LeetCode 解题报告

t236

题目大意是给定一棵二叉树及两个结点，求它们最近的公共祖先。

思路一：后序遍历

求祖先的问题首先想到的是后序遍历，后序遍历的方式是：左->右->根，遍历到当前结点的时候，栈中存储的是该结点的所有父结点。

故这道题目的思路就非常明确了。用一个栈保存还未进行处理的结点，从根结点开始采用后序遍历的思想，遍历到要求的结点时，遍历停止，这个时候就把该结点的所有父结点都保存在了栈中。两个结点的的最近公共祖先是离栈顶最近的第一次相同的元素。

代码：

void PostOrder(TreeNode\* root,TreeNode\* cur,vector<TreeNode\*>& parent){

//后序遍历这棵二叉树

TreeNode\* temp=root;

TreeNode\* v=NULL;

while(temp||!parent.empty()){

if(temp){

parent.push\_back(temp);

temp=temp->left;

}else{

temp=\*(parent.end()-1);

if(temp->right&&temp->right!=v){

temp=temp->right;

}

else{

parent.erase(parent.end()-1);

if(temp==cur){

parent.push\_back(cur);

return;

}

else{

v=temp;

temp=NULL;

}

}

}

}

}

思路二：递归

当从root开始遍历的时候，有三种情况：

1. 如果root为NULL,那么root肯定不是公共祖先，直接返回；
2. 如果root的左子树存在p，右子树存在q，则root肯定是最近的公共祖先；
3. 如果p、q都在root的左子树或者右子树，则递归遍历左子树或者右子树。

代码：class Solution {

public:

TreeNode\* lowestCommonAncestor(TreeNode\* root, TreeNode\* p, TreeNode\* q) {

if(root==NULL) return NULL;

else if(root==p) return p;

else if(root==q) return q;

TreeNode\* lc=lowestCommonAncestor(root->left,p,q);

TreeNode\* rc=lowestCommonAncestor(root->right,p,q);

if(!lc||!rc) return lc==NULL?rc:lc;

else return root;

}

};

t238

题目大意：对数组中的每个元素求除自身外的其他元素的乘积

要求：

1. 不能用除法
2. 要求线性的时间复杂度（不能依次遍历求解）
3. 要求空间复杂度为O（1）

思路：

先从左往右将数组扫描一遍，记录下每个元素左边元素连乘的积；

然后再从数组右边开始扫描，记录下每个元素右边元素连乘的积。

因为:output[i]=left[i]\*right[i]

left[i+1]=left[i]\*nums[i]

right[i]=right[i+1]\*nums[i+1]

class Solution {

public:

vector<int> productExceptSelf(vector<int>& nums) {

int n=nums.size();

vector<int> output(n,1);

for(int i=1;i<n;i++){

//先将当前元素的所有左边元素的乘积统计

output[i]=output[i-1]\*nums[i-1];

}

int right=1;

for(int i=n-2;i>=0;i--){

right\*=nums[i+1];

output[i]\*=right;

}

return output;

}

};

t240

题目大意：在一个矩阵中找到指定的数，这个矩阵从左到右从上到下，数依次增大，如果找到指定的数则返回true,否则返回false。

思路：最简单的方法是暴力搜索，时间复杂度是O（mn），肯定过不去。可以从左下角开始往右上搜索，如果是当前值则返回true，如果目标值比当前值大，则往右边搜索，如果目标值比当前值小，则往上搜索，时间复杂度是O（m+n）。

代码：class Solution {

public:

bool searchMatrix(vector<vector<int>>& matrix, int target) {

//从左下角往右上角找，大的往右，小的往上

int row=matrix.size();

if(row==0) return false;

int column=matrix[0].size();

int i=row-1;

int j=0;

while(i>=0&&j<column){

if(target>matrix[i][j]) j++;

else if(target<matrix[i][j]) i--;

else break;

}

if(i<0||j>=column) return false;

else return true;

}

};

t241

分治法的思想：对于一个规模为n的大问题，分解成同类型的k个不同的小问题。对每个小问题求解，然后合并成整个问题的解。

vector<int> res;

vector<int> lres=diffWaysToCompute(input.substr(0,i));

vector<int> rres=diffWaysToCompute(input.substr(i+1));

没有符号的情况有一个将string字符串转化为10进制数的库函数stoi()。