Institut for Matematik og Datalogi
Syddansk Universitet

DM553 Assignment 4
DM508 Assignment 2

This is your last assignment in DM553 or DM508. The assignment is due at 8:15 on Tuesday, May 26. You may write this either in Danish or English. Write your full name (or names if you do it in a group of two or three) 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 or DM508 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. If you have questions about the assignment, come to Joan Boyar or Christian Nørskov.

Please note that you must have this assignment approved in order to pass DM553 or DM508. If it is not turned in on time, or if you do not get it approved, it will count as your only retry (assuming you still have one left; otherwise you will not have passed this portion of the course) in the course, and you must have it approved on your only allowed retry for this assignment.

Assignment

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

1. Prove that there is no algorithm (based on comparisons) for finding all three of the largest, second largest and third largest keys (of n keys), which only does $n + \lceil \log_2(n) \rceil - 2$ comparisons in the worst case. (Hint: We can already make the algorithm answer incorrectly if there is any key but the maximum which has not "lost" to something and make the maximum win over at least $\lceil \log_2(n) \rceil$ keys. We can also make the algorithm answer incorrectly if there is more than one of these $\lceil \log_2(n) \rceil$

keys which does not "lose" to at least two keys. Try choosing which one of these should be second largest and then which possibilities exist for third largest.)

2. Consider the following algorithm:

```
Input: An integer array $A$ of length $n$
for $i = 2$ to $n$ do
    $j = i$;
    while $j > 1$ and $A[j] < A[j-1]$ do
        temp $\leftarrow A[j]$;
        $A[j] \leftarrow A[j-1]$;
        $A[j-1] \leftarrow$ temp;
    end while
end for
return($A$)
```

   Give two proofs that this is not an algorithm for sorting an array. One proof should refer to a lower bound result you already know. The other should not (what should an adversary do?).

3. Recall that Gadget–Assembly is a game where the player is given a set of parts (via pictures) and a goal set of gadgets (also via pictures). The player should create each of the specified gadgets, one at a time from the parts, using as few moves as possible. Each gadget contains a power supply (which is one of the parts the player is given) and two other of his/her parts. Combining two parts or a part with something already assembled counts as one move. Taking one part off something already assembled or partly assembled is free if the result is something the player has previously had in that form.

   The optimzation version of Gadget–Assembly problem is, given a set of parts and a goal set of gadgets, give a set of moves to create the gadgets with the lowest possible score (number of moves).

   Give an approximation algorithm for this problem. Define $s(I)$ to be the smallest integer such that there exists a set of $s(I)$ parts (not including the power supply) such that all gadgets in the input I contain at least one part in that set. Let $|I|$ denote the number of gadgets in I. Your approximation algorithm should make at most $|I| + 2s(I)$ moves. Prove that it is correct and makes no more moves than this.