Notebook - Maratona de Programação

Estomazil? Jesus Baiano Surgiu!

Contents

1 Árvore

1.1 Lca Binarylifting

```
#include <bits/stdc++.h>
using namespace std;
#define sws ios_base::sync_with_stdio(false);cout.tie(NULL);cin.tie(NULL);
#define mp make_pair
#define pb push_back
#define rep(i, a, b) for (int i = a; i < b; i++)
#define dbg(msg,x) cout<<msg<<" "<<x<<endl;
#define output(x) for(auto c:x){cout<<c<<" ";}cout<<"
     ";
#define ll long long
#define ff first
#define ss second
#define pq priority_queue
typedef vector<int> vi;
typedef vector<bool> vb;
typedef pair<int, int> pii;
typedef vector<pair<int,int> > vpp;
const int MAXN=10;

int n,l;
int timer=0;
vi adj[MAXN];
int tin[MAXN],tout[MAXN];
vector<vi>up;

void dfs(int v,int p){

    tin[v]=timer++;
    dbg("v",v)
    dbg("tin",tin[v]);
    cout<<endl;
    up[v][0]=p;

    for(int i=1;i<=l;i++){
        up[v][i]=up[up[v][i-1]][i-1];
    }

    for(auto c: adj[v]){
        if(c!=p)dfs(c,v);
    }

    tout[v]= timer++;
    dbg("v",v)
    dbg("tout",tout[v]);
    cout<<endl;


}

bool is_ancestor(int u,int v){
    return (tin[u]<=tin[v] && tout[u]>=tout[v]);
}

int lca(int u,int v){
    if(is_ancestor(u,v))return u;
    if(is_ancestor(v,u))return v;
    for(int i=l;i>=0;--i){
        if(!is_ancestor(up[u][i],v)){
            u=up[u][i];
        }
    }

    return up[u][0];
}

void preprocess(int root){
    timer=0;
    l= ceil(log2(n));
    up.assign(n, vector<int>(l + 1));
    dfs(root,root);
}

void edge(int a,int b){
    adj[a].pb(b);
    adj[b].pb(a);
}

int main(){
    sws
    n=10;
    adj[1].pb(7);
    adj[7].pb(1);

    adj[2].pb(7);
    adj[7].pb(2);

    adj[6].pb(7);
    adj[7].pb(6);

    adj[6].pb(5);
    adj[5].pb(6);

    adj[6].pb(3);
    adj[3].pb(6);

    adj[1].pb(9);
    adj[9].pb(1);

    adj[9].pb(4);
    adj[4].pb(9);

    adj[4].pb(8);
    adj[8].pb(4);

    // edge(1,2);
    // edge(2,3);
    // edge(3,4);
    // edge(4,5);
    // edge(5,6);
    // edge(6,7);
    // edge(8,7);


    rep(i,1,10){
        cout<<i<<": ";
        for(auto c: adj[i])cout<<c<<" ";
        cout<<endl;
    }

    preprocess(1);

    dbg("lca",lca(8,5));
}
```

1.2 Fenwicktree

```
const int MAXN = 100000;
int t[MAXN +1];
int n;

void build(int a[]) {
    copy(a,a+n+1,t);

    for(int i=1; i<n+1; i++) {
        int p = i + (i&-i);
        if(p<n) {
            t[p] += t[i];
        }
    }
}
```

```

void update(int pos, int newValue) {
    while(pos<n) {
        t[pos] += newValue;
        pos += (pos&-pos);
    }
}

int sum(int a) {
    int ans=0;
    while(a>0) {
        ans += t[a];
        a -= (a&-a);
    }
    return ans;
}

int sum(int l, int r) {
    return sum(r) - sum(l-1);
}


    /*  MAIN
    *   Recebe entrada e
    *   aloca no array a
    *   int delta = newValue-a[pos+1];
    */
```

2 DP

2.1 Lis

```
int lis(vector<int> const& a) {
    int n = a.size();
    const int INF = 1e9;
    vector<int> d(n+1, INF);
    d[0] = -INF;

    for (int i = 0; i < n; i++) {
        for (int j = 1; j <= n; j++) {
            if (d[j-1] < a[i] && a[i] < d[j])
                d[j] = a[i];
        }
    }

    int ans = 0;
    for (int i = 0; i <= n; i++) {
        if (d[i] < INF)
            ans = i;
    }
    return ans;
}
```

3 Template

3.1 Template

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

4 EDs

4.1 Seglazy

```
const int MAX= 2e5+5;

vector<ll> lazy(4*MAX,-1);
ll tree[4*MAX], numeros[MAX];

void prop(int l, int r, int no){
    if(lazy[no] == -1){
        if(l != r){
            lazy[2*no] = lazy[no];
            lazy[2*no+1] = lazy[no];
        }
        tree[no] = (l-r+1)*lazy[no];
        lazy[no] = -1;
    }
}

void build(int l, int r, int no){
    if(l == r){
        tree[no] = numeros[l];
        return;
    }
    int meio = (l+r)/2;
    build(l,meio,2*no);
    build(meio+1,r,2*no+1);
    tree[no] = tree[2*no] + tree[2*no+1];
}

void update(int a, int b, int x, int l, int r, int no){
    prop(l, r, no);
    if(r< a or l > b){
        return;
    }
    if(l>=a and r <=b){
        lazy[no] = x;
        prop(l, r, no);
        return;
    }
    int meio = (l+r)/2;
    update(a,b,x,l,meio,2*no);
    update(a,b,x,meio+1,r,2*no+1);
    tree[no] = tree[2*no+1] + tree[2*no];
}

ll querie(int id, int l, int r, int no){
    prop(l, r, no);
    if(l == r){
        return tree[no];
    }
    int meio = (l+r)/2;
    if(id <= meio){
        return querie(id,l,meio,2*no);
    }
    else{
        return querie(id,meio+1,r,2*no+1);
    }
}
```

4.2 Ordered Set

```
#include <bits/extc++.h>

using namespace __gnu_pbds; // or pb_ds;

template<typename T, typename B = null_type>
using ordered_set = tree<T, B, less<T>, rb_tree_tag, tree_order_statistics_node_update>; //
```

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

4.3 Dsu

```
class DSU {
    vector<int> parent;
    vector<int> card;

public:
    DSU(int n): parent(n+1), card(n+1,1) {
        for(int i = 1; i <= n; i++)
            parent[i] = i;
    }

    /* O(log n) */
    int find_set(int x) {
        if(x == parent[x])
            return x;

        return parent[x] = find_set(parent[x]);
    }

    bool same_set(int a, int b) {
        return find_set(a) == find_set(b);
    }

    /* O(log n) */
    void join_sets(int a, int b) {
        a = find_set(a);
        b = find_set(b);

        if(card[a] < card[b])
            swap(a,b);

        card[a] += card[b];
        parent[b] = a;
    }
};
```

5 Grafo

5.1 Floydwarshal

```
#include <bits/stdc++.h>
using namespace std;
#define sws                           \
    ios_base::sync_with_stdio(false); \
    cout.tie(NULL);                   \
    cin.tie(NULL);
#define mp make_pair
#define pb push_back
#define rep(i, a, b) for (int i = a; i < b; i++)
#define dbg(msg, x) cout << msg << " " << x << endl;
#define output(x)           \
    for (auto c : x)        \
    {                       \
        cout << c << " "; \
    }                       \
    cout << " ";
#define ff first
#define ss second
typedef vector<int> vi;
typedef pair<int, int> pii;
const int MAXN = (int)1e3;
vector<pii> adj[MAXN];

int main()
{
    sws int n, qtdA;
    cin >> n >> qtdA;
    int adj[n + 1][n + 1];

    for (int i = 1; i <= n; i++)
    {
        for (int j = 1; j <= n; j++)
        {
            adj[i][j] = 0;
        }
    }
    rep(i, 0, qtdA)
    {
        int org, dest, w;
        cin >> org >> dest >> w;

        adj[org][dest] = w;
        adj[dest][org] = w;
    }

    int dists[n + 1][n + 1];


    rep(i, 1, n + 1)
    {
        rep(j, 1, n + 1)
        {
            if (i == j)
            {
                dists[i][j] = 0;
                continue;
            }
            if (adj[i][j])
            {
                dists[i][j] = adj[i][j];
                continue;
            }
            dists[i][j] = (int)1e5;
        }
    }


    for (int k = 1; k <= n; k++)
    {
        for (int i = 1; i <= n; i++)
        {
            for (int j = 1; j <= n; j++)
            {
                dists[i][j] = min(dists[i][k] + dists[k][j], dists[i][j]);
            }
        }
    }

    for (int i = 1; i <= n; i++)
    {
        for (int j = 1; j <= n; j++)
        {
            cout << dists[i][j] << " ";
        }
        cout << endl;
    }
}
```

5.2 Dijkstra

```
for(int i = 1; i <= n; i++) distance[i] = INF;
distance[x] = 0;
q.push({0, x});
while(!q.empty()) {
    int a = q.top().second;
    q.pop();
    if(processed[a]) {
        continue;
    }
    processed[a] = true;
    for(auto u : adj[a]) {
        int b = u.first, w = u.second;
        if(distance[a] + w < distance[b]) {
            distance[b] = distance[a] + w;
            q.push({-distance[b], b});
        }
    }
}
```

5.3 Dinic

```
const int N = 500;

struct Dinic {
    struct Edge{
        int from, to; ll flow, cap;
    };
    vector<Edge> edge;

    vector<int> g[N];
    int ne = 0;
    int lvl[N], vis[N], pass;
    int qu[N], px[N], qt;

    ll run(int s, int sink, ll minE) {
        if(s == sink) return minE;

        ll ans = 0;

        for(; px[s] < (int)g[s].size(); px[s]++) {
            int e = g[s][ px[s] ];
            auto &v = edge[e], &rev = edge[e^1];
            if(lvl[v.to] != lvl[s]+1 || v.flow >= v.cap)
                continue;           // v.cap - v.flow < lim
            ll tmp = run(v.to, sink,min(minE, v.cap-v.flow));
            v.flow += tmp, rev.flow -= tmp;
            ans += tmp, minE -= tmp;
            if(minE == 0) break;
        }
        return ans;
    }
    bool bfs(int source, int sink) {
        qt = 0;
        qu[qt++] = source;
        lvl[source] = 1;
        vis[source] = ++pass;
        for(int i = 0; i < qt; i++) {
            int u = qu[i];
            px[u] = 0;
            if(u == sink) return true;
            for(auto& ed : g[u]) {
                auto v = edge[ed];
                if(v.flow >= v.cap || vis[v.to] == pass)
                    continue; // v.cap - v.flow < lim
                vis[v.to] = pass;
                lvl[v.to] = lvl[u]+1;
                qu[qt++] = v.to;
            }
        }
        return false;
    }
    ll flow(int source, int sink) {
        reset_flow();
        ll ans = 0;
        //for(lim = (1LL << 62); lim >= 1; lim /= 2)
        while(bfs(source, sink))
            ans += run(source, sink, LLINF);
        return ans;
    }
    void addEdge(int u, int v, ll c, ll rc) {
        Edge e = {u, v, 0, c};
        edge.pb(e);
        g[u].push_back(ne++);

        e = {v, u, 0, rc};
        edge.pb(e);
        g[v].push_back(ne++);
    }
    void reset_flow() {
        for(int i = 0; i < ne; i++)
            edge[i].flow = 0;
        memset(lvl, 0, sizeof(lvl));
        memset(vis, 0, sizeof(vis));
        memset(qu, 0, sizeof(qu));
        memset(px, 0, sizeof(px));
        qt = 0; pass = 0;
    }
    vector<pair<int, int>> cut() {
        vector<pair<int, int>> cuts;
        for (auto [from, to, flow, cap]: edge) {
            if (flow == cap and vis[from] == pass and vis[to] < pass and cap>0) {
                cuts.pb({from, to});
            }
        }
        return cuts;
    }
    void ans(int source, int sink, int total){
        flow(source,sink);
        for(int i =0;i<edge.size();i++){
            if(edge[i].flow == 1){
                matriz[edge[i].from][edge[i].to-total] = 'X';
            }
        }
    }
};
```

5.4 Finding Bridges

```
int n; // number of nodes
vector<vector<int>> adj; // adjacency list of graph

vector<bool> visited;
vector<int> tin, low;
int timer;

void dfs(int v, int p = -1) {
    visited[v] = true;
    tin[v] = low[v] = timer++;
    for (int to : adj[v]) {
        if (to == p) continue;
        if (visited[to]) {
            low[v] = min(low[v], tin[to]);
        } else {
            dfs(to, v);
            low[v] = min(low[v], low[to]);
            if (low[to] > tin[v])
                IS_BRIDGE(v, to);
        }
    }
}

void find_bridges() {
    timer = 0;
    visited.assign(n, false);
    tin.assign(n, -1);
    low.assign(n, -1);
    for (int i = 0; i < n; ++i) {
        if (!visited[i])
            dfs(i);
    }
}
```