代码模板

并查集

```
struct DSU {
    std::vector<int> f, siz;

    DSU() {}
    DSU(int n) {
        init(n);
    }

    void init(int n) {
        f.resize(n);
        std::iota(f.begin(), f.end(), 0);
        siz.assign(n, 1);
    }

    int find(int x) {
        while (x != f[x]) {
            x = f[x] = f[f[x]];
        }
        return x;
    }

    bool same(int x, int y) {
        return find(x) == find(y);
    }

    bool merge(int x, int y) {
        x = find(x);
        y = find(y);
        if (x == y) {
            return false;
        }
        siz[x] += siz[y];
        f[y] = x;
        return true;
    }

    int size(int x) {
        return siz[find(x)];
    }
};
```

使用方法：

1.把代码模板放到最前面进行定义

2.在主函数中声明并查集

3.使用对应的方法

实例：

```
#include <bits/stdc++.h>
using namespace std;
#define IOS                         \
    ios_base::sync_with_stdio(0); \
    cin.tie(0);                   \
    cout.tie(0);
typedef long long i64;
typedef pair<int, int> pii;
const int N = 2e5 + 10;
// dont use umap!!!

// 1. 先把模板贴过来
struct DSU {
    std::vector<int> f, siz;

    DSU() {}
    DSU(int n) {
        init(n);
    }

    void init(int n) {
        f.resize(n);
        std::iota(f.begin(), f.end(), 0);
        siz.assign(n, 1);
    }

    int find(int x) {
        while (x != f[x]) {
            x = f[x] = f[f[x]];
        }
        return x;
    }

    bool same(int x, int y) {
        return find(x) == find(y);
    }

    bool merge(int x, int y) {
        x = find(x);
        y = find(y);
        if (x == y) {
            return false;
        }
        siz[x] += siz[y];
        f[y] = x;
        return true;
```

```
    }

    int size(int x) {
        return siz[find(x)];
    }
};

signed main()
{
    int n;
    cin >> n;

    // 2. 声明一个并查集，括号内是构造函数，表示并查集的大小(是从0开始计算的所以要n+1)
    DSU dsu(n + 1);

    // 合并x，y
    int x, y;
    dsu.merge(x, y);

    // 查询x的根节点
    dsu.find(x);

    // 判断x,y是否在同一集合
    if(dsu.same(x, y)) cout << "Yes\n";

    // 求x所在集合大小
    int siz = dsu.size(x);

    return 0;
}
```

讲义标准程序

A 并查集

```
int f[N], siz[N];

int find(int x)
{
    return f[x] == x ? x : f[x] = find(f[x]);
}

void merge(int x,int y)
{
    x = find(x), y = find(y);
    if(x == y) return;
    if(x > y) swap(x, y);
    f[y] = x;
    siz[x] += siz[y];
}

signed main()
{
    IOS;
    int n, m;
    cin >> n >> m;
    for(int i = 1; i <= n; i ++) f[i] = i, siz[i] = 1;
    while(m--)
    {
        int op, x, y;
        cin >> op;
        if(op == 1)
        {
            cin >> x >> y;
            merge(x, y);
        }
        else if(op == 2)
        {
            cin >> x;
            cout << find(x) << '\n';
        }
        else
        {
            cin >> x;
            cout << siz[find(x)] << '\n';
        }
```

```
        }
        return 0;
    }
```

B 合并集合

```
vector<int> primes;
int vis[N], f[N];

int find(int u)
{
    return f[u] == u ? u : f[u] = find(f[u]);
}

void merge(int x,int y)
{
    x = find(x);
    y = find(y);
    if(x == y) return;
    f[y] = x;
}

signed main()
{
    IOS;
    int a, b, c;
    cin >> a >> b >> c;
    for(int i = 1; i <= b; i ++) f[i] = i;
    for(int i = 2; i <= b; i ++)
    {
        if(!vis[i]) primes.push_back(i);
        for(auto p : primes)
        {
            if(i * p > b) break;
            vis[i * p] = 1;
            if(i % p == 0) break;
        }
    }
    for(int p : primes)
    {
        if(p < c) continue;
        for(int i = (a + p - 1) / p * p; i <= b; i += p)
        {
            int x = i, y = i + p;
            if(y <= b) merge(x, y);
        }
    }
    set<int> st;
    for(int i = a; i <= b; i ++)
        st.insert(find(f[i]));
```

```
    cout << st.size() << '\n';
    return 0;
}
```

D 团伙

```
int f[N], siz[N];

int find(int x)
{
    return f[x] == x ? x : f[x] = find(f[x]);
}

void merge(int x,int y)
{
    x = find(x), y = find(y);
    if(x == y) return;
    if(x > y) swap(x, y);
    f[y] = x;
    siz[x] += siz[y];
}

signed main()
{
    IOS;
    int n, m;
    cin >> n >> m;
    for(int i = 1; i <= 2 * n; i ++) f[i] = i, siz[i] = 1;
    while(m--)
    {
        string op;
        int x, y;
        cin >> op;
        if(op == "F")
        {
            cin >> x >> y;
            merge(x, y);
        }
        else if(op == "E")
        {
            cin >> x >> y;
            merge(x, y + n);
            merge(y, x + n);
        }
    }
    int ans = 0;
    for(int i = 1; i <= n; i ++)
    {
        if(f[i] == i) ans ++;
    }
```

```
    cout << ans << '\n';
    return 0;
}
```

F 银河英雄传说

```
const int N = 30010;

int m;
int f[N], siz[N], d[N];

int find(int x)
{
    if (f[x] != x)
    {
        int root = find(f[x]);
        d[x] += d[f[x]];
        f[x] = root;
    }
    return f[x];
}

int main()
{
    cin.tie(0) -> sync_with_stdio(0);
    cin >> m;

    for (int i = 1; i < N; i ++ )
    {
        f[i] = i;
        siz[i] = 1;
    }

    while (m -- )
    {
        string op;
        int a, b;
        cin >> op >> a >> b;
        if (op == "M")
        {
            int fa = find(a), fb = find(b);
            if (fa != fb)
            {
                d[fa] = siz[fb];
                siz[fb] += siz[fa];
                f[fa] = fb;
            }
        }
        else
        {
```

```
            int fa = find(a), fb = find(b);
            if (fa != fb) cout << "-1\n";
            else cout << max(0, abs(d[a] - d[b]) - 1) << '\n';
        }
    }

    return 0;
}
```

G 最小生成树

Kruskal模板

```
int f[N], siz[N];

struct edge
{
    int u, v, w;
    bool operator<(edge &e)
    {
        return w < e.w;
    }
};

int find(int x)
{
    return f[x] == x ? x : f[x] = find(f[x]);
}

void merge(int x,int y)
{
    x = find(x), y = find(y);
    if(x == y) return;
    if(x > y) swap(x, y);
    f[y] = x;
    siz[x] += siz[y];
}

using i64 = long long;

signed main()
{
    IOS;
    int n, m;
    cin >> n >> m;
    for(int i = 1; i <= n; i ++) f[i] = i, siz[i] = 1;
    vector<edge> a;
    while(m--)
    {
        int u, v, w;
        cin >> u >> v >> w;
        a.push_back({u, v, w});
    }
    sort(a.begin(), a.end());
    i64 ans = 0;
    int cnt = 0;
    for(auto[u, v, w] : a)
    {
        u = find(u), v = find(v);
```

```
            if(u != v)
            {
                cnt ++;
                ans += w;
                merge(u, v);
            }
        }
        // if(cnt != n - 1) cout << -1 << '\n';
        if(siz[find(1)] != n) cout << -1 << '\n';
        else cout << ans << '\n';
        return 0;
    }
```