# Modern Education Society's Wadia College of Engineering, Pune

## 210256: DATA STRUCTURES and ALGORITHM LABORATORY
## (2019 C0URSE)

| | |
|---|---|
| **NAME OF STUDENT:** | **CLASS:** |
| **SEMESTER/YEAR:** | **ROLL NO:** |
| **DATE OF PERFORMANCE:** | **DATE OF SUBMISSION:** |
| **EXAMINED BY:** | **EXPERIMENT NO: D18** |

**TITLE: Implementation of Optimal Binary Search Tree (OBST)**

**AIM/PROBLEM STATEMENT:** 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 ?

**OBJECTIVES:**

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

**OUTCOMES:**

1. Define class for Extended binary search tree using Object Oriented features.
2. Analyze working of functions.

**PRE-REQUISITES:**

1. Knowledge of C++ programming.
2. Knowledge of OBST

**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 trees, we will consider "extended binary search trees", which have the keys stored at their internal nodes. Suppose "n" keys k1, k2, … kn are stored at the internal nodes of a binary search tree. It is assumed that the keys are given in sorted order, so that k1< k2 < … < kn.

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 following figure by squares:
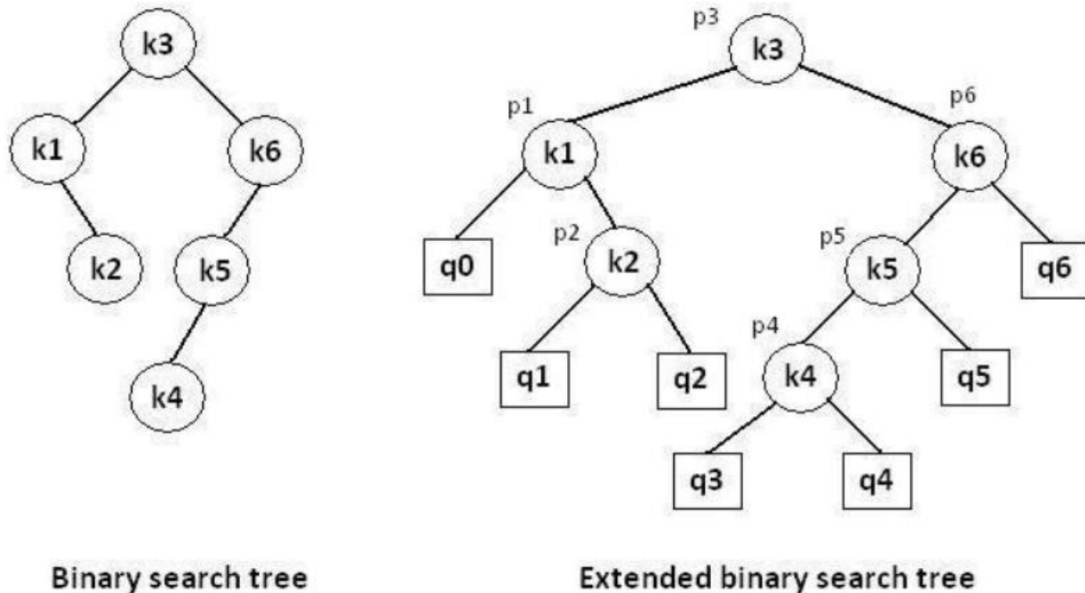


Binary search tree                    Extended binary search tree

**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 accessed is known, weights can be assigned to each node of the extended tree (p1 … p6). They represent the relative frequencies of searches terminating at each node, that is, they mark the successful searches.

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

1 – the key is found, so the corresponding weight „p" is incremented;

2 – the key is not found, so the corresponding „q" value is incremented.


**GENERALIZATION:**

The terminal node in the extended tree that is the left successor of k1 can be interpreted as representing all key values that are not stored and are less than k1. Similarly, the terminal node in the extended tree that is the 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.

**COMPLEXITY ANALYSIS:**

The algorithm requires $O(n^2)$ time and $O(n^2)$ storage. Therefore, as n increases it will run out of storage even before it runs out of time. The storage needed can be reduced by almost half by implementing the two-dimensional arrays as one-dimensional arrays.

**QUESTIONS:**

1. Find the optimal binary search tree for N = 6, having keys k1 … k6 and weights p1 = 10, p2 = 3, p3 = 9, p4 = 2, p5 = 0, p6 = 10; q0 = 5, q1 = 6, q2 = 4, q3 = 4, q4 = 3, q5 = 8, q6 = 0. The following figure shows the arrays as they would appear after the initialization and their final disposition.

W indicates weighted indicates

C indicates cost

R indicates root

| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| k | | 3 | 7 | 10 | 15 | 20 | 25 |
| p | - | 10 | 3 | 9 | 2 | 0 | 10 |
| q | 5 | 6 | 4 | 4 | 3 | 8 | 0 |

| R | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| 0 | | 1 | | | | | |
| 1 | | | 2 | | | | |
| 2 | | | | 3 | | | |
| 3 | | | | | 4 | | |
| 4 | | | | | | 5 | |
| 5 | | | | | | | 6 |
| 6 | | | | | | | |

| W | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| 0 | 5 | 21 | 28 | 41 | 46 | 54 | 64 |
| 1 | | 6 | 13 | 26 | 31 | 39 | 49 |
| 2 | | | 4 | 17 | 22 | 30 | 40 |
| 3 | | | | 4 | 9 | 17 | 27 |
| 4 | | | | | 3 | 11 | 21 |
| 5 | | | | | | 8 | 18 |
| 6 | | | | | | | 0 |

| C | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| 0 | | | | | | | |
| 1 | | | | | | | |
| 2 | | | | | | | |