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 eliminacao gauss xor loteria

```
#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");
}
```

1.3 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);
}
```

1.4 line sweep k th fence fail

```
#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);
}
```

1.5 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;
}
```

1.6 pd garcom internet trouble

```
#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];
    }
}
```

1.7 pd intervalos hiperatcive

```
#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;
}
```

1.8 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;
}
```

1.9 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");
}
```

1.10 pd string guardioes curiosos

```
#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));
}
```

1.11 pd string palindromo

```
#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 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));
}
```

1.13 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;
}
```

1.14 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";
    }
}
```

1.15 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);
        }
}
```