CSOR W4246–Fall, 2022
Eleni Drinea

Homework 2 Theoretical (120 points)

Out: Monday, October 3, 2022
Due: 11:59pm, Tuesday, October 18, 2022 (HARD DEADLINE)

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 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 midterm exam, and possibly in interviews. For the same reason, **you should avoid collaboration** with your classmates, to the extent possible. I encourage you to work on problems 1 and 2 immediately. You can work on problem 3 after Thursday's (Oct 6) lecture. Once we introduce Dynamic Programming in class (Oct 11), you can work on exercise 4. The deadline was slightly extended to ensure you will have time to work on the recommended exercises too. I strongly encourage you to work on all of them in the order they appear first entirely on your own, then in small study groups if needed.
4. You should submit your assignment as a **pdf** file to 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.

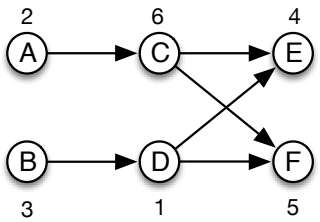
Homework Problems

1. (25 points) Given an undirected (unweighted) graph $G = (V, E)$ and two nodes $s, t \in V$, give an $O(n + m)$ algorithm that computes *the number* of shortest s-t paths in G.

2. (35 points) You are given a directed graph $G = (V, E)$ in which each node $u \in V$ has an associated price p_u which is a positive integer. Define the array `cost` as follows: for each $u \in V$,

   $$\texttt{cost}[u] = \text{price of the cheapest node reachable from } u \text{ (including } u \text{ itself)}$$

   For instance, in the graph below (with prices shown for each vertex), you can confirm by inspection that the `cost` values of the nodes A, B, C, D, E, F are $2, 1, 4, 1, 4, 5$ respectively.



   Your goal is to design an algorithm that fills in the entire array `cost`.

   (a) (20 points) Give a linear-time algorithm that works when the input is a Directed Acyclic Graph (DAG).

   (b) (15 points) Extend this to a linear-time algorithm that works for all directed graphs.
   Hint: Recall the "hierarchical" structure of a directed graph G, i.e., think of the meta-graph of its SCCs. Consider one SCC. What do you observe about the `cost` *values of the nodes of the SCC?*

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 for all $e \in E$, also have non-negative vertex costs c_v 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 $G = (V, E, w, c)$ and a source (origin) vertex $s \in V$ outputs an array $dist$ such that for every vertex u, $dist[u]$ is the minimum weight of any path from s to u (e.g., $dist[s] = c_s$). Thus $dist[u]$ is the weight of the shortest s-u path under the modified definition for the weight of a path given above.

4. (35 points) You are given a string of n characters $s[1, \ldots, n]$, which you believe to be a corrupted text document in which all punctuation has vanished (e.g., "itwasthebestoftimes"). You wish to reconstruct the document using a dictionary available in the form of a Boolean function `dict(·)`: for any string w,

$$\texttt{dict}(w) = \begin{cases} 1, & \text{if } w \text{ is a valid word} \\ 0, & \text{otherwise} \end{cases}$$

   Give a $O(n^2)$ dynamic programming algorithm that answers **yes** if the string $s[\cdot]$ can be reconstituted as a sequence of valid words, and **no** otherwise. You may map **yes** and **no** to 1 and 0, respectively.

   If the algorithm answers **yes**, you should also output the corresponding sequence of words. (Assume calls to `dict` take constant time.)

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

1. `https://leetcode.com/problems/climbing-stairs/`. Everyone is encouraged to work on this exercise before attempting problem 4!

2. You are going on a long trip. You start on the road at mile post 0. Along the way there are n hotels, at mile posts $a_1 < a_2 < \cdots < a_n$, where each a_i is measured from the starting point. The only places you are allowed to stop are at these hotels, but you can choose which of the hotels you stop at. Your destination is the final hotel (at distance a_n) and you must stop there.

   You'd like to travel 200 miles a day, but this may not be possible, depending on the spacing of the hotels. If you travel x miles during a day, the *penalty* for that day is $(200 - x)^2$. You

want to plan your trip so as to minimize the total penalty—that is, the sum, over all travel days, of the daily penalties. Give an efficient algorithm that determines the total penalty of an optimal sequence of hotels at which to stop, and returns such an optimal sequence.

3. 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:

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

4. 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.

5. 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.