

ALP3 U14

Übungsgruppe: Qianli Wang und Nazar Sopiha

January 28, 2020

1 Aufgabe 92: Adjazenzlisten

Inspirieren durch Radixsort:

Dieser Algorithmus ist schon im Vergleich zum Radixsort vereinfacht, weil in allen Adjazenzlisten jede Kante gleiches i hat.

Wir können dann folgendes machen:

1. Verteile jedes j in jeder Adjazenzliste in 10 Fächern nach der letzten Ziffer
2. Füge Fach 0, 1, ..., 9 zu einer Liste zusammen.
3. Iterieren m mal, wobei m die Anzahl von Bits von Maximum in der Adjazenzliste ist.

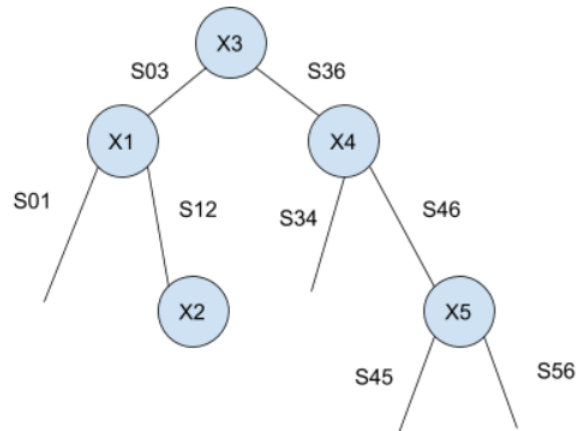
2 Aufgabe 93: Optimaler binärer Suchbaum

Vom Algorithmus erzeugte Tabelle:

Sij	1	2	3	4	5	6
0	0	4	6	17	31	37
1	/	0	1	8	22	26
2	/	/	0	6	19	23
3	/	/	/	0	7	11
4	/	/	/	/	0	2
5	/	/	/	/	/	0

Die Zwischenwerte im Algorithmus, also S_{ij} in der Tabelle bedeuten das Kosten des optimalen Suchbaums für Schlüssel im Intervall $x_i < x < x_j$.

Optimaler binärer Suchbaum:



3 Aufgabe 97: Zahlen mit Münzen

Wir nehmen an, dass man möglich wenige Münze bekommen sollte.

Teilprobleme: A_k bedeutet die Liste von Münzen mit dem zu zahlenden Betrag k .

Randbedingung: $A_{M_i} = [M_i]$

```

1 initialize(betrag, M):
2     memo = [[NIL] * betrag]
3     for each m in M:
4         memo[m] = [m]
5     return memo
6
7 changeMoney(betrag):
8     res = {}
9     if(memo[betrag] != NIL): //Memoization
10         extend memo[betrag] into res // this is a list
11     return
12     for i:= (m.length - 1) to 0 do:
13         if(M[i] <= betrag):
14             append M[i] into res // this is a number
15     return changeMoney(betrag - M[i])
  
```

Python Code sehen Sie unter: <https://github.com/qiauw99/WS2019-20/blob/master/DataStructure/U14/U14.py>