剑指offer相关题目

- 链表相关题目：
  - 面试题5:输入一个链表的头节点，从尾到头反过来打印出每个节点的值。
    - 解法一：将每个链表中的数据放进栈中，之后出栈即可。

```
void ListToStack(LinkLinst &L,          LinkStack &S)
{
    LNode *p;
    p = L;
    while (p) {
        SNode *q = new SNode;
        q -> data = p -> data;
        q -> next = S;
        S = q;
        p = p -> next;
    }
}
```

    - 解法二：递归实现。（递归本质上就是一个栈结构）

```
void PrintListReversingly_Recursively(LinkLinst &L)
{
    if (L != NULL) {
        if (L -> next != NULL) {
PrintListReversingly_Recursively(L -> next);
        }
    cout << L -> data << endl;;
    }
}
```

    -完整代码：面试题五
  - 面试题13:给定单项链表的头指针和一个节点指针，定义一个函数在O(1)时间删除该节点。
    - 解法：假如我们想删除节点i，i的下一个节点是j。我们可以把节点j的内容复制给i，把i的next指向j的next。如果我们要删除的节点位于尾部，我们只能从

头节点开始，遍历到尾部的前一个节点，并删除尾部节点。如果只有一个节点我们就直接删除它。
    - 完整代码：面试题13
- 面试题15：输入一个链表，输出该链表中倒数第k个节点。本题从1开始计数，即链表的尾节点是倒数第一个节点。例如1、2、3、4、5、6。倒数第三个节点是4。
    - 解法：首先要清楚倒数第k个节点就是正数第n-k+1个节点。所以定义两个指针p，q。首先让p走到第k－1个节点，之后q从头出发，p继续往下走，当p走到尾部，q正好指向倒数第k个节点。
    - 注意：1.输入的链表是否为空。 2.节点个数是否少于k个。 3.k值不能为0.
    - 关键代码：

```
void FindKthToTail(LinkList &L, int k)
{
    if (L != NULL && k != 0) {
        LNode *p = L;
        LNode *q = L;
        for (int i = 0; i < k; i ++) {
            if (p!= NULL) {
                p = p -> next;
            }else{
                return;
            }
        }

        while (p) {
            q = q -> next;
            p = p -> next;
        }
        cout << q -> data;
    }
}
```

    - 完整代码：面试题15
- 面试题16:定义一个函数，输入一个链表的头结点，反转该链表并输出反转后链表。
    - 解法：定义三个指针，分别指向当前结点、当前结点的前一个结点、当前结点的后一个结点。
    - 关键代码：

```
LNode* ReverseLinkList(LinkList &L)
{
    LNode *p, *pPrev, *head, *pNext;
    p = L;
    pPrev = head = NULL;
    while (p) {
        pNext = p -> next;
        if (!pNext) {
            head = p;
        }
        p -> next = pPrev;
        pPrev = p;
        p = pNext;
    }
    return head;
}
```

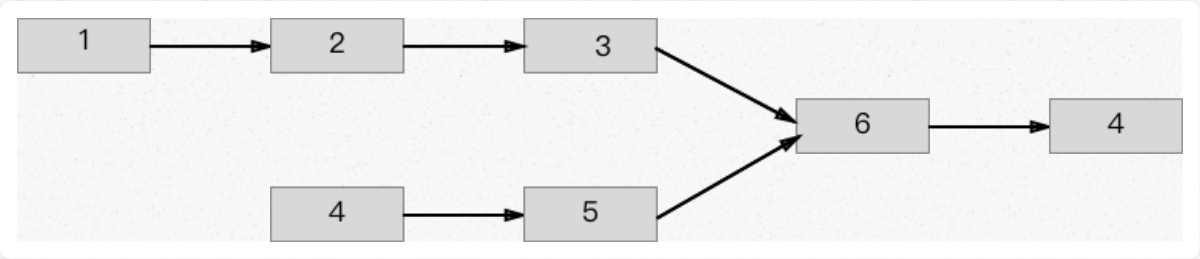
- 完整代码：面试题16

- 面试题17：输入两个递增的排序链表，合并这两个链表并使新链表中的结点依然是按照升序排列的。
  - 解法：两个链表中的头节点值较小的节点作为合并后连标的头节点。两个链表剩余的节点依旧是顺序的，所以可以把较小的节点连接到合并后的链表中。因为每次的步骤是一样的，所以可以使用递归。
  - 关键代码：

```
LNode * Merge(LNode *p, LNode *q)
{
    if (p == NULL) {
        return q;
    }else if (q == NULL){
        return p;
    }
    LNode *r;
    if (p -> data < q -> data) {
        r = p;
        r -> next = Merge(p -> next, q);
    }else{
        r = q;
        r -> next = Merge(p, q -> next);
    }
    return r;
}
```

- 完整代码：面试题17

- 面试题37:输入两个链表，找出它们的第一个公共节点。(如图)



  - 解题思路：首先遍历两个链表得到它们的长度，就能知道哪个链表长，长多少。第二次遍历的时候，先在较长的链表上先走若干步，接着再同时在两个链表上遍历，找到的第一个相同的节点就是它们的公共节点。

  - 主要算法

```
void FindFirstCommonNode(ListNode *p, ListNode *q)
{
int lengthList1 = GetListLength(p);
int lengthList2 = GetListLength(q);

int listDiff;
ListNode *listLong, *listShort;

if (lengthList1 > lengthList2) {
    listDiff = lengthList1 - lengthList2;
    listLong = p;
    listShort = q;
}else{
    listDiff = lengthList2 - lengthList1;
    listLong = q;
    listShort = p;
}

for (int i = 0; i < listDiff; i++) {
    listLong = listLong -> m_pNext;
}

while (listLong != NULL && listShort != NULL && listLong != lis
tShort) {
    listLong = listLong -> m_pNext;
    listShort = listShort -> m_pNext;
}

cout << listLong -> m_Key;

}
```

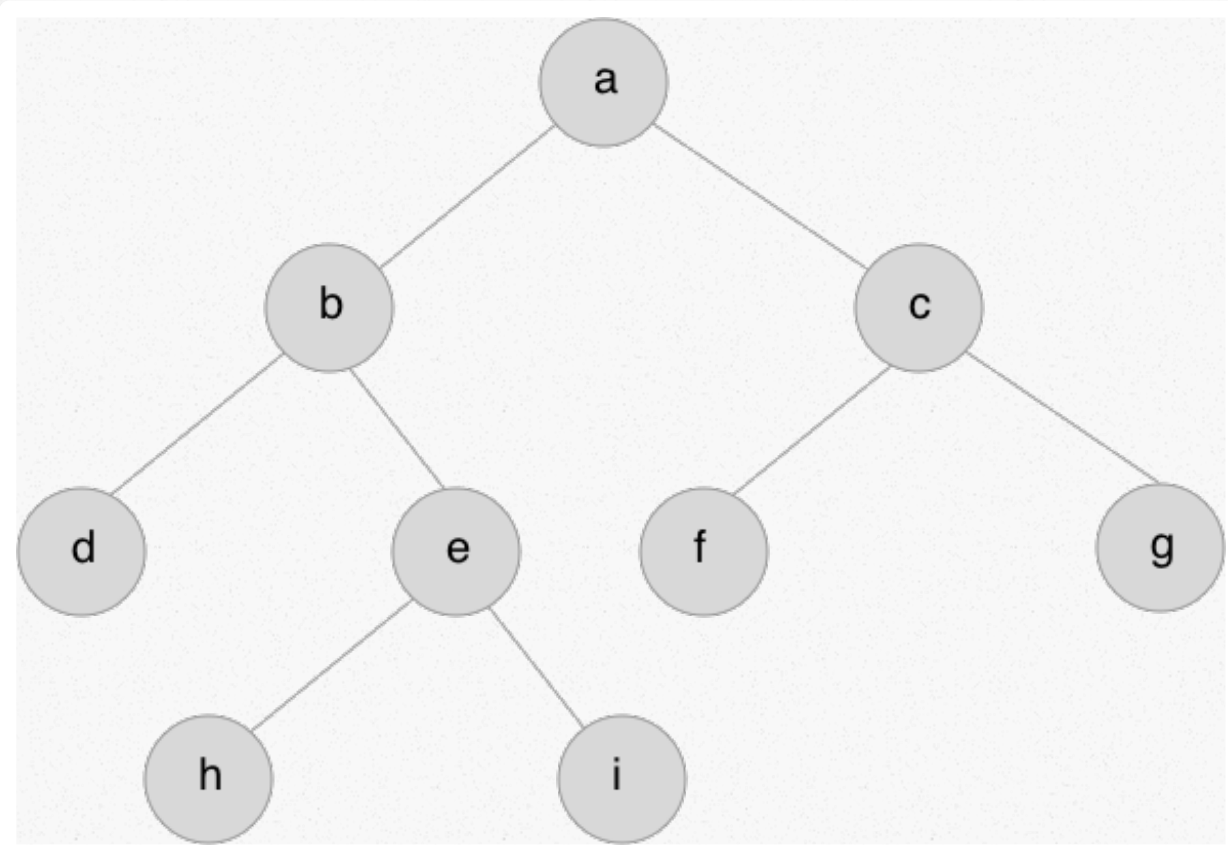
    - 完整代码：面试题37

- 面试题三
    - 题目：在一个二维数组中，每一行都按照从左到右递增的顺序排序，每一列都按照从上到下的顺序排序。请完成一个函数，输入这样的一个二维数组盒一个整数，判断数组中是否含有该整数。
    - 解题思路：首先选取数组中右上角的数字。如果该数字等于要查找的数字，结束查找；如果该数字大于要查找的数字，剔除该数字所在的列；如果该数字小于要查找的数字，剔除该数字所在的行。
    - 代码：

```
bool Find(int * matrix, int rows, int cloumns, int number)
{
    bool found = false;

    if (matrix != NULL && rows > 0 && cloumns > 0 ) {
        int row = 0;
        int cloumn = cloumns - 1;

        while (row < rows && cloumn >= 0) {
        if (matrix[row * cloumns + cloumn] == number) {
            found = true;
            break;
        }else if (matrix[row * cloumns + cloumn] > number)
            cloumn --;
        else
            row ++;
        }
    }

    return found;
}
```

    - 完整代码：面试题三

- 面试题58：给定一颗二叉树和其中的一个节点，如何找出中序遍历的下一个节点 。（中序遍历序列为：d、b、h、e、i、a、f、c、g）



- 解题思路:

1. 如果一个节点有右子树，那么它的下一个结点就是它的右子树的最左节点。例如：b的下一个节点是h。
2. 如果一个节点没有右子树，并且节点是它的父结点的左子结点，那么它的下一个节点就是它的父结点。例如：d的下一个结点是b。
3. 如果一个节点即没有右子树，并且还是它父结点的右子结点，我们可以沿着指向父结点的指针一直向上遍历，直到找到一个是它父结点的左子结点的结点。如果这样的结点存在，那么这个结点的父结点就是我们要找的下一个结点。例如：找i的下一个节点，i的父结点是e，但是它是b的右节点，所以我们继续向上遍历到b，b是它父结点e的左子结点，所以b为i的下一个节点；g的过程是类似的直到遍历的根节点，所以g没有下一个结点。

- 主要算法:

```
void getNet(BinaryTreeNode *node)
{
    if (node == NULL) {
        return;
    }
    BinaryTreeNode *nextNode = NULL;
    if (node -> Right != NULL) {
        BinaryTreeNode *right = node -> Right;
        while (right -> Left != NULL) {
```

```
            right = right -> Left;
        }
        nextNode = right;
        //cout << "the next node is " << right -> value << endl;
    }else if(node -> Parent != NULL){
        BinaryTreeNode *currentNode = node;
        BinaryTreeNode *parentNode = node -> Parent;
        while (parentNode != NULL && currentNode == parentNode ->
Right) {
            currentNode = parentNode;
            parentNode = parentNode -> Parent;
        }
        nextNode = parentNode;
       // cout << "the next node is " << parentNode -> value << en
dl;
    }
    if (nextNode) {
        cout << "tne next node is " << nextNode -> value << endl;
    }else{
        cout << "the next node is null" << endl;
    }

}
```

- 完整代码：面试题58