二叉搜索树的操作集

1.函数接口定义

```
typedef struct TNode *Position;
typedef Position BinTree;
struct TNode{
    ElementType Data;
    BinTree Left;
    BinTree Right;
};

BinTree Insert( BinTree BST, ElementType X );
BinTree Delete( BinTree BST, ElementType X );
Position Find( BinTree BST, ElementType X );
Position FindMin( BinTree BST );
Position FindMax( BinTree BST );
```

- 函数Insert将X插入二叉搜索树BST并返回结果树的根结点指针；
- 函数Delete将X从二叉搜索树BST中删除，并返回结果树的根结点指针；- 如果X不在树中，则打印一行Not Found并返回原树的根结点指针；
- 函数Find在二叉搜索树BST中找到X，返回该结点的指针；如果找不到则- 返回空指针；
- 函数FindMin返回二叉搜索树BST中最小元结点的指针；
- 函数FindMax返回二叉搜索树BST中最大元结点的指针。

2.操作实现

```
BinTree Insert(BinTree BST, ElementType X)
{
    if (BST == NULL)
    {
        BST = (BinTree)malloc(sizeof(struct TNode));
        BST->Data = X;
        BST->Left = BST->Right = NULL;
    }
    else if (X < BST->Data)
        BST->Left = Insert(BST->Left, X);
    else if (X > BST->Data)
        BST->Right = Insert(BST->Right, X);
    return BST;
}

BinTree Delete(BinTree BST, ElementType X)
{
    Position Tmp;
    if (!BST)
        printf("Not Found\n");
    else
    {
        if (X < BST->Data)
            BST->Left = Delete(BST->Left, X);
        else if (X > BST->Data)
            BST->Right = Delete(BST->Right, X);
        else
        {
            if (BST->Left && BST->Right)
            {
                Tmp = FindMin(BST->Right);
                BST->Data = Tmp->Data;
                BST->Right = Delete(BST->Right, BST->Data);
            }
            else
            {
                Tmp = BST;
                if (!BST->Left)
                    BST = BST->Right;
                else
                    BST = BST->Left;
                free(Tmp);
            }
        }
    }
    return BST;
}

Position Find(BinTree BST, ElementType X)
{
    if (BST == NULL)
        return NULL;
    if (BST->Data == X)
        return BST;
    else if (X < BST->Data)
        return Find(BST->Left, X);
    else
        return Find(BST->Right, X);
}

Position FindMin(BinTree BST)
{
    if (BST)
        while (BST->Left)
            BST = BST->Left;
    return BST;
}

Position FindMax(BinTree BST)
{
    if (BST)
        while (BST->Right)
            BST = BST->Right;
    return BST;
```

```
}
```