PTA | 程序设计类实验辅助教学平台
PROGRAMMING TEACHING ASSISTANT

声微丶饭否

< 题目集列表

题目集概况

题目列表

提交列表

排名

共 100 分

判断题(共 26 分) 13/13

✗ ✓ ✗ ✓ ✓ ✓ ✗
✓ ✓ ✓ ✓ ✓ ✓

A. 单选题(共 60 分) 20/20

✓ ✓ ✓ ✓ ✗ ✓ ✗
✓ ✓ ✗ ✓ ✓ ✓ ✗
✓ ✗ ✓ ✗ ✗ ✓

程序填空题(共 6 分) 1/1

✓

函数题(共 8 分) 0/1

✗

浙江大学2019-20学年春夏学期《高级数据结构与算法分析》课程期末考试试卷

判断题 13 | A. 单选题 20 | 程序填空题 1 | 函数题 1

5-1 The function `FindKey` is to check if a given `key` is in a B+ Tree with its root pointed by `root`. Return `true` if `key` is in the tree, or `false` if not. The B+ tree structure is defin
as following:

```
static int order = DEFAULT_ORDER;
typedef struct BpTreeNode BpTreeNode;
struct BpTreeNode {
    BpTreeNode** childrens;  /* Pointers to childrens. This field is not used by leaf nodes. */
    ElementType* keys;
    BpTreeNode* parent;
    bool isLeaf;  /* 1 if this node is a leaf, or 0 if not */
    int numKeys;  /* This field is used to keep track of the number of valid keys. In an internal node, the number of valid pointers is always numKeys + 1. *
};


bool FindKey(BpTreeNode * const root, ElementType key){
    if (root == NULL) {
            return false;
    }
    int i = 0;
    BpTreeNode * node = root;
    while ( !node->isLeaf (3分)) {
        i = 0;
        while (i < node->numKeys) {
            if ( key >= node->keys[i] (3分)) i++;
            else break;
        }
        node = node->childrens[i];
    }
    for(i = 0; i < node->numKeys; i++){
        if(node->keys[i] == key)
            return true;
    }
    return false;
}
```

5-1 答案正确 (6 分) 创建提问