

Übung 7

Aufgabe 5 (Optimaler Suchbaum):

7 + 2 + 1 = 10 Punkte

Gegeben sind folgende Knoten mit dazugehörigen Zugriffswahrscheinlichkeiten:

Knoten	l_0	N_1	l_1	N_2	l_2	N_3	l_3	N_4	l_4	N_5	l_5
Werte	$(-\infty, 1)$	1	(1,2)	2	(2,3)	3	(3,4)	4	(4,5)	5	$(5, \infty)$
Wahrscheinlichkeit	0	0.1	0	0.2	0	0.1	0	0.3	0	0.3	0

alle qs sind = 0

Konstruieren Sie einen optimalen Suchbaum wie folgt.

man optimiert jedes Mal für das k => das k, das C minimiert, wird auch als R gewählt

- a) Füllen Sie untenstehende Tabellen für $W_{i,j}$ und $C_{i,j}$ nach dem Verfahren aus der Vorlesung aus. Geben Sie in $C_{i,j}$ ebenfalls **alle möglichen Wurzeln** des optimalen Suchbaums für $\{i, \dots, j\}$ an.

$W_{i,j}$	j	0	1	2	3	4	5
i	1	0	0.1	0.3	0.4	0.7	1.0
	2	—	0	0.2	0.3	0.6	0.9
	3	—	—	0	0.1	0.4	0.7
	4	—	—	—	0	0.3	0.6
	5	—	—	—	—	0	0.3
	6	—	—	—	—	—	0

$C_{ij} (R_{ij})$	0	1	2	3	4	5
1	0	0.1(1)	0.4(2)	0.6(2)	1.3(2,4)	1.9 (4)
2	—	0	0.2(2)	0.4(2)	1.0(4)	1.6 (4)
3	—	—	0	0.1(3)	0.5(4)	1.1 (4)
4	—	—	—	0	0.3(4)	0.9 (4,5)
5	—	—	—	—	0	0.3 (5)
6	—	—	—	—	—	0

- b) Geben Sie einen optimalen Suchbaum für die Knoten mit den gegebenen Zugriffswahrscheinlichkeiten und der gegebenen Reihenfolge der Knoten graphisch an.
- c) Ist der optimale Suchbaum für die Knoten mit den gegebenen Zugriffswahrscheinlichkeiten und der gegebenen Reihenfolge der Knoten eindeutig? Geben Sie dazu eine kurze Begründung an.

Aufgabe 6 (Bucketsort):

10 Punkte

Sortieren Sie das folgende Array mithilfe von Bucketsort. Geben Sie dazu an, welche Buckets Sie verwenden, für welche Intervalle diese stehen und welche Elemente in welcher Reihenfolge in diese Buckets eingefügt werden. Geben Sie zusätzlich den Inhalt der Bucket an, nachdem diese sortiert worden sind. Zuletzt geben Sie das vollständig sortierte Array an.

Sie dürfen ein beliebiges Sortierverfahren nutzen um die einzelnen Buckets zu sortieren. Es ist nicht notwendig dazu Zwischenschritte anzugeben.

0.12	0.26	0.69	0.86	0.93	0.52	0.34	0.25	0.43	0.31
------	------	------	------	------	------	------	------	------	------

Aufgabe 7 (Mastertheorem - Wiederholung):

2 + 2 + 2 + 2 + 2 = 10 Punkte

Bestimmen Sie mithilfe des Mastertheorems die beste mögliche O -Klasse für folgende Rekursionsgleichungen. Geben Sie zusätzlich an, welches p Sie gewählt haben. Begründen Sie, warum weder durch eine andere Wahl von p , noch durch die Anwendung eines anderen Falls des Mastertheorems eine bessere O -Klasse erreicht werden kann.

a)

$$T(1) = 1$$

$$T(n) = 2 \cdot T\left(\frac{n}{4}\right) + n + 4 \quad \text{für } n > 1$$

b)

$$T(1) = 1$$

$$T(n) = 9 \cdot T\left(\frac{n}{3}\right) + n \cdot \sqrt{n} \quad \text{für } n > 1$$

c)

$$T(1) = 1$$

$$T(n) = 16 \cdot T\left(\frac{n}{2}\right) + 12 \cdot n^4 + 16 \cdot n^3 + 100 \quad \text{für } n > 1$$

d)

$$T(1) = 1$$

$$T(n) = 6 \cdot T\left(\frac{n}{6}\right) + \frac{n}{2} \quad \text{für } n > 1$$

e)

$$T(1) = 1$$

$$T(n) = 8 \cdot T\left(\frac{n}{4}\right) + \frac{n^2}{6} + n \cdot \sqrt{n} \quad \text{für } n > 1$$

Aufgabe 8 (Abstrakte Datentypen):

5 + 5 = 10 Punkte

Ein abstrakter Datentyp wird definiert durch die Spezifikationen der Methoden mit denen man auf diesen Datentyp zugreifen kann. Daher kann ein abstrakter Datentyp mehr als eine Implementierung haben, die unterschiedliche Vor- oder Nachteile, insbesondere unterschiedliche Laufzeitkomplexitäten haben können. Wir haben in dieser Aufgabe abstrakte Datentypen (mit möglicherweise etwas anderen Methodennamen) mit einer unüblichen und ineffizienten Implementierung angegeben. Welche abstrakten Datentypen haben wir hier implementiert?

a) Die Methoden `insert`, `delete` und `max` sind die Methoden für binäre Suchbäume wie in der Vorlesung und den Übungen vorgestellt.

```
class ADT_three:
    def __init__(self):
        self.root = None

    def push(x,p,adt_three):
        adt_three.root = insert(adt_three.root,p,x)

    def pop(adt_three):
        if adt_three.root is None:
            return None
        m = max(adt_three.root)
        adt_three.root = delete(m.key,adt_three.root)
        return m.value
```

Geben Sie an, welcher abstrakte Datentyp hier implementiert wurde. Begründen Sie auch ihre Antwort.

b)

```
class ADT_four:
    def __init__(self):
        self.dict_1 = {}
        self.dict_2 = {}

    def all(adf_four):
        all_list = []
        for key in adf_four.dict_2.keys():
            if adf_four.dict_2[key]:
                all_list.append(key)
        return all_list

    def all_n(v, adf_four):
        i = 0
        all_n_list = []
        while str(v)+str(i) in adf_four.dict_1:
            all_n_list.append(adf_four.dict_1[str(v)+str(i)])
            i += 1
        return all_n_list

    def add(v, adf_four):
        adf_four.dict_2[v] = True

    def connect(v1, v2, adf_four):
        i=0
        while str(v1)+str(i) in adf_four.dict_1 and adf_four.dict_1[str(v1)+str(i)] != v2:
            i += 1
        adf_four.dict_1[str(v1)+str(i)] = v2
```

Geben Sie an, welcher abstrakte Datentyp hier implementiert wurde. Begründen Sie auch ihre Antwort.