Algorithms and Complexity
2017
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, April 21 17.00,** to Johan or Jonas in written or printed form (personally or physical mailbox). 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. The report should be written in English.

In all problems you should give an analysis of the time complexity of your algorithm and you should be able to argue for its correctness. When making reductions you are free to use any NP-Complete problems mentioned in the course book or in the lecture notes.

1. Super connectors

We have a computer network. We fix a number C and say that a computer is a super connector if it has at least C connections. (Connections are edges in the network.) A group S of super connectors is a set of super connectors all connected to each other. We can formulate a decision problem like this: Given a network in form of a graph G and integers C, K, is there a group of size K of super connectors in G? Show that this problem is NP-Complete by reducing the known problem CLIQUE to this problem.

2. Different types of matchings

The problem TRIPARTITE MATCHING can be described (in a traditional way) as this: Assume that we have a set $\{m_1, m_2, ..., m_n\}$ of n men, a set $\{w_1, w_2, ..., w_n\}$ of n women and a set $\{d_1, d_2, ..., d_n\}$ of n dogs. We assume that some three-groups of one man, one woman and one dog, like (m_3, w_9, d_2) get along well as a group and some others do not. Let us call a three-group whose members get along well a feasible three-group. (In the traditional description, they can form a household.) The problem is to decide if we can find a selection of n feasible three-group such that each man, woman and dog is in exactly one three-group in the selection. Such a selection is then called a tripartite matching. So the problem TRIPARTITE MATCHING has the sets $\{m_1, m_2, ..., m_n\}$, $\{w_1, w_2, ..., w_n\}$, $\{d_1, d_2, ..., d_n\}$ and the set of feasable three-groups as input and the problem is to decide if there is a tripartite matching or not.

This problem is known to be NP-complete. It is a variant of the ordinary problem BIPARTITE MATCHING (or just MATCHING) where we have two-groups (men and women). We can generalise the problem to four-groups, and so on.

a. Let us look at FOURPARTITE MATCHING where we have the men, women, dogs and a set $\{c, c_2, ..., c_n$ of n cats and a set of feasible four-groups. Fourpartite matchings are defined in the same way as tripartite matchings. It is easy to se that FOURPARTITE MATCHING is in NP. Show that FOURPARTITE MATCHING is NP-complete by reducing from TRIPARTITE MATCHING to FOURPARTITE MATCHING.

b. Let us assume that $P \neq NP$. The problem BIPARTITE MATCHING can be solved efficiently, which means that the problem is in P. Explain carefully why it is impossible to reduce from TRIPARTITE MATCHING to BIPARTITE MATCHING.

3. The Plus Minus Game

We will define a very special two person game which we will call "The Plus Minus Game". We have two players A and B. They each have n cards such that each card has a natural number written on it. (To be more exact, each card has one of the numbers $0, 1, 2, 3...$ and different cards can have the same number.) A and B can see eacothers cards. The players take turns and makes moves. In a move a player chooses one of his/her cards and the chooses a sign Plus or Minus. We have a so called "Bank". If the player chooses a card with number k and chooses Plus, then k is added to the bank. If the player chooses Minus, k is subtracted from the bank. The bank is 0 when the game starts. Player A makes the first move. We say that player B wins if the bank is 0 after the last move (which is made by B) and player A wins otherwise.

We can now ask the question: Does player A have a winning strategy or not. To be more specific, we assume we are given natural numbers $a_1, a_2, ..., a_n, b_1, b_2, ..., b_n$ corresponding to the players cards and want to know if A has a winning strategy or not. This is a well defined decision problem since the game runs for $2n$ moves and the set of possible cards and signs to choose from is finite. The problem is not neccesarily in NP. But we can still show that it probably is impossible to solve algoritmmically in polynomial time by showing that it is NP-Hard. This means that an NP-Complete problem can be reduced to it (which means that all NP problems can by reduced to it, by Cook's Theorem). Your task is to show that the problem is NP-Hard.

4. The intricate amusement park

A certain amusement park has n different attractions. There are connections (short roads) between the attractions such that the park can be represented as a graph G with the attractions as nodes and the connections as edges. The owner of the park is interested in finding possible ways of walking through the park along the connections such that no attractions are visited more than once. This concepts obviously is the same thing as paths in G. The owner would like to find long paths, probably for recommendation to the visitors. But how can she find long paths? It turns out that her clever daughter has designed an algorithm $PathFinder$ that takes any G and any positive integer K as input and decides if there is a path of length $\geq K$ in G or not. So the computation $PathFinder(G, K)$ returns Yes if there is such a path and No otherwise. This is nice, but what the owner would really like to have an algorithm for finding the actual path of maximum length (or at least one such path if there are more than one). But now the clever daughter, who is the only one who understands how PathFinder works, has gone on a hiking trip to Mongolia. Your task is to help the owner and construct an algorithm $LongestPathFinder(G)$ which returns a longest path in G as the set of edges in the path. The algorithm should use a limited number of calls to $PathFinder$. The number of calls and all other computations made should be as small as possible. You should give a suitable upper bound for the complexity on the form $O(p(|V|, |E|)\ T(|V|, |E|))$ where $T(|V|, |E|)$ is the maximum value of the complexity of $PathFinder(G, K)$ when $0 < K < |V|$. Of course, V and E are the set of nodes and edges of G.