Universiteit Twente
Afdeling Informatica, Faculteit EWI
donderdag 16 maart 2017

Semester 2016-2017
Tentamen

Answers

MOD7:ADS

1

10 pt

Consider the following algorithm (with * for multiplication, // for integer division (eg. 7//2 = 3), and **2 for square):

```
def func(n):
    if n==0:
        return 1
    else:
        if n<4:
            return n
        else:
            return 2*func(n//4) + 6 + func(n//4)**2
```

1. Give a recursive expression for the time complexity of this algorithm, expressed in the number of arithmetical operations.

   Answer:

   Note that `func(n div 4)` is called twice. Furthermore, 6 aritmetical operations are used (a multiplication, two additions, two divisions, and a square), so the recurrence relation becomes $T(n) = 2 \cdot T(\lfloor n/4 \rfloor) + 6$.

2. What is the complexity class of this algorithm?

   Answer:

   Assume n is a power of 4 (just for convenience) so $T(n) = 2 \cdot T(n/4) + 6$. We apply the Master theorem with $b = 2$ and $c = 4$, so $E = log2/log4 = 1/2$. Now $6 \in O(n^{1/2-\epsilon})$ for some ϵ, so we have case 1, so $T(n) \in \Theta(n^{1/2})$.

2a

5 pt

Suppose in a heap you update an arbitrary element (say with index i). Describe (in words or in pseudocode) an algorithm that repairs (if necessary) the heap property.

Answer:

Suppose the heap is given by array E. There are three possibilities:

1. E is still a heap; then you are ready.
2. The new value k of $E[i]$ is bigger than its parent. Then swap $E[i]$ with its parent. Repeat this until k is in a position where it is smaller than its parent (or it is the root); now you have again a heap/
3. The new value k of $E[i]$ is smaller that its parent, but also bigger than one of its children. Now call *Heapify(E,i)*.

2b 5 pt

Given a binary search tree with positive keys, and a key k that does not occur in the tree. Give a function that yields: the biggest key in the tree, smaller than k (or zero if there is no such key). Hint: traverse the tree as if you want to insert k, and keep track of what you encounter.

Answer:

```
pred(T,k):

x = T.root
max = 0

while x != null:
    if k < x.key:
        x = x.left
    else:
        max = x.key
        x = x.right
return max
```

3 10 pt

Suppose you want to put songs on a cd. Suppose you can choose from n songs, where song i takes t_i minutes. You want to fill the cd as much as possible, which means that you want to put as much minutes of music on it as possible. Assume a cd may contain at most 80 minutes of music.

1. suppose $C(i,k)$ indicates the minimal remainder (so the amount of unused minutes) if still k minutes need to be filled with songs chosen from the set $\{1,\ldots,i\}$. Explain that
$$C(i,k) = min\{C(i-1,k), C(i-1,k-t_i)\}$$

Answer:

Either you do not choose song i, and then the remainder is $C(i-1,k)$, or you do choose song i, and then the remainder is $C(i-1,k-t_i)$. Now take the minimum of these two options.

2. Give a polynomial algorithm, based on dynamic programming, that calculates the maximal amount of minutes you can put on the cd.

Answer:

```
int cdchoice1(int[] t, int n)
{ for (k=0;k<=80;k++) C[0,k]=k;

  for (i=1;i<=n;i++)
   for (k=0;k<=80;k++)
    if (k-t[i]<0) then C[i,k] = C[i-1,k]
                  else C[i,k] = min(C[i-1,k], C[i-1, k-t[i])
  return 80-C[n,80]
}
```