CSOR W4246–Fall, 2020
Eleni Drinea

Homework 2 (140 points)

Out: Monday, November 16, 2020
Due: 11:59pm, Monday, November 30, 2020

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.
   - You're encouraged to analyze space but points will not be deducted unless the problem explicitly asks for a space analysis.
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 actual code.
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 when working on the first three problems. I also encourage you to work on all the recommended exercises.
4. You should submit your 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 and 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. You should list your collaborators on your write-up.

Homework Problems

1. (25 points) You are managing a team of engineers and need to assign tasks to them for n weeks. Every week i for $1 \leq i \leq n$, you can assign to the team one of two kinds of tasks: an *easy* task with revenue $\ell_i > 0$ or a *difficult* task with revenue $h_i > 0$. There is one constraint: if you pick a difficult task in week i, then you must assign no task to your team in week $i-1$. That is, your team must be idle during week $i-1$, thus earn revenue 0 in week $i-1$.

   Given sets of revenues $\ell_1, \ell_2, \ldots, \ell_n$ and $h_1, h_2, \ldots, h_n$, you want to assign tasks to your team during the n weeks so that the total revenue V of the tasks is **maximized**, subject to the constraint above. Give an efficient algorithm that computes the optimal revenue V. (It is ok for your team to start with a difficult task in week 1.)

2. (35 points) There is a number pyramid. The top floor has 1 positive integer. The second floor has 2 positive integers, the third floor 3, and so on and so forth.

   Here is an example of a pyramid with 4 floors:

   6
   3 8
   7 1 0
   2 6 4 4

   You start from the top floor. At each step, you can go either diagonally down to the left or diagonally down to the right. This way you can form a *route* from the top of the pyramid to its base. Note that there are multiple routes that end somewhere in the base of the pyramid. Your goal is to maximize the sum of the integers on any route.

   Design and analyze an efficient algorithm to compute the maximum sum on any *route* from the top to the base of the pyramid. You should also return an optimal route. You may assume that there are n floors in the pyramid.

   In the above example, the maximum sum is 22 achieved by route $6 \to 3 \to 7 \to 6$.

3. (30 points) You are given three strings X, Y, Z of lengths m, n, r respectively.

   Give an efficient algorithm to compute the length of the longest common subsequence of the three strings.

   A subsequence of string $s_1 s_2 \ldots s_n$ is a subset of the characters of the string taken in order, of 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, the longest common subsequence of "abc", "bdc" and "bacd" is "bc" and its length is 2.

4. (30 points) You're taking n courses, each with a final project that still has to be done. Each project will receive an integer grade on a scale of 1 to $g > 1$ (a higher number is a better grade). Your goal is to maximize your average grade on the n projects.

   You have a total of $H > n$ hours to work on the n projects, and you want to decide how to divide up this time. You have determined that, if you spend $h \leq H$ hours on project i, you'll get a grade of $f_i(h)$ (assume that H, h are positive integers and the functions f_i are non-decreasing: if $h < h'$ then $f_i(h) \leq f_i(h')$).

   Given n, H, g, $f_1, \ldots, f_n$, give a $O(nH^2)$ algorithm that computes the maximum average grade you can achieve by deciding how many hours you should spend on eac

5. (20 points) In the shortest-paths algorithm we are concerned with the *total length* of the path between a source (origin) node s and every other node. Suppose instead that we are concerned with the length of the *longest edge* in a path between the source node and every node. That is, the *bottleneck* of a path is defined to be the length of the longest edge in the path.

   Design an efficient algorithm to solve the single-source smallest bottleneck problem, i.e., find the paths from a source to every other node such that each path has the smallest possible bottleneck. (You may assume that the graph is undirected.)

EXTRA CREDIT exercises: *You do NOT have to solve this problem. If you do, you will receive extra credit for hw2t.*

1. (25 points) You are given a strongly connected directed graph $G = (V, E, w)$ with positive edge weights along with a particular node $x \in V$. Give an efficient algorithm for finding shortest paths between *all pairs of nodes*, with the restriction that these paths must all pass through x.

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

1. A server has n customers waiting to be served. The service time for customer i is t_i minutes. So if the customers are served in order of increasing i, the i-th customer spends $\sum_{j=1}^{i} t_j$ minutes waiting to be served.

   Given n, $\{t_1, t_2, \ldots, t_n\}$, design an efficient algorithm to compute the optimal order in which to process the customers so that the total waiting time below is minimized:

   $$T = \sum_{i=1}^{n} \text{(time spent waiting by customer } i\text{)}$$

2. 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.

3. 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.

4. Given two strings $x = x_1 x_2 \cdots x_m$ and $y = y_1 y_2 \cdots y_n$, we wish to find the length of their longest common substring, that is, the largest k for which there are indices i and j such that

   $$x_i x_{i+1} \cdots x_{i+k-1} = y_j y_{j+1} \cdots y_{j+k-1}.$$

   Give an efficient algorithm for this problem.