＆代表参数传递

递归 函数调用＆全局有效

不加＆则在该层递归嵌套或函数中

JZ1

class Solution {

public:

bool Find(int target, vector<vector<int> > array) {

for(int i=0;i<array.size();i++)

{

for(int j=0;j<array[i].size();j++)

if(target==array[i][j])

return true;

}

return false;

}

};

———————————————————————————————————————

JZ2

class Solution {

public:

/\*\*

\* 代码中的类名、方法名、参数名已经指定，请勿修改，直接返回方法规定的值即可

\*

\*

\* @param s string字符串

\* @return string字符串

\*/

string replaceSpace(string s) {

// write code here

string s1;

for(int i=0;i<s.size();i++)

{

if(s[i] == ' ')

{s1+='%';

s1+='2';

s1+='0';

}

else

s1+=s[i];

}

return s1;

}

};

———————————————————————————————————————

JZ3

/\*\*

\* struct ListNode {

\* int val;

\* struct ListNode \*next;

\* ListNode(int x) :

\* val(x), next(NULL) {

\* }

\* };

\*/

class Solution {

public:

vector<int> printListFromTailToHead(ListNode\* head) {

ListNode \*p;

ListNode \*p1;

ListNode \*temp;

vector<int> v;

p = head;

p1 = head;

temp = NULL;

while(p!=NULL)

{

p1 = p->next;

p->next = temp;

temp = p;

p = p1;

}

while(temp!= NULL)

{

v.push\_back(temp->val);

temp = temp->next;

}

return v;

}

};

———————————————————————————————————————

JZ4

/\*\*

\* Definition for binary tree

\* struct TreeNode {

\* int val;

\* TreeNode \*left;

\* TreeNode \*right;

\* TreeNode(int x) : val(x), left(NULL), right(NULL) {}

\* };

\*/

class Solution {

public:

TreeNode\* reConstructBinaryTree(vector<int> pre,vector<int> vin) {

vector<int> pre\_left,pre\_right,vin\_left,vin\_right;

if(pre.empty()) return nullptr;

TreeNode\* root = new TreeNode(pre[0]);

int head = 0;

for(int i=0;i<vin.size();i++){

if(vin[i] == pre[0]){

head = i;

break;

}

}

for(int i=0;i<head;i++){

pre\_left.push\_back(pre[i+1]);

vin\_left.push\_back(vin[i]);

}

for(int i=head+1;i<pre.size();i++){

pre\_right.push\_back(pre[i]);

vin\_right.push\_back(vin[i]);

}

root->left = reConstructBinaryTree(pre\_left, vin\_left);

root->right = reConstructBinaryTree(pre\_right, vin\_right);

return root;

}

};

———————————————————————————————————————

JZ5

class Solution

{

public:

int num;

void push(int node) {

s1.push(node);

}

int pop() {

if(s2.empty())

{while(!s1.empty())

{s2.push(s1.top());

s1.pop();

}}

num = s2.top();

s2.pop();

return num;

}

private:

stack<int> s1;

stack<int> s2;

};

———————————————————————————————————————

JZ6

递归

class Solution {

public:

int sort1(vector<int> arr ,int l,int r)

{

if(l<r)

{

int temp = (l+r)/2;

if(arr[temp]<arr[r])

return sort1(arr, l, temp);

else if(arr[temp]>arr[r])

return sort1(arr, temp+1, r);

else

return sort1(arr, l ,r-1);

}

return arr[l];

}

int minNumberInRotateArray(vector<int> arr) {

if(arr.size() == 0)

return 0;

else

return sort1(arr,0,arr.size()-1);

}

};

While法

class Solution {

public:

int sort1(vector<int> arr ,int l,int r)

{

while(l<r)

{

int temp = (l+r)/2;

if(arr[temp] < arr[r])

r = temp;

else if(arr[temp] > arr[r])

l = temp+1;

else

r = r-1;

}

return arr[l];

}

int minNumberInRotateArray(vector<int> arr) {

if(arr.size() == 0)

return 0;

else

return sort1(arr,0,arr.size()-1);

}

};

———————————————————————————————————————

JZ7

class Solution {

public:

int Fibonacci(int n) {

int a[999];

a[0] = 0;

a[1] = 1;

for(int i=0;i<40;i++)

{

a[i+2]=a[i+1]+a[i];

}

return a[n];

}

};

———————————————————————————————————————

JZ8

class Solution {

public:

int jumpFloor(int number) {

int a[999];

a[1]=1;

a[2]=2;

for(int i=1;i<number+1;i++)

{

a[i+2] = a[i+1]+a[i];

}

return a[number];

}

};

———————————————————————————————————————

JZ9

class Solution {

public:

int jumpFloorII(int number) {

int a[999];

a[1] = 1;

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

a[i+1] = a[i]\*2;

}

return a[number];

}

};

———————————————————————————————————————

JZ10

class Solution {

public:

int rectCover(int number) {

int a[9999];

a[0] = 0;

a[1] = 1;

a[2] = 2;

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

a[i+2] = a[i+1] + a[i];

}

return a[number];

}

};

———————————————————————————————————————

JZ11

class Solution {

public:

int NumberOf1(int n) {

int num = 0;

while(n != 0){

num++;

n = n & (n-1);

}

return num;

}

};

———————————————————————————————————————

JZ12

class Solution {

public:

double Power(double base, int exponent) {

double num = 1;

if(exponent >= 0){

for(int i=0;i<exponent;i++){

num \*= base;

}}

else{

exponent = -exponent;

for(int i = 0;i<exponent;i++){

num \*= base;

}

num = 1/num;

}

return num;

}

};

———————————————————————————————————————

JZ13

class Solution {

public:

/\*\*

\* 代码中的类名、方法名、参数名已经指定，请勿修改，直接返回方法规定的值即可

\*

\*

\* @param array int整型vector

\* @return int整型vector

\*/

vector<int> reOrderArray(vector<int>& arr) {

vector<int> v1,v2;

for(int i=0;i<arr.size();i++){

if(arr[i] % 2 == 0)

v2.push\_back(arr[i]);

else

v1.push\_back(arr[i]);

}

for(int i=0;i<v2.size();i++){

v1.push\_back(v2[i]);

}

return v1;

}

};

———————————————————————————————————————

JZ14

/\*\*

\* struct ListNode {

\* int val;

\* struct ListNode \*next;

\* ListNode(int x) : val(x), next(nullptr) {}

\* };

\*/

class Solution {

public:

/\*\*

\* 代码中的类名、方法名、参数名已经指定，请勿修改，直接返回方法规定的值即可

\*

\*

\* @param pHead ListNode类

\* @param k int整型

\* @return ListNode类

\*/

ListNode\* FindKthToTail(ListNode\* pHead, int k) {

// write code here

ListNode\* p;

p = pHead;

int num = 0;

int temp = 0;

while(p != NULL){

num++;

p = p->next;

}

if(num < k) return nullptr;

temp = num - k;

p = pHead;

while(temp != 0){

temp--;

p = p->next;

}

return p;

}

};

———————————————————————————————————————

JZ15

/\*

struct ListNode {

int val;

struct ListNode \*next;

ListNode(int x) :

val(x), next(NULL) {

}

};\*/

class Solution {

public:

ListNode\* ReverseList(ListNode\* pHead) {

ListNode\* p;

ListNode\* p1;

ListNode\* temp;

p = pHead;

p1 = p;

temp = NULL;

while(p!=NULL){

p1 = p->next;

p->next = temp;

temp = p;

p = p1;

}

return temp;

}

};

———————————————————————————————————————

JZ16

/\*

struct ListNode {

int val;

struct ListNode \*next;

ListNode(int x) :

val(x), next(NULL) {

}

};\*/

class Solution {

public:

ListNode\* Merge(ListNode\* p1, ListNode\* p2) {

ListNode\* temp;

ListNode\* node = new ListNode(-1);

temp =node;

while(p1 && p2){

if(p1->val > p2->val){

temp->next = p2;

temp = p2;

p2 = p2->next;

}

else{

temp->next = p1;

temp = p1;

p1 = p1->next;

}

}

if(p1)

temp->next = p1;

else if(p2)

temp->next = p2;

return node->next;

}

};

递归

/\*

struct ListNode {

int val;

struct ListNode \*next;

ListNode(int x) :

val(x), next(NULL) {

}

};\*/

class Solution {

public:

ListNode\* Merge(ListNode\* p1, ListNode\* p2) {

if(!p1) return p2;

if(!p2) return p1;

if(p1->val < p2->val){

p1->next = Merge(p1->next, p2);

return p1;

}

else{

p2->next = Merge(p1, p2->next);

return p2;

}

}

};

———————————————————————————————————————

JZ17

/\*

struct TreeNode {

int val;

struct TreeNode \*left;

struct TreeNode \*right;

TreeNode(int x) :

val(x), left(NULL), right(NULL) {

}

};\*/

class Solution {

public:

bool panduan(TreeNode\* root1,TreeNode\* root2){

if(!root2) return true;

if(!root1) return false;

if(root1->val != root2->val) return false;

return (panduan(root1->left,root2->left) && panduan(root1->right,root2->right));

}

bool HasSubtree(TreeNode\* pRoot1, TreeNode\* pRoot2) {

if(!pRoot1 || !pRoot2) return false;

else

return (panduan(pRoot1,pRoot2) || HasSubtree(pRoot1->left, pRoot2) || HasSubtree(pRoot1->right, pRoot2));

}

};

———————————————————————————————————————

JZ18

/\*\*

\* struct TreeNode {

\* int val;

\* struct TreeNode \*left;

\* struct TreeNode \*right;

\* TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}

\* };

\*/

class Solution {

public:

/\*\*

\* 代码中的类名、方法名、参数名已经指定，请勿修改，直接返回方法规定的值即可

\*

\*

\* @param pRoot TreeNode类

\* @return TreeNode类

\*/

TreeNode\* Mirror(TreeNode\* pRoot) {

if(!pRoot) return nullptr;

TreeNode\* root = new TreeNode(pRoot->val);

root->left = Mirror(pRoot->right);

root->right = Mirror(pRoot->left);

return root;

}

};

———————————————————————————————————————

JZ19

class Solution {

public:

void print(vector<vector<int> > mat,int r,int m,int n,vector<int> &v){

if(m == 0||n == 0) return;

if(m == 1){

for(int i=0;i<n;i++)

v.push\_back(mat[r-1+i][r-1]);

return;

}

if(n == 1){

for(int i=0;i<m;i++)

v.push\_back(mat[r-1][r-1+i]);

return;

}

for(int i=0;i<m;i++){

v.push\_back(mat[r-1][r-1+i]);

}

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

v.push\_back(mat[r-1+i][r-1+m-1]);

}

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

v.push\_back(mat[r-1+n-1][r-1+m-1-i]);

}

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

v.push\_back(mat[r-1+n-1-i][r-1]);

}

return print(mat, r+1, m-2, n-2, v);

}

vector<int> printMatrix(vector<vector<int> > matrix) {

vector<int> v;

print(matrix,1,matrix[0].size(),matrix.size(),v);

return v;

}

};

———————————————————————————————————————

JZ20

class Solution {

public:

vector<int> v;

int num;

void push(int value) {

v.push\_back(value);

}

void pop() {

v.pop\_back();

}

int top() {

return v[v.size()-1];

}

int min() {

num = v[0];

for(int i=0;i<v.size()-1;i++){

if(v[i+1] < num)

num = v[i+1];

}

return num;

}

};

———————————————————————————————————————

JZ21

class Solution {

public:

bool IsPopOrder(vector<int> pushV,vector<int> popV) {

stack<int> s;

int j = 0;

for(int i=0;i<pushV.size();i++){

if(pushV[i] == popV[j]){

j++;

while(!s.empty() && s.top() == popV[j]){

s.pop();

j++;

}

}

else s.push(pushV[i]);

}

if(s.empty()) return true;

else return false;

}

};

简单法

class Solution {

public:

bool IsPopOrder(vector<int> pushV,vector<int> popV) {

int i=0,j=0;

stack<int> s;

for(i = 0;i < pushV.size();i++){

s.push(pushV[i]);

while(!s.empty() && s.top() == popV[j]){

s.pop();

j++;

}

}

return s.empty();

}

};

———————————————————————————————————————

JZ22

/\*

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) {

deque<TreeNode\*> q;

vector<int> v;

if(root == NULL) return {};

q.push\_back(root);

while(!q.empty()){

if(q[0]->left != NULL)

q.push\_back(q[0]->left);

if(q[0]->right != NULL)

q.push\_back(q[0]->right);

v.push\_back(q[0]->val);

q.pop\_front();

}

return v;

}

};

———————————————————————————————————————

JZ23

class Solution {

public:

bool panduan(vector<int> v,int l,int r){

if(l<r){

int temp = v[r];

int i;

for(i=l;i<r;i++){

if(temp < v[i]) break;

}

for(int j=i;j<r;j++){

if(v[j] < temp) return false;

}

return (panduan(v,l,i-1) && panduan(v,i,r-1));

}

return true;

}

bool VerifySquenceOfBST(vector<int> sequence) {

if(sequence.empty()) return false;

return panduan(sequence,0,sequence.size()-1);

}

};

———————————————————————————————————————

JZ24

/\*

struct TreeNode {

int val;

struct TreeNode \*left;

struct TreeNode \*right;

TreeNode(int x) :

val(x), left(NULL), right(NULL) {

}

};\*/

class Solution {

public:

void bianli(TreeNode\* root,int sum,int target,vector<int> v,vector<vector<int>> &v2 ){

if(!root) return;

sum += root->val;

v.push\_back(root->val);

if(sum == target && !root->left && !root->right) v2.push\_back(v);

bianli(root->left, sum, target, v, v2);

bianli(root->right, sum, target, v, v2);

return;

}

vector<vector<int> > FindPath(TreeNode\* root,int expectNumber) {

vector<int> v;

vector<vector<int> > v2;

if(!root) return { };

bianli(root,0,expectNumber,v,v2);

return v2;

}

};

———————————————————————————————————————

JZ25

/\*

struct RandomListNode {

int label;

struct RandomListNode \*next, \*random;

RandomListNode(int x) :

label(x), next(NULL), random(NULL) {

}

};

\*/

class Solution {

public:

RandomListNode\* Clone(RandomListNode\* pHead) {

RandomListNode\* p1 = pHead;

RandomListNode\* p2;

RandomListNode\* head = new RandomListNode(-1);

while(p1 != NULL){

RandomListNode\* node = new RandomListNode(p1->label);

node->next = p1->next;

p1->next = node;

p1 = node->next;

}

p1 = pHead;

while(p1 != NULL){

if(p1->random)

p1->next->random = p1->random->next;

p1 = p1->next->next;

}

p1 = pHead;

p2 = head;

while(p1 != NULL){

p2->next = p1->next;

p2 = p2->next;

p1->next = p1->next->next;

p1 = p1->next;

}

return head->next;

}

};

———————————————————————————————————————

JZ26

/\*

struct TreeNode {

int val;

struct TreeNode \*left;

struct TreeNode \*right;

TreeNode(int x) :

val(x), left(NULL), right(NULL) {

}

};\*/

class Solution {

public:

map<int, TreeNode\*> m;

vector<int> v;

void bianli(TreeNode\* root){

if(!root) return;

bianli(root->left);

m[root->val] = root;

v.push\_back(root->val);

bianli(root->right);

return;

}

TreeNode\* Convert(TreeNode\* pRootOfTree) {

if(!pRootOfTree) return {};

bianli(pRootOfTree);

m[v[0]]->right = m[v[1]];

m[v[v.size()-1]]->left = m[v[v.size()-2]];

for(int i=1;i<v.size()-1;i++){

m[v[i]]->left = m[v[i-1]];

m[v[i]]->right = m[v[i+1]];

}

return m[v[0]];

}

};

———————————————————————————————————————

JZ27

———————————————————————————————————————

JZ28

class Solution {

public:

void quick(vector<int> &v,int l,int r){

if(l<r){

int i=l;

int j=r;

int temp = v[i];

while(i<j){

while(i<j && temp <= v[j])

j--;

v[i] = v[j];

while(i<j && temp > v[i])

i++;

v[j] = v[i];

}

v[i] = temp;

quick(v, l, i-1);

quick(v,i+1,r);

}

}

int MoreThanHalfNum\_Solution(vector<int> num) {

quick(num,0,num.size()-1);

return num[num.size()/2];

}

};

class Solution {

public:

void quick(vector<int> &v,int l,int r){

if(l<r){

int i=l;

int j=r;

int temp = v[l];

while(i<j){

while(i<j && temp <= v[j])

j--;

while(i<j && temp >= v[i])

i++;

swap(v[i], v[j]);

}

swap(v[l], v[i]);

quick(v, l, i-1);

quick(v,i+1,r);

}

}

int MoreThanHalfNum\_Solution(vector<int> num) {

quick(num,0,num.size()-1);

return num[num.size()/2];

}

};

———————————————————————————————————————

JZ29

class Solution {

public:

vector<int> GetLeastNumbers\_Solution(vector<int> input, int k) {

vector<int> v;

sort(input.begin(),input.end());

for(int i=0;i<k;i++){

v.push\_back(input[i]);

}

return v;

}

};

———————————————————————————————————————

JZ31

class Solution {

public:

int NumberOf1Between1AndN\_Solution(int n) {

int m = 1;

int a,b;

int sum = 0;

while(n/m != 0){

a = n/m;

b = n%m;

sum += (a+8)/10\*m+(a%10 == 1 ? b+1:0);

m \*= 10;

}

return sum;

}a

};

———————————————————————————————————————

JZ33

class Solution {

public:

int GetUglyNumber\_Solution(int index) {

if(!index) return 0;

int dp[index+1];

int p2 = 0,p3 = 0,p5 = 0;

dp[0] = 1;

int temp;

for(int i=0;i<index;i++){

temp = min(2\*dp[p2], min(3\*dp[p3], 5\*dp[p5]));

if(temp == 2\*dp[p2]) dp[i+1] = 2\*dp[p2++];

if(temp == 3\*dp[p3]) dp[i+1] = 3\*dp[p3++];

if(temp == 5\*dp[p5]) dp[i+1] = 5\*dp[p5++];

}

return dp[index-1];

}

};

———————————————————————————————————————

JZ34

class Solution {

public:

int FirstNotRepeatingChar(string str) {

int count[999];

for(int i=0;i<str.length();i++){

count[str[i]]++;

}

for(int i=0;i<str.length();i++){

if(count[str[i]] == 1) return i;

}

return -1;

}

};

———————————————————————————————————————

JZ35

class Solution {

public:

int InversePairs(vector<int> data) {

long long int dp[999999];

dp[0] = 0;

dp[1] = 0;

int len = data.size();

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

long long int sum = 0;

for(int j=0;j<i;j++){

if(data[i] < data[j]) sum++;

}

dp[i+1] = dp[i] + sum;

}

return (dp[len]%1000000007);

}

};

———————————————————————————————————————

JZ36

/\*

struct ListNode {

int val;

struct ListNode \*next;

ListNode(int x) :

val(x), next(NULL) {

}

};\*/

class Solution {

public:

ListNode\* FindFirstCommonNode( ListNode\* pHead1, ListNode\* pHead2) {

ListNode\* p1;

ListNode\* p2;

p1 = pHead1;

p2 = pHead2;

while(p1 != p2){

if(p1) p1 = p1->next;

else p1 = pHead2;

if(p2) p2 = p2->next;

else p2 = pHead1;

}

return p1;

}

};

傻逼方法

/\*

struct ListNode {

int val;

struct ListNode \*next;

ListNode(int x) :

val(x), next(NULL) {

}

};\*/

class Solution {

public:

ListNode\* FindFirstCommonNode( ListNode\* pHead1, ListNode\* pHead2) {

ListNode\* head = new ListNode(-1);

ListNode\* p1;

ListNode\* p2;

p1 = pHead1;

p2 = pHead2;

while(p1){

while(p2){

if(p1->val == p2->val) return p1;

p2 = p2->next;

}

p2 = pHead2;

p1 = p1->next;

}

return {};

}

};

———————————————————————————————————————

JZ37

class Solution {

public:

int GetNumberOfK(vector<int> data ,int k) {

int l = 0;

int r = data.size()-1;

if(data.empty()) return 0;

while(data[r] > k && l<r) r--;

while(data[l] < k && l<r) l++;

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

return (r-l+1);

}

};

———————————————————————————————————————

JZ38

/\*

struct TreeNode {

int val;

struct TreeNode \*left;

struct TreeNode \*right;

TreeNode(int x) :

val(x), left(NULL), right(NULL) {

}

};\*/

class Solution {

public:

vector<int> bianli(TreeNode\* root,int n,vector<int> &v){

if(!root) {

v.push\_back(n);

return {};}

n++;

bianli(root->left,n,v);

bianli(root->right,n,v);

return v;

}

int TreeDepth(TreeNode\* pRoot) {

vector<int> v;

bianli(pRoot,0,v);

sort(v.begin(), v.end());

return v[v.size()-1];

}

};

———————————————————————————————————————

JZ39

class Solution {

public:

int depth(TreeNode\* root){

if(!root) return 0;

int left = depth(root->left);

if(left == -1) return -1;

int right = depth(root->right);

if(right == -1) return -1;

if(abs(left - right) > 1) return -1;

return max(left,right)+1;

}

bool IsBalanced\_Solution(TreeNode\* pRoot) {

return (depth(pRoot) != -1);

}

};

Hashmap：

class Solution {

public:

map<TreeNode\*, int> m;

int depth(TreeNode\* root){

if(!root) return 0;

int l = depth(root->left);

int r = depth(root->right);

m[root] = max(l,r)+1;

return m[root];

}

bool bianli(TreeNode\* root){

if(!root) return true;

return (abs(m[root->left]-m[root->right])<=1 && bianli(root->left) && bianli(root->right));

}

bool IsBalanced\_Solution(TreeNode\* pRoot) {

depth(pRoot);

return bianli(pRoot);

}

};

———————————————————————————————————————

JZ40

class Solution {

public:

/\*\*

\* 代码中的类名、方法名、参数名已经指定，请勿修改，直接返回方法规定的值即可

\*

\*

\* @param array int整型vector

\* @return int整型vector

\*/

vector<int> FindNumsAppearOnce(vector<int>& array) {

map<int ,int > m;

vector<int> v;

for(int i=0;i<array.size();i++){

m[array[i]]=0;

}

for(int i=0;i<array.size();i++){

m[array[i]]++;

}

for(int i=0;i<array.size();i++){

if(m[array[i]] == 1) v.push\_back(array[i]);

}

return v;

}

};

———————————————————————————————————————

JZ41

class Solution {

public:

vector<vector<int> > FindContinuousSequence(int sum) {

vector<vector<int> > num;

vector<int> num1;

for(int i=sum-1;i>0;i--){

int flag = 0;

num1.clear();

for(int j=1;j<sum;j++){

if((2\*j+i-1)\*i/2 == sum){

for(int k=0;k<i;k++){

num1.push\_back(j+k);

}

flag=1;

break;

}

}

if(flag)

num.push\_back(num1);

}

return num;

}

};

———————————————————————————————————————

JZ42

class Solution {

public:

vector<int> FindNumbersWithSum(vector<int> arr,int sum) {

vector<int> v;

int flag = 0;

if(arr.empty()) return {};

for(int i=0;i<arr.size()-1;i++){

for(int j=i+1;j<arr.size();j++){

if(arr[i] + arr[j] == sum){

v.push\_back(arr[i]);

v.push\_back(arr[j]);

flag = 1;

}

}

if(flag) break;

}

if(v.empty()) return {};

return v;

}

};

———————————————————————————————————————

JZ43

class Solution {

public:

string LeftRotateString(string str, int n) {

char temp;

if(str.empty()) return "";

for(int i=0;i<n;i++){

temp = str[0];

for(int j=0;j<str.size()-1;j++){

str[j] = str[j+1];

}

str[str.size()-1] = temp;

}

return str;

}

};

———————————————————————————————————————

JZ44

class Solution {

public:

string ReverseSentence(string str) {

vector<char> v;

stack<char> s1;

string s;

for(int i=str.size()-1;i>-1;i--){

s1.push(str[i]);

if(str[i-1] == ' '){

while(!s1.empty()){

v.push\_back(s1.top());

s1.pop();

}

v.push\_back(' ');

i--;

}

if(i == 0){

while(!s1.empty()){

v.push\_back(s1.top());

s1.pop();

}

}

}

s.assign(v.begin(),v.end());

return s;

}

};

———————————————————————————————————————

JZ45

class Solution {

public:

bool IsContinuous( vector<int> num ) {

int temp = 0;

sort(num.begin(), num.end());

for(int i=0;i<num.size()-1;i++){

if(num[i] == 0) temp++;

else if(num[i+1] == num[i]) return false;

else if(num[i+1] != num[i]+1) temp -= num[i+1]-num[i]-1;

}

if(temp >= 0) return true;

return false;

}

};

———————————————————————————————————————

JZ46

class Solution {

public:

int LastRemaining\_Solution(int n, int m) {

list<int> l;

if(!n) return -1;

for(int i=0;i<n;i++) {

l.push\_back(i);

}

int it = 0;

while(n-1){

it = (it + m - 1) % n;

auto begin = l.begin();

advance(begin, it);

l.erase(begin);

n--;

}

return l.front();

}

};

动态规划

class Solution {

public:

int LastRemaining\_Solution(int n, int m) {

int dp[n+1];

if(!n) return -1;

dp[1] = 0;

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

dp[i+1]=(dp[i]+m)%(i+1);

}

return dp[n];

}

};

———————————————————————————————————————

JZ47

class Solution {

public:

int sum(int &sum1,int n){

sum1 += n;

n && sum(sum1,n-1);

return sum1;

}

int Sum\_Solution(int n) {

int sum2 = 0;

return sum(sum2,n);

}

};

class Solution {

public:

int Sum\_Solution(int n) {

int sum = n;

n && (sum += Sum\_Solution(n-1));

return sum;

}

};

———————————————————————————————————————

JZ48

class Solution {

public:

int Add(int num1, int num2) {

while(num2 !=0 )

{

int temp = (num1 & num2) << 1;

num1 ^= num2;

num2 = temp;

}

return num1;

}

};

———————————————————————————————————————

JZ49

class Solution {

public:

int StrToInt(string s) {

int sum = 0;

if(s[0] == '+' || s[0] == '-'){

for(int i=1;i<s.size();i++){

if('0'> s[i] || '9' < s[i]) return 0;

else sum = (s[i]-'0') + sum\*10;

}

if(s[0] == '+') return sum;

else if(s[0] == '-') return -sum;

else return 0;

}

else if('0'<s[0] && s[0]<='9'){

for(int i=0;i<s.size();i++){

if('0'> s[i] || '9' < s[i]) return 0;

else sum = (s[i]-'0') + sum\*10;

}

return sum;

}

else return 0;

}

};

———————————————————————————————————————

JZ50

class Solution {

public:

/\*\*

\* 代码中的类名、方法名、参数名已经指定，请勿修改，直接返回方法规定的值即可

\*

\*

\* @param numbers int整型vector

\* @return int整型

\*/

int duplicate(vector<int>& num) {

// write code here

for(int i=0;i<num.size();i++){

for(int j=i+1;j<num.size();j++){

if(num[i] == num[j])

return num[i];

}

}

return -1;

}

};

———————————————————————————————————————

JZ51

class Solution {

public:

vector<int> multiply(const vector<int>& A) {

vector<int> v;

for(int i=0;i<A.size();i++){

int num = 1;

for(int j=0;j<A.size();j++){

if(j!=i)

num \*= A[j];

}

v.push\_back(num);

}

return v;

}

};

———————————————————————————————————————

JZ52

———————————————————————————————————————

JZ53

class Solution {

public:

/\*\*

\* 代码中的类名、方法名、参数名已经指定，请勿修改，直接返回方法规定的值即可

\*

\*

\* @param str string字符串

\* @return bool布尔型

\*/

bool isNumeric(string str) {

int dian = 0;

int e = 0;

int fuhao = 0;

int num=0;

for(int i=0;i<str.size();i++){

if('0'>str[i] || str[i]>'9'){

if(str[i] == 'e' || str[i] == 'E') e++;

else if(str[i] == '+' || str[i] == '-') fuhao++;

else if(str[i] == '.') dian++;

else return false;

}

if(str[i]>='0' &&str[i]<='9')num++;

}

if(dian >= 2) return false;

if(str[0] == '+'|| str[0] == '-') fuhao--;

for(int i=0;i<str.size();i++){

if(str[i] == 'e'|| str[i] == 'E'){

if(str[i-1]>'9'||str[i-1]<'0') return false;

e--;

if(i == str.size()-1) return false;

if(str[i+1] == '+'|| str[i+1] == '-') {fuhao--;

if(str[i+2]>'9'||str[i+2]<'0') return false;}

for(int j=i+1;j<str.size();j++){

if(str[j] == '.') return false;

}

}

}

if(e != 0) return false;

if(fuhao != 0) return false;

if(num == 0) return false;

return true;

}

};

———————————————————————————————————————

JZ54

class Solution

{

public:

//Insert one char from stringstream

map<char,int> m;

vector<char> v;

void Insert(char ch) {

m[ch]++;

v.push\_back(ch);

}

//return the first appearence once char in current stringstream

char FirstAppearingOnce() {

for(int i=0;i<v.size();i++){

if(m[v[i]] == 1) return v[i];

}

return '#';

}

};

———————————————————————————————————————

JZ55

/\*

struct ListNode {

int val;

struct ListNode \*next;

ListNode(int x) :

val(x), next(NULL) {

}

};

\*/

class Solution {

public:

map<ListNode\*,int> m;

ListNode\* EntryNodeOfLoop(ListNode\* pHead) {

while(m[pHead] != 2){

m[pHead]++;

pHead = pHead->next;

if(!pHead) return nullptr;

// if(pHead->next == pHead) return pHead;

}

return pHead;

}

};

———————————————————————————————————————

JZ56

简化

/\*

struct ListNode {

int val;

struct ListNode \*next;

ListNode(int x) :

val(x), next(NULL) {

}

};

\*/

class Solution {

public:

map<int,int> m;

ListNode\* deleteDuplication(ListNode\* pHead) {

ListNode\* p;

p = pHead;

ListNode\* p1 = new ListNode(-1);

while(p){

m[p->val]++;

p = p->next;

}

p = pHead;

p1->next = p;

pHead = p1;

while(p){

while(p && m[p->val] >= 2){

p1->next = p1->next->next;

p = p1->next;

}

if(p && m[p->val] == 1)

{p1 = p1->next;

p = p->next;}

}

return pHead->next;

}

};

/\*

struct ListNode {

int val;

struct ListNode \*next;

ListNode(int x) :

val(x), next(NULL) {

}

};

\*/

class Solution {

public:

map<int,int> m;

ListNode\* deleteDuplication(ListNode\* pHead) {

ListNode\* p;

p = pHead;

ListNode\* p1 = new ListNode(-1);

while(p){

m[p->val]++;

p = p->next;

}

p = pHead;

p1->next = p;

pHead = p1;

while(p){

while(m[p->val] >= 2){

p1->next = p1->next->next;

p = p1->next;

if(!p) break;

}

if(!p) break;

if(m[p->val] == 1)

{p1 = p1->next;

p = p->next;}

}

return pHead->next;

}

};

JZ57

/\*

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:

vector<TreeLinkNode\*> v;

void bianli(TreeLinkNode\* root){

if(!root) return;

bianli(root->left);

v.push\_back(root);

bianli(root->right);

return;

}

TreeLinkNode\* GetNext(TreeLinkNode\* pNode) {

TreeLinkNode\* p1;

TreeLinkNode\* p2;

p2 = pNode;

while(p2){

p1 = p2;

p2 = p2->next;

}

bianli(p1);

if(v.size() == 1) return nullptr;

for(int i=0;i<v.size();i++){

if(v[i] == pNode) return v[i+1];

}

return nullptr;

}

};

———————————————————————————————————————

JZ58

/\*

struct TreeNode {

int val;

struct TreeNode \*left;

struct TreeNode \*right;

TreeNode(int x) :

val(x), left(NULL), right(NULL) {

}

};

\*/

class Solution {

public:#常规创建二叉树方法

TreeNode\* jingxiang(TreeNode\* proot){

if(!proot) return nullptr;

TreeNode\* root = new TreeNode(proot->val);

root->left = jingxiang(proot->right);

root->right = jingxiang(proot->left);

return root;

}

bool panduan(TreeNode\* root,TreeNode\* root1){

if(!root && !root1) return true;

if(!root || !root1) return false;

if(root1->val != root->val) return false;

return panduan(root->left, root1->left) && panduan(root->right, root1->right);

}

bool isSymmetrical(TreeNode\* pRoot) {

TreeNode\* root = jingxiang(pRoot);

return panduan(pRoot,root);

}

};

/\*

struct TreeNode {

int val;

struct TreeNode \*left;

struct TreeNode \*right;

TreeNode(int x) :

val(x), left(NULL), right(NULL) {

}

};

最优

\*/

class Solution {

public:

bool panduan(TreeNode\* left,TreeNode\* right){

if(!left && !right) return true;

if(!left || !right) return false;

if(left->val != right->val) return false;

return panduan(left->left,right->right) && panduan(left->right, right->left);

}

bool isSymmetrical(TreeNode\* pRoot) {

if(!pRoot) return true;

return panduan(pRoot->left, pRoot->right);

}

};

———————————————————————————————————————

JZ59

/\*

struct TreeNode {

int val;

struct TreeNode \*left;

struct TreeNode \*right;

TreeNode(int x) :

val(x), left(NULL), right(NULL) {

}

};

\*/

class Solution {

public:

map<TreeNode\*,int> m;

void biaoji(TreeNode\* root,int n){

if(!root) return;

m[root] = n;

n++;

biaoji(root->left,n);

biaoji(root->right,n);

return;

}

void bianli(TreeNode\* root,vector<TreeNode\*> &v){

queue<TreeNode\*> q;

TreeNode\* temp;

q.push(root);

while(!q.empty()){

temp = q.front();

v.push\_back(temp);

q.pop();

if(temp->left) q.push(temp->left);

if(temp->right) q.push(temp->right);

}

return;

}

vector<vector<int> > Print(TreeNode\* pRoot) {

vector<TreeNode\*> v;

vector<int> temp;

vector<vector<int>>vv;

stack<int> s;

if(!pRoot) return {};

biaoji(pRoot,1);

bianli(pRoot, v);

for(int i=0;i<v.size();i++){

if(m[v[i]]%2==1){

temp.push\_back(v[i]->val);}

else{

s.push(v[i]->val);

}

if(m[v[i]]!=m[v[i+1]]){

while(!s.empty()){

temp.push\_back(s.top());

s.pop();

}

vv.push\_back(temp);

temp.clear();

}

}

return vv;

}

};

———————————————————————————————————————

JZ60

/\*

struct TreeNode {

int val;

struct TreeNode \*left;

struct TreeNode \*right;

TreeNode(int x) :

val(x), left(NULL), right(NULL) {

}

};

\*/

class Solution {

public:

map<TreeNode\*, int> m;

void bianli(TreeNode\* root,int n){

if(!root) return;

m[root] = n;

n++;

bianli(root->left,n);

bianli(root->right,n);

}

vector<vector<int> > Print(TreeNode\* pRoot) {

if(!pRoot) return {};

queue<TreeNode\*> q;

vector<TreeNode\*> v;

vector<int> v1;

vector<vector<int> > vv;

TreeNode\* temp;

q.push(pRoot);

bianli(pRoot,1);

while(!q.empty()){

temp = q.front();

q.pop();

v.push\_back(temp);

if(temp->left) q.push(temp->left);

if(temp->right) q.push(temp->right);

}

for(int i=0;i<v.size();i++){

v1.push\_back(v[i]->val);

if(m[v[i]] != m[v[i+1]] || i==v.size()-1) {

vv.push\_back(v1);

v1.clear();

}

}

return vv;

}

};

———————————————————————————————————————