ADS2021Midterm-bjj&yds

开始时间	2022/04/15 15:00:00	结束时间	2022/04/29 12:00:00	答题时长	45分钟
答卷类型	标准答案	总分	100		

判断题

得分：暂无　总分：30

1-1 For an AVL tree, the balance factors of all the non-leaf nodes are 0 iff the tree is a complete binary tree. (3分)

○ T ◉ F

1-2 While accessing a term stored in a B+ tree in an inverted file index, range searchings are expensive. (3分)

○ T ◉ F

1-3 The root of a B+ tree of order m has at most m subtrees. (3分)

◉ T ○ F

1-4 For one operation, if its amortized time bound is $O(logN)$, then its worst-case time bound must be $O(logN)$. (3分)

○ T ◉ F

1-5 In a red-black tree, if an internal black node is of degree 1, then it must have only 1 descendant node. (3分)

◉ T ○ F

1-6 To solve a problem by dynamic programming instead of recursions, the key approach is to store the results of computations for the subproblems so that we only have to compute each different subproblem once. Those solutions can be stored in an array or a hash table. (3分)

◉ T ○ F

1-7 To implement a binomial queue, left-child-next-sibling structure is used to represent each binomial tree. (3分)

◉ T ○ F

1-8 A perfectly balanced tree forms if keys 1 to 2^k-1 are inserted in order into an initally empty skew heap. (3分)

◉ T ○ F

1-9 For the recurrence equation $T(N)=aT(N/b)+f(N)$, if $af(N/b)=f(N)$, then $T(N)=\Theta(NlogbN)$. (3分)

○ T ◉ F

1-10 Finding the minimum key from a splay tree will result in a tree with its root having no left subtree. (3分)

◉ T ○ F

单选题

得分：暂无　总分：50

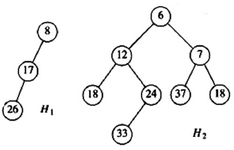
2-1 During inserting { 42, 26, 8, 70, 102, 56, 2} into an initially empty AVL tree, which of the following statements is true? (5分)

◉ A. The resulted AVL tree is also a completed binary tree;
○ B. There are 4 rotations: LL,RR,LR,LL
○ C. There are 4 rotations: LL,RR,LL,RL
○ D. There are 3 rotations: LL,RR,RL

2-2 A B+ tree of order 3 with 21 numbers has at most __ nodes of degree 2. (5分)

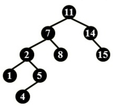
○ A. 1
○ B. 5
○ C. 6
◉ D. 7

2-3 Merge the two leftist heaps in the following figure. Which one of the following statements is FALSE? (5分)



○ A. 6 is the root with 7 being its right child
◉ B. 37 is the left child of 7
○ C. the depths of 24 and 8 are the same
○ D. the null path length of 7 is the same as that of 12

2-4 For the result of accessing the keys 5 and 8 in order in the splay tree in the following figure, which one of the following statements is FALSE? (5分)



○ A. 8 is the root
◉ B. 11 is the parent of 7
○ C. 2 and 7 are siblings
○ D. 5 is the parent of 7

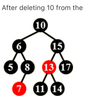
2-5 A queue can be implemented by using two stacks S_A and S_B as follows: (5分)

- To enqueue x, we push x onto S_A.
- To dequeue from the queue, we pop and return the top item from S_B. However, if S_B is empty, we first fill it (and empty S_A) by popping the top item from S_A, pushing this item onto S_B, and repeat until S_A is empty.

Assuming that push and pop operations take $O(1)$ worst-case time, please select a potential function ϕ which can help us prove that enqueue and dequeue operations take $O(1)$ amortized time (when starting from an empty queue).

◉ A. $\phi=2|S_A|$
○ B. $\phi=|S_A|$
○ C. $\phi=2|S_B|$
○ D. $\phi=|S_B|$

2-6 After deleting 10 from the red-black tree given in the figure, which one of the following statements must be FALSE? (5分)



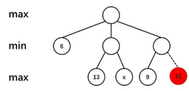
○ A. 11 is the parent of 6, and 14 is red
○ B. 8 is the parent of 15, and 7 is black
◉ C. 8 is the parent of 15, and there are 2 red nodes in the tree
○ D. 11 is the parent of 15, and there are 2 red nodes in the tree

2-7 There are 8000 documents in the database. The statistic data for one query are shown in the following table. The precision is: __ (5分)

	Relevant	Irrelevant
Retrieved	1000	1000
Not Retrieved	2000	4000

○ A. 12.5%
○ B. 20%
○ C. 33%
◉ D. 50%

2-8 Given the following game tree, the red node will be pruned with α-β pruning algorithm if and only if __. (5分)



○ A. $6\le x\le 13$
○ B. $x\ge 13$
○ C. $6\le x\le 9$
◉ D. $x\ge 9$

2-9 To solve the optimal binary search tree problem, we have the recursive equation $c_{ij}=\min_{i\le k\le j}\{w_{ij}+c_{i,k-1}+c_{k+1,j}\}$. To solve this equation in an interative way, we must fill up a table as follows: (5分)

○ A.
```
for i= 1 to n-1 do:
  for j= i to n do;
    for l= i to j do
```

○ B.
```
for j= 1 to n-1 do;
  for i= 1 to j do;
    for l= i to j do
```

◉ C.
```
for k= 1 to n-1 do;
  for i= 1 to n-k do;
    set j = i+k;
    for l= i to j do
```

○ D.
```
for k= 1 to n-1 do;
  for i= 1 to n do;
    set j = i+k;
    for l= i to j do
```

2-10 To solve a problem with input size N by divide and conquer, an algorithm divides the problem into 2 subproblems with size $\sqrt{N}$ (assuming it is an integer) and the time recurrences is (5分)

$$T(N)=2T(\sqrt{N})+\log(N).$$

What is the overall time complexity of this algorithm?

○ A. $O(\log(N))$
○ B. $O((\log(N))^2)$
◉ C. $O(\log(N)\log\log(N))$
○ D. $O(\sqrt{N}\log(N))$

程序填空题

得分：暂无　总分：20

5-1 IsRBTree (3)

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

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 BH; /* black height */
    colors Color;
};
typedef PtrToRBNode RBTree;
```

Please fill in the blanks.

```
bool IsRBTree( RBTree T )
{
    int LeftBH, RightBH;
    if ( !T ) return true;
    if ( T->Color == black ) T->BH = 1;
    else {
        if ( T->Left && (T->Left->Color == red)) return false;
        if ( T->Right && (T->Right->Color == red)  (3分) ) return false;
    }
    if ( !T->Left && !T->Right ) return true;
    if (  IsRBTree(T->Left) && IsRBTree(T->Right)  (4分)) {
        if ( T->Left ) LeftBH = T->Left->BH;
        else LeftBH = 0;
        if ( T->Right ) RightBH = T->Right->BH;
        else RightBH = 0;
        if ( LeftBH == RightBH ) {
            T->BH += LeftBH  (3分);
            return true;
        }
        else return false;
    }
    else return false;
}
```

5-2 The function `BinQueue_Merge` is to merge two binomial queues `H1` and `H2`, and return `H1` as the resulting queue.

```
BinQueue BinQueue_Merge( BinQueue H1, BinQueue H2 )
{   BinTree T1, T2, Carry = NULL;
    int i, j;
    H1->CurrentSize += H2-> CurrentSize;
    for ( i=0, j=1; j<= H1->CurrentSize; i++, j*=2 ) {
        T1 = H1->TheTrees[i]; T2 = H2->TheTrees[i];
        switch( 4*!!Carry + 2*!!T2 + !!T1 ) {
        case 0:
        case 1: break;
        case 2:  H1->TheTree[i] = T2; H2->TheTree[i] = NULL  (5分); break;
        case 4: H1->TheTrees[i] = Carry; Carry = NULL; break;
        case 3: Carry = CombineTrees( T1, T2 );
                H1->TheTrees[i] = H2->TheTrees[i] = NULL; break;
        case 5: Carry = CombineTrees( T1, Carry );
                H1->TheTrees[i] = NULL; break;
        case 6: Carry = CombineTrees( T2, Carry );
                H2->TheTrees[i] = NULL; break;
        case 7: H1->TheTrees[i] = Carry;
                Carry = CombineTrees( T1, T2 )  (5分);
                H2->TheTrees[i] = NULL; break;
        } /* end switch */
    } /* end for-loop */
    return H1;
}
```