Competitive programming Notebook •



Meia noite eu te conto

C	Contents			5.12 Random	
1	Geometry 1.1 Convex Hull	2 2		5.14 Split	12 13
2	DP 2.1 Lcs	2 2 3	6	5.16 Template Full	
	2.3 Edit Distance 2.4 Digit Dp	3 3	Ū	6.1 Dsu	13 14
	2.5 Range Dp	4 4 4		6.3 Big K	1
3	2.8 Knapsack	5 5		6.6 Lazy Addition To Segment 6.7 Range Query Point Update 6.8 Persistent	17
	3.1 Bfs	5 5		6.9 Dynamic Implicit Sparse	19 20
	3.3 Is Prime 3.4 Lca 3.5 Dijkstra	6 6 6		6.11 Minimum And Amount	
	3.6 Floyd Warshall 3.7 Ford Fulkerson 3.8 Has Negative Cycle 3.9 Dinic	6 7 7 7	7	String 2 7.1 Is Substring	
	3.10 Dfs	8			
4	Math 4.1 Sieve	9 9			
	4.3 Ceil	9 9 9			
	4.6 Log Any Base	10 10			
5		10 10			
		10 11 11			
	5.4 Min Priority Queue	11			
	5.7 Base Converter	11 11			
	1	12 12			

Geometry

```
Convex Hull
                                                             67
1 // Convex Hull - Monotone Chain
                                                             68
                                                             69
2 //
_{\rm 3} // Convex Hull is the subset of points that forms the ^{70}
        smallest convex polygon
                                                             71
_4 // which encloses all points in the set.
5 //
6 // https://cses.fi/problemset/task/2195/
7 // https://open.kattis.com/problems/convexhull (
                                                             74
       counterclockwise)
9 // O(n log(n))
10
11 typedef long long ftype;
                                                             79
13 struct Point {
                                                             8.0
       ftype x, y;
                                                             81
14
                                                             82
15
       Point() {};
                                                             83
17
       Point(ftype x, ftype y) : x(x), y(y) {};
                                                             8.5
18
                                                             86
19
       bool operator < (Point o) {</pre>
           if (x == o.x) return y < o.y;
20
           return x < o.x;</pre>
                                                             87
21
                                                             88
22
                                                             89
24
       bool operator == (Point o) {
25
           return x == o.x && y == o.y;
                                                             90
                                                             91
26
27 };
                                                             92
                                                             93 }
29 ftype cross(Point a, Point b, Point c) {
       // v: a -> c
30
       // w: a -> b
3.1
32
       // v: c.x - a.x, c.y - a.y
       // w: b.x - a.x, b.y - a.y
34
       return (c.x - a.x) * (b.y - a.y) - (c.y - a.y) *
36
       (b.x - a.x);
37
38
39 ftype dir(Point a, Point b, Point c) {
      // 0 -> colineares
40
41
       // -1 -> esquerda
       // 1 -> direita
42
43
       ftype cp = cross(a, b, c);
45
       if (cp == 0) return 0;
46
                                                             12
       else if (cp < 0) return -1;
47
48
       else return 1;
                                                             1.4
49 }
                                                             15
50
51 vector < Point > convex_hull(vector < Point > points) {
52
       sort(points.begin(), points.end());
       points.erase( unique(points.begin(), points.end() 19
53
       ), points.end()); // somente pontos distintos
                                                             20
       int n = points.size();
5.4
                                                             21
                                                             22
       if (n == 1) return { points[0] };
56
                                                             23
                                                             24
       vector < Point > upper_hull = {points[0], points
                                                             25
58
       [1]};
                                                             26
       for (int i = 2; i < n; i++) {</pre>
                                                             27
           upper_hull.push_back(points[i]);
60
                                                             28
                                                             29
           int sz = upper_hull.size();
                                                             3.0
                                                             31
```

```
while (sz >= 3 && dir(upper_hull[sz-3],
upper_hull[sz-2], upper_hull[sz-1]) == -1) {
        upper_hull.pop_back();
        upper_hull.pop_back();
        upper_hull.push_back(points[i]);
        sz - - :
    }
vector < Point > lower_hull = {points[n-1], points[n
-211:
for (int i = n-3; i >= 0; i--) {
    lower_hull.push_back(points[i]);
    int sz = lower_hull.size();
    while (sz >= 3 && dir(lower_hull[sz-3],
lower_hull[sz-2], lower_hull[sz-1]) == -1) {
        lower_hull.pop_back();
        lower_hull.pop_back();
        lower_hull.push_back(points[i]);
    }
}
// reverse(lower_hull.begin(), lower_hull.end());
// counterclockwise
for (int i = (int)lower_hull.size() - 2; i > 0; i
--) {
    upper_hull.push_back(lower_hull[i]);
return upper_hull;
DP
```

2

64

6.5

2.1 Lcs

```
1 // LCS (Longest Common Subsequence)
2 //
3 // maior subsequencia comum entre duas strings
4 //
5 // tamanho da matriz da dp eh |a| x |b|
6 // lcs(a, b) = string da melhor resposta
7 // dp[a.size()][b.size()] = tamanho da melhor
      resposta
8 //
9 // https://atcoder.jp/contests/dp/tasks/dp_f
10 //
11 // O(n^2)
13 string lcs(string a, string b) {
      int n = a.size();
       int m = b.size();
      int dp[n+1][m+1];
      pair < int , int > p[n+1][m+1];
      memset(dp, 0, sizeof(dp));
      memset(p, -1, sizeof(p));
      for (int i = 1; i <= n; i++) {</pre>
           for (int j = 1; j <= m; j++) {</pre>
               if (a[i-1] == b[j-1]) {
                   dp[i][j] = dp[i-1][j-1] + 1;
                   p[i][j] = {i-1, j-1};
               } else {
                   if (dp[i-1][j] > dp[i][j-1]) {
                       dp[i][j] = dp[i-1][j];
                       p[i][j] = \{i-1, j\};
```

```
} else {
                                                                  return dp(number, 0, 10, 0, 0);
32
                                                           42
                       dp[i][j] = dp[i][j-1];
                                                           43 }
                       p[i][j] = {i, j-1};
34
                                                           44
                                                           45 int main() {
                                                                  ios::sync_with_stdio(false);
               }
                                                           46
                                                                  cin.tie(NULL);
           }
37
                                                           47
      }
                                                           48
                                                                  ll a, b; cin >> a >> b;
39
                                                           49
                                                                  cout << solve(b) - solve(a-1) << '\n';
      // recuperar resposta
40
                                                           50
41
                                                           51
       string ans = "";
                                                           52
                                                                  return 0:
42
43
      pair < int , int > curr = {n, m};
                                                           53
44
       while (curr.first != 0 && curr.second != 0) {
                                                                  Edit Distance
                                                             2.3
45
46
          auto [i, j] = curr;
47
                                                           1 // Edit Distance / Levenshtein Distance
           if (a[i-1] == b[j-1]) {
                                                            2 //
               ans += a[i-1];
49
                                                           3 // numero minimo de operacoes
                                                            4 // para transformar
5.1
                                                            5 // uma string em outra
           curr = p[i][j];
                                                           6 //
                                                           7 // tamanho da matriz da dp eh |a| x |b|
54
                                                           8 // edit_distance(a.size(), b.size(), a, b)
      reverse(ans.begin(), ans.end());
                                                           9 //
56
                                                           10 // https://cses.fi/problemset/task/1639
57
      return ans;
                                                           11 //
58 }
                                                           12 // O(n^2)
  2.2 Digit Dp 2
                                                           13
                                                           14 int tb[MAX][MAX];
                                                           15
1 // Digit DP 2: https://cses.fi/problemset/task/2220
                                                           int edit_distance(int i, int j, string &a, string &b)
2 //
_{\mbox{\scriptsize 3}} // Number of integers between a and b
                                                                  if (i == 0) return j;
4 // where no two adjacents digits are the same
                                                                  if (j == 0) return i;
                                                           18
                                                           19
6 #include <bits/stdc++.h>
                                                                  int &ans = tb[i][j];
                                                           20
                                                           2.1
8 using namespace std;
                                                                  if (ans != -1) return ans;
                                                           22
9 using ll = long long;
                                                           23
                                                                  ans = min({
                                                           24
11 const int MAX = 20; // 10^18
                                                           25
                                                                      edit_distance(i-1, j, a, b) + 1,
                                                                      edit_distance(i, j-1, a, b) + 1,
                                                           26
13 ll tb[MAX][MAX][2][2];
                                                                      edit_distance(i-1, j-1, a, b) + (a[i-1] != b[
14
                                                                  j-1])
15 ll dp(string& number, int pos, int last_digit, bool
                                                                  });
                                                           28
      under, bool started) {
       if (pos >= (int)number.size()) {
                                                           3.0
                                                                  return ans;
           return 1:
18
19
                                                             2.4 Digit Dp
      11& mem = tb[pos][last_digit][under][started];
20
      if (mem != -1) return mem;
      mem = 0:
                                                            1 // Digit DP 1: https://atcoder.jp/contests/dp/tasks/
22
                                                                  dp_s
      int limit = 9;
                                                            2 //
24
      if (!under) limit = number[pos] - '0';
                                                            3 // find the number of integers between 1 and K (
2.5
                                                                  inclusive)
      for (int digit = 0; digit <= limit; digit++) {</pre>
                                                            _{4} // where the sum of digits in base ten is a multiple
27
           if (started && digit == last_digit) continue;
                                                                 of D
29
           bool is_under = under || (digit < limit);</pre>
                                                            6 #include <bits/stdc++.h>
30
31
           bool is_started = started || (digit != 0);
                                                            8 using namespace std;
32
           mem += dp(number, pos+1, digit, is_under,
       is_started);
                                                           10 const int MOD = 1e9+7;
3.5
                                                           12 string k;
36
      return mem;
                                                           13 int d:
37 }
                                                           14
                                                           15 int tb[10010][110][2]:
38
39 ll solve(ll ubound) {
      memset(tb, -1, sizeof(tb));
                                                           int dp(int pos, int sum, bool under) {
40
      string number = to_string(ubound);
                                                                  if (pos >= k.size()) return sum == 0;
41
                                                           18
```

```
cin.tie(NULL):
19
                                                           3.5
20
       int& mem = tb[pos][sum][under];
                                                           36
      if (mem != -1) return mem;
                                                                  cin >> n >> s:
                                                           3.7
                                                                  memset(tb, -1, sizeof(tb));
      mem = 0;
                                                           38
                                                           39
                                                                  cout << dp(0, n-1) << '\n';
      int limit = 9;
24
                                                           40
       if (!under) limit = k[pos] - '0';
                                                           41
                                                                  return 0;
                                                           42 }
26
       for (int digit = 0; digit <= limit; digit++) {</pre>
27
          mem += dp(pos+1, (sum + digit) % d, under | ( 2.6 Lis Binary Search
      digit < limit));
           mem %= MOD;
                                                            int lis(vector<int> arr) {
3.0
                                                            2
                                                                  vector < int > dp;
31
                                                            3
32
      return mem;
                                                                  for (auto e : arr) {
33 }
                                                                      int pos = lower_bound(dp.begin(), dp.end(), e
34
                                                                  ) - dp.begin();
35 int main() {
      ios::sync_with_stdio(false);
                                                                      if (pos == (int)dp.size()) {
      cin.tie(NULL);
37
                                                                           dp.push_back(e);
                                                                      } else {
38
                                                            9
      cin >> k >> d;
39
                                                                           dp[pos] = e;
                                                           10
40
      memset(tb, -1, sizeof(tb));
                                                           12
42
       cout << (dp(0, 0, false) - 1 + MOD) % MOD << '\n' 14
43
                                                                  return (int)dp.size();
44
                                                              2.7
                                                                   Lis Segtree
      return 0;
45
46 }
                                                            1 #include <bits/stdc++.h>
  2.5 Range Dp
                                                            3 using namespace std;
1 // Range DP 1: https://codeforces.com/problemset/
      problem/1132/F
                                                            5 const int MAX = 2e5+17;
3 // You may apply some operations to this string
                                                            7 struct segTree {
                                                                  int size;
4 // in one operation you can delete some contiguous
      substring of this string
                                                                  vector < int > tree;
_{5} // if all letters in the substring you delete are
                                                                  void init(int n) {
      equal
6 // calculate the minimum number of operations to
                                                           12
                                                                      size = 1;
                                                                      while (size < n) size *= 2;</pre>
      delete the whole string s
                                                           1.3
                                                                      tree.assign(2 * size, 0LL);
                                                           14
8 #include <bits/stdc++.h>
                                                           15
                                                           16
10 using namespace std;
                                                           17
                                                                  int merge(int a, int b) {
                                                                      return max(a, b);
11
                                                           1.8
12 const int MAX = 510;
                                                           19
                                                           20
14 int n, tb[MAX][MAX];
                                                           21
                                                                  void build(vector<int> &arr, int x, int lx, int
15 string s;
                                                                  rx) {
                                                                      if (rx - lx == 1) {
16
                                                           22
int dp(int left, int right) {
                                                                           if (lx < (int)arr.size())</pre>
      if (left > right) return 0;
                                                                               tree[x] = arr[lx];
1.8
                                                           2.4
                                                           25
19
       int & mem = tb[left][right];
20
                                                                           return;
      if (mem != -1) return mem;
21
                                                           27
23
      mem = 1 + dp(left+1, right); // gastar uma
                                                           29
                                                                      int m = (1x + rx) / 2;
                                                                      build(arr, 2 * x + 1, lx, m);
      operaÃgÃčo arrumando sÃş o cara atual
                                                           30
                                                                      build(arr, 2 * x + 2, m, rx);
24
       for (int i = left+1; i <= right; i++) {</pre>
                                                           31
           if (s[left] == s[i]) {
                                                           3.2
2.5
26
               mem = min(mem, dp(left+1, i-1) + dp(i,
                                                                      tree[x] = merge(tree[2 * x + 1], tree[2 * x +
       right));
                                                                   2]);
           }
                                                           34
28
      }
                                                           35
                                                                  void build(vector<int> &arr) {
29
                                                           36
       return mem;
                                                           37
                                                                      build(arr, 0, 0, size);
30
31 }
                                                           38
                                                           39
33 int main() {
                                                                  void update(int i, int v, int x, int lx, int rx)
                                                           40
      ios::sync_with_stdio(false);
34
```

```
3
                                                                    Graph
           if (rx - 1x == 1) {
4.1
42
                tree[x] = v;
43
                return;
                                                               3.1
                                                                      \mathbf{Bfs}
           }
44
                                                             vector < vector < int >> adj; // adjacency list
           int m = (1x + rx) / 2;
46
                                                                   representation
           if (i < m) {</pre>
               update(i, v, 2 * x + 1, lx, m);
                                                            2 int n; // number of nodes
48
                                                             3 int s; // source vertex
           } else {
49
                update(i, v, 2 * x + 2, m, rx);
                                                             5 queue < int > q;
51
                                                             6 vector < bool > used(n + 1);
            tree[x] = merge(tree[2 * x + 1], tree[2 * x + 7 vector < int > d(n + 1), p(n + 1);
53
        2]);
                                                             9 q.push(s);
       }
5.4
                                                             10 used[s] = true;
55
                                                             11 p[s] = -1;
56
       void update(int i, int v) {
                                                             12 while (!q.empty()) {
           update(i, v, 0, 0, size);
5.7
                                                                  int v = q.front();
                                                             13
                                                             14
                                                                    q.pop();
5.9
                                                                   for (int u : adj[v]) {
       60
                                                                        if (!used[u]) {
61
                                                            17
                                                                            used[u] = true;
           if (lx >= 1 && rx <= r) return tree[x];</pre>
62
                                                                            q.push(u);
                                                             18
                                                             19
                                                                            d[u] = d[v] + 1;
            int m = (1x + rx) / 2;
64
                                                                            p[u] = v;
            int s1 = query(1, r, 2 * x + 1, lx, m);
int s2 = query(1, r, 2 * x + 2, m, rx);
                                                             20
6.5
                                                                        }
                                                             21
66
                                                                    }
                                                             22
67
                                                             23 }
           return merge(s1, s2);
                                                             24
       }
69
                                                            25 // restore path
70
                                                            26 if (!used[u]) {
       int query(int 1, int r) {
71
                                                                    cout << "No path!";</pre>
           return query(1, r, 0, 0, size);
                                                            27
72
                                                             28 } else {
                                                             29
                                                                    vector < int > path;
74 }:
                                                             30
                                                             31
                                                                   for (int v = u; v != -1; v = p[v])
7.6
                                                                        path.push_back(v);
                                                             32
77 int main() {
                                                             33
       ios::sync_with_stdio(false);
                                                                    reverse(path.begin(), path.end());
       cin.tie(NULL);
                                                             34
79
                                                             35
80
                                                                    cout << "Path: ";
                                                             36
       int n, arr[MAX], aux[MAX]; cin >> n;
8.1
                                                                    for (int v : path)
                                                             3.7
       for (int i = 0; i < n; i++) {</pre>
82
                                                                        cout << v << " ";
                                                             38
83
           cin >> arr[i];
                                                             39 }
84
            aux[i] = arr[i];
85
                                                               3.2 Kruskal
       }
86
       sort(aux, aux+n);
88
                                                             1 // need: DSU
89
       segTree st;
90
                                                             3 struct Edge {
       st.init(n);
91
                                                                    int u, v;
                                                                   ll weight;
       int ans = 0;
93
       for (int i = 0; i < n; i++) {</pre>
94
                                                                    Edge() {}
            int it = lower_bound(aux, aux+n, arr[i]) -
95
       aux;
                                                                    Edge(int u, int v, ll weight) : u(u), v(v),
            int lis = st.query(0, it) + 1;
96
                                                                    weight(weight) {}
97
           st.update(it, lis);
98
                                                             11
                                                                   bool operator < (Edge const& other) {</pre>
99
                                                             12
                                                                        return weight < other.weight;</pre>
           ans = max(ans, lis);
100
                                                             13
                                                             14 };
                                                             15
103
       cout << ans << '\n';
                                                             vector < Edge > kruskal(vector < Edge > edges, int n) {
104
                                                                    vector < Edge > result;
                                                             17
       return 0;
105
                                                                    11 cost = 0;
                                                             18
106 }
                                                             19
                                                                    sort(edges.begin(), edges.end());
                                                             20
         Knapsack
                                                                    DSU dsu(n);
                                                             21
                                                             22
```

23

24

for (auto e : edges) {

if (!dsu.same(e.u, e.v)) {

```
cost += e.weight;
                                                                 }
2.5
                                                            46
26
               result.push_back(e);
                                                            47
                                                                 int lca(int a, int b) {
               dsu.unite(e.u, e.v);
                                                            48
                                                                   if (dep[a] > dep[b]) swap(a, b);
           }
                                                            49
28
       }
                                                            50
                                                                   b = jump(b, dep[b] - dep[a]);
30
                                                            51
                                                                   if (a == b) return a;
31
       return result;
32 }
                                                            5.3
                                                                   for (int i = MAXE-1; i >= 0; i--) {
                                                            54
  3.3 Is Prime
                                                            55
                                                                    if (up[a][i] != up[b][i]) {
                                                                      a = up[a][i];
                                                            56
                                                            57
                                                                       b = up[b][i];
1 bool is_prime(ll n) {
                                                            58
                                                                     }
      if (n <= 1) return false;</pre>
                                                            59
       if (n == 2) return true;
                                                            60
                                                                   return up[a][0];
                                                            61
      for (11 i = 2; i*i <= n; i++) {</pre>
                                                            62
           if (n % i == 0)
                                                            63
               return false;
                                                                 int dist(int a, int b) {
                                                                  return dep[a] + dep[b] - 2 * dep[lca(a, b)];
                                                            6.5
9
                                                            66
                                                                 }
      return true;
10
                                                            67 };
11 }
                                                                     Dijkstra
                                                               3.5
  3.4 Lca
                                                             1 const int INF = 1e9+17;
1 // LCA
                                                             vector<vector<pair<int, int>>> adj; // {neighbor,
2 //
                                                                  weight}
3 // lowest common ancestor between two nodes
4 //
                                                             4 void dijkstra(int s, vector<int> & d, vector<int> & p
5 // edit_distance(n, adj, root)
6 //
                                                                   int n = adj.size();
7 // https://cses.fi/problemset/task/1688
                                                                   d.assign(n, INF);
                                                             6
8 //
                                                                   p.assign(n, -1);
9 // O(log N)
10
                                                                   d[s] = 0;
                                                            9
11 struct LCA {
                                                                   set < pair < int , int >> q;
                                                            10
12
    const int MAXE = 31;
                                                                   q.insert({0, s});
                                                            11
    vector < vector < int >> up;
13
                                                                   while (!q.empty()) {
    vector < int > dep;
                                                                       int v = q.begin()->second;
                                                            1.3
1.5
                                                            14
                                                                       q.erase(q.begin());
    LCA(int n, vector < vector < int >> & adj, int root = 1)
16
                                                            1.5
                                                            16
                                                                       for (auto edge : adj[v]) {
      up.assign(n+1, vector<int>(MAXE, -1));
                                                            1.7
                                                                            int to = edge.first;
18
       dep.assign(n+1, 0);
                                                                            int len = edge.second;
                                                            18
19
                                                            19
       dep[root] = 1;
                                                                            if (d[v] + len < d[to]) {</pre>
                                                            2.0
       dfs(root, -1, adj);
                                                                                q.erase({d[to], to});
22
                                                            22
                                                                                d[to] = d[v] + len;
      for (int j = 1; j < MAXE; j++) {</pre>
23
                                                                                p[to] = v;
                                                            23
         for (int i = 1; i <= n; i++) {</pre>
24
                                                                                q.insert({d[to], to});
                                                            24
           if (up[i][j-1] != -1)
                                                                            }
                                                            25
             up[i][j] = up[ up[i][j-1] ][j-1];
26
                                                                       }
                                                            26
27
                                                                   }
                                                            27
      }
28
29
30
                                                                    Floyd Warshall
                                                               3.6
    void dfs(int x, int p, vector<vector<int>>& adj) {
31
      up[x][0] = p;
33
       for (auto e : adj[x]) {
                                                             1 const long long LLINF = 0x3f3f3f3f3f3f3f3f3f1LL;
         if (e != p) {
34
                                                             3 for (int i = 0; i < n; i++) {</pre>
35
          dep[e] = dep[x] + 1;
                                                                   for (int j = 0; j < n; j++) {
           dfs(e, x, adj);
36
                                                             4
37
         }
                                                                       adj[i][j] = 0;
      }
38
39
40
    int jump(int x, int k) { // jump from node x k
                                                            9 long long dist[MAX][MAX];
41
                                                            10 for (int i = 0; i < n; i++) {
       for (int i = 0; i < MAXE; i++) {</pre>
                                                                 for (int j = 0; j < n; j++) {
42
        if (k&(1 << i) && x != -1) x = up[x][i];</pre>
                                                                       if (i == j)
43
      }
                                                                           dist[i][j] = 0;
44
                                                            13
       return x;
                                                                       else if (adj[i][j])
45
                                                            14
```

```
dist[i][j] = adj[i][j];
1.5
                                                            5.6
16
                                                            57
                                                                       return total;
               dist[i][j] = LLINF;
17
                                                            58
                                                            59 };
18
       }
19 }
                                                                    Has Negative Cycle
20
21 for (int k = 0; k < n; k++) {</pre>
       for (int i = 0; i < n; i++) {</pre>
22
                                                             1 // Edson
          for (int j = 0; j < n; j++) {</pre>
23
               dist[i][j] = min(dist[i][j], dist[i][k] + 2
                                                             3 using edge = tuple<int, int, int>;
        dist[k][j]);
25
                                                             5 bool has_negative_cycle(int s, int N, const vector<</pre>
26
                                                                   edge > & edges)
27 }
                                                             7
                                                                   const int INF { 1e9+17 };
  3.7 Ford Fulkerson
                                                             8
                                                                   vector < int > dist(N + 1, INF);
1 // Ford-Fulkerson
                                                                   dist[s] = 0;
                                                            1.0
2 //
3 // max-flow / min-cut
                                                            12
                                                                   for (int i = 1; i <= N - 1; i++) {</pre>
4 //
                                                                       for (auto [u, v, w] : edges) {
                                                            13
5 // MAX nÃşs
                                                                            if (dist[u] < INF && dist[v] > dist[u] +
                                                            14
                                                                   w) {
7 // https://cses.fi/problemset/task/1694/
                                                                                dist[v] = dist[u] + w;
8 //
                                                                           }
                                                            16
9 // O(m * max_flow)
                                                            17
                                                                       }
10
                                                                   }
                                                            18
11 using ll = long long;
                                                            19
12 const int MAX = 510;
                                                                   for (auto [u, v, w] : edges) {
                                                            20
                                                                       if (dist[u] < INF && dist[v] > dist[u] + w) {
                                                            21
14 struct Flow {
                                                            22
                                                                           return true;
       int n;
15
                                                                       }
                                                            23
      11 adj[MAX][MAX];
16
                                                            24
      bool used[MAX];
17
                                                            25
                                                                   return false;
                                                            26
      Flow(int n) : n(n) {};
19
                                                            27 }
20
21
       void add_edge(int u, int v, ll c) {
                                                              3.9
                                                                    Dinic
          adj[u][v] += c;
           adj[v][u] = 0; // cuidado com isso
                                                            1 // Dinic / Dinitz
24
                                                             2 //
                                                            3 // max-flow / min-cut
      11 dfs(int x, int t, ll amount) {
26
          used[x] = true;
                                                             4 //
27
                                                             5 // https://cses.fi/problemset/task/1694/
           if (x == t) return amount;
                                                             6 //
29
                                                             7 // O(E * V^2)
           for (int i = 1; i <= n; i++) {</pre>
3.1
32
               if (adj[x][i] > 0 && !used[i]) {
                                                            9 using ll = long long;
                   ll sent = dfs(i, t, min(amount, adj[x10 const ll FLOW_INF = 1e18 + 7;
33
      ][i]));
                                                            12 struct Edge {
                   if (sent > 0) {
                                                                  int from, to;
35
                                                            13
                        adj[x][i] -= sent;
                                                            14
                                                                   ll cap, flow;
                        adj[i][x] += sent;
                                                                   Edge* residual; // a inversa da minha aresta
3.7
                                                            1.5
                                                            16
38
                        return sent;
                                                            17
                                                                   Edge() {};
39
                   }
40
                                                            18
               }
                                                                   Edge(int from, int to, ll cap) : from(from), to(
                                                            19
42
           }
                                                                   to), cap(cap), flow(0) {};
43
                                                            20
44
           return 0;
                                                            21
                                                                   ll remaining_cap() {
                                                                       return cap - flow;
45
                                                            22
46
       11 max_flow(int s, int t) { // source and sink
47
                                                            24
           11 total = 0;
                                                                   void augment(ll bottle_neck) {
                                                            25
                                                                       flow += bottle_neck;
           11 \text{ sent} = -1;
49
                                                            26
50
                                                            27
                                                                       residual -> flow -= bottle_neck;
           while (sent != 0) {
                                                            28
               memset(used, 0, sizeof(used));
52
                                                            29
               sent = dfs(s, t, INT_MAX);
                                                                   bool is_residual() {
                                                            30
               total += sent;
                                                                       return cap == 0;
5.4
                                                            3.1
           }
55
                                                            32
```

```
33 };
                                                                         while (bfs(s, t)) {
                                                                             fill(next.begin(), next.end(), 0);
34
35 struct Dinic {
36
       int n;
                                                             104
                                                                              while (ll sent = dfs(s, t, FLOW_INF)) {
                                                                                  max_flow += sent;
       vector < vector < Edge *>> adj;
       vector < int > level , next;
38
                                                             106
                                                                         }
       Dinic(int n): n(n) {
40
                                                             108
            adj.assign(n+1, vector < Edge *>());
                                                                         return max_flow;
41
                                                             109
            level.assign(n+1, -1);
                                                             110
                                                                     }
            next.assign(n+1, 0);
43
                                                                     // path recover
45
                                                             113
                                                                     vector <bool > vis:
       void add_edge(int from, int to, ll cap) {
                                                                     vector < int > curr;
46
                                                             114
            auto e1 = new Edge(from, to, cap);
47
            auto e2 = new Edge(to, from, 0);
                                                                     bool dfs2(int x, int& t) {
48
                                                             116
49
                                                                         vis[x] = true;
            e1->residual = e2:
                                                                         bool arrived = false;
50
                                                             118
            e2 - > residual = e1;
                                                                         if (x == t) {
52
                                                             120
            adj[from].push_back(e1);
                                                                             curr.push_back(x);
53
            adj[to].push_back(e2);
                                                                              return true;
54
5.5
                                                             123
       bool bfs(int s, int t) {
                                                                         for (auto e : adj[x]) {
57
            fill(level.begin(), level.end(), -1);
                                                                             if (e->flow > 0 && !vis[e->to]) { // !e->
58
                                                                     is_residual() &&
59
            queue < int > q;
                                                                                  bool aux = dfs2(e->to, t);
            q.push(s);
61
                                                             128
            level[s] = 1;
                                                                                  if (aux) {
62
                                                                                      arrived = true;
            while (q.size()) {
                                                                                      e -> flow - -:
64
                int curr = q.front();
6.5
                q.pop();
                                                                             }
                                                                         }
67
                                                             134
                for (auto edge : adj[curr]) {
                    if (edge->remaining_cap() > 0 &&
                                                                         if (arrived) curr.push_back(x);
       level[edge->to] == -1) {
                         level[edge->to] = level[curr] +
                                                                         return arrived;
                                                             138
       1:
                                                             139
                         q.push(edge->to);
                                                             140
                                                                     vector < vector < int >> get_paths(int s, int t) {
                    }
                                                             141
                }
                                                                         vector < vector < int >> ans;
73
                                                             142
74
            }
                                                             143
75
                                                                         while (true) {
                                                             144
76
            return level[t] != -1;
                                                                              curr.clear();
                                                             145
       }
                                                                             vis.assign(n+1, false);
                                                             146
       11 dfs(int x, int t, 11 flow) {
                                                                             if (!dfs2(s, t)) break;
7.9
                                                             148
80
            if (x == t) return flow;
                                                             149
                                                                             reverse(curr.begin(), curr.end());
81
            for (int& cid = next[x]; cid < (int)adj[x].</pre>
                                                                             ans.push_back(curr);
82
       size(); cid++) {
                auto& edge = adj[x][cid];
83
                                                             153
                11 cap = edge->remaining_cap();
                                                             154
                                                                         return ans;
84
8.5
                                                             155
                if (cap > 0 && level[edge->to] == level[x_{156} };
       ] + 1) {
                                                                3.10
87
                    ll sent = dfs(edge->to, t, min(flow,
       cap)); // bottle neck
                    if (sent > 0) {
88
                                                              1 // DFS
                         edge -> augment(sent);
89
                                                              2 //
                         return sent;
                                                              3 // Percorre todos os vertices
                    }
91
                                                              4 // priorizando profundidade
                }
                                                              5 //
            }
93
                                                              6 // O(n+m)
94
            return 0;
                                                              8 vector < vector < int >> g;
96
                                                              9 vector < bool > vis;
                                                              1.0
       11 solve(int s, int t) {
98
                                                              void dfs(int s){
           11 max_flow = 0;
99
                                                                    if(vis[s]) return;
                                                              12
100
                                                                     vis[s] = true;
                                                              13
```

```
for(auto v : g[s]){
                                                                     for (int i = 2; i*i < lim; i++) {</pre>
1.4
                                                               g
15
           dfs(v);
                                                              10
                                                                          if (isprime[i]) {
16
                                                              11
                                                                              primes.push_back(i);
17 }
                                                                               for (int j = i+i; j < lim; j += i) {</pre>
                                                                                   isprime[j] = false;
       Math
                                                              14
                                                                          }
                                                              16
       Sieve
  4.1
                                                                     }
                                                              17
                                                              18
                                                                     return primes;
                                                              19
1 // nao "otimizado"
                                                              20 }
3 vector < bool > sieve(int lim=1e5+17) {
                                                                 4.5
                                                                      2sat
       vector < bool > isprime(lim+1, true);
                                                              1 // 2SAT
       isprime[0] = isprime[1] = false;
                                                              2 //
                                                               3 // verifica se existe e encontra soluÃğÃčo
       for (int i = 2; i*i < lim; i++) {</pre>
                                                               _4 // para f	ilde{\mathtt{A}}şrmulas booleanas da forma
           if (isprime[i]) {
                                                               5 // (a or b) and (!a or c) and (...)
               for (int j = i+i; j < lim; j += i) {</pre>
10
                                                               6 //
                    isprime[j] = false;
                                                               7 // indexado em 0
                                                               8 // n(a) = 2*x e n(~a) = 2*x+1
           }
13
                                                               9 // a = 2; n(a) = 4; n(\tilde{a}) = 5; n(a)^1 = 5; n(\tilde{a})
      }
14
                                                                     ^1 = 4
1.5
                                                              10 //
       return isprime;
                                                              // https://cses.fi/problemset/task/1684/
17 }
                                                              12 // https://codeforces.com/gym/104120/problem/E
                                                              13 // (add_eq, add_true, add_false e at_most_one n$\tilde{A}$co
  4.2
       Fexp
                                                                     foram testadas)
                                                              14 //
using ll = long long;
                                                              15 // O(n + m)
3 ll fexp(ll base, ll exp, ll m) {
                                                              17 struct sat {
      ll ans = 1;
                                                                     int n, tot;
      base %= m;
                                                                     {\tt vector} {<\tt vector} {<\tt int} {\gt\gt} {\tt adj} \;, \; {\tt adjt}; \;\; // \;\; {\tt grafo} \;\; {\tt original} \;,
                                                              1.9
                                                                      grafo transposto
       while (exp > 0) {
                                                                     vector < int > vis, comp, ans;
           if (exp % 2 == 1) {
                                                                     stack<int> topo; // ordem topolÃşgica
                                                              21
               ans = (ans * base) \% m;
                                                              22
1.0
                                                              23
                                                                     sat() {}
                                                                      sat(int n_{-}) : n(n_{-}), tot(n), adj(2*n), adjt(2*n)
                                                              24
           base = (base * base) % m;
12
                                                                     {}
           exp /= 2;
14
                                                              26
                                                                     void dfs(int x) {
15
                                                                          vis[x] = true;
                                                              27
16
       return ans;
                                                              28
17 }
                                                                          for (auto e : adj[x]) {
                                                              29
                                                              3.0
                                                                              if (!vis[e]) dfs(e);
  4.3 Ceil
                                                              31
                                                              32
                                                                          topo.push(x);
                                                              33
using ll = long long;
                                                              34
                                                              35
_{\rm 3} // avoid overflow
                                                                     void dfst(int x, int& id) {
                                                              3.6
4 ll division_ceil(ll a, ll b) {
                                                                          vis[x] = true;
                                                              37
       return 1 + ((a - 1) / b); // if a != 0
                                                              38
                                                                          comp[x] = id;
6 }
                                                              39
                                                                          for (auto e : adjt[x]) {
8 int intceil(int a, int b) {
                                                              41
                                                                              if (!vis[e]) dfst(e, id);
       return (a+b-1)/b;
                                                              42
10 }
                                                                     }
                                                              43
                                                              44
  4.4 Generate Primes
                                                              45
                                                                     void add_impl(int a, int b) \{ // a \rightarrow b = (!a or )
1 // crivo nao otimizado
                                                                          a = (a >= 0 ? 2*a : -2*a-1);
                                                                          b = (b >= 0 ? 2*b : -2*b-1);
                                                              47
3 vector<int> generate_primes(int lim=1e5+17) {
                                                              48
       vector < int > primes;
                                                                          adj[a].push_back(b);
      vector < bool > isprime(lim+1, true);
                                                                          adj[b^1].push_back(a^1);
                                                              50
                                                              51
       isprime[0] = isprime[1] = false;
                                                                          adjt[b].push_back(a);
                                                              52
                                                                          adjt[a^1].push_back(b^1);
                                                              53
```

5.4

55

56 57

59

62

63

64

66

67

68

69

72 73

74

7.5

76

78 79

80

8.1

82

84

85

86

87

89

90

91 92

93

94

96

97

98

99

103

104

105

106

108

110

111

112

113

114

115

117

118

119

```
}
                                                                       id++:
                                                   120
                                                                   }
void add_or(int a, int b) { // a or b
                                                              }
   add_impl(~a, b);
                                                               for (int i = 0; i < tot; i++) {</pre>
                                                                   if (comp[2*i] == comp[2*i+1]) return {
void add_nor(int a, int b) { // a nor b = !(a or
                                                           false, {}};
b)
                                                                   ans[i] = (comp[2*i] > comp[2*i+1]);
    add_or(~a, b), add_or(a, ~b), add_or(~a, ~b);127
}
                                                              return {true, ans};
                                                   129
void add_and(int a, int b) { // a and b
    add_or(a, b), add_or(~a, b), add_or(a, ~b); 131 };
                                                      4.6 Log Any Base
void add_nand(int a, int b) { // a nand b = !(a
and b)
                                                    int intlog(double base, double x) {
    add_or(~a, ~b);
                                                          return (int)(log(x) / log(base));
                                                    3 }
void add_xor(int a, int b) { // a xor b = (a != b 4.7 Divisors
    add_or(a, b), add_or(~a, ~b);
                                                    vector<ll> divisors(ll n) {
                                                          vector <11> ans;
void add_xnor(int a, int b) { // a xnor b = !(a
                                                          for (ll i = 1; i*i <= n; i++) {</pre>
xor b) = (a = b)
                                                              if (n%i == 0) {
    add_xor(~a, b);
                                                                   ll value = n/i;
                                                                   ans.push_back(i);
void add_true(int a) { // a = T
                                                    9
                                                                   if (value != i) {
    add_or(a, ~a);
                                                    10
                                                                       ans.push_back(value);
                                                              }
                                                    12
void add_false(int a) { // a = F
                                                    13
    add_and(a, ~a);
                                                    14
                                                    15
                                                          return ans;
                                                    16 }
// magia - brunomaletta
void add_true_old(int a) { // a = T (n sei se
                                                            Factorization
                                                      4.8
funciona)
    add_impl(~a, a);
                                                    1 // nson
                                                    3 using ll = long long;
void at_most_one(vector<int> v) { // no max um
verdadeiro
                                                    5 vector < pair < 11, int >> factorization(11 n) {
    adj.resize(2*(tot+v.size()));
                                                          vector < pair < 11, int >> ans;
    for (int i = 0; i < v.size(); i++) {</pre>
        add_impl(tot+i, ~v[i]);
                                                          for (11 p = 2; p*p <= n; p++) {</pre>
        if (i) {
                                                               if (n\%p == 0) {
            add_impl(tot+i, tot+i-1);
                                                                   int expoente = 0;
            add_impl(v[i], tot+i-1);
        }
                                                                   while (n\%p == 0) {
                                                    12
    }
                                                    13
                                                                       n /= p;
    tot += v.size();
                                                                       expoente++;
                                                    14
                                                    15
                                                    16
pair < bool , vector < int >> solve() {
                                                    17
                                                                   ans.push_back({p, expoente});
    ans.assign(n, -1);
                                                              }
                                                    18
    comp.assign(2*tot, -1);
                                                    19
    vis.assign(2*tot, 0);
                                                    20
    int id = 1;
                                                          if (n > 1) {
                                                    21
                                                               ans.push_back({n, 1});
    for (int i = 0; i < 2*tot; i++) if (!vis[i])</pre>
                                                   2.3
dfs(i);
                                                    24
                                                          return ans;
                                                    25
    vis.assign(2*tot, 0);
                                                    26 }
    while (topo.size()) {
        auto x = topo.top();
                                                           General
                                                      5
        topo.pop();
        if (!vis[x]) {
                                                            Compilation Flags
                                                      5.1
            dfst(x, id);
```

```
1 /*
      // josÃľ
2
      g++ -std=c++17 -Wshadow -Wall -Wextra -Wformat=2
      -Wconversion -fsanitize=address, undefined -fno-
      sanitize - recover - Wfatal - errors
      // maxwell
7 */
  5.2 Get Subset Sums
using ll = long long;
3 vector<11> get_subset_sums(int 1, int r, vector<11> & 12
      vector < 11 > ans;
      int len = r-l+1;
                                                          16
      for (int i = 0; i < (1 << len); i++) {</pre>
          11 sum = 0;
          for (int j = 0; j < len; j++) {</pre>
10
                                                          20
               if (i&(1 << j)) {</pre>
11
                                                          21
                   sum += arr[1 + j];
12
                                                          22
                                                          23
          }
1.4
                                                          24
1.5
                                                          25
16
          ans.push_back(sum);
                                                          26
1.7
                                                          27
                                                          2.8
19
      return ans;
                                                          29
20 }
                                                          30
                                                          31
  5.3 Next Permutation
                                                          3.3
1 // output: 1,2,3; 1,3,2; 2,1,3; 2,3,1; 3,1,2; 3,2,1; 34 // verifica se um nÞmero estÃą na base especificada
s vector < int > arr = {1, 2, 3};
                                                          36
4 int n = arr.size();
                                                          37
                                                          3.8
                                                          39
      for (auto e : arr) {
                                                          40
          cout << e << ' ';
                                                          41
                                                          42
      cout << '\n';
                                                          43
11 } while (next_permutation(arr.begin(), arr.end())); 44
  5.4 Min Priority Queue
                                                          46
                                                          47
1 template < class T> using min_priority_queue =
      priority_queue < T , vector < T > , greater < T > >;
  5.5 Xor 1 To N
_{1} // XOR sum from 1 to N
2 ll xor_1_to_n(ll n) {
      if (n % 4 == 0) {
          return n;
                                                           4
      } else if (n % 4 == 1) {
          return 1;
      } else if (n % 4 == 2) {
          return n + 1;
10
                                                           9
      return 0;
                                                           10
12
                                                          11
                                                          12
  5.6 Input By File
                                                          1.3
freopen("file.in", "r", stdin);
                                                          15
2 freopen("file.out", "w", stdout);
                                                          16
```

5.7 Base Converter

```
1 const string digits = "0123456789
      ABCDEFGHIJKLMNOPQRSTUVWXYZ";
 3 11 tobase10(string number, int base) {
      map < char , int > val;
       for (int i = 0; i < digits.size(); i++) {</pre>
           val[digits[i]] = i;
       ll ans = 0, pot = 1;
       for (int i = number.size() - 1; i >= 0; i--) {
           ans += val[number[i]] * pot;
           pot *= base;
       return ans;
17 }
19 string frombase10(ll number, int base) {
       if (number == 0) return "0";
       string ans = "";
       while (number > 0) {
           ans += digits[number % base];
           number /= base;
       reverse(ans.begin(), ans.end());
       return ans:
32 }
35 bool verify_base(string num, int base) {
       map < char , int > val;
       for (int i = 0; i < digits.size(); i++) {</pre>
           val[digits[i]] = i;
       for (auto digit : num) {
           if (val[digit] >= base) {
               return false;
       return true;
48 }
```

5.8Last True

```
last_true(2, 10, [](int x) { return x * x <= 30; })
  ; // outputs 5
[1, r]
if none of the values in the range work, return lo
  - 1
f(1) = true
f(2) = true
f(3) = true
f(4) = true
f(5) = true
f(6) = false
f(7) = false
f(8) = false
```

```
last_true(1, 8, f) = 5
                                                          4 using namespace __gnu_pbds;
   last_true(7, 8, f) = 6
19 */
                                                          6 typedef tree <
21 int last_true(int lo, int hi, function < bool(int) > f)
                                                                null_type,
                                                                less<int>.
                                                                rb_tree_tag,
      while (lo < hi) {
                                                                tree_order_statistics_node_update> ordered_set;
23
          int mid = lo + (hi - lo + 1) / 2;
                                                         13 void Erase(ordered_set& a, int x){
          if (f(mid)) {
                                                                int r = a.order_of_key(x);
                                                         14
              lo = mid;
                                                         15
                                                                auto it = a.find_by_order(r);
            else {
                                                         16
                                                                a.erase(it);
              hi = mid - 1:
                                                         17 }
3.0
                                                         18
      }
                                                         19 /*
31
                                                                order_of_key(k) // Number of items strictly
32
      return lo;
33 }
                                                                smaller than k.
                                                                find_by_order(k) // K-th element in a set (
       Template
  5.9
                                                                counting from zero).
                                                         22 */
                                                         23
#include <bits/stdc++.h>
                                                         24 /* os parametros sÃčo, na ordem:
                                                         25 1) int -> tipo do valor que quero inserir(key), pode
3 using namespace std;
                                                                ser int, double, pii e etc
                                                         26 2) null_type -> Ãľ para usar essa Ãarvore como set/
5 int main() {
                                                                multiset. DÃą pra usar essa ED como map tmb
     ios::sync_with_stdio(false);
                                                                trocando isso pra um tipo map
      cin.tie(NULL);
                                                         27 3) less<int> -> forma de comparaÃgÃčo dos elementos.
                                                               less<int>, less_equal<int>, greater,
                                                                greater_equal e etc
1.0
                                                         28 4) rb_tree_tag -> tipo de Ãąrvore a ser usado, usar a
      return 0;
                                                                rb para as operaÃğÃţes serem O(logN)
12 }
                                                         29 5) tree_order_statistics_node__update -> contÃlm
                                                                vÃąrias operaÃğÃţes para atualizar "tree-based
  5.10 Bitwise
                                                                container", usado pra manter algo como o numero
                                                                de nodes em alguma subÃąrvore
1 #define lcm(a,b) (a*b)/gcd(a,b)
2 #define msb(n) (32 - builtin_clz(n))
                                                         31 usar less<equal> em (3) para usar como set e
                                                                less_equal < TIPO > para usar como multiset (
  5.11 Goto
                                                               permitir elementos repetidos).
                                                         32 se em (3) colocar greater_equal, entao o order_of_key
                                                                (k) retorna a quantidade de valores maiores que k
1 while (t--) {
   for (int d = 0; d < 11; d++) {
                                                                 (ÃI preferivel usar less_equal e retornar o
                                                                tamanho do intervalo - order)
      if (n % 11 == 0) {
                                                         33 */
        cout << "YES" << endl;</pre>
                                                         3.4
        goto done;
                                                         35 /* Como usar
                                                         _{\rm 36} .find_by_order(k) -> retorna um ITERADOR para o
                                                                elemento na posiÃğÃčo kth (contando do 0, ou seja
     n -= 111;
                                                                , k = 0 retorna o menor)
     if (n < 0) break;
                                                                Usar *(find_by_order(k)) para saber QUAL ÃL o
                                                         37
10
                                                                numero na posiÃğÃčo.
    cout << "NO" << endl;
                                                         38
12
                                                             .order_of_key(k) -> retorna o numero de valores na
                                                         39
    done:;
                                                                Ãąrvore que estÃčo ESTRITAMENTE menores que k;
14 }
                                                                retorna um inteiro
                                                         40
                                                         41
  5.12 Random
                                                             TambÃľm Ãľ possivel usar as funÃğÃţes que o set
                                                         42
                                                                comum possui, como o .insert().
1 random_device dev;
                                                             D\tilde{A}q pra percorrer os valores inseridos na \tilde{A}qrvore
2 mt19937 rng(dev());
                                                               usando o for(auto p: tree);
                                                             Obs.: O .erase funciona, mas precisa de um iterador
4 uniform_int_distribution < mt19937::result_type > dist
                                                                para o elemento! Para isso tem a funÃgÃčo Erase
      (1, 6); // distribution in range [1, 6]
                                                                implementada.
                                                         45 */
6 int val = dist(rng);
                                                            5.14 Split
  5.13 Ordered Set
                                                          vector<string> split(string s, char key=' ') {
                                                                vector < string > ans;
#include <ext/pb_ds/assoc_container.hpp>
                                                               string aux = "";
2 #include <ext/pb_ds/tree_policy.hpp>
                                                          3
```

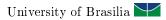
1 /*

```
first_true(2, 10, [](int x) { return x * x >= 30;
      for (int i = 0; i < (int)s.size(); i++) {</pre>
6
          if (s[i] == key) {
                                                                }); // outputs 6
               if (aux.size() > 0) {
                  ans.push_back(aux);
                                                               [1, r]
                   aux = "";
              }
                                                               if none of the values in the range work, return hi
                                                           6
10
          } else {
               aux += s[i];
          }
                                                               f(1) = false
13
      }
                                                           9
                                                              f(2) = false
                                                               f(3) = false
15
                                                          10
16
      if ((int)aux.size() > 0) {
                                                               f(4) = false
                                                              f(5) = false
17
          ans.push_back(aux);
                                                          12
                                                              f(6) = true
                                                          13
18
                                                          f(7) = true
19
      return ans;
                                                          15
                                                              f(8) = true
20
21 }
                                                          16 */
                                                          1.7
  5.15
           Read
                                                          int first_true(int lo, int hi, function <bool(int) > f)
                                                                 {
                                                                 hi++;
                                                          19
1 /*
                                                                 while (lo < hi) {</pre>
                                                          20
      RELER O ENUNCIADO!
                                                                    int mid = lo + (hi - lo) / 2;
                                                          2.1
      WA? coloca long long que passa;
      testar casos de borda, n = 0? n = 1? todos os
4
                                                                     if (f(mid)) {
                                                          2.3
      numeros iguais?
                                                                         hi = mid;
                                                          24
      Uma resposta Ãştima pode ter tamanho 2?
                                                                     } else {
                                                          25
      pode ser DP? nÃco Ãľ guloso de mais?
                                                                         lo = mid + 1;
                                                          26
      dÃa pra modelar como grafo?
                                                          27
                                                                     }
      pode ser fluxo?
                                                          28
      as vezes compensa mais fazer um brute (em python) _{29}\,
                                                                 return lo;
1.0
      nada funcionou? coda do zero
11
12 */
                                                             6
                                                                  \mathbf{DS}
  5.16 Template Full
                                                             6.1
                                                                  Dsu
#include <bits/stdc++.h>
2 #define debug(x) cout << "[" << #x << " = " << x << " 1 /*
                                                           2 DSU - Disjoint Set Union (or Union Find)
3 #define ff first
                                                           4 find(x) -> find component that x is on
4 #define ss second
                                                           5 join(a, b) -> union of a set containing 'a' and set
                                                                containing b
6 using namespace std;
7 using ll = long long;
                                                           7 find / join with path compreension -> O(inv_Ackermann
8 using ld = long double;
                                                                 (n)) [0(1)]
9 using pii = pair<int,int>;
                                                           8 find / join without path compreension -> O(logN)
10 using vi = vector<int>;
11
                                                          10 https://judge.yosupo.jp/submission/126864
12 using tii = tuple < int, int, int >;
                                                          11 */
                                                          12
14 const int oo = (int)1e9 + 17; //INF to INT
                                                          13 struct DSU {
15 const 11 00 = 0x3f3f3f3f3f3f3f3f1LL; //INF to LL
                                                          14
                                                                 int n = 0, components = 0;
                                                          1.5
17 void solve(){
                                                                 vector < int > parent;
                                                          16
18
                                                          17
                                                                 vector < int > size;
19 }
                                                          18
20
                                                                 DSU(int nn){
                                                          19
21 int main() {
                                                          20
                                                                    n = nn;
      ios::sync_with_stdio(false);
22
                                                          21
                                                                     components = n;
23
      cin.tie(NULL);
                                                          22
                                                                     size.assign(n + 5, 1);
24
                                                                     parent.assign(n + 5, 0);
                                                          23
      int t = 1;
25
                                                          24
                                                                     iota(parent.begin(), parent.end(), 0);
      cin >> t;
                                                          2.5
      while (t - - ) {
27
                                                          26
28
           solve();
                                                                 int find(int x){
                                                          27
29
                                                          28
                                                                     if(x == parent[x]) {
30 }
                                                                         return x;
                                                          29
                                                          3.0
  5.17 First True
                                                                     //path compression
                                                          31
```

32

33

return parent[x] = find(parent[x]);



```
34
35
       void join(int a, int b){
           a = find(a);
36
           b = find(b);
           if(a == b) {
               return;
39
           if(size[a] < size[b]) {</pre>
41
                swap(a, b);
42
           parent[b] = a;
44
           size[a] += size[b];
46
           components -= 1;
47
48
       int sameSet(int a, int b) {
49
50
           a = find(a);
           b = find(b);
5.1
           return a == b;
53
54
55 };
         Ordered Set
1 // Ordered Set
```

```
2 //
3 // set roubado com mais operacoes
4 //
5 // para alterar para multiset
6 // trocar less para less_equal
7 //
8 // ordered_set < int > s
9 //
10 // order_of_key(k) // number of items strictly
      smaller than k \rightarrow int
11 // find_by_order(k) // k-th element in a set (
      counting from zero) -> iterator
12 //
13 // https://cses.fi/problemset/task/2169
_{15} // O(log N) para insert, erase (com iterator),
      order_of_key, find_by_order
17 using namespace __gnu_pbds;
18 template <typename T>
using ordered_set = tree < T, null_type, less < T > ,
      rb_tree_tag,tree_order_statistics_node_update>;
```

6.3 Big K

```
1 struct SetSum {
      11 s = 0;
      multiset <11> mt;
       void add(ll x){
           mt.insert(x);
           s += x;
      }
       int pop(11 x){
           auto f = mt.find(x);
9
           if(f == mt.end()) return 0;
10
           mt.erase(f);
           s -= x;
           return 1;
1.3
14
15 };
16
17 struct BigK {
      int k;
1.8
       SetSum gt, mt;
       BigK(int _k){
20
           k = _k;
21
```

```
void balancear(){
          while((int)gt.mt.size() < k && (int)mt.mt.
24
       size()){
                auto p = (prev(mt.mt.end()));
               gt.add(*p);
26
27
               mt.pop(*p);
28
           while((int)mt.mt.size() && (int)gt.mt.size()
29
           *(gt.mt.begin()) < *(prev(mt.mt.end())) ){
30
31
               11 u = *(gt.mt.begin());
               11 v = *(prev(mt.mt.end()));
32
33
               gt.pop(u); mt.pop(v);
3.4
                gt.add(v); mt.add(u);
35
36
       void add(ll x){
3.7
           mt.add(x);
           balancear();
39
40
       void rem(ll x){
41
           //x = -x;
42
           if(mt.pop(x) == 0)
               gt.pop(x);
44
           balancear();
45
46
47 };
```

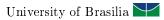
6.4 Segment With Maximum Sum

```
1 // Description:
2 // Query - get sum of segment that is maximum among
      all segments
3 // E.g
4 // Array: 5 -4 4 3 -5
_{5} // Maximum segment sum: 8 because 5 + (-4) + 4 = 8
_{\rm 6} // Update - update element at position id to a value
8 // Problem:
9 // https://codeforces.com/edu/course/2/lesson/4/2/
      practice/contest/273278/problem/A
11 // Complexity:
12 // O(log n) for both query and update
_{14} // How to use:
15 // Segtree seg = Segtree(n);
16 // seg.build(v);
18 // Notes
19 // The maximum segment sum can be a negative number
_{20} // In that case, taking zero elements is the best
      choice
21 // So we need to take the maximum between 0 and the
      query
22 // max(0LL, seg.query(0, n).max_seg)
24 using ll = long long;
25
26 typedef ll ftype_node;
28 struct Node {
      ftype_node max_seg;
29
30
       ftype_node pref;
3.1
      ftype_node suf;
32
      ftype_node sum;
33
      Node(ftype_node max_seg, ftype_node pref,
34
      ftype_node suf, ftype_node sum) : max_seg(max_seg
      ), pref(pref), suf(suf), sum(sum) {};
35 };
```

```
original
37 typedef Node ftype;
                                                                           // seg tamanho potencia de dois
                                                                           if (ini < (int)v.size()) {</pre>
38
                                                           104
                                                                               seg[pos] = Node(v[ini], v[ini], v[ini
39 struct Segtree {
      vector < ftype > seg;
                                                                  ], v[ini]);
       int n:
41
                                                           106
                                                                           }
       const ftype NEUTRAL = Node(0, 0, 0, 0);
42
                                                                           return;
                                                                       }
43
                                                           108
       Segtree(int n) {
44
                                                           109
           int sz = 1;
                                                                       int e = 2*pos + 1;
45
                                                           110
           // potencia de dois mais proxima
                                                                       int d = 2*pos + 2;
46
                                                           111
           while (sz < n) sz *= 2;
                                                                       int m = ini + (fim - ini) / 2;
48
           this -> n = sz;
                                                           113
                                                                       build(e, ini, m, v);
49
                                                           114
5.0
           // numero de nos da seg
                                                           115
                                                                       build(d, m + 1, fim, v);
           seg.assign(2*sz, NEUTRAL);
51
                                                           116
       }
52
                                                                       seg[pos] = f(seg[e], seg[d]);
53
                                                           118
54
       ftype f(ftype a, ftype b) {
           ftype_node max_seg = max({a.max_seg, b.
5.5
                                                           120
                                                                   ftype query(int p, int q) {
                                                                       return query(0, 0, n - 1, p, q);
       max_seg, a.suf + b.pref});
           ftype_node pref = max(a.pref, a.sum + b.pref)122
                                                           123
           ftype_node suf = max(b.suf, b.sum + a.suf); 124
                                                                   void update(int id, int val) {
57
           ftype_node sum = a.sum + b.sum;
                                                                       update(0, 0, n - 1, id, val);
58
                                                           125
59
                                                           126
60
           return Node(max_seg, pref, suf, sum);
                                                           127
       }
                                                                   void build(vector<int> &v) {
61
                                                           128
                                                                       build(0, 0, n - 1, v);
                                                           129
       ftype query(int pos, int ini, int fim, int p, int130
63
        q) {
           if (ini >= p && fim <= q) {</pre>
                                                                   void debug() {
64
                                                           132
               return seg[pos];
                                                           133
                                                                       for (auto e : seg) {
                                                                           cout << e.max_seg << ', ', << e.pref << ', '
           }
                                                           134
                                                                    << e.suf << ' ' << e.sum << '\n';
67
           if (q < ini || p > fim) {
               return NEUTRAL;
                                                                       cout << '\n';
69
                                                           136
70
                                                           137
                                                           138 };
           int e = 2*pos + 1;
72
                                                              6.5
                                                                    Lazy Dynamic Implicit Sparse
           int d = 2*pos + 2;
           int m = ini + (fim - ini) / 2;
7.4
75
                                                            1 // Description:
           return f(query(e, ini, m, p, q), query(d, m + 2 // Indexed at one
76
        1, fim, p, q));
                                                            4 // When the indexes of the nodes are too big to be
                                                                  stored in an array
       void update(int pos, int ini, int fim, int id,
                                                            5 // and the queries need to be answered online so we
       int val) {
                                                                  can't sort the nodes and compress them
80
           if (ini > id || fim < id) {</pre>
                                                            _{\rm 6} // we create nodes only when they are needed so there
               return;
81
                                                                  'll be (Q*log(MAX)) nodes
82
                                                            _{7} // where Q is the number of queries and MAX is the
                                                                  maximum index a node can assume
           if (ini == id && fim == id) {
84
                seg[pos] = Node(val, val, val, val);
85
                                                            _{9} // Query - get sum of elements from range (1, r)
86
                                                                  inclusive
                return:
87
                                                            _{
m 10} // Update - update element at position id to a value
           }
                                                                  val
89
           int e = 2*pos + 1;
                                                            12 // Problem:
           int d = 2*pos + 2;
91
                                                            13 // https://oj.uz/problem/view/IZhO12_apple
           int m = ini + (fim - ini) / 2;
92
                                                           15 // Complexity:
           update(e, ini, m, id, val);
94
                                                            16 // O(log n) for both query and update
           update(d, m + 1, fim, id, val);
96
                                                            18 // How to use:
           seg[pos] = f(seg[e], seg[d]);
                                                            _{\rm 19} // MAX is the maximum index a node can assume
97
       }
98
                                                            20 // Create a default null node
99
                                                            21 // Create a node to be the root of the segtree
       void build(int pos, int ini, int fim, vector<int>22
100
        &v) {
                                                           23 // Segtree seg = Segtree(MAX);
           if (ini == fim) {
                                                           24 // seg.create();
               // se a posiÃğÃčo existir no array
                                                           25 // seg.create();
```

```
return:
27 const int MAX = 1e9+10;
                                                           93
                                                                      }
28 const long long INF = 1e18+10;
                                                           94
                                                           95
                                                                      if (ini >= p && fim <= q) {</pre>
                                                                          lazy[pos] = apply_lazy(lazy[pos], val, 1)
30 typedef long long ftype;
31
                                                                           seg[pos] = apply_lazy(seg[pos], val, fim
32 struct Segtree {
      vector < ftype > seg, d, e, lazy;
                                                                  - ini + 1);
33
      const ftype NEUTRAL = 0;
34
                                                           98
       const ftype NEUTRAL_LAZY = -INF;
                                                                          return;
                                                           99
      int n:
36
                                                          100
                                                                      int m = (ini + fim) >> 1;
      Segtree(int n) {
38
          this -> n = n;
                                                                      if (e[pos] == 0) e[pos] = create();
40
                                                          104
                                                                      update(e[pos], ini, m, p, q, val);
41
42
      ftype apply_lazy(ftype a, ftype b, int len) {
           if (b == NEUTRAL_LAZY) return a;
                                                                      if (d[pos] == 0) d[pos] = create();
43
           else return b * len;
                                                                      update(d[pos], m + 1, fim, p, q, val);
45
                                                          109
                                                                      seg[pos] = f(seg[e[pos]], seg[d[pos]]);
46
                                                          110
      void propagate(int pos, int ini, int fim) {
                                                                  }
           if (seg[pos] == 0) return;
48
                                                          112
                                                                  ftype query(int p, int q) {
                                                          113
           if (ini == fim) {
50
                                                          114
                                                                      return query(1, 1, n, p, q);
5.1
               return:
                                                          115
52
                                                          116
                                                                  void update(int p, int q, int val) {
           int m = (ini + fim) >> 1;
                                                                      update(1, 1, n, p, q, val);
                                                          118
5.5
                                                          119
           if (e[pos] == 0) e[pos] = create();
                                                          120 };
           if (d[pos] == 0) d[pos] = create();
5.7
                                                             6.6 Lazy Addition To Segment
58
           lazy[e[pos]] = apply_lazy(lazy[e[pos]], lazy[
      pos], 1);
                                                            1 // Description:
           lazy[d[pos]] = apply_lazy(lazy[d[pos]], lazy[ 2 // Query - get sum of elements from range (1, r)
      pos], 1);
                                                                  inclusive
                                                            _{3} // Update - add a value val to elementos from range (
           seg[e[pos]] = apply_lazy(seg[e[pos]], lazy[
                                                                 l, r) inclusive
      posl. m - ini + 1):
           seg[d[pos]] = apply_lazy(seg[d[pos]], lazy[
                                                            5 // Problem:
      pos], fim - m);
                                                            6 // https://codeforces.com/edu/course/2/lesson/5/1/
                                                                 practice/contest/279634/problem/A
           lazy[pos] = NEUTRAL_LAZY;
                                                            8 // Complexity:
66
                                                           9 // O(log n) for both query and update
      ftype f(ftype a, ftype b) {
68
          return a + b;
                                                           11 // How to use:
7.0
                                                           12 // Segtree seg = Segtree(n);
                                                           13 // seg.build(v);
      ftype create() {
72
                                                           14
           seg.push_back(0);
73
                                                           15 // Notes
           e.push_back(0);
                                                           16 // Change neutral element and f function to perform a
           d.push_back(0);
7.5
                                                                   different operation
           lazy.push_back(-1);
                                                           1.7
           return seg.size() - 1;
                                                           18 const long long INF = 1e18+10;
78
79
                                                           20 typedef long long ftype;
      ftype query(int pos, int ini, int fim, int p, int _{21}
80
       q) {
                                                           22 struct Segtree {
8.1
           propagate(pos, ini, fim);
                                                                  vector<ftype> seg;
                                                           23
           if (q < ini || p > fim) return NEUTRAL;
82
                                                           24
                                                                  vector < ftype > lazy;
           if (pos == 0) return 0;
                                                                  int n;
                                                           25
           if (p <= ini && fim <= q) return seg[pos];</pre>
84
                                                                  const ftype NEUTRAL = 0;
           int m = (ini + fim) >> 1;
                                                                  const ftype NEUTRAL_LAZY = -INF;
           return f(query(e[pos], ini, m, p, q), query(d_{28}
86
       [pos], m + 1, fim, p, q));
                                                           29
                                                                  Segtree(int n) {
                                                                      int sz = 1;
87
                                                           30
                                                                      while (sz < n) sz *= 2;
       void update(int pos, int ini, int fim, int p, int 32
                                                                      this -> n = sz;
       q, int val) {
           propagate(pos, ini, fim);
                                                                      seg.assign(2*sz, NEUTRAL);
                                                           34
91
           if (ini > q || fim < p) {</pre>
                                                                      lazy.assign(2*sz, NEUTRAL_LAZY);
                                                           35
```

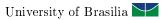
```
}
                                                                       update(e, ini, m, p, q, val);
36
37
                                                           103
                                                                       update(d, m + 1, fim, p, q, val);
       ftype apply_lazy(ftype a, ftype b, int len) {
38
                                                           104
           if (b == NEUTRAL_LAZY) return a;
                                                                       seg[pos] = f(seg[e], seg[d]);
39
           if (a == NEUTRAL_LAZY) return b * len;
                                                           106
           else return a + b * len:
41
                                                                  void build(int pos, int ini, int fim, vector<int>
42
                                                           108
                                                                   & v ) {
43
       void propagate(int pos, int ini, int fim) {
                                                                       if (ini == fim) {
44
           if (ini == fim) {
                                                                           if (ini < (int)v.size()) {</pre>
                                                           110
               return;
                                                                               seg[pos] = v[ini];
46
48
                                                           113
                                                                           return:
           int e = 2*pos + 1;
49
                                                           114
           int d = 2*pos + 2;
5.0
                                                           115
           int m = ini + (fim - ini) / 2;
                                                                       int e = 2*pos + 1;
51
                                                           116
                                                                       int d = 2*pos + 2;
                                                                       int m = ini + (fim - ini) / 2;
           lazy[e] = apply_lazy(lazy[e], lazy[pos], 1); 118
53
           lazy[d] = apply_lazy(lazy[d], lazy[pos], 1); 119
                                                                       build(e, ini, m, v);
5.5
           seg[e] = apply_lazy(seg[e], lazy[pos], m -
                                                                       build(d, m + 1, fim, v);
56
       ini + 1);
           seg[d] = apply_lazy(seg[d], lazy[pos], fim - 123
                                                                       seg[pos] = f(seg[e], seg[d]);
57
       m):
58
                                                           125
           lazy[pos] = NEUTRAL_LAZY;
                                                           126
                                                                  ftype query(int p, int q) {
59
       }
                                                                      return query(0, 0, n - 1, p, q);
60
                                                           127
61
                                                           128
       ftype f(ftype a, ftype b) {
62
                                                           129
           return a + b;
                                                                  void update(int p, int q, int val) {
63
                                                           130
                                                                       update(0, 0, n - 1, p, q, val);
64
                                                           131
6.5
                                                           132
       ftype query(int pos, int ini, int fim, int p, int133
66
                                                                  void build(vector<int> &v) {
           propagate(pos, ini, fim);
                                                                       build(0, 0, n - 1, v);
67
                                                           135
68
                                                           136
           if (ini >= p && fim <= q) {</pre>
69
                                                           137
               return seg[pos];
70
                                                           138
                                                                  void debug() {
           }
                                                           139
                                                                       for (auto e : seg) {
                                                                           cout << e << ' ';
72
                                                           140
           141
                                                                       cout << '\n';
               return NEUTRAL;
7.4
                                                           142
                                                                       for (auto e : lazy) {
75
                                                           143
7.6
                                                           144
                                                                           cout << e << ', ';
           int e = 2*pos + 1;
                                                           145
           int d = 2*pos + 2;
                                                           146
                                                                       cout << '\n';
           int m = ini + (fim - ini) / 2;
                                                                       cout << '\n';
7.9
                                                           147
           return f(query(e, ini, m, p, q), query(d, m +149 );
81
        1, fim, p, q));
                                                                    Range Query Point Update
                                                              6.7
82
83
       void update(int pos, int ini, int fim, int p, int 1 // Description:
        q, int val) {
                                                            2 // Indexed at zero
           propagate(pos, ini, fim);
85
                                                            _3 // Query - get sum of elements from range (1, r)
86
                                                                  inclusive
           if (ini > q || fim < p) {</pre>
87
                                                            4 // Update - update element at position id to a value
               return;
                                                                  val
           }
89
                                                            6 // Problem:
90
           if (ini >= p && fim <= q) {</pre>
9.1
                                                            7 // https://codeforces.com/edu/course/2/lesson/4/1/
               lazy[pos] = apply_lazy(lazy[pos], val, 1)
92
                                                                  practice/contest/273169/problem/B
               seg[pos] = apply_lazy(seg[pos], val, fim 9 // Complexity:
93
       - ini + 1);
                                                            10 // O(\log n) for both query and update
94
               return;
95
                                                            12 // How to use:
           }
                                                            13 // Segtree seg = Segtree(n);
96
                                                            14 // seg.build(v);
           int e = 2*pos + 1;
                                                            1.5
           int d = 2*pos + 2;
99
                                                            16 // Notes
           int m = ini + (fim - ini) / 2;
100
                                                            _{\rm 17} // Change neutral element and f function to perform a
                                                                   different operation
```



```
return:
                                                           8.5
19 // If you want to change the operations to point
                                                           86
      query and range update
                                                           87
                                                                      int e = 2*pos + 1;
20 // Use the same segtree, but perform the following
                                                           88
      operations
                                                                      int d = 2*pos + 2;
                                                                      int m = ini + (fim - ini) / 2;
21 // Query - seg.query(0, id);
                                                           90
22 // Update - seg.update(1, v); seg.update(r + 1, -v); 91
                                                                      build(e, ini, m, v);
                                                           92
24 typedef long long ftype;
                                                                      build(d, m + 1, fim, v);
                                                           93
                                                           94
26 struct Segtree {
                                                                      seg[pos] = f(seg[e], seg[d]);
                                                           95
      vector < ftype > seg;
                                                           96
28
      int n;
                                                           97
      const ftype NEUTRAL = 0;
                                                           98
                                                                  ftype query(int p, int q) {
                                                                      return query(0, 0, n - 1, p, q);
3.0
                                                          99
      Segtree(int n) {
                                                          100
31
           int sz = 1;
           while (sz < n) sz *= 2;
                                                                  void update(int id, int val) {
33
                                                          102
                                                                      update(0, 0, n - 1, id, val);
           this -> n = sz;
3.5
                                                          104
           seg.assign(2*sz, NEUTRAL);
                                                          105
36
                                                                  void build(vector<int> &v) {
37
                                                          106
                                                                      build(0, 0, n - 1, v);
38
      ftype f(ftype a, ftype b) {
                                                          108
           return a + b;
40
                                                          109
                                                                  void debug() {
41
                                                          110
                                                                      for (auto e : seg) {
42
      ftype query(int pos, int ini, int fim, int p, int112
                                                                          cout << e << ' ';
43
          if (ini >= p && fim <= q) {</pre>
                                                                      cout << '\n';
44
                                                          114
               return seg[pos];
                                                          115
45
           7
                                                          116 };
46
47
                                                                   Persistent
                                                             6.8
           if (q < ini || p > fim) {
              return NEUTRAL;
49
                                                           1 // Description:
5.1
                                                            2 // Persistent segtree allows for you to save the
           int e = 2*pos + 1;
52
                                                                  different versions of the segtree between each
           int d = 2*pos + 2;
                                                                 update
           int m = ini + (fim - ini) / 2;
54
                                                            3 // Indexed at one
                                                            _4 // Query - get sum of elements from range (1, r)
           return f(query(e, ini, m, p, q), query(d, m +
56
                                                                  inclusive
       1, fim, p, q));
                                                            _{5} // Update - update element at position id to a value
5.7
                                                                 val
58
      void update(int pos, int ini, int fim, int id,
                                                           7 // Problem:
59
      int val) {
                                                            8 // https://cses.fi/problemset/task/1737/
          if (ini > id || fim < id) {</pre>
                                                           10 // Complexity:
61
               return;
           }
62
                                                           11 // O(log n) for both query and update
63
                                                           12
           if (ini == id && fim == id) {
64
                                                           13 // How to use:
               seg[pos] = val;
                                                           14 // vector < int > raiz(MAX); // vector to store the
66
                                                                  roots of each version
               return;
                                                           15 // Segtree seg = Segtree(INF);
           }
68
                                                           16 // raiz[0] = seg.create(); // null node
69
                                                           17 // curr = 1; // keep track of the last version
           int e = 2*pos + 1;
                                                           18
71
           int d = 2*pos + 2;
                                                           19 // raiz[k] = seg.update(raiz[k], idx, val); //
           int m = ini + (fim - ini) / 2;
72
                                                                 updating version k
7.3
                                                           20 // seg.query(raiz[k], l, r) // querying version k
           update(e, ini, m, id, val);
7.4
                                                           21 // raiz[++curr] = raiz[k]; // create a new version
           update(d, m + 1, fim, id, val);
                                                                 based on version k
76
           seg[pos] = f(seg[e], seg[d]);
                                                           23 const int MAX = 2e5+17;
78
                                                           24 const int INF = 1e9+17;
79
      void build(int pos, int ini, int fim, vector < int > 26 typedef long long ftype;
80
       &v) {
           if (ini == fim) {
                                                           28 struct Segtree {
               if (ini < (int)v.size()) {</pre>
82
                                                                 vector<ftype> seg, d, e;
                   seg[pos] = v[ini];
                                                                  const ftype NEUTRAL = 0;
83
                                                           3.0
84
                                                                  int n;
                                                           31
```

```
32
33
      Segtree(int n) {
                                                            9 // Query - get sum of elements from range (1, r)
          this -> n = n;
34
                                                                  inclusive
                                                            10 // Update - update element at position id to a value
3.5
                                                                  val
       ftype f(ftype a, ftype b) {
37
                                                            11
                                                            12 // Problem:
          return a + b;
                                                           13 // https://cses.fi/problemset/task/1648
39
40
                                                           15 // Complexity:
       ftype create() {
41
           seg.push_back(0);
                                                           16 // O(log n) for both query and update
42
           e.push_back(0);
                                                           18 // How to use:
44
           d.push_back(0);
                                                            _{\rm 19} // MAX is the maximum index a node can assume
           return seg.size() - 1;
45
       }
46
                                                            21 // Segtree seg = Segtree(MAX);
47
       ftype query(int pos, int ini, int fim, int p, int 22
48
                                                           23 typedef long long ftype;
           if (q < ini || p > fim) return NEUTRAL;
           if (pos == 0) return 0;
                                                           25 const int MAX = 1e9+17;
50
           if (p <= ini && fim <= q) return seg[pos];</pre>
                                                           26
           int m = (ini + fim) >> 1;
                                                            27 struct Segtree {
           return f(query(e[pos], ini, m, p, q), query(d 28
                                                                  vector <ftype > seg, d, e;
53
       [pos], m + 1, fim, p, q));
                                                                  const ftype NEUTRAL = 0;
54
                                                            3.0
                                                                  int n;
       int update(int pos, int ini, int fim, int id, int 32
                                                                  Segtree(int n) {
56
       val) {
                                                                       this -> n = n;
                                                           3.3
           int novo = create();
                                                                       create();
57
                                                                       create():
5.8
                                                           3.5
           seg[novo] = seg[pos];
                                                           36
           e[novo] = e[pos];
60
                                                           3.7
           d[novo] = d[pos];
                                                           38
                                                                   ftype f(ftype a, ftype b) {
6.1
                                                           39
                                                                      return a + b;
           if (ini == fim) {
63
                                                           40
               seg[novo] = val;
                                                            41
               return novo:
                                                                  ftype create() {
6.5
                                                           42
                                                           43
                                                                       seg.push_back(0);
66
                                                                       e.push_back(0);
                                                           44
           int m = (ini + fim) >> 1;
                                                            45
                                                                       d.push_back(0);
68
69
                                                                       return seg.size() - 1;
           if (id <= m) e[novo] = update(e[novo], ini, m _{47}
70
       , id, val);
          else d[novo] = update(d[novo], m + 1, fim, id 49
                                                                   ftype query(int pos, int ini, int fim, int p, int
       , val);
                                                                       if (q < ini || p > fim) return NEUTRAL;
                                                                       if (pos == 0) return 0;
           seg[novo] = f(seg[e[novo]], seg[d[novo]]);
7.3
                                                           5.1
                                                            52
                                                                       if (p <= ini && fim <= q) return seg[pos];</pre>
                                                                       int m = (ini + fim) >> 1;
7.5
           return novo;
                                                           5.3
76
                                                            54
                                                                       return f(query(e[pos], ini, m, p, q), query(d
                                                                   [pos], m + 1, fim, p, q));
       ftype query(int pos, int p, int q) {
78
                                                           55
          return query(pos, 1, n, p, q);
                                                            56
                                                                  void update(int pos, int ini, int fim, int id,
8.0
                                                           5.7
                                                                   int val) {
81
                                                                       if (ini > id || fim < id) {</pre>
       int update(int pos, int id, int val) {
82
                                                           5.8
           return update(pos, 1, n, id, val);
83
                                                           5.9
                                                                           return:
84
                                                           60
85 }:
                                                           61
                                                                       if (ini == fim) {
                                                           62
  6.9 Dynamic Implicit Sparse
                                                                           seg[pos] = val;
                                                           63
                                                           64
                                                                           return;
1 // Description:
                                                           65
                                                           66
2 // Indexed at one
                                                                       int m = (ini + fim) >> 1;
_4 // When the indexes of the nodes are too big to be
                                                           68
       stored in an array
                                                            69
_{5} // and the queries need to be answered online so we
                                                                       if (id <= m) {</pre>
                                                                           if (e[pos] == 0) e[pos] = create();
      can't sort the nodes and compress them
                                                                           update(e[pos], ini, m, id, val);
_{\rm 6} // we create nodes only when they are needed so there ^{72}
                                                                       } else {
       'll be (Q*log(MAX)) nodes
                                                           73
                                                                           if (d[pos] == 0) d[pos] = create();
_{7} // where Q is the number of queries and MAX is the
                                                            74
                                                           7.5
                                                                           update(d[pos], m + 1, fim, id, val);
      maximum index a node can assume
```

```
}
                                                                            return seg[pos];
7.6
                                                            5.5
                                                             56
                                                                        }
           seg[pos] = f(seg[e[pos]], seg[d[pos]]);
78
                                                            5.7
79
                                                            5.8
                                                                        if (q < ini || p > fim) {
                                                                            return NEUTRAL;
      ftype query(int p, int q) {
81
                                                            60
82
           return query(1, 1, n, p, q);
                                                            61
                                                                        int e = 2*pos + 1;
83
                                                            62
                                                                        int d = 2*pos + 2;
84
                                                            63
      void update(int id, int val) {
                                                                        int m = ini + (fim - ini) / 2;
                                                            64
           update(1, 1, n, id, val);
86
                                                            65
                                                                        return f(query(e, ini, m, p, q), query(d, m +
88 };
                                                                    1, fim, p, q));
                                                             67
          Lazy Assignment To Segment
                                                             68
                                                                    void update (int pos, int ini, int fim, int p, int
const long long INF = 1e18+10;
                                                                     q, int val) {
                                                                        propagate(pos, ini, fim);
                                                             7.0
3 typedef long long ftype;
                                                                        if (ini > q || fim < p) {</pre>
                                                            72
                                                            73
                                                                            return:
5 struct Segtree {
                                                                        }
                                                             74
      vector < ftype > seg;
                                                            7.5
      vector < ftype > lazy;
                                                                        if (ini >= p && fim <= q) {</pre>
                                                            76
      int n:
                                                                            lazy[pos] = apply_lazy(lazy[pos], val, 1)
      const ftype NEUTRAL = 0;
      const ftype NEUTRAL_LAZY = -INF;
1.0
                                                                             seg[pos] = apply_lazy(seg[pos], val, fim
11
                                                                    - ini + 1);
      Segtree(int n) {
12
                                                             79
           int sz = 1:
13
                                                                            return:
14
           // potencia de dois mais proxima
                                                            8.0
                                                                        }
                                                             81
1.5
           while (sz < n) sz *= 2;
           this -> n = sz;
                                                            82
16
                                                                        int e = 2*pos + 1;
                                                            83
           // numero de nos da seg
                                                            84
                                                                        int d = 2*pos + 2;
18
                                                                        int m = ini + (fim - ini) / 2;
           seg.assign(2*sz, NEUTRAL);
                                                            85
           lazy.assign(2*sz, NEUTRAL_LAZY);
20
                                                                        update(e, ini, m, p, q, val);
                                                            87
21
                                                                        update(d, m + 1, fim, p, q, val);
                                                             88
22
      ftype apply_lazy(ftype a, ftype b, int len) {
                                                             89
23
                                                                        seg[pos] = f(seg[e], seg[d]);
           if (b == NEUTRAL_LAZY) return a;
                                                             90
24
                                                             91
                                                                   }
           if (a == NEUTRAL_LAZY) return b * len;
25
           else return b * len;
                                                             92
                                                                    void build(int pos, int ini, int fim, vector<int>
                                                             93
27
                                                                     &v) {
28
                                                                        if (ini == fim) {
      void propagate(int pos, int ini, int fim) {
                                                             94
29
                                                                            // se a posi\tilde{\mathbb{A}}ğ\tilde{\mathbb{A}}čo existir no array
           if (ini == fim) {
30
31
               return;
                                                                   original
                                                                            // seg tamanho potencia de dois
           }
32
                                                                            if (ini < (int)v.size()) {</pre>
33
                                                             97
                                                                                 seg[pos] = v[ini];
                                                            98
           int e = 2*pos + 1;
34
                                                                            }
           int d = 2*pos + 2;
                                                            99
3.5
                                                                            return:
           int m = ini + (fim - ini) / 2;
                                                            100
                                                                        }
37
           lazy[e] = apply_lazy(lazy[e], lazy[pos], 1); 102
                                                                        int e = 2*pos + 1;
           lazy[d] = apply_lazy(lazy[d], lazy[pos], 1); 103
39
                                                                        int d = 2*pos + 2;
                                                            104
40
                                                                        int m = ini + (fim - ini) / 2;
           seg[e] = apply_lazy(seg[e], lazy[pos], m -
      ini + 1);
                                                                        \verb|build(e, ini, m, v);|\\
           seg[d] = apply_lazy(seg[d], lazy[pos], fim - 107
                                                            108
                                                                        build(d, m + 1, fim, v);
      m);
                                                            109
43
                                                            110
                                                                        seg[pos] = f(seg[e], seg[d]);
44
           lazy[pos] = NEUTRAL_LAZY;
45
46
                                                            112
                                                            113
                                                                    ftype query(int p, int q) {
      ftype f(ftype a, ftype b) {
47
                                                                        return query(0, 0, n - 1, p, q);
                                                            114
           return a + b;
                                                            115
49
50
                                                                    void update(int p, int q, int val) {
       ftype query(int pos, int ini, int fim, int p, int^{117}
51
                                                                        update(0, 0, n - 1, p, q, val);
       a) {
           propagate(pos, ini, fim);
                                                            119
                                                                    void build(vector<int> &v) {
           if (ini >= p && fim <= q) {</pre>
54
```



```
return NEUTRAL:
           build(0, 0, n - 1, v);
                                                             5.1
                                                             52
                                                                         }
124
                                                             5.3
                                                                         int e = 2*pos + 1;
       void debug() {
                                                             5.4
           for (auto e : seg) {
                                                             55
                                                                         int d = 2*pos + 2;
                cout << e << ' ';
                                                                         int m = ini + (fim - ini) / 2;
                                                             56
128
                                                             57
            cout << '\n';
                                                                         return f(query(e, ini, m, p, q), query(d, m +
129
                                                             5.8
           for (auto e : lazy) {
130
                                                                     1, fim, p, q));
                cout << e << ' ';
                                                             59
131
132
                                                             60
            cout << '\n';
                                                             61
                                                                    void update(int pos, int ini, int fim, int id,
            cout << '\n';
                                                                    int val) {
134
                                                                         if (ini > id || fim < id) {</pre>
135
                                                             62
136 };
                                                             63
                                                                             return;
                                                             64
           Minimum And Amount
   6.11
                                                             65
                                                                         if (ini == id && fim == id) {
                                                             66
 1 // Description:
                                                                             seg[pos] = mp(val, 1);
 2 // Query - get minimum element in a range (1, r)
                                                             68
                                                                             return:
                                                             69
       inclusive
                                                                         }
 _{\mbox{\scriptsize 8}} // and also the number of times it appears in that
                                                             70
                                                                         int e = 2*pos + 1;
 _4 // Update - update element at position id to a value ^{72}
                                                                         int d = 2*pos + 2;
                                                             7.3
       val
                                                             7.4
                                                                         int m = ini + (fim - ini) / 2;
 5
 6 // Problem:
                                                             75
                                                                         update(e, ini, m, id, val);
 7 // https://codeforces.com/edu/course/2/lesson/4/1/
                                                             76
                                                                         update(d, m + 1, fim, id, val);
       practice/contest/273169/problem/C
                                                             77
                                                             7.8
                                                                         seg[pos] = f(seg[e], seg[d]);
 9 // Complexity:
                                                             79
10 // O(\log n) for both query and update
                                                             80
                                                             81
12 // How to use:
                                                             82
                                                                    void build(int pos, int ini, int fim, vector<int>
                                                                     &v) {
13 // Segtree seg = Segtree(n);
                                                                         if (ini == fim) {
14 // seg.build(v);
                                                                             if (ini < (int)v.size()) {</pre>
                                                             8.4
                                                                                 seg[pos] = mp(v[ini], 1);
                                                             85
16 #define pii pair<int, int>
17 #define mp make_pair
                                                             86
                                                                             return;
                                                             87
18 #define ff first
                                                             88
                                                                         }
19 #define ss second
                                                             8.9
                                                                         int e = 2*pos + 1;
                                                             90
21 const int INF = 1e9+17;
                                                                         int d = 2*pos + 2;
                                                             91
                                                                         int m = ini + (fim - ini) / 2;
                                                             92
23 typedef pii ftype;
                                                             93
24
                                                                         build(e, ini, m, v);
25 struct Segtree {
                                                             94
       vector < ftype > seg;
                                                                         build(d, m + 1, fim, v);
26
       int n;
                                                             96
                                                             97
                                                                         seg[pos] = f(seg[e], seg[d]);
       const ftype NEUTRAL = mp(INF, 0);
28
                                                                    }
                                                             98
29
       Segtree(int n) {
                                                             99
           int sz = 1;
                                                                    ftype query(int p, int q) {
                                                             100
31
                                                                        return query(0, 0, n - 1, p, q);
            while (sz < n) sz *= 2;
32
                                                            102
           this -> n = sz;
3.3
                                                            103
34
                                                                    void update(int id, int val) {
                                                            104
            seg.assign(2*sz, NEUTRAL);
35
                                                            105
                                                                         update(0, 0, n - 1, id, val);
36
                                                            106
38
       ftype f(ftype a, ftype b) {
                                                                    void build(vector<int> &v) {
                                                            108
           if (a.ff < b.ff) return a;</pre>
39
                                                            109
                                                                         build(0, 0, n - 1, v);
            if (b.ff < a.ff) return b;</pre>
40
                                                            110
41
           return mp(a.ff, a.ss + b.ss);
                                                            111
                                                                    void debug() {
43
                                                                        for (auto e : seg) {
44
                                                                             cout << e.ff << ', ' << e.ss << '\n';
45
       ftype query(int pos, int ini, int fim, int p, int114
        q) {
                                                                         cout << '\n';
                                                            116
           if (ini >= p && fim <= q) {</pre>
46
                                                                    }
                return seg[pos];
                                                            117
47
                                                            118 };
49
           if (q < ini || p > fim) {
50
```

6.12 Segtree2d

```
67
1 // Description:
_{2} // Indexed at zero
                                                             68
_{3} // Given a N x M grid, where i represents the row and ^{69}
       j the column, perform the following operations
                                                             7.0
4 // update(j, i) - update the value of grid[i][j]
5 // query(j1, j2, i1, i2) - return the sum of values
       inside the rectangle
6 // defined by grid[i1][j1] and grid[i2][j2] inclusive 73
                                                             74
8 // Problem:
9 // https://cses.fi/problemset/task/1739/
11 // Complexity:
                                                             78
12 // Time complexity:
_{13} // O(log N * log M) for both query and update
_{14} // O(N * M) for build
15 // Memory complexity:
16 // 4 * M * N
                                                             8.1
                                                             82
18 // How to use:
19 // Segtree2D seg = Segtree2D(n, n);
                                                             83
20 // vector<vector<int>> v(n, vector<int>(n));
                                                             85
21 // seg.build(v);
22
                                                             86
23 // Notes
24 // Indexed at zero
                                                             87
                                                             88
2.5
26 struct Segtree2D {
                                                             89
       const int MAXN = 1025;
                                                             90
27
       int N, M;
                                                             9.1
28
       vector<vector<int>> seg;
                                                             93
3.0
                                                             94
31
                                                             95
       Segtree2D(int N, int M) {
32
           this -> N = N;
                                                             96
                                                             97
34
           this -> M = M:
           seg.resize(2*MAXN, vector<int>(2*MAXN));
                                                             98
35
36
37
       void buildY(int noX, int lX, int rX, int noY, int
       1Y, int rY, vector < vector < int >> &v) {
           if(1Y == rY){
39
40
                if(1X == rX){
                    seg[noX][noY] = v[rX][rY];
41
                                                             104
                    seg[noX][noY] = seg[2*noX+1][noY] +
43
       seg[2*noX+2][noY];
               }
44
           }else{
45
                int m = (1Y+rY)/2;
47
               buildY(noX, 1X, rX, 2*noY+1, 1Y, m, v);
48
               buildY(noX, 1X, rX, 2*noY+2, m+1, rY, v); 109
49
50
                seg[noX][noY] = seg[noX][2*noY+1] + seg[111]
       noX][2*noY+2];
           }
53
       }
54
                                                            114
       void buildX(int noX, int lX, int rX, vector<</pre>
       vector < int >> &v) {
           if(1X != rX){
                                                             116
               int m = (1X+rX)/2;
57
               buildX(2*noX+1, 1X, m, v);
59
                buildX(2*noX+2, m+1, rX, v);
                                                             118
61
           buildY(noX, 1X, rX, 0, 0, M - 1, v);
63
       }
64
                                                             122
65
```

```
void updateY(int noX, int lX, int rX, int noY,
int lY, int rY, int y){
    if(1Y == rY){
        if(1X == rX){
            seg[noX][noY] = !seg[noX][noY];
        lelse {
            seg[noX][noY] = seg[2*noX+1][noY] +
seg[2*noX+2][noY];
        }
    }else{
        int m = (1Y+rY)/2;
        if (y <= m) {
            updateY(noX, 1X, rX, 2*noY+1,1Y, m, y
);
        else if(m < y)
            updateY(noX, 1X, rX, 2*noY+2, m+1, rY
, y);
        seg[noX][noY] = seg[noX][2*noY+1] + seg[
noX][2*noY+2];
    }
void updateX(int noX, int lX, int rX, int x, int
    int m = (1X+rX)/2;
    if(1X != rX){
        if(x \le m)
            updateX(2*noX+1, 1X, m, x, y);
        else if(m < x){
            updateX(2*noX+2, m+1, rX, x, y);
    }
    updateY(noX, 1X, rX, 0, 0, M - 1, y);
}
int queryY(int noX, int noY, int lY, int rY, int
aY, int bY){
    if(aY <= 1Y && rY <= bY) return seg[noX][noY</pre>
1:
    int m = (1Y+rY)/2;
    if(bY <= m) return queryY(noX, 2*noY+1, 1Y, m</pre>
, aY, bY);
    if(m < aY) return queryY(noX, 2*noY+2, m+1,</pre>
rY, aY, bY);
    return queryY(noX, 2*noY+1, 1Y, m, aY, bY) +
queryY(noX, 2*noY+2, m+1, rY, aY, bY);
int queryX(int noX, int 1X, int rX, int aX, int
bX, int aY, int bY){
    if(aX <= 1X && rX <= bX) return queryY(noX,</pre>
0, 0, M - 1, aY, bY);
    int m = (1X+rX)/2;
    if(bX <= m) return queryX(2*noX+1, 1X, m, aX,</pre>
 bX, aY, bY);
    if(m < aX) return queryX(2*noX+2, m+1, rX, aX</pre>
, bX, aY, bY);
    return queryX(2*noX+1, 1X, m, aX, bX, aY, bY)
 + queryX(2*noX+2, m+1, rX, aX, bX, aY, bY);
void build(vector<vector<int>> &v) {
```

66

```
buildX(0, 0, N - 1, v);
123
                                                            2.4
124
       }
                                                             25
                                                                   void add(int x) {
125
                                                             26
       int query(int aX, int bX, int aY, int bY) {
                                                             27
                                                                        int curr = 0;
126
           return queryX(0, 0, N - 1, aX, bX, aY, bY);
                                                                        for (int i = 31; i >= 0; i--) {
128
                                                             29
                                                                            int b = ((x&(1 << i)) > 0);
       void update(int x, int y) {
130
                                                             3.1
           updateX(0, 0, N - 1, x, y);
                                                                            if (trie[curr][b] == 0)
131
                                                            32
132
                                                             33
                                                                                 trie[curr][b] = nxt++;
133 };
                                                             34
                                                             35
                                                                            paths [curr]++;
        String
                                                             36
                                                                            curr = trie[curr][b];
                                                             37
                                                            38
        Is Substring
                                                                        paths [curr]++;
                                                             39
                                                             40
                                                                        finish[curr]++;
                                                             41
 1 // equivalente ao in do python
                                                                   void rem(int x) {
 3 bool is_substring(string a, string b){ // verifica se
                                                                        int curr = 0;
                                                             44
        a Ã1 substring de b
       for(int i = 0; i < b.size(); i++){</pre>
                                                                        for (int i = 31; i >= 0; i--) {
                                                            46
           int it = i, jt = 0; // b[it], a[jt]
                                                                            int b = ((x&(1 << i)) > 0);
                                                            47
                                                             48
            while(it < b.size() && jt < a.size()){</pre>
                                                                            paths [curr] - -;
                                                             49
                if(b[it] != a[jt])
                                                                            curr = trie[curr][b];
                                                             50
                    break;
                                                            51
                                                            52
                it++:
                                                                        paths [curr] --;
                                                            5.3
12
                it++:
                                                                        finish[curr] - -;
                                                             54
13
                                                                   }
                                                            5.5
                if(jt == a.size())
14
                                                            56
                    return true;
                                                            57
                                                                    int search(int x) {
           }
16
                                                                        int curr = 0;
                                                            58
       }
17
                                                             59
18
                                                                        for (int i = 31; i >= 0; i--) {
                                                            60
       return false;
19
                                                                            int b = ((x&(1 << i)) > 0);
                                                            61
20 }
                                                            62
                                                                            if (trie[curr][b] == 0) return false;
                                                             63
         Triexor
                                                             64
                                                                            curr = trie[curr][b];
                                                            6.5
 1 // TrieXOR
 2 //
                                                             67
                                                                        return (finish[curr] > 0);
 3 // adiciona, remove e verifica se existe strings
                                                             68
       binarias
                                                             69
 4 // max_xor(x) = maximiza o xor de x com algum valor
                                                             7.0
       da trie
                                                                    int max_xor(int x) { // maximum xor with x and
 5 //
                                                                   any number of trie
 6 // raiz = 0
                                                                        int curr = 0, ans = 0;
 7 //
                                                             73
 8 // https://codeforces.com/problemset/problem/706/D
                                                                        for (int i = 31; i >= 0; i--) {
                                                             74
                                                                            int b = ((x&(1 << i)) > 0);
 9 //
10 // O(|s|) adicionar, remover e buscar
                                                                            int want = b^1;
                                                            7.6
                                                                            if (trie[curr][want] == 0 || paths[trie[
12 struct TrieXOR {
                                                             78
                                                                    curr][want]] == 0) want ^= 1;
       int n, alph_sz, nxt;
13
                                                                            if (trie[curr][want] == 0 || paths[trie[
       vector < vector < int >> trie;
                                                             7.9
14
       vector < int > finish, paths;
                                                                    curr][want]] == 0) break;
15
                                                                            if (want != b) ans |= (1 << i);</pre>
                                                             80
       TrieXOR() {}
                                                             8.1
                                                                            curr = trie[curr][want];
18
                                                             82
19
       TrieXOR(int n, int alph_sz = 2) : n(n), alph_sz( 83
                                                                        }
       alph_sz) {
           nxt = 1;
                                                                        return ans;
           trie.assign(n, vector<int>(alph_sz));
                                                                   }
21
                                                            86
           finish.assign(n * alph_sz, 0);
                                                            87 };
22
           paths.assign(n * alph_sz, 0);
23
```