CS 430 Introduction to Algorithms **Spring Semester, 2021**

Second Extra Homework for Remote Students

Assigned: April 14 *Due: April 28*

.

Please respect the following guidelines for writing pseudocode:

1. C/Java/Python instructions are fine. But do not write object-oriented additions. Do not declare or use any class. Declare only procedures (if necessary) and explain in words what each procedure does, and what is the use of each parameter.
2. One instruction per line
3. Match the brackets with a horizontal line
4. Number your lines
5. Write down if your array is indexed $0 \ldots n-1$ or $1 \ldots n$.
6. If you decide to use procedures/functions/methods from the class (such as, say, DELETE-MAX(S) for S a max-heap), explain clearly what they do and in what running time. Explain the data structures, if any.

Problem 1 Present an algorithm with running time $O(n)$ for the following problem: Given array $A[1..n]$, find entries $1 \leq i < j \leq n$ such that $A[j] - A[i]$ is maximum among all such pairs $i < j$. As an example, if $A =< 5, 9, 1, 4, 7, 2 >$, output $i = 3$ and $j = 5$.

Make sure to include pseudocode, correctness and running time arguments. A $O(n \log n)$-time algorithm is worth partial credit. An $O(n^2)$-time algorithm is trivial and only gives half of score if complete.

Hint: Dynamic programming.

Problem 2 In an undirected graph $G = (V, E)$, a ***clique*** is a subset A of vertices such that any two distinct vertices of A are adjacent. A ***maximal clique*** M is a clique such that, if we were to add any additional vertex to M, then it would not be a clique any longer. Every graph has a maximal clique. (Can you see this? This question is not part of the exercise, but it is worth thinking about.) Give an efficient algorithm that computes a maximal clique for an input graph G.

Present full pseudocode (using the format in the notes) and analyze the running time. A correctness proof is not required (but the algorithm must be correct). $O(|V|+|E|)$-time is achievable and needed for full marks. $O(|V|+|E|\log|E|)$-time can get up to 95% of the marks, and any algorithm that is polynomial up to 85% of the marks.

Problem 3 $G=(V,E)$ is an undirected graph whose edges have weight w. A subgraph of G is called *spanning* if it has V as its vertex set. A spanning subgraph of G is identified with its set of edges. A spanning tree is *minimum* if the total weight of its edges is minimum among all the spanning trees.

Assume no two edges have the same weight. Prove that the minimum spanning tree is unique in this case. Show by example that the second-best minimum spanning tree need not be unique. Explain why your example has the desired properties: show the minimum spanning tree, and at least two distinct second-best minimum spanning trees.

Problem 4 Suppose we are given a weighted directed graph $G=(V,E,c)$ with positive costs on the edges, and two nodes $u,v \in V$. Give an efficient (polynomial) algorithm for computing the number of shortest $u-v$ paths in G. Do not attempt to list all these paths!

Present pseudocode, analyze the running time, and prove correctness.