Problem Set 5

> **Due date:** Electronic submission of the pdf file of this homework is due on **10/8/2021 before 11:59pm** on ecampus.

Name: [redacted]

Resources. (All people, books, articles, web pages, etc. that have been consulted when producing your answers to this homework)

On my honor, as an Aggie, I have neither given nor received any unauthorized aid on any portion of the academic work included in this assignment. Furthermore, I have disclosed all resources (people, books, web sites, etc.) that have been used to prepare this homework.

Signature: [redacted]

Make sure that you describe all solutions in your own words. Typesetting in LaTeX is required. Re-read chapters 15 and 16 in our textbook. Read Chapter 17.

Problem 1 (20 points)**.** Solve Exercise 15.3-4 on page 389 of our textbook.

Solution. For problem 15.3-4, we want to find an instance of the matrix chain multiplication problem for which the greedy approach yields to a suboptimal solution. So let's take the matrices, A_1, A_2, A_3, and A_4, where the dimensions would be p_0, p_1, p_2,p_3, p_4. Let's say A_1 is 5x6, A_2 6x3, A_3 is 3x2, then we can take $(A_1A_2)A_3 = 120$. On the flip side, suppose we grouped it in a different way, where $(A_1(A_2A_3))$, which is equal to 96. This number is less than Professor Capulet's method, therefore showing that her greedy approach is the suboptimal solution.

Problem 2 (20 points)**.** Solve Exercise 15.4-1 on page 396 of our textbook.

Solution. For problem 15.4-1, we must determine an LCS of $\langle 1, 0, 0, 1, 0, 1, 0, 1\rangle$ and $\langle 0, 1, 0, 1, 1, 0, 1, 1, 0\rangle$. The best approach to find the LCS, is by first observing the first list, where it contains a 00 consecutively, where as the second list does not have that. A nested loop could be used to parse through these lists, where you would have (i,j). In the second list, there is 11 consecutively, that occurs twice in the list, but does not appear in the first list. Futhermore, in LCS you would skip past 3 elements at least for every double, and ensuring theres no 2 same elements side by side would result in $\langle 1, 0, 1, 0, 1, 0\rangle$.

Problem 3 (20 points)**.** Solve Exercise 15.4-5 on page 397 of our textbook.

Solution. For problem 15.4-5, we must give an $O(n^2)$ time algorithm to find the longest monotonically increasing subsequence of a sequence of n numbers. So let's denote the longest monotonically increasing subsequence. There is an array s, of the double array which has the length of [n][2]. So considering the array s[i][0], which stores the length of the longest increasing subsequence which ends at i, and s[i][1] contains the index of the number which is the previous position of the longest increase subsequent. For pseudocode, you could run a quicksort algorithm, take the length of LCS, and print. This short method of quicksort, and determining the length, etc has the total complexity of $O(n^2)$.

Problem 4 (20 points)**.** Solve Exercise 16.2-5 on page 428 of our textbook.

Solution. For problem 16.2-5, we have to describe an efficient algorithm based on a given set. So to begin, we should sort the points by obtaining a new array. The first interval would be $[y_1, y_1 + 1]$, so considering the leftmost interval, if there is no point on the leftmost interval then you would go to the next interval which would be $[y_1, y_1 + 1]$, and so on. The most efficient algorithm to utilize to successfully perform this is the greedy algorithm, since the rightmost element should be part of the interval and won't perform better than the interval $[y_1, y_1 + 1]$. This further shows that any subproblem would also be optimal, and also stands considering the points are greater that $y_1 + 1$, hence this algorithm is the most efficient to use. Furthermore, the complexity of this algorithm is not worse than the polynomial time.

Problem 5 (20 points)**.** Solve (a) Exercise 17.1-3 on page 456 and (b) Exercise 17.2-2 on page 459 of our textbook.

Solution. a) For problem 17.1-3, we must use the aggregate analysis to determine the amortized cost per operation. For the aggregate method, in the sequence of the operations there are $2^{(\log_2 n)+1}$, where 1,2,3,4,...., $2^{\log_2 n)+1}$. The total costs are $\sum_{i=0}^{(\log_2 n)} 2i = 2^{(\log_2 n)+1} - 1 \leq 2^{\log_2 n+1} = 2$n. The remaining operation has the cost of 1 and its less than n such operations. Thus, the total cost of operations is T(n) $\leq 2n + n = 3n = O(n)$. Therefore $O(1)$ is amortized cost per operation.
b) For problem 17.2-2, we must use the accounting method for the same problem. We can modify every operation with 3, where 1 is paid for the operations of the non power of '2'. 2 is put still and paid later for the operations of power '2'. There for the amortized cost per operation using accounting method is 3, and complexity is $O(1)$.
****

Discussions on canvas are always encouraged, especially to clarify concepts that were introduced in the lecture. However, discussions of homework problems on canvas should not contain spoilers. It is okay to ask for clarifications concerning homework questions if needed. Make sure that you write the solutions in your own words.

Checklist:
☐ Did you add your name?
☐ Did you disclose all resources that you have used?
(This includes all people, books, websites, etc. that you have consulted)
☐ Did you sign that you followed the Aggie honor code?
☐ Did you solve all problems?
☐ Did you submit the pdf file of your homework?