Sabancı University
Faculty of Engineering and Natural Sciences
CS301 – Algorithms
Midterm Exam

November 30, 2020 @ 20:15–21:00

PLEASE NOTE:

- Provide only the requested information and nothing more.
- Unreadable, unintelligible and irrelevant answers will not be considered.

NAME: ____ **ID:** _________

Question	Learning Outcome	Maximum Points	Points
1	1	10	
2	1	20	
3	3	10	
4	3	20	
	Total	100	

Question 1) [10 points] Derive the recurrence for the running time of the algorithm given below. Give 1-2 sentence explanation.

```
void looseFunction(n):
   if n <= 1:
      return
   if n > 6:
      looseFunction(n/3)
   else:
      looseFunction(n/3)

   for i in range(n):
      dummy(); // Constant Time Complexity Function O(1)

   looseFunction(n/6)
   looseFunction(n/6)
```

Question 2) [20 points] Using the substitution method, show that the following recurrence is super-linear. Show your steps.

$$T(n) = T(n/3) + T(2n/3) + \Theta(n)$$

[NA: If you do not upload any answer for Question 2, you will get 20% of the points.]

Question 3) [10 points] Please recall the *Matrix Chain Multiplication Problem* that we studied in the class. In this problem, we are given a matrix multiplication in the form

$$A^1_{(p_0 \times p_1)} \times A^2_{(p_1 \times p_2)} \times \cdots \times A^n_{(p_{n-1} \times p_n)}$$

where for each $1 \leq i \leq n$, $A^i_{(p_{i-1} \times p_i)}$ is a p_{i-1} by p_i matrix. We are asked to find the smallest number of scalar multiplications required to multiply these n matrices with the given dimensions.

We saw a solution for this problem using dynamic programming. As in the case of all dynamic programming algorithms, the solution is based on the recurrence which was the following:

$$m[i,j] = \begin{cases} 0 & \text{if } i = j \\ \min\limits_{i \leq k < j} \{m[i,k] + m[k+1,j] + (p_{i-1} \times p_k \times p_j)\} & \text{if } i < j \end{cases}$$

Using this recurrence and by showing your calculations, fill in the following dynamic programming table for the multiplication of the following 4 matrices with the given sizes:

$$A^1_{(3\times5)} \times A^2_{(5\times2)} \times A^3_{(2\times20)} \times A^4_{(20\times3)}$$

$m[i,j]$:

i \ j	1	2	3	4
1				
2				
3				
4				

Question 4) [20 points]

Please recall the *Longest Common Subsequence* (LCS) problem that we studied in the class.

> Given two subsequences $A = \langle a_1, a_2, \ldots, a_n \rangle$ and $B = \langle b_1, b_2, \ldots, b_m \rangle$ find a sequence $C = \langle c_1, c_2, \ldots, c_k \rangle$ such that C is a subsequence of both A and B and k (i.e. the length of C) is maximized.

We first introduced the notation A_i to denote $A_i = \langle a_1, a_2, \ldots, a_i \rangle$, i.e. the part of the sequence A between its first element a_1 and its i^{th} element a_i. Similarly, we use B_i and C_i to denote the part of the sequences B and C between their first elements and their i^{th} elements.

We then made the following observations:

- If $a_n = b_m$ (i.e. the last elements of A and B are the same), then we need to have $c_k = a_n = b_m$ as well, i.e. the last element of an LCS C must be the same. Furthermore, C_{k-1} must be an LCS of A_{n-1} and B_{m-1}.
- If $a_n \neq b_m$ (i.e. the last elements of A and B are different), then we either have
  - C is an LCS of A_{n-1} and B, or
  - C is an LCS of A and B_{m-1}

Based on these observations, we were able to write the recurrence $s[i,j]$ (LCS for A_i and B_j) that we can use for our dynamic programming solution as follows:

$$s[i,j] = \begin{cases} \langle\ \rangle & \text{if } i = 0 \text{ or } j = 0 \\ s[i-1, j-1] \cdot a_i & \text{if } i,j > 0 \text{ and } a_i = b_j \\ \text{longer of } s[i-1,j] \text{ and } s[i,j-1] & \text{otherwise} \end{cases}$$

Now, let us introduce the notation ${}_iA$ to denote ${}_iA = \langle a_i, a_{i+1}, \ldots, a_n \rangle$, i.e. the trailing part of A starting with element a_i and running upto the last element a_n. Note that ${}_nA = \langle a_n \rangle$ and ${}_{n+1}A = \langle\ \rangle$. Assume that we similarly use ${}_iB$ and ${}_iC$ to denote the sequences B and C starting with their i^{th} elements as well.

For an LCS $C = \langle c_1, c_2, \ldots, c_k \rangle$ of sequence $A = \langle a_1, a_2, \ldots, a_n \rangle$ and $B = \langle b_1, b_2, \ldots, b_m \rangle$ we also made the following observations in the class:

- If $a_1 = b_1$ (i.e. the first elements of A and B are the same), then we must have $c_1 = a_1 = b_1$ as well, i.e. the first element of an LCS C must be the same. Furthermore, ${}_2C$ must be an LCS of ${}_2A$ and ${}_2B$.
- If $a_1 \neq b_1$ (i.e. the first elements of A and B are different), then we either have
  - C is an LCS of ${}_2A$ and B, or
  - C is an LCS of A and ${}_2B$

Although we made these observations as well, we did not write the recurrence that one would obtain by using this approach. Now it is time for you to write the recurrence for $p[i, j]$ (LCS for ${}_iA$ and ${}_jB$) that we can use for our dynamic programming solution.

Please fill–in the blanks in the following template recurrence:

$$p[i,j] = \begin{cases} \langle\,\rangle & \text{if } \_\_\_\_\_\_\_\_\_\_\_\_ \\ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ & \text{if } \_\_\_\_\_\_\_\_\_\_\_\_ \\ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ & \text{otherwise} \end{cases}$$