Algorithms and Complexity
Hösten 2010
Mästarprov 2: Complexity

Mästarprov 2 should be solved individually in written form and presented orally. No collaboration is allowed.

Written solutions should be handed in latest on **Friday, December 3rd 17.00,** in my postbox or in the student reception in Osquars backe 2 plane 2. Be sure to save a copy of your solutions. Mästarprov 2 is a mandatory and rated part of the course. The test consists of four tasks. The test is roughly graded as follows: Two task correctly solved give an E. Three tasks correctly solved give a C and all tasks correctly solved give an A. You can read more about the grading criteria and the final grade on the course web page.

1. A Variant of SUBSET SUM
If you take an NP-Complete problem and slightly modify it, it might happen that the problem can be solved in polynomial time. For instance, if we take the problem INDEPENDENT SET and specialize it by asking if there is an independent set of size 3, the problem can be solved in polynomial time.
We know that the problem SUBSET SUM, i.e., given positive integers $a_1, a_2, ..., a_n, M$, decide if there is a subset sum equal to M is NP-Complete. Now let's look at a variant of the problem:

Input: Positive integers $a_1, a_2, ..., a_n$. A positive integer M. An integer d such that $1 \leq d \leq M$.
Goal: Is there a subset of $a_1, a_2, ..., a_n$ with sum in the range $[M - d, M]$.

We now might ask if this problem, we can call it INTERVAL SUBSET SUM, can be solved in polynomial time. But we can prove that it is NP-Complete by reducing the ordinary SUBSET SUM problem to INTERVAL SUBSET SUM. Your task is to show how such a reduction can be done. We give some hints:
Assume that we have an instance $a_1, a_2, ..., a_n, M$ of SUBSET SUM. We make a special instance $a'_1, a'_2, ..., a'_n, M', d$ of INTERVAL SUBSET SUM.
How should $a'_1, a'_2, ..., a'_n, M', d$ be chosen so that the instance of SUBSET SUM has a solution $\Leftrightarrow$ the instance of INTERVAL SUBSET SUM has a solution?

2. Another game We have a one person game that is played on an $m \times n$ board. On each of its mn positions lies either a blue chip, a red chip or nothing at all. You play by removing chips from the board. You win if you can remove chips so that each column contains only chips of a single color and each row contains at least one chip. Winning may or may not be possible, depending upon the initial configuration. We can now state the following problem: Given an initial configuration on an $m \times n$ board, decide if we can reach a winning position or not. Decide if there is an efficient algorithm that solves the problem.

Either construct such an algorithm or show that the problem i NP-hard by reducing some known NP-Complete problem (for instance one in the text book) to the problem.

3. Experimental Cooking
We want to try some experimental cooking. We have a list of n possible ingredients and we want to mix them. But they cannot be mixed in any crazy way, some don't go well with others. We write down an $n \times n$ matrix D called the *discord matrix*. The element D_{ij} is a real number between 0 and 1 where 1 means "i and j don't go together at all" and 0 means "i and j go together perfectly". When we now try to experiment we want to choose as many ingredients as possible, but at the same time have as little *total discord* as possible. We define total discord as the sum all discords between the chosen ingredients. We define the following problem:

EXPERIMENTAL COOKING
Input: An $n \times n$ matrix D with real numbers between 0 and 1. An integer k. A real number t.
Goal: Is there a subset S of $\{1, , ..., n\}$ of size k such that if $dsum$ is the sum of all D_{ij} such that $i < j, i \in S, j \in S$ then $dsum \leq t$?

So for instance, if

$$D = \begin{pmatrix} 0.0 & 0.4 & 0.2 & 0.9 & 1.0 \\ 0.4 & 0.0 & 0.1 & 1.0 & 0.2 \\ 0.2 & 0.1 & 0.0 & 0.8 & 0.5 \\ 0.9 & 1.0 & 0.8 & 0.0 & 0.2 \\ 1.0 & 0.2 & 0.5 & 0.2 & 0.0 \end{pmatrix}$$

and $S = \{1, 3, 5\}$ then $dsum = 0.2 + 1.0 + 0.5 = 1.7$. Show that this problem is NP-Complete. (It is possible to use at least one of the NP-Complete problems mentioned in the course.)

4. Finding Spanning Trees
You work at a company with a lot of computers arranged in a network. Some of the computers are connected and some are not. That means that the computers and their connections form a graph. Your boss wants you to mark certain edges as particularly important. The edges should form a *spanning tree* T in the graph and the set of *leaves* in T should be a given set L. (A *leaf* in a tree is a node of degree one.) Your boss asks you to design an algorithm that, given a graph G and a set L of nodes in G, finds a spanning tree in G with its set of leaves equal to L.You tell him that this problem, actually is NP-Complete so you cannot find an efficient algorithm doing it. Some days later your boss has bought an algorithm $TF(G, L)$ which solves the problem from the company *TreeFinders inc*. The algorithm is rather slow. It seems to have time complexity $T(n)$ where n is the number of nodes in the graph. There is a problem with the algorithm: It is a decision algorithm which output yes or no telling us if there is a spanning tree or not. You want to find the actual spanning tree (if there is one). Your assignment is to write an algorithm which finds a spanning tree with leaves in L if there is one. It should call the algorithm $TF(G, L)$ a polynomial number of times. More exactly, your algorithm should have time complexity $O(n^k T(n))$ for some integer k.

Solutions (Sketch)

1. The simplest would be to set $d = 0$.
But that is not allowed.
We can use this trick:
Set $a_i' = 2a_i$, $M' = 2M$ and $d = 1$.
This reduction works.

---

2. We reduce 3-SAT to our problem.

Let the columns correspond to variables and the rows to clauses. If we have a clause $x_5 \vee \bar{x}_7 \vee \bar{x}_{10}$

we put a blue chip in column 5 and red chips in columns 7 and 10 in the row corresponding to the clause.

It can be seen that a winning position corresponds to a value assignment satisfying all clauses, and vice versa.

3. Reduce INDEPENDENT SET to our problem.
Let D be the incidence matrix of the graph G and $t = 0$.

---

4. One way to do it is like this.

```
If FT(G, L) = FALSE Return ∅
G1 ← G, T ← E(G)
Foreach e ∈ T
    If FT(G1, L) = TRUE
        T ← T \ e, G1 ← G1 \ e
    Else
        G1 ← G1 \ e
Return T
```