

# Competitive Programming Notebook

As Meninas Superpoderosas

## Contents

<b>1 Graph</b>	<b>2</b>	7.6 Lis Segtree . . . . .	11
1.1 Bfs . . . . .	2	7.7 Range Dp . . . . .	12
1.2 Floyd Warshall . . . . .	2	<b>8 DS</b>	<b>12</b>
1.3 2sat . . . . .	2	8.1 Dsu . . . . .	12
1.4 Lca . . . . .	3	8.2 Ordered Set . . . . .	12
1.5 Dinic . . . . .	4	8.3 Kruskal . . . . .	12
1.6 Dijkstra . . . . .	5		
1.7 Ford Fulkerson . . . . .	5		
1.8 Has Negative Cycle . . . . .	5		
<b>2 Primitives</b>	<b>6</b>		
<b>3 Geometry</b>	<b>6</b>		
3.1 Convex Hull . . . . .	6		
<b>4 Math</b>	<b>6</b>		
4.1 Log Any Base . . . . .	6		
4.2 Generate Primes . . . . .	6		
4.3 Factorization . . . . .	7		
4.4 Sieve . . . . .	7		
4.5 Ceil . . . . .	7		
4.6 Fexp . . . . .	7		
4.7 Is Prime . . . . .	7		
4.8 Divisors . . . . .	7		
<b>5 General</b>	<b>7</b>		
5.1 Min Priority Queue . . . . .	7		
5.2 Random . . . . .	7		
5.3 Next Permutation . . . . .	7		
5.4 Base Converter . . . . .	8		
5.5 Get Subsets Sum Iterative . . . . .	8		
5.6 Last True . . . . .	8		
5.7 Xor 1 To N . . . . .	8		
5.8 Input By File . . . . .	8		
5.9 Template . . . . .	9		
5.10 First True . . . . .	9		
<b>6 String</b>	<b>9</b>		
6.1 Split . . . . .	9		
6.2 Is Substring . . . . .	9		
6.3 Trie Xor . . . . .	9		
<b>7 DP</b>	<b>10</b>		
7.1 Digit Dp . . . . .	10		
7.2 Lcs . . . . .	10		
7.3 Lis Binary Search . . . . .	11		
7.4 Edit Distance . . . . .	11		
7.5 Digit Dp 2 . . . . .	11		

# 1 Graph

## 1.1 Bfs

```

1 vector<vector<int>> adj; // adjacency list
  representation
2 int n; // number of nodes
3 int s; // source vertex
4
5 queue<int> q;
6 vector<bool> used(n + 1);
7 vector<int> d(n + 1), p(n + 1);
8
9 q.push(s);
10 used[s] = true;
11 p[s] = -1;
12 while (!q.empty()) {
13     int v = q.front();
14     q.pop();
15     for (int u : adj[v]) {
16         if (!used[u]) {
17             used[u] = true;
18             q.push(u);
19             d[u] = d[v] + 1;
20             p[u] = v;
21         }
22     }
23 }
24
25 // restore path
26 if (!used[u]) {
27     cout << "No path!";
28 } else {
29     vector<int> path;
30
31     for (int v = u; v != -1; v = p[v])
32         path.push_back(v);
33
34     reverse(path.begin(), path.end());
35
36     cout << "Path: ";
37     for (int v : path)
38         cout << v << " ";
39 }

```

## 1.2 Floyd Warshall

```

1 const long long LLINF = 0x3f3f3f3f3f3f3fLL;
2
3 for (int i = 0; i < n; i++) {
4     for (int j = 0; j < n; j++) {
5         adj[i][j] = 0;
6     }
7 }
8
9 long long dist[MAX][MAX];
10 for (int i = 0; i < n; i++) {
11     for (int j = 0; j < n; j++) {
12         if (i == j)
13             dist[i][j] = 0;
14         else if (adj[i][j])
15             dist[i][j] = adj[i][j];
16         else
17             dist[i][j] = LLINF;
18     }
19 }
20
21 for (int k = 0; k < n; k++) {
22     for (int i = 0; i < n; i++) {
23         for (int j = 0; j < n; j++) {
24             dist[i][j] = min(dist[i][j], dist[i][k] +

```

```

25         }
26     }
27 }

```

## 1.3 2sat

```

1 // 2SAT
2 //
3 // verifica se existe e encontra solução
4 // para fórmulas booleanas da forma
5 // (a or b) and (!a or c) and (...)
6 //
7 // indexado em 0
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)
10 // ^1 = 4
11 // https://cses.fi/problemset/task/1684/
12 // https://codeforces.com/gym/104120/problem/E
13 // (add_eq, add_true, add_false e at_most_one não
14 // foram testadas)
15 // O(n + m)
16
17 struct sat {
18     int n, tot;
19     vector<vector<int>> adj, adjt; // grafo original,
20     // grafo transposto
21     vector<int> vis, comp, ans;
22     stack<int> topo; // ordem topológica
23
24     sat() {}
25     sat(int n_) : n(n_), tot(n), adj(2*n), adjt(2*n) {}
26
27     void dfs(int x) {
28         vis[x] = true;
29
30         for (auto e : adj[x]) {
31             if (!vis[e]) dfs(e);
32         }
33
34         topo.push(x);
35     }
36
37     void dfst(int x, int& id) {
38         vis[x] = true;
39         comp[x] = id;
40
41         for (auto e : adjt[x]) {
42             if (!vis[e]) dfst(e, id);
43         }
44     }
45
46     void add_impl(int a, int b) { // a -> b = (!a or b)
47         a = (a >= 0 ? 2*a : -2*a-1);
48         b = (b >= 0 ? 2*b : -2*b-1);
49
50         adj[a].push_back(b);
51         adj[b^1].push_back(a^1);
52
53         adjt[b].push_back(a);
54         adjt[a^1].push_back(b^1);
55     }
56
57     void add_or(int a, int b) { // a or b
58         add_impl(~a, b);
59     }
60
61     void add_nor(int a, int b) { // a nor b = !(a or b)
62         add_or(~a, b), add_or(a, ~b), add_or(~a, ~b);

```

```

62     }
63
64     void add_and(int a, int b) { // a and b
65         add_or(a, b), add_or(~a, b), add_or(a, ~b);
66     }
67
68     void add_nand(int a, int b) { // a nand b = !(a
69         and b)
70         add_or(~a, ~b);
71     }
72
73     void add_xor(int a, int b) { // a xor b = (a != b
74         )
75         add_or(a, b), add_or(~a, ~b);
76     }
77
78     void add_xnor(int a, int b) { // a xnor b = !(a
79         xor b) = (a == b)
80         add_xor(~a, b);
81     }
82
83     void add_true(int a) { // a = T
84         add_or(a, ~a);
85     }
86
87     void add_false(int a) { // a = F
88         add_and(a, ~a);
89     }
90
91     // magia - brunomaletta
92     void add_true_old(int a) { // a = T (n sei se
93         funciona)
94         add_impl(~a, a);
95     }
96
97     void at_most_one(vector<int> v) { // no max um
98         verdadeiro
99         adj.resize(2*(tot+v.size()));
100         for (int i = 0; i < v.size(); i++) {
101             add_impl(tot+i, ~v[i]);
102             if (i) {
103                 add_impl(tot+i, tot+i-1);
104                 add_impl(v[i], tot+i-1);
105             }
106         }
107         tot += v.size();
108     }
109
110     pair<bool, vector<int>> solve() {
111         ans.assign(n, -1);
112         comp.assign(2*tot, -1);
113         vis.assign(2*tot, 0);
114         int id = 1;
115
116         for (int i = 0; i < 2*tot; i++) if (!vis[i])
117             dfs(i);
118
119         vis.assign(2*tot, 0);
120         while (topo.size()) {
121             auto x = topo.top();
122             topo.pop();
123
124             if (!vis[x]) {
125                 dfst(x, id);
126                 id++;
127             }
128         }
129
130         for (int i = 0; i < tot; i++) {
131             if (comp[2*i] == comp[2*i+1]) return {
132                 false, {} };
133             ans[i] = (comp[2*i] > comp[2*i+1]);
134         }
135     }

```

```

128
129         return {true, ans};
130     }
131 };

```

## 1.4 Lca

```

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 {
12     const int MAXE = 31;
13     vector<vector<int>>> up;
14     vector<int> dep;
15
16     LCA(int n, vector<vector<int>>>& adj, int root =
17         1) {
18         up.assign(n+1, vector<int>(MAXE, -1));
19         dep.assign(n+1, 0);
20
21         dep[root] = 1;
22         dfs(root, -1, adj);
23
24         for (int j = 1; j < MAXE; j++) {
25             for (int i = 1; i <= n; i++) {
26                 if (up[i][j-1] != -1)
27                     up[i][j] = up[ up[i][j-1] ][j-1];
28             }
29         }
30     }
31
32     void dfs(int x, int p, vector<vector<int>>>& adj)
33     {
34         up[x][0] = p;
35         for (auto e : adj[x]) {
36             if (e != p) {
37                 dep[e] = dep[x] + 1;
38                 dfs(e, x, adj);
39             }
40         }
41     }
42
43     int jump(int x, int k) { // jump from node x k
44         times
45         for (int i = 0; i < MAXE; i++) {
46             if (k && (1 << i) && x != -1) x = up[x][i];
47         }
48         return x;
49     }
50
51     int lca(int a, int b) {
52         if (dep[a] > dep[b]) swap(a, b);
53         b = jump(b, dep[b] - dep[a]);
54
55         if (a == b) return a;
56
57         for (int i = MAXE-1; i >= 0; i--) {
58             if (up[a][i] != up[b][i]) {
59                 a = up[a][i];
60                 b = up[b][i];
61             }
62         }
63         return up[a][0];
64     }
65 }

```

```

64     int dist(int a, int b) {
65         return dep[a] + dep[b] - 2 * dep[lca(a, b)];
66     }
67 };

```

## 1.5 Dinic

```

1  // Dinic / Dinitz
2  //
3  // max-flow / min-cut
4  //
5  // https://cses.fi/problemset/task/1694/
6  //
7  // O(E * V^2)
8
9  using ll = long long;
10 const ll FLOW_INF = 1e18 + 7;
11
12 struct Edge {
13     int from, to;
14     ll cap, flow;
15     Edge* residual; // a inversa da minha aresta
16
17     Edge() {};
18
19     Edge(int from, int to, ll cap) : from(from), to(to), cap(cap), flow(0) {};
20
21     ll remaining_cap() {
22         return cap - flow;
23     }
24
25     void augment(ll bottle_neck) {
26         flow += bottle_neck;
27         residual->flow -= bottle_neck;
28     }
29
30     bool is_residual() {
31         return cap == 0;
32     }
33 };
34
35 struct Dinic {
36     int n;
37     vector<vector<Edge*>> adj;
38     vector<int> level, next;
39
40     Dinic(int n): n(n) {
41         adj.assign(n+1, vector<Edge*>());
42         level.assign(n+1, -1);
43         next.assign(n+1, 0);
44     }
45
46     void add_edge(int from, int to, ll cap) {
47         auto e1 = new Edge(from, to, cap);
48         auto e2 = new Edge(to, from, 0);
49
50         e1->residual = e2;
51         e2->residual = e1;
52
53         adj[from].push_back(e1);
54         adj[to].push_back(e2);
55     }
56
57     bool bfs(int s, int t) {
58         fill(level.begin(), level.end(), -1);
59         queue<int> q;
60
61         q.push(s);
62         level[s] = 1;
63
64         while (q.size()) {
65             int curr = q.front();

```

```

66             q.pop();
67
68             for (auto edge : adj[curr]) {
69                 if (edge->remaining_cap() > 0 &&
70                     level[edge->to] == -1) {
71                     level[edge->to] = level[curr] +
72                     1;
73                     q.push(edge->to);
74                 }
75             }
76
77             return level[t] != -1;
78         }
79
80         ll dfs(int x, int t, ll flow) {
81             if (x == t) return flow;
82
83             for (int& cid = next[x]; cid < (int)adj[x].
84                 size(); cid++) {
85                 auto& edge = adj[x][cid];
86                 ll cap = edge->remaining_cap();
87
88                 if (cap > 0 && level[edge->to] == level[x]
89                     + 1) {
90                     ll sent = dfs(edge->to, t, min(flow,
91                         cap)); // bottle neck
92                     if (sent > 0) {
93                         edge->augment(sent);
94                         return sent;
95                     }
96                 }
97             }
98
99             return 0;
100         }
101
102         ll solve(int s, int t) {
103             ll max_flow = 0;
104
105             while (bfs(s, t)) {
106                 fill(next.begin(), next.end(), 0);
107
108                 while (ll sent = dfs(s, t, FLOW_INF)) {
109                     max_flow += sent;
110                 }
111             }
112
113             return max_flow;
114         }
115
116         // path recover
117         vector<bool> vis;
118         vector<int> curr;
119
120         bool dfs2(int x, int& t) {
121             vis[x] = true;
122             bool arrived = false;
123
124             if (x == t) {
125                 curr.push_back(x);
126                 return true;
127             }
128
129             for (auto e : adj[x]) {
130                 if (e->flow > 0 && !vis[e->to]) { // !e->
131                     is_residual() &&
132                     bool aux = dfs2(e->to, t);
133
134                     if (aux) {
135                         arrived = true;
136                         e->flow--;
137                     }
138                 }
139             }
140         }

```

```

133     }
134 }
135
136     if (arrived) curr.push_back(x);
137
138     return arrived;
139 }
140
141 vector<vector<int>> get_paths(int s, int t) {
142     vector<vector<int>> ans;
143
144     while (true) {
145         curr.clear();
146         vis.assign(n+1, false);
147
148         if (!dfs2(s, t)) break;
149
150         reverse(curr.begin(), curr.end());
151         ans.push_back(curr);
152     }
153
154     return ans;
155 }
156 };

```

## 1.6 Dijkstra

```

1 const int INF = 1e9+17;
2 vector<vector<pair<int, int>>> adj; // {neighbor,
   weight}
3
4 void dijkstra(int s, vector<int> & d, vector<int> & p
   ) {
5     int n = adj.size();
6     d.assign(n, INF);
7     p.assign(n, -1);
8
9     d[s] = 0;
10    set<pair<int, int>> q;
11    q.insert({0, s});
12    while (!q.empty()) {
13        int v = q.begin()->second;
14        q.erase(q.begin());
15
16        for (auto edge : adj[v]) {
17            int to = edge.first;
18            int len = edge.second;
19
20            if (d[v] + len < d[to]) {
21                q.erase({d[to], to});
22                d[to] = d[v] + len;
23                p[to] = v;
24                q.insert({d[to], to});
25            }
26        }
27    }
28 }

```

## 1.7 Ford Fulkerson

```

1 // Ford-Fulkerson
2 //
3 // max-flow / min-cut
4 //
5 // MAX nÃşs
6 //
7 // https://cses.fi/problemset/task/1694/
8 //
9 // O(m * max_flow)
10
11 using ll = long long;
12 const int MAX = 510;

```

```

13 struct Flow {
14     int n;
15     ll adj[MAX][MAX];
16     bool used[MAX];
17
18     Flow(int n) : n(n) {};
19
20     void add_edge(int u, int v, ll c) {
21         adj[u][v] += c;
22         adj[v][u] = 0; // cuidado com isso
23     }
24
25     ll dfs(int x, int t, ll amount) {
26         used[x] = true;
27
28         if (x == t) return amount;
29
30         for (int i = 1; i <= n; i++) {
31             if (adj[x][i] > 0 && !used[i]) {
32                 ll sent = dfs(i, t, min(amount, adj[x
33 ] [i]));
34
35                 if (sent > 0) {
36                     adj[x][i] -= sent;
37                     adj[i][x] += sent;
38
39                     return sent;
40                 }
41             }
42         }
43
44         return 0;
45     }
46
47     ll max_flow(int s, int t) { // source and sink
48         ll total = 0;
49         ll sent = -1;
50
51         while (sent != 0) {
52             memset(used, 0, sizeof(used));
53             sent = dfs(s, t, INT_MAX);
54             total += sent;
55         }
56
57         return total;
58     }
59 };

```

## 1.8 Has Negative Cycle

```

1 // Edson
2
3 using edge = tuple<int, int, int>;
4
5 bool has_negative_cycle(int s, int N, const vector<
   edge>& edges)
6 {
7     const int INF { 1e9+17 };
8
9     vector<int> dist(N + 1, INF);
10    dist[s] = 0;
11
12    for (int i = 1; i <= N - 1; i++) {
13        for (auto [u, v, w] : edges) {
14            if (dist[u] < INF && dist[v] > dist[u] +
15 w) {
16                dist[v] = dist[u] + w;
17            }
18        }
19    }
20
21    for (auto [u, v, w] : edges) {

```

```

21         if (dist[u] < INF && dist[v] > dist[u] + w) {
22             return true;
23         }
24     }
25     return false;
26 }

```

## 2 Primitives

## 3 Geometry

### 3.1 Convex Hull

```

1 // Convex Hull - Monotone Chain
2 //
3 // Convex Hull is the subset of points that forms the
4 // smallest convex polygon
5 // which encloses all points in the set.
6 //
7 // https://cses.fi/problemset/task/2195/
8 // https://open.kattis.com/problems/convexhull (
9 // counterclockwise)
10 //
11 // 0(n log(n))
12
13 typedef long long ftype;
14
15 struct Point {
16     ftype x, y;
17
18     Point() {}
19     Point(ftype x, ftype y) : x(x), y(y) {}
20
21     bool operator<(Point o) {
22         if (x == o.x) return y < o.y;
23         return x < o.x;
24     }
25
26     bool operator==(Point o) {
27         return x == o.x && y == o.y;
28     }
29 };
30
31 ftype cross(Point a, Point b, Point c) {
32     // v: a -> c
33     // w: a -> b
34
35     // v: c.x - a.x, c.y - a.y
36     // w: b.x - a.x, b.y - a.y
37
38     return (c.x - a.x) * (b.y - a.y) - (c.y - a.y) *
39           (b.x - a.x);
40 }
41
42 ftype dir(Point a, Point b, Point c) {
43     // 0 -> colineares
44     // -1 -> esquerda
45     // 1 -> direita
46
47     ftype cp = cross(a, b, c);
48
49     if (cp == 0) return 0;
50     else if (cp < 0) return -1;
51     else return 1;
52 }
53
54 vector<Point> convex_hull(vector<Point> points) {
55     sort(points.begin(), points.end());
56     points.erase( unique(points.begin(), points.end())
57                  , points.end()); // somente pontos distintos

```

```

54 int n = points.size();
55
56 if (n == 1) return { points[0] };
57
58 vector<Point> upper_hull = {points[0], points
59 [1]};
60 for (int i = 2; i < n; i++) {
61     upper_hull.push_back(points[i]);
62
63     int sz = upper_hull.size();
64
65     while (sz >= 3 && dir(upper_hull[sz-3],
66 upper_hull[sz-2], upper_hull[sz-1]) == -1) {
67         upper_hull.pop_back();
68         upper_hull.pop_back();
69         upper_hull.push_back(points[i]);
70         sz--;
71     }
72
73 vector<Point> lower_hull = {points[n-1], points[n
74 -2]};
75 for (int i = n-3; i >= 0; i--) {
76     lower_hull.push_back(points[i]);
77
78     int sz = lower_hull.size();
79
80     while (sz >= 3 && dir(lower_hull[sz-3],
81 lower_hull[sz-2], lower_hull[sz-1]) == -1) {
82         lower_hull.pop_back();
83         lower_hull.pop_back();
84         lower_hull.push_back(points[i]);
85         sz--;
86     }
87 }
88
89 // reverse(lower_hull.begin(), lower_hull.end());
90 // counterclockwise
91
92 for (int i = (int)lower_hull.size() - 2; i > 0; i
93 --) {
94     upper_hull.push_back(lower_hull[i]);
95 }
96
97 return upper_hull;
98 }

```

## 4 Math

### 4.1 Log Any Base

```

1 int intlog(double base, double x) {
2     return (int)(log(x) / log(base));
3 }

```

### 4.2 Generate Primes

```

1 // crivo nao otimizado
2
3 vector<int> generate_primes(int lim=1e5+17) {
4     vector<int> primes;
5     vector<bool> isprime(lim+1, true);
6
7     isprime[0] = isprime[1] = false;
8
9     for (int i = 2; i*i < lim; i++) {
10         if (isprime[i]) {
11             primes.push_back(i);
12
13             for (int j = i+i; j < lim; j += i) {
14                 isprime[j] = false;

```

```

15     }
16 }
17 }
18
19 return primes;
20 }

```

### 4.3 Factorization

```

1 // nson
2
3 using ll = long long;
4
5 vector<pair<ll, int>> factorization(ll n) {
6     vector<pair<ll, int>> ans;
7
8     for (ll p = 2; p*p <= n; p++) {
9         if (n%p == 0) {
10             int expoente = 0;
11
12             while (n%p == 0) {
13                 n /= p;
14                 expoente++;
15             }
16
17             ans.push_back({p, expoente});
18         }
19     }
20
21     if (n > 1) {
22         ans.push_back({n, 1});
23     }
24
25     return ans;
26 }

```

### 4.4 Sieve

```

1 // nao "otimizado"
2
3 vector<bool> sieve(int lim=1e5+17) {
4     vector<bool> isprime(lim+1, true);
5
6     isprime[0] = isprime[1] = false;
7
8     for (int i = 2; i*i < lim; i++) {
9         if (isprime[i]) {
10             for (int j = i+i; j < lim; j += i) {
11                 isprime[j] = false;
12             }
13         }
14     }
15
16     return isprime;
17 }

```

### 4.5 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 }

```

### 4.6 Fexp

```

1 using ll = long long;
2
3 ll fexp(ll base, ll exp, ll m) {
4     ll ans = 1;
5     base %= m;
6
7     while (exp > 0) {
8         if (exp % 2 == 1) {
9             ans = (ans * base) % m;
10        }
11
12        base = (base * base) % m;
13        exp /= 2;
14    }
15
16    return ans;
17 }

```

### 4.7 Is Prime

```

1 bool is_prime(ll n) {
2     if (n <= 1) return false;
3     if (n == 2) return true;
4
5     for (ll i = 2; i*i <= n; i++) {
6         if (n % i == 0)
7             return false;
8     }
9
10    return true;
11 }

```

### 4.8 Divisors

```

1 vector<ll> divisors(ll n) {
2     vector<ll> ans;
3
4     for (ll i = 1; i*i <= n; i++) {
5         if (n%i == 0) {
6             ll value = n/i;
7
8             ans.push_back(i);
9             if (value != i) {
10                 ans.push_back(value);
11             }
12         }
13     }
14
15     return ans;
16 }

```

## 5 General

### 5.1 Min Priority Queue

```

1 template<class T> using min_priority_queue =
    priority_queue<T, vector<T>, greater<T>>;

```

### 5.2 Random

```

1 random_device dev;
2 mt19937 rng(dev());
3
4 uniform_int_distribution<mt19937::result_type> dist
    (1, 6); // distribution in range [1, 6]
5
6 int val = dist(rng);

```

### 5.3 Next Permutation

```

1 // output: 1,2,3; 1,3,2; 2,1,3; 2,3,1; 3,1,2; 3,2,1;
2
3 vector<int> arr = {1, 2, 3};
4 int n = arr.size();
5
6 do {
7     for (auto e : arr) {
8         cout << e << ' ';
9     }
10    cout << '\n';
11 } while (next_permutation(arr.begin(), arr.end()));

```

## 5.4 Base Converter

```

1 const string digits = "0123456789
  ABCDEFGHIJKLMNOPQRSTUVWXYZ";
2
3 ll tobase10(string number, int base) {
4     map<char, int> val;
5     for (int i = 0; i < digits.size(); i++) {
6         val[digits[i]] = i;
7     }
8
9     ll ans = 0, pot = 1;
10
11    for (int i = number.size() - 1; i >= 0; i--) {
12        ans += val[number[i]] * pot;
13        pot *= base;
14    }
15
16    return ans;
17 }
18
19 string frombase10(ll number, int base) {
20     if (number == 0) return "0";
21
22     string ans = "";
23
24     while (number > 0) {
25         ans += digits[number % base];
26         number /= base;
27     }
28
29     reverse(ans.begin(), ans.end());
30
31     return ans;
32 }
33
34 // verifica se um número está na base especificada
35 bool verify_base(string num, int base) {
36     map<char, int> val;
37     for (int i = 0; i < digits.size(); i++) {
38         val[digits[i]] = i;
39     }
40
41     for (auto digit : num) {
42         if (val[digit] >= base) {
43             return false;
44         }
45     }
46
47     return true;
48 }

```

## 5.5 Get Subsets Sum Iterative

```

1 vector<ll> get_subset_sums(int l, int r, vector<ll>&
  arr) {
2     vector<ll> ans;
3
4     int len = r-l+1;
5     for (int i = 0; i < (1 << len); i++) {

```

```

6         ll sum = 0;
7
8         for (int j = 0; j < len; j++) {
9             if (i & (1 << j)) {
10                 sum += arr[l + j];
11             }
12         }
13
14         ans.push_back(sum);
15     }
16
17     return ans;
18 }

```

## 5.6 Last True

```

1 // Binary Search (last_true)
2
3 // last_true(2, 10, [](int x) { return x * x <= 30;
4 // }); // outputs 5
5 // [1, r]
6 //
7 // if none of the values in the range work, return lo
8 // - 1
9 //
10 // f(1) = true
11 // f(2) = true
12 // f(3) = true
13 // f(4) = true
14 // f(5) = true
15 // f(6) = false
16 // f(7) = false
17 // f(8) = false
18 //
19 // last_true(1, 8, f) = 5
20 // last_true(7, 8, f) = 6
21
22 int last_true(int lo, int hi, function<bool(int)> f)
23 {
24     lo--;
25     while (lo < hi) {
26         int mid = lo + (hi - lo + 1) / 2;
27
28         if (f(mid)) {
29             lo = mid;
30         } else {
31             hi = mid - 1;
32         }
33     }
34
35     return lo;
36 }

```

## 5.7 Xor 1 To N

```

1 // XOR sum from 1 to N
2 ll xor_1_to_n(ll n) {
3     if (n % 4 == 0) {
4         return n;
5     } else if (n % 4 == 1) {
6         return 1;
7     } else if (n % 4 == 2) {
8         return n + 1;
9     }
10
11     return 0;
12 }

```

## 5.8 Input By File

```

1 freopen("file.in", "r", stdin);
2 freopen("file.out", "w", stdout);

```



## 5.9 Template

```

1 #include <bits/stdc++.h>
2
3 using namespace std;
4
5 int main() {
6     ios::sync_with_stdio(false);
7     cin.tie(NULL);
8
9
10
11     return 0;
12 }
```

## 5.10 First True

```

1 // Binary Search (first_true)
2 //
3 // first_true(2, 10, [](int x) { return x * x >= 30;
4 //     }); // outputs 6
5 //
6 // [l, r]
7 // if none of the values in the range work, return hi
8 //     + 1
9 //
10 // f(4) = false
11 // f(5) = false
12 // f(6) = true
13 // f(7) = true
14 int first_true(int lo, int hi, function<bool(int)> f)
15 {
16     hi++;
17     while (lo < hi) {
18         int mid = lo + (hi - lo) / 2;
19
20         if (f(mid)) {
21             hi = mid;
22         } else {
23             lo = mid + 1;
24         }
25     }
26     return lo;
27 }
```

## 6 String

### 6.1 Split

```

1 vector<string> split(string s, char key=' ') {
2     vector<string> ans;
3     string aux = "";
4
5     for (int i = 0; i < (int)s.size(); i++) {
6         if (s[i] == key) {
7             if (aux.size() > 0) {
8                 ans.push_back(aux);
9                 aux = "";
10            }
11        } else {
12            aux += s[i];
13        }
14    }
15
16    if ((int)aux.size() > 0) {
17        ans.push_back(aux);
18    }
19
20    return ans;
21 }
```

21 }

## 6.2 Is Substring

```

1 // equivalente ao in do python
2
3 bool is_substring(string a, string b){ // verifica se
4     a Ã substring de b
5     for(int i = 0; i < b.size(); i++){
6         int it = i, jt = 0; // b[it], a[jt]
7
8         while(it < b.size() && jt < a.size()){
9             if(b[it] != a[jt])
10                 break;
11
12             it++;
13             jt++;
14
15             if(jt == a.size())
16                 return true;
17         }
18     }
19
20     return false;
21 }
```

## 6.3 Trie Xor

```

1 // TrieXOR
2 //
3 // adiciona, remove e verifica se existe strings
4 // binarias
5 // max_xor(x) = maximiza o xor de x com algum valor
6 // da trie
7 //
8 // raiz = 0
9 //
10 // https://codeforces.com/problemset/problem/706/D
11 //
12 // 0(|s|) adicionar, remover e buscar
13
14 struct TrieXOR {
15     int n, alph_sz, nxt;
16     vector<vector<int>> trie;
17     vector<int> finish, paths;
18
19     TrieXOR() {}
20
21     TrieXOR(int n, int alph_sz = 2) : n(n), alph_sz(
22         alph_sz) {
23         nxt = 1;
24         trie.assign(n, vector<int>(alph_sz));
25         finish.assign(n * alph_sz, 0);
26         paths.assign(n * alph_sz, 0);
27     }
28
29     void add(int x) {
30         int curr = 0;
31
32         for (int i = 31; i >= 0; i--) {
33             int b = ((x >> i) & 1);
34
35             if (trie[curr][b] == 0)
36                 trie[curr][b] = nxt++;
37
38             paths[curr]++;
39             curr = trie[curr][b];
40         }
41
42         finish[curr]++;
43     }
44 }
```

```

42
43 void rem(int x) {
44     int curr = 0;
45
46     for (int i = 31; i >= 0; i--) {
47         int b = ((x & (1 << i)) > 0);
48
49         paths[curr]--;
50         curr = trie[curr][b];
51     }
52
53     paths[curr]--;
54     finish[curr]--;
55 }
56
57 int search(int x) {
58     int curr = 0;
59
60     for (int i = 31; i >= 0; i--) {
61         int b = ((x & (1 << i)) > 0);
62
63         if (trie[curr][b] == 0) return false;
64
65         curr = trie[curr][b];
66     }
67
68     return (finish[curr] > 0);
69 }
70
71 int max_xor(int x) { // maximum xor with x and
72     // any number of trie
73     int curr = 0, ans = 0;
74
75     for (int i = 31; i >= 0; i--) {
76         int b = ((x & (1 << i)) > 0);
77         int want = b^1;
78
79         if (trie[curr][want] == 0 || paths[trie[
80 curr][want]] == 0) want ^= 1;
81         if (trie[curr][want] == 0 || paths[trie[
82 curr][want]] == 0) break;
83         if (want != b) ans |= (1 << i);
84
85         curr = trie[curr][want];
86     }
87
88     return ans;
89 }

```

## 7 DP

### 7.1 Digit Dp

```

1 // Digit DP 1: https://atcoder.jp/contests/dp/tasks/
2 // dp_s
3 // find the number of integers between 1 and K (
4 // inclusive)
5 // where the sum of digits in base ten is a multiple
6 // of D
7
8 #include <bits/stdc++.h>
9
10 using namespace std;
11
12 const int MOD = 1e9+7;
13
14 string k;
15 int d;
16
17 int tb[10010][110][2];

```

```

16
17 int dp(int pos, int sum, bool under) {
18     if (pos >= k.size()) return sum == 0;
19
20     int& mem = tb[pos][sum][under];
21     if (mem != -1) return mem;
22     mem = 0;
23
24     int limit = 9;
25     if (!under) limit = k[pos] - '0';
26
27     for (int digit = 0; digit <= limit; digit++) {
28         mem += dp(pos+1, (sum + digit) % d, under | (
29 digit < limit));
30         mem %= MOD;
31     }
32
33     return mem;
34 }
35
36 int main() {
37     ios::sync_with_stdio(false);
38     cin.tie(NULL);
39
40     cin >> k >> d;
41
42     memset(tb, -1, sizeof(tb));
43
44     cout << (dp(0, 0, false) - 1 + MOD) % MOD << '\n'
45 ;
46
47     return 0;
48 }

```

### 7.2 Lcs

```

1 // LCS (Longest Common Subsequence)
2 //
3 // maior subsequencia comum entre duas strings
4 //
5 // tamanho da matriz da dp eh |a| x |b|
6 // lcs(a, b) = string da melhor resposta
7 // dp[a.size()][b.size()] = tamanho da melhor
8 // resposta
9 //
10 // https://atcoder.jp/contests/dp/tasks/dp_f
11 //
12 // 0(n^2)
13
14 string lcs(string a, string b) {
15     int n = a.size();
16     int m = b.size();
17
18     int dp[n+1][m+1];
19     pair<int, int> p[n+1][m+1];
20
21     memset(dp, 0, sizeof(dp));
22     memset(p, -1, sizeof(p));
23
24     for (int i = 1; i <= n; i++) {
25         for (int j = 1; j <= m; j++) {
26             if (a[i-1] == b[j-1]) {
27                 dp[i][j] = dp[i-1][j-1] + 1;
28                 p[i][j] = {i-1, j-1};
29             } else {
30                 if (dp[i-1][j] > dp[i][j-1]) {
31                     dp[i][j] = dp[i-1][j];
32                     p[i][j] = {i-1, j};
33                 } else {
34                     dp[i][j] = dp[i][j-1];
35                     p[i][j] = {i, j-1};
36                 }
37             }
38         }
39     }
40 }

```

```

37     }
38 }
39
40 // recuperar resposta
41
42 string ans = "";
43 pair<int, int> curr = {n, m};
44
45 while (curr.first != 0 && curr.second != 0) {
46     auto [i, j] = curr;
47
48     if (a[i-1] == b[j-1]) {
49         ans += a[i-1];
50     }
51
52     curr = p[i][j];
53 }
54
55 reverse(ans.begin(), ans.end());
56
57 return ans;
58 }

```

### 7.3 Lis Binary Search

```

1 int lis(vector<int> arr) {
2     vector<int> dp;
3
4     for (auto e : arr) {
5         int pos = lower_bound(dp.begin(), dp.end(), e
6         ) - dp.begin();
7
8         if (pos == (int)dp.size()) {
9             dp.push_back(e);
10        } else {
11            dp[pos] = e;
12        }
13    }
14
15    return (int)dp.size();
16 }

```

### 7.4 Edit Distance

```

1 // Edit Distance / Levenshtein Distance
2 //
3 // numero minimo de operacoes
4 // para transformar
5 // uma string em outra
6 //
7 // tamanho da matriz da dp eh |a| x |b|
8 // edit_distance(a.size(), b.size(), a, b)
9 //
10 // https://cses.fi/problemset/task/1639
11 //
12 // O(n^2)
13
14 int tb[MAX][MAX];
15
16 int edit_distance(int i, int j, string &a, string &b)
17 {
18     if (i == 0) return j;
19     if (j == 0) return i;
20
21     int &ans = tb[i][j];
22
23     if (ans != -1) return ans;
24
25     ans = min({
26         edit_distance(i-1, j, a, b) + 1,
27         edit_distance(i, j-1, a, b) + 1,
28         edit_distance(i-1, j-1, a, b) + (a[i-1] != b[
29         j-1])
30     });
31 }

```

```

28     });
29
30     return ans;
31 }

```

### 7.5 Digit Dp 2

```

1 // Digit DP 2: https://cses.fi/problemset/task/2220
2 //
3 // Number of integers between a and b
4 // where no two adjacent digits are the same
5
6 #include <bits/stdc++.h>
7
8 using namespace std;
9 using ll = long long;
10
11 const int MAX = 20; // 10^18
12
13 ll tb[MAX][MAX][2][2];
14
15 ll dp(string& number, int pos, int last_digit, bool
16     under, bool started) {
17     if (pos >= (int)number.size()) {
18         return 1;
19     }
20
21     ll& mem = tb[pos][last_digit][under][started];
22     if (mem != -1) return mem;
23     mem = 0;
24
25     int limit = 9;
26     if (!under) limit = number[pos] - '0';
27
28     for (int digit = 0; digit <= limit; digit++) {
29         if (started && digit == last_digit) continue;
30
31         bool is_under = under || (digit < limit);
32         bool is_started = started || (digit != 0);
33
34         mem += dp(number, pos+1, digit, is_under,
35         is_started);
36     }
37
38     return mem;
39 }
40
41 ll solve(ll ubound) {
42     memset(tb, -1, sizeof(tb));
43     string number = to_string(ubound);
44     return dp(number, 0, 10, 0, 0);
45 }
46
47 int main() {
48     ios::sync_with_stdio(false);
49     cin.tie(NULL);
50
51     ll a, b; cin >> a >> b;
52     cout << solve(b) - solve(a-1) << '\n';
53
54     return 0;
55 }

```

### 7.6 Lis Segtree

```

1 int n, arr[MAX], aux[MAX]; cin >> n;
2 for (int i = 0; i < n; i++) {
3     cin >> arr[i];
4     aux[i] = arr[i];
5 }
6
7 sort(aux, aux+n);

```

```

8
9 Segtree st(n); // seg of maximum
10
11 int ans = 0;
12 for (int i = 0; i < n; i++) {
13     int it = lower_bound(aux, aux+n, arr[i]) - aux;
14     int lis = st.query(0, it) + 1;
15
16     st.update(it, lis);
17
18     ans = max(ans, lis);
19 }
20
21 cout << ans << '\n';

```

## 7.7 Range Dp

```

1 // Range DP 1: https://codeforces.com/problemset/
  // problem/1132/F
2 //
3 // You may apply some operations to this string
4 // in one operation you can delete some contiguous
  // substring of this string
5 // if all letters in the substring you delete are
  // equal
6 // calculate the minimum number of operations to
  // delete the whole string s
7
8 #include <bits/stdc++.h>
9
10 using namespace std;
11
12 const int MAX = 510;
13
14 int n, tb[MAX][MAX];
15 string s;
16
17 int dp(int left, int right) {
18     if (left > right) return 0;
19
20     int& mem = tb[left][right];
21     if (mem != -1) return mem;
22
23     mem = 1 + dp(left+1, right); // gastar uma
  // operação arrumando a cara atual
24     for (int i = left+1; i <= right; i++) {
25         if (s[left] == s[i]) {
26             mem = min(mem, dp(left+1, i-1) + dp(i,
27             right));
28         }
29     }
30
31     return mem;
32 }
33
34 int main() {
35     ios::sync_with_stdio(false);
36     cin.tie(NULL);
37
38     cin >> n >> s;
39     memset(tb, -1, sizeof(tb));
40     cout << dp(0, n-1) << '\n';
41
42     return 0;
43 }

```

## 8 DS

### 8.1 Dsu

```

1 struct DSU {

```

```

2     int n;
3     vector<int> link, sizes;
4
5     DSU(int n) {
6         this->n = n;
7         link.assign(n+1, 0);
8         sizes.assign(n+1, 1);
9
10        for (int i = 0; i <= n; i++)
11            link[i] = i;
12    }
13
14    int find(int x) {
15        while (x != link[x])
16            x = link[x];
17
18        return x;
19    }
20
21    bool same(int a, int b) {
22        return find(a) == find(b);
23    }
24
25    void unite(int a, int b) {
26        a = find(a);
27        b = find(b);
28
29        if (a == b) return;
30
31        if (sizes[a] < sizes[b])
32            swap(a, b);
33
34        sizes[a] += sizes[b];
35        link[b] = a;
36    }
37 };

```

### 8.2 Ordered Set

```

1 // Ordered Set
2 //
3 // set roubado com mais operacoes
4 //
5 // para alterar para multiset
6 // trocar less para less_equal
7 //
8 // ordered_set<int> s
9 //
10 // order_of_key(k) // number of items strictly
  // smaller than k -> int
11 // find_by_order(k) // k-th element in a set (
  // counting from zero) -> iterator
12 //
13 // https://cses.fi/problemset/task/2169
14 //
15 // O(log N) para insert, erase (com iterator),
  // order_of_key, find_by_order
16
17 using namespace __gnu_pbds;
18 template <typename T>
19 using ordered_set = tree<T, null_type, less<T>,
  rb_tree_tag, tree_order_statistics_node_update>;
20
21 void erase(ordered_set& a, int x){
22     int r = a.order_of_key(x);
23     auto it = a.find_by_order(r);
24     a.erase(it);
25 }

```

### 8.3 Kruskal

```

1 struct Edge {

```

```
2     int u, v;
3     ll weight;
4
5     Edge() {}
6
7     Edge(int u, int v, ll weight) : u(u), v(v),
8     weight(weight) {}
9
10    bool operator<(Edge const& other) {
11        return weight < other.weight;
12    }
13
14    vector<Edge> kruskal(vector<Edge> edges, int n) {
15        vector<Edge> result;
16
17        ll cost = 0;
18
19        sort(edges.begin(), edges.end());
20        DSU dsu(n);
21
22        for (auto e : edges) {
23            if (!dsu.same(e.u, e.v)) {
24                cost += e.weight;
25                result.push_back(e);
26                dsu.unite(e.u, e.v);
27            }
28        }
29
30        return result;
31    }
```