

5-1 B+ Tree - Find Key 分数 4

作者 杨洋 单位 浙江大学

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 defined 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) 2分 ) {
        i = 0;
        while (i < node->numKeys) {
            if ( key >= node->keys[i] 2分 ) i++;
            else break;
        }
        node = node->childrens[i];
    }
    for(i = 0; i < node->numKeys; i++){
        if(node->keys[i] == key)
            return true;
    }
    return false;
}
```

答案正确: 4分

创建提问

5-2 Decode Count 分数 6

作者 陈越 单位 浙江大学

Suppose that a string of English letters is encoded into a string of numbers. To be more specific, `A-Z` are encoded into `0-25`. Since it is not a prefix code, the decoded result may not be unique. For example, `1213487` can be decoded as `BCBDEAH`, `MBDEAH`, `BCNEAH`, `BVDEAH` or `MNEAH`. Note that `07` is not `7`, hence cannot be decoded as `H`.

The function `DecodeCount` is supposed to return the number of different ways (modulo `BASE` to avoid overflow) we can decode `NumStr`, where `NumStr` is a string consisting of only the numbers `0-9`. Please complete the following program.

```
int DecodeCount( char NumStr[] )
{
    int L, i;
    int dp[MAXN]; //dp[i] is the solution from NumStr[i] to the end

    L = strlen(NumStr);
    if (L==0) return 0;
    if (L==1) return 1;
    dp[L-1] = 1;
    if (NumStr[L-2]!='1' && (NumStr[L-2]!='2' || NumStr[L-1]>'5'))
        dp[L-2] = 1;
    else dp[L-2] = 2;
    for (i=L-3; i>=0; i--) {
        if (NumStr[i]!='1' && (NumStr[i]!='2' || NumStr[i+1]>'5'))
            dp[i] = dp[i+1] + dp[i+2] 2分 ;
        else dp[i] = dp[i+1] 2分 ;
        dp[i] %= BASE; //to avoid overFlow
    }
    return dp[0] 2分 ;
}
```

部分正确: 2分

创建提问

https://pintia.cn/problem-sets/1513428827367215104/exam/problems/type/5#p-1513428827430129664

1/1