



Notebook - Maratona de Programação

Estomazil? Jesus Baiano Surgiu!

Contents

1	Árvore	2
1.1	Lca Binarylifting	2
1.2	Fenwicktree	2
2	DP	3
2.1	Lis	3
3	Template	3
3.1	Template	3
4	EDs	3
4.1	Seglazy	3
4.2	Ordered Set	3
4.3	Dsu	4
5	Grafo	4
5.1	Floydwarshal	4
5.2	Dijkstra	4
5.3	Dinic	5
5.4	Finding Bridges	5

1 Árvore

1.1 Lca Binarylifting

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 #define sws ios_base::sync_with_stdio(false);cout.tie
  (NULL);cin.tie(NULL);
4 #define mp make_pair
5 #define pb push_back
6 #define rep(i, a, b) for (int i = a; i < b; i++)
7 #define dbg(msg,x) cout<<msg<<" "<<x<<endl;
8 #define output(x) for(auto c:x){cout<<c<<" ";}cout<<"
  ";
9 #define ll long long
10 #define ff first
11 #define ss second
12 #define pq priority_queue
13 typedef vector<int> vi;
14 typedef vector<bool> vb;
15 typedef pair<int, int> pii;
16 typedef vector<pair<int,int> > vpp;
17 const int MAXN=10;
18
19 int n,l;
20 int timer=0;
21 vi adj[MAXN];
22 int tin[MAXN],tout[MAXN];
23 vector<vi>up;
24
25 void dfs(int v,int p){
26
27     tin[v]=timer++;
28     dbg("v",v)
29     dbg("tin",tin[v]);
30     cout<<endl;
31     up[v][0]=p;
32
33     for(int i=1;i<=l;i++){
34         up[v][i]=up[up[v][i-1]][i-1];
35     }
36
37     for(auto c: adj[v]){
38         if(c!=p)dfs(c,v);
39     }
40
41     tout[v]= timer++;
42     dbg("v",v)
43     dbg("tout",tout[v]);
44     cout<<endl;
45
46 }
47
48 bool is_ancestor(int u,int v){
49     return (tin[u]<=tin[v] && tout[u]>=tout[v]);
50 }
51
52
53 int lca(int u,int v){
54     if(is_ancestor(u,v))return u;
55     if(is_ancestor(v,u))return v;
56     for(int i=l;i>=0;--i){
57         if(!is_ancestor(up[u][i],v)){
58             u=up[u][i];
59         }
60     }
61
62     return up[u][0];
63 }
64
65 void preprocess(int root){
66     timer=0;
```

```
67     l= ceil(log2(n));
68     up.assign(n, vector<int>(l + 1));
69     dfs(root,root);
70 }
71
72 void edge(int a,int b){
73     adj[a].pb(b);
74     adj[b].pb(a);
75 }
76
77 int main(){
78     sws
79     n=10;
80     adj[1].pb(7);
81     adj[7].pb(1);
82
83     adj[2].pb(7);
84     adj[7].pb(2);
85
86     adj[6].pb(7);
87     adj[7].pb(6);
88
89     adj[6].pb(5);
90     adj[5].pb(6);
91
92     adj[6].pb(3);
93     adj[3].pb(6);
94
95     adj[1].pb(9);
96     adj[9].pb(1);
97
98     adj[9].pb(4);
99     adj[4].pb(9);
100
101     adj[4].pb(8);
102     adj[8].pb(4);
103
104     // edge(1,2);
105     // edge(2,3);
106     // edge(3,4);
107     // edge(4,5);
108     // edge(5,6);
109     // edge(6,7);
110     // edge(8,7);
111
112
113     rep(i,1,10){
114         cout<<i<<" ";
115         for(auto c: adj[i])cout<<c<<" ";
116         cout<<endl;
117     }
118
119     preprocess(1);
120
121     dbg("lca",lca(8,5));
122 }
```

1.2 Fenwicktree

```
1 const int MAXN = 100000;
2 int t[MAXN +1];
3 int n;
4
5 void build(int a[]) {
6     copy(a,a+n+1,t);
7
8     for(int i=1; i<n+1; i++) {
9         int p = i + (i&-i);
10        if(p<n) {
11            t[p] += t[i];
12        }
13    }
14 }
```

```

15
16 void update(int pos, int newValue) {
17     while(pos<n) {
18         t[pos] += newValue;
19         pos += (pos&-pos);
20     }
21 }
22
23 int sum(int a) {
24     int ans=0;
25     while(a>0) {
26         ans += t[a];
27         a -= (a&-a);
28     }
29     return ans;
30 }
31
32 int sum(int l, int r) {
33     return sum(r) - sum(l-1);
34 }
35
36
37 /* MAIN
38 * Recebe entrada e
39 * aloca no array a
40 * int delta = newValue - a[pos+1];
41 */

```

2 DP

2.1 Lis

```

1 int lis(vector<int> const& a) {
2     int n = a.size();
3     const int INF = 1e9;
4     vector<int> d(n+1, INF);
5     d[0] = -INF;
6
7     for (int i = 0; i < n; i++) {
8         for (int j = 1; j <= n; j++) {
9             if (d[j-1] < a[i] && a[i] < d[j])
10                 d[j] = a[i];
11         }
12     }
13
14     int ans = 0;
15     for (int i = 0; i <= n; i++) {
16         if (d[i] < INF)
17             ans = i;
18     }
19     return ans;
20 }

```

3 Template

3.1 Template

```

1 #define sws std::ios::sync_with_stdio(false); cin.tie
2 (NULL); cout.tie(NULL);
3 #define input(x) for (auto &it : x) cin >> it;
4 #define output(x) for (auto &it : x) cout << it << '
5 ';
6 #define rep(i, a, b) for (int i = a; i < b; i++)
7 #define dbg(msg, x) cout << msg << " " << x << endl;
8 const double PI = acos(-1);
9 const int INF = 0x3f3f3f3f;
10 const long long LLINF = 0x3f3f3f3f3f3f3f3f;

```

4 EDs

4.1 Seglazy

```

1 const int MAX= 2e5+5;
2
3 vector<ll> lazy(4*MAX,-1);
4 ll tree[4*MAX], numeros[MAX];
5
6 void prop(int l, int r, int no){
7     if(lazy[no] == -1){
8         if(l != r){
9             lazy[2*no] = lazy[no];
10            lazy[2*no+1] = lazy[no];
11        }
12        tree[no] = (l-r+1)*lazy[no];
13        lazy[no] = -1;
14    }
15 }
16
17 void build(int l, int r, int no){
18     if(l == r){
19         tree[no] = numeros[l];
20         return;
21     }
22     int meio = (l+r)/2;
23     build(l,meio,2*no);
24     build(meio+1,r,2*no+1);
25     tree[no] = tree[2*no] + tree[2*no+1];
26 }
27
28 void update(int a, int b, int x, int l, int r, int no
29 ){
30     prop(l, r, no);
31     if(r<a or l > b){
32         return;
33     }
34     if(l>=a and r <=b){
35         lazy[no] = x;
36         prop(l, r, no);
37         return;
38     }
39     int meio = (l+r)/2;
40     update(a,b,x,l,meio,2*no);
41     update(a,b,x,meio+1,r,2*no+1);
42     tree[no] = tree[2*no+1] + tree[2*no];
43 }
44
45 ll querie(int id, int l, int r, int no){
46     prop(l, r, no);
47     if(l == r){
48         return tree[no];
49     }
50     int meio = (l+r)/2;
51     if(id <= meio){
52         return querie(id,l,meio,2*no);
53     }
54     else{
55         return querie(id,meio+1,r,2*no+1);
56     }
57 }

```

4.2 Ordered Set

```

1 #include <bits/extc++.h>
2
3 using namespace __gnu_pbds; // or pb_ds;
4
5 template<typename T, typename B = null_type>
6 using ordered_set = tree<T, B, less<T>, rb_tree_tag,
7     tree_order_statistics_node_update>; //

```

```

        order_of_key(k) : Number of items strictly
        smaller than k.
7 // find_by_order(k) : K-th element in a set (counting
    from zero).

```

4.3 Dsu

```

1 class DSU {
2     vector<int> parent;
3     vector<int> card;
4
5 public:
6     DSU(int n): parent(n+1), card(n+1,1) {
7         for(int i = 1; i <= n; i++)
8             parent[i] = i;
9     }
10
11     /* O(log n) */
12     int find_set(int x) {
13         if(x == parent[x])
14             return x;
15
16         return parent[x] = find_set(parent[x]);
17     }
18
19     bool same_set(int a, int b) {
20         return find_set(a) == find_set(b);
21     }
22
23     /* O(log n) */
24     void join_sets(int a, int b) {
25         a = find_set(a);
26         b = find_set(b);
27
28         if(card[a] < card[b])
29             swap(a,b);
30
31         card[a] += card[b];
32         parent[b] = a;
33     }
34 };

```

5 Grafo

5.1 Floydwarshal

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define sws \
4     ios_base::sync_with_stdio(false); \
5     cout.tie(NULL); \
6     cin.tie(NULL);
7 #define mp make_pair
8 #define pb push_back
9 #define rep(i, a, b) for (int i = a; i < b; i++)
10 #define dbg(msg, x) cout << msg << " " << x << endl;
11 #define output(x) \
12     for (auto c : x) \
13     { \
14         cout << c << " "; \
15     } \
16     cout << " ";
17 #define ff first
18 #define ss second
19 typedef vector<int> vi;
20 typedef pair<int, int> pii;
21 const int MAXN = (int)1e3;
22 vector<pii> adj[MAXN];
23
24 int main()
25 {

```

```

26 sws int n, qtdA;
27 cin >> n >> qtdA;
28 int adj[n + 1][n + 1];
29
30 for (int i = 1; i <= n; i++)
31 {
32     for (int j = 1; j <= n; j++)
33     {
34         adj[i][j] = 0;
35     }
36 }
37 rep(i, 0, qtdA)
38 {
39     int org, dest, w;
40     cin >> org >> dest >> w;
41
42     adj[org][dest] = w;
43     adj[dest][org] = w;
44 }
45
46 int dists[n + 1][n + 1];
47
48 rep(i, 1, n + 1)
49 {
50     rep(j, 1, n + 1)
51     {
52         rep(k, 1, n + 1)
53         {
54             if (i == j)
55             {
56                 dists[i][j] = 0;
57                 continue;
58             }
59             if (adj[i][j])
60             {
61                 dists[i][j] = adj[i][j];
62                 continue;
63             }
64             dists[i][j] = (int)1e5;
65         }
66     }
67 }
68
69 for (int k = 1; k <= n; k++)
70 {
71     for (int i = 1; i <= n; i++)
72     {
73         for (int j = 1; j <= n; j++)
74         {
75             dists[i][j] = min(dists[i][k] + dists
76 [k][j], dists[i][j]);
77         }
78     }
79 }
80
81 for (int i = 1; i <= n; i++)
82 {
83     for (int j = 1; j <= n; j++)
84     {
85         cout << dists[i][j] << " ";
86     }
87 }

```

5.2 Dijkstra

```

1 for(int i = 1; i <= n; i++) distance[i] = INF;
2 distance[x] = 0;
3 q.push({0, x});
4 while(!q.empty()) {
5     int a = q.top().second;
6     q.pop();
7     if(processed[a]) {

```

```

8         continue;
9     }
10    processed[a] = true;
11    for(auto u : adj[a]) {
12        int b = u.first, w = u.second;
13        if(distance[a] + w < distance[b]) {
14            distance[b] = distance[a] + w;
15            q.push({-distance[b], b});
16        }
17    }
18 }

```

5.3 Dinic

```

1  const int N = 500;
2
3  struct Dinic {
4      struct Edge{
5          int from, to; ll flow, cap;
6      };
7      vector<Edge> edge;
8
9      vector<int> g[N];
10     int ne = 0;
11     int lvl[N], vis[N], pass;
12     int qu[N], px[N], qt;
13
14     ll run(int s, int sink, ll minE) {
15         if(s == sink) return minE;
16
17         ll ans = 0;
18
19         for(; px[s] < (int)g[s].size(); px[s]++) {
20             int e = g[s][ px[s] ];
21             auto &v = edge[e], &rev = edge[e^1];
22             if(lvl[v.to] != lvl[s]+1 || v.flow >= v.
cap)
23                 continue;          // v.cap - v.flow
24             < lim
25             ll tmp = run(v.to, sink, min(minE, v.cap - v
.flow));
26             v.flow += tmp, rev.flow -= tmp;
27             ans += tmp, minE -= tmp;
28             if(minE == 0) break;
29         }
30         return ans;
31     }
32     bool bfs(int source, int sink) {
33         qt = 0;
34         qu[qt++] = source;
35         lvl[source] = 1;
36         vis[source] = ++pass;
37         for(int i = 0; i < qt; i++) {
38             int u = qu[i];
39             px[u] = 0;
40             if(u == sink) return true;
41             for(auto& ed : g[u]) {
42                 auto v = edge[ed];
43                 if(v.flow >= v.cap || vis[v.to] ==
pass)
44                     continue; // v.cap - v.flow < lim
45                 vis[v.to] = pass;
46                 lvl[v.to] = lvl[u]+1;
47                 qu[qt++] = v.to;
48             }
49         }
50         return false;
51     }
52     ll flow(int source, int sink) {
53         reset_flow();
54         ll ans = 0;
55         //for(lim = (1LL << 62); lim >= 1; lim /= 2)
while(bfs(source, sink))

```

```

56         ans += run(source, sink, LLINF);
57         return ans;
58     }
59     void addEdge(int u, int v, ll c, ll rc) {
60         Edge e = {u, v, 0, c};
61         edge.pb(e);
62         g[u].push_back(ne++);
63
64         e = {v, u, 0, rc};
65         edge.pb(e);
66         g[v].push_back(ne++);
67     }
68     void reset_flow() {
69         for(int i = 0; i < ne; i++)
70             edge[i].flow = 0;
71         memset(lvl, 0, sizeof(lvl));
72         memset(vis, 0, sizeof(vis));
73         memset(qu, 0, sizeof(qu));
74         memset(px, 0, sizeof(px));
75         qt = 0; pass = 0;
76     }
77     vector<pair<int, int>> cut() {
78         vector<pair<int, int>> cuts;
79         for (auto [from, to, flow, cap]: edge) {
80             if (flow == cap and vis[from] == pass and
vis[to] < pass and cap>0) {
81                 cuts.pb({from, to});
82             }
83         }
84         return cuts;
85     }
86     void ans(int source, int sink, int total){
87         flow(source, sink);
88         for(int i = 0; i < edge.size(); i++){
89             if(edge[i].flow == 1){
90                 matriz[edge[i].from][edge[i].to - total
] = 'X';
91             }
92         }
93     }
94 };

```

5.4 Finding Bridges

```

1  int n; // number of nodes
2  vector<vector<int>> adj; // adjacency list of graph
3
4  vector<bool> visited;
5  vector<int> tin, low;
6  int timer;
7
8  void dfs(int v, int p = -1) {
9      visited[v] = true;
10     tin[v] = low[v] = timer++;
11     for (int to : adj[v]) {
12         if (to == p) continue;
13         if (visited[to]) {
14             low[v] = min(low[v], tin[to]);
15         } else {
16             dfs(to, v);
17             low[v] = min(low[v], low[to]);
18             if (low[to] > tin[v])
19                 IS_BRIDGE(v, to);
20         }
21     }
22 }
23
24 void find_bridges() {
25     timer = 0;
26     visited.assign(n, false);
27     tin.assign(n, -1);
28     low.assign(n, -1);
29     for (int i = 0; i < n; ++i) {

```

```
30     if (!visited[i])
31         dfs(i);
32     }
33 }
```