

MIN AND MAX IN BST

To find the minimum and maximum value in a Binary Search tree we use a property called "inorder" traversal.

→ The first (leftmost bottom) element is the minimum.

→ The last (Rightmost bottom) element is the maximum.

→ Algorithm: (working of steps):

Step 1: Create a BST

Step 2: Sorting of BST in "inorder" method.

Step 3: visit each node in inorder traversal technique.

Step 4: The left most in the bottom is minimum.

Step 5: The Rightmost in the bottom is maximum.

→ Program: (C):

```
#include <stdio.h>
```

```
#include <limits.h>
```

```
#include <stdlib.h>
```

```
struct Node
```

```
{
```

```
    int data;
```

```
    struct Node* left, right;
```

```
};
```

```
int minValue(struct Node* root)
{
    if (root == NULL) {
        return (-1);
    }
    struct Node* curr = root;
    while (curr->left != NULL)
    {
        curr = curr->left;
    }
    return (curr->data);
}
```

```
int maxValue(struct Node* root)
{
    if (root == NULL)
    {
        return (-1);
    }
    struct Node* curr = root;
    while (curr->right != NULL)
    {
        curr = curr->right;
    }
    return (curr->data);
}
```

```
struct Node* createNode(int val)
{
    struct Node* node = (struct Node*) malloc (sizeof(struct Node));
    node->data = val;
    node->left = node->right = NULL;
    return (node);
}
```

}

int main()

{

struct Node* root = createNode(5);

root->left = createNode(4);

root->right = createNode(6);

root->left->left = createNode(3);

root->right->right = createNode(7);

root->left->left->left = createNode(1);

printf("%d\n", minValue(root));

printf("%d\n", maxValue(root));

return(0);

}

→ Output :

1

7

Complexity :

- Time = $O(h) \Rightarrow h$ - height
- Space = $O(1)$

→ Representation of BST :

