COMPSCI 311: Introduction to Algorithms	Fall 2019

Homework 6

Released 11/23/2019 — Due 12/11/2019 11:59pm in Gradescope

Instructions. You may work in groups, but you must write solutions yourself. List collaborators on your submission. Also list any sources of help (including online sources) other than the textbook and course staff.

If you are asked to design an algorithm, please provide: (a) the pseudocode or precise description in words of the algorithm, (b) an explanation of the intuition for the algorithm, (c) a proof of correctness, (d) the running time of your algorithm and (e) justification for your running time analysis.

This homework has 110 points, but the A level will be no higher than 90, so you may think of any ten of the points as extra credit if you like.

Submissions. Please submit a PDF file. You may submit a scanned handwritten document, but a typed submission is preferred. Please assign pages to questions in Gradescope.

1. **(20 points) Use All Letters** Consider a game where you are given a set of tiles, each with a printed letter, and you have to arrange them to make words, much like in Scrabble. It may be easy to make *some* words, but becomes much harder if you have to use *all* given letters.

More precisely, given a *multiset* of symbols (letters) L from an alphabet Σ (thus, the same letter may appear in L multiple times), and a set of words $W \subseteq \Sigma^*$, UseAllLetters asks if it is possible to use all letters from L to make words belonging to W. You may form the same word multiple times (so if you have g and d twice, and four o's, you might spell *good* twice if it is in W). Prove that UseAllLetters is NP-complete.

2. **(20 points) The Last Token** Consider the following game played on a graph G where each node can hold an arbitrary number of tokens. A move consists of removing two tokens from one node (that has at least two tokens) and adding *one* token to some neighboring node. The LastToken problem asks whether, given a graph G and an initial number of tokens $0 \leq t(v) \leq n^2$ for each vertex v, there is a sequence of moves that results in only one token being left in G. Prove that LastToken is NP-complete.

3. **(20 points) Intersection Inference** (K&T Exercise 8.16)

4. **(20 points) Independent Set** (K&T Exercise 8.22)

5. **(30 points) Triangle-TSP** Recall that the input to the TSP problem is a function d, where $d(x, y)$ is the distance from node x to node y. We require that $d(x, x) = 0$ and that $d(x, y) > 0$ if $x \neq y$.

Such a distance function is said to obey the *triangle inequality* if $d(x, z) \leq d(x, y) + d(y, z)$ for any three nodes x, y, and z. The TriangleTSP problem is the special case of TSP where the distance function satisfies the triangle inequality, so it is $\{(d, k)\colon d$ satisfies the triangle inequality and there is a tour of all the cities with total distance at most $k\}$.

(a) Prove that TriangleTSP is NP-complete. (Hint: Look at the reduction in lecture from HAM-Circuit to ordinary TSP.)

(b) The *2-Approximate TSP* problem is to input a distance function d and output a tour of the cities that has total distance at most twice that of the optimal tour.

Prove that the 2-Approximate TSP problem is NP-hard, meaning that if there were a polynomial-time algorithm for it, P would equal NP. (The most natural way to prove this is to use such an algorithm to solve the NP-complete Hamiltonian cycle problem.)

(c) If your solution to (b) was correct, the distance function you constructed did *not* satisfy the triangle inequality. We know that because for symmetric distances, the 2-Approximate TriangleTSP problem is in P, and we don't believe that you proved P equals NP. Prove that 2-Approximate TriangleTSP is in P, assuming the distance function is symmetric. (Hint: Consider a minimum spanning tree for the weighted graph that has all possible edges, with each edge weighted according to d.)

6. **(0 points).** How long did it take you to complete this assignment?