ALP3 U14

Übungsgruppe: Qianli Wang und Nazar Sopiha

February 6, 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.

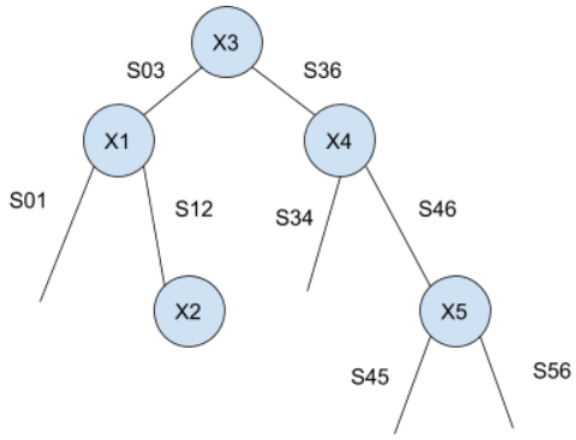
2 Aufgabe 93: Optimaler binärer Suchbaum

Vom Algorthmus erzeugte Tabelle:

```
function x = SolveLinearSystem(A, b)
    n = length(b);
    x = zeros(n, 1);
    y = zeros(n, 1);
    % decomposition of matrix, Doolittle's Method
    for i = 1:1:n
        for j = 1:1:(i - 1)
            alpha = A(i,j);
            for k = 1:1:(j - 1)
                alpha = alpha - A(i,k)*A(k,j);
            end
            A(i,j) = alpha/A(j,j);
        end
        for j = i:1:n
            alpha = A(i,j);
            for k = 1:1:(i - 1)
                alpha = alpha - A(i,k)*A(k,j);
            end
            A(i,j) = alpha;
        end
    end
    %A = L+U-I
    % find solution of Ly = b
    for i = 1:1:n
        alpha = 0;
        for k = 1:1:i
            alpha = alpha + A(i,k)*y(k);
        end
        y(i) = b(i) - alpha;
    end
    % find solution of Ux = y
    for i = n:(-1):1
        alpha = 0;
        for k = (i + 1):1:n
            alpha = alpha + A(i,k)*x(k);
        end
        x(i) = (y(i) - alpha)/A(i, i);
    end
end
```

Die Zwischenwerte im Algorithmus, also S_{ij} in der Tabelle bedeuten das Kosten des optimalen Suchbaums für Schlüssel im Intervall $x_i < x < x_j$.
Die mittlere Suchzeit: $\frac{37}{20}$

Optimaler binärer Suchbaum:



3 Aufgabe 97: Zahlen mit Münzen

Wir nehmen an, dass man möglich weinige Münze bekommen sollte.
Teilprobleme: A_k bedeutet die Liste von Münzen mit dem zu zahlenden Betrag k.
Randbedingung: $A_{M_i} = [M_i]$

```
coins= [1,2,5,10]

changeMoney(nums):
  dp = [inf] * nums
  dp[0] = 0

  for i = 1 to ()nums - 1):
    for j 0 to length(coins):
      if(coins[j] <= nums):
        dp[i] = min(dp[i], dp[i - coins[j]] + 1)
      endif
    endfor
  endfor
  return dp[nums - 1]
```

Python Code sehen Sie unter: *https://github.com/qiaw99/WS2019-20/blob/master/DataStructure/U14/U14_97.py*