CSOR W4231.001–Spring, 2021
Eleni Drinea

Homework 4 (155 points)

Out: Friday, March 12, 2021
Due: 11:59pm, Friday, March 26, 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. (30 points) Consider a directed graph constructed by the following discrete-time process. (Ideas along these lines have been used for building models to explain statistics of the Internet graph and growth of peer-to-peer networks.)

   You start with a single node v_1. At every time step t with $1 < t \leq n$, a single node v_t is added to the graph, together with a single outgoing edge; the destination of the edge is a node chosen uniformly at random among the existing nodes in the graph.

   Let G be the graph at the end of time step n.

   (a) What is the expected number of edges entering node v_j in G? Give an exact expression in terms of n and j, as well as an asymptotic expression using Θ notation.

   (b) What is the expected number of nodes with no incoming edges in G?

2. (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.

3. (30 points) You're asked to design a good strategy for a seller in an online auction. There are n buyers in this auction, each with a distinct bid $b_i > 0$. The buyers appear in a random order and the seller must decide immediately whether to accept the current buyer's bid or not. If he accepts, he makes a profit of b_i and the auction terminates. Otherwise, the bid is withdrawn and the seller may see the next buyer.

   Design a strategy (algorithm) that guarantees that the seller accepts the highest of the n bids with probability at least $1/4$, regardless of n, which you may assume even for simplicity.

4. (35 points) You are given some data to analyze. You have organized the process of analyzing the data so that it consists of n tasks that have to be performed sequentially by using dedicated hardware: you will use a processor P_i to perform task i, for every i.

   You can spend D dollars to perform the analysis, where D is a polynomial in n. Each processor is relatively cheap but may fail to complete its task with some probability, independently of the other processors. Specifically, P_i costs c_i dollars and succeeds to complete its task with probability s_i, while it fails with probability $1 - s_i$.

   (a) (3 points) What is the probability that the process of analyzing the data will be completed successfully?

   (b) Note that you can improve this success probability by using $p_i \geq 1$ identical processors P_i for task i instead of just one.

i. (4 points) What is the probability that task i will be completed successfully now?

ii. (4 points) What is the probability that the process of analyzing the data will be completed successfully now?

iii. (24 points) Your goal now is clear: on input $s_1, \dots, s_n$, integers $c_1, \dots, c_n$ and integer D, you want to assign values to $p_1, p_2, \dots, p_n$ so that the success probability of analyzing the data is maximized while you're not spending more than D dollars. Design and analyze the time and space complexity of an efficient algorithm that computes the maximum success probability of the entire process of analyzing the data while spending no more than D dollars.

5. (30 points) The following randomized algorithm returns the k-th smallest number from a set S of n distinct integers, for any k.

Algorithm 1

```
k-th order statistic $(S, k)$
1: Select an item $a_i \in S$ uniformly at random
2: for each item $a_j \in S$ do
3:     Put $a_j$ in $S^-$ if $a_j < a_i$
4:     Put $a_j$ in $S^+$ if $a_j > a_i$
5: end for
6: if $|S^-| = k - 1$ then          // $a_i$ is the $k$-th smallest item
7:     return $a_i$
8: else if $|S^-| \geq k$ then          // the $k$-th smallest item lies in $S^-$
9:     k-th order statistic $(S^-, k)$
10: else                              // the $k$-th smallest item lies in $S^+$
11:     k-th order statistic $(S^+, k - 1 - |S^-|)$
12: end if
```

(a) (5 points) Show how to compute the median of S using this algorithm.

(b) (25) Analyze the expected running time of `k-th order statistic` (S, k).

Hint: We will say that a subproblem is of type j if its input consists of at most $n\left(\frac{3}{4}\right)^j$ items but more than $n\left(\frac{3}{4}\right)^{j+1}$ items.

Upper bound the expected time of `k-th order statistic` *on a subproblem of type j, excluding the time spent on recursive calls. Equivalently, upper bound the expected time until an item a_i is selected such that $1/4$ of the input can be thrown out (thus the input shrinks by a factor of $3/4$).*

Next obtain an upper bound to the expected running time of the entire algorithm by summing up the expected time spent on every subproblem.

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. 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. 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.