#include<stdlib.h>

#include<stdio.h>

#include<iostream>

#include<iomanip>

#include<cmath>

#include<algorithm>

#include<set>

#include<deque>

#include<functional>

#include<vector>

#include<queue>

#include<map>

#include<string>

#include<cstring>

#include<string.h>

#include<random>

//#include<bits/stdc++.h>

using namespace std;

#define mem(a,b) memset((a),b,sizeof((a)))

#define clr(a) (a).clear()

#define ll long long

#define IOS ios::sync\_with\_stdio(0);cin.tie(0);cout.tie(0)

#ifndef ONLINE\_JUDGE

freopen("in.txt", "r", stdin);

#endif

字符串和数字互相转换函数

char str[];int num;

字符串转换为整数 stoi(str); // sscanf (str, "%d", &num);

字符串转换为浮点数 stod(str); // sscanf (str, "%lf", &num);

字符串转换为长整数 stol(str); // sscanf (str, "%lld", &num);

整数转字符串sprintf(str,”%d”,num);

浮点数转字符串sprintf(str,”%lf”,num);

长整数转字符串sprints(str, "%lld", num);

随机数生成方法

1.随机小数

static uniform\_real\_distribution<double> x(1,10); //括号内为闭区间范围

static default\_random\_engine e;

double w = x(e);

2.随机整数

static uniform\_int\_distribution<int> x(1,10); //括号内为闭区间范围

static default\_random\_engine e;

int w = x(e);

**矩阵快速幂**

struct mat

{

ll s[15][15];

};

mat mul(mat a, mat b)

{

mat c;

mem(c.s, 0);

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

for (int j = 0; j <= n + 1; ++j)

for (int k = 0; k <= n + 1; ++k)

c.s[i][j] = (c.s[i][j] + a.s[i][k] \* b.s[k][j]) % mod;

return c;

}

mat pow(mat a, int p)

{

mat b;

mem(b.s, 0);

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

b.s[i][i] = 1;

while (p)

{

if (p & 1)

b = mul(b, a);

a = mul(a, a);

p >>= 1;

}

return b;

}

**BFS**

const int dx[] = { 1,-1,0,0 };

const int dy[] = { 0,0,1,-1 };

char maze[100][100];

bool visited[100][100];

int n, m;

int main()

{

while (cin >> n >> m)

{

int flag=0;

memset(visited, false, sizeof(visited));

deque<tuple<int, int, int>>queue;

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

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

{

cin >> maze[i][ii];

if (maze[i][ii] == 'S')

{

visited[i][ii] = true;

queue.push\_back(make\_tuple(i, ii, 0));

}

}

while (!queue.empty())

{

auto x = queue.front();

queue.pop\_front();

if (maze[get<0>(x)][get<1>(x)] == 'E')

{

cout << get<2>(x) << endl;

flag=1;

}

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

{

int newx = get<0>(x) + dx[i];

int newy = get<1>(x) + dy[i];

if (newx >= 0 && newx < n&&newy >= 0 && newy < m&&maze[newx][newy] != 'X'&&visited[newx][newy] == false)

{

queue.push\_back(make\_tuple(newx, newy, get<2>(x) + 1));

visited[newx][newy] = true;

}

}

}

if(flag==0)

cout << "Impossible" << endl;

}

}

**DFS**

char maze[100][100];

bool visited[100][100];

int n, m;

bool dfs(int x, int y)

{

if (visited[x][y])

return false;

visited[x][y] = true;

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

{

for (int ii = -1; ii <= 1; ++ii)

{

if (x + i >= 0 && x + i < n&&y + ii >= 0 && y + ii < m&&maze[x + i][y + ii] == 'X')

dfs(x + i, y + ii);

}

}

return true;

}

int main()

{

memset(visited, false, sizeof(visited));

while (cin >> n >> m)

{

int counter = 0;

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

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

cin >> maze[i][ii];

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

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

if (maze[i][ii] == 'X')

if (dfs(i, ii))

++counter;

cout << counter << endl;

}

}

**最大上升子序列logn\* n^2复杂度**

long long a[1005];

long long dp[1005];

using namespace std;

int main()

{

int n;

while (scanf\_s("%d", &n) && n != 0)

{

memset(dp, 0, sizeof(dp));

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

{

scanf\_s("%lld", &a[i]);

dp[i] = a[i];

}

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

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

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

dp[i] = max(dp[i], dp[j] + a[i]);

long long ans = -10000;

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

if (dp[i] > ans)

ans = dp[i];

printf("%lld\n", ans);

}

}

**接馅饼**

int pie[15][100000];

int maxp[15][100000];

int max3(int a, int b, int c)

{

int max = (a > b ? a : b);

return (max > c ? max : c);

}

int main()

{

int num,p,t;

while (scanf\_s("%d", &num) && num != 0)

{

int maxt = -1;

memset(pie, 0, sizeof(pie));

memset(maxp, 0, sizeof(maxp));

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

{

scanf\_s("%d%d", &p, &t);

pie[p][t]++;

if (maxt < t)

maxt = t;

}

for (int i = 0; i <= 10; i++)

maxp[i][maxt] = pie[i][maxt];

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

for (int j = 0; j <= 10; j++)

maxp[j][i] = pie[j][i] + max3(maxp[j + 1][i + 1], maxp[j][i + 1], maxp[j - 1][i + 1]);

printf("%d\n", maxp[5][0]);

}

}

**01背包**

int dp[1002];

int main()

{

int n, v;

while (cin >> n >> v)

{

memset(dp, 0, sizeof(dp));

int val[1002], cost[1002];

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

cin >> val[i] >> cost[i];

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

for(int j = v; j >= cost[i]; j--)

dp[j] = max(dp[j], dp[j - cost[i]] + val[i]);

cout << dp[v] << endl;

}

}

**完全背包**

int dp[10010];

int main()

{

int n, v;

while (cin>>v>>n)

{

int cost[55], value[55];

memset(dp, 0, sizeof(dp));

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

cin >> cost[i] >> value[i];

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

{

for (int j = cost[i]; j <= v; j++)

{

if (j >= cost[i])

dp[j] = max(dp[j], dp[j - cost[i]] + value[i]);

}

}

cout << dp[v] << endl;

}

}

**多重背包**

int dp[10010];

int main()

{

int n, v;

int cas;

for(cin>>cas;cas>0;cas--)

{

cin >> v >> n;

int cost[110], value[110],num[110];

memset(dp, 0, sizeof(dp));

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

cin >> cost[i] >> value[i] >> num[i];

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

{

int sum = num[i];

for (int k = 1; sum > 0; k \*= 2)

{

int temp = min(k, sum);

for (int j = v; j >= temp \* cost[i]; j--)

dp[j] = max(dp[j], dp[j - temp \* cost[i]] + value[i] \* temp);

sum -= temp;

}

}

cout << dp[v] << endl;

}

}

**三维01背包**

int main()

{

int N,W,E;

int w[110],e[110],v[110];

int dp[2][505][50];

while (scanf("%d%d%d", &N, &W, &E) != EOF)

{

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

scanf("%d%d%d",v+i, w+i, e+i);

}

memset(dp, 0, sizeof(dp));

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

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

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

dp[i%2][j][k] = max(dp[(i+1)%2][j][k], (j+w[i]<=W&&k+e[i]<=E) ? v[i] + dp[(i+1)%2][j+w[i]][k+e[i]] : 0);

}

}

}

int ans = 0;

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

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

ans = max(ans, dp[(N - 1)%2][j][k]);

}

}

printf("%d\n", ans);

}

}

//最长公共子序列

int lcs(String str1, String str2) {

    int len1 = str1.length();

    int len2 = str2.length();

    int c[][] = new int[len1+1][len2+1];

    for (int i = 0; i <= len1; i++) {

        for( int j = 0; j <= len2; j++) {

            if(i == 0 || j == 0) {

                c[i][j] = 0;

            } else if (str1.charAt(i-1) == str2.charAt(j-1)) {

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

            } else {

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

            }

        }

    }

    return c[len1][len2];

}

//最长公共子串

int lcs(String str1, String str2) {

    int len1 = str1.length();

    int len2 = str2.length();

    int result = 0;     //记录最长公共子串长度

    int c[][] = new int[len1+1][len2+1];

    for (int i = 0; i <= len1; i++) {

        for( int j = 0; j <= len2; j++) {

            if(i == 0 || j == 0) {

                c[i][j] = 0;

            } else if (str1.charAt(i-1) == str2.charAt(j-1)) {

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

                result = max(c[i][j], result);

            } else {

                c[i][j] = 0;

            }

1. }
2. }
3. **return** result;
4. }

**欧拉素数筛**

bool IsPrime[1000010];

int Prim[1000010]; //储存第i个素数

void euler\_prime(int n)

{

int num = 0;

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

{

if (!IsPrime[i])

Prim[num++] = i;

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

{

if (i \* Prim[j] > n)

break;

IsPrime[i \* Prim[j]] = true;

if (i % Prim[j] == 0)

break;

}

}

}

**快速幂**

int quick\_pow(int x, int n){ int res = 1, p = x; while(n) { if(n & 1) res \*= p; p \*= p; n >>= 1; } return res;}

**快速幂取模**

int quick\_pow(int x, int n,int k)

{

int res = 1, p = x;

while (n)

{

if (n & 1)

{

res \*= p;

res %= k;

}

p \*= p;

n >>= 1;

}

return res%k;

}

大组合数取模  
ll f[100005], inv[100005];

const ll mod = 1e9 + 7;

const int maxn = 1e5 + 5;

ll q\_pow(ll a, int b) {

ll ret = 1;

for (; b; b >>= 1, a = a \* a % mod) {

if (b & 1) ret = ret \* a % mod;

}

return ret;

}

void init()

{

f[0] = inv[0] = 1;

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

f[i] = f[i - 1] \* i % mod;

inv[maxn - 1] = q\_pow(f[maxn - 1], mod - 2);

for (int i = maxn - 2; i; --i)

inv[i] = inv[i + 1] \* (i + 1) % mod;

}

ll C(int n, int m)

{

if (n < m)

return 0;

return f[n] \* inv[m] % mod \* inv[n - m] % mod;

}

**快速阶乘及位数**

int main()

{

int num; //计算阶乘的数字

while (cin >> num)

{

int result[40000]; //保存结算结果的数组

int height = 1; //结果的最高位

result[0] = 1;

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

{

int res = 0; //进位

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

{

int buf = result[j] \* i + res; //计算结果

result[j] = buf % 10; //取当前位

res = buf / 10; //计算进位

}

while (res)

{

result[height++] = res % 10; //取当前位

res /= 10; //计算进位

}

}

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

{

cout << result[k];

}

cout << endl;

cout << "length=" << height << endl;

}

}

**大数阶乘取模**

const int a[100] =

{

682498929,491101308,76479948,723816384,67347853,27368307,625544428,199888908,888050723,927880474,

281863274,661224977,623534362,970055531,261384175,195888993,66404266,547665832,109838563,933245637,724691727,

368925948,268838846,136026497,112390913,135498044,217544623,419363534,500780548,668123525,128487469,30977140,

522049725,309058615,386027524,189239124,148528617,940567523,917084264,429277690,996164327,358655417,568392357,

780072518,462639908,275105629,909210595,99199382,703397904,733333339,97830135,608823837,256141983,141827977,

696628828,637939935,811575797,848924691,131772368,724464507,272814771,326159309,456152084,903466878,92255682,

769795511,373745190,606241871,825871994,957939114,435887178,852304035,663307737,375297772,217598709,624148346,

671734977,624500515,748510389,203191898,423951674,629786193,672850561,814362881,823845496,116667533,256473217,

627655552,245795606,586445753,172114298,193781724,778983779,83868974,315103615,965785236,492741665,377329025,

847549272,698611116

};

const int MOD = 1000000007;

int main()

{

long long n, p;

long long now;

while (cin >> n >> p)

{

if (p == 1000000007)

{

if (n >= p)

{

cout << "0";

}

if (n < 10000000)

now = 1;

else now = a[n / 10000000 - 1];

for (int i = n / 10000000 \* 10000000 + 1; i <= n; i++)

now = now \* i%MOD;

}

else

{

now = 1;

if (n >= p) now = 0;

else

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

now = now \* i%p;

}

cout << now << endl;

}

}

**欧几里得算法/辗转相除法（求最大公因数）**

int gcd(int a, int b)

{ if(b == 0) return a; return gcd(b, a % b);}

**求最小公倍数**

int lcm(int x, int y)

{ return x / gcd(x, y) \* y;}

**扩展欧几里得算法//求ax+by=c特解**

int exgcd(int a, int b, int &x, int &y) {

if(!b) {

x = 1, y = 0;

return a;

}

int d = exgcd(b, a % b, y, x);

y -= a / b \* x;

return d;

}

bool solve(long long a, long long b, long long c, long long &x, long long&y)

{

long long n = exgcd(a, b, x, y);

if (c%n)

return false;

long long k = c / n;

x \*= k;

y \*= k;

return true;

}

int main()

{

long long a, b, c;

while (cin >> a >> b >> c)

{

long long x, y;

if (!solve(a, b, c, x, y))

cout << "-1" << endl;

else cout << x << y << endl;

}

}

**单模板匹配KMP**

vector<int> finds(string pattern, string text)

{

int n = pattern.size();

vector<int>next(n + 1, 0);

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

{

int j = i;

while (j > 0)

{

j = next[j];

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

{

next[i + 1] = j + 1;

break;

}

}

}

vector<int>positions;

int m = text.size();

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

{

if (j < n&&text[i] == pattern[j])

j++;

else

{

while (j > 0)

{

j = next[j];

if (text[i] == pattern[j])

{

j++;

break;

}

}

}

if (j == n)

positions.push\_back(i - n + 1);

}

return positions;

}

int main()

{

string pattern;

string text;

int n;

for (cin >> n; n > 0; n--)

{

cin >> pattern >> text;

vector<int>ans = finds(pattern, text);

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

cout << ans[i] << " ";

cout << ans[ans.size() - 1] << endl;

}

}

**Manacher 最大回文串长度**

int N, p[10010];

char str[10010], b[10010];

void init()

{

int i;

for (i = 0; str[i]; i++) b[2 \* i + 1] = '#', b[2 \* i + 2] = str[i];

N = 2 \* i + 1;

b[0] = '$', b[N] = b[N + 1] = '#';

}

void solve()

{

int i, id, max = 0, ans = 0;

for (i = 1; i <= N; i++)

{

p[i] = i < max ? min(max - i, p[2 \* id - i]) : 1;

while (b[i + p[i]] == b[i - p[i]]) ++p[i];

if (i + p[i] > max) max = i + p[i], id = i;

ans=max(ans, p[i] - 1);

}

printf("%d\n", ans);

}

int main()

{

while (scanf("%s", str) == 1)

{

init();

solve();

}

return 0;

}

**AC自动机**

int out[505];

int f[505];

int g[505][26];

int buildMatchingMachine(string arr[], int k)

{

memset(out, 0, sizeof out);

memset(g, -1, sizeof g);

int states = 1;

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

{

const string &word = arr[i];

int currentState = 0;

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

{

int ch = word[j] - 'a';

if (g[currentState][ch] == -1)

g[currentState][ch] = states++;

currentState = g[currentState][ch];

}

out[currentState] |= (1 << i);

}

for (int ch = 0; ch < 26; ++ch)

if (g[0][ch] == -1)

g[0][ch] = 0;

memset(f, -1, sizeof f);

deque<int> q;

for (int ch = 0; ch < 26; ++ch)

{

if (g[0][ch] != 0)

{

f[g[0][ch]] = 0;

q.push\_back(g[0][ch]);

}

}

while (q.size())

{

int state = q.front();

q.pop\_front();

for (int ch = 0; ch <= 26; ++ch)

{

if (g[state][ch] != -1)

{

int failure = f[state];

while (g[failure][ch] == -1)

failure = f[failure];

failure = g[failure][ch];

f[g[state][ch]] = failure;

out[g[state][ch]] |= out[failure];

q.push\_back(g[state][ch]);

}

}

}

return states;

}

int findNextState(int currentState, char nextInput)

{

int answer = currentState;

int ch = nextInput - 'a';

while (g[answer][ch] == -1)

answer = f[answer];

return g[answer][ch];

}

void searchWords(string arr[], int k, string text)

{

buildMatchingMachine(arr, k);

int currentState = 0;

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

{

currentState = findNextState(currentState, text[i]);

if (out[currentState] == 0)

continue;

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

{

if (out[currentState] & (1 << j))

{

cout << "Word " << arr[j] << " appears from "

<< i - arr[j].size() + 1 << " to " << i << endl;

}

}

}

}

int main()

{

string arr[] = { "he", "she", "hers", "his" };

string text = "ahishers";

int k = sizeof(arr) / sizeof(arr[0]);

searchWords(arr, k, text);

}

**寻找最小加边数//畅通工程，并查集**

int a[1005];

int find(int x)

{

if(x!=a[x])

a[x] = find(a[x]);

return a[x];

}

int main()

{

int n, m;

while (cin >> n >> m)

{

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

a[i] = i;

int j = n - 1;

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

{

int x, y;

cin >> x >> y;

x = find(x);

y = find(y);

if (x != y)

{

a[y] = x;

j--;

}

}

cout << j << endl;

}

}

**最小生成树**

int a[1010];

struct stru

{

int u, v, w;

}road[5050];

int find(int x)

{

if (x != a[x])

a[x] = find(a[x]);

return a[x];

}

bool cmp(const stru a, const stru b)

{

return a.w < b.w;

}

int main()

{

int n;

while (cin >> n && n != 0)

{

memset(road, 0, sizeof(road));

memset(a, 0, sizeof(a));

int m = n \* (n - 1) / 2;

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

a[i] = i;

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

cin >> road[i].u >> road[i].v >> road[i].w;

sort(road, road + m, cmp);

int sum = 0;

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

{

int x = find(a[road[i].u]), y = find(a[road[i].v]);

if (x != y)

{

sum += road[i].w;

a[x] = y;

}

}

cout << sum << endl;

}

}

**最大生成树**

using namespace std;

struct edge

{

int u, v, w;

}e[100010];

int a[30010];

int find(int x)

{

if (x != a[x])

a[x] = find(a[x]);

return a[x];

}

bool cmp(const edge a, const edge b)

{

return a.w > b.w;

}

int main()

{

int n, m, r;

int cas;

for(cin>>cas;cas>0;cas--)

{

scanf("%d%d%d", &n, &m, &r);

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

scanf("%d%d%d", &e[i].u, &e[i].v, &e[i].w);

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

a[i] = i;

sort(e, e + r, cmp);

long long sum = 0;

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

{

int x = find(e[i].u+1), y = find(e[i].v+n+1);

if (x != y)

{

sum += e[i].w;

a[x] = y;

}

}

printf("%d\n", sum);

}

}

**单源最短路：SPFA 可处理负权边**

int q[10001], dis[201], a[201][201], b[201][201];

bool vis[201];

int n, m, s, t;

void spfa(int s) {

for (int i = 0; i <= n; i++) dis[i] = 99999999;

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

int i, v, head = 0, tail = 1;

while (head<tail)

{

head++;

v = q[head];

vis[v] = 0;

for (i = 1; i <= b[v][0]; i++)

if (dis[b[v][i]] > dis[v] + a[v][b[v][i]])

{

dis[b[v][i]] = dis[v] + a[v][b[v][i]];

if (vis[b[v][i]] == 0)

{

tail++;

q[tail] = b[v][i];

vis[b[v][i]] = 1;

}

}

}

}

int main() {

int x, y, z;

while (cin >> n >> m)

{

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

{

cin >> x >> y >> z;

if (a[x][y] != 0 && z > a[x][y])

continue;

b[x][0]++; b[x][b[x][0]] = y;

a[x][y] = z;

b[y][0]++; b[y][b[y][0]] = x; a[y][x] = z;

}

cin >> s >> t;

spfa(s);

if (dis[t] != 99999999)

cout << dis[t] << endl;

else

cout << -1 << endl;

}

}

**多源最短路Floyd**

int G[1002][1002];

void Floyd(int n)

{

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

{

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

{

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

{

G[i][j] = min(G[i][j], G[i][k] + G[k][j]);

}

}

}

}

int main()

{

int dist[1002][1002];

int m, n;

while (cin >> n >> m, m + n)

{

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

{

G[i][i] = 0;

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

G[i][j] = G[j][i] = 0xfffffff;

}

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

{

int a, b, c;

cin >> a >> b >> c;

G[a][b] = min(G[a][b], c);

G[b][a] = G[a][b];

}

Floyd(n);

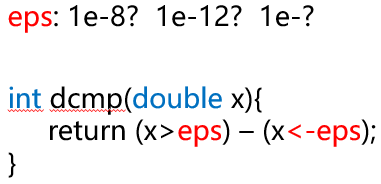
cout << G[1][n] << endl;

}

}

**计算几何**

精度误差



存点 存线

struct Point{ struct Line{

double x, y; // z Point a, b;

} }

点积：a · b = |a| |b| cosθ 叉积：a × b = |a| |b| sinα

负交换律： a × b = - b × a

double dot(const Point& a, const Point& b){ double det(const Point& a, const Point& b){

return a.x\*b.x + a.y\*b.y; return a.x\*b.y - a.y\*b.x;

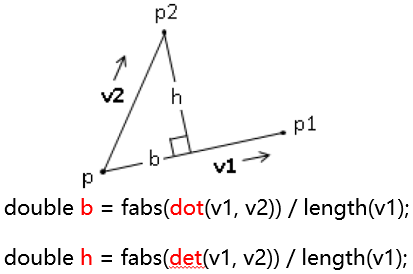
} }

a · b > 0 : 夹角为锐角 P × Q ＞0 P在Q的顺时针方向

a · b = 0 : 夹角为直角 P × Q = 0 P与Q共线(0°/180°)

a · b < 0 : 夹角为钝角 P × Q ＜0 P在Q的逆时针方向

投影和高度



线段绕端点逆时针旋转θ 判断直线是否相交

tx = A.b.x – A.a.x; ty = A.b.y-A.a.y; double s1 = det(A.a - B.a, B.b - B.a);

A.b.x = tx cos θ - ty sin θ + A.a.x; double s2 = det(A.b - B.a, B.b - B.a);

A.b.y = tx sin θ + ty cos θ + A.a.y; Point p = (s1\*A.b - s2\*A.a)/(s1 – s2);

再判断P是否在两直线上