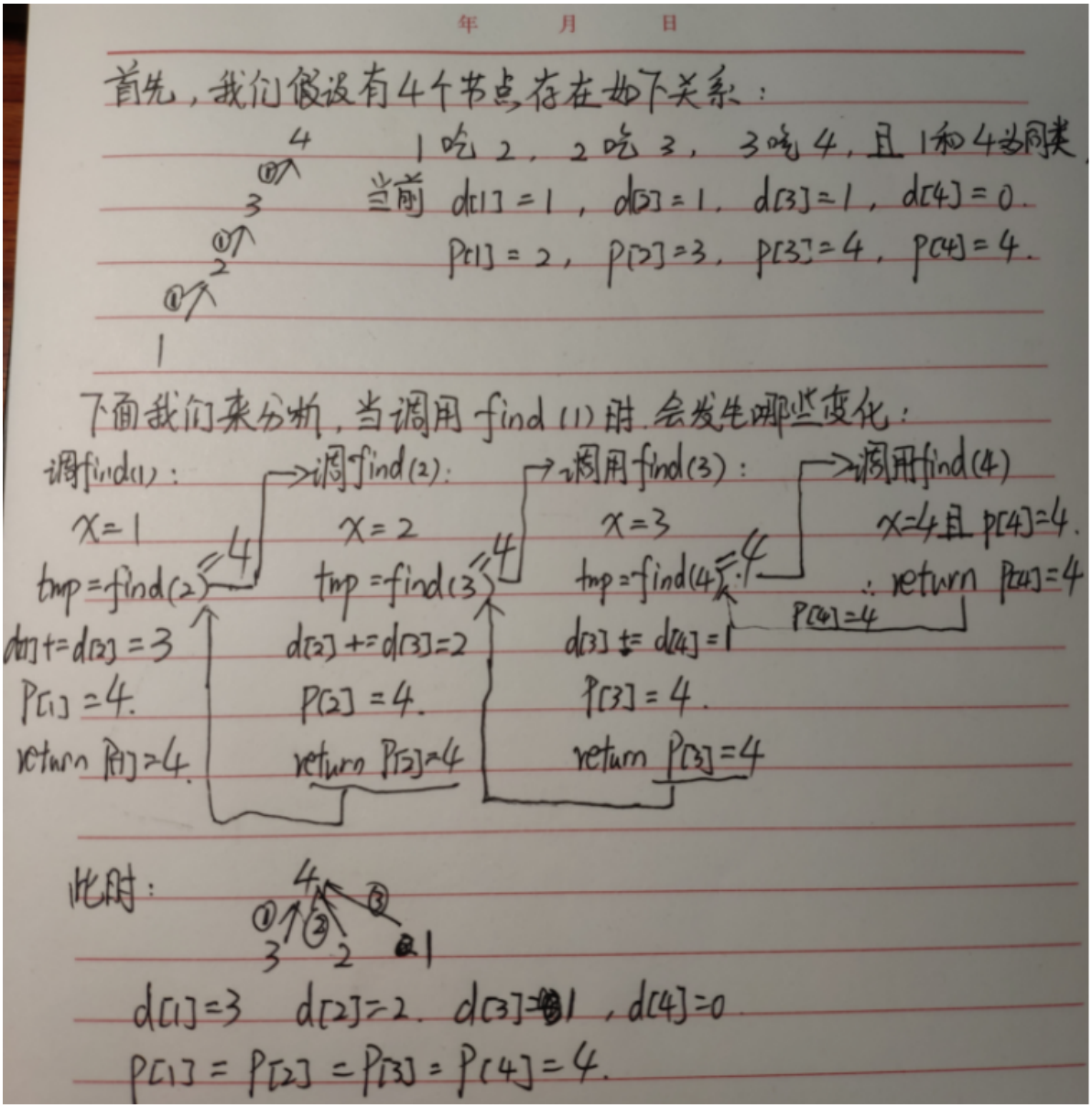
普通并查集

并查集维护大小

```cpp
#include<bits/stdc++.h>
using namespace std;

const int N = 10010;

int n, m, vol;
int v[N], w[N];
int p[N];
int f[N];

int find(int x){
    if (p[x] != x) p[x] = find(p[x]);
    return p[x];
}

int main(){
    cin >> n >> m >> vol;

    for (int i = 1; i <= n; i ++ ){
        p[i] = i;
        cin >> v[i] >> w[i];
    }

    while (m -- ){
        int a, b; cin >> a >> b;
        int pa = find(a), pb = find(b);
        if (pa != pb){
            v[pb] += v[pa];
            w[pb] += w[pa];
            p[pa] = pb;
        }
    }

    for (int i = 1; i <= n; i ++ )
        if (p[i] == i)
            for (int j = vol; j >= v[i]; j -- )
                f[j] = max(f[j], f[j - v[i]] + w[i]);

    cout << f[vol] << endl;

    return 0;
}
```

边权并查集



代码

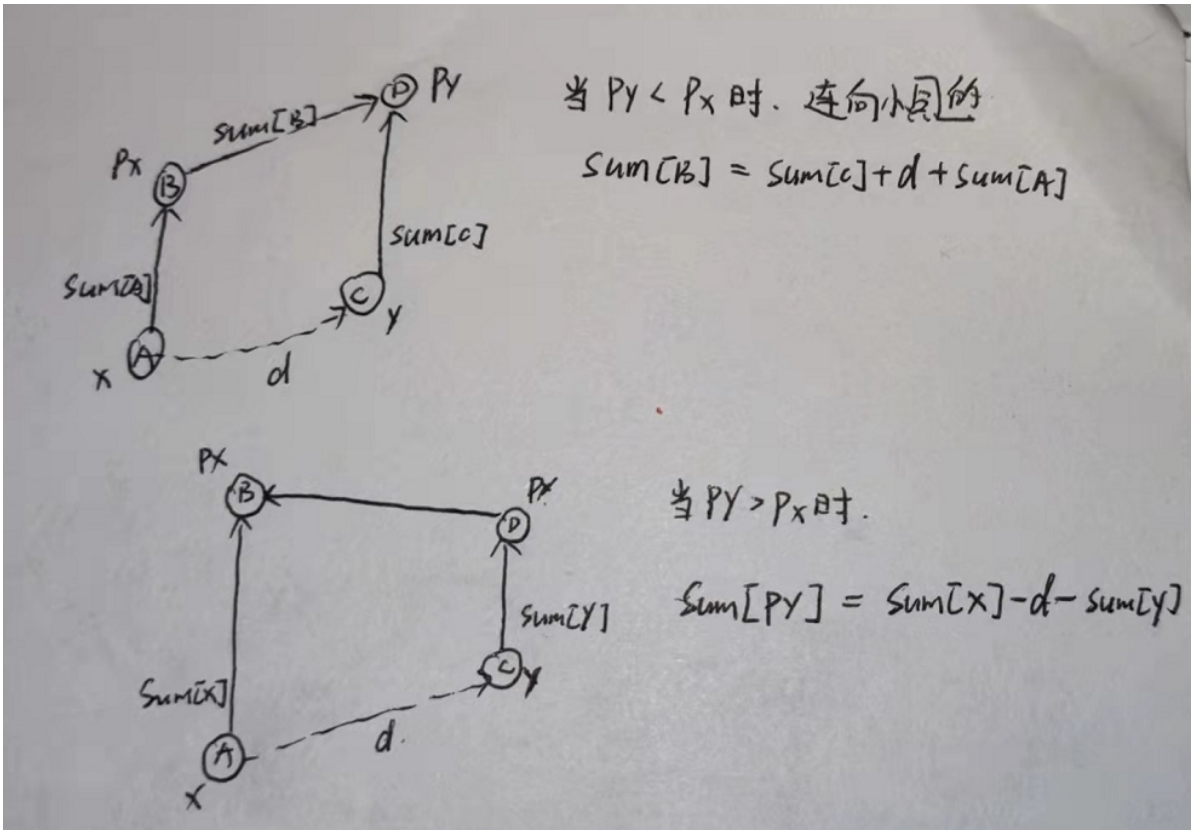
```
memset(d,1,sizeof d)
int find(int x)
{
    if(p[x] != x)
    {
        int tmp = find(p[x]);
        d[x] += d[p[x]];
        p[x] = tmp;
    }
    return p[x];
}
```

拓展域并查集

```
//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.
#include <bits/stdc++.h>
using namespace std;
int fa[200000];
int n,m,k,x,y,ans;
int get(int x)
```

```
{
    if(x==fa[x])
        return x;
    return fa[x]=get(fa[x]);
}
void merge(int x,int y)
{
    fa[get(x)]=get(y);
}
int main()
{
    cin>>n>>m;
    for(int i=1;i<=3*n;i++)
        fa[i]=i;
    for(int i=1;i<=m;i++)
    {
        scanf("%d%d%d",&k,&x,&y);
        if(x>n || y>n)
            ans++;
        else if(k==1)
        {
            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.
                ans++;
            else
            {
                merge(x,y);
                merge(x+n,y+n);
                merge(x+n+n,y+n+n);
            }
        }
        else
        {
            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
                ans++;//都是假话
            else
            {
                merge(x,y+n+n);      // y的天敌域加入x
                merge(x+n,y);        // x捕食域加入y
                merge(x+n+n,y+n);    // x的天敌域为y的捕食域
            }
        }
    }
    cout<<ans<<endl;
}
//x是同类域.
//x+n是捕食域
//x+n+n是天敌域
```

带权并查集




```cpp
#include<bits/stdc++.h>
using namespace std;

int w,n,m;
const int maxn=1000;
int p[maxn],sum[maxn];

int find(int x){
    if(x!=p[x]){
        int px=find(p[x]);
        sum[x]+=sum[p[x]];
        p[x]=px;
    }
    return p[x];
}

bool merge(int x,int y,int d){
    int px=find(x),py=find(y);
    if(px==py){
        if(sum[x]-sum[y]!=d) return 1;
    }
    else{
        if(px<py){
            p[px]=py;
            sum[px]=sum[y]+d-sum[x];
        }
        else{
            p[py]=px;
            sum[py]=sum[x]-d-sum[y];
        }
    }
    return 0;
}
```

```cpp

int main(){
    cin>>w;
    while(w--){
        int n,m; cin>>n>>m;
        for(int i=0;i<=n;i++){
            p[i]=i,sum[i]=0;
        }

        bool bz=true;
        while(m--){
            int s,t,v; cin>>s>>t>>v;
            s--; //区域
            if(merge(s,t,v)) bz=false;
        }
        if(bz) cout<<"true"<<endl;
        else cout<<"false"<<endl;
    }

    return 0;
}
```

并查集模板

```cpp
struct DSU {
    std::vector<int> p, siz;
    DSU(int n) : p(n+1), siz(n+1, 1) { std::iota(p.begin(), p.end(), 0); }
    int find(int x) {
        return p[x] == x ? x : p[x] = find(p[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];
        p[y] = x;
        return true;
    }
    int size(int x) { return siz[find(x)]; }
};
```

可持久化并查集