Institut for Matematik og Datalogi
Syddansk Universitet

Exam Assignment 3
Complexity and Computability — 2020

This is the third of three sets of problems (assignments) which together with the oral exam in June constitute the exam in DM553. This first set of problems may be solved in groups of up to three. You are encouraged to work in groups to have someone to discuss this with; you can communicate online with people in your group.

The assignment is due at 23:59 on Sunday, May 17. You may write this either in Danish or English. Write your full name and email address (or names and addresses if you do it together — up to three people may work together) clearly on the first page of your assignment (on the top, if it is not a cover page). Turn it in as a PDF file via Blackboard through your DM553 course. The assignment hand-in is in the menu for the course and is called "SDU Assignment". Keep the receipt it gives you proving that you turned your assignment in on time. Blackboard will not allow you to turn in an assignment late.

Cheating on this assignment is viewed as cheating on an exam. Do not talk with anyone outside of your group (except Joan Boyar or David Hammer) about the assignment, and do not show your solutions to anyone outside your group. If you have questions about the assignment, please ask Joan Boyar or David Hammer.

Assignment 3

Do the following problems. Write clear, complete answers, but not longer than necessary.

1. Consider the following game, which we will call "Cave Tunnels". Every time you start this game, you are given a new map of a cave system with n *rooms*, some of which are connected by *tunnels*, plus a set of k tokens, which we will call *blockers*. Your goal is to *control* as many

of the tunnels as possible. You can only control a tunnel by placing a blocker in both of the rooms the tunnel connects.

(a) Prove that this problem is NP-hard, by defining a recognition version of the problem (a decision problem) which is NP-Complete, and proving that the problem you define is NP-Complete. Call the decision problem you define "Decide Cave Tunnels".

(b) Show that if there is a polynomial time algorithm for Decide Cave Tunnels, then there is a polynomial time algorithm for the evaluation version of Cave Tunnels (find the maximum number of tunnels which can be controlled with k blockers).

(c) Show that if you can find a polynomial time algorithm to solve Decide Cave Tunnels, then you can also find a polynomial time algorithm to place the blockers to control a maximum number of tunnels in the Cave Tunnels game.

(d) Suppose, in Decide Cave Tunnels, each room has at most two tunnels going off from it (incident to it). Is the recognition version of the problem still NP-Complete? Prove your answer. (You may assume that $P \neq NP$ and that it is possible to get from any room to any other using the tunnels and other rooms.)

2. Suppose you have designed a medicine which you would like to test as a cure for the corona virus. Suppose you have n people and you know $2k$ have corona virus and you want to find k of them to give your medicine to. (We assume that after finding the k, you will give them the medicine, and you can compare the results with the others, testing until you have found the k others. We'll ignore the methodalogical problems in this.) Assume that testing is a slow process and that you cannot test more than one person at a time.

(a) In this part assume that you need to test k people with positive results before you are done. In the worst case, how many people do you have to test before finding k that test positive? Give matching upper and lower bounds. An upper bound is a algorithm definition with a worst case analysis proving the result you claim. For the lower bound, use an adversary argument.

(b) Do the same as in the previous part, proving matching upper and lower bounds, but in this part assume that after you have found

$n - 2k$ that tested negative, all others will test positive, and you do not need to test them.

3. Suppose you wanted to use an information theoretic lower bound for the first part of the previous question. Assume that you are only considering algorithms that never ask about the k people with the highest identification numbers. What result would you get, doing it in the obvious way? (You do not have to approximate; it is OK if there are some factorial symbols in your answer.)