Institut for Matematik og Datalogi
Syddansk Universitet

Assignment 3 — Introduction to Computer Science 2015

This is your third assignment in DM534/DM558. The assignment is due at **8:15 on Thursday, November 5**. You may write this either in Danish or English. It must be made in LaTeX. Write your full name, your section number (D1, D2, or D3), and your "instruktor"s name (Kristine Vitting Klinkby Knudsen, Mathias W. Svendsen, or Jesper With Mikkelsen) clearly on the first page of your assignment (on the top, if it's not a cover page). You should turn it in as a PDF file via Blackboard through your DM534/DM558 course. The assignment hand-in is in the menu for the course and is called "SDU Assignment". Choose the correct one for your section number, D1, D2 or D3. 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. You are allowed to talk about course material with your fellow students, but working together on this assignment is cheating. If you have questions about the assignment, come to Joan Boyar or your "instruktor" for DM534/DM558.

Please note that you must have this assignment approved in order to pass DM534/DM558. 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 single allowed retry for this assignment. Note that you have only two retries in total for the assignments in DM534.

Assignment 3

Do either the first problem or the second more challening problem. Write your solutions in LaTeX. Do not include the statements of the problems or other information not asked for in the problems, but make your answers clear and precise.

1. Consider a course with a large number of students and a project which is graded with results which are non-negative integer scores (but with varying maximum scores from year to year, so do not consider the maximum possible as known). In the professor's records for this course,

he/she wants to record for each student whether or not that student got the highest (or tied for the highest) score on the project. The professor creates an additional list for this with Boolean values `True` and `False`, where `True` in location i means that ith student got the highest score and `False` means that the score was not highest. Suppose the scores are in a list called `score` and the additional list is called `highest`. Let `score`$[i]$ denote the ith entry of `score` and `highest`$[j]$ denote the jth entry of `highest`.

(a) Consider the following algorithm for solving the professor's problem:

```
procedure Rate(score, highest):
{ Input: The list score contains non-negative integers }
{ Output: Boolean list, highest, with True meaning that the
            corresponding value in the list score was highest }

    N := 1
    while (N ≤ length(score))
    begin
        highest[N] := True
        j := 1
        while (j ≤ length(score) and score[j] ≤ score[N])
            j := j + 1
        if j ≤ length(score)
            then highest[N] := False
        N := N + 1
    end
```

Use as the fundamental operation the comparison of entries in `score`.

i. Give a list of length 4 where this algorithm performs 7 fundamental operations.

ii. In general (for arbitrary list length), on which lists does this algorithm perform the least number of fundamental operations?

iii. Suppose that `score` has n entries. Express the best case running time (the minimum number of fundamental operations as a function of n) of this algorithm using Θ notation. Explain your answer.

iv. Give a list of length 4 where this algorithm performs 16 fundamental operations.

v. In general (for arbitrary list length), on which lists does this algorithm perform the greatest number of fundamental operations?

vi. Suppose that `score` has n entries. Express the worst case running time of this algorithm (the maximum number of fundamental operations as a function of n) using Θ notation. Explain your answer. Your explanation should include both why the algorithm cannot take more time than this and why it in some cases takes this much time.

(b) Someone interested in the running time of algorithms would not have written the above algorithm. It is not $\Theta(n)$ and it is possible to solve this problem in $\Theta(n)$.

i. Write an algorithm to solve this problem in $\Theta(n)$. Use the same notation as above.

ii. Analyze your algorithm (explain why it does $\Theta(n)$ fundamental operations in the worst case).

(c) Include your LaTeX code for this assignment at the end.

2. Suppose you have a list `Customers` and a function `where` which tells where the customer is from by calling `where(Customers`$[i]$`)` for customer i. Suppose further that there are only three possibilities, `Odense`, `Fyn`, and `Other`, where `Fyn` means on Fyn, but not in Odense Kommune, and `Other` means not on Fyn. Suppose the owner would like a partially sorted list of these customers with all customers from Odense first, all other customers from Fyn next, and the remaining customers last. Suppose the number of customers is n, and let the fundamental operation being calls to the function `where`.

(a) Write a $\Theta(n)$ algorithm to solve this problem. Do not use an extra list; just move entries around in the original list. Argue that the algorithm is $\Theta(n)$.

Hint: Consider the partition procedure in QuickSort. This idea works if there are only two locations: Keep a left index and a right index into the list, and a loop invariant saying that all entries to the left of the left index are of type 1 and all entries to the right of the right index are of type 2. Extend this idea to three types.

(b) Include your LaTeX code for this assignment at the end.