16010122828

Binary Search Tree- basic operations traversal and applications

1. **count number of nodes**

```c
#include <stdio.h>
#include <stdlib.h>
struct node {
    int val;
    struct node* left;
    struct node* right;
};
struct node* newNode(int val) {
    struct node* temp = (struct node*) malloc(sizeof(struct node));
    temp->val = val;
    temp->left = NULL;
    temp->right = NULL;
    return temp;
}
struct node* insert(struct node* root, int val) {
    if (root == NULL) return newNode(val);
    if (val < root->val)
        root->left  = insert(root->left, val);
    else if (val > root->val)
        root->right = insert(root->right, val);
    return root;
}
int countNodes(struct node* root) {
    if (root == NULL)
        return 0;
    return countNodes(root->left) + countNodes(root->right) + 1;
}
int main() {
    struct node* root = NULL;
    int n,i,val;
    printf("Enter the number of nodes:");
    scanf("%d",&n);
    printf("Enter the values:");
    for(i=0;i<n;i++){
        scanf("%d",&val);
        root=insert(root,val);
    }
    int nodeCount = countNodes(root);
    printf("Number of nodes: %d\n", nodeCount);
    return 0;
}
```

```
Output
/tmp/HLNGjYHEJN.o
Enter the number of nodes:4
Enter the values:45
56
12
23
Number of nodes: 4
```

2. compute height of tree

```c
#include <stdio.h>
#include <stdlib.h>
struct node {
    int val;
    struct node* left;
    struct node* right;
};
struct node* newNode(int val) {
    struct node* temp = (struct node*) malloc(sizeof(struct node));
    temp->val = val;
    temp->left = NULL;
    temp->right = NULL;
    return temp;
}
struct node* insert(struct node* root, int val) {
    if (root == NULL) return newNode(val);
    if (val < root->val)
        root->left  = insert(root->left, val);
    else if (val > root->val)
        root->right = insert(root->right, val);
    return root;
}
int height(struct node* root) {
    if (root == NULL)
        return 0;
    int leftHeight = height(root->left);
    int rightHeight = height(root->right);
    if (leftHeight > rightHeight)
        return leftHeight + 1;
    else
        return rightHeight + 1;
}
```

```c
int main() {
    struct node* root = NULL;
    int n,i,val;
    printf("Enter the number of nodes:");
    scanf("%d",&n);
    printf("Enter the values:");
    for(i=0;i<n;i++){
        scanf("%d",&val);
        root=insert(root,val);
    }
    int treeHeight = height(root);
    printf("Height of tree: %d\n", treeHeight);
    return 0;
}
```

Output

```
/tmp/HLNGjYHEJN.o
Enter the number of nodes:5
Enter the values:1
2
5
7
8
Height of tree: 5
```