Einzelprüfung „Theoretische Informatik / Algorithmen (vertieft)“

Einzelprüfungsnummer 66115 / 2015 / Frühjahr

Thema 2 / Aufgabe 5

(Sortieren mit Stapel)

Stichwörter: Stapel (Stack), Algorithmische Komplexität (O-Notation)

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

Stapel	Schlange
`boolean isEmpty()`	`boolean isEmpty()`
`void push(int e)`	`enqueue(int e)`
`int pop()`	`int dequeue()`
`int top()`	`int head()`

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(Stapel 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.

Lösungsvorschlag

```
q := neue Schlange
while s not empty:
    q.enqueue(S.pop())
while q not empty:
    while s not empty and s.top() < q.head():
        q.enqueue(s.pop())
    s.push(q.dequeue)
```

Als Java-Code

```
  /**
   * So ähnlich wie der <a href=
   * "https://www.geeksforgeeks.org/sort-stack-using-temporary-stack/">Stapel-
↪   Sortiert-Algorithmus
   * der nur Stapel verwendet</a>, nur mit einer Warteschlange.
   *
   * @param s Der Stapel, der sortiert wird.
   */
  public static void sort(Stapel s) {
    Schlange q = new Schlange();
    while (!s.isEmpty()) {
      q.enqueue(s.pop());
```

```
    }
    while (!q.isEmpty()) {
      // Sortiert aufsteigend. Für absteigend einfach das „kleiner"
      // Zeichen umdrehen.
      while (!s.isEmpty() && s.top() < q.head()) {
        q.enqueue(s.pop());
      }
      s.push(q.dequeue());
    }
  }
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2015/fruehjahr/schlange/Sort.java`

Klasse Sort

```
public class Sort {

  /**
   * So ähnlich wie der <a href=
   * "https://www.geeksforgeeks.org/sort-stack-using-temporary-stack/">Stapel-
↪   Sortiert-Algorithmus
   * der nur Stapel verwendet</a>, nur mit einer Warteschlange.
   *
   * @param s Der Stapel, der sortiert wird.
   */
  public static void sort(Stapel s) {
    Schlange q = new Schlange();
    while (!s.isEmpty()) {
      q.enqueue(s.pop());
    }
    while (!q.isEmpty()) {
      // Sortiert aufsteigend. Für absteigend einfach das „kleiner"
      // Zeichen umdrehen.
      while (!s.isEmpty() && s.top() < q.head()) {
        q.enqueue(s.pop());
      }
      s.push(q.dequeue());
    }
  }

  public static Stapel stapelBefüllen(int[] zahlen) {
    Stapel s = new Stapel();
    for (int i : zahlen) {
      s.push(i);
    }
    return s;
  }

  public static void zeigeStapel(Stapel s) {
    while (!s.isEmpty()) {
      System.out.print(s.pop() + " ");
    }
```

```
    System.out.println();
  }

  public static void main(String[] args) {
    Stapel s1 = stapelBefüllen(new int[] { 4, 2, 1, 5, 3 });
    sort(s1);
    zeigeStapel(s1);

    Stapel s2 = stapelBefüllen(new int[] { 1, 2, 6, 3, 9, 11, 4 });
    sort(s2);
    zeigeStapel(s2);
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2015/fruehjahr/schlange/Sort.java`

Klasse Schlange

```
public class Schlange {

  public Element head;

  public Schlange() {
    head = null;
  }

  public int head() {
    if (head.getNext() == null) {
      return head.getValue();
    }
    Element element = head;
    Element previous = head;
    while (element.getNext() != null) {
      previous = element;
      element = element.getNext();
    }
    element = previous.getNext();
    return element.getValue();
  }

  /**
   * @param value Eine Zahl, die zur Schlange hinzugefügt werden soll.
   */
  public void enqueue(int value) {
    Element element = new Element(value);
    element.setNext(head);
    head = element;
  }

  /**
   * @return Das Element oder null, wenn der Schlange leer ist.
   */
```

```
  public int dequeue() {
    if (head.getNext() == null) {
      int result = head.getValue();
      head = null;
      return result;
    }
    Element element = head;
    Element previous = null;
    while (element.getNext() != null) {
      previous = element;
      element = element.getNext();
    }
    element = previous.getNext();
    previous.setNext(null);
    return element.getValue();
  }

  /**
   * @return Wahr wenn der Schlange leer ist.
   */
  public boolean isEmpty() {
    return head == null;
  }

  public static void main(String[] args) {
    Schlange s = new Schlange();
    s.enqueue(1);
    s.enqueue(2);
    s.enqueue(3);
    System.out.println(s.head());
    System.out.println(s.dequeue());
    System.out.println(s.head());
    System.out.println(s.dequeue());
    System.out.println(s.head());
    System.out.println(s.dequeue());
  }

}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2015/fruehjahr/schlange/Schlange.java`

Klasse Element

```
public class Element {
  public int value;

  public Element next;

  public Element() {
    this.next = null;
  }
```

```
  public Element(int value, Element element) {
    this.value = value;
    this.next = element;
  }

  public Element(int value) {
    this.value = value;
    this.next = null;
  }

  public int getValue() {
    return value;
  }

  public Element getNext() {
    return next;
  }

  public void setNext(Element element) {
    next = element;
  }

}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2015/fruehjahr/schlange/Element.java`

Test-Klasse

```
import static org.junit.Assert.*;

import org.junit.Test;

public class TestCase {

  @Test
  public void testeStapel() {
    Stapel s = new Stapel();
    s.push(1);
    s.push(2);
    s.push(3);

    assertEquals(false, s.isEmpty());

    assertEquals(3, s.top());
    assertEquals(3, s.pop());

    assertEquals(2, s.top());
    assertEquals(2, s.pop());

    assertEquals(1, s.top());
    assertEquals(1, s.pop());
    assertEquals(true, s.isEmpty());
```

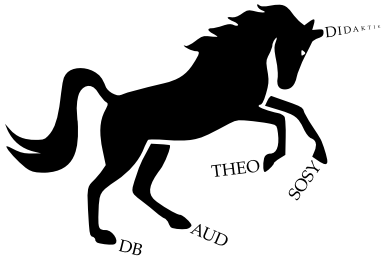
```java
  }

  @Test
  public void testeSchlange() {
    Schlange s = new Schlange();
    s.enqueue(1);
    s.enqueue(2);
    s.enqueue(3);

    assertEquals(false, s.isEmpty());

    assertEquals(1, s.head());
    assertEquals(1, s.dequeue());

    assertEquals(2, s.head());
    assertEquals(2, s.dequeue());

    assertEquals(3, s.head());
    assertEquals(3, s.dequeue());
    assertEquals(true, s.isEmpty());
  }

}
```

Code-Beispiel auf Github ansehen: `src/test/java/org/bschlangaul/examen/examen_66115/jahr_2015/fruehjahr/schlange/TestCase.java`

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

Lösungsvorschlag

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



Die Bschlangaul-Sammlung
Hermine Bschlangaul and Friends

Eine freie Aufgabensammlung mit Lösungen von Studierenden für Studierende zur Vorbereitung auf die 1. Staatsexamensprüfungen des Lehramts Informatik in Bayern.

Hilf mit! Die Hermine schafft das nicht allein! Das ist ein Community-Projekt! Verbesserungsvorschläge, Fehlerkorrekturen, weitere Lösungen sind herzlich willkommen - egal wie - per Pull-Request oder per E-Mail an hermine.bschlangaul@gmx.net.Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden: `https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2015/03/Thema-2/Aufgabe-5.tex`