

1.有n栋房子排成一行，假如第i栋房子的高度是正整数a[i]，小明站在房子的顶部时，可以看到左右高度小于等于当前高度的房子，直到左右出现更高的房子，例如，有四栋房子a,b,c,d高度分别为1,4,3,3,那么在房子a顶部时，无法看到其他的房子；

在房子b顶部时，可以看到其他所有房子，在房子c时可以看到房子d，在房子d时可以看到房子c,现在小明想知道分别站在每个房子顶部时可以看到左右房子的数量。

#include<iostream>

#include<vector>

using namespace std;

int main(){

int n, tmp = 0;

cin >> n;

vector<int> body(n, 0), res;

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

{

cin >> body[i];

}

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

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

if(body[j] <= body[i])

tmp++;

else

break;

}

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

{

if(body[j] <= body[i])

tmp++;

else

break;

}

res.push\_back(tmp);

tmp = 0;

}

for(auto num : res)

cout << num << endl;

return 0;

}

2.你有n种无门槛优惠券，每种优惠券有一个面值ai，当一件商品的售价大于等于ai时，你可以出示这种优惠券，抵折后优惠券不会被回收，可以继续使用，现在，你想要买m种商品，每件商品的售价是bi，请问你最少需要花多少钱？

第一行两个整数n，m

第二行n个正整数，代表n种无门槛优惠券的面值(不保证排序)

第三行m个正整数，代表m件商品的价格(不保证排序)

输出合理使用优惠券后购买m件商品的最小价格

示例：

输入：

3 4

50 100 200

99 199 200 300

输出：

248

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

static auto \_ = [](){

ios::sync\_with\_stdio(false);

cin.tie(NULL);

return nullptr;

}();

int main(){

int n, m, res = 0;

bool flag = false;

cin >> n;

cin >> m;

vector<int> cards(n ,0);

vector<int> products(m, 0);

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

cin >> cards[i];

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

cin >> products[i];

if(!is\_sorted(cards.begin(), cards.end()))

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

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

{

int l = 0, r = n-1, mid = -1;

while(l <= r)

{

mid = (l + r) >> 1;

if(mid-1 >= 0 && cards[mid] > products[i] && cards[mid-1] <= products[i]){

res += products[i]-cards[mid-1];

flag = true;

break;

}

if(cards[mid] > products[i])

r = mid-1;

else if(cards[mid] == products[i]){

flag = true;

break;

}

else

l = mid+1;

}

if(flag)

flag = false;

else{

res += cards[0] > products[i] ? products[i] : products[i] - cards[n-1];

}

}

cout << res << endl;

return 0;

}

思路 排序优惠券并进行二分查找

3.数列变换

<https://blog.csdn.net/wuprogrammer/article/details/105481885>

#include <iostream>

#include <vector>

using namespace std;

static auto \_ = [](){

ios::sync\_with\_stdio(false);

cin.tie(NULL);

return nullptr;

}();

int main(){

int t, m;

cin >> t;

bool flag = false, hasNum = false;

vector<int> nums1, nums2;

vector<string> res;

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

{

cin >> m;

nums1.resize(m);

nums2.resize(m);

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

cin >> nums1[j];

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

cin >> nums2[j];

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

{

if(nums1[j] != nums2[j]){

if(hasNum){

flag = true;

break;

}

if(nums1[j] > nums2[j]){

hasNum = true;

flag = true;

break;

}

else{

hasNum = true;

int tmp = nums2[j] - nums1[j];

for(int k = j+1; k < m; ++k)

{

if(nums2[k] - nums1[k] != tmp){

if(nums2[k] != nums1[k])

flag = true;

else

j = k;

break;

}

if(k == m-1)

j = m-1;

}

}

if(flag)

break;

}

}

if(flag || (!flag && !hasNum)){

flag = false;

res.push\_back("NO");

}

else

res.push\_back("YES");

hasNum = false;

}

for(auto str : res)

cout << str << endl;

return 0;

}

4.拆木棍

在你面前从左到右摆着n根长短不一的木棍，你每次可以折断一个木棍，并将折断后得到的两根木棍一左一右放在原来的位置，折断后两根木棍的长度必须为整数，且他们之和为折断前的木棍的长度，你希望最终从左到右的木棍长度单调不减，那么你需要折断多少次呢？

第一行是个数n，表示开始时有多少根木棍(1<=n<=3000)，第二行有n个数，从第一个到第n个分别代表木棍的长度，任意木棍长度l的范围为(1<=l<=3000)

输出一行一个数，你最少所需折断的木棍数

#include <iostream>

#include <vector>

#include <stack>

using namespace std;

int main(){

int n, res = 0;

cin >> n;

vector<int> body(n, 0);

stack<int> st;

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

cin >> body[i];

for(int i = n-1; i >= 0; --i)

{

if(!st.empty() && st.top() < body[i]){

int t = (body[i] - 1) / st.top();

res += t;

st.push(body[i] / (t+1));

}

else

st.push(body[i]);

}

cout << res << endl;

return 0;

}

<https://leetcode-cn.com/circle/discuss/eXOcnD/>

机器人正在玩一个古老的基于DOS的游戏。游戏中有N+1座建筑——从0到N编号，从左到右排列。编号为0的建筑高度为0个单位，编号为i的建筑的高度为H(i)个单位。

起初， 机器人在编号为0的建筑处。每一步，它跳到下一个（右边）建筑。假设机器人在第k个建筑，且它现在的能量值是E, 下一步它将跳到第个k+1建筑。它将会得到或者失去正比于与H(k+1)与E之差的能量。如果 H(k+1) > E 那么机器人就失去 H(k+1) - E 的能量值，否则它将得到 E - H(k+1) 的能量值。

游戏目标是到达第个N建筑，在这个过程中，能量值不能为负数个单位。现在的问题是机器人以多少能量值开始游戏，才可以保证成功完成游戏？

#include <iostream>

#include <vector>

using namespace std;

int main(){

    int n;

    cin >> n;

    vector<int> body(n, 0);

    for(int i = 0; i < n; ++i)

        cin >> body[i];

    int res = 0;

    for(int i = n-1; i >= 0; --i)

    {

        res = (res + body[i]) % 2 == 0 ? (res+body[i]) / 2 : (res+body[i])/2 + 1;

    }

    cout << res << endl;

    return 0;

}

算法笔试题

<https://blog.csdn.net/my_sunshine26/article/details/104886835>

第一题

抖音上不同的用户类型我们有不同的用户模型文件。

我们有一个模型配置文件，里面有很多的不同的用户类型和模型文件。我们需要找出每个模型对应的是哪些用户类型。

给定一行输入，格式是：a b

a表示这个用户的用户类型，b表示这个用户对应的模型文件。

请你输出每个模型文件对应的用户类型。

**输入描述**

输入第1行:用户类型N

接下来的N行:用户类型 模型文件

**输出描述**

每行输出：用户类型 模型文件1 模型文件2 。。。

注意:每个模型文件可能对应多个用户类型，用户类型之间用空格 切分。

注意2:如果有多个用户类型输出，用户类型之间的排序按照字典序。

#include <bits/stdc++.h>

using namespace std;

static auto \_ = [](){

ios::sync\_with\_stdio(false);

return nullptr;

}();

int main(){

int n, count = 1;

cin >> n;

vector<set<string>> files;

unordered\_map<string, int> body;

vector<string> fileName;

string a = "" ,b = "";

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

cin >> a;

cin >> b;

if(body[b] == 0){

body[b] = count++;

fileName.push\_back(b);

set<string> tmp;

tmp.insert(a);

files.push\_back(tmp);

}

else

files[body[b]-1].insert(a);

}

for(int i = 0; i < count-1; ++i){

cout << fileName[i];

for(auto it = files[i].begin(); it != files[i].end(); ++it)

cout << " " << \*it;

if(i != count-2)

cout << endl;

}

return 0;

}

第二题

旅行者穿越沙漠的过程中需要不断地消耗携带的饮用水，到达终点前会经过几个绿洲，每个绿洲均设有水分补给站可以为旅行者提供水分补给并收取一定的费用。

沿途共有n个补给站，每个补给站收取的费用都一样，但是提供的水量不尽相同。起点到终点的距离为D公里，postion[i]表示 第i个补给站距离起点的距离，单位为公里，supply[]表示 第i个补给站可以提供的水量，单位为升。

假设旅行者在起点时携带了W升的水，每行走1公里需要消耗1升的水量，身上可携带的水量没有上限，且携带的水量多少不会对体能消耗产生影响，鉴于每次进补给站花费的钱都是一样多，期望用最少的补给次数到达终点，请帮忙计算最少的补给次数。

**输入描述**

第一行输入整数D和W, D表示起点到终点的距离，W表示初始携带的水 第二行输入数组postion，长度为N，分别表示N个补给站分别距离起点的距离 第三行输入数组supply,长度为N,分别表示N个补给站分别可以供给的水量 数据范围: 1 <=D, W<= 10^8，0<= N <=1000，0 < position[i],supply[i] < D

**输出描述**

输出一个整数表示最少的补给次数，若无法到达终点则返回-1

#include <bits/stdc++.h>

using namespace std;

static auto \_ = [](){

ios::sync\_with\_stdio(false);

return nullptr;

}();

int main(){

int D, W, N = 0, res = 0;

cin >> D;

cin >> W;

vector<int> distance(2000, 0), supply(2000, 0);

priority\_queue<int> helper;

while(cin >> distance[N++]);

N = (N - 1) / 2;

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

supply[i] = distance[i+N];

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

if(distance[i] > W){

while(W - distance[i] < 0){

if(!helper.empty()){

W += helper.top();

helper.pop();

res++;

}

else{

cout << -1 << endl;

return 0;

}

}

}

helper.push(supply[i]);

}

while(W - D < 0){

if(!helper.empty()){

W += helper.top();

helper.pop();

res++;

}

else{

cout << -1 << endl;

return 0;

}

}

cout << res << endl;

return 0;

}

第三题

**输入描述**

每一行输入都是用空格隔开的整数

第一行给出迷宫地图的长和宽，均为正整数

之后每一行的每一个数字，都代表迷宫的一格

-2表示起点，-3表示终点，-1表示不可通过的障碍物，0表示可通过的道路，大于0的正整数代表传送门，并且保证成对出现，在传送门上，可以仅用一步传送到另一一个相同数字的传送门]的位置。

**输出描述**

输出最少要多少步能够从起点走到终点。

输出-1如果没有任何办法从起点走到终点。

**测试用例**

**input**

4 3

1 0 -1 1

-2 0 -1 -3

2 2 0 0

**output**

3

#include <bits/stdc++.h>

using namespace std;

static auto \_ = [](){

ios::sync\_with\_stdio(false);

return nullptr;

}();

pair<int, int> tarPos;

unordered\_map<int, vector<int>> trans;

int n, m, res = INT\_MAX;

void dfs(int r, int c, vector<vector<bool> >& isCome, vector<vector<int> >& body, int tmp, vector<vector<bool> >& isTrans){

if(r == tarPos.first && c == tarPos.second){

res = min(res, tmp);

//cout << res << endl;

return;

}

if(body[r][c] > 0 && !isTrans[r][c])

{

int rTmp = trans[body[r][c]][0], cTmp = trans[body[r][c]][1];

isTrans[r][c] = true;

if(r == rTmp && c == cTmp){

if(!isTrans[trans[body[r][c]][2]][trans[body[r][c]][3]])

dfs(trans[body[r][c]][2], trans[body[r][c]][3], isCome, body, tmp+1, isTrans);

}

else{

if(!isTrans[rTmp][cTmp])

dfs(rTmp, cTmp, isCome, body, tmp+1, isTrans);

}

isTrans[r][c] = false;

}

if(r-1 >= 0 && body[r-1][c] != -1 && !isCome[r-1][c]){

isCome[r-1][c] = true;

dfs(r-1, c, isCome, body, tmp+1, isTrans);

isCome[r-1][c] = false;

}

if(r+1 < n && body[r+1][c] != -1 && !isCome[r+1][c]){

isCome[r+1][c] = true;

dfs(r+1, c, isCome, body, tmp+1, isTrans);

isCome[r+1][c] = false;

}

if(c+1 < m && body[r][c+1] != -1 && !isCome[r][c+1]){

isCome[r][c+1] = true;

dfs(r, c+1, isCome, body, tmp+1, isTrans);

isCome[r][c+1] = false;

}

if(c-1 >= 0 && body[r][c-1] != -1 && !isCome[r][c-1]){

isCome[r][c-1] = true;

dfs(r, c-1, isCome, body, tmp+1, isTrans);

isCome[r][c-1] = false;

}

}

int main(){

cin >> m;

cin >> n;

vector<vector<int> > body(n, vector<int>(m, 0));

vector<vector<bool> > isCome(n, vector<bool>(m, false));

vector<vector<bool> > isTrans(n, vector<bool>(m, false));

pair<int, int> begPos;

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

{

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

{

cin >> body[i][j];

if(body[i][j] == -2){

begPos.first = i;

begPos.second = j;

}

else if(body[i][j] == -3)

{

tarPos.first = i;

tarPos.second = j;

}

else if(body[i][j] > 0)

{

trans[body[i][j]].push\_back(i);

trans[body[i][j]].push\_back(j);

}

}

}

isCome[begPos.first][begPos.second] = true;

dfs(begPos.first, begPos.second, isCome, body, 0, isTrans);

if(res != INT\_MAX)

cout << res << endl;

else

cout << -1 << endl;

return 0;

}

第四题

**题目描述**

连连看是一款经典益智游戏。玩家需要把两个相同的方块用最多三根线段相连，线段必须水平或垂直，不能伸出棋盘，也不能被别的方块阻挡;这样，这两个方块就会被消除，并且玩家会获得得分。

现在有一个连连看的棋盘，和玩家每次操作的方块选择记录，你需要判断每次操作的两个方块是否能被消除。

2020/5/2面试

第一题

**输入描述**

第一行两个数: n, m,代表连连看的棋盘大小为n\*m。

接下来n行，每行一个长度为m的字符串，代表棋盘上方块的位置。输入只包含”.”和大写字母。”.”代表此处没有方块，大写字母代表方块，不同的字母代表不同的方块。

接下来一行一个数: q,玩家操作次数。

接下来q行每行四个数: x1, y1, x2, y2,代表这次操作的两个方块的坐标。若能消除，则会清除对应方块。

数据保证每次操作的两个方块坐标一定不同。

**输出描述**

输出q行，对于每个操作，若能消除，输出'YES'， 否则输出'NO'。

#include <bits/stdc++.h>

using namespace std;

static auto \_ = [](){

ios::sync\_with\_stdio(false);

return nullptr;

}();

int main(){

int n,m,N,x1,y1,x2,y2;

cin>>n>>m;

string \* map=new string[n];

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

{

cin >> map[i];

}

cin >> N;

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

{

cin >> x1 >> y1 >> x2 >> y2;

x1-=1;

x2-=1;

y1-=1;

y2-=1;

if (map[x1][y1]!=map[x2][y2])

{

cout<<"NO"<<endl;

continue;

}

int temp;

if (x1>x2){

temp=x1;x1=x2;x2=temp;

temp=y1;y1=y2;y2=temp;

}

int j=x1,k=x2;

for(; j < x2; j++)

if(map[j][y1]!='.') break;

for(; k > x1; k--)

if(map[k][y2]!='.') break;

if(k <= j)

{

int maxy = y1 > y2 ? y1 : y2;

int miny = y1 > y2 ? y2 : y1;

int flag = 0;

for(; k <= j; k++){

int d = miny + 1;

for(; d <= maxy - 1; d++)

{

if(map[k][d]!='.')

{

flag = 1;

break;

}

}

if(d != maxy - 1)

{

flag=0;

}

}

if(!flag)

{

map[x1][y1] = '.';

map[x2][y2] = '.';

cout << "YES" << endl;

continue;

}

}

if(y1 > y2){

temp = x1;

x1 = x2;

x2 = temp;

temp = y1;

y1 = y2;

y2 = temp;

}

j = y1;

k = y2;

for(; j < y2; j++)

if(map[x1][j] != '.') break;

for (; k > y1 ; k--)

if(map[x2][k] != '.') break;

if (k <= j)

{

int maxx = x1 > x2 ? x1 : x2;

int minx = x1 > x2 ? x2 : x1;

int flag=0;

for(; k <= j; k++){

int d = minx + 1;

for(; d <= maxx - 1; d++)

{

if(map[d][k]!='.')

{

flag=1;

break;

}

}

if(d != maxx - 1)

{

flag=0;

}

}

if(!flag)

{

map[x1][y1]='.';

map[x2][y2]='.';

cout<<"YES"<<endl;

continue;

}

}

cout<<"NO"<<endl;

}

}

第五题

升序数组中第一个都是1，后续都是连续的素数，对每一个m < n都对应一个有理数m/n,现在给定这个数组和一个k，要求返回第k小的有理数

输入格式：

第一行是一个正整数N，从第二行开始每行有若干个正整数，依次存放K，a1…an，一共有n行，也就是n组参数。

K是输入参数表示需要寻找的顺序第K小的有理数，a1-an是以1开始后续的n-1个素数

输出：

输出有n行，每行两个数字m和n，空格隔开

#include <bits/stdc++.h>

using namespace std;

int main(){

int N, tmp, k, count = 0;

cin >> N;

vector<double> body;

unordered\_map<double, vector<int>> A;

priority\_queue<double> helper;

int n = home.size() / N;

char c;

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

{

cin >> tmp;

k = tmp;

while((cin >> tmp).get(c)){

body.push\_back(tmp);

if(c == '\n')

break;

}

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

{

for(int jj = j+1; jj < body.size(); ++jj){

double temp = body[j] / body[jj];

A[temp].push\_back(body[j]);

A[temp].push\_back(body[jj]);

if(count < k){

count++;

helper.push(temp);

}

else{

helper.push(temp);

helper.pop();

}

}

}

count = 0;

cout << A[helper.top()][0] << " " << A[helper.top()][1] << endl;

while(!helper.empty())

helper.pop();

body.resize(0);

A.clear();

}

return 0;

}

第六题

安卓手机屏幕密码采用下边9个点中画线至少包括M至多N个点形成一个密码

1 2 3

4 5 6

7 8 9

任意两点都可以连线并符合如下限制

1. 每个密码路径中每个点只能用一次
2. 如果二号点没有被使用过，直接链接1和3是非法的，类似还有4-6，7-9，1-7，3-9，1-9，2-8等
3. 如果2号点用过了，则1-3是可行的，1和6可以直接连接，因为中间没有其他点，求给定m,n限定条件下的合法密码数(M是最少点数，N是最多点数)

输入：

4

1 1 2 3 4

3 1 2 3

2 1 2 3 5

1 1 2 3 5

输出：

1 4

2 3

1 3

1 5

#include <bits/stdc++.h>

using namespace std;

vector<int> body;

int main(){

int N, tmp;

char c;

cin >> N;

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

while((cin >> tmp).get(c)){

body.push\_back(tmp);

if(c == '\n')

break;

}

/\* 针对这行的操作代码 \*/

body.clear();

}

return 0;

}

将n转为m进制

#include <bits/stdc++.h>

using namespace std;

int main(){

int n,m;

cin >> n >> m;

int bit[1000];

int k=0;

while (n)

{

if(n%(m)>=0)

{

bit[k]=n%(m);

n=n/(m);

}

else

{

bit[k]=n-(n/(m)+1)\*(m);

n=n/m+1;

}

k++;

}

for(int j=k-1;j>=0;j--)

{

cout<<bit[j];

}

cout<<endl;

return 0;

}

第一题：给定一个整数数组，数组中有正负数随机分布，要求将正数负数间隔分布，且不改变其原来的顺序。例如【1， 2， 3， -2， -4， -6， 7， 8】 => 【1，-2， 2， -4， 3， -6， 7， 8】

#include <bits/stdc++.h>

using namespace std;

int main(){

int tmp, pos1 = 0, pos2 = 0;

vector<int> nums, posNums, negNums;

while(cin >> tmp){

nums.push\_back(tmp);

}

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

{

if(nums[i] < 0)

negNums.push\_back(nums[i]);

else if(nums[i] > 0)

posNums.push\_back(nums[i]);

}

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

if(i % 2 == 0 ){

if(pos1 < posNums.size())

nums[i] = posNums[pos1++];

else

nums[i] = negNums[pos2++];

}

else{

if(pos2 < negNums.size())

nums[i] = negNums[pos2++];

else

nums[i] = posNums[pos1++];

}

}

string res = "";

res += "[";

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

if(i != nums.size()-1)

res += to\_string(nums[i]) + " ";

else

res += to\_string(nums[i]) + "]";

}

cout << res << endl;

return 0;

}

第二题：翻转二叉树，给定一个二叉树按层输出的结果，要求将此二叉树翻转并输出其按层输出的结果。例如【1， 2， 3， #， 7， 8， 9】（‘#’表示空位） => 【1， 3， 2， 9， 8， 7， #】

#include <bits/stdc++.h>

using namespace std;

struct TreeNode{

char val;

TreeNode \*left, \*right;

TreeNode(int value) : val(value), left(nullptr), right(nullptr){}

};

void createTree(TreeNode \*p, int i, vector<char> &tree){

if(tree[i] != '#' && i < tree.size()){

p = new TreeNode(tree[i]);

createTree(p->left, 2\*i+1, tree);

createTree(p->right, 2\*i+2, tree);

}

}

void dfs(TreeNode \*p){

if(!p)

return;

dfs(p->left);

dfs(p->right);

if(p->left && p->right){

TreeNode \*q = p->left;

p->left = p->right;

p->right = q;

}

else if(p->left && !p->right){

p->right = p->left;

p->left = nullptr;

}

else if(!p->left && p->right){

p->left = p->right;

p->right = nullptr;

}

}

int main(){

char c;

string res = "[";

vector<char> tree;

TreeNode \*root = nullptr;

while(cin >> c){

tree.push\_back(c);

}

createTree(root, 0, tree);

dfs(root);

queue<TreeNode\*> body;

int tmp, count = 1;

body.push(root);

res += to\_string(root->val) + " ";

/\*while(!body.empty())

{

tmp = count;

count = 0;

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

root = body.front();

body.pop();

if(root->left){

body.push(root->left);

count++;

res += to\_string(root->left->val) + " ";

}

else

res += "# ";

if(root->right){

body.push(root->right);

count++;

res += to\_string(root->right->val) + " ";

}

else

res += "# ";

}

}

res[res.size()-1] = ']';\*/

cout << res << endl;

return 0;

}

2020/5/8字节跳动笔试题

第一题：

初始时候手里放着一叠扑克牌（从上向下垂直摆放），顺序未知，通过以下步骤形成一个新的序列（也是从上向下垂直摆放）。

1. 把手里牌堆得最上边一张放到桌子上边牌堆的最上边（第一张直接放在桌面上）
2. 把手里牌堆的最上边一张放到手中牌堆的最下边
3. 反复重复上边两个操作，直到手中牌全都放到了桌子上，形成了新的一叠牌

这时我们能看到桌面上的牌的从上到下的顺序，要推算出原始手中牌的顺序。

例如，我们看到桌子上最终的牌堆从上到下的顺序是：4 2 3 1

那原始在手中时候的从上到下的顺序就是1 2 3 4

输入描述：

第一行数字N，表示当前这组测试用例有多少个测试用例，后续会有N行的用空格分隔的数字，表示在桌子上的牌堆从上向下的顺序。

输出描述：

每行是用空格分隔的数字，表示对应的输入用例的输出，也就是原始在手上时候堆从上向下的顺序

第二题：

题目描述

给定一个无向无环图，求图的直径。

这里直径是指最大的两个节点间的边的数量。

第一行是N，表示测试数组的数量，每组的各入如下

每组的第一行是M，表示节点的个数，后续是M-1行的正整数n1，n2表示n1和n2之间有一条边。

输出描述：

每行一个输出，表示对应测试图数据的直径

#include <bits/stdc++.h>

using namespace std;

int ans = 0;

void dfs(int start, int index, vector<vector<int>> &v, vector<int> &fa, vector<int> &jl)

{

int s = v[start].size();

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

{

if(v[start][i] != fa[start])

{

fa[v[start][i]] = start;

dfs(v[start][i], index + 1, v, fa, jl);

fa[v[start][i]] = 0;

}

}

if(index > ans)

ans = index;

jl[start] = index;

return;

}

int main(){

int N, M, l, r;

cin >> N;

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

{

vector<vector<int>> v;

vector<int> fa;

vector<int> jl;

cin >> M;

v.resize(M+1);

fa.resize(M+1);

jl.resize(M+1);

for(int j = 1; j < M; ++j)

{

cin >> l;

cin >> r;

v[l].push\_back(r);

v[r].push\_back(l);

}

dfs(1, 0, v, fa, jl);

int end;

for(int j = 1; j <= M; j++)

{

if(jl[j] == ans)

{

end = j;

break;

}

}

dfs(end, 0, v, fa, jl);

cout << ans << endl;

ans = 0;

}

system("pause");

return 0;

}

第三题：

求图的直径(可能有环)

#include <bits/stdc++.h>

using namespace std;

static auto \_ = [](){

ios::sync\_with\_stdio(false);

cin.tie(NULL);

return nullptr;

}();

int main(){

int N, M, lNode, rNode, ans = 0;//N:图的个数 M:每个图的结点数

cin >> N;

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

{

cin >> M;//M:图中结点的个数

vector<vector<int> > dis(M+1, vector<int>(M+1, 0));

for(int j = 0; j < M; ++j)

{

cin >> lNode;

cin >> rNode;

dis[lNode][rNode] = 1;

dis[rNode][lNode] = 1;

}

//M+1:图中边的个数

for(int j = 1; j < M+1; ++j)

for(int k = 1; k < M+1; ++k){

if(dis[j][k] != 1 && j != k)

dis[j][k] = 10000;

}

for(int j = 1; j < M+1; ++j)

for(int k = 1; k < M+1; ++k)

for(int t = 1; t < M+1; ++t)

if(dis[k][t] > dis[k][j]+dis[j][t])

dis[k][t] = dis[k][j]+dis[j][t];

for(int j = 1; j < M+1; ++j)

for(int k = 1; k < M+1; ++k)

if(dis[j][k] != 10000)

ans = max(ans, dis[j][k]);

cout << ans << endl;

// for(int j = 1; j < M+1; ++j){

// for(int k = 1; k < M+1; ++k)

// cout << dis[j][k] << " ";

// cout << endl;

// }

ans = 0;

dis.clear();

}

system("pause");

return 0;

}

#include <bits/stdc++.h>

using namespace std;

static auto \_ = [](){

ios::sync\_with\_stdio(false);

cin.tie(NULL);

return nullptr;

}();

int main(){

int N, tmp;

char c;

vector<int> body;

cin >> N;

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

{

while((cin >> tmp).get(c)){

body.push\_back(tmp);

if(c == '\n')

break;

}

for(auto num : body)

cout << num << ' ';

cout << endl;

body.clear();

}

system("pause");

return 0;

}

知识点总结：

1.fork()样例：

int main(){

fork();

fork()&&fork()||fork();

fork();

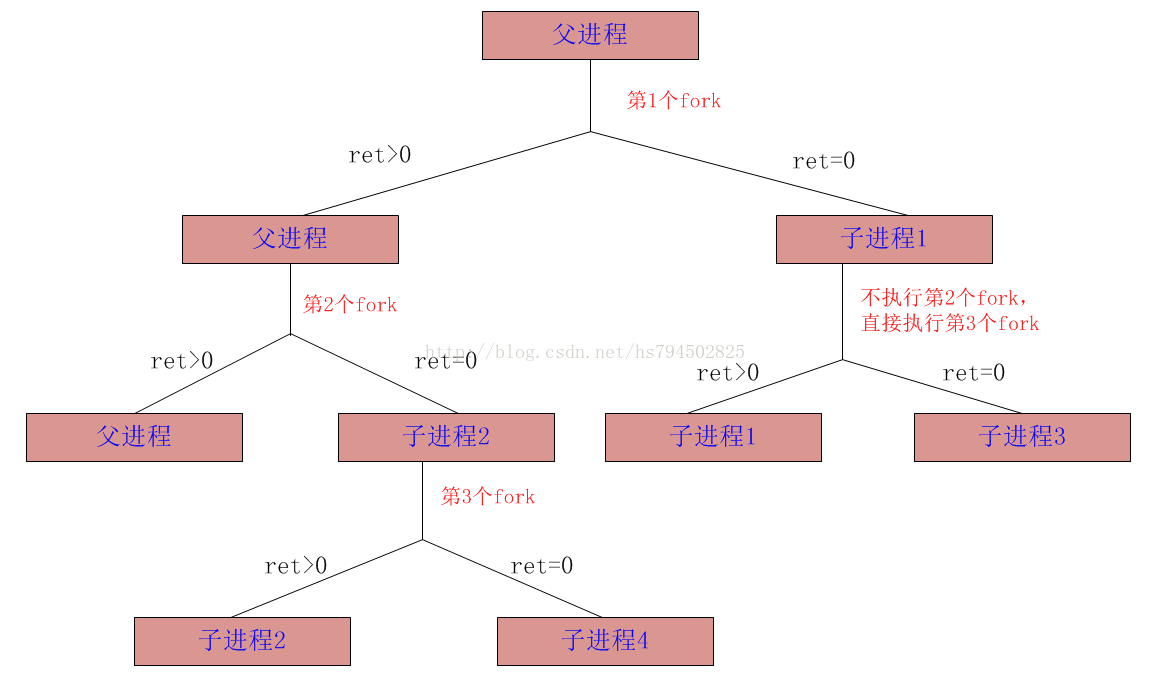
return 0;

}

求一共生成几个进程

第一行产生一个子进程，目前共2个进程

第二行每个进程又生成了4个新进程，目前共10个进程如图：



第三行每个进程又产生了一个子进程，共20个进程。

总共增加了19个新进程。

2.蓄水池算法

3.udp和tcp的区别

2020.5.15面试经历

1.自我介绍(…)

2.询问实习时间(…)

3. c++如何实现多态:(说了静态多态和动态多态,函数重载，函数模板编程，虚函数动态绑定机制)

4.linux IO机制，了解IO复用吗，select和epoll区别(简要说了5中IO复用模型)

5.进程线程通信机制(线程通过锁，全局变量，进程通过管道，IPC，socket)

6.你了解操作系统哪些线程锁(读写锁，自旋锁，互斥锁)

7.项目介绍(说了自己的项目)

8.网络编程基本过程(说了自己项目里TCP同步通信模型，服务器:socket->bind->listen->accept->数据收发->close客户端:socket->connect->数据收发->close)

8.git 常用命令 git add, git status, git commit, git push, git reset(git reset 答得不好)

9.从10亿数据取一部分，如何去重，建立表是在内存中还是文件中(开始说用map，后来说在文件中建立hash表，面试官说效率低，总之这部分答得不太好)

10.从10亿数据取前k个用什么结构，大顶堆(构建大顶堆，说了时间复杂度)

11.数据库了解哪些概念:(答了acid特性，InnoDB，myISAM引擎及其区别，数据库索引结构B+树，为什么用B+树不用二叉搜索树)

12.数据库的四种隔离机制(读未提交，读已提交，可重复读，串行化)

13.TCP/IP是哪层协议，具体介绍一下内部机制，TCP为什么是三次握手四次挥手(基本都说上了)

14. 自己有哪些爱好,平时玩游戏吗（说了下游戏开发相关的,引擎开发，UE4）

15.19年以来游戏界热词(说了光追,DLSS的概念)

16.说一下自己的优缺点(…)

腾讯/360笔试题

作者：normal\_yuan  
链接：[https://www.nowcoder.com/discuss/409909](https://www.nowcoder.com/discuss/409909" \t "_blank)  
来源：牛客网

**1.最强的不一定是最后的赢家。**  
某赛事有n名选手参加，但是不同于其他的比赛，本比赛采取的是擂台赛的形式，n名选手排成一排，每次队伍的第一位和第二位选手进行比赛，输的一方会排到队尾。  
当某位选手取得m连胜时，他将成为最后的赢家，且游戏结束，请问截止到游戏结束，共会进行多少次比赛。  
两位选手的比赛结果由他们的战斗力决定，n位选手的战斗力是一个1~n的排列，也就是说他们的战斗力两两不同，不会有平局的情况。

**//输入第一行包含两个正整数n，m，分别代表参赛选手数量和取得连胜的要求。(1<=n<=100000，1<=m<=10^9)**

**//输入第二行包含n个正整数，中间用空格隔开，第i个数表示队伍的第i位选手的战斗力，整体是一个1~n的排列。**

#include <bits/stdc++.h>

using namespace std;

static auto \_ = [](){

ios::sync\_with\_stdio(false);

cin.tie(NULL);

return nullptr;

}();

int main(){

int n, m, res = 0, tmp, l1, l2, winner, currVal= 0;

cin >> n;

cin >> m;

queue<int> body;

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

cin >> tmp;

body.push(tmp);

}

l1 = body.front();

body.pop();

l2 = body.front();

body.pop();

winner = max(l1, l2);

while(currVal != m){

if(l1 > l2){

body.push(l2);

if(winner == l1)

currVal++;

else{

currVal = 1;

winner = l1;

}

}

else{

body.push(l1);

if(winner == l2)

currVal++;

else{

currVal = 1;

winner = l2;

}

}

l1 = winner;

l2 = body.front();

body.pop();

res++;

}

cout << res << endl;

return 0;

}

每天1小时C++primer，半小时STL源码解析(0:00-1:30)

每天2-4道编程题可以串时间做(13:30-14:00, 20:00-23:00)(leetcode 2道，牛客网2道)

每天1小时Unix网络编程(下午14:00-15:00)

重点：每天2小时陈硕书 + Github源码(15:00-17:00)

每天半小时牛客面经(20:00-23:00)