Exercises week 2

Complexity 2011-2012

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

These are the exercises that will be discussed in week 2. The exercise marked with (†) is a hard exercise. 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 **Tuesday February 14, 15h30**.

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. Assume you have functions f and g such that $f(n)$ is in $O(g(n))$. For each of the following statements, decide whether you think it is true or false and give a proof or a counter-example.

1. $\log_2 f(n)$ is $O(\log_2 g(n))$
2. $2^{f(n)}$ is $O(2^{g(n)})$
3. $f(n)^2$ is $O(g(n)^2)$

Exercise 2. Given a sequence v of n integers and an integer x, we want to obtain the index of the first occurrence of x in v, and -1 in case x does not occur in v. The function `search` below solves this problem.

```
int search(int v[], int n, int x) {
       int i = 0, found = 0;
       while (i < n && !found) {
        if (v[i] == x) found = 1;
        i++;
       }
       if (!found) return -1;
       else return i-1;
 }
```

(i) What will be the number of operations performed in the worst and best cases.

(ii) If the sequence v is ordered, then it is possible to improve the algorithm. Explain how and determine the complexity of the new algorithm.

Exercise 3. (*) Consider the `Bubble sort` algorithm, where N is the size of the input vector:

```
void bubble_sort(int A[], int N) {
  for (i=0 ; i<N ; i++)
    for(j=N-1 ; j>i ; j--)
      if (A[j] < A[j-1])
         swap(A,j,j-1);
}


void swap(int A[], int i, int j) {
  int aux=A[i];
  A[i]=A[j];
  A[j]=aux;
}
```

(i) Study the behavior of the algorithm above in the best and worst case scenarios using the notation Θ.

(ii) How would you characterize the global behavior of this algorithm using the notation O and Ω? What is the relation with the answer you gave in (i)?

(iii) Imagine the algorithm checks, in the outermost cycle, whether `A` is sorted. Repeat (i) under this assumption.

(iv) Analyze the following alternative version of the algorithm

```
void bubble_sort(int A[], int N) {
      for (i=0 ; i<N ; i++)
         for(j=N-1 ; j>i ; j--)
            if (A[j] < A[i])
             swap(A,j,i);
}
```

Exercise 4. (†)

Given three arrays of n (floating point) real numbers (the numbers can be positive or negative), write an algorithm to determine if there are three numbers, one from each array whose sum is 0. Design the most efficient algorithm you can think of to solve this problem. (It can be done in $\Theta(n^2)$ time.)

Hint: Given two sorted lists determine an algorithm, in $\Theta(N)$, to check if they have an element in common.