COMP 3711 Final, Spring 2022, Wednesday May 25

Part 1: 4:30-5:45 pm

P1, 4% Complexity	P2, 10% D&C	P3, 10% Sorting	P4, 10% Greedy	P5, 10% Stable Match	P6, 6% BFS, DFS

Problem 1, Complexity, 4%

Let $T_A(n)$ and $T_B(n)$ denote the time complexities of two algorithms A and B respectively, with respect to the input size n. Below are 4 different cases of time complexities for each algorithm. For each of the following 4 cases, write "A", "B", or "U", where:

- "A" means that algorithm A is faster;
- "B" means that algorithm B is faster;
- "U" means that we do not know which algorithm is faster.

Case	$T_A(n)$	$T_B(n)$
1	$\Theta(n^{2.1})$	$\Theta(n^2\log^3 n)$
2	$\Theta(n^9)$	$\Theta(2^{\sqrt{n}})$
3	$\Omega(n^3)$	$O(n^{2.1}\log^3 n)$
4	$O(n^3)$	$O(n^{2.1}\log^3 n)$

Case 1:
Case 2:
Case 3:
Case 4:

Problem 2, Divide and Conquer, 10%

You are given an array $A[1..n]$, where $A[1]=0$, $A[n]=0$, and all the other elements are distinct positive numbers, i.e., $A[i]>0$, $\forall\ i\ 1<i<n$.
$A[i]$ is a *peak element*, if it is larger than both its neighbors, i.e., $A[i] > A[i-1]$ and $A[i] > A[i+1]$. For example, if $A=[0, 8, 9, 2, 1, 3, 0]$, the peak elements are $A[3]=9$ and $A[6]=3$.
Describe a O(log$n$) Divide and Conquer algorithm for returning the position of a peak element in A (no need to find all peak elements). Include the pseudo-code and briefly describe the recurrence for the running time $T(n)$.

Problem 3, Sorting, 10%

You are given an array $A[1..n]$, where all elements are distinct positive numbers (not necessarily integers). Describe an $O(n\log n)$ algorithm that re-arranges the elements of A, so that (i) every element at an even position is a *peak element*, and (ii) the peak elements are in increasing order.
For example, if $A=[7, 1, 3, 4, 5, 6, 2]$, valid outputs include [1,**3**,2,**5**,4,**7**,6] or [1,**5**,2,**6**,3,**7**,4] (the peak elements are underlined). Recall from the previous exercise that $A[i]$ is a *peak element*, if $A[i] > A[i-1]$ and $A[i] > A[i+1]$). Discuss the overall complexity of your approach.
Hint: You can use any sorting algorithm as a black box.

Problem 4, Greedy, 10%

Let two sequences $X = (x_1, \dots, x_m)$ and $Y = (y_1, \dots, y_n)$, where $m\leq n$. Write the pseudocode for a greedy algorithm that determines if X is a subsequence of Y. For example, if X = ACDA and Y = ADCBDAB, then the output should be "yes", whereas if X = CDBB, it should be "no". For full credits your algorithm should run in O(n) time.

Problem 5, Stable Matching, 10%
Consider three men M1, M2, M3 and three women W1, W2, W3 with the following preferences:

For men
M1: W1, W2, W3
M2: W1, W2, W3
M3: W2, W1, W3

For women
W1: M1, M2, M3
W2: M2, M1, M3
W3: M2, M3, M1

Question 1, 5%: Show that the assignment (M1, W1), (M2, W3), (M3, W2) is unstable.
Question 2, 5%: Find a stable matching and explain whether it is optimal for the men, the women or both.

Problem 6, BFS, DFS, 6%
Consider the following adjacency lists representing a directed graph G:
adj(s) = [v_1, v_3, v_4],
adj(v_1) = [ ],
adj(v_2) = [v_4],
adj(v_3) = [v_2, v_5],
adj(v_4) = [v_3],
adj(v_5) = [s].

Question 1, 3% What is the order of nodes visited by Breadth First Search (BFS) starting from node s in G. Nodes in the same list are visited according to their lexicographic order.
Question 2, 3% What is the order of nodes visited by Depth First Search (DFS) starting from node s in G. Nodes in the same list are visited according to their lexicographic order.