Design and Analysis of Algorithms **out:** April 2, 2015

Problem Set 7

Lecturer: Daniel A. Spielman **due: 2:30** PM, April 9, 2015

Instructions. Read Carefully

1. Make sure that your yale **netid** appears on **every** page of your problem set, and that it is written clearly.
2. The problem sets are due at the beginning of class. Solution sets will be provided in class. For this reason, late problem sets are not accepted.
3. You will **hand in each problem separately**, so make sure that they are not stapled together and that no problem appears on the back of another. There will be 4 boxes in the front of the class, one for each problem. Put each problem in the correct box.
4. Alternatively, you may submit the problem set electronically via the Classes server. If you do so, submit it as a **pdf** and make sure that your **netid** appears on **every** page. **Each problem must be in its own document.**
5. Cite any resources that you use, other than the textbook, TA and instructor.
6. You **must not** search the web for solutions to similar problems given out in other classes.

Collaboration Policy

You must write the solutions to the problems independently and in your own words.

However, you are allowed to discuss the problem sets in small groups of no more than 4 students. To ensure that you understand the solutions you submit, you are forbidden from taking written notes during your discussions. You must list at the top of the problem set everyone (other than course staff) with whom you have discussed the problem set. Failure to list people with whom you have discussed a problem set is considered a violation of academic honesty.

Problem 1: Generalized Dominoes

If you are unfamiliar with the games played with dominoes, I recommend that you quickly look at them on the web. In this problem, we will consider a generalization of dominoes that is to be played by one player on a graph.

For us, a domino is a directed edge with two endpoints: a start and an end. Each endpoint is assigned a number. The input to the problem is a directed graph $G = (V, E)$ and a list of allowed types of dominoes. The problem is to place one type of domino on each directed edge so that all of the numbers that meet at a vertex are the same. For example, if the graph has vertex set $V = \{a, b, c\}$ and directed edge set $E = \{(a, b), (b, c), (a, c)\}$, and the set of allowable dominoes is $\{(1, 1), (1, 2)\}$, then you are allowed to place dominoes on the edges like:

$$\begin{gathered}(a, b) \to (1, 1)\\ (a, c) \to (1, 2)\\ (b, c) \to (1, 2)\end{gathered}$$

In contrast, the following placement would not be allowed

$$\begin{gathered}(a, b) \to (1, 2)\\ (a, c) \to (1, 2)\\ (b, c) \to (1, 1),\end{gathered}$$

because both numbers 1 and 2 appear at node c. It is also not allowed because both 1 and 2 appear at node b.

Of course, you could just put the domino $(1, 1)$ on every edge.

The input to the Domino problem is a graph and a list of allowable dominoes. The output is "yes" if there is an allowable assignment of dominoes to every edge. Prove that the Domino problem in NP-Complete.

Problem 2: Short Lattice Vectors

Most of you are aware that almost all the cryptography we use is based on the hardness of number theoretic problems like factoring. Some of you are probably aware that there is concern that these problems might actually not be hard. If we could build a general purpose quantum computer, we could factor large numbers quickly and thereby break these crypto schemes.

Fortunately, there are other approaches to designing crypto systems that rely on very different problems. The best contenders to replace number theoretic problems are lattice problems. I will give you a taste of these.

The input to a lattice problem is a set of vectors $v_1, \ldots, v_n$, where each $v_j \in \mathbf{Z}^d$. That is, each vector is a d-dimensional vector with integer entries. In case you are wondering what d is, it is the dimension, and it is usually big and typically comparable to n. A typical lattice problem that is hard is determining whether or not there is an integer sum of these vectors that is short. That is, whether there exist integers $a_1, \ldots, a_n$ so that $\sum_i a_j v_j$ has small norm. You will prove that a special version of this problem is hard.

The input to the Short Lattice Vector Problem (SLVP) is a list of integer vectors $v_1, \ldots, v_n$, a target norm T, and an integer k. The problem is to determine whether there exist $a_1, \ldots, a_n$ in $\{0, 1\}$ so that

$$\sum_{j=1}^{n} a_j = k \quad \text{and} \quad \left\| \sum_{j=1}^{n} a_j v_j \right\|^2 \leq T, \tag{1}$$

for some target T.

Recall that if $x(i)$ is the ith coordinate of a vector x, then

$$\|x\| = \sqrt{\sum_i x(i)^2} \quad \text{and so} \quad \|x\|^2 = \sum_i x(i)^2.$$

Prove that SLVP is NP-Complete.

Hint: Here is a fact about norms that might prove useful:

$$\sum_{i=1}^{d} x(i)^2 \geq \frac{1}{d} \left(\sum_{i=1}^{d} |x(i)| \right)^2,$$

with equality if and only if all of the $x(i)$ are equal.

Problem 3: k-subset sum

Recall that in the Subset Sum problem we are given as input a list of integers $x_1, \ldots, x_n$ along with a target integer T, and must determine if there is a subset of these integers A so that

$$\sum_{i \in A} x_i = T.$$

I now define a variant of this problem that I call the k-Subset Sum problem. In this problem there are 3 inputs: the list of integers $x_1, \ldots, x_n$, a target integer T, and another integer k. The answer is "yes" if these is a set A of size k so that

$$\sum_{i \in A} x_i = T.$$

It is easy to reduce the Subset Sum problem to the k-Subset Sum problem: you just try every k between 1 and n. In this problem, I ask you to go the other way.

Give a polynomial-time reduction from the k-Subset Sum problem to the Subset Sum problem. This must be a direct reduction. You may not go through other problems, like SAT or 3DM.

Problem 4: Stable Matching Made Hard

There are many variants of the Stable Matching problem that are hard. For examples see the paper by Eytan Ronn, "NP-complete stable matching problems", *Journal of Algorithms*, 1990.

In this problem, I will ask you to examine a variant that I hope does not appear in that paper. In the Hard Stable Matching Problem (HSMP), you are given a list of n people and n companies. The problem is to find a perfect stable matching among them. But, the matching must satisfy some extra conditions. These conditions will correspond to forbidden submatchings.

A sub-matching is a list of pairs of people and companies. A matching contains such a sub-matching if each of the pairs in the sub-matching is in the matching. For example, consider a situtation in which people p_1 and p_2 hate each other, and that companies c_1 and c_3 are in the same building. We might want to forbid matching those people with those companies. Thus, we might forbid matching p_1 to c_1 and p_2 to c_3. In this case, the forbidden sub-matching is $\{(p_1, c_1), (p_2, c_3)\}$. A matching that contains both of these pairs would not be allowed. Under the justification I gave, we should also forbid $\{(p_1, c_3), (p_2, c_1)\}$.

The input to HSMP is a list of the preferences of n people and n companies, and a list of forbidden submatchings. The answer is "yes" if there is a perfect stable matching that does not contain any of the forbidden submatchings. Prove that HSMP is NP-Complete.

Hint: Look back at problem 2b of Problem Set 1.