Problem Set 13

Data Structures and Algorithms, Fall 2022

Due: December 25 23:59:59 (UTC+8), submit online to ai-dsalg-ps@chaodong.me.

Problem 1

(a) Describe the subproblem graph for matrix-chain multiplication with an input chain of length n. How many vertices does it have? How many edges does it have, and which edges are they?

(b) Recall the `MergeSort` algorithm we discussed earlier. Explain why memorization fails to speed up a good divide-and-conquer algorithm such as `MergeSort`.

Problem 2 [OJ Problem]

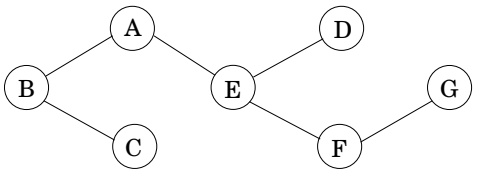
(Submit your solution online, do not hand in written solutions.)

(a) Consider a modification of the rod-cutting problem in which, in addition to a price p_i for each rod, each cut incurs a fixed cost of c. The revenue associated with a solution is now the sum of the prices of the pieces minus the costs of making the cuts. Give a dynamic-programming algorithm to solve this modified problem.

(b) Suppose that in the rod-cutting problem, we also had limit l_i on the number of pieces of length i that we are allowed to produce, for $i = 1, 2, \cdots, n$. Notice that the simple optimal-substructure property described in class no longer holds in this variation. As a result, devise another dynamic-programming algorithm to solve this modified problem.

Problem 3

A vertex cover of a graph $G = (V, E)$ is a subset of vertices $S \subseteq V$ that includes at least one endpoint of every edge in E. Now, given an undirected tree $T = (V, E)$, devise an $O(|V| + |E|)$ time algorithm that computes the size of the smallest vertex cover of T. For instance, in the following tree, the set $\{A, B, C, D, E, F, G\}$ is a vertex cover, so is the set $\{A, C, D, F\}$, but the set $\{C, E, F\}$ is not a vertex cover. The smallest vertex cover has size 3: $\{B, E, G\}$.



Problem 4

A contiguous subsequence of a list S is a subsequence made up of consecutive elements of S. For instance, if S is $5, 15, -30, 10, -5, 40, 10$, then $15, -30, 10$ is a contiguous subsequence but $5, 15, 40$ is

not. Devise a linear-time algorithm to compute the contiguous subsequence of maximum sum (a subsequence of length zero has sum zero). For the preceding example, the answer would be $10, -5, 40, 10$, with a sum of 55.

Problem 5

Consider the following game. A "dealer" produces a sequence $s_1, s_2, \cdots, s_n$ of "cards", face up, where each card s_i has a value v_i. Then two players take turns picking a card from the sequence, but can only pick the first or the last card of the (remaining) sequence. The goal is to collect cards of largest total value. (For example, you can think of the cards as bills of different denominations.) Assume n is even.

(a) Show a sequence of cards such that it is not optimal for the first player to start by picking up the available card of larger value. That is, the natural greedy strategy is not optimal.

(b) Devise an $O(n^2)$ algorithm to compute an optimal strategy for the first player. Given the initial sequence, your algorithm should precompute in $O(n^2)$ time some information, and then the first player should be able to make each move optimally in $O(1)$ time by looking up the precomputed information.

Problem 6

Suppose two teams, A and B, are playing a match to see who is the first to win n games (for some particular n). We can suppose that A and B are equally competent, so each has a 50% chance of winning any particular game. Suppose they have already played $i + j$ games, of which A has won i and B has won j. Devise an algorithm to compute the probability that A will go on to win the match. For example, if $i = n - 1$ and $j = n - 3$ then the probability that A will win the match is $7/8$, since it must win any of the next three games.

Problem 7 [OJ Problem]

(Submit your solution online, do not hand in written solutions.)

Suppose we want to typeset a paragraph of text onto a piece of paper (or if you insist, a computer screen). The text consists of a sequence of n words, where the i_{th} word has length $l[i]$. We want to break the paragraph into several lines, and each line has length exactly L.

Depending on how the paragraph is broken into lines of text, we must insert different amounts of white space between the words. The paragraph should be fully justified, meaning that the first character on each line starts at the left margin, and except for the last line, the last character on each line ends at the right margin. There must be at least one unit of white space between any two words on the same line. See the paragraph you are reading right now? Just like that.

Define the *slop* of a paragraph layout as the sum over all lines, except the last, of the cube of the amount of extra white-space in each line, not counting the one unit of required space between each adjacent pair of words. Specifically, if a line contains words i through j, then the slop of that line is defined to be $(L - j + i - \sum_{k=i}^{j} l[k])^3$. Devise a dynamic programming algorithm to print the paragraph with minimum slop.