Data Structure - Binary Search

Binary search is a fast search algorithm with run-time complexity of O(log n). This search algorithm works on the principle of divide and conquer. For this algorithm to work properly the data collection should be in sorted form.

Binary Search

Binary search search a particular item by comparing the middle most item of the collection. If match occurs then index of item is returned. If middle item is greater than item then item is searched in sub-array to the right of the middle item other wise item is search in sub-array to the left of the middle item. This process continues on sub-array as well until the size of subarray reduces to zero.

Binary Search

How binary search works? For a binary search to work, it is mandatory for the target array to be sorted. We shall learn the process of binary search with an pictorial example. The below given is our sorted array and assume that we need to search location of value 31 using binary search.
Binary search First, we shall determine the half of the array by using this formula
mid = low + (high - low) / 2 Here it is, 0 + (9 - 0) / 2 = 4 (integer value of 4.5). So 4 is the mid of array.

Binary Search

Binary search Now we compare the value stored at location 4, with the value being searched i.e. 31. We find that value at location 4 is 27, which is not a match. Because value is greater than 27 and we have a sorted array so we also know that target value must be in upper portion of the array.

Binary Search

Binary search We change our low to mid + 1 and find the new mid value again.
low = mid + 1 mid = low + (high - low) / 2 Our new mid is 7 now. We compare the value stored at location 7 with our target value 31.
Binary search The value stored at location 7 is not a match, rather it is less that what we are looking for. So the value must be in lower part from this location.

Binary Search

Binary search So we calculate the mid again. This time it is 5.
Binary search We compare the value stored ad location 5 with our target value. We find that it is a match.
Binary search We conclude that the target value 31 is stored at location 5.
Binary search halves the searchable items and thus reduces the count of comparisons to be made to very less numbers.

Data Structure - Binary Search Tree

Binary Search

A binary search tree (BST) is a tree in which all nodes follows the below mentioned properties

- The left sub-tree of a node has key less than or equal to its parent node's key.
- The right sub-tree of a node has key greater than or equal to its parent node's key.

Thus, a binary search tree (BST) divides all its sub-trees into two segments; left sub-tree and right sub-tree and can be defined as

$$left_subtree(keys) \leq node(key) \leq right_subtree(keys)$$

Representation of a Binary Search Tree

BST is a collection of nodes arranged in a way where they maintain BST properties. Each node has key and associated value. While searching, the desired key is compared to the keys in BST and if found, the associated value is retrieved.
An example of BST

Binary Search Tree We observe that the root node key (27) has all less-valued keys on the left sub-tree and higher valued keys on the right sub-tree.

Basic Operations Following are basic primary operations of a tree which are following.

Search search an element in a tree.

Insert insert an element in a tree.

Preorder Traversal traverse a tree in a preorder manner.

Inorder Traversal traverse a tree in an inorder manner.

Postorder Traversal traverse a tree in a postorder manner.

Node Define a node having some data, references to its left and right child nodes.

```
struct node {
int data;
struct node *leftChild;
struct node *rightChild;
};
```