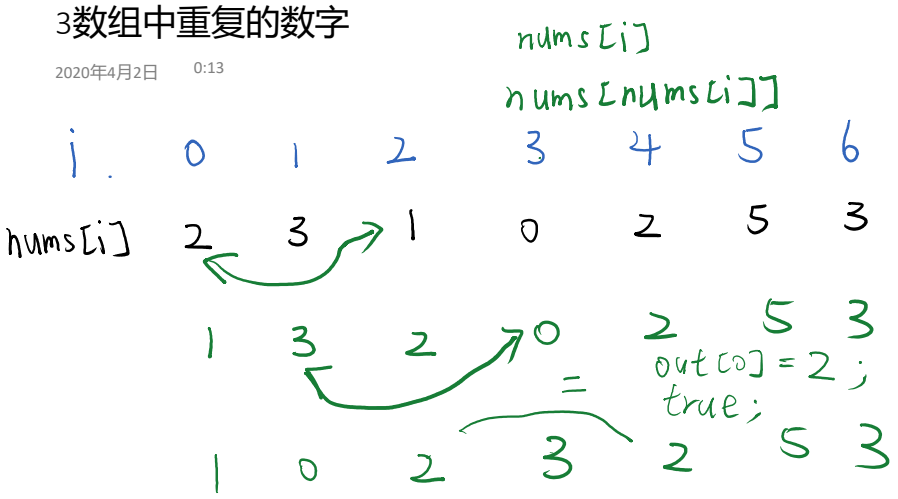
3数组中重复的数字

2020年4月2日 0:13

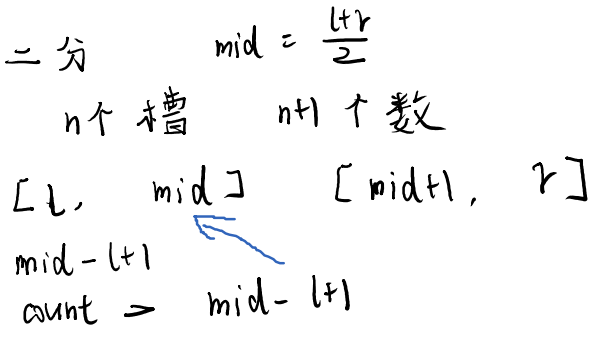



```
class Solution {
public:
    int duplicateInArray(vector<int>& nums) {
        int n = nums.size();
        //如果数组中某个元素超出范围 返回 -1
        for(auto x : nums)
            if(x < 0 || x >= n)
                return -1;

        for(int i = 0; i < n; i ++)
        {
            while(nums[i] != i && nums[nums[i]] != nums[i])
                swap(nums[i], nums[nums[i]]);
            if(nums[i] != i && nums[nums[i]] == nums[i])
                return nums[i];
        }


        // for(int i = 0; i < n; i ++)
        // {
        //     while (nums[i] != i) {
        //         if (nums[i] == nums[nums[i]])
        //             return nums[i];
        //         swap(nums[i], nums[nums[i]]);
        //     }
        // }

        return -1;//如果没有找到 也需要返回 -1
    }
};
```

```
class Solution {
public:
    // Parameters:
    //        numbers:     an array of integers
    //        length:      the length of array numbers
    //        duplication: (Output) the duplicated number in the array number
    // Return value:       true if the input is valid, and there are some duplications in the array number
    //                     otherwise false
    bool duplicate(int nums[], int n, int* out) {
        for(int i = 0; i < n; i ++)
        {
            while(nums[i] != i)
            {
                if(nums[i] == nums[nums[i]])
                {
                    out[0] = nums[i];
                    return true;
                }
                else
                    swap(nums[i], nums[nums[i]]);
            }
        }
        return false;
    }
};
```


不修改数组找出重复的数字

在一个长度为n+1的数组里的所有数字都在1~n的范围内，所以数组中至少有一个数字是重复的。从数组中找出任意一个重复的数字，但不能修改输入的数组。如{2,3,5,4,3,2,6,7}得到2或者3.




```
#include <iostream>
#include <vector>

using namespace std;

int duplicateInArray0(vector<int>& nums) {
    //暴力 n*n
    int n = nums.size();
    for(int i = 0; i < n - 1; i ++)
        for(int j = i + 1; j < n; j ++)
```

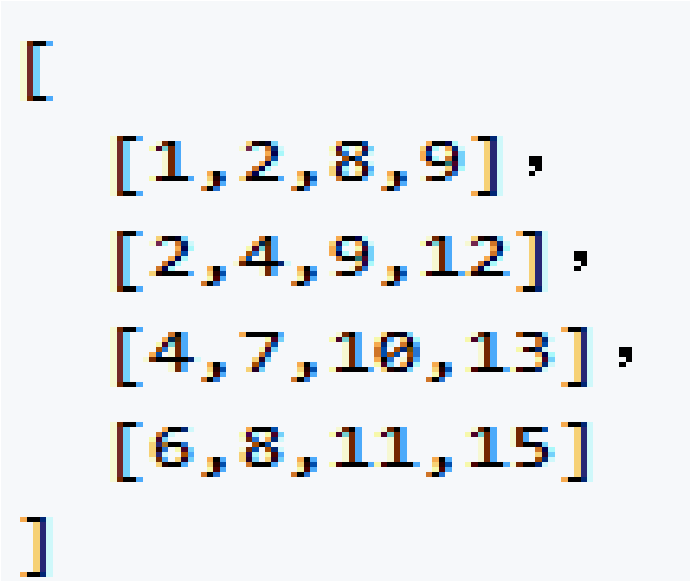
```
            if(nums[i] ^ nums[j] == 0)
                return nums[i];

        return 0;
}

int duplicateInArray(vector<int>& nums) {

    //二分 n* logn
    int l = 1, r = nums.size() - 1;

    while(l < r)
    {
        int mid = l + r >> 1; //[l, mid]  [mid+ 1, r]
        int count = 0;

        for(int i = 0; i < nums.size(); i ++)
        {
            if(nums[i] >= l && nums[i] <= mid)
                count ++;
        }

        if(count > mid - l + 1) r = mid;
        else l = mid + 1;
    }

    return l;
}

int main()
{
    int a[8] = {2, 3, 5, 4, 3, 2, 6, 7};
    vector<int> nums;
    //将a的所有元素插入到b中
    nums.insert(nums.begin(), a, a+9);
    cout <<"暴力解答：" << duplicateInArray(nums) <<endl;
    cout <<"二分解答："<<duplicateInArray(nums) <<endl;


    return 0;
}
```

4二维数组中的查找

2020年4月2日 1:18

题目描述

在一个二维数组中（每个一维数组的长度相同），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递增的顺序排序。请完成一个函数，输入这样的一个二维数组和一个整数，判断数组中是否含有该整数。




```
class Solution {
public:
    bool Find(int target, vector<vector<int> > array) {
        if(array.empty() || array[0].empty()) return false;

        int i = 0, j = array[0].size() - 1;
        while(i <= array.size() - 1  && j >= 0)
        {
            if(array[i][j] == target) return true;
            if(array[i][j] > target) j --;
            else i ++;
        }

        return false;
    }
};
```

5替换空格

2020年4月2日 1:42

请实现一个函数，将一个字符串中的每个空格替换成“%20”。例如，当字符串为We Are Happy.则经过替换之后的字符串为We%20Are%20Happy。

```
class Solution {
public:
        void replaceSpace(char *str,int length) {
    int count = 0;

    for(int i = 0; i < length; i ++)
    {
       if(str[i] == ' ')
          count ++;
    }
    // 1
    //%20 3
    for(int i = length - 1; i >= 0; i --)
    {
       if(str[i] != ' ')
          str[i + count * 2] = str[i];
       if(str[i] == ' ')
       {
          count --;
          str[i + count * 2] = '%';
          str[i + count * 2 + 1] = '2';
          str[i + count * 2 + 2] = '0';
       }
    }

  }
};
```

6从尾到头打印链表

2020年4月2日 2:27

题目描述

输入一个链表，按链表从尾到头的顺序返回一个ArrayList。

来自 <https://www.nowcoder.com/practice/d0267f7f55b3412ba93bd35cfa8e8035?tpId=13&tqId=11156&tPage=1&rp=1&ru=%2Fta%2Fcoding-interviews&qru=%2Fta%2Fcoding-interviews%2Fquestion-ranking>

```
/**
*  struct ListNode {
*        int val;
*        struct ListNode *next;
*        ListNode(int x) :
*              val(x), next(NULL) {
*        }
*  };
*/
class Solution {
public:
    vector<int> printListFromTailToHead(ListNode* head) {
        /*
        ListNode* p = head;
        stack<int> stk;
        vector<int> result;

        while(p != NULL)
        {
            stk.push(p -> val);
            p = p -> next;
        }

        int len = stk.size();
        for(int i = 0; i < len; i ++)
        {
            result.push_back(stk.top());
            stk.pop();
        }

        return result;
        */
        ListNode* p = head;
        vector<int> result;
        while(p != NULL)
        {
            result.push_back(p -> val);
            p = p -> next;
        }

        return vector<int>(result.rbegin(), result.rend());
    }
};
```

7重建二叉树

2020年4月2日 3:00

输入某二叉树的前序遍历和中序遍历的结果，请重建出该二叉树。假设输入的前序遍历和中序遍历的结果中都不含重复的数字。例如输入前序遍历序列{1,2,4,7,3,5,6,8}和中序遍历序列{4,7,2,1,5,3,8,6}，则重建二叉树并返回。

来自 <https://www.nowcoder.com/practice/8a19cbe657394eeaac2f6ea9b0f6fcf6?tpId=13&tqId=11157&tPage=1&rp=1&ru=/ta/coding-interviews&qru=/ta/coding-interviews/question-ranking>

```
给定：
前序遍历是：[3, 9, 20, 15, 7]
中序遍历是：[9, 3, 15, 20, 7]

返回：[3, 9, 20, null, null, 15, 7, null, null, null, null]
返回的二叉树如下所示：
    3
   / \
  9  20
    /  \
   15   7
```

```
/**
 * Definition for binary tree
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
 * };
 */
class Solution {
public:
    unordered_map<int,int> pos;
    TreeNode* reConstructBinaryTree(vector<int> pre,vector<int> vin) {
        int n = pre.size();
        for(int i = 0; i < n; i ++)
            pos[vin[i]] = i;

        return dfs(pre, vin, 0, n - 1, 0, n - 1);
    }

    TreeNode* dfs(vector<int> pre,vector<int> vin, int pl, int pr, int vl, int vr)
    {
        //前序 根左右  pre[pl]
        //中序 左根右
```

```
        if(pl > pr) return NULL;
        //找根节点
        TreeNode* root = new TreeNode(pre[pl]);
        //左子树的长度k
        int k = pos[pre[pl]] - vl;
        //vl + k 根节点
        root -> left = dfs(pre, vin, pl + 1, pl + k, vl, vl + k - 1);
        root -> right = dfs(pre, vin,pl + k + 1, pr, vl + k + 1, vr);

        return root;
    }

};
```

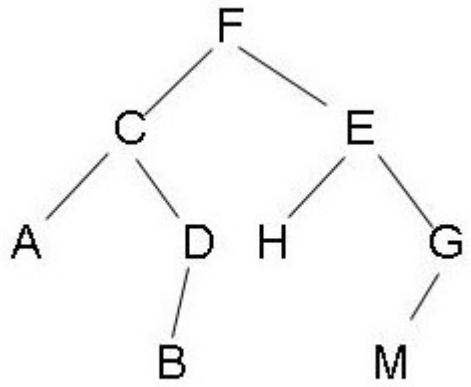
8二叉树的下一个节点

2020年4月2日 3:00

题目描述

给定一个二叉树和其中的一个结点，请找出中序遍历顺序的下一个结点并且返回。注意，树中的结点不仅包含左右子结点，同时包含指向父结点的指针。

来自 <https://www.nowcoder.com/practice/9023a0c988684a53960365b889ceaf5e?tpId=13&tqId=11210&tPage=1&rp=1&ru=/ta/coding-interviews&qru=/ta/coding-interviews/question-ranking>




```
/*
struct TreeLinkNode {
    int val;
    struct TreeLinkNode *left;
    struct TreeLinkNode *right;
    struct TreeLinkNode *next;
    TreeLinkNode(int x) :val(x), left(NULL), right(NULL), next(NULL) {

    }
};
*/
class Solution {
public:
    TreeLinkNode* GetNext(TreeLinkNode* p)
    {
        //右子树存在 右子树最左边的结点
        if(p -> right)
        {
            p = p -> right;
            while(p -> left) p = p -> left;
            return p;
        }

        //右子树不存在 只有左子树
        while(p -> next)
        {
```

```
            //p不是根节点
            if(p == p -> next -> left)
                return p -> next;
            p = p -> next;
        }

        return NULL;
    }
};
```

9用两个栈实现队列

2020年4月2日 3:00

题目描述

用两个栈来实现一个队列，完成队列的Push和Pop操作。 队列中的元素为int类型。

来自 <https://www.nowcoder.com/practice/54275ddae22f475981afa2244dd448c6?tpId=13&tqId=11158&tPage=1&rp=1&ru=%2Fta%2Fcoding-interviews&qru=%2Fta%2Fcoding-interviews%2Fquestion-ranking>

```
class Solution
{
public:
    //先进后出 杯子 栈
    //先进先出 管道 队列


    void push(int node) {
        stack1.push(node);
    }

    void copy(stack<int> &a, stack<int> &b)
    {
        while(a.size())
        {
            b.push(a.top());
            a.pop();
        }
    }

    int pop() {
        //1 ——》 2
        // 2 top  pop
        // 2 ——》 1
        copy(stack1, stack2);
        int res = stack2.top();
        stack2.pop();
        copy(stack2, stack1);

        return res;
    }

private:
    stack<int> stack1;
    stack<int> stack2;
};
```

```
class MyQueue {
public:

    stack<int> stack1, stack2;
    /** Initialize your data structure here. */
    MyQueue() {

    }

    void copy(stack<int> &a, stack<int> &b)
    {
        while(a.size())
        {
            b.push(a.top());
            a.pop();
        }
    }

    /** Push element x to the back of queue. */
    void push(int x) {
        stack1.push(x);
    }

    /** Removes the element from in front of queue and returns that element. */
    int pop() {
        //1 ——》 2
        // 2 top  pop
        // 2 ——》 1
        copy(stack1, stack2);
        int res = stack2.top();
        stack2.pop();
        copy(stack2, stack1);

        return res;
    }

    /** Get the front element. */
    int peek() {
        copy(stack1, stack2);
        int res = stack2.top();
        copy(stack2, stack1);

        return res;
    }

    /** Returns whether the queue is empty. */
    bool empty() {
        return stack1.empty();
    }
};

/**
 * Your MyQueue object will be instantiated and called as such:
 * MyQueue obj = MyQueue();
 * obj.push(x);
 * int param_2 = obj.pop();
 * int param_3 = obj.peek();
 * bool param_4 = obj.empty();
 */
```

作者：满血小进

链接：https://www.acwing.com/activity/content/code/content/261715/

来源：AcWing

10斐波那契数列

2020年4月2日 3:21

大家都知道斐波那契数列，现在要求输入一个整数n，请你输出斐波那契数列的第n项（从0开始，第0项为0）。

n<=39

来自 <https://www.nowcoder.com/practice/c6c7742f5ba7442aada113136ddea0c3?tpId=13&tqId=11160&tPage=1&rp=1&ru=%2Fta%2Fcoding-interviews&qru=%2Fta%2Fcoding-interviews%2Fquestion-ranking>

```
class Solution {
public:
    int Fibonacci(int n) {
        if(n == 0) return 0;
        if(n == 1) return 1;
        int first = 0, second = 1;
        int res = 0;
        for(int i = 2; i <= n; i ++)
        {
            res = first + second;
            first = second;
            second = res;
        }

        return res;
    }
};
```

11旋转数组的最小数字

2020年4月2日 3:39

```
class Solution {
public:
    int minNumberInRotateArray(vector<int> nums) {
        int n = nums.size() - 1;
        if(n < 0) return 0;

        while(nums[n] == nums[0] && n > 0) n --;
        if(nums[n] >= nums[0]) return nums[0];

        int l = 0, r = n;
        while(l < r)
        {
            int mid = l + r >> 1; //[l, mid] [mid+1, r]
            if(nums[mid] < nums[0]) r = mid;
            else l = mid + 1;
        }
        return nums[r];
    }
};
```

12矩阵中的路径

2020年4月2日 3:48

题目描述

请设计一个函数，用来判断在一个矩阵中是否存在一条包含某字符串所有字符的路径。路径可以从矩阵中的任意一个格子开始，每一步可以在矩阵中向左，向右，向上，向下移动一个格子。如果一条路径经过了矩阵中的某一个格子，则该路径不能再进入该格子。 例如 $\begin{bmatrix} a & b & c & e \\ s & f & c & s \\ a & d & e & e \end{bmatrix}$ 矩阵中包含一条字符串"bcced"的路径，但是矩阵中不包含"abcb"路径，因为字符串的第一个字符b占据了矩阵中的第一行第二个格子之后，路径不能再次进入该格子。

```
class Solution {
public:
    bool hasPath(char* matrix, int rows, int cols, char* str)
    {
        if(rows == 1 && cols == 1)
            if(matrix[0] == str[0]) return true;
            else return false;

        //matrix 一维
        //str 目标字符串  str  \0
        for(int i = 0; i < rows; i ++)
            for(int j = 0; j < cols; j ++)
                if(dfs(matrix, rows, cols, str, 0, i, j))
                    return true;

        return false;
    }

    bool dfs(char* matrix, int rows, int cols, char* str, int u, int x, int y)
    {
        if(str[u] == '\0') return true;

        int dx[4] = {-1, 0, 1,0}, dy[4] = {0, 1, 0, -1};
        for(int i = 0; i < 4; i ++)
        {
            int a = x + dx[i], b = y + dy[i];
            if(a >= 0 && a < rows && b >= 0 && b < cols && matrix[a * cols + b] == str[u])
            {
                char t = matrix[a * cols + b];
                matrix[a * cols + b] = '*';
                if(dfs(matrix, rows, cols, str, u + 1, a, b))
                    return true;
                matrix[a * cols + b] = t;
            }
        }
```

```
        return false;
    }


};
```

13机器人的运动范围

2020年4月3日 13:59

题目描述

地上有一个m行和n列的方格。一个机器人从坐标0,0的格子开始移动，每一次只能向左，右，上，下四个方向移动一格，但是不能进入行坐标和列坐标的数位之和大于k的格子。 例如，当k为18时，机器人能够进入方格（35,37），因为3+5+3+7 = 18。但是，它不能进入方格（35,38），因为3+5+3+8 = 19。请问该机器人能够达到多少个格子?

来自 <https://www.nowcoder.com/practice/6e5207314b5241fb83f2329e89fdecc8?tpId=13&tqId=11219&tPage=1&rp=1&ru=/ta/coding-interviews&qru=/ta/coding-interviews/question-ranking>

```
class Solution {
public:

    int get_sum(pair<int, int> p)
    {
        int s = 0;
        while(p.first)
        {
            s += p.first % 10;
            p.first /= 10;
        }
        while(p.second)
        {
            s += p.second % 10;
            p.second /= 10;
        }

        return s;
    }

    int movingCount(int threshold, int rows, int cols)
    {
        int res = 0;

        if(!cols || !rows) return 0;

        queue<pair<int, int>> q;
        vector<vector<bool>> st(rows, vector<bool>(cols, false));

        //bfs
        int dx[4] = {-1, 0, 1, 0}, dy[4] = {0, 1, 0, -1};
        q.push({0, 0});
```

```
        //bfs
        while(q.size())
        {
            pair<int, int> t = q.front();
            q.pop();

            if(st[t.first][t.second] || get_sum(t) > threshold) continue;

            res ++;
            st[t.first][t.second] = true;

            for(int i = 0; i < 4; i ++)
            {
                int x = t.first + dx[i], y = t.second + dy[i];
                if(x >= 0 && x< rows && y >= 0 && y < cols) q.push({x, y});
            }
        }

        return res;
    }
};
```

14剪绳子

2020年4月3日 15:53

题目描述

给你一根长度为n的绳子，请把绳子剪成整数长的m段（m、n都是整数，n>1并且m>1），每段绳子的长度记为k[0],k[1],...,k[m]。请问k[0]xk[1]x...xk[m]可能的最大乘积是多少？例如，当绳子的长度是8时，我们把它剪成长度分别为2、3、3的三段，此时得到的最大乘积是18。

输入描述:

输入一个数n，意义见题面。 (2 <= n <= 60)

输出描述:

输出答案。

示例1

输入

复制

```
8
```

输出

复制

```
18
```

来自 <https://www.nowcoder.com/practice/57d85990ba5b440ab888fc72b0751bf8?tpId=13&tqId=33257&tPage=1&rp=1&ru=/ta/coding-interviews&qru=/ta/coding-interviews/question-ranking>

```
class Solution {
public:
    int cutRope(int n) {
        //数学
        //子绳子
        // >=5
        //ni    3 * (ni - 3) = 3*ni - 9 >  ni;    2*ni > 9
        // == 4
        // 2  2   1  3
        // 3 > 1 *2
        // 2 > 1*1
        /*
        if(n <= 3) return n - 1;

        int res = 1;
        if(n % 3 == 1) res *= 4, n -= 4;
        else if(n % 3 == 2) res *= 2, n -= 2;
        while(n)
        {
            res *= 3;
            n -= 3;
        }
```

```
        return res;
        */

        //动态规划
        if(n <= 3) return n - 1;

        int dp[n];
        dp[1] = 1;
        dp[2] = 2;
        dp[3] = 3;
        int res = 0;//记录最大的乘积
        for(int i = 4; i <= n ; i ++)
        {
           for(int j = 1; j <= i/2; j ++)
              res = max(res, dp[j] * dp[i - j]);
           dp[i] = res;
        }

        return dp[n];
    }
};
```

15二进制中1的个数

2020年4月3日 16:14

输入一个整数，输出该数二进制表示中1的个数。其中负数用补码表示。

来自 <https://www.nowcoder.com/practice/8ee967e43c2c4ec193b040ea7fbb10b8?tpId=13&tqId=11164&tPage=1&rp=1&ru=/ta/coding-interviews&qru=/ta/coding-interviews/question-ranking>

```
class Solution {
public:
    int  NumberOf1(int n) {

        int count = 0;
        while(n)
        {
            count ++;
            n = n & (n - 1);
        }
        return count;
    }
};
```

16数值的整数次方

2020年4月3日 16:44

给定一个double类型的浮点数base和int类型的整数exponent。求base的exponent次方。
保证base和exponent不同时为0

来自 <https://www.nowcoder.com/practice/1a834e5e3e1a4b7ba251417554e07c00?tpId=13&tqId=11165&tPage=1&rp=1&ru=/ta/coding-interviews&qru=/ta/coding-interviews/question-ranking>

```
class Solution {
public:
    double Power(double base, int exponent) {
        double res = 1;
        for(int i = 0; i < abs(exponent); i ++) res *= base;
        if(exponent < 0) res = 1/ res;
        return res;
    }
};
```

17打印从1到最大的n位数

2020年4月3日　　16:49

输入数字 n，按顺序打印出从 1 到最大的 n 位十进制数。比如输入 3，则打印出 1、2、3 一直到最大的 3 位数 999。

示例 1:

输入: n = 1
输出: [1,2,3,4,5,6,7,8,9]

说明：

用返回一个整数列表来代替打印
n 为正整数

来源：力扣（LeetCode）
链接：https://leetcode-cn.com/problems/da-yin-cong-1dao-zui-da-de-nwei-shu-lcof
著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。

打印从1到最大的n位数

> 输入数字 n，按顺序打印出从 1 到最大的 n 位十进制数。比如输入 3，则打印出 1、2、3 一直到最大的 3 位数 999。

n*n*可能很大的话，就采用大数加法的写法。
有一个注意的地方，就是判断是否到最大值那里，如果一直用strcmp来与最大值进行比较，每次比较的复杂度都是O(n)*O*(*n*)的，这样不可取。
有两种方法优化：

- 如果i=0*i*=0的时候还有进位，那就到达最大值了
- 可以算出总共有多少个数，然后for就行

```
class Solution {
public:
    vector<int> printNumbers(int n) {
        char *number = new char[n+1];
        vector<int> ans;
        memset(number,'0',n);
        number[n] = '\0';
        while(!checkNumber(number,n)){
            ans.push_back(changeNumber(number,n));
        }
        return ans;
```

```
    }
    int changeNumber(char *number,int n){
        int res = 0;
        for(int i = 0;i < n;++i){
            res = res*10 + number[i]-'0';
        }
        return res;
    }
    bool checkNumber(char *number,int n){
        int nLength = n;
        bool isOverFlow = 0;
        int isTakeOver=1;
        for(int i = nLength - 1;i >= 0;--i){
            number[i] = number[i] + isTakeOver;
            if(number[i] > '9'){
                if(i == 0) {
                    isOverFlow = 1;
                    break;
                }
                number[i] = '0';
                isTakeOver = 1;
            }else{
                isTakeOver = 0;
            }
        }
        return isOverFlow;
    }
};
```

还有一种全排列的写法：

```
class Solution {

public:
    vector<int> printNumbers(int n) {
        char *str = new char[n+1];
        vector<int>ans;
        dfsChooseNum(0,n,str,ans);
        return ans;
    }
    void dfsChooseNum(int pos,const int& n,char* now,vector<int>& ans){
        if(pos == n){
            int res = 0;
            for(int i = 0;i < n;++i){
                res = res*10 + now[i] - '0';
            }
            if(res) ans.push_back(res);
```

```
17            return;
18        }
19        for(int i=0;i<=9;++i){
20            now[pos] = '0' + i;
21            dfsChooseNum(pos+1,n,now,ans);
22        }
23    }
24 };
```

来自 <https://blog.nowcoder.net/n/08fe46d509a3435ab4e46ac164357d42#打印从1到最大的n位数>

18在O(1)时间删除链表结点

2020年4月7日 0:53

给定单向链表的一个节点指针，定义一个函数在O(1)时间删除该结点。

假设链表一定存在，并且该节点一定不是尾节点。

样例

输入：链表 1->4->6->8

删掉节点：第2个节点即6（头节点为第0个节点）

输出：新链表 1->4->8

来自 <https://www.acwing.com/problem/content/85/>

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode(int x) : val(x), next(NULL) {}
 * };
 */
class Solution {
public:
    void deleteNode(ListNode* node) {
        //一般 前驱next指针 = 当前next指针
        //没有前驱结点，当前结点=下一个结点 然后删除下一个结点
        node -> val = node -> next -> val;
        node -> next = node -> next -> next;

    }
};
```

18删除链表中重复的节点

2020年4月7日 12:00

在一个排序的链表中，存在重复的结点，请删除该链表中重复的结点，重复的结点不保留。

样例1

输入：1->2->3->3->4->4->5

输出：1->2->5

样例2

输入：1->1->1->2->3

输出：2->3

来自 <https://www.acwing.com/problem/content/27/>

```
/*
struct ListNode {
    int val;
    struct ListNode *next;
    ListNode(int x) :
        val(x), next(NULL) {
    }
};
*/
class Solution {
public:
    ListNode* deleteDuplication(ListNode* head)
    {
        auto dummy = new ListNode(-1);
        dummy -> next = head;

        auto p = dummy;
        while(p -> next)
        {
            auto q = p -> next;
            while(q && p -> next -> val == q -> val) q = q -> next;

            if(p -> next -> next == q) p = p -> next;
            else p -> next = q;
        }

        return dummy -> next;
    }
};
```

19正则表达式匹配

2020年4月7日　　0:53

请实现一个函数用来匹配包括'.'和'*'的正则表达式。模式中的字符'.'表示任意一个字符，而'*'表示它前面的字符可以出现任意次（包含0次）。 在本题中，匹配是指字符串的所有字符匹配整个模式。例如，字符串"aaa"与模式"a.a"和"ab*ac*a"匹配，但是与"aa.a"和"ab*a"均不匹配

来自 <https://www.nowcoder.com/profile/2983379/wrongset/283780110?page=1&offset=4&tags=>

```
class Solution {
public:
    bool match(char* str, char* patten)
    {
        //两者都为空 true
        if(*str == '\0' && *patten == '\0')
            return true;
        //str不空 patten空
        if(*str != '\0' && *patten == '\0')
            return false;
        //str空 patten不空
        //        a* *:0
        //*:0...

        if(*(patten + 1) != '*')//匹配当前字符
        {
            if(*str == *patten || (*str != '\0' && *patten =='.'))   //str : any   patten :.
                return match(str + 1, patten + 1);
            else
                return false;
        }
        else//patten +1 == *
        {
            if(*str == *patten || (*str != '\0' && *patten =='.'))
                return match(str, patten + 2) || match(str + 1, patten);
                    // aa   a*aa
                    //aaa  .*a
            else//aa   c*aa
                return match(str, patten + 2);
        }

    }
};
```

来自 <https://www.nowcoder.com/profile/2983379/wrongset/283780110?page=1&offset=4&tags=>

20表示数值的字符串

2020年4月7日 3:10

请实现一个函数用来判断字符串是否表示数值（包括整数和小数）。例如，字符串"+100","5e2","-123","3.1416"和"-1E-16"都表示数值。 但是"12e","1a3.14","1.2.3","+-5"和"12e+4.3"都不是。

来自 <https://www.nowcoder.com/profile/2983379/wrongset/283782461?page=1&offset=3&tags=>

```
class Solution {
public:
    bool isNumeric(char* string)
    {
        /*思路
        1. 连续..  no
        2.  e. no
        3. 连续 ee  EE   no
        4. e '\0'  no
        */
        if(string == NULL)
            return false;
        if(*string == '-' || *string == '+')
        {
            string ++;
            if(*string == '-' || *string == '+')
                return false; //+-15
        }
        //+-
        if(*string == '\0')
            return false;

        int dot = 0, e = 0, num = 0;
        while(*string  != '\0')
        {
            if(*string >= '0' && *string <= '9')
            {
                string ++;
                num ++;
            }
            else if(*string == '.')
            {
                //..  10e3.4
                if(dot > 0 || e > 0)
                    return false;
```

```
            string ++;
            dot ++;
        }
        //E
        else if(*string == 'e' || *string == 'E')
        {
            // e10  10e10e
            if(num == 0 || e > 0)
                return false;
            string ++;
            e ++;
            if(*string == '-' || *string == '+')
            {
                string ++;//10e-10
                //10e--10
                if(*string == '-' || *string == '+')
                    return false;
            }
            if(*string == '\0')//10e
                return false;
        }
        else //as10
            return false;
    }
    return true;
  }
};
```

来自 <https://www.nowcoder.com/profile/2983379/wrongset/283782461?page=1&offset=3&tags=>

21调整数组顺序使奇数位于偶数前面

2020年4月7日 3:11

输入一个整数数组，实现一个函数来调整该数组中数字的顺序，使得所有的奇数位于数组的前半部分，所有的偶数位于数组的后半部分，并保证奇数和奇数，偶数和偶数之间的相对位置不变。

来自 <https://www.nowcoder.com/profile/2983379/wrongset/283784310?page=1&offset=2&tags=>

```
class Solution {
public:
    void reOrderArray(vector<int> &array) {
        /*
        queue<int> q1, q2;

        for(int i = 0; i < array.size(); i ++)
        {
            if(array[i] % 2 == 1)
                q1.push(array[i]);
            else
                q2.push(array[i]);
        }

        array.clear();
        while(q1.size())
        {
            array.push_back(q1.front());
            q1.pop();
        }
        while(q2.size())
        {
            array.push_back(q2.front());
            q2.pop();
        }
        */

        //排序前后的稳定性
        //是不是奇数
        //前面偶数 后面奇数  交换
        for(int i = 0; i < array.size(); i ++)
        {
            for(int j = array.size() - 1; j > i; j --)
            {
                if(array[j] % 2 == 1 && array[j - 1] % 2 == 0)
```

```
                swap(array[j],array[j - 1]);
            }
        }

    }
};
```

来自 <https://www.nowcoder.com/profile/2983379/wrongset/283784310?page=1&offset=2&tags=>

22链表中倒数第k个节点

2020年4月7日 4:13

输入一个链表，输出该链表中倒数第k个结点。

来自 <https://www.nowcoder.com/profile/2983379/wrongset/283786102?page=1&offset=1&tags=>

```
/*
struct ListNode {
    int val;
    struct ListNode *next;
    ListNode(int x) :
            val(x), next(NULL) {
    }
};*/
class Solution {
public:
    ListNode* FindKthToTail(ListNode* head, unsigned int k) {
        int n = 0;
        for(ListNode* p = head; p; p = p -> next) n ++;

        if(k > n) return nullptr;
        ListNode* p = head;
        for(int i = 0; i < n - k; i ++) p = p -> next;

        return p;
    }
};
```

来自 <https://www.nowcoder.com/profile/2983379/wrongset/283786102?page=1&offset=1&tags=>

23链表中环的入口结点

2020年4月7日 4:41

题目描述

给一个链表，若其中包含环，请找出该链表的环的入口结点，否则，输出null。

来自 <https://www.nowcoder.com/practice/253d2c59ec3e4bc68da16833f79a38e4?tpId=13&tqId=11208&tPage=1&rp=1&ru=/ta/coding-interviews&qru=/ta/coding-interviews/question-ranking>

```
/*
struct ListNode {
    int val;
    struct ListNode *next;
    ListNode(int x) :
        val(x), next(NULL) {
    }
};
*/
class Solution {
public:
    ListNode* EntryNodeOfLoop(ListNode* head)
    {
        /*
        //快慢指针
        auto fast = head, slow = head;
        while(fast && slow)
        {
            fast = fast -> next;
            slow = slow -> next;
            if(fast) fast = fast -> next;
            else break;
            //相遇点
            if(fast == slow)
            {
                slow = head;
                while(fast != slow)
                {
                    fast = fast -> next;
                    slow = slow -> next;
                }
                return slow;
            }
        }
        return 0;
        */
        //hash表
        unordered_map<ListNode*,int> h;
        int id = 1;
        for(auto i = head; i ; i = i -> next, id ++)
        {
            if(h[i] != 0)
```

```
            {
                return i;
            }
            else
                h[i]  = id;
        }

        return NULL;
    }
};
```

24反转链表

2020年4月7日 4:43

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的头结点。

思考题：

- 请同时实现迭代版本和递归版本。

样例

输入:1->2->3->4->5->NULL

输出:5->4->3->2->1->NULL

来自 <https://www.acwing.com/problem/content/33/>

```
/*
struct ListNode {
        int val;
        struct ListNode *next;
        ListNode(int x) :
                        val(x), next(NULL) {
        }
};*/
class Solution {
public:
   ListNode* ReverseList(ListNode* pHead) {
      //找到前驱结点
      ListNode* pre = nullptr;

      auto cur = pHead;
      while(cur)
      {
         auto next = cur -> next;
         cur -> next = pre;
         pre = cur;
         cur = next;
      }
      return pre;
   }
};
```

25合并两个排序的链表

2020年4月7日 4:51

输入两个递增排序的链表，合并这两个链表并使新链表中的结点仍然是按照递增排序的。

样例

输入：1->3->5 , 2->4->5

输出：1->2->3->4->5->5

来自 <https://www.acwing.com/problem/content/34/>

```
/*
struct ListNode {
        int val;
        struct ListNode *next;
        ListNode(int x) :
                        val(x), next(NULL) {
        }
};*/
class Solution {
public:
   ListNode* Merge(ListNode* p1, ListNode* p2)
   {
      //归并思想
      ListNode* dummy = new ListNode(-1);
      auto cur = dummy;
      while(p1 && p2)
      {
         if(p1 -> val <= p2 -> val)
         {
            cur -> next = p1;
            cur = cur -> next;
            p1 = p1 -> next;
         }
         else{
            cur -> next = p2;
            cur = cur -> next;
            p2 = p2 -> next;
         }
      }

      if(p1) cur -> next = p1;
      else cur -> next = p2;

      return dummy -> next;
```

```
    }
};
```

26树的子结构

2020年4月7日 5:05

输入两棵二叉树A，B，判断B是不是A的子结构。

我们规定空树不是任何树的子结构。

样例

树A：

```
  8
 /\
 8  7
/\
9  2
 /\
 4  7
```

树B：

```
 8
/\
9  2
```

返回 **true** ,因为B是A的子结构。

来自 <https://www.acwing.com/problem/content/35/>

```
/*
struct TreeNode {
        int val;
        struct TreeNode *left;
        struct TreeNode *right;
        TreeNode(int x) :
                        val(x), left(NULL), right(NULL) {
        }
};*/
class Solution {
public:
   bool HasSubtree(TreeNode* t1, TreeNode* t2)
   {
      if(!t1 || !t2) return false;
      if(isPart(t1, t2)) return true;

      return HasSubtree(t1 -> left, t2) || HasSubtree(t1 -> right, t2);
   }
```

```
    //只能比较t1  t2各自为根结点的树
    bool isPart(TreeNode* t1, TreeNode* t2)
    {
        if(!t2) return true;//t2已经遍历到了叶节点结束
        if(!t1 || t1 -> val != t2 -> val) return false;
        //比较左右子孩子
        return isPart(t1 -> left, t2 -> left) && isPart(t1 -> right, t2 -> right);
    }
};
```

27二叉树的镜像

2020年4月7日 8:18

输入一个二叉树，将它变换为它的镜像。

样例

输入树:

```
     8
    / \
   6  10
  / \ / \
 5  7 9 11
```

[8,6,10,5,7,9,11,null,null,null,null,null,null,null,null]

输出树:

```
     8
    / \
   10  6
  / \ / \
 11 9 7  5
```

[8,10,6,11,9,7,5,null,null,null,null,null,null,null,null]

来自 <https://www.acwing.com/problem/content/37/>

```
/*
struct TreeNode {
        int val;
        struct TreeNode *left;
        struct TreeNode *right;
        TreeNode(int x) :
                        val(x), left(NULL), right(NULL) {
        }
};*/
class Solution {
public:
   void Mirror(TreeNode *root) {
      if(!root) return;

      Mirror(root -> left);
      Mirror(root -> right);
      //交换左右结点
      swap(root -> left, root -> right);
   }
};
```

28对称的二叉树

2020年4月7日 8:26

请实现一个函数，用来判断一棵二叉树是不是对称的。

如果一棵二叉树和它的镜像一样，那么它是对称的。

样例

如下图所示二叉树[1,2,2,3,4,4,3,null,null,null,null,null,null,null,null]为对称二叉树：

```
    1
   / \
  2   2
 / \ / \
3  4 4  3
```

如下图所示二叉树[1,2,2,null,4,4,3,null,null,null,null,null,null]不是对称二叉树：

```
    1
   / \
  2   2
   \ / \
   4 4  3
```

来自 <https://www.acwing.com/problem/content/38/>

```
/*
struct TreeNode {
    int val;
    struct TreeNode *left;
    struct TreeNode *right;
    TreeNode(int x) :
            val(x), left(NULL), right(NULL) {
    }
};
*/
class Solution {
public:
    bool isSymmetrical(TreeNode* root)
    {
        if(!root) return true;

        return dfs(root -> left, root -> right);
    }

    bool dfs(TreeNode* t1, TreeNode* t2)
    {
        if(!t1 || !t2) return !t1 && !t2;
```

```
        if(t1 -> val != t2 -> val) return false;

        return dfs(t1 -> left, t2 -> right) && dfs(t1 -> right, t2 -> left);
    }

};
```

29顺时针打印矩阵

2020年4月7日 8:53

输入一个矩阵，按照从外向里以顺时针的顺序依次打印出每一个数字。

样例

输入：
[
[1, 2, 3, 4],
[5, 6, 7, 8],
[9,10,11,12]
]
输出：[1,2,3,4,8,12,11,10,9,5,6,7]

来自 <https://www.acwing.com/problem/content/39/>

```
class Solution {
public:
    vector<int> printMatrix(vector<vector<int> > matrix) {
        vector<int> res;
        //定义方向
        //碰壁换方向 右下左上

        if(matrix.empty()) return res;

        int n = matrix.size(), m = matrix[0].size();
        vector<vector<bool>> state(n, vector<bool>(m, false));
        int dx[4] = {-1, 0, 1, 0}, dy[4] = {0, 1, 0, -1};
        int x = 0, y = 0, d = 1;

        for(int i = 0; i < n * m; i ++)
        {
            res.push_back(matrix[x][y]);
            state[x][y] = true;

            int a = x + dx[d], b = y + dy[d];
            if(a < 0 || a >= n || b < 0 || b >= m || state[a][b] == true)
            {
                d = (d + 1) % 4;
                a = x + dx[d], b = y + dy[d];
            }

            x = a, y = b;
        }

        return res;
    }
};
```

30包含min函数的栈

2020年4月7日 9:32

设计一个支持push，pop，top等操作并且可以在O(1)时间内检索出最小元素的堆栈。

- push(x)–将元素x插入栈中
- pop()–移除栈顶元素
- top()–得到栈顶元素
- getMin()–得到栈中最小元素

样例

```
MinStack minStack = new MinStack();
minStack.push(-1);
minStack.push(3);
minStack.push(-4);
minStack.getMin();   --> Returns -4.
minStack.pop();
minStack.top();      --> Returns 3.
minStack.getMin();   --> Returns -1.
```

来自 <https://www.acwing.com/problem/content/90/>

```
class MinStack {
public:
    /** initialize your data structure here. */
    stack<int> stk, min_stk;
    MinStack() {

    }

    void push(int x) {
        stk.push(x);
        if(min_stk.empty() || min_stk.top() >= x) min_stk.push(x);
    }

    void pop() {

        if(stk.top() == min_stk.top()) min_stk.pop();
        stk.pop();

    }

    int top() {
        return stk.top();
    }

    int getMin() {
        return min_stk.top();
    }
```

```
};

/**
 * Your MinStack object will be instantiated and called as such:
 * MinStack obj = new MinStack();
 * obj.push(x);
 * obj.pop();
 * int param_3 = obj.top();
 * int param_4 = obj.getMin();
 */
```

31栈的压入、弹出序列

2020年4月7日 9:40

输入两个整数序列，第一个序列表示栈的压入顺序，请判断第二个序列是否可能为该栈的弹出顺序。

假设压入栈的所有数字均不相等。

例如序列1,2,3,4,5是某栈的压入顺序，序列4,5,3,2,1是该压栈序列对应的一个弹出序列，但4,3,5,1,2就不可能是该压栈序列的弹出序列。

注意：若两个序列长度不等则视为并不是一个栈的压入、弹出序列。若两个序列都为空，则视为是一个栈的压入、弹出序列。

样例

输入：[1,2,3,4,5]
[4,5,3,2,1]
输出：true

来自 <https://www.acwing.com/problem/content/40/>

```
class Solution {
public:
    bool isPopOrder(vector<int> pushV,vector<int> popV) {
        if(pushV.size() != popV.size()) return false;

        int i = 0;
        stack<int> stk;

        for(auto x : pushV)
        {
            stk.push(x);
            while(stk.size() && stk.top() == popV[i])
            {
                stk.pop();
                i ++;
            }
        }

        return stk.empty();
    }
};
```

32从上往下打印二叉树

2020年4月12日 13:56

题目描述

从上往下打印出二叉树的每个节点，同层节点从左至右打印。

来自 <https://www.nowcoder.com/practice/7fe2212963db4790b57431d9ed259701?tpId=13&tqId=11175&tPage=2&rp=1&ru=%2Fta%2Fcoding-interviews&qru=%2Fta%2Fcoding-interviews%2Fquestion-ranking>

```
/*
struct TreeNode {
        int val;
        struct TreeNode *left;
        struct TreeNode *right;
        TreeNode(int x) :
                        val(x), left(NULL), right(NULL) {
        }
};*/
class Solution {
public:
   vector<int> PrintFromTopToBottom(TreeNode* root) {
      vector<int> res;
      if(!root) return res;

      //非空
      queue<TreeNode*> q;
      q.push(root);

      //bfs 广度优先遍历
      while(q.size())
      {
         auto t = q.front();
         q.pop();
         res.push_back(t -> val);
         if(t -> left) q.push(t -> left);
         if(t -> right) q.push(t -> right);
      }

      return res;
   }
};
```

32不分行从上往下打印二叉树

2020年4月7日 10:30

从上往下打印出二叉树的每个结点，同一层的结点按照从左到右的顺序打印。

样例

输入如下图所示二叉树[8, 12, 2, null, null, 6, null, 4, null, null, null]

```
  8
 / \
12  2
   /
  6
 /
4
```

输出：[8, 12, 2, 6, 4]

来自 <https://www.acwing.com/problem/content/41/>

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
 * };
 */
class Solution {
public:
    vector<int> printFromTopToBottom(TreeNode* root) {
        vector<int> res;
        if(!root)  return res;

        queue<TreeNode*> q;
        q.push(root);

        while(q.size())
        {
            auto t = q.front();
            q.pop();
            res.push_back(t -> val);
            if(t -> left) q.push(t -> left);
            if(t -> right) q.push(t -> right);
        }


        return res;
    }
```

```
};
```

作者：满血小进
链接：https://www.acwing.com/activity/content/code/content/268943/
来源：AcWing

32分行从上往下打印二叉树

2020年4月7日 10:32

44. 分行从上往下打印二叉树

- 题目
- 提交记录
- 讨论
- 题解
- 视频讲解

从上到下按层打印二叉树，同一层的结点按从左到右的顺序打印，每一层打印到一行。

样例

输入如下图所示二叉树[8, 12, 2, null, null, 6, null, 4, null, null, null]

```
  8
 / \
12  2
  /
 6
/
4
```

输出：[[8], [12, 2], [6], [4]]

来自 <https://www.acwing.com/problem/content/42/>

```
/*
struct TreeNode {
    int val;
    struct TreeNode *left;
    struct TreeNode *right;
    TreeNode(int x) :
            val(x), left(NULL), right(NULL) {
    }
};
*/
class Solution {
public:
        vector<vector<int> > Print(TreeNode* root) {
            vector<vector<int> > res;
            if(!root) return res;

            queue<TreeNode*> q;
```

```
        q.push(root);
        q.push(nullptr);

        vector<int> layer;
        while(q.size())
        {
            auto t = q.front();
            q.pop();
            //读到nullptr表示分行 把layer push到 res
            if(!t)
            {
                if(layer.empty()) break;
                res.push_back(layer);
                layer.clear();
                q.push(nullptr);
                continue;
            }
            layer.push_back(t -> val);
            if(t -> left) q.push(t -> left);
            if(t -> right) q.push(t -> right);
        }

        return res;
    }

};
```

32之字形打印二叉树

2020年4月7日 10:32

请实现一个函数按照之字形顺序从上向下打印二叉树。

即第一行按照从左到右的顺序打印，第二层按照从右到左的顺序打印，第三行再按照从左到右的顺序打印，其他行以此类推。

样例

输入如下图所示二叉树[8, 12, 2, null, null, 6, 4, null, null, null, null]

```
   8
  / \
 12  2
    / \
   6   4
```

输出：[[8], [2, 12], [6, 4]]

来自 <https://www.acwing.com/problem/content/43/>

```
/*
struct TreeNode {
    int val;
    struct TreeNode *left;
    struct TreeNode *right;
    TreeNode(int x) :
            val(x), left(NULL), right(NULL) {
    }
};
*/
class Solution {
public:
    vector<vector<int> > Print(TreeNode* root) {
        vector<vector<int> > res;
        if(!root) return res;
        queue<TreeNode*> q;
        q.push(root);
        q.push(nullptr);

        vector<int> layer;

        bool zigzag = false;
        while(q.size())
        {
            auto t = q.front();
            q.pop();
            if(!t)
            {
                if(layer.empty()) break;
                if(zigzag) reverse(layer.begin(), layer.end());
```

```
                res.push_back(layer);
                layer.clear();
                zigzag = !zigzag;
                q.push(nullptr);
                continue;
            }
            layer.push_back(t -> val);
            if(t -> left) q.push(t -> left);
            if(t -> right) q.push(t -> right);
        }

        return res;
    }

};
```

33二叉搜索树的后序遍历序列

2020年4月8日 3:15

输入一个整数数组，判断该数组是不是某二叉搜索树的后序遍历的结果。

如果是则返回true，否则返回false。

假设输入的数组的任意两个数字都互不相同。

样例

输入：[4, 8, 6, 12, 16, 14, 10]

输出：true

来自 <https://www.acwing.com/problem/content/44/>

```
class Solution {
public:
  //后续遍历 左右根
  //二叉搜索树特点 左孩子 < 父亲结点 < 右孩子
  //左右根
  //数组最后一个数字就是根节点/父亲结点
  //左右子树分界点 与根节点相比较
  vector<int> seq;
  bool VerifySquenceOfBST(vector<int> sequence) {
    seq = sequence;
    if(seq.empty()) return false;
    return dfs(0, seq.size() - 1);
  }

  bool dfs(int l, int r)
  {
    if(l >= r) return true;

    //父亲结点
    int father = seq[r];
    //左子树 都小于
    int k = l;
    while(k < r && seq[k] < father) k ++;
    //右子树 都大于
    for(int i = k; i < r; i ++)
      if(seq[i] < father)
        return false;

    return dfs(l, k - 1) && dfs(k, r - 1);
  }
};
```

34二叉树中和为某一值的路径

2020年4月8日 3:23

输入一棵二叉树和一个整数，打印出二叉树中结点值的和为输入整数的所有路径。

从树的根结点开始往下一直到叶结点所经过的结点形成一条路径。

样例

给出二叉树如下所示，并给出num=22。

```
     5
    / \
   4   6
  /   / \
 12  13  6
/ \   / \
9   1 5   1
```

输出：[[5,4,12,1],[5,6,6,5]]

来自 <https://www.acwing.com/problem/content/45/>

```
/*
struct TreeNode {
        int val;
        struct TreeNode *left;
        struct TreeNode *right;
        TreeNode(int x) :
                        val(x), left(NULL), right(NULL) {
        }
};*/
class Solution {
public:
   vector<vector<int> > res;
   vector<int> path;
   vector<vector<int> > FindPath(TreeNode* root,int sum) {
      dfs(root, sum);
      return res;
   }

   void dfs(TreeNode* root,int sum)
   {
      if(!root) return;
      path.push_back(root -> val);
      sum -= root -> val;
      //sum == 0 且找到了叶节点 确定一条路径
      if(!sum && !root -> left && !root -> right) res.push_back(path);

      dfs(root -> left, sum);
```

```
        dfs(root ->right, sum);
        path.pop_back();
    }
};
```

35复杂链表的复刻

2020年4月8日 3:41

请实现一个函数可以复制一个复杂链表。

在复杂链表中，每个结点除了有一个指针指向下一个结点外，还有一个额外的指针指向链表中的任意结点或者null。

注意:

- 函数结束后原链表要与输入时保持一致。

来自 <https://www.acwing.com/problem/content/89/>

具体分为三步：

（1）在旧链表中创建新链表，此时不处理新链表的兄弟结点

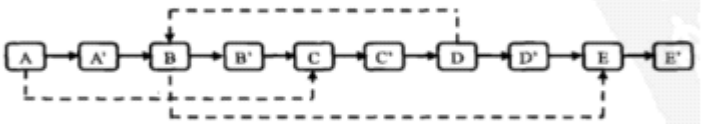


图 4.9 复制复杂链表的第一步

（2）根据旧链表的兄弟结点，初始化新链表的兄弟结点

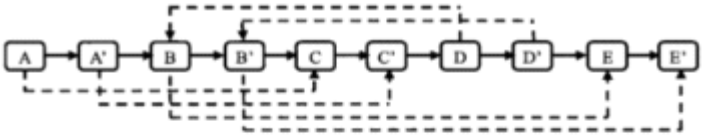


图 4.10 复制复杂链表的第二步

（3）从旧链表中拆分得到新链表

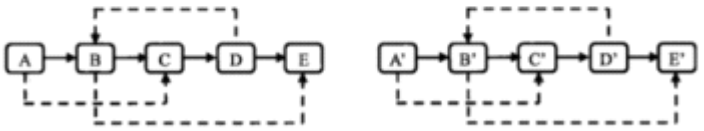


图 4.11 复制复杂链表的第三步

```
/*
struct RandomListNode {
    int label;
    struct RandomListNode *next, *random;
    RandomListNode(int x) :
            label(x), next(NULL), random(NULL) {
    }
};
*/
class Solution {
public:
    RandomListNode* Clone(RandomListNode* head)
    {
```

```
    //1. 复制每个结点 next
       //在两个结点之间新建一个结点 复制前驱
    for(auto p = head; p;)
    {
       auto newp = new RandomListNode(p -> label);
       //在p 和 p -> next之间插入newp
       auto temp = p -> next;
       p -> next = newp;
       newp -> next = temp;
       p = temp;
    }

    //random
    //a -> next -> ramdom = a -> random -> next;
    for(auto p = head; p; p = p -> next -> next)
    {
       if(p -> random)
          p -> next -> random = p -> random -> next;
    }

    auto dummy = new RandomListNode(-1);

    auto cur = dummy;
    for(auto p = head; p; p = p -> next)
    {
       cur -> next = p -> next;//A'
       cur = cur -> next;
       p -> next = p -> next -> next;
    }

    return dummy -> next;
  }
};
```

36二叉搜索树与双向链表

2020年4月8日 4:23

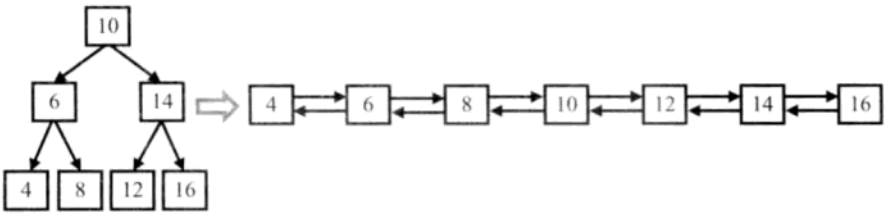
输入一棵二叉搜索树，将该二叉搜索树转换成一个排序的双向链表。

要求不能创建任何新的结点，只能调整树中结点指针的指向。

注意：

- 需要返回双向链表最左侧的节点。

例如，输入下图中左边的二叉搜索树，则输出右边的排序双向链表。



来自 <https://www.acwing.com/problem/content/87/>

```
/*
struct TreeNode {
        int val;
        struct TreeNode *left;
        struct TreeNode *right;
        TreeNode(int x) :
                        val(x), left(NULL), right(NULL) {
        }
};*/
class Solution {
public:
  TreeNode* Convert(TreeNode* root)
  {
    if(!root) return NULL;
    auto pairs = dfs(root);
    return pairs.first;
  }

  pair<TreeNode*,TreeNode*> dfs(TreeNode* root)
  {
    if(!root -> left && ! root -> right) return {root, root};
    if(root -> left && root -> right)
    {
      auto lsides = dfs(root -> left), rsides = dfs(root -> right);
      //左边返回
      //{1，2} 2 ——> root   root -> 2
      lsides.second -> right = root;
      root -> left = lsides.second;
      //右边返回
```

```
            //{1，2} 1——> root  root -> 1
            rsides.first -> left = root;
            root -> right = rsides.first;
            return {lsides.first, rsides.second};
        }
        //只有左子树
        if(root -> left)
        {
            auto lsides = dfs(root -> left);
            //左边返回
            //{1，2} 2 ——> root   root -> 2
            lsides.second -> right = root;
            root -> left = lsides.second;

            return {lsides.first, root};
        }
        //只有右子树
        if(root -> right)
        {
            auto rsides = dfs(root -> right);
            //右边返回
            //{1，2} 1——> root  root -> 1
            rsides.first -> left = root;
            root -> right = rsides.first;
            return {root, rsides.second};
        }
    }
};
```

37序列化二叉树

2020年4月8日 4:23

请实现两个函数，分别用来序列化和反序列化二叉树。

您需要确保二叉树可以序列化为字符串，并且可以将此字符串反序列化为原始树结构。

样例

你可以序列化如下的二叉树

```
  8
 / \
12  2
   / \
  6   4
```

为："[8, 12, 2, null, null, 6, 4, null, null, null, null]"

注意:

- 以上的格式是AcWing序列化二叉树的方式，你不必一定按照此格式，所以可以设计出一些新的构造方式。

来自 <https://www.acwing.com/problem/content/46/>

```
/*
struct TreeNode {
    int val;
    struct TreeNode *left;
    struct TreeNode *right;
    TreeNode(int x) :
            val(x), left(NULL), right(NULL) {
    }
};
*/
class Solution {
public:
    char* Serialize(TreeNode *root) {
        string res;
        dfs1(root, res);
        char* p = new char[res.size() + 1];
        strcpy(p, res.c_str());

        return p;
    }
```

```
    void dfs1(TreeNode* root, string &res)
    {
        if(!root)
        {
            //#
            res += "#,";
            return;
        }

        res += to_string(root -> val) + ",";
        dfs1(root -> left, res);
        dfs1(root -> right, res);
    }

    //反序列化
    TreeNode* Deserialize(char *str) {
        int idx = 0;
        return dfs2(str, idx);
    }

    TreeNode* dfs2(char* str, int &idx)
    {
        //确定长度 23 长度2 3 长度1
        int len = idx;
        while(str[len] != ',') len ++;
        //空结点
        if(str[idx] == '#')
        {
            idx = len + 1;
            return NULL;
        }
        //非空结点
        //计算数值
        int num = 0;
        //考虑符号+-
        int sign = 1;
        if(idx < len && str[idx] == '-') sign = -1, idx ++;
        // ‘234’
        for(int i = idx; i < len; i ++) num = num * 10 + str[i] - '0';
        num *= sign;
        //idx走到下一个数字
        idx = len + 1;
        //构建树
        auto root = new TreeNode(num);
        root -> left = dfs2(str, idx);
        root -> right = dfs2(str, idx);

        return root;
    }

};
```

38字符串的排列

2020年4月8日 4:23

题目描述

输入一个字符串,按字典序打印出该字符串中字符的所有排列。例如输入字符串abc,则打印出由字符a,b,c所能排列出来的所有字符串abc,acb,bac,bca,cab和cba。

输入描述:

输入一个字符串,长度不超过9(可能有字符重复),字符只包括大小写字母。

```
class Solution {
public:
    vector<string> res;
    string path;

    vector<string> Permutation(string str) {
        if(str == "") return res;  //[""] []
        //全排列
        //可能重复字母 靠在一起 好判断
        sort(str.begin(),str.end());
        path.resize(str.size());
        //全排列函数
        dfs(str, 0, 0, 0);

        //字典顺序
        sort(res.begin(), res.end());
        return res;
    }

    void dfs(string &str, int idx, int start, int state)
    {
        //已经找到一个排列
        if(idx == str.size())
        {
            res.push_back(path);
            return;
        }
        //单个排列没有排完
        //相同字母而言 前后关系
        //可能重复排序
        //通过控制相对顺序不变确保不重复
        if(idx == 0 || str[idx] != str[idx - 1]) start = 0;
        //顺序往后
        for(int i = start; i < str.size(); i ++)
            if(!(state >> i & 1))
            {
                path[i] = str[idx];
                dfs(str, idx + 1, i + 1, state + (1 << i));
            }
    }
};
```

39数组中出现次数超过一半的数字

2020年4月8日 4:23

题目描述

数组中有一个数字出现的次数超过数组长度的一半，请找出这个数字。例如输入一个长度为9的数组{1,2,3,2,2,2,5,4,2}。由于数字2在数组中出现了5次，超过数组长度的一半，因此输出2。如果不存在则输出0。

来自 <https://www.nowcoder.com/practice/e8a1b01a2df14cb2b228b30ee6a92163?tpId=13&tqId=11181&tPage=1&rp=1&ru=/ta/coding-interviews&qru=/ta/coding-interviews/question-ranking>

```
class Solution {
public:
    int MoreThanHalfNum_Solution(vector<int> numbers) {
        int val, cnt = 0;
        for(auto x : numbers)
        {
            if(val == x)
                cnt ++;
            else{
                if(cnt) cnt --;
                else{
                    val = x;
                    cnt = 1;
                }
            }
        }

        cnt = 0;
        for(auto x : numbers)
        {
            if(x == val)
                cnt ++;
        }

        if(cnt * 2 > numbers.size())
            return val;

        return 0;

    }
};
```

40最小的k个数

2020年4月8日 4:23

题目描述

输入n个整数，找出其中最小的K个数。例如输入4,5,1,6,2,7,3,8这8个数字，则最小的4个数字是1,2,3,4,。

来自 <https://www.nowcoder.com/practice/6a296eb82cf844ca8539b57c23e6e9bf?tpId=13&tqId=11182&tPage=2&rp=1&ru=%2Fta%2Fcoding-interviews&qru=%2Fta%2Fcoding-interviews%2Fquestion-ranking>

```
class Solution {
public:
    vector<int> GetLeastNumbers_Solution(vector<int> input, int k) {
        //暴力解答
        /*
        vector<int> res;
        if(k > input.size()) return res;

        sort(input.begin(),input.end());
        for(int i = 0; i < k; i ++)
        {
            res.push_back(input[i]);
        }

        return res;
        */
        //大根堆 k
        vector<int> res;
        if(k > input.size()) return res;
        priority_queue<int> heap;

        //找到最小的k个数字
        for(auto x : input)
        {
            heap.push(x);
            if(heap.size() > k) heap.pop();
        }
        //从大到小输入
        while(heap.size()) res.push_back(heap.top()), heap.pop();
        //从小到大
        reverse(res.begin(), res.end());

        return res;

    }
};
```

41数据流中的中位数

2020年4月8日 11:27

如何得到一个数据流中的中位数?

如果从数据流中读出奇数个数值，那么中位数就是所有数值排序之后位于中间的数值。

如果从数据流中读出偶数个数值，那么中位数就是所有数值排序之后中间两个数的平均值。

样例

输入：1, 2, 3, 4
输出：1,1.5,2,2.5

解释：每当数据流读入一个数据，就进行一次判断并输出当前的中位数。

来自 <https://www.acwing.com/problem/content/88/>

```
class Solution {
public:

  //例子
  //输入：1, 2, 3, 4
  //输出：1,1.5,2,2.5

  //大小根堆
  //小根堆 较大的数字 最小的是根节点
  //大根堆 较小的数字 最大的是根节点
  //大根堆 - 小根堆 = 1/0

  priority_queue<int> maxheap;
  priority_queue<int, vector<int>, greater<int>> minheap;

  void Insert(int num)
  {
    //都无脑先插入 大根堆
    maxheap.push(num);

    //大根堆 - 小根堆 = 1/0
    if(maxheap.size() - minheap.size() > 1)
    {
      //从大根堆中拿最大的元素到小根堆
      minheap.push(maxheap.top());
```

```
            maxheap.pop();
        }

        //插入的元素较大
        //大根堆的根 > 小根堆的根
        //交换
        while(minheap.size() && maxheap.top() > minheap.top())
        {
            int max = maxheap.top(), min = minheap.top();
            maxheap.pop(), minheap.pop();
            maxheap.push(min), minheap.push(max);
        }
    }

    double GetMedian()
    {
        //奇数
        if((maxheap.size() + minheap.size()) % 2 == 1) return maxheap.top();
        //偶数
        return (maxheap.top() + minheap.top()) /2.0;
    }

};
```

42连续子数组的最大和

2020年4月9日 4:11

输入一个 **非空** 整型数组，数组里的数可能为正，也可能为负。

数组中一个或连续的多个整数组成一个子数组。

求所有子数组的和的最大值。

要求时间复杂度为O(n)。

样例

输入：[1, -2, 3, 10, -4, 7, 2, -5]
输出：18

来自 <https://www.acwing.com/problem/content/description/50/>

```
class Solution {
public:
  //面试高频
  int FindGreatestSumOfSubArray(vector<int> array) {
    int res = INT_MIN, s = 0;
    //x 10  -11 3 4
    //s 10  -1  3 7
    //r 10   10 10 10
    for(auto x : array)
    {
      if(s < 0) s = 0;
      s += x;
      res = max(res, s);
    }

    return res;
  }
};
```

43从1到n整数中1出现的次数

2020年4月9日 4:11

题目描述

求出1~13的整数中1出现的次数,并算出100~1300的整数中1出现的次数？为此他特别数了一下1~13中包含1的数字有1、10、11、12、13因此共出现6次,但是对于后面问题他就没辙了。ACMer希望你们帮帮他,并把问题更加普遍化,可以很快的求出任意非负整数区间中1出现的次数（从1 到 n 中1出现的次数）。

来自 <https://www.nowcoder.com/practice/bd7f978302044eee894445e244c7eee6?tpId=13&tqId=11184&tPage=2&rp=1&ru=%2Fta%2Fcoding-interviews&qru=%2Fta%2Fcoding-interviews%2Fquestion-ranking>

```
class Solution {
public:
  //13  6
  //left i  right
  //i 1  left * 10^i + right + 1
  //i 0 left *10 ^ i + 10^i
  int NumberOf1Between1AndN_Solution(int n)
  {
    if(!n) return 0;
    //n >= 1
    vector<int> num;
    while(n)
    {
      num.push_back(n % 10);
      n /= 10;
    }

    int res = 0;
    for(int i = num.size() - 1; i >= 0; i --)
    {
      int left = 0, right = 0, x = 1;
      //高位到低位 left
      for(int j = num.size() - 1; j > i; j --)
        left = left * 10 + num[j];
      //计算低位 right
      for(int k = i - 1; k >= 0; k --)
      {
        right = right * 10 + num[k];
        x *= 10;
      }

      //计算位数
      res = res + left * x;
      if(num[i] == 1) res += right + 1;
      else if(num[i] > 1) res += x;

    }
    return res;
  }
};
```

44数字序列中某一位的数字

2020年4月9日 4:16

数字以0123456789101112131415...的格式序列化到一个字符序列中。

在这个序列中，第5位（从0开始计数）是5，第13位是1，第19位是4，等等。

请写一个函数求任意位对应的数字。

样例

输入：13

输出：1

来自 <https://www.acwing.com/problem/content/52/>

```
class Solution {
public:
    int digitAtIndex(int n) {
     if(!n) return 0;
     //确定是几位数
     //i 位数
     long long i = 1, s = 9, base = 1;
     while(n > i * s)
     {
        n -= i * s;
        i ++;
        s *= 10;
        base *= 10;
     }
     //确定是几位数的第几个数
     //number 确定第几个数
     //eg 4 位数的 第 1000 位属于哪个数
     // 1000 / 4 = 250
     // n + i - 1 / i下取整 == n/ i上取整
     int number = base + (n + i - 1)/ i - 1;
     //确定是该数的第几个数字
     //r如果取模 != 0 就是 对应的数字 否则 == 0就是最后一位 即 i
     int r = n % i ? n % i : i;
     for(int j = 0; j < i - r; j ++) number /= 10;

     return number % 10;
    }
};
```

45把数组排成最小的数

2020年4月9日 6:22

输入一个正整数数组，把数组里所有数字拼接起来排成一个数，打印能拼接出的所有数字中最小的一个。

例如输入数组[3, 32, 321]，则打印出这3个数字能排成的最小数字321323。

样例

输入：[3, 32, 321]

输出：321323

注意：输出数字的格式为字符串。

来自 <https://www.acwing.com/problem/content/54/>

```
class Solution {
public:
    static bool cmp(int a, int b)
    {
        string as = to_string(a), bs = to_string(b);
        //33 321
        //“33321” “32133”
        return as + bs < bs + as;
    }

    string PrintMinNumber(vector<int> nums) {
        //33  321
        //33321 32133
        sort(nums.begin(), nums.end(),cmp);

        string res;
        for(auto x : nums) res += to_string(x);
        return res;
    }
};
```

46把数字翻译成字符串

2020年4月9日 6:22

给定一个数字，我们按照如下规则把它翻译为字符串：

0翻译成“a”，1翻译成“b”，……，11翻译成“l”，……，25翻译成“z”。

一个数字可能有多个翻译。例如12258有5种不同的翻译，它们分别是“bccfi”、“bwfi”、“bczi”、“mcfi”和“mzi”。

请编程实现一个函数用来计算一个数字有多少种不同的翻译方法。

样例

输入：“12258”
输出：5

来自 <https://www.acwing.com/problem/content/55/>

```
class Solution {
public:
    int getTranslationCount(string s) {
        //动态规划
        //fi  = fi-1 +  fi-2
        // >=10  <=25
        int n = s.size();
        vector<int> f(n + 1);
        f[0] = 1;
        for(int i = 1; i <= n; i ++)
        {
            f[i] = f[i - 1];
            //什么时候加上f i -2
            //s[i - 2]s[i - 1]  >= 10 <= 25
            int t = (s[i - 2] - '0') * 10 + s[i - 1] - '0';
            if(t >= 10 && t <= 25)
                f[i] += f[i - 2];
        }

        return f[n];
    }
};
```

47礼物的最大价值

2020年4月9日 6:22

在一个m×n的棋盘的每一格都放有一个礼物，每个礼物都有一定的价值（价值大于0）。

你可以从棋盘的左上角开始拿格子里的礼物，并每次向右或者向下移动一格直到到达棋盘的右下角。

给定一个棋盘及其上面的礼物，请计算你最多能拿到多少价值的礼物？

注意：

- $m,n>0$

样例：

输入：
[
  [2,3,1],
  [1,7,1],
  [4,6,1]
]
输出：19

解释：沿着路径 2→3→7→6→1 可以得到拿到最大价值礼物。

来自 <https://www.acwing.com/problem/content/56/>

```
class Solution {
public:
    int getMaxValue(vector<vector<int>>& grid) {
        int x = grid.size(), y = grid[0].size();

        vector<vector<int>> f(x + 1, vector<int>(y + 1));
        for(int i = 1; i <= x; i ++)
            for(int j = 1; j <= y; j ++)
                f[i][j] = max(f[i -1][j], f[i][j - 1]) + grid[i - 1][j - 1];
        //f[1][1]  grid[0][0]

        return f[x][y];
    }
};
```

作者：满血小进

链接：https://www.acwing.com/activity/content/code/content/271815/

来源：AcWing

48最长不含重复字符的子字符串

2020年4月9日 6:22

请从字符串中找出一个最长的不包含重复字符的子字符串，计算该最长子字符串的长度。

假设字符串中只包含从‘ a’ 到‘ z’ 的字符。

样例

输入：“abcabc”

输出：3

来自 <https://www.acwing.com/problem/content/57/>

```
class Solution {
public:
    int longestSubstringWithoutDuplication(string s) {
        unordered_map<char, int> hash;

        int res = 0;
        for(int i = 0, j = 0; j < s.size(); j ++)
        {
            hash[s[j]] ++;
            while(hash[s[j]] > 1)//s[j]字符出现两次
            {
                hash[s[i]] --;
                i ++;;
            }
            res = max(res, j - i + 1);
        }

        return res;
    }
};
```

49丑数

2020年4月9日 6:22

我们把只包含质因子2、3和5的数称作丑数（Ugly Number）。

例如6、8都是丑数，但14不是，因为它包含质因子7。

求第n个丑数的值。

样例

输入：5

输出：5

注意：习惯上我们把1当做第一个丑数。

来自 <https://www.acwing.com/problem/content/58/>

```
class Solution {
public:
    int getUglyNumber(int n) {
        vector<int> chou(1, 1);

        int i = 0, j = 0, k = 0;
        //1
        while(-- n)// n - 1次
        {
            int t = min(chou[i] * 2, min(chou[j] * 3, chou[k] * 5));

            chou.push_back(t);
            if(chou[i] * 2 == t) i ++;
            if(chou[j] * 3 == t) j ++;
            if(chou[k] * 5 == t) k ++;
        }

        return chou.back();
    }
};
```

50字符串中第一个只出现一次的字符

2020年4月9日 6:22

在字符串中找出第一个只出现一次的字符。

如输入"abaccdeff"，则输出b。

如果字符串中不存在只出现一次的字符，返回#字符。

样例：

输入："abaccdeff"
输出：'b'

来自 <https://www.acwing.com/problem/content/59/>

```
class Solution {
public:
    char firstNotRepeatingChar(string s) {
        //hash
        unordered_map<char, int> hash;
        for(auto c : s) hash[c] ++;

        char res = '#';
        for(auto c : s)
        {
            if(hash[c] == 1)
            {
                res = c;
                break;
            }
        }

        return res;
    }
};
```

50字符流中第一个只出现一次的字符

2020年4月10日 8:14

请实现一个函数用来找出字符流中第一个只出现一次的字符。

例如，当从字符流中只读出前两个字符”go”时，第一个只出现一次的字符是’g’。

当从该字符流中读出前六个字符”google”时，第一个只出现一次的字符是’l’。

如果当前字符流没有存在出现一次的字符，返回#字符。

样例

输入："google"
输出："ggg#ll"

解释：每当字符流读入一个字符，就进行一次判断并输出当前的第一个只出现一次的字符。

来自 <https://www.acwing.com/problem/content/60/>

```
class Solution{
public:

  unordered_map<char, int> hash;
  queue<char> q;//不重复的字符 队首就是第一个不重复的字符
  //Insert one char from stringstream
  void insert(char ch){
    hash[ch] ++;
    if(hash[ch] > 1)
    {
      //双指针算法
      while(q.size() && hash[q.front()] > 1) q.pop();
    }
    else //== 1
      q.push(ch);
  }
  //return the first appearence once char in current stringstream
  char firstAppearingOnce(){
    if(q.empty()) return '#';
    return q.front();
  }
};
```

50第一个只出现一个的字符

2020年4月10日 9:17

题目描述

在一个字符串(0<=字符串长度<=10000，全部由字母组成)中找到第一个只出现一次的字符,并返回它的位置, 如果没有则返回 -1（需要区分大小写）.

来自 <https://www.nowcoder.com/practice/1c82e8cf713b4bbeb2a5b31cf5b0417c?tpId=13&tqId=11187&tPage=1&rp=1&ru=/ta/coding-interviews&qru=/ta/coding-interviews/question-ranking>

```
class Solution {
public:
    int FirstNotRepeatingChar(string str) {
        //针对第一次出现的字符/数字
        unordered_map<char, int> hash;
        for(auto c : str) hash[c] ++;

        int res = -1;
        int index = 0;
        for(auto c : str)
        {
            if(hash[c] == 1)
            {
                res = index;
                break;
            }
            index ++;
        }

        return res;
    }
};
```

51数组中的逆序对

2020年4月10日 8:15

在数组中的两个数字如果前面一个数字大于后面的数字，则这两个数字组成一个逆序对。

输入一个数组，求出这个数组中的逆序对的总数。

样例

输入：[1,2,3,4,5,6,0]

输出：6

来自 <https://www.acwing.com/problem/content/61/>

```
class Solution {
public:

    int merge(vector<int>& nums, int l, int r)
    {
        if(l >= r) return 0;
        int mid = l + r >> 1;
        int res = merge(nums, l, mid) + merge(nums, mid + 1, r);
        int i = l, j = mid + 1;
        vector<int> temp;
        while(i <= mid && j <= r)
        {
            if(nums[i] <= nums[j]) temp.push_back(nums[i ++]);
            else{
                temp.push_back(nums[j ++]);
                res += mid - i + 1;
            }
        }

        while(i <= mid) temp.push_back(nums[i ++]);
        while(j <= r) temp.push_back(nums[j ++]);

        i = l;
        for(auto x : temp) nums[i ++] = x;

        return res;


    }

    int inversePairs(vector<int>& nums) {
        //暴力解n^2
        // int res = 0;
        // for(int i = 0; i < nums.size(); i ++)
        //     for(int j = i + 1; j < nums.size(); j ++)
        //         if(nums[i] > nums[j])
        //             res ++;
```

```
        // return res;

        //递归 n*logn
        return merge(nums, 0, nums.size() - 1);
    }
};
```

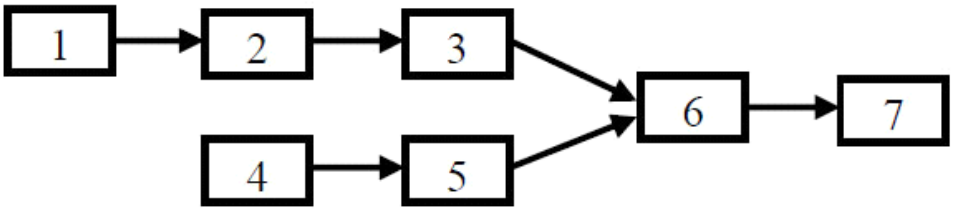
52两个链表的第一个公共结点

2020年4月10日 8:15

题目描述

输入两个链表，找出它们的第一个公共结点。（注意因为传入数据是链表，所以错误测试数据的提示是用其他方式显示的，保证传入数据是正确的）

来自 <https://www.nowcoder.com/practice/6ab1d9a29e88450685099d45c9e31e46?tpId=13&tqId=11189&tPage=2&rp=1&ru=%2Fta%2Fcoding-interviews&qru=%2Fta%2Fcoding-interviews%2Fquestion-ranking>




```
/*
struct ListNode {
        int val;
        struct ListNode *next;
        ListNode(int x) :
                        val(x), next(NULL) {
        }
};*/
class Solution {
public:
   ListNode* FindFirstCommonNode( ListNode* pHead1, ListNode* pHead2) {
      auto p = pHead1, q = pHead2;
      while(p != q)
      {
         if(p) p = p -> next;
         else p = pHead2;

         if(q) q = q -> next;
         else q = pHead1;
      }

      return p;
   }
};
```

53数字在排序数组中出现的次数

2020年4月10日 8:16

题目描述

统计一个数字在排序数组中出现的次数。

来自 <https://www.nowcoder.com/practice/70610bf967994b22bb1c26f9ae901fa2?tpId=13&tqId=11190&tPage=2&rp=1&ru=%2Fta%2Fcoding-interviews&qru=%2Fta%2Fcoding-interviews%2Fquestion-ranking>

```
class Solution {
public:
    int GetNumberOfK(vector<int> data ,int k) {
        //二分
        //left 左边 < k  右边 >=k
        //right 左边 <=k 右边 > k
        //right - left + 1
        if(data.empty()) return 0;
        //二分
        int l = 0, r = data.size() - 1;
        //找左端点
        while(l < r)
        {
            int mid = (l + r) / 2;
            if(data[mid] < k) l = mid + 1;
            else r = mid;
        }

        if(data[l] != k) return 0;
        int left = l;

        //找右端点
        l = 0, r = data.size() - 1;
        while(l < r)
        {
            int mid = (l + r + 1) / 2;
            if(data[mid] <= k) l = mid;
            else r = mid - 1;
        }

        int right = r;

        return right - left + 1;
    }
};
```

53.0到n-1中缺失的数字

2020年4月10日 8:16

一个长度为n-1的递增排序数组中的所有数字都是唯一的，并且每个数字都在范围0到n-1之内。

在范围0到n-1的n个数字中有且只有一个数字不在该数组中，请找出这个数字。

样例

输入：[0,1,2,4]
输出：3

来自 <https://www.acwing.com/problem/content/64/>

```
class Solution {
public:
    int getMissingNumber(vector<int>& nums) {
        if(nums.empty()) return 0;
        //二分 log n
        int l = 0, r = nums.size() - 1;

        while(l < r)
        {
            int mid = l + r >> 1;
            if(nums[mid] != mid) r = mid;
            else l = mid + 1;
        }

        //如果缺失的是最后一个数字，需要 r++
        if(nums[r] == r) r++;

        return r;
    }
};
```

53数组中数值和下标相等的元素

2020年4月10日 8:16

假设一个单调递增的数组里的每个元素都是整数并且是唯一的。

请编程实现一个函数找出数组中任意一个数值等于其下标的元素。

例如，在数组[-3, -1, 1, 3, 5]中，数字3和它的下标相等。

样例

输入：[-3, -1, 1, 3, 5]

输出：3

注意:如果不存在，则返回-1。

来自 <https://www.acwing.com/problem/content/65/>

```
class Solution {
public:
    int getNumberSameAsIndex(vector<int>& nums) {
        //暴力解答 n
        // int i = 0;
        // for(i = 0; i < nums.size(); i ++)
        // {
        //     if(nums[i] == i)
        //     {
        //         return i;
        //     }
        // }

        // return -1;

        //二分 log n
        // i - 1    i      i + 1
        // nums1   numsi    nums2
        //  <0     0       >= 0
        int l = 0, r = nums.size() - 1;
        while(l < r)
        {
            int mid = l + r >> 1;
            if(nums[mid] - mid >= 0) r = mid;
            else l = mid + 1;
        }
        if(nums[l] == l) return l;
        return -1;
    }
};
```

54二叉搜索树的第k大结点

2020年4月10日 8:16

题目描述

给定一棵二叉搜索树，请找出其中的第k小的结点。例如， （5，3，7，2，4，6，8） 中，按结点数值大小顺序第三小结点的值为4。

来自 <https://www.nowcoder.com/practice/ef068f602dde4d28aab2b210e859150a?tpId=13&tqId=11215&tPage=1&rp=1&ru=/ta/coding-interviews&qru=/ta/coding-interviews/question-ranking>

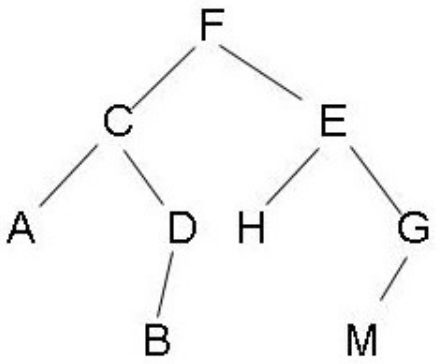
```
/*
struct TreeNode {
    int val;
    struct TreeNode *left;
    struct TreeNode *right;
    TreeNode(int x) :
            val(x), left(NULL), right(NULL) {
    }
};
*/
class Solution {
public:
    TreeNode* res = NULL;

    TreeNode* KthNode(TreeNode* root, int k)
    {
        dfs(root, k);
        return res;
    }

    void dfs(TreeNode* root, int &k)
    {
        if(!root) return;
        //中序遍历 左根右
        dfs(root -> left, k);
        //根
        k --;
        if(k == 0) res = root;
        //右
        if(k > 0) dfs(root -> right, k);
    }
};
```

55二叉树的深度

2020年4月10日 8:16

输入一棵二叉树，求该树的深度。从根结点到叶结点依次经过的结点（含根、叶结点）形成树的一条路径，最长路径的长度为树的深度。

来自 <https://www.nowcoder.com/practice/435fb86331474282a3499955f0a41e8b?tpId=13&tqId=11191&tPage=1&rp=1&ru=/ta/coding-interviews&qru=/ta/coding-interviews/question-ranking>




```
/*
struct TreeNode {
        int val;
        struct TreeNode *left;
        struct TreeNode *right;
        TreeNode(int x) :
                        val(x), left(NULL), right(NULL) {
        }
};*/
class Solution {
public:
    int TreeDepth(TreeNode* root)
    {
        //递归
        //max（left, right） + 1
        if(!root) return 0;
        return max(TreeDepth(root -> left), TreeDepth(root -> right)) + 1;
    }
};
```

55平衡二叉树

2020年4月10日 8:17

题目描述

输入一棵二叉树，判断该二叉树是否是平衡二叉树。

来自 <https://www.nowcoder.com/practice/8b3b95850edb4115918ecebdf1b4d222?tpId=13&tqId=11192&tPage=2&rp=1&ru=%2Fta%2Fcoding-interviews&qru=%2Fta%2Fcoding-interviews%2Fquestion-ranking>

```
class Solution {
public:
    bool res = true;
    bool IsBalanced_Solution(TreeNode* root) {
        dfs(root);
        return res;
    }

    int dfs(TreeNode* root)
    {
        if(!root) return 0;
        int left = dfs(root -> left), right = dfs(root -> right);
        if(abs(left - right) > 1) res = false;
        return max(left, right) + 1;
    }
};
```

56数组中只出现一次的两个数字

2020年4月10日 8:17

题目描述

一个整型数组里除了两个数字之外，其他的数字都出现了两次。请写程序找出这两个只出现一次的数字。

来自 <https://www.nowcoder.com/practice/e02fdb54d7524710a7d664d082bb7811?tpId=13&tqId=11193&tPage=2&rp=1&ru=%2Fta%2Fcoding-interviews&qru=%2Fta%2Fcoding-interviews%2Fquestion-ranking>

```
class Solution {
public:
    void FindNumsAppearOnce(vector<int> data,int* num1,int *num2) {
        //异或
        //^   x^x = 0
        // x ^ 0  = x
        // x ^ y = z
        int sum = 0;//num1^ num2
        for(auto x : data) sum ^= x;
        //怎么拆分
        //二进制表示中一定有一位存在不同 1  0
        int k = 0;
        *num1 = 0;
        while(!(sum >> k & 1))  k ++;//通过第k位是否为1分组
        for(auto x : data)
            if(x >> k & 1)
                *num1 ^= x;//num1

        *num2 = sum ^ *num1;// num1^num2^num1 = num2
    }
};
```

56数组中唯一只出现一次的数字

2020年4月10日 8:18

在一个数组中除了一个数字只出现一次之外，其他数字都出现了三次。

请找出那个只出现一次的数字。

你可以假设满足条件的数字一定存在。

思考题：

- 如果要求只使用 $O(n)$ 的时间和额外 $O(1)$ 的空间，该怎么做呢？

样例

输入：[1,1,1,2,2,2,3,4,4,4]
输出：3

来自 <https://www.acwing.com/problem/content/70/>

```
class Solution {
public:
    int findNumberAppearingOnce(vector<int>& nums) {
        //只出现一次的 hash
        //
        // unordered_map<int, int> hash;
        // for(auto x : nums) hash[x] ++;
        // for(auto x : nums)
        //     if(hash[x] == 1)
        //         return x;

        // return 0;

        //整数 32位
        //二进制
        //nums  3  1
        //32  31  30 .。 。 。   1
        //%3
        //%3 == 0    0
        //%3 == 1    1
        // int bitSum[32] = {0};
        // //统计每一位1的个数
        // for(int i = 0; i < nums.size(); i ++)
```

```
        // {
        //     int bitMask = 1; //向左移位 &
        //     for(int j = 31; j >= 0; j --)
        //     {
        //         int bit = nums[i] & bitMask;
        //         if(bit)//bit == 1
        //             bitSum[j] += 1;//%3
        //         bitMask = bitMask << 1;//1——》  10   100 1000
        //     }
        // }

        // int res = 0;

        // for(int i = 0; i < 32; i ++)
        // {
        //     res = res << 1;
        //     res += bitSum[i] % 3;
        // }

        // return res;


        //自动机
        int ones = 0, twos = 0;
        //{ones, twos}
        //{0,0} {1,0} {0,1} {0,0}
        for(auto x : nums)
        {
            ones = (ones ^ x) & ~twos;
            twos = (twos ^ x) & ~ones;
        }

        return ones;

    }
};
```

57和为S的两个数字

2020年4月10日 8:18

输入一个数组和一个数字s，在数组中查找两个数，使得它们的和正好是s。

如果有多对数字的和等于s，输出任意一对即可。

你可以认为每组输入中都至少含有一组满足条件的输出。

样例

输入：[1,2,3,4] , sum=7
输出：[3,4]

来自 <https://www.acwing.com/problem/content/71/>

```
class Solution {
public:
    vector<int> findNumbersWithSum(vector<int>& nums, int target) {
        //暴力解答 n*n
        // for(int i = 0; i < nums.size(); i ++)
        //     for(int j = 0; j < i; j ++)
        //         if(nums[i] + nums[j] == target)
        //             return vector<int>{nums[i], nums[j]};

        unordered_map<int, int> hash;
        for(int i = 0; i < nums.size(); i ++)
        {
            hash[nums[i]] ++;
            if(hash[target - nums[i]] > 0) return vector<int>{nums[i], target - nums[i]};
        }
    }
};
```

57和为S的连续正数序列

2020年4月10日 8:18

输入一个正数s，打印出所有和为s的连续正数序列（至少含有两个数）。

例如输入15，由于1+2+3+4+5=4+5+6=7+8=15，所以结果打印出3个连续序列1～5、4～6和7～8。

样例

输入：15

输出：[[1,2,3,4,5],[4,5,6],[7,8]]

来自 <https://www.acwing.com/problem/content/72/>

```
class Solution {
public:
    vector<vector<int> > findContinuousSequence(int sum) {
        vector<vector<int> > res;
        for(int i = 1, j = 1, s = 1; i < sum; i ++)
        {
            while(s < sum) j ++, s += j;
            if(s == sum && j - i > 0)
            {
                vector<int> line;
                for(int k = i; k <= j; k ++) line.push_back(k);
                res.push_back(line);
            }

            s -= i;
        }

        return res;
    }
};
```

作者：满血小进

链接：https://www.acwing.com/activity/content/code/content/274244/

来源：AcWing

58翻转单词顺序

2020年4月10日 8:19

题目描述

牛客最近来了一个新员工Fish，每天早晨总是会拿着一本英文杂志，写些句子在本子上。同事Cat对Fish写的内容颇感兴趣，有一天他向Fish借来翻看，但却读不懂它的意思。例如，“student. a am I”。后来才意识到，这家伙原来把句子单词的顺序翻转了，正确的句子应该是“I am a student.”。Cat对一一的翻转这些单词顺序可不在行，你能帮助他么?

来自 <https://www.nowcoder.com/practice/3194a4f4cf814f63919d0790578d51f3?tpId=13&tqId=11197&tPage=3&rp=1&ru=%2Fta%2Fcoding-interviews&qru=%2Fta%2Fcoding-interviews%2Fquestion-ranking>

```
class Solution {
public:
    string ReverseSentence(string str) {
        //翻转所有字符
        reverse(str.begin(), str.end());
        //翻转单个单词
        for(int i = 0; i < str.size(); i ++)
        {
            int j = i;
            while(j < str.size() && str[j] != ' ') j ++;
            //翻转单个单词
            reverse(str.begin() + i, str.begin() + j);
            i = j;
        }

        return str;
    }
};
```

58左旋转字符串

2020年4月10日 8:19

题目描述

汇编语言中有一种移位指令叫做循环左移（ROL），现在有个简单的任务，就是用字符串模拟这个指令的运算结果。对于一个给定的字符序列S，请你把其循环左移K位后的序列输出。例如，字符序列S=”abcXYZdef”,要求输出循环左移3位后的结果，即“XYZdefabc”。是不是很简单？OK，搞定它！

来自 <https://www.nowcoder.com/practice/12d959b108cb42b1ab72cef4d36af5ec?tpId=13&tqId=11196&tPage=1&rp=1&ru=/ta/coding-interviews&qru=/ta/coding-interviews/question-ranking>

```
class Solution {
public:
    string LeftRotateString(string str, int n) {
        //翻转整个字符串
        reverse(str.begin(), str.end());
        //翻转前半部分
        reverse(str.begin(), str.begin() + str.size() - n);
        //翻转后半部分
        reverse(str.begin() + str.size() - n, str.end());

        return str;
    }
};
```

59滑动窗口的最大值

2020年4月10日 8:19

给定一个数组和滑动窗口的大小，请找出所有滑动窗口里的最大值。

例如，如果输入数组[2, 3, 4, 2, 6, 2, 5, 1]及滑动窗口的大小3,那么一共存在6个滑动窗口，它们的最大值分别为[4, 4, 6, 6, 6, 5]。

注意：

- 数据保证k大于0，且k小于等于数组长度。

样例

输入：[2, 3, 4, 2, 6, 2, 5, 1] , k=3
输出: [4, 4, 6, 6, 6, 5]

来自 <https://www.acwing.com/problem/content/75/>

```
class Solution {
public:
    vector<int> maxInWindows(const vector<int>& num, unsigned int size)
    {
        vector<int> res; // num[i]
        //双向队列 pop
        deque<int> q;//保存num 中的index
        for(int i = 0; i < num.size(); i ++)
        {
            //队头什么时候出队
            while(!q.empty() && i - q.front() >= size) q.pop_front();
            //维护一个队列的单调性
            //4,2,6
            //q 0 1 newindex
            //什么时候pop队尾
            //当新进来的index对应的值比队尾的值>= 队尾pop
            while(!q.empty() && num[i] >= num[q.back()]) q.pop_back();

            q.push_back(i);//每次都push  index

            //i >= size - 1
            //i从0开始
            if(i >= size - 1) res.push_back(num[q.front()]);
        }
        return res;
    }
};
```

59队列的最大值

2020年4月11日 17:28

请定义一个队列并实现函数 max_value 得到队列里的最大值，要求函数max_value、 push_back 和 pop_front 的均摊时间复杂度都是O(1)。

若队列为空，pop_front 和 max_value 需要返回 -1

示例 1：

输入:
["MaxQueue","push_back","push_back","max_value","pop_front","max_value"]
[[],[1],[2],[],[],[]]
输出: [null,null,null,2,1,2]

示例 2：

输入:
["MaxQueue","pop_front","max_value"]
[[],[],[]]
输出: [null,-1,-1]

限制：

1 <= push_back,pop_front,max_value的总操作数 <= 10000
1 <= value <= 10^5

来源：力扣（LeetCode）
链接：https://leetcode-cn.com/problems/dui-lie-de-zui-da-zhi-lcof

60n个骰子的点数

2020年4月10日 9:17

面试题60. n个骰子的点数

难度简单37

把n个骰子扔在地上，所有骰子朝上一面的点数之和为s。输入n，打印出s的所有可能的值出现的概率。

你需要用一个浮点数数组返回答案，其中第 i 个元素代表这 n 个骰子所能掷出的点数集合中第 i 小的那个的概率。

示例 1:
输入: 1
输出: [0.16667,0.16667,0.16667,0.16667,0.16667,0.16667]
示例 2:
输入: 2
输出:
[0.02778,0.05556,0.08333,0.11111,0.13889,0.16667,0.13889,0.11111,0.08333,0.05556,0.02778]

限制:
1 <= n <= 11

通过次数5,000

提交次数9,358

来自 <https://leetcode-cn.com/problems/nge-tou-zi-de-dian-shu-lcof/>

```
class Solution {
public:
  vector<double> twoSum(int n) {
    //暴力递归
    // vector<int> res;
    // for(int i = n; i <= n * 6; i ++)
    //     res.push_back(dfs(n, i));
    vector<vector<int>> f(n + 1, vector<int>(n * 6 + 1));
    f[0][0] = 1;
    for(int i = 1; i <= n; i ++)
      for(int j = 1; j <= i * 6; j ++)
        for(int k = 1; k <= min(j, 6); k ++)
          f[i][j] += f[i - 1][j - k];//前i次总和为j的方案次数

    vector<int> res;
    for(int i = n; i <= n * 6; i ++) res.push_back(f[n][i]);
```

```
    int sum = 0;
    for(auto x : res)
      sum += x;
    vector<double> out;
    for(auto x : res)
      out.push_back(double(x)/ double(sum));
    return out;
  }
  int dfs(int n, int sum)
  {
    if(sum < 0) return 0;
    if(n == 0) return !sum;

    int res = 0;
    for(int i = 1; i <= 6; i ++)
      res += dfs(n - 1, sum - i);
    return res;
  }
};
```

61扑克牌的顺子

2020年4月11日 17:23

从扑克牌中随机抽5张牌，判断是不是一个顺子，即这5张牌是不是连续的。

2～10为数字本身，A为1，J为11，Q为12，K为13，大小王可以看做任意数字。

为了方便，大小王均以0来表示，并且假设这副牌中大小王均有两张。

样例1

输入：[8,9,10,11,12]

输出：true

样例2

输入：[0,8,9,11,12]

输出：true

来自 <https://www.acwing.com/problem/content/77/>

```
class Solution {
public:
    bool IsContinuous( vector<int> nums ) {
        if(nums.empty()) return false;
        //大小王 0
        //去除0
        sort(nums.begin(), nums.end());
        int k = 0;
        while(nums[k] == 0) k ++;
        //找从k + 1位开始的 有没有重复数字
        for(int i = k + 1; i < nums.size(); i ++)
        {
            if(nums[i] == nums[i - 1])
                return false;
        }

        //确保最大值-最小值 <= 4
        return nums.back() - nums[k] <= 4;
    }
};
```

62圆圈中最后剩下的数字

2020年4月11日 17:23

0, 1, ..., n-1这n个数字(n>0)排成一个圆圈，从数字0开始每次从这个圆圈里删除第m个数字。

求出这个圆圈里剩下的最后一个数字。

样例

输入：n=5 , m=3

输出：3

来自 <https://www.acwing.com/problem/content/78/>

```
class Solution {
public:
  int lastRemaining(int n, int m) {
    //旧编号0  1 ...    m-1  m  m+1.... n -1
    //新编号-m -m+1 ... m-1  0  1  .。。
    if(n == 1) return 0;
    //新编号 + m
    return (lastRemaining(n - 1, m)+ m) % n;
  }
};
```

63股票的最大利润

2020年4月11日 17:23

假设把某股票的价格按照时间先后顺序存储在数组中，请问买卖该股票一次可能获得的最大利润是多少?

示例 1:

输入: [7,1,5,3,6,4]
输出: 5
解释: 在第 2 天（股票价格 = 1）的时候买入，在第 5 天（股票价格 = 6）的时候卖出，最大利润 = 6-1 = 5 。
注意利润不能是 7-1 = 6, 因为卖出价格需要大于买入价格。
示例 2:

输入: [7,6,4,3,1]
输出: 0
解释: 在这种情况下, 没有交易完成, 所以最大利润为 0。

来源：力扣（LeetCode）
链接：https://leetcode-cn.com/problems/gu-piao-de-zui-da-li-run-lcof


```
class Solution {
public:
  int maxProfit(vector<int>& nums) {
    if(nums.empty()) return 0;
    int maxv = 0;
    for(int i = 1, minv = nums[0]; i < nums.size(); i ++)
    {
      maxv = max(maxv, nums[i] - minv);
      minv = min(minv, nums[i]);
    }
    return maxv;
  }
};
```

64求1+2+...+n

2020年4月11日 17:23

求1+2+...+n,要求不能使用乘除法、for、while、if、else、switch、case等关键字及条件判断语句（A?B:C）。

样例

输入：10
输出：55

来自 <https://www.acwing.com/problem/content/80/>

```
class Solution {
public:
  int sumNums(int n) {
    int res = n;
    (n > 0) &&  (res+=sumNums(n - 1));
    return res;
  }
};
```

65不用加减乘除做加法

2020年4月11日 17:23

写一个函数，求两个整数之和，要求在函数体内不得使用 + 、 - 、×、÷ 四则运算符号。

样例

输入：num1 = 1 , num2 = 2

输出：3

来自 <https://www.acwing.com/problem/content/81/>

```
class Solution {
public:
  int add(int num1, int num2) {
    while (num2) {
      int sum = num1 ^ num2;
      int carray = (unsigned int)(num1 & num2) << 1;
      num1 = sum;
      num2 = carray;
    }
    return num1;
  }
};
```

66构建乘积数组

2020年4月11日 17:23

给定一个数组A[0, 1, ..., n-1]，请构建一个数组B[0, 1, ..., n-1]，其中B中的元素B[i]=A[0]×A[1]×... ×A[i-1]×A[i+1]×...×A[n-1]。

不能使用除法。

样例

输入：[1, 2, 3, 4, 5]

输出：[120, 60, 40, 30, 24]

思考题：

- 能不能只使用常数空间？（除了输出的数组之外）

来自 <https://www.acwing.com/problem/content/82/>

```
class Solution {
public:
  vector<int> constructArr(vector<int>& A) {
    if(A.empty()) return vector<int>();

    int n = A.size();
    vector<int> B(n);

    // i 左边
    for(int i = 0, p = 1; i < n; i ++)
    {
      //a1 *...* ai-1
      B[i] = p;
      p *= A[i];
    }

   //i右边
    //ai+1 *....* an-1
    for(int i = n - 1,p = 1; i >= 0; i --)
    {
      B[i] *= p;
      p *= A[i];
    }

    return B;
  }
};
```

67 把字符串转换成整数

2020年4月11日 17:23

题目描述

将一个字符串转换成一个整数，要求不能使用字符串转换整数的库函数。 数值为0或者字符串不是一个合法的数值则返回0

输入描述:

输入一个字符串,包括数字字母符号,可以为空

输出描述:

如果是合法的数值表达则返回该数字，否则返回0

示例1

输入

复制

```
+2147483647
  1a33
```

输出

复制

```
2147483647
  0
```

来自 <https://www.nowcoder.com/practice/1277c681251b4372bdef344468e4f26e?tpId=13&tqId=11202&tPage=3&rp=1&ru=%2Fta%2Fcoding-interviews&qru=%2Fta%2Fcoding-interviews%2Fquestion-ranking>

```
class Solution {
public:
    int StrToInt(string str) {
        int k = 0;
        long long num = 0;
        int sign = 1;
        if(str[k] == '-')  k ++, sign = -1;
        else if(str[k] == '+')  k ++, sign = 1;

        while(k < str.size())
        {
            if(str[k] >= '0' && str[k] <= '9')
            {
                num  = num * 10 + str[k] - '0';
                k ++;
            }
            else{
                num = 0;
                break;
            }
        }
        num *= sign;
        return num;
    }
};
```

68树中两个结点的最低公共祖先

2020年4月11日 17:23

给出一个二叉树，输入两个树节点，求它们的最低公共祖先。

一个树节点的祖先节点包括它本身。

注意：

- 输入的二叉树不为空；
- 输入的两个节点一定不为空，且是二叉树中的节点；

样例

二叉树[8, 12, 2, null, null, 6, 4, null, null, null, null]如下图所示：

```
  8
 / \
12  2
   / \
  6   4
```

1. 如果输入的树节点为2和12，则输出的最低公共祖先为树节点8。

2. 如果输入的树节点为2和6，则输出的最低公共祖先为树节点2。

来自 <https://www.acwing.com/problem/content/84/>

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
 * };
 */
class Solution {
public:
  TreeNode* lowestCommonAncestor(TreeNode* root, TreeNode* p, TreeNode* q) {
    if(!root) return NULL;
    if(root == p || root == q) return root;
    auto left = lowestCommonAncestor(root -> left, p, q);
    auto right = lowestCommonAncestor(root -> right, p, q);
    if(left && right) return root;
    if(left) return left;
```

```
        return right;
    }
};
```