

Competitive Programming Notebook

Contents

1 Range queries	2	7.8 Dijkstra	15
1.1 Segment Tree Comprimida	2	7.9 Dinic	15
1.2 Segment Tree Base	2	7.10 Dsu	16
1.3 Prefix Sum 2d	3	7.11 Pontes	17
2 String	3	7.12 Floyd Warshall	17
2.1 Suffix Array	3	7.13 Scc	17
2.2 Trie	4		
2.3 3 Hash	4		
2.4 Hash Sem Ordem	4		
3 Geometria	5		
3.1 Convex Hull	5		
3.2 Ponto	6		
4 Buscas e stl	6		
4.1 Subset Sum	6		
4.2 Busca Binaria	7		
4.3 Kadane	7		
4.4 Stack Monotonica	7		
5 Matematica	8		
5.1 Matrizes	8		
5.2 Fexp E Comuns	8		
5.3 Divisores	8		
5.4 Fatoracao Prima E Spf	8		
5.5 Crivo	9		
5.6 Combinacao	9		
5.7 Modulo Struct	9		
6 Dp	10		
6.1 Operacoes-bitwise	10		
6.2 Digit Dp	10		
6.3 Knapsack 2d	10		
6.4 Knapsack 1d	11		
6.5 Lcis	11		
6.6 Moedas	11		
6.7 Lcs	11		
6.8 Tree Matching	12		
7 Grafos	12		
7.1 Bicolorabilidade	12		
7.2 Prim	12		
7.3 Dfs	13		
7.4 Bfs	13		
7.5 Achar Ciclos	14		
7.6 Toposort	14		
7.7 Euler Tour E Lca	14		

1 Range queries

1.1 Segment Tree Comprimida

```

1 #include <bits/stdc++.h>
2 #define endl '\n'
3 using namespace std;
4
5 void update(int no, int l, int r, int pos, int val,
6             vector<int> &segtree){
7     if((pos < l) or (r < pos)){
8         return;
9     }
10    if(l == r){
11        segtree[no] += val;
12        return;
13    }
14
15    int mid = (l + r)/2;
16    update(2* no, l, mid, pos, val, segtree);
17    update((2* no) + 1, mid + 1, r, pos, val, segtree);
18    segtree[no] = segtree[2 * no] + segtree[(2 * no)+
19        1];
20 }
21
22 int query(int no, int l, int r, int lq, int rq,
23           vector<int> &segtree){
24     if((rq < l) or (r < lq)){
25         return 0;
26     }
27     if((lq <= l) and (r <= rq)){
28         return segtree[no];
29     }
30
31     int mid = (l + r)/2;
32     int ans = query(2 * no, l, mid, lq, rq, segtree);
33     ans += query((2 * no) + 1, mid + 1, r, lq, rq,
34                 segtree);
35
36     return ans;
37 }
38
39 int main(){
40     ios_base::sync_with_stdio(false);
41     cin.tie(NULL);
42
43     int n, q, sal, k, j;
44     char c;
45     vector<int> all;
46     vector<int> orig;
47     vector<tuple<char, int, int>> queries;
48     sort(all.begin(), all.end());
49     all.erase(unique(all.begin(), all.end()), all.end());
50
51     int range;
52     vector<int> segtree(4 * (all.size()), 0);
53     for(int i = 0; i < orig.size(); i++){
54         range = lower_bound(all.begin(), all.end(),
55                             orig[i]) - all.begin();
56         update(1, 1, all.size(), range, 1, segtree);
57     }
58
59     return 0;
60 }

```

1.2 Segment Tree Base

```

1 #include <bits/stdc++.h>
2 #define endl '\n'
3 using namespace std;

```

```

4
5 void build(int no, int l, int r, vector<long long int>
6            &segtree, vector<long long int> &orig){
7     if(l == r){
8         segtree[no] = orig[l];
9         return;
10    }
11
12    int mid = (l + r)/2;
13    build(2 * no, l, mid, segtree, orig);
14    build((2 * no) + 1, mid + 1, r, segtree, orig);
15    segtree[no] = segtree[2 * no] + segtree[(2 * no)
16        + 1];
17 }
18
19 void update(int no, int l, int r, int pos, long long
20             int val, vector<long long int> &segtree){
21     if((pos < l) or (r < pos)){
22         return;
23     }
24     if(l == r){
25         segtree[no] = val;
26         return;
27     }
28
29     int mid = (l + r)/2;
30     update(2* no, l, mid, pos, val, segtree);
31     update((2* no) + 1, mid + 1, r, pos, val, segtree);
32
33     segtree[no] = segtree[2 * no] + segtree[(2 * no)+
34        1];
35 }
36
37 long long int query(int no, int l, int r, int lq, int
38                     rq, vector<long long int> &segtree){
39     if((rq < l) or (r < lq)){
40         return 0;
41     }
42     if((lq <= l) and (r <= rq)){
43         return segtree[no];
44     }
45
46     int mid = (l + r)/2;
47     long long int ans = query(2 * no, l, mid, lq, rq,
48                             segtree);
49     ans += query((2 * no) + 1, mid + 1, r, lq, rq,
50                 segtree);
51
52     return ans;
53 }
54
55 int main(){
56     ios_base::sync_with_stdio(false);
57     cin.tie(NULL);
58     int n;
59     vector<long long int> segtree(4 * n);
60     //ADICIONAR X EM L E -X EM R+1 EH IGUAL A
61     ADICIONAR X EM [L, R]
62     //SEGMENTO PARA ACHAR MENORES:
63     // for(int i = 0; i < n; i++){
64     //     range = upper_bound(mansort.begin(),
65                             mansort.end(), man[i]) - mansort.begin();
66     //     dir[i] = query(1, 1, n, range, n, segtree);
67     // }
68
69     // update(1, 1, n, range, 1, segtree);
70
71     //SegEuler -> usar updates ao inves do build (ou
72     mapear vetor a[euler_in[i]] -> orig[i]
73     // query em euler_in[i], euler_out[i]
74     // update em euler_in[i]

```

```
65     return 0;
66 }
```

1.3 Prefix Sum 2d

```
1 #include <bits/stdc++.h>
2 #define endl '\n'
3 using namespace std;
4
5 int main(){
6     ios_base::sync_with_stdio(false);
7     cin.tie(NULL);
8
9     int xi, yi, yf, xf, n, q;
10    vector<vector<int>> prefsum(n + 1, vector<int>(n
11    + 1, 0));
12    vector<vector<int>> floresta;
13    for(int i = 1; i <= n; i++){
14        for(int j = 1; j <= n; j++){
15            prefsum[i][j] = floresta[i - 1][j - 1] +
16            prefsum[i - 1][j] + prefsum[i][j - 1] - prefsum[i
17            - 1][j - 1];
18        }
19    }
20    for(int i = 0; i < q; i++){
21        cin >> yi;
22        cin >> xi;
23        cin >> yf;
24        cin >> xf;
25
26        cout << (prefsum[yf][xf] - prefsum[yf][xi -
27        1] - prefsum[yi - 1][xf] + prefsum[yi - 1][xi -
28        1]) << endl;
29    }
30
31    return 0;
32 }
```

2 String

2.1 Suffix Array

```
1 #include <bits/stdc++.h>
2 #define endl '\n'
3 #define ll long long int
4 using namespace std;
5
6 struct SA{
7     int sz;
8     vector<ll> suffarr, endrank, lcp;
9     SA(int s){
10         sz = s;
11         suffarr = vector<ll>(sz+1); //espaço p/
12         sentinela vai ser apagado
13         endrank = vector<ll>(sz);
14         lcp = vector<ll>(sz);
15     }
16     void suffarray(string &s){
17         //ajuste a sentinela se necessario
18         s.push_back(char(0));
19         const ll n = s.size();
20         vector<ll> rank(n), temp(n);
21         vector<pair<ll, ll>> etc(n);
22         for(int i = 0; i < n; i++){
23             etc[i] = {s[i], i};
24         }
25         sort(etc.begin(), etc.end());
26         for(int i = 0; i < n; i++){
27             suffarr[i] = etc[i].second;
```

```
28         rank[suffarr[0]] = 0;
29         for(int i = 1; i < n; i++){
30             rank[suffarr[i]] = rank[suffarr[i-1]] + (
31             etc[i].first != etc[i-1].first);
32         }
33
34         for(int k = 1; k < n; k *= 2){
35             int classes = rank[suffarr[n-1]] + 1;
36             vector<ll> cnt(classes, 0);
37             for(int i = 0; i < n; i++) cnt[rank[i
38             ]]]++;
39
40             vector<ll> pos(classes, 0);
41             for(int i = 1; i < classes; i++) pos[i] =
42             pos[i-1] + cnt[i-1];
43
44             for(int i = 0; i < n; i++){
45                 int j = suffarr[i] - k;
46                 if(j < 0) j+=n;
47                 temp[pos[rank[j]]++] = j;
48             }
49             suffarr = temp;
50
51             temp[suffarr[0]] = 0;
52             for(int i = 1; i < n; i++){
53                 pair<ll, ll> prev = {rank[suffarr[i
54                 -1]], rank[(suffarr[i-1] + k) % n]};
55                 pair<ll, ll> curr = {rank[suffarr[i
56                 ]], rank[(suffarr[i] + k) % n]};
57
58                 temp[suffarr[i]] = temp[suffarr[i-1]]
59                 + (curr != prev);
60             }
61             rank = temp;
62         }
63         s.pop_back();
64         suffarr.erase(suffarr.begin());
65         return;
66     }
67 }
68
69 //lcp[i] = lcp de i-1 e i; lcp[0] deve ser
70 ignorado
71 void lcpv(string &s){
72     s.push_back(char(0));
73     int n = suffarr.size();
74
75     for(int i = 0; i < n; i++){
76         endrank[suffarr[i]] = i;
77     }
78     ll k = 0;
79
80     for(int i = 0; i < n; i++){
81         if(endrank[i] == 0) lcp[0] = 0;
82         else{
83             ll j = suffarr[endrank[i] - 1];
84             while(s[i+k] == s[j+k]) k++;
85             lcp[endrank[i]] = k;
86             if(k > 0) k--;
87         }
88     }
89     s.pop_back();
90     return;
91 }
92
93 //retorna tamanho do lcp entre sufixos da posicao
94 i e j
95 //so pra mostrar que eh uma query de minimo no
96 lcp
97 //ao inves disso use seg
98 ll get_lcp(int i, int j){
99     int l = endrank[i];
100    int r = endrank[j];
101    if(l > r) swap(l, r);
```

```

92         if(i==j) return (sz - suffarr[l]);
93
94         ll ans = lcp[l+1];
95         for(int k = l+2; k <= r; k++){
96             ans = min(lcp[k], ans);
97         }
98         return ans;
99     }
100 };

```

2.2 Trie

```

1  #include <bits/stdc++.h>
2  #define endl '\n'
3  using namespace std;
4
5  struct Node{
6      int next[26];
7      int subtree = 0;
8  };
9
10 void add(string s, vector<Node> &trie){
11     int curr = 0;
12
13     for(auto c : s){
14         if(trie[curr].next[c - 'a'] == 0){
15             trie[curr].next[c - 'a'] = trie.size();
16             trie.push_back(Node());
17         }
18
19         trie[curr].subtree += 1;
20         curr = trie[curr].next[c - 'a'];
21     }
22     trie[curr].subtree += 1;
23 }
24
25 int query(string s, vector<Node> &trie){
26     int curr = 0;
27
28     for(auto c : s){
29         if(trie[curr].next[c - 'a'] == 0){
30             return 0;
31         }
32         curr = trie[curr].next[c - 'a'];
33     }
34     return trie[curr].subtree;
35 }
36
37 int main() {
38     ios_base::sync_with_stdio(false);
39     cin.tie(NULL);
40
41     vector<Node> trie(1);
42     //trie pode ser modificada com DFS para propagar
43     //mudancas
44     //TRIE DE XOR -> max(busca diferentes) e min(
45     //busca igual)
46     return 0;
47 }

```

2.3 3 Hash

```

1  #include <bits/stdc++.h>
2  #define endl '\n'
3  using namespace std;
4
5  int main() {
6      ios_base::sync_with_stdio(false);
7      cin.tie(NULL);
8
9

```

```

10     const long long int mod = 1000000009;
11     const long long int mod2 = 1000000007;
12     const long long int mod3 = 999999937;
13     string s;
14     cin >> s;
15     const long long int k = 277;
16     const long long int l = 149;
17     const long long int p = 37;
18     vector<long long int> pot(s.size() + 1);
19     vector<long long int> pot2(s.size() + 1);
20     pot[0] = 1;
21     pot2[0] = 1;
22     for(int p = 1; p <= s.size(); p++){
23         pot[p] = (pot[p - 1] * k) % mod;
24         pot2[p] = (pot2[p - 1] * l) % mod2;
25     }
26     vector<long long int> hashupto1(s.size());
27     vector<long long int> hashupto2(s.size());
28     hashupto1[0] = s[0];
29     hashupto2[0] = s[0];
30     for(int i = 1; i < s.size(); i++){
31         hashupto1[i] = ((hashupto1[i - 1] * k) % mod)
32         + s[i];
33         hashupto1[i] = hashupto1[i] % mod;
34         hashupto2[i] = ((hashupto2[i - 1] * l) % mod2
35         + s[i]);
36         hashupto2[i] = hashupto2[i] % mod2;
37     }
38     //hash(1..r) = pref(r) - (pref(1 - 1) * (k^(r-1
39     +1))) % MOD
40     //aa = hashupto1[i + (pref.size() - 1)] - ((
41     hashupto1[i - 1] * pot[pref.size()]) % mod);
42     //aa = (((aa % mod) + mod) % mod);
43     //bb = hashupto2[i + (pref.size() - 1)] - ((
44     hashupto2[i - 1] * pot2[pref.size()]) % mod2);
45     //bb = (((bb % mod2) + mod2) % mod2);
46     //achar periodos facil: hash(0..(n-p)) == hash(p
47     ...n) -> so se n for parcial
48
49     return 0;
50 }

```

2.4 Hash Sem Ordem

```

1  #include <bits/stdc++.h>
2  #define endl '\n'
3  using namespace std;
4
5
6
7  int main() {
8      ios_base::sync_with_stdio(false);
9      cin.tie(NULL);
10
11     const long long int k = 277;
12     const long long int l = 149;
13     const long long int p = 37;
14
15     const long long int mod = 1000000009;
16     const long long int mod2 = 1000000007;
17     const long long int mod3 = 999999937;
18
19     int n, q;
20     cin >> n;
21     cin >> q;
22     vector<long long int> num(n);
23     vector<long long int> bnum(n);
24     vector<long long int> pot(n+1);
25     vector<long long int> pot2(n+1);
26     vector<long long int> pot3(n+1);
27     pot[0] = 1;
28     pot[1] = 1;

```

```

29 pot[2] = 1;
30 for(int po = 1; po <= n; po++){
31     pot[po] = (pot[po - 1] * k) % mod;
32     pot2[po] = (pot2[po - 1] * 1) % mod2;
33     pot3[po] = (pot3[po - 1] * p) % mod3;
34 }
35 for(int i = 0; i < n; i++){
36     cin >> num[i];
37 }
38 for(int i = 0; i < n; i++){
39     cin >> bnum[i];
40 }
41 vector<long long int> hashsuml(n+1);
42 vector<long long int> hashsumk(n+1);
43 vector<long long int> hashsump(n+1);
44 hashsumk[0] = 0;
45 hashsuml[0] = 0;
46 hashsump[0] = 0;
47 hashsumk[1] = (num[0] * pot[num[0]]) % mod;
48 hashsuml[1] = (num[0] * pot2[num[0]]) % mod2;
49 hashsump[1] = (num[0] * pot3[num[0]]) % mod3;
50
51 vector<long long int> bhashsuml(n+1);
52 vector<long long int> bhashsumk(n+1);
53 vector<long long int> bhashsump(n+1);
54 bhashsumk[0] = 0;
55 bhashsuml[0] = 0;
56 bhashsump[0] = 0;
57 bhashsumk[1] = (bnum[0] * pot[bnum[0]]) % mod;
58 bhashsuml[1] = (bnum[0] * pot2[bnum[0]]) % mod2;
59 bhashsump[1] = (bnum[0] * pot3[bnum[0]]) % mod3;
60 for(int i = 1; i < n; i++){
61     hashsumk[i+1] = ((hashsumk[i] + ((num[i] *
62     pot[num[i]]) % mod)) % mod);
63     hashsuml[i+1] = ((hashsuml[i] + ((num[i] *
64     pot2[num[i]]) % mod2)) % mod2);
65     hashsump[i+1] = ((hashsump[i] + ((num[i] *
66     pot3[num[i]]) % mod3)) % mod3);
67 }
68 for(int i = 1; i < n; i++){
69     bhashsumk[i+1] = ((bhashsumk[i] + ((bnum[i] *
70     pot[bnum[i]]) % mod)) % mod);
71     bhashsuml[i+1] = ((bhashsuml[i] + ((bnum[i] *
72     pot2[bnum[i]]) % mod2)) % mod2);
73     bhashsump[i+1] = ((bhashsump[i] + ((bnum[i] *
74     pot3[bnum[i]]) % mod3)) % mod3);
75 }
76 return 0;
77 }

```

3 Geometria

3.1 Convex Hull

```

1 #include <bits/stdc++.h>
2 #define endl '\n'
3 using namespace std;
4
5 struct Point{
6     //alterar tipos se preciso
7     long long int x, y;
8     Point(long long int x, long long int y){
9         this->x = x;
10        this->y = y;
11    }
12    Point operator+(Point o){ return Point(x + o.x, y
13    + o.y); }
14    Point operator-(Point o){ return Point(x - o.x, y
15    - o.y); }
16    Point operator*(long long int k){ return Point(k*
17    x, k*y); }

```

```

15 double len(){ return hypot(x, y); }
16 long long int cross(Point o){ return ((x * o.y) -
17 (y*o.x)); }
18 bool operator<(Point o){ return (tie(x, y) < tie(o
19 .x, o.y)); }
20 bool operator==(Point o){ return (tie(x, y) ==
21 tie(o.x, o.y)); }
22 };
23
24 int orientation(Point a, Point b, Point c){
25     Point ab = b - a;
26     Point bc = c - b;
27     long long int v = ab.cross(bc);
28
29     if(v < 0){
30         return -1; //horario
31     }
32     if(v > 0){
33         return 1; //antihorario
34     }
35     return 0;
36 }
37
38 bool cw(Point a, Point b, Point c, bool
39 include_collinear){
40     int o = orientation(a, b, c);
41     return ((o < 0) || (include_collinear && (o == 0)
42 ));
43 }
44
45 bool ccw(Point a, Point b, Point c, bool
46 include_collinear){
47     int o = orientation(a, b, c);
48     return ((o > 0) || (include_collinear && (o == 0)
49 ));
50 }
51
52 void convex_hull(vector<Point> &a, bool
53 include_collinear=false){
54     if(a.size() == 1){
55         return;
56     }
57     sort(a.begin(), a.end());
58     Point p1 = a[0];
59     Point p2 = a.back();
60     vector<Point> up, down;
61     up.push_back(p1);
62     down.push_back(p2);
63
64     for(int i = 1; i < (int)a.size(); i++){
65         if((i == a.size() - 1) || (cw(p1, a[i], p2,
66 include_collinear))){
67             while((up.size() >= 2) && !(cw(up[up.size
68 () - 2], up[up.size() - 1], a[i],
69 include_collinear))){
70                 up.pop_back();
71             }
72             up.push_back(a[i]);
73         }
74         if((i == a.size() - 1) || (ccw(p1, a[i], p2,
75 include_collinear))){
76             while((down.size() >= 2) && !(ccw(down[
77 down.size() - 2], down[down.size() - 1], a[i],
78 include_collinear))){
79                 down.pop_back();
80             }
81             down.push_back(a[i]);
82         }
83     }
84     if(include_collinear && (up.size() == a.size())){
85         reverse(a.begin(), a.end());
86         return;
87     }
88 }

```

```

74     a.clear();
75     for(int i = 0; i < (int)up.size(); i++){
76         a.push_back(up[i]);
77     }
78     for(int i = down.size() - 2; i > 0; i--){
79         a.push_back(down[i]);
80     }
81 }
82
83 bool insidetriangle(Point a, Point b, Point c, Point
point) {
84     long long int s1 = abs((b-a).cross(c-b));
85     long long int area1 = abs((point - a).cross(point
- b));
86     long long int area2 = abs((point - b).cross(point
- c));
87     long long int area3 = abs((point - c).cross(point
- a));
88     long long int s2 = area1 + area2 + area3;
89     return s1 == s2;
90 }
91
92 bool isinside(vector<Point> &hull, Point p){
93     int n = hull.size();
94     if(n == 1){
95         return (hull.front() == p);
96     }
97     int l = 1;
98     int r = n - 1;
99     int mid;
100    while(abs(r - l) > 1){
101        mid = (r+l)/2;
102        Point tomid = hull[mid] - hull[0];
103        Point topoint = p - hull[0];
104        if((topoint.cross(tomid) < 0){
105            //a esquerda
106            r = mid;
107        } else{
108            l = mid;
109        }
110    }
111    //Point vec = hull[r] - hull[l];
112    //Point tovec = p - hull[l];
113    //return (tovec.cross(vec) > 0);
114    return insidetriangle(hull[0], hull[l], hull[r],
p);
115 }
116
117 int main() {
118     ios_base::sync_with_stdio(false);
119     cin.tie(NULL);
120
121     return 0;
122 }

```

3.2 Ponto

```

1  #include <bits/stdc++.h>
2  #define endl '\n'
3  using namespace std;
4
5  //trocar os long longs por outro tipo desejado, ou:
6  //template<typename T>
7  struct Point{
8      long long int x, y;
9      Point(long long int x, long long int y){
10         this->x = x;
11         this->y = y;
12     }
13     Point operator+(Point o){ return Point(x + o.x, y
+ o.y); }
14     Point operator-(Point o){ return Point(x - o.x, y
- o.y); }

```

```

15     Point operator*(long long int k){ return Point(k*
x, k*y); }
16     double len(){ return hypot(x, y); }
17     long long int cross(Point o){ return ((x * o.y) -
(y*o.x)); }
18     bool operator<(Point o){ return(tie(x, y) < tie(o
.x, o.y)); }
19     bool operator==(Point o){ return (tie(x, y) ==
tie(o.x, o.y)); }
20 };
21
22 int main() {
23     ios_base::sync_with_stdio(false);
24     cin.tie(NULL);
25
26     int t;
27     long long int x1, x2, x3, y1, y2, y3;
28     for(int k = 0; k < t; k++){
29         Point p1 = Point(x2 - x1, y2 - y1);
30         Point p2 = Point(x3 - x1, y3 - y1);
31         //produto vetorial
32         long long int check = p2.cross(p1);
33         if(check == 0){
34             cout << "TOUCH" << endl;
35         } else if(check > 0){
36             cout << "RIGHT" << endl;
37         } else{
38             cout << "LEFT" << endl;
39         }
40     }
41
42     //numerador = abs(((y2 - y) * cx) - ((x2 - x) *
cy) + (x2 * y) - (y2 * x));
43     //denominador = sqrt(((y2 - y) * (y2 - y)) + ((x2
- x) * (x2 - x)));
44     //dist = numerador/denominador;
45     //distancia entre reta formada pelos pontos (x, y
) e (x2, y2) ate o ponto (cx, cy)
46
47     return 0;
48 }

```

4 Buscas e stl

4.1 Subset Sum

```

1  #include <bits/stdc++.h>
2  #define endl '\n'
3  using namespace std;
4
5  void subsetsum(int n, vector<int> &all, vector<long
long int> &atual){
6      if(n == all.size()){
7          //verificacao
8          return;
9      }
10
11     subsetsum(n + 1, all, atual);
12     atual.push_back(all[n]);
13     subsetsum(n + 1, all, atual);
14     atual.pop_back();
15 }
16
17 int main() {
18     ios_base::sync_with_stdio(false);
19     cin.tie(NULL);
20
21     //subset sum com bitmask
22     int n, arr[100];
23     long long int sum;
24     vector<long long int> vec;
25     for (int i = 0; i < (1 << n); i++) {

```

```

26     for (int j = 0; j < n; j++) {
27         if (i & (1 << j)) {
28             sum += arr[j];
29         }
30     }
31     vec.push_back(sum);
32     sum = 0;
33
34 }
35
36
37 return 0;
38 }

```

4.2 Busca Binaria

```

1  #include <bits/stdc++.h>
2  #define endl '\n'
3  using namespace std;
4
5  int main() {
6      ios_base::sync_with_stdio(false);
7      cin.tie(NULL);
8
9      int n, p;
10     int l = 0;
11     int r = n - 1;
12     int mid;
13     bool pos;
14     int maximpossivel = 0;
15     vector<int> sortado;
16     while(r >= l){
17         mid = l + (r - l)/2;
18         pos = true;
19
20         //CHECAGEM
21
22         if(!pos){
23             l = mid + 1;
24             maximpossivel = max(maximpossivel, mid);
25         } else{
26             r = mid - 1;
27         }
28     }
29     upper_bound(sortado.begin(), sortado.end(), p);
30     //PRIMEIRO ELEMENTO >= P
31     lower_bound(sortado.begin(), sortado.end(), p);
32     //PRIMEIRO ELEMENTO > P
33     //subtrair .begin() retorna indice
34     //subtrair upper do lower retorna quantidade
35
36     return 0;
37 }

```

4.3 Kadane

```

1  #include <bits/stdc++.h>
2  #include <iostream>
3  using namespace std;
4
5  int main() {
6      ios_base::sync_with_stdio(false);
7      cin.tie(NULL);
8
9      int len, elemento;
10     long long maxsum, prevsum;
11     int arr[len];
12     prevsum = arr[0];
13     maxsum = arr[0];
14     //maior soma em subarray
15     for (int j = 1; j < len; j++){
16         if((prevsum + arr[j]) < arr[j]){

```

```

17         prevsum = arr[j];
18     } else{
19         prevsum += arr[j];
20     }
21
22     if (prevsum > maxsum){
23         maxsum = prevsum;
24     }
25 }
26 cout << maxsum << endl;
27
28 return 0;
29 }

```

4.4 Stack Monotonica

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  int main() {
6      ios_base::sync_with_stdio(false);
7      cin.tie(NULL);
8
9      int n;
10     vector<long long int> tabuas(n);
11     stack<long long int> monotesq;
12     stack<long long int> monotdir;
13     vector<long long int> mindir(n);
14     vector<long long int> minesq(n);
15
16     for(int i = 0; i < n; i++){
17         while((monotesq.size() > 0) and (tabuas[i] <
18             tabuas[monotesq.top()])){
19             monotesq.pop();
20         }
21
22         if(monotesq.size() > 0){
23             minesq[i] = monotesq.top();
24         } else{
25             minesq[i] = -1;
26         }
27         monotesq.push(i);
28     }
29     //GUARDA MENOR MAIS PROXIMO A ESQUERDA
30
31     for(int i = 1; i <= n; i++){
32         while((monotdir.size() > 0) and (tabuas[n - i]
33             ] <= tabuas[monotdir.top()])){
34             monotdir.pop();
35         }
36
37         if(monotdir.size() > 0){
38             mindir[n - i] = monotdir.top();
39         } else{
40             mindir[n - i] = n;
41         }
42         monotdir.push((n - i));
43     }
44     //VERSAO INVERTIDA
45
46     long long int ar;
47     long long int maxarea = 0;
48     for(int i = 0; i < n; i++){
49         ar = (mindir[i] - minesq[i] - 1) * tabuas[i];
50         maxarea = max(ar, maxarea);
51     }
52     cout << maxarea << endl;
53
54     return 0;
55 }

```

5 Matemática

5.1 Matrizes

```

1 #include <bits/stdc++.h>
2 #define endl '\n'
3 using namespace std;
4
5 vector<vector<long long int>> mult(vector<vector<long
    long int>> &a, vector<vector<long long int>> &b,
    long long int MOD){
6     vector<vector<long long int>> res(a.size(),
    vector<long long int>(b[0].size()));
7
8     for(int i = 0; i < a.size(); i++){
9         for(int j = 0; j < b[0].size(); j++){
10             res[i][j] = 0;
11             for(int k = 0; k < a[0].size(); k++){
12                 res[i][j] += (a[i][k] * b[k][j]) %
    MOD;
13             }
14             res[i][j] = res[i][j] % MOD;
15         }
16     }
17
18     return res;
19 }
20
21 vector<vector<long long int>> fexp(vector<vector<long
    long int>> &a, long long int e, long long int
    MOD){
22     vector<vector<long long int>> ans(4, vector<long
    long int>(4, 0));
23     ans[0][0] = 1;
24     ans[1][1] = 1;
25     ans[2][2] = 1;
26     ans[3][3] = 1;
27
28     while(e){
29         if(e & 1){
30             ans = mult(a, ans, MOD);
31         }
32         a = mult(a, a, MOD);
33         e >>= 1;
34     }
35
36     return ans;
37 }
38
39 int main() {
40     ios_base::sync_with_stdio(false);
41     cin.tie(NULL);
42
43     //recorrencia em matrizes
44     //matrix T * matrix (f0) = matrix (f2)
45     //          (f1)          (f1)
46
47     return 0;
48 }

```

5.2 Fexp E Comuns

```

1 #include <bits/stdc++.h>
2 #define endl '\n'
3 using namespace std;
4
5 const long long int mod = 1000000007;
6
7 long long int fexp(long long int a, long long int b){
8     long long int ans = 1;
9     while(b != 0){
10         if(b & 1){

```

```

11         ans = (ans * a) % mod;
12     }
13     a = (a * a) % mod;
14     b >>= 1;
15 }
16 return ans;
17 }
18
19 long long int gcd(long long int a, long long int b){
20     if(!b){
21         return a;
22     } else{
23         return gcd(b, a % b);
24     }
25     //ja implementado em __gcd()
26 }
27
28 long long int lcm(long long int a, long long int b){
29     return ((a*b)/gcd(a,b));
30 }
31
32 int main() {
33     ios_base::sync_with_stdio(false);
34     cin.tie(NULL);
35
36     return 0;
37 }
38 }

```

5.3 Divisores

```

1 #include <bits/stdc++.h>
2 #define endl '\n'
3 using namespace std;
4
5 int main() {
6     ios_base::sync_with_stdio(false);
7     cin.tie(NULL);
8
9     int n;
10    vector<int> divs;
11    for(int i = 1; (i * i) <= n; i++){
12        if(n % i == 0){
13            divs.push_back(i);
14            if(i != n/i){
15                divs.push_back(n/i);
16            }
17        }
18    }
19
20    return 0;
21 }

```

5.4 Fatoracao Prima E Spf

```

1 #include <bits/stdc++.h>
2 #define endl '\n'
3 #define ll long long int
4 using namespace std;
5
6 int main() {
7     ios_base::sync_with_stdio(false);
8     cin.tie(NULL);
9
10    ll maxn;
11    //SHORTEST PRIME FACTOR
12    vector<ll> spf(maxn+1, 0);
13    ll curr;
14    for(ll i = 2; i <= maxn; i++){
15        if(spf[i] == 0){
16            spf[i] = i;
17            curr = i*i;

```



```

18         while(curr <= maxn){
19             if(spfc[curr] == 0){
20                 spfc[curr] = i;
21             }
22             curr += i;
23         }
24     }
25 }
26
27 //FATORACAO MELHORADA COM SPF -> nlogn
28 ll v;
29 vector<map<ll, int>> fat(maxn+1);
30 for(int i = 2; i <= maxn; i++){
31     v = i;
32     while(v > 1){
33         porfavor[i][spf[v]]++;
34         v = v/spf[v];
35     }
36 }
37
38 //FATORACAO PRIMA PADRAO -> n * sqrt(n)
39 int n;
40 vector<int> primos;
41 map<int, int> freq;
42 for(int i = 2; i*i <= n; i++){
43     int cnt = 0;
44     while(n % i == 0){
45         n /= i;
46         cnt++;
47     }
48     if(cnt > 0){
49         freq[i] += cnt;
50         primos.push_back(i);
51     }
52 }
53 if(n > 1){
54     primos.push_back(n);
55     freq[n]++;
56 }
57
58 return 0;
59 }

```

5.5 Crivo

```

1 #include <bits/stdc++.h>
2 #define endl '\n'
3 using namespace std;
4
5 int main() {
6     ios_base::sync_with_stdio(false);
7     cin.tie(NULL);
8
9     int lim;
10    vector<bool> isprime;
11    isprime[0] = false;
12    isprime[1] = false;
13    vector<int> primes;
14    //crivo base, acha primos
15    for(int i = 2; i < lim; i++){
16        if(isprime[i]){
17            primes.push_back(i);
18            for(int j = i*2; j < lim; j+=i){
19                isprime[j] = false;
20            }
21        }
22    }
23
24    //crivo da soma dos divisores no intervalo
25    vector<int> sumdivisor;
26    for(int i = 1; i < lim; i++){
27        for(int j = i; j < lim; j+= i){
28            sumdivisor[j] += i;

```

```

29        }
30    }
31
32    //crivo da quantidade de divisores dos números no
    intervalo
33    vector<int> numdivisors;
34    for(int i = 1; i < lim; i++){
35        for(int j = i; j < lim; j+= i){
36            numdivisors[j]++;
37        }
38    }
39
40    return 0;
41 }

```

5.6 Combinacao

```

1 #include <bits/stdc++.h>
2 #define endl '\n'
3 using namespace std;
4
5 const long long int mod = 1000000007;
6
7 long long int comb(long long int n, long long int i){
8     long long int denom = 1;
9     long long int num = 1;
10
11     for(int j = 0; j < i; j++){
12         num *= (n - j);
13         num /= (j + 1);
14     }
15     //COMBINACAO ITERATIVA
16     return num;
17 }
18
19 long long int fexp(long long int a, long long int b);
20
21 int main() {
22     ios_base::sync_with_stdio(false);
23     cin.tie(NULL);
24
25     long long int n, m;
26     vector<long long int> fatn(2000100);
27     fatn[0] = 1;
28     for(long long int i = 1; i < fatn.size(); i++){
29         fatn[i] = (fatn[i - 1] * i) % mod;
30     }
31     //combinacao = (n!/((i!*(n-i)))
32     //combinacao com repeticao C(n, i) = C(n + i - 1,
        i);
33     long long int aa = ((fatn[m] * fatn[n - 1])) %
        mod;
34     long long int bb = fexp(aa, mod - 2);
35     long long int combrep = (fatn[n + m - 1] * bb) %
        mod;
36
37     long long int comb = ((fatn[n])/((fatn[n-m] * fatn
        [m])));
38     //para operacoes com modulo eh preciso ao inves
        de dividir, multiplicar
39     //pelo inverso modular (n ^ mod-2)
40
41     return 0;
42 }

```

5.7 Modulo Struct

```

1 #include <bits/stdc++.h>
2 #define ll long long
3 #define endl '\n'
4 using namespace std;
5

```

```

6  const ll mod = 1e9 + 7;
7
8  struct mint{
9      ll val;
10     mint(){
11         this->val = 0;
12     }
13     mint(ll val){
14         this->val = ((val % mod) + mod) % mod;
15     }
16
17     mint operator+(mint v){
18         return (((val + v.val) % mod) + mod) % mod;
19     }
20     mint operator-(mint v){
21         return (((val - v.val) % mod) + mod) % mod;
22     }
23     mint operator*(mint v){
24         return (((val * v.val) % mod) + mod) % mod;
25     }
26     friend ostream& operator<<(ostream& os, const
27     mint& m) {
28         return os << m.val;
29     }
30
31     friend istream& operator>>(istream& is, mint& m)
32     {
33         ll x;
34         is >> x;
35         m = mint(x);
36         return is;
37     }
38 };

```

6 Dp

6.1 Operacoes-bitwise

```

1  #include <bits/stdc++.h>
2  #define endl '\n'
3  using namespace std;
4
5
6  int main() {
7      ios_base::sync_with_stdio(false);
8      cin.tie(NULL);
9
10     //2^n = (1<<n)
11     int n, i, mask;
12     for(int mask = 0; mask < (1<<n); mask++){
13         //iterar pela mask n
14
15         if(mask & (1<<i)); //se bit i for 1
16         mask = mask | (1<<i); //ligar bit i
17         mask = mask ^ (1<<i); //flipar bit i
18
19         return 0;
20 }

```

6.2 Digit Dp

```

1  #include <bits/stdc++.h>
2  #define endl '\n'
3  using namespace std;
4
5  //dp para numeros sem 2 digitos adjacentes iguais
6  long long int rec(int pos, int lastdig, bool start,
7  bool smaller, vector<int> &num, vector<vector<
8  vector<vector<long long int>>>> &dp){
9      if(pos >= num.size()){
10         return 1;

```

```

9      }
10
11     if(dp[pos][lastdig][start][smaller] != -1){
12         return dp[pos][lastdig][start][smaller];
13     }
14
15     long long int ans = 0;
16     if(smaller){
17         for(int i = 0; i <= 9; i++){
18             if(start){
19                 if(i > 0){
20                     ans += rec(pos + 1, i, false,
21                     smaller, num, dp);
22                 } else{
23                     ans += rec(pos + 1, i, true,
24                     smaller, num, dp);
25                 }
26             } else{
27                 if(i != lastdig){
28                     ans += rec(pos + 1, i, start,
29                     smaller, num, dp);
30                 }
31             }
32         }
33     } else{
34         for(int i = 0; i <= num[pos]; i++){
35             if(start){
36                 if(i > 0){
37                     ans += rec(pos + 1, i, false, !(i
38                     == num[pos]), num, dp);
39                 } else{
40                     ans += rec(pos + 1, i, true, !(i
41                     == num[pos]), num, dp);
42                 }
43             } else{
44                 if(i != lastdig){
45                     ans += rec(pos + 1, i, start, !(i
46                     == num[pos]), num, dp);
47                 }
48             }
49         }
50     }
51     dp[pos][lastdig][start][smaller] = ans;
52     return dp[pos][lastdig][start][smaller];
53 }
54
55 int main() {
56     ios_base::sync_with_stdio(false);
57     cin.tie(NULL);
58
59     //digit dp: iterar pelos digitos
60     //lembrar da ideia base: quando chegar na
61     //posiçãõ
62     //nãõ pode mudar o valor!!
63
64     return 0;
65 }

```

6.3 Knapsack 2d

```

1  #include <bits/stdc++.h>
2  #define endl '\n'
3  using namespace std;
4
5  long long int knapsack(vector<tuple<long long int,
6  long long int>> &itens, vector<vector<long long
7  int>> &dp, long long int w, int i){
8      if(w == 0){
9         return 0;
10     }
11     if(i >= itens.size()){
12         return 0;

```

```

11     }
12     if(dp[w][i] != -1){
13         return dp[w][i];
14     }
15
16     long long int ans = knapsack(itens, dp, w, i + 1)
17     ;
18     if(get<0>(itens[i]) <= w){
19         ans = max(ans, (get<1>(itens[i]) + knapsack(
20     itens, dp, (w - get<0>(itens[i])), i + 1)));
21     }
22     dp[w][i] = ans;
23     return ans;
24 }
25
26 int main() {
27     ios_base::sync_with_stdio(false);
28     cin.tie(NULL);
29
30     return 0;
31 }

```

6.4 Knapsack 1d

```

1 #include <bits/stdc++.h>
2 #define endl '\n'
3 using namespace std;
4
5 long long int knap(map<int, int> &custo, vector<long
6     long int> &dp, int w){
7     if(w == 0){
8         return 0;
9     }
10    if(dp[w] != -1){
11        return dp[w];
12    }
13
14    long long int ans = 0;
15    for(auto i : custo){
16        if((w - i.first) >= 0){
17            ans = max(ans, (knap(custo, dp, (w - i.
18    first))) + i.second);
19        }
20    }
21    dp[w] = ans;
22    return dp[w];
23 }
24
25 int main() {
26     ios_base::sync_with_stdio(false);
27     cin.tie(NULL);
28
29     //mesmos principios da digit, lembre-se do kongey
30     donk
31     //estados, mudan as, dp[i][j] -> dp[i- 1][j], dp
32     [i - 1][j + 1], dp[i - 1][j - 1]
33
34     return 0;
35 }

```

6.5 Lcis

```

1 #include <bits/stdc++.h>
2 #define endl '\n'
3 using namespace std;
4
5 int main() {
6     ios_base::sync_with_stdio(false);
7     cin.tie(NULL);
8

```

```

9     int n, pos;
10    cin >> n;
11    vector<int> v(n);
12    for(int i = 0; i < n; i++){
13        cin >> v[i];
14    }
15    vector<int> aux;
16    aux.push_back(v[0]);
17    for(int i = 1; i < n; i++){
18        pos = upper_bound(aux.begin(), aux.end(), v[i
19    ]) - aux.begin();
20        if(pos >= aux.size()){
21            if(v[i] != aux[pos - 1]){
22                aux.push_back(v[i]);
23            }
24        } else{
25            if(v[i] != aux[pos - 1]){
26                aux[pos] = v[i];
27            }
28        }
29    }
30    cout << aux.size() << endl;
31
32    return 0;
33 }

```

6.6 Moedas

```

1 #include <bits/stdc++.h>
2 #define endl '\n'
3 using namespace std;
4
5 //dp das moedas
6 long long int coinsum(vector<long long int> &dp,
7     vector<long long int> &moedas, long long int w){
8     if(w == 0){
9         return 0;
10    }
11    if(dp[w] != -1){
12        return dp[w];
13    }
14
15    long long int ans = INT_MAX;
16    for(int i = 0; i < moedas.size(); i++){
17        if(w - moedas[i] >= 0){
18            ans = min(ans, coinsum(dp, moedas, w -
19    moedas[i] + 1);
20        }
21    }
22    dp[w] = ans;
23    return dp[w];
24 }
25
26 int main() {
27     ios_base::sync_with_stdio(false);
28     cin.tie(NULL);
29
30     return 0;
31 }

```

6.7 Lcs

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 const int MAXN = 5*1e3 + 5;
5 int memo[MAXN][MAXN];
6
7 string s, t;
8
9 inline int LCS(int i, int j){
10    if(i == s.size() || j == t.size()) return 0;

```

```

11     if(memo[i][j] != -1) return memo[i][j];
12
13     if(s[i] == t[j]) return memo[i][j] = 1 + LCS(i+1,
14         j+1);
15
16     return memo[i][j] = max(LCS(i+1, j), LCS(i, j+1));
17 }
18
19 int LCS_It(){
20     for(int i=s.size()-1; i>=0; i--)
21         for(int j=t.size()-1; j>=0; j--)
22             if(s[i] == t[j])
23                 memo[i][j] = 1 + memo[i+1][j+1];
24             else
25                 memo[i][j] = max( memo[i+1][j], memo[
26                     i][j+1] );
27
28     return memo[0][0];
29 }
30
31 string RecoverLCS(int i, int j){
32     if(i == s.size() || j == t.size()) return "";
33
34     if(s[i] == t[j]) return s[i] + RecoverLCS(i+1, j
35         +1);
36
37     if(memo[i+1][j] > memo[i][j+1]) return RecoverLCS
38         (i+1, j);
39
40     return RecoverLCS(i, j+1);
41 }
42
43 //creditos para SamuelH12
44 /*****
45 LCS - Longest Common Subsequence
46
47 Complexity: O(N^2)
48
49 * Recursive:
50 memset(memo, -1, sizeof memo);
51 LCS(0, 0);
52
53 * Iterative:
54 LCS_It();
55
56 * RecoverLCS
57 Complexity: O(N)
58 Recover one of all the possible LCS
59 *****/

```

6.8 Tree Matching

```

1 #include <bits/stdc++.h>
2 #define endl '\n'
3 using namespace std;
4
5 void tmaxu(int n, int p, vector<vector<long long int
6     >> &dp, vector<vector<int>> &adj){
7     long long int sum1 = 0;
8     long long int sum2 = 0;
9
10    for(auto v : adj[n]){
11        if(v != p){
12            tmaxu(v, n, dp, adj);
13            sum1 += max(dp[0][v], dp[1][v]);
14        }
15    }
16    dp[0][n] = sum1;
17
18    for(auto v : adj[n]){
19        if(v != p){
20            sum2 = max(sum2, (1 + dp[0][v] + dp[0][n]
21                - max(dp[0][v], dp[1][v])));

```

```

20    }
21    }
22    dp[1][n] = sum2;
23 }
24
25 long long int tmatch(int n, vector<vector<long long
26     int>> &dp, vector<vector<int>> &adj){
27     tmaxu(n, -1, dp, adj);
28     return max(dp[0][1], dp[1][1]);
29 }
30
31 int main() {
32     ios_base::sync_with_stdio(false);
33     cin.tie(NULL);
34
35     return 0;

```

7 Grafos

7.1 Bicolorabilidade

```

1 #include <bits/stdc++.h>
2 #define endl '\n'
3 using namespace std;
4
5 bool dfs(int v, vector<vector<int>> &adj, vector<bool>
6     > &visitado, vector<bool> &cor){
7     visitado[v] = true;
8
9     for(auto u : adj[v]){
10        if(!visitado[u]){
11            cor[u] = !cor[v];
12            if(!dfs(u, adj, visitado, cor)){
13                return false;
14            }
15        } else if(cor[u] == cor[v]){
16            return false;
17        }
18    }
19    return true;
20 }
21
22 void dfs2(int v, long long int dist, vector<bool> &
23     cor, vector<bool> &visitado, vector<vector<tuple<
24     int, long long int>>> &adj){
25     visitado[v] = true;
26     for(auto u : adj[v]){
27        if(!visitado[get<0>(u)]){
28            if((dist + get<1>(u)) % 2 == 0){
29                cor[get<0>(u)] = cor[1];
30            } else{
31                cor[get<0>(u)] = !cor[1];
32            }
33            dfs2(get<0>(u), dist + get<1>(u), cor,
34                visitado, adj);
35        }
36    }
37 }
38
39 int main() {
40     ios_base::sync_with_stdio(false);
41     cin.tie(NULL);
42
43     return 0;

```

7.2 Prim

```

1 #include <bits/stdc++.h>
2 #define endl '\n'

```

```

3 using namespace std;
4
5 int main() {
6     ios_base::sync_with_stdio(false);
7     cin.tie(NULL);
8
9     int n, m, a, b;
10    long long int c;
11    vector<vector<tuple<long long int, int>>> adj(n +
12    1);
13    vector<int> na_arvore;
14    vector<bool> visitado(n + 1, false);
15    priority_queue<tuple<long long int, int>> minheap
16    ;
17    for(int i = 0; i < m; i++){
18        adj[a].push_back(tuple(-(c), b));
19        adj[b].push_back(tuple(-(c), a));
20    }
21    minheap.push(tuple(0, 1));
22    long long int custo, d;
23    int o;
24    custo = 0;
25    while(minheap.size() > 0){
26        d = get<0>(minheap.top());
27        o = get<1>(minheap.top());
28        minheap.pop();
29
30        if(!visitado[o]){
31            visitado[o] = true;
32            na_arvore.push_back(o);
33            custo += -(d);
34
35            for(auto v : adj[o]){
36                if(!visitado[get<1>(v)]){
37                    minheap.push(v);
38                }
39            }
40
41            //OBS: KRUSKALL
42            // sort arestas
43            // for custo u, v em arestas:
44            // if(find(u) != find(v)):
45            //     join(u, v)
46            //     total = total + custo
47            //prim expande uma arvore, kruskall cria e vai
48            // juntando
49
50    return 0;
51 }

```

7.3 Dfs

```

1 #include <bits/stdc++.h>
2 #define endl '\n'
3 using namespace std;
4
5 void dfs(int n, vector<vector<int>> &adj, vector<bool>
6 &vis){
7     vis[n] = true;
8     for(auto i : adj[n]){
9         if(!vis[i]){
10            dfs(i, adj, vis);
11        }
12    }
13    return;
14 }
15
16 int main() {
17     ios_base::sync_with_stdio(false);
18     cin.tie(NULL);

```

```

19
20     return 0;
21 }

```

7.4 Bfs

```

1 #include <bits/stdc++.h>
2 #define endl '\n'
3 using namespace std;
4
5 int main() {
6     ios_base::sync_with_stdio(false);
7     cin.tie(NULL);
8
9     int n, m;
10    string s;
11    tuple<int, int, int> start;
12    char c;
13    vector<vector<char>> mapa(n);
14    queue<tuple<int, int, int>> fila;
15    vector<vector<bool>> visitado(n, vector<bool>(m,
16    false));
17
18    visitado[get<0>(start)][get<1>(start)] = true;
19    fila.push(start);
20    tuple<int, int> end;
21    int x, y, camada, dist;
22    dist = 0;
23    while(fila.size() > 0){
24        y = get<0>(fila.front());
25        x = get<1>(fila.front());
26        camada = get<2>(fila.front());
27        fila.pop();
28
29        if(mapa[y][x] == 'B'){
30            dist = camada;
31            end = tuple(y, x);
32            break;
33        }
34
35        if((y + 1 < n)){
36            if(!visitado[y + 1][x]){
37                visitado[y + 1][x] = true;
38                fila.push(tuple(y + 1, x, camada + 1)
39            );
40            }
41
42            if((y - 1 >= 0)){
43                if(!visitado[y - 1][x]){
44                    visitado[y - 1][x] = true;
45                    fila.push(tuple(y - 1, x, camada + 1)
46                );
47            }
48
49            if((x + 1 < m)){
50                if(!visitado[y][x + 1]){
51                    visitado[y][x + 1] = true;
52                    fila.push(tuple(y, x + 1, camada + 1)
53                );
54            }
55
56            if((x - 1 >= 0)){
57                if(!visitado[y][x - 1]){
58                    visitado[y][x - 1] = true;
59                    fila.push(tuple(y, x - 1, camada + 1)
60                );
61            }
62        }

```

```

63 //DIAMETRO DA ARVORE:
64 // ACHAR PONTO U MAIS DISTANTE DE INICIAL
65 // ACHAR PONTO V MAIS DISTANTE DE U
66 // DIAMETRO SERA U, V
67
68 //LEMBRAR DE MULTISOURCE
69
70 return 0;
71 }

```

7.5 Achar Ciclos

```

1 #include <bits/stdc++.h>
2 #define endl '\n'
3 using namespace std;
4
5 int dfs(bool &cicblock, int &cicstart, int cidade,
6 int anterior, vector<int> &ciclo, vector<bool> &
7 visitado, vector<vector<int>> &adj){
8     if(visitado[cidade]){
9         if(cicstart == 0){
10             ciclo.push_back(cidade);
11             cicstart = cidade;
12         }
13         return cidade;
14     } else{
15         int fim = 0;
16         visitado[cidade] = true;
17
18         for(auto i : adj[cidade]){
19             if(i != anterior){
20                 fim = dfs(cicblock, cicstart, i,
21 cidade, ciclo, visitado, adj);
22
23                 if(cidade == cicstart){
24                     //ciclo.push_back(cidade);
25                     cicblock = true;
26                 }
27
28                 if(fim != -1){
29                     if(!cicblock){
30                         ciclo.push_back(cidade);
31                         return cidade;
32                     }
33                 }
34             }
35         }
36
37         return -1;
38     }
39 }
40
41 int main() {
42     ios_base::sync_with_stdio(false);
43     cin.tie(NULL);
44
45     //def dfs(atual, anterior):
46     //     if(visitado[atual]): return
47     //     visitado[atual] = true
48     //     for nxt in adj[atual]:
49     //         if(nxt != anterior):
50     //             fim = dfs(nxt, atual)
51     //             if(fim != -1): ciclo.adiciona(atual)
52     //             if(fim == atual OU fim ==
53 JA_TERMINOU): retorne JA_TERMINOU
54     //     retorne -1
55
56     return 0;
57 }

```

7.6 Toposort

```

1 #include <bits/stdc++.h>
2 #define endl '\n'
3 using namespace std;
4
5 int main() {
6     ios_base::sync_with_stdio(false);
7     cin.tie(NULL);
8
9     int n, m, a, b;
10    vector<int> grau_entrada(n + 1, 0);
11    vector<vector<int>> requisitos(n + 1);
12    for(int i = 0; i < m; i++){
13        cin >> a;
14        cin >> b;
15        requisitos[a].push_back(b);
16        grau_entrada[b] += 1;
17    }
18
19    queue<int> fila;
20    for(int i = 1; i <= n; i++){
21        if(grau_entrada[i] == 0){
22            fila.push(i);
23        }
24    }
25
26    vector<int> toposort;
27    int u;
28    while(fila.size() > 0){
29        u = fila.front();
30        fila.pop();
31
32        toposort.push_back(u);
33        for(auto v : requisitos[u]){
34            grau_entrada[v]--;
35            if(grau_entrada[v] == 0){
36                fila.push(v);
37            }
38        }
39    }
40
41    if(toposort.size() == n){
42        for(int i = 0; i < n; i++){
43            cout << toposort[i] << " ";
44        }
45        cout << endl;
46    } else{
47        cout << "IMPOSSIBLE" << endl;
48    }
49
50    return 0;
51 }
52 }

```

7.7 Euler Tour E Lca

```

1 #include <bits/stdc++.h>
2 #define endl '\n'
3 using namespace std;
4
5 int tempo = 0;
6
7 void euler(int v, vector<vector<long long int>> &adj,
8 vector<bool> &vis, vector<long long int> &
9 euler_in, vector<long long int> &euler_out){
10     vis[v] = true;
11     tempo++;
12     euler_in[v] = tempo;
13     //euler tour
14     for(auto u : adj[v]){
15         if(!vis[u]){
16             euler(u, adj, vis, euler_in, euler_out);
17         }
18     }
19 }

```

```

17 //tempo++; -> desnecessario
18 euler_out[v] = tempo;
19 }
20
21 int lca(int a, int b, vector<long long int> &euler_in,
22         vector<long long int> &euler_out, vector<vector<
23         <long long int>> &pai){
24     if((euler_in[a] <= euler_in[b]) and (euler_out[a]
25         >= euler_out[b])){
26         return a;
27     } else if((euler_in[b] <= euler_in[a]) and (
28         euler_out[b] >= euler_out[a])){
29         return b;
30     } else{
31         int cursor = a;
32         for(int i = 20; i >= 0; i--){
33             if(!((euler_in[pai[cursor]][i] <=
34                 euler_in[b]) and (euler_out[pai[cursor]][i] >=
35                 euler_out[b]))){
36                 cursor = pai[cursor][i];
37             }
38         }
39         return pai[cursor][0];
40     }
41 }
42
43 int main(){
44     ios_base::sync_with_stdio(false);
45     cin.tie(NULL);
46
47     int n, q, p, a, b;
48     //AJUSTAR VALOR DE ITERAÇÃO DE ACORDO COM LOGN
49     cin >> n >> q;
50     vector<vector<long long int>> pai(n+1, vector<
51     long long int>(21, -1));
52     vector<long long int> euler_in(n+1);
53     vector<long long int> euler_out(n+1);
54     vector<vector<long long int>> adj(n+1);
55     vector<bool> vis(n+1, false);
56     pai[1][0] = 1;
57     for(int i = 2; i <= n; i++){
58         cin >> p;
59         pai[i][0] = p;
60         adj[p].push_back(i);
61         adj[i].push_back(p);
62     }
63     //precalcular pais
64     for(int j = 1; j <= 20; j++){
65         for(int i = 1; i <= n; i++){
66             if(pai[i][j-1] != -1){
67                 pai[i][j] = pai[pai[i][j-1]][j-1];
68             }
69         }
70     }
71     return 0;
72 }

```

7.8 Dijkstra

```

1 #include <bits/stdc++.h>
2 #define endl '\n'
3 using namespace std;
4
5 int main() {
6     ios_base::sync_with_stdio(false);
7     cin.tie(NULL);
8
9     int n, m;
10    vector<long long int> distancias(n + 1, LLONG_MAX
11    );
12    vector<vector<tuple<long long int, int>>> adj(n +
13    1);

```

```

12    distancias[1] = 0;
13    priority_queue<tuple<long long int, int>> minheap
14    ;
15    minheap.push(tuple(0, 1));
16    long long int d, c;
17    int a, b, o;
18    for(int i = 0; i < m; i++){
19        cin >> a;
20        cin >> b;
21        cin >> c;
22
23        adj[a].push_back(tuple(c, b));
24    }
25    while(minheap.size() > 0){
26        d = -(get<0>(minheap.top()));
27        o = get<1>(minheap.top());
28        minheap.pop();
29
30        if(d <= distancias[o]){
31            for(auto v : adj[o]){
32                if(distancias[get<1>(v)] > distancias
33                [o] + get<0>(v)){
34                    distancias[get<1>(v)] =
35                    distancias[o] + get<0>(v);
36                    minheap.push(tuple(-(distancias[
37                    get<1>(v)]), get<1>(v)));
38                }
39            }
40        }
41    }
42    // LEMBRE DE GRAPH MODELLING
43    // ADICIONAR ESTADOS COMO EM DP
44
45    return 0;
46 }

```

7.9 Dinic

```

1 #include <bits/stdc++.h>
2 #define endl '\n'
3 #define ll long long int
4 using namespace std;
5
6 struct aresta{
7     ll u, v, c;
8     aresta(){
9         u = 0;
10        v = 0;
11        c = 0;
12    }
13    aresta(ll u2, ll v2, ll c2){
14        u = u2;
15        v = v2;
16        c = c2;
17    }
18 };
19
20 struct Dinic{
21     ll n, src, sink;
22     //vetor all permite achar real e inversa
23     facilmente
24     vector<aresta> all;
25     vector<vector<int>> adj;
26     vector<ll> level;
27     //proxima aresta disponivel
28     vector<int> nxt;
29
30     Dinic(ll n2, ll src2, ll sink2){
31         n = n2;
32         src = src2;
33         sink = sink2;
34         adj = vector<vector<int>>(n+1);
35     }

```

```

35 void add(ll u, ll v, ll c, bool dir){
36     //no dinic sao necessarios arestas opostas de
37     capacidade 0
38     all.push_back(aresta(u, v, c));
39     adj[u].push_back(all.size()-1);
40     //se for nao direcionado trocar 0 por c
41     ll invcap = 0;
42     if(dir){
43         invcap = c;
44     }
45     all.push_back(aresta(v, u, invcap));
46     adj[v].push_back(all.size()-1);
47 }
48
49 //bfs para gerar o vetor lvl por meio de arestas
50 //nao saturadas
51 //deve sinalizar se chegou na pia (caso contrario
52 //o algoritmo terminou)
53 bool bfs(){
54     level = vector<ll>(n+1, -1);
55     queue<tuple<int, ll>> fila;
56     level[src] = 0;
57     fila.push(tuple<int, ll>(src, 0));
58
59     while(!fila.empty()){
60         int curr = get<0>(fila.front());
61         ll d = get<1>(fila.front());
62         fila.pop();
63
64         for(auto e : adj[curr]){
65             if((all[e].c > 0) && (level[all[e].v]
66             == -1)){
67                 level[all[e].v] = d+1;
68                 fila.push(tuple<int, ll>(all[e].v,
69                 d+1));
70             }
71         }
72     }
73
74     return (level[sink] != -1);
75 }
76
77 ll dfs(int curr, ll bottleneck){
78     if((curr == sink) or (bottleneck == 0)){
79         return bottleneck;
80     }
81
82     //atualiza prox aresta, lembrar de resetar
83     for(int &curledge = nxt[curr]; curledge < (
84     adj[curr].size()); curledge++){
85         int ar = adj[curr][curledge];
86         int nv = all[ar].v;
87         //apenas arestas que progridem
88         if((level[nv] == (level[curr] + 1))){
89             ll check = dfs(nv, min(bottleneck,
90             all[ar].c));
91             if(check > 0){
92                 all[ar].c -= check;
93                 //flipar o ultimo bit checa
94                 inverso
95                 all[ar ^ 1].c += check;
96                 return check;
97             }
98         }
99     }
100     //blocking flow
101     return 0;
102 }
103
104 ll maxflow(){
105     ll flow = 0;
106     while(bfs()){
107         ll flow += dfs(src, 1e12);
108     }
109 }
110
111 mincut = maxflow
112 apos ultima bfs, level[i] == -1 se esta do lado da
113 fonte
114
115 emparelhamento max:
116 criar src e sink, direcionar arestas src->lado 1,
117 lado 2-> sink
118 todas arestas tem peso 1
119 emp = maxflow
120
121 cobertura minima do bipartido = emparelhamento
122 grids sao bipartidos
123
124 ind - max independent set
125 cob(n) = n - ind(n)
126
127 em um grafo com arestas unitarias:
128 maxflow = quantidade de caminhos aresta-disjuntos s-t
129 cap(corte minimo) = maximo de arestas removidos que
130 quebra os caminhos de s-t
131
132 caminhos vertice-disjuntos = maxflow onde vertices
133 sao separados em v1 e v2 com
134 aresta unitaria de v1-v2, v1 onde entra e v2 onde sai
135
136 fechamento maximo: reduzir para min cut
137 lado da fonte: o que adquiriu
138 lado do ralo: o que ignorou
139
140 se valor real i >= 0 : fonte -> i (valor)
141 else i -> ralo (-valor)
142
143 dependencias: se precisa de A pra ter B, aresta: B ->
144 A (infinito)
145
146 */

```

7.10 Dsu

```

1 #include <bits/stdc++.h>
2 #define endl '\n'
3 using namespace std;
4
5 int find(int n, vector<int> &rep){
6     if(n == rep[n]){
7         return rep[n];
8     } else{
9         rep[n] = find(rep[n], rep);
10        return rep[n];
11    }
12 }
13
14 void join(int n, int v, vector<int> &rep, map<int,
15 int> &size){
16     n = find(n, rep);
17     v = find(v, rep);

```



```

18     if(n == v){
19         return;
20     }
21
22     if(size[n] < size[v]){
23         swap(v, n);
24     }
25
26     rep[v] = n;
27     size[n] += size[v];
28 }
29
30 int main() {
31     ios_base::sync_with_stdio(false);
32     cin.tie(NULL);
33
34     int n, m;
35     map<int, int> size;
36     vector<int> rep(n + 1);
37     for(int i = 1; i <= n; i++){
38         rep[i] = i;
39         size[i]++;
40     }
41
42     return 0;
43 }

```

7.11 Pontes

```

1 #include <bits/stdc++.h>
2 #define endl '\n'
3 #define ll long long int
4 using namespace std;
5
6 int t = 0;
7 vector<ll> pontes;
8
9 void dfs(int v, int p, vector<vector<tuple<ll, int>>>
    &adj, vector<bool> &vis, vector<int> &in, vector
    <int> &low){
10     vis[v] = true;
11     t++;
12     in[v] = t;
13     low[v] = t;
14
15     bool skip = false;
16     for(auto [id, u] : adj[v]){
17         if(u==p && !skip){
18             skip = true;
19         } else{
20             if(vis[u]){
21                 low[v] = min(low[v], in[u]);
22             } else{
23                 dfs(u, v, adj, vis, in, low);
24                 low[v] = min(low[v], low[u]);
25                 if(low[u] > in[v]){
26                     pontes.push_back(id);
27                 }
28             }
29         }
30     }
31 }

```

7.12 Floyd Warshall

```

1 #include <bits/stdc++.h>
2 #define endl '\n'
3 using namespace std;
4
5 int main() {
6     ios_base::sync_with_stdio(false);
7     cin.tie(NULL);

```

```

8
9     int n, m, a, b;
10     long long int c;
11     vector<vector<long long int>> distancias(n + 1,
12     vector<long long int>(n + 1, LLONG_MAX));
13     for(int i = 0; i < m; i++){
14         cin >> a;
15         cin >> b;
16         cin >> c;
17         // lembrar arestas duplas
18         distancias[a][b] = min(c, distancias[a][b]);
19         distancias[b][a] = min(c, distancias[b][a]);
20     }
21
22     for(int k = 1; k <= n; k++){
23         for(int u = 1; u <= n; u++){
24             for(int v = 1; v <= n; v++){
25                 if((distancias[u][k] != LLONG_MAX)
26                 and (distancias[k][v] != LLONG_MAX)){
27                     distancias[u][v] = min(distancias
28                     [u][v], (distancias[u][k] + distancias[k][v]));
29                 }
30             }
31         }
32     }
33
34     return 0;
35 }

```

7.13 Scc

```

1 #include <bits/stdc++.h>
2 #define endl '\n'
3 #define ll long long int
4 using namespace std;
5
6 void dfs(int v, vector<vector<int>> &adj, vector<bool>
    &vis, vector<int> &order){
7     vis[v] = true;
8     for(auto u : adj[v]){
9         if(!vis[u]){
10             dfs(u, adj, vis, order);
11         }
12     }
13     order.push_back(v);
14
15     return;
16 }
17
18 void kosaraju(vector<vector<int>> &adj, vector<int> &
    c, vector<vector<int>> &comps, vector<vector<int>
    >> &dag){
19     int n = adj.size();
20     vector<int> order;
21     vector<bool> vis(n, false);
22
23     for(int i = 1; i < n; i++){
24         if(!vis[i]){
25             dfs(i, adj, vis, order);
26         }
27     }
28
29     vector<vector<int>> adj_transp(n);
30     for(int v = 1; v < n; v++){
31         for(int u : adj[v]){
32             adj_transp[u].push_back(v);
33         }
34     }
35     vis = vector<bool>(n, false);

```

```

36     reverse(order.begin(), order.end());
37
38     for(auto v : order){
39         if(!vis[v]){
40             vector<int> comp;
41             dfs(v, adj_transp, vis, comp);
42             comps.push_back(comp);
43         }
44     }
45
46     vector<bool> mark(comps.size(), false);
47     for(int i = 0; i < comps.size(); i++){
48         for(auto v : comps[i]){
49             c[v] = i;
50         }
51     }
52
53     for(int i = 1; i < n; i++){
54         for(auto v : adj[i]){
55             if(c[i]==c[v] || mark[c[v]]) continue;
56
57             mark[c[v]] = true;
58             dag[c[i]].push_back(c[v]);
59         }
60         for(auto v : adj[i]) mark[c[v]] = false;
61     }
62 }
63
64 int main(){
65     ios_base::sync_with_stdio(false);
66     cin.tie(NULL);
67
68     //2SAT
69     //i = x
70     //i^1 = ~x
71     if(!pos){
72         cout << "IMPOSSIVEL" << endl;
73     } else{
74

```

```

75         vector<bool> val(m+1);
76         for(int i = 1; i <= 2*m + 1; i+=2){
77             val[i/2] = co[i] < co[i^1];
78         }
79         for(int i = 1; i <= m; i++){
80             if(val[i]){
81                 cout << "S ";
82             } else{
83                 cout << "N ";
84             }
85         }
86         cout << endl;
87     }
88
89     /*
90     void add_or(int u, int v){
91         E[eval(~u)].push_back(eval(v));
92         E[eval(~v)].push_back(eval(u));
93     }
94     void add_nand(int u, int v) {
95         E[eval(u)].push_back(eval(~v));
96         E[eval(v)].push_back(eval(~u));
97     }
98     void set_true (int u){ E[eval(~u)].push_back(eval
99         (u)); }
100     void set_false(int u){ set_true(~u); }
101     void add_imply(int u, int v){ E[eval(u)].
102         push_back(eval(v)); }
103     void add_and (int u, int v){ set_true(u);
104         set_true(v); }
105     void add_nor (int u, int v){ add_and(~u, ~v); }
106     void add_xor (int u, int v){ add_or(u, v);
107         add_nand(u, v); }
108     void add_xnor (int u, int v){ add_xor(u, ~v); }
109     */
110
111     return 0;
112 }

```