CS 577: Introduction to Algorithms Fall 2020

Homework 9

Instructor: Dieter van Melkebeek TA: Ryan Moreno

This homework covers reductions. **Problems 1 and 3 must be submitted for grading by 11:59pm on 11/16**. Please refer to the homework guidelines on Canvas for detailed instructions.

Warm-up problems

1. Consider the satisfiability problem for Boolean formulas. Show how to reduce search to decision in polynomial time.

2. Suppose you are given two sequences of nonnegative integers $a_1, a_2, \ldots, a_n$ and $b_1, b_2, \ldots, b_n$ and two targets s and t. You want to decide whether there exists $I \subseteq [n]$ such that $\Sigma_{i \in I} a_i = s$ and $\Sigma_{i \in I} b_i = t$. Design a polynomial-time reduction from this problem to the subset sum problem.

   The subset sum problem takes as input a sequence of nonnegative integers $a'_1, a'_2, \ldots, a'_{n'}$ and a target t' and determines whether there exists $I' \subseteq [n']$ such that $\Sigma_{i \in I'} a'_i = t'$.

Regular problems

3. You're an avid bicyclist who just moved to Madison. You plan to go on a long bike ride every weekend in an effort to explore the city. You've picked up a Madison cycling guide that has a list of the best bike routes. Each route includes the names of the landmarks it passes by (and each route only passes by a given landmark once). You are excited to explore Madison, so you refuse to take a weekend off of biking until you have passed by each landmark at least once. Your friend wants to visit you in w weeks, and she has planned a backpacking trip to take you on (which would preclude you from biking that weekend). She wants to know if you will be finished biking past all of the landmarks by the time she arrives. So, she has developed an algorithm to answer this question. Formally, her algorithm takes as input the number k, as well as m bike routes which together pass by n landmarks. Bike route $i \in [m]$ passes by landmarks $L_i \subseteq [n]$. Her algorithm outputs yes if there is a set of k bike routes that together pass by all n landmarks, and it outputs no otherwise.

   (a) Your friend tells you that, yes, it is possible for you to visit all n landmarks using only w of those m bike routes. However, she doesn't tell you which bike routes to use or whether you can use fewer than w routes. Your first ride is tomorrow, so you need to plan your routes now, and you want to plan the fewest routes possible to cover all landmarks. Unfortunately, your friend is camping with limited cell service, so she can't explain how her algorithm works. However, you can text her a number k' and a set of m' bike routes that together pass by n' landmarks, and she will respond whether or not it is possible to visit all n' landmarks using only k' of those m' bike routes.

   Formulate your bike route planning problem as an optimization version of your friend's decision problem. Design a polynomial time reduction from the optimization problem to the decision problem.

(b) After completing the backpacking trip with your friend, you want to get back into biking every weekend. You enjoyed the bike guidebook and will use those routes again. However, since you've already visited each landmark once, you only want to visit each landmark at most one more time. As you plan your next set of bike routes, you want to choose the maximum number of bike routes such that none of these bike routes visit the same landmark.

Formally, you are given m bike routes, which together pass by n unique landmarks. Bike route $i \in [m]$ passes by landmarks $L_i \subseteq [n]$. Your goal is to return a set of bike routes $B \subseteq [m]$ such that none of the bike routes pass by the same landmark and $|B|$ is maximized. Show how to reduce this problem to Independent Set in polynomial time.

4. You are nearing graduation and want to finalize the courses you need to take. You've already gotten all the core courses out of the way, so you only need to take T more credits out of the remaining n courses available. Each course $i \in [n]$ uses c_i credits. Also, taking more credits will increase your tuition, so you want to pick a course schedule that uses exactly T credits. Furthermore, you'd like to have time to go on adventures during your final semesters, so you want to pick a schedule with as few courses as possible. MyUW just released a new feature that given a list of courses and their credits, an integer t, and an integer m, determines if there is some subset of at most m of these courses so that the total sum of their credits is exactly t.

   (a) Design a polynomial-time algorithm that outputs a course schedule with total credit equal to T that uses as few courses as possible. You may assume you can use MyUW's new feature in constant time. In other words, you need to design a polynomial-time reduction from the problem of finding courses to the problem encoded in MyUW's new feature.

   (b) Notice that MyUW's new feature is really solving a variant of the subset sum problem. Namely, given a list of integers, $a_1, a_2, \ldots, a_n$, an integer, T, and an integer, m, determine if there is a subset $I \subseteq [n]$ with $|I| \leq m$ such that $\Sigma_{i \in I} a_i = T$. Show that this variation reduces in polynomial time to the classical subset sum problem.

   The subset sum problem takes as input a sequence of nonnegative integers $a'_1, a'_2, \ldots, a'_{n'}$ and a target t' and determines whether there exists $I' \subseteq [n']$ such that $\Sigma_{i \in I'} a'_i = t'$.

5. Consider the variant of the Traveling Salesperson optimization problem in which the tour does not need to end in the same city as it starts in. Show that this variant and the original one are equivalent under polynomial-time reductions.

   The standard Traveling Salesperson problem is as follows: Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?

Challenge problem

6. Recall scheduling to minimize the maximum lateness. Suppose we want to minimize the *sum* of the latenesses instead of the *maximum* lateness. Give a polynomial-time reduction from the partition problem to this problem.

Programming problem

7. SPOJ problem The Courier (problem code COURIER).
   Hint: Whereas the trivial algorithm for the Traveling Salesperson Problem takes time $\Theta(n!)$, there is a dynamic programming algorithm that only takes time $O(n^2 \cdot 2^n)$.