CSE 100: Algorithm Design and Analysis
Midterm 3

Spring 2020

- This is a take-home exam with no proctoring. You can start at any time once you get access to this exam. You have to submit your by **4:30pm, Sat, 4/11**, through catcourses (midterm 3 under Assignments). You can resubmit again until the deadline. If there are any technical issues with uploading, it is extremely important to immediately contact the instructor or the TAs, and email solutions to them
- This is an open-book exam. The **only** resources you can use during the exam are all **course materials** uploaded to catcourses plus the **textbook**. In particular, this means that you are NOT allowed to search for solutions on the internet. No calculator is allowed. You have to take the exam by yourself.
- There are 6 problems in total. Every problem is of at least intermediate difficulty, meaning that my expectation from you won't be high for this exam. That said, you could earn some partial points if you show progress even if you can't solve problems completely.
- Please make your answers concise as well as precise. If there is something in the question that you believe is open to interpretation, then please go ahead and interpret, but state your assumptions in your answer.
- If you have questions, you can email the instructor and the TAs. However, we may not answer your questions on time as we can't be available all the time. In particular, do not expect reply on Saturday.

You're required to take the following honor pledge prior to taking the exam.

> *By completing this exam, I acknowledge and confirm that I will not give or receive any unauthorized assistance on this examination. I will conductmyself within the guidelines of the university academic integrity guidelines.*

1. (10 points) Dr. Sponge claims that in the tree representation of an optimal prefix-free code, every leaf node must have a sibling. Is the claim true or not? **Answer Yes or No, and explain why.**

   Sol. Yes. For the sake of contradiction suppose there is a leaf node x with no sibling in the tree. Let p be x's parent node and q be p's parent node. Then, by removing p and making x a child of q, we can reduce the depth pf x, meaning that we use less bits for encoding the character associated with the leaf node x. This contradicts the optimality of the tree.

   Note. 'No' can be a correct answer if there is only one character.

   We will not give partial points just for yes or no. Your explanation is what we're looking for.

2. (10 points) Suppose we want to compress a text consisting of 6 characters, a, b, c, d, e, f using the Huffman Algorithm. Give an example for which the algorithm produces at least one codeword of length 5. In other words, you are being asked to give a set of the character frequencies that results in the deepest tree.

   Sol. There are millions of examples. One simple example is 1, 1, 2, 4, 8, 16. In fact, assuming that $f_1, f_2, ...f_6$ are the frequencies sorted in increasing order, it can be a correct solution as long as $f_3 \geq f_2 + f_1$, $f_4 \geq f_3 + f_2 + f_1$, $f_5 \geq f_4 + ... + f_1$, $f_6 \geq f_5 + ... + f_1$.

3. (10 points) In the rod cutting problem, we are given as input, $p_1, p_2, ..., p_n$, where p_i denotes the price of a rod/piece of length i. We are interested in cutting a given rod of length n into pieces of integer lengths in order to maximize the revenue; here we are only interested in finding the maximum revenue. Cutting is free. Let r_i denote the maximum revenue one can get out of a rod of length i. To solve the problem using DP (dynamic programming), we can use the following recursion:

$$r_j = \begin{cases} \max_{1 \leq i \leq j}(p_i + r_{j-i}) & \text{if } j \geq 1 \\ 0 & \text{otherwise } (j = 0) \end{cases}$$

   But Dr. Sponge thinks that perhaps he has a more efficient recursion and proposes the following:

$$r_j = \begin{cases} \max\{p_j, \max_{1 \leq i \leq \lfloor j/2 \rfloor}(r_i + r_{j-i})\} & \text{if } j \geq 2 \\ p_1 & \text{otherwise } (j = 1) \end{cases}$$

   Does this proposed recursion correctly compute r_j? Answer Yes or No, and explain why.

   Sol. Yes. It is correct. When given a rod of length j, clearly we have an option to sell it with no cut for p_j. Suppose we cut. Then, we will get two pieces and let i be the length of the shorter one, which automatically makes the other piece of length $j - i$. Clearly, $i \leq \lfloor j/2 \rfloor$. The max revenues we can get out of the two pieces are r_i and r_{j-i}, respectively, thus giving us a revenue of $r_i + r_{j-i}$. The recursion is nothing but taking the maximum over all these possible choices.

4. (20 points) In the weighted version of Interval Selection problem, we are given n intervals, $I_1 = (s_1, f_1), I_2 = (s_2, f_2), ..., I_n = (s_n, f_n)$ where they are ordered in increasing order of their finish times, i.e., $f_1 \leq f_2 \leq ... \leq f_n$. Further each interval I_i is associated with a certain weight/profit $w_i \geq 0$. Our goal is to choose a subset of intervals with the maximum total

weight. If we let $M(i)$ denote the maximum weight of any subset of mutually disjoint intervals from $I_0, I_1, I_2, ..., I_i$ (here, I_0 is a 'dummy' interval of zero weight that is disjoint from all other intervals), we have the following recurrence:

$$M(i) = \begin{cases} 0 & \text{if } i = 0 \\ \max\{M(j) + w_i, M(i-1)\} & \text{otherwise,} \end{cases} \quad (1)$$

where I_j is the interval with the largest finish time that ends before I_i starts. Give a **pseudocode of a top-down DP algorithm based on memoization** using this recursion.

Sol.

$F(n)$

Let $M[0] = 0$, $M[1], M[2], ...M[n] = -\infty$.
Return $AF(n)$.

$AF(i)$

If $M[i] \geq 0$ return $M[i]$
Let j be $\max\{j' \mid 0 \leq j' < i \text{ and } e_{j'} \leq s_i\}$
(this can be computed a simple for loop)
Let $M[i] = \max\{AF(j) + w_i, AF(i-1)\}$.
return $M[i]$.

Note: The running time of the pseudocode is $O(n^2)$. You can find the above j in $O(\log n)$ time using binary search. Then, the RT can be improved to $O(n \log n)$ time. You will get the full 20 points even without using binary search. But if you used binary search, you will earn 3 extra points. (you should actually show the pseudocode of a binary search)

5. (20 points) Recall that in the matrix chain multiplication, the input is a sequence of n matrices, $A_1, A_2, ..., A_n$ where A_i is $p_{i-1} \times p_i$, and the goal is to fully parenthesize the product $A_1 A_2 \cdots A_n$ such that the number of multiplications is minimized. To solve the problem, using a DP, we computed $m[i, j]$ and $s[i, j]$ — recall that $m[i, j]$ was the minimum number of multiplications needed to compute the product $A_i A_{i+1} \cdots A_j$, and $s[i, j] = k$ implies that there is an optimal solution for $A_i A_{i+1} \cdots A_j$ that is constructed by $A_{i...k} \times A_{k+1...j}$.[1]

Unfortunately, we have **lost** all $m[i, j]$ values, but luckily we still have $s[i, j]$ values. Our goal is to compute $m[1, n]$ as fast as possible. **Give a pseudocode of your algorithm. What is your algorithm's asymptotic running time?** Your algorithm must run faster than you compute $m[1, n]$ from scratches using DP.

Sol. For simplicity, let's assume that we pretend that the array s can be globally accessed.

ComputeM(i, j)
If $i == j$ then return 0
Else return ComputeM$(i, s[i, j])$ + ComputeM$(s[i, j] + 1, j)$ + $p_{i-1} p_{s[i,j]} p_j$

Then, we call ComputeM$(1, n)$.

The running time is $O(n)$.

Pseudocode: 15 pts. RT: 5 pts.

[1] Here, $A_{i...k}$ denotes $A_i A_{i+1} ... A_k$, and $A_{k+1...j}$ is also analogously defined.

Note: Although you don't need to explain, the reason is as follows. Considering the recursion tree. There is no overlap whatsoever, and one function call splits each subproblem into smaller ones. Since the atomic (bottom) case involves one matrix and there are at most $O(n)$ function calls. Each function call takes $O(1)$ time (without considering its recursive calls). Therefore, the RT is $O(n)$.

6. (30 points) A Variant of Rod cutting. Recall in the rod cutting problem, we're given a rod of length n along with an array $\{p_i\}_{1\le i\le n}$, in which p_i denotes the price you can charge for a rod/piece of length i. The goal is to cut the given rod of length n into smaller pieces (or do nothing) so that the total price of the pieces is maximized. But things have changed. Cutting has become so expensive and difficult. You have to hire the cut master to cut the rod. The master is lazy and doesn't want to cut many times, so he charged k^2 dollars if he makes k cuts. Now your goal is to figure out the best way to cut the rod so as to maximize your profit, namely your revenue minus the total expenses for cutting.

   Example: Suppose $n = 10$ and you cut the rod into pieces of lengths, 2, 3, 5. Then your profit is $p_2 + p_3 + p_5 - 2^2$. If you cut it into 10 pieces, all of length 1, then your profit is $10p_1 - 9^2$.

   Give a DP based algorithm. Your algorithm only needs to compute the maximum profit. Analyze the running time of your algorithm.

   Sol. There are several ways. One possible solution is the following. Let $r_{i,k}$ denote the max profit we can get out of a rod of length i by making exactly k cuts. Here, $0 \le i \le n$ and $0 \le k \le n-1$. Clearly, $r_{i,0} = p_i$ for all $i \ge 0$; let $p_0 = 0$ for convenience. Also, $r_{0,k} = 0$ for all k. Further, $r_{i,k} = \max_{1\le j\le i-1}(p_j + r_{j-i,k-1} - (k^2 - (k-1)^2))$ for other pairs of i and k.

   We can set up a DP table with entries corresponding to $r_{i,k}$ and can fill it using a double loop with k being for the outer loop; of course, we use the above recursion. The number of entries is $O(n^2)$. And each entry takes $O(n)$ time to compute. Thus, the RT is $O(n^3)$ to compute all the entries. And we just need to return $\max_{0\le k\le n-1} r_{n,k}$ as the max profit.

   Main recursion is worth 15 points. The RT analysis and DP algorithm description is worth 15 points.

   We note that there are several alternative solutions. For example, you can alternatively define $r_{i,k}$ to be the max revenue you can get out of length i by making exactly k cuts, without factoring in the cutting cost. Then, the main recurrence will look like $r_{i,k} = \max_{1\le j\le i-1}(p_j + r_{j-i,k-1})$. And at the end, we return $\max_{0\le k\le n-1}(r_{n,k} - k^2)$ as the max profit.