Übung 9

Algorithmen und Programmierung

Jonas Henschel, Jens Pönisch, Dominik Gorgosch, Bastian Felix Bachmann, Arvid Horn, Billy Naumann

Aufgabe 1

Beschreiben Sie kurz einen Algorithmus für die Suche des Maximums in einem Feld. Bestimmen Sie seine Komplexität im Worst-Case in Abhängigkeit von der Eingabegröße. Setzen Sie für Vergleiche, Zuweisungen, Rechenoperationen und Arrayzugriffe jeweils die Kosten von 1 an.

Aufgabe 2

Bestimmen Sie die Komplexität des folgenden Algorithmus im Worst-Case. Was ist sein Ziel? Setzen Sie für Vergleiche, Zuweisungen, Rechenoperationen, Arrayzugriffe, return-Statements und Funktionsaufrufe jeweils die Kosten von 1 an.

Ausgelagerte Berechnung. Bestimmen Sie die Komplexität für den Worst Case für die Funktion `allIncluded()`.

```
1  int func1(char *string){
2    int i=0;
3    while (string[i]!='\0') i++;
4    return i;
5  }
6
7  void func2(int arg, char *string){
8    for(int i=0; i < arg; i++) string[i]=string[i]+12;
9  }
10
11 void allIncluded(char *string1, char *string2){
12   int a = func1(string1);
13   for (int i=0; i < a; i++) string2[i]=string1[i];
14   string2[a]='\0';
15   func2(a, string2);
16 }
```

Aufgabe 3

Bestimmen Sie die Komplexität des folgenden Algorithmus im Best-Case und im Worst-Case. Was ist sein Ziel? Setzen Sie für Vergleiche, Zuweisungen, Rechenoperationen und Arrayzugriffe jeweils die Kosten von 1 an.

```
1  void algorithmus(int data[], int length) {
2    int i, k, t, m;
3
4    for( i = 0; i < length-1; i++) {
5      m = i;
6      for( k = i+1; k < length; k++) {
7        if( data[k] < data[m])
8          m = k;
9      }
10     t = data[m];
11     data[m] = data[i];
12     data[i] = t;
13   }
14 }
```

Aufgabe 4

Bestimmen Sie die Komplexität des folgenden Algorithmus im Best-Case und im Worst-Case. Sie können davon ausgehen, dass das Array `data` aufsteigend sortiert ist. Setzen Sie für Vergleiche, Zuweisungen, Rechenoperationen und Arrayzugriffe die Kosten von 1 an.

```
1  int algo(int data[], int len, int x) {
2    int left = 0;
3    int right = len - 1;
4    while (left <= right) {
5      int middle = left + ((right - left) / 2);
6      if (data[middle] == x) return middle;
7      else if (x < data[middle]) right = middle - 1;
8      else left = middle + 1;
9    }
10   return -1;
11 }
```