

Competitive Programming Notebook

As Meninas Superpoderosas

Contents

1 DS	2	8 Graph	10
1.1 Bigk	2	8.1 Dinic	10
1.2 Ordered Set	2	8.2 Ford Fulkerson	11
1.3 Kruskal	2	8.3 Bfs	12
1.4 Dsu	3	8.4 2sat	12
2 String	3	8.5 Has Negative Cycle	13
2.1 Split	3	8.6 Dijkstra	13
2.2 Hash	3	8.7 Floyd Warshall	13
2.3 Trie Xor	4	8.8 Lca	14
2.4 Is Substring	4		
3 Math	5		
3.1 Log Any Base	5		
3.2 Is Prime	5		
3.3 Factorization	5		
3.4 Sieve	5		
3.5 Divisors	5		
3.6 Fexp	5		
3.7 Generate Primes	5		
3.8 Ceil	5		
4 Primitives	6		
5 General	6		
5.1 Last True	6		
5.2 Random	6		
5.3 Template	6		
5.4 Input By File	6		
5.5 Interactive	6		
5.6 Get Subsets Sum Iterative	6		
5.7 Xor 1 To N	6		
5.8 Next Permutation	6		
5.9 Min Priority Queue	7		
5.10 First True	7		
5.11 Base Converter	7		
6 Geometry	7		
6.1 Convex Hull	7		
7 DP	8		
7.1 Edit Distance	8		
7.2 Range Dp	8		
7.3 Lcs	9		
7.4 Digit Dp	9		
7.5 Digit Dp 2	9		
7.6 Lis Binary Search	10		
7.7 Lis Segtree	10		

1 DS

1.1 Bigk

```

1 struct SetSum {
2     ll sum;
3     multiset<ll> ms;
4
5     SetSum() {}
6
7     void add(ll x) {
8         sum += x;
9         ms.insert(x);
10    }
11
12    int rem(ll x) {
13        auto it = ms.find(x);
14
15        if (it == ms.end()) {
16            return 0;
17        }
18
19        sum -= x;
20        ms.erase(it);
21        return 1;
22    }
23
24    ll getMin() { return *ms.begin(); }
25
26    ll getMax() { return *ms.rbegin(); }
27
28    ll getSum() { return sum; }
29
30    int size() { return (int)ms.size(); }
31 };
32
33 struct BigK {
34     int k;
35     SetSum gt, mt;
36
37     BigK(int k): k(k) {}
38
39     void balance() {
40         while (gt.size() > k) {
41             ll mn = gt.getMin();
42             gt.rem(mn);
43             mt.add(mn);
44         }
45
46         while (gt.size() < k && mt.size() > 0) {
47             ll mx = mt.getMax();
48             mt.rem(mx);
49             gt.add(mx);
50         }
51     }
52
53     void add(ll x) {
54         gt.add(x);
55         balance();
56     }
57
58     void rem(ll x) {
59         if (mt.rem(x) == 0) {
60             gt.rem(x);
61         }
62
63         balance();
64     }
65
66     // be careful, O(abs(oldK - newK) * log)
67     void setK(int _k) {
68         k = _k;

```

```

69         balance();
70     }
71
72     // O(log)
73     void incK() { setK(k + 1); }
74
75     // O(log)
76     void decK() { setK(k - 1); }
77 };

```

1.2 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
11 // smaller than k -> int
12 // find_by_order(k) // k-th element in a set (
13 // counting from zero) -> iterator
14 //
15 // https://cses.fi/problemset/task/2169
16 //
17 // O(log N) para insert, erase (com iterator),
18 // order_of_key, find_by_order
19
20 using namespace __gnu_pbds;
21 template <typename T>
22 using ordered_set = tree<T, null_type, less<T>,
23 rb_tree_tag, tree_order_statistics_node_update>;
24
25 void erase(ordered_set& a, int x){
26     int r = a.order_of_key(x);
27     auto it = a.find_by_order(r);
28     a.erase(it);
29 }

```

1.3 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
15 vector<Edge> kruskal(vector<Edge> edges, int n) {
16     vector<Edge> result;
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 }

```

```

28
29     return result;
30 }

```

1.4 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 };

```

2 String

2.1 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 }

```

2.2 Hash

```

1 struct Hash {
2     ll MOD, P;
3     int n; string s;
4     vector<ll> h, hi, p;
5     Hash() {}
6     Hash(string s, ll MOD, ll P = 31): s(s), MOD(MOD),
7     , P(P), n(s.size()), h(n), hi(n), p(n) {
8         for (int i=0;i<n;i++) p[i] = (i ? P*p[i-1]:1)
9         % MOD;
10        for (int i=0;i<n;i++)
11            h[i] = (s[i] + (i ? h[i-1]:0) * P) % MOD;
12        for (int i=n-1;i>=0;i--)
13            hi[i] = (s[i] + (i+1<n ? hi[i+1]:0) * P)
14            % MOD;
15    }
16
17    int query(int l, int r) {
18        ll hash = (h[r] - (l ? h[l-1]*p[r-l+1]%MOD :
19        0));
20        return hash < 0 ? hash + MOD : hash;
21    }
22
23    int query_inv(int l, int r) {
24        ll hash = (hi[l] - (r+1 < n ? hi[r+1]*p[r-1
25        +1] % MOD : 0));
26        return hash < 0 ? hash + MOD : hash;
27    }
28 }
29
30 struct DoubleHash {
31     const ll MOD1 = 90264469;
32     const ll MOD2 = 25699183;
33
34     Hash hash1, hash2;
35
36     DoubleHash();
37
38     DoubleHash(string s) : hash1(s, MOD1), hash2(s,
39     MOD2) {}
40
41     pair<int, int> query(int l, int r) {
42         return { hash1.query(l, r), hash2.query(l, r)
43         };
44     }
45
46     pair<int, int> query_inv(int l, int r) {
47         return { hash1.query_inv(l, r), hash2.
48         query_inv(l, r) };
49     }
50 }
51
52 struct TripleHash {
53     const ll MOD1 = 90264469;
54     const ll MOD2 = 25699183;
55     const ll MOD3 = 81249169;
56
57     Hash hash1, hash2, hash3;
58
59     TripleHash();
60
61     TripleHash(string s) : hash1(s, MOD1), hash2(s,
62     MOD2), hash3(s, MOD3) {}
63
64     tuple<int, int, int> query(int l, int r) {
65         return { hash1.query(l, r), hash2.query(l, r)
66         , hash3.query(l, r) };
67     }
68
69     tuple<int, int, int> query_inv(int l, int r) {
70         return { hash1.query_inv(l, r), hash2.
71         query_inv(l, r), hash3.query_inv(l, r) };
72     }
73 }
74
75 struct HashK {

```

```

63 vector<ll> primes; // more primes = more hashes
64 vector<Hash> hash;
65
66 HashK();
67
68 HashK(string s, vector<ll> primes): primes(primes) {
69     for (auto p : primes) {
70         hash.push_back(Hash(s, p));
71     }
72 }
73
74 vector<int> query(int l, int r) {
75     vector<int> ans;
76
77     for (auto h : hash) {
78         ans.push_back(h.query(l, r));
79     }
80
81     return ans;
82 }
83
84 vector<int> query_inv(int l, int r) {
85     vector<int> ans;
86
87     for (auto h : hash) {
88         ans.push_back(h.query_inv(l, r));
89     }
90
91     return ans;
92 }
93 };

```

2.3 Trie Xor

```

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

```

```

36         curr = trie[curr][b];
37     }
38
39     paths[curr]++;
40     finish[curr]++;
41 }
42
43 void rem(int x) {
44     int curr = 0;
45
46     for (int i = 31; i >= 0; i--) {
47         int b = ((x << 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 << 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
any number of trie
72     int curr = 0, ans = 0;
73
74     for (int i = 31; i >= 0; i--) {
75         int b = ((x << i) > 0);
76         int want = b^1;
77
78         if (trie[curr][want] == 0 || paths[trie[
curr][want]] == 0) want ^= 1;
79         if (trie[curr][want] == 0 || paths[trie[
curr][want]] == 0) break;
80         if (want != b) ans |= (1 << i);
81
82         curr = trie[curr][want];
83     }
84
85     return ans;
86 }
87 };

```

2.4 Is Substring

```

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

```

```

15         return true;
16     }
17 }
18
19 return false;
20 }

```

3 Math

3.1 Log Any Base

```

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

```

3.2 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 }

```

3.3 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 }

```

3.4 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 }

```

3.5 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 }

```

3.6 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 }

```

3.7 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 }

```

3.8 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 Primitives

5 General

5.1 Last True

```

1 // Binary Search (last_true)
2
3 // last_true(2, 10, [](int x) { return x * x <= 30;
4 //     }); // outputs 5
5 //
6 // [1, r]
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     return lo;
35 }

```

5.2 Random

```

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

```

5.3 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 }

```

5.4 Input By File

```

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

```

5.5 Interactive

```

1 // you should use cout.flush() every cout
2 int query(int a) {
3     cout << "? " << a << '\n';
4     cout.flush();
5     char res; cin >> res;
6     return res;
7 }
8
9 // using endl you don't need
10 int query(int a) {
11     cout << "? " << a << endl;
12     char res; cin >> res;
13     return res;
14 }

```

5.6 Get Subsets Sum Iterative

```

1 vector<ll> get_subset_sums(int l, int r, vector<ll>&
2     arr) {
3     vector<ll> ans;
4
5     int len = r-l+1;
6     for (int i = 0; i < (1 << len); i++) {
7         ll sum = 0;
8
9         for (int j = 0; j < len; j++) {
10             if (i & (1 << j)) {
11                 sum += arr[l + j];
12             }
13         }
14         ans.push_back(sum);
15     }
16
17     return ans;
18 }

```

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 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()));

```

5.9 Min Priority Queue

```

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

```

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 // [1, 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
15 int first_true(int lo, int hi, function<bool(int)> f)
16 {
17     hi++;
18     while (lo < hi) {
19         int mid = lo + (hi - lo) / 2;
20
21         if (f(mid)) {
22             hi = mid;
23         } else {
24             lo = mid + 1;
25         }
26     }
27     return lo;
28 }

```

5.11 Base Converter

```

1 const string digits = "0123456789
2 ABCDEFGHIJKLMNOPQRSTUVWXYZ";
3
4 ll tobase10(string number, int base) {
5     map<char, int> val;
6     for (int i = 0; i < digits.size(); i++) {
7         val[digits[i]] = i;
8     }
9
10    ll ans = 0, pot = 1;
11
12    for (int i = number.size() - 1; i >= 0; i--) {
13        ans += val[number[i]] * pot;
14        pot *= base;
15    }
16
17    return ans;
18 }
19
20 string frombase10(ll number, int base) {
21     if (number == 0) return "0";

```

```

21 string ans = "";
22
23 while (number > 0) {
24     ans += digits[number % base];
25     number /= base;
26 }
27
28 reverse(ans.begin(), ans.end());
29
30 return ans;
31 }
32
33 // verifica se um número está na base especificada
34 bool verify_base(string num, int base) {
35     map<char, int> val;
36     for (int i = 0; i < digits.size(); i++) {
37         val[digits[i]] = i;
38     }
39
40     for (auto digit : num) {
41         if (val[digit] >= base) {
42             return false;
43         }
44     }
45
46     return true;
47 }

```

6 Geometry

6.1 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 // https://cses.fi/problemset/task/2195/
7 // https://open.kattis.com/problems/convexhull (
8 // counterclockwise)
9 //
10 // O(n log(n))
11
12 typedef long long ftype;
13
14 struct Point {
15     ftype x, y;
16
17     Point() {}
18     Point(ftype x, ftype y) : x(x), y(y) {}
19
20     bool operator<(Point o) {
21         if (x == o.x) return y < o.y;
22         return x < o.x;
23     }
24
25     bool operator==(Point o) {
26         return x == o.x && y == o.y;
27     }
28 };
29
30 ftype cross(Point a, Point b, Point c) {
31     // v: a -> c
32     // w: a -> b
33
34     // v: c.x - a.x, c.y - a.y
35     // w: b.x - a.x, b.y - a.y
36
37     return (c.x - a.x) * (b.y - a.y) - (c.y - a.y) *
38           (b.x - a.x);

```

```

37 }
38
39 ftype dir(Point a, Point b, Point c) {
40     // 0 -> colineares
41     // -1 -> esquerda
42     // 1 -> direita
43
44     ftype cp = cross(a, b, c);
45
46     if (cp == 0) return 0;
47     else if (cp < 0) return -1;
48     else return 1;
49 }
50
51 vector<Point> convex_hull(vector<Point> points) {
52     sort(points.begin(), points.end());
53     points.erase( unique(points.begin(), points.end())
54                 , points.end()); // somente pontos distintos
55     int n = points.size();
56
57     if (n == 1) return { points[0] };
58
59     vector<Point> upper_hull = {points[0], points[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
74     vector<Point> lower_hull = {points[n-1], points[n-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         upper_hull.push_back(lower_hull[i]);
94     }
95
96     return upper_hull;
97 }

```

7 DP

7.1 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[j-1])
29     });
30
31     return ans;
32 }

```

7.2 Range Dp

```

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

```



```

35     cin.tie(NULL);
36
37     cin >> n >> s;
38     memset(tb, -1, sizeof(tb));
39     cout << dp(0, n-1) << '\n';
40
41     return 0;
42 }

```

7.3 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 // https://atcoder.jp/contests/dp/tasks/dp_f
10 //
11 // O(n^2)
12
13 string lcs(string a, string b) {
14     int n = a.size();
15     int m = b.size();
16
17     int dp[n+1][m+1];
18     pair<int, int> p[n+1][m+1];
19
20     memset(dp, 0, sizeof(dp));
21     memset(p, -1, sizeof(p));
22
23     for (int i = 1; i <= n; i++) {
24         for (int j = 1; j <= m; j++) {
25             if (a[i-1] == b[j-1]) {
26                 dp[i][j] = dp[i-1][j-1] + 1;
27                 p[i][j] = {i-1, j-1};
28             } else {
29                 if (dp[i-1][j] > dp[i][j-1]) {
30                     dp[i][j] = dp[i-1][j];
31                     p[i][j] = {i-1, j};
32                 } else {
33                     dp[i][j] = dp[i][j-1];
34                     p[i][j] = {i, j-1};
35                 }
36             }
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 }

```

7.4 Digit Dp

```

1 // Digit DP 1: https://atcoder.jp/contests/dp/tasks/
  dp_s
2 //
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
5
6 #include <bits/stdc++.h>
7
8 using namespace std;
9
10 const int MOD = 1e9+7;
11
12 string k;
13 int d;
14
15 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 | (
  digit < limit));
29         mem %= MOD;
30     }
31
32     return mem;
33 }
34
35 int main() {
36     ios::sync_with_stdio(false);
37     cin.tie(NULL);
38
39     cin >> k >> d;
40
41     memset(tb, -1, sizeof(tb));
42
43     cout << (dp(0, 0, false) - 1 + MOD) % MOD << '\n'
  ;
44
45     return 0;
46 }

```

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
  under, bool started) {
16     if (pos >= (int)number.size()) {
17         return 1;
18     }

```

```

19
20 ll& mem = tb[pos][last_digit][under][started];
21 if (mem != -1) return mem;
22 mem = 0;
23
24 int limit = 9;
25 if (!under) limit = number[pos] - '0';
26
27 for (int digit = 0; digit <= limit; digit++) {
28     if (started && digit == last_digit) continue;
29
30     bool is_under = under || (digit < limit);
31     bool is_started = started || (digit != 0);
32
33     mem += dp(number, pos+1, digit, is_under,
34 is_started);
35 }
36 return mem;
37 }
38
39 ll solve(ll ubound) {
40     memset(tb, -1, sizeof(tb));
41     string number = to_string(ubound);
42     return dp(number, 0, 10, 0, 0);
43 }
44
45 int main() {
46     ios::sync_with_stdio(false);
47     cin.tie(NULL);
48
49     ll a, b; cin >> a >> b;
50     cout << solve(b) - solve(a-1) << '\n';
51
52     return 0;
53 }

```

7.6 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        return (int)dp.size();
15    }

```

7.7 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';

```

8 Graph

8.1 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(
20 to), cap(cap), flow(0) {};
```

```

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     q.push(s);
61     level[s] = 1;
62
63     while (q.size()) {
64         int curr = q.front();
65         q.pop();
66
67         for (auto edge : adj[curr]) {
68             if (edge->remaining_cap() > 0 &&
69 level[edge->to] == -1) {
70                 level[edge->to] = level[curr] +
71 1;
72                 q.push(edge->to);
73             }
74         }
75     }
76     return level[t] != -1;
77 }
78
79 ll dfs(int x, int t, ll flow) {
80     if (x == t) return flow;
81
82     for (int& cid = next[x]; cid < (int)adj[x].
83 size(); cid++) {
84         auto& edge = adj[x][cid];
85         ll cap = edge->remaining_cap();
86
87         if (cap > 0 && level[edge->to] == level[x
88 ] + 1) {
89             ll sent = dfs(edge->to, t, min(flow,
90 cap)); // bottle neck
91             if (sent > 0) {
92                 edge->augment(sent);
93                 return sent;
94             }
95         }
96     }
97     return 0;
98 }
99
100 ll solve(int s, int t) {
101     ll max_flow = 0;
102
103     while (bfs(s, t)) {
104         fill(next.begin(), next.end(), 0);
105         while (ll sent = dfs(s, t, FLOW_INF)) {
106             max_flow += sent;
107         }
108     }
109     return max_flow;
110 }
111
112 // path recover
113 vector<bool> vis;
114 vector<int> curr;
115
116 bool dfs2(int x, int& t) {
117     vis[x] = true;
118     bool arrived = false;
119
120     if (x == t) {
121         curr.push_back(x);
122         return true;
123     }
124
125     for (auto e : adj[x]) {
126         if (e->flow > 0 && !vis[e->to]) { // !e->
is_residual() &&

```

```

127         bool aux = dfs2(e->to, t);
128
129         if (aux) {
130             arrived = true;
131             e->flow--;
132         }
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 }

```

8.2 Ford Fulkerson

```

1 // Ford-Fulkerson
2 //
3 // max-flow / min-cut
4 //
5 // MAX nÃss
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
14 struct Flow {
15     int n;
16     ll adj[MAX][MAX];
17     bool used[MAX];
18
19     Flow(int n) : n(n) {};
20
21     void add_edge(int u, int v, ll c) {
22         adj[u][v] += c;
23         adj[v][u] = 0; // cuidado com isso
24     }
25
26     ll dfs(int x, int t, ll amount) {
27         used[x] = true;
28
29         if (x == t) return amount;
30
31         for (int i = 1; i <= n; i++) {
32             if (adj[x][i] > 0 && !used[i]) {
33                 ll sent = dfs(i, t, min(amount, adj[x
34 ][i]));
35
36                 if (sent > 0) {
37                     adj[x][i] -= sent;
38                     adj[i][x] += sent;
39
40                     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 };

```

8.3 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 }

```

8.4 2sat

```

1 // 2SAT
2 //
3 // verifica se existe e encontra solu~ão
4 // para f~ormulas 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)
  ^1 = 4
10 //
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 n~ão
  foram testadas)
14 //
15 // O(n + m)
16
17 struct sat {
18     int n, tot;
19     vector<vector<int>> adj, adjt; // grafo original,
  grafo transposto
20     vector<int> vis, comp, ans;
21     stack<int> topo; // ordem topol~ogica
22
23     sat() {}
24     sat(int n_) : n(n_), tot(n), adj(2*n), adjt(2*n)
  {}
25
26     void dfs(int x) {
27         vis[x] = true;
28
29         for (auto e : adj[x]) {
30             if (!vis[e]) dfs(e);
31         }
32
33         topo.push(x);
34     }
35
36     void dfst(int x, int& id) {
37         vis[x] = true;
38         comp[x] = id;
39
40         for (auto e : adjt[x]) {
41             if (!vis[e]) dfst(e, id);
42         }
43     }
44
45     void add_impl(int a, int b) { // a -> b = (!a or
  b)
46         a = (a >= 0 ? 2*a : -2*a-1);
47         b = (b >= 0 ? 2*b : -2*b-1);
48
49         adj[a].push_back(b);
50         adj[b^1].push_back(a^1);
51
52         adjt[b].push_back(a);
53         adjt[a^1].push_back(b^1);
54     }
55
56     void add_or(int a, int b) { // a or b
57         add_impl(~a, b);
58     }
59
60     void add_nor(int a, int b) { // a nor b = !(a or
  b)
61         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
  and b)
69         add_or(~a, ~b);
70     }
71
72     void add_xor(int a, int b) { // a xor b = (a != b

```

```

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

```

8.5 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] +
w) {
15                dist[v] = dist[u] + w;
16            }
17        }
18    }
19
20    for (auto [u, v, w] : edges) {
21        if (dist[u] < INF && dist[v] > dist[u] + w) {
22            return true;
23        }
24    }
25
26    return false;
27 }

```

8.6 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 }

```

8.7 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                             dist[k][j]);
26         }
27     }

```

8.8 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++) {

```

```

25         if (up[i][j-1] != -1)
26             up[i][j] = up[ up[i][j-1] ][j-1];
27     }
28 }
29
30 void dfs(int x, int p, vector<vector<int>>& adj)
31 {
32     up[x][0] = p;
33     for (auto e : adj[x]) {
34         if (e != p) {
35             dep[e] = dep[x] + 1;
36             dfs(e, x, adj);
37         }
38     }
39 }
40
41 int jump(int x, int k) { // jump from node x k
42     times
43     for (int i = 0; i < MAXE; i++) {
44         if (k && (1 << i) && x != -1) x = up[x][i];
45     }
46     return x;
47 }
48
49 int lca(int a, int b) {
50     if (dep[a] > dep[b]) swap(a, b);
51     b = jump(b, dep[b] - dep[a]);
52
53     if (a == b) return a;
54
55     for (int i = MAXE-1; i >= 0; i--) {
56         if (up[a][i] != up[b][i]) {
57             a = up[a][i];
58             b = up[b][i];
59         }
60     }
61     return up[a][0];
62 }
63
64 int dist(int a, int b) {
65     return dep[a] + dep[b] - 2 * dep[lca(a, b)];
66 }
67 };

```