University of Freiburg
Dept. of Computer Science
Prof. Dr. F. Kuhn
P. Bamberger, P. Schneider

Algorithms and Data Structures
Summer Term 2019
Sample Solution Exercise Sheet 1

Exercise 1: Bubblesort

The following pseudocode describes the BUBBLESORT algorithm which takes as input an Array A of length n.

Algorithm 1 BUBBLESORT($A[0, \ldots, n-1]$)

```
for i = 0 to n − 2 do
    for j = 0 to n − 2 do
        if A[j] > A[j + 1] then
            swap(A[j], A[j + 1])
```

(a) Assume BUBBLESORT runs on input $A = [27, 8, 19, 5, 23, 12]$. Give A after the end of each iteration of the outer for-loop.

(b) Give an upper and a lower bound for the (worst-case) runtime of BUBBLESORT as a function of n. Explain your answer.

Sample Solution

(a) $[27, 8, 19, 5, 23, 12]$
$[8, 19, 5, 23, 12, 27]$
$[8, 5, 19, 12, 23, 27]$
$[5, 8, 12, 19, 23, 27]$

(b) The phrases "upper bound" and "lower bound" are maybe a little misguiding here. The part "**if** $A[j] > A[j+1]$ **then** swap$(A[j], A[j+1])$" takes constant time and is executed $(n-2)^2$ times. So the algorithm takes quadratic time in any case. By saying that this is an "upper bound" we actually mean that the algorithm is correct, i.e., any input array is sorted after the $(n-1)^2$ iterations. To see this, we observe that after the first iteration of the outer for-loop, the largest element has been moved to the end, after the second iteration, the second largest has been moved to the end and so on. So after $n-1$ iterations, the array is ordered.

On the other hand, for an array A of length n with $A[i] > A[i+1]$, we need $\sum_{i=1}^{n-1} i = \frac{n^2-n}{2} \geq c \cdot n^2$ swap operations until A is sorted. So we say that the lower bound is also quadratic, i.e., we cannot stop the algorithm after $o(n^2)$ steps and have the guarantee that the input array is ordered.

Exercise 2: Insertion Sort

The following pseudocode describes the InsertionSort algorithm which takes as input an Array A of length n.

Algorithm 2 InsertionSort($A[0, \ldots, n-1]$)

```
for i = 0 to n − 2 do
    pos = i + 1
    while pos > 0 and A[pos] < A[pos − 1] do
        swap(A[pos], A[pos − 1])
        pos = pos−1
```

(a) Assume InsertionSort runs on input $A = [27, 8, 19, 5, 23, 12]$. Give A after the end of each iteration of the for-loop.

(b) Give an upper and a lower bound for the (worst-case) runtime of InsertionSort as a function of n. Explain your answer.

Sample Solution

(a) $[27, 8, 19, 5, 23, 12]$

$[8, 27, 19, 5, 23, 12]$

$[8, 19, 27, 5, 23, 12]$

$[5, 8, 19, 27, 23, 12]$

$[5, 8, 19, 23, 27, 12]$

$[5, 8, 12, 19, 23, 27]$

(b) We first argue that the algorithm is correct. One can show inductively that after the ith for-loop, the array $A[0, \ldots, i]$ is sorted. It follows that after $n-1$ iterations, A is sorted. The algorithm executes at most $\sum_{i=1}^{n-1} i = \frac{n^2-n}{2} \leq c \cdot n^2$ swap operations, so we have a quadratic upper bound.

On the other hand, for an array A of length n with $A[i] > A[i+1]$, we need in fact $\sum_{i=1}^{n-1} i = \frac{n^2-n}{2} \geq c' \cdot n^2$ swap operations until A is sorted, so we also have a quadratic lower bound.