

## 普通并查集

---

### 并查集维护大小

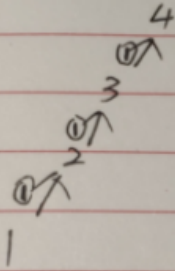
---

```
1  #include<bits/stdc++.h>
2  using namespace std;
3
4  const int N = 10010;
5
6  int n, m, vol;
7  int v[N], w[N];
8  int p[N];
9  int f[N];
10
11 int find(int x){
12     if (p[x] != x) p[x] = find(p[x]);
13     return p[x];
14 }
15
16 int main(){
17     cin >> n >> m >> vol;
18
19     for (int i = 1; i <= n; i ++ ){
20         p[i] = i;
21         cin >> v[i] >> w[i];
22     }
23
24     while (m -- ){
25         int a, b; cin >> a >> b;
26         int pa = find(a), pb = find(b);
27         if (pa != pb){
28             v[pb] += v[pa];
29             w[pb] += w[pa];
30             p[pa] = pb;
31         }
32     }
33
34     for (int i = 1; i <= n; i ++ )
35         if (p[i] == i)
36             for (int j = vol; j >= v[i]; j -- )
37                 f[j] = max(f[j], f[j - v[i]] + w[i]);
38
39     cout << f[vol] << endl;
40
41     return 0;
42 }
```

### 边权并查集

---

首先, 我们假设有4个节点存在如下关系:

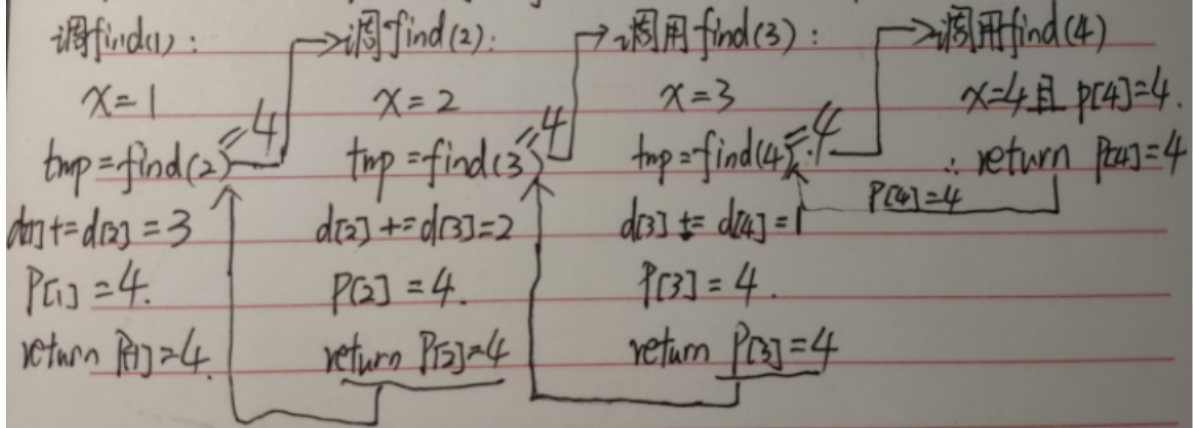


1 吃 2, 2 吃 3, 3 吃 4, 且 1 和 4 为同类

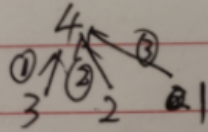
当前  $d[1]=1, d[2]=1, d[3]=1, d[4]=0$

$p[1]=2, p[2]=3, p[3]=4, p[4]=4$

下面我们来分析, 当调用  $\text{find}(1)$  时会发生哪些变化:



此时:



$d[1]=3, d[2]=2, d[3]=1, d[4]=0$

$p[1]=p[2]=p[3]=p[4]=4$

代码

```

1  memset(d,1,sizeof d)
2  int find(int x)
3  {
4      if(p[x] != x)
5      {
6          int tmp = find(p[x]);
7          d[x] += d[p[x]];
8          p[x] = tmp;
9      }
10     return p[x];
11 }
  
```

## 拓展域并查集

```

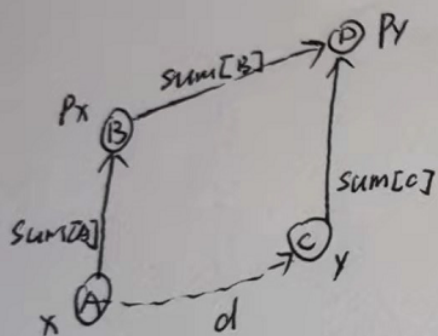
1  //这里我们将三个域,分别转化为了n,n+n,n+n+n.因为读入方面特别烦.
2  #include <bits/stdc++.h>
3  using namespace std;
4  int fa[200000];
5  int n,m,k,x,y,ans;
6  int get(int x)
  
```

```

7  {
8      if(x==fa[x])
9          return x;
10     return fa[x]=get(fa[x]);
11 }
12 void merge(int x,int y)
13 {
14     fa[get(x)]=get(y);
15 }
16 int main()
17 {
18     cin>>n>>m;
19     for(int i=1;i<=3*n;i++)
20         fa[i]=i;
21     for(int i=1;i<=m;i++)
22     {
23         scanf("%d%d%d",&k,&x,&y);
24         if(x>n || y>n)
25             ans++;
26         else if(k==1)
27         {
28             if(get(x)==get(y+n) || get(x)==get(y+n+n)) //如果x,y是同类,但是x是y
//的捕食中的动物,或者x是y天敌中的动物,那么错误.
29                 ans++;
30             else
31             {
32                 merge(x,y);
33                 merge(x+n,y+n);
34                 merge(x+n+n,y+n+n);
35             }
36         }
37         else
38         {
39             if(x==y || get(x)==get(y) || get(x)==get(y+n)) //x就是y,或者他们是
//同类,再或者是y的同类中有x
40                 ans++; //都是假话
41             else
42             {
43                 merge(x,y+n+n); // y的天敌域加入x
44                 merge(x+n,y); // x捕食域加入y
45                 merge(x+n+n,y+n); // x的天敌域为y的捕食域
46             }
47         }
48     }
49     cout<<ans<<endl;
50 }
51 //x是同类域.
52 //x+n是捕食域
53 //x+n+n是天敌域

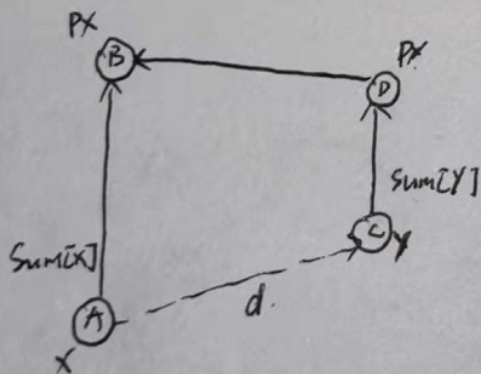
```

## 带权并查集



当  $p_y < p_x$  时. 连向小点的

$$\text{sum}[B] = \text{sum}[C] + d + \text{sum}[A]$$



当  $p_y > p_x$  时.

$$\text{sum}[py] = \text{sum}[x] - d - \text{sum}[y]$$

```

1  #include<bits/stdc++.h>
2  using namespace std;
3
4  int w,n,m;
5  const int maxn=1000;
6  int p[maxn],sum[maxn];
7
8  int find(int x){
9      if(x!=p[x]){
10         int px=find(p[x]);
11         sum[x]+=sum[p[x]];
12         p[x]=px;
13     }
14     return p[x];
15 }
16
17 bool merge(int x,int y,int d){
18     int px=find(x),py=find(y);
19     if(px==py){
20         if(sum[x]-sum[y]!=d) return 1;
21     }
22     else{
23         if(px<py){
24             p[px]=py;
25             sum[px]=sum[y]+d-sum[x];
26         }
27         else{
28             p[py]=px;
29             sum[py]=sum[x]-d-sum[y];
30         }
31     }
32     return 0;
33 }

```

```

34
35 int main(){
36     cin>>w;
37     while(w--){
38         int n,m; cin>>n>>m;
39         for(int i=0;i<=n;i++){
40             p[i]=i,sum[i]=0;
41         }
42
43         bool bz=true;
44         while(m--){
45             int s,t,v; cin>>s>>t>>v;
46             s--; //区域
47             if(merge(s,t,v)) bz=false;
48         }
49         if(bz) cout<<"true"<<endl;
50         else cout<<"false"<<endl;
51     }
52
53     return 0;
54 }

```

## 并查集模板

```

1 struct DSU {
2     std::vector<int> p, siz;
3     DSU(int n) : p(n+1), siz(n+1, 1) { std::iota(p.begin(), p.end(), 0); }
4     int find(int x) {
5         return p[x] == x ? x : p[x] = find(p[x]);
6     }
7     bool same(int x, int y) { return find(x) == find(y); }
8     bool merge(int x, int y) {
9         x = find(x);
10        y = find(y);
11        if (x == y) return false;
12        siz[x] += siz[y];
13        p[y] = x;
14        return true;
15    }
16    int size(int x) { return siz[find(x)]; }
17 };

```

## 可持久化并查集