CSC373 Summer '22
Assignment 4

Due Date: August 13, 2022, 11:59pm ET

Instructions

1. Typed assignments are preferred (e.g., PDFs created using LaTeX or Word), especially if your handwriting is possibly illegible or if you do not have access to a good quality scanner. Either way, you need to submit a single PDF named "hwk4.pdf" on MarkUS at `https://markus.teach.cs.toronto.edu/2022-05`

2. You will receive 20% of the points for a (sub)question when you leave it blank (or cross off any written solution) and write "I do not know how to approach this problem." If you leave it blank but do not write this or a similar statement, you will receive 10%. This does not apply to any bonus (sub)questions.

3. You may receive partial credit for the work that is clearly on the right track. But if your answer is largely irrelevant, you will receive 0 points.

Q1 [15 Points] Complement of an NP Problem

For a decision problem D_0, define $\overline{D_0}$ to be its complement problem i.e., a YES instance of D_0 is a NO instance of $\overline{D_0}$ and a NO instance of D_0 is a YES instance of $\overline{D_0}$. Below, "$\leq_K$" denotes a *Karp* reduction as defined in class (i.e., a polynomial time reduction that only produces a single instance and preserves the YES/NO value).

(a) [7.5 Points] Give a decision problem D_0 such that $D_0 \leq_K \overline{D_0}$.
Define D_0 precisely, describe the reduction $D_0 \leq_K \overline{D_0}$ in detail, and prove that your reduction is correct.

(b) [2.5 Points] Does your decision problem D_0 from part (a) belong to NP? Justify.

(c) [5 Points] Suppose that D_1 is a decision problem such that $D_1 \leq_p \overline{D_1}$ and D_1 is NP-complete. What can you conclude? Make the strongest claim you can and prove it.

Q2 [15 Points] Exponential Algorithms for NP Problems

Show that for every decision problem $D \in$ NP, there is some polynomial $p(n)$ and some algorithm A (both of which depend on D) such that A solves D in worst-case time $O\big(2^{p(n)}\big)$.
HINT: This involves mostly "unrolling" the definitions, so write up your answer carefully. In your answer, you must use the formal definition of NP in terms of polynomial-time *verifiers* – for this, you can refer to class slides or page 1064 of the text.

Q3 [20 Points] Showing NP-Completeness I

A web server has a number of simultaneous "requests" to reply to. The server has to send its replies in "packets," each one of which has a fixed positive integer size limit L. Each packet can contain

more than one reply (so multiple requests can be replied to in a single packet), but each individual reply has its own positive integer size s_i. We would like to use as few packets as possible to send all of the replies.

This problem can be formulated formally as a decision problem FEWPACKETS ("FP" for short), as follows:

Input: Positive integer *packet size limit* L, positive integers *reply sizes* $s_1, \ldots, s_n$, positive integer *number of packets* k.

Output: Is there some partition of $\{1, \ldots, n\}$ into packets $P_1, \ldots, P_k$, where each packet has size at most L– formally: $\exists P_1, \ldots, P_k, P_1 \cup \cdots \cup P_k = \{1, \ldots, n\} \wedge (\forall i, j, P_i \cap P_j = \emptyset) \wedge \forall i, \sum_{j \in P_i} s_j \leq L$?

For example, $(8, \{2, 5, 4, 5\}, 3) \in$ FP because we can use packets $P_1 = \{1, 3\}$ (with size $s_1 + s_3 = 2 + 4 \leq 8$), $P_2 = \{2\}$ (with size $s_2 = 5 \leq 8$), and $P_3 = \{4\}$ (with size $s_4 = 5 \leq 8$). However, $(8, \{2, 5, 4, 5\}, 2) \notin$ FP because there is no way to partition every reply into only 2 packets, each one of size ≤ 8 – even though the total size of all replies is only 16.

Write a *detailed* proof that FP is NP-complete: first show that FP $\in$ NP, and then use a reduction from SUBSET-SUM to show that FP is NP-hard. You may assume that the following "positive integer" version of SUBSET-SUM is NP-complete:

Input: Positive integer W (target sum) and a set S of n positive integers $\{x_1, \ldots, x_n\}$.

Output: Is there some subset $\emptyset \neq S' \subseteq S$ that sums to exactly W?

Q4 [20 Points] Showing NP-Completeness II

Imagine that we have a network of processors, each one of which needs access to a common database in order to carry out its work (for example, a network of banking machines accessing client information). To simplify, assume all operations are queries (no operation changes the data). Our problem is to determine which processor(s) should store the database (these processors will be called "servers") in order to have reasonably fast access time (including the load on each server), as well as having as little duplication of the database as possible.

At one extreme, each processor could store its own copy of the database. This would ensure the fastest possible access time, but at the cost of replicating the database many times. At the other extreme, the database could be stored in a single server (all other processors would connect to the server to access the data). This would ensure the smallest amount of duplication, but at the cost of longer access times and server load. In either case, there is a significant cost if the network is large.

A reasonable middle ground between the two extremes would be to fix a distance parameter d and to store the database on as few servers as possible, chosen so that every processor is within distance d of at least one server (the distance is measured as the minimum number of links required to send communications from one processor to the next). Unfortunately, it seems impossible to solve this problem efficiently.

Formally, show that the following SERVERLOCATION problem is NP-complete. You may use any NP-complete problem from class for your reduction.

Input: An undirected graph G (the network of processors) and non-negative integers k and d.

Output: Is there some subset S of no more than k vertices (the "servers") such that every processor is within distance d of some server in S? (The "distance" between vertices u, v is measured as the number of edges on a shortest path from u to v, or infinity (∞) if there is no path from u to v.)