14

17 18

23

. 13

Competitive programming Notebook



Meia noite eu te conto

Contents 6.9 Xor 1 To N				
	String 1.1 Is Substring	. 2		6.10 Next Permutation 6.11 Template Full 6.12 Goto 6.13 Split 6.14 Last True
2	Graph 2.1 Lca 2.2 Ford Fulkerson 2.3 Dinic 2.4 Floyd Warshall 2.5 Has Negative Cycle 2.6 Is Prime 2.7 Bfs 2.8 Dijkstra 2.9 Kruskal 2.10 Dfs	3 3 4 5 5 5 5 5 5 5 5	7	6.15 Template 6.16 Ordered Set 6.16 Ordered Set 6.17 Get Subset Sums 6.17 Get Subset Sums 6.17 Get Subset Sums 6.17 Get Subset Sums 7.1 Segtree2d 7.2 Minimum And Amount 7.3 Segment With Maximum Sum 7.3 Segment With Maximum Sum 7.4 Persistent 7.5 Range Query Point Update 7.5 Lazy Assignment To Segment 7.5 Segment
3	Math 3.1 Divisors 3.2 Fexp 3.3 Factorization 3.4 Ceil 3.5 Log Any Base 3.6 2sat 3.7 Sieve 3.8 Generate Primes	6 6 6 6 7		7.7 Dynamic Implicit Sparse
4	DP 4.1 Lis Binary Search 4.2 Digit Dp 4.3 Lcs 4.4 Digit Dp 2 4.5 Lis Segtree 4.6 Knapsack 4.7 Range Dp 4.8 Edit Distance	8 8 9 9 10 10		
5	Geometry 5.1 Convex Hull	11 . 11		
6	General 6.1 Base Converter 6.2 First True 6.2 First True 6.3 Random 6.4 Bitwise 6.4 Bitwise 6.5 Min Priority Queue 6.6 Input By File 6.7 Read 1.0 Read	. 12 . 12 . 12 . 12 . 12		

String

1.1 Is Substring

1 // equivalente ao in do python

```
_3 bool is_substring(string a, string b){ // verifica se_{49}
       a Ã1 substring de b
       for(int i = 0; i < b.size(); i++){</pre>
           int it = i, jt = 0; // b[it], a[jt]
                                                               52
           while(it < b.size() && jt < a.size()){</pre>
                                                               54
                if(b[it] != a[jt])
                                                               55
                                                               56
10
                                                               57
                it++;
11
                                                               58
12
                jt++;
                                                               59
13
                                                               60
                if(jt == a.size())
                                                               61
                    return true;
                                                               62
16
           }
                                                               63
       }
17
                                                               64
18
19
       return false;
                                                               66
```

3 // adiciona, remove e verifica se existe strings

1.2 Triexor

1 // TrieXOR

20 }

2 //

```
binarias
4 // max_xor(x) = maximiza o xor de x com algum valor
5 //
_6 // raiz = 0
7 //
8 // https://codeforces.com/problemset/problem/706/D
9 //
10 // O(|s|) adicionar, remover e buscar
12 struct TrieXOR {
      int n, alph_sz, nxt;
13
      vector < vector < int >> trie;
      vector < int > finish, paths;
15
      TrieXOR() {}
17
18
      TrieXOR(int n, int alph_sz = 2) : n(n), alph_sz(
19
      alph_sz) {
          nxt = 1;
          trie.assign(n, vector<int>(alph_sz));
21
           finish.assign(n * alph_sz, 0);
22
           paths.assign(n * alph_sz, 0);
23
24
      void add(int x) {
26
          int curr = 0;
28
           for (int i = 31; i >= 0; i--) {
29
               int b = ((x&(1 << i)) > 0);
30
31
               if (trie[curr][b] == 0)
                   trie[curr][b] = nxt++;
33
34
               paths[curr]++;
35
               curr = trie[curr][b];
36
           }
38
           paths[curr]++;
40
           finish[curr]++;
41
```

```
void rem(int x) {
          int curr = 0;
          for (int i = 31; i >= 0; i--) {
              int b = ((x&(1 << i)) > 0);
              paths[curr]--;
              curr = trie[curr][b];
          paths[curr]--;
          finish[curr]--;
      int search(int x) {
          int curr = 0;
          for (int i = 31; i >= 0; i--) {
              int b = ((x&(1 << i)) > 0);
               if (trie[curr][b] == 0) return false;
              curr = trie[curr][b];
          return (finish[curr] > 0);
      int max_xor(int x) { // maximum xor with x and
      any number of trie
          int curr = 0, ans = 0;
          for (int i = 31; i >= 0; i--) {
              int b = ((x&(1 << i)) > 0);
               int want = b^1;
              if (trie[curr][want] == 0 || paths[trie[
      curr][want]] == 0) want ^= 1;
              if (trie[curr][want] == 0 || paths[trie[
      curr][want]] == 0) break;
              if (want != b) ans |= (1 << i);</pre>
              curr = trie[curr][want];
          return ans;
      }
87 };
```

Graph

Lca2.1

42 43

44

45

47

67

68

69 70

71

73

75

76 77

78

79

80 81

82

83

84

```
1 // LCA
2 //
3 // lowest common ancestor between two nodes
4 //
5 // edit_distance(n, adj, root)
6 //
7 // https://cses.fi/problemset/task/1688
8 //
9 // O(log N)
10
11 struct LCA {
    const int MAXE = 31;
12
13
    vector < vector < int >> up;
    vector < int > dep;
14
    LCA(int n, vector<vector<int>>& adj, int root = 1)
      up.assign(n+1, vector<int>(MAXE, -1));
17
```

Flow(int n) : n(n) {};

19

```
dep.assign(n+1, 0);
18
                                                           20
                                                           21
                                                                   void add_edge(int u, int v, ll c) {
19
                                                                       adj[u][v] += c;
       dep[root] = 1;
20
                                                           22
      dfs(root, -1, adj);
                                                                       adj[v][u] = 0; // cuidado com isso
                                                           23
21
                                                           24
      for (int j = 1; j < MAXE; j++) {</pre>
23
                                                           25
        for (int i = 1; i <= n; i++) {</pre>
                                                                   11 dfs(int x, int t, ll amount) {
                                                           26
           if (up[i][j-1] != -1)
                                                                       used[x] = true;
25
                                                           27
             up[i][j] = up[up[i][j-1]][j-1];
26
                                                           28
        }
                                                                       if (x == t) return amount;
27
      }
28
                                                            30
    }
                                                                       for (int i = 1; i <= n; i++) {</pre>
                                                                           if (adj[x][i] > 0 && !used[i]) {
30
    void dfs(int x, int p, vector<vector<int>>& adj) { 33
                                                                               ll sent = dfs(i, t, min(amount, adj[x
31
      up[x][0] = p;
                                                                   ][i]));
32
      for (auto e : adj[x]) {
33
34
         if (e != p) {
                                                                                if (sent > 0) {
           dep[e] = dep[x] + 1;
                                                                                    adj[x][i] -= sent;
35
                                                           36
           dfs(e, x, adj);
                                                                                    adj[i][x] += sent;
        }
37
                                                           38
      }
                                                                                    return sent;
                                                            39
38
    }
                                                                               }
39
                                                            40
                                                                           }
40
                                                            41
    int jump(int x, int k) { // jump from node x k
                                                                       }
                                                           43
      for (int i = 0; i < MAXE; i++) {</pre>
                                                                       return 0;
42
                                                            44
        if (k&(1 << i) && x != -1) x = up[x][i];
43
                                                           45
44
                                                            46
      return x;
                                                                   11 max_flow(int s, int t) { // source and sink
45
                                                            47
    }
                                                                       11 total = 0;
46
                                                            48
                                                                       11 \text{ sent} = -1;
                                                            49
47
    int lca(int a, int b) {
48
                                                            50
      if (dep[a] > dep[b]) swap(a, b);
                                                                       while (sent != 0) {
                                                           51
49
      b = jump(b, dep[b] - dep[a]);
                                                           52
                                                                           memset(used, 0, sizeof(used));
                                                                           sent = dfs(s, t, INT_MAX);
                                                           53
51
      if (a == b) return a;
                                                                           total += sent;
53
                                                           55
       for (int i = MAXE-1; i >= 0; i--) {
54
                                                           56
        if (up[a][i] != up[b][i]) {
                                                           57
                                                                       return total;
          a = up[a][i];
                                                                   }
                                                           58
56
57
           b = up[b][i];
                                                            59 };
        }
58
                                                              2.3
                                                                    Dinic
59
60
      return up[a][0];
61
                                                            1 // Dinic / Dinitz
62
                                                            2 //
63
                                                            3 // max-flow / min-cut
    int dist(int a, int b) {
                                                            4 //
      return dep[a] + dep[b] - 2 * dep[lca(a, b)];
65
                                                            5 // https://cses.fi/problemset/task/1694/
66
                                                            6 //
67 };
                                                            7 // O(E * V^2)
        Ford Fulkerson
                                                            9 using 11 = long long;
                                                            10 const 11 FLOW_INF = 1e18 + 7;
1 // Ford-Fulkerson
                                                            11
2 //
                                                            12 struct Edge {
3 // max-flow / min-cut
                                                                   int from, to;
                                                            13
4 //
                                                                   ll cap, flow;
                                                           14
5 // MAX nÃşs
                                                                   Edge* residual; // a inversa da minha aresta
6 //
                                                            16
7 // https://cses.fi/problemset/task/1694/
                                                                   Edge() {};
                                                            17
8 //
                                                            18
9 // O(m * max_flow)
                                                                   Edge(int from, int to, ll cap) : from(from), to(
                                                            19
                                                                   to), cap(cap), flow(0) {};
using ll = long long;
                                                            20
12 const int MAX = 510;
                                                            21
                                                                   ll remaining_cap() {
13
                                                            22
                                                                       return cap - flow;
14 struct Flow {
                                                           23
      int n;
                                                            24
      11 adj[MAX][MAX];
                                                                   void augment(ll bottle_neck) {
16
                                                            25
                                                                       flow += bottle_neck;
      bool used[MAX];
                                                            26
                                                                       residual->flow -= bottle_neck;
18
                                                            27
```

```
29
                                                             97
30
       bool is_residual() {
                                                             98
                                                                     11 solve(int s, int t) {
                                                                         11 \max_{flow} = 0;
31
           return cap == 0;
                                                             99
                                                             100
32
33 };
                                                                         while (bfs(s, t)) {
                                                                             fill(next.begin(), next.end(), 0);
34
                                                             102
35 struct Dinic {
       int n;
                                                                             while (ll sent = dfs(s, t, FLOW_INF)) {
36
                                                             104
                                                                                  max_flow += sent;
       vector < vector < Edge * >> adj;
37
                                                             105
                                                                             }
       vector < int > level, next;
38
                                                             106
                                                                         }
39
                                                             107
40
       Dinic(int n): n(n) {
                                                             108
41
           adj.assign(n+1, vector < Edge *>());
                                                             109
                                                                         return max_flow;
           level.assign(n+1, -1);
                                                             110
42
43
           next.assign(n+1, 0);
                                                            111
                                                                    // path recover
44
                                                             112
45
                                                             113
                                                                     vector < bool > vis;
                                                                     vector < int > curr;
       void add_edge(int from, int to, ll cap) {
46
                                                             114
           auto e1 = new Edge(from, to, cap);
                                                             115
           auto e2 = new Edge(to, from, 0);
                                                                    bool dfs2(int x, int& t) {
48
                                                             116
                                                                         vis[x] = true;
                                                             117
49
           e1->residual = e2;
                                                                         bool arrived = false;
50
                                                             118
           e2->residual = e1;
51
                                                            119
                                                                         if (x == t) {
           adj[from].push_back(e1);
                                                                             curr.push_back(x);
53
                                                            121
           adj[to].push_back(e2);
                                                             122
                                                                             return true;
54
      }
                                                                         }
55
                                                             123
56
                                                             124
       bool bfs(int s, int t) {
                                                                         for (auto e : adj[x]) {
57
                                                                             if (e->flow > 0 && !vis[e->to]) { // !e->
           fill(level.begin(), level.end(), -1);
58
                                                             126
                                                                     is_residual() &&
59
           queue < int > q;
                                                                                  bool aux = dfs2(e->to, t);
60
           q.push(s);
                                                             128
61
           level[s] = 1;
                                                             129
                                                                                  if (aux) {
                                                                                      arrived = true;
63
                                                             130
           while (q.size()) {
                                                                                      e->flow--;
               int curr = q.front();
                                                                                  }
65
                q.pop();
                                                                             }
66
                                                                         }
67
                                                             134
                for (auto edge : adj[curr]) {
68
                                                             135
                    if (edge->remaining_cap() > 0 &&
                                                             136
                                                                         if (arrived) curr.push_back(x);
       level[edge->to] == -1) {
                        level[edge->to] = level[curr] +
                                                                         return arrived;
70
                                                             138
       1;
                                                             139
                                                                    7
                        q.push(edge->to);
71
                                                             140
                    }
                                                             141
                                                                     vector < vector < int >> get_paths (int s, int t) {
72
               }
                                                                         vector < vector < int >> ans;
                                                             142
           }
                                                                         while (true) {
                                                             144
           return level[t] != -1;
                                                                             curr.clear();
76
                                                             145
       }
                                                                             vis.assign(n+1, false);
77
                                                             146
78
                                                             147
       ll dfs(int x, int t, ll flow) {
                                                                             if (!dfs2(s, t)) break;
79
           if (x == t) return flow;
80
                                                             149
                                                                             reverse(curr.begin(), curr.end());
81
                                                             150
           for (int& cid = next[x]; cid < (int)adj[x].</pre>
                                                            151
                                                                             ans.push_back(curr);
82
       size(); cid++) {
                auto& edge = adj[x][cid];
83
                                                             153
               11 cap = edge->remaining_cap();
84
                                                            154
                                                                         return ans:
                                                                    }
85
               if (cap > 0 && level[edge->to] == level[x_{156} };
86
      ] + 1) {
                                                                2.4 Floyd Warshall
                    ll sent = dfs(edge->to, t, min(flow,
       cap)); // bottle neck
                    if (sent > 0) {
                                                              const long long LLINF = 0x3f3f3f3f3f3f3f3f3f1LL;
                        edge ->augment(sent);
89
                        return sent;
                                                              3 for (int i = 0; i < n; i++) {</pre>
90
                    }
                                                                    for (int j = 0; j < n; j++) {
91
                                                              4
               }
92
                                                                         adj[i][j] = 0;
           }
93
                                                                    }
                                                              6
94
                                                              7 }
95
           return 0:
       }
96
                                                              9 long long dist[MAX][MAX];
```

3 int s; // source vertex

```
10 for (int i = 0; i < n; i++) {
11
      for (int j = 0; j < n; j++) {
                                                           5 queue < int > q;
          if (i == j)
                                                           6 vector < bool > used(n + 1);
12
              dist[i][j] = 0;
                                                           7 vector < int > d(n + 1), p(n + 1);
13
           else if (adj[i][j])
              dist[i][j] = adj[i][j];
                                                           9 q.push(s);
15
16
                                                          10 used[s] = true;
               dist[i][j] = LLINF;
                                                          11 p[s] = -1;
17
                                                          12 while (!q.empty()) {
18
19 }
                                                                 int v = q.front();
                                                          13
                                                                 q.pop();
20
                                                          14
21 for (int k = 0; k < n; k++) {
                                                                 for (int u : adj[v]) {
      for (int i = 0; i < n; i++) {</pre>
                                                          16
                                                                     if (!used[u]) {
          for (int j = 0; j < n; j++) {
                                                                          used[u] = true;
                                                          17
               dist[i][j] = min(dist[i][j], dist[i][k] + 18
24
                                                                          q.push(u);
       dist[k][j]);
                                                                          d[u] = d[v] + 1;
                                                          19
                                                                          p[u] = v;
          }
                                                                     }
26
                                                          21
27 }
                                                          22
                                                          23 }
        Has Negative Cycle
                                                          24
                                                          25 // restore path
                                                          26 if (!used[u]) {
1 // Edson
                                                                 cout << "No path!";</pre>
                                                          28 } else {
3 using edge = tuple<int, int, int>;
                                                                 vector < int > path;
                                                          29
5 bool has_negative_cycle(int s, int N, const vector
                                                                 for (int v = u; v != -1; v = p[v])
                                                          31
      edge > & edges)
                                                                      path.push_back(v);
6 {
                                                          33
      const int INF { 1e9+17 };
                                                          34
                                                                 reverse(path.begin(), path.end());
                                                          35
      vector<int> dist(N + 1, INF);
                                                                 cout << "Path: ";</pre>
                                                          36
      dist[s] = 0;
10
                                                          37
                                                                 for (int v : path)
                                                                     cout << v << " ";
                                                          38
      for (int i = 1; i <= N - 1; i++) {</pre>
12
          for (auto [u, v, w] : edges) {
13
               if (dist[u] < INF && dist[v] > dist[u] +
                                                             2.8
                                                                   Dijkstra
      w) {
                   dist[v] = dist[u] + w;
16
               }
                                                           const int INF = 1e9+17;
          }
                                                           vector < vector < pair < int , int >>> adj; // {neighbor ,
      }
                                                                 weight}
19
      for (auto [u, v, w] : edges) {
20
                                                           4 void dijkstra(int s, vector<int> & d, vector<int> & p
          21
                                                                 ) {
               return true;
22
                                                                 int n = adj.size();
23
                                                                 d.assign(n, INF);
      }
24
                                                                 p.assign(n, -1);
      return false;
26
                                                                 d[s] = 0;
                                                           9
27 }
                                                                 set < pair < int , int >> q;
                                                           10
                                                                 q.insert({0, s});
                                                          11
  2.6 Is Prime
                                                          12
                                                                 while (!q.empty()) {
                                                                     int v = q.begin()->second;
                                                          13
                                                          14
                                                                      q.erase(q.begin());
1 bool is_prime(ll n) {
      if (n <= 1) return false;</pre>
                                                          15
                                                                      for (auto edge : adj[v]) {
      if (n == 2) return true;
                                                          16
                                                                          int to = edge.first;
                                                          17
                                                                          int len = edge.second;
                                                          18
      for (11 i = 2; i*i <= n; i++) {</pre>
                                                          19
           if (n % i == 0)
                                                                          if (d[v] + len < d[to]) {</pre>
                                                          20
              return false;
                                                                              q.erase({d[to], to});
                                                          21
                                                                              d[to] = d[v] + len;
9
                                                                              p[to] = v;
                                                          23
      return true;
10
                                                                              q.insert({d[to], to});
11 }
                                                          25
                                                                          }
  2.7 Bfs
                                                                     }
                                                          26
                                                          27
                                                                 }
vector<vector<int>> adj; // adjacency list
      representation
                                                                   Kruskal
                                                             2.9
2 int n; // number of nodes
```

```
1 // need: DSU
3 struct Edge {
      int u, v;
      ll weight;
       Edge() {}
      Edge(int u, int v, ll weight) : u(u), v(v),
      weight(weight) {}
10
       bool operator < (Edge const& other) {</pre>
12
          return weight < other.weight;</pre>
13
14 };
15
16 vector < Edge > kruskal (vector < Edge > edges, int n) {
       vector < Edge > result;
17
18
       11 cost = 0;
19
       sort(edges.begin(), edges.end());
20
       DSU dsu(n);
22
      for (auto e : edges) {
           if (!dsu.same(e.u, e.v)) {
24
               cost += e.weight;
25
               result.push_back(e);
26
               dsu.unite(e.u, e.v);
27
           }
      }
29
30
       return result;
31
32 }
  2.10 Dfs
1 // DFS
2 //
3 // Percorre todos os vertices
_4 // priorizando profundidade
5 //
6 // O(n+m)
8 vector < vector < int >> g;
9 vector < bool > vis;
void dfs(int s){
      if(vis[s]) return;
12
13
      vis[s] = true;
      for(auto v : g[s]){
14
15
           dfs(v);
```

3 Math

}

16

17 }

3.1 Divisors

```
return ans;
16 }
```

3.2

Fexp

14

```
using ll = long long;
3 ll fexp(ll base, ll exp, ll m) {
      ll ans = 1;
      base %= m;
5
      while (exp > 0) {
          if (exp % 2 == 1) {
               ans = (ans * base) % m;
10
           base = (base * base) % m;
12
13
           exp /= 2;
14
15
16
      return ans;
17 }
```

3.3 Factorization

```
1 // nson
3 using 11 = long long;
5 vector<pair<1l, int>> factorization(11 n) {
       vector<pair<11, int>> ans;
       for (11 p = 2; p*p <= n; p++) {</pre>
           if (n\%p == 0) {
9
                int expoente = 0;
10
                while (n\%p == 0) {
12
13
                    n /= p;
                    expoente++;
14
15
16
                ans.push_back({p, expoente});
17
           }
18
       }
19
20
       if (n > 1) {
21
22
           ans.push_back({n, 1});
23
24
25
       return ans;
26 }
```

3.4 Ceil

```
1 using ll = long long;
2
3 // avoid overflow
4 ll division_ceil(ll a, ll b) {
5    return 1 + ((a - 1) / b); // if a != 0
6 }
7
8 int intceil(int a, int b) {
9    return (a+b-1)/b;
10 }
```

3.5 Log Any Base

```
int intlog(double base, double x) {
   return (int)(log(x) / log(base));
}
```

```
3.6
         2sat
                                                                      }
                                                               66
                                                               67
                                                                      void add_nand(int a, int b) { // a nand b = !(a
                                                               68
1 // 2SAT
2 //
3 // verifica se existe e encontra soluÃgÃčo
                                                                           add_or(~a, ~b);
_4 // para f	ilde{\mathtt{A}}şrmulas booleanas da forma
                                                               70
                                                               71
5 // (a or b) and (!a or c) and (...)
                                                                      void add_xor(int a, int b) { // a xor b = (a != b
                                                               72
6 //
                                                                      )
7 // indexado em 0
                                                                           add_or(a, b), add_or(~a, ~b);
8 // n(a) = 2*x e n(~a) = 2*x+1
                                                                      }
9 // a = 2 ; n(a) = 4 ; n(~a) = 5 ; n(a)~1 = 5 ; n(~a)
                                                               74
       ^1 = 4
                                                                      void add_xnor(int a, int b) { // a xnor b = !(a
                                                               76
                                                                      xor b) = (a = b)
11 // https://cses.fi/problemset/task/1684/
                                                                           add_xor(~a, b);
12 // https://codeforces.com/gym/104120/problem/E
                                                               77
                                                               78
13 // (add_eq, add_true, add_false e at_most_one nÃčo
       foram testadas)
                                                                      void add_true(int a) { // a = T
14 //
                                                               80
15 // 0(n + m)
                                                               81
                                                                           add_or(a, ~a);
                                                                      }
                                                               82
16
17 struct sat {
                                                               83
                                                                      void add_false(int a) { // a = F
       int n, tot;
18
                                                                           add_and(a, ~a);
       vector < vector < int >> adj, adjt; // grafo original, 85
19
        grafo transposto
                                                               87
20
       vector<int> vis, comp, ans;
                                                                      // magia - brunomaletta
                                                               88
21
       stack<int> topo; // ordem topolÃşgica
                                                                      void add_true_old(int a) { // a = T (n sei se
                                                                      funciona)
       sat() {}
23
                                                                           add_impl(~a, a);
       \mathtt{sat}(\mathtt{int}\ \mathtt{n}_{\mathtt{-}}) : \mathtt{n}(\mathtt{n}_{\mathtt{-}}), \mathtt{tot}(\mathtt{n}), \mathtt{adj}(\mathtt{2*n}), \mathtt{adjt}(\mathtt{2*n}) 90
24
                                                                      }
                                                               91
25
                                                               92
                                                                      void at_most_one(vector<int> v) { // no max um
       void dfs(int x) {
                                                               93
26
                                                                      verdadeiro
           vis[x] = true;
                                                               94
                                                                           adj.resize(2*(tot+v.size()));
28
                                                                           for (int i = 0; i < v.size(); i++) {</pre>
           for (auto e : adj[x]) {
                                                               95
                                                                                add_impl(tot+i, ~v[i]);
                if (!vis[e]) dfs(e);
                                                               96
30
                                                                                if (i) {
           }
                                                               97
                                                               98
                                                                                    add_impl(tot+i, tot+i-1);
32
                                                                                    add_impl(v[i], tot+i-1);
           topo.push(x);
                                                               99
33
       }
                                                              100
                                                                           7
35
       void dfst(int x, int& id) {
                                                                           tot += v.size();
           vis[x] = true;
                                                              103
37
                                                              104
           comp[x] = id;
38
                                                                      pair < bool, vector < int >> solve() {
39
                                                                           ans.assign(n, -1);
                                                              106
           for (auto e : adjt[x]) {
40
                                                                           comp.assign(2*tot, -1);
                if (!vis[e]) dfst(e, id);
                                                                           vis.assign(2*tot, 0);
           }
42
                                                                           int id = 1;
       }
                                                              109
43
                                                              110
44
                                                                           for (int i = 0; i < 2*tot; i++) if (!vis[i])</pre>
       void add_impl(int a, int b) { // a -> b = (!a or 111
45
                                                                      dfs(i);
       b)
           a = (a >= 0 ? 2*a : -2*a-1);
46
                                                                           vis.assign(2*tot, 0);
           b = (b >= 0 ? 2*b : -2*b-1);
                                                                           while (topo.size()) {
                                                              114
48
                                                                               auto x = topo.top();
                                                              115
           adj[a].push_back(b);
49
                                                              116
                                                                               topo.pop();
           adj[b^1].push_back(a^1);
                                                              117
           adjt[b].push_back(a);
                                                              118
                                                                               if (!vis[x]) {
                                                                                    dfst(x, id);
                                                              119
           adjt[a^1].push_back(b^1);
                                                              120
                                                                                    id++;
       }
54
                                                                           }
       void add_or(int a, int b) { // a or b
56
57
           add_impl(~a, b);
                                                                           for (int i = 0; i < tot; i++) {</pre>
58
                                                                                if (comp[2*i] == comp[2*i+1]) return {
                                                                      false, {}};
60
       void add_nor(int a, int b) { // a nor b = !(a or
                                                                                ans[i] = (comp[2*i] > comp[2*i+1]);
           add_or(~a, b), add_or(a, ~b), add_or(~a, ~b);127
       }
62
                                                                           return {true, ans};
                                                              129
63
       void add_and(int a, int b) { // a and b
                                                              130
64
           add_or(a, b), add_or(~a, b), add_or(a, ~b); 131 };
65
```

3.7 Sieve

```
1 // nao "otimizado"
3 vector < bool > sieve(int lim=1e5+17) {
       vector < bool > isprime(lim+1, true);
       isprime[0] = isprime[1] = false;
       for (int i = 2; i*i < lim; i++) {</pre>
           if (isprime[i]) {
               for (int j = i+i; j < lim; j += i) {</pre>
                    isprime[j] = false;
           }
       }
14
15
16
       return isprime;
17 }
```

Generate Primes

```
1 // crivo nao otimizado
3 vector<int> generate_primes(int lim=1e5+17) {
       vector < int > primes;
       vector < bool > isprime(lim+1, true);
       isprime[0] = isprime[1] = false;
       for (int i = 2; i*i < lim; i++) {</pre>
9
           if (isprime[i]) {
10
               primes.push_back(i);
11
                for (int j = i+i; j < lim; j += i) {</pre>
                    isprime[j] = false;
14
           }
16
17
       }
18
19
       return primes;
20 }
```

DP4

Lis Binary Search

```
int lis(vector<int> arr) {
      vector<int> dp;
2
      for (auto e : arr) {
          int pos = lower_bound(dp.begin(), dp.end(), e _5 // tamanho da matriz da dp eh |a| x |b|
      ) - dp.begin();
          if (pos == (int)dp.size()) {
              dp.push_back(e);
          } else {
              dp[pos] = e;
10
11
      }
12
13
14
      return (int)dp.size();
15 }
  4.2 Digit Dp
```

```
1 // Digit DP 1: https://atcoder.jp/contests/dp/tasks/ 19
      dp_s
2 //
_{\mbox{\scriptsize 3}} // find the number of integers between 1 and K (
      inclusive)
```

```
_{4} // where the sum of digits in base ten is a multiple
       of D
6 #include <bits/stdc++.h>
8 using namespace std;
10 const int MOD = 1e9+7;
11
12 string k;
13 int d:
15 int tb[10010][110][2];
16
17 int dp(int pos, int sum, bool under) {
       if (pos >= k.size()) return sum == 0;
18
19
       int& mem = tb[pos][sum][under];
20
       if (mem != -1) return mem;
21
22
       mem = 0;
23
       int limit = 9;
24
       if (!under) limit = k[pos] - '0';
25
       for (int digit = 0; digit <= limit; digit++) {</pre>
27
           mem += dp(pos+1, (sum + digit) % d, under | (
28
       digit < limit));</pre>
           mem %= MOD;
29
30
31
32
       return mem;
33 }
34
35 int main() {
       ios::sync_with_stdio(false);
36
       cin.tie(NULL);
37
38
39
       cin >> k >> d;
40
       memset(tb, -1, sizeof(tb));
41
42
       cout << (dp(0, 0, false) - 1 + MOD) % MOD << '\n'</pre>
43
44
       return 0;
45
46 }
```

4.3 Lcs

```
1 // LCS (Longest Common Subsequence)
2 //
3 // maior subsequencia comum entre duas strings
4 //
6 // lcs(a, b) = string da melhor resposta
7 // dp[a.size()][b.size()] = tamanho da melhor
       resposta
9 // https://atcoder.jp/contests/dp/tasks/dp_f
10 //
11 // O(n<sup>2</sup>)
12
13 string lcs(string a, string b) {
       int n = a.size();
       int m = b.size();
16
       int dp[n+1][m+1];
       pair < int , int > p[n+1][m+1];
       memset(dp, 0, sizeof(dp));
20
       memset(p, -1, sizeof(p));
21
22
       for (int i = 1; i <= n; i++) {</pre>
23
```

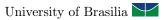
```
for (int j = 1; j <= m; j++) {</pre>
                                                                   }
24
                                                           34
25
               if (a[i-1] == b[j-1]) {
                                                           35
                   dp[i][j] = dp[i-1][j-1] + 1;
26
                                                           36
                                                                   return mem;
                   p[i][j] = \{i-1, j-1\};
                                                           37 }
               } else {
                   if (dp[i-1][j] > dp[i][j-1]) {
                                                           39 ll solve(ll ubound) {
29
                        dp[i][j] = dp[i-1][j];
                                                                   memset(tb, -1, sizeof(tb));
30
                                                           40
                                                                   string number = to_string(ubound);
                       p[i][j] = {i-1, j};
31
                                                           41
                   } 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
35
                   }
                                                            45 int main() {
               }
36
                                                           46
                                                                   ios::sync_with_stdio(false);
           }
                                                                   cin.tie(NULL);
                                                           47
37
      }
38
                                                            48
                                                                   ll a, b; cin >> a >> b;
                                                            49
39
40
      // recuperar resposta
                                                            50
                                                                   cout << solve(b) - solve(a-1) << '\n';</pre>
41
                                                           51
                                                                   return 0;
42
       string ans = "";
                                                           52
      pair < int , int > curr = {n, m};
                                                           53 }
43
44
                                                              4.5
                                                                    Lis Segtree
      while (curr.first != 0 && curr.second != 0) {
45
           auto [i, j] = curr;
46
                                                            #include <bits/stdc++.h>
           if (a[i-1] == b[j-1]) {
48
               ans += a[i-1];
49
                                                            3 using namespace std;
50
51
                                                            5 const int MAX = 2e5+17;
           curr = p[i][j];
      }
53
                                                            7 struct segTree {
54
                                                                  int size;
      reverse(ans.begin(), ans.end());
55
                                                                   vector < int > tree;
                                                            9
56
                                                            10
57
      return ans;
                                                                   void init(int n) {
                                                            11
58 }
                                                                       size = 1;
                                                                       while (size < n) size *= 2;</pre>
                                                            13
  4.4 Digit Dp 2
                                                            14
                                                                       tree.assign(2 * size, OLL);
                                                            15
1 // Digit DP 2: https://cses.fi/problemset/task/2220
                                                           16
2 //
                                                                   int merge(int a, int b) {
3 // Number of integers between a and b
                                                            18
                                                                       return max(a, b);
4 // where no two adjacents digits are the same
                                                            19
                                                            20
6 #include <bits/stdc++.h>
                                                                   void build(vector<int> &arr, int x, int lx, int
                                                            21
                                                                   rx) {
8 using namespace std;
                                                                       if (rx - lx == 1) {
                                                            22
                                                                           if (lx < (int)arr.size())</pre>
9 using 11 = long long;
                                                            23
                                                                               tree[x] = arr[lx];
                                                            24
11 const int MAX = 20; // 10^18
                                                            25
                                                            26
                                                                           return;
13 ll tb[MAX][MAX][2][2];
                                                            27
                                                            28
                                                                       int m = (lx + rx) / 2;
15 ll dp(string& number, int pos, int last_digit, bool
                                                           29
      under, bool started) {
                                                                       build(arr, 2 * x + 1, lx, m);
                                                            30
      if (pos >= (int)number.size()) {
                                                                       build(arr, 2 * x + 2, m, rx);
16
                                                            31
           return 1;
                                                            32
17
      }
                                                                       tree[x] = merge(tree[2 * x + 1], tree[2 * x +
                                                                    2]);
19
      11& mem = tb[pos][last_digit][under][started];
                                                                   }
      if (mem != -1) return mem;
21
                                                            35
      mem = 0;
                                                                   void build(vector<int> &arr) {
                                                            36
22
23
                                                            37
                                                                       build(arr, 0, 0, size);
      int limit = 9;
                                                            38
24
      if (!under) limit = number[pos] - '0';
                                                                   void update(int i, int v, int x, int lx, int rx)
26
      for (int digit = 0; digit <= limit; digit++) {</pre>
           if (started && digit == last_digit) continue; 41
                                                                       if (rx - lx == 1) {
28
                                                                           tree[x] = v;
29
                                                           42
           bool is_under = under || (digit < limit);</pre>
                                                                           return;
           bool is_started = started || (digit != 0);
                                                                       }
31
                                                            44
                                                                       int m = (lx + rx) / 2;
           mem += dp(number, pos+1, digit, is_under,
33
                                                            46
      is_started);
                                                                       if (i < m) {</pre>
                                                            47
```

substring of this string

```
update(i, v, 2 * x + 1, lx, m);
                                                           _{5} // if all letters in the substring you delete are
48
49
           } else {
                                                                  equal
                update(i, v, 2 * x + 2, m, rx);
                                                            6 // calculate the minimum number of operations to
           }
                                                                  delete the whole string s
           tree[x] = merge(tree[2 * x + 1], tree[2 * x + 8 #include <bits/stdc++.h>
                                                           10 using namespace std;
54
55
                                                           11
       void update(int i, int v) {
                                                           12 const int MAX = 510;
           update(i, v, 0, 0, size);
57
                                                           13
                                                           14 int n, tb[MAX][MAX];
59
                                                            15 string s;
       int query(int 1, int r, int x, int lx, int rx) { 16
60
           if (lx >= r || l >= rx) return 0;
61
                                                           17 int dp(int left, int right) {
           if (lx >= l && rx <= r) return tree[x];</pre>
                                                                  if (left > right) return 0;
62
                                                           18
                                                                  int& mem = tb[left][right];
           int m = (1x + rx) / 2;
64
                                                           20
           int s1 = query(1, r, 2 * x + 1, lx, m);
                                                           21
                                                                  if (mem != -1) return mem;
           int s2 = query(1, r, 2 * x + 2, m, rx);
66
                                                           22
                                                                  mem = 1 + dp(left+1, right); // gastar uma
                                                           23
67
           return merge(s1, s2);
                                                                  operaÃgÃčo arrumando sÃş o cara atual
                                                                  for (int i = left+1; i <= right; i++) {</pre>
       }
69
                                                           24
                                                                      if (s[left] == s[i]) {
                                                           25
       int query(int 1, int r) {
                                                                           mem = min(mem, dp(left+1, i-1) + dp(i,
71
                                                           26
           return query(1, r, 0, 0, size);
                                                                  right));
72
73
                                                                      }
74 };
                                                           28
                                                           29
75
76
                                                           30
                                                                  return mem:
                                                           31 }
77 int main() {
       ios::sync_with_stdio(false);
78
                                                           32
       cin.tie(NULL);
                                                           33 int main() {
79
80
                                                           34
                                                                  ios::sync_with_stdio(false);
       int n, arr[MAX], aux[MAX]; cin >> n;
                                                                  cin.tie(NULL);
81
                                                           35
       for (int i = 0; i < n; i++) {</pre>
82
                                                           36
           cin >> arr[i];
                                                                  cin >> n >> s;
83
                                                           37
                                                                  memset(tb, -1, sizeof(tb));
                                                           38
84
                                                                  cout << dp(0, n-1) << '\n';
           aux[i] = arr[i];
                                                           39
85
       }
                                                           40
86
                                                           41
                                                                  return 0;
       sort(aux, aux+n);
                                                           42 }
88
89
90
       segTree st;
                                                                    Edit Distance
       st.init(n);
91
92
       int ans = 0:
93
                                                            1 // Edit Distance / Levenshtein Distance
       for (int i = 0; i < n; i++) {</pre>
                                                            2 //
           int it = lower_bound(aux, aux+n, arr[i]) -
95
                                                            3 // numero minimo de operacoes
                                                            4 // para transformar
           int lis = st.query(0, it) + 1;
96
                                                            5 // uma string em outra
97
                                                            6 //
           st.update(it, lis);
                                                            7 // tamanho da matriz da dp eh |a| x |b|
99
                                                            8 // edit_distance(a.size(), b.size(), a, b)
           ans = max(ans, lis);
100
                                                            9 //
                                                           10 // https://cses.fi/problemset/task/1639
                                                           11 //
       cout << ans << '\n';
                                                           12 // O(n^2)
104
105
       return 0;
                                                           14 int tb[MAX][MAX];
106 }
                                                           15
                                                           int edit_distance(int i, int j, string &a, string &b)
         Knapsack
                                                            17
                                                                  if (i == 0) return j;
                                                                  if (j == 0) return i;
                                                           18
         Range Dp
                                                            19
                                                           20
                                                                  int &ans = tb[i][j];
 1 // Range DP 1: https://codeforces.com/problemset/
                                                           21
       problem/1132/F
                                                                  if (ans != -1) return ans;
 2 //
                                                           23
 3 // You may apply some operations to this string
                                                                  ans = min({
                                                           24
 4 // in one operation you can delete some contiguous
                                                                      edit_distance(i-1, j, a, b) + 1,
                                                           25
```

26

 $edit_distance(i, j-1, a, b) + 1,$



```
edit_distance(i-1, j-1, a, b) + (a[i-1] != b[58]
                                                                  vector < Point > upper_hull = {points[0], points
      j-1])
                                                                   [1]};
      });
                                                                   for (int i = 2; i < n; i++) {</pre>
28
                                                            60
                                                                       upper_hull.push_back(points[i]);
29
30
      return ans;
                                                            61
31 }
                                                                       int sz = upper_hull.size();
                                                            62
                                                            63
                                                                       while (sz >= 3 && dir(upper_hull[sz-3],
       Geometry
                                                            64
                                                                   upper_hull[sz-2], upper_hull[sz-1]) == -1) {
                                                                           upper_hull.pop_back();
                                                            65
  5.1 Convex Hull
                                                                           upper_hull.pop_back();
                                                            66
                                                            67
                                                                           upper_hull.push_back(points[i]);
1 // Convex Hull - Monotone Chain
                                                            68
                                                                           SZ - -:
                                                            69
2 //
_{\rm 3} // Convex Hull is the subset of points that forms the ^{70}
                                                            71
       smallest convex polygon
_{\rm 4} // which encloses all points in the set.
                                                                   vector < Point > lower_hull = {points[n-1], points[n
6 // https://cses.fi/problemset/task/2195/
                                                                   for (int i = n-3; i \ge 0; i--) {
                                                                       lower_hull.push_back(points[i]);
7 // https://open.kattis.com/problems/convexhull (
                                                            74
      counterclockwise)
8 //
                                                                       int sz = lower_hull.size();
                                                            76
                                                            77
9 // O(n log(n))
                                                                       while (sz >= 3 && dir(lower_hull[sz-3],
                                                                  lower_hull[sz-2], lower_hull[sz-1]) == -1) {
11 typedef long long ftype;
                                                            79
                                                                           lower_hull.pop_back();
12
13 struct Point {
                                                            80
                                                                           lower_hull.pop_back();
                                                                           lower_hull.push_back(points[i]);
      ftype x, y;
                                                            81
14
                                                            82
                                                                           sz--;
15
                                                                       }
      Point() {};
                                                            83
                                                                  }
                                                            84
      Point(ftype x, ftype y) : x(x), y(y) {};
17
                                                            85
18
                                                                   // reverse(lower_hull.begin(), lower_hull.end());
                                                            86
      bool operator < (Point o) {</pre>
19
           if (x == o.x) return y < o.y;</pre>
                                                                    // counterclockwise
20
           return x < o.x;</pre>
                                                            87
                                                                   for (int i = (int)lower_hull.size() - 2; i > 0; i
                                                            88
22
                                                                  --) {
23
                                                                       upper_hull.push_back(lower_hull[i]);
24
      bool operator == (Point o) {
                                                            89
          return x == o.x && y == o.y;
                                                            90
                                                                  7
25
                                                            91
26
                                                            92
                                                                   return upper_hull;
27 };
                                                            93 }
29 ftype cross(Point a, Point b, Point c) {
                                                                   General
      // v: a -> c
                                                              6
30
31
      // w: a -> b
32
                                                                    Base Converter
                                                              6.1
      // v: c.x - a.x, c.y - a.y
33
      // w: b.x - a.x, b.y - a.y
34
                                                            const string digits = "0123456789
35
                                                                  ABCDEFGHIJKLMNOPQRSTUVWXYZ";
      return (c.x - a.x) * (b.y - a.y) - (c.y - a.y) *
36
      (b.x - a.x);
37 }
                                                            3 11 tobase10(string number, int base) {
                                                                  map < char , int > val;
38
39 ftype dir(Point a, Point b, Point c) {
                                                                   for (int i = 0; i < digits.size(); i++) {</pre>
      // 0 -> colineares
                                                                       val[digits[i]] = i;
40
                                                            6
      // -1 -> esquerda
                                                            7
41
      // 1 -> direita
42
                                                                  ll ans = 0, pot = 1;
43
                                                            9
      ftype cp = cross(a, b, c);
                                                            10
                                                                   for (int i = number.size() - 1; i >= 0; i--) {
45
                                                            11
      if (cp == 0) return 0;
                                                                       ans += val[number[i]] * pot;
                                                            12
46
       else if (cp < 0) return -1;
                                                                       pot *= base;
47
                                                            13
      else return 1;
                                                            14
48
49 }
                                                            15
50
                                                            16
                                                                  return ans;
51 vector < Point > convex_hull(vector < Point > points) {
                                                           17 }
      sort(points.begin(), points.end());
52
                                                            18
      points.erase( unique(points.begin(), points.end() 19 string frombase10(ll number, int base) {
      ), points.end()); // somente pontos distintos
                                                                  if (number == 0) return "0";
                                                           20
      int n = points.size();
54
                                                            21
                                                                   string ans = "";
                                                            22
      if (n == 1) return { points[0] };
56
                                                            23
                                                                   while (number > 0) {
                                                            24
57
```

6.4 Bitwise

```
1 #define lcm(a,b) (a*b)/gcd(a,b)
           ans += digits[number % base];
25
26
          number /= base;
                                                           2 #define msb(n) (32 - builtin_clz(n))
27
                                                                  Min Priority Queue
                                                             6.5
28
      reverse(ans.begin(), ans.end());
30
                                                           1 template < class T > using min_priority_queue =
31
      return ans;
                                                                 priority_queue < T, vector < T > , greater < T >>;
32 }
                                                                  Input By File
33
                                                             6.6
34 // verifica se um nÞmero estÃą na base especificada
35 bool verify_base(string num, int base) {
                                                           1 freopen("file.in", "r", stdin);
      map < char , int > val;
                                                           2 freopen("file.out", "w", stdout);
      for (int i = 0; i < digits.size(); i++) {</pre>
37
          val[digits[i]] = i;
38
                                                             6.7
                                                                    Read
39
40
                                                           1 /*
41
      for (auto digit : num) {
                                                                 RELER O ENUNCIADO!
                                                           2
          if (val[digit] >= base) {
42
                                                           3
                                                                 WA? coloca long long que passa;
43
               return false;
                                                                 testar casos de borda, n = 0? n = 1? todos os
                                                           4
          }
44
                                                                 numeros iguais?
      }
45
                                                                 Uma resposta Ãştima pode ter tamanho 2?
                                                           5
46
                                                           6
                                                                 pode ser DP? nÃčo Ãľ guloso de mais?
      return true;
47
                                                                 d\ddot{A}q pra modelar como grafo?
48 }
                                                                 pode ser fluxo?
                                                                 as vezes compensa mais fazer um brute (em python)
        First True
                                                          10
                                                                 nada funcionou? coda do zero
                                                          12 */
    first_true(2, 10, [](int x) { return x * x >= 30;
      }); // outputs 6
                                                                  Compilation Flags
                                                             6.8
    [1, r]
                                                           1 /*
                                                                 // josÃľ
    if none of the values in the range work, return hi
                                                                 g++ -std=c++17 -Wshadow -Wall -Wextra -Wformat=2
                                                                 -Wconversion -fsanitize=address, undefined -fno-
                                                                 sanitize-recover -Wfatal-errors
    f(1) = false
    f(2) = false
                                                                 // maxwell
                                                           5
    f(3) = false
10
    f(4) = false
11
                                                           7 */
    f(5) = false
    f(6) = true
13
                                                                   Xor 1 To N
                                                             6.9
    f(7) = true
14
    f(8) = true
15
                                                           _{1} // XOR sum from 1 to N
16 */
                                                           2 ll xor_1_to_n(ll n) {
17
                                                                 if (n % 4 == 0) {
18 int first_true(int lo, int hi, function < bool(int) > f)
                                                                     return n;
       {
                                                                 } else if (n % 4 == 1) {
      hi++:
19
                                                                     return 1;
      while (lo < hi) {</pre>
                                                                   else if (n % 4 == 2) {
          int mid = lo + (hi - lo) / 2;
21
                                                                     return n + 1;
                                                           9
          if (f(mid)) {
23
                                                          10
              hi = mid;
24
                                                          11
                                                                 return 0;
          } else {
                                                          12 }
26
               lo = mid + 1:
27
                                                             6.10 Next Permutation
28
      }
      return lo;
29
30 }
                                                           1 // output: 1,2,3; 1,3,2; 2,1,3; 2,3,1; 3,1,2; 3,2,1;
  6.3 Random
                                                           3 vector<int> arr = {1, 2, 3};
                                                           4 int n = arr.size();
1 random_device dev;
                                                           6 do {
2 mt19937 rng(dev());
                                                                 for (auto e : arr) {
                                                                     cout << e << ' ';
4 uniform_int_distribution < mt19937::result_type > dist
      (1, 6); // distribution in range [1, 6]
                                                                 cout << '\n';
                                                          11 } while (next_permutation(arr.begin(), arr.end()));
6 int val = dist(rng);
```

Template Full

6.11

```
6.14 Last True
#include <bits/stdc++.h>
2 #define debug(x) cout << "[" << #x << " = " << x << "</pre>
      ] "
3 #define ff first
                                                               last_true(2, 10, [](int x) { return x * x <= 30; })
4 #define ss second
                                                                 ; // outputs 5
6 using namespace std;
                                                               [1, r]
                                                           4
7 using 11 = long long;
8 using ld = long double;
                                                               if none of the values in the range work, return lo
                                                           6
9 using pii = pair < int , int >;
10 using vi = vector<int>;
                                                               f(1) = true
                                                           8
using tii = tuple <int,int,int>;
                                                               f(2) = true
                                                               f(3) = true
13
                                                           10
14 const int oo = (int)1e9 + 17; //INF to INT
                                                               f(4) = true
                                                          11
15 const 11 00 = 0x3f3f3f3f3f3f3f3f3fLL; //INF to LL
                                                               f(5) = true
                                                          12
                                                               f(6) = false
                                                          13
17 void solve(){
                                                               f(7) = false
                                                          14
18
                                                               f(8) = false
                                                          1.5
19 }
20
                                                               last_true(1, 8, f) = 5
                                                          17
21 int main() {
                                                               last_true(7, 8, f) = 6
                                                          18
      ios::sync_with_stdio(false);
22
                                                          19 */
      cin.tie(NULL);
                                                          20
24
                                                          21 int last_true(int lo, int hi, function < bool(int) > f)
      int t = 1;
25
                                                                 {
      cin >> t;
26
                                                                 lo--;
                                                          22
      while(t--){
                                                                 while (lo < hi) {</pre>
27
                                                          23
          solve();
                                                                     int mid = lo + (hi - lo + 1) / 2;
                                                          24
      }
29
                                                          25
30 }
                                                                     if (f(mid)) {
                                                          26
                                                          27
                                                                         lo = mid;
  6.12 Goto
                                                                     } else {
                                                          28
                                                          29
                                                                          hi = mid - 1;
                                                                     }
                                                          30
1 while (t--) {
                                                                 }
                                                          31
    for (int d = 0; d < 11; d++) {</pre>
                                                          32
                                                                 return lo;
      if (n % 11 == 0) {
                                                          33 }
        cout << "YES" << endl;</pre>
         goto done;
                                                                     Template
                                                             6.15
                                                           1 #include <bits/stdc++.h>
      n = 111;
      if (n < 0) break;
                                                           3 using namespace std;
    }
10
    cout << "NO" << endl;
11
                                                           5 int main() {
                                                                 ios::sync_with_stdio(false);
13
    done:;
                                                                 cin.tie(NULL);
14 }
  6.13 Split
                                                          10
                                                                 return 0:
                                                          11
vector<string> split(string s, char key=' ') {
                                                          12 }
      vector<string> ans;
                                                                    Ordered Set
                                                             6.16
      string aux = "";
      for (int i = 0; i < (int)s.size(); i++) {</pre>
                                                           #include <ext/pb_ds/assoc_container.hpp>
           if (s[i] == key) {
                                                           #include <ext/pb_ds/tree_policy.hpp>
               if (aux.size() > 0) {
                   ans.push_back(aux);
                                                           4 using namespace __gnu_pbds;
                   aux = "":
9
              }
10
                                                           6 typedef tree <
                                                                 int,
          } else {
11
               aux += s[i];
                                                                 null_type,
           }
                                                                 less<int>,
13
                                                           9
      }
14
                                                          10
                                                                 rb_tree_tag,
                                                                 tree_order_statistics_node_update> ordered_set;
15
                                                          11
      if ((int)aux.size() > 0) {
                                                          12
16
          ans.push_back(aux);
                                                          void Erase(ordered_set& a, int x){
17
      }
                                                               int r = a.order_of_key(x);
18
                                                          14
                                                                 auto it = a.find_by_order(r);
19
                                                          15
                                                                 a.erase(it);
20
      return ans;
                                                          16
21 }
                                                          17 }
```

```
18
                                                              18
19 /*
                                                              19
                                                                     return ans;
       order_of_key(k) // Number of items strictly
                                                             20 }
20
       smaller than k.
       find_by_order(k) // K-th element in a set (
                                                                     DS
       counting from zero).
22 */
                                                                7.1
                                                                       Segtree2d
24 /* os parametros sÃčo, na ordem:
25 1) int -> tipo do valor que quero inserir(key), pode
                                                              1 // Description:
      ser int, double, pii e etc
                                                              2 // Indexed at zero
26 2) null_type -> Ãľ para usar essa Ãąrvore como set/
                                                              _{\rm 3} // Given a N x M grid, where i represents the row and
      \verb|multiset|. D\tilde{A} \verb|q| pra usar essa ED como map tmb|
                                                                     j the column, perform the following operations
       trocando isso pra um tipo map
                                                              4 // update(j, i) - update the value of grid[i][j]
27 3) less<int> -> forma de compara\tilde{A}g\tilde{A}čo dos elementos.
                                                              _{5} // query(j1, j2, i1, i2) - return the sum of values
      less<int>, less_equal<int>, greater,
                                                                     inside the rectangle
       greater_equal e etc
                                                              6 // defined by grid[i1][j1] and grid[i2][j2] inclusive
_{28} 4) rb_tree_tag -> tipo de 	ilde{\mathtt{A}}ąrvore a ser usado, usar a _{7}
       rb para as operaÃğÃţes serem O(logN)
                                                              8 // Problem:
29 5) tree_order_statistics_node__update -> contÃľm
                                                              9 // https://cses.fi/problemset/task/1739/
       vÃarias operaÃgÃtes para atualizar "tree-based
       container", usado pra manter algo como o numero
                                                              11 // Complexity:
       de nodes em alguma subÃarvore
                                                              12 // Time complexity:
                                                              _{13} // O(log N * log M) for both query and update
31 usar less<equal> em (3) para usar como set e
                                                              _{14} // O(N * M) for build
       less_equal <TIPO > para usar como multiset (
                                                              15 // Memory complexity:
       permitir elementos repetidos).
                                                              16 // 4 * M * N
_{
m 32} se em (3) colocar greater_equal, entao o order_of_key _{
m 17}
       (k) retorna a quantidade de valores maiores que k _{18} // How to use:
        (ÃI preferivel usar less_equal e retornar o
                                                              19 // Segtree2D seg = Segtree2D(n, n);
       tamanho do intervalo - order)
                                                             _{20} // vector<vector<int>> v(n, vector<int>(n));
33 */
                                                             21 // seg.build(v);
34
35 /* Como usar
                                                             23 // Notes
   .find_by_order(k) -> retorna um ITERADOR para o
                                                              24 // Indexed at zero
       elemento na posi\tilde{\mathbf{A}}g\tilde{\mathbf{A}}čo kth (contando do 0, ou seja_{25}
       k = 0 retorna o menor)
                                                             26 struct Segtree2D {
       Usar *(find_by_order(k)) para saber QUAL ÃL o
                                                                    const int MAXN = 1025;
                                                             27
      numero na posiÃğÃčo.
                                                                     int N, M;
                                                             28
38
39
   .order_of_key(k) -> retorna o numero de valores na
                                                             30
                                                                    vector < vector < int >> seg;
       Ãąrvore que estÃčo ESTRITAMENTE menores que k;
       retorna um inteiro
40
                                                             32
                                                                     Segtree2D(int N, int M) {
41
                                                                         this -> N = N;
                                                             33
   TambÃľm Ãľ possivel usar as funÃgÃţes que o set
42
                                                              34
                                                                         this -> M = M:
       comum possui, como o .insert().
                                                                         seg.resize(2*MAXN, vector<int>(2*MAXN));
                                                             35
   DÃą pra percorrer os valores inseridos na Ãąrvore
                                                              36
      usando o for(auto p: tree);
                                                              37
   {\tt Obs.:} \ {\tt O} \ . {\tt erase} \ {\tt funciona} \, , \ {\tt mas} \ {\tt precisa} \ {\tt de} \ {\tt um} \ {\tt iterador}
                                                                     void buildY(int noX, int lX, int rX, int noY, int
       para o elemento! Para isso tem a funÃgÃčo Erase
                                                                      1Y, int rY, vector < vector < int >> &v) {
       implementada.
                                                                         if(1Y == rY){
                                                              39
                                                                              if(1X == rX){
                                                              40
                                                                                  seg[noX][noY] = v[rX][rY];
                                                             41
  6.17 Get Subset Sums
                                                              42
                                                                             }else{
                                                                                  seg[noX][noY] = seg[2*noX+1][noY] +
using ll = long long;
                                                                     seg[2*noX+2][noY];
                                                                             }
3 vector<ll> get_subset_sums(int 1, int r, vector<ll> & 45
                                                                         }else{
       arr) {
                                                                             int m = (1Y+rY)/2;
       vector <11> ans;
                                                              47
                                                                              buildY(noX, 1X, rX, 2*noY+1, 1Y, m, v);
                                                              48
                                                                             buildY(noX, 1X, rX, 2*noY+2, m+1, rY, v);
       int len = r-l+1;
                                                              49
       for (int i = 0; i < (1 << len); i++) {</pre>
                                                             50
           11 sum = 0;
                                                             51
                                                                             seg[noX][noY] = seg[noX][2*noY+1] + seg[
                                                                    noX][2*noY+2];
           for (int j = 0; j < len; j++) {</pre>
10
                                                                         }
                if (i&(1 << j)) {</pre>
                    sum += arr[1 + j];
                                                             54
                                                                     void buildX(int noX, int lX, int rX, vector<</pre>
                                                             55
           }
                                                                    vector < int >> &v) {
14
                                                                         if(1X != rX){
                                                                             int m = (1X+rX)/2;
16
           ans.push_back(sum);
```

60

61

62

64

66

68

71

72

73

74

76

78

80

81

83

84 85

86

87

89

90

91

92 93

94

95 96

97

98

99

100

104

106

107

108

109

111

113

114

116

117

```
buildX(2*noX+1, 1X, m, v);
        buildX(2*noX+2, m+1, rX, v);
                                                    118
    }
                                                    119
    buildY(noX, 1X, rX, 0, 0, M - 1, v);
                                                    120
}
                                                    121
void updateY(int noX, int lX, int rX, int noY,
int lY, int rY, int y){
                                                    124
    if (1Y == rY){
        if (1X == rX) {
                                                    126
             seg[noX][noY] = !seg[noX][noY];
                                                    127
        }else{
                                                    128
             seg[noX][noY] = seg[2*noX+1][noY] +
                                                    129
seg[2*noX+2][noY];
                                                    130
        }
                                                    131
    }else{
                                                    132
        int m = (1Y+rY)/2;
                                                    133 }:
        if(y \le m)
             updateY(noX, 1X, rX, 2*noY+1,1Y, m, y
);
        else if(m < v)
             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 23 typedef pii ftype;
aY, int bY){
    if(aY <= 1Y && rY <= bY) return seg[noX][noY 25 struct Segtree {
];
                                                     26
                                                     27
    int m = (1Y+rY)/2;
                                                     28
    if(bY <= m) return queryY(noX, 2*noY+1, 1Y, m<sub>30</sub>
, aY, bY);
                                                     31
   if(m < aY) return queryY(noX, 2*noY+2, m+1,</pre>
rY, aY, bY);
                                                     33
    return queryY(noX, 2*noY+1, 1Y, m, aY, bY) +
                                                     35
queryY(noX, 2*noY+2, m+1, rY, aY, bY);
                                                     36
                                                     37
                                                     38
int queryX(int noX, int lX, int rX, int aX, int
                                                     39
bX, int aY, int bY){
                                                     40
    if(aX <= lX && rX <= bX) return queryY(noX,</pre>
                                                     41
0, 0, M - 1, aY, bY);
                                                     42
                                                     43
    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 _{47}
```

```
, bX, aY, bY);
    return queryX(2*noX+1, lX, m, aX, bX, aY, bY)
 + queryX(2*noX+2, m+1, rX, aX, bX, aY, bY);
void build(vector<vector<int>> &v) {
    buildX(0, 0, N - 1, v);
int query(int aX, int bX, int aY, int bY) {
    return queryX(0, 0, N - 1, aX, bX, aY, bY);
void update(int x, int y) {
    updateX(0, 0, N - 1, x, y);
```

Minimum And Amount

```
1 // Description:
_{2} // Query - get minimum element in a range (1, r)
      inclusive
_{\rm 3} // and also the number of times it appears in that
      range
4 // Update - update element at position id to a value
6 // Problem:
{\scriptsize 7~//~https://codeforces.com/edu/course/2/lesson/4/1/}
      practice/contest/273169/problem/C
9 // Complexity:
10 // O(\log n) for both query and update
_{12} // How to use:
13 // Segtree seg = Segtree(n);
14 // seg.build(v);
16 #define pii pair <int, int>
17 #define mp make_pair
18 #define ff first
19 #define ss second
21 const int INF = 1e9+17;
      vector<ftype> seg;
       int n;
       const ftype NEUTRAL = mp(INF, 0);
       Segtree(int n) {
           int sz = 1;
           while (sz < n) sz *= 2;
           this -> n = sz:
           seg.assign(2*sz, NEUTRAL);
       ftype f(ftype a, ftype b) {
           if (a.ff < b.ff) return a;</pre>
           if (b.ff < a.ff) return b;</pre>
           return mp(a.ff, a.ss + b.ss);
       ftype query(int pos, int ini, int fim, int p, int
        q) {
           if (ini >= p && fim <= q) {</pre>
               return seg[pos];
```

50

51

53

55

56

58

60

61

63

65

66

68

70

71

75

76

77

78

79

80 81

82

83

84

86

88

89

90

91

93

94

95

96

98

99

100

102

104

106

108

109

110

111

112

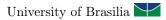
113

114

115

116

```
}
                                                  118 };
                                                            Segment With Maximum Sum
                                                     7.3
    if (q < ini || p > fim) {
        return NEUTRAL;
                                                   1 // Description:
                                                    2 // Query - get sum of segment that is maximum among
    int e = 2*pos + 1;
                                                          all segments
    int d = 2*pos + 2;
                                                    3 // E.g
    int m = ini + (fim - ini) / 2;
                                                    4 // Array: 5 -4 4 3 -5
                                                    _{5} // Maximum segment sum: 8 because 5 + (-4) + 4 = 8
    return f(query(e, ini, m, p, q), query(d, m + _6 // Update - update element at position id to a value
 1, fim, p, q));
                                                         val
}
                                                    8 // Problem:
void update(int pos, int ini, int fim, int id,
                                                    9 // https://codeforces.com/edu/course/2/lesson/4/2/
int val) {
                                                          practice/contest/273278/problem/A
    if (ini > id || fim < id) {</pre>
                                                   10
                                                   11 // Complexity:
        return;
    }
                                                   _{12} // O(log n) for both query and update
    if (ini == id && fim == id) {
                                                   ^{14} // How to use:
        seg[pos] = mp(val, 1);
                                                   15 // Segtree seg = Segtree(n);
                                                   16 // seg.build(v);
        return;
                                                   17
    }
                                                   18 // Notes
                                                   19 // The maximum segment sum can be a negative number
    int e = 2*pos + 1;
                                                   20 // In that case, taking zero elements is the best
    int d = 2*pos + 2;
                                                          choice
    int m = ini + (fim - ini) / 2;
                                                   21 // So we need to take the maximum between 0 and the
                                                         query
    update(e, ini, m, id, val);
                                                   22 // max(OLL, seg.query(0, n).max_seg)
    update(d, m + 1, fim, id, val);
                                                   23
                                                   24 using ll = long long;
    seg[pos] = f(seg[e], seg[d]);
}
                                                   26 typedef ll ftype_node;
void build(int pos, int ini, int fim, vector<int> 28 struct Node {
                                                          ftype_node max_seg;
                                                   29
    if (ini == fim) {
                                                          ftype_node pref;
                                                   30
        if (ini < (int)v.size()) {</pre>
                                                          ftype_node suf;
                                                   31
            seg[pos] = mp(v[ini], 1);
                                                   32
                                                          ftype_node sum;
        }
                                                   33
        return:
                                                   34
                                                          Node(ftype_node max_seg, ftype_node pref,
    }
                                                          ftype_node suf, ftype_node sum) : max_seg(max_seg
                                                          ), pref(pref), suf(suf), sum(sum) {};
    int e = 2*pos + 1;
                                                   35 };
    int d = 2*pos + 2;
    int m = ini + (fim - ini) / 2;
                                                  37 typedef Node ftype;
                                                   38
    build(e, ini, m, v);
                                                   39 struct Segtree {
    build(d, m + 1, fim, v);
                                                          vector<ftype> seg;
                                                   40
                                                   41
    seg[pos] = f(seg[e], seg[d]);
                                                          const ftype NEUTRAL = Node(0, 0, 0, 0);
                                                   42
}
                                                   43
                                                          Segtree(int n) {
                                                   44
ftype query(int p, int q) {
                                                   45
                                                              int sz = 1;
    return query(0, 0, n - 1, p, q);
                                                              // potencia de dois mais proxima
                                                   46
                                                              while (sz < n) sz *= 2;
                                                   47
                                                              this->n = sz;
void update(int id, int val) {
                                                   49
    update(0, 0, n - 1, id, val);
                                                              // numero de nos da seg
                                                   50
                                                   51
                                                              seg.assign(2*sz, NEUTRAL);
                                                   52
void build(vector<int> &v) {
                                                   53
    build(0, 0, n - 1, v);
                                                          ftype f(ftype a, ftype b) {
                                                   54
                                                              ftype_node max_seg = max({a.max_seg, b.
                                                          max_seg, a.suf + b.pref});
void debug() {
                                                              ftype_node pref = max(a.pref, a.sum + b.pref)
                                                   56
    for (auto e : seg) {
        cout << e.ff << ' ' << e.ss << '\n';
                                                              ftype_node suf = max(b.suf, b.sum + a.suf);
                                                   57
                                                              ftype_node sum = a.sum + b.sum;
    cout << '\n';</pre>
                                                   59
}
                                                              return Node(max_seg, pref, suf, sum);
                                                   60
```



```
}
                                                                   void build(vector<int> &v) {
61
                                                            128
                                                            129
                                                                        build(0, 0, n - 1, v);
62
       ftype query(int pos, int ini, int fim, int p, int130
63
        a) {
                                                                   void debug() {
              (ini >= p && fim <= q) {
                return seg[pos];
                                                                       for (auto e : seg) {
65
                                                           133
                                                                            cout << e.max_seg << ' ' ' << e.pref << ' '
           }
66
                                                            134
                                                                    << e.suf << ' ' << e.sum << '\n';
67
            if (q < ini || p > fim) {
68
                                                            135
                return NEUTRAL;
                                                                        cout << '\n';
                                                            136
           }
70
                                                            137
                                                            138 };
           int e = 2*pos + 1;
                                                                    Persistent
                                                               7.4
           int d = 2*pos + 2;
74
           int m = ini + (fim - ini) / 2;
                                                             1 // Description:
            return f(query(e, ini, m, p, q), query(d, m + _2 // Persistent segtree allows for you to save the
        1, fim, p, q));
                                                                   different versions of the segtree between each
77
                                                                   update
78
                                                             3 // Indexed at one
       void update(int pos, int ini, int fim, int id,
                                                             4 // Query - get sum of elements from range (1, r)
       int val) {
                                                                   inclusive
           if (ini > id || fim < id) {</pre>
80
                                                             _{5} // Update - update element at position id to a value
                return;
                                                                   val
           }
82
                                                             7 // Problem:
83
            if (ini == id && fim == id) {
84
                                                             8 // https://cses.fi/problemset/task/1737/
                seg[pos] = Node(val, val, val, val);
85
                                                            10 // Complexity:
                return;
87
                                                            _{11} // O(log n) for both query and update
           }
88
89
                                                            13 // How to use:
            int e = 2*pos + 1;
90
                                                            14 // vector <int > raiz(MAX); // vector to store the
            int d = 2*pos + 2;
                                                                   roots of each version
            int m = ini + (fim - ini) / 2;
92
                                                            15 // Segtree seg = Segtree(INF);
                                                            16 // raiz[0] = seg.create(); // null node
           update(e, ini, m, id, val);
94
                                                            17 // curr = 1; // keep track of the last version
           update(d, m + 1, fim, id, val);
95
                                                            19 // raiz[k] = seg.update(raiz[k], idx, val); //
96
           seg[pos] = f(seg[e], seg[d]);
97
                                                                   updating version k
                                                            _{\rm 20} // seg.query(raiz[k], l, r) // querying version k
99
                                                             21 // raiz[++curr] = raiz[k]; // create a new version
       void build(int pos, int ini, int fim, vector<int>
100
                                                                  based on version k
        &v) {
            if (ini == fim) {
                                                            23 const int MAX = 2e5+17:
                // se a posiÃğÃčo existir no array
                                                            24 const int INF = 1e9+17;
       original
                                                            25
                // seg tamanho potencia de dois
                                                            26 typedef long long ftype;
                if (ini < (int)v.size()) {</pre>
104
                                                            27
                    seg[pos] = Node(v[ini], v[ini], v[ini<sub>28</sub> struct Segtree {
       ], v[ini]);
                                                                   vector<ftype> seg, d, e;
                                                            29
               }
106
                                                                   const ftype NEUTRAL = 0;
                                                            30
                return;
                                                            31
                                                                   int n:
           }
108
                                                            32
109
                                                            33
                                                                   Segtree(int n) {
            int e = 2*pos + 1;
110
                                                                       this -> n = n;
                                                            34
            int d = 2*pos + 2;
                                                            35
            int m = ini + (fim - ini) / 2;
                                                            36
113
                                                                   ftype f(ftype a, ftype b) {
                                                            37
            build(e, ini, m, v);
114
                                                            38
                                                                       return a + b;
           build(d, m + 1, fim, v);
                                                            39
116
                                                            40
            seg[pos] = f(seg[e], seg[d]);
                                                                   ftype create() {
                                                            41
       }
                                                                        seg.push_back(0);
118
                                                            42
119
                                                            43
                                                                        e.push_back(0);
120
       ftype query(int p, int q) {
                                                            44
                                                                        d.push_back(0);
           return query(0, 0, n - 1, p, q);
121
                                                            45
                                                                        return seg.size() - 1;
       }
                                                            46
123
                                                            47
       void update(int id, int val) {
124
                                                                   ftype query(int pos, int ini, int fim, int p, int
                                                            48
            update(0, 0, n - 1, id, val);
125
126
                                                                        if (q < ini || p > fim) return NEUTRAL;
                                                            49
127
                                                                        if (pos == 0) return 0;
                                                            50
```

```
if (p <= ini && fim <= q) return seg[pos]; 26 struct Segtree {</pre>
51
           int m = (ini + fim) >> 1;
                                                           27
                                                                  vector < ftype > seg;
           return f(query(e[pos], ini, m, p, q), query(d28
                                                                  int n;
       [pos], m + 1, fim, p, q);
                                                                  const ftype NEUTRAL = 0;
                                                                  Segtree(int n) {
                                                           31
       int update(int pos, int ini, int fim, int id, int 32
                                                                      int sz = 1;
       val) {
                                                                      while (sz < n) sz *= 2;
           int novo = create();
                                                                      this -> n = sz;
           seg[novo] = seg[pos];
                                                                      seg.assign(2*sz, NEUTRAL);
59
                                                           36
           e[novo] = e[pos];
                                                           37
           d[novo] = d[pos];
61
                                                           38
                                                                  ftype f(ftype a, ftype b) {
                                                           39
62
63
           if (ini == fim) {
                                                           40
                                                                     return a + b;
               seg[novo] = val;
                                                           41
64
               return novo;
                                                           42
           }
                                                                  ftype query(int pos, int ini, int fim, int p, int
66
                                                           43
                                                                   q) {
                                                                      if (ini >= p && fim <= q) {</pre>
          int m = (ini + fim) >> 1;
68
                                                           44
                                                                          return seg[pos];
69
           if (id <= m) e[novo] = update(e[novo], ini, m 46</pre>
70
       , id, val);
           else d[novo] = update(d[novo], m + 1, fim, id 48
                                                                      if (q < ini || p > fim) {
       , val);
                                                                          return NEUTRAL;
72
                                                           50
           seg[novo] = f(seg[e[novo]], seg[d[novo]]);
73
                                                           51
                                                                      int e = 2*pos + 1;
74
                                                           52
           return novo;
                                                                      int d = 2*pos + 2;
                                                                      int m = ini + (fim - ini) / 2;
      }
76
                                                           54
                                                           55
      ftype query(int pos, int p, int q) {
                                                                      return f(query(e, ini, m, p, q), query(d, m +
78
                                                           56
          return query(pos, 1, n, p, q);
                                                                   1, fim, p, q));
79
                                                           57
81
                                                           58
       int update(int pos, int id, int val) {
                                                                  void update(int pos, int ini, int fim, int id,
82
           return update(pos, 1, n, id, val);
                                                                  int val) {
83
                                                                      if (ini > id || fim < id) {</pre>
84
85 };
                                                                          return;
                                                           61
                                                           62
  7.5 Range Query Point Update
                                                                      if (ini == id && fim == id) {
                                                           64
                                                                          seg[pos] = val;
1 // Description:
2 // Indexed at zero
                                                                          return;
3 // Query - get sum of elements from range (1, r)
                                                           67
                                                                      }
      inclusive
_4 // Update - update element at position id to a value ^{69}
                                                                      int e = 2*pos + 1;
      val
                                                                      int d = 2*pos + 2;
                                                           71
                                                                      int m = ini + (fim - ini) / 2;
                                                           72
6 // Problem:
7 // https://codeforces.com/edu/course/2/lesson/4/1/
                                                           73
                                                                      update(e, ini, m, id, val);
      practice/contest/273169/problem/B
                                                           74
                                                                      update(d, m + 1, fim, id, val);
9 // Complexity:
                                                           76
                                                           77
                                                                      seg[pos] = f(seg[e], seg[d]);
_{10} // O(log n) for both query and update
                                                           78
                                                           79
12 // How to use:
                                                                  void build(int pos, int ini, int fim, vector<int>
13 // Segtree seg = Segtree(n);
                                                           80
                                                                   &v) {
14 // seg.build(v);
                                                                      if (ini == fim) {
                                                           81
                                                                          if (ini < (int)v.size()) {</pre>
16 // Notes
                                                                              seg[pos] = v[ini];
_{17} // Change neutral element and f function to perform a ^{83}
                                                                          }
       different operation
                                                                          return:
                                                           85
                                                                      }
19 // If you want to change the operations to point
                                                           87
      query and range update
                                                                      int e = 2*pos + 1;
20 // Use the same segtree, but perform the following
                                                                      int d = 2*pos + 2;
      operations
                                                                      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
                                                                      build(d, m + 1, fim, v);
24 typedef long long ftype;
                                                           93
                                                           94
```

```
seg[pos] = f(seg[e], seg[d]);
95
96
97
       ftype query(int p, int q) {
98
           return query(0, 0, n - 1, p, q);
99
100
       void update(int id, int val) {
            update(0, 0, n - 1, id, val);
103
104
106
       void build(vector<int> &v) {
           build(0, 0, n - 1, v);
108
109
       void debug() {
110
            for (auto e : seg) {
                cout << e << ' ';
112
            cout << '\n';
114
116 };
```

7.6 Lazy Assignment To Segment

const long long INF = 1e18+10;

```
70
3 typedef long long ftype;
                                                             71
                                                             72
5 struct Segtree {
                                                             73
                                                             74
      vector<ftype> seg;
       vector<ftype> lazy;
                                                             75
                                                             76
       int n;
       const ftype NEUTRAL = 0;
9
       const ftype NEUTRAL_LAZY = -INF;
       Segtree(int n) {
12
                                                             79
13
           int sz = 1;
           // potencia de dois mais proxima
                                                             80
14
           while (sz < n) sz *= 2;
                                                             81
           this->n = sz;
16
                                                             83
           // numero de nos da seg
                                                             84
18
                                                             85
           seg.assign(2*sz, NEUTRAL);
19
20
           lazy.assign(2*sz, NEUTRAL_LAZY);
                                                             86
21
                                                             88
       ftype apply_lazy(ftype a, ftype b, int len) {
23
24
           if (b == NEUTRAL_LAZY) return a;
                                                             90
                                                             91
           if (a == NEUTRAL_LAZY) return b * len;
25
                                                             92
           else return b * len;
26
                                                             93
28
       void propagate(int pos, int ini, int fim) {
                                                             94
          <u>if</u> (ini == fim) {
                                                             95
30
               return;
31
           }
                                                             96
32
                                                             97
33
           int e = 2*pos + 1;
                                                             98
           int d = 2*pos + 2;
35
           int m = ini + (fim - ini) / 2;
36
           lazy[e] = apply_lazy(lazy[e], lazy[pos], 1); 102
38
39
           lazy[d] = apply_lazy(lazy[d], lazy[pos], 1); 103
40
           seg[e] = apply_lazy(seg[e], lazy[pos], m -
                                                            105
41
       ini + 1):
                                                            106
           seg[d] = apply_lazy(seg[d], lazy[pos], fim - 107
42
      m):
                                                            108
                                                            109
43
           lazy[pos] = NEUTRAL_LAZY;
                                                            110
                                                            111
       }
45
                                                            112
46
```

```
ftype f(ftype a, ftype b) {
   return a + b;
ftype query(int pos, int ini, int fim, int p, int
q) {
    propagate(pos, ini, fim);
    if (ini >= p && fim <= q) {</pre>
        return seg[pos];
    if (q < ini || p > fim) {
        return NEUTRAL;
    7
    int e = 2*pos + 1;
    int d = 2*pos + 2;
    int m = ini + (fim - ini) / 2;
    return f(query(e, ini, m, p, q), query(d, m +
 1, fim, p, q));
void update(int pos, int ini, int fim, int p, int
 q, int val) {
    propagate(pos, ini, fim);
    if (ini > q || fim < p) {</pre>
        return;
    if (ini >= p && fim <= q) {</pre>
        lazy[pos] = apply_lazy(lazy[pos], val, 1)
        seg[pos] = apply_lazy(seg[pos], val, fim
- ini + 1);
        return;
    }
    int e = 2*pos + 1;
    int d = 2*pos + 2;
    int m = ini + (fim - ini) / 2;
    update(e, ini, m, p, q, val);
    update(d, m + 1, fim, p, q, val);
    seg[pos] = f(seg[e], seg[d]);
void build(int pos, int ini, int fim, vector<int>
    if (ini == fim) {
        // se a posiÃğÃčo existir no array
original
        // seg tamanho potencia de dois
        if (ini < (int)v.size()) {</pre>
            seg[pos] = v[ini];
        return;
    }
    int e = 2*pos + 1;
    int d = 2*pos + 2;
    int m = ini + (fim - ini) / 2;
    build(e, ini, m, v);
    build(d, m + 1, fim, v);
    seg[pos] = f(seg[e], seg[d]);
}
```

47 48

49

50

51

53

54

55 56

57

58

59

60

61

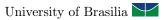
62

63

65

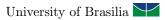
66

67



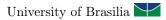
```
ftype query(int p, int q) {
                                                            41
114
           return query(0, 0, n - 1, p, q);
                                                            42
                                                                   ftype create() {
                                                                       seg.push_back(0);
115
                                                            43
                                                                       e.push_back(0);
116
                                                            44
       void update(int p, int q, int val) {
                                                                       d.push_back(0);
                                                            45
           update(0, 0, n - 1, p, q, val);
                                                                       return seg.size() - 1;
118
                                                            46
119
                                                            47
120
                                                            48
                                                                   ftype query(int pos, int ini, int fim, int p, int
       void build(vector<int> &v) {
121
                                                            49
           build(0, 0, n - 1, v);
                                                                       if (q < ini || p > fim) return NEUTRAL;
123
                                                            50
124
                                                            51
                                                                       if (pos == 0) return 0;
                                                                       if (p <= ini && fim <= q) return seg[pos];</pre>
125
       void debug() {
                                                            52
                                                                       int m = (ini + fim) >> 1;
           for (auto e : seg) {
                                                            53
126
                cout << e << ' ';
127
                                                            54
                                                                       return f(query(e[pos], ini, m, p, q), query(d
                                                                   [pos], m + 1, fim, p, q));
128
            cout << '\n';</pre>
                                                            55
           for (auto e : lazy) {
130
                                                            56
131
                cout << e << ' ';
                                                            57
                                                                   void update(int pos, int ini, int fim, int id,
132
                                                                   int val) {
           cout << '\n';
                                                                       if (ini > id || fim < id) {</pre>
                                                            58
133
            cout << '\n';</pre>
134
                                                            59
                                                                           return;
135
                                                            60
136 };
                                                                       if (ini == fim) {
                                                            62
   7.7 Dynamic Implicit Sparse
                                                                           seg[pos] = val;
                                                            63
                                                            64
                                                                           return;
 1 // Description:
                                                            65
                                                                       }
 2 // Indexed at one
                                                            67
                                                                       int m = (ini + fim) >> 1;
 _{4} // When the indexes of the nodes are too big to be
                                                            68
       stored in an array
                                                            69
                                                                       if (id <= m) {</pre>
                                                            70
 _{5} // and the queries need to be answered online so we
       can't sort the nodes and compress them
                                                            71
                                                                           if (e[pos] == 0) e[pos] = create();
                                                                           update(e[pos], ini, m, id, val);
 _{6} // we create nodes only when they are needed so there ^{72}
       '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
                                                            75
                                                                           update(d[pos], m + 1, fim, id, val);
       maximum index a node can assume
                                                            77
 9 // Query - get sum of elements from range (1, r)
                                                                       seg[pos] = f(seg[e[pos]], seg[d[pos]]);
       inclusive
_{10} // Update - update element at position id to a value _{79}
                                                            80
       val
                                                                   ftype query(int p, int q) {
                                                            81
                                                                       return query(1, 1, n, p, q);
12 // Problem:
                                                            82
13 // https://cses.fi/problemset/task/1648
                                                            83
                                                            84
                                                                   void update(int id, int val) {
15 // Complexity:
16 // O(log n) for both query and update
                                                            86
                                                                       update(1, 1, n, id, val);
                                                            87
                                                            88 };
18 // How to use:
_{19} // MAX is the maximum index a node can assume
                                                                    Lazy Addition To Segment
21 // Segtree seg = Segtree(MAX);
22
23 typedef long long ftype;
                                                            1 // Description:
                                                             2 // Query - get sum of elements from range (1, r)
25 const int MAX = 1e9+17;
                                                                   inclusive
                                                             _{\rm 3} // Update - add a value val to elementos from range (
27 struct Segtree {
                                                                  l, r) inclusive
       vector<ftype> seg, d, e;
                                                             5 // Problem:
29
       const ftype NEUTRAL = 0;
       int n;
                                                             6 // https://codeforces.com/edu/course/2/lesson/5/1/
30
                                                                  practice/contest/279634/problem/A
       Segtree(int n) {
32
           this -> n = n;
                                                            8 // Complexity:
33
                                                            9 // O(log n) for both query and update
34
           create();
           create();
35
                                                            11 // How to use:
36
                                                            12 // Segtree seg = Segtree(n);
37
       ftype f(ftype a, ftype b) {
                                                            13 // seg.build(v);
           return a + b;
39
```

15 // Notes



```
_{16} // Change neutral element and f function to perform a _{84}
                                                                  void update(int pos, int ini, int fim, int p, int
       different operation
                                                                    q, int val) {
                                                                       propagate(pos, ini, fim);
18 const long long INF = 1e18+10;
                                                           86
                                                           87
                                                                       if (ini > q || fim < p) {</pre>
20 typedef long long ftype;
                                                           88
                                                                           return;
21
                                                            89
22 struct Segtree {
                                                           90
      vector<ftype> seg;
                                                                       if (ini >= p && fim <= q) {</pre>
23
                                                           91
      vector<ftype> lazy;
                                                                           lazy[pos] = apply_lazy(lazy[pos], val, 1)
                                                            92
      int n;
25
      const ftype NEUTRAL = 0;
                                                                           seg[pos] = apply_lazy(seg[pos], val, fim
27
       const ftype NEUTRAL_LAZY = -INF;
                                                                   - ini + 1);
28
29
      Segtree(int n) {
                                                           95
                                                                           return;
           int sz = 1;
                                                           96
30
           while (sz < n) sz *= 2;
                                                                       int e = 2*pos + 1;
           this->n = sz;
32
                                                           98
33
                                                                       int d = 2*pos + 2;
34
           seg.assign(2*sz, NEUTRAL);
                                                                       int m = ini + (fim - ini) / 2;
                                                           100
           lazy.assign(2*sz, NEUTRAL_LAZY);
35
                                                                       update(e, ini, m, p, q, val);
36
                                                                       update(d, m + 1, fim, p, q, val);
37
                                                           103
      ftype apply_lazy(ftype a, ftype b, int len) {
           if (b == NEUTRAL_LAZY) return a;
                                                                       seg[pos] = f(seg[e], seg[d]);
39
                                                           105
           if (a == NEUTRAL_LAZY) return b * len;
                                                           106
40
41
           else return a + b * len;
                                                           107
                                                                   void build(int pos, int ini, int fim, vector<int>
42
                                                           108
                                                                    &v) {
      void propagate(int pos, int ini, int fim) {
                                                                       if (ini == fim) {
44
                                                                           if (ini < (int)v.size()) {</pre>
          if (ini == fim) {
45
                                                                                seg[pos] = v[ini];
46
               return;
                                                           111
           }
47
                                                                           return;
           int e = 2*pos + 1;
                                                                       }
49
                                                           114
           int d = 2*pos + 2;
           int m = ini + (fim - ini) / 2;
                                                                       int e = 2*pos + 1;
51
                                                           116
                                                                       int d = 2*pos + 2;
                                                           117
           lazy[e] = apply_lazy(lazy[e], lazy[pos], 1); 118
                                                                       int m = ini + (fim - ini) / 2;
           lazy[d] = apply_lazy(lazy[d], lazy[pos], 1); 119
54
                                                                       build(e, ini, m, v);
                                                                       build(d, m + 1, fim, v);
56
           seg[e] = apply_lazy(seg[e], lazy[pos], m -
      ini + 1):
                                                           122
           seg[d] = apply_lazy(seg[d], lazy[pos], fim - 123
                                                                       seg[pos] = f(seg[e], seg[d]);
      m):
                                                           124
           lazy[pos] = NEUTRAL_LAZY;
                                                                   ftype query(int p, int q) {
59
                                                           126
60
      }
                                                                       return query(0, 0, n - 1, p, q);
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
      ftype query(int pos, int ini, int fim, int p, int133
66
                                                                   void build(vector<int> &v) {
                                                           134
                                                                       build(0, 0, n - 1, v);
           propagate(pos, ini, fim);
68
                                                           136
           if (ini >= p && fim <= q) {</pre>
                                                           137
70
               return seg[pos];
                                                           138
                                                                  void debug() {
                                                                       for (auto e : seg) {
71
                                                           139
                                                                           cout << e << ' ';
                                                           140
           if (q < ini || p > fim) {
                                                           141
               return NEUTRAL;
                                                                       cout << '\n';</pre>
                                                           142
           }
                                                                       for (auto e : lazy) {
75
                                                           143
                                                                           cout << e << ' ';
           int e = 2*pos + 1;
                                                           145
           int d = 2*pos + 2;
                                                                       cout << '\n';
                                                           146
                                                                       cout << '\n';
           int m = ini + (fim - ini) / 2;
79
                                                           147
80
                                                           148
           return f(query(e, ini, m, p, q), query(d, m +149 };
       1, fim, p, q));
                                                                     Lazy Dynamic Implicit Sparse
82
83
```

```
1 // Description:
                                                           64
2 // Indexed at one
                                                           65
                                                                      lazy[pos] = NEUTRAL_LAZY;
                                                           66
_4 // When the indexes of the nodes are too big to be
                                                           67
                                                                  ftype f(ftype a, ftype b) {
      stored in an array
_{5} // and the queries need to be answered online so we
                                                                      return a + b;
                                                           69
      can't sort the nodes and compress them
                                                           70
_{6} // we create nodes only when they are needed so there _{71}
      'll be (Q*log(MAX)) nodes
                                                                  ftype create() {
                                                           72
_{7} // where Q is the number of queries and MAX is the
                                                                      seg.push_back(0);
      maximum index a node can assume
                                                                      e.push_back(0);
                                                           74
                                                                      d.push_back(0);
9 // Query - get sum of elements from range (1, r)
                                                           76
                                                                      lazy.push_back(-1);
      inclusive
                                                                      return seg.size() - 1;
                                                           77
_{10} // Update - update element at position id to a value _{78}
      val
                                                           79
                                                                  ftype query(int pos, int ini, int fim, int p, int
12 // Problem:
                                                                   q) {
13 // https://oj.uz/problem/view/IZh012_apple
                                                           81
                                                                      propagate(pos, ini, fim);
                                                           82
                                                                      if (q < ini || p > fim) return NEUTRAL;
15 // Complexity:
                                                                      if (pos == 0) return 0;
                                                           83
_{16} // O(log n) for both query and update
                                                                      if (p <= ini && fim <= q) return seg[pos];</pre>
                                                                      int m = (ini + fim) >> 1;
                                                           85
18 // How to use:
                                                                      return f(query(e[pos], ini, m, p, q), query(d
19 // MAX is the maximum index a node can assume
                                                                  [pos], m + 1, fim, p, q));
20 // Create a default null node
                                                           87
21 // Create a node to be the root of the segtree
                                                           88
                                                                  void update(int pos, int ini, int fim, int p, int
                                                           89
23 // Segtree seg = Segtree(MAX);
                                                                   q, int val) {
24 // seg.create();
                                                           90
                                                                      propagate(pos, ini, fim);
25 // seg.create();
                                                           91
                                                                      if (ini > q || fim < p) {</pre>
                                                           92
                                                                          return;
27 const int MAX = 1e9+10;
                                                                      }
                                                           93
28 const long long INF = 1e18+10;
                                                           94
                                                                      if (ini >= p && fim <= q) {</pre>
29
                                                           95
30 typedef long long ftype;
                                                                          lazy[pos] = apply_lazy(lazy[pos], val, 1)
31
32 struct Segtree {
                                                                          seg[pos] = apply_lazy(seg[pos], val, fim
                                                           97
      vector<ftype> seg, d, e, lazy;
                                                                  - ini + 1);
      const ftype NEUTRAL = 0;
34
                                                           98
      const ftype NEUTRAL_LAZY = -INF;
                                                                          return;
                                                                      }
36
      int n;
                                                          100
37
                                                          101
      Segtree(int n) {
                                                                      int m = (ini + fim) >> 1;
38
          this -> n = n;
39
                                                                      if (e[pos] == 0) e[pos] = create();
40
                                                          104
                                                                      update(e[pos], ini, m, p, q, val);
41
      ftype apply_lazy(ftype a, ftype b, int len) {
                                                                      if (d[pos] == 0) d[pos] = create();
43
          if (b == NEUTRAL_LAZY) return a;
           else return b * len;
                                                                      update(d[pos], m + 1, fim, p, q, val);
                                                          108
44
45
                                                                      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) {
49
                                                          113
           if (ini == fim) {
                                                          114
                                                                      return query(1, 1, n, p, q);
50
               return:
                                                          115
          1
                                                          116
53
                                                          117
                                                                  void update(int p, int q, int val) {
           int m = (ini + fim) >> 1;
                                                                      update(1, 1, n, p, q, val);
54
                                                          118
                                                          119
           if (e[pos] == 0) e[pos] = create();
                                                          120 };
56
           if (d[pos] == 0) d[pos] = create();
                                                             7.10 Big K
58
          lazy[e[pos]] = apply_lazy(lazy[e[pos]], lazy[
      pos], 1);
                                                            1 struct SetSum {
          lazy[d[pos]] = apply_lazy(lazy[d[pos]], lazy[
60
                                                                  11 s = 0;
      pos], 1);
                                                                  multiset <11> mt;
61
                                                                  void add(ll x){
           seg[e[pos]] = apply_lazy(seg[e[pos]], lazy[
                                                                      mt.insert(x);
      pos], m - ini + 1);
          seg[d[pos]] = apply_lazy(seg[d[pos]], lazy[
63
                                                                  }
      pos], fim - m);
                                                                  int pop(ll x){
```



```
auto f = mt.find(x);
                                                           18 template <typename T>
9
10
           if(f == mt.end()) return 0;
                                                           19 using ordered_set = tree<T,null_type,less<T>,
11
          mt.erase(f);
                                                                  rb_tree_tag,tree_order_statistics_node_update>;
           s -= x;
                                                              7.12 Dsu
           return 1;
      }
14
15 };
                                                            1 /*
16
                                                            2 DSU - Disjoint Set Union (or Union Find)
17 struct BigK {
      int k;
                                                            4 find(x) -> find component that x is on
      SetSum gt, mt;
19
                                                            _{5} join(a, b) -> union of a set containing 'a' and set
20
      BigK(int _k){
                                                                  containing b
21
          k = _k;
22
                                                            7 find / join with path compreension -> O(inv_Ackermann
23
      void balancear(){
                                                                 (n)) [0(1)]
          while((int)gt.mt.size() < k && (int)mt.mt.</pre>
24
                                                            8 find / join without path compreension -> O(logN)
      size()){
               auto p = (prev(mt.mt.end()));
25
                                                           10 https://judge.yosupo.jp/submission/126864
               gt.add(*p);
27
               mt.pop(*p);
                                                           12
          }
28
                                                           13 struct DSU {
           while((int)mt.mt.size() && (int)gt.mt.size()
29
                                                           14
      & &
                                                                  int n = 0, components = 0;
           *(gt.mt.begin()) < *(prev(mt.mt.end())) ){
                                                                  vector<int> parent;
                                                           16
               11 u = *(gt.mt.begin());
31
                                                           17
                                                                  vector < int > size;
               11 v = *(prev(mt.mt.end()));
32
                                                           18
33
               gt.pop(u); mt.pop(v);
                                                                  DSU(int nn){
                                                           19
               gt.add(v); mt.add(u);
34
                                                                      n = nn;
                                                           20
          }
                                                                      components = n;
                                                           21
      }
36
                                                           22
                                                                      size.assign(n + 5, 1);
      void add(ll x){
37
                                                                      parent.assign(n + 5, 0);
                                                           23
          mt.add(x):
38
                                                                      iota(parent.begin(), parent.end(), 0);
                                                           24
           balancear();
39
                                                                  }
                                                           25
                                                           26
      void rem(ll x){
41
                                                                  int find(int x){
                                                           27
           //x = -x;
42
                                                           28
                                                                      if(x == parent[x]) {
           if(mt.pop(x) == 0)
43
                                                           29
                                                                          return x;
               gt.pop(x);
44
                                                           30
           balancear();
45
                                                                      //path compression
                                                           31
      }
46
                                                           32
                                                                      return parent[x] = find(parent[x]);
47 };
                                                           33
                                                           34
         Ordered Set
  7.11
                                                                  void join(int a, int b){
                                                           35
                                                                      a = find(a);
                                                           36
1 // Ordered Set
                                                                      b = find(b);
                                                           37
2 //
                                                                      if(a == b) {
                                                           38
3 // set roubado com mais operacoes
                                                           39
                                                                           return;
4 //
                                                                      }
                                                           40
                                                                      if(size[a] < size[b]) {</pre>
5 // para alterar para multiset
                                                           41
6 // trocar less para less_equal
                                                           42
                                                                           swap(a, b);
7 //
                                                           43
8 // ordered_set <int> s
                                                                      parent[b] = a;
9 //
                                                                      size[a] += size[b];
                                                           45
10 // order_of_key(k) // number of items strictly
                                                           46
                                                                      components -= 1;
      smaller than k -> int
                                                           47
                                                                  }
11 // find_by_order(k) // k-th element in a set (
                                                           48
      counting from zero) -> iterator
                                                           49
                                                                  int sameSet(int a, int b) {
                                                                      a = find(a);
                                                           50
13 // https://cses.fi/problemset/task/2169
                                                           51
                                                                      b = find(b);
14 //
                                                                      return a == b;
                                                           52
15 // O(log N) para insert, erase (com iterator),
                                                           53
      order_of_key, find_by_order
                                                           54
                                                           55 };
17 using namespace __gnu_pbds;
```