CS 170 HW 10

Due on 2018-04-09, at 11:59 pm

1 (★) Study Group

List the names and SIDs of the members in your study group.

2 (★) Runtime of NP

True or False (with proof): Suppose we can show for some fixed k, an NP-complete problem P has a time $O(n^k)$ algorithm. Then every language in NP has a $O(n^k)$ time algorithm.

3 (★★★) 2-SAT and Variants

While we showed that 3-SAT last week was NP-complete, today we will explore the variant 2-SAT, where each clause contains at most 2 literals (hereby called a 2-clause).

(a) Show that 2-SAT is in P.

(b) The problem of Max-2-SAT is defined as follows. Let $C_1, \ldots, C_m$ be a collection of 2-clauses and k a non-negative integer . Output 1 if and only if there exists an assignment of the literals such that at least k of the clauses are satisfied.

Show that Max-2-SAT is NP-complete. Reduce from Max-Cut, which is the problem of determining if given input a graph G and an integer k, determining if there exists a cut of weight at least k.

4 (★★★) More Reductions

Given a set S of non-negative integers $[a_1, a_2, \ldots, a_n]$, consider the following problems:

1 **Partition**: Determine whether there is a subset $P \subseteq S$ such that $\sum_{i \in P} a_i = \sum_{j \in S \setminus P} a_j$

2 **Subset Sum**: Given some integer k, determine whether there is a subset $P \subseteq S$ such that $\sum_{i \in P} a_i = k$

3 **Knapsack**: Given some set of items each with weight w_i and value v_i, as well as fixed numbers W and V there is some subset P such that $\sum_{i \in P} w_i \leq W$ and $\sum_{i \in P} v_i \geq V$

For each of the following briefly describe your reduction and provide a few sentences to justify runtime and correctness.

a Find a linear time reduction from Subset Sum to Partition.

b Find a linear time reduction from Subset Sum to Knapsack.

5 (★★★★) An approximation algorithm for Subset-Sum

The **SUBSETSUM** problem is defined as: Given $A = \{a_1, \ldots, a_m\}$ positive integers, and a positive integer t (the "target"), find a subset $A' \subseteq A$ s.t. $\sum_{a \in A'} a = t$.

(a) Give a Dynamic Programming algorithm for Subset Sum. Argue that if the inputs (the a_i and t) are given in unary, this algorithm runs in polynomial time.

(b) **SUBSETSUM** (with the integers given in ordinary binary notation) is known to be **NP**-complete; e.g., there is a reduction from 3-SAT. So, instead of solving it exactly, we are going to come up with an approximation algorithm. For a fixed $\epsilon > 0$, we want to *decide* if a subset $A' \subseteq A$ exists s.t.

$$\sum_{a \in A'} a \in [t(1-\epsilon), t(1+\epsilon)]$$

Specifically, we are going to build a streaming algorithm. In a streaming algorithm, the elements $a_1, \ldots, a_m$ are received one at a time and we take some immediate action when they are received. (We'll be more formal about the model later in the course.)

Here is a sketch for the algorithm. **Fill in the details and prove its correctness.**

> The algorithm maintains T, a set of non-intersecting intervals. Initially, T is a set consisting of a single interval: $\{[t(1-\epsilon), t(1+\epsilon)]\}$.
> While the stream hasn't ended:
>
> (a) Let $a_j \leftarrow$ the next element in the stream.
>
> (b) If there is an interval $[\alpha, \beta] \in T$ such that $a_j \in [\alpha, \beta]$ then output YES.
>
> (c) Update T using a_j. [**You need to explain how T is updated.**]
>
> Otherwise, output NO.

(c) Now we are going to prove that this algorithm runs in **poly**$(n, 1/\epsilon)$ time and space where n is the length of the input.

Give good space and time bounds on this algorithm (in particular be explicit about the polynomial and the $O()$ above). Hint: first find an upper bound on $|T|$.

(d) We would like to also extract the set A' when the answer is YES. Using the previous algorithm as a subroutine extract a set A' that achieves the approximation bound if one exists. Give a total runtime bound.

6 (★★★★★) Steiner Tree

The Steiner tree problem is the following:

Input: An undirected graph $G = (V, E)$ with non-negative edge weights $\text{wt} : E \to \mathbb{N}$, a set $S \subseteq V$ of special nodes.

Output: A Steiner tree for S whose total weight is minimal

A Steiner tree for S is a tree composed of edges from G that spans (connects) all of the special nodes S. In other words, a Steiner tree is a subset $E' \subseteq E$ of edges, such that for every $s, t \in S$, there is a path from s to t using only edges from E'. The total weight of the Steiner tree is the sum of the weights included in the tree, i.e., $\sum_{e \in E'} \text{wt}(e)$.

The Steiner tree problem has many applications in different areas, including creating genealogy trees to represent the evolutionary tree of life, designing efficient networks, to even planning water pipes or heating ducts in buildings. Unfortunately, it is NP-hard.

Here is an approximation algorithm for this problem:

1. Compute the shortest distance between all pairs of special nodes. Use these distances to create a modified and complete graph $G' = (S, E_S)$, which uses the special nodes as vertices, and the weight of the edge between two vertices is the shortest distance between them.

2. Find the minimal spanning tree of G'. Call it M.

3. Reinterpret M to give us a Steiner tree for G: for edge in M, say an edge (r, s), select the edges in G that correspond to the shortest path from r to s. Put together all the selected edges to get a Steiner tree.

Prove that this algorithm achieves an approximation ratio of 2.

(Note that the efficiency of an approximation algorithm does not contradict NP-completeness because it gives an approximate (rather than an exact) solution to the problem.)