Algorithms and Complexity
2019
Mästarprov1: Algorithms

Mästarprov 1 should be solved individually in written form and presented orally. No collaboration is allowed.

Written solutions should be handed in latest on **Thursday, February 21th 23.00,** on Canvas.

Mästarprov 1 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.

1. You are given an undirected graph G with n nodes. The graph is represented with adjacency lists. Is the graph a tree or not? You want to decide it algorithmically in time $O(n)$.

Observe that this can easily be done in time $O(|E|)$, but you are asked to do better than this. Also observe that you know n from the start but you don't know $|E|$ *explicitely.* (Can be computed though.)

Describe an $O(n)$-algorithm that solves the problem.

2. We will now look at a variant of the max-flow problem. Assume that we have a network of n computers connected in the graph G. One computer s is a *sender* and a set R of computers are *receivers.* Then, of course, there are other computers. To state it formally, $R \subset V(G),\ s \notin R$. We assume that all computers in R are connected to s in the network. We are now going to send information from s to all nodes in R. We will think of this information sent as a flow and the value of the flow as a transmission rate. For each node $v \in R$ we have a demand that the node should get a flow d_v. We can assume that s can send an unlimited amount of information. The communication protocol we use works so that each computer in the network processes the information a certain time before sending it forward. This means that for each node in $v \in V(G) - R - \{s\}$ there is a number α_v such that the flow trough v cannot be greater that α_v. (So observe that the *capacities* are on the nodes instead of the edges). Your task is now to construct an efficient algorithm that decides if it is possible to get a flow of this type, i.e., a flow from s to the nodes in R such that each $v \in R$ gets at least d_v. Analyse the complexity of your algorithm carefully.

3. In this problem we have a set S of n objects $a_1, a_2, ..., a_n$. The objects could be any type of objects, but we assume that they can be classified into different *kinds*. The classification can be described by a function $eq(a_i, a_j)$ that returns 1 if the objects are of the same kind and 0 otherwise. We assume that eq is an $O(1)$-algorithm (constant time). The set S has a *dominant kind* if there is a kind such that a majority of the elements are of this kind. How can we decide efficiently if S has a dominant kind or not? There are some obvious methods that have time complexity $O(n^2)$.

Your task is to find a divide-and-conquer approach that gives an algorithm that solves the problem in $O(n \log n)$ time.

4. A family plans to make a road trip. On the road they will travel, which we can think of as a line, there are cities in positions $x_1, x_2, ..., x_n$ where the x_i:s are numbers and $i < j \Rightarrow x_i < x_j$. The family wants to stay one day each in certain cities. They estimate their enjoyment of a stay in city i to p_i where p_i is a real number > 0. They must choose which cities to stay in. Their *enjoyment sum* is the sum of the p_i:s for the cities they choose. But they have one restriction. They want the distance between any two chosen cities to be at least K where K s a fixed number. Give an efficient algorithm that decides the maximum enjoyment sum the family can get.