Exercises week 5

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 13 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 propositions

$$(p \to q) \to (q \to p), \quad (p \to q) \vee (q \to p), \quad ((p \to q) \to p) \to p, \quad \neg((p \to q) \vee p) \vee (\neg(p \to q) \wedge q).$$

Give the truth tables for the three propositions and conclude whether they are valid, satisfiable or neither.

Exercise 2. Given that for every computable f there exists G_f as in the slides then show that there exists G'_f such that $G'_f \in \Omega(f)$. Here, by $G'_f \in \Omega(f)$ we mean that for every algorithm p of G'_f, $T_p \in \Omega(f)$.

Exercise 3. (†)

1. Given f, g such that $f < g$, find h which satisfies $f < h < g$ ($f < g$ means $f \in O(g)$ and $g \notin O(f)$).
2. Find a function f such that $g \in O(f)$ for every computable g [Hint: f is not computable].

Exercise 4. (†) **Longest common subsequence**

1. Give an algorithm that finds *the length* of the longest common subsequence of two sequences such that the worst-case space complexity is at most of order $O(\min\{n, m\})$ and the worst-case time complexity is at most $O(n \cdot m)$, where n is the length of the first input sequence and m of the second. Prove these upper bounds.
2. Give an algorithm that finds a longest common subsequence in the same bounds.
3. Sketch how to find the difference between two textfiles using the LCS-algorithm. That is, given an *original* and *changed* textfile, find the lines that are inserted and deleted. (Like `diff` or the history on Wikipedia.)

Exercise 5. (*) Recall last week's exercise about heaps.

Given the index i of a node, it is easy to find the indices of the parent node $P(i)$, the left child node $L(i)$ and the right child node $R(i)$.

(i) Give L, R and P using multiplication, division, addition and/or floor.

(ii) The merit of heaps is that some useful operations related to them are easy and fast. This week, we will explicitly study the functions `heapify` and `popmin`, which we included in last week's assignment without showing the algorithms.

```
int popmin(int A[], int N) {
 /* N is the size of the array */
 int r = A[0];
 A[0] = A[N-1];
 fix(A,N-1,0);
 return r;
}

void heapify(int A[], int N) {
 /* N is the size of the array */
 for(i=N-1, i>=0,i--)
     fix(A,N,i);
}

void fix(int A[], int N, int i)  {
 /* N is the size of the array */
 int j = i;
 if (R(i) < N) {
    if (A[i] >= A[L(i)] ||  A[i] >= A[R(i)])
       if (A[L(i)] >= A[R(i)])
              j = R(i);
       else   j = L(i);
 }
 else {
  if (L(i) < N && A[i] >= A[L(i)])
     j = L(i);
 }

 if (j != i){
  swap(A, i,j);
  fix(A,N, j);
 }
}
```

`heapify` reorders an array such that it becomes a heap; `popmin` returns and removes the least element from the heap. Consider the following arrays

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

(a) Apply `popmin` to A_1, A_2, A_3 and A_4. Give the return values and the changed arrays.

(b) Apply `heapify` to A_1, A_2, A_3 and A_4. Give the changed arrays.

(c) Describe what `fix` does.

(d) Prove that `popmin` runs worst-case in time of order $O(\log n)$.

(e) Prove that `heapify` runs worst-case in time of order $O(n)$. (Hint: If $S = \sum_{i=0}^{n} i2^i$, look at $2S - S$ to derive a closed formula for S.)

(f) Prove that `heapify` runs worst-case in time of order $\Omega(n)$.