



Notebook - Maratona de Programação

Py tá $O(N)$

Contents

1	Strings	2
1.1	Kmp	2
1.2	Z-function	2
2	Math	2
2.1	Sieve Of Eratosthenes	2
3	ED	2
3.1	Dsu	2
4	Grafos	2
4.1	Kruskall	2
4.2	Dijkstra	3
4.3	Bellman Ford	3
4.4	Floyd Warshall	3
4.5	Lca	3
5	Template	4
5.1	Template	4
6	Algoritmos	4
6.1	Binary Search Last True	4
6.2	Kadane	4
6.3	Binary Exponentiation	4
6.4	Delta-encoding	4
6.5	Binary Search First True	4

1 Strings

1.1 Kmp

```
1 vector<int> prefix_function(string s) {
2     int n = (int)s.length();
3     vector<int> pi(n);
4     for (int i = 1; i < n; i++) {
5         int j = pi[i-1];
6         while (j > 0 && s[i] != s[j])
7             j = pi[j-1];
8         if (s[i] == s[j])
9             j++;
10        pi[i] = j;
11    }
12    return pi;
13 }
```

1.2 Z-function

```
1 vector<int> z_function(string s) {
2     int n = (int) s.length();
3     vector<int> z(n);
4     for (int i = 1, l = 0, r = 0; i < n; ++i) {
5         if (i <= r)
6             z[i] = min(r - i + 1, z[i - l]);
7         while (i + z[i] < n && s[z[i]] == s[i + z[i]
8         ])
9             ++z[i];
10        if (i + z[i] - 1 > r)
11            l = i, r = i + z[i] - 1;
12    }
13    return z;
14 }
```

2 Math

2.1 Sieve Of Eratosthenes

```
1 int n;
2 vector<bool> is_prime(n+1, true);
3 is_prime[0] = is_prime[1] = false;
4 for (int i = 2; i <= n; i++) {
5     if (is_prime[i] && (long long)i * i <= n) {
6         for (int j = i * i; j <= n; j += i)
7             is_prime[j] = false;
8     }
9 }
```

3 ED

3.1 Dsu

```
1 #include <bits/stdc++.h>
2
3 using namespace std;
4
5 const int MAX = 1e6+17;
6
7 struct DSU {
8     int n;
9     vector<int> link, sizes;
10
11     DSU(int n) {
12         this->n = n;
13         link.assign(n+1, 0);
14         sizes.assign(n+1, 1);
15
16         for (int i = 0; i <= n; i++)
```

```
17         link[i] = i;
18     }
19
20     int find(int x) {
21         while (x != link[x])
22             x = link[x];
23
24         return x;
25     }
26
27     bool same(int a, int b) {
28         return find(a) == find(b);
29     }
30
31     void unite(int a, int b) {
32         a = find(a);
33         b = find(b);
34
35         if (a == b) return;
36
37         if (sizes[a] < sizes[b])
38             swap(a, b);
39
40         sizes[a] += sizes[b];
41         link[b] = a;
42     }
43
44     int size(int x) {
45         return sizes[x];
46     }
47 };
48
49 int main() {
50     ios::sync_with_stdio(false);
51     cin.tie(NULL);
52
53     int cities, roads; cin >> cities >> roads;
54     vector<int> final_roads;
55     int ans = 0;
56     DSU dsu = DSU(cities);
57     for (int i = 0, a, b; i < roads; i++) {
58         cin >> a >> b;
59         dsu.unite(a, b);
60     }
61
62     for (int i = 2; i <= cities; i++) {
63         if (!dsu.same(1, i)) {
64             ans++;
65             final_roads.push_back(i);
66             dsu.unite(1, i);
67         }
68     }
69
70     cout << ans << '\n';
71     for (auto e : final_roads) {
72         cout << "1 " << e << '\n';
73     }
74
75 }
```

4 Grafos

4.1 Kruskall

```
1 vector<int> parent, rank;
2
3 void make_set(int v) {
4     parent[v] = v;
5     rank[v] = 0;
6 }
7
8 int find_set(int v) {
```

```

9     if (v == parent[v])
10         return v;
11     return parent[v] = find_set(parent[v]);
12 }
13
14 void union_sets(int a, int b) {
15     a = find_set(a);
16     b = find_set(b);
17     if (a != b) {
18         if (rank[a] < rank[b])
19             swap(a, b);
20         parent[b] = a;
21         if (rank[a] == rank[b])
22             rank[a]++;
23     }
24 }
25
26 struct Edge {
27     int u, v, weight;
28     bool operator<(const Edge &other) {
29         return weight < other.weight;
30     }
31 };
32
33 int n;
34 vector<Edge> edges;
35
36 int cost = 0;
37 vector<Edge> result;
38 parent.resize(n);
39 rank.resize(n);
40 for (int i = 0; i < n; i++)
41     make_set(i);
42
43 sort(edges.begin(), edges.end());
44
45 for (Edge e : edges) {
46     if (find_set(e.u) != find_set(e.v)) {
47         cost += e.weight;
48         result.push_back(e);
49         union_sets(e.u, e.v);
50     }
51 }

```

4.2 Dijkstra

```

1 const int INF = 1000000000;
2 vector<vector<pair<int, int>>> adj;
3
4 void dijkstra(int s, vector<int> &d, vector<int> &p) {
5     int n = adj.size();
6     d.assign(n, INF);
7     p.assign(n, -1);
8
9     d[s] = 0;
10    set<pair<int, int>> q;
11    q.insert({0, s});
12    while (!q.empty()) {
13        int v = q.begin()->second;
14        q.erase(q.begin());
15
16        for (auto edge : adj[v]) {
17            int to = edge.first;
18            int len = edge.second;
19
20            if (d[v] + len < d[to]) {
21                q.erase({d[to], to});
22                d[to] = d[v] + len;
23                p[to] = v;
24                q.insert({d[to], to});
25            }
26        }
27    }
28 }

```

```

27     }
28 }

```

4.3 Bellman Ford

```

1 struct edge
2 {
3     int a, b, cost;
4 };
5
6 int n, m, v;
7 vector<edge> e;
8 const int INF = 1000000000;
9
10 void solve()
11 {
12     vector<int> d (n, INF);
13     d[v] = 0;
14     for (int i=0; i<n-1; ++i)
15         for (int j=0; j<m; ++j)
16             if (d[e[j].a] < INF)
17                 d[e[j].b] = min (d[e[j].b], d[e[j].a]
18                                     + e[j].cost);
19 }

```

4.4 Floyd Warshall

```

1 for (int k = 0; k < n; ++k) {
2     for (int i = 0; i < n; ++i) {
3         for (int j = 0; j < n; ++j) {
4             if (d[i][k] < INF && d[k][j] < INF)
5                 d[i][j] = min(d[i][j], d[i][k] + d[k]
6                                 ][j]);
7         }
8     }
9 }

```

4.5 Lca

```

1 int n, l;
2 vector<vector<int>> adj;
3
4 int timer;
5 vector<int> tin, tout;
6 vector<vector<int>> up;
7
8 void dfs(int v, int p)
9 {
10     tin[v] = ++timer;
11     up[v][0] = p;
12     for (int i = 1; i <= l; ++i)
13         up[v][i] = up[up[v][i-1]][i-1];
14
15     for (int u : adj[v]) {
16         if (u != p)
17             dfs(u, v);
18     }
19
20     tout[v] = ++timer;
21 }
22
23 bool is_ancestor(int u, int v)
24 {
25     return tin[u] <= tin[v] && tout[u] >= tout[v];
26 }
27
28 int lca(int u, int v)
29 {
30     if (is_ancestor(u, v))
31         return u;
32     if (is_ancestor(v, u))
33         return v;

```

```

34     for (int i = 1; i >= 0; --i) {
35         if (!is_ancestor(up[u][i], v))
36             u = up[u][i];
37     }
38     return up[u][0];
39 }
40
41 void preprocess(int root) {
42     tin.resize(n);
43     tout.resize(n);
44     timer = 0;
45     l = ceil(log2(n));
46     up.assign(n, vector<int>(l + 1));
47     dfs(root, root);
48 }

```

5 Template

5.1 Template

```

1 #include<bits/stdc++.h>
2
3 using namespace std;
4
5 const int MAX = 2e5+17;
6
7 int main() {
8     ios::sync_with_stdio(false);
9     cin.tie(NULL);
10
11
12     return 0;
13 }
14 }

```

6 Algoritmos

6.1 Binary Search Last True

```

1 int last_true(int lo, int hi, function<bool(int)> f)
2 {
3     lo--;
4     while (lo < hi) {
5         int mid = lo + (hi - lo + 1) / 2;
6         if (f(mid)) {
7             lo = mid;
8         } else {
9             hi = mid - 1;
10        }
11    }
12    return lo;
13 }

```

6.2 Kadane

```

1 int ans = a[0], ans_l = 0, ans_r = 0;
2 int sum = 0, minus_pos = -1;
3
4 for (int r = 0; r < n; ++r) {
5     sum += a[r];
6     if (sum > ans) {
7         ans = sum;
8         ans_l = minus_pos + 1;
9         ans_r = r;
10    }

```

```

11    if (sum < 0) {
12        sum = 0;
13        minus_pos = r;
14    }
15 }

```

6.3 Binary Exponentiation

```

1 long long power(long long a, long long b) {
2     long long res = 1;
3     while (b > 0) {
4         if (b & 1)
5             res = res * a;
6         a = a * a;
7         b >>= 1;
8     }
9     return res;
10 }

```

6.4 Delta-encoding

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 int main(){
5     int n, q;
6     cin >> n >> q;
7     int [n];
8     int delta[n+2];
9
10    while(q--){
11        int l, r, x;
12        cin >> l >> r >> x;
13        delta[l] += x;
14        delta[r+1] -= x;
15    }
16
17    int curr = 0;
18    for(int i=0; i < n; i++){
19        curr += delta[i];
20        v[i] = curr;
21    }
22
23    for(int i=0; i < n; i++){
24        cout << v[i] << ' ';
25    }
26    cout << '\n';
27
28    return 0;
29 }

```

6.5 Binary Search First True

```

1 int first_true(int lo, int hi, function<bool(int)> f)
2 {
3     hi++;
4     while (lo < hi) {
5         int mid = lo + (hi - lo) / 2;
6         if (f(mid)) {
7             hi = mid;
8         } else {
9             lo = mid + 1;
10        }
11    }
12    return lo;

```