Algorithmics

Sebastian Iwanowski
FH Wedel

3. Solutions for the dictionary problem
3.4 Optimal binary search trees

Algorithmics 3

3.4 Optimal binary search trees

Problem:

i. Let $S = \{a_1, a_2, \ldots, a_n\}$ be a linearily ordered set with predetermined probablities p_i for the occurrence of a_i und q_i für the occurrence of an element a in between: $a_i < a < a_{i+1}$.
ii. Construct a binary search tree which minimizes the expected response time (i.e. number of comparisons with elements a_i).

Required tree properties:

The tree should not only find the position of elements contained in the given dictionary, but also locate the position where new elements would be placed:
Inner nodes correspond to elements contained, leaves correspond to elements in between

Solution by the algorithm of Bellman (1957)

Time for the construction of the search tree: $O(n^3)$ (easy to prove)

Improvement: $O(n^2)$

References:

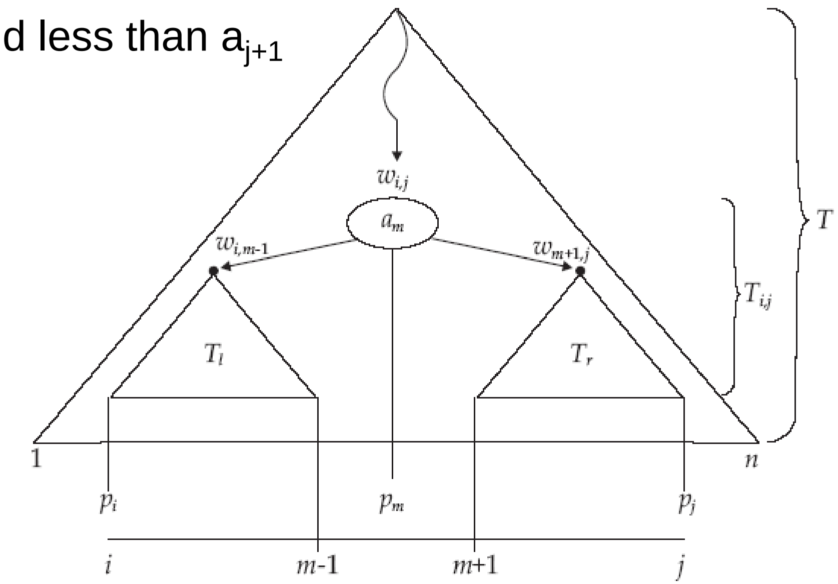
Skript Alt S. 65 – 70 (ch. 3.3) in German: Other references are less clear

Cormen 15.5 (ch. Dynamic Programming)

Knuth 6.2.2 (Binary Tree Searching)

Bellman's Algorithm for optimal binary search trees:

$T_{i,j}$: optimal subtree for search items greater than a_{i-1} and less than a_{j+1}



Special cases:

$T_{i,i}$: optimal subtree for search items greater than a_{i-1} and less than a_{i+1}.
This tree consist of one node comparing with a_i

$T_{i,i-1}$: optimal subtree for search items greater than a_{i-1} and less than a_i.
This tree is empty and corresponds to a leaf.

$T_{i,n}$: optimal subtree for search items greater than a_{i-1}

$T_{1,j}$: optimal subtree for search items less than a_{j+1}

$T_{1,n}$: optimal subtree for all search items

aus: Skript Alt

Bellman's Algorithm for optimal binary search trees:

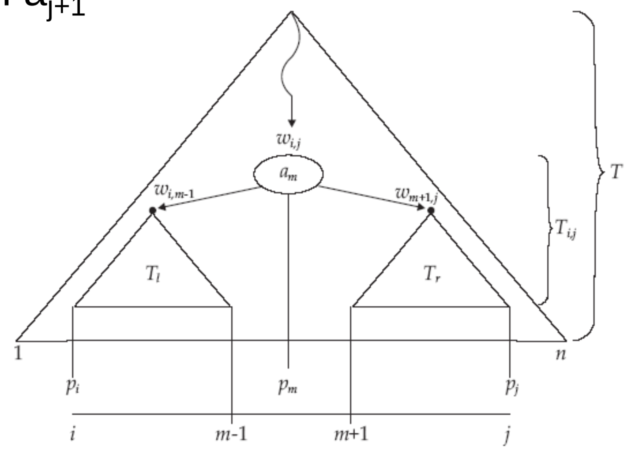
$T_{i,j}$: optimal subtree for search items greater than a_{i-1} and less than a_{j+1}

$r_{i,j}$: index m of the root of $T_{i,j}$: The item to be compared with is a_m

$P(T_{i,j})$: expected costs for $T_{i,j}$ if $T_{i,j}$ is chosen

$w_{i,j}$: probability that $T_{i,j}$ is chosen

$c_{i,j}$: expected costs for $T_{i,j}$ if no precondition is known



Lemma 3.3.5: If $T_{i,j}$ is optimal, then each subtree is also optimal.

Algorithm 3: [Bellman, 1957] Iterative Suche nach dem optimalen Suchbaum T.

for $i = 0, \ldots, n$ **do**
  $w_{i+1,i} = q_i$
  $c_{i+1,i} = 0$
end for
for $k = 0, \ldots, n-1$ **do**
  for $i = 1, \ldots, n-k$ **do**
    $j = i + k$
    Bestimme m mit $i \leq m \leq j$, so dass $c_{i,m-1} + c_{m+1,j}$ minimal ist.
    $r_{i,j} = m$
    $w_{i,j} = w_{i,m-1} + w_{m+1,j} + p_m$
    $c_{i,j} = c_{i,m-1} + c_{m+1,j} + w_{i,j}$
  end for
end for

Initialization for empty trees corresponding to the intervals in between the search keys

k+1 is the number of elements considered in $T_{i,j}$

This is improved in Knuth

Assertion 3.3.6:

$w_{i,j} = w_{i,m-1} + p_m + w_{m+1,,j}$

$c_{i,j} = w_{i,j} \bullet P(T_{i,j})$
$= w_{i,j} \bullet (1 + P(T_{i,m-1}) + P(T_{m+1,j}))$
$= w_{i,j} + c_{i,m-1} + c_{m+1,,j}$

Lemma 3.3.7:

$r_{i,j-1} \leq r_{i,j} \leq r_{i+1,j}$

aus: Skript Alt

Example from Skript Alt:

p_1=0 p_2=0,1 p_3=0,2 p_4=0,2

q_0=0,1 q_1=0,1 q_2=0,1 q_3=0,1 q_4=0,1

i	0	1	2	3	4
$Init$	$w_{1,0} = 0,1$ $c_{1,0} = 0$	$w_{2,1} = 0,1$ $c_{2,1} = 0$	$w_{3,2} = 0,1$ $c_{3,2} = 0$	$w_{4,3} = 0,1$ $c_{4,3} = 0$	$w_{5,4} = 0,1$ $c_{5,4} = 0$
$k = 0$		$r_{1,1} = 1$ $w_{1,1} = 0,2$ $c_{1,1} = 0,2$	$r_{2,2} = 2$ $w_{2,2} = 0,3$ $c_{2,2} = 0,3$	$r_{3,3} = 3$ $w_{3,3} = 0,4$ $c_{3,3} = 0,4$	$r_{4,4} = 4$ $w_{4,4} = 0,4$ $c_{4,4} = 0,4$
$k = 1$			$r_{1,2} = 2$ $w_{1,2} = 0,4$ $c_{1,2} = 0,6$	$r_{2,3} = 3$ $w_{2,3} = 0,6$ $c_{2,3} = 0,9$	$r_{3,4} = 3$ $w_{3,4} = 0,7$ $c_{3,4} = 1,1$
$k = 2$				$r_{1,3} = 2$ $w_{1,3} = 0,7$ $c_{1,3} = 1,3$	$r_{2,4} = 3$ $w_{2,4} = 0,9$ $c_{2,4} = 1,6$
$k = 3$					$r_{1,4} = 3$ $w_{1,4} = 1$ $c_{1,4} = 2$

Tabelle 3.1: Tabelle zur Speicherung der Berechnungen des Algorithmus

Resulting construction of search tree: