Notebook - Maratona de Programação

De galinha em galinha, o grão enche o papo

Contents

1 Algoritmos

1.1 Busca Binaria

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

bool check(int valor, int x) {
    return valor <= x;
}

int bb(int a, int b, int x){
    int l = a;
    int r = b;
    while (l < r) {
        int mid = (l + r) / 2;
        if (check(mid, x)) r = mid;
        else l = mid + 1;
    }
    return l;
}

bool check(int valor) {
    return valor <= 10;
}

int bb_menor(int a, int b){
    int l = a;
    int r = b;
    while (l < r) {
        int mid = (l + r) / 2;
        if (check(mid)) r = mid;
        else l = mid + 1;
    }

    return l;
}


int bb_maior(int a, int b){
    int l = a;
    int r = b;
    while (l < r) {
        int mid = (l + r) / 2;
        if (!check(mid)) r = mid;
        else l = mid + 1;
    }
}
```

1.2 Busca Binaria Double

```cpp
//
// Complexidade : O(NlogN)

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

typedef long long ll;
const int MAX = 1e5+1;
const double EPS = 0.0000001;

vector<int> v(100001);
int n;
ll check(double x){
    ll sum = 0;
    for(int i=0; i<n; i++){
        sum += (v[i]/x);
    }
    return sum;
}

int main(){
    int k;
    cin>>n>>k;

    for(int i=0; i<n; i++)cin>>v[i];

    double l=0.0000000, r=10000000.0000000;
    double mid;
    while(r-l>EPS){
        mid = (double)((l + r)/2);
        if (check(mid)>=k){
            l=mid;
        }
        else{
            r = mid;
        }
    }
    cout<<fixed<<setprecision(7)<<mid<<endl;

    return 0;
}
```

1.3 Busca Binaria Resposta

```cpp
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
#define loop(i,a,n) for(int i=a; i < n; i++)

ll upperbound(ll maior, ll k, vector<ll> tabuas){
    ll mid = 0, l = 0, r = maior, count = 300;
    ll aux;
    while((l < r) && (count--)){
        aux = 0;
        mid = (l+r)/2;
        loop(i,0,tabuas.size()){
            if(mid > 0){aux += (tabuas[i]/mid);}
        }
        if(aux >= k){l = mid;}
        else{r = mid;}
    }

    ll aux_2 = 0;
    loop(i,0,tabuas.size()){
        aux_2 += (tabuas[i]/(mid+1));
    }
    if(aux_2 >= k){return mid+1;}

    if(aux < k){
        int aux_2 = 0;
        loop(i,0,tabuas.size()){
            if(mid - 1 > 0){aux_2 += (tabuas[i]/(mid
-1));}
        }
        if(aux_2 >= k){return mid-1;}
    }

    return mid;
}

int main(){
    ios::sync_with_stdio(false);
    cin.tie( NULL);
    cout.tie(NULL);
    int n; cin >> n;
    ll k; cin >> k;
    vector<ll> tabuas(n);
    ll maior = 0;
    loop(i,0,n){
        cin >> tabuas[i];
        maior = max(maior,tabuas[i]);
    }
    cout << upperbound(maior,k,tabuas);
}
```

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 Exponentiaton

```
1 int fast_exp(int base, int e){
2     if(e == 0) return 1;
3     if(e % 2) return base * fast_exp(base * base,e/2)
    ;
4     else return fast_exp(base * base, e/2);
5 }
```

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          }
21
22          for(int i=1; i<MAX; i++){
23              for(int j=1; j<MAX; j++){
24                  psum[i][j] += psum[i][j-1];
25              }
26          }
27
28          for(int i=0; i<q; i++){
29              int x1, y1, x2, y2;
30              cin>>x1>>y1>>x2>>y2;
31              x2--; y2--;
32
33              int soma = psum[x1][y1] + psum[x2][y2] -
     psum[x2][y1] - psum[x1][y2];
34              cout<<soma<<endl;
35          }
36      }
37      return 0;
38 }
```

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
     );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(pega>npega) pegou[idx][dias] = true;
30
31      return tab[idx][dias] = max(pega, 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 }
```

2.3 Lis

```
1 multiset<int> S;
2 for(int i=0;i<n;i++){
3      auto it = S.upper_bound(vet[i]); // low for inc
4      if(it != S.end())
5          S.erase(it);
6      S.insert(vet[i]);
7 }
8 // size of the lis
9 int ans = S.size();
10
11 ////////////////////////// see that later
12 // https://codeforces.com/blog/entry/13225?#comment
     -180208
13
14 vi LIS(const vi &elements){
15      auto compare = [&](int x, int y) {
16          return elements[x] < elements[y];
17      };
18      set< int, decltype(compare) > S(compare);
```

```
19
20     vi previous( elements.size(), -1 );
21     for(int i=0; i<int( elements.size() ); ++i){
22         auto it = S.insert(i).first;
23         if(it != S.begin())
24             previous[i] = *prev(it);
25         if(*it == i and next(it) != S.end())
26             S.erase(next(it));
27     }
28
29     vi answer;
30     answer.push_back( *S.rbegin() );
31     while ( previous[answer.back()] != -1 )
32         answer.push_back( previous[answer.back()] );
33     reverse( answer.begin(), answer.end() );
34     return answer;
35 }
```

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
32     printf("%lld\n", ans);
33 }
```

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
   itens para se considerar
12     if(vis[i][P]) return dp[i][P];
13     vis[i][P] = 1;
14
```

```
15     // primeira possibilidade, nao adicionar o
   elemento
16     dp[i][P] = solve(i+1, P);
17
18     // segunda possibilidade, adicionar o elemento.
19     // Lembrar de tirar o maximo com o valor ja
   calculado da primeira possibilidade
20     if(P >= p[i])
21         dp[i][P] = max(dp[i][P], solve(i+1, P - p[i])
    + v[i]);
22
23     return dp[i][P];
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   Bit

```
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 }
```

3.2   Dsu

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 const int MAX = 1e5+10;
5
6 int parent[MAX];
7 int sz[MAX];
8
9 void make(int v){
10     parent[v] = v;
11     sz[v] = 1;
12 }
13
14 int find(int v){
```

```
15    if (v == parent[v])
16        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 int main(){
32
33    return 0;
34 }
```

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]+x);
14            lazy[i] = 0;
15        }
16
17   public:
18       void build(int i = 1, int l = 0, int r = n-1){
19            if(l == r){
20                st[i] = a[l]; //leaf node.
21                lazy[i] = 0;
22            }
23            else{
24                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]);
    //parent node.
29            }
30            return;
31        }
32
33       ll query(int l, int r, int i = 1, int auxl =
    0, int auxr = n-1){
34            if(l <= auxl && r >= auxr){ //total
    overlap.
35                if(lazy[i]){
36                    push(i);
37                }
38                return st[i];
39            }
40            else if(auxr < l || auxl > r){ //no
    overlap.
41                return LLINF;
42            }
43            else{ //partial overlap
44                int auxmid = (auxr+auxl)/2;
45                push(i);
46                return merge(query(l, r, 2*i, auxl,
    auxmid), query(l, r, 2*i + 1, auxmid+1, auxr));
47            }
48        }
49
50       void update(int l, int r, ll x, int i = 1,
    int auxl = 0, int auxr = n-1){
51            if(l <= auxl && auxr <= r){ //total
    overlap.
52                push(i,x);
53            }
54            else if(auxr < l || auxl > r){ //no
    overlap.
55                return;
56            }
57            else{ //partial overlap
58                int auxmid = (auxr+auxl)/2;
59                update(l, r, x, 2*i, auxl, auxmid);
60                update(l, r, x, 2*i + 1, auxmid+1,
    auxr);
61                st[i] = merge(st[2*i],st[2*i+1]);
62            }
63        }
64 };
65
66 int main(){
67    int q; cin >> n >> q;
68    SegTree seg;
69    for(int i = 0; i < n; i++){
70        cin >> a[i];
71    }
72    seg.build();
73    for(int i = 0; i < q; i++){
74        int op; cin >> op;
75        if(op == 1){
76            int l, r, x; cin >> l >> r >> x;
77            seg.update(l-1,r-1,x);
78        }
79        else{
80            int k; cin >> k;
81            cout << seg.query(k-1,k-1) << "\n";
82        }
83    }
84 }
```

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

```
56
57     return 0;
58 }
```

3.6   Segtree 2

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

3.7   Segtree Lazy Propagation

```
1  #include <bits/stdc++.h>
2  #define ll long long
3
4  using namespace std;
5
6  const int MAX = 1e5; // tamanho maximo do vetor
7  const ll LLINF = 0x3f3f3f3f3f3f3f3f;
8
9  // End Template //
10
11  vector<ll> lazy(4*MAX, 0);
12  ll tree[4*MAX], vet[MAX];
13  int N;
14
15  ll merge(ll a, ll b){
16      return a + b;
17  }
18
19  void build(int l=0, int r=N-1, int no=1){
20      if(l==r){
21          tree[no] = vet[l];
22          return;
23      }
24      int mid = (l+r)/2;
25      build(l, mid, 2*no);
26      build(mid+1, r, 2*no+1);
27
28      tree[no] = merge(tree[2*no], tree[2*no+1]);
29  }
30
31  void prop(int l, int r, int no){
32      if(lazy[no]!=0){
33          tree[no] = (r-l+1)*lazy[no];
34          if(l!=r){
35              lazy[2*no] = lazy[2*no+1] = lazy[no];
36          }
37          lazy[no] = 0;
38      }
39  }
40
41  void update(int A, int B, int x, int l=0, int r=N-1,
42          int no=1){
42      prop(l, r, no);
43      // caso 1
44      if(B<l or r<A) return;
45      // caso 2
46      if(A<=l and r<=B){
47          lazy[no] = x;
48          prop(l, r, no);
49          return;
50      }
51      // caso 3
52      int mid = (l+r)/2;
53
54      update(A, B, x, l, mid, 2*no);
55      update(A, B, x, mid+1, r, 2*no+1);
56
57      tree[no] = merge(tree[2*no], tree[2*no+1]);
58  }
59
60  ll query(int A, int B, int l=0, int r=N-1, int no=1){
61      prop(l, r, no);
62      // caso 1
63      if(B<l or r<A) return 0;
64      // caso 2
65      if(A<=l and r<=B) return tree[no];
66      // caso 3
67      int mid = (l+r)/2;
68
69      return merge(query(A, B, l, mid, 2*no),
70                   query(A, B, mid+1, r, 2*no+1));
71  }
72
73
74  int32_t main()
75  {
76
77      int Q, opt, a, b, l, r, k, idx;
78      cin >> N >> Q;
79      vector<int> vaux(N);
80      for(int i=0;i<N;i++){
81          cin >> vaux[i];
82          vet[i] = vaux[i];
83      }
84      for(int i=0; i<N; i++){
85          if(i==0)vet[i] = vaux[i];
86          else vet[i] = vet[i-1] + vaux[i];
87      }
88      build();
89
90      for(int i=0;i<Q;i++){
91          cin >> opt;
92          if(opt==1){ // update
93              cin >> idx >> k;
94              idx--;
95              int soma = -vaux[idx] + k;
96
97              vaux[idx] = k;
98              update(idx, N-1, soma);
99          }else{ // query
100             cin >> l >> r;
101             l--;r--; // indice indexado em 0
102             cout << query(l, r) << endl;
103         }
104     }
105     for(int i=0; i<N; i++){
106         cout<<vet[i]<<" ";
107     }
108     cout<<endl;
109
110     return 0;
```

4 Grafos

4.1 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  */
4
5  #include <bits/stdc++.h>
6  using namespace std;
7
8  // pode usar uma tuple
9  struct Edge {
10     // [de onde vem, pra onde vai, peso]
11     int from, to, custo;
12
13     Edge(int a=0, int b=0,int c=0 ){
14         from = a;
15         to=b;
16         custo = c;
17     }
18 };
19
20 int main(){
21
22     int n, m;
23     cin>>n>>m;
24     vector<Edge> arestas(m);
25
26     for(int i=0; i<m; i++){
27         int a, b, c;
28         cin>>a>>b>>c;
29         arestas[i] = Edge(a, b, c);
30     }
31
32     vector<int> distancia(n + 1, 100000000);
33     distancia[1]=0;
34     for(int i=0; i<n-1; i++){
35         for(auto aresta : arestas){
36             if (distancia[aresta.from] + aresta.custo
     < distancia[aresta.to]){
37                 distancia[aresta.to] = distancia[
     aresta.from] + aresta.custo;
38             }
39         }
40     }
41
42     for(int i=1; i<=n; i++){
43         cout<<"Distancia ate o vertice "<<i<<" "<<
     distancia[i]<<endl;
44     }
45
46     return 0;
47 }
```

4.2 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 }
```

4.3 Binary Lifting

```
1  vector<int> adj[MAX];
2  int up[MAX][30], parent[MAX];
3
4  void process(int n){
5      for(int v=1; v<=n; v++){
6          up[v][0]= parent[v];
7          for(int i=1; i<30; i++){
8              up[v][i] = up[ up[v][i-1] ][i-1];
9          }
10     }
11 }
12
13 int jump(int n, int k){
14     for(int i=0; i<30; i++){
15         if(k & (1 << i)){
16             n = up[n][i];
17         }
18     }
19     if(n == 0) return -1;
20     return n;
21 }
22
23 int32_t main(){
24
25     int n, q; cin>>n>>q;
26
27     parent[1] = 0;
28     for(int i=1; i<=n-1; i++){
29         int x;
30         cin>>x;
31         parent[i+1] = x;
32
33         adj[i+1].pb(x);
34         adj[x].pb(i+1);
35     }
36     process(n);
37     for(int i=0; i<q; i++){
38         int a, b;
39         cin>>a>>b;
40
41         cout<<(jump(a,b))<<endl;
42     }
43 }
```

4.4 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
       (NULL); cout.tie(NULL);
6  #define pb push_back
```

```cpp
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 }
```

4.5   Dfs

```cpp
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[x]){
19             dfs(i);
20         }
21     }
22 }
23
24 void dfsStack(int x){
25     stack<int> s;
26     s.push(x);
27     while(!s.empty()){
```

```cpp
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 }
```

4.6   Dfs Tree

```cpp
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 }
```

4.7   Diametro Arvore Bfs

```cpp
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 }
```

4.8 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
       ()) - dist.begin());
34
35     dist[v] = 0;
36     dfs(v, -1, dist);
37
38     return *max_element(dist.begin(), dist.end());
39 }*/
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 }
```

4.9 Dijkstra

```
1  // Acha o menor caminho de um ponto inicial para
      todos os outros
2  // Complexidade: O(NlogN)
3
4  #include <bits/stdc++.h>
5  using namespace std;
6  #define ll long long
7
8  const int N = 100005;
9  const ll oo = 1e18;
10
11 ll d[N]; // vetor onde guardamos as distancias
12
13 int n; // numeros de vertices
14 vector<pair<int, ll>> g[N];
15
16 void dijkstra(int start){
17     for(int u = 1; u <= n; u++)
18         d[u] = oo;
19
20     priority_queue<pair<ll, int>, vector<pair<ll, int
       >>,
21     greater<pair<ll, int>> > pq;
22
23     d[start] = 0;
24     pq.emplace(d[start], start);
25
26     ll dt, w;
27     int u, v;
28     while(!pq.empty()){
29         tie(dt, u) = pq.top(); pq.pop();
30         if(dt > d[u]) continue;
31         for(auto edge : g[u]){
32             tie(v, w) = edge;
33             if(d[v] > d[u] + w){
34                 d[v] = d[u] + w;
35                 pq.emplace(d[v], v);
36             }
37         }
38     }
39 }
40
41 int main(){
42
43     // le o input, qnt de vertices, arestas
44     // e vertice inicial(start)
45     int start = 0; // inicial
46     dijkstra(start);
47
48     for(int u = 1; u <= n; u++){
49         printf("Distancia de %d para %d: %lld\n",
       start, u, d[u]);
50     }
51
52 }
```

4.10 Floyd Warshall

```
1  /*
2  Algoritmo de caminho mais curto com todos os pares.
3  Complexidade: O(³N)
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
         por k;
39
40                 // relaxar distancia de i pra j
41                 dist[i][j] = min(dist[i][j], dist[i][
         k] + dist[k][j]);
42             }
43         }
44     }
45         return 0;
46 }
```

4.11 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
53 int main(){
54
55     // ler a entrada
56     cin >> n >> m;
57
58     for(int i = 1;i <= m;i++)
59         cin >> aresta[i].x >> aresta[i].y >> aresta[i
      ].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) ){
      // se estiverem em componentes distintas
72             join(aresta[i].x, aresta[i].y);
73
74             mst[++size] = aresta[i];
75         }
76
77     }
78
79     // imprimir a MST
80     for(int i = 1;i < n;i++) cout << mst[i].x << " "
      << mst[i].y << " " << mst[i].dis << "\n";
81     return 0;
82 }
```

4.12 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   Math

5.1   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 }
```

5.2   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 algarimos eh divisivel por 9
9 // 10 -> se o ultimo algarimo eh 0
```

5.3   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 }
```

5.4   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 }
```

5.5   Mdc

```
1 // Greatest common divisor / MDC
2
```

```cpp
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)
```

5.6   Mmc

```cpp
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  }
```

5.7   Pa

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

5.8   Pg

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

5.9   Pollard-rho

```cpp
1  ll a[MAX];
2
3  ll mul(ll a, ll b, ll m){
4      ll ret = a*b - (ll)((long double)1/m*a*b+0.5)*m;
5      return ret < 0 ? ret+m : ret;
6  }
7
8  ll pow(ll x, ll y, ll m) {
9      if (!y) return 1;
10     ll ans = pow(mul(x, x, m), y/2, m);
11     return y%2 ? mul(x, ans, m) : ans;
12 }
13
14 bool prime(ll n) {
15     if (n < 2) return 0;
16     if (n <= 3) return 1;
17     if (n % 2 == 0) return 0;
18
19     ll r = __builtin_ctzll(n - 1), d = n >> r;
20     for (int a : {2, 325, 9375, 28178, 450775,
       9780504, 795265022}) {
21         ll x = pow(a, d, n);
22         if (x == 1 or x == n - 1 or a % n == 0)
       continue;
23
24         for (int j = 0; j < r - 1; j++) {
25             x = mul(x, x, n);
26             if (x == n - 1) break;
27         }
28         if (x != n - 1) return 0;
29     }
30     return 1;
```

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

5.10   Primos

```cpp
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 }
```

6 Strings

6.1 Suffix Array

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

#define ll long long
#define sws ios::sync_with_stdio(false);cin.tie( NULL
    );cout.tie(NULL);
#define print(x) for (auto &it : x) cout<<it<<' ';<<
    cout<<endl;
#define loop(i,a,n) for(int i=a; i < n; i++)
#define pb(x) push_back(x);
#define vi vector<int>
#define mp(x,y) make_pair(x,y)
#define pii pair<int,int>
#define pqi priority_queue<int, vector<int>, greater<
    int>>
const ll MOD = 1e9+7;
const int INF = 0x3f3f3f3f;
const ll LLINF = 0x3f3f3f3f3f3f3f3f;

vector<int> suffix_array(string s) {
    s += "$";
    int n = s.size(), N = max(n, 260);
    vector<int> sa(n), ra(n);
    for (int i = 0; i < n; i++) sa[i] = i, ra[i] = s[
    i];

    for (int k = 0; k < n; k ? k *= 2 : k++) {
        vector<int> nsa(sa), nra(n), cnt(N);

        for (int i = 0; i < n; i++) nsa[i] = (nsa[i]-
    k+n)%n, cnt[ra[i]]++;
        for (int i = 1; i < N; i++) cnt[i] += cnt[i
    -1];
        for (int i = n-1; i+1; i--) sa[--cnt[ra[nsa[i
    ]]]] = nsa[i];

        for (int i = 1, r = 0; i < n; i++) nra[sa[i]]
     = r += ra[sa[i]] !=
            ra[sa[i-1]] or ra[(sa[i]+k)%n] != ra[(sa[
    i-1]+k)%n];
        ra = nra;
        if (ra[sa[n-1]] == n-1) break;
    }
    return vector<int>(sa.begin()+1, sa.end());
}

vector<int> kasai(string s, vector<int> sa) {
    int n = s.size(), k = 0;
    vector<int> ra(n), lcp(n);
    for (int i = 0; i < n; i++) ra[sa[i]] = i;

    for (int i = 0; i < n; i++, k -= !!k) {
        if (ra[i] == n-1) { k = 0; continue; }
        int j = sa[ra[i]+1];
        while (i+k < n and j+k < n and s[i+k] == s[j+
    k]) k++;
        lcp[ra[i]] = k;
    }
    return lcp;
}


int32_t main(){
    sws;
    string s;
    cin>>s;

    vector<int> suf = suffix_array(s);
```

```cpp
    vector<int> lcp = kasai(s, suf);

    ll ans = 0;
    for(int i=0; i<s.size(); i++){
        if(islower(s[suf[i]])){
            int sz = s.size()-suf[i];
            ans += (sz - lcp[i]);
        }
    }
    cout<<ans<<endl;
}
```

7 Template

7.1 Template

```cpp
#include <bits/stdc++.h>
using namespace std;
//alias comp='g++ -std=c++17 -g -O2 -Wall -fsanitize=
    address -o out'

#define int long long
#define endl "\n"
#define sws std::ios::sync_with_stdio(false); cin.tie
    (NULL); cout.tie(NULL);
#define all(x) x.begin(), x.end()
#define input(x) for (auto &it : x) cin >> it
#define print(x,y) loop(i,0,y){cout << x[i] << " ";}
    cout << "\n";
#define dbg(msg, x) cout << msg << " = " << x << endl
#define pb push_back
#define mp make_pair
#define ff first
#define ss second
#define TETO(a, b) ((a) + (b-1))/(b)
#define loop(i,a,n) for(int i=a; i < n; i++)
typedef long long ll;
typedef vector<int> vi;
typedef pair<int,int> pii;
typedef priority_queue<int, vector<int>, greater<int
    >> pqi;

const ll MOD = 1e9+7;
const int MAX = 1e4+5;
const ll LLINF = 0x3f3f3f3f3f3f3f3f;
const double PI = acos(-1);


int32_t main(){ sws;


    return 0;
}
```

8 zExtra

8.1 Getline

```cpp
#include <bits/stdc++.h>
using namespace std;
// Sempre usar cin.ignore() entre um cin e um getline
int main() {

    string s1; cin>>s1;
    cin.ignore();
    while (true) {
        string s; getline(cin, s);
        if (s == "PARO") break;
        cout<<"A"<<endl;
    }
```

```cpp
    string s2; cin>>s2;
    cin.ignore();
    while (true) {
        string s3; getline(cin, s3);
        if (s3 == "PARO") break;
        cout<<"A"<<endl;
    }
}
```