@Bellman\_Ford最短路

struct Edge {

int u, v, w;

};

vector<Edge> edge;

int dis[MAXN], u, v, w;

const int INF = 0x3f3f3f3f;

bool bellmanford(int n, int s) {

memset(dis, 0x3f, sizeof(dis));

dis[s] = 0;

bool flag = false; // 判断一轮循环过程中是否发生松弛操作

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

flag = false;

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

u = edge[j].u, v = edge[j].v, w = edge[j].w;

if (dis[u] == INF) continue;

// 无穷大与常数加减仍然为无穷大

// 因此最短路长度为 INF 的点引出的边不可能发生松弛操作

if (dis[v] > dis[u] + w) {

dis[v] = dis[u] + w;

flag = true;

}

}

// 没有可以松弛的边时就停止算法

if (!flag) {

break;

}

}

// 第 n 轮循环仍然可以松弛时说明 s 点可以抵达一个负环

return flag;

}

bool spfa(int s)

{

memset(cnt, 0, sizeof cnt);

fill(dis, dis + n + 1, inf);

dis[s] = 0, vis[s] = 1;

queue<int> q;

q.push(s);

while (!q.empty())

{

int u = q.front();

q.pop(), vis[u] = 0;

for (auto [v, w] : g[u])

{

if (dis[v] - w > dis[u])

{

dis[v] = dis[u] + w;

cnt[v] = cnt[u] + 1;

if (cnt[v] >= n)

return false;

if (!vis[v])

q.push(v), vis[v] = 1;

}

}

}

return true;

}最短哈密顿回路

#include<bits/stdc++.h>

using namespace std;

const int MAXN=25,MAXM=(1<<20),inf=0x3f;//定义变量，inf为无限

int n,a[MAXN][MAXN],f[MAXM][MAXN];

int main(){

scanf("%d",&n);

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

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

scanf("%d",&a[i][j]);

memset(f,inf,sizeof(f));//一开始f数组都是无限的

f[1][0]=0;//还没开始旅程，为0

for(int i=1;i<(1<<n);i++)//枚举状态

{

for(int j=0;j<n;j++)//枚举每个点

{

if(!((i>>j)&1)) continue;//经过了

for(int k=0;k<n;k++)//上一次经过了哪些点？

if(((i^(1<<j))>>k)&1)//枚举从上一个经过的节点走到j节点

f[i][j]=min(f[i][j],f[i^(1<<j)][k]+a[k][j]);//状态转移

}

}

printf("%d\n",f[(1<<n)-1][n-1]);//out

return 0;

}

奶牛聚会

#include <iostream>

using namespace std;

const int N = 1005;

int mp[N][N], n, m, k, res;

int ans[N], vis[N], cow[N];

void dfs(int u) {

ans[u]++;

vis[u] = 1;

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

if (mp[u][i] &&!vis[i]) {

dfs(i);

}

}

}

int main() {

cin >> k >> n >> m;

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

cin >> cow[i];

}

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

int ver, from;

cin >> from >> ver;

mp[from][ver] = 1;

}

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

fill(vis, vis + N, 0);

vis[cow[i]] = 1;

dfs(cow[i]);

}

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

if (ans[i] == k) {

res++;

}

}

cout << res << endl;

return 0;

}

线段树

#include <bits/stdc++.h>

#define let auto &

#define var auto

#define in :

#define to0(a) memset(a, 0, sizeof(a))

#define to1(a) memset(a, -1, sizeof(a))

#define N 400005

using namespace std;

using ll = long long;

using cint = const int;

cint inf = INT\_MAX;

#define int ll

int t[N];

int a[N];

int lazy[N];

void build(int l, int r, int p)

{

if (l == r)

{

t[p] = a[l];

return;

}

int mid = l + ((r - l) >> 1);

int lc = (p << 1), rc = lc | 1;

build(l, mid, lc);

build(mid + 1, r, rc);

t[p] = t[lc] + t[rc];

}

void add(int l, int r, int i, int j, int p, int c)

{

if (l <= i && j <= r)

{

t[p] += (j - i + 1) \* c, lazy[p] += c;

return;

}

int mid = i + ((j - i) >> 1); // mid=(i+j)/2

int lchild = (p << 1), rchild = lchild | 1;

if (lazy[p] && i != j)

{

t[lchild] += lazy[p] \* (mid - i + 1);

t[rchild] += lazy[p] \* (j - mid);

lazy[lchild] += lazy[p], lazy[rchild] += lazy[p];

lazy[p] = 0;

}

if (l <= mid)

add(l, r, i, mid, lchild, c);

if (mid < r)

add(l, r, mid + 1, j, rchild, c);

t[p] = t[lchild] + t[rchild];

}

int getsum(int l, int r, int i, int j, int p)

{

if (l <= i && j <= r)

return t[p];

int mid = i + ((j - i) >> 1);

int lchild = (p << 1), rchild = lchild | 1;

if (lazy[p])

{

t[lchild] += lazy[p] \* (mid - i + 1);

t[rchild] += lazy[p] \* (j - mid);

lazy[lchild] += lazy[p], lazy[rchild] += lazy[p];

}

lazy[p] = 0;

int sum = 0;

if (l <= mid)

sum = getsum(l, r, i, mid, lchild);

if (mid < r)

sum += getsum(l, r, mid + 1, j, rchild);

// cout<<sum<<endl;

return sum;

}

signed main()

{

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

cin.tie(nullptr), cout.tie(nullptr);

int n, m, x, y, cmd, k;

cin >> n >> m;

to0(lazy);

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

cin >> a[i];

build(1, n, 1);

// output();

// for(int i=1;i<=9;i++)

// cout<<t[i]<<endl;

while (m--)

{

cin >> cmd;

if (cmd == 1)

{

cin >> x >> y >> k;

add(x, y, 1, n, 1, k);

}

else

{

cin >> x >> y;

cout << getsum(x, y, 1, n, 1) << endl;

}

}

return 0;

}

食物链计数

int n;

queue<int> q;

vector<vector<int>> g;

vector<int> rudu, chudu, dp;

void input()

{

int m;

cin >> n >> m;

g.resize(n + 1), rudu.resize(n + 1), dp.resize(n + 1);

int a, b;

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

{

cin >> a >> b;

g[a].push\_back(b);

++rudu[b];

}

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

{

if (!rudu[i])

{

dp[i] = 1;

q.push(i);

}

if (!g[i].size())

chudu.push\_back(i);

}

}

int bfs()

{

int res = 0, a;

while (!q.empty())

{

a = q.front();

q.pop();

for (auto b : g[a])

{

--rudu[b];

dp[b] = (dp[b] + dp[a]) % mod;

if (!rudu[b])

{

q.push(b);

}

}

}

for (auto a : chudu)

{

res = (res + dp[a]) % mod;

}

return res;

}

**平时实验代码**

前缀和试炼之子矩阵的和

ll arr[N][N];

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

cin>>n>>m>>q;

to0(arr[0]);

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

arr[i][0] = 0;

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

cin >> arr[i][j];

arr[i][j]+=(arr[i][j-1]+arr[i-1][j]-arr[i-1][j-1]);

}}

while(q--){

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

cout<<arr[x2][y2]-arr[x2][y1-1]-arr[x1-1][y2]+arr[x1-1][y1-1]<<endl;

}

纸片游戏

int dfs(int m)

{

int a, dfsf, dfsb, p = m % 2;

if (m == 1)

{

// cout<<"res:"<<s<<endl<<endl;

return s;

}

// arr.print();

a = arr.front();

arr.pop\_front();

if (p == 0)

s += a;

// cout<<arr.front()<<' ';

// cout<<s<<endl;

dfsf = dfs(m - 1);

if (p == 0)

s -= a;

arr.push\_front();

a = arr.back();

arr.pop\_back();

if (p == 0)

s += a;

dfsb = dfs(m - 1);

if (p == 0)

s -= a;

arr.push\_back();

if (p == 0)

{

return max(dfsf, dfsb);

}

else

{

return min(dfsf, dfsb);

}

}

n的阶乘右边有多少个连续的零。

int max2(int n)

{

int i = 0;

while ((1 << i) <= n)

++i;

return i - 1;

}

int max5(int n)

{

int i = 0, a = 1;

while (a <= n)

++i, a \*= 5;

// cout<<a<<endl;

return a / 5;

}

signed main()

{

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

cin.tie(nullptr), cout.tie(nullptr);

int n;

cin >> n;

int five = 0, twoMax = max2(n), two = 0, fiveMax = max5(n);

for (int i = twoMax; i >= 1; --i)

{

two += n / (1 << i);

}

for (int i = fiveMax; i >= 5; i /= 5)

{

five += n / i;

}

cout << min(two, five) << endl;

return 0;

}

2出现的次数

int num2(int n)

{

int res = 0;

int ten = 1;

int next = 1;

while (n)

{

res += ((n / 10 + (n % 10 > 2)) \* ten + (n % 10 == 2) \* next); // 本位为2

next = n % 10+1;

n /= 10;

ten \*= 10;

}

return res;

}

cin >> l >> r;

cout << num2(r) - num2(l - 1) << endl;

字符串反转

a = input().split()

for i in range(len(a)):

a[i] = a[i][::-1]

print(' '.join(a))

矩阵幂求和

int n, k, m, n2;

struct matrix

{

int arr[N][N];

matrix()

{

for (int i = n2 - 1; i >= 0; --i)

{

to0(arr[i]);

}

}

};

matrix operator\*(const matrix &a, const matrix &b)

{

matrix res;

for (int i = 0; i < n2; i++)

{

for (int j = 0; j < n2; j++)

{

for (int p = 0; p < n2; p++)

{

res.arr[i][j] = (res.arr[i][j] + a.arr[i][p] \* b.arr[p][j]) % m;

}

}

}

return res;

}

void init(matrix &mat)

{

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

{

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

{

cin >> mat.arr[i][j];

}

}

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

{

mat.arr[i][i + n] = mat.arr[i + n][i + n] = 1;

}

}

matrix pow(matrix a)

{

matrix res;

for (int i = 0; i < n2; i++)

{

res.arr[i][i] = 1;

}

while (k)

{

if (k & 1)

{

res = res \* a;

}

a = a \* a;

k >>= 1;

}

return res;

}

void sub(matrix &mat)

{

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

{

mat.arr[i][i + n] = mat.arr[i][i + n] + m - 1;

for (int j = n; j < n2; j++)

{

mat.arr[i][j] %= m;

}

}

}

void output(matrix mat)

{

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

{

for (int j = n; j < n2; j++)

{

cout << mat.arr[i][j] << ' ';

}

cout <<endl;

}

}

signed main()

{

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

cin >> n >> k >> m;

++k;

n2 = 2 \* n;

matrix res;

init(res);

res = pow(res);

sub(res);

output(res);

return 0;

}

分馅饼

void solve()

{

cin >> n >> f;

++f;

double l = 0.0, r = 0.0;

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

{

cin >> pie[i];

pie[i] \*= pie[i];

r += pie[i];

}

sort(pie, pie + n);

r /= f;

double m = (l + r) / 2, eat;

while (r - l > 1e-6)

{

// cout << l << ' ' << r << endl;

m = (l + r) / 2;

eat = 0;

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

{

if (pie[i] < m)

{

break;

}

eat += floor(pie[i] / m);

if (eat > f)

{

break;

}

}

if (eat >= f) // 可以分给更多的朋友

{

l = m;

}

else

{

r = m;

}

// m = (l + r) / 2;

}

// cout<<m<<endl;

cout << fixed << setprecision(4) << m \* pi << endl;

}

密码破译

int n, res;

string s;

void dfs(int l, int r, char c, int counter){

if (r - l == 1)

{

if (s[l] != c)

++counter;

res = min(res, counter);

return;

}

if (counter >= res)

{

return;

}

int m = (l + r) >> 1;

int change = 0;

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

{

if (s[i] != c)

{

++change;

}

}

// 4 5 6 7

dfs(m, r, c + 1, counter + change);

change = 0;

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

{

if (s[i] != c)

{

++change;

}

}

dfs(l, m, c + 1, counter + change);

}

void solve(){

cin >> n >> s;

res = n;

dfs(0, n, 'a', 0);

cout << res << endl;

}

拼凑的旋律：用第二行的数拼第一行

int n, m;

int a[N], b[N];

array<vector<int>, N> a2b; // a里的元素由哪些b构成

int db[N]; // 十进制转二进制，如0->1<<0=1b,1->1<<1=10b,2->1<<2=100b,...,7->1<<7=10000000b

// int bd[8]; // 二进制转十进制，同上

int res = 0;

void alla2b(int goala, int last, int start, int con)

/\*

找到所有a能由b组成的组合

goala:要找的a[i]

last:还需要多少凑成a[i]

start:上一个找到的j的下一个，从这儿开始循环

con:使用情况，二进制数表示

\*/

{

if (last == 0)

{

a2b[goala].push\_back(con);

// cout << "ok" << endl;

return;

}

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

{

if (last < b[j])

return;

con += db[j];

alla2b(goala, last - b[j], j + 1, con);

con -= db[j];

}

}

void dfs(int start, int counter, int con)

/\*

start:上一个找到的i的下一个，从这儿开始循环

counter:个数

\*/

{

// cout<<counter<<endl;

if (counter == n)

{

res = n;

return;

}

for (int i = start; i < n; i++) // 循环找a由哪些b组成

{

for (var each in a2b[i]) // 循环找这样每一种情况

{

// cout << each << endl;

// cout << (con | each) << endl;

// 如已经使用了1001，而找到了1010，那么新情况就是1001|1010=1011

// 此时1001&1010=1000，那么说明有重叠，不行

if (con & each) // 有重叠

{

// cout << "no" << endl;

}

else

{

// cout << "ok" << endl;

dfs(i + 1, counter + 1, con | each); // 新情况

}

}

}

res = max(res, counter);

}

signed main()

{

cin >> n >> m;

for (int i = 0, j = 1; i < 8; ++i, j <<= 1)

{

db[i] = j;

}

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

{

cin >> a[i];

}

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

{

cin >> b[i];

}

sort(a, a + n), sort(b, b + m);

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

{

alla2b(i, a[i], 0, 0);

}

dfs(0, 0, 0);

cout << res << endl;

// output\_a2b();

}

运算优先级

string s;

int lens;

vector<vector<vector<int>>> f; // f[i][j][k]:延长长度，从第几开始，有哪些结果

vector<int> nums; // 数字

vector<char> signs; // 运算符

int n = 0; // 运算符的个数,数字个数比运算符个数多一个

void readstring(int num, int i)

{

if (i == lens)

{

nums.push\_back(num);

return;

}

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

{

nums.push\_back(num);

signs.push\_back(s[i]);

++n;

readstring(0, i + 1);

}

else

{

readstring(num \* 10 + s[i] - '0', i + 1);

}

}

signed main()

{

cin >> s;

lens = s.size();

readstring(0, 0);

f.resize(n + 2, vector<vector<int>>(n + 1));

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

{

f[0][i].push\_back(nums[i]);

// debug << "nums[i]" << nums[i] << endl;

}

for (int l = 1; l <= n; l++) // 长度-1

{

for (int l0 = 0; l0 < l; ++l0) // 如l=2,l0=0->1。

{

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

// 把[l,i]分为[l0,i]?[l-l0-1,i+l0+1]。2\*3-4\*5

// 例如：[2,1]=[0,1]?[1,2]==>l=2,i=1,l0=0,?=signes[i+l0]=signs[1]

//

{

if (signs[i + l0] == '+')

{

for (var a in f[l0][i])

{

for (var b in f[l - l0 - 1][i + l0 + 1])

{

f[l][i].push\_back(a + b);

}}}

else if (signs[i + l0] == '-')

{

for (var a in f[l0][i])

{

for (var b in f[l - l0 - 1][i + l0 + 1]) {

f[l][i].push\_back(a - b);

}

}

}

else if (signs[i + l0] == '\*') {

for (var a in f[l0][i]) {

for (var b in f[l - l0 - 1][i + l0 + 1]) {

f[l][i].push\_back(a \* b);

}}}}}}

// 最后只剩[n,0]

std::sort(f[n][0].begin(), f[n][0].end());

for (int i = 0, l = f[n][0].size(); i < l; i++)

{

cout << f[n][0][i] << endl;

}

return 0;

}

八皇后第n解

void dfs(int depth)

{

if (depth == n)

{

string str="00000000";

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

{

str[i]=arr[i]+1+'0';

}

result.push\_back(str);

return;

}

for (int i = 0; i < n; i++) // put column(i,depth)

{

int j = n;

for (j = 0; j < depth; j++) // check (arr[j],j)

{

if (i == arr[j] || abs(i - arr[j]) == abs(j - depth))

{

// cout<<i<<','<<depth<<'\t'<<arr[j]<<','<<j<<endl;

break;

}

}

if (j == depth)

{

arr[depth] = i;

dfs(depth + 1);

}

}

}

第k小的数

void QuickSelect(int s[], int k, int left, int right)

{

int i, j, pivot;

if (left <= right)

{

pivot = s[left];

i = left;

j = right;

while (i < j)

{

while (i < j && s[j] > pivot)

{

j--;

};

if (i < j)

{

s[i] = s[j];

i++;

}

while (i < j && s[i] < pivot)

{

i++;

};

if (i < j)

{

s[j] = s[i];

j--;

}

}

s[i] = pivot;

}

if (k < (i + 1))

{

QuickSelect(s, k, left, i - 1);

}

else if (k == (i + 1))

{

return;

}

else

{

QuickSelect(s, k, i + 1, right);

}

}

石子游戏：拿走非零平方数的博弈：

void init()

{

to0(square);

to0(win);

for (int i = 1, j = 1; j = i \* i, j <= N; ++i)

{

win[j] = square[j] = true;

// cout<<j<<endl;

nums.push\_back(j);

}

}

bool dfs(int n)

{

// cout<<n<<':'<<endl;

if (square[n])

return true;

for (var res in nums)

{

// cout<<res<<endl;

if (res >= n)

return false;

if (win[n - res] == false)

{

// cout<<res<<'+'<<n-res<<endl;

return true;

}

}

return false;

}

01背包

int w[n], v[n], f[m + 1];

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

{

cin >> w[i] >> v[i];

}

to0(f);

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

{

for (int j = m; j >= w[i]; --j)

{

f[j] = max(f[j], f[j - w[i]] + v[i]);

}

}

cout << f[m] << endl;

return 0;}

最长公共子序列

int c[n+1][m+1];

for (int i = 0; i <= n; i++) c[i][0] = 0;

for (int j = 0; j <= m; j++) c[0][j] = 0;

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

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

if (x[i - 1] == y[j - 1])

{

c[i][j] = c[i - 1][j - 1] + 1;

}

else

{

c[i][j] = max(c[i - 1][j], c[i][j - 1]);

}}}

cout << c[n][m];

调制四果汤：花尽量少的钱来购买配料，调出来的必须具有四果汤的所有特征。

init();//十进制转二进制d2b初始化

string s;

int goal = (1 << t) - 1;

int dp[goal + 5], value[n], various[n];

to1(dp);

int m;

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

{

cin >> m >> s;

int fruits = 0;

for (int i = s.size() - 1; i >= 0; --i)

{

fruits |= d2b[s[i] - 'A'];

}

value[i] = m;

various[i] = fruits;

dp[fruits] = m;

}

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

{

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

// for (int j = goal; j >= 1; --j)

{

if (dp[j] == inf)

continue;

int k = various[i] | j;

if (dp[k] == -1 || dp[k] > value[i] + dp[j])

dp[k] = value[i] + dp[j];

}

}

cout << dp[goal] << endl;

上课别摸鱼：九键键盘

int c[10] = {0, 0, 3, 3, 3, 3, 3, 4, 3, 4};

cin >> s;

int l = s.size();

ll f[l];

f[0] = 1;

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

{

f[i] = f[i - 1];

for (int j = 1; j < c[s[i] - '0']; j++)

{

if (s[i] != s[i - j])

break;

if (i > j)

f[i] += f[i - j-1];

else

{

f[i] += 1;

break;

}

}

f[i] %= mod;

// cout<<f[i]<<endl;

}

cout << f[l - 1] << endl;

贪心：选择其中任意个数（不能多次选择同一个位置的数），对a[i]做a[i] := a[i] \* i的操作，使得变换后的序列的乘积能够整除2^n，并且要使得选中的数的个数最少。

inline int howmuch2(int a)

{

int res = 0;

while (a)

{

if (a & 1)

return res;

a >>= 1;

++res;

}

return res;

}

int solve()

{

int origin = 0, res = 0;

int n, m;

int arr[35];

to0(arr);

cin >> n;

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

{

cin >> m;

origin += howmuch2(m);

++arr[howmuch2(i)];

}

// cout<<origin<<endl;

if (origin >= n)

{

return 0;

}

for (int i = 34; i >= 1; i--)

{

int t = origin + arr[i] \* i;

if (t >= n)

{

return res + (n - origin + i - 1) / i;

}

origin = t;

res += arr[i];

}

return -1;

}

共享单车

int bike()

{

vector<int> dp(n + 1, INT\_MAX);

dp[0] = 0;

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

{

// 像我一样，不买通票，直接付

dp[i] = dp[i - 1] + price \* d[i];

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

{

for (int k = min(t[j], i); k >= 1; --k)

dp[i] = min(dp[i], dp[i - k] + c[j]);

}

}

return dp[n];

}

神奇的拼图

int f[N][N];

bool check(int c)

{

memset(f, 0, sizeof(f));

int x = 0, y;

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

{

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

{

for (int k = j; k <= min(m, j + c / a[i]); k++)

{

f[i][k] = max(f[i][k], f[i - 1][j] + (c + a[i] \* (j - k)) / b[i]);

y = k;

}

}

x = y;

}

return f[n][m] >= m;

}

int search()

{

int l = 1, r = 1e7, mid;

while (l <= r)

{

// cout<<l<<' '<<r<<endl;

mid = (l + r) / 2;

if (check(mid))

{

r = mid - 1;

}

else

{

l = mid+1;

}

// cout << l << ' ' << r << endl;

}

return l;

}

装配线调度

int n;

cin >> n;

int t[n + 1][2][2];

for (int i = 0; i < 2; i++)

{

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

{

cin >> t[j][i][i];

}

}

for (int i = 0; i < 2; i++)

{

int k = 1 - i;

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

{

cin >> t[j][i][k];

}

}

int leave[2], f[n + 1][2];

cin >> f[1][0] >> f[1][1] >> leave[0] >> leave[1];

for (int j = 0; j < 2; j++)

f[1][j] += t[1][j][j];

for (int i = 2; i <= n; i++)

{

for (int j = 0; j < 2; j++)

{

f[i][j] = t[i][j][j]+min(f[i-1][j], f[i - 1][1-j] + t[i-1][1-j][j]);

}

}

int res = min(f[n][0] + leave[0], f[n][1] + leave[1]);

cout << res << endl;

矩阵链乘法

int n;

cin >> n;

int dim[n + 1];

vector<vector<int>> m(n + 1, vector<int>(n + 1, inf));

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

{

cin >> dim[i - 1] >> dim[i];

m[i][i] = 0;

}

for (int c = 2; c <= n; c++)

{

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

{

int j = i + c - 1;

for (int k = i; k < j; k++)

{

m[i][j] = min(m[i][j], m[i][k] + m[k + 1][j] + dim[i - 1] \* dim[k] \* dim[j]);

}

// cout<<m[i][j]<<endl;

}

}

cout << m[1][n] << endl;

并查集

struct dsu

{

vector<int> pa, size;

explicit dsu(int s) : pa(s)

{

iota(pa.begin(), pa.end(), 0);

size.resize(s);

}

int find(int x)

{

return pa[x] == x ? x : find(pa[x]);

}

void unite(int x, int y)

{

x = find(x), y = find(y);

if (x == y)

return;

if (size[x] < size[y])

swap(x, y);

pa[y] = x;

size[x] += size[y];

}

};

堆排序：前m小的数

vector<int> a;

int n, m;

void heapsort(int l, int r)

{

int parent = l;

int child = (parent << 1) | 1;

while (child <= r)

{

if (child + 1 <= r && a[child] > a[child + 1])

++child;

if (a[parent] < a[child])

return;

else

{

swap(a[parent], a[child]);

parent = child;

child = (parent << 1) | 1;

}

}

}

void heapify()

{

for (int i = ((n - 2) >> 1); i >= 0; --i)

heapsort(i, n - 1);

}

signed main()

{

cin >> n >> m;

a.resize(n);

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

cin >> a[i];

heapify();

bool first = false;

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

{

swap(a[0], a[n - i]);

heapsort(0, n - i - 1);

cout << a[n - i];

if (first)

first = true;

else

cout << ' ';

}

cout<<endl;

return 0;

}

Led：贪心

if (n <= 2)

res = n / 2;

else

switch (n % 3){

case 0:

res = n / 3 \* 7;

break;

case 1:

res = (n / 3 - 1) \* 7 + 4;

break;

case 2:

res = (n / 3) \* 7 +1;

break;

default:

break;

}

相同数组和

void solve()

{

int n, q, l, r;

cin >> n >> q;

vector<int> arr(n + 1), numof1(n + 1);

vector<long long> sum(n + 1); // 不开long long\_ \_ \_

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

{

cin >> arr[i];

sum[i] = sum[i - 1] + arr[i];

numof1[i] = numof1[i - 1] + (arr[i] == 1);

}

while (q--)

{

cin >> l >> r;

if (l == r || r - l + 1 + numof1[r] - numof1[l-1] > sum[r] - sum[l-1])

cout << "NO" << endl;

else

cout << "YES" << endl;

}}

小a的字符串：先保证少，然后靠前删

const int a = 'a', z = 'z';

int arr[27], counter[27];

void solve()

{

string s;

int n, summ = 0;

to0(arr), to0(counter);

cin >> s >> n;

for (char c : s)

{

++arr[int(c) - a];

summ += (c - a + 1);

}

if (summ > n)

for (int i = 25; i >=0; i--)

{

int b = (summ - n + i) / (i + 1);

if (b > arr[i])

{

counter[i] = arr[i];

summ -= arr[i] \* (i + 1);

}

else

{

counter[i] = b;

break;

}

}

for (auto c : s)

{

if (counter[c - a])

--counter[c - a];

else

cout << c;

}

cout << endl;

}

最大高度

int n, m, s;

vector<int> height(N, \_inf);

int cmp\_inf;

struct edge{ int v, w;};

vector<vector<edge>> e;

void dfs(int u){

for (edge i : e[u]){

int v = i.v, h = i.w;

int minn = min(h, height[u]);

if (minn > height[v]){

height[v] = minn;

dfs(v);

}}}

void bfs(int u)

{

queue<int> q;

q.push(u);

while (!q.empty())

{

u = q.front();

q.pop();

for (edge i : e[u])

{

int v = i.v, h = i.w;

int minn = min(h, height[u]);

if (minn > height[v])

{

height[v] = minn;

q.push(v);

}

}

}

}

signed main()

{

cin >> n >> m >> s;

e.resize(n + 1);

int a, b, c;

while (m--)

{

cin >> a >> b >> c;

e[a].push\_back({b, c});

e[b].push\_back({a, c});

}

height[s] = inf;

bfs(s);

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

{

if (i == s)

continue;

if (height[i] == \_inf)

cout << "unreachable" << endl;

else

cout << height[i] << endl;

}

return 0;

}

Dijkstra：第K所学生公寓可能有火情，通知N个消防队派遣分队，从驻扎公寓沿着最近的路线往第K公寓。

现在已知任意2所学生公寓之间的通行时间，计算第一个消防分队到达第K公寓的通行时间。

int n, m, p, q;

struct edge

{

int v, w;

};

edge makeedge(int v, int w)

{

edge e;

e.v = v, e.w = w;

return e;

}

struct node{

int dis, u;

bool operator>(const node &a) const

{

return dis > a.dis;

}

};

void solve()

{

cin >> n >> m >> p >> q;

priority\_queue<node, vector<node>, greater<node>> qq;

vector<int> zhuzha;

vector<vector<edge>> e(m + 1);

int a, b, c;

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

{

cin >> a;

zhuzha.push\_back(a);

}

while (p--)

{

cin >> a >> b >> c;

e[a].push\_back({b, c});

e[b].push\_back({a, c});

}

vector<int> dis(m + 1, inf);

vector<bool> vis(m + 1);

dis[q] = 0;

qq.push({0, q});

while (!qq.empty())

{

int u = qq.top().u;

qq.pop();

if (vis[u])

continue;

vis[u] = true;

for (auto ed : e[u])

{

int v = ed.v, w = ed.w;

if (dis[v] - dis[u] > w)

{

dis[v] = dis[u] + w;

qq.push({dis[v], v});

}

}

}

int res = inf;

for (auto x : zhuzha)

{

res = min(dis[x], res);

}

cout << res << endl;

}

铺设光纤：最小生成树

int n, a[N][N];

struct node

{

int u, v, len;

bool operator>(const node &a) const

{

return len > a.len;

}

};

node make\_node(int a, int b, int c)

{

node nd;

nd.u = a, nd.v = b, nd.len = c;

return nd;

}

priority\_queue<node, vector<node>, greater<node>> q;

struct dsu

{

vector<int> pa, size;

explicit dsu(int s) : pa(s)

{

iota(pa.begin(), pa.end(), 0);

size.resize(s);

}

int find(int x)

{

return pa[x] == x ? x : find(pa[x]);

}

void unite(int x, int y)

{

// pa[find(x)]=find(y);

x = find(x), y = find(y);

if (x == y)

return;

if (size[x] < size[y])

swap(x, y);

pa[y] = x;

size[x] += size[y];

}

};

signed main()

{

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

cin.tie(nullptr), cout.tie(nullptr);

cin >> n;

dsu dd(n + 1);

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

{

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

{

cin >> a[i][j];

if (i != j) 暴雨: n栋建筑，m条道路。1号点出发，最初的生命值为H，当他沿着一条道路从建筑ai到bi时，中途冲刺的过程会被暴摧残掉ci的生命值，当他的生命值小于0时，他将无法回到n号点，而每经过一栋建筑i时，他都会在该建筑获取fi个奖励。

有多条路径，每条路有一个fi最大的建筑物，找到这些所有路径的里面的最小值。

cint inf = 0x3fffffff;

struct Edge{

int u, v, d;

Edge() {}

Edge(int a, int b, int c) : u(a), v(b), d(c) {}

};

vector<Edge> g[N];

int dis[N], f[N];

int n, m, h, l = 0, r = 0;

struct node

{

int dis, pos;

bool operator>(const node &a) const

{

return dis > a.dis;

}

};

inline bool check(int lim)

{

if (f[1] > lim)

return false;

fill(dis, dis + n + 1, inf);

dis[1] = 0;

priority\_queue<node, vector<node>, greater<node>> q;

q.push({0, 1});

while (!q.empty())

{

node top = q.top();

q.pop();

int u = top.pos;

if (dis[u] < top.dis)

continue;

for (auto edg : g[u])

{

int v = edg.v, d = edg.d;

if (f[v] <= lim && dis[v] - d > dis[u])

{

dis[v] = dis[u] + d;

q.push({dis[v], v});

}

}

}

return dis[n] != dis[0] && dis[n] <= h;

}

void input()

{

cin >> n >> m >> h;

int u, v, d;

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

{

cin >> f[i];

r = max(r, f[i]);

}

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

{

cin >> u >> v >> d;

g[u].push\_back(Edge(u, v, d));

g[v].push\_back(Edge(v, u, d));

}

}

signed main()

{

input();

if (!check(r))

{

cout << "AFK" << endl;

return 0;

}

while (r - l > 1)

{

int mid = (l + r) >> 1;

if (check(mid))

r = mid;

else

l = mid;

}

cout << r << endl;

return 0;

}

{

q.push(make\_node(i, j, a[i][j]));

}

}

}

int res = 0;

while (!q.empty())

{

node nd = q.top();

q.pop();

if (dd.find(nd.u) != dd.find(nd.v))

{

res += nd.len;

dd.unite(nd.u, nd.v);

}

}

cout<<res<<endl;

return 0;

}

奖励4090

int n, m, f[M], w, v, s;

to0(f);

cin >> n >> m;

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

{

cin >> w >> v >> s;

for (int j = max(w \* s, m); j >= w; --j)

{

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

{

int wsum = w \* k;

if (wsum > j)

break;

f[j] = max(f[j], f[j - wsum] + k \* v);

}

}

}

cout << f[m] << endl;

宁宁与古榕树

struct node

{

int childsum, childedge, res;

vector<int> edge;

node() { childsum = childedge = 0; }

};

int n;

bool vis[N];

vector<node> tree;

void input()

{

memset(vis, 0, sizeof vis);

cin >> n;

tree.resize(n + 1);

int u, v;

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

{

cin >> u >> v;

tree[u].edge.push\_back(v);

tree[v].edge.push\_back(u);

}

}

void dfs(int u)

{

for (auto v : tree[u].edge)

{

if (!vis[v])

{

vis[v] = true;

dfs(v);

tree[u].childedge += (tree[v].childedge + 1);

tree[u].childsum += tree[v].childsum;

}

}

tree[u].childsum += tree[u].childedge;

}

void dfs2(int u, int parentsum, int parentedge)

{

parentsum += parentedge;

tree[u].res = tree[u].childsum + parentsum;

for (auto v : tree[u].edge)

{

if (vis[v])

{

vis[v] = false;

dfs2(v, parentsum - tree[v].childedge - tree[v].childsum + tree[u].childsum - 1, parentedge - tree[v].childedge + tree[u].childedge);

}

}

}

signed main()

{

input();

vis[1] = true;

dfs(1);

vis[1] = false;

dfs2(1, 0, 0);

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

{

cout << tree[i].res << endl;

}}

差分约束

struct edge

{

int v, w, tail;

edge() {}

edge(int a, int b, int c) : v(a), w(b), tail(c) {}

};

int n, m, cnt;

edge e[N];

bool vis[N];

int dis[N], in[N], head[N];

bool spfa(int u)

{

in[u] = 1;

vis[u] = true;

dis[u] = 0;

queue<int> q;

q.push(u);

while (!q.empty())

{

u = q.front();

q.pop();

vis[u] = false;

for (int j = head[u]; j; j = e[j].tail)

{

int v = e[j].v, w = e[j].w;

if (dis[v] > dis[u] + w)

{

dis[v] = dis[u] + w;

if (!vis[v])

{

q.push(v);

vis[v] = true;

++in[v];

if (in[v] == n + 1)

return false;

}

}

}

}

return true;

}

void add(int u, int v, int w)

{

e[cnt] = edge(v, w, head[u]);

head[u] = cnt++;

}

void input()

{

// memset(head, -1, sizeof head);

cnt = 1;

cin >> n >> m;

int u, v, w;

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

dis[i] = INT\_MAX;

memset(vis, 0, sizeof vis);

memset(in, 0, sizeof in);

while (m--)

{

cin >> u >> v >> w;

add(v, u, w);

}

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

add(n + 1, i, 0);

}

signed main()

{

input();

if (!spfa(n + 1))

return 0;

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

cout << dis[i] << ' ';

// cout << dis[n] << endl;

return 0;

}

二分图最大匹配

int n, m, k;

vector<int> may\_be\_cp[N];

int lover[N], loved[N];

int cp = 0;

inline void input()

{

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

may\_be\_cp[i].clear();

memset(lover, 0, sizeof lover);

memset(loved, 0, sizeof loved);

int u, v;

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

{

cin >> u >> v;

may\_be\_cp[u].push\_back(v);

}

}

bool find\_lover(int i, int ntr)

{

if (lover[i] == ntr)

return false;

lover[i] = ntr;

for (int j : may\_be\_cp[i])

{

if (!loved[j] || find\_lover(loved[j], ntr))

{

loved[j] = i;

return true;

}

}

return false;

}

signed main()

{

while (cin >> k >> n >> m)

{

cp = 0;

input();

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

{

if (find\_lover(i, i))

++cp;

}

cout << cp << endl;

}

return 0;

}

口罩运输

int n, m, s, t;

ll dis[N], res;

bool vis[N];

int pre[N], head[N], flag[300][300];

struct node

{

int to, net;

ll val;

node();

node(int a, int b, ll c) : to(a), net(b), val(c){};

};

vector<node> e;

inline void add(int u, int v, ll w)

{

e.push\_back(node(v, head[u], w));

head[u] = e.size() - 1;

e.push\_back(node(u, head[v], 0));

head[v] = e.size() - 1;

}

inline bool bfs()

{

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

vis[i] = false;

queue<int> q;

q.push(s);

vis[s] = 1;

dis[s] = 3000000000;

while (!q.empty())

{

int x = q.front();

q.pop();

for (int i = head[x]; i; i = e[i].net)

{

if (e[i].val <= 0)

continue;

int v = e[i].to;

if (vis[v])

continue;

dis[v] = min(dis[x], e[i].val);

pre[v] = i;

q.push(v);

vis[v] = true;

if (v == t)

return true;

}

}

return false;

}

inline void update()

{

int x = t;

while (x != s)

{

int v = pre[x];

e[v].val -= dis[t];

e[v ^ 1].val += dis[t];

x = e[v ^ 1].to;

}

res += dis[t];

}

signed main()

{

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

cin.tie(nullptr), cout.tie(nullptr);

cin >> m >> n;

s = 1, t = n;

for (int i = 0; i < 2; i++)

e.push\_back(node(0, 0, 0));

int u, v;

ll w;

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

{

cin >> u >> v >> w;

if (flag[u][v] == 0)

{

add(u, v, w);

flag[u][v] = e.size() - 1;

}

else

{

e[flag[u][v] - 1].val += w;

}

}

while (bfs())

{

update();

}

cout << res << endl;

return 0;

}

八皇后

int n;

int arr[N];

int res=0;

void dfs(int depth)

{

if (depth == n)

{

++res;

return;

}

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

{

int j = n;

for (j = 0; j < depth; j++)

{

if (i == arr[j] || abs(i - arr[j]) == abs(j - depth))

{

break;

}

}

if (j == depth)

{

arr[depth] = i;

dfs(depth + 1);

}

}

}

算24

bool cmp(float x, float y, float &z, int i)

{

switch (i)

{

case 0:

z = x + y;

return true;

break;

case 1:

z = x - y;

return true;

break;

case 2:

z = y - x;

return true;

break;

case 3:

z = x \* y;

return true;

break;

case 4:

if (iszero(y))

return false;

z = x / y;

return true;

break;

case 5:

if (iszero(x))

return false;

z = y / x;

return true;

default:

return false;

break;

}

}

bool dfs(int d)

{

if (d == 1)

{

// if (b[0] == 24)

// return true;

return iszero(b[0] - 24);

}

float last[4];

for (int k = 0; k < d; k++)

last[k] = a[k];

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

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

{

float x = a[i], y = a[j];

int l = 0;

for (int k = 0; k < d; k++)

{

if (k != i && k != j)

{

b[l++] = a[k];

}

}

for (int k = 0; k < l; k++)

{

a[k] = b[k];

}

float z = 0;

for (int k = 0; k < 6; k++)

{

if (!cmp(x, y, z, k))

continue;

b[l] = a[l] = z;

if (dfs(d - 1))

return true;

}

for (int k = 0; k < d; k++)

a[k] = b[k] = last[k];

}

return false;

}

多重背包

int n, m;

cin >> n >> m;

int s[n], v[n], w[n], dp[m + 1];

memset(dp, 0, sizeof dp);

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

{

cin >> w[i] >> v[i] >> s[i];

if (s[i] \* w[i] > m) // 质量大按完全背包随便装

{

for (int j = w[i]; j <= m; ++j)

{

dp[j] = max(dp[j], dp[j - w[i]] + v[i]);

}

}

else // 质量小按01背包逐个装

{

for (int j = m; j >= w[i]; --j)

{

for (int k = s[i]; k >= 0; k--)

{

if (j >= k \* w[i])

dp[j] = max(dp[j], dp[j - k \* w[i]] + k \* v[i]);

}

}

}

}

cout << dp[m] << endl;

合唱队形

signed main()

{

int n;

int arr[N], up[N], down[N], res = 0;

cin >> n;

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

cin >> arr[i];

fill(up, up + n + 1, 1);

fill(down, down + n + 1, 1);

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

{

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

{

if (arr[j] < arr[i])

{

up[i] = max(up[i], up[j] + 1);

}

}

}

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

{

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

{

if (arr[i] > arr[j])

{

down[i] = max(down[i], down[j] + 1);

}

}

}

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

{

res = max(res, up[i-1] + down[i]);

}

cout << n-res << endl;

return 0;

}

排队打水

int n, a;

long long res = 0;

cin >> n;

vector<int> arr;

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

{

cin >> a;

arr.push\_back(a);

}

sort(arr.begin(), arr.end(),greater<int>());

for (int i = arr.size() - 1; i >= 0; --i)

{

res += arr[i] \* i;

}

cout<<res<<endl;

Kotlin程序设计（没看清但是过了）

def fun(a: list) -> tuple:

"""

:return: 变量名，开始，结束，步长

"""

s, t, step = 0, 0, 1

if 'downTo' in a: # for (i in 2 downTo -2 step 1)，i=2，1，0，-1，-2

step = -1

t = a[5]

s = int(a[3])

else:

s, t = a[3].split('..')

s = int(s)

if t[-1] == ')':

t = int(t[:-1])

else:

t = int(t)

# t += step

if 'step' in a: # for (i in 0..5 step 2)，i=0，2，4

step = step \* int(a[-1][:-1])

return a[1][1:], s, t, step

def my\_length(s: int, t: int, d: int) -> int:

return (t - s) // d+1

def my\_add(s: int, t: int, d: int) -> int:

n = my\_length(s, t, d)

return int(d / 2 \* n \*\* 2 + (s - d / 2) \* n)

input()

input()

s1 = input().split()

s2 = input().split()

s3 = input().split()

input()

input()

a1 = fun(s1)

a2 = fun(s2)

answer = 0

if a1[0] in s3:

answer = my\_add(a1[1], a1[2], a1[3]) \* my\_length(a2[1], a2[2], a2[3])

elif a2[0] in s3:

answer = my\_add(a2[1], a2[2], a2[3]) \* my\_length(a1[1], a1[2], a1[3])

print(answer)

山水厦大：模板库版本

int n;

cin >> n;

int arr[n][n];

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

{

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

{

cin >> arr[i][j];

}

}

vector<int> p(n);

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

p[i] = i;

int ans = INT\_MAX;

do

{

int sum = 0;

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

{

sum += arr[p[i]][p[i + 1]];

}

ans = min(ans, sum);

} while (next\_permutation(p.begin() + 1, p.end() - 1));

cout << ans;

宁宁与校庆气球

int n, m, empty\_num = 0;

/\*

做上首个球的标记。如果是第一个球，则空盒数+1。否则p+1,弹出一个球

\*/

bool bottomball[N \* 2];

int ball\_which\_box[N \* 2];

int balls[N \* 2], vistopball[N]; // 位置-球（p+1的bottom不为true时，即为下一个球）(p->ball)，第一个球在哪个p(ball->p)

void push\_ball(int p)

{

int q = vistopball[balls[p]];

// cout << "push:" << p << ' ' << balls[p] << ' ' << q << endl;

if (q) // 已经出现过的顶部球，可以凑对

{

if (ball\_which\_box[q] == ball\_which\_box[p]) // 同一个盒子

return;

if (bottomball[p + 1])

++empty\_num;

else

push\_ball(p + 1);

if (bottomball[q + 1])

++empty\_num;

else

push\_ball(q + 1);

}

else

{

vistopball[balls[p]] = p;

}

}

signed main()

{

cin >> n >> m;

memset(vistopball, 0, sizeof vistopball);

memset(bottomball, 0, sizeof bottomball);

int k, p = 1;

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

{

cin >> k;

int first = p;

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

{

cin >> balls[p];

ball\_which\_box[p] = i;

++p;

}

bottomball[p] = true;

push\_ball(first);

}

if (empty\_num == m)

cout << "Yes" << endl;

else

cout << "No" << endl;

return 0;

}

离心优化

int n, m, maxnum = 0, mincost = 0; // summ是状态压缩

int costs[maxN];

int d2b[7][7], arr[7][7];

vector<int> pairs;

int pairs\_size;

void init()

{

memset(costs, -1, sizeof costs);

d2b[1][1] = 1;

for (int i = 1; i < m \* n; i++) // 状态压缩

{

d2b[i / m + 1][i % m + 1] = d2b[(i - 1) / m + 1][(i - 1) % m + 1] << 1;

}

}

void dfs(int s, int vis, int num, int cost)

{

// cout<<s-1<<' '<<vis<<' '<<num<<' '<<cost<<endl;

if (num >= maxnum)

{

if (num > maxnum)

{

maxnum = num;

mincost = cost;

}

else if (cost < mincost)

{

mincost = cost;

}

}

// cout<<s<<endl;

for (int i = s; i < pairs\_size; i++)

{

// cout <<"i:" << i << endl;

if (vis & pairs[i])

continue;

int next\_vis = vis | pairs[i];

if (num && costs[next\_vis] != -1) // 访问过了

continue;

dfs(i + 1, next\_vis, num + 2, cost + costs[pairs[i]]);

}

}

signed main()

{

int guan2;

cin >> n >> m;

init();

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

{

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

{

cin >> arr[i][j];

}

}

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

{

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

{

if (arr[i][j] != -1 && arr[i][j + 1] != -1)

{

guan2 = d2b[i][j] | d2b[i][j + 1];

pairs.push\_back(guan2);

costs[guan2] = arr[i][j] + arr[i][j + 1];

}

}

}

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

{

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

{

if (arr[i][j] != -1 && arr[i + 1][j] != -1)

{

guan2 = d2b[i][j] | d2b[i + 1][j];

pairs.push\_back(guan2);

costs[guan2] = arr[i][j] + arr[i + 1][j];

}

}

}

pairs\_size = pairs.size();

// dp or dfs?

dfs(0, 0, 0, 0);

cout << maxnum << ' ' << mincost;

return 0;

}

离心优化

int n;

long long m = LONG\_LONG\_MIN, maxnum = 0ll, secnum = 0ll;

scanf("%llu", &n);

long long arr[n + 1];

vector<int> edge[n + 1];

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

{

scanf("%llu", &arr[i]);

m = max(m, arr[i]);

}

int u, v;

long long res = m + 2ll;

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

{

scanf("%llu%llu", &u, &v);

edge[u].push\_back(v);

edge[v].push\_back(u);

}

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

{

if (arr[i] == m)

++maxnum;

else if (arr[i] == m - 1ll)

++secnum;

}

long long maxnum\_old = maxnum, secnum\_old = secnum;

for (int r = 1; r <= n; r++)

{

if (arr[r] == m)

--maxnum;

else if (arr[r] == m - 1ll)

--secnum;

for (int v : edge[r])

{

if (arr[v] == m)

--maxnum, ++secnum;

else if (arr[v] == m - 1ll)

--secnum;

}

if (maxnum == 0ll)

{

if (secnum == 0ll)

res = min(res, m);

else

res = min(res, m + 1ll);

}

maxnum = maxnum\_old, secnum = secnum\_old;

}

printf("%llu", res);

多多建设农场

struct edge

{

int u, v, val;

edge(){}

edge(int a, int b, int c) : u(a), v(b), val(c) {}

bool operator>(const edge &a) const

{

return val > a.val;

}

};

int n;

int arr[N][N];

inline void input()

{

cin >> n;

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

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

cin >> arr[i][j];

}

vector<bool> vis;

priority\_queue<edge, vector<edge>, greater<edge>> q;

void add(int u)

{

vis[u] = true;

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

{

if (u == v)

continue;

if (!vis[v])

q.push(edge(u, v, arr[u][v]));

}

}

long long bfs()

{

vis.resize(n);

int u = 0;

long long res = 0;

edge top;

add(u);

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

{

u = 0;

while (vis[u])

{

top = q.top();

q.pop();

u = top.v;

}

add(u);

res += top.val;

}

return res;

}

矩阵中的最长递增路径

int arr[N][N], dp[N][N];

struct node

{

int x, y, val;

node();

node(int a, int b, int c) : x(a), y(b), val(c){};

bool operator>(const node a) const

{

return val > a.val;

}

};

int n, m, dx[] = {0, 0, -1, 1}, dy[] = {1,-1,0,0};

inline bool cmp(int x0, int y0, int x, int y)

{

return x >= 0 && x < n && y >= 0 && y < m && arr[x][y] > arr[x0][y0] && dp[x0][y0] + 1 > dp[x][y];

}

void dfs(int x0, int y0)

{

for (int t = 0; t < 4; t++)

{

int x = x0 + dx[t], y = y0 + dy[t];

if (cmp(x0, y0, x, y))

{

dp[x][y] = dp[x0][y0] + 1;

dfs(x, y);

}

}

}

priority\_queue<node, vector<node>, greater<node>> q;

int main()

{

memset(dp, 0, sizeof dp);

cin >> n >> m;

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

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

{

cin >> arr[i][j];

q.push(node(i, j, arr[i][j]));

}

while (!q.empty())

{

node top = q.top();

q.pop();

int x = top.x, y = top.y;

if (!dp[x][y])

{

dp[x][y] = 1;

dfs(x, y);

}

}

int res = 1;

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

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

{

res = max(res, dp[i][j]);

}

cout << res << endl;

return 0;

}

Bfs试炼之微博转发

int n, m;

struct node

{

int v, dis;

};

vector<int> g[N];

void input()

{

int j, b;

cin >> n >> m;

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

{

cin >> j;

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

{

cin >> b;

g[b].push\_back(i);

}

}

}

int bfs(int s)

{

int res = 0;

queue<node> q;

vector<bool> vis(n + 1);

vis[s]=true;

q.push({s, 0});

while (!q.empty())

{

int u = q.front().v, dis = q.front().dis;

q.pop();

if (dis == m)

break;

for (int v : g[u])

{

if (!vis[v])

{

q.push({v, dis + 1});

++res;

vis[v] = true;

}

}

}

return res;

}

signed main()

{

input();

int q;

cin >> q;

int arr[q];

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

cin >> arr[i];

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

cout << bfs(arr[i]) << endl;

return 0;

}

宁宁与塔罗牌

const int N = 20;

int n;

vector<vector<int>> g;

typedef pair<int, int> p;

p arr[N];

bitset<N> vis;

bool win(bool except) // true宁宁赢，false对方赢

{

bitset<N> temp = vis;

for (int i = temp.\_Find\_first(); i != temp.size(); i = temp.\_Find\_next(i))

{

for (int j : g[i])

{

if (!vis[j])

continue;

vis.reset(i);

vis.reset(j);

if (win(!except) == except) // 对方没辙

{

vis.set(i), vis.set(j);

return except;

}

vis.set(i), vis.set(j);

}

}

return !except;

}

signed main()

{

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

cin.tie(nullptr), cout.tie(nullptr);

cin >> n;

int a, b;

g.resize(75);

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

{

cin >> a >> b;

arr[i] = p(a, b);

}

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

{

vis.set(i);

}

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

{

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

{

if (arr[i].first == arr[j].first || arr[i].second == arr[j].second)

{

g[i].push\_back(j);

g[j].push\_back(i);

break;

}

}

}

if (win(true))

cout << "YES" << endl;

else

cout << "NO" << endl;

return 0;

}

二部图最小

const int white = 0, red = 1, blue = 2;

typedef int color;

int res = 0;

int nums[3];

vector<vector<int>> g;

vector<color> colors;

bool dfs(int u)

{

color nextcolor = 3 - colors[u];

for (auto v : g[u])

{

if (colors[v] == colors[u])

return false;

if (colors[v] == white)

{

++nums[nextcolor];

colors[v] = nextcolor;

if (!dfs(v))

{

return false;

}

}

}

return true;

}

signed main()

{

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

cin.tie(nullptr), cout.tie(nullptr);

int n, m;

cin >> n >> m;

g.resize(n + 1);

colors.resize(n + 1);

fill(colors.begin(), colors.end(), white);

int a, b;

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

{

cin >> a >> b;

g[a].push\_back(b);

g[b].push\_back(a);

}

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

{

if (colors[i] == white)

{

nums[red] = 1;

nums[blue] = 0;

colors[i] = red;

if (!dfs(i))

{

cout << "Impossible" << endl;

return 0;

}

res += min(nums[red], nums[blue]);

}

}

cout << res << endl;

return 0;

}

盒子

int n;

vector<vector<int>> arr(200005, vector<int>(2));

bool solve()

{

sort(arr.begin(), arr.begin() + n);

int pos = 1;

priority\_queue<int, vector<int>, greater<int>> q;

for (int i = 0; i < n || !q.empty();)

{

for (; i < n && arr[i][0] == pos; i++)

{

q.push(arr[i][1]);

}

if (q.empty())

{

pos = arr[i][0];

}

else

{

if (q.top() < pos)

return false;

q.pop();

++pos;

}

}

return true;

}

丰庭晚餐

int solve()

{

dp[1][0] = m;

dp[1][1] = 0;

for (int i = 2; i <= n; i++)

{

dp[i][1] = dp[i - 1][0];

dp[i][0] = ((dp[i - 1][0] + dp[i - 1][1]) % mod) \* (m - 1) % mod;

}

return (dp[n][0] + dp[n][1]) % mod;

}

宁宁与塔罗牌

#include <stdio.h>

#include <string.h>

#define MAX\_NUM 16

int num\_cards; // 卡牌数量

int cards[MAX\_NUM], count\_cards[MAX\_NUM]; // cards数组存储卡牌，count\_cards数组存储每种卡牌的数量

int mycount[(1 << MAX\_NUM)]; // mycount数组存储每个状态下卡牌数量的总和

int vis\_check[(1 << MAX\_NUM)], ok[(1 << MAX\_NUM)]; // vis\_check和ok数组用于状态检查

// 检查函数，判断当前状态S是否符合条件

int check(int S) {

static int vis[(1 << MAX\_NUM)], ok[(1 << MAX\_NUM)]; // 用于记录状态和判断状态是否合法

if (vis[S] || S == 0) return ok[S]; // 如果状态已被访问过或为初始状态，则直接返回判断结果

vis[S] = 1;

int arr[MAX\_NUM], arr\_size = 0; // arr数组存储卡牌种类，arr\_size表示种类数量

memset(count\_cards, 0, sizeof count\_cards); // 初始化每种卡牌数量为0

for (int i = 0; i < num\_cards; i++) {

if (S >> i & 1) {

count\_cards[cards[i]]++; // 统计当前状态下每种卡牌的数量

}

}

for (int i = 0; i < 13; i++) {

if (count\_cards[i]) arr[arr\_size++] = i; // 记录有卡牌的种类

}

if (mycount[S] == 1) {

ok[S] = 1; // 如果卡牌总数为1，则合法

} else if (mycount[S] == 2) {

ok[S] = (arr\_size == 1); // 如果卡牌总数为2且种类为1，则合法

} else if (mycount[S] == 3) {

ok[S] = (arr\_size == 1); // 如果卡牌总数为3且种类为1，则合法

} else if (mycount[S] == 5) {

if (arr\_size == 5) {

ok[S] = 1; // 如果卡牌总数为5且有序，则合法

for (int i = 1; i < 5; i++)

if (arr[i - 1] + 1 != arr[i])

ok[S] = 0; // 如果不是顺子，则非法

} else if (arr\_size == 2) {

ok[S] = (count\_cards[arr[0]] == 3 || count\_cards[arr[1]] == 3); // 如果有2种卡牌，其中一种数量为3，则合法

}

} else if (mycount[S] == 6) {

int n = arr\_size;

if (n == 2 || n == 3) {

ok[S] = 1; // 如果卡牌总数为6，且有2种或3种卡牌，则合法

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

if (arr[i - 1] + 1 != arr[i])

ok[S] = 0; // 如果不是顺子，则非法

}

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

if (count\_cards[arr[i]] != 6 / n)

ok[S] = 0; // 如果数量不合理，则非法

}

}

}

return ok[S]; // 返回当前状态的合法性

}

// 动态规划函数，计算满足条件的状态总数

long long dp(int S) {

if (S == 0) return 1; // 如果为初始状态，则返回1

static long long f[(1 << MAX\_NUM)]; // 用于存储状态总数

static int vis[(1 << MAX\_NUM)]; // 用于记录状态是否访问过

if (vis[S])

return f[S];

vis[S] = 1;

f[S] = 0;

for (int nS = S; nS > 0; nS = ((nS - 1) & S)) {

if ((nS & (S & -S)) == 0) {

continue;

}

if (check(nS)) {

f[S] += dp(S ^ nS); // 递归计算满足条件的状态总数

}

}

return f[S]; // 返回当前状态下满足条件的状态总数

}

int main() {

for (int i = 0; i < 13; i++) {

int count;

scanf("%d", &count);

while (count--) {

cards[num\_cards++] = i; // 输入卡牌数量和种类

}

}

for (int i = 0; i < (1 << num\_cards); i++)

mycount[i] = mycount[i >> 1] + (i & 1); // 计算每种状态下卡牌数量的总和

printf("%lld\n", dp((1 << num\_cards) - 1)); // 输出满足条件的状态总数

return 0;

}