

Competitive Programming Notebook

Contents

1	Grafos	2
1.1	Prim	2
1.2	Dijkstra	2
1.3	Floyd Warshall	2
1.4	Bicolorabilidade	3
1.5	Bfs	3
1.6	Dsu	3
1.7	Toposort	4
1.8	Dfs	4
1.9	Achar Ciclos	4
2	Buscas e stl	5
2.1	Stack Monotonica	5
2.2	Busca Binaria	5
2.3	Subset Sum	5
2.4	Kadane	6
3	String	6
3.1	3 Hash	6
3.2	Trie	6
3.3	Hash Sem Ordem	7
4	Matematica	7
4.1	Fatoracao Prima	7
4.2	Divisores	7
4.3	Combinacao	8
4.4	Fexp E Comuns	8
4.5	Crivo	8
4.6	Soma De Digitos	9
4.7	Matrizes	9
5	Range queries	9
5.1	Prefix Sum 2d	9
5.2	Segment Tree Comprimida	9
5.3	Segment Tree Base	10
6	Dp	10
6.1	Digit Dp	10
6.2	Knapsack 2d	11
6.3	Operacoes-bitwise	11
6.4	Knapsack 1d	11
6.5	Moedas	12
6.6	Lcis	12
6.7	Lcs	12

1 Grafos

1.1 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 + 1);
12    vector<int> na_arvore;
13    vector<bool> visitado(n + 1, false);
14    priority_queue<tuple<long long int, int>> minheap;
15
16    for(int i = 0; i < m; i++){
17        adj[a].push_back(tuple(-(c), b));
18        adj[b].push_back(tuple(-(c), a));
19    }
20    minheap.push(tuple(0, 1));
21    long long int custo, d;
22    int o;
23    custo = 0;
24    while(minheap.size() > 0){
25        d = get<0>(minheap.top());
26        o = get<1>(minheap.top());
27        minheap.pop();
28
29        if(!visitado[o]){
30            visitado[o] = true;
31            na_arvore.push_back(o);
32            custo += -(d);
33
34            for(auto v : adj[o]){
35                if(!visitado[get<1>(v)]){
36                    minheap.push(v);
37                }
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 juntando
48
49    return 0;
50 }

```

1.2 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
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29

```

```

11    vector<vector<tuple<long long int, int>>> adj(n + 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[o] + get<0>(v)){
33                    distancias[get<1>(v)] = distancias[o] + get<0>(v);
34                    minheap.push(tuple(-(distancias[get<1>(v)]), get<1>(v)));
35                }
36            }
37        }
38        // LEMBRE DE GRAPH MODELLING
39        // ADICIONAR ESTADOS COMO EM DP
40
41    return 0;
42 }

```

1.3 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, vector<long long int>(n + 1, LLONG_MAX));
12    for(int i = 0; i < m; i++){
13        cin >> a;
14        cin >> b;
15        cin >> c;
16
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) and (distancias[k][v] != LLONG_MAX)){
26                    distancias[u][v] = min(distancias[u][v], (distancias[u][k] + distancias[k][v]));
27                }
28            }
29        }
30    }
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50

```

```

30 // analisar quantidade, serve pra poucos
31 // lrtices e saber distância entre todas
32
33
34 return 0;
35 }

```

1.4 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>
> &visitado, vector<bool> &cor){
6     visitado[v] = true;
7
8     for(auto u : adj[v]){
9         if(!visitado[u]){
10             cor[u] = !cor[v];
11             if(!dfs(u, adj, visitado, cor)){
12                 return false;
13             }
14         } else if(cor[u] == cor[v]){
15             return false;
16         }
17     }
18     return true;
19 }
20
21 void dfs2(int v, long long int dist, vector<bool> &
cor, vector<bool> &visitado, vector<vector<tuple<
int, long long int>>> &adj){
22     visitado[v] = true;
23     for(auto u : adj[v]){
24         if(!visitado[get<0>(u)]){
25             if((dist + get<1>(u)) % 2 == 0){
26                 cor[get<0>(u)] = cor[1];
27             } else{
28                 cor[get<0>(u)] = !cor[1];
29             }
30             dfs2(get<0>(u), dist + get<1>(u), cor,
visitado, adj);
31         }
32     }
33 }
34
35 int main() {
36     ios_base::sync_with_stdio(false);
37     cin.tie(NULL);
38
39     return 0;
40 }

```

1.5 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,
false));

```

```

16
17    visitado[get<0>(start)][get<1>(start)] = true;
18    fila.push(start);
19    tuple<int, int> end;
20    int x, y, camada, dist;
21    dist = 0;
22    while(fila.size() > 0){
23        y = get<0>(fila.front());
24        x = get<1>(fila.front());
25        camada = get<2>(fila.front());
26        fila.pop();
27
28        if(mapa[y][x] == 'B'){
29            dist = camada;
30            end = tuple(y, x);
31            break;
32        }
33
34        if((y + 1 < n)){
35            if(!visitado[y + 1][x]){
36                visitado[y + 1][x] = true;
37                fila.push(tuple(y + 1, x, camada + 1)
);
38            }
39        }
40
41        if((y - 1 >= 0)){
42            if(!visitado[y - 1][x]){
43                visitado[y - 1][x] = true;
44                fila.push(tuple(y - 1, x, camada + 1)
);
45            }
46        }
47
48        if((x + 1 < m)){
49            if(!visitado[y][x + 1]){
50                visitado[y][x + 1] = true;
51                fila.push(tuple(y, x + 1, camada + 1)
);
52            }
53        }
54
55        if((x - 1 >= 0)){
56            if(!visitado[y][x - 1]){
57                visitado[y][x - 1] = true;
58                fila.push(tuple(y, x - 1, camada + 1)
);
59            }
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 }

```

1.6 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
19     if(n == v){
20         return;
21     }
22
23     if(size[n] < size[v]){
24         swap(v, n);
25     }
26
27     rep[v] = n;
28     size[n] += size[v];
29 }
30
31 int main() {
32     ios_base::sync_with_stdio(false);
33     cin.tie(NULL);
34
35     int n, m;
36     map<int, int> size;
37     vector<int> rep(n + 1);
38     for(int i = 1; i <= n; i++){
39         rep[i] = i;
40         size[i]++;
41     }
42
43     return 0;

```

1.7 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    if(toposort.size() == n){
41        for(int i = 0; i < n; i++){
42            cout << toposort[i] << " ";
43        }
44        cout << endl;
45    } else{
46        cout << "IMPOSSIBLE" << endl;
47    }
48
49    return 0;
50 }

```

1.8 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
14     return;
15 }
16
17 int main() {
18     ios_base::sync_with_stdio(false);
19     cin.tie(NULL);
20
21     return 0;

```

1.9 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){

```

```

26         if(!cicblock){
27             ciclo.push_back(cidade);
28             return cidade;
29         }
30     }
31 }
32 }
33
34     return -1;
35 }
36 }
37
38 int main() {
39     ios_base::sync_with_stdio(false);
40     cin.tie(NULL);
41
42     //def dfs(atual, anterior):
43     //  if(visitado[atual]): return
44     //  visitado[atual] = true
45     //  for nxt in adj[atual]:
46     //      if(nxt != anterior):
47     //          fim = dfs(nxt, atual)
48     //          if(fim != -1): ciclo.adiciona(atual)
49     //          if(fim == atual OU fim ==
43     JA_TERMINOU): retorne JA_TERMINOU
44     //
50     //  retorne -1
51
52     return 0;
53 }
54 }

```

2 Buscas e stl

2.1 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){

```

```

36         mindir[n - i] = monotdir.top();
37     } else{
38         mindir[n - i] = n;
39     }
40     monotdir.push((n - i));
41 }
42 //VERSAO INVERTIDA
43
44 long long int ar;
45 long long int maxarea = 0;
46 for(int i = 0; i < n; i++){
47     ar = (mindir[i] - minesq[i] - 1) * tabuas[i];
48     maxarea = max(ar, maxarea);
49 }
50 cout << maxarea << endl;
51
52 return 0;
53 }
54 }

```

2.2 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)/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 }

```

2.3 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
6     long int> &atual){
7     if(n == all.size()){
8         //verificacao
9         return;
10    }

```

```

10 subsetsum(n + 1, all, atual);
11 atual.push_back(all[n]);
12 subsetsum(n + 1, all, atual);
13 atual.pop_back();
14 }
15
16 int main() {
17     ios_base::sync_with_stdio(false);
18     cin.tie(NULL);
19
20     //subset sum com bitmask
21     int n, arr[100];
22     long long int sum;
23     vector<long long int> vec;
24     for (int i = 0; i < (1 << n); i++) {
25         for (int j = 0; j < n; j++) {
26             if (i & (1 << j)) {
27                 sum += arr[j];
28             }
29         }
30         vec.push_back(sum);
31         sum = 0;
32     }
33
34     return 0;
35 }
36
37
38 }

```

2.4 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 }

```

3 String

3.1 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
47    return 0;
48 }

```

3.2 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 }

```

3.3 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] * l) % 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 }

```

4 Matematica

4.1 Fatoracao Prima

```

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> primos;
11     map<int, int> freq;
12     for(int i = 2; i*i <= n; i++){
13         int cnt = 0;
14         while(n % i == 0){
15             n /= i;
16             cnt++;
17         }
18         if(cnt > 0){
19             freq[i] += cnt;
20             primos.push_back(i);
21         }
22     }
23
24     return 0;
25 }

```

4.2 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 }

```

4.3 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 }

```

4.4 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 / (gcd(a, b) * b));
30 }
31
32 int main() {
33     ios_base::sync_with_stdio(false);
34     cin.tie(NULL);
35
36     return 0;
37 }

```

4.5 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++){
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 nÃºmeros 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     return 0;
40 }
41 }

```

4.6 Soma De Digitos

```

1  #include <bits/stdc++.h>
2  #define endl '\n'
3  using namespace std;
4
5  long long int digsum(long long int num){
6      string s = to_string(num);
7      long long int sum = 0;
8      for(int i = 0; i < s.size(); i++){
9          sum += s[i] - '0';
10     }
11     return sum;
12 }
13
14 int digsum2(int n) {
15     while(n>=10) {
16         int temp = 0;
17         while (n > 0) {
18             temp += n % 10;
19             n /= 10;
20         }
21         n = temp;
22     }
23     return n;
24 }

```

4.7 Matrices

```

1  #include <bits/stdc++.h>
2  #define endl '\n'
3  using namespace std;
4
5  vector<vector<long long int>> mult(vector<vector<long
6      long int>> &a, vector<vector<long long int>> &b,
7      long long int MOD){
8      vector<vector<long long int>> res(a.size(),
9      vector<long long int>(b[0].size()));
10
11     for(int i = 0; i < a.size(); i++){
12         for(int j = 0; j < b[0].size(); j++){
13             res[i][j] = 0;
14             for(int k = 0; k < a[0].size(); k++){
15                 res[i][j] += (a[i][k] * b[k][j]) %
16                 MOD;
17             }
18             res[i][j] = res[i][j] % MOD;
19         }
20     }
21     return res;
22 }
23
24 vector<vector<long long int>> fexp(vector<vector<long
25     long int>> &a, long long int e, long long int
26     MOD){
27     vector<vector<long long int>> ans(4, vector<long
28     long int>(4, 0));
29     ans[0][0] = 1;
30     ans[1][1] = 1;
31     ans[2][2] = 1;
32     ans[3][3] = 1;
33
34     while(e){
35         if(e & 1){
36             ans = mult(a, ans, MOD);
37         }
38         a = mult(a, a, MOD);
39         e /= 2;
40     }
41     return ans;
42 }

```

```

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 Range queries

5.1 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 }

```

5.2 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 }

```

```

14     int mid = (l + r)/2;
15     update(2* no, l, mid, pos, val, segtree);
16     update((2* no) + 1, mid + 1, r, pos, val, segtree);
17     segtree[no] = segtree[2 * no] + segtree[(2 * no) + 1];
18 }
19
20 int query(int no, int l, int r, int lq, int rq,
21 vector<int> &segtree){
22     if((rq < l) or (r < lq)){
23         return 0;
24     }
25     if((lq <= l) and (r <= rq)){
26         return segtree[no];
27     }
28
29     int mid = (l + r)/2;
30     int ans = query(2 * no, l, mid, lq, rq, segtree);
31     ans += query((2 * no) + 1, mid + 1, r, lq, rq, segtree);
32
33     return ans;
34 }
35
36 int main(){
37     ios_base::sync_with_stdio(false);
38     cin.tie(NULL);
39
40     int n, q, sal, k, j;
41     char c;
42     vector<int> all;
43     vector<int> orig;
44     vector<tuple<char, int, int>> queries;
45     sort(all.begin(), all.end());
46     all.erase(unique(all.begin(), all.end()), all.end());
47     int range;
48     vector<int> segtree(4 * (all.size()), 0);
49     for(int i = 0; i < orig.size(); i++){
50         range = lower_bound(all.begin(), all.end(), orig[i]) - all.begin();
51         update(1, 1, all.size(), range, 1, segtree);
52     }
53
54     return 0;
55 }

```

```

20 }
21 if(l == r){
22     segtree[no] = val;
23     return;
24 }
25
26 int mid = (l + r)/2;
27 update(2* no, l, mid, pos, val, segtree);
28 update((2* no) + 1, mid + 1, r, pos, val, segtree);
29 segtree[no] = segtree[2 * no] + segtree[(2 * no) + 1];
30
31 }
32
33 long long int query(int no, int l, int r, int lq, int rq, vector<long long int> &segtree){
34     if((rq < l) or (r < lq)){
35         return 0;
36     }
37     if((lq <= l) and (r <= rq)){
38         return segtree[no];
39     }
40
41     int mid = (l + r)/2;
42     long long int ans = query(2 * no, l, mid, lq, rq, segtree);
43     ans += query((2 * no) + 1, mid + 1, r, lq, rq, segtree);
44
45     return ans;
46 }
47
48 int main(){
49     ios_base::sync_with_stdio(false);
50     cin.tie(NULL);
51     int n;
52     vector<long long int> segtree(4 * n);
53     //ADICIONAR X EM L E -X EM R+1 ÃŁ IGUAL A
54     ADICIONAR X EM [L, R]
55     //SEGTREE PARA ACHAR MENORES:
56     // for(int i = 0; i < n; i++){
57     //     range = upper_bound(mansort.begin(), mansort.end(), man[i]) - mansort.begin();
58     //     dir[i] = query(1, 1, n, range, n, segtree);
59     //     update(1, 1, n, range, 1, segtree);
60     // }
61
62     return 0;
63 }

```

5.3 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> &segtree, vector<long long int> &orig){
6     if(l == r){
7         segtree[no] = orig[l];
8         return;
9     }
10
11     int mid = (l + r)/2;
12     build(2 * no, l, mid, segtree, orig);
13     build((2 * no) + 1, mid + 1, r, segtree, orig);
14     segtree[no] = segtree[2 * no] + segtree[(2 * no) + 1];
15 }
16
17 void update(int no, int l, int r, int pos, long long int val, vector<long long int> &segtree){
18     if((pos < l) or (r < pos)){
19         return;

```

6 Dp

6.1 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, bool smaller, vector<int> &num, vector<vector<vector<long long int>>>> &dp){
7     if(pos >= num.size()){
8         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,
smaller, num, dp);
21             } else{
22                 ans += rec(pos + 1, i, true,
smaller, num, dp);
23             }
24         } else{
25             if(i != lastdig){
26                 ans += rec(pos + 1, i, start,
smaller, num, dp);
27             }
28         }
29     }
30 } else{
31     for(int i = 0; i <= num[pos]; i++){
32         if(start){
33             if(i > 0){
34                 ans += rec(pos + 1, i, false, !(i
== num[pos]), num, dp);
35             } else{
36                 ans += rec(pos + 1, i, true, !(i
== num[pos]), num, dp);
37             }
38         } else{
39             if(i != lastdig){
40                 ans += rec(pos + 1, i, start, !(i
== num[pos]), num, dp);
41             }
42         }
43     }
44 }
45 dp[pos][lastdig][start][smaller] = ans;
46 return dp[pos][lastdig][start][smaller];
47 }
48 }
49
50 int main() {
51     ios_base::sync_with_stdio(false);
52     cin.tie(NULL);
53
54     //digit dp: iterar pelos digitos
55     //lembrar da ideia base: quando chegar na
posiÃ§Ã£o
56     //nÃ£o pode mudar o valor!!
57
58     return 0;
59 }

```

6.2 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,
long long int>> &itens, vector<vector<long long
int>> &dp, long long int w, int i){
6     if(w == 0){
7         return 0;
8     }
9     if(i >= itens.size()){
10         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 if(get<0>(itens[i]) <= w){
18     ans = max(ans, (get<1>(itens[i]) + knapsack(
itens, dp, (w - get<0>(itens[i])), i + 1)));
19 }
20 dp[w][i] = ans;
21 return ans;
22 }
23
24 int main() {
25     ios_base::sync_with_stdio(false);
26     cin.tie(NULL);
27
28     return 0;
29 }

```

6.3 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 }

```

6.4 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
long int> &dp, int w){
6     if(w == 0){
7         return 0;
8     }
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.
first))) + i.second);
18        }
19    }
20    dp[w] = ans;
21    return dp[w];
22 }
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 26
    donk
30 //estados, mudan as, dp[i][j] -> dp[i- 1][j], dp 27
    [i - 1][j + 1], dp[i - 1][j - 1] 28
31 29 cout << aux.size() << endl;
32 return 0; 30
33 } 31 return 0;
    32 }

```

6.5 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    long long int ans = INT_MAX;
15    for(int i = 0; i < moedas.size(); i++){
16        if(w - moedas[i] >= 0){
17            ans = min(ans, coinsum(dp, moedas, w -
18 moedas[i]) + 1);
19        }
20    }
21    dp[w] = ans;
22    return dp[w];
23 }
24 int main() {
25     ios_base::sync_with_stdio(false);
26     cin.tie(NULL);
27
28     return 0;
29 }

```

6.6 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];

```

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

```