ProblemSet List

ProblemSet landing

Problem List

Submissions

Rank List

True-or-False (Total 35 score) 10/10

1 2 3 4 5 6 7 8 9 10

Multiple-Choice - 1 (Total 40 score) 8/8

1 2 3 4 5 6 7 8

Fill-in-Blank - P (Total 25 score) 2/2

1 2

ZJUADS_cy2019_MidTermExam

True-or-False 10 | Multiple-Choice - 1 8 | Fill-in-Blank - P 2

5-1 The functions `BinQueue_Find` and `Recur_Find` are to find `X` in a binomial queue `H`. Return the node pointer if found, otherwise return NULL.

Author: 陈越
Organization: 浙江大学
Time Limit: 400 ms
Memory Limit: 64 MB

```
BinTree BinQueue_Find( BinQueue H, ElementType X )
{
    BinTree T, result = NULL;
    int i, j;

    for( i=0, j=1; j<=H->CurrentSize; i++, j*=2) {  /* for each tree in H */
        T= H->TheTrees[i];
        if ( X >= T->Element (5分) ){  /* if need to search
            result = Recur_Find(T, X);
            if ( result != NULL ) return result;
        }
    }
    return result;
}

BinTree Recur_Find( BinTree T, ElementType X )
{
    BinTree result = NULL;
    if ( X==T->Element ) return T;
    if ( T->LeftChild!=NULL ){
        result = Recur_Find(T->LeftChild, X);
        if ( result!=NULL ) return result;
    }
    if ( T->NextSibling!=NULL (5分) )
        result = Recur_Find(T->NextSibling, X);
    return result;
}
```

5-2 The functions `IsRBT` is to check if a given binary search tree `T` is a red-black tree. Return `true` if `T` is, or `false` if not.

Author: 陈越
Organization: 浙江大学
Time Limit: 400 ms
Memory Limit: 64 MB

The red-black tree structure is defined as the following:

```
typedef enum { red, black } colors;
typedef struct RBNode *PtrToRBNode;
struct RBNode{
    int Data;
    PtrToRBNode Left, Right, Parent;
    int BlackHeight;
    colors Color;
};
typedef PtrToRBNode RBTree;
```

Please fill in the blanks.

```
bool IsRBT( RBTree T )
{
    int LeftBH, RightBH;
    if ( !T ) return true;
    if ( T->Color == black ) T->BlackHeight = 1;
    else {
         if ( T->Left && (T->Left->Color == red) (5分)) ret
         if ( T->Right && (T->Right->Color == red) ) return false;
    }
    if ( !T->Left && !T->Right ) return true;
    if ( IsRBT(T->Left) && IsRBT(T->Right) (5分)) {
       if ( T->Left ) LeftBH = T->Left->BlackHeight;
       else LeftBH = 0;
       if ( T->Right ) RightBH = T->Right->BlackHeight;
       else RightBH = 0;
       if ( LeftBH == RightBH ) {
          T->BlackHeight = LeftBH + (T->Color == bla (5分);
          return true;
       }
       else return false;
    }
    else return false;
}
```