* Assignment No.: 9 *.

○ Title:-

Given sequence k = k1 < k2 <... kn n storted keys, with a search probability it for each key ki build the Binary search tree that the least search cost given the access probability for each key ?

○ objective:-

1) To understand concept of OBST
2) To understand concept & features like extends binary search tree.

○ learning outcome:

1) Define class for extends binary search tree using object oriented features
2) Analysis working of function.

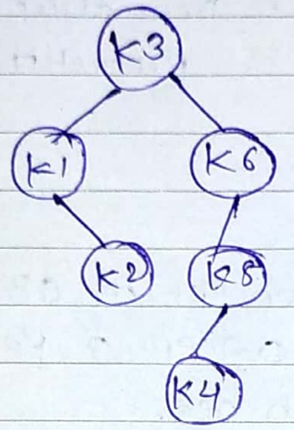
○ Theory :-

An optimal binary search tree is a binary search tree for which the nodes are arranged on levels.

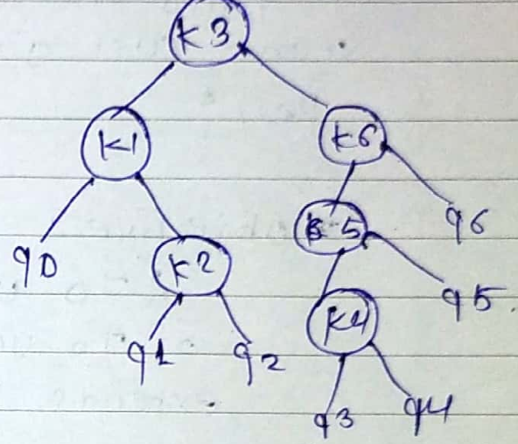
Such that the tree cost is minimum. for the purpose of a better presentation of optimal binary search tree, "which have the keys stored at their internal nodes suppose "n". key. k1 k2 . . . . kn are stored at the internal nodes of a storted order. so that. k1<k2 . . kn

An extended binary search tree is obtained from the binary search tree is obtained from the binary search tree by adding dussor nodes to each.

of its terminal nodes as indicates in the following by



Binary Search tree



failure node.

* **In the Extended tree-:**

i) The squores represents terminal nodes, These terminal node represent unsuccessful seacrches of the tree of key Volume. The searches did not end successful, that is because that represent key value that are not actually stored in the tree.

ii) The round node represent internal nodes. these are actually key stored in the tree.

iii) Assumming that the relative frequency with each key value is accesed is known weights can be assigned to each node of the extended tree (P1.P6). They represent terminating at each node,- that is, they mark the successful searches.

iv) If the user interface a perticular key in the case 2 can occur.

v) 1- the key is found. So the corresponding weight 'p' is incremented.

vi) 2- The key is not found. So the corresponding 'q' value is incremented.

* Generalization :-

The terminal node is the extended tree that is the left successor of k1 can be interrupted as representing all key values that are not stored and are less than k1. Similerly the terminal node in the extended tree that is right successor of kn, represents all key values not stored in the tree that are greater than kn. The terminal node that is successes between ki and ki-1 in an inorder traversal represent all key values not stored that lie between ki and ki-1

* Algorithm :-

we have the following procedures for determing R(i,j) & C(i,j) with 0 <= i <= j << n;

Procedure compute Root (n, p, q, R, c);
begin
for i:= 0 to n do
    c(i,i) ← 0.
  w(i,i) ← q(i)
for m = 0 to n do.
for i:= 0 to (n-m) do.

$j \leftarrow i+m$

$w(i,j) \leftarrow w(i,j-1)+p(j)+q(j).$

* find $c(i,j)$ and $R(i,j)$, which minimize the three cost

end

end.

The following functions builds an optimal binary search tree.

function constructor.(R, j, i).

begin

* build a new internal node N labeled (i,j)

if i=k then

* build a new leaf node 'N labeled (i,i)

else

N ← constructor (R,i, k).

* N is the left childe N labeled (j,i)

else

N ← Constructor (R, k+1, j)

N is the right child of node N

return N.

- Complexity analysis :-

The algorithm requirs $O(n^2)$ time and $O(n^2)$ storage. Therefore, as 'n' increases it will run out of storage even before it runs out of time.

- Input :

i) No. of element.

ii) key values.

iii) key probability.

* output:

Create binary search tree having optimal search cost.

- Example:

$w(w_1, w_2, w_3, w_4)$ = (do, if, ~~else~~ int, while).
$p(p_1, p_2, p_3, p_4)$ = (3, 3, 1, 1).
$q(q_0, q_1, q_2, q_3, q_4)$ = (2, 3, 1, 1, 1).

identifiers:

	0	1	2	3	4	5	6
0	d	o	\0				
1	i	f	\0				
2	i	n	t	\0			
3	w	h	i	l	e	\0	
4							
5							
6							

probability p

1	2	3	4	5
3	3	1	1	

failure q

0	1	2	3	4	5
2	3	1	1	1	

$w_{00} = 2$ $w_{11} = 3$ $w_{22} = 1$ $w_{33} = 1$ w_{44} 1
$c_{00} = 0$ $c_{11} = 0$ $c_{22} = 0$ $c_{33} = 0$ c_{44} 0
$r_{00} = 0$ $r_{11} = 0$ $r_{22} = 0$ $r_{33} = 0$ r_{44} 0.

$w[i][i] = q[i]$
$c[i][i] = r[i][i] = 0.$
$w[i][i+1] = q[i] + q[i+1] + p[i+1].$
$r[i][i+1] = i+1;$

$c[i][i+1] = q[i] + q[i+1] + p[i+1].$

$j-i = 1, \quad i=0, j=1.$
$w[i][i] = q[i] + q[i+1] + p[i+1].$
$w(0,1) = p(1) + q(1) + w(0,0).$
$= 3+3+2.$
$\underline{w(0,1) = 8}$
$c(i,j) = \min.$
$c[i][i+1] = q[i] + q[i+1] + p[i+1].$
$c[0][1] = q[0] + q[1] + p[1]$
$= 2,3+3.$

$\underline{\underline{c[0][1] = 8}}$
$\underline{\underline{r(0,1) = 1}}$

$i=1, j=2.$
$w[1][2] = q[1] + q[2] + p[2]$
$= 3+1+1$
$w(1,2) = 7$

$c(1)(2) = q[1] + q[2] + p[2].$
$= 3 + 1 + 3$
$c(1,2) = 7.$
$r(1,2) = 2$

$\underline{i=2, j=3}$

$i=2, j=3.$

$w(2,3) = q[2] + q[3] + p[3]$

$= 1+1+1$

$w(2,3) = \underline{\underline{3}}$

$c[2,3] = q[2] + q[3] + p[3]$

$= 1+1+1$

$\underline{\underline{c[2,3] = 3}}$.

$i=3, j=4 \quad w[3,4] = q[3] + q[4] + p[4]$

$= 1+1+1 = 3$

$c[3][4] = q[3] + q[4] + p[4].$

$= 1+1+1 = \underline{\underline{3}}.$

$\underline{\underline{r(3,4) = 4}}.$

$i=4, j=5.$

$w(4,5) = q[4] + p[5] + q[5]$

$= 1+0+0$

$r(4,5) = \underline{\underline{5}} \quad , w(4,5) = \underline{\underline{1}},$

		0	1	2	3	4	5	6..
	0	2	8	12	14	16		
	1		3	7	9	11		
	2			1	3	5		
w	3				1	3		
(weight)	4					1	1	
	5							
	6							

cost (c)

	0	1	2	3	4	5	6
0	0	8	19	25	22		
1		0	7	12	19		
2			0	3	8		
3				0	3		
4					0	1	
5							
6							

root (r).

	0	1	2	3	4	5	6
0	0	1	1	2	2		
1		0	2	2	2		
2			0	3	3		
3				0	4		
4					0	5	
5							
6							

* Conclusion:

This program gives us the knowledge OBST Extended binary search tree.