Übungen zur Vorlesung
Einführung in die Algorithmik WS 15/16

Hasso-Plattner-Institut
Potsdam
12. Januar 2016

Moritz Eissenhauer

Lösung zu Übungsblatt 11

Lösung zu Aufgabe 1

Trennbar

Input: Array w mit Buchstaben **Output:** Boolean der angibt ob das Wort zerlegbar ist

```
1: A ← Array mit Länge n + 1
2: A[n] ← True
3: for i := n − 1 to 0 do
4:    for j := 0 to n − i do
5:       A[i] ← A[i]∨ (Teilwort von w ab i mit Länge j ist Wort ∧A[i + j + 1])
6:    end for
7: end for
8: return A[0]
```

Laufzeit: Das Erstellen des Array brauch $\mathcal{O}(n)$, die beiden Schleifen laufen jeweils $\mathcal{O}(n)$ mal, die anderen Anweisungen haben konstante Laufzeit. die gesamte Laufzeit ist damit $\mathcal{O}(n^2)$.

Lösung zu Aufgabe 2

a)

Preorder: 37, 29, 23, 17, 28, 31, 46, 42, 40, 41, 47, 49
Inorder: 17, 23, 28, 29, 31, 37, 40, 41, 42, 46, 47, 49
Postorder: 17, 28, 23, 31, 29, 41, 40, 42, 49, 47, 46, 37

b)

Jeder Knoten und jeder Null Knoten wird genau ein mal als Parameter übergeben. Die kosten in jedem Rekursionsschritt sind dabei konstant. Die Laufzeit lässt sich folglich mit $\Theta(n)$ abschätzen.

c)

ZZ. Die Schlüssel von T werden sortiert ausgegeben

Beweis. Für jeden Schlüssel n werden zuerst alle Schlüssel des linken Teilbaumes, dann n und dann alle Schlüssel des rechten Teilbaumes ausgegeben. Da per Definition alle schlüssel im linken Teilbaum kleiner sind und alle schlüssel im rechten Teilbaum größer sind gilt das n hinsichtlich seiner Teilbäume sortiert ist.
Da dies für alle Schlüssel n gilt und '$<$' transitiv ist, ist die Ausgabe sortiert. □

Lösung zu Aufgabe 3
ZZ. Jeder gerichtete Binärbaum mit k Knoten hat genau $k+1$ viee Kinder-null-Zeiger (KnZ).

Beweis. **Indunktionsanfang:**
Ein Baum min $k = 1$ Knoten hat $2 = k + 1$ KnZ. **Induktionsannahme:**
Die Aussage gilt für festes k. **Indunktionsschritt:**
Jeder Baum mit $k + 1$ Knoten kann aus einem Baum mit k Knoten durch das Anfügen eines Knoten erstellt werden. Um ein Knoten anzufügen muss ein KnZ entfernt werden. An dem neuen Knoten entstehen zwei neue KnZ. Die Anzahl der KnZ bei dem neuen Baum (mit $k+1$ Knoten) ist also: $(k + 1) - 1 + 2 = k + 2$. Die Aussage gilt also für einen Baum mit $k + 1$ Knoten.
Wenn man annimmt, dass ein Leerer Baum einen KnZ hat gilt die Aussage damit für alle k. □

Lösung zu Aufgabe 4

a)

Man sortiere A aufsteigend. Aus dem sortierten Array ließt man wie folgt rekursiv einen Baum ab:
Für das Teilarray von n bis m ist $\lceil \frac{m-n}{2} \rceil$ der Wurzelknoten, das Teilarray n bis $\lceil \frac{m-n}{2} \rceil - 1$ der linke Teilbaum und das Teilarray $\lceil \frac{m-n}{2} \rceil + 1$ bis m der rechte Teilbaum. Der gesamte Baum ist das Teilarray von 1 bis n.
Die Höhe des Baums ist $O(\log n)$ da nach interpretation des Arrays der linke Teilbaum immer höchstens um 1 höher als der linke ist.
Die Laufzeit ist $\mathcal{O}(n \log n)$, da Sortieren der einzige Schritt des Algorithmus ist. Man könnte natürlich alle Schlüssel noch naheinander in einen Baum einfügen der nach dem genannten Schema aufgebaut ist, dies würde in $o(n \log n)$ gehen.

b)

ZZ. Es gibt keinen Algorithmus, welcher in $o(n \log n)$ Schritten einen Suchbaum konstruiert

Beweis. Angenommen es gibt einen solchen Algorithmus A, dann könnten man erst A und dann Inorder ausführen. Dies würde die Schlüssel in sortierter Reihnfolge ausgeben. Die Laufzeit wäre $o(n \log n) + \Theta(n) \in o(n \log n)$. Dies verletzt die untere Schranke des Sortierens.
Da die Annahme zum Widerspruch führt, kann es keinen solchen Algorithmus geben. □

Lösung zu Aufgabe 5
In der anderen Datei. Der letzte Baum is b) die anderen a)