THE HONG KONG POLYTECHNIC UNIVERSITY Fall 2020
COMP3011: Design and Analysis of Algorithms

Assignment 4

Due: Monday, 07 December 2020, 3:30 p.m.

To submit the assignment, upload a .pdf-, .doc-, .docx-, .txt-, or .jpg-file with your solutions to Blackboard before the deadline. The filename should start with "A4_99999999Z" (replace "99999999Z" by your student ID number). Include your name and student ID number at the top of each page. Scanned, handwritten solutions are OK as long as they are easily readable. You may specify your algorithms in pseudocode or in plain English.

The maximum possible score is 15 marks. Late submissions will not be accepted. The assignment is to be completed individually, i.e., every student must turn in his or her own solutions. Students are allowed to discuss solution strategies with other students but they are not allowed to copy each others' answers.

1. In this question, consider the two algorithms for computing convex hulls named Graham's scan (the MERGE-SORT-based implementation) and Jarvis's march that were presented in Lecture 10. Show how to define, for every positive integer $n \geq 3$, a sequence of n points in the plane for which Jarvis's march is asymptotically faster than Graham's scan. Also, explain why Jarvis's march is faster in this particular case. *(2 marks)*

2. Three points in the plane are called *collinear* if they lie on the same straight line. For example, the three points $(0, 1)$, $(1, 3)$, $(3, 7)$ are collinear. Describe an algorithm that solves the following problem efficiently and analyze its time complexity.

   Input: A sequence Q of points in the plane.
   Output: "YES" if Q contains three points that are collinear; "NO" otherwise.

   Let n denote the number of points in the input, i.e., $n = |Q|$. To get full marks, the worst-case running time of your algorithm must be $O(n^2 \lg n)$. Note that two or more points in the input Q may have the same coordinates. *(3 marks)*

3. Let Independent-Set and U-LCS be the two decision problems defined below. U-LCS is "unbounded" in the sense that the number of input sequences as well as the size of the alphabet that the symbols belong to can be arbitrarily large and are not fixed in advance.

   Independent-Set:
   Input: A positive integer k and an undirected graph G with vertex set V and edge set E.
   Output: "YES" if there exists a subset S of V with $|S| \geq k$ such that no two vertices in S are adjacent in G; "NO" otherwise.

   The Unbounded Longest Common Subsequence Decision Problem (U-LCS):
   Input: A positive integer ℓ and a set $\mathcal{X}$ of sequences of symbols.
   Output: "YES" if there exists a common subsequence of $\mathcal{X}$ of length $\geq \ell$; "NO" otherwise.

   Independent-Set is known to be $\mathcal{NP}$-complete (you do not need to show it here). Prove that U-LCS is $\mathcal{NP}$-complete. *(3 marks)*

4. According to question 3. above, Independent-Set is $\mathcal{NP}$-complete. Is the restriction of Independent-Set to instances in which the input graph G has no cycles of length 3 still $\mathcal{NP}$-complete? If yes, give a proof; if not, describe a polynomial-time algorithm that solves it. *(3 marks)*

(Continued $\longrightarrow$)

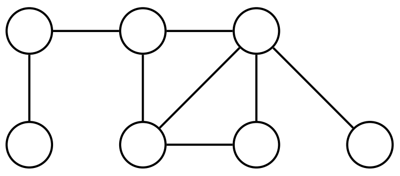
5. If $\mathcal{P} \neq \mathcal{NP}$, would it be possible for a decision problem to belong to $\mathcal{P}$ and also be $\mathcal{NP}$-complete? Justify your answer. *(1 mark)*

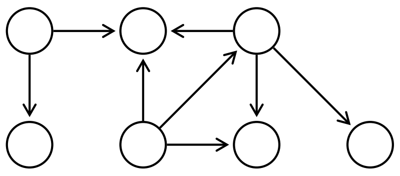
6. Define the following optimization problem.

   Input: A connected, undirected graph G.

   Output: An assignment of a direction to each edge in G such that the number of vertices with at least one outgoing edge is as small as possible.

   As an example, if the input is the undirected graph on the left then the directed graph on the right is an optimal solution. It has three vertices with at least one outgoing edge, and this is the minimum possible because it's impossible to assign directions to all the edges so that only two vertices get outgoing edges.

   Describe a polynomial-time 2-approximation algorithm for the problem. Prove that your algorithm's approximation ratio is at most 2 and that it runs in polynomial time.

   Hint: Use one of the approximation algorithms that was presented during the lectures. *(3 marks)*