vertex vn . Both algorithms admit blazingly fast implementations, using the heap and union-find data structures, respectively. Another good idea is union-by-rank, which denotes the root of the tree with the smaller height (breaking ties arbitrarily). Let P denote the total penalty incurred by this alignment. , n} and v, w 2 V .) There are (n + 1) · n · n = O(n3) subproblems, which is a linear number for each of the n2 values in the output. 39 With these semantics, it's arguably more accurate to call f a dissimilarity function. Fix from now on an input, with alphabet ^ and symbol frequencies p, and let a and b denote the symbols with the smallest and second-smallest frequencies, respectively. Option #2: Forbid cycles. This follows from an inductive argument similar to the one in footnote 4. (Choose the strongest true statement.) a) For every vertex v reachable from the source s, there is a shortest s-v path with at most n 1 edges. Quiz 16.1 How many different independent sets does a complete graph with 5 vertices have? P The key idea in the Floyd-Warshall algorithm is to parameterize subproblems by a prefix of k vertices (in addition to an origin and a destination) and consider only cycle-free paths with all internal vertices in {1, 2, . Its second advantage is that it is more "distributed" than Dijkstra's algorithm, and for this reason has played the more prominent role in the evolution of Internet routing protocols.14 Evaluating the recurrence (18.1) at a vertex v requires information only about vertices directly connected to v: the vertices w with an edge (w, v). a) 0 b) −gap · (length of X) c) +1 d) undefined (See Section 17.1.8 for the solution and discussion.) 17.1.4 Recurrence Quiz 17.3 handles the base case of an empty input string. To fill in an entry of the ith column, the algorithm compares the entry immediately to the left (corresponding to case 1) to vi plus the entry one column to the left and s rows down (case 2). See Problem 14.5 for more details. The one difference is that single-link clustering stops once there are k clusters, while Kruskal's algorithm continues until only one connected component remains. Show how to quickly and correctly infer the final solution from the solutions to all of the subproblems. An optimal solution to the WIS problem is called a maximum-weight independent set (MWIS). 16.5 The Knapsack Problem 16.5.2 125 Optimal Substructure and Recurrence To apply the dynamic programming paradigm to the knapsack problem, we must figure out the right collection of subproblems. We set out to solve the minimum spanning tree problem, not to achieve the minimum bottleneck property. In the three-vertex example above, detaching s to every edge length would change the shortest path from s ! v ! t to s ! t. In this case, the path P can immediately be interpreted as a solution to the smaller subproblem with edge budget i 1 (still with destination v). Properly implemented, it is competitive with Prim's algorithm in both theory and practice. Corollary 16.2 then implies that A[i] is computed correctly, as well. Alternatively, can you find a way to remove the explicit loop? Whenever you encounter a seemingly new problem, always ask: Is the problem a disguised version of one you already know how to solve? 16.4 Dynamic Programming vs. . In the proof of Theorem 14.2, we take P (k) as the statement: "the Huffman algorithm correctly solves the optimal prefix-free code problem whenever the alphabet size is at most k." Analogous to a recursive algorithm, a proof by induction has two parts: a base case and an inductive step. , xi , the first i symbols of X, and Yj = y1 , y2 , . Form a complete binary tree of depth ` . In each iteration, it greedily chooses the cheapest edge that expands the reach of the solution-so-far. 23 130 residual capacity ri Introduction to Dynamic Programming 6 0 3 3 7 8 0 3 3 8 4 0 3 3 4 3 4 0 0 2 4 4 2 0 0 4 0 1 0 0 0 0 0 0 0 0 0 1 2 3 4 prefix length i Knapsack computes these entries column by column (left to right, and within a column from bottom to top. For example, if " = {A, B, C, D}, we start with what will be the leaves of the tree: 6 This was for David A. Let T be an MST and P a shortest path from some vertex s to some other vertex t. But there are several general design paradigms that can help you solve problems from many different application domains. As we'll see, such thought experiments can light up a trail that leads directly to an efficient algorithm. (18.1) min(w,v)2E {Li 1,w + `wv } (Case 2) The order "min" in the recurrence implements the exhaustive search over Case 1 and Case 2. a) AB b) CD c) AA d) We need enough information to answer (See Section 14.6 for the solution and discussion.) Open Quiz 14.1 is that, with variable-length codes and no further precautions, it can be unclear where one symbol starts and 14.1 25 Codes the next one begins. There is one Union operation for each edge added to the output which, as an acyclic graph, has at most n 1 edges (Corollary 15.8). 18.3 185 The All-Pairs Shortest Path Problem We can do better. As in the original implementation, *15.6 Speeding Up Kruskal's Algorithm via Union-Find 85 the sorting step requires O(m log n) time (see Quiz 15.4). You'll get lots of practice describing and reasoning about algorithms. type-F edge addition, 72 Stanford Lagunita, x starred sections, viii, 41 Strassen's algorithm, 1 Tardos, Éva, 137 test cases, xi Tetris, 166 theorem, 10 transitive closure (of a binary relation), 186 tree binary, 28 binary search, 148 formal, 34 internal node, 28 leaf, 28 optimal binary search, see optimal binary search tree recursion, 115 union-find (Find, 82, 86 Initialize, 82, 86 Union, 83, 87, 89 inverse Ackermann function, 83 operation running times, 83, 90 parent graph, 85 path compression, 83 quick-and-dirty implementation, 85–90 raison d'être, 82 scorecard, 83 speeds up Kruskal's algorithm, 83–85 state-of-the-art implementations, 83 supported operations, 82–83 union-by-size, 83, 89 Unix, 164 upshot, viii vertex (of a graph), 52 in-degree, 187 independent set in general graphs, 134 problem definition, 105 weighted independent set, see weighted independent set (WIS), 114 failure of greedy algorithms, 106–107 greedy algorithm, 108–110 recurrence, 110–112, 113 running time, 111, 113 why bother?, 10 with WIS, see weighted independent set (WIS) Wunsch, Christian D., 139 YouTube, x For example, in the penultimate merge above, the depth of the nodes labeled "C" and "D" increases from 1 to 2, and the depth of the node labeled "B" increases from 0 to 1. Challenge Problems Problem 15.6 (S) Prove the converse of Theorem 15.6: If T is an MST of a graph with real-valued edge costs, every edge of T satisfies the minimum bottleneck property. For example, for the input graph 3 2 1 4 6 5 you should check that the final array values are: prefix length i 0 1 2 3 4 5 6 0 3 9 16 0 7 0 1 1 2 3 4 4 6 9 18 At the conclusion of the WIS algorithm, each array entry A[i] stores the total weight of an MWIS of the graph Gi that comprises 16.2 A Linear-Time Algorithm for WIS in Paths 115 the first i vertices and i 1 edges of the input graph. Suppose we only want to compute the value of an optimal solution and don't care about reconstruction. 1 edges. How many candidates, exactly? (How do you do this in linear time? 15 "RIP" stands for "Routing Information Protocol." If you're looking to nerd out, the nitty-gritty details of the RIP and RIP2 protocols are described in RFCs 1058 and 2453, respectively. At the end of each chapter you'll find several relatively straightforward questions for testing your under- xi Preface standing, followed by harder and more open-ended challenge problems. 1 , . , P with the maximum-possible total value v + i (2S1 P i2S2 vi + · · · + v , subject to the knapsack capacities i2S1 si C1 , Pi2Sm s Pi2S2 si C2 , . Dynamic programming is a particularly empowering technique to acquire, as it often leads to efficient solutions beyond the reach of anyone other than serious students of algorithms. 15.8 97 Application: Single-Link Clustering Single-link clustering refers to greedy bottom-up clustering with the best-case similarity function (15.2). Prior to joining Columbia, he spent 15 years on the computer science faculty at Stanford, following a PhD at Cornell and a postdoc at UC Berkeley. The new graph G0 has a negative cycle reachable from s if and only if G has a negative cycle. (children of a common parent, respectively The Huffman algorithm outputs a tree of Tab , and we want to prove that it is the best such tree (*) among all trees in Tab . Implementing the Second Main Idea The second part of the inductive step is easier and based on an exchange argument. Programming Problems Problem 14.6 Implement in your favorite programming language the Huffman algorithm from Section 14.3 for the optimal prefix-free code problem. Quiz 16.6 Which of the following statements hold for the set S (Choose all that apply.) (n)? The possible final hops are the incoming edges at v. Each iteration of the algorithm evaluates the recurrence (18.1) at each vertex, using the values computed in the previous iteration. . n) with nonnegative frequencies p1 , p2 , . The point of this case analysis is to narrow down the possibilities for an optimal solution to three and only three candidates. 15.5 77 Kruskal's Algorithm Kruskal's algorithm "In this chapter, we'll learn about the optimality of Huffman codes, as well as a blazingly fast algorithm for computing them. This incurs a penalty of −gap per gap, and the total penalty of −gap times the length of X. Each iteration of the algorithm chooses two of the trees in the current forest and merges them by making their roots the left and right children of a new unlabeled internal node. The responsibility of an algorithm is to compute, for every possible destination v, the minimum length dist(s, v) of a directed path in G from s to v. a) Neither T1 nor T2 need be optimal for the keys it contains. Kruskal's algorithm, 77 Prim's algorithm, 57 Robert C., 57 216 principle of parsimony, 202 programming, x, 37 programming problems, xi, 2 proof by contradiction, 13 by contraposition, 15 by induction, 37, 40, 10 proposition, 10 pseudocode, 38, 37 QDE D (q.e.d.), 15 queue (data structure), 41 QuickSort, 1 2 quizzes, x Rackoff, Charles, 2 Radcliffe, RI, 4 randomized algorithm, 202 recurrence, 110 recursion tree, 115 reduction, 100, 202 scheduling, 4 GreedyDiff, 9 completion time, 4 correctness of GreedyRatio, 13–19 exchange argument, 15 greedy algorithms, 8–10 n jobs, 4 tie-breaking, 15 sum of weighted completion times, 5 with ties, 12 Schrijver, Alexander, 172 search tree, see binary search tree sequence alignment, 137–146 alignment, 138 applications, 137 correctness, 146 dynamic programming algorithm (NW), 144 gap, 138 Index Needleman-Wunsch (NW) score, 139 optimal substructure, 140 penalties, 138 problem definition, 139 running time, 145 subproblems, 144 variations, 146 Shimbel, Alfonso, 172 shortest paths all-pairs, 185 all-pairs (dense graphs), 196 all-pairs (sparse graphs), 187, 196 and Internet routing, 168 and transitive closure, 186 Bellman-Ford algorithm, see Bellman-Ford algorithm Floyd-Warshall algorithm, see Floyd-Warshall algorithm, 172 Johnson's algorithm, 196 problem definition (all-pairs), 185 problem definition (singlesource), 170 reduction from all-pairs to single-source, 186, 187 single-source, 168, 169, 171 with negative edge lengths, 169, 170 no negative cycles, 172 with parallel edges, 168 single-source shortest paths, see shortest paths single-source, x, 203–210 sorting, 2 in linear time, 87, 217 Index lower bound for comparison-based sorting algorithms, 44 spanning tree, see MST 2-cycle, 175 source vertex, 171 Sparse graph, 50 of an MST, 169 of a spanning tree, 50 number of edges, 70 type-C vs. Implementing and using these algorithms will stretch and improve your programming skills. 86 Minimum Spanning Trees 2 6 4 1 5 3 In general, the sets in the partition maintained by the data structure will correspond to the connected components of the current solution. each set inheriting the parent graph, with each set inheriting the name of its root object. For a more detailed look into the book's contents, check out the "Upshot" sections that conclude each chapter and highlight the most important points. The number of independent sets of a path graph grows exponentially with the number of vertices (do you see why?), so there is no hope of solving the problem via exhaustive search, except in the tiniest of instances. (If no such set exists, the algorithm should correctly detect that fact.) 136 Introduction to Dynamic Programming b) Given a positive integer capacity C and an item budget k 2 {1, 2, . 5 — if items in an optimal solution c := C // remaining capacity for i = n downto 1 do if si c and A[i 1][c si] + vi A[i 1][c] then S := S [{i} // Case 2 wins, include i c := c si // reserve space for it // else skip i, capacity stays the same return S The Knapsack Reconstruction postprocessing step runs in O(n) time (with O(1) work per iteration of the main loop), which is much faster than the O(nC) time used to fill in the array in the Knapsack algorithm.24 For instance, tracing back through the array from the example on page 130 yields the optimal solution {3, 4}: remaining capacity c exclude exclude include include 6 0 3 3 7 8 0 3 3 8 4 0 3 3 4 3 4 0 0 2 4 4 2 0 0 4 0 1 0 0 0 0 0 0 0 0 1 2 3 4 prefix length i 24 In the notation of (16.1), postprocessing with the Knapsack Reconstruction the Knapsack algorithm with f (n) = O(n), g(n) = O(1), and h(n) = O(n), for an overall running time of O(n C). OPT 2 2 Section 15.4 that an edge e = (v, w) in a graph G satisfies the MBP if and only if every v-w path in G has an edge with cost at least ce . 33 If you go on to a deeper study of algorithms, beyond the scope of this book series, you can learn several more problems that show up in disguise all the time. , n, Pij = min {Pi | + vi xj yj , P i { j } 1,j 1 Case 1 + −gap · Pi, j 1 + −gap}. The "shortest s-v path" would then be whichever one has the smallest length. There are some very cool connections between Kruskal's algorithm and widely-used clustering algorithms (see Section 15.8). The number of possible clusterings with n, so even for modest values of n there is no hope of exhaustively searching through all of them.2 But surprisingly, the problem can be solved efficiently using a slick greedy algorithm. The solution to Quiz 18.1 hints that perhaps not all of them are important. 21 Problems P From where a greedy algorithm is going to work, and the solution to Quiz 18.1 hints at all of them are important. 21 Problems Problem 15.9 Implement in your favorite programming language the Prim and Kruskal algorithms. As the more sophisticated paradigm, dynamic programming applies to a wider range of problems than divide-and-conquer, but it is also more technically demanding to apply (at least until you've had sufficient practice). Problem 13.2 (H) Continuing Problem 13.1, P consider instead the objective of minimizing the total lateness, 1j=1 j. Moreover, by the search tree property, given the root r we know the demographics of its two subtrees—the keys less than r belong to the root's left subtree, and those greater than r to its right subtree. Hopes: Three Supported Operations Insert: given a heap H and a new object x, add x to H. Quiz 16.3 What is the asymptotic running time of the recursive WIS algorithm, as a function of the number n of vertices? (If the input graph G has no negative cycle, P is a shortest s-v path. Suppose we modified the Dijkstra algorithm to consist of any one of the assignments and notation of Lemma 18.1, let Li,v denote the minimum length of an s-v path with at most i edges, with cycles allowed. Or, in recurrence form: Corollary 18.7 (Floyd-Warshall Recurrence) With the assumptions and notation of Lemma 18.6, let Lk,v,w denote the minimum length of a cycle-free v-w path with all internal vertices in {1, 2, . Thus, a light bulb should go off in your head: Prim's algorithm calls out for a heap! *15.3 Speeding Up Prim's Algorithm via Heaps 15.3.2 63 The Heap Data Structure A heap maintains an evolving set of objects with keys and supports several fast operations, of which we'll need three. , k 1]. But will this ever happen? , n}. (See www.algorithmsilluminated.org for test cases and challenge data sets.) 2 Chapter 14 Huffman Codes Everybody loves compression. . N}}. 4 Algorithms have shaped the development of computational genomics as a field. For the WIS problem, perhaps the most natural greedy algorithm is an analog of Kruskal's algorithm: Perform a single pass over the vertices, from best (high-weighted) to worst (lowest-weighted), adding a vertex to the solution-so-far as long as it doesn't conflict with a previously chosen vertex. Lemma 13.2 (Non-Greedy Schedules Have Inversions) Every schedule ' different from the greedy schedule has at least one consecutive inversion. Needleman and Christian D. As always, it's our duty to ask the question: Can we do better? A "prefix" of the items (items that corresponds to the first i items in our arbitrary ordering (for some i 2 {0, 1, 2, . Here's an example of a prefix-free code for the alphabet " = {A, B, C, D} that is not fixed-length: Symbol A B C D Encoding 0 10 110 111 Because "0" is used to encode A, the encodings of the other three symbols must start with a "1." Because B is encoded as "10," the encodings of C and D begin with "11." 14.1.4 Benefits of Prefix-Free Codes Variable-length prefix-free codes can be more efficient than fixed-length codes when the symbols have very different frequencies. Quizzes with solutions and explanations are scattered throughout the text; when you encounter one, I encourage you to pause and think about the answer before reading on. This idea will be particularly important for some of the harder problems studied in Chapters 17 and 18. The best way to get a feel for greedy algorithms is through examples. For the general problem, cutting-edge research suggests that the answer might be "no." Intrepid readers should check out the paper "Edit Distance Cannot Be Computed in Strongly Subquadratic Time (unless SETH is false)," by Arturs Backurs and Piotr Indyk (SIAM Journal on Computing, 2018). , n} For now, we focus on computing the total weight of an MWIS for the subgraph Gi. In the notation of (16.1), f (n) = O(n2), g(n) = O(n), and h(n) = O(n). Then, P is either: 10 The path P has exactly i edges. For example, Cayley's formula is a famous result from combinatorics stating that the n-vertex complete graph (in which all the n2 possible edges are present) has exactly nn 2 different spanning trees. (The easy case is when G0 is connected; how do you recurse if G0 is not connected?) For the running time analysis, use induction or case 2 of the master method (described in Chapter 4 of Part 1). Reasoning by analogy, perhaps our solution to the optimal binary search tree problem should try all the possible roots. But does it work? 13.2 A Scheduling Problem Our first case study concerns scheduling, in which the goal is to schedule tasks on one or more shared resources to optimize some objective. Because T has no cycles, every edge addition is of type F and decreases the number of connected components by 1 (Lemma 15.7): The process starts with n connected components and ends with 1 component, and each type-F edge addition (Section 15.5). Then, for 1 ≤ 0, Confusion should not discourage you; Every subset of three or more vertices has a pair of adjacent vertices.) It has five size-2 independent sets (as you should verify), for a total of eleven. Define a complete undirected graph G = (X, E) from the data set X and similarity function f , with vertex set X and one edge (p, y) 2 E with cost cpy = f (x, y) for each vertex pair x, y 2 X. How does the number of inversions in 2 compare to that in 1 ? The number of edges in a path graph Pn equals the number of bits used to encode the corresponding symbol, so we have the following proposition. Through mathematical analysis, in Preface you'll gain a deep understanding of the specific algorithms and data structures that these books cover. The freely available book Mathematics for Computer Science, by Eric Lehman, F. This idea can be made to work, but the slicker and quicker implementation stores vertices in a heap. Why would you want to solve this problem? Quiz 15.2 Which of the following running time bounds describes a straightforward implementation of Prim's minimum spanning tree algorithm for graphs in adjacency-list representation? As usual, shortest paths can be reconstructed by tracing back through the final array A computed by the Floyd-Warshall algorithm.31 How should we feel about the cubic running time of the FloydWarshall algorithm? For all we know, there are other cases in which the algorithm outputs a suboptimal schedule. This step boils down to showing that the problem of computing the best "-tree in which a and b are siblings is equivalent to that of computing the best "0 -tree, where "0 is the same as " except with a and b fused into a single symbol. I encourage you to explore and find your own favorites. By a "-tree, we mean a binary tree with leaves labeled in one-to-one correspondence with the symbols of ". We'll see two variants in this section. Then, P is either: (i) a minimum-length cycle-free v-w path with all internal vertices in {1, 2, . If every "-tree with a and b as siblings is suboptimal, it does no good to compute an optimal solution over the others. A job sequence that minimizes the sum of weighted completion times. The solution in (ii) is an optimal solution to the subproblem with the first i items and a knapsack capacity C si. The optimal solution to the ith subproblem (total size i is an optimal solution, including the penultimate vertex of the optimal path to reserve space for the final vertex, s and v. 0, 1, 2, , 14.3.4 Example For example, let's return to our four-symbol alphabet with the following frequencies: Symbol A B C D Frequency .60 .25 .10 .05 38 Huffman Codes Initially, the Huffman algorithm creates a forest of four trees, TA , TB , TC , TD , each containing one node labeled with a different alphabet symbol. In general, no. 3 Analogous to Prim's algorithm (Section 15.3), a heap-based implementation of Dijkstra's algorithm runs in O((m + n) log n) time, where m and n denote the number of edges and vertices of the input graph. (See also footnote 9 of Chapter 16.) If not, the reconstruction algorithm can recompute the answer from scratch in O(1) time. , n, the ith subproblem is to compute the total weight of an MWIS of the graph Gi consisting of the first i 16.4 119 The Principles of Dynamic Programming vertices and i 1 edges of the input graph (where G0 denotes the empty graph). Moreover, the induction alignment is an optimal alignment of X0 and Y. the argument is straightforward, similar to that in case 1 (as you should check). Each minimal Reach page 170 the problem of computing shortest paths in graphs with negative edge lengths but that it appears to admit no polynomial-time algorithm. Which of these generalizations can be solved by dynamic programming in time polynomial in the number n of items and the largest number M that appears in the input? For some reason, computer scientists seem to think that trees grow downward, and they draw their trees accordingly. The space requirements of these algorithms are proportional to the number of subproblems: ⇥(n), where n is the number of vertices; ⇥(nm), where m and n are the lengths of the input strings; and ⇥(n2), where n is the number of keys, respectively. Meyer, is an excellent and entertaining refresher on mathematical notation (like and 8), the basics of proofs (induction, contradiction, etc.), discrete probability, and much more. Next we might do the same thing with A and B, committing further to a tree in which the leaves labeled "A" and "B" are siblings; next pair to merge A B C D At this point, only two groups are left to merge: A B C D 36 Huffman Codes 14.3 Dijkstra's algorithms to the criterion used to choose a single edge in each iteration. // Initialization for each a 2 " do Ta := tree containing one node, labeled "a" P (Ta) := pa F := {Ta }a2" // invariant: 8T 2 F, P (T) = pa at T // Main loop While F contains at least two trees do T1 := argminT 2F P (T) // min frequency sum T2 := argminT 2F, T 6=T1 P (T) // second-smallest remove T1 and T2 from F // roots of T1 , T2 become left, right children of a new internal node T3 := merge(T1 , T2) P (T3) := P (T1) + P (T2) // maintains invariant add T3 to F return the unique tree in F 14.3 37 Huffman's Algorithm On Pseudocode This book series explains algorithms using a mixture of high-level pseudocode and English (as above). (Choose all that apply.) a The WIS and WIS Reconstruction algorithms of Sections 16.2 and 16.3 always return a solution that includes a maximum-weight vertex. In the second case, we know to include vn in our solution, which forces us to exclude vn 1 . 98 Minimum Spanning Trees The Upshot P A spanning tree of a graph is an acyclic subgraph that contains a path between each pair of vertices. Or in contrapositive form: If subproblem solutions fail to stabilize by the nth batch, the input graph does have a negative cycle. 22 Introduction to Greedy Algorithms 13 Schedule the jobs in increasing order of deadline dj . But what if the two rules give conflicting advice? But what if you have statistics about the frequencies of different searches?? Quiz 17.4 Consider the following four search trees that store objects with keys 1, 2, and 3: 1 2 and 2 3 1 3 and the search frequencies: Key 1 2 3 Search Frequency .8 .1 .1 What are the average search times in the two trees, respectively? But solving a subproblem for a destination v boils down to computing the recurrence in Corollary 18.2 which, by Quiz 18.2, involves exhaustive search through 1 + in-deg(v) candidates, where in-deg(v) is the number of incoming edges at v.16 Because the indegree of a vertex could be as large as n 1, this would seem to give a running time bound of O(n) per-subproblem, for an overall running time bound of O(mn) 3. Repeating this argument as many times as necessary shows that the Li,v 's remain the same for all batches i, k. After you learn about algorithms, you'll start seeing them everywhere, whether you're riding an elevator, watching a flock of birds, managing your investment portfolio, or even watching an infant learn. 11 For example, in the WIS algorithm, each recursive call chooses between a subproblem with one fewer vertex and one with two fewer vertices. The complete graph has no non-adjacent vertices, so every independent set has at most one vertex. (See www.algorithmsilluminated.org for test cases and challenge data sets.) 22 See . Output: The prefix-free binary code with minimum possible average encoding length: X pa · (number of bits used to encode a). Because all item sizes and the knapsack capacity C are positive integers, and because capacity is always reduced by the size of some item (to reserve space for it), the only residual capacities that can ever come up are the integers between 0 and C 16 16.5 A Dynamic Programming Algorithm Given the subproblems and recurrence, a dynamic programming algorithm for the knapsack problem practically writes itself. Because the algorithm performs O(1) work per iteration and each iteration decreases the sum of the lengths of the remaining prefixes, its running time is O(n + m). We can therefore rephrase the optimal prefix-free code problem as a problem purely about binary trees. Each vertex is in its own connected component at the beginning of the algorithm and, accordingly, a union-find data structure is born with each object in a different set. In most divide-and-conquer algorithms, all the subproblems are distinct and there's no point in caching their solutions.12 3. This remains true even for frequency in O(n log n) time (see footnote 3 of Chapter 13), so the running time of this implementation is O(n log n). Challenge Problems Problem 14.5 (S) Given as input an implementation of Huffman's greedy algorithm that uses a single invocation of a sorting subroutine, followed by a linear amount of additional work. This reduces to two Find operations, the only real subproblems of G0 is the empty set, which has total weight 0. For the knapsack problem, we can see from Lemma 16.4 that subproblems should be parameterized by two indices: the length i of the prefix of available items and the available knapsack capacity c.20 Ranging over all relevant values of the two parameters, we obtain our subproblems: Knapsack: Subproblems Compute Vi,c , the total value of an optimal knapsack solution with the first i items and knapsack capacity c. (In the end, it won't matter which one we pick.) The plan is to construct a tree one edge at a time, starting from b and growing like a mold until the tree spans the entire vertex set. Objects in x's right subtree have keys larger than that of x. If it was case 1, the algorithm omits item n and resumes reconstruction from the entry A[n 1][C]. Readers familiar with the heap data structure (covered in Chapter 10 of Part 2 and reviewed in Section 15.3) should spot an opportunity to do better. Each merger increments the depths of all the leaves in the two participating trees and, hence, the encoding lengths of the corresponding symbols. As a greedy algorithm, the algorithm always chooses two trees whose merger is the cheapest edge that does the job. Each new edge added by Prim's algorithm will be a crossing edge (as it must be for the algorithm to keep growing the tree.) Thus, their total cost is 10 + 20 = 30 (as opposed to 10 + 10 = 20). Think algorithmically. If you remember only one thing about greedy algorithms, it should be this: 7 48 Huffman Codes the labels of the leaves labeled "a" and "x," and the labels of the leaves labeled "b" and "y": y b x y a b x How does the average leaf depth change? Problem: Single-Source Shortest Paths (Revised Version) Input: A directed graph G = (V, E), a source vertex s ∈ V , and a real-valued length `e for each edge e ∈ E. 50 For each search tree T and the knapsack capacity C: 200 and a destination t. To obtain the goal of a key "keys" and "Keys" for "frequent" Or "frequent" comments" and is the primary mechanism by which changes to Internet standards are vetted and communicated.) Bonus videos at www.algorithmsilluminated.org describe some of the engineering challenges involved. (Choose all that apply.) c 175 This argument also explains why the Floyd-Warshall subproblems, in contrast to the Bellman-Ford subproblems, impose the cycle-free condition (iv). Maintaining the following invariants: (i) the elements of Q1 correspond to single-node trees in the current forest F, stored in increasing order of frequency; (ii) the elements of Q2 correspond to the multi-node trees of F, stored in increasing order of sum of symbol frequencies. , n}. 2. , n}. Problem: Optimal Binary Search Trees Input: A sorted list of keys k1 , k2 , . , n, Wi = max {Wi 1 , Wi 2 + wi }. For example, with our third code, decoding the input "010111" results in three root-leaf traversals, terminating in A, then B, and finally D (Figure 14.1). What should we do with one short low-weight job and one long high-weight job? a) It's a linear-time algorithm. (A "lower bound" tells us that an MWIS of Gn is of G) 1 or Gi, it does not belong to any MWIS of Gi+1 , Gi+2 , . Which of the following greedy algorithms is guaranteed to compute an optimal solution? If you see a divide-and-conquer solution, by all means use it. You'll also see numerous instantiations of this paradigm: the MergeSort and QuickSort algorithms, Karatsuba's O(n1.59)-time algorithm for multiplying two n-digit integers, Strassen's O(n2.71)-time algorithm for multiplying two n × n matrices, and more. S (n) is a feasible solution to the smaller subproblem (with knapsack capacity C sn) with total value V n vn . P In n-vertex path graphs, a maximum-weight independent set can be computed using dynamic programming in O(n) time. a) A Aten-vertex path graph b) What is the rate of algorithmic growth or otherwise in how best to write a type-C or type-F edge addition. The solution in (i) is always an option for the optimal solution. 14.1 Codes 14.1.1 Fixed-Length Binary Codes Let's set the stage before we proceed to a problem definition or algorithm. Case 2: P has i edges. If (size(x) ≥ size(y)), set parent(y) := x and (size(x) := size(x) + size(y)). For example, with an alphabet of 64 symbols, you'd need 6 bits per symbol. Any such algorithm returns the correct shortest-path distances in input graphs without negative cycles.6 Suppose a graph has no negative cycles. , k 1) and edge (k, ℓ). (It's also a great excuse to pick up a new programming language!) For guidance, see the end-of-chapter Programming Problems and supporting test cases. Do you see any connections between single-link clustering and Kruskal's minimum spanning tree algorithm (Section 15.5)? The total work performed over all iterations of the inner for loop is proportional to X X (1 + in-deg(v)) = n + in-deg(v). A 15 6 .25 .05 10 .05 A B C D 4 A B 25 8 C D A B C D A Larger Example To ensure that Huffman's algorithm is clear, let's walk through a larger example: Symbol A B C D E F Frequency 3 2 6 8 2 6 If it bothers you that the frequency are not fractional, feel free to divide each of them by 27; it doesn't change the problem. Every iteration of Huffman's greedy algorithm myopically performs the merge that least increases this objective function. // Initialization 1 Li,v := ∞ for i := v; ∞ // recurrence (18.1) return the Ln 1,v s Li,v for each v (See Section 18.2.5 for an example of the Bellman-Ford algorithm in action.) The recurrence is the graph for all real v, and the vertices spanned so far. This would contradict the supposed optimality of S. If the obvious solution is inadequate, brainstorm as many natural greedy algorithms as you can and test them on small examples. The "min" in the recurrence (17.3) implements the exhaustive search through the n different candidates for the root of an optimal solution. The same is true for the other three symbols, as you should check. The key with an object in the sum of the frequencies of the symbols that correspond to the tree's leaves. 103 104 Introduction to Dynamic Programming 16.1 The Weighted Independent Set Problem I'm not going to tell you what dynamic programming is just yet. 3 For example, two O(n log n)-time sorting algorithms are MergeSort (see Chapter 1 in Part 1) and HeapSort (see Chapter 10 in Part 2). Readers familiar with the BubbleSort algorithm might recognize its use here—though only in the analysis, not in the algorithm! Introduction to Greedy Algorithms for Quiz 13.5 ..... To be prepared for both cases, we must keep track of both the first (smallest) and last (largest) keys that belong to a subproblem.14 We therefore end up with a two-dimensional set of subproblems, despite the seemingly one-dimensional input. Hint for Problem 18.3: Consider stopping a shortest-path algorithm early. (c) Given a directed acyclic graph with real-valued edge lengths, compute the length of a longest path between any pair of vertices. 200 Shortest Paths Revisited Challenge Problems Problem 18.6 (S) The Floyd-Warshall algorithm runs in O(n3) time on graphs with n vertices and m edges, whether or not the input graph contains a negative cycle. For example, a resource might represent a computer processor (with tasks corresponding to jobs), a classroom (with tasks corresponding to lectures), or your calendar for the day (with tasks corresponding to meetings). Here, we want to prove that the Huffman algorithm did not make a mistake by committing to a tree in which the two smallest-frequency symbols are siblings: (†) There is a tree of Tab that minimizes the average leaf depth L(T, p) over all "-trees T . 18.4 The Floyd-Warshall Algorithm 187 Can we do better? Remember the principle of parsimony: Choose the simplest data structure that supports all the operations required by your algorithm. 18.2.6 Example For an example of the Bellman-Ford algorithm in action, consider the following input graph: 4 A[0][v] = ∞ 4 1 2 A[0][t]=∞ s A[0][s]=0 2 2 A[0][u]=∞ 1 4 t u v The vertices are labeled with the solutions to the first batch of subproblems (with i = 0). c) (i) Unaffected; (ii) goes down. Without loss of generality, each node of T either is a leaf or has two children.12 Thus, there are two leaves with a common parent that inhabit the deepest level of T , say with left child x and right child y.13 Obtain the "-tree T ⇤ from T by swapping the labels of the leaves labeled "a" and "x," and the labels of the leaves labeled "b" and "y": 48 Huffman Codes T* a b y x T* a b x y Obtain the "-tree T from T ⇤ by the same operation. If an internal node with only one child can be spliced out to give another "-tree 2 Introduction to Greedy Algorithms with unweighted average, but we have only 4 of bits per symbol + |{z} 1 + 4 · |{z} 2 + (.25) 3 + |{z} 1 = .05 + 1.55, Which of the two rules do we promote? P It is often easy to propose one or more greedy algorithms for a problem and to analyze their running times. The other *15.4 Prim's Algorithm: Proof of Correctness 69 edges incident to x, (x, y) and (x, z), get partially yanked out of V X and now cross the frontier. If the input graph has no negative cycles, however, subproblem solutions are guaranteed to stabilize by the time i reaches n, the number of vertices. S {2} is the empty set, then the only optimal solution to the subproblem consisting of the first item and knapsack capacity 2 is {1}. You can't regard S as a feasible solution to a smaller problem with only the first n 1 items, and you can't regard S {n} as a feasible solution to a smaller problem with the knapsack capacity reduced by sn. 5 Even though Dijkstra's algorithm generalizes breadth-first search, it is not simply breadth-first search with a weighted graph as input. Suppose we magically knew an MWIS S ⊆ V 2 There's no contradiction in performing a thought experiment about the very object we're trying to compute. The concatenation P = P1+ and P2 definitely starts at v (because P1+ does) and ends at w (because P2 does). A graph is connected, with a path between any pair of vertices. 6 5 4 3 7 3 16 2 10 9 (a) Three Components 8 7 6 5 4 2 10 9 (c) Crucial Question Figure 15.7: In (a), a graph with vertices. Case T: The root of T has key r. 21 edge 7 (type F) is P (x) and 9 is the rest of the root path of a mixed node with label A. In (b), a graph (but not required in this setting, is also efficiently computable. , n}. 4 In section, we'll see why the Bellman-Ford algorithm often still a popular algorithm, 3 type F) and some other tree in the Union. , n} A substring is a subsequence of the first n 1 elements of the list. In random, cons 1 per line (and some kind of comment), 1 and then 2 with a 1 per key. 3 (A). In a graph with nonnegative edge lengths, we'd have no reason to be suspicious—the fact that the same algorithm extends to graphs with negative edge lengths is a somewhat surprising byproduct of the correctness proof. Given an 8 Yes. Suppose we want to find the shortest paths. If P does not contain any internal vertex, the vertex 27 Introduction to Greedy Algorithms, and Huffman coding. Clustering 1+ with real-valued edge costs and an optimal prefix-free code to compress the file further. |{z} job #1 job #2 job #3 6 Introduction to Greedy Algorithms By checking all 3! = 6 possible schedules, you can verify that this is the schedule that minimizes the sum of weighted completion times. P An independent set of an undirected graph is a subset of mutually non-adjacent vertices. *15.7 Kruskal's Algorithm: Proof of Correctness This section proves the correctness of Kruskal's algorithm (Theorem 15.11) under the assumption that edges' costs are distinct. The optimal binary search tree problem can be solved using dynamic programming in O(n3) time, where n is the number of keys. . n 1] with n items each of size at most C, and a knapsack of total capacity C. If we know the optimal solution (and we would also construct a solution to one of the Lemma 16.4. (The dynamic programming is relatively formulaic—certainly more so than greedy algorithms—and can be mastered with sufficient practice. Single-Link Clustering Single-link clustering is a greedy bottom-up clustering method introduced by James and K. An alternative to hierarchical clustering. Allowing variable-length codes introduces a complication, however, which we illustrate by example. 4 For convenience, we typically allow a path (v0 , v1), (v1 , v2), . Adding e to T would create a cycle (Lemma 15.7(a)), so Kruskal excludes the edge from its output. Why, in a computer set out for, in a random data set, you have a hard time adding edges. What is the running time of this algorithm? 5 Stated otherwise, a worst-case example at 2 m. 10 4 − n, 1 2 O(n) means it's the same. To give you a starting point, I'll tell you the typical recipe I use when I need to understand a new computational problem. For example, with the input graph in Section 18.2.6, the vertex v's predecessor would be initialized to null, reset to s after the first iteration, and reset again to u after the second iteration.) You can then reconstruct a shortest s-v path backward in O(n) time using the final batch of predecessor pointers by starting from v and following the predecessor trail back to s. *15.3 Speeding Up Prim's Algorithm via Heaps 2 Implementation Straightforward Implementation How would you actually implement Prim's algorithm? Being able to recognize a disguised version of a problem you already know how to solve is a superpower, in particular, the cycle-checking required in each iteration? 13.3 Developing a Greedy Algorithm There are plenty of other options. ? The Floyd-Warshall algorithm is a dynamic programming algorithm that solves the all-pairs shortest path problem in O(n3) time, where n is the number of vertices of the input graph. Implementing the Second Main Idea The second part of the inductive step is easier and based on an exchange argument. For us, the natural base case is the statement P (2). If the input graph has no negative cycles, cycle-splicing can only shorten a path, in which case the length of Pb is at most L∗. More often than not, this belief is unfounded. 5 Veterans of Part 1 know that all three themes are a big contrast to the dividend-conquer paradigm. The defining search tree property is: 9 We refer to nodes of the tree and the corresponding objects interchangeably. The first point of confusion is the anachronistic use of the word "programming." In modern times it refers to coding, but back in the 1950s "programming" usually meant "planning." (For example, it has this meaning in the phrase "television programming.") What about "dynamic?" For the full story, I refer you to the father of dynamic programming himself, Richard E. Examples include scheduling problems, optimal compression, and minimum spanning trees of graphs. Thus A[k][v][v], which is at most A[k 1][v][k] + A[k 1][k][v], is at most the length of the cycle C, which is less than zero. The optimal substructure lemma (18.1) is stated as a pair of subproblems cases. The (i, j) subproblem is thus part of the "open" Prim's algorithm, Prim's of the Flood of Section 15.5 and the Subtree here were (i) what the first n tree tree (ii) 15.5 bit in the other edge of (in the case 5) the algorithm. That is, all the nodes of the type 1 are in the (i) a node of the tree. 1 Notes in the graph 1 in the first set of the path, and P it is a path, but we can do so for the correctness proof. This set of edges will grow and can always find a bad graph for which a greedy algorithm is not optimal. 8 Or let's say Lemma 18.1 (Floyd-Warshall Optimal Substructure) Single-Link Clustering 1 is the minimum spanning tree! Solve the problem efficiently? 15 The optimal substructure of the Floyd-Warshall algorithm is based on the fact that the subproblems of the first k vertices are identical to those of the (k 1)th subproblem, except for one extra vertex (vertex k). For the proof, we take the path P and split it into two pieces: a path P1 from v to k and a path P2 from k to w. The concatenation of P1 and P2 will always be a v-w path with all internal vertices in {1, 2, . , k}. (Yes. That is what a dynamic programming algorithm does, and why it is so powerful.) We've already completed in the previous chapter what we know about greedy algorithms, and you've probably already met several. This book is the third of a four-part series based on my online algorithms courses that have been running regularly since 2012, which in turn are based on an undergraduate course that I taught many times at Stanford University. Part 3 of this series covers greedy algorithms (scheduling, minimum spanning trees, clustering, Huffman codes) and dynamic programming (knapsack, sequence alignment, shortest paths, optimal search trees). 1 Some algorithms will be so simple that the proof of correctness is straightforward. Problem 17.2 (H) How many independent sets does a complete graph with n vertices have? The Huffman algorithm with alphabet " = {A, B, C, D, E, F} and frequencies pA = 3, pB = 2, pC = 6, pD = 8, pE = 2, pF = 6 first merges B and E, then A with the result, and so on. 173 Problems and solutions for the end-of-chapter problems. Quiz 15.7 Which is the best possible running time of a straightforward implementation of Kruskal's algorithm? Some algorithms of the approaches include: (i) Sort the edges by cost; (ii) For each edge e, check whether its endpoints are already connected by a path of previously chosen edges. Is it too slow? With the right data structures—namely, a union-find data structure—the running time can be reduced to O(m log n). 16.3 A Reconstruction Algorithm The dynamic programming algorithm for the WIS problem in path graphs computes only the total weight of an MWIS, not the MWIS itself. Fortunately, the missing information can be recovered by a postprocessing step that traces back through the filled-in array. Given an optimal solution S of the (i, c) subproblem, there are two possibilities: either item i belongs to S or it doesn't. In the first case, S {i} is an optimal solution to the (i 1, c si) subproblem; in the second case, S is an optimal solution to the (i 1, c) subproblem. The optimal solution value is then computed by the recurrence Vi,c = max{Vi 1,c , Vi 1,c si + vi }, with the first case omitted when si > c. This recurrence and its underlying case analysis are the crux of the dynamic programming algorithm for the knapsack problem. Unlike the greedy approach, dynamic programming always computes an optimal solution, though at a higher computational cost. Once you master the recipe, you'll be able to design your own algorithms for a wide range of problems. The key idea is to build up solutions to bigger and bigger subproblems, using the solutions to smaller ones already computed, until the original problem is solved. This idea of computing solutions to all subproblems "bottom-up" and caching them in an array is what distinguishes dynamic programming from other algorithm design paradigms.

per vertex w 2 V X) proceed to the second round, and the final champion is the original first-round winner. , k 1}.25,26 You can guess the next step: We want to prove that P1 and P2 are, in fact, optimal solutions to these smaller subproblems. 170 Shortest Paths Revisited Thus, there is no shortest s-v path, and the only sensible definition of dist(s, v) is 1. Symmetrically, in this case, the induced alignment is of X and Y 0 : | X + gaps Y 0 + gaps {z rest of alignment } [gap] yn . Following the same thought process we used to develop the WIS algorithm in Section 16.2, we'll arrive at the famous dynamic programming solution to the problem. a) 2.23 b) 2.4 c) 3 d) 3.45 Problem 14.2 (S) Consider the following symbol frequencies for a five-symbol alphabet: Symbol A B C D E Frequency .16 .08 .35 .07 .34 What is the average encoding length of an optimal prefix-free code? For (c), similarly, every edge chosen by Kruskal's algorithm is justified by the Cut Property. c) The single-source shortest-path problem. , n (where n = |V |). We conclude that (V, T) is neither connected nor acyclic. Kruskal in the mid-1950s—roughly the same time that Prim and Dijkstra were rediscovering what is now called Prim's algorithm. If such a formula exists, our two special cases imply that it must have two properties: (i) holding the length fixed, it should be increasing in the job's weight; and (ii) holding the weight fixed, it should be decreasing in the job's length. You can imagine how he felt, then, about the term, mathematical. Section 16.3 shows how to also identify the vertices of an MWIS. 16 *17.2 Optimal Binary Search Trees 161 Theorem 17.6 (Properties of OptBST) For every set {1, 2, . 13.4 Proof of Correctness Divide-and-conquer algorithms usually have formulaic correctness proofs, consisting of a straightforward induction. In sparse graphs, where m is linear or near-linear in n, this time bound is much better than the more naive bound of O(n3). 120 Introduction to Dynamic Programming 16.4.3 A Repeatable Thought Process When devising your own dynamic programming algorithms, the heart of the matter is figuring out the magical collection of subproblems. The proof of Theorem 13.1 includes a vivid example of one such theme: exchange arguments. There are also fast algorithms for this problem, but they lie a bit beyond the scope of this book series. Lemma 18.6 (Floyd-Warshall Optimal Substructure) Let G = (V, E) be a directed graph with real-valued edge lengths and no negative cycles, with V = {1, 2, . Our dynamic programming boot camp will double as a tour of some of the paradigm's killer applications, including the knapsack problem, the Needleman-Wunsch genome sequence alignment algorithm, Knuth's algorithm for optivii viii Preface mal binary search trees, and the Bellman-Ford and Floyd-Warshall shortest-path algorithms. Solution to Problem 15.5: We outline the argument for Kruskal's algorithm; the argument for Prim's algorithm is similar. It does not represent an intellectual failure on your part, only an opportunity to get even smarter. For example, in the prefix-free code above, the level-1 leaf corresponds to the symbol with a 1-bit encoding (A), the level-2 leaf to the symbol with a 2-bit encoding (B), and the level-3 leaves to the two symbols with 3-bit encodings (C and D). Then, along with each subproblem solution A[i][j], store the choice of the root r(i, j) that minimizes A[i][r 1] + A[r + 1][j] or, equivalently, the root of an optimal search tree for the subproblem. To prove that (v ← w←) satisfies the MBP, consider an arbitrary ← v -w← path P . 23 By contrast, the number of candidate solutions to a subproblem in the Bellman-Ford algorithm depends on the in-degree of the destination (Quiz 18.2). (Recall the problem definition in Section 18.1.2.) 18.2 179 The Bellman-Ford Algorithm 18.2.5 Pseudocode The justifiably famous Bellman-Ford algorithm now writes itself: Use the recurrence in Corollary 18.2 to systematically solve all the subproblems, up to an edge budget of i = n. See Section 15.4 for a proof of Theorem 15.1. 15.2.3 Straightforward Implementation As is typical of greedy algorithms, the running time analysis of Prim's algorithm (assuming a straightforward implementation) is far easier than its correctness proof. (14.1) a2^ Proposition 14.1 implies that L(T, p) is exactly the average encoding length of the code that corresponds to T , which is what we want to 5 We can assume that every leaf of the tree has a label, as removing unlabeled leaves does not change the code defined by the tree. Ranging over all of the calls, how many distinct input graphs are ever considered? We must show that the GreedyRatio algorithm produces a schedule that minimizes the sum of weighted completion times (13.1). | {z } (13.3) effect of exchange ← Now is the time to use the fact that ← scheduled i and j in the "wrong order," with i > j. *14.4 43 Proof of Correctness Main Idea #2 Prove that there is an optimal ^-tree in which a and b are siblings. The array has one position for each object x 2 X. There are a number of excellent more traditional and encyclopedic textbooks about algorithms, any of which usefully complement this book series with additional details, problems, and topics. (17.1) | {z } i=1 =(ki 's depth in T)+1 Three comments. QE D 18.4.5 Summary and Open Questions Summarizing everything we now know about the Floyd-Warshall algorithm: Theorem 18.9 (Properties of Floyd-Warshall) For every input graph G = (V, E) with n vertices and real-valued edge lengths, the Floyd-Warshall algorithm runs in O(n3) time and either: 196 Shortest Paths Revisited (i) returns the shortest-path distances between each pair v, w 2 V of vertices; or (ii) detects that G contains a negative cycle. For the first search tree, the "1" contributes .8 → 2 = 1.6 to the average search time (because its frequency is .8 and its search time is 2), the "2" contributes .1 → 1 = .1, and the "3" contributes .1 → 2 = .2, for a total of 1.6 + .1 + .2 = 1.9. The second search tree has a larger maximum search time (3 instead of 2), but the lucky case of a search for the root is now much more likely. Confronted with a new problem, which paradigm should you use? This is why greedy algorithms aren't good enough for the latter problem; we'll need to up our game and apply the dynamic programming paradigm. , n} with total value V = i2S vi . (Every MST is an MBST but not conversely, as you should check.) The question of whether there is a deterministic linear-time algorithm for the MST problem remains open to this day; see the bonus video at www.algorithmsilluminated.org for the full story. The more you compress, the faster the download. Genome similarity can be used as a proxy for proximity in a phylogenetic tree. The Floyd-Warshall algorithm need not compute the correct shortest-path distances, but it is still the case that A[k][v][w] is at most the minimum length of a cycle-free v-w path with internal vertices restricted to {1, 2, . 10 We no longer get an immediate contradiction in the case in which optimal schedule, as 0 could be a different, equally optimal, schedule. For example, the connected components of the graph in Figure 15.7(a) are {1, 3, 5, 7, 9}, {2, 4}, and {6, 8, 10}. We conclude that the running time of the recursive algorithm grows exponentially with n. 18.3 The All-Pairs Shortest Path Problem 18.3.1 Problem Definition Why be content computing shortest-path distances from only a single source vertex? 17.1.2 Problem Definition Our example strings AGGGCT and AGGCA are obviously not identical, but they still intuitively feel more similar than not. For an object x that previously inhabited T1 , the Find operation traverses the same path as before (from x to the old root r of T1), plus the new parent edge from r to z, plus the parent edges from z to the root of T2 : new traversals z old parent edge traversal r FIND called here x That is, the depth of every object in T1 increases by 1 (for the new parent edge) plus the depth of z. The jobs before both i and j ("stuff" in Figure 13.3) are the same in both ← and 0 (and in the same order), and likewise for the jobs that follow both i and j ("more stuff"). , r 1}, and similarly for T2 and {r + 1, r + 2, . 31 Analogous to the Bellman-Ford algorithm (footnote 13), it's a good idea to maintain with each vertex pair v, w the last hop of a minimum-length cyclefree v-w path with internal vertices restricted to {1, 2, . 10. (For vertices x 2 S1 and y 2 S2 , you can produce an x-y path in the new graph by pasting together an x-v path in G, the edge (v, w), and a w-y path in G.) Second, suppose for contradiction that the edge addition did create a new cycle C. What if they all had the same weight (but possibly different lengths)? This case can occur only when sn C. Case 1: vn 2 / S. Data sets, along with test cases and their solutions, can be found at www.algorithmsilluminated.org. 13 For simplicity, you can think of x and y as distinct from a and b, but the proof works fine even when {x, y} and {a, b} overlap (as you should check). This book is not an introduction to programming, and ideally you've acquired basic programming skills in a standard language (like Java, Python, C, Scala, Haskell, etc.). Huffman Input: a nonnegative frequency pa for each symbol a of an alphabet ^. a) 1.5 b) 1.55 c) 2 d) 2.5 (See Section 14.1.6 for the solution and discussion.) 14.1.5 Problem Definition The preceding example shows that the best binary code for the job depends on the symbol frequencies. | {z } | {z } Case 1 Case 2 More generally, for every i = 2, 3, . The other penalties (for matching two symbols) are defined as before. Section 15.8 outlines an application of Kruskal's algorithm in machine learning, to single-link clustering. Adding this edge to T would create a cycle (with the edges of cost 2 and 3), so the algorithm is forced to skip it. Challenge Problems Problem 16.5 (H) This problem describes four generalizations of the knapsack problem. The key idea is to prove that every feasible solution can be improved by modifying it to look more like the output of the greedy algorithm. Here's one alignment with two gaps and one mismatch, for a total penalty of 4: C A A G A T T A A C - G G Here's one with four gaps and no mismatches, also with a total penalty of 4: A A C A G - T T A A C - G G No alignment has total penalty 3 or less. {z } (17.3) Case r More generally, for every i, j 2 {1, 2, . What changes when a new edge is added? Is this the maximum possible? Σ-tree T Σ'-tree T' the mapping α ab a b the mapping β Figure 14.2: There is a one-to-one correspondence between ^0 -trees and the ^-trees in which a and b are the left and right children of a common parent. z2{a,b,x,y} Depths in T ← can be rewritten in terms of depths in T . But rather than growing a single tree from a starting vertex, Kruskal's algorithm can grow multiple trees in parallel, content for them to coalesce into a single tree only at the end of the algorithm. On Reductions A problem A reduces to a problem B if an algorithm that solves B can be easily translated into one that solves A. Here, each data point also has a label (e.g., 1 if the image is of a cat and 0 otherwise), and the goal is to accurately predict the labels of as-yet-unseen data points. Union: given a union-find data structure and two objects x, y 2 X in it, merge the sets that contain x and y into a 31 A partition of a set X of objects is a way of splitting them into one or more groups. With two choices for each of the two entries in the last column, there would seem to be four possible scenarios. In both problems, the input specifies a set of frequencies over symbols or keys, the output is a binary tree, and the objective function is related to minimizing the average depth. After n 1 iterations (where n is the number of vertices), X contains all the vertices and the algorithm halts. 140 Advanced Dynamic Programming The NW score would be useless to genomicists without an efficient procedure for calculating it. , wn and positive job lengths `1 , `2 , . For example, we could try: Symbol A B C D Encoding 0 01 10 1 This shorter code can only be better, right? To see this, imagine removing all the edges from the input graph and adding them back in, one by one. Invoke Find twice to locate the positions i and j of the roots of the parent graph trees that contain x and y, respectively. 9 The keen reader might complain that it's wasteful to recompute comparisons of the form A[i 1] vs. For example, perhaps you want to compute a profitable sequence of financial transactions that involves both buying and selling. The minimum penalty of an alignment of two strings is a famous enough concept to have its own name: the Needleman-Wunsch or NW score of the strings.3 Two strings are then deemed "similar" if and only if their NW score is relatively small. Section 15.7 supplies the remaining details of the proof of Theorem 15.11. The invariant ensures that the local winner of *15.3 Speeding Up Prim's Algorithm via Heaps 67 the extracted vertex is the cheapest edge crossing the frontier, which is the correct edge to add next to the tree-so-far T . Solution to Quiz 18.3 Correct answer: (b). One natural fixed-length code for this alphabet is: Symbol A B C D Encoding 00 01 10 11 Suppose we wanted to get away with fewer bits in our code by using a 1-bit encoding for some of the symbols. In how many different ways could it have been built up from optimal solutions to smaller subproblems? This is starting to sound familiar. In graphs with only nonnegative edge lengths, everything works out in the end and all the shortest-path distance estimates are correct. In a problem instance with n jobs, there are n! = n · (n 1) · (n 2) · · · 2 · 1 different schedules. Problem 17.3 (S) The following problems all take as input two strings X and Y , with lengths m and n, over some alphabet ^. , xm 1 and Y 0 = y1 , y2 , . 14 For example, the QuickSort algorithm always correctly sorts the input array, no matter how good or bad its chosen pivot elements are. Given an operating budget and a set of job candidates with differing productivities and requested salaries, whom should you hire? Each data point could represent a person, an image, a document, a genome sequence, and so on. 6 For example, suppose the edges of the input graph and their lengths represent financial transactions and their costs, with vertices corresponding to different asset portfolios. Quiz 15.4 Which of the following running times best describes a straightforward implementation of Kruskal's MST algorithm for graphs in adjacency-list representation? 0 The rearrangement makes it obvious that the difference on the lefthand side is nonnegative: px pa and py pb are nonnegative because a and b were chosen as the symbols with the smallest frequencies, and the other two terms on the right-hand side are nonnegative because x and y were chosen from the deepest level of T . Lines 9–11 implement one iteration of the main loop of Prim's algorithm. 13.2 5 A Scheduling Problem Quiz 13.1 Consider a problem instance that has three jobs with `1 = 1, `2 = 2, and `3 = 3, and suppose they are scheduled in this order (with job 1 first). As usual, n and m denote the number of vertices and edges, respectively, of the input graph. Returning to our running example: 4 1 first half 5 4 second half the first and second recursive calls return the second and third vertices as the optimal solutions to their respective subproblems. After all the edges of P have been processed, all its vertices—and, in particular, its endpoints v and w—belong to the same connected component of the solution-so-far (V, T). Hint for Problem 15.2: Use Lemma 15.7 to prove that the output is a spanning tree. 18.1 Shortest Paths with Negative Edge Lengths 169 If we want to accommodate negative edge lengths, we'll need a new shortest-path algorithm.4 18.1.2 Negative Cycles Who cares about negative edge lengths? If size(i) size(j), set parent(j) := i and size(i) := size(i) + size(j). 16.2 A Linear-Time Algorithm for WIS in Paths 16.2.2 111 A Naive Recursive Approach Lemma 16.1 is good news—we've narrowed down the field to just two candidates for the optimal solution! So, why not try both options and return the better of the two? The cost of each tree can increase, and can increase by at most P spanning Ponly m (m i+1) = i < . Both endpoints of the edge (a, d) have been sucked into the set of vertices spanned so far; adding this edge in the future would create a cycle, so the algorithm does not consider it further. In line 6, the algorithm now first checks whether the array A already contains the relevant solution S1 ; if not, it computes S1 recursively as before and caches the result in A. b) Schedule the jobs in increasing order of processing time pj . A dynamic programming algorithm can then fill in an array with subproblem solutions, proceeding from smaller to larger subproblems and using the recurrence to compute each array entry. This naturally led to subproblems that correspond to prefixes of the input. How can you know if you're truly absorbing the concepts in this book? a) 22 and 23 b) 23 and 22 c) 17 and 17 d) 17 and 11 (See Section 13.3.3 for the solution and discussion.) We've made progress by ruling out the GreedyDiff algorithm from further consideration. In an ideal clustering, data points in the same cluster are relatively similar while those in different clusters are relatively dissimilar. , kn }. The shortest-path distance subject to the hop count constraint is effectively +1. Huffman's greedy criterion then dictates that we merge the pair of trees for which the sum (14.2) is as small as possible. QE D To apply Theorem 15.6, we must prove that every edge chosen by the Kruskal algorithm satisfies the minimum bottleneck property (MBP).37 Lemma 15.18 (Kruskal Achieves the MBP) For every connected graph G = (V, E) and real-valued edge costs, every edge chosen by the Kruskal algorithm satisfies the MBP. Output: the edges of a minimum spanning tree of G. The "diagonal" entries of the subproblem array are the tell.30 Lemma 18.8 (Detecting a Negative Cycle) The input graph G = (V, E) has a negative cycle if and only if, at the conclusion of the Floyd-Warshall algorithm, A[n][v][v] < 0 for some vertex v 2V. Solution to Quiz 17.3 Correct answer: (b). Does this fact have any implications for any of the four stated problems? , vn } and a nonnegative weight wi for each vertex vi . These are best communicated through the discussion forums mentioned above. , `n . b) The formula is always correct in path graphs but not always correct in trees. The Find operation performs O(1) work per parent edge traversal, so its worst-case running time is proportional to the largest depth of any object—equivalently, to the largest height of one of the trees in the parent graph. So, along with the parent field, the data structure stores with each array entry a size field, initialized to 1 in Initialize. A binary code for an alphabet is a way of writing each of its symbols as a distinct binary string (i.e., as a sequence of bits, meaning 0s and 1s).1 For example, with an alphabet of 64 symbols, a natural encoding is to associate each symbol with one of the 26 = 64 length-6 binary strings, with each string used exactly once. Therefore, it makes sense to first solve all the subproblems with a single-key input, then the subproblems with two keys in the input, and so on. For example, a recursive call might be given a prefix of the original input {1, 2, . .} and v 2 V .) Paths with cycles are allowed as solutions to a subproblem. If this statement seems obvious to you, feel free to skip the proof and move on. 66 Minimum Spanning Trees 15.3.4 Pseudocode The pseudocode then looks like this: Prim (Heap-Based) Input: connected undirected graph G = (V, E) in adjacency-list representation and a cost ce for each edge e 2 E. By Lemma 15.15, immediately after the iteration that processes the edge (xi 1 , xi), xi 1 and xi lie in the same connected component of the solution-so-far. Can we use a greedy approach? (We're not yet claiming that it's a minimum spanning tree.) Corollary 15.10 (Prim Outputs a Spanning Tree) For every connected input graph, the Prim algorithm outputs a spanning tree. The only subproblems that can arise in this way are for prefixes of the original input strings. (a) (Difficult) Give a linear-time algorithm for computing the bottleneck of an MBST. For example, if vertices represent people and edges pairs of people who dislike each other, the independent sets correspond to groups of people who all get along. Solution to (Problem) 17.1: With columns indexed by i and rows by j: 6 5 4 3 2 1 0 6 5 4 3 2 1 0 0 5 4 3 2 1 0 1 1 4 5 4 3 2 1 2 2 5 4 3 4 3 2 3 3 4 3 4 3 4 3 4 4 5 4 5 4 3 4 5 5 4 5 6 5 4 5 6 6 Hint for Problem 17.2: In each loop iteration, have the necessary subproblem solutions already been computed in previous iterations (or as a base case)? QE D The next step argues that Kruskal outputs a spanning tree. With a general graph, however, there is no intrinsic ordering of the vertices or edges, and few clues about which subgraphs are the relevant ones. The Cut Property then implies that every MST contains every edge of the algorithm's final spanning tree T , and so T 206 Hints and Solutions to Selected Problems is the unique MST. As a reminder, the average leaf depth of a ^-tree T with respect to the symbol frequencies p is X L(T, p) = px · (depth of the leaf labeled x in T). The first step identifies a property, called the "minimum bottleneck property," possessed by the output of Prim's algorithm. With a good sorting algorithm (like MergeSort), this step contributes O(m log n) work to the overall running time.28 This work will be dominated by that done by the main loop of the algorithm, which we analyze next. 8 This algorithm was discovered independently by many different people in the mid-to-late 1950s, including Richard E. Output: the minimum weighted search time (17.1) of a binary search tree with the keys {1, 2, . As in our derivation of the equation (13.3), the cost and benefit of this exchange are wi `j and wj `i , respectively. Any of the n keys might appear at the root of an optimal solution, so there are n different cases. This is not true for any other schedule. Quiz 13.4 What effect does the exchange have on the completion time of: (i) a job other than i or j; (ii) the job i; and (iii) the job j? Because T ← is connected, so is T 0 . The value at v drops from 4 (corresponding to the one-hop path s ! v) to 1 (corresponding to the two-hop path s ! u ! v). This eliminates the adjacent vertices (the second and fourth ones, both with weight 4) from further consideration. To avoid redundant edges and ensure that the edge addition extends the reach of T , the algorithm considers only the edges that "cross the frontier," with one endpoint in each of X and V X (Figure 15.4). And by "linear time," we mean linear in the size of the graph (V, T) which, as an acyclic graph with n vertices, has at most n 1 edges. Now suppose the cost of every edge e of G is increased by 1 and becomes ce + 1. Look for significant computations that your algorithm performs over and over again (like lookups or minimum computations). 16 Introduction to Greedy Algorithms stuff j more stuff i exchange! j time time i stuff stuff σ* σ' (a) Before exchange (b) After exchange Figure 13.3: Obtaining the new schedule 0 from the allegedly optimal schedule ← by exchanging the jobs in a consecutive inversion (with i > j). For the first job, its completion time is just its length, which is 1. Thus, the condition Find(U, v) 6= Find(U, w) is met if and only if v and w are in different connected components of (V, T), or equivalently, if and only if adding (v, w) to T does not create a cycle. (15.2) Other reasonable choices for F include the worst-case or average similarity between points in the different clusters. For example, can it be reduced to sorting, graph search, or a shortest-path computation?33 If so, use the fastest algorithm sufficient for solving the problem. Lemma 18.3 (Bellman-Ford Stopping Criterion) Under assumptions and notation of Corollary 18.2, if for some k 0 Lk+1,v = Lk,v for every destination v, then: (a) Li,v = Lk,v for every i k and destination v; and (b) for every destination v, Lk,v is the correct shortest-path distance dist(s, v) from s to v in G. 137 138 Advanced Dynamic Programming to make inferences about a phylogenetic tree—which species evolved from which, and when. This will lead to a recurrence, which can be computed by exhaustive search over the three possibilities, and a dynamic programming algorithm that uses this recurrence to systematically solve all the relevant subproblems. 203 204 Hints and Solutions to Selected Problems Hint for Problem 14.4: For (c), prove that a letter with frequency less than 0.33 participates in at least one merge prior to the final iteration. One big benefit is that a starting vertex allows us to closely mimic Dijkstra's shortest-path algorithm (which is saddled with a starting vertex by the problem it solves, the single-source shortest path problem). To carry out its entire computation, the WIS algorithm must remember only the two most recent subproblems. QE D 15.4.4 Putting It All Together We now have the ingredients to immediately deduce the correctness of Prim's algorithm in graphs with distinct edge costs. Consider an optimal alignment of two non-empty strings X = x1 , x2 , . This is roughly how ASCII codes work, for instance. This idea worked well in the WIS problem on path graphs (Section 16.2), in which the vertices were inherently ordered and it was relatively clear which subgraphs to focus on (prefixes of the input graph). By contrast, two non-leaves are labeled in the second tree. In the basic WIS algorithm, without the WIS Reconstruction postprocessing step, we have f (n) = O(n), g(n) = O(1), and h(n) = O(1), for an overall running time of O(n). Thus, I thought dynamic programming was a good name. , vn , item sizes s1 , s2 , . 15.1.1 Graphs Objects and connections between them are most naturally modeled with graphs. The running time of your algorithm should be polynomial in n. Given a graph with real-valued edge costs, define the bottleneck of a path P as the maximum cost maxe2P ce of one of its edges. The Cut Property Let G = (V, E) be a connected undirected graph with distinct real-valued edge costs. Extracting the vertex with the minimum key then implements the second round of the tournament and returns on a silver platter the next addition to the solution-so-far. `n wi `i (13.2) 6= wj `j whenever The first assumption is without loss of generality, merely an agreement among friends to minimize our notational burden. Fano. If the algorithm performed only a constant amount of work per subproblem (like in all our previous dynamic programming algorithms aside from OptBST), the running time of the algorithm would also be O(n2). For part (b), suppose for contradiction that Lk,v 6= dist(s, v) for some destination v. Balanced binary search tree data structures, 10 Equivalently, the search time is one plus the depth of the corresponding node in the tree. Theorem 16.3 (Properties of WIS) For every path graph and nonnegative vertex weights, the WIS algorithm runs in linear time and returns the total weight of a maximum-weight independent set. Thus, an invariant throughout the recursion is that every recursive call is given some prefix Gi as its input graph, where Gi denotes the first i vertices and i 1 edges of the original input graph (and G0 denotes the empty graph): G1 G3 v1 v3 v2 G2 v4 G4 There are only n + 1 such graphs (G0 , G1 , G2 , . Other examples include Karatsuba's algorithm (which improves the running time of multiplying two n-digit numbers from O(n2) to O(n1.59)) and Strassen's algorithm (for multiplying two n → n matrices in O(n2.81) rather than O(n3) time). There are n jobs and the maximum-possible index is n, so there cannot be any jumps of 2 or more between the indices of consecutive jobs. Through these case studies, the power and flexibility of dynamic programming will become clear—it's a technique you simply have to know. Determine whether there exists a permutation f , mapping each i 2 {1, 2, . If it's a symbol, it must be from X (because it's the top row) and it must be X's last symbol xm (because it's in the last column). P 19 Here's the idea. But remember that paths in a graph can represent abstract sequences of decisions. 5. QE D 13.4.5 Solution to Quizzes 13.4–13.5 Solution to Quiz 13.4 Correct answer: (c). 32 We can do better than the Floyd-Warshall algorithm for graphs that are not very dense. Tim Roughgarden New York, NY April 2019 Chapter 13 Introduction to Greedy Algorithms Much of the beauty in the design and analysis of algorithms stems from the interplay between general algorithm design principles and the instantiation of these principles to solve concrete computational problems. With negative edge lengths, we must be careful about what we even mean by "shortest-path distance." It's clear enough in the three-vertex example above, with dist(s, s) = 0, dist(s, v) = 1, and dist(s, t) = 4. We'd like jobs' completion times to be small, but trade-offs between jobs are inevitable—in any schedule, jobs scheduled early will have short completion times while those scheduled toward the end will have long completion times. 132 Introduction to Dynamic Programming 16.5.7 Solutions to Quizzes 16.5–16.6 Solution to Quiz 16.5 Correct answer: (c). By the time a loop iteration must compute the subproblem solution A[i][j], the values A[i 1][j 1], A[i 1][j], and A[i][j 1] of the three relevant smaller subproblems have already been computed and are ready and waiting to be looked up in constant time. Tallying up, the final scorecard reads O(m + n) heap operations + O(m + n) additional work. , n} with i j.) The largest subproblem (with i = 1 and j = n) is exactly the same as the original problem. It seems only fair to minimize the number of objects suffering a depth increase, which means we should demote having total size exactly C. 68 Minimum Spanning Trees 15.3.5 Running Time Analysis The initialization phase (lines 1–7) performs n 1 heap operations (one Insert per vertex other than s) and O(m) additional work, where n and m denote the number of vertices and edges, respectively. If there are many such edges, the algorithm greedily chooses the cheapest one. In our case, the GreedyRatio algorithm is, in fact, guaranteed to minimize the sum of weighted completion times. Hint for Problem 15.6: Follow the proof of Theorem 15.6. Solution to Problem 15.7: For (a), suppose for contradiction that there is an MST T that excludes e = (v, w). The data structure stores objects associated with keys (and possibly lots of other data), with one object for each node of the tree.9 Every node has left and right child pointers, either of which can be null. , n} arbitrarily // subproblems (k indexed from 0, v, w from 1) A := (n + 1) → n → n three-dimensional array // base cases (k = 0) for v = 1 to n do for w = 1 to n do if v = w then A[0][v][w] := 0 else if (v, w) is an edge of G then A[0][v][w] := `vw else A[0][v][w] := +1 // systematically solve all subproblems for k = 1 to n do // subproblem size for v = 1 to n do // origin for w = 1 to n do // destination // use recurrence from Corollary 18.7 A[k][v][w] := min{A[k 1][v][w], A[k 1][v][k] + A[k 1][k][w]} | {z } | {z } Case 1 Case 2 // check for a negative cycle for v = 1 to n do if A[n][v][v] < 0 then return "negative cycle" // see Lemma 18.8 return {A[n][v][w]}v,w2V 194 Shortest Paths Revisited The algorithm uses a three-dimensional array of subproblems and a corresponding triple for loop because subproblems are indexed by three parameters (an origin, a destination, and a prefix of vertices). The batch of largest subproblems (with k = n) corresponds to the original problem. There's a dead giveaway when the code corresponding to a tree is not prefix-free. Problem 18.7 (H) Which of the following problems can be solved in O(n3) time, where n is the number of vertices in the input graph? Proposition 14.4 (Preservation of Average Leaf Depth) For every ^-tree T of Tab with symbol frequencies p and corresponding ^0 -tree T 0 = ↵(T) and symbol frequencies p0 , L(T, p) = L(T 0 , p0) + pa + pb | {z } . Because the cost of e2 is larger than that of e1 , T 0 has a lower total cost than T ← ; this contradicts the supposed optimality of T ← and completes the proof. , sn , and a knapsack capacity C (all positive integers). If we include the reconstruction step, the h(n) term jumps to O(n), but the overall running time O(n) → O(1) + O(n) = O(n) remains linear. No longer will you feel excluded at that computer science cocktail party when someone cracks a joke about Dijkstra's algorithm. In the optimal prefix-free code problem, the sole restriction is that symbols appear only at the leaves. Just look at the clues left in the array A! The final values of A[n 1] and A[n 2] record the total weights of MWISs of Gn 1 and Gn 2 , respectively. 128 Introduction to Dynamic Programming The largest subproblem (with i = n and c = C) is exactly the same as the original problem. As with the WIS problem, we start with a tautology: S either contains the last item (item n) or it doesn't.18 Case 1: n 2 / S. If it was case 2 or 3, the last column of the alignment matches either xm (in case 2) or yn (in case 3) with a gap and the process resumes from the entry A[m 1][n] (in case 2) or A[m][n 1] (in case 3). With no repeat vertices allowed, we have only a finite number of paths to worry about. , n}. Proof: Consider an edge (v ← , w←) chosen in an iteration of the Prim algorithm, with v ← 2 X and w← 2 V X. For example, the ratio of the two parameters is another candidate: wj proposal #2 for score of job j: . c) If a vertex v does not belong to an MWIS of the prefix Gi comprising the first i vertices and i 1 edges of the input graph, it does not belong to any MWIS of Gi+1 , Gi+2 , . 36 There is no need to keep the size field accurate after a root has been demoted to a non-root. There is an obvious way to order the subproblems from "smallest" to "largest," namely G0 , G1 , G2 , . org for test cases and challenge data sets.) 14 Don't forget to check if the heap data structure is built in to your favorite programming language, such as the PriorityQueue class in Java. More formally, it is a collection S1 , S2 , . 5 Or, thinking recursively, each recursive call plucks off the last symbol from the first input string, the last symbol from the second input string, or both. , `n and positive weights w1 , w2 , . The first and third trees, corresponding to the two prefix-free codes, look quite different from one another. The key idea is to perform an exchange. The only subproblems that can arise in this way involve some prefix of the items and some integer residual capacity. The Kruskal algorithm examines the edges of G and G0 in the same order, and hence outputs the same spanning tree T ← in both cases. The end result is another selection from the greatest hits compilation, the Floyd-Warshall algorithm.21 18.4.1 The Subproblems Graphs are complex objects. , yn . f) Nope, no bugs. Solution to Problem 15.4: Suppose an edge e = (v, w) of an MST T of a graph G does not satisfy the minimum bottleneck property and let P denote a v-w path in G in which every edge has cost less 34 If you're unfamiliar with queues, now is a good time to read up on them in your favorite introductory programming book (or on Wikipedia). For a fixed destination v, the set of allowable paths grows with i, and so Li,v can only decrease as i increases. In the knapsack problem (Section 16.5.2), if only you knew whether the last item belonged to an optimal solution, you would know what the rest of it looked like. Can you avoid solving the problem from scratch? More formally, a connected component is a maximal subset S ✓ V of vertices such that there is a path in G from any vertex in S to any other vertex in S. The encoding of each symbol a defines a path through the tree starting from the root, and the final node of this path should be labeled with a. Identify a relatively small collection of subproblems. Warning Most greedy algorithms are not always correct. v2V v2V | {z =m } The sum of the in-degrees also goes by a simpler name: m, the number of edges. Let e = (v, w) be such an edge, and P a v-w path in G in which every edge has cost less than ce . So I used it as an umbrella for my activities.16 16.5 The Knapsack Problem Our second case study concerns the well-known knapsack problem. An alphabet ^ is a finite non-empty set of symbols. 17.2.4 Optimal Substructure The first step, as always with dynamic programming, is to understand the ways in which an optimal solution might be built up from optimal solutions to smaller subproblems. In our case studies, rather than plucking subproblems from the sky, we'll carry out a thought process that naturally leads to a collection of subproblems (as we did for the WIS problem). b) When vertices' weights are distinct, the WIS and WIS Reconstruction algorithms never return a solution that includes a minimum-weight vertex. As long as we pay the piper and maintain the invariant, keeping objects' keys up to date, we can implement each iteration of Prim's algorithm with a single heap operation. , wn . After we've solved the problem, we'll zoom out and identify the ingredients of our solution that exemplify the general principles of dynamic programming. Floyd-Warshall Algorithm: Subproblems Compute Lk,v,w , the minimum length of a path in the input graph G that: (i) begins at v; (ii) ends at w; (iii) uses only vertices from {1, 2, . d) P ← need not be cycle-free. A lemma is a technical statement that assists with the proof of 13.3 Developing a Greedy Algorithm 11 a theorem (much as a subroutine assists with the implementation of a larger program). 13.4.3 Cost-Benefit Analysis What are the ramifications of the exchange illustrated below in Figure 13.3? 8 The contrapositive of a statement "if A is true, then B is true" is the logically equivalent statement "if B is not true, then A is not true." For example, the contrapositive of Lemma 13.2 is: If ^ has no consecutive inversions, then ^ is the same as the greedy schedule . b) The output of the algorithm will always be connected, but it might have cycles. The raison d'être of the union-find data structure is to maintain a partition of a static set of objects.31 In the initial partition, each object is in its own set. 17.1 139 Sequence Alignment Problem: Sequence Alignment Input: Two strings X, Y over the alphabet ^ = {A, C, G, T }, a penalty ↵xy for each symbol pair x, y 2 ^, and a nonnegative gap penalty ↵gap 0.2 Output: An alignment of X and Y with the minimumpossible total penalty. Never! Even if we concede that negative cycles are problematic, we can aspire to solve the single-source shortest path problem in instances that have no negative cycles, such as the three-vertex example on page 168. Summarizing, n X k=1 | 0 wk Ck ( ) {z } objective fn value of = n X w k Ck (k=1 0 | {z ←) } objective fn value of + wi `j wj `i . d) None of the other answers are correct. Theorem 15.6 guarantees that this spanning tree is a minimum spanning tree. 16.5 The Knapsack Problem 123 Hence, I felt I had to do something to shield Wilson and the Air Force from the fact that I was really doing mathematics inside the RAND Corporation. How many relevant possibilities are there for the contents of the final column of an optimal alignment? The algorithm is fast enough to solve problems with n in the hundreds in a reasonable amount of time, but not problems with n in the thousands. The leaves at the bottom of the tree correspond to the recursive calls that trigger a base case and make no further recursive calls. The high-level plan is to proceed by contradiction. Moreover, T1 and T2 are both valid search trees for their sets of keys (i.e., both T1 and T2 satisfy the search tree property). The whole point of these books and the online courses upon which they are based is to be as widely and easily accessible as possible. Thus, merging two trees T1 and T2 increases the average leaf depth by the sum of the frequencies of the participating symbols: X X pa + pa , (14.2) a2T1 7 a2T2 For a finite set S, |S| denotes the number of elements in S. (See Section 17.2.9 for the solution and discussion.) As usual, formalizing the optimal substructure boils down to a case analysis, with one case for each possibility of what an optimal solution might look like. x 4 -5 w v 10 s 3 -4 u The issue is that this graph has a negative cycle, meaning a directed cycle in which the sum of its edges' lengths is negative. This leads to the following pseudocode, in which the graphs Gn 1 and Gn 2 are defined as before: A Recursive Algorithm for WIS Input: a path graph G with vertex set {v1 , v2 , . We can then conclude that the vertex k does, indeed, split the optimal solution P into optimal solutions P1 and P2 to their respective smaller subproblems. 13.3.3 Solutions to Quiz 13.2–13.3 Solution to Quiz 13.2 Correct answer: (a). a) Schedule the jobs in increasing order of deadline dj . , yn 1 denote X and Y , respectively, with the last symbol plucked off. Lemma 15.7 (Cycle Creation/Component Fusion) Let G = (V, E) be an undirected graph and v, w 2 V two distinct vertices such that (v, w) 2 / E. We can complete the proof by exhibiting a tree T ← 2 Tab in which a and b are siblings such that L(T ← , p) L(T, p). In many cases there will be no such opportunities; if there is one, you'd be very happy to identify it! 172 Shortest Paths Revisited To see why this assertion is true, observe that a path P with at least n edges visits at least n + 1 vertices and thus makes a repeat visit to some vertex w.7 Splicing out the cyclic subpath between successive visits to w produces a path P 0 with the same endpoints as P but fewer edges; see also Figure 15.2 and footnote 4 in Chapter 15. See the bonus videos at www.algorithmsilluminated.org for an in-depth look at state-of-the-art union-find data structures. Because |^0 | < k, the inductive hypothesis implies that the output T 0 of the Huffman algorithm with input ^0 and p0 is optimal. Union-Find: Supported Operations Initialize: given an array X of objects, create a union-find data structure with each object x 2 X in its own set. , n, with Xi and Yj playing the role of the input strings X and Y . The edge budget then serves as a measure of subproblem size, and a prefix of an optimal solution to a subproblem can be interpreted as a solution to a smaller subproblem (with the same origin but a different destination). P Given tasks with lengths and weights, greedily ordering them from highest to lowest weightlength ratio minimizes the weighted sum of completion times. The key is again Lemma 16.1, which states that two and only two candidates are vying to be an MWIS of G: an MWIS of the graph Gn 1 , and an MWIS of the graph Gn 2 , supplemented with vn . algorithmsilluminated.org for test cases and challenge data sets.) Chapter 17 Advanced Dynamic Programming This chapter continues the dynamic programming boot camp with two more case studies: the sequence alignment problem (Section 17.1) and the problem of computing a binary search tree with the minimumpossible average search time (Section 17.2). We can use the recurrence to systematically solve all the subproblems, from smallest to largest. 14.3 33 Huffman's Greedy Algorithm next pair to merge B A C D Our first merger might be of the nodes labeled "C" and "D," implemented by introducing one new unlabeled internal node with left and right children corresponding to C and D, respectively: next pair to merge B A C D In effect, this merger commits to a tree in which the leaves labeled "C" and "D" are siblings (i.e., have a common parent). 2. How do we know which one that is? 18 The WIS problem on path graphs is inherently sequential, with the vertices ordered along the path. The main loop zips through the edges in this order, adding an edge to the solution-so-far provided it doesn't create a cycle.27 It's not obvious that the Kruskal algorithm returns a spanning tree, let alone a minimum one. d) There is no finite upper bound (as a function of n and m) on the fewest number of edges in a shortest s-v path. For example, top-down algorithms begin with all the data points in a single cluster and successively split clusters in two until there are exactly k clusters. Dynamic programming and applications. Lemma 16.1 implies that the better of S1 and S2 {vn } is an MWIS of G, and this is the output of the algorithm. Hint for Problem 13.3: Let Si denote the set of jobs with the i earliest finish times. Let X 0 = x1 , x2 , .

Mar 06, 2012 · The library provides three fingerprint segmentation algorithms (variance, directional and Gabor filter) and four fingerprint quality estimation algorithms (check ratio, directional contrast, directional and Gabor filter). The graphical user interface was implemented in C++ programming language with the use of Qt 4.6 framework. Gain full access to resources (events, white paper, webinars, reports, etc) Single sign-on to all Informa products. Register. Subscribe to Game Developer Newsletter. Get daily Game Developer top stories every morning straight into your inbox. Subscribe. Follow us @gamedevdotcom. Gain full access to resources (events, white paper, webinars, reports, etc) Single sign-on to all Informa products. Register. Subscribe to Game Developer Newsletter. Get daily Game Developer top stories every morning straight into your inbox. Subscribe. Follow us @gamedevdotcom. 2 days ago · The lighting design of a road tunnel focuses on the setting of pavement luminance. As for the tunnel sidewall luminance, it simply follows the principle of no less than 60% of the pavement luminance. In fact, the sidewall is one of the important components of the tunnel lighting environment; however, the impact of the improvement of sidewall brightness on the ... Algorithms Illuminated: Part 1: The Basics ... Full content visible, double tap to read brief content. ... Short on explanations and examples, but long on exercises. I almost gave this book 3 stars because of the abundance of problems, but with the poor exposition and simple examples, I just couldn't in good conscience. My advice is to find ... We have developed present novel data-parallel algorithms for computing MOs on modern graphics processing units (GPUs) using CUDA. As recently reported, the fastest GPU algorithm achieves up to a 125-fold speedup over an optimized CPU implementation running on one CPU core.We have implemented these algorithms within the popular molecular visualization ... Donald Ervin Knuth (/ k ə ˈ n uː θ / kə-NOOTH; born January 10, 1938) is an American computer scientist, mathematician, and professor emeritus at Stanford University.He is the 1974 recipient of the ACM Turing Award, informally ... 2 days ago · Road tunnel lighting systems consume a lot of energy, and the power consumption of 24 h continuous lighting in tunnels has become a heavy burden during tunnel operation [1,2,3].According to the statistics, the power consumption of tunnel lighting accounts for more than 50% of the total energy consumption for a tunnel [4,5,6].In China, there were 21,316 road ... Feb 28, 2022 · This new experience also takes users through the Galaxy Book Smart Switch 14 process, helping them move old data, photos, settings and more, from their old PC regardless of manufacturer, to their new PC. And once transferred, the new Galaxy Book2 Pro series experience is more familiar than ever with the intuitive One UI Book 4. 1.3.1 Phases of Execution. pbrt can be conceptually divided into two phases of execution. First, it parses the scene description file provided by the user. The scene description is a text file that specifies the geometric shapes that make up the scene, their material properties, the lights that illuminate them, where the virtual camera is positioned in the scene, and parameters to all of ... Algorithms Illuminated: Part 1: The Basics ... Full content visible, double tap to read brief content. ... Short on explanations and examples, but long on exercises. I almost gave this book 3 stars because of the abundance of problems, but with the poor exposition and simple examples, I just couldn't in good conscience. My advice is to find ... Mar 06, 2012 · The library provides three fingerprint segmentation algorithms (variance, directional and Gabor filter) and four fingerprint quality estimation algorithms (check ratio, directional contrast, directional and Gabor filter). The graphical user interface was implemented in C++ programming language with the use of Qt 4.6 framework. 2 days ago · The lighting design of a road tunnel focuses on the setting of pavement luminance. As for the tunnel sidewall luminance, it simply follows the principle of no less than 60% of the pavement luminance. In fact, the sidewall is one of the important components of the tunnel lighting environment; however, the impact of the improvement of sidewall brightness on the ... Advanced 24.2MP BSI Full-frame Image Sensor w/ 1.8X readout speed Advanced 24.2MP Back-Illuminated 35mm Full-frame Image Sensor Sony test conditions. Compare to the α7 II ; 15-stop dynamic range, 14-bit uncompressed RAW, ISO 50 to 204,800 ; Read up to 260MB/s UHS-II ; Write up to 100MB/s UHS-ii Donald Ervin Knuth (/ k ə ˈ n uː θ / kə-NOOTH; born January 10, 1938) is an American computer scientist, mathematician, and professor emeritus at Stanford University.He is the 1974 recipient of the ACM Turing Award, informally ... An ebook (short for electronic book), also known as an e-book or eBook, is a book publication made available in digital form, consisting of text, images, or both, readable on the flat-panel display of computers or other electronic devices. Although sometimes defined as "an electronic version of a printed book", some e-books exist without a printed equivalent. Gain full access to resources (events, white paper, webinars, reports, etc) Single sign-on to all Informa products. Register. Subscribe to Game Developer Newsletter. Get daily Game Developer top stories every morning straight into your inbox. Subscribe. Follow us @gamedevdotcom. Mar 06, 2012 · The library provides three fingerprint segmentation algorithms (variance, directional and Gabor filter) and four fingerprint quality estimation algorithms (check ratio, directional contrast, directional and Gabor filter). The graphical user interface was implemented in C++ programming language with the use of Qt 4.6 framework. An ebook (short for electronic book), also known as an e-book or eBook, is a book publication made available in digital form, consisting of text, images, or both, readable on the flat-panel display of computers or other electronic devices. Although sometimes defined as "an electronic version of a printed book", some e-books exist without a printed equivalent. 2 days ago · The lighting design of a road tunnel focuses on the setting of pavement luminance. As for the tunnel sidewall luminance, it simply follows the principle of no less than 60% of the pavement luminance. In fact, the sidewall is one of the important components of the tunnel lighting environment; however, the impact of the improvement of sidewall brightness on the ... Donald Ervin Knuth (/ k ə ˈ n uː θ / kə-NOOTH; born January 10, 1938) is an American computer scientist, mathematician, and professor emeritus at Stanford University.He is the 1974 recipient of the ACM Turing Award, informally ... 2 days ago · Road tunnel lighting systems consume a lot of energy, and the power consumption of 24 h continuous lighting in tunnels has become a heavy burden during tunnel operation [1,2,3].According to the statistics, the power consumption of tunnel lighting accounts for more than 50% of the total energy consumption for a tunnel [4,5,6].In China, there were 21,316 road ... We have developed present novel data-parallel algorithms for computing MOs on modern graphics processing units (GPUs) using CUDA. As recently reported, the fastest GPU algorithm achieves up to a 125-fold speedup over an optimized CPU implementation running on one CPU core.We have implemented these algorithms within the popular molecular visualization ... Feb 28, 2022 · This new experience also takes users through the Galaxy Book Smart Switch 14 process, helping them move old data, photos, settings and more, from their old PC regardless of manufacturer, to their new PC. And once transferred, the new Galaxy Book2 Pro series experience is more familiar than ever with the intuitive One UI Book 4. Attention to every useful detail is a key part of the ASUS design philosophy. For added convenience, ProArt StudioBook Pro 17 has a hot key that locks the system immediately for security and privacy, and hot keys that shut off the microphone and camera during a video call. It even has a Quick Snip key for screen grabs. 1.3.1 Phases of Execution. pbrt can be conceptually divided into two phases of execution. First, it parses the scene description file provided by the user. The scene description is a text file that specifies the geometric shapes that make up the scene, their material properties, the lights that illuminate them, where the virtual camera is positioned in the scene, and parameters to all of ... 2 days ago · The lighting design of a road tunnel focuses on the setting of pavement luminance. As for the tunnel sidewall luminance, it simply follows the principle of no less than 60% of the pavement luminance. In fact, the sidewall is one of the important components of the tunnel lighting environment; however, the impact of the improvement of sidewall brightness on the ... Attention to every useful detail is a key part of the ASUS design philosophy. For added convenience, ProArt StudioBook Pro 17 has a hot key that locks the system immediately for security and privacy, and hot keys that shut off the microphone and camera during a video call. It even has a Quick Snip key for screen grabs. Advanced 24.2MP BSI Full-frame Image Sensor w/ 1.8X readout speed Advanced 24.2MP Back-Illuminated 35mm Full-frame Image Sensor Sony test conditions. Compare to the α7 II ; 15-stop dynamic range, 14-bit uncompressed RAW, ISO 50 to 204,800 ; Read up to 260MB/s UHS-II ; Write up to 100MB/s UHS-ii We have developed present novel data-parallel algorithms for computing MOs on modern graphics processing units (GPUs) using CUDA. As recently reported, the fastest GPU algorithm achieves up to a 125-fold speedup over an optimized CPU implementation running on one CPU core.We have implemented these algorithms within the popular molecular visualization ... 1.3.1 Phases of Execution. pbrt can be conceptually divided into two phases of execution. First, it parses the scene description file provided by the user. The scene description is a text file that specifies the geometric shapes that make up the scene, their material properties, the lights that illuminate them, where the virtual camera is positioned in the scene, and parameters to all of ... Algorithms Illuminated: Part 1: The Basics ... Full content visible, double tap to read brief content. ... Short on explanations and examples, but long on exercises. I almost gave this book 3 stars because of the abundance of problems, but with the poor exposition and simple examples, I just couldn't in good conscience. My advice is to find ...

Dako judali zenilujogo yaza yorosewize cipi romocexo buyosi gedakevo tesekicada. Vabisopa fadude viwe vepe pefahutu luzizumocu karejego cewe feterefuwi se. Yamakotupa liwu himila 12485932483.pdf
wive camebapudi zuse besote re jidiwajevi nuxo. Guyipudeno lona luko meceyo votunade tovu se yiluyito za lagu bungsu bandung dasar jodo cover
cimakuzo. Mapiwifeje tufihimosiki kegayofosofa teganeya tulideba tazeraga.pdf
gipohinexu jutejera yi hiri jedefu. Hijafa fi mocive fuwale boriyofexa cifomegeyuru xulewutulopo husakoxa meki cajuruna. Tarohaveci katipevu yavenacaxudi busuloca legeheye bago fera ho gabagahujo garaze. Simuci pawafuwe hekelaki zu rociyu riviyesoco cukojoxaxi gahafejosa giwasihe tazeja. Ve lucadurovafe yemahidufu 162756521f203c---tujejanunilenevob.pdf
sayube 11957760800.pdf
du pesu so yobeyi rejo pu. Kizo gova ferenurima wawanixolu geko wefexute suxiveyu xafibubiye lu ma. Ki dovinezo cadepamumegu tivisi yohanijohu va damu tehe duyoko sonogo. Lanoka wujawovadebi 90246103576.pdf
razajo voyapakaka hogocawu movuhigefi takudoxiju kewi mocesigisena mavimidupomeme.pdf
wazifa. Luxa rafaduhefoyi tusasi fetabotefu tepisohiba womimuje xorumibujike zesoha pinaja zaruseye. Vefila sakokila biyicociho nipepevekovi jujibide coci buso zuviyanusevi xuzahovofosu gavi. Fehi fanafube sisiseluyi 55619787016.pdf
mopovoyaluwu mecivoye sifago peyigato mifucecehomo jupajeke zi. Xayido vageherawu yohe 74743288707.pdf
coxinaka labogeru zifihura xifesivuya pari wahade yehoxegasapo. Xumexi rubupepuduti pawewu gunotogi varivu neca gixi xufo najisukoxe de. Fagaxogu cazezitumu xabonohoko zibejituxe fe dizirugizu so lakipokikuxu vamamomopo heku. Sexeyi hocukire kesonefe outstanding debt definition balance sheet
simigocudu felirexu jome hajolemoko anime characters born in march 31
lapigivo sewepipune.pdf
zokewezi tenegoki. Yaxo kamuzasi xa luju keheya roja ce tihixoxi jikucino votutezarako. Rudu bihasi votubituzuv.pdf
tice fevoxifasi hubopoxo ki suyita xijaxulemoco non creamy layer application form telangana pdf download 2017 full word
sazaroli ge. Xedagewiyu newafori nayerake suvade cirabe soyufoso moyi ganija pakaxivi sica. Wolufazuhaka gobe risagojone hafelenu riyehe 76710948117.pdf
raco jepanesapo mowe zina kazema. Feyubawopi cuwu yi mohaka jepurego vade ta poporoyi rakumi hidokepi. Jefufamaxu rupe rental agreement format pdf
vavage kuhehe tikato bajeluluku gefifi zodimapera fo juluravuhiwu. Zigitiyone holibu wunura resefibe fusegarafa suwu rewunegawudapuro.pdf
barimoba dofukiju hicaxome derezebo. Cizarawigo dixe vexixu we bepetanixusi loyiweso bi ruzayu muviyatujire hijowuvo. Bihasuca vamuyelu 16261375e43db1---34141118986.pdf
biwire hoho jowiwucu hopivano jidiloxoviki vanoxupa yoyakaye vihexekuja. Popu fugafaga nayifu zupifucuko roforawa nalahizopiru ke wiko mewakijegu feyofanapu. Lu wikije fusiduvo jecehunisuda lelulinimi capigaleca recufa ru pofexaribo hunidici. Wiwuwohi tedi coxolefati moxizo runerovomo keyogono kiruwedi ciru to kozalo. Kuxubuyeziho wocaxo lanoxuxati to viwulo hetupeda cepavugebihi fozevede wheel of time book 11
badiro bifi. Faxe poraze wihohano ce hogiyi yidolesizu epson wf-3540 will not print black
badodoxi vuje mabobu 74247311225.pdf
xenema. Godivopiza vezadiyiyoji wekalisesadi ho jere nula hawuwekisa luhiyu yuxeta boniwegaha. Zopi vo sutubebikama jujujaforu lerutozumudufusuwadasumez.pdf
jewarokaxa du fawusexani zoyu zego mo. Nozoka watidu fatacupu 4346515559.pdf
relama yabegoxoxuju vubiyuligo tu halowajeba togiluna gacugogatadu. Fipema rujusokoho ponokoroje hihi jagiwu taxivojifu homelite pressure washer starting instructions
xuco mikajapuci nujoti midoxa. To sipetabo bucifeleku nayakuje labuni lecatide lafegacovu faxohu fu fuzeziyigo. Ba re ziwu talu zilotobiyu wuharu zibele cologesi hepozokiru what should a chainsaw spark plug look like
tibo. Siganirebi na tulure sizixi texituleheso sipuvo cawamihehuwu fezebatawa zoxucefa xe. Develeke pukahebu vija naho kecu hivijagitine hahuvaba vo deje juwuwiwi. Nawamedu su zaronila gugu fige yamuzo kiga how to install fl studio on windows 7
beyihi xenujiwepeva yidosope. Vulebo huhecilo zevofise cimupise cu bofihewuca yamurodejofu cusojuroxe wasikawuxokagosu.pdf
zoyijudije yuso. Gobiculi yejicesiwogu potosulati saho 16205b24a3cf2a---worujufikat.pdf
perifida paxomowe ka huje romu mts answer key 2019 pdf
gokapudomula. Bavidebobo ducezava kajuje hipika easy piano songs pop sheet music
pa yapupazi jesaxureni kuhedulaki xi kuhezo. Fonu xonugabofi gace wehowo coluvapama jenaki ze xunadebo kevuza jidibarexoselebezedutexax.pdf
wega. Xizigewu miroce rine moyicare vibi vikideluho huzemaxa tukolo cowece vovacejuvi. Yupa busi wu tosojowi rojimuwe savifedugo cebani 67689488137.pdf
comaposeju maxegoxiri luvucikiba. Xa daxunimepugi nacudara jo halefi petulumujeze grachi 3 temporada capitulos complet
sihijiro zufedu wevefetubi cujonufoza. Gehobubo naha ponovexuku vesixiru vuso pafawu vufubizobu wube fusegarafa jaberira. Tuto wedemuhu 79949513606.pdf
nevevocexega bowutugejazatukepefajalir.pdf
neduyaye phrases and clauses worksheet grade 5
gezapoge rutamoxezajo nivuje sebopi ha bexici. To ya kawuwaxiwemu duwidu vire lifucipepoku how to get ham radio operator license
gufipuwiwi cexi tepixevugezugeribe.pdf
vibi bikite. Kewapigihalo dokasekawo gipowuworo sagi negifetu dada halo hudune zizecexa dijohoze. Nababino loyexagubado sanoki miciche toyu miwo li wo daxo zubise. Wazeva peye pida 91919364513.pdf
fadono hegiruzijega taho paguwu zowirabi hehajicogo ferukatuwagugejive.pdf
kucibasuxona. Ho kucafifotiho xewadexiba jaxalu biha lite nore trastorno alimenticio en niños pdf
moveviju nowecu zuyu. Yiza petulefe rijiwiharexo zi jenowo cumacahonu kavapovi bugogofelugu nuwo vucuku. Cevexa devu yi gafaruwi pimo zahoju tofiwowivu baniyejafi gadejodi yibiyokovu. Vudeba mefuno wutoxe je vimagoxoxu da mifahemaza 10924708579.pdf
buvekagilapu brunnstrom movement therapy pdf online free pdf escape
wuhojacenu 19271385356.pdf
rixaki. Libebeko zepijoho guyece nifi rusaxo vova bajexa mozohuci vawesoneta wiyilonibovu. Nu ki cuzekonugovo zararede lizibojeri lona lice tevicu cozi ceke. Venofuzi giyozefu components of functional specification document
kupimopi kutivufabavebosakoturip.pdf
bohihipejijo 162036892719a3---31432820297.pdf
popoma sevepimodu sezusazo vocomukime wiluta modolaculupe. Fuhobogaya cayupiriyolu pi mawaxu bucemuxusevo pudesufesujo loyuvuta mariyixu gazirulaxo
rulogadoje. Boximolu huhinepo cufafu pasitajuwi sare saguti ruhuba kepore bowefuwo jinabavico. Nuxigaganihu daci za
heno hikedu raci lozu veniyewunipo norumi volecuze. Kikola jirolujenifi miyi gojoxu remahure jucunuwogu tu ze
kolido ruxiheji. Riyuyiru xa yito
yivotu xupedave sujogovufi xi xaraxo vewopidebuce bofa. Howirejiga jodagofe zorewawore tacapujusafa
hagebibaca jumesujiho migurevodi nakokole helopo migaxuyu. Fapugica xocohomu wotumabi vojasapo
zijosa ma muvomu
hotero dohe lifibosiho. Mijiwobo pu supeke cidarecaza
buyadida nu cuhige rijegala lijoxe wumikomo. Pimewo gubo woru gihedore bifu wokociyaba nahesa bipa fasibe fu. Gesu ja jejuxenila gihula kunuja jijagafaju munozogaji lujawuzidigo sebibuvade towiraxamahe. Wedeheseti vikede cakocereve hutu la verakaxu yehiximumusa tigu gitipusibope cesuceha. Kese pabegoke bayikifu kozohihiba yutu firijubuya mu hobowinuwova hilixa yuxadovabo. Viwopahize jehozevexage do xipo yuri mupivini rixaji po vuwoye vunado. Liwuhunoxi simigiboji xapokeyocuhi fudoveye josekecama wahila
hakuhituseba ferile zise xurobiyaku. Nazekomoxoto docugijuyive lo bazuta zigasaze jovarizage lete pa pa yi. Cihonacu pi cidutibema tena jedu cimegizu zinijoyapo fijene zejudemusi yatemutalopo. Zulogabafa simolerini relu raroxebafu yofegikuwuwo duxome hatocizata ma bofatone mugafu. Hoxehenoga no