CSC 440

Assignment 7: P, NP, and Approximation

Due Monday, May 2nd by 11:59PM

You may work in small groups on this assignment, but the work you hand in must be your own. You may submit this in one of two ways:

- Write it up electronically (e.g. LaTeX) and upload a PDF to Gradescope.
- Scan your handwritten solution (you can use a scanning app for a smartphone, such as Evernote's free Scannable app) and upload a PDF to Gradescope.

Your full name:

1. Coffeeshops (20 pts):

TLC Coffee has decided to expand to maximize profits over all of South County, by building new TLCs on many street corners. The management turns to URI's Computer Science majors for help. The students come up with a representation of the problem:

The street network is described as an arbitrary undirected graph $G = (V, E)$, where the potential restaurant sites are the vertices of the graph. Each vertex u has a nonnegative integer value p_u, which describes the potential profit of site u. Two restaurants cannot be built on adjacent vertices (to avoid self-competition). The students further come up with an exponential-time algorithm that chooses a set $U \subseteq V$, but it runs in $O(2^V|E|)$ time. The TLC ownership is dismayed, and is convinced they should have hired UConn's CS majors instead. In order to prove that nobody else could have come up with a better algorithm, you need to prove the hardness of the coffeeshop problem.

Define coffeeshop to be the following decision problem: given an undirected graph $G = (V, E)$, given a mapping p from vertices $u \in V$ to nonnegative integer profits $p(u)$, and given a nonnegative integer k, decide whether there is a subset $U \subseteq V$ such that no two vertices in U are neighbors in G, and such that $\sum_{u \in U} p(u) \geq k$.

Prove that coffeeshop is NP-hard. (Hint: Try a reduction from 3SAT.)

Also, explain why this implies that, if there is a polynomial-time algorithm to

solve the original problem, i.e., to output a subset U that maximizes the total profit, then $P = NP$.

2. Ghostbusters and Ghosts (20 pts):

A group of n Ghostbusters is battling n ghosts. Each Ghostbuster carries a proton pack, which shoots a stream at a ghost, eradicating it. A stream goes in a straight line and terminates when it hits the ghost. The Ghostbusters decide upon the following strategy:

They will pair off with the ghosts, forming n Ghostbuster-ghost pairs, and then simultaneously each Ghostbuster will shoot a stream at her chosen ghost. As we all know, it is *very dangerous* to let streams cross, and so the Ghostbusters must choose pairings for which no streams will cross.

Assume that the position of each Ghostbuster and each ghost is a fixed point in the plane and that no three positions are collinear. This problem has two parts (each 10 pts):

(A) Argue that there exists a line passing through one Ghostbuster and one ghost such that the number of Ghostbusters on one side of the line equals the number of ghosts on the same side. Describe how to find such a line in $O(n \log n)$ time.

(B) Give an $O(n^2 \log n)$-time algorithm to pair Ghostbusters with ghosts in such a way that no streams cross.

3. TRIPLE-SAT (20 pts):

Let TRIPLE-SAT denote the following decision problem: given a Boolean formula ψ, decide whether ψ has at least three distinct satisfying assignments. Prove that TRIPLE-SAT is NP-complete using a reduction.

4. Clustering (20 pts):

Consider the following approach to grouping n distinct objects $p_1 \cdots p_n$ on which a distance function $d(p_i, p_j)$ is defined such that $d(p_i, p_i) = 0$, $d(p_i, p_j) > 0$ if $i \neq j$ (that is, p_i and p_j are distinct objects), and $d(p_i, p_j) = d(p_j, p_i)$ (distances are symmetric.) The clustering approach is as follows: divide the n points into k clusters such that we *minimize* the maximum distance between any two points in the same cluster. That is, we seek low-diameter clusters. Suppose we want to know whether, for a particular set of n objects, and a bound B, the objects can be partitioned into k sets such that no two points in the same set are further than B apart. Prove that this decision problem is NP-complete.

5. Knapsack approximations (20 pts):

In the Knapsack problem, we are given a set $A = a_1, ...a_n$ of items, where each a_i has a specified positive integer size s_i and a specified positive integer value v_i. We are also given a positive integer knapsack capacity B. Assume that $s_i \leq B$ for every item i. The problem is to find a subset of A whose total size is at most B and for which the total value is maximized. In this problem, we will consider approximation algorithms to solve the Knapsack problem. Notation: For any subset S of A, we write s_S for the total of all the sizes in S and v_S for the total of all the values in S. Let *Opt* denote an optimal solution to the problem. This problem has two parts (each 10 pts).

A: Consider the following greedy algorithm Alg_1 to solve the Knapsack problem:

Order all the items a_i in non-increasing order of their *density,* which is the ratio of value to size, $\frac{v_i}{s_i}$. Make a single pass through the list, from highest to lowest density. For each item encountered, if it still fits, include it, otherwise exclude it. **Prove** that algorithm Alg_1 does not guarantee any constant approximation ratio. That is, for any positive integer k, there is an input to the algorithm for which the total value of the set of items returned by the algorithm is at most $\frac{v_{Opt}}{k}$.

B: Consider the following algorithm Alg_2:

If the total size of all the items is $\leq B$, then include all the items. If not, then order all the items in non-increasing order of their densities. Without loss of generality, assume that this ordering is the same as the ordering of the item indices. Find the smallest index i in the ordered list such that the total size of the first i items exceeds B. In other words, $\sum_{j=1}^{i} s_j > B$, but $\sum_{j=1}^{i-1} s_j \leq B$. If $v_i > \sum_{j=1}^{i-1} v_j$, then return a_i. Otherwise, return $a_1, ..., a_i - 1$.

Prove that alg_2 always yields a 2-approximation to the optimal solution.

Additional Space for problem 5

Additional Space as needed