Theoretical Computer Science – Exercise 12

SS 2022
Jochen Schmidt

Technische Hochschule Rosenheim

Please prepare the following exercises at home prior to the tutorial:

Exercise 1

Which of the following statements are true?

$3n + 8n^2 + 100n^3 = O(n^2 \log n)$	$3n + 8n^2 + 100n^3 = \Theta(n^2 \log n)$	$3n + 8n^2 + 100n^3 = \Omega(n^2 \log n)$
$3n + 8n^2 + 100n^3 = O(n^3)$	$3n + 8n^2 + 100n^3 = \Theta(n^3)$	$3n + 8n^2 + 100n^3 = \Omega(n^3)$
$3n + 8n^2 + 100n^3 = O(n^4)$	$3n + 8n^2 + 100n^3 = \Theta(n^4)$	$3n + 8n^2 + 100n^3 = \Omega(n^4)$

Exercise 2

Given: An unsorted array containing natural numbers.
Objective: Find the maximum and minimum number simultaneously.

a) Specify an algorithm based on the divide-and-conquer principle that halves the number of elements in each step. What is its time complexity?
b) How does the algorithm and its time complexity change when four partitions are used instead of two?
c) Show how both variants perform using the following array: 30, 7, 6, 11, 4, 19, 5, 14, 10, 8

We will do the following exercises together during the tutorial:

Exercise 3

The following C code implements a function for sorting an integer array a with n entries:

```
void sort (int *a, int n) {
    int i, t, s = 1;
    while (s != 0) {
        s = 0;
        for (i = 1; i < n; i++) {
            if (a[i] < a[i - 1]) {
                t = a[i];
                a[i] = a[i - 1];
                a[i - 1] = t;
                s = 1;
            }
        }
    }
}
```

Specify the time complexity in O-notation for the worst and the best case (meaning: the behavior of the code depends on how the data in the array is pre-sorted). Justify your result!

Exercise 4

The following function recursively calculates the arithmetic mean of all numbers stored in array `a`. The `start` and `end` parameters specify the indices of the array part to be considered. For instance, if you have 10 numbers stored in an array `arr`, then the function is called as follows (`m` then contains the mean value):

```
float m = average(arr, 0, 9);
```

Determine the time complexity of the function in O-notation.

```
float average(float a[], int start, int end) {
      int n = end - start + 1;
      if(n == 1) return a[start];
      else {
            int center = n / 2 - 1;
            float m1 = average(a, start, start + center);
            float m2 = average(a, start + center + 1, end);
            float m  = (m1 * (center + 1) +
                        m2 * (end - (start + center + 1) + 1)) / n;
            return m;
      }
}
```