

Introduction to algorithms and datastructures

Solutions

IT University of Copenhagen

June 13, 2001

This exam consists of 3 exercises containing in total 13 subexercises. Each of these 13 subexercises are given the same weight in the evaluation. You have 4 hours to complete the exam. Remember to number the pages and write your name and CPR number on every page. The exam consists of 7 numbered pages.

CLR refers to “Introduction to Algorithms” by Cormen, Leiserson and Rivest, 18. press, 1997.

For exercises in which algorithms must be specified, the asymptotic time complexity of the specified solution will be taken into account when grading. Exercises asking for time complexity must be answered using O-notation. It is weighted in the evaluation that growth rates in the O-notation are expressed with least possible asymptotic growth.

Exercise 1

This exercise is about O -notation.

a) What is $O(\log n) + O(1) + O(n)$?

Let there be given a heap, which supports the following operations BUILD-HEAP, EXTRACT-MIN and DECREASE-KEY.

b) Assume that the heap above supports EXTRACT-MIN in $O(\log \log n)$ time, DECREASE-KEY in $O(1)$ time and BUILD-HEAP in linear time. Give the time complexity for Dijkstra's algorithm in a graph with m edges and n nodes if the above heap is used.

Look at the following method:

$F1(n)$

1 **if** $n > 2$

2 **then return** $(F1(n - 1) + F1(n - 2)) \bmod n$

3 **else return** 2

c) Explain how $F1$ can be implemented with a running time of $O(n)$.

Look at the following three methods:

F2(n)

```
1 if  $n > 0$ 
2   then F2( $\lfloor n/2 \rfloor$ )
3     F2( $\lfloor n/2 \rfloor$ )
```

F3(n)

```
1 if  $n > 0$ 
2   then for  $i \leftarrow 1$  to  $n$ 
3     do  $j \leftarrow 1$ 
4       F3( $\lfloor n/2 \rfloor$ )
5       F3( $\lfloor n/2 \rfloor$ )
```

F4(n)

```
1 if  $n > 0$ 
2   then for  $i \leftarrow 1$  to  $n$ 
3     do  $j \leftarrow 1$ 
4       F4( $\lfloor n/2 \rfloor$ )
```

d) Assume that the procedures are only called with an integer n . For each of the four procedures above give the time complexity in n .

Assignment 2

This assignment is about heaps and amortized analysis. A heap is defined as a data structure, which supports the operations `EXTRACT-MAX` and `INSERT`.

`INSERT(A, k)` : Inserts the value k in the heap A .

`EXTRACT-MAX(A)` : Removes and returns a maximum value from the heap A .

Let these two operations be implemented as described in CLR chapter 7.

a) Illustrate the execution of `EXTRACT-MAX(A)` for the heap A in figure 1. Use the style from figure e7.5 in CLR page 151. Remark that in this figure insertion is shown. You have to show extraction.

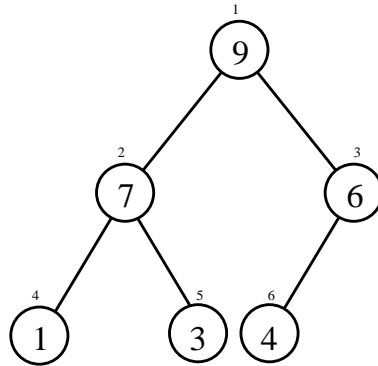


Figure 1: The heap A , which `EXTRACT-MAX` are to be executed at.

Now the heap must support another operation:

`REMOVELARGEST(A, k)` : Removes the k largest elements from A . If A has less than k elements, A are emptied completely.

b) Assume that the heap can contain up to n elements. Describe how `REMOVELARGEST(A, k)` can be supported with a worst case running time of $O(k \log n)$.

The heap must now support another operation:

`EXTRACTMORETHAN(A, x)` : Removes and returns all values from the heap A , which are larger than x .

c) Assume that the heap can contain up to n values. Describe how the operations `INSERT`, `EXTRACT-MAX` and `EXTRACTMORETHAN` can be implemented in such a way, that the heap operations has an amortized running time of $O(\log n)$. That is m heap operations are done in time $O(m \log n)$

The heap must now support another operation:

EXTRACT-MIN(A) : Removes and returns a minimal value from the heap A .

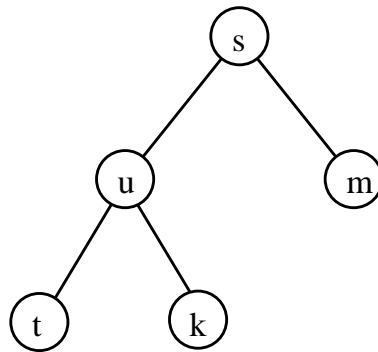
d) Assume that the heap can contain up to n values. Describe how the operations **INSERT**, **EXTRACT-MAX** and **EXTRACT-MIN** can be implemented in such a way that the heap operations has a worst case time complexity of $O(\log n)$ per operation. This means, that each of the operations must be supported in time $O(\log n)$.

Exercise 3

This exercise is about recursion and rooted binary trees. Each node has either two or no children. The node x 's left child is $left[x]$, and its right child is $right[x]$. If the node x doesn't have any children, $left[x]$ and $right[x]$ has the special value NIL and x is called a leaf. If x has any children it is called an internal node. In case the root node for a tree are NIL, the tree is empty. For each node in the tree a color are assigned; the node x has color $color[x]$.

Look at the following procedure:

```
PRINT( $x$ )
1  if  $x \neq$  NIL
2    then print  $color[x]$ .
3      if  $left[x] \neq$  NIL
4        then PRINT( $left[x]$ )
5          PRINT( $right[x]$ )
```



Figur 2: Tree With nodes, that has colors denoted by letters.

Let x be the root node for the tree in figure 2. A call to the method PRINT(x) will print the color sequence “sutkm”.

a) Change the method PRINT, in such a way that the recursive run-through by the call to PRINT(x) for the root node x in the tree from figure 2 prints the color sequence “smukt’ in stead.

With the definition of binary trees we use in this exercise, we can calculate the number of nodes and leafs in a tree using the following two procedures:

NODES(x)

```
1 if  $x = \text{NIL}$ 
2   then return 0
3   else return 1+NODES( $left[x]$ ) + NODES( $right[x]$ )
```

LEAVES(x)

```
1 if  $x = \text{NIL}$ 
2   then return 0
3   else if  $left[x] = \text{NIL}$ 
4     then return 1
5     else return LEAVES( $left[x]$ ) +LEAVES( $right[x]$ )
```

b) An internal node in a tree is a node which aren't a leaf. Construct a method INTERNAL(x), which given a root node x for a tree, returns the number of internal nodes in the tree. Give the time complexity of your solution.

In the following three subexercises, effective recursive methods are to be constructed.

c) A tree has a red root path, if there are a path in the tree from the root x to a leaf, where all nodes on the path from x to the leaf has the color red. Construct a method REDROOTPATH(x), which returns the number 1 if the tree with root x has a red root path. If such a path doesn't exist, the method must return 0. Give the time complexity of your solution.

d) A tree has a red leaf path, if there are a path between two different leafs in the tree, where all the nodes on this path has the color red. Construct a procedure REDLEAFPATH(x), which returns the number 1, if the tree with root x has a red leaf path. If such a path doesn't exist, the method shall return a number different from 1. Give the time complexity of your solution.

e) A tree is said to be complete if all leafs has the same depth. Construct a method COMPLETE(x), which given a root node x for a tree returns -1 if the tree is not complete and otherwise the height of the tree. Give the time complexity of your solution.