

## Aufgabe 4

Gegeben seien die Standardstrukturen Stapel (Stack) und Schlange (Queue) mit folgenden Standardoperationen:

Stapel	Schlange
<code>boolean isEmpty()</code>	<code>boolean isEmpty()</code>
<code>void push(int e)</code>	<code>enqueue(int e)</code>
<code>int pop()</code>	<code>int dequeue()</code>
<code>int top()</code>	<code>int head()</code>

Beim Stapel gibt die Operation `top()` das gleiche Element wie `pop()` zurück, bei der Schlange gibt `head()` das gleiche Element wie `dequeue()` zurück. Im Unterschied zu `pop()`, beziehungsweise `dequeue()`, wird das Element bei `top()` und `head()` nicht aus der Datenstruktur entfernt.

- (a) Geben Sie in Pseudocode einen Algorithmus `sort(Stack s)` an, der als Eingabe einen Stapel `s` mit `n` Zahlen erhält und die Zahlen in `s` sortiert. (Sie dürfen die Zahlen wahlweise entweder aufsteigend oder absteigend sortieren.) Verwenden Sie als Hilfsdatenstruktur ausschließlich eine Schlange `q`. Sie erhalten volle Punktzahl, wenn Sie außer `s` und `q` keine weiteren Variablen benutzen. Sie dürfen annehmen, dass alle Zahlen in `s` verschieden sind.

```
1  q := neue Schlange
2  while s not empty:
3      q.enqueue(s.pop())
4  while q not empty:
5      while s not empty and s.top() < q.head():
6          q.enqueue(s.pop())
7      s.push(q.dequeue())
```

### Als Java-Code

```
5  /**
6   * So ähnlich wie der <a href=
7   * "https://www.geeksforgeeks.org/sort-stack-using-temporary-
8   ↪ stack/">Stapel-Sortiert-Algorithmus
9   * der nur Stapel verwendet</a>, nur mit einer Warteschlange.
10  *
11  * @param s Der Stapel, der sortiert wird.
12  */
13  public static void sort(Stack s) {
14      Schlange q = new Schlange();
15      while (!s.isEmpty()) {
16          q.enqueue(s.pop());
17      }
18      while (!q.isEmpty()) {
19          // Sortiert aufsteigend. Für absteigend einfach das „kleiner“
20          // Zeichen umdrehen.
21          while (!s.isEmpty() && s.top() < q.head()) {
22              q.enqueue(s.pop());
23          }
24          s.push(q.dequeue());
25      }
26  }
```

```
24     }  
25 }
```

- (b) Analysieren Sie die Laufzeit Ihrer Methode in Abhängigkeit von  $n$ .

Zeitkomplexität:  $\mathcal{O}(n^2)$ , da es zwei ineinander verschachtelte `while`-Schleifen gibt, die von der Anzahl der Elemente im Stapel abhängen.