Optimal Binary Search Trees

Dr. K. RAGHAVA RAO
Professor in CSE,
School of Computing
KL University
krraocse@gmail.com
http://mcadaa.blog.com

DYNAMIC PROGRAMMING

- **P**roblems like knapsack problem, shortest path can be solved by greedy method in which optimal decisions can be made one at a time.
- For many problems, it is not possible to make stepwise decision in such a manner that the sequence of decisions made is optimal.

DYNAMIC PROGRAMMING (Contd..)

Example:

- Suppose a shortest path from vertex i to vertex j is to be found.
- Let Ai be vertices adjacent from vertex i. which of the vertices of Ai should be the second vertex on the path?
- One way to solve the problem is to enumerate all decision sequences and pick out the best.
- In dynamic programming the principle of optimality is used to reduce the decision sequences.

DYNAMIC PROGRAMMING (Contd..)

Principle of optimality:

- An optimal sequence of decisions has the property that whatever the initial state and decisions are, the remaining decisions must constitute an optional decision sequence with regard to the state resulting from the first decision.
- In the greedy method only one decision sequence is generated.
- In dynamic programming many decision sequences may be generated.

OPTIMAL BINARY SEARCH TREES

- Definition: **binary search tree (BST)** A binary search tree is a binary tree; either it is empty or each node contains an identifier and

(i) all identifiers in the left sub tree of T are less than the identifiers in the root node T.

(ii) all the identifiers the right sub tree are greater than the identifier in the root node T.

(iii) the right and left sub tree are also BSTs.

ALGORITHM TO SEARCH FOR AN IDENTIFIER IN THE TREE 'T'.

Procedure SEARCH (<u>T</u> <u>X</u> <u>I</u>)

// Search T for X, each node had fields LCHILD, IDENT, RCHILD//

// Return address i pointing to the identifier X// //Initially T is pointing to tree.

//ident(i)=X or i=0 //

i ← T

Algorithm to search for an identifier in the tree 'T'(Contd..)

While i $\neq$ 0 do

case : X < Ident(i) : i $\leftarrow$LCHILD(i)

: X = IDENT(i) : RETURN i

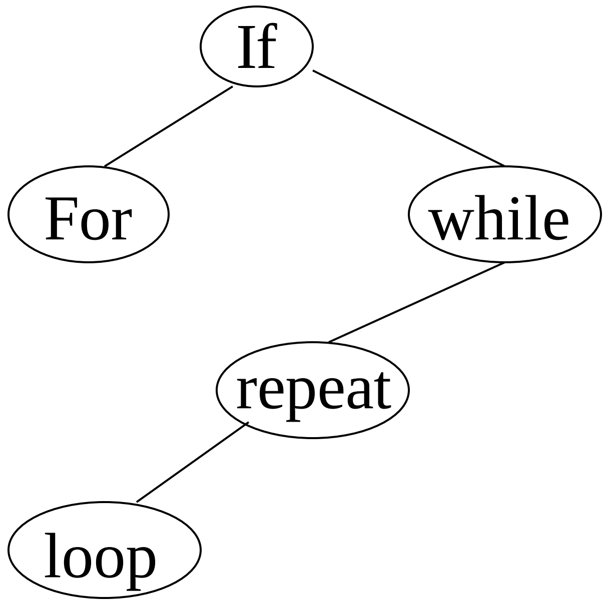
: X > IDENT(i) : I $\leftarrow$ RCHILD(i)

endcase

repeat

end SEARCH

Optimal Binary Search trees - Example



if each identifier is searched with equal probability the average number of comparisons for the above tree are $\frac{1+2+2+3+4}{5} = 12/5$.

OPTIMAL BINARY SEARCH TREES (Contd..)

- Let us assume that the given set of identifiers are $\{a_1, a_2, \ldots\ldots\ldots a_n\}$ with $a_1 < a_2 < \ldots\ldots\ldots < a_n$.
- Let P_i be the probability with which we are searching for a_i.
- Let Q_i be the probability that identifier x being searched for is such that $a_i < x < a_{i+1}$ $0 \leq i \leq n$, and $a_0 = -\infty$ and $a_{n+1} = +\infty$.

OPTIMAL BINARY SEARCH TREES (Contd..)

- Then $\sum Q_i$ is the probability of an unsuccessful search.

  $0 \leq i \leq n$

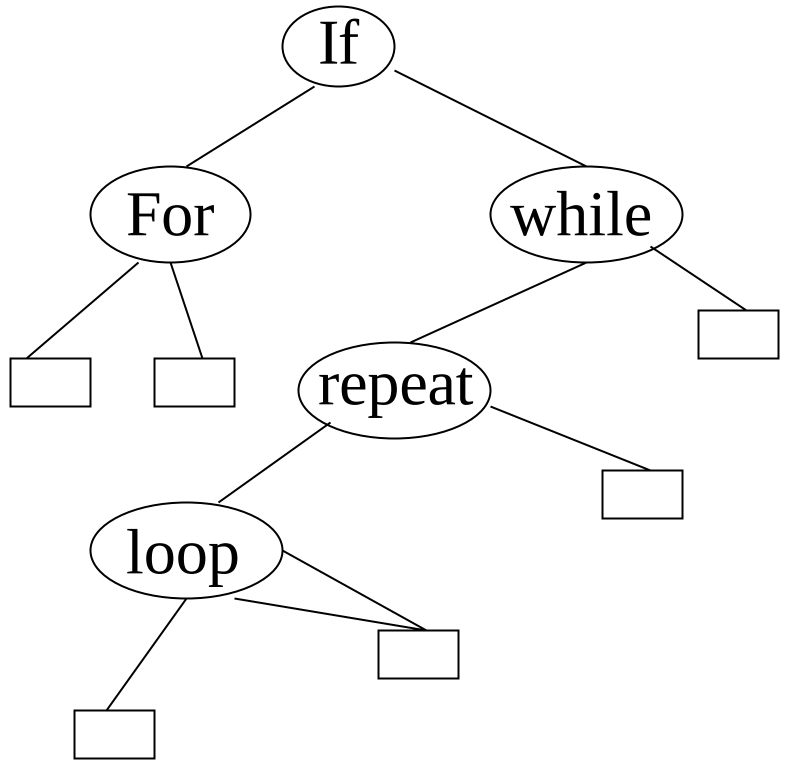
  $\sum P(i) + \sum Q(i) = 1.$ Given the data,

  $1 \leq i \leq n$ $0 \leq i \leq n$

let us construct one optimal binary search tree for $(a_1 \ldots\ldots\ldots. a_n)$.

- In place of empty sub tree, we add external nodes denoted with squares.
- Internet nodes are denoted as circles.

OPTIMAL BINARY SEARCH TREES (Contd..)



Construction of optimal binary search trees

- A BST with n identifiers will have n internal nodes and n+1 external nodes.
- Successful search terminates at internal nodes unsuccessful search terminates at external nodes.
- If a successful search terminates at an internal node at level L, then L iterations of the loop in the algorithm are needed.
- Hence the expected cost contribution from the internal nodes for a_i is P(i) * level(a_i).

OPTIMAL BINARY SEARCH TREES (Contd..)

- Unsuccessful searche terminates at external nodes i.e. at i = 0.
- The identifiers not in the binary search tree may be partitioned into n+1 equivalent classes

  E_i $0 \le i \le n$.

  E_o contains all X such that $X \le a_i$

  E_i contains all X such that $a < X <= a_{i+1}$ $1 \le i \le n$

  E_n contains all X such that $X > a_n$

OPTIMAL BINARY SEARCH TREES (Contd..)

- For identifiers in the same class E_i, the search terminate at the same external node.
- If the failure node for E_i is at level L, then only L-1 iterations of the while loop are made

  $\therefore$ The cost contribution of the failure node for E_i is Q(i) * level (E_i)   -1)

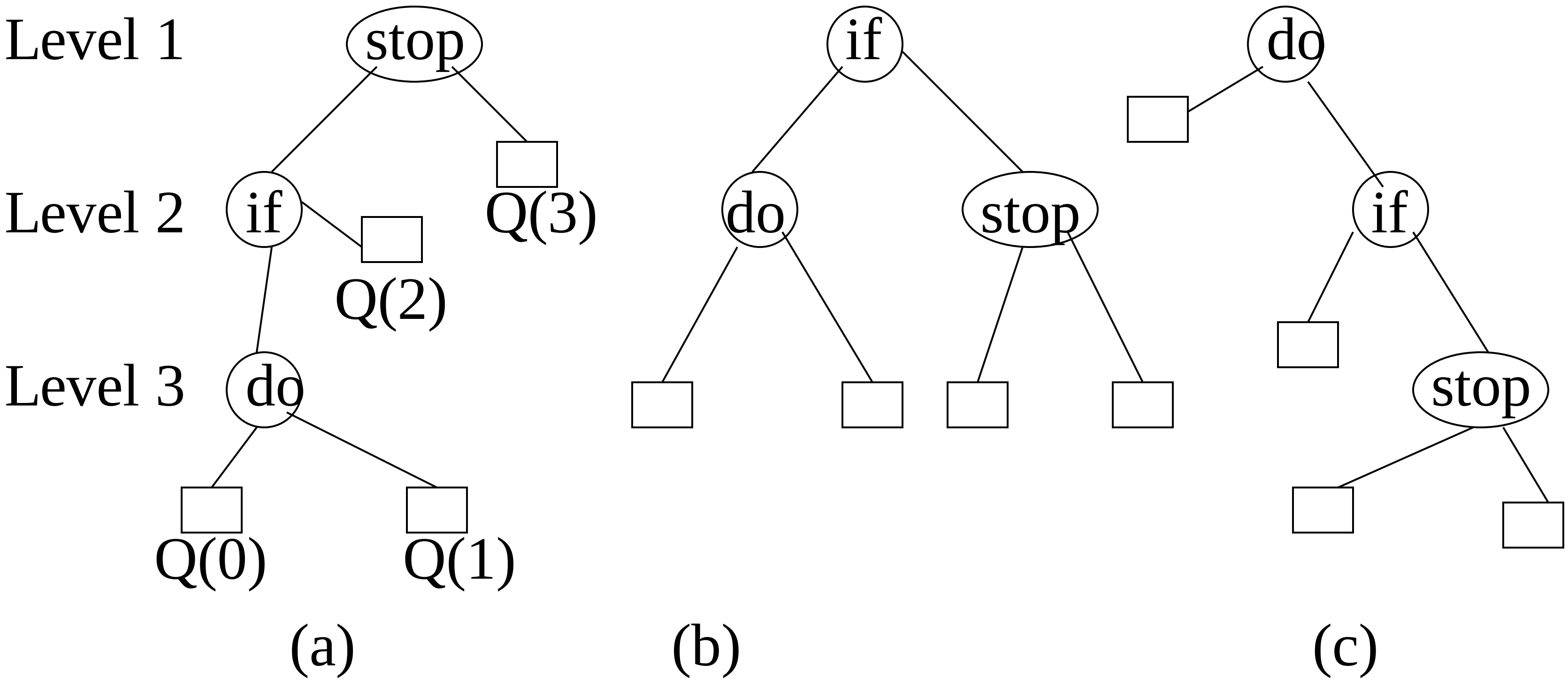
OPTIMAL BINARY SEARCH TREES (Contd..)

- Thus the expected cost of a binary search tree is:

$$\sum_{1 \le i \le n} P(i) * \text{level}(a_i) + \sum_{0 \le i \le n} Q(i) * \text{level}(E_i) - 1) \quad \ldots\ldots(2)$$

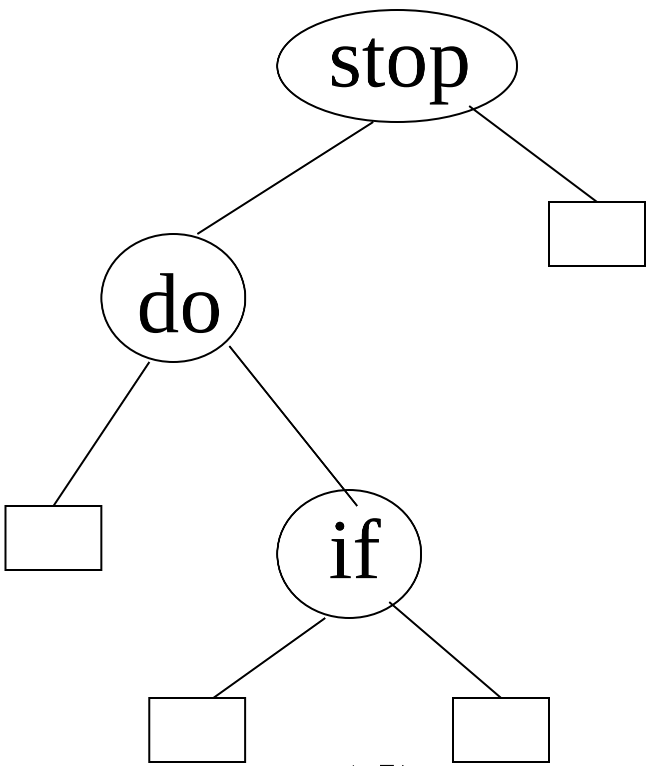
- An optimal binary search tree for $\{a_1 \ldots\ldots, a_n\}$ is a BST for which (2) is minimum .
- Example: Let $\{a_1, a_2, a_3\} = \{\text{do, if, stop}\}$

OPTIMAL BINARY SEARCH TREES (Contd..)

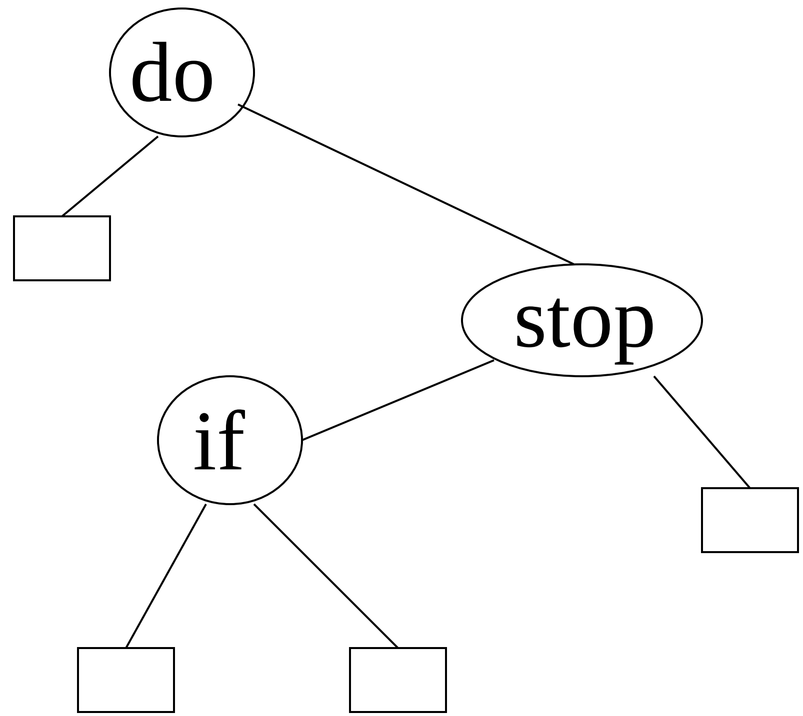


{a1,a2,a3}={do,if,stop}

OPTIMAL BINARY SEARCH TREES (Contd..)



(d)



(c)

OPTIMAL BINARY SEARCH TREES (Contd..)

- With equal probability P(i)=Q(i)=1/7.
- Let us find an OBST out of these.
- Cost(tree a)=∑P(i)*level a(i) +∑Q(i)*level (Ei) -1

  1≤i≤n 0≤i≤n

  (2-1) (3-1) (4-1) (4-1)

  =1/7[1+2+3 + 1 + 2 + 3 + 3] = 15/7
- Cost (tree b) =17[1+2+2+2+2+2+2] =13/7
- Cost (tree c) =cost (tree d) =cost (tree e) =15/7

∴ tree b is optimal.

OPTIMAL BINARY SEARCH TREES (Contd..)

- If P(1) =0.5 ,P(2) =0.1, P(3) =0.005 , Q(0) =.15 , Q(1) =.1, Q(2) =.05 and Q(3) =.05 find the OBST.
- Cost (tree a) = .5 x 3 +.1 x 2 +.05 x 3 +.15x3 +.1x3 +.05x2 +.05x1 = 2.65
- Cost (tree b) =1.9 , Cost (tree c) =1.5 ,Cost (tree d) =2.05 ,
- Cost (tree e) =1.6 Hence tree C is optimal.

OPTIMAL BINARY SEARCH TREES (Contd..)

- To obtain a OBST using Dynamic programming we need to take a sequence of decisions regard. The construction of tree.
- First decision is which of a_i is be as root.
- Let us choose a_k as the root . Then the internal nodes for $a_1,\ldots\ldots,a_{k-1}$ and the external nodes for classes $E_o,E_1,\ldots\ldots,E_{k-1}$ will lie in the left subtree L of the root.
- The remaining nodes will be in the right subtree R.

OPTIMAL BINARY SEARCH TREES (Contd..)

Define

$$\text{Cost}(L) = \sum_{1 \le i \le k} P(i) * \text{level}(ai) + \sum_{0 \le i \le k} Q(i) * (\text{level}(E_i) - 1)$$

$$\text{Cost}(R) = \sum_{k \le i \le n} P(i) * \text{level}(ai) + \sum_{k \le i \le n} Q(i) * (\text{level}(E_i) - 1)$$

- Tij be the tree with nodes $a_{i+1}, \ldots, a_j$ and nodes corresponding to $E_i, E_{i+1}, \ldots, E_{j.}$
- Let W(i,j) represents the weight of tree $T_{ij.}$

OPTIMAL BINARY SEARCH TREES (Contd..)

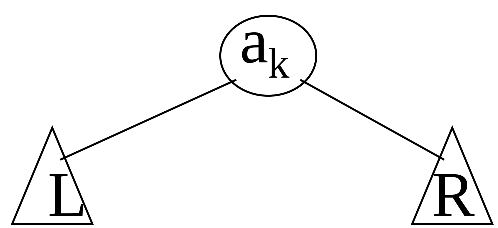
$$W(i,j)=P(i+1)+\ldots+P(j)+Q(i)+Q(i+1)\ldots Q(j)=Q(i)+\sum_{l=i+1}^{j}[Q(l)+P(l)]$$

- The expected cost of the search tree in (a) is (let us call it T) is

  P(k)+cost(l)+cost(r)+W(0,k-1)+W(k,n)

  W(0,k-1) is the sum of probabilities corresponding to nodes and nodes belonging to equivalent classes to the left of a_k.

  W(k,n) is the sum of the probabilities corresponding to those on the right of a_k.



(a) OBST with root a_k