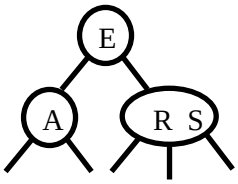
CENG 218 Spring 2023 Homework 2
Due date: 26th of May 2023, 23:59

Submit photos or scans of your solutions via MS-Teams. Homeworks submitted after the due date will not be evaluated. Write all your explanations and comments in English! Text in Turkish will not be evaluated. Submit a single pdf file or a single zip file.

Q1 (20 points). Below is a 2-3 tree, into which 4 elements were inserted so far. Please insert C,H,X,P and L in this tree in the given order. Show all intermediate steps with your drawings.



Q2 (35 points).

Matrix-chain multiplication problem:
We are given a matrix sequence (chain) $A_1, A_2, \ldots, A_n$ and we wish to compute the product $A_1A_2\ldots A_n$
Any order gives the same product but the order changes the amount of scalar multiplications we do.
Illustration: Consider the problem of a chain $<A_1, A_2, A_3>$
Dimensions of A_1, A_2 and A_3 are 10x100, 100x5, and 5x50 respectively.
Multiplication of $((A_1A_2)A_3)$ takes 7500 scalar multiplications, whereas $(A_1(A_2A_3))$ takes 75000.
The problem is finding the multiplication order that takes minimum amount of scalar multiplications.

a) (10 points) What is the complexity of the brute-force algorithm? I.e. How much time does it take to try out all alternatives to find the best order?
b) (10 points) Please refer to Section 15.2 in your textbook and shortly explain how this problem is solved with DP? Give the formulation.
c) (10 points) Solve the example below using DP. Do not only write the asnwer but also show the table constructed with bottom-up approach.

matrix	A_1	A_2	A_3	A_4	A_5	A_6
dimension	30×35	35×15	15×5	5×10	10×20	20×25

d) (5 points) What is the complexity of bottom-up dynamic programming algorithm?

Q3 (20 points).

In Merge-sort we are recursively dividing the problem into two subproblems and solve them. Explain why a dynamic programming algorithm, using memoization for example, does not speed up a good divide-and-conquer approach such as merge-sort?

Q4 (25 points).

Remember the activity selection problem: We have a set $S = \{a_1, a_2, \ldots, a_n\}$ of n proposed ***activities*** that wish to use a resource. Each activity a_i has a ***start time*** s_i and a ***finish time*** f_i .

Activities a_i and a_j are ***compatible*** if the intervals $[s_i , f_i)$ and $[s_j , f_j)$ do not overlap. We wish to select a maximum-size subset of compatible activities. Consider the following set of activities:

i	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
s_i	3	1	4	3	5	6	7	8
f_i	5	4	7	9	9	10	11	12

a) Consider the following greedy approach: "Selecting the activity of least duration from those that are compatible with previously selected activities". Does this approach provide an optimal solution? Explain your answer.

b) Consider the following greedy approach: "Selecting the last activity to start". Does this approach provide an optimal solution? Explain your answer.