



## Notebook - Maratona de Programação

DSUm balão da cor sim cor não

### Contents

<b>1</b>	<b>Algoritmos</b>	<b>2</b>			
1.1	Busca Binaria . . . . .	2	5.7	Diametro Arvore Bfs . . . . .	11
1.2	Busca Binaria Double . . . . .	2	5.8	Diametro Arvore Dfs . . . . .	11
1.3	Busca Binaria Resposta . . . . .	2	5.9	Dijkstra . . . . .	12
1.4	Busca Ternaria . . . . .	3	5.10	Floyd Warshall . . . . .	12
1.5	Delta . . . . .	3	5.11	Kosaraju . . . . .	13
1.6	Fast Exponentiaton . . . . .	3	5.12	Kruskall . . . . .	13
1.7	Psum . . . . .	3	5.13	Lca . . . . .	14
1.8	Psum2d . . . . .	3	5.14	Topo Sort . . . . .	14
<b>2</b>	<b>DP</b>	<b>4</b>	<b>6</b>	<b>Math</b>	<b>15</b>
2.1	Dp . . . . .	4	6.1	Combinatoria . . . . .	15
2.2	Knapsack . . . . .	4	6.2	Divisibilidade . . . . .	15
2.3	Lis . . . . .	4	6.3	Divisores . . . . .	15
2.4	Mochila Iterativa . . . . .	5	6.4	Fatora . . . . .	15
2.5	Mochila Recursiva . . . . .	5	6.5	Mdc . . . . .	15
<b>3</b>	<b>ED</b>	<b>5</b>	6.6	Mmc . . . . .	15
3.1	Bitwise . . . . .	5	6.7	Pa . . . . .	15
3.2	Dsu . . . . .	5	6.8	Pg . . . . .	15
3.3	Lazy Seg . . . . .	6	6.9	Pollard-rho . . . . .	15
3.4	Merge Sort . . . . .	6	6.10	Primos . . . . .	16
3.5	Ordered Set . . . . .	7	<b>7</b>	<b>Strings</b>	<b>16</b>
3.6	Segtree 1 . . . . .	7	7.1	Suffix Array . . . . .	16
3.7	Segtree 2 . . . . .	7	7.2	Trie . . . . .	17
3.8	Segtree Lazy Propagation . . . . .	8	<b>8</b>	<b>Template</b>	<b>17</b>
<b>4</b>	<b>Geometria</b>	<b>9</b>	8.1	Template . . . . .	17
4.1	Geometria . . . . .	9	<b>9</b>	<b>zExtra</b>	<b>18</b>
<b>5</b>	<b>Grafos</b>	<b>9</b>	9.1	Getline . . . . .	18
5.1	Bellman Ford . . . . .	9			
5.2	Bfs . . . . .	10			
5.3	Binary Lifting . . . . .	10			
5.4	Bridges . . . . .	10			
5.5	Dfs . . . . .	11			
5.6	Dfs Tree . . . . .	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 const int MAX = 1e5+1;
9 const double EPS = 0.0000001;
10
11 vector<int> v(100001);
12 int n;
13 ll check(double x){
14     ll sum = 0;
15     for(int i=0; i<n; i++){
16         sum += (v[i]/x);
17     }
18     return sum;
19 }
20
21 int main(){
```

```
22     int k;
23     cin>>n>>k;
24
25     for(int i=0; i<n; i++)cin>>v[i];
26
27     double l=0.0000000, r=10000000.0000000;
28     double mid;
29     while(r-l>EPS){
30         mid = (double)((l + r)/2);
31         if (check(mid)>=k){
32             l=mid;
33         }
34         else{
35             r = mid;
36         }
37     }
38     cout<<fixed<<setprecision(7)<<mid<<endl;
39
40     return 0;
41 }
```

## 1.3 Busca Binaria Resposta

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 #define loop(i,a,n) for(int i=a; i < n; i++)
5
6 ll upperbound(ll maior, ll k, vector<ll> tabuas){
7     ll mid = 0, l = 0, r = maior, count = 300;
8     ll aux;
9     while((l < r) && (count--)){
10         aux = 0;
11         mid = (l+r)/2;
12         loop(i,0,tabuas.size()){
13             if(mid > 0){aux += (tabuas[i]/mid);}
14         }
15         if(aux >= k){l = mid;}
16         else{r = mid;}
17     }
18
19     ll aux_2 = 0;
20     loop(i,0,tabuas.size()){
21         aux_2 += (tabuas[i]/(mid+1));
22     }
23     if(aux_2 >= k){return mid+1;}
24
25     if(aux < k){
26         int aux_2 = 0;
27         loop(i,0,tabuas.size()){
28             if(mid - 1 > 0){aux_2 += (tabuas[i]/(mid
29             -1));}
30         }
31         if(aux_2 >= k){return mid-1;}
32     }
33
34     return mid;
35 }
36
37 int main(){
38     ios::sync_with_stdio(false);
39     cin.tie( NULL);
40     cout.tie(NULL);
41     int n; cin >> n;
42     ll k; cin >> k;
43     vector<ll> tabuas(n);
44     ll maior = 0;
45     loop(i,0,n){
46         cin >> tabuas[i];
47         maior = max(maior,tabuas[i]);
48     }
49     cout << upperbound(maior,k,tabuas);
50 }
```

## 1.4 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 }
```

## 1.5 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 }
```

## 1.6 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) { // iterativo
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 }
```

## 1.7 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 }
```

## 1.8 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     for(int i=1; i<MAX; i++){
21         for(int j=1; j<MAX; j++){
22             psum[i][j] += psum[i][j-1];
23         }
24     }
25
26     for(int i=0; i<q; i++){
27         int x1, y1, x2, y2;
28         cin>>x1>>y1>>x2>>y2;
29         x2--; y2--;
30
31         int soma = psum[x1][y1] + psum[x2][y2] -
32         psum[x2][y1] - psum[x1][y2];
33         cout<<soma<<endl;
34     }
35     return 0;
36 }

```

## 2 DP

### 2.1 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 }

```

### 2.2 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
7 );cout.tie(NULL);
8 #define pb(x) push_back(x);
9 #define pii pair<int,int>
10 const int N = 1e3+5;
11
12 int n, t;
13 int tab[N][N];
14 bool pegou[N][N];
15 vector<pair<int,int>> v;
16
17 vector<int> resposta;
18
19 int dp(int idx, int dias){
20     if(idx >= n) return 0;
21     if(tab[idx][dias] != -1) return tab[idx][dias];
22
23     int pega=0;
24     if(dias+v[idx].first <= t){
25         pega = dp(idx+1, dias+v[idx].first)+v[idx].
26         second;
27     }
28
29     int npega = dp(idx+1, dias);
30
31     if(pega>npega) pegou[idx][dias] = true;
32
33     return tab[idx][dias] = max(pega, npega);
34 }
35
36 int32_t main(){
37     memset(tab, -1, sizeof(tab));
38     cin>>n>>t;
39     for(int i=0; i<n; i++){
40         int ti, di;
41         cin>>ti>>di;
42
43         v.push_back({ti, di});
44     }
45     dp(0, 0);
46     int i = 0, j = 0;
47     vector<int> ans;
48     // retornar os valores
49     while(i < n){
50         if(pegou[i][j]){
51             j += v[i].first;
52             ans.push_back(i+1);
53         }
54         i++;
55     }
56     cout<<ans.size()<<endl;
57     for(int i=0; i<ans.size(); i++){
58         cout<<ans[i]<<" ";
59     }
60 }

```

### 2.3 Lis

```

1 // Longest increase sequence
2 // O(nlogn)
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 }

```

```

10 // size of the lis
11 int ans = S.size();
12
13 ////////////////////////////////////////////////// see that later
14 // https://codeforces.com/blog/entry/13225?#comment
15 // -180208
16 vi LIS(const vi &elements){
17     auto compare = [&](int x, int y) {
18         return elements[x] < elements[y];
19     };
20     set< int, decltype(compare) > S(compare);
21
22     vi previous( elements.size(), -1 );
23     for(int i=0; i<int( elements.size() ); ++i){
24         auto it = S.insert(i).first;
25         if(it != S.begin())
26             previous[i] = *prev(it);
27         if(*it == i and next(it) != S.end())
28             S.erase(next(it));
29     }
30
31     vi answer;
32     answer.push_back( *S.rbegin() );
33     while ( previous[answer.back()] != -1 )
34         answer.push_back( previous[answer.back()] );
35     reverse( answer.begin(), answer.end() );
36     return answer;
37 }

```

## 2.4 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     printf("%lld\n", ans);
32 }

```

## 2.5 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])
25             + v[i]);
26
27     return dp[i][P];
28 }
29
30 int main() {
31     int C; scanf("%d %d", &n, &C);
32     for(int i = 1; i <= n; i++)
33         scanf("%d %d", &p[i], &v[i]);
34     printf("%lld\n", solve(1, C));
35 }

```

## 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
19 // binario
20 void possibleSubsets(char S[], int N) {
21     for(int i = 0; i < (1 << N); ++i) { // i = [0, 2^
22         N - 1]
23         for(int j = 0; j < N; ++j)
24             if(i & (1 << j)) // se o j-esimo bit de
25                 i esta setado, printamos S[j]
26                 cout << S[j] << " ";
27         cout << endl;
28     }
29 }

```

### 3.2 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);
22         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
42 int main(){
43     DSU dsu(10);
44     return 0;
45 }

```

### 3.3 Lazy Seg

```

1 //Seg Tree. Considering I = 1, L = 0 and R = N-1; I
  is the first index in st.
2 class SegTree{
3     private:
4         ll st[4*MAX], lazy[4*MAX];
5
6         ll merge(ll a, ll b){
7             return min(a,b);
8         }
9
10        void push(int i, long long x = 0){
11            st[i] += (lazy[i]+x);
12            if(2*i < 4*MAX) lazy[2*i] += (lazy[i]+x);
13            if(2*i+1 < 4*MAX) lazy[2*i+1] += (lazy[i]
14            ]+x);
15            lazy[i] = 0;
16        }
17    public:
18        void build(int i = 1, int l = 0, int r = n-1)
19        {
20            if(l == r){
21                st[i] = a[l]; //leaf node.
22                lazy[i] = 0;
23            }
24            else{
25                int mid = (r+l)/2;

```

```

25                lazy[i] = 0;
26                build(2*i, l, mid);
27                build(2*i + 1, mid+1, r);
28                st[i] = merge(st[2*i], st[2*i + 1]);
29            }
30        }
31        return;
32    }
33
34    ll query(int l, int r, int i = 1, int auxl =
35    0, int auxr = n-1){
36        if(l <= auxl && r >= auxr){ //total
37            overlap.
38                if(lazy[i]){
39                    push(i);
40                }
41                return st[i];
42            }
43            else if(auxr < l || auxl > r){ //no
44                overlap.
45                    return LLINF;
46            }
47            else{ //partial overlap
48                int auxmid = (auxr+auxl)/2;
49                push(i);
50                return merge(query(l, r, 2*i, auxl,
51                auxmid), query(l, r, 2*i + 1, auxmid+1, auxr));
52            }
53        }
54
55        void update(int l, int r, ll x, int i = 1,
56        int auxl = 0, int auxr = n-1){
57            if(l <= auxl && auxr <= r){ //total
58                overlap.
59                    push(i,x);
60            }
61            else if(auxr < l || auxl > r){ //no
62                overlap.
63                    return;
64            }
65            else{ //partial overlap
66                int auxmid = (auxr+auxl)/2;
67                update(l, r, x, 2*i, auxl, auxmid);
68                update(l, r, x, 2*i + 1, auxmid+1,
69                auxr);
70                st[i] = merge(st[2*i],st[2*i+1]);
71            }
72        }
73    }
74 };
75
76 int main(){
77     int q; cin >> n >> q;
78     SegTree seg;
79     for(int i = 0; i < n; i++){
80         cin >> a[i];
81     }
82     seg.build();
83     for(int i = 0; i < q; i++){
84         int op; cin >> op;
85         if(op == 1){
86             int l, r, x; cin >> l >> r >> x;
87             seg.update(l-1,r-1,x);
88         }
89         else{
90             int k; cin >> k;
91             cout << seg.query(k-1,k-1) << "\n";
92         }
93     }
94 }

```

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

```

### 3.5 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;

```

### 3.6 Segtree 1

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 class SegTree{
5     vector<int> seg;
6     int size;
7     int elem_neutro = 0;
8
9     int merge(int a, int b){
10         return a^b;
11     }
12     void update(int idx, int val, int stl, int str, int no){
13         if(stl == idx and str==idx){
14             seg[no] = val;
15             return;
16         }

```

```

17         if(stl>idx or str<idx) return;
18
19         int mid = (stl+str)/2;
20         update(idx, val, stl, mid, 2*no);
21         update(idx, val, mid+1, str, 2*no+1);
22
23         seg[no] = merge(seg[2*no], seg[2*no+1]);
24     }
25
26     int query(int l, int r, int stl, int str, int no)
27     {
28         if(str<l or stl>r) return elem_neutro;
29         if(stl>=l and str<=r) return seg[no];
30
31         int mid = (stl+str)/2;
32         int x = query(l, r, stl, mid, 2*no);
33         int y = query(l, r, mid+1, str, 2*no+1);
34         return merge(x, y);
35     }
36     public:
37     SegTree(int n): seg(4*n, 0){size=n;}
38     int query(int l, int r){return query(l, r, 0, size-1, 1);}
39     void update(int idx, int val){update(idx, val, 0, size-1, 1);}
40     void out(){for(int i=0; i<size; i++){cout<<query(i, i)<<" ";cout<<endl;}}
41 };
42
43 int32_t main(){
44     int n, q;
45     cin>>n>>q;
46     SegTree seg(n);
47     for(int i=0; i<n; i++){
48         int x; cin>>x;
49         seg.update(i,x);
50     }
51     for(int i=0; i<q; i++){
52         int a, b;
53         cin>>a>>b;
54
55         cout<<seg.query(a-1, b-1)<<endl;
56     }
57     return 0;
58 }

```

### 3.7 Segtree 2

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 //SEG-TREES are used when we want to apply queries in
4 //intervals(segmentes) of a vector, such as
5 //getting the min value, getting the max value or
6 //getting the sum of this segment, and also doing
7 //updates to these segments in a efficient O
8 //complexity.
9
10 //It takes O(n) to build a segment tree.
11 //It takes O(log n) to answer a query and to update a
12 //segment.
13 //Also, an important thing to notice is that we dont
14 //actually implement a tree, we do it in the form
15 //of an array.
16 //Queries are done in the same way despite of the seg
17 //tree type, therefore what actually changes is
18 //how we build the tree considering what we want
19 //such as: max, min, sum...
20
21 const int INF = 0x3f3f3f3f;
22 const int MAX = 200005;
23 int n;
24 int a[MAX];

```

```

16
17 // Min Seg Tree. Considering L = 1 and R = N;
18 class MinSegTree{
19     private:
20         int st[4*MAX];
21     public:
22         void build(int i, int l, int r){
23             if(l == r){
24                 st[i] = a[l]; //leaf node.
25             }
26             else{
27                 int mid = (r+l)/2;
28                 build(2*i, l, mid);
29                 build(2*i + 1, mid+1, r);
30                 st[i] = min(st[2*i], st[2*i + 1]); //
parent node.
31             }
32             return;
33         }
34
35         int getmin(int i, int auxl, int auxr, int l,
int r){
36             if(l <= auxl && r >= auxr){ //total
overlap.
37                 return st[i];
38             }
39             else if(auxr < l || auxl > r){ //no
overlap.
40                 return INF;
41             }
42             else{ //partial overlap
43                 int auxmid = (auxr+auxl)/2;
44                 return min( getmin(2*i, auxl, auxmid,
l, r), getmin(2*i+1, auxmid+1, auxr, l, r));
45             }
46         }
47
48         void update(int i, int v, int x, int l, int r
){
49             if(l == r){
50                 st[i] = x;
51             }
52             else{
53                 int mid = (r+l)/2;
54                 if(v <= mid){
55                     update(2*i, v, x, l, mid);
56                 }
57                 else{
58                     update(2*i+1, v, x, mid+1, r);
59                 }
60                 st[i] = min(st[2*i],st[2*i + 1]);
61             }
62             return;
63         }
64 };
65
66 int main(){
67     int n, q; cin >> n >> q;
68     MinSegTree seg;
69     for(int i = 1; i < n+1; i++){
70         cin >> a[i];
71     }
72     seg.build(1,1,n);
73     for(int i = 0; i < q; i++){
74         int op; cin >> op;
75         if(op == 1){
76             int v, x; cin >> v >> x;
77             seg.update(1,v,x,1,n);
78         }
79         else{
80             int l, r; cin >> l >> r;
81             cout << seg.getmin(1,1,n,l,r) << "\n";
82         }

```

```

83     }
84 }

```

### 3.8 Segtree Lazy Propagation

```

1 const int MAX = 2e5+20;
2 vector<int> lazy(4*MAX, 0);
3 int tree[4*MAX], vet[MAX];
4 int n;
5
6 int merge(int a, int b){
7     return a + b; //seg de soma
8 }
9
10 void build(int l=0, int r=n-1, int no=1){
11     if(l==r){
12         tree[no] = vet[l];
13         lazy[no] = 0;
14         return;
15     }
16     int mid = (l+r)/2;
17     build(l, mid, 2*no);
18     build(mid+1, r, 2*no+1);
19
20     tree[no] = merge(tree[2*no], tree[2*no+1]);
21 }
22
23 void prop(int l, int r, int no){
24     if(lazy[no]!=0){
25         tree[no] += (r-l+1)*lazy[no]; //update de
soma
26         if(l!=r){
27             lazy[2*no] += lazy[no]; //update de soma
28             lazy[2*no+1] += lazy[no]; //update de
soma
29         }
30         lazy[no] = 0;
31     }
32 }
33
34 void update(int A, int B, int x, int l=0, int r=n-1,
int no=1){
35     prop(l, r, no);
36     // caso 1
37     if(B<l or r<A) return;
38     // caso 2
39     if(A<=l and r<=B){
40         lazy[no] += x; //update de soma
41         prop(l, r, no);
42         return;
43     }
44     // caso 3
45     int mid = (l+r)/2;
46
47     update(A, B, x, l, mid, 2*no);
48     update(A, B, x, mid+1, r, 2*no+1);
49
50     tree[no] = merge(tree[2*no], tree[2*no+1]);
51 }
52
53 int query(int A, int B, int l=0, int r=n-1, int no=1)
{
54     prop(l, r, no);
55     // caso 1
56     if(B<l or r<A) return 0;
57     // caso 2
58     if(A<=l and r<=B) return tree[no];
59     // caso 3
60     int mid = (l+r)/2;
61
62     return merge(query(A, B, l, mid, 2*no),
query(A, B, mid+1, r, 2*no+1));
63 }
64 }

```



```

65
66
67 int32_t main(){
68     int q;
69     cin>>n>>q;
70     for(int i=0;i<n;i++) cin >> vet[i];
71     build();
72     while(q--){
73         int opt; cin>>opt;
74         if(opt == 1){
75             int l, r, u;
76             cin>>l>>r>>u;
77             l--; r--;
78             update(l, r, u);
79         }else{
80             int k; cin>>k;
81             k--;
82             cout<< query(k, k)<<endl;
83         }
84     }
85     return 0;
86 }

```

## 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+
11         o.x, y+o.y}; }
12     point operator-(const point &o) const{ return {x-
13         o.x, y-o.y}; }
14     point operator*(ld t) const{ return {x*t, y*t}; }
15     point operator/(ld t) const{ return {x/t, y/t}; }
16     ld operator*(const point &o) const{ return x * o.
17         x + y * o.y; }
18     ld operator^(const point &o) const{ return x * o.
19         y - y * o.x; }
20 };
21
22 // line l(point(x1, y1), point(x2, y2));
23 struct line{
24     point a, b;
25     line(){}
26     line(point a, point b) : a(a), b(b){}
27 };
28
29 // ponto e em relacao a linha l
30 // counterclockwise
31 int ccw(line l, point e){
32     // -1=dir; 0=colinear; 1=esq;
33     point a = l.b-l.a, b=e-l.a;
34     ld tmp = a ^ b;
35     return (tmp > EPS) - (tmp < -EPS);
36 }
37
38 // se o ponto ta em cima da linha
39 bool isinseg(point p, line l){
40     point a = l.a-p, b = l.b-p;
41     return ccw(l, p) == 0 and (a * b) <= 0;
42 }
43
44 // se o seg de r intersecta o seg de s
45 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
44         true;
45
46     return (ccw(r, s.a)>0) != (ccw(r, s.b)>0) and
47         (ccw(s, r.a)>0) != (ccw(s, r.b)>0);
48 }
49
50 // area do poligono
51 ld area_polygon(vector<point> vp){
52     ld area = 0;
53     for(int i=1; i<vp.size()-1; i++){
54         area += (vp[0]-vp[i]) ^ (vp[0]-vp[i+1]);
55     }
56     return (abs(area)/2);
57 }
58
59 // localizacao do ponto no poligono
60 int point_polygon(vector<point> vp, point p){
61     // -1=outside; 0=boundary; 1=inside;
62     int sz = vp.size();
63     int inter = 0;
64     for(int i=0; i<sz; i++){
65         int j = (i+1)%sz;
66         line l(vp[i], vp[j]);
67         if(isinseg(p, l)) return 0;
68
69         if(vp[i].x <= p.x and p.x < vp[j].x and ccw(l
70             , p) == 1) inter++;
71         else if(vp[j].x <= p.x and p.x < vp[i].x and
72             ccw(l, p) == -1) inter++;
73     }
74
75     if(inter%2==0) return -1;
76     else return 1;
77 }

```

## 5 Grafos

### 5.1 Bellman Ford

```

1  /*
2  Algoritmo de busca de caminho minimo em um digrafo (
3  grafo orientado ou dirigido) ponderado, ou seja,
4  cujas arestas tem peso, inclusive negativo.
5
6  Acha ciclo negativo
7  O(V*E)
8  */
9
10 int d[MAX];
11 int parent[MAX];
12 vector<pair<int,int>> adj[MAX];
13
14 int32_t main(){ sws;
15     int n, m;
16     cin>>n>>m;
17     for(int i=1; i<=n; i++){
18         d[i] = LLINF;
19     }
20     for(int i=0; i<m; i++){
21         int a, b, c;
22         cin>>a>>b>>c;
23         adj[a].pb({b,c});
24     }
25     d[1] = 0;
26
27     int src_cycle = -1;
28     for(int j=1; j<=n and src_cycle; j++){
29         src_cycle = 0;
30         for(int u=1; u <= n; u++){
31             for(auto [v, w]: adj[u]){
32                 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 }

```

## 5.2 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 }

```

## 5.3 Binary Lifting

```

1 vector<int> adj[MAXN];
2 const int LOG = 30;
3 int up[MAXN][LOG], parent[MAXN];
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 }

```

## 5.4 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);
7 #define pb push_back
8 const int MAX = 1e5+5;
9
10 vector<int> adj[MAX];
11 int timer=0;
12 int low[MAX], tin[MAX];
13 bool bridge=false;
14 bool visited[MAX];
15
16 void dfs(int v, int p = -1) {
17     visited[v] = true;
18     tin[v] = low[v] = timer++;
19     for (int to : adj[v]) {
20         if (to == p) continue;
21         if (visited[to]) {
22             low[v] = min(low[v], tin[to]);
23         } else {
24             dfs(to, v);
25             low[v] = min(low[v], low[to]);
26             if (low[to] > tin[v]){
27                 //IS_BRIDGE(v, to);
28             }
29         }
30     }
31 }
32
33 int32_t main(){ sws;
34     int n, m;
35     cin>>n>>m;
36
37     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 }

```

## 5.5 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 }

```

## 5.6 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 (
6 // ou => u e sobem pra cima de u; backedges[u] == 0
7 // => u eh ponte
8 void dfs(int u, int p) {
9     if(vis[u]) return;
10    pai[u] = p;
11    h[u] = h[p]+1;
12    vis[u] = 1;
13
14    for(auto v : g[u]) {
15        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 }

```

## 5.7 Diametro Arvore Bfs

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 typedef long long ll;
5 typedef vector<int> vi;
6 typedef pair<int,int> pii;
7 const int MAX = 1e5+10;
8 const ll MOD = 1e9+5;
9
10 vector<int> adj[MAX];
11
12 pair<int, int> bfs(int s, int N){
13
14     vi dist(N + 1, MAX); dist[s] = 0;
15     queue<int> q; q.push(s);
16     int last = s;
17
18     while(!q.empty()){
19         auto u = q.front(); q.pop();
20         last = u;
21
22         for(auto v: adj[u]){
23             if(dist[v]==MAX){
24                 dist[v]=dist[u]+1;
25                 q.push(v);
26             }
27         }
28     }
29
30     return {last, dist[last]};
31 }
32
33 int diameter(int N){
34     auto [v, _] = bfs(1, N);
35     auto [w, D] = bfs(v, N);
36
37     return D;
38 }

```

## 5.8 Diametro Arvore Dfs

```

1 // DIAMETRO ARVORE - DFS
2
3 #include <bits/stdc++.h>
4 using namespace std;
5
6 typedef long long ll;
7 typedef vector<int> vi;
8 typedef pair<int,int> pii;
9 const int MAX = 1e5+10;
10 const ll MOD = 1e9+5;
11 const ll LLINF = 0x3f3f3f3f3f3f3f3f;
12
13 int to_leaf[MAX];
14 int max_length[MAX];
15 int dist[MAX];
16
17 vector<int> adj(MAX);

```

```

18 /*
19 void dfs(int u, int p, vector<int> &dist){
20     for(auto [v, w] : adj[u]){
21         if(v!=p){
22             dist[v] = dist[u] + w;
23             dfs(v, u, dist);
24         }
25     }
26 }
27
28 int solve(int n){
29     vector<int> dist(n+1, 0);
30
31     dfs(0, -1, dist);
32
33     auto v = (int)(max_element(dist.begin(), dist.end()
34     ()) - dist.begin());
35
36     dist[v] = 0;
37     dfs(v, -1, dist);
38
39     return *max_element(dist.begin(), dist.end());
40 }*/
41 void dfs(int u, int p){
42     vi ds;
43
44     for(auto v: adj[u]){
45         if(v==p) continue;
46
47         dfs(v, u);
48         ds.pb(to_leaf[v]);
49     }
50     sort(ds.begin(), ds.end());
51
52     to_leaf[u] = ds.empty() ? 0 : ds.back() + 1;
53
54     auto N = ds.size();
55
56     switch(N){
57         case 0:
58             max_length[u]=0;
59             break;
60         case 1:
61             max_length[u] = ds.back() + 1;
62             break;
63         default:
64             max_length[u] = ds[N-1] + ds[N-2] + 2;
65     }
66 }
67
68 int diameter(int root, int N){
69     dfs(root, 0);
70
71     int d=0;
72
73     for(int u=1; u<=N; u++){
74         d= max(d, max_length[u]);
75     }
76 }

```

## 5.9 Dijkstra

```

1 // Acha o menor caminho de um ponto inicial para
2 // todos os outros
3 // Complexidade: O(NlogN)
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     }

```

## 5.10 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 = 100000000; // 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 comeca, termina, custo;
29     cin>>comeca>>termina>>custo;
30
31     // grafo direcionado
32     dist[comeca][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;
40             // relaxar distancia de i pra j
41             dist[i][j] = min(dist[i][j], dist[i][k]
42                             + dist[k][j]);
43         }
44     }
45     return 0;
46 }

```

## 5.11 Kosaraju

```

1 // Acha componentes fortemente conexas
2 // ou seja, que tem caminho entre todos os pares de
  vertices
3 // O(n+m)
4 bool visited[MAX];
5 int n;
6 vector<int> adj[MAX];
7 vector<int> adj2[MAX]; // grafo inverso
8 stack<int> st;
9 int conex[MAX]; // conexo de cada vertice
10 void dfs(int u){
11     visited[u] = true;
12     for(auto v : adj[u]){
13         if(!visited[v]) dfs(v);
14     }
15     st.push(u);
16 }
17 // strongly connected component
18 void scc(int u, int c){
19     // cout<<u<<" ";
20     visited[u] = true;
21     conex[u] = c;
22     for(auto v : adj2[u]){
23         if(!visited[v]) scc(v, c);
24     }
25 }
26
27 void kosaraju(){
28     for(int i=1; i<=n; i++) visited[i]=false;
29     for(int i=1; i<=n; i++) if(!visited[i]) dfs(i);
30
31     for(int i=1; i<=n; i++) visited[i]=false;
32     while(!st.empty()){
33         int u = st.top(); st.pop();
34         if(!visited[u]) scc(u, u);
35     }
36 }
37 int main(){
38     int m;
39     cin>>n>>m;
40
41     for(int i=0; i<m; i++){
42         int a, b;
43         cin>>a>>b;
44         adj[a].pb(b);
45         adj2[b].pb(a);
46     }

```

```

47
48     kosaraju();
49
50     return 0;
51 }

```

## 5.12 Kruskall

```

1 /*
2 Busca uma arvore geradora minima para um grafo conexo
  com pesos.
3 */
4
5 #include <iostream>
6 #include <algorithm>
7
8 using namespace std;
9
10 struct t_aresta{
11     int dis;
12     int x, y;
13 };
14
15 bool comp(t_aresta a, t_aresta b){ return a.dis < b.
    dis; }
16
17 //-----
18 #define MAXN 50500
19 #define MAXM 200200
20
21 int n, m; // numero de vertices e arestas
22 t_aresta aresta[MAXM];
23
24 // para o union find
25 int pai[MAXN];
26 int peso[MAXN];
27
28 // a arvore
29 t_aresta mst[MAXM];
30 //-----
31
32 // funcoes do union find
33 int find(int x){
34     if(pai[x] == x) return x;
35     return pai[x] = find(pai[x]);
36 }
37
38 void join(int a, int b){
39
40     a = find(a);
41     b = find(b);
42
43     if(peso[a] < peso[b]) pai[a] = b;
44     else if(peso[b] < peso[a]) pai[b] = a;
45     else{
46         pai[a] = b;
47         peso[b]++;
48     }
49 }
50
51
52 int main(){
53
54     // ler a entrada
55     cin >> n >> m;
56
57     for(int i = 1; i <= m; i++){
58         cin >> aresta[i].x >> aresta[i].y >> aresta[i]
59         ].dis;
60
61
62     // inicializar os pais para o union-find

```

```

63     for(int i = 1; i <= n; i++) pai[i] = i;
64
65     // ordenar as arestas
66     sort(aresta+1, aresta+m+1, comp);
67
68     int size = 0;
69     for(int i = 1; i <= m; i++){
70
71         if( find(aresta[i].x) != find(aresta[i].y) ){
72             // se estiverem em componentes distintas
73             join(aresta[i].x, aresta[i].y);
74
75             mst[++size] = aresta[i];
76         }
77     }
78
79     // imprimir a MST
80     for(int i = 1; i < n; i++) cout << mst[i].x << " "
81     << mst[i].y << " " << mst[i].dis << "\n";
82     return 0;
83 }

```

### 5.13 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 //ll weight[SIZE];
15 int pai[SIZE];
16 vector<pair<int,int>> adj[SIZE];
17 int up[SIZE][LOG];
18
19 //
20 void dfs(int u, int p) {
21     for(auto edge : adj[u]) {
22         int v, w;
23         tie(v, w) = edge;
24         if(v != p){
25             up[v][0] = u;
26             //weight[v] = weight[u] + w;
27             depth[v] = depth[u] + 1;
28             for(int i=1; i<LOG; i++){
29                 up[v][i] = up[ up[v][i-1] ][i-1];
30             }
31             dfs(v, u);
32         }
33     }
34 }
35
36 int lca(int a, int b) {
37     if(depth[a] < depth[b]) swap(a,b);
38     int k = depth[a] - depth[b];
39     for(int i=0; i<LOG; i++){
40         if(k & (1 << i)){
41             a = up[a][i];
42         }
43     }
44     if(a == b) return a;
45     for (int i = LOG-1; i >= 0; i--) {

```

```

46         if(up[a][i] != up[b][i]) {
47             a = up[a][i];
48             b = up[b][i];
49         }
50     }
51     return up[a][0];
52 }
53
54 ll dist(int u, int v){
55     return depth[u] + depth[v] - 2*depth[lca(u,v)];
56     // return weight[u] + weight[v] - 2*weight[lca(u,v)
57     ];
58 }
59 int main() {
60     int n; cin>>n;
61
62     for(int i=0; i<n-1; i++){
63         int x, y, z;
64         cin>>x>>y>>z;
65         adj[x].push_back({y, z});
66         adj[y].push_back({x, z});
67     }
68     // raiz
69     dfs(1, 0);
70
71     int q; cin>>q;
72     while(q--){
73         int a, b, c;
74         cin>>a>>b>>c;
75         long long x = dist(a, b) + dist(b, c);
76         cout<<x<<endl;
77     }
78 }

```

### 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 }

```

### 6.2 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

```

### 6.3 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 }

```

### 6.4 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 }

```

### 6.5 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)

```

### 6.6 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 }

```

### 6.7 Pa

```

1 // Termo Geral
2 // An = A1 + (n-1)*d
3
4 // Soma
5 // Sn = (n/2)(2*A1+(n-1)*d)
6
7 // Soma de 1 a N
8 int pa(int k){
9     return (k*(k+1))/2;
10 }

```

### 6.8 Pg

```

1 // Termo Geral
2 // An = A1 * r^(n-1)
3
4 // Soma
5 // (A(r^n(n)-1))/(r-1)

```

### 6.9 Pollard-rho

```

1 // O(sqrt(N) * logN)
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);

```

```

24         if (x == 1 or x == n - 1 or a % n == 0)
25             continue;
26         for (int j = 0; j < r - 1; j++) {
27             x = mul(x, x, n);
28             if (x == n - 1) break;
29         }
30         if (x != n - 1) return 0;
31     }
32     return 1;
33 }
34
35 ll rho(ll n) {
36     if (n == 1 or prime(n)) return n;
37     auto f = [n](ll x) {return mul(x, x, n) + 1;};
38
39     ll x = 0, y = 0, t = 30, prd = 2, x0 = 1, q;
40     while (t % 40 != 0 or __gcd(prd, n) == 1) {
41         if (x==y) x = ++x0, y = f(x);
42         q = mul(prd, abs(x-y), n);
43         if (q != 0) prd = q;
44         x = f(x), y = f(f(y)), t++;
45     }
46     return __gcd(prd, n);
47 }
48
49 vector<ll> fact(ll n) {
50     if (n == 1) return {};
51     if (prime(n)) return {n};
52     ll d = rho(n);
53     vector<ll> l = fact(d), r = fact(n / d);
54     l.insert(l.end(), r.begin(), r.end());
55     return l;
56 }
57
58 int main(){
59     set<ll> primes;
60     int M, N, K; cin >> M >> N >> K;
61     loop(i,0,N){
62         cin >> a[i];
63         vector<ll> aprimes = fact(a[i]);
64         for(auto prime : aprimes){
65             primes.insert(prime);
66         }
67     }
68     int m, n, d;
69     loop(i,0,K) cin >> m >> n >> d;
70     for(auto prime : primes){
71         cout << prime << " ";
72     }
73 }
74 }

```

## 6.10 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

```

```

19 bool is_prime(int num){
20     for(int i = 2; i*i<= num; i++) {
21         if(num % i == 0) {
22             return false;
23         }
24     }
25     return true;
26 }

```

## 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
23     for (int k = 0; k < n; k ? k *= 2 : k++) {
24         vector<int> nsa(sa), nra(n), cnt(N);
25
26         for (int i = 0; i < n; i++) nsa[i] = (nsa[i]-
27             k+n)%n, cnt[ra[i]]++;
28         for (int i = 1; i < N; i++) cnt[i] += cnt[i
29             -1];
30         for (int i = n-1; i+1; i--) sa[--cnt[ra[nsa[i
31             ]]]] = nsa[i];
32         for (int i = 1, r = 0; i < n; i++) nra[sa[i]]
33             = r += ra[sa[i]] !=
34             ra[sa[i-1]] or ra[(sa[i]+k)%n] != ra[(sa[
35             i-1]+k)%n];
36         ra = nra;
37         if (ra[sa[n-1]] == n-1) break;
38     }
39     return vector<int>(sa.begin()+1, sa.end());
40 }
41
42 vector<int> kasai(string s, vector<int> sa) {
43     int n = s.size(), k = 0;
44     vector<int> ra(n), lcp(n);
45     for (int i = 0; i < n; i++) ra[sa[i]] = i;
46
47     for (int i = 0; i < n; i++, k -= !!k) {
48         if (ra[i] == n-1) { k = 0; continue; }
49         int j = sa[ra[i]+1];
50         while (i+k < n and j+k < n and s[i+k] == s[j+
51             k]) k++;
52         lcp[ra[i]] = k;
53     }
54     return lcp;
55 }

```



```

50 }
51
52
53 int32_t main(){
54     sws;
55     string s;
56     cin>>s;
57
58     vector<int> suf = suffix_array(s);
59     vector<int> lcp = kasai(s, suf);
60
61     ll ans = 0;
62     for(int i=0; i<s.size(); i++){
63         if(islower(s[suf[i]])){
64             int sz = s.size()-suf[i];
65             ans += (sz - lcp[i]);
66         }
67     }
68     cout<<ans<<endl;
69 }

```

## 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];

```

```

50     }
51     return cur;
52 }
53
54 int count_pref(string s){
55     return pref[find(s)];
56 }
57
58 Trie(int qtd_char_=26, char c_min_ = 'a', int
59     max_size_=MAX):
60     c_min(c_min_), qtd_char(qtd_char_), max_size(
61     max_size_){
62     trie.resize(max_size, vector<int>(qtd_char));
63     pref.resize(max_size);
64     end.resize(max_size);
65 }
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 }

```

## 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
10 #define int long long
11 #define endl "\n"
12 #define loop(i,a,n) for(int i=a; i < n; i++)
13 #define input(x) for (auto &it : x) cin >> it
14 #define pb push_back
15 #define all(x) x.begin(), x.end()
16 #define ff first
17 #define ss second
18 #define mp make_pair
19 #define TETO(a, b) ((a) + (b-1))/(b)
20 #define dbg(msg, x) cout << msg << " = " << x << endl
21 #define print(x,y) loop(i,0,y){cout << x[i] << " ";}
22     cout << "\n";
23
24 typedef long long ll;
25 typedef long double ld;
26 typedef vector<int> vi;
27 typedef pair<int,int> pii;
28 typedef priority_queue<int, vector<int>, greater<int>
29     >> pqi;
30
31 const ll MOD = 1e9+7;
32 const int MAX = 1e4+5;
33 const ll LLINF = 0x3f3f3f3f3f3f3f3f;
34 const double PI = acos(-1);
35
36 int32_t main(){ sws;
37
38     return 0;
39 }

```

## 9 zExtra

### 9.1 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 }
```