



Notebook - Maratona de Programação

DSUm balão da cor sim cor não

Contents

1 Algoritmos	2	5.6 Bfs	12
1.1 Busca Binaria	2	5.7 Bridges	12
1.2 Busca Binaria Double	2	5.8 Dfs	12
1.3 Busca Ternaria	2	5.9 Dfs Tree	13
1.4 Delta	2	5.10 Dijkstra	13
1.5 Fast Exponentiation	3	5.11 Euler Path	13
1.6 Psum	3	5.12 Floyd Warshall	14
1.7 Psum2d	3	5.13 Kosaraju	14
2 DP	3	5.14 Topo Sort	15
2.1 Convex Hull Opt	3	6 Math	15
2.2 Dp	4	6.1 Combinatoria	15
2.3 Knapsack	4	6.2 Dec To Bin	15
2.4 Lis	5	6.3 Divisibilidade	15
2.5 Mochila Iterativa	5	6.4 Divisores	15
2.6 Mochila Recursiva	5	6.5 Fatora	16
3 ED	6	6.6 Mdc	16
3.1 Bitwise	6	6.7 Mmc	16
3.2 Delayed	6	6.8 Pa	16
3.3 Dsu	6	6.9 Pg	16
3.4 Merge Sort	6	6.10 Pollard-rho	16
3.5 Mo	7	6.11 Primos	17
3.6 Ordered Set	7	7 Strings	17
3.7 Segtree	7	7.1 Suffix Array	17
3.8 Sqrt Decomposition	8	7.2 Trie	17
3.9 Xortrie	8	7.3 Zfunction	18
4 Geometria	9	8 Template	18
4.1 Geometria	9	8.1 Template	18
5 Grafos	9	9 zExtra	19
5.1 Binary Lifting	9	9.1 Formulasmat	19
5.2 Diametro	10	9.2 Getline	19
5.3 Kruskall	10		
5.4 Lca	11		
5.5 Bellman Ford	11		

1 Algoritmos

1.1 Busca Binaria

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 bool check(int valor, int x) {
5     return valor <= x;
6 }
7
8 int bb(int a, int b, int x){
9     int l = a;
10    int r = b;
11    while (l < r) {
12        int mid = (l + r) / 2;
13        if (check(mid, x)) r = mid;
14        else l = mid + 1;
15    }
16    return l;
17 }
18
19 bool check(int valor) {
20     return valor <= 10;
21 }
22
23 int bb_menor(int a, int b){
24     int l = a;
25     int r = b;
26     while (l < r) {
27         int mid = (l + r) / 2;
28         if (check(mid)) r = mid;
29         else l = mid + 1;
30     }
31
32     return l;
33 }
34
35
36 int bb_maior(int a, int b){
37     int l = a;
38     int r = b;
39     while (l < r) {
40         int mid = (l + r) / 2;
41         if (!check(mid)) r = mid;
42         else l = mid + 1;
43     }
44 }
```

1.2 Busca Binaria Double

```
1 //
2 // Complexidade : O(NlogN)
3
4 #include <bits/stdc++.h>
5 using namespace std;
6
7 typedef long long ll;
8 typedef long double ld;
9 const ld EPS = 1e-9;
10
11 ll check(ld x, vector<int> &v){
12     ll sum = 0;
13     for(int i=0; i<n; i++){
14         sum += (v[i]/x);
15     }
16     return sum;
17 }
18
19 int main(){
20     int n, k;
21     cin>>n>>k;
```

```
22     vector<int> v(n);
23     for(int i=0; i<n; i++)cin>>v[i];
24
25     ld l=0.0000000, r=10000000.0000000;
26     ld mid;
27     while(r-l>EPS){
28         mid = (ld)((l + r)/2);
29         if (check(mid, v)>=k){
30             l=mid;
31         }
32         else{
33             r = mid;
34         }
35     }
36     cout<<fixed<<setprecision(7)<<mid<<endl;
37
38     return 0;
39 }
```

1.3 Busca Ternaria

```
1 // Uma busca em uma curva, avaliando dois pontos
   diferentes
2 // Complexidade: O(Nlog3N)
3
4 double check(vector<int> v, vector<int> t, double x){
5     double ans = 0;
6     for(int i=0; i<v.size(); i++){
7         ans = max(ans, (double)(abs(v[i]-x) + t[i]));
8     }
9     return ans;
10 }
11
12 int32_t main(){ sws;
13
14     int t; cin>>t;
15     while(t--){
16         int n; cin>>n;
17         vector<int> v(n);
18         vector<int> t(n);
19         input(v);
20         input(t);
21
22         double ans = 0.0;
23         double l=0.0, r=1e9;
24         while(r-l >= EPS){
25
26             double mid1 = (double) l + (r - l) / 3;
27             double mid2 = (double) r - (r - l) / 3;
28
29             double x1 = check(v, t, mid1);
30             double x2 = check(v, t, mid2);
31
32             if(x1 < x2){
33                 r = mid2;
34             }else{
35                 l = mid1;
36                 ans = l;
37             }
38         }
39         cout<<fixed<<setprecision(7);
40         cout<<ans<<endl;
41     }
42     return 0;
43 }
```

1.4 Delta

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 int main(){
```

```

5     int n, q;
6     cin >> n >> q;
7     vector<int> v(n,0);
8     vector<int> delta(n+2, 0);
9
10    while(q--){
11        int l, r, x;
12        cin >> l >> r >> x;
13        delta[l] += x;
14        delta[r+1] -= x;
15    }
16
17    int atual = 0;
18    for(int i=0; i < n; i++){
19        atual += delta[i];
20        v[i] = atual;
21    }
22
23    for(int i=0; i < n; i++){
24        cout << v[i] << " ";
25    }
26    cout << endl;
27
28    return 0;
29 }

```

1.5 Fast Exponentiation

```

1 // recursivo
2 int fast_exp(int base, int e, int m){
3     if(!e) return 1;
4     int ans = fast_exp(base * base % m, e/2, m);
5     if(e % 2) return base * ans % m;
6     else return ans;
7 }
8 //iterativo
9 int fast_exp(int base, int e, int m) {
10    int ret = 1;
11    while (e) {
12        if (e & 1) ret = (ret * base) % m;
13        e >>= 1;
14        base = (base * base) % m;
15    }
16    return ret;
17 }

```

1.6 Psum

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 #define input(x) for (auto &it : x) cin >> it
5 typedef long long ll;
6 vector<ll> psum(1e5);
7
8 int solve(int l, int r){
9     if(l==0) return psum[r];
10    else return psum[r] - psum[l-1];
11 }
12
13 int main(){
14
15     int n, q;
16     cin>>n>>q;
17
18     vector<int> v(n);
19     input(v);
20     for(int i=0; i<n; i++){
21         if(i==0) psum[i] = v[i];
22         else psum[i] = psum[i-1] + v[i];
23     }
24     while(q--){

```

```

25         int l, r;
26         cin>>l>>r;
27
28         cout<<(solve(l,r))<<endl;
29     }
30
31     return 0;
32 }

```

1.7 Psum2d

```

1 int psum[MAX][MAX];
2
3 int32_t main(){ sws;
4     int t; cin>>t;
5     while(t--){
6         memset(psum, 0, sizeof(psum));
7         int n, q; cin>>n>>q;
8
9         for(int i=0; i<n; i++){
10            int x, y;
11            cin>>x>>y;
12
13            psum[x][y] += x*y;
14        }
15
16        for(int i=1; i<MAX; i++){
17            for(int j=1; j<MAX; j++){
18                psum[i][j] += psum[i-1][j];
19            }
20
21            for(int i=1; i<MAX; i++){
22                for(int j=1; j<MAX; j++){
23                    psum[i][j] += psum[i][j-1];
24                }
25            }
26
27            for(int i=0; i<q; i++){
28                int x1, y1, x2, y2;
29                cin>>x1>>y1>>x2>>y2;
30                x2--; y2--;
31
32                int soma = psum[x1][y1] + psum[x2][y2] -
33                    psum[x2][y1] - psum[x1][y2];
34                cout<<soma<<endl;
35            }
36        }
37    }
38    return 0;
39 }

```

2 DP

2.1 Convex Hull Opt

```

1 // utiliza-se convexhull tricky geralmente para dp 0(
2 // 2n), onde para cada elemento, percorre os
3 // elementos anteriores à ele.
4 // o objetivo é iterar pelo j e transformar o i em
5 // constante para criar retas e assim, encontrar o
6 // max. ou min.
7 // convex foi feito para achar o max, caso queira o
8 // min. troque o sinal de todos os j's
9 // reta ax + b, onde x é em função de i. Transforma em
10 // um for ós, onde os i's são atribuídas em dp[i]
11 // e soma-se à ela o cht.eval(x da reta)
12 // logo depois, faz cht.insert_line(a da reta, b da
13 // reta)
14
15 // algoritmo
16 const ll is_query = -(1LL<<62);
17 struct line {
18     ll m, b;

```

```

11 mutable function<const line*> succ;
12 bool operator<(const line& rhs) const {
13     if (rhs.b != is_query) return m < rhs.m;
14     const line* s = succ();
15     if (!s) return 0;
16     ll x = rhs.m;
17     return b - s->b < (s->m - m) * x;
18 }
19 };
20
21 struct dynamic_hull : public multiset<line> { // will
    maintain upper hull for maximum
22     const ll inf = LLONG_MAX;
23     bool bad(iterator y) {
24         auto z = next(y);
25         if (y == begin()) {
26             if (z == end()) return 0;
27             return y->m == z->m && y->b <= z->b;
28         }
29         auto x = prev(y);
30         if (z == end()) return y->m == x->m && y->b
<= x->b;
31
32         /* compare two lines by slope, make sure
    denominator is not 0 */
33         ll v1 = (x->b - y->b);
34         if (y->m == x->m) v1 = x->b > y->b ? inf : -
inf;
35         else v1 /= (y->m - x->m);
36         ll v2 = (y->b - z->b);
37         if (z->m == y->m) v2 = y->b > z->b ? inf : -
inf;
38         else v2 /= (z->m - y->m);
39         return v1 >= v2;
40     }
41     void insert_line(ll m, ll b) {
42         auto y = insert({ m, b });
43         y->succ = [=] { return next(y) == end() ? 0 :
&*next(y); };
44         if (bad(y)) { erase(y); return; }
45         while (next(y) != end() && bad(next(y)))
erase(next(y));
46         while (y != begin() && bad(prev(y))) erase(
prev(y));
47     }
48     ll eval(ll x) {
49         auto l = *lower_bound((line) { x, is_query })
;
50         return l.m * x + l.b;
51     }
52 };
53
54 // antes do convex
55 vll dp(n+1, LLINF);
56 for(int i=1; i<=n; i++){
57     ll x, a, b; tie(x, a, b) = v[i];
58     ll ans = LLINF; dp[i] = x*b + a;
59     for(int j=i-1; j>=1; j--){
60         ll x_bef, a_bef, b_bef; tie(x_bef, a_bef,
b_bef) = v[j];
61         ll val = -x_bef * b;
62         ans = min(ans, val + dp[j]);
63     }
64     dp[i] = min(dp[i], ans + x*b + a);
65 }
66
67 return 0;
68 }
69
70 // depois do convex
71 cht.insert_line(0, 0); // primeiro valor (no caso

```

dessa questao exemplo eh 0, 0)

```

74
75 for(int i=1; i<=n; i++){
76     ll x, a, b; tie(x, a, b) = v[i];
77     dp[i] = x * b + a - cht.eval(b);
78     cht.insert_line(x, -dp[i]);
79 }

```

2.2 Dp

```

1 // DP - Dynamic Programming
2
3 #include <bits/stdc++.h>
4 using namespace std;
5
6 typedef long long ll;
7 const int MAX = 110;
8
9 int n;
10 int tab[MAX];
11 vector<int> v;
12
13 ll dp(int i){
14     if(i>=n) return 0;
15     if(tab[i] != -1) return tab[i];
16
17     int pega = v[i] + dp(i+2);
18     int npega = dp(i+1);
19
20     tab[i] = max(pega, npega);
21     return tab[i];
22 }
23
24 int main(){
25     memset(tab, -1, sizeof(tab));
26     cin>>n;
27
28     v.assign(n, 0);
29
30     cout<<dp(0)<<endl;
31
32     return 0;
33 }

```

2.3 Knapsack

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 #define int long long
5 #define ll long long
6 #define sws ios::sync_with_stdio(false);cin.tie( NULL
);cout.tie(NULL);
7 #define pb(x) push_back(x);
8 #define pii pair<int,int>
9 const int N = 1e3+5;
10
11 int n, t;
12 int tab[N][N];
13 bool pegou[N][N];
14 vector<pair<int,int>> v;
15
16 vector<int> resposta;
17
18 int dp(int idx, int dias){
19     if(idx >= n) return 0;
20     if(tab[idx][dias] != -1) return tab[idx][dias];
21
22     int pega=0;
23     if(dias+v[idx].first <= t){
24         pega = dp(idx+1, dias+v[idx].first)+v[idx].
second;

```

```

25     }
26
27     int npega = dp(idx+1, dias);
28
29     if(pegas > npega) pegou[idx][dias] = true;
30
31     return tab[idx][dias] = max(pegas, npega);
32 }
33
34 int32_t main(){
35     memset(tab, -1, sizeof(tab));
36     cin >> n >> t;
37     for(int i=0; i<n; i++){
38         int ti, di;
39         cin >> ti >> di;
40
41         v.push_back({ti, di});
42     }
43     dp(0, 0);
44     int i = 0, j = 0;
45     vector<int> ans;
46     // retornar os valores
47     while(i < n){
48         if(pegou[i][j]){
49             j += v[i].first;
50             ans.push_back(i+1);
51         }
52         i++;
53     }
54     cout << ans.size() << endl;
55     for(int i=0; i<ans.size(); i++){
56         cout << ans[i] << " ";
57     }
58 }
59 }

```

2.4 Lis

```

1 // Longest increase sequence
2 // O(n log n)
3 multiset<int> S;
4 for(int i=0; i<n; i++){
5     auto it = S.upper_bound(vet[i]); // upper -
6     // longest strictly increase sequence
7     if(it != S.end())
8         S.erase(it);
9     S.insert(vet[i]);
10 }
11 // size of the lis
12 int ans = S.size();
13
14 // return the elements in LIS
15 // see that later
16 // https://codeforces.com/blog/entry/13225?comment=180208
17
18 vi LIS(const vi &elements){
19     auto compare = [&](int x, int y) {
20         return elements[x] < elements[y];
21     };
22     set<int, decltype(compare)> S(compare);
23
24     vi previous(elements.size(), -1);
25     for(int i=0; i<int(elements.size()); ++i){
26         auto it = S.insert(i).first;
27         if(it != S.begin())
28             previous[i] = *prev(it);
29         if(*it == i and next(it) != S.end())
30             S.erase(next(it));
31     }
32
33     vi answer;
34     answer.push_back(*S.rbegin());
35 }

```

```

34 while (previous[answer.back()] != -1 )
35     answer.push_back( previous[answer.back()] );
36 reverse( answer.begin(), answer.end() );
37 return answer;
38 }

```

2.5 Mochila Iterativa

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 const int maxn = 110, maxp = 1e5+10;
5 const long long inf = 0x3f3f3f3f3f3f3f3f; // ~= 10^18
6
7 int v[maxn], p[maxn];
8 long long dp[maxn][maxp];
9
10 int main() {
11     int n, C; scanf("%d %d", &n, &C);
12     for(int i = 1; i <= n; i++)
13         scanf("%d %d", &p[i], &v[i]);
14
15     long long ans = 0;
16     // inicializando o vetor
17     for(int i = 1; i <= n; i++)
18         for(int P = p[i]; P <= C; P++)
19             dp[i][P] = -inf;
20     // definindo o caso base
21     dp[0][0] = 0;
22
23     for(int i = 1; i <= n; i++) {
24         for(int P = 0; P <= C; P++) {
25             dp[i][P] = dp[i-1][P];
26             if(P >= p[i])
27                 dp[i][P] = max(dp[i][P], dp[i-1][P-p[i]] + v[i]);
28             ans = max(ans, dp[i][P]);
29         }
30     }
31
32     printf("%lld\n", ans);
33 }

```

2.6 Mochila Recursiva

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 const int maxn = 110, maxp = 1e5+10;
5
6 int v[maxn], p[maxn], n;
7 long long dp[maxn][maxp];
8 bool vis[maxn][maxp];
9
10 long long solve(int i, int P) {
11     if(i == n+1) return 0; // caso base, nao ha mais
12     // itens para se considerar
13     if(vis[i][P]) return dp[i][P];
14     vis[i][P] = 1;
15
16     // primeira possibilidade, nao adicionar o
17     // elemento
18     dp[i][P] = solve(i+1, P);
19
20     // segunda possibilidade, adicionar o elemento.
21     // Lembrar de tirar o maximo com o valor ja
22     // calculado da primeira possibilidade
23     if(P >= p[i])
24         dp[i][P] = max(dp[i][P], solve(i+1, P - p[i]) + v[i]);
25
26     return dp[i][P];
27 }

```

```

24 }
25
26 int main() {
27     int C; scanf("%d %d", &n, &C);
28     for(int i = 1; i <= n; i++)
29         scanf("%d %d", &p[i], &v[i]);
30     printf("%lld\n", solve(1, C));
31 }

```

3 ED

3.1 Bitwise

```

1 // Bitwise Operations
2
3 #include <bits/stdc++.h>
4 using namespace std;
5
6
7 // Verificar se o bit esta ligado
8 bool isSet(int bitPosition, int number) {
9     bool ret = ((number & (1 << bitPosition)) != 0);
10    return ret;
11 }
12
13 // Ligar o bit
14 bool setBit(int bitPosition, int number) {
15     return (number | (1 << bitPosition));
16 }
17
18 // Gerando todos os subconjuntos de um conjunto em
    binario
19 void possibleSubsets(char S[], int N) {
20     for(int i = 0; i < (1 << N); ++i) { // i = [0, 2^N - 1]
21         for(int j = 0; j < N; ++j)
22             if(i & (1 << j)) // se o j-esimo bit de
                i esta setado, printamos S[j]
23                 cout << S[j] << " ";
24         cout << endl;
25     }
26 }
27
28 // x & (~x+1) -> first set bit

```

3.2 Delayed

```

1 // adiciona elementos em um multiset, e calcula o
    numero de elementos menor que x no set
2 // O(raiz(QlogQ))
3
4 class Delayed{
5     ll q;
6     vector<ll> a, delayed;
7 public:
8     void merge(){
9         for(auto x : delayed){
10             a.pb(x);
11         }
12         sort(all(a));
13         delayed = {};
14     }
15
16     void add(ll x){
17         delayed.pb(x);
18         if(delayed.size() * delayed.size() > q){
19             merge();
20         }
21     }
22
23     ll get(ll x){

```

```

24         ll ans = 0;
25         ll pos = lower_bound(a.begin(), a.end(), x) -
            a.begin();
26         if(!pos){ans = 0;} else{ans = pos;}
27         for(auto it: delayed){
28             if(it < x){ans++;}
29         }
30         return ans;
31     }
32
33     Delayed(ll q){
34         this->q = q;
35     }
36 };

```

3.3 Dsu

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 // Complexidade
5 // build : O(N)
6 // find : O(logN)
7 class DSU{
8     vector<int> parent, sz;
9 public:
10    void make(int v){
11        parent[v] = v;
12        sz[v] = 1;
13    }
14
15    int find(int v){
16        if (v == parent[v]) return v;
17        return parent[v] = find(parent[v]);
18    }
19
20    void union_(int a, int b){
21        a = find(a), b = find(b);
22
23        if(sz[b]>sz[a]) swap(a,b);
24        if (a != b){
25            sz[a] += sz[b];
26            parent[b] = a;
27        }
28    }
29
30    bool same(int a, int b){
31        a = find(a), b = find(b);
32        return a == b;
33    }
34
35    DSU(int n): parent(n+1), sz(n+1){
36        for(int i=1; i<=n; i++) make(i);
37    }
38 };
39
40
41 int main(){
42     DSU dsu(10);
43     return 0;
44 }

```

3.4 Merge Sort

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 #define INF 1000000000
5
6 void merge_sort(vector<int> &v){
7     if(v.size()==1) return;
8

```

```

9     vector<int> v1, v2;
10
11     for(int i=0; i<v.size()/2; i++) v1.push_back(v[i
12     ]);
13     for(int i=v.size()/2; i<v.size(); i++) v2.
14     push_back(v[i]);
15
16     merge_sort(v1);
17     merge_sort(v2);
18
19     v1.push_back(INF);
20     v2.push_back(INF);
21
22     int ini1=0, ini2=0;
23
24     for(int i=0; i<v.size(); i++){
25         if(v1[ini1]<v2[ini2]){
26             v[i] = v1[ini1];
27             ini1++;
28         }else{
29             v[i] = v2[ini2];
30             ini2++;
31         }
32     }
33     return;
34 }

```

3.5 Mo

```

1 // Contar uma certa ocorrencia em queries de L a R
2 // O(K*(N+Q)), onde K = raiz(N)
3
4 // Problema: quantos numeros x existem tal que
5 // x ocorre exatamente x vezes no subarray
6
7 int block;
8 bool comp(tuple<int,int,int> a, tuple<int,int,int> b)
9 {
10     int l, r, idx, ll, rr, idx2;
11     tie(l, r, idx) = a;
12     tie(ll, rr, idx2) = b;
13
14     if(l/block != ll/block){
15         return l/block < ll/block;
16     }
17     return (l/block & 1) ? r < rr : r > rr;
18 }
19
20 class MO{
21 public:
22     vector<int> a;
23     int ans = 0;
24     unordered_map<int, int> cnt;
25     vector<tuple<int,int,int>> queries;
26
27     void add(int x){
28         if(cnt[x] == x) ans--;
29         cnt[x]++;
30         if(cnt[x] == x) ans++;
31     }
32
33     void del(int x){
34         if(cnt[x] == x) ans--;
35         cnt[x]--;
36         if(cnt[x] == x) ans++;
37     }
38
39     vector<int> get(){
40         vector<int> qans(queries.size());
41         sort(all(queries), comp);
42         int l=0, r=-1;
43         for(auto q: queries){
44             int ll, rr, idx;
45             tie(ll, rr, idx) = q;

```

```

46             while(r < rr) add(a[++r]);
47             while(l > ll) add(a[--l]);
48             while(r > rr) del(a[r--]);
49             while(l < ll) del(a[l++]);
50             qans[idx] = ans;
51         }
52         return qans;
53     }
54
55     MO(vector<int> a, vector<tuple<int,int,int>>
56     queries){
57         this->a = a;
58         this->queries = queries;
59         block = (int)sqrt((int)a.size());
60     }
61 };
62
63 int32_t main(){ sws;
64     int n, m;
65     cin>>n>>m;
66     vector<int> a(n);
67     for(int i=0; i<n; i++)cin>>a[i];
68     vector<tuple<int,int,int>> queries;
69     for(int i=0; i<m; i++){
70         int l, r;
71         cin>>l>>r;
72         queries.push_back({l-1, r-1, i});
73     }
74     MO mo(a, queries);
75     vector<int> ans = mo.get();
76     for(int i=0; i<m; i++){
77         cout<<ans[i]<<endl;
78     }
79     return 0;
80 }

```

3.6 Ordered Set

```

1 // disable define int long long
2 #include <ext/pb_ds/assoc_container.hpp>
3 #include <ext/pb_ds/tree_policy.hpp>
4 using namespace __gnu_pbds;
5 template <class T>
6     using ord_set = tree<T, null_type, less<T>,
7     rb_tree_tag,
8     tree_order_statistics_node_update>;
9
10 // k-th maior elemento - O(logN) - idx em 0
11 s.find_by_order(k)
12
13 // qtd elementos < k - O(logN)
14 s.order_of_key(k)
15
16 ord_set<int> s;

```

3.7 Segtree

```

1 // Build: O(N)
2 // Queries: O(log N)
3 // Update: O(log N)
4
5 // indexada em 0
6
7 class SegTree{
8     int n, elem_neutro = 0;
9     vector<int> tree, lazy, v;
10
11     int merge(int a, int b){
12         return a+b; //seg de soma
13     }
14
15     void build(int l, int r, int no){
16         if(l==r){

```

```

17         tree[no] = v[l];
18         return;
19     }
20     int mid = (l+r)/2;
21     build(l, mid, 2*no);
22     build(mid+1, r, 2*no+1);
23
24     tree[no] = merge(tree[2*no], tree[2*no+1]);
25 }
26
27 void update(int A, int B, int x, int l, int r,
28 int no){
29     prop(l, r, no);
30     if(B<l or r<A) return;
31     if(A<=l and r<=B){
32         lazy[no] += x; //update de soma
33         prop(l, r, no);
34         return;
35     }
36     int mid = (l+r)/2;
37
38     update(A, B, x, l, mid, 2*no);
39     update(A, B, x, mid+1, r, 2*no+1);
40
41     tree[no] = merge(tree[2*no], tree[2*no+1]);
42 }
43
44 void prop(int l, int r, int no){
45     if(lazy[no]!=0){
46         tree[no] += (r-l+1)*lazy[no]; //update de
47         soma
48         if(l!=r){
49             lazy[2*no] += lazy[no]; //update de
50             soma
51             lazy[2*no+1] += lazy[no]; //update de
52             soma
53         }
54         lazy[no] = 0;
55     }
56 }
57
58 int query(int A, int B, int l, int r, int no){
59     prop(l, r, no);
60     if(B<l or r<A) return elem_neutro;
61     if(A<=l and r<=B) return tree[no];
62     int mid = (l+r)/2;
63
64     return merge(query(A, B, l, mid, 2*no),
65                 query(A, B, mid+1, r, 2*no+1));
66 }
67
68 public:
69     SegTree(vector<int> &v){
70         this->n=v.size();
71         this->v=v;
72         tree.assign(4*n, 0);
73         lazy.assign(4*n, 0);
74         build(0, n-1, 1);
75     }
76
77     int query(int l, int r){return query(l, r, 0,
78 n-1, 1);}
79
80     void update(int l, int r, int val){update(l,
81 r, val, 0, n-1, 1);}
82
83     void out(){for(int i=0; i<n; i++){cout<<query
84 (i, i)<<" ";cout<<endl;}}
85 };
86
87 int32_t main(){
88     int n, q;
89     cin>>n>>q;
90     vector<int> v(n);
91     for(int i=0; i<n; i++)cin>>v[i];
92     SegTree seg(v);

```

```

83     while(q--){
84         int op; cin>>op;
85         if(op == 1){
86             int l, r, val;
87             cin>>l>>r>>val;
88             l--; r--;
89             seg.update(l, r, val);
90         }else{
91             int idx;
92             cin>>idx;
93             idx--;
94             cout<<seg.query(idx, idx)<<endl;
95         }
96     }
97     return 0;
98 }

```

3.8 Sqrt Decomposition

```

1 // Acha o elemento minimo do segmento de l a r
2 // O(N/K + K), onde K = raiz(N)
3
4 class Sqrt{
5     vector<int> a, b;
6     int n, k;
7
8     public:
9     void build(){
10         b.resize((n/k)+1);
11         for(int i=0; i<=(n/k); i++){
12             b[i] = LLINF;
13         }
14         for(int i=0; i<n; i++){
15             b[i/k] = min(b[i/k], a[i]);
16         }
17     }
18
19     void update(int idx, int val){
20         a[idx] = val;
21         int blockId = idx/k;
22         b[blockId] = LLINF;
23         for(int i=blockId*k; i<min(blockId+k, n); i
24 ++){
25             b[blockId] = min(b[blockId], a[i]);
26         }
27     }
28
29     int query(int l, int r){
30         int ans = LLINF;
31         int i = l;
32         while(i <= r){
33             if(i%k==0 and i+k-1<=r){
34                 ans = min(ans, b[i/k]);
35                 i+=k;
36             }else{
37                 ans = min(ans, a[i]);
38                 i++;
39             }
40         }
41         return ans;
42     }
43
44     Sqrt(vector<int> a){
45         this->a = a;
46         this->n = (int)a.size();
47         this->k = sqrt(n);
48         build();
49     }
50 };

```

3.9 Xortrie


```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 const int MAX = (2e5+5)*30;
5
6
7 class Trie{
8     int trie[MAX][2], pref[MAX];
9
10    int node = 1;
11
12    public:
13    void add(int num){
14        int cur = 1;
15        for(int i=30; i>=0; i--){
16            int bit = ((num &(1<<i)) >= 1);
17            if(!pref[trie[cur][bit]]) trie[cur][bit]
= ++node;
18            cur = trie[cur][bit];
19            pref[cur]++;
20        }
21    }
22
23    void erase(int num){
24        int cur = 1;
25        for(int i=30; i>=0; i--){
26            int bit = ((num &(1<<i)) >= 1);
27            cur = trie[cur][bit];
28            pref[cur]--;
29        }
30    }
31
32    int find(int num){
33        int cur = 1;
34        int ans = 0;
35        for(int i=30; i>=0; i--){
36            int bit = ((num &(1<<i)) >= 1);
37            if(pref[trie[cur][bit^1]]){
38                cur = trie[cur][bit^1];
39                ans += 1<<i;
40            }else
41                cur = trie[cur][bit];
42        }
43        return ans;
44    }
45 };

```

4 Geometria

4.1 Geometria

```

1 const long double EPS = 1e-9;
2 typedef long double ld;
3
4 // point p(x, y);
5 struct point {
6     ld x, y;
7     int id;
8     point(ld x=0, ld y=0): x(x), y(y){}
9
10    point operator+(const point &o) const{ return {x+
o.x, y+o.y}; }
11    point operator-(const point &o) const{ return {x-
o.x, y-o.y}; }
12    point operator*(ld t) const{ return {x*t, y*t}; }
13    point operator/(ld t) const{ return {x/t, y/t}; }
14    ld operator*(const point &o) const{ return x * o.
x + y * o.y; }
15    ld operator^(const point &o) const{ return x * o.
y - y * o.x; }
16 };
17

```

```

18 // line l(point(x1, y1), point(x2, y2));
19 struct line{
20     point a, b;
21     line(){}
22     line(point a, point b) : a(a), b(b){}
23 };
24
25 // ponto e em relacao a linha l
26 // counterclockwise
27 int ccw(line l, point e){
28     // -1=dir; 0=colinear; 1=esq;
29     point a = l.b-l.a, b=e-l.a;
30     ld tmp = a ^ b;
31     return (tmp > EPS) - (tmp < -EPS);
32 }
33
34 // se o ponto ta em cima da linha
35 bool isinseg(point p, line l){
36     point a = l.a-p, b = l.b-p;
37     return ccw(l, p) == 0 and (a * b) <= 0;
38 }
39
40 // se o seg de r intersecta o seg de s
41 bool interseg(line r, line s) {
42     if (isinseg(r.a, s) or isinseg(r.b, s)
43         or isinseg(s.a, r) or isinseg(s.b, r)) return
true;
44
45     return (ccw(r, s.a)>0) != (ccw(r, s.b)>0) and
(ccw(s, r.a)>0) != (ccw(s, r.b)>0);
46 }
47
48
49 // area do poligono
50 ld area_polygon(vector<point> vp){
51     ld area = 0;
52     for(int i=1; i<vp.size()-1; i++){
53         area += (vp[0]-vp[i]) ^ (vp[0]-vp[i+1]);
54     }
55     return (abs(area)/2);
56 }
57
58 // localizacao do ponto no poligono
59 int point_polygon(vector<point> vp, point p){
60     // -1=outside; 0=boundary; 1=inside;
61     int sz = vp.size();
62     int inter = 0;
63     for(int i=0; i<sz; i++){
64         int j = (i+1)%sz;
65         line l(vp[i], vp[j]);
66         if(isinseg(p, l)) return 0;
67
68         if(vp[i].x <= p.x and p.x < vp[j].x and ccw(l
, p) == 1) inter++;
69         else if(vp[j].x <= p.x and p.x < vp[i].x and
ccw(l, p) == -1) inter++;
70     }
71
72     if(inter%2==0) return -1;
73     else return 1;
74 }

```

5 Grafos

5.1 Binary Lifting

```

1 vector<int> adj[MAX];
2 const int LOG = 30;
3 int up[MAX][LOG], parent[MAX];
4
5 void process(int n){
6     for(int v=1; v<=n; v++){
7         up[v][0] = parent[v];

```

```

8         for(int i=1; i<LOG; i++){
9             up[v][i] = up[ up[v][i-1] ][i-1];
10        }
11    }
12 }
13
14 int jump(int n, int k){
15     for(int i=0; i<LOG; i++){
16         if(k & (1 << i)){
17             n = up[n][i];
18         }
19     }
20     if(n == 0) return -1;
21     return n;
22 }
23
24 int32_t main(){
25
26     int n, q; cin>>n>>q;
27
28     parent[1] = 0;
29     for(int i=1; i<=n-1; i++){
30         int x;
31         cin>>x;
32         parent[i+1] = x;
33
34         adj[i+1].pb(x);
35         adj[x].pb(i+1);
36     }
37     process(n);
38     for(int i=0; i<q; i++){
39         int a, b;
40         cin>>a>>b;
41
42         cout<<(jump(a,b))<<endl;
43     }
44 }

```

5.2 Diametro

```

1 // Acha o caminho mais longo de uma ponta ate outra
   ponta de uma arvore
2 // Complexidade: O(N)
3 // Lembrar de checar N == 1, diametro = 0
4 #include <bits/stdc++.h>
5 using namespace std;
6 const int MAX = 1e5+10;
7
8 vector<int> adj[MAX];
9 /*pair<int, int> bfs(int s, int N){
10     vi dist(N + 1, MAX); dist[s] = 0;
11     queue<int> q; q.push(s);
12     int last = s;
13
14     while(!q.empty()){
15         auto u = q.front(); q.pop();
16         last = u;
17         for(auto v: adj[u]){
18             if(dist[v]==MAX){
19                 dist[v]=dist[u]+1;
20                 q.push(v);
21             }
22         }
23     }
24     return {last, dist[last]};
25 }
26
27 int diameter_bfs(int N){
28     auto [v, _] = bfs(1, N);
29     auto [w, D] = bfs(v, N);
30
31     return D;
32 }*/

```

```

33
34 void dfs(int u, int p, vector<int> &dist){
35     for(auto v : adj[u]){
36         if(v == p) continue;
37         dist[v] = dist[u] + 1;
38         dfs(v, u, dist);
39     }
40 }
41
42 int diameter(int n){
43     vector<int> dist(n+1);
44     dfs(1, 0, dist);
45     // get farthest node from root
46     auto v = (int)(max_element(dist.begin(), dist.end()
47         ) - dist.begin());
48     // start from farthest node
49     dist[v] = 0;
50     dfs(v, 0, dist);
51     return *max_element(dist.begin(), dist.end());
52 }
53
54 int32_t main(){ sws;
55     int n; cin>>n;
56     for(int i=0; i<n-1; i++){
57         int a, b;
58         cin>>a>>b;
59         adj[a].pb(b);
60         adj[b].pb(a);
61     }
62     if(n == 1) cout<<0<<endl;
63     else cout<<diameter(n)<<endl;
64     return 0;
65 }

```

5.3 Kruskall

```

1 // Arvore geradora minima (arvore conexa com peso
   minimo)
2 // O(MlogN)
3
4 #include <bits/stdc++.h>
5 using namespace std;
6
7 int n;
8 class DSU{
9     vector<int> parent, sz;
10 public:
11     void make(int v){
12         parent[v] = v;
13         sz[v] = 1;
14     }
15
16     int find(int v){
17         if (v == parent[v]) return v;
18         return parent[v] = find(parent[v]);
19     }
20
21     void union_(int a, int b){
22         a = find(a), b = find(b);
23
24         if(sz[b]>sz[a]) swap(a,b);
25         if (a != b){
26             sz[a] += sz[b];
27             parent[b] = a;
28         }
29     }
30
31     bool same(int a, int b){
32         a = find(a), b = find(b);
33         return a == b;
34     }
35
36     DSU(int n): parent(n+1), sz(n+1){

```

```

37     for(int i=1; i<=n; i++) make(i);
38 }
39 };
40
41 // {a, b, weight}
42 vector<tuple<int,int,int>> MST(vector<tuple<int,int,
43 int>> &v){
44     DSU dsu(n);
45     sort(v.begin(), v.end());
46     vector<tuple<int,int,int>> ans;
47     for(int i=0; i<v.size(); i++){
48         int w, a, b;
49         tie(w, a, b) = v[i];
50         if(!dsu.same(a, b)){
51             dsu.union_(a, b);
52             ans.push_back({a, b, w});
53         }
54     }
55     return ans;
56 }
57 int32_t main(){
58     int m;
59     cin>>n>>m;
60     DSU dsu(n);
61     vector<tuple<int,int,int>> vt;
62     for(int i=0; i<m; i++){
63         int a, b, w;
64         cin>>a>>b>>w;
65         // {weight, a, b}
66         vt.push_back({w, a, b});
67     }
68     vector<tuple<int,int,int>> ans = MST(vt);
69     return 0;
70 }

```

5.4 Lca

```

1  /*
2  Lowest Common ancestor (LCA) - dado uma Arvore cuja
   raiz eh um vertice arbitrario e dois vertices u,v
   que a pertencem, diga qual eh o no mais baixo(
   relativo a raiz) que eh ancestral de u,v.
3  */
4  // Complexidades:
5  // build - O(n log(n))
6  // lca - O(log(n))
7
8  #include <bits/stdc++.h>
9  using namespace std;
10 #define ll long long
11 const int SIZE = 2e5+5;
12 const int LOG = 30; // log2(SIZE)+1;
13 int depth[SIZE];
14 vector<pair<int,int>> adj[SIZE];
15 int up[SIZE][LOG];
16
17 void dfs(int u, int p) {
18     for(auto edge : adj[u]) {
19         int v, w;
20         tie(v, w) = edge;
21         if(v != p){
22             up[v][0] = u;
23             //weight[v] = weight[u] + w;
24             depth[v] = depth[u] + 1;
25             for(int i=1; i<LOG; i++){
26                 up[v][i] = up[ up[v][i-1] ][i-1];
27             }
28             dfs(v, u);
29         }
30     }
31 }
32

```

```

33 int lca(int a, int b) {
34     if(depth[a] < depth[b]) swap(a,b);
35     int k = depth[a] - depth[b];
36     for(int i=0; i<LOG; i++){
37         if(k & (1 << i)){
38             a = up[a][i];
39         }
40     }
41     if(a == b) return a;
42     for (int i = LOG-1; i >= 0; i--) {
43         if(up[a][i] != up[b][i]) {
44             a = up[a][i];
45             b = up[b][i];
46         }
47     }
48     return up[a][0];
49 }
50
51 ll dist(int u, int v){
52     return depth[u] + depth[v] - 2*depth[lca(u,v)];
53     // return weight[u] + weight[v] -2*weight[lca(u,v)
54     ]];
55 }
56
57 int main() {
58     int n; cin>>n;
59     for(int i=0; i<n-1; i++){
60         int x, y, z;
61         cin>>x>>y>>z;
62         adj[x].push_back({y, z});
63         adj[y].push_back({x, z});
64     }
65     // raiz
66     dfs(1, 0);
67
68     int q; cin>>q;
69     while(q--){
70         int a, b, c;
71         cin>>a>>b>>c;
72         long long x = dist(a, b) + dist(b, c);
73         cout<<x<<endl;
74     }
75 }

```

5.5 Bellman Ford

```

1  /*
2  Algoritmo de busca de caminho minimo em um digrafo (
   grafo orientado ou dirigido) ponderado, ou seja,
   cujas arestas tem peso, inclusive negativo.
3  Acha ciclo negativo
4  O(V*E)
5  */
6
7  int d[MAX];
8  int parent[MAX];
9  vector<pair<int,int>> adj[MAX];
10
11 int32_t main(){ sws;
12     int n, m;
13     cin>>n>>m;
14     for(int i=1; i<=n; i++){
15         d[i] = LLINF;
16     }
17     for(int i=0; i<m; i++){
18         int a, b, c;
19         cin>>a>>b>>c;
20         adj[a].pb({b,c});
21     }
22     d[1] = 0;
23
24     int src_cycle = -1;

```

```

25 for(int j=1; j<=n and src_cycle; j++){
26     src_cycle = 0;
27     for(int u=1; u <= n; u++){
28         for(auto [v, w]: adj[u]){
29             if(d[u] + w < d[v]){
30                 d[v] = d[u] + w;
31                 parent[v] = u;
32                 src_cycle = v;
33             }
34         }
35     }
36 }
37 // there is no negative cycle
38 if(!src_cycle){cout<<"NO"<<endl;}
39 else {
40     // there is negative cycle
41     cout<<"YES"<<endl;
42     vector<int> v;
43     int a = src_cycle;
44     for(int i = 0; i < n; i++)
45         src_cycle = parent[src_cycle];
46
47     int atual=src_cycle;
48     while(true){
49         v.pb(atual);
50         if(atual == src_cycle && v.size()>1)break
51     };
52     atual = parent[atual];
53 }
54 reverse(all(v));
55 print(v, (int)v.size());
56
57 return 0;
58 }

```

5.6 Bfs

```

1 #include <bits/stdc++.h>[]
2 using namespace std;
3
4 //-----
5 #define MAXN 50050
6
7 int n, m;
8 bool visited[MAXN];
9 vector<int> lista[MAXN];
10 //-----
11
12 void bfs(int x){
13
14     queue<int> q;
15     q.push(x);
16     while(!q.empty()){
17         int v = q.front();
18         q.pop();
19         visited[v] = true;
20         for(auto i : lista[v]){
21             if(!visited[i]){
22                 q.push(i);
23             }
24         }
25     }
26 }

```

5.7 Bridges

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 #define endl "\n"
5 #define sws std::ios::sync_with_stdio(false); cin.tie
6 (NULL); cout.tie(NULL);

```

```

6 #define pb push_back
7 const int MAX = 1e5+5;
8
9 vector<int> adj[MAX];
10 int timer=0;
11 int low[MAX], tin[MAX];
12 bool bridge=false;
13 bool visited[MAX];
14
15 void dfs(int v, int p = -1) {
16     visited[v] = true;
17     tin[v] = low[v] = timer++;
18     for (int to : adj[v]) {
19         if (to == p) continue;
20         if (visited[to]) {
21             low[v] = min(low[v], tin[to]);
22         } else {
23             dfs(to, v);
24             low[v] = min(low[v], low[to]);
25             if (low[to] > tin[v]){
26                 //IS_BRIDGE(v, to);
27             }
28         }
29     }
30
31 int32_t main(){ sws;
32     int n, m;
33     cin>>n>>m;
34
35     for(int i=0; i<m; i++){
36         int a, b;
37         cin>>a>>b;
38
39         adj[a].pb(b);
40         adj[b].pb(a);
41     }
42     for(int i=1; i<=n; i++){
43         if(!visited[i]) dfs(i);
44     }
45     if(bridge)cout<<"YES"<<endl;
46     else cout<<"NO"<<endl;
47
48     return 0;
49 }

```

5.8 Dfs

```

1 #include <iostream>
2 #include <vector>
3 #include <stack>
4
5 using namespace std;
6
7 //-----
8 #define MAXN 50050
9
10 int n, m;
11 bool visited[MAXN];
12 vector<int> lista[MAXN];
13 //-----
14
15 void dfs(int x){
16     visited[x] = true;
17     for(auto i : lista[x]){
18         if(!visited[i]){
19             dfs(i);
20         }
21     }
22 }
23
24 void dfsStack(int x){
25     stack<int> s;
26     s.push(x);

```

```

27 while(!s.empty()){
28     int v = s.top();
29     s.pop();
30     visited[v] = true;
31     for(auto i : lista[v]){
32         if(!visited[i]){
33             s.push(i);
34         }
35     }
36 }
37 }

```

5.9 Dfs Tree

```

1 const int MAX = 1e5;
2 int desce[MAX], sobe[MAX], vis[MAX], h[MAX];
3 int backedges[MAX], pai[MAX];
4
5 // backedges[u] = backedges que comecam embaixo de (
   // ou =) u e sobem pra cima de u; backedges[u] == 0
   // => u eh ponte
6 void dfs(int u, int p) {
7     if(vis[u]) return;
8     pai[u] = p;
9     h[u] = h[p]+1;
10    vis[u] = 1;
11
12    for(auto v : g[u]) {
13        if(p == v or vis[v]) continue;
14        dfs(v, u);
15        backedges[u] += backedges[v];
16    }
17    for(auto v : g[u]) {
18        if(h[v] > h[u]+1)
19            desce[u]++;
20        else if(h[v] < h[u]-1)
21            sobe[u]++;
22    }
23    backedges[u] += sobe[u] - desce[u];
24 }

```

5.10 Dijkstra

```

1 // Acha o menor caminho de um ponto inicial para
   // todos os outros
2 // Complexidade: O(|V|+|E|*log|V|)
3
4 #include <bits/stdc++.h>
5 using namespace std;
6 #define ll long long
7 typedef pair<int,int> pii;
8
9 const int N = 100005;
10 const ll oo = 1e18;
11
12 ll d[N]; // vetor onde guardamos as distancias
13
14 int n; // numeros de vertices
15 vector<pair<int, ll>> adj[N];
16
17 void dijkstra(int start){
18     for(int u = 1; u <= n; u++){
19         d[u] = oo;
20
21         priority_queue<pii,
22             vector<pii>,
23             greater<pii> > pq;
24
25         d[start] = 0;
26         pq.push({d[start], start});
27
28         ll dt, w;

```

```

29 int u, v;
30 while(!pq.empty()){
31     auto [dt, u] = pq.top(); pq.pop();
32     if(dt > d[u]) continue;
33     for(auto [v, w] : adj[u]){
34         if(d[v] > d[u] + w){
35             d[v] = d[u] + w;
36             pq.push({d[v], v});
37         }
38     }
39 }
40 }
41
42 int main(){
43
44     // le o input, qnt de vertices, arestas
45     // e vertice inicial(start)
46     int start = 0; // inicial
47     dijkstra(start);
48
49     for(int u = 1; u <= n; u++){
50         printf("Distancia de %d para %d: %lld\n",
51             start, u, d[u]);
52     }
53 }

```

5.11 Euler Path

```

1 // Acha um caminho em que visita todas as arestas
   // somente uma vez
2
3 class EulerPath{
4     int n, m, id=0;
5     bool impossible=false, directed;
6     vector<int> in, out, deg;
7     vector<pair<int,int>> adj[MAX], path;
8     vector<bool> visited;
9     int src = -1;
10    public:
11    void add(int a, int b){
12        if(directed){
13            adj[a].pb({b, id});
14            out[a]++, in[b]++;
15        }else{
16            adj[a].pb({b, id}), adj[b].pb({a, id});
17            deg[a]++, deg[b]++;
18        }
19        id++;
20    }
21
22    void dfs(int p, int u){
23        while(!adj[u].empty()){
24            pair<int, int> p = adj[u].back(); adj[u].
25            pop_back();
26            int v, id; tie(v, id) = p;
27            if(visited[id]) continue;
28            visited[id] = true;
29            dfs(u, v);
30
31            if(path.size() and path.back().first != u)
32                impossible=true;
33            path.pb({p, u});
34        }
35
36        // exists, path
37        vector<int> findEulerPath(){
38            for(int i=1; i<=n; i++) if(deg[i]%2 != 0)
39                return {};
40            dfs(-1, src);
41            if((path.size() != m+1) or impossible) return
42                {};
43            vector<int> ans;

```

```

40         reverse(all(path));
41         for(int i=0; i<path.size(); i++){
42             ans.pb(path[i].second);
43         }
44         return ans;
45     }
46 }
47 EulerPath(int _n, int _m, bool _directed, int
_src=-1):
48 in(n+1), out(n+1), deg(n+1), visited(m, 0),
49 n(_n), m(_m), directed(_directed), src(_src){}
50 };
51
52 int32_t main(){ sws;
53     int n, m;
54     cin>>n>>m;
55     EulerPath ep(n, m, true, 1);
56     for(int i=0; i<m; i++){
57         int a, b;
58         cin>>a>>b;
59         ep.add(a, b);
60     }
61     vector<int> ans = ep.findEulerPath();
62     if(ans.size()){
63         print(ans, ans.size());
64     }else{
65         cout<<"IMPOSSIBLE"<<endl;
66     }
67
68     return 0;
69 }

```

5.12 Floyd Warshall

```

1  /*
2  Algoritmo de caminho mais curto com todos os pares.
3  Complexidade:  $O(N^3)$ 
4  */
5
6  #include <bits/stdc++.h>
7  using namespace std;
8
9  const int oo = 1000000000; // infinito
10
11 int main(){
12
13     int n, m; cin>>n>>m;
14
15     vector<vector<int>> dist(n+1, vector<int> (n+1));
16
17     for(int i=0; i<n+1; i++){
18         for(int j=0; j<n+1; j++){
19             dist[i][j] = oo;
20         }
21     }
22
23     for(int i=0; i<n+1; i++){
24         dist[i][i]=0;
25     }
26
27     for(int i=0; i<m; i++){
28         int começa, termina, custo;
29         cin>>começa>>termina>>custo;
30
31         // grafo direcionado
32         dist[começa][termina] = custo;
33     }
34
35     for(int k=1; k<=n; k++){ // intermediario
36         for(int i=1; i<=n; i++){
37             for(int j=1; j<=n; j++){
38                 //(i,k,j) = ir de i pra j passando
39                 por k;

```

```

39
40         // relaxar distancia de i pra j
41         dist[i][j] = min(dist[i][j], dist[i][
42         k] + dist[k][j]);
43     }
44 }
45     return 0;
46 }

```

5.13 Kosaraju

```

1  // Acha componentes fortemente conexas
2  // ou seja, que tem caminho entre todos os pares de
   vertices
3  //  $O(n+m)$ 
4
5  // SCC from BenQ
6  class SCC{
7      int N;
8      public:
9      vector<int> adj[MAX], radj[MAX];
10     stack<int> st;
11     vector<bool> visited;
12     // todas as componentes
13     vector<int> comps;
14     // componente do vertice
15     vector<int> comp;
16
17     void add(int x, int y) {
18         adj[x].pb(y), radj[y].pb(x);
19     }
20     void dfs(int u){
21         visited[u] = true;
22         for(auto v: adj[u]) if(!visited[v]) dfs(v);
23         st.push(u);
24     }
25     void dfs2(int u, int c){
26         comp[u] = c;
27         for(auto v: radj[u]) if(comp[v] == -1) dfs2(v
, c);
28     }
29     void gen() {
30         for(int i=1; i<=N; i++) if(!visited[i]) dfs(i
);
31         while(!st.empty()){
32             int u = st.top(); st.pop();
33             if(comp[u] == -1){
34                 dfs2(u, u);
35                 comps.pb(u);
36             }
37         }
38     }
39     SCC(int n){
40         N = n+1;
41         comp.assign(N, -1);
42         visited.assign(N, false);
43     }
44 };
45
46 int32_t main(){ sws;
47     int n, m;
48     cin>>n>>m;
49     SCC scc(n);
50     for(int i=0; i<m; i++){
51         int a, b;
52         cin>>a>>b;
53         scc.add(a, b);
54     }
55     int comp=0;
56     vector<int> ans(n+1);
57     scc.gen();
58     cout<<scc.comps.size()<<endl;

```

```

59     for(int i=1; i<=n; i++){
60         if(!ans[scc.comp[i]]){
61             ans[scc.comp[i]] = ++comp;
62         }
63     }
64     for(int i=1; i<=n; i++){
65         cout<<ans[scc.comp[i]]<<" ";
66     }
67     cout<<endl;
68     return 0;
69 }

```

5.14 Topo Sort

```

1 // topological sort
2 // retorna uma ordenacao topologica
3 // caso for um dag, se nao, retorna vazio se tiver
  ciclo
4 // O(n+m)
5 // indexado em 1 os vertices
6
7 int n;
8 int visited[MAX];
9 vector<int> adj[MAX];
10 int pos=0;
11 vector<int> ord;
12 bool has_cycle=false;
13
14 void dfs(int v){
15     visited[v] = 1;
16     for(auto u : adj[v]){
17         if(visited[u] == 1) has_cycle=true;
18         else if(!visited[u]) dfs(u);
19     }
20     ord[pos--] = v;
21     visited[v] = 2;
22 }
23
24 vector<int> topo_sort(int n){
25     ord.assign(n, 0);
26     has_cycle = false;
27     pos = n-1;
28     for(int i=1; i<=n; i++){
29         if(!visited[i]) dfs(i);
30     }
31
32     if(has_cycle) return {};
33     else return ord;
34 }
35
36 int main(){
37     int m;
38     cin>>n>>m;
39
40     for(int i=0; i<m; i++){
41         int a, b;
42         cin>>a>>b;
43         adj[a].pb(b);
44     }
45
46     vector<int> ans = topo_sort(n);
47
48     return 0;
49 }

```

6 Math

6.1 Combinatoria

```

1 // quantidade de combinacoes possiveis sem repeticao
  de 2 numeros

```

```

2 int comb(int k){
3     if(k==1 or k==0) return 0;
4     return (k*(k-1))/2;
5 }
6
7 int fat[MAX], ifat[MAX];
8
9 void factorial(){
10     fat[0] = 1;
11     for(int i=0; i<MAX; i++){
12         if(i > 0) fat[i] = (i * fat[i-1]) % MOD;
13         ifat[i] = fast_exp(fat[i], MOD-2, MOD);
14     }
15 }
16
17 // N escolhe K
18 int choose(int n, int k){
19     if(k > n or k<0) return 0;
20     return (((fat[n] * ifat[k]) % MOD) * ifat[n-k]) %
      MOD;
21 }

```

6.2 Dec To Bin

```

1 int binary_to_decimal(long long n) {
2     int dec = 0, i = 0, rem;
3
4     while (n!=0) {
5         rem = n % 10;
6         n /= 10;
7         dec += rem * pow(2, i);
8         ++i;
9     }
10
11     return dec;
12 }
13
14 long long decimal_to_binary(int n) {
15     long long bin = 0;
16     int rem, i = 1;
17
18     while (n!=0) {
19         rem = n % 2;
20         n /= 2;
21         bin += rem * i;
22         i *= 10;
23     }
24
25     return bin;
26 }
27
28 // copiei da nathalia, tem que ver se funciona
  certinho

```

6.3 Divisibilidade

```

1 // 2 -> se eh par
2 // 3 -> se a soma dos algarismos eh divisivel por 3
3 // 4 -> se os dois ultimos algarismos eh divisivel
  por 4
4 // 5 -> se o ultima algarismo eh 0 ou 5
5 // 6 -> se eh par e a soma dos algarismos eh
  divisivel por 3
6 // 7 -> se o dobro do ultimo algarismo subtraido do
  numero sem o ultimo algarismo eh divisivel por 7
7 // 8 -> se os 3 ultimos algarismos eh divisivel por 8
8 // 9 -> se a soma dos algarismos eh divisivel por 9
9 // 10 -> se o ultimo algarimo eh 0

```

6.4 Divisores

```

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

```

```

2 using namespace std;
3
4 vector<long long> get_divisors(long long n){
5     vector<long long> divs;
6     for(long long i = 1; i*i <= n; i++){
7         if(n%i == 0){
8             divs.push_back(i);
9             long long j = n/i;
10            if(j != i)
11                divs.push_back(j);
12        }
13    }
14    return divs;
15 }

```

6.5 Fatora

```

1 map<int,int> fatora(int n){
2     map<int,int> fact;
3     for(int i = 2; i*i <= n; i++){
4         while(n%i == 0){
5             fact[i]++;
6             n /= i;
7         }
8     }
9     if(n > 1)
10        fact[n]++;
11    return fact;
12 }

```

6.6 Mdc

```

1 // Greatest common divisor / MDC
2
3 long long gcd(long long a, long long b){
4     return b ? gcd(b, a % b) : a;
5 }
6
7 // or just use __gcd(a,b)

```

6.7 Mmc

```

1 // Least Common Multiple - MMC
2 #include <bits/stdc++.h>
3 using namespace std;
4
5 long long lcm(long long a, long long b){
6     return (a/__gcd(a,b)*b);
7 }

```

6.8 Pa

```

1 // Termo Geral
2 //  $A_n = A_1 + (n-1)*d$ 
3
4 // Soma
5 //  $S_n = (n/2)(2*A_1 + (n-1)*d)$ 
6
7 // Soma de 1 a K
8 int pa(int k){
9     return (k*(k+1))/2;
10 }

```

6.9 Pg

```

1 // Termo Geral
2 //  $A_n = A_1 * r^{(n-1)}$ 
3
4 // Soma
5 //  $(A(r^n - 1))/(r - 1)$ 

```

6.10 Pollard-rho

```

1 //  $O(\sqrt{N} * \log N)$ 
2
3 ll a[MAX];
4
5 ll mul(ll a, ll b, ll m){
6     ll ret = a*b - (ll)((long double)1/m*a*b+0.5)*m;
7     return ret < 0 ? ret+m : ret;
8 }
9
10 ll pow(ll x, ll y, ll m) {
11     if (!y) return 1;
12     ll ans = pow(mul(x, x, m), y/2, m);
13     return y%2 ? mul(x, ans, m) : ans;
14 }
15
16 bool prime(ll n) {
17     if (n < 2) return 0;
18     if (n <= 3) return 1;
19     if (n % 2 == 0) return 0;
20
21     ll r = __builtin_ctzll(n - 1), d = n >> r;
22     for (int a : {2, 325, 9375, 28178, 450775,
23                 9780504, 795265022}) {
24         ll x = pow(a, d, n);
25         if (x == 1 or x == n - 1 or a % n == 0)
26             continue;
27
28         for (int j = 0; j < r - 1; j++) {
29             x = mul(x, x, n);
30             if (x == n - 1) break;
31         }
32         if (x != n - 1) return 0;
33     }
34     return 1;
35 }
36
37 ll rho(ll n) {
38     if (n == 1 or prime(n)) return n;
39     auto f = [n](ll x) {return mul(x, x, n) + 1;};
40
41     ll x = 0, y = 0, t = 30, prd = 2, x0 = 1, q;
42     while (t % 40 != 0 or __gcd(prd, n) == 1) {
43         if (x==y) x = ++x0, y = f(x);
44         q = mul(prd, abs(x-y), n);
45         if (q != 0) prd = q;
46         x = f(x), y = f(f(y)), t++;
47     }
48     return __gcd(prd, n);
49 }
50
51 vector<ll> fact(ll n) {
52     if (n == 1) return {};
53     if (prime(n)) return {n};
54     ll d = rho(n);
55     vector<ll> l = fact(d), r = fact(n / d);
56     l.insert(l.end(), r.begin(), r.end());
57     return l;
58 }
59
60 int main(){
61     set<ll> primes;
62     int M, N, K; cin >> M >> N >> K;
63     loop(i,0,N){
64         cin >> a[i];
65         vector<ll> aprimes = fact(a[i]);
66         for(auto prime : aprimes){
67             primes.insert(prime);
68         }
69     }
70     int m, n, d;

```



```

70     loop(i,0,K) cin >> m >> n >> d;
71     for(auto prime : primes){
72         cout << prime << " ";
73     }
74 }

```

6.11 Primos

```

1 // PRIMALIDADE
2
3 #include <bits/stdc++.h>
4 using namespace std;
5
6 const int MAX = 1e5+7;
7
8 void crivo(){
9     vector<int> crivo(MAX, 1);
10    for(int i=2; i*i<=MAX; i++){
11        if(crivo[i]==1){
12            for(int j=i+i; j<MAX; j+=i){
13                crivo[j]=0;
14            }
15        }
16    }
17 }
18 bool isPrime[MAX];
19 vector<int> generate_primes() {
20     vector<int> primes;
21     isPrime[1]=isPrime[0]=1;
22     for(int i=2; i<MAX; i++){
23         if(!isPrime[i]){
24             primes.pb(i);
25             for(int j=i+i; j<MAX; j+=i){
26                 isPrime[j]=1;
27             }
28         }
29     }
30     return primes;
31 }
32
33 bool is_prime(int num){
34     for(int i = 2; i*i<= num; i++) {
35         if(num % i == 0) {
36             return false;
37         }
38     }
39     return true;
40 }
41 }

```

7 Strings

7.1 Suffix Array

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 #define ll long long
5 #define sws ios::sync_with_stdio(false);cin.tie( NULL
6 #define print(x) for (auto &it : x) cout<<it<<' ';<<
7 #define loop(i,a,n) for(int i=a; i < n; i++)
8 #define pb(x) push_back(x);
9 #define vi vector<int>
10 #define mp(x,y) make_pair(x,y)
11 #define pii pair<int,int>
12 #define pqi priority_queue<int, vector<int>, greater<
13 const ll MOD = 1e9+7;
14 const int INF = 0x3f3f3f3f;

```

```

15 const ll LLINF = 0x3f3f3f3f3f3f3f3f;
16
17 vector<int> suffix_array(string s) {
18     s += "$";
19     int n = s.size(), N = max(n, 260);
20     vector<int> sa(n), ra(n);
21     for (int i = 0; i < n; i++) sa[i] = i, ra[i] = s[
22         i];
23
24     for (int k = 0; k < n; k ? k *= 2 : k++) {
25         vector<int> nsa(sa), nra(n), cnt(N);
26
27         for (int i = 0; i < n; i++) nsa[i] = (nsa[i]-
28             k+n)%n, cnt[ra[i]]++;
29         for (int i = 1; i < N; i++) cnt[i] += cnt[i
30             -1];
31         for (int i = n-1; i+1; i--) sa[--cnt[ra[nsa[i
32             ]]]] = nsa[i];
33
34         for (int i = 1, r = 0; i < n; i++) nra[sa[i]]
35             = r += ra[sa[i]] !=
36             ra[sa[i-1]] or ra[(sa[i]+k)%n] != ra[(sa[
37                 i-1]+k)%n];
38         ra = nra;
39         if (ra[sa[n-1]] == n-1) break;
40     }
41     return vector<int>(sa.begin()+1, sa.end());
42 }
43
44 vector<int> kasai(string s, vector<int> sa) {
45     int n = s.size(), k = 0;
46     vector<int> ra(n), lcp(n);
47     for (int i = 0; i < n; i++) ra[sa[i]] = i;
48
49     for (int i = 0; i < n; i++, k -= !!k) {
50         if (ra[i] == n-1) { k = 0; continue; }
51         int j = sa[ra[i]+1];
52         while (i+k < n and j+k < n and s[i+k] == s[j+
53             k]) k++;
54         lcp[ra[i]] = k;
55     }
56     return lcp;
57 }
58
59 int32_t main(){
60     sws;
61     string s;
62     cin>>s;
63
64     vector<int> suf = suffix_array(s);
65     vector<int> lcp = kasai(s, suf);
66
67     ll ans = 0;
68     for(int i=0; i<s.size(); i++){
69         if(islower(s[suf[i]])){
70             int sz = s.size()-suf[i];
71             ans += (sz - lcp[i]);
72         }
73     }
74     cout<<ans<<endl;
75 }

```

7.2 Trie

```

1 // Constroi e procura por uma string em uma arvore
2 // Trie t;
3 // Trie t(qtd_char, c_min, max_size)
4 // qtd_char = qntd maxima de caracteres
5 // c_min = menor caractere
6 // max_size = tamanho maximo de strings
7
8 // Complexidade - O(N*|s|*qtd_char)

```

```

9
10 #include <bits/stdc++.h>
11 using namespace std;
12
13 #define sws std::ios::sync_with_stdio(false); cin.tie
14 (NULL); cout.tie(NULL);
15 const int MAX = 2005;
16
17 class Trie{
18     int node = 1;
19     char c_min;
20     int qtd_char, max_size;
21     vector<vector<int>> trie;
22     vector<int> pref, end;
23
24 public:
25     void add(string s){
26         int cur = 1;
27         for(auto c: s){
28             if(!trie[cur][c-c_min]){
29                 trie[cur][c-c_min] = ++node;
30             }
31             cur = trie[cur][c-c_min];
32             pref[cur]++;
33         }
34         end[cur]++;
35     }
36
37     void erase(string s){
38         int cur = 1;
39         for(auto c: s){
40             cur = trie[cur][c-c_min];
41             pref[cur]--;
42         }
43         end[cur]--;
44     }
45
46     int find(string s){
47         int cur = 1;
48         for(auto c: s){
49             if(!trie[cur][c-c_min]) return 0;
50             cur = trie[cur][c-c_min];
51         }
52         return cur;
53     }
54
55     int count_pref(string s){
56         return pref[find(s)];
57     }
58
59     Trie(int qtd_char_=26, char c_min_ = 'a', int
60         max_size_=MAX):
61         c_min(c_min_), qtd_char(qtd_char_), max_size(
62         max_size_){
63         trie.resize(max_size, vector<int>(qtd_char));
64         pref.resize(max_size);
65         end.resize(max_size);
66     }
67 };
68
69 int32_t main(){ sws;
70     Trie t;
71     t.add("abcd");
72     t.add("ad");
73     t.erase("ad");
74     cout<<t.count_pref("a")<<endl;
75
76     return 0;
77 }

```

7.3 Zfunction

```
1 // complexidades
```

```

2 // z - O(|s|)
3 // match: O(|s|+|p|)
4 vector<int> z_func(string s){
5     int n = s.size();
6     vector<int> z(n);
7     int l=0, r=0;
8     for (int i = 1, i < n; i++) {
9         if (i <= r)
10             z[i] = min(z[i - 1], r - i + 1);
11         while (i + z[i] < n && s[z[i]] == s[i + z[i]
12             ])
13             z[i]++;
14         if (i + z[i] - 1 > r)
15             l = i, r = i + z[i] - 1;
16     }
17     return z;
18 }
19 // string matching
20 // quantas vezes B aparece em A
21 int32_t main(){ sws;
22     string a, b;
23     cin>>a>>b;
24
25     string s = b + '$' + a;
26     vector<int> z = z_func(s);
27     int ans = 0;
28     for(int i=0; i<z.size(); i++){
29         if(z[i] == b.size())ans++;
30     }
31     cout<<ans<<endl;
32
33     return 0;
34 }

```

8 Template

8.1 Template

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 //alias comp='g++ -std=c++17 -g -O2 -Wall -
4 Wconversion -Wshadow -fsanitize=address,undefined
5 -fno-sanitize-recover -ggdb -o out'
6
7 #define sws std::ios::sync_with_stdio(false); cin.tie
8 (NULL); cout.tie(NULL);
9 #define int long long
10 #define endl "\n"
11 #define input(x) for (auto &it : x) cin >> it
12 #define pb push_back
13 #define all(x) x.begin(), x.end()
14 #define ff first
15 #define ss second
16 #define TETO(a, b) ((a) + (b-1))/(b)
17 #define dbg(msg, x) cout << msg << " = " << x << endl
18 #define print(x,y) for (auto &it : x) cout << it
19
20 typedef long long ll;
21 typedef long double ld;
22 typedef vector<int> vi;
23 typedef pair<int,int> pii;
24 typedef priority_queue<int, vector<int>, greater<int
25 >> pqi;
26
27 const ll MOD = 1e9+7;
28 const int MAX = 1e4+5;
29 const ll LLINF = 0x3f3f3f3f3f3f3f3f;
30 const double PI = acos(-1);
31
32 int32_t main(){ sws;
33
34
35
36
37
38
39
40

```

```

31     return 0;
32 }

```

9 zExtra

9.1 Formulasmat

```

1  int gcd(int a, int b) {
2      if (b == 0) return a;
3      return gcd(b, a % b);
4  }
5
6  // number of elements
7  long long sum_of_n_first_squares(int n) {
8      return (n * (n - 1) * (2 * n - 1)) / 6;
9  }
10
11 // first element, last element, number of elements
12 long long sum_pa(int a1, int an, int n) {
13     return ((a1 + an) * n) / 2;
14 }
15
16 // first element, number of elements, ratio
17 long long general_term_pa(int a1, int n, int r) {
18     return a1 + (n - 1) * r;
19 }
20
21 // first term, numbers of elements, ratio
22 long long sum_pg(int a1, int n, int q) {
23     return (a1 * (fexp(q, n) - 1)) / (q - 1);
24 }
25
26 // -1 < q < 1
27 // first term, ratio
28 long long sum_infinite_pg(int a1, double q) {
29     return a1 * (1 - q);
30 }
31
32 // first term, number of elements, ratio
33 long long general_term_pg(int a1, int n, int q) {
34     return a1 * fexp(q, n - 1);
35 }
36
37 // first element of original pa, first element of
38 // derived pa, number of elements of original pa,
39 // ratio of derived pa
40 long long sum_second_order_pa(int a1, int b1, int n,
41                                int r) {
42     return a1 * n + (b1 * n * (n - 1)) / 2 + (r * n * (n
43         - 1) * (n - 2)) / 6;
44 }
45
46 // log
47 int intlog(double base, double x) {
48     return (int)(log(x) / log(base));
49 }
50
51 // sum from one to n
52 (n * (n + 1)) / 2
53
54 // gcd
55 long long gcd(long long a, long long b){
56     return b ? gcd(b, a % b) : a;
57 }
58
59 // or just use __gcd(a,b)
60
61 // lcm
62 long long lcm(long long a, long long b){
63     return (a / __gcd(a,b) * b);
64 }

```

```

60 }
61
62 // distancia manhattan
63 // https://vjudge.net/contest/539684#problem/H
64
65 // distancia euclidiana
66
67 // GEOMETRIA
68 // seno
69 a / sen(a) = b / sen(b) = c / sen(c)
70
71 //cosseno
72 a^2 = b^2 + c^2 - 2*b*c*cos(a)
73
74 // area losango
75 A = (1/2) * diagonal_maior * diagonal_menor
76
77 // volume prisma
78 V = B * H
79
80 //volume esfera
81 V = (4/3) * PI * R^3
82
83 //volume piramide
84 V = (1/3) * B * H
85
86 //volume cone
87 V = (1/3) * PI * R^2 * H
88
89 //condicao de existencia
90 a - b | < c < a + b
91
92 // combinacao sem rep.
93 C(n x) = n! / (x! (n-x)!)
94
95 // combinacao com rep.
96 C(n m) = (m + n - 1)! / (n! (m-1)!)
97
98 // perm sem rep
99 p = n!
100
101 // perm com rep
102 p = n! / (rep1! rep2! ... repn!)
103
104 // perm circ
105 P = (n-1)!

```

9.2 Getline

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  // Sempre usar cin.ignore() entre um cin e um getline
4  int main() {
5
6      string s1; cin>>s1;
7      cin.ignore();
8      while (true) {
9          string s; getline(cin, s);
10         if (s == "PARO") break;
11         cout<<"A"<<endl;
12     }
13     string s2; cin>>s2;
14     cin.ignore();
15     while (true) {
16         string s3; getline(cin, s3);
17         if (s3 == "PARO") break;
18         cout<<"A"<<endl;
19     }
20 }

```