OPTIMAL BINARY SEARCH TREES.

①

Binary search tree is a tool for implementing binary search techniques.

Given a fixed set of identifiers, we wish to create a binary search tree organization. We may expect different binary search trees for the same identifier set to have different performance characteristics.

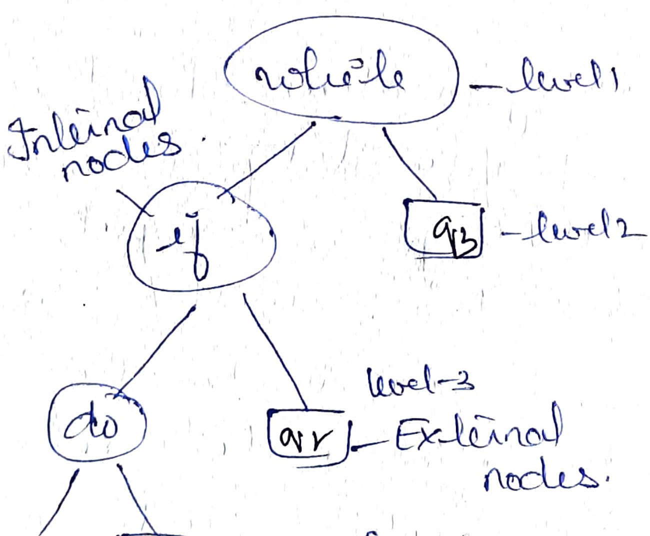
We want total time spent on searching to be as low as possible.

$n = 3$;

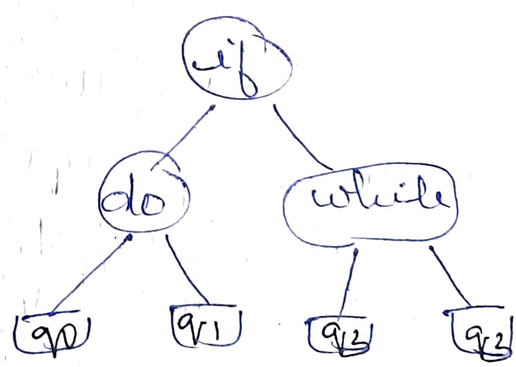
Identifiers set $(a_1, a_2, a_3) = (\text{do, if, while})$

(frequencies) or Probability of successful search $\Rightarrow (P_1, P_2, P_3) = (0.5, 0.1, 0.05)$

Probability of unsuccessful search $\Rightarrow (q_0, q_1, q_2, q_3) = (0.15, 0.1, 0.05, 0.05)$



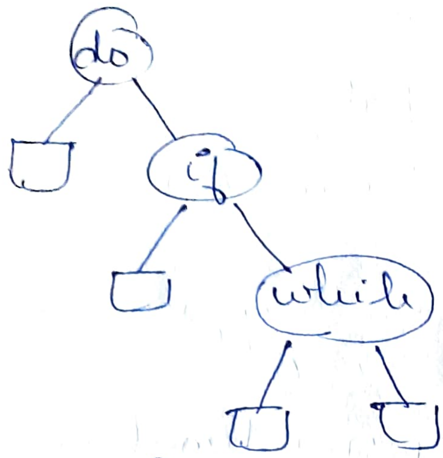
(a)



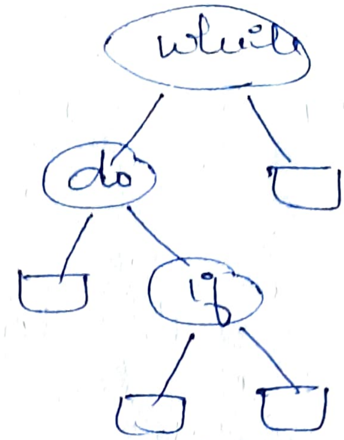
(b)

→ If there are 'n' identifiers, then there will be exactly n+1 external nodes.

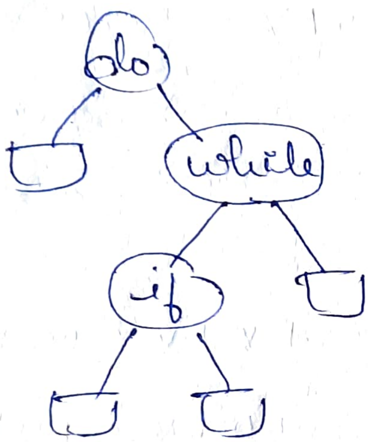
Unsuccessful search → search for an identifier not in the tree



(c)



(d)



(e).

Sometimes we may search for value which is not in the tree. Therefore External nodes are added in place of every empty sub tree in BST, to represent values not in a tree.

Ex q_0 represents values all less than do, q_3 rep all values greater than while

→ No. of binary search trees possible with 'n' nodes are = $\frac{1}{n+1}\, {}^{2n}C_n$

$\frac{n!}{(n-r)!\, r!}$

above Ex n=3

$$\frac{1}{3+1}\, {}^{2\,3}C_3$$

$$= \frac{1}{4}\, {}^{6}C_3 = \frac{1}{4}\, \frac{6\times5\times4\times\cancel{3\times2\times1}}{3\times2\times1\times\cancel{3\times2\times1}}$$

$$= \frac{1}{\cancel{4}}\, \frac{\overset{20\,5}{\cancel{120}}}{6} = \underline{5}$$

→ Time of search will determine the cost of tree.

→ Successful search terminates at internal nodes

→ Unsuccessful " " at External "

Cost (T) = Cost (successful search) + cost (unsuccessful search).

$$= \sum_{i=1}^{n} P_i * level(a_i) + \sum_{i=0}^{n} q_i * [level(E_i) - 1]$$

P(i) - Probability of searching an internal node
q(i) - " " " " and external "

Cost (Ta) = 0.5 * 3 + 0.1 * 2 + 0.05 * 1 +
0.15 * 3 + 0.1 * 3 + 0.05 * 2 + 0.05 * 1.

= 2.65

Cost (Tb) = 0.5 * 2 + 0.1 * 1 * 0.05 * 2 +
0.15 * 2 + 0.1 * 2 + 0.05 * 2 + 0.05 * 2

= 1.0 + 0.1 + 0.10 + 0.30 + 0.2 + 0.10 + 0.10

= 1.9.

Cost (Tc) = 1.5

Cost (Td) = 2.15

Cost (Te) = 1.6.

∴ Tree (c) is optimal even though it has higher levels than Tree (b). because of probability of nodes.

As no. of nodes in the binary search tree increases, it is ~~not~~ very tedious work to construct all possible binary search trees and finding out the optimal binary search tree.

Therefore we can directly construct the optimal binary search tree by making use of three functions w, c, r.

$w(i,i) = q(i)$
$r(i,i) = 0$ where $0 \le i \le n$.
$c(i,i) = 0$. n = no. of identifiers.

$w(i, i+1) = q(i) + q(i+1) + p(i+1)$
$r(i, i+1) = i+1$
$c(i, i+1) = q(i) + q(i+1) + p(i+1)$

$w(i,j) = w(i, j-1) + P(j) + q(j)$ → Sum of Probabilities.

$c(i,j) = w(i,j) + \min_{i < k \le j} \{ c(i, k-1) + c(k,j) \}$
↓ Cost OBST containing the keys from i to j, so we have to find $c(1,4)$

$r(i,j) = k$.
↳ Gives node value.

→ This process can be repeated until $w(0,n)$, $c(0,n)$, $r(0,n)$ are obtained.

Example :-

$n = 4$

$(a_1, a_2, a_3, a_4) = $ (do, if, int, while)

$(P_1, P_2, P_3, P_4) = (3, 3, 1, 1)$

$(q_0, q_1, q_2, q_3, q_4) = (2, 3, 1, 1, 1)$

So we have to calculate $w(0,4)$, $c(0,4)$, $r(0,4)$.

compute w, c, r for $j-i=0$, $j-i=1$, $j-i=2$, $j-i=3$, $j-i=4$ ..... $j-i=n$.

here $0 \le i \le 4$.

Step1: Compute w, c, r for $j-i=0$.

$w(0,0) = q(0) = 2$

$w(1,1) = q(1) = 3$

$w(2,2) = q(2) = 1$

$w(3,3) = q(3) = 1$

$w(4,4) = q(4) = 1$.

$w(i,i) = q(i)$

$r(i,i) = 0$

$c(i,i) = 0$.

$r(0,0) = 0$, $r(1,1) = 0$, $r(2,2) = 0$, $r(3,3) = 0$

$r(4,4) = 0$.

$c(0,0) = 0$, $c(1,1) = 0$, $c(2,2) = 0$, $c(3,3) = 0$

$c(4,4) = 0$.

$w(0,1)$
$(1,2)$
$(2,3)$
$(3,4)$.

Step 2:- Compute w, c, r for $j-i=1$.

$$w(i, i+1) = q(i) + q(i+1) + P(i+1)$$
$$r(i, i+1) = i+1$$
$$c(i, i+1) = q(i) + q(i+1) + P(i+1).$$

$$w(0,1) = q(0) + q(1) + P(1)$$
$$= 2+3+3 = 8$$

$$w(1,2) = q(1) + q(2) + P(2)$$
$$= 3+1+3 = 7$$

$$w(2,3) = q(2) + q(3) + P(3)$$
$$= 1+1+1 = 3.$$

$$w(3,4) = q(3) + q(4) + P(4)$$
$$= 1+1+1 = 3.$$

$r(0,1) = 1$, $r(1,2) = 2$, $r(2,3) = 3$, $r(3,4) = 4$.

$c(0,1) = 8$, $c(1,2) = 7$, $c(2,3) = 3$, $c(3,4) = 3$.

Step 3 :- w, c, r for $j-i=2$

$(0,2)$
$(1,3)$
$(2,4)$.

$$w(i,j) = w(i, j-1) + P(j) + q(j)$$
$$c(i,j) = w(i,j) + \min_{i<k\le j} \{c(i, k-1) + c(k,j)\}.$$
$$r(i,j) = k.$$

$$w(0,2) = w(0,1) + P(2) + q(2)$$
$$= 8+3+1 = 12.$$

$$w(1,3) = w(1,2) + P(3) + q(3)$$
$$= 7+1+1 = 9$$

$$w(3,4) = w(2,3) + p(4) + q(4)$$
$$= 3 + 1 + 1 = 5$$

$$c(0,2) = w(0,2) + \min_{\substack{k=1,2 \\ 0<k\leq 2}} \{ c(0,0) + c(1,2),$$
$$c(0,1) + c(2,2) \}$$

$$= 12 + \min \{ 0+7, 8+0 \}$$
$$= 12 + \min \{7, 8\}$$
$$= 12 + 7 = \underline{\underline{19}}$$

$$c(1,3) = w(1,3) + \min_{\substack{k=2,3 \\ 1<k\leq 3}} \{ c(1,1) + c(2,3),$$
$$c(1,2) + c(3,3) \}$$

$$= 9 + \min \{ 0+3, 7+0 \}$$
$$= 9 + 3 = \underline{\underline{12}}$$

$$c(2,4) = w(2,4) + \min_{\substack{k=3,4 \\ 2<k\leq 4}} \{ c(2,2) + c(3,4),$$
$$c(2,3) + c(4,4) \}$$

$$= 5 + \min \{ 0+3, 3+0 \}$$
$$= 5 + 3 = \underline{\underline{8}}$$

$r(0,2) = 1$, $r(1,3) = 2$, $r(2,4) = 3$.

Step 4:- compute w, c, r for $j-i=3$. $(0,3)$ $(1,4)$.

$w(0,3) = w(0,2) + P(3) + q(3)$
$= 12 + 1 + 1 = 14$.

$w(1,4) = w(1,3) + P(4) + q(4)$
$= 9 + 1 + 1 = 11$

$c(0,3) = w(0,3) + \min\limits_{0<k\leq 3}^{k=1,2,3} \{ c(0,0) + c(1,3),$
$c(0,1) + c(2,3),$
$c(0,2) + c(3,3) \}$

$= 14 + \min\{0+12, 8+3, 19+0\}$
$= 14 + \min\{12, 11, 19\}$
$= 14 + 11 = \underline{\underline{25}}$

$c(1,4) = w(1,4) + \min\limits_{1<k\leq 4}^{k=2,3,4} \{ c(1,1) + c(2,4),$
$c(1,2) + c(3,4),$
$c(1,3) + c(4,4) \}$

$= 11 + \min\{0+8, 7+3, 12+0\}$
$= 11 + \min\{8, 10, 12\}$
$= 11 + 8 = \underline{\underline{19}}$

$r(0,3) = 2, \quad r(1,4) = 2$.

Step 4 :- Compute w, c, r for $j - i = 4$

$w(0,4) = w(0,3) + P(4) + q(4)$

$= 14 + 1 + 1 = \underline{16}$

$c(0,4) = w(0,4) + \min\limits_{0 < k \leq 4} \{ c(0,0) + c(1,4),$ $\quad k = 1,2,3,4$

$c(0,1) + c(2,4),$

$c(0,2) + c(3,4),$

$c(0,3) + c(4,4) \}$

$= 16 + \min\{ 0 + 19,\ 8 + 8,\ 19 + 3,\ [illegible] + 1 \}$

$= 16 + \min\{ 19, 16, 22, 25 \}$

$= 16 + 16 = \underline{32}$

$r(0,4) = \underline{2}$

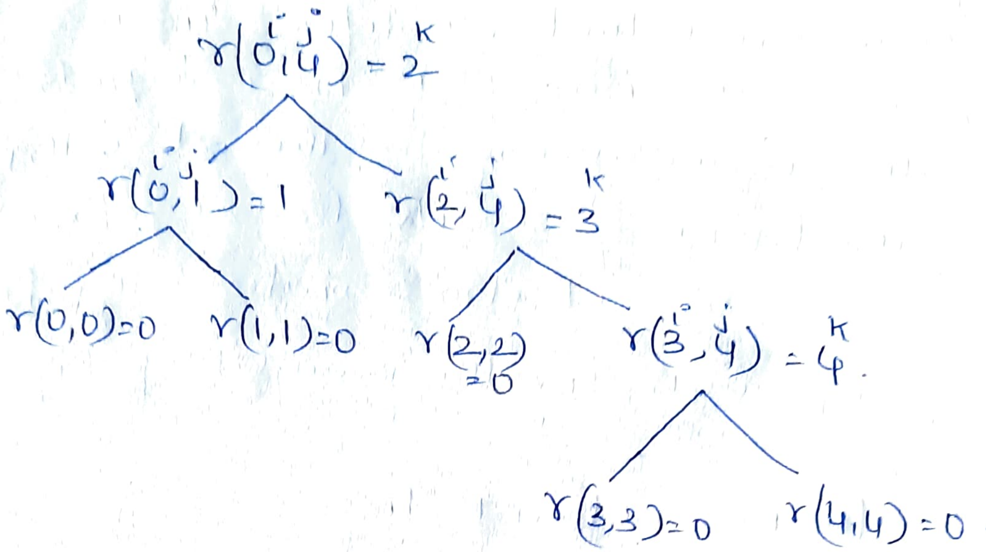
Tree Construction :- (OBST)

$r(i,j) = k$

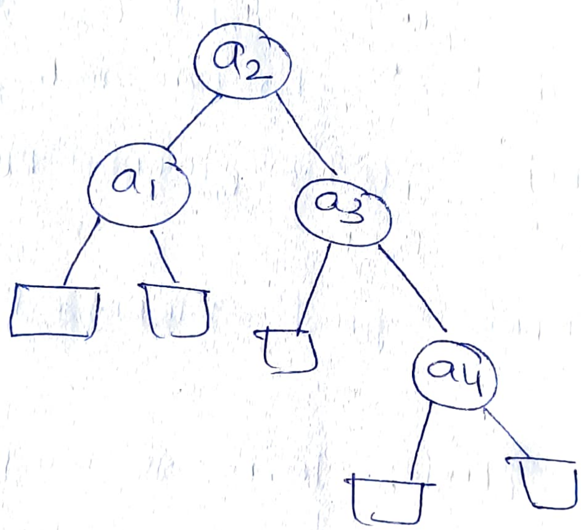
$r(i, k-1)$ $\quad r(k, j)$

[illegible]

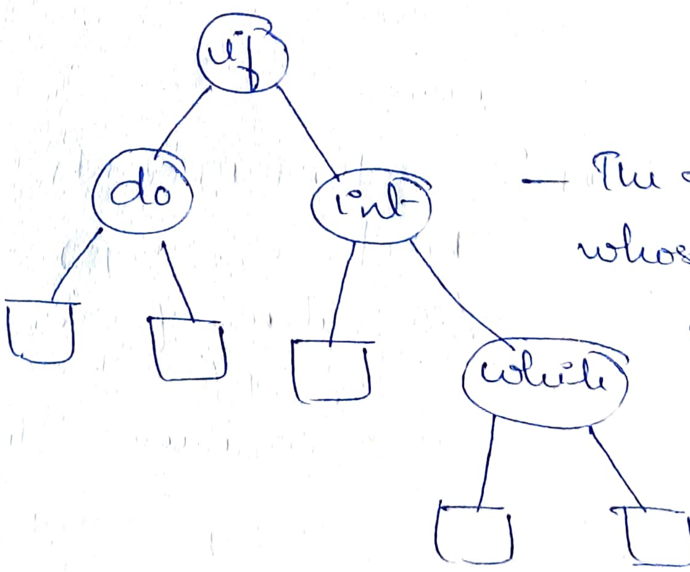
$r(0,n)$



⇓



⇓



— The final OBST whose cost $c(0,4) = \underline{\underline{32}}$

Algorithm OBST (P, q, n)

// Given n distinct identifiers $a_1 < a_2 < \dots a_n$ and
// probabilities $P[i]$, $1 \le i \le n$, and $q[i]$, $0 \le i \le n$,
// this algorithm computes the cost $c[i,j]$, of
// optimal binary search trees t_{ij} for identifiers
// $a_{i+1}, \dots a_j$. It also computes $r[i,j]$, the root
// of t_{ij}. $w[i,j]$ is the weight of t_{ij}.

```
{
  for i:=0 to n-1 do
  {
     // Initialize
     w[i,i]:= q[i]; r[i,i]=0; c[i,i]=0;
     // optimal trees with one node
     w[i, i+1]:= q[i] + q[i+1] + P[i+1];
     r[i, i+1]:= i+1;
     c[i, i+1]:= q[i] + q[i+1] + P[i+1];
  }
  w[n,n]:= q[n]; r[n,n]:=0; c[n,n]:=0;
  for m:=2 to n do // Find optimal trees
                    // with m nodes.
   for i:=0 to n-m do
   {
      j:= i+m;
      w[i,j]:= w[i, j-1] + P[j] + q[j];
       k:= Find(c, r, i, j);
          // A value of l in the range
          // r[i, j-1] <= l <= r[i+1, j]
          // that minimizes c[i, l-1] + c[l, j];
      c[i,j]:= w[i,j] + c[i, k-1] + c[k, j]
      r[i,j]:= k;
   }
   write (c[0,n], w[0,n], r[0,n]);
}
```

```
Algorithm Find (c, r, i, j)
{ min := ∞;
  for m := r[i, j-1] to r[i+1, j] do
    if (c[i, m-1] + c[m, j]) < min then
    {
      min := c[i, m-1] + c[m, j];
        l := m;
    }
    return l;
}.
```

Compute $c(i,j)$ for $j-i = 1, 2, \ldots n$
when $j-i = m$