O-1

Date:- / /
Page No.:-

01-1

Title :- Constructing an Optimal Binary Search tree using Dynamic Programming.

Problem Statement :- Given sequence $K = K_1 < K_2 < \dots < k_n$ of n sorted object/keys, witha search probability P_i for each key k_i. Build the Binary search tree that has the least search cost given the access probability for each key?

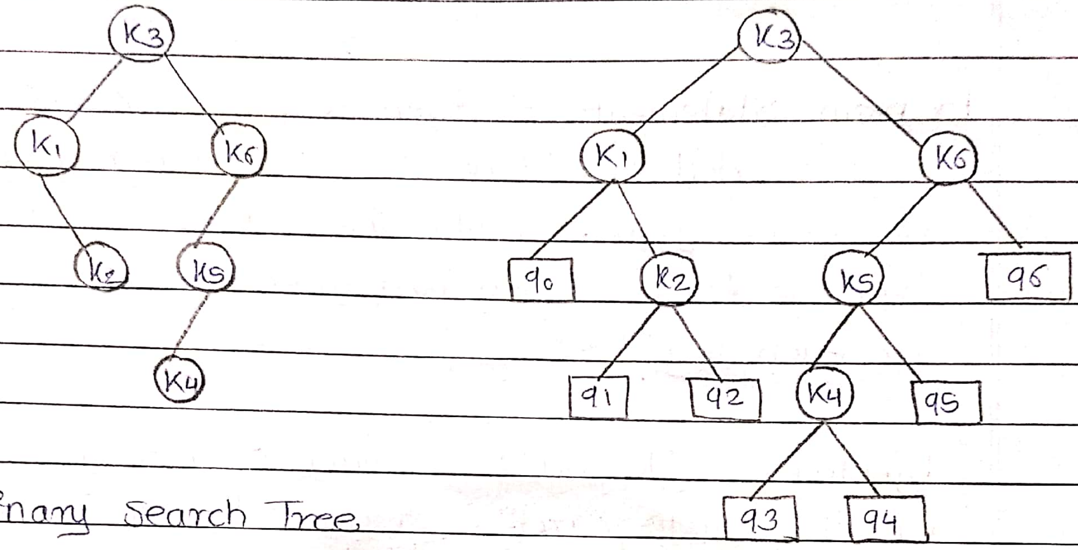
Objectives :- To build a binary search ~~thr~~ tree that has the least search cost given the access probability for each key?

Software Requirement :-

Theory :- An optimal binary search tree is binary search tree for which the nodes are arranged on level such that the tree cost is minimum.
For the purpose of a better presentation of optimal binary search trees, we will consider "extended binary search trees", which have the keys stored at their internal nodes. Suppose 'n' keys $k_1, k_2, \dots k_n$ are stored at the internal nodes of a binary search tree. It is assumed that the keys are given in sorted order, so that $k_1 < k_2 < \dots < k_n$.
An extended binary search tree is obtained from the binary search tree by adding successor nodes to each of its terminal nodes as indicated in the

Date:- / /
Page No.:-

Following figure by squares:



Binary Search Tree.

Extended Binary Search Tree.

- In the extended tree, the squares represent terminal nodes. These terminal nodes represent unsuccessful searches of the tree for key values. The searches did not end successfully, that is, because they represent key values that are not actually stored in the tree.
- The round nodes represent internal nodes, these are the actual keys stored in the tree.
- Assuming that the relative frequency with which each key value is known, weights can be assigned to each node of the extended tree (P_1 .... P_6). They represent the relative frequencis of searches terminating at each node, that is, they mark the successful searches.

Date:- / /
Page No.:-

01-2

If the user searches a particular key in the tree, 2 cases can occur.

1. The key is found, so the correspoind weight 'p' is increamented.
   ponding
2. The key is not found, so the corresponding weight 'q' is increamented.

Generalization:- The terminal node in the extended tree that is the left successor of k_1, can be interpreted as representing all key values that are not stored and are less than k_1. Similarly, the terminal node in the extended tree that is the right successor of k_n, represents all the key values not stored in the thr tree that are greater than k_n. The terminal node that is success between k_i and k_{i-1} in an inorder traversal represent all key values not stored that lie between k_i and k_{i-1}.

Algorithms :-

1. Start
2. Create an nxn matrix c, where c[i][j] represents the cost of the binary search tree containing only the keys k_i through k_j.
3. Initialize the diagonal entries of c to be the search probabilities for the corresponding keys, i.e. c[i][i] = p_i for all i.
4. For each chain length L = 1, 2, ..., n-1 and each sharing index i = 1, 2, ... n-L, compute the cost

of the binary search tree containing keys ki through Ki+L.

5. The minimum cost of the binary search tree containing all keys is c[1][n].

6. Stop.

The function returns the root node of the binary search tree and the minimum cost of the tree. The minimum cost may not be unique, as there may be multiple binary search trees with the same minimum cost.

Conclusion :- By using dynamic programming, we can construct an optimal binary search tree that has the least search ~~and~~ cost given the access probability for each key. This algorithm has a time complexity of $O(n^3)$ and space complexity $O(n^2)$.