



并查集 (DSU)

用于快速判断元素是否属于同一个集合。

支持操作

1. 将两个元素放到同一个集合中。
2. 查询两个元素是否属于同一个集合。

代表元素思想

选择一个元素作该集合的代表元素，其他需要添加的元素作为这个代表元素的儿子，形成一颗树。

把判断集合与元素间的从属转化为集合与集合间的关系，通过递归查找。

实现方式

1. 路径压缩

简单来说，每次都递归查找太麻烦了！我们记录下每次查询的结果，查询时就可以直接调用。

2. 按秩合并

用于在合并两个集合时尽量保持并查集的树高度较小，防止形成长链。

通过上述两种方式，我们可以避免可能途径的超长递归链，以此减少查询的时间开销。

Code

```
namespace dsu{
    int fa[maxn];
    inline void init(int x){for(int i=1;i<=x;i++)fa[i] = i;}
    inline int find(int a){return fa[a]==a?a:fa[a] = find(fa[a]);}
    inline void merge(int a,int b){fa[find(a)] = find(b);}
    inline bool check(int a,int b){return find(a)==find(b);}
}
```

例题

[P3367 【模板】并查集](#)

```
#include <cstdio>
```

```
#define int long long
using namespace std;
const int maxn = 2e5+5;
int n,m,opt,x,y;
namespace dsu{
    int fa[maxn];
    inline void init(int x){for(int i=1;i<=x;i++)fa[i] = i;}
    inline int find(int a){return fa[a]==a?a:fa[a] = find(fa[a]);}
    inline void merge(int a,int b){fa[find(a)] = find(b);}
    inline bool check(int a,int b){return find(a)==find(b);}
}

signed main(){
    scanf("%lld%lld",&n,&m);
    for(int i=1;i<=n;i++)dsu::fa[i]=i;
    while(m--){
        scanf("%d%d%d",&opt,&x,&y);
        if(opt==1) dsu::merge(x,y);
        else printf("%c\n",dsu::check(x,y)?'Y':'N');
    }
    return 0;
}
```