Assignment No. 8

Problem Definition:

Given sequence k = k1 <k2 < ... < kn of n sorted keys, with a search probability pi for each key ki . Build the Binary search tree that has the least search cost given the access probability for each key.

Theory:

A Binary Search Tree (BST) is a tree where the key values are stored in the internal nodes. The external nodes are null nodes. The keys are ordered lexicographically, i.e. for each internal node all the keys in the left sub-tree are less than the keys in the node, and all the keys in the right sub-tree are greater.

When we know the frequency of searching each one of the keys, it is quite easy to compute the expected cost of accessing each node in the tree. An optimal binary search tree is a BST, which has minimal expected cost of locating each node

Search time of an element in a BST is ***O(n)***, whereas in a Balanced-BST search time is ***O(log n)***. Again the search time can be improved in Optimal Cost Binary Search Tree, placing the most frequently used data in the root and closer to the root element, while placing the least frequently used data near leaves and in leaves.

Dynamic method

Let us understand OBST problem using the dynamic approach.

We can consider the table mentioned below with the frequencies and the keys.

	1	2	3	4
Keys(k)	10	20	30	40
Frequencies(pi)	4	2	6	3

Formula:-

1)w(i,i)=q(i)

2) c(i,i)=0

3) r(i,i)=0

4) w(i,j)=p(j)+q(j)+w(i,j-1)

5) C(i,j)= min {c(i,k-1)+c(k,j)}+w(i,j) ….. i<k<=j

6) r(i,j)=k

We will then go on to form a matrix with i and j,
Let us start by calculating the values for when j-i =0.
When j-i is 0,
i=0, j=0
i=1, j=1
i=2, j=2
i=3, j=3
i=4, j=4
Hence, c[0,0] = 0, and same for c[1,1], c[2,2], c[3,3], c[4,4].

Then, we calculate for when j-i=1,
When j-i=1,
i=0, j=1
i=1, j=2
i=2, j=3
i=3, j=4
Hence, c[0,1] = 4, c[1,2] = 2, c[2,3]= 6, c[3,4]=3.

Then, we calculate for when j-i=,
When j-i=2,
i=0, j=2
i=1, j=3
i=2, j=4
Hence, c[0,2] = 8.

Here, only two binary trees will be possible out of which cost of [0,2] is the smallest one, so we choose it.

Following the same method, we keep on testing for different values of j-i, and find the minimum cost in each case.

The general formula for finding the minimal cost is:

C[i,j] = min{c[i, k-1] + c[k,j]} + w(i,j)

Pseudocode for OBST

Optimal-Binary-Search-Tree(p, q, n)

C[1…n + 1, 0…n],
w[1…n + 1, 0…n],
root[1…n + 1, 0…n]

```
for i = 1 to n + 1 do
  C[i, i - 1] := q_i - 1
  w[i, i - 1] := q_i - 1

for l = 1 to n do
  for i = 1 to n – l + 1 do
    j = i + l – 1 c[i, j] := ∞
    w[i, i] := w[i, i -1] + p_j + q_j

   for r = i to j do
      t := c[i, r - 1] + c[r + 1, j] + w[i, j]
      if t < c[i, j]
        c[i, j] := t
        root[i, j] := r
return c and root
```

Conclusion: OBST for given keys can be constructed using dynamic programming approach.