二叉查找树类的定义

```
/*  二叉查找树结点类声明*/
template<class T>
class BSTNode
{
public:
        BSTNode<T> *llink ;
        BSTNode<T> *rlink ;
        T key ;
        BSTNode(const T & item,BSTNode<T> *lptr = NULL,BSTNode<T> *rptr = NULL): //  构造函数
              key(item), llink(lptr), rlink(rptr)     { }
} ;
/*  二叉查找树类 BSTree 声明*/
template <class T>
class BSTree
{
private:
       BSTNode<T>    *root;   //  指向二叉树根结点的指针
        T stop;                        //构造二叉树时，若输入 stop 则停止输入
public:
        BSTree(BSTNode<T> * t=NULL): root(t) { }        //构造函数
        BSTNode<T>* Search_Insert(T k);                 //二叉查找树的查找和插入
        void Delete(BSTNode<T>* q);                     //在树中删除 q 指向的结点
        void OptimalBST(int p[],int q[], int n );       //构造最优二叉查找树
        //其他操作
        BSTNode<T> *GetRoot() { return root ; }
        void SetRoot(BSTNode<T> * t) {    root=t ; }
        void CreateBSTree(T tostop) ;                   //  创建一棵二叉树
        void InOrder(BSTNode<T> *t)const ;              //  中根遍历并输出以结点 t 为根的子树
        BSTNode<T>* Father(BSTNode<T> *t,BSTNode<T> *p);
        T GetStop() { return stop ; }
        void SetStop(T tostop) {    stop=tostop ; }
} ;
```

算法 T（二叉查找树的查找与插入）

```
//本算法在二叉查找树中查找一个给定的变元 k，如果查找成功，则返回指向结点 k 的指针
//如果 k 不在树中，则在树的适当位置插入包含 k 的一个新结点，返回空。
template <class T>
BSTNode<T>*    BSTree<T>::Search_Insert(T k)
{
        if(root==NULL)          //如果树为空，则直接插入结点 k
        {
                root=new BSTNode<T>(k,NULL,NULL);
                return NULL;
        }
        BSTNode<T> *p=root;
        while(p!=NULL)
        {
                if(k==p->key)    return p;
                if(k<p->key)
                {
                        if(p->llink==NULL) break;
                        else p=p->llink;
                }
                else            //此时 k>p->key
                {
                        if(p->rlink==NULL) break;
                        else p=p->rlink;
                }
        }
        //至此，查找不成功，将包含关键词 k 的新结点插入树中
        BSTNode<T> *q=new BSTNode<T>(k,NULL,NULL);
        if(k<p->key)    p->llink=q;
        else    p->rlink=q;
        return NULL;
}
```