

Präsenzaufgaben 12

05./06.06.2023

Die Lösung der Aufgaben wird am Ende der Übung von Ihnen vorgestellt.

Aufgabe 1: Fragen

- a) Bei welchen Kantengewichten funktioniert der Dijkstra-Algorithmus nicht mehr?
- b) Nennen Sie die minimale Laufzeitkomplexität der folgenden Funktion zur Suche eines Zeichens in einem Character-Feld in Abhängigkeit von der Feldlänge n . Benutzen Sie die O-Notation.

```
public static boolean contains(char[] text, char c) {  
    if (text.length == 0) {  
        return false;  
    }  
    if (text[0] == c) {  
        return true;  
    }  
  
    // 1. Buchstaben abschneiden und rekursiv aufrufen  
    return contains(Arrays.copyOfRange(text, 1, text.length), c);  
}
```

- c) In welchem Baum-Durchlauf wird das Wurzelement als erstes Element durchlaufen?
- d) Geben Sie für die folgende Funktion die Laufzeitkomplexität in Abhängigkeit von x an.

```
public static int test (int x) {  
    int ret = 0;  
    while (x > 0) {  
        ret += x % 2;  
        x /= 2;  
    }  
    return ret;  
}
```

- e) Welche Datenstruktur würden Sie zur Implementierung einer Prioritätswarteschlange verwenden?
- f) Kreuzen Sie an: welche der nachfolgenden regulären Ausdrücke (PCRE) erkennen den Text **aa**?

Ausdruck	erkennt aa
$a+b*c?aa$	()
$a*b*c?aa$	()
$ab*c?a?$	()

Aufgabe 2:

Suchen Sie mit dem Verfahren von Boyer-Moore-Sunday das jeweils angegebene Muster in den zugehörigen Texten. Geben Sie jeweils das *last*-Feld und die wesentlichen Schritte des Suchvorgangs an. Beispiel:

Suchen Sie in dem Text **nnrfrfenffnrufufein** das Pattern **fein**.

last-Feld:

e	f	i	n	andere
1	0	2	3	-1

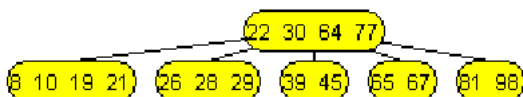
Suchvorgang:

n	n	r	f	r	f	e	n	f	f	e	n	r	u	f	u	f	e	i	n
f	e	i	n																
				f	e	i	n												
								f	e	i	n								
													f	e	i	n			
															f	e	i	n	

- a) Suchen Sie in dem Text **eriraaalraierleiblei** das Pattern **blei**.
- b) Suchen Sie in dem Text **aeaaublbuarurualblau** das Pattern **blau**.
- c) Suchen Sie in dem Text **llebeirriblbiieelbeberiibleibe** das Pattern **bleibe**.

Aufgabe 3:

Nehmen Sie für die folgenden Teile a) – d) den folgenden B-Baum der Ordnung 2 als Ausgangsbaum:



- a) Entfernen Sie vom abgebildeten Baum die 67.
- b) Entfernen Sie vom abgebildeten Baum die 39.
- c) Entfernen Sie vom abgebildeten Baum die 22.
- d) Fügen Sie zum abgebildeten Baum die 2 hinzu.

Aufgabe 4:

Mit dem Boyer-Moore-Verfahren soll in einer `ArrayList<Integer>` eine Folge von Zahlen gesucht werden. Das Pattern ist ebenfalls als `ArrayList<Integer>` gegeben. Programmieren Sie eine Klasse `LastTabelle`, die dafür eine Last-Tabelle erstellt. Die Klasse soll folgende Methoden besitzen:

```
public LastTabelle(ArrayList<Integer> s) //erzeugt eine Last-Tabelle aus
                                         //der angegebenen Zahlenliste
public int get(int c) //gibt den Tabellenwert zur Zahl c zurueck
```

Bedingung: Es ist nicht sinnvoll, ein Feld mit der Länge des Integer-Wertebereichs zu erstellen. Finden Sie eine bessere Lösung. Die Laufzeitkomplexität soll für den Konstruktor $O(n)$ sein (n ist die Stringlänge). Für die `get`-Funktion ist $O(1)$ verlangt. Hinweis: Eine gut programmierte Lösung ist sehr kurz. Das eigentliche Suchverfahren ist für eine typische Klausuraufgabe zu lang und muss nicht programmiert werden.

Aufgabe 5:

Gegeben sei folgendes Gerüst für einen binären Suchbaum:

```
public class BinarySearchTree {
    private TreeNode root;
    ...
}

public class TreeNode {
    public int value;
    public TreeNode left;
    public TreeNode right;
}
```

Der Klasse `BinarySearchTree` soll eine Methode

```
public Node getNextNode(Node n)
```

hinzugefügt werden, die den Knoten zurückgibt, der dem Knoten n in aufsteigender Reihenfolge (In-Order) nachfolgt. Bedingung ist, dass diese Methode bei einem ausgeglichenen Suchbaum eine Zeitkomplexität von maximal $O(\log n)$ im worst case hat. Die Klassen dürfen auch keine neuen Attribute erhalten.

Schreiben Sie einen passenden Algorithmus in Textform oder Pseudocode auf.

Hinweis: Sie brauchen eine Fallunterscheidung:

- Der Knoten n hat ein rechtes Kind (einfacher Fall).
- Der Knoten n hat kein rechtes Kind (schwerer Fall). Hier muss man auch überprüfen, ob n schon der größte Knoten des Baums ist.