Exercises week 4

Complexity 2011-2012

A. Silva, H. Barendregt, B. Westerbaan & B. Westerbaan

Exercises marked with (†) are harder exercises. Exercises marked with a **(*)** can be handed in, we will correct them and give them back to you the week after. This week the answers should be handed in before **March 6 at 23h59 (CET time)**.

Handing in your answers: There are two options: e-mail to `alexandra@cs.ru.nl` or put your solutions in the post box of Alexandra (more detailed instructions are in the first week exercise sheet: `http://alexandrasilva.org/files/teaching/complexity2012/ex1.pdf`).

Exercise 1. Consider the following recurrence relation:

$$T(n) = \begin{cases} c_1 & n \leq 1 \\ c_2 + 5T(\lfloor n/3 \rfloor) & n > 1 \end{cases}$$

Find the solution, in terms of O notation, for this recurrence (justify your answer).

Exercise 2. (*) Consider the following algorithm which computes Fibonacci numbers

```
int fib (int n)  {
  if (n==0 || n==1) return 1;
  else return fib(n-1) + fib(n-2);
}
```

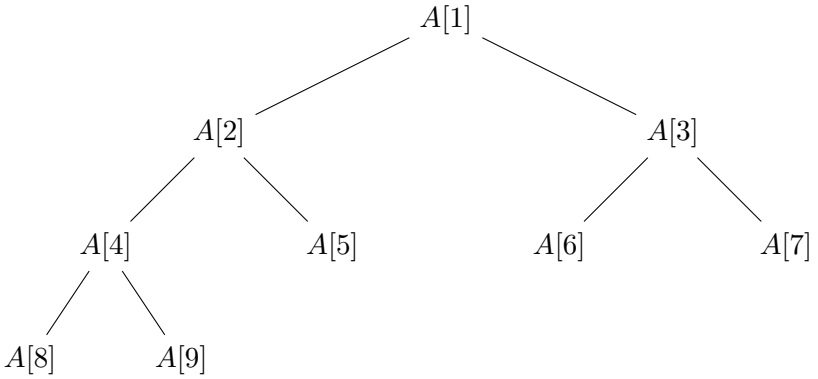
Write a recurrence relation for this algorithm. Draw the recursion tree and obtain a complexity bound from the tree. Prove your complexity bound using the substitution method.

Exercise 3. Consider the following function:

```
void example(int A[], int N) {
 /* N is the size of the array */
    int i;
    Node p;
    for (i = 1; i <= N; i++)
        p = insert(A,N,i,p);
    convert(p,A,1);
  }
```

Assuming $T_{\texttt{insert}} = O(\lg N)$ and $T_{\texttt{convert}} = O(N^2)$, analyze the execution time of the function `example` in the worst case.

Exercise 4. (*) A *complete binary tree* is a binary tree of which all levels are completely filled, except possibly the last, on which all nodes are packed to the left. Complete binary trees can be represented as arrays in a natural way: enumerate the nodes line by line from left to right. A 9 element array A represents the following complete binary tree.



A *heap* is an array, which interpreted as complete binary tree has the following property: the value of every node is less than or equal to the values of its childnodes.

(i) Given the following arrays

$A_1 = [1,2,3,4,5,6,7,8]$ $A_2 = [1,2,5,1,3,2,1,8]$ $A_3 = [3,5,7,9,6,9]$ $A_4 = [9,8,7,6,5,4,3,2,1]$

(a) Draw the arrays A_1, A_2, A_3 and A_4 interpreted as complete binary trees.

(b) Which ones of A_1, A_2, A_3 and A_4 are heaps?

(ii) One application of heaps is a simple sorting algorithm.

```
void HeapSort(int A[], int N) {
 /* N is the size of the array */
 int B[N];
 for (i=1, i< N, i++)
    B[i] = A[i];  /*copy A to B */
 heapify(B);
 for (i=1, i< N, i++)
    A[i] = popmin(B);
}
```

`heapify` reorders an array such that it becomes a heap; `popmin` returns and removes the least element from the heap. Assuming that `heapify` and `popmin` run worst-case in time of order, respectively, $O(n)$ and $O(\lg n)$, prove that HeapSort runs worst-case in time of order $O(n \log n)$.

Exercise 5. (†) Show how to sort n integers in the range 0 to $n^3 - 1$ in $O(n)$ time.

Exercise 6. (†) Let $T\colon \mathbb{N} \to \mathbb{N}$ be monotonic increasing. Show the following implications; the assumption is that the inequalities hold for all even numbers n.

$$\begin{array}{lrcll} (1) & T(n) & \leq & T(n/2) + 12 & \Rightarrow \; T \in O(\lg n). \\ (2) & T(n) & \leq & 2T(n/2) + 7 & \Rightarrow \; T \in O(n). \\ (3) & T(n) & \leq & 2T(n/2) + 8n & \Rightarrow \; T \in O(n \lg n). \end{array}$$