Institut for Matematik og Datalogi
Syddansk Universitet

Assignment 4
Complexity and Computability — 2016

This is your fourth assignment in DM553. **The assignment is due at noon on Thursday, May 26**. You may write this either in Danish or English. Write your full name (or names if you do it together) clearly on the first page of your assignment (on the top, if it's not a cover page). Turn it in as a PDF file via Blackboard through your DM553 course (only one per group). 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. If you have questions about the assignment, come to Joan Boyar or Christian Kudahl.

Please note that you must have this assignment approved in order to pass DM553. If it is not turned in on time, or if you do not get it approved, it will count as one of your two retries in the course, and you must have it approved on your only allowed retry for this assignment.

Assignment 4

1. On page 214 of CLRS, there is an algorithm, MINIMUM, for finding the minimum element in an array. Design an algorithm A which first uses MINIMUM to find the minimum element in the array and then finds the second smallest element. (You do not have to use MINIMUM as a black box. You may assume that you can see which comparisons are performed and what the results of those comparisons are.) Design your algorithm such that on at least one ordering of the inputs, your algorithm only uses $n-1$ comparisons in all (including the comparisons done by MINIMUM).

   (a) On which ordering does your algorithm do only $n-1$ comparisons?

(b) What is the largest number of comparisons it does? Use (a modification of) the adversary for finding the largest and second largest (to make it work for smallest and second smallest) to show which input would produce this large number of comparisons.

2. Recall that in the game Cardstone,, two players have monsters battling each other. A *state* in the game is represented by two arrays, one for you and one for your opponent. Your array is called A, your opponent's array is called B.

   Array A contains all your monsters (one in each space). Array B contains all your opponent's monsters (also one in each space). A monster is represented by an attack value and a health value separated by a slash.

   On your turn, each of your monsters may attack (no more than once each). You may decide the order in which they attack. A monster attacks one opponent's monster of your choice. When it does, they both have their health reduced by the attack value of the monster they fight. If a monster's health is reduced to 0 or below, it dies and is removed from the game.

   The Max-Board-Clear problem we consider is the following: Given a state in the game, give an assignment for each of your monsters to one of your opponents monsters for which your monsters kill the maximum number of your opponents monsters in a single turn. (Do not worry about how many of yours die.)

   (a) Since there is no proof that $P \neq NP$, we do not expect to be able to easily find an exponential lower bound on the complexity of Max-Board-Clear. Prove an $\Omega(n \log n)$ lower bound using an information theoretic argument. To do this, consider the case where both you and your opponent have n monsters and only one of yours will be needed to kill one of your opponent's. The test operation you should use is testing if a given subset (which can have size one) of your monsters can kill a specified one of your opponent's monsters.

   (b) Give a polynomial time 2-approximation algorithm for Max-Board-Clear for the special case where all of your opponent's monsters

have the same health values. Prove that it is a 2-approximation algorithm.

Hint: Consider for each of your opponent's monsters the last monster which attacked it (the one causing it to die) and how many monsters it could have killed. Then consider for all of your opponent's monsters, all of your monsters that attacked them and were not the monster that killed them and how many monsters they could have killed.

As an extra (optional) challenge, try to find a similar 2-approximation algorithm for the more general case where there are no restrictions on the health values of your opponent's monsters. Prove it.