
CS 473: Algorithms, Fall 2009

HBS 10

This HBS contains review problems for midterm 2. Try to solve 3 of the first 6 problems during the hbs. Note that the problems are ordered by topic, not difficulty.

Problem 1 ([4.13]). We have n jobs J_1, J_2, \dots, J_n which we need to schedule on a machine. Each job J_i has a processing time t_i and a weight w_i . A schedule for the machine is an ordering of the jobs. Given a schedule, let C_i denote the finishing time of job J_i . For example, if job J_j is the first job in the schedule, its finishing time C_j is equal to t_j ; if job J_j follows job J_i in the schedule, its finishing time C_j is equal to $C_i + t_j$. The weighted completion time of the schedule is $\sum_{i=1}^n w_i C_i$.

Give an efficient algorithm that finds a schedule with minimum weighted completion time.

Example. Suppose there are two jobs: the first takes time $t_1 = 1$ and has weight $w_1 = 10$, while the second job takes time $t_2 = 3$ and has weight $w_2 = 2$. Then scheduling job 1 first yields a weighted completion time of $10 \cdot 1 + 2 \cdot 4 = 18$, while scheduling job 2 first yields a weighted completion time of $10 \cdot 4 + 2 \cdot 3 = 46$. Therefore J_1, J_2 is the schedule with minimum weighted completion time.

Problem 2 ([4.28]). Let $G = (V, E)$ be an undirected, connected graph such that each edge of G is colored either red or blue. Let k be an integer. Give a polynomial-time algorithm that either returns a spanning tree T of G such that T has exactly k red edges or it correctly reports that no such tree exists.

Problem 3 ([6.20]). It's nearing the end of the semester and you're taking n courses, each with a final project that still has to be done. Each project will receive an integer grade from 1 to g , higher numbers being better grades.

You have only $H > n$ (assume H is a positive integer) hours in which to work on the n projects and you must decide how to best allocate your time. To help, you've come up with a set of non-decreasing functions $\{f_i : i = 1, 2, \dots, n\}$, where $f_i(h)$ denotes the grade you'll get on the i th project if you spend $h \leq H$ hours on it.

Given these functions $\{f_i\}$, design a polynomial (in n, g, H) time algorithm to compute how many hours to spend on each project (in integer values only) so that your average grade is maximized.

Problem 4 ([7.12]). Let $G = (V, E)$ be a directed network with source s and sink t . Each edge in G has unit capacity (that is, $c(e) = 1$, for each edge $e \in E$). Let k be an integer.

Your goal is to delete k edges so as to reduce the maximum $s - t$ flow in G by as much as possible. Give a polynomial-time algorithm that finds a set of edges $F \subseteq E$ so that $|F| = k$ and the maximum $s - t$ flow in $G' = (V, E - F)$ is as small as possible.

Problem 5. Let the number of papers submitted to a conference be n and the number of available reviewers be m . Each reviewer has a list of papers that she can review and each paper should be reviewed by three different reviewers. Also, each reviewer can review at most 5 papers. Design and analyze an algorithm to make the assignment or decide no feasible assignment exists.

Hint. Can you model this problem as a network flow problem?

Problem 6 ([7.15]). Suppose you live in an apartment with n people (including yourself), $\{p_1, p_2, \dots, p_n\}$, and you would like to divide the task of cooking dinner. Over the next n days, $\{d_1, d_2, \dots, d_n\}$, each of you is supposed to cook dinner for the group exactly once, so that someone cooks on each of the days.

Each resident has provided you with a schedule $S_i \subset \{d_1, \dots, d_n\}$ informing you of the days when they are available to cook. A *feasible dinner schedule* is an assignment of each person to a different day, so that each person cooks on exactly one day, someone cooks each day, and if p_i cooks on day d_j , then $d_j \in S_i$.

- (a) Describe a bipartite graph G so that G has a perfect matching if and only if there is a feasible dinner schedule.
- (b) Suppose your friend creates a dinner schedule, in which $n - 2$ people are assigned to different nights on which they are available. However, for the remaining two people, p_i, p_j , and days d_k, d_l , you find that he assigned both p_i, p_j to d_k and no one to d_l . You would like to fix his mistake without having to recompute everything from scratch. Show that it's possible, using this "almost correct" schedule, to compute in only $O(n^2)$ time whether there exists a feasible dinner schedule (and output it) or not.

You can skip the remaining problems during the hbs. You can use them (and the previous problems) to practice for the exam.

Problem 7 ([4.3]). You are consulting for a trucking company that ships packages between New York and Boston. The trucks have a fixed limit W on the maximum amount of weight they can carry. The packages arrive at the New York station one by one, and each package i has weight w_i . The packages need to be shipped to Boston in the order in which they arrive and at most one truck can be in the station at any time.

At the moment, the company is using the following greedy algorithm for packing. They pack the packages in the order they arrive, and whenever the next package does not fit, they send the truck on its way and they bring an empty truck into the station.

Prove that, for a given set of packages with specified weights, the greedy algorithm used by the company minimizes the number of trucks needed.

Problem 8 ([4.10]). Let $G = (V, E)$ be an undirected graph with costs $c_e \geq 0$ on the edges $e \in E$. Assume you are given an MST T of G . You are also given an edge $e \notin E$ with cost c .

- (a) Give an $O(|E|)$ algorithm to test if T remains an MST for $G + e$, where $G + e$ is the graph obtained from G by adding the edge e . Can you give an $O(|V|)$ algorithm?
- (b) Suppose that T is no longer an MST for $G + e$. Give an $O(|E|)$ algorithm that updates T to the new MST of $G + e$.

Problem 9 ([6.12]). Suppose we want to replicate a file over a collection of n servers S_1, S_2, \dots, S_n . Each server S_i has a *placement cost* c_i to place a copy of the file at S_i . If a user requests the file from S_i and no copy is present, then servers $S_{i+1}, S_{i+2}, S_{i+3}, \dots$ are searched in order until a copy is found, say at S_j (where $j > i$). This results in an *access cost* a_i , where

$$a_i = \begin{cases} j - i & \text{if } S_i \text{ does not contain a copy the file} \\ 0 & \text{if } S_i \text{ contains a copy of the file} \end{cases}$$

We require that S_n contains the file, so that all searches will terminate.

Given $I \subset \{1, \dots, n - 1\}$, a subset of servers to place the file on, we define the total cost

$$c(I) = c_n + \sum_{i \in I} c_i + \sum_{i=1}^n a_i$$

to be the sum of the placement costs for each server that contains the file (recall that S_n must contain the file), plus the sum of the access costs associated with all n servers. Design a polynomial-time algorithm to compute a subset I of servers to contain the file that minimizes total cost.

Problem 10 ([6.18]). Consider the sequence alignment problem over a four-letter alphabet $\{z_1, z_2, z_3, z_4\}$, with a given gap cost and given mismatch costs. Assume that each of these parameters is a positive integer.

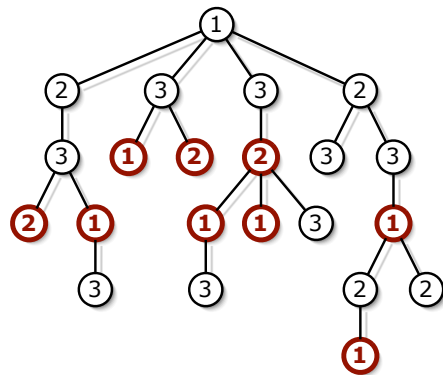
Suppose you are given two strings $A = a_1a_2\dots a_m$ and $B = b_1b_2\dots b_n$ and a proposed alignment between them. Give an $O(mn)$ algorithm to decide whether this alignment is the unique minimum-cost alignment between A and B .

Problem 11. You and your eight-year-old nephew Elmo decide to play a simple card game. At the beginning of the game, the cards are dealt face up in a long row. Each card is worth a different number of points. After all the cards are dealt, you and Elmo take turns removing either the leftmost or rightmost card from the row, until all the cards are gone. At each turn, you can decide which of the two cards to take. The winner of the game is the player that has collected the most points when the game ends.

Having never taken an algorithms class, Elmo follows the obvious greedy strategy: when it's his turn, Elmo always takes the card with the higher point value. Your task is to find a strategy that will beat Elmo whenever possible.

- (a) Prove that you should not also use the greedy strategy. That is, show that there is a game that you can win, but only if you do not follow the same greedy strategy as Elmo.
- (b) Give an algorithm to determine, given the initial sequence of cards, the maximum number of points that you can collect playing against Elmo.

Problem 12. Let T be a rooted tree. A valid labeling of T is a labeling that assigns to each node in T a label in $\{1, 2, 3\}$, such that every node has a different label than its parent. The *cost* of a valid labeling is the number of nodes that have smaller label than their parents.



A tree labeling with cost 9. Bold nodes have smaller labels than their parents. This is *not* the optimal labeling for this tree.

Given a rooted tree T , find the minimum cost of any valid labeling of T .

Problem 13 ([7.11]). Assume your friend implements an augmenting path based maximum flow algorithm that *only includes the forward edges* in the residual graph. That is, it searches for $s - t$ paths in a graph \tilde{G}_f consisting only of edges e for which $f(e) < c_e$, and it terminates when there is no augmenting path consisting entirely of such edges. Your friend has called this the Forward-Edge-Only Algorithm.

The Forward-Edge-Only Algorithm does not always return a flow of *maximum* value, but your friend claims that it always returns a flow whose value is greater than a fixed fraction of optimal. In particular, she claims:

There is an absolute constant $b > 1$ (independent of the input network), so that on every instance of the Maximum-Flow Problem, the Forward-Edge-Only Algorithm is guaranteed to find a flow of value at least $1/b$ times the maximum flow value (regardless of how it chooses its forward-edge augmenting paths).

Do you believe this? Either give a proof of this statement or its negation.

Problem 14 ([7.14]). In the *Escape Problem* you are given a directed graph $G = (V, E)$ and two disjoint collections of nodes $X \subset V$, $S \subset V$, denoted *populated*, *safe nodes* respectively. We would like to find a set of paths (called *evacuation routes*) from populated nodes to safe nodes, so that (i) each node in X is the tail of one path, (ii) the last node on each path lies in S , and (iii) the paths do not share any edges.

- (a) Design a polynomial-time algorithm that, given G, X, S , correctly decides whether a set of evacuation routes exists.
- (b) Suppose we impose the stronger condition that the paths composing our evacuation routes also do not share any *nodes* (note then, that they cannot share any *edges*). Design a polynomial-time algorithm to decide whether such a set of evacuation routes exists. Also, provide an example with the same G, X, S , in which the answer is yes to part (a), but no to (b).

Hint. Can you model this problem as a network flow problem?

Problem 15. A box i can be specified by the values of its sides, say (i_1, i_2, i_3) . We know all the side lengths are larger than 10 and smaller than 20 (i.e. $10 < i_1, i_2, i_3 < 20$). Geometrically, you know what it means for one box to nest in another: It is possible if you can rotate the smaller so that it fits inside the larger in each dimension. Of course, nesting is recursive, that is if i nests in j and j nests in k then i nests in k . After doing some nesting operations, we say a box is visible if it is not nested in any other one. Given a set of boxes (each specified by the lengths of their sides) the goal is to find a set of nesting operations to minimize the number of visible boxes. Give an efficient algorithm to do this.

Hint. Construct a bipartite graph $G = (L \cup R, E)$ such that, for each box i , G has a vertex $\ell_i \in L$ and $r_i \in R$. The graph G has an edge (ℓ_i, r_j) for each pair i, j such that box i nests in box j . What does a maximum matching M in G tell you?