CSOR W4231.001–Spring, 2021
Eleni Drinea

Homework 3 (145 points)

Out: Friday, February 19, 2021
Due: 11:59pm, Friday, March 12, 2021

Homework Instructions.

1. For all algorithms that you are asked to "give" or "design", you should
   - Describe your algorithm clearly in English.
   - Give pseudocode.
   - Argue correctness.
   - Give the best upper bound that you can for the running time.
2. **If you give a Dynamic Programming algorithm, the above requirements are modified as follows:**
   (a) Clearly define the subproblems in English.
   (b) Explain the recurrence in English. (This counts as a proof of correctness; feel free to give an inductive proof of correctness too for practice but points will not be deducted if you don't). Then give the recurrence in symbols.
   (c) State boundary conditions.
   (d) Analyze time.
   (e) Analyze space.
   (f) If you're filling in a matrix, explain the order to fill in subproblems in English.
   (g) Give pseudocode.
3. **You should not use any external resources for this homework.** Failure to follow this instruction will have a negative impact on your performance in the exam (and possibly in interviews). For the same reason, you should avoid collaborating with your classmates, at least not before you have thought through the problems for a while on your own. I also encourage you to work on all the recommended exercises.
4. You should submit this assignment as a **pdf** file on Gradescope. Other file formats will not be graded, and will automatically receive a score of 0.
5. I recommend you type your solutions using LaTeX. For every assignment, you will earn 5 extra credit points if you type your solutions using LaTeX or other software that prints equations and algorithms neatly. If you do not type your solutions, make sure that your hand-writing is very clear and that your scan is high quality.
6. You should write up the solutions **entirely on your own**. Collaboration is limited to discussion of ideas only. You should adhere to the department's academic honesty policy (see the course syllabus). Similarity between your solutions and solutions of your classmates or solutions posted online will result in receiving a 0 in this assignment, and possibly further disciplinary actions. There will be no exception to this policy and it may be applied retro-actively if we have reasons to re-evaluate this homework.

Homework Problems

1. (25 points) Give an algorithm that takes as input a tree on n nodes and determines whether it has a *perfect matching.*

   A *perfect matching* is a subset of the edges of the tree where every node appears exactly once; for example, if T is a tree with 4 nodes and edges (v_1, v_2), (v_2, v_3), (v_3, v_4), then $M = \{(v_1, v_2), (v_3, v_4)\}$ is a perfect matching.

2. (25 points) Let S, S' be two sequences of symbols from some alphabet with lengths n, m respectively.

   Give an efficient algorithm that decides whether S' is a *subsequence* of S. A *subsequence* of a sequence $S = s_1 s_2 \ldots s_n$ is any subset of the symbols in S taken in order, that is, it has the form $s_{i_1} s_{i_2} \ldots s_{i_k}$, where $1 \leq i_1 < i_2 < \ldots < i_k \leq n$.

   (For example, $S' = ABC$ is a subsequence of $S = ACACBBCB$.)

3. (25 points) In Internet routing, there are delays on lines but also, delays at routers. This motivates us to consider graphs that, on top of non-negative edge weights $w_e > 0$ for all edges $e \in E$, also have non-negative vertex costs $c_v > 0$ for all $v \in V$. We now define the weight of a path to be the sum of its edge weights, *plus* the costs of all vertices on the path (including the endpoints).

   Give an efficient algorithm that on input a directed graph with node costs and edge weights $G = (V, E, w, c)$ and an origin vertex $s \in V$ outputs an array *dist* such that for every vertex u, $\mathit{dist}[u]$ is the minimum weight of any path from s to u (e.g., $\mathit{dist}[s] = c_s$). Thus $\mathit{dist}[u]$ is the weight of the shortest s-u path under the modified definition for the weight of a path given above.

4. (30 points) You are given a line L that represents a long hallway in an art gallery and a sorted set $X = \{x_1, x_2, \ldots, x_n\}$ of real numbers that specify the positions of paintings in this hallway. Suppose that a guard can protect all the paintings within distance at most 1 of their position on both sides.

   Give an efficient algorithm that returns

   (a) the minimum number of guards to protect all the paintings; and

   (b) a placement of the guards that achieves this minimum number.

5. (30 points) There are n libraries $L_1, L_2, \ldots, L_n$. We want to store copies of a book in some libraries. Storing a copy at L_i incurs a purchase cost c_i (assume integer $c_i > 0$). A copy of the book is always stored at L_n. If a user requests the book from L_i and L_i does not have it, then $L_{i+1}, L_{i+2}, \ldots$ are searched sequentially until a copy of the book is found at L_j for some $j > i$. This results in a user delay of $j - i$. (Note that, in this case, no library L_k with an index k smaller than i is searched; also, if the user finds the book at L_i, then the user delay is 0.)

We define the **total cost** as the sum of the purchase costs and the *user delays associated with all n servers.* For example, if there are 4 libraries, and copies of the book are stored at L_1 and L_4, the total cost is $c_1 + c_4 + 2 + 1$.

Give an efficient algorithm that returns

(a) the minimum **total cost**; and

(b) a set of libraries where copies of the books must be placed to achieve the minimum total cost.

RECOMMENDED exercises: *do NOT return, they will not be graded.*

1. Give an efficient algorithm to compute the length of the **longest** increasing subsequence of a sequence of numbers $a_1, \ldots, a_n$. A subsequence is any subset of these numbers taken in order, of the form $a_{i_1}, a_{i_2}, \ldots, a_{i_k}$ where $1 \leq i_1 < i_2 < \ldots < i_k \leq n$ and an increasing subsequence is one in which the numbers are getting strictly larger.

   For example, the longest increasing subsequence of $5, 2, 8, 6, 3, 6, 7$ is $2, 3, 6, 7$ and its length is 4.

2. Consider an array A with n numbers, some of which (but not all) may be negative. We wish to find indices i and j such that

$$\sum_{k=i}^{j} A[k]$$

   is maximized. Give an efficient algorithm for this problem.

3. Problem 16.1 from your textbook (pp. 446-447).