Capybara dreaming

Contents

1 Problemas

1.1 coloracao galaxy collision

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

using namespace std;

const int MAXN1 = 60000, MAXN2 = 60000, MAXM = 200000;
int n1, n2, edges, last[MAXN1], prevs[MAXM], head[MAXM], matching[MAXN2], dist[MAXN1], Q[MAXN1], used[MAXN1],
    vis[MAXN1];

typedef pair<int, int> ii;

void init(int _n1, int _n2)
{
    n1 = _n1;
    n2 = _n2;
    edges = 0;
    fill(last, last + n1, -1);
}

void addAresta(int u, int v)
{
    head[edges] = v;
    prevs[edges] = last[u];
    last[u] = edges++;
}

void bfs()
{
    fill(dist, dist + n1, -1);
    int sizeQ = 0;
    for (int u = 0; u < n1; ++u)
    {
        if (!used[u])
        {
            Q[sizeQ++] = u;
            dist[u] = 0;
        }
    }
    for (int i = 0; i < sizeQ; i++)
    {
        int u1 = Q[i];
        for (int e = last[u1]; e >= 0; e = prevs[e])
        {
            int u2 = matching[head[e]];
            if (u2 >= 0 && dist[u2] < 0)
            {
                dist[u2] = dist[u1] + 1;
                Q[sizeQ++] = u2;
            }
        }
    }
}

bool dfs(int u1)
{
    vis[u1] = true;
    for (int e = last[u1]; e >= 0; e = prevs[e])
    {
        int v = head[e];
        int u2 = matching[v];
        if (u2 < 0 || !vis[u2] && dist[u2] == dist[u1] + 1 && dfs(u2))
        {
            matching[v] = u1;
            used[u1] = true;
            return true;
        }
    }
    return false;
}

int maxMatching()
{
    fill(used, used + n1, false);
    fill(matching, matching + n2, -1);
    for (int res = 0;;)
    {
        bfs();
        fill(vis, vis + n1, false);
        int f = 0;
```

```
        for (int u = 0; u < n1; ++u)
            if (!used[u] && dfs(u))
                ++f;
        if (!f)
            return res;
        res += f;
    }
}
int main() {
        int n, i, j, a, b;
        map<ii, int> pontos;

        vector<ii> validos;

        for(i = -5; i <= 5; i++)
                for(j = -5; j <= 5; j++)
                        if(hypot(i, j) <= 5 && (i || j))
                                validos.push_back(ii(i, j));

        while(scanf("%d", &n) > 0) {
                pontos.clear();

                for(i = 0; i < n; i++)
                        scanf("%d %d", &a, &b), pontos[ii(a, b)] = i;

                init(n + 1, n + 1);

                for(auto &it : pontos)
                        for(auto &dif : validos) {
                                auto x = it.first.first + dif.first;
                                auto y = it.first.second + dif.second;

                                if(pontos.count(ii(x, y)))
                                        addAresta(it.second, pontos[ii(x, y)]);
                        }

                printf("%d\n", maxMatching() / 2);
        }
}
```

1.2 convex hull trick

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

using namespace std;
int pointer;         //Keeps track of the best line from previous query
vector<long long> M; //Holds the slopes of the lines in the envelope
vector<long long> B; //Holds the y-intercepts of the lines in the envelope

//Returns true if either line l1 or line l3 is always better than line l2
bool bad(int l1, int l2, int l3)
{
    /*
        intersection(l1,l2) has x-coordinate (b1-b2)/(m2-m1)
        intersection(l1,l3) has x-coordinate (b1-b3)/(m3-m1)
        set the former greater than the latter, and cross-multiply to
        eliminate division
        */
    return (B[l3] - B[l1]) * (M[l1] - M[l2]) < (B[l2] - B[l1]) * (M[l1] - M[l3]);
}

//Adds a new line (with lowest slope) to the structure
void add(long long m, long long b)
{
    //First, let's add it to the end
    M.push_back(m);
    B.push_back(b);
    //If the penultimate is now made irrelevant between the antepenultimate
    //and the ultimate, remove it. Repeat as many times as necessary
    while (M.size() >= 3 && bad(M.size() - 3, M.size() - 2, M.size() - 1))
    {
        M.erase(M.end() - 2);
        B.erase(B.end() - 2);
    }
}

//Returns the minimum y-coordinate of any intersection between a given vertical
//line and the lower envelope
long long query(long long x)
{
    //If we removed what was the best line for the previous query, then the
    //newly inserted line is now the best for that query
    if (pointer >= M.size())
        pointer = M.size() - 1;
    //Any better line must be to the right, since query values are
    //non-decreasing
    while (pointer < M.size() - 1 &&
           M[pointer + 1] * x + B[pointer + 1] < M[pointer] * x + B[pointer])
        pointer++;
    return M[pointer] * x + B[pointer];
}

int main()
{
    int M, N, i;

    pair<int, int> a[50000];
    pair<int, int> rect[50000];

    freopen("acquire.in", "r", stdin);
    freopen("acquire.out", "w", stdout);

    scanf("%d", &M);

    for (i = 0; i < M; i++)
        scanf("%d %d", &a[i].first, &a[i].second);
    //Sort first by height and then by width (arbitrary labels)

    sort(a, a + M);

    for (i = 0, N = 0; i < M; i++)
    {
        /*
                When we add a higher rectangle, any rectangles that are also
                equally thin or thinner become irrelevant, as they are
                completely contained within the higher one; remove as many
                as necessary
                */
        while (N > 0 && rect[N - 1].second <= a[i].second)
            N--;
        rect[N++] = a[i]; //add the new rectangle
    }

    long long cost;
```

```
    add(rect[0].second, 0);
    //initially, the best line could be any of the lines in the envelope,
    //that is, any line with index 0 or greater, so set pointer=0

    pointer = 0;

    for (i = 0; i < N; i++) //discussed in article
    {
        cost = query(rect[i].first);
        if (i < N - 1)
            add(rect[i + 1].second, cost);
    }

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

    return 0;
}
```

1.3 eliminacao gauss xor loteria

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

using namespace std;

typedef vector<int> vi;
typedef vector<vi> vvi;

int main()
{
    int n, k, matriz[10000][50], linha, coluna, rank, i, j, e;

    scanf("%d %d", &n, &k);

    for (i = 0; i < n; i++)
        for (j = 0; j < k; j++)
            scanf("%d", &e), matriz[i][j] = e & 1;

    linha = coluna = rank = 0;

    while (linha < n && coluna < k)
    {
        for (i = linha; i < n; i++)
            if (matriz[i][coluna] == 1)
                break;

        if (i == n || matriz[i][coluna] == 0)
        {
            puts("S");

            return 0;
        }

        for (j = 0; j < k; j++)
            swap(matriz[linha][j], matriz[i][j]);

        for (i = 0; i < n; i++)
        {
            if (i == linha || matriz[i][coluna] == 0)
                continue;

            for (j = 0; j < k; j++)
                matriz[i][j] ^= matriz[linha][j];
        }

        linha++, coluna++;
    }

    if(linha >= k && n > linha) {
        puts("N");

        return 0;
    }

    puts("S");
}
```

1.4 fft laboratorio biotecnologia

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

using namespace std;

const int MAX_DIST = 1 << 23;

typedef complex<double> cpx;
const double pi = acos(-1.0);

char txt[100000];
int maxDist;

// in:     vector de entrada
// type:   1 = Transformada, -1 = Transformada inversa
void fft (vector<cpx> &a, bool invert) {
        int n = (int) a.size();

        for (int i=1, j=0; i<n; ++i) {
                int bit = n >> 1;
                for (; j>=bit; bit>>=1)
                        j -= bit;
                j += bit;

                if (i < j)
                        swap (a[i], a[j]);
        }

        for (int len=2; len<=n; len<<=1) {
                double ang = 2*pi/len * (invert ? -1 : 1);
                cpx wlen (cos(ang), sin(ang));
                for (int i=0; i<n; i+=len) {
                        cpx w (1);
                        for (int j=0; j<len/2; ++j) {
                                cpx u = a[i+j],  v = a[i+j+len/2] * w;
                                a[i+j] = u + v;
                                a[i+j+len/2] = u - v;
                                w *= wlen;
                        }
                }
        }

        if (invert)
                for (int i=0; i<n; ++i)
                        a[i] /= n;
}

int main()
{
        int soma = 0, acc = 0, tam = 0, c;
    vector<cpx> fftEsq(MAX_DIST), fftDir(MAX_DIST);

        while((c = getchar()) >= 'a') {
                soma += txt[tam++] = c - 96;
                fftEsq[soma] = cpx(1, 0);
        }

        fftDir[soma] = cpx(1, 0);

        for(int i = 0; i < tam; i++) {
                acc += txt[i];

                fftDir[soma - acc] = cpx(1, 0);
        }

        int shiftAmount, lim = 2 * soma;

        for (shiftAmount = 0; (lim >> shiftAmount) != 0; shiftAmount++)
                ;

        maxDist = 1 << shiftAmount;

        fftEsq.resize(maxDist);
        fftDir.resize(maxDist);

        fft(fftEsq, false);
        fft(fftDir, false);

        for (int i = 0; i < maxDist; i++)
                fftEsq[i] = fftDir[i] * fftEsq[i];

        fft(fftEsq, 1);

        int total = 0;

        assert(lim < maxDist);
```

```
        for (int i = soma + 1; i <= lim; i++)
        {
                if (fftEsq[i].real() > 0.01)
                        total++;
        }
        printf("%d\n", total);
}
```

1.5 geometria bloco preto branco

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

using namespace std;

struct Bloco {
    long long qt;
    char c;
};

Bloco bloco[200000];
long long totB, totW;
long long n;

long long getX(long long y) {
        return ceil((totB * y)*1.0/(totW));
}
long long getY(long long x) {
        return ceil((totW * x)*1.0/(totB));
}
int main() {
    long long t;
    cin >> t;

    while(t--) {
        cin >> n;

        totB = totW = 0;

        for(int i=0; i<n; i++) {
            scanf("%lld %c", &bloco[i].qt, &bloco[i].c);
            if(bloco[i].c == 'W')
                totW += bloco[i].qt;
            else
                totB += bloco[i].qt;
        }

        if(totW == 0 || totB == 0) {
            cout << totW + totB << endl;
            continue;
        }

                long long x = 0, y = 0;
                long long x_a = 0, y_a = 0;
                long long cont = 0;

        for(int i=0; i<n; i++) {
                if(i < n-1 && bloco[i].c == bloco[i+1].c) {
                        if(bloco[i].c == 'B')
                                x_a += bloco[i].qt;
                        else
                                y_a += bloco[i].qt;
                }
                else {
                        if(bloco[i].c == 'B') {
                                x_a += bloco[i].qt;
                                double local = getX(y);
                                if(local >= x && local < x + x_a)
                                        if((totB * y) % totW == 0)
                                                cont++;
                                x += x_a;
                                x_a = 0;
                        }
                        else {
                                y_a += bloco[i].qt;
                                double local = getY(x);
                                if(local >= y && local < y + y_a)
                                        if((totW * x) % totB == 0)
                                                cont++;
                                y += y_a;
                                y_a = 0;
                        }
                }
        }

        cout << cont << endl;
    }

    return 0;
}
```

1.6 geometria bolo do marcelo

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

using namespace std;

#define D(x) cout << #x << " = " << x << endl;

typedef pair<int, int> Ponto;

//O dobro da área definida pelo triangulo de pontos pontos a, b e c (com sinal).
long long area2(Ponto &a, Ponto &b, Ponto &c) {
    return (long long) (b.first - a.first) * (c.second - a.second) - (long long) (b.second - a.second) * (c.
        first - a.first);
}
//Retorna a área do polígono p definido pelos pontos p[i, f]
long long polygonArea2(vector<Ponto> &p, int i, int f) {
    long long s = 0.0;

    Ponto& primeiro = p[i];
    Ponto origem = Ponto(0, 0);

    for (; i != f; i = (i + 1) % p.size())
        s += area2(origem, p[i], p[(i + 1) % p.size()]);

    s += area2(origem, p[i], primeiro);

    return s;
}
long long corta(vector<Ponto> &p, long long &areaTot) {
    int atual = 0;
    int k = 0;
    int m = p.size();

    long long area = 0, areaVelha = 0;
    long long resp = 1ll<<62;

    for (atual = 0; atual < m; atual++) {
        while (true) {
            if (2 * area > areaTot) {
                areaVelha = area - area2(p[atual], p[(m + k - 1) % m], p[k]);

                resp = min(resp, max(areaTot - area, min(areaVelha, areaTot - areaVelha)));

                break;
            }
            k = (k + 1) % m;

            area += area2(p[atual], p[(m + k - 1) % m], p[k]);
        }
        area -= area2(p[atual], p[(atual + 1) % m], p[k]);
    }
    return resp;
}
int main() {
    int n;

    scanf("%d", &n);

    vector<Ponto> pontos(n);

    for (auto &it : pontos)
        scanf("%d %d", &it.first, &it.second);

    auto area = polygonArea2(pontos, 0, pontos.size() - 1);
    auto resp = corta(pontos, area);

    printf("%lld %lld\n", area - resp, resp);

    return 0;
}
```

1.7 geometria fila mineira

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

#define D(x) cout << #x << " = " << x << endl

using namespace std;

typedef pair<int, int> ii;
typedef vector<int> vi;
typedef vector<vi> vvi;
typedef vector<ii> vii;
typedef vector<vii> vvii;

#define pb push_back

vii pontos;

double f(double theta)
{
    double y_max = -1e100, y_min = 1e100, y;
    double sint = sin(theta), cost = cos(theta);

    for(auto &it : pontos) {
        y = -sint * it.first + cost * it.second;

        y_max = max(y, y_max);
        y_min = min(y, y_min);
    }

    return (y_max - y_min) / 2;
}

double gss(double a, double b)
{
    double r = (sqrt(5) - 1) / 2; //=.618...=golden ratio-1
    double x1 = b - r * (b - a), x2 = a + r * (b - a);
    double f1 = f(x1), f2 = f(x2);

    // while(fabs(f1 - f2) > e)
    for(int it = 0; it < 100; it++)
    {
        //change to > to find maximum
        if (f1 < f2)
        {
            b = x2;
            x2 = x1;
            f2 = f1;
            x1 = b - r * (b - a);
            f1 = f(x1);
        }
        else
        {
            a = x1;
            x1 = x2;
            f1 = f2;
            x2 = a + r * (b - a);
            f2 = f(x2);
        }
    }

    // return f1;
    return (f2 + f1) / 2;
}

int main()
{
    int n;

    cin >> n;

    pontos = vii(n);

    for(auto &it : pontos)
        cin >> it.first >> it.second;

    auto resp = gss(0, 90);
    resp = min(resp, gss(90, 180));
    resp = min(resp, gss(180, 270));
    resp = min(resp, gss(270, 360));

    printf("%.2lf\n", resp);
}
```

1.8 grafo caminho unico entre 2 vertices

```
#include <bits/stdc++.h>
using namespace std;

typedef vector<int> vi;
typedef pair<int, int> ii;
typedef vector<ii> vii;
typedef vector<vii> vvii;

#define D(x) cout << #x " = " << x << endl;

vvii grafo;

vi dfs_num, dfs_low, dfs_parent, S, visited, pontes;
int dfsNC, root, rootChildren;

void tarjan(int u) {
        dfs_low[u] = dfs_num[u] = dfsNC++;

        S.push_back(u);

        visited[u] = 1;

        for(auto &it : grafo[u]) {
                if(dfs_num[it.first] == -1) {
                        dfs_parent[it.first] = u;

                        if(u == root)
                                rootChildren++;

                        tarjan(it.first);

                        if(dfs_low[it.first] > dfs_num[u])
                                pontes[it.second] = 1;

                        dfs_low[u] = min(dfs_low[u], dfs_low[it.first]);
                }
                else if(it.first != dfs_parent[u])
                        dfs_low[u] = min(dfs_low[u], dfs_low[it.first]);
        }
}

int dfs1(int atual, int destino) {
        if(atual == destino)
                return 1;

        if(visited[atual])
                return 0;

        visited[atual] = 1;

        int r = 0;

        for(auto &it : grafo[atual])
                if(pontes[it.second])
                        r += dfs1(it.first, destino);

        return r;
}

int main()
{
        int r, c, q, a, b, i;

        while(scanf("%d %d %d", &r, &c, &q) && r) {
                grafo = vvii(r);

                pontes = vi(c + 1, 0);

                while(c--)
                        scanf("%d %d", &a, &b),
                        grafo[a-1].push_back({b-1, c}),
                        grafo[b-1].push_back({a-1, c});

                dfs_num = vi(r, -1);
                dfs_parent = vi(r, -1);
                dfs_low = vi(r, 0);
                visited = vi(r, 0);
                dfsNC = 0;

                for(i = 0; i < r; i++)
                        if(dfs_num[i] == -1)
                                root = i, rootChildren = 0, tarjan(i);

                while(q--) {
                        visited = vi(r, 0);
                        scanf("%d %d", &a, &b);
```

```
                printf("%c\n", dfs1(a - 1, b - 1) ? 'Y' : 'N');
            }
            puts("-");
        }
}
```

1.9 grafo centro arvore junte dois reinos

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

using namespace std;

#define pb push_back

typedef vector<int> vi;
typedef vector<vi> vvi;

vvi a_esq, a_dir;
vi grau_esq, grau_dir, tam_esq, tam_dir, dist_esq, dist_dir;
int centro_esq, centro_dir;

int dfs_esq(int atual, int pai) {
        int r = 0;

        for(auto &it : a_esq[atual])
                if(it != pai)
                        r = max(r, dfs_esq(it, atual));

        return tam_esq[atual] = r + 1;
}

int dfs_dir(int atual, int pai) {
        int r = 0;

        for(auto &it : a_dir[atual])
                if(it != pai)
                        r = max(r, dfs_dir(it, atual));

        return tam_dir[atual] = r + 1;
}

void calc_dist_esq(int atual, int maior) {
        dist_esq[atual] = maior;

        for(auto &it : a_esq[atual])
                if(dist_esq[it] == -1 && it != centro_esq)
                        calc_dist_esq(it, maior + 1);
}

void calc_dist_dir(int atual, int maior) {
        dist_dir[atual] = maior;

        for(auto &it : a_dir[atual])
                if(dist_dir[it] == -1 && it != centro_dir)
                        calc_dist_dir(it, maior + 1);
}

int main() {
        ios::sync_with_stdio(false);
        cin.tie(0);

        long long soma;
        int n, q, i, a, b, maxi, max_dir, max_esq;

        while(cin >> n >> q) {
                multiset<int> valores_esq, valores_dir;

                a_esq = vvi(n);
                a_dir = vvi(q);

                grau_esq = vi(n);
                grau_dir = vi(q);

                tam_esq = vi(n);
                tam_dir = vi(q);

                dist_esq = vi(n, -1);
                dist_dir = vi(q, -1);

                for(i = 1; i < n; i++)
                        cin >> a >> b,
                        a--,
                        b--,
                        a_esq[a].pb(b),
                        grau_esq[a]++,
                        a_esq[b].pb(a),
                        grau_esq[b]++;

                for(i = 1; i < q; i++)
                        cin >> a >> b,
                        a--,
                        b--,
                        a_dir[a].pb(b),
```

```
                grau_dir[a]++,
                a_dir[b].pb(a),
                grau_dir[b]++;

        queue<int> fila;

// Centro da arvore esquerda
        for(i = 0; i < n; i++)
                if(a_esq[i].size() <= 1) // <= por árvore vazia!
                        fila.push(i);

        while(fila.size() > 1) {
                for(auto &it : a_esq[fila.front()]) {
                        grau_esq[it]--;

                        if(grau_esq[it] == 1)
                                fila.push(it);
                }

                fila.pop();
        }

        centro_esq = fila.front();
        fila.pop();

// Centro da arvore direita
        for(i = 0; i < q; i++)
                if(a_dir[i].size() <= 1) // <= por árvore vazia!
                        fila.push(i);

        while(fila.size() > 1) {
                for(auto &it : a_dir[fila.front()]) {
                        grau_dir[it]--;

                        if(grau_dir[it] == 1)
                                fila.push(it);
                }

                fila.pop();
        }

        centro_dir = fila.front();
        fila.pop();

        dfs_esq(centro_esq, -1);
        dfs_dir(centro_dir, -1);

        valores_esq.insert(0);
        for(auto &it : a_esq[centro_esq])
                valores_esq.insert(-tam_esq[it]);

        for(auto &it : a_esq[centro_esq]) {
                valores_esq.erase(valores_esq.find(-tam_esq[it]));

                calc_dist_esq(it, -(*valores_esq.begin()) + 1);

                valores_esq.insert(-tam_esq[it]);
        }

        dist_esq[centro_esq] = -(*valores_esq.begin());

        valores_dir.insert(0);
        for(auto &it : a_dir[centro_dir])
                valores_dir.insert(-tam_dir[it]);

        for(auto &it : a_dir[centro_dir]) {
                valores_dir.erase(valores_dir.find(-tam_dir[it]));

                calc_dist_dir(it, -(*valores_dir.begin()) + 1);

                valores_dir.insert(-tam_dir[it]);
        }

        dist_dir[centro_dir] = -(*valores_dir.begin());

        soma = maxi = max_esq = max_dir = 0;

        for(auto &it : dist_esq)
                max_esq = max(max_esq, it);

        for(auto &it : dist_dir)
                max_dir = max(max_dir, it);

        maxi = max(max_esq, max_dir);

        vi v_esq(max_esq + 1, 0), v_dir(max_dir + 1, 0);

        for(auto &i : dist_esq)
```

```
                        v_esq[i]++;

                for(auto &i : dist_dir)
                        v_dir[i]++;

                for(int i = 0; i <= max_esq; i++)
                        for(int j = 0; j <= max_dir; j++)
                                if(v_esq[i] && v_dir[j])
                                        soma += v_esq[i] * v_dir[j] * max(i + j + 1, maxi);

                printf("%.3lf\n", 1.0 * soma / (q * n));
        }
}
```

1.10 grundy jogo da velha 1d

```
#include <iostream>
#include <cstring>
using namespace std;

#define MAX 10000

bool ganhou(char jogo[])
{
        char *p;

        if ((p=strstr(jogo,"XX"))!=NULL)
                return true;
        if ((p=strstr(jogo,"X.X"))!=NULL)
                return true;
        return false;
}

int main()
{
        int grundyxx[MAX+1];
        int grundy_x[MAX+1];
        int grundy__[MAX+1];

        grundyxx[2] = 0;
        grundyxx[3] = 0;
        grundyxx[4] = 0;

        for(int i=5;i<=MAX;i++) {
                bool s[MAX+1]={false};
                for(int j=2;j<=(i-1)/2;j++)
                        s[grundyxx[j] ^ grundyxx[i-j-1]]=true;
                for(int j=0;j<i;j++)
                        if (!s[j]) {
                                grundyxx[i] = j;
                                break;
                        }
        }

        grundy_x[1] = 0;
        grundy_x[2] = 0;
        grundy_x[3] = 1;
        for(int i=4;i<=MAX;i++) {
                bool s[MAX+1]={false};
                s[grundyxx[i-1]]=true;
                for(int j=2;j<=i-3;j++)
                        s[grundyxx[j] ^ grundy_x[i-j-1]]=true;
                for(int j=0;j<i;j++)
                        if (!s[j]) {
                                grundy_x[i] = j;
                                break;
                        }
        }

        grundy__[3] = 1;
        for(int i=4;i<=MAX;i++) {
                if (grundy_x[i-1]==0)   //Jogar X-
                        grundy__[i] = 1;
                else {
                        grundy__[i] = 0;
                        for(int j=1;j<=(i-1)/2;j++)
                                if ((grundy_x[j] ^ grundy_x[i-j-1]) == 0) {     //Jogar (-)X(-)
//cout << i << " " << j << "(" << grundy_x[j] << " " << grundy_x[i-j-1] << ")" << endl;
                                        grundy__[i] = 1;
                                        break;
                                }
                }
        }

//      cout << "Done!" << endl;

        int n, g = 0;
        char jogo[MAX+1];
        int i, j;

        cin >> n;
        while(n) {
                cin >> jogo;

                if (ganhou(jogo)) {
```

```
                        cout << 'S' << endl;
                        cin >> n;
                        continue;
                }
                for(i=0;jogo[i]!='\0';i++)
                        if(jogo[i]=='X')
                                break;
                if (jogo[i]=='\0') {    //sem X
                        g = grundy__[i];
//                      cout << "--  " << i << " g=" << grundy__[i] << endl;
                }
                else {
                        if (i>0) { //nao comeca com X
                                g = grundy_x[i];
//                              cout << "-x  " << i << " g=" << grundy_x[i] << endl;
                        }
                        else
                                g = 0;
                        for(j=i+1;jogo[j]!='\0';j++) {
                                if (jogo[j]=='X') {
                                        g ^= grundyxx[j-i-1];
//                                      cout << "x-x " << j-i-1 << " g=" << grundyxx[j-i-1] << endl;
                                        i = j;
                                }
                        }
                        if (jogo[j-1]!='X') { //nao termina com X
                                g ^= grundy_x[j-i-1];
//                              cout << "x-  " << j-i-1 << " g=" << grundy_x[j-i-1] << endl;
                        }
                }
                cout << (g?'S':'N') << endl;
                cin >> n;
        }
        return 0;
}
```

1.11 guloso azeitona do vovo pepe

```cpp
#include <bits/stdc++.h>
using namespace std;
int main()
{
    int c, n, corte, atual, i, lim, azeitonas[20000];
    scanf("%d %d", &c, &n);
    lim = c / n;
    for(i = 0; i < n; i++)
        scanf("%d", &azeitonas[i]);
    for(corte = 0; corte <= lim; corte++) {
        atual = azeitonas[0] - corte;
        for(i = 0; i < n; i++) {
            if(!(azeitonas[i] >= atual && azeitonas[i] < atual + lim))
                goto a;
            atual += lim;
        }
        puts("S");
        return 0;
        a:;
    }
    puts("N");
}
```

1.12 line sweep k th fence fail

```cpp
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>

using namespace std;
using namespace __gnu_pbds;

#define D(x) //cout << #x << " = " << x << endl

typedef pair<int, int> Ponto;
typedef pair<Ponto, int> Flor;

typedef tree<
        pair<int, int>,
        null_type,
        less<pair<int, int>>,
        rb_tree_tag,
        tree_order_statistics_node_update>
        ordered_set;

int main() {
        vector<Flor> flores;
        vector<Ponto> cercas;

        int p, v, aux;
        long long resp;

        scanf("%d %d", &p, &v);

        ordered_set eventos;

        flores = vector<Flor>(p);
        cercas = vector<Ponto>(v);

        for(int i = 0; i < p; i++)
                scanf("%d %d", &flores[i].first.first, &flores[i].first.second),
                flores[i].second = i + 1;

        for(auto &it : cercas)
                scanf("%d %d", &it.first, &it.second);

        sort(flores.begin(), flores.end());
        sort(cercas.begin(), cercas.end());

        auto flor = flores.begin();
        auto cerca = cercas.begin();

        resp = 0;
        aux = 1;

        while(flor != flores.end())
        {
                if(cerca == cercas.end())
                {
                        resp += flor->second;

                        ++flor;
                }
                else if(flor->first < *cerca)
                {
                        auto lb = eventos.lower_bound({flor->first.second, 5000000});
                        auto order = eventos.order_of_key(*lb);

                        if(lb == eventos.end() || order % 2 == 0)
                                resp += flor->second;

                        ++flor;
                }
                else
                {
                    auto lb = eventos.lower_bound({cerca->second, 0});

                    if(lb != eventos.end() && lb->first == cerca->second)
                        eventos.erase(lb);
                    else
                            eventos.insert({cerca->second, aux++});

                        cerca++;
                }
        }

        printf("%lld\n", resp);
}
```

1.13 math integracao simpson

```java
import java.util.function.DoubleFunction;

public class SimpsonIntegration {

    public static double integrate(DoubleFunction<Double> f, double a, double b) {
        double eps = 1e-10;
        double m = (a + b) / 2;
        double am = simpsonIntegration(f, a, m);
        double mb = simpsonIntegration(f, m, b);
        double ab = simpsonIntegration(f, a, b);
        if (Math.abs(am + mb - ab) < eps)
            return ab;
        return integrate(f, a, m) + integrate(f, m, b);
    }

    static double simpsonIntegration(DoubleFunction<Double> f, double a, double b) {
        return (f.apply(a) + 4 * f.apply((a + b) / 2) + f.apply(b)) * (b - a) / 6;
    }

    // Usage example
    public static void main(String[] args) {
        System.out.println(integrate(x -> Math.sin(x), 0, Math.PI / 2));
    }
}
```

1.14 math pascal

```
from sys import stdin

valores = []

fatorial = [1]

for i in range(1, 33):
        fatorial.append(fatorial[-1] * i)

def calc(n, x, produto, anterior):
        if(n == 0):
                valores.append(fatorial[h - 1] // produto)
                return

        if(x == 0):
                return

        for i in range(0, min(anterior + 1, n + 1)):
                calc(n - i, x - 1, produto * fatorial[i], i)

for line in stdin:
        d, h = map(int, line.split(' '))

        calc(h - 1, d, 1, h)

        valores = list(set(valores))
        valores.sort()

        for x in valores:
                print(x)

        valores = []
```

1.15 meet in the middle caxeiro indoring

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

using namespace std;

#define ultimo combi.size()-1
#define kultimo pos-1

long long dist_total, l;
int n;
bool visitado_combinacao[20];
long long mat_adj[20][20];
int para1, para2;

bool achei;

unordered_map<int, unordered_map<long long, int>[15]> puto;

int vet[20];

int preCP[20];

int elevado() {
        preCP[0] = 1;

        for(int i=1; i<20; i++)
                preCP[i] = preCP[i-1] * 2;
}

void combinacao(int val, int pos) {
    if(n-val + pos < para1)
        return;

        if(pos == para1) {
                int jaSei = 0;
            for(int i=0; i<pos; i++)
                jaSei = jaSei | preCP[vet[i]];

            do {
                dist_total = mat_adj[0][vet[0]];
                for(int i=1; i<pos; i++)
                    dist_total += mat_adj[vet[i]][vet[i-1]];
                    puto[jaSei][vet[pos-1]][dist_total] = 1;
                } while(next_permutation(vet, vet+pos));

                return;
        }

        for(int i=val; i<n; i++) {
            vet[pos] = i;
            combinacao(i + 1, pos+1);
        }
}

void combinacao2(int val, int pos) {
    if(n-val + pos < para2)
        return;

        if(pos == para2) {
            int jaSei = 0;
            for(int i=1; i<n; i++) {
                if(!visitado_combinacao[i])
                        jaSei = jaSei | preCP[i];
            }

            do {
                dist_total = mat_adj[0][vet[0]];
                for(int i=1; i<pos; i++)
                    dist_total += mat_adj[vet[i]][vet[i-1]];

                    for(int it=1; it<n; it++) {
                        if(!visitado_combinacao[it])
                                        if (puto[jaSei][it].count(l-dist_total-mat_adj[it][vet[pos-1]]) != 0) {
                                                puts("possible");
                                                exit(0);
                                        }
                        }

                } while(next_permutation(vet, vet+pos));

                return;
        }

        for(int i=val; i<n; i++) {
            vet[pos] = i;
```

```
            visitado_combinacao[i] = true;
            combinacao2(i + 1, pos+1);
            visitado_combinacao[i] = false;
        }
}

int main() {
            elevado();
         cin >> n >> l;
            if(n%2 == 0) {
                para1 = n/2.0-1;
                    para2 = n/2.0;
            }
            else {
                para1 = n/2.0;
                    para2 = n/2.0;
            }
                achei = false;
                for(int i=0; i<n; i++)
                        for(int j=0; j<n; j++)
                                scanf("%lld", &mat_adj[i][j]);
                if(n == 2) {
                        if(mat_adj[0][1]*2 == l)
                                puts("possible");
                        else
                                puts("impossible");
                }
                else {
                combinacao(1, 0);
                fill(visitado_combinacao, visitado_combinacao+n+1, false);
                visitado_combinacao[0] = true;
                vet[0] = 0;
                combinacao2(1, 0);

                if(!achei)
                        puts("impossible");
                }
        return 0;
}
```

1.16 pd digitos digits counting

```
#include <bits/stdc++.h>
using namespace std;
struct SEQ {
        vector<int> qt;
        SEQ() {
                qt.assign(10,0);
        }
        SEQ operator - (const SEQ B) {
                SEQ novo;
                for(int i=0; i<=9; i++)
                        novo.qt[i] = qt[i] - B.qt[i];
                return novo;
        }
        SEQ operator + (const SEQ B) {
                SEQ novo;
                for(int i=0; i<=9; i++)
                        novo.qt[i] = qt[i] + B.qt[i];
                return novo;
        }
        void operator = (const SEQ B) {
                for(int i=0; i<=9; i++)
                        qt[i] = B.qt[i];
        }
};
int exp(int a) {
        int cont = 1;
        for(int i=0; i<a; i++)
                cont *= 10;
        return cont;
}
SEQ andre(int n, int tam) {
        if(tam == 0) {
                SEQ novo;
                novo.qt[0]++;
                return novo;
        }
        int pot = exp(tam-1);
        int pri = n/pot;
        int rest = n%pot;
        SEQ cont;
        cont.qt[pri] = rest+1;
        for(int i=pri-1; i>=0; i--)
                cont.qt[i] += pot;
        for(int i=0; i<=9; i++)
                cont.qt[i] += pri*(tam-1)*(pot/10);

        cont.qt[0] -= pot;
        return cont + andre(rest, tam-1);
}
int main() {
        int a, b, aux_a, aux_b;
        int qt_a, qt_b;
        while((scanf("%d%d", &a, &b) && a+b)) {
                aux_a = a-1;
                aux_b = b;
                qt_a = qt_b = 0;
                while(aux_a) {
                        qt_a++;
                        aux_a /= 10;
                }
                while(aux_b) {
                        qt_b++;
                        aux_b /= 10;
                }
                SEQ novo = andre(b, qt_b) - andre(a-1, qt_a);
```

```
                cout << novo.qt[0];
                for(int i=1; i<=9; i++)
                        cout << " " << novo.qt[i];
                cout << endl;
        }
        return 0;
}
```

1.17 pd garcom internet trouble

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

using namespace std;

long long custo[6010][6010], pd[6010][6010], valores[6100], acumulado[6100];
int k_opt[6010][6010] = {0}, b, n;

/*
        PD Original

    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= n; j++)
            for (int k = 0; k < i; k++)
                if (b + pd[k][j - 1] + custo[k + 1][i] < pd[i][j])
                {
                    pd[i][j] = b + pd[k][j - 1] + custo[k + 1][i];
                    k_opt[i][j] = k;
                }
*/

void resolve(int i, int j, int opt_l, int opt_r)
{
    k_opt[i][j] = opt_l;

    for (int k = opt_l; k <= opt_r; k++)
        if (b + pd[k][j - 1] + custo[k + 1][i] < pd[i][j])
        {
            pd[i][j] = b + pd[k][j - 1] + custo[k + 1][i];
            k_opt[i][j] = k;
        }
}

void calcula(int i, int l, int r, int opt_l, int opt_r)
{
    if (l > r)
        return;

    int m = (l + r) / 2;

    resolve(i, m, opt_l, opt_r);

    calcula(i, l, m - 1, opt_l, k_opt[i][m]);
    calcula(i, m + 1, r, k_opt[i][m], opt_r);
}

int main()
{
    int c;

    while (scanf("%d %d %d", &n, &b, &c) > 0)
    {
        for (int i = 0; i <= n; i++)
        {
            fill(pd[i], pd[i] + n + 1, 0ll);
            fill(custo[i], custo[i] + n + 1, 0ll);
        }

        for (int i = 0; i < n; i++)
            scanf("%lld", &valores[i]);

        acumulado[0] = 0;

        for (int i = 0; i < n; i++)
            acumulado[i + 1] = valores[i] + acumulado[i];

        for (int i = 0; i < n; i++)
        {
            pd[i][0] = valores[0] * i;
            pd[i][n - 1] = valores[n - 1] * (n - 1 - i);

            for (int j = 1; j < i; j++)
                pd[i][j] = pd[i][j - 1] + valores[j] * (i - j);
            for (int j = n - 2; j > i; j--)
                pd[i][j] = pd[i][j + 1] + valores[j] * (j - i);

            if (i < n - 1)
                pd[i][i] += pd[i][i + 1];
            if (i)
                pd[i][i] += pd[i][i - 1];
        }

        for (int i = 0; i < n; i++)
        {
            custo[i][i] = 0;
```

```
            for (int j = 0; j < i; j++)
            {
                auto val_mediana = (acumulado[i + 1] - acumulado[j]) / 2 + acumulado[j];

                auto mediana = distance(acumulado, upper_bound(acumulado, acumulado + n, val_mediana)) - 1;

                auto val = pd[mediana][mediana];

                if (i < n - 1)
                    val -= pd[mediana][i + 1];

                if (j)
                    val -= pd[mediana][j - 1];

                custo[i + 1][j + 1] = c * val;
            }
            for (int j = i + 1; j < n; j++)
            {
                auto val_mediana = (acumulado[j + 1] - acumulado[i]) / 2 + acumulado[i];

                auto mediana = distance(acumulado, upper_bound(acumulado, acumulado + n, val_mediana)) - 1;

                auto val = pd[mediana][mediana];

                if (j < n - 1)
                    val -= pd[mediana][j + 1];

                if (i)
                    val -= pd[mediana][i - 1];

                custo[i + 1][j + 1] = c * val;
            }
        }
        for (int i = 0; i <= n; i++)
            fill(pd[i], pd[i] + n + 1, 1ll << 60);

        pd[0][0] = 0;

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

        for (int j = 1; j <= n; j++)
            cout << pd[n][j] << " \n"[j == n];
    }
}
```

1.18 pd intervalos hiperatcive

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

using namespace std;

int m, n, s, f, a ,b;
vector<pair<int, int> > intervalos;
map<pair<int, int>, int> pd;

int pdzona(int ii, int ini) {
        if(intervalos[ii].first == m)
                return 1;

        auto h = make_pair(ii, ini);
        if(pd.count(h) != 0)
                return pd[h];

        int cont = 0;
        for(int j=ii+1; j<n; j++) {
                if(intervalos[j].first > intervalos[ii].first && intervalos[j].second > ini && intervalos[j].
                    second <= intervalos[ii].first) {
                        cont = (cont+pdzona(j, intervalos[ii].first))%100000000;
                }
        }

        return pd[h] = cont;
}

int main() {
        while(cin >> m >> n) {
                if(m+n == 0)
                        return 0;
                intervalos.clear();
                pd.clear();

                for(int i=0; i<n; i++) {
                        cin >> a >> b;
                        intervalos.push_back(make_pair(b, a));
                }

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

                int cont = 0;
                for(int i=0; i<n; i++) {
                        if(intervalos[i].second == 0)
                                cont = (cont+pdzona(i, 0))%100000000;
                }

                cout << cont << endl;
        }

        return 0;
}
```

1.19 pd math prob war quarta mineira

```
#include <stdio.h>
#include <math.h>

long int probPerder[4][4][4] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 15, 21, 0, 0, 55,
    161, 0, 0, 55, 1241, 0, 0, 0, 0, 0, 0, 125, 91, 0, 0, 370, 290, 636, 0, 370, 1250, 6156, 0, 0, 0, 0, 0,
    1025, 271, 0, 0, 4030, 2090, 1656, 0, 3123, 11104, 10663, 21766};
double probs[4][4][4] = {0}, chances[10002][10002] = {0};

inline int max(int a, int b) {
        return a > b ? a : b;
}

inline int min(int a, int b) {
        return a < b ? a : b;
}

inline double dmax(double a, double b) {
        return a > b ? a : b;
}

inline double dmin(double a, double b) {
        return a < b ? a : b;
}

double chance(int atk, int def) {
        if(def <= 0)
                return 1;
        if(atk <= 0)
                return 0;

        return chances[atk][def];
}

double calc(int atk, int def, int da, int dd) {
        int i, j, qtd = min(da, dd);
        double prob = 0;

        for(i = 0; i <= qtd; i++)
                prob += probs[da][dd][i] * chance(atk-i, def-qtd+i);

        return prob;
}

double minimo(int atk, int def, int da) {
        int i, j, qtd = min(def, 3);
        double r = 2;

        for(i = 1; i <= qtd; i++)
                r = dmin(r, calc(atk, def, da, i));

        return r;
}

double maximo(int atk, int def) {
        int i, j, qtd = min(atk, 3);
        double r = 0;

        for(i = 1; i <= qtd; i++)
                r = dmax(r, minimo(atk, def, i));

        return r;
}

int main() {
        int i, j, k, a, b;

        for(i = 1; i <= 3; i++)
                for(j = 1; j <= 3; j++)
                        for(k = 0; k < 4; k++)
                                probs[i][j][k] = probPerder[i][j][k]/(pow(6, i+j));

        scanf("%d %d", &a, &b);
        a--;

// min-max para atk <= 3 || def <= 3
        for(i = 1; i <= 50; i++)
                for(j = 1; j <= 3; j++) {
                        chances[i][j] = maximo(i, j);
                        chances[j][i] = maximo(j, i);
                }

        for(i = 51; i <= max(a, b); i++)
                chances[i][1] = chances[i][2] = chances[i][3] = 1;

// guloso para atk > 3 && def > 3
        for(i = 4; i <= a; i++)
```

```
                for(j = (a+b)%3-(i%3); j <= b; j+=3) {
                        if(j <= 3)
                                continue;

                        chances[i][j] = probs[3][3][0] * chances[i][j-3] + probs[3][3][1] * chances[i-1][j-2] +
                            probs[3][3][2] * chances[i-2][j-1] + probs[3][3][3] * chances[i-3][j];

                        if(chances[i][j] < 0.000001)
                                break;
                }
        printf("%.04lf\n", chances[a][b]);
}
```

1.20 pd minmax bottomup cartoes

```
#include <bits/stdc++.h>
#define D(x) cout << #x << " = " << x << endl;
using namespace std;
int main() {
        int n;
        long long cartoes[12000];
        long long A[12000];
        long long W[12000];
        while(cin >> n) {
                for(int i=0; i<n; i++) {
                        scanf("%lld", &cartoes[i]);
                        A[i] = W[i] = 0;
                }
                for(int i=1; i<n; i++) {
                        for(int j=0; j<n-i; j++) {
                                if(i%2 == 0)
                                        W[j] = min(A[j], A[j+1]);
                                else
                                        A[j] = max( (cartoes[j] + W[j+1]),
                                                    (cartoes[j+i] + W[j]));
                        }
                }
                cout << A[0] << endl;
        }
        return 0;
}
```

1.21 pd segtree keep it energized

```
#include <bits/stdc++.h>
using namespace std;
#define MAX 1000000 // O valor aqui tem que ser >= 2 * tamanho do maior n
#define D(x) cout << #x << " = " << x << endl
int init[MAX], tree[MAX], lazy[MAX];
typedef vector<int> vi;
struct Loja {
        int l, s, c;
};
void build_tree(int node, int a, int b)
{
        if (a > b)
                return;
        // Se folha
        if (a == b)
        {
                tree[node] = 1 << 30;
                return;
        }
        build_tree(node * 2, a, (a + b) / 2);
        build_tree(node * 2 + 1, 1 + (a + b) / 2, b);
        tree[node] = min(tree[node * 2], tree[node * 2 + 1]);
}
void update_tree(int node, int a, int b, int i, int j, int value)
{
        // Se fora do intervalo - retorna
        if (a > b || a > j || b < i)
                return;
        if (a >= i && b <= j)
        {
                tree[node] = value;
                return;
        }
        // Atualiza os filhos.
        update_tree(node * 2, a, (a + b) / 2, i, j, value);
        update_tree(1 + node * 2, 1 + (a + b) / 2, b, i, j, value);
        // Atualiza o pai.
        tree[node] = min(tree[node * 2], tree[node * 2 + 1]);
}
int query_tree(int node, int a, int b, int i, int j)
{
        // Se fora do intervalo
        if (a > b || a > j || b < i)
        {
                return 1<<30;
        }
        if (a >= i && b <= j)
                return tree[node];
        int q1 = query_tree(node * 2, a, (a + b) / 2, i, j);
        int q2 = query_tree(1 + node * 2, 1 + (a + b) / 2, b, i, j);
        return min(q1, q2);
}
int main() {
        int n, m;
        scanf("%d %d", &n, &m);
        build_tree(1, 0, n);
        auto custos = vi(n);
        auto acumulado = vi(n + 1);
        for(auto &it : custos)
                scanf("%d", &it);
        acumulado[n] = 0;
```

```
        for(int i = n - 1; i >= 0; i--)
                acumulado[i] = acumulado[i + 1] + custos[i];

        auto lojas =  vector<Loja>(m);
        auto pacotes = vector<vector<pair<int, int> > >(n);
        auto alcance = vi(m);
        auto count = 0;

        for(auto &it : lojas) {
                scanf("%d %d %d", &it.l, &it.s, &it.c);

                pacotes[it.l - 1].push_back({count, it.c});

                auto eu = lower_bound(acumulado.rbegin(), acumulado.rend(), acumulado[it.l - 1] - it.s);

                alcance[count++] = distance(eu, acumulado.rend()) - 1;
        }

        auto pd = vi(n + 1, 1 << 30);

        pd[n] = 0;

        update_tree(1, 0, n, n, n, 0);

        for(int i = n - 1; i >= 0; i--) {
                for(auto pacote : pacotes[i])
                        pd[i] = min(pd[i], query_tree(1, 0, n, i, alcance[pacote.first]) + pacote.second);

                update_tree(1, 0, n, i, i, pd[i]);
        }
        if(pd[0] != 1 << 30)
                cout << pd[0] << endl;
        else
                puts("-1");
}
```

1.22 pd string guardioes curiosos

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

using namespace std;

typedef vector<int> vl;
typedef vector<vl> vvl;

vvl pd;
int k, tam;

vvl pascal(101, vl(101, 0));

long long conta(int n, int m)
{
    if (n == 0)
        return 0;

    if (m == 1)
        return n;
    if (n == 1 && m < k || m == 0)
        return 1;

    if (pd[n][m] != -1)
        return pd[n][m];

    long long resp = 0;

    for (int i = 0; i < min(k, m + 1); i++)
        resp = (resp + (conta(n - 1, m - i) * pascal[m][i]) % 1000000007) % 1000000007;

    return pd[n][m] = resp;
}

int main()
{
    int i, j;

    pascal[0][0] = 1;

    for (i = 0; i < 101; i++)
        pascal[i][0] = 1;

    for (i = 1; i < 101; i++)
        for (j = 1; j < 101; j++)
            pascal[i][j] = (pascal[i - 1][j - 1] + pascal[i - 1][j]) % 1000000007;

    scanf("%d %d", &tam, &k);

    pd = vvl(tam + 1, vl(tam + 1, -1));

    printf("%d\n", conta(tam, tam - 2));
}
```

1.23 segtree acordes intergalaticos

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

using namespace std;

#define MAX 1000000 // O valor aqui tem que ser >= 4 * tamanho do maior n
#define ELEMENTO_NEUTRO 0

vector<int> freq(9);

struct No
{
    int F[9] = {0};

    No(int a)
    {
        F[a] = 1;
    }

    No() {}

    No operator+(const No &a) const
    {
        No novo;

        for (int i = 0; i < 9; i++)
            novo.F[i] = F[i] + a.F[i];

        return novo;
    }

    No operator=(const No &a)
    {
        for (int i = 0; i < 9; i++)
            this->F[i] = a.F[i];

        return *this;
    }

    void update(int max)
    {
        int temp[9];

        for (int i = 0; i < 9; i++)
            temp[i] = F[i];

        for (int i = 0; i < 9; i++)
            F[(i + max) % 9] = temp[i];
    }

    int val()
    {
        int max = 0;

        for (int i = 1; i < 9; i++)
            if (F[i] >= F[max])
                max = i;

        return max;
    }
};

int init[MAX], lazy[MAX];
No tree[MAX];

void build_tree(int node, int a, int b)
{
    if (a > b)
        return;

    if (a == b)
    {
        tree[node] = No(init[a]);
        lazy[node] = 0;
        return;
    }

    build_tree(node * 2, a, (a + b) / 2);
    build_tree(node * 2 + 1, 1 + (a + b) / 2, b);

    //Atualizacão do pai - verificar operacão
    tree[node] = tree[node * 2] + tree[node * 2 + 1];
    lazy[node] = 0;
}

void update_tree(int node, int a, int b, int i, int j, int val)
{
    //Atualizacão atrasada - verificar operacão
```

```
    if (lazy[node] != 0)
    {
        tree[node].update(lazy[node]);

        if (a != b)
        {
            lazy[node * 2] += lazy[node];
            lazy[node * 2 + 1] += lazy[node];
        }

        lazy[node] = 0;

        return;
    }

    if (a > b || a > j || b < i)
        return;

    //Atualizacão do nó - verificar operacão
    if (a >= i && b <= j)
    {
        tree[node].update(val);

        if (a != b)
        {
            lazy[node * 2] += val;
            lazy[node * 2 + 1] += val;
        }

        return;
    }

    update_tree(node * 2, a, (a + b) / 2, i, j, val);
    update_tree(1 + node * 2, 1 + (a + b) / 2, b, i, j, val);

    //Atualizacão do pai - verificar operacão
    tree[node] = tree[node * 2] + tree[node * 2 + 1];
}

int query_tree(int node, int a, int b, int i, int j)
{
    if (a > b || a > j || b < i)
    {
        return 0;
    }

    //Atualizacão atrasada - verificar operacão
    if (lazy[node] != 0)
    {
        tree[node].update(lazy[node]);

        if (a != b)
        {
            lazy[node * 2] += lazy[node];
            lazy[node * 2 + 1] += lazy[node];
        }

        lazy[node] = 0;
    }

    if (a == i && b == j && a == b)
        return tree[node].val();

    int q1 = query_tree(node * 2, a, (a + b) / 2, i, j);
    int q2 = query_tree(1 + node * 2, 1 + (a + b) / 2, b, i, j);

    //Retorno da arvore - verificar operacão
    return q1 + q2;
}

void most_freq(int node, int a, int b, int i, int j)
{
    if (a > b || a > j || b < i)
    {
        return;
    }

    //Atualizacão atrasada - verificar operacão
    if (lazy[node] != 0)
    {
        tree[node].update(lazy[node]);

        if (a != b)
        {
            lazy[node * 2] += lazy[node];
            lazy[node * 2 + 1] += lazy[node];
        }
```

```
        lazy[node] = 0;
    }

    if (a >= i && b <= j)
    {
        for (int i = 0; i < 9; i++)
            freq[i] += tree[node].F[i];

        return;
    }

    most_freq(node * 2, a, (a + b) / 2, i, j);
    most_freq(1 + node * 2, 1 + (a + b) / 2, b, i, j);
}

int main()
{
    int n, q, a, b;

    scanf("%d %d", &n, &q);

    fill(init, init + n, 1);

    build_tree(1, 0, n - 1);

    while (q--)
    {
        scanf("%d %d", &a, &b);

        int most = 0;

        for (auto &it : freq)
            it = 0;

        most_freq(1, 0, n - 1, a, b);

        for (int i = 1; i < 9; i++)
            if (freq[i] >= freq[most])
                most = i;

        update_tree(1, 0, n - 1, a, b, most);
    }

    for (int i = 0; i < n; i++)
        printf("%d\n", query_tree(1, 0, n - 1, i, i));
}
```

1.24 segtree homem elefante rato

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

using namespace std;

#define MAX 6000000

typedef struct tDado {
    int h, e, r;

    tDado operator++(int a) {
        int aux = e;
        e = h;
        h = r;
        r = aux;
    }
    tDado operator+(const tDado &a) {
        tDado b;
        b.h = h + a.h;
        b.e = e + a.e;
        b.r = r + a.r;
        return b;
    }
    tDado operator=(const tDado &a) {
        h = a.h;
        e = a.e;
        r = a.r;
        return *this;
    }
    tDado operator=(int a) {
        h = a == 0;
        e = a == 1;
        r = a == 2;
        return *this;
    }
} tDado;

tDado init[MAX], tree[MAX];
int lazy[MAX];

void build_tree(int node, int a, int b) {
    if(a > b)
        return;

    if(a == b) {
        tree[node] = init[a];
                lazy[node] = 0;
        return;
    }

    build_tree(node*2, a, (a+b)/2);
    build_tree(node*2+1, 1+(a+b)/2, b);

    tree[node] = tree[node*2] + tree[node*2+1];
        lazy[node] = 0;
}

void update_tree(int node, int a, int b, int i, int j, int value) {
    if(lazy[node] != 0) {
                for(int k = 0; k < lazy[node] % 3; k++)
                        tree[node]++;

                if(a != b) {
                        lazy[node*2] += lazy[node];
                lazy[node*2+1] += lazy[node];
                }

                lazy[node] = 0;
        }

        if(a > b || a > j || b < i)
        return;

        if(a >= i && b <= j) {
                tree[node]++;

                if(a != b) {
                        lazy[node*2]++;
                        lazy[node*2+1]++;
                }

                return;
        }

    if(a == b) {
        tree[node]++;
```

```
        return;
    }

    update_tree(node*2, a, (a+b)/2, i, j, value);
    update_tree(1+node*2, 1+(a+b)/2, b, i, j, value);

    tree[node] = tree[node*2] + tree[node*2+1];
}

tDado query_tree(int node, int a, int b, int i, int j) {

    if(a > b || a > j || b < i) {
        tDado a;

        a = -1;

        return a;
    }

        if(lazy[node] != 0) {
                for(int k = 0; k < lazy[node] % 3; k++)
                        tree[node]++;

                if(a != b) {
                        lazy[node*2] += lazy[node];
                lazy[node*2+1] += lazy[node];
                }

                lazy[node] = 0;
        }

    if(a >= i && b <= j)
        return tree[node];

    tDado q1 = query_tree(node*2, a, (a+b)/2, i, j);
    tDado q2 = query_tree(1+node*2, 1+(a+b)/2, b, i, j);

    return q1 + q2;
}

int main () {
    char op;
    int n, m, i, a, b;
    tDado resp;

    while(scanf("%d %d", &n, &m) > 0) {
        for(i = 0; i < n; i++)
            init[i] = 0;

        build_tree(1, 0, n-1);

        for(i = 0; i < m; i++) {
            scanf(" %c %d %d", &op, &a, &b);

            if(op == 'M')
                update_tree(1, 0, n-1, a-1, b-1, 0);
            else {
                resp = query_tree(1, 0, n-1, a-1, b-1);

                printf("%d %d %d\n", resp.h, resp.e, resp.r);
            }
        }

        printf("\n");
    }

    return 0;
}
```

1.25 simulacao mochilas nwerc

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

using namespace std;

long long mdc(long long a, long long b)
{
    long long remainder;
    while (b != 0)
    {
        remainder = a % b;
        a = b;
        b = remainder;
    }
    return a;
}

int main() {
        multiset<long long> original;

        long long n, s, t;
        long long aux;

        cin >> n >> s >> t;

        for(long long i=0; i<n; i++) {
                scanf("%lld", &aux);
                original.insert(aux);
        }

        long long melhor = 1LL<<60, pior = -1, media = 0;
        long long t_mod;

        for(auto it = original.begin(); it != original.end(); it++) {
                multiset<long long> esteira = original;

                long long tempo = *it;

                long long diferenca;
                auto quem = esteira.erase(esteira.lower_bound(tempo));
                auto fulano = esteira.begin();
                tempo += t;

                while(!esteira.empty()) {
                        t_mod = tempo%s;

                        quem = esteira.lower_bound(t_mod);
                        if(quem == esteira.end())
                                quem = esteira.begin();

                        if(*quem < t_mod)
                                tempo += s + *quem - t_mod;
                        else
                                tempo += *quem - t_mod;

                        esteira.erase(quem);
                        tempo += t;
                }

                tempo -= *it;
                melhor = min(melhor, tempo);

                if(it == original.begin()) {
                        auto asd = original.end();
                        asd--;
                        diferenca = *it + s - *asd;
                }
                        else {
                        fulano = it;
                        fulano--;
                        diferenca = *it - *fulano;
                }

                pior = max(pior, tempo + diferenca - 1);
                media += diferenca*tempo + diferenca*(diferenca-1)/2;
        }

        if(original.count(*original.begin()) == original.size()) {
                pior = melhor+s-1;
                media = s*melhor + s*(s-1)/2;
        }

        cout << melhor << "\n" << pior << endl;
```

```
        long long mmm = mdc(media, s);
        printf("%lld/%lld\n", media/mmm, s/mmm);
        return 0;
}
```

1.26 string aho crosahick

```
public class AhoCorasick {

        static final int ALPHABET_SIZE = 26;

        Node[] nodes;
        int nodeCount;

        public static class Node {
            int parent;
            char charFromParent;
            int suffLink = -1;
            int[] children = new int[ALPHABET_SIZE];
            int[] transitions = new int[ALPHABET_SIZE];
            boolean leaf;

            {
                Arrays.fill(children, -1);
                Arrays.fill(transitions, -1);
            }
        }

        public AhoCorasick(int maxNodes) {
            nodes = new Node[maxNodes];
            // create root
            nodes[0] = new Node();
            nodes[0].suffLink = 0;
            nodes[0].parent = -1;
            nodeCount = 1;
        }

        public void addString(String s) {
            int cur = 0;
            for (char ch : s.toCharArray()) {
                int c = ch - 'a';
                if (nodes[cur].children[c] == -1) {
                    nodes[nodeCount] = new Node();
                    nodes[nodeCount].parent = cur;
                    nodes[nodeCount].charFromParent = ch;
                    nodes[cur].children[c] = nodeCount++;
                }
                cur = nodes[cur].children[c];
            }
            nodes[cur].leaf = true;
        }

        public int suffLink(int nodeIndex) {
            Node node = nodes[nodeIndex];
            if (node.suffLink == -1)
                node.suffLink = node.parent == 0 ? 0 : transition(suffLink(node.parent), node.charFromParent);
            return node.suffLink;
        }

        public int transition(int nodeIndex, char ch) {
            int c = ch - 'a';
            Node node = nodes[nodeIndex];
            if (node.transitions[c] == -1)
                node.transitions[c] = node.children[c] != -1 ? node.children[c] : (nodeIndex == 0 ? 0 :
                    transition(suffLink(nodeIndex), ch));
            return node.transitions[c];
        }

        // Usage example
        public static void main(String[] args) {
            AhoCorasick ahoCorasick = new AhoCorasick(1000);
            ahoCorasick.addString("bc");
            ahoCorasick.addString("abc");

            String s = "tabcbc";
            int node = 0;
            List<Integer> positions = new ArrayList<>();
            for (int i = 0; i < s.length(); i++) {
                node = ahoCorasick.transition(node, s.charAt(i));
                if (ahoCorasick.nodes[node].leaf)
                    positions.add(i);
            }
            System.out.println(positions);
        }
    }
}
```

1.27 string matching game of matchings

```
s=raw_input()
m=int(raw_input())
p=map(int,raw_input().split())
n=len(s)
#befp[i]=max {j: j<i and p[j]=p[i]}
last=[-1]*27
befp=[-1]*m
for i in range(m):
    ch=p[i]
    befp[i]=last[ch]
    last[ch]=i
#befs[i]=max {j: j<i and s[j]=s[i]}
last=[-1]*27
befs=[-1]*n
for i in range(n):
    ch=ord(s[i])-ord('a')
    befs[i]=last[ch]
    last[ch]=i
#f[i]=failure function
f=[0]*m
idx=0
for i in range(1,m):
    match=False
    while idx>0 and not match:
        match=True
        if befp[i]!=-1 and idx>=(i-befp[i]):
            at=idx-(i-befp[i])
            if p[idx]!=p[at]:
                idx=f[idx-1]
                match=False
        if match and befp[idx]!=-1:
            at=i-(idx-befp[idx])
            if p[at]!=p[i]:
                idx=f[idx-1]
                match=False
    idx+=1
    f[i]=idx
#kmp
idx=0
ans=0
for i in range(n):
    match=False
    while idx>0 and not match:
        match=True
        if befs[i]!=-1 and idx>=(i-befs[i]):
            at=idx-(i-befs[i])
            if p[idx]!=p[at]:
                idx=f[idx-1]
                match=False
        if match and befp[idx]!=-1:
            at=i-(idx-befp[idx])
            if s[at]!=s[i]:
                idx=f[idx-1]
                match=False
    idx+=1
    if idx==m:
        #match at i-m+1
        ans+=1
        idx=f[m-1]
print ans
```

1.28 string quebrar em palindromos russa

```
#include <bits/stdc++.h>
using namespace std;
typedef vector<int> vi;
#define D(x) cout << #x << " = " << x << endl
int main() {
	ios::sync_with_stdio(false);
	int n;
	string entrada;
	cin >> n;
	cin >> entrada;
	vi apareceu(255);
	for(auto &it : entrada)
		apareceu[it]++;
	int impares = 0, palindromos = 1, tamanho;
	vi usaveis;
	for(int i = 0; i < 255; i++)
		if(apareceu[i] % 2)
			usaveis.push_back(i), impares++, apareceu[i]--;
	palindromos = max(impares, palindromos);
	if(palindromos > 1)
		while(palindromos < entrada.size()) {
			if((entrada.size() % palindromos) == 0 && (entrada.size() / palindromos) % 2 == 1)
				break;
			else
				palindromos += 2;
		}
	if(entrada.size() % palindromos)
		palindromos = entrada.size();
	cout << palindromos << endl;
	tamanho = entrada.size() / palindromos;
	int j = 0;
	for(int i = 0; i < palindromos; i++) {
		string resposta;
		if(tamanho % 2) {
			if(usaveis.size())
				resposta = usaveis.back(), usaveis.pop_back();
			else {
				for(int k = 0; k < 255; k++)
					if(apareceu[k]) {
						usaveis.push_back(k);
						resposta = k;
						apareceu[k] -= 2;
						break;
					}
			}
		}
		for(; j < 255; j++) {
			while(apareceu[j]) {
				resposta += ((char) j);
				apareceu[j] -= 2;
				if(tamanho <= resposta.size() * 2)
					break;
			}
			if(tamanho <= resposta.size() * 2)
				break;
		}
		reverse(resposta.begin(), resposta.end());
		cout << resposta;
		if(tamanho % 2)
```

```
                        resposta.erase(resposta.end() - 1, resposta.end());

                reverse(resposta.begin(), resposta.end());
                cout << resposta;

                cout << " \n"[i == palindromos-1];
        }
}
```

1.29 string trie dicionario portunhol

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

using namespace std;

long long sss1, sss2;
long long tot_pt[256];
long long tot_es[256];

struct No
{
    char c;
    vector<No> filhos;
};

void conta_pt(No &atual, long long qt) {
    if(atual.filhos.size() == 0) {
        sss1 += qt;
        return;
    }

    bool soma = true;

    for(auto &it : atual.filhos) {
        if(soma)
            conta_pt(it, qt+1);
        else
            conta_pt(it, 1);

        soma = false;
    }
}
void conta_es(No &atual, long long qt) {
    if(atual.filhos.size() == 0) {
        sss2 += qt;
        return;
    }

    bool soma = true;

    for(auto &it : atual.filhos) {
        if(soma)
            conta_es(it, qt+1);
        else
            conta_es(it, 1);

        soma = false;
    }
}
void total_pt(No &atual, bool faz) {
    if(faz)
        tot_pt[atual.c]++;

    for(auto &it : atual.filhos)
       total_pt(it, true);
}
void total_es(No &atual, bool faz) {
    if(faz)
        tot_es[atual.c]++;

    for(auto &it : atual.filhos)
       total_es(it, true);
}
int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);

    int n, i, m;
    string pal;

    No pt;
    No es;

    pt.c = es.c = '-';

    // Lê n palavras e monta a arvore de todas as palavras, adicionando $ a cada fim de palavra
    while(cin >> n >> m)
    {
        if(n+m == 0)
            return 0;
```

```
        sss1 = sss2 = 0;

        fill(tot_es + 'a', tot_es + 'z' + 1, 0);
        fill(tot_pt + 'a', tot_pt + 'z' + 1, 0);

        pt.filhos.clear();
        es.filhos.clear();

        No *atual;

        for(i = 0; i < n; i++)
        {
            atual = &pt;

            cin >> pal;

            for(auto &it : pal)
            {
                for(auto &it2 : atual->filhos) {
                    if(it2.c == it) {
                        atual = &it2;
                        goto prox1;
                    }
                }

                atual->filhos.push_back(No());
                atual = &(atual->filhos.back());
                atual->c = it;

                prox1:;
            }
        }

        for(i = 0; i < m; i++)
        {
            atual = &es;

            cin >> pal;

            for(int j=pal.size()-1; j >= 0; j--)
            {
                for(auto &it2 : atual->filhos) {
                    if(it2.c == pal[j]) {
                        atual = &it2;
                        goto prox2;
                    }
                }

                atual->filhos.push_back(No());
                atual = &(atual->filhos.back());
                atual->c = pal[j];

                prox2:;
            }
        }

        for(auto it : pt.filhos) {
            conta_pt(it, 1);
            total_pt(it, false);
        }

        for(auto it : es.filhos) {
            conta_es(it, 1);
            total_es(it, false);
        }

        long long soma_tot = 0;

        for(int j = 'a'; j <= 'z'; j++)
            soma_tot -= tot_pt[j] * tot_es[j];

        soma_tot += sss1 * sss2;

        cout << soma_tot << '\n';
    }
}
```

1.30 suffix array growing strings

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

using namespace std;

#define PB push_back

typedef vector<int> vi;

const int MAX = 1005000;

bool comp(string a, string b)
{
    return a.size() < b.size();
}
vi txt;                  //input
int iSA[MAX], SA[MAX];   //output
int cnt[MAX], prox[MAX]; //internal
bool bh[MAX], b2h[MAX];

// Compares two suffixes according to their first characters
bool smaller_first_char(int a, int b)
{
    return txt[a] < txt[b];
}

void suffixSort(int n)
{
    for (int i = 0; i < n; ++i)
        SA[i] = i;

    sort(SA, SA + n, smaller_first_char);

    for (int i = 0; i < n; ++i)
    {
        bh[i] = i == 0 || txt[SA[i]] != txt[SA[i - 1]];
        b2h[i] = false;
    }

    for (int h = 1; h < n; h <<= 1)
    {
        int buckets = 0;

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

            while (j < n && !bh[j])
                j++;

            prox[i] = j;
            buckets++;
        }

        if (buckets == n)
            break;

        for (int i = 0; i < n; i = prox[i])
        {
            cnt[i] = 0;

            for (int j = i; j < prox[i]; ++j)
                iSA[SA[j]] = i;
        }

        cnt[iSA[n - h]]++;
        b2h[iSA[n - h]] = true;

        for (int i = 0; i < n; i = prox[i])
        {
            for (int j = i; j < prox[i]; ++j)
            {
                int s = SA[j] - h;

                if (s >= 0)
                {
                    int head = iSA[s];
                    iSA[s] = head + cnt[head]++;
                    b2h[iSA[s]] = true;
                }
            }
            for (int j = i; j < prox[i]; ++j)
            {
                int s = SA[j] - h;
```

```
                if (s >= 0 && b2h[iSA[s]])
                    for (int k = iSA[s] + 1; !bh[k] && b2h[k]; k++)
                        b2h[k] = false;
            }
        }
        for (int i = 0; i < n; ++i)
        {
            SA[iSA[i]] = i;
            bh[i] |= b2h[i];
        }
    }
    for (int i = 0; i < n; ++i)
    {
        iSA[SA[i]] = i;
    }
}

int lcp[MAX];
void getlcp(int n)
{
    for (int i = 0; i < n; ++i)
        iSA[SA[i]] = i;

    lcp[0] = 0;

    for (int i = 0, h = 0; i < n; ++i)
    {
        if (iSA[i] > 0)
        {
            int j = SA[iSA[i] - 1];

            while (i + h < n && j + h < n && txt[i + h] == txt[j + h])
                h++;

            lcp[iSA[i]] = h;

            if (h > 0)
                h--;
        }
    }
}

int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);

    int n, resp;

    while (cin >> n && n)
    {
        txt.clear();

        map<int, int> posicoesIniciais, posicoesFinais, aVisitar;
        vi pd(n, 1);
        vector<string> palavras(n);
        resp = 0;

        for (auto &it : palavras)
            cin >> it;

        sort(palavras.begin(), palavras.end(), comp);

        for (int i = 0; i < n; i++)
        {
            posicoesIniciais[txt.size()] = i;

            for (auto &it : palavras[i])
                txt.PB(it + 20000);

            // Fim de palavra (menor que todas as letras e crescente)
            // pra ordernar certo;
            txt.PB(1 + i);

            posicoesFinais[txt.size()] = i;
        }

        suffixSort(txt.size());

        getlcp(txt.size());

        for (int i = 0; i < txt.size() - 1; i++)
        {
            auto inicial = posicoesIniciais.find(SA[i]);

            if (inicial != posicoesIniciais.end() && lcp[i + 1] == palavras[inicial->second].size())
                aVisitar[inicial->second] = i;
```

```
        }
        for (auto &it : aVisitar)
        {
            for (int j = it.second + 1; j < txt.size(); j++)
                if (lcp[j] < palavras[it.first].size())
                    break;
                else
                {
                    auto atual = posicoesFinais.upper_bound(SA[j])->second;
                    pd[atual] = max(pd[atual], pd[it.first] + 1);
                }
        }
        for (auto &it : pd)
            resp = max(resp, it);
        cout << resp << "\n";
    }
}
```

1.31 trie cellphone typing

```
#include <bits/stdc++.h>
using namespace std;
struct No {
        map<char, No> filhos;
};
int dfs(No &atual, int nivel, char v) {
        int resp = 0;
        for(auto &it : atual.filhos) {
                if(it.first == '$')
                        resp += nivel;
                resp += dfs(it.second, nivel + (atual.filhos.size() > 1), it.first);
        }
        return resp;
}
int main() {
        ios::sync_with_stdio(false);
        cin.tie(0);
        int n, i;
        string pal;
        while(cin >> n) {
                No inicio, *atual;
                inicio.filhos['-'] = No();
                for(i = 0; i < n; i++)
                {
                        atual = &inicio;
                        cin >> pal;
                        for(auto &it : pal) {
                                if(atual->filhos.count(it))
                                        atual = &(atual->filhos[it]);
                                else {
                                        atual->filhos[it] = No();
                                        atual = &(atual->filhos[it]);
                                }
                        }
                        atual->filhos['$'] = No();
                }
                printf("%.2lf\n", dfs(inicio, 0, '-') * 1.0 / n);
        }
}
```