2022-2 HI-ARC 중급스터디

7주차. 분리 집합

이지은 (leeju1013)

목차

집합의 표현

성공 스페셜 저지

골드 IV

시간 제한	메모리 제한	제출	정답
2 초	128 MB	68615	21689

문제

초기에 {0}, {1}, {2}, ... {n} 이 각각 n+1개의 집합을 이루고 있다. 여기에 합집합 연산과, 두 원소가 같은 집합에 포함되어 있는지를 확인하는 연산을 수행하려고 한다.

집합을 표현하는 프로그램을 작성하시오.

입력

첫째 줄에 n(1 ≤ n ≤ 1,000,000), m(1 ≤ m ≤ 100,000)이 주어진다. m은 입력으로 주어지는 연산의 개수이다. 다음 m개의 줄에는 각각의 연산이 주어진다. 합집합은 0 a b의 형태로 입력이 주어진다. 이는 a가 포함되어 있는 집합과, b가 포함되어 있는 집합을 합친다는 의미이다. 두 원소가 같은 집합에 포함되어 있는지를 확인하는 연산은 1 a b의 형태로 입력이 주어진다. 이는 a와 b가 같은 집합에 포함되어 있는지를 확인하는 연산이다. a와 b는 n 이하의 자연수 또는 0이며 같을 수도 있다.

출력

1로 시작하는 입력에 대해서 한 줄에 하나씩 YES/NO로 결과를 출력한다. (yes/no 를 출력해도 된다)

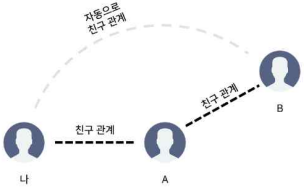
예제 입력 1

```
7 8
0 1 3
1 1 7
0 7 6
1 7 1
0 3 7
0 4 2
0 1 1
1 1 1
```

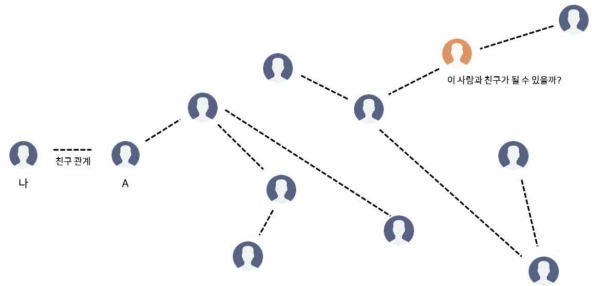
1, 2, 3, 4, 5, 6, 7
1-3, 2, 4, 5, 6, 7
NO
1-3, 2, 4, 5, 6-7
NO
1-3-6-7, 2, 4, 5
1-3-6-7, 2-4, 5
1-3-6-7, 2-4, 5
YES

예제 출력 1

```
NO
NO
YES
```

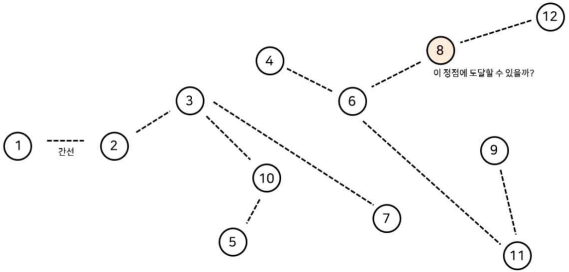


A와 B가 친구 관계이고, 내가 A와 친구 관계이면
자동으로 나와 B는 친구 관계가 되는 메신저 프로그램

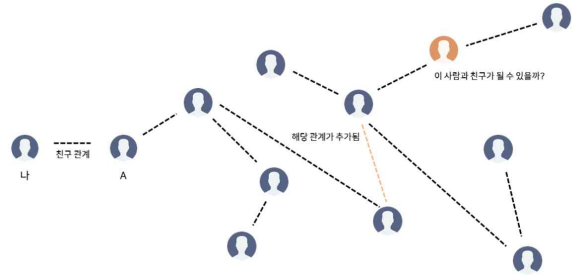


내가 A와 친구 관계를 맺었을 때,
주황색으로 표시된 사람과 친구 관계인지
알 수 있는 방법은?

(출처)



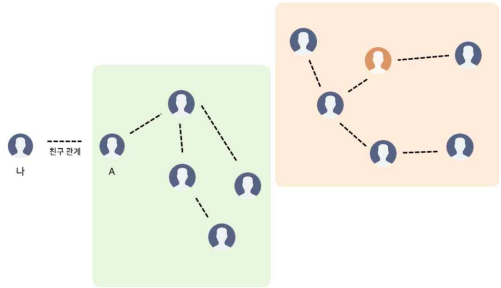
친구 관계를 그래프로 나타낸 후
나를 시작으로 그래프 탐색(BFS, DFS)을 통해
목표 정점까지 도달할 수 있는 지 확인하면 됨.



새로운 친구 관계가 생긴다면?

주황색 사람과 친구 관계가 될 수 있는 지 알기 위해
다시 그래프 탐색을 진행해야함.

시간 오래 걸림...

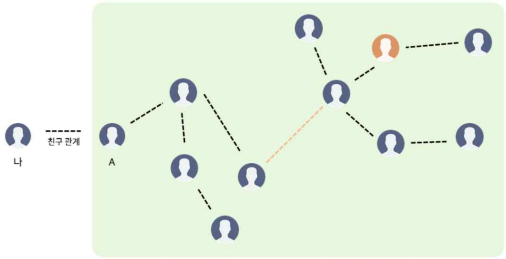


친구 관계를 그룹으로 표현해보자!

친구 관계를 맺으면 같은 그룹에 속하게 됨.

다른 사용자와 친구 관계인지 알려면
속한 그룹만 비교하면 됨.

주황색으로 표시된 사람과 나는 다른 그룹이므로
친구 관계가 아니라는 것을 쉽게 알 수 있음.



주황색 점선으로 표시된 새로운 친구 관계 생성 시
두 그룹이 합쳐지며 친구 관계가 갱신됨.

주황색으로 표시된 사람과 내가 같은 그룹이므로
친구 관계임을 알 수 있음.

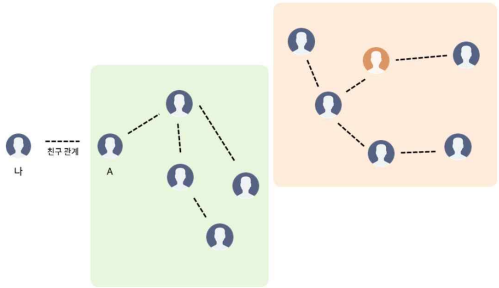
1. 분리 집합(Disjoint Set)

- What?

: 서로 겹치는 원소가 없는 집합들을 관리하는 자료구조
(= 서로소 집합)

즉, 전체 집합 U에 대해
겹치는 부분이 발생하지 않도록
모든 원소들을 분리한 부분집합

ex) 나를 제외한 총 사용자 11명을
두 그룹으로 겹치지 않게 나누었음.



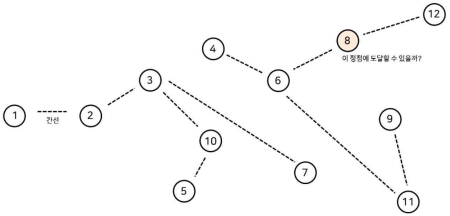
2. Union-Find 알고리즘

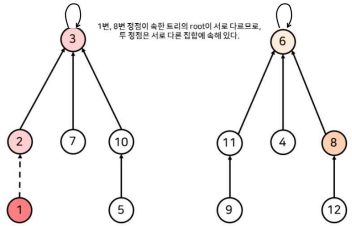
- What?
: 두 노드가 같은 집합에 속하는지 판별하는 알고리즘

- How?
: 각 그룹을 트리의 형태로 표현함

- Find 연산
: 노드 x가 어느 집합에 포함되어 있는지 찾기

- Union 연산
: 노드 x가 포함된 집합과 노드 y가 포함된 집합을 합치기



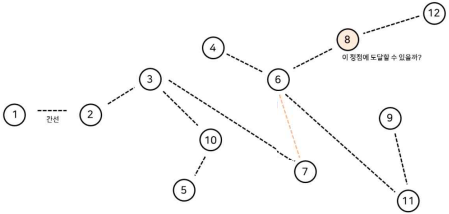


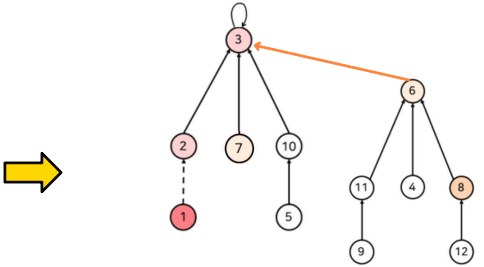
n	1	2	3	4	5	6	7	8	9	10	11	12
Parent[n]	2	3	3	6	10	6	3	6	11	3	6	8

각 그룹을 트리(계층형 구조의 그래프) 형태로 표현!
각 노드는 부모 노드를 가지며, 트리의 root 노드로 그룹을 구별함.(root노드의 부모는 자기 자신으로 설정)

1번 노드(나)와 2번 노드가 친구 관계를 맺었으므로, 1번 노드가 2번 노드의 자식 노드로 연결되어있음.

내가 8번 노드와 친구 관계인지 알고싶다면?
내가 속한 트리의 root(3번 노드)와 8번 노드가 속한 트리의 root(6번 노드)를 비교하면 됨.
현재 root가 서로 다르므로 나와 8번 노드는 친구 관계가 아님.





n	1	2	3	4	5	6	7	8	9	10	11	12
Parent[n]	2	3	3	6	10	3	3	6	11	3	6	8

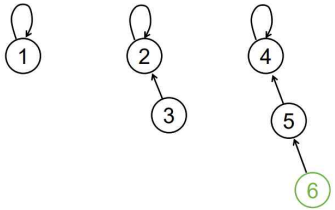
6, 7번 노드 간 친구 관계가 새로 생성된다면?
각각의 root를 찾아준 후(**find**), 두 트리의 root가 다른 경우에는 한 root의 부모 노드를 다른 하나로 지정.(**union**)

이제 내가 속한 트리의 root와 8번 노드가 속한 트리의 root가
3번 노드로 같아졌으므로 나와 8번 노드는 친구 관계임.

2. Union-Find 알고리즘

- Find 연산
  : 주어진 노드가 포함된 집합의 대푯값(root 노드) 구하기

```
int find(int num){
    if(num==par[num]) return num; // num이 루트노드이면, 그대로 반환
    return find(par[num]); // num이 루트노드가 아니라면, 부모노드에 대해 재귀실행하여 루트노드를 반환
}
```



find(1)의 결과 : 1
find(3)의 결과 : 2
find(6)의 결과 : 4

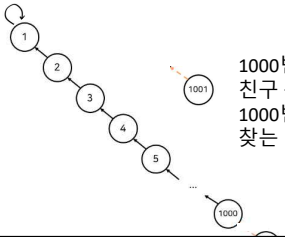
집합마다 유일한 대푯값(root 노드)이 존재하며
나머지 다른 원소에서 대표로 가는 경로가 존재함.

2. Union-Find 알고리즘

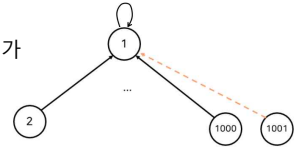
- Find 연산

  -> 최적화: 경로 압축(Path Compression)

```
int find(int num){
    if(num==par[num]) return num; // num이 루트노드이면, 그대로 반환
    return par[num] = find(par[num]); // 경로압축(부모노드를, 최종적으로 찾을 root노드로 갱신)
}
```



1000번 노드와 1001번 노드가
친구 관계를 맺으려 할 때,
1000번 노드의 root노드를
찾는 데에 O(N)이 걸림.

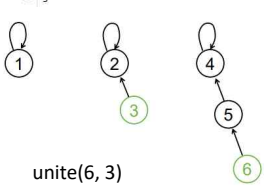


한 노드에 대해 find연산을
실행할 때마다
그 노드의 부모 노드를
항상 root노드로 만들어줌.
-> O(1)

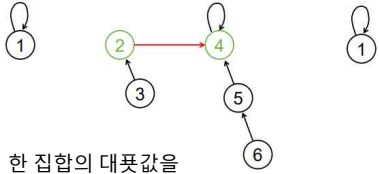
2. Union-Find 알고리즘

- Union 연산
  : 두 집합(트리)을 병합하기

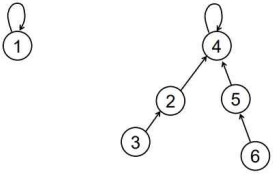
```
1 void unite(int a, int b) {
2     a = find(a); // a가 속한 트리의 루트노드 찾기
3     b = find(b); // b가 속한 트리의 루트노드 찾기
4
5     par[b] = a; // 서로 다른 트리에 속한다면(a!=b) 한 쪽의 트리를 다른 쪽에 붙이기
6 }
```



unite(6, 3)



한 집합의 대푯값을
다른 집합의 대푯값으로 linking



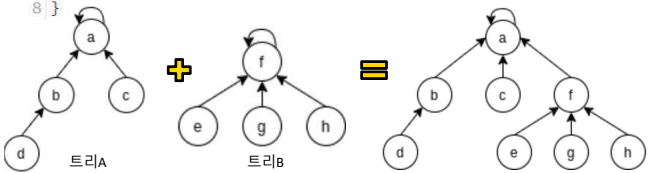
2. Union-Find 알고리즘

- Union 연산

  -> 최적화: Union by Rank

```
void unite(int a, int b){
    a=find(a), b=find(b); // a, b가 속한 트리의 루트노드 찾기
    if(height[a] < height[b]) swap(a,b); //높이는 항상 a>=b
    par[b]=a; // 높이가 높은 트리 a에 높이가 낮은 트리 b를 붙이기 (b의 부모가 a)

    if(height[a] == height[b]) // 높이가 a==b였다면, 합친 후 트리의 높이를 1 증가
        height[a]++;
}
```



트리A 트리B

find 연산의 시간복잡도는
트리의 높이에 의해 결정됨.

union 연산 시 항상
높이가 낮은 트리를 높은 트리에 붙이자!

쉬는 시간

1. 분리 집합
2. Union-Find
   - 1717. 집합의 표현
   - 18116. 로봇 조립

집합의 표현

성공 스페셜 저지

골드 IV

시간 제한	메모리 제한	제출	정답
2 초	128 MB	68615	21689

문제

초기에 {0}, {1}, {2}, ... {n} 이 각각 n+1개의 집합을 이루고 있다. 여기에 합집합 연산과, 두 원소가 같은 집합에 포함되어 있는지를 확인하는 연산을 수행하려고 한다.

집합을 표현하는 프로그램을 작성하시오.

입력

첫째 줄에 n(1 ≤ n ≤ 1,000,000), m(1 ≤ m ≤ 100,000)이 주어진다. m은 입력으로 주어지는 연산의 개수이다. 다음 m개의 줄에는 각각의 연산이 주어진다. 합집합은 0 a b의 형태로 입력이 주어진다. 이는 a가 포함되어 있는 집합과, b가 포함되어 있는 집합을 합친다는 의미이다. 두 원소가 같은 집합에 포함되어 있는지를 확인하는 연산은 1 a b의 형태로 입력이 주어진다. 이는 a와 b가 같은 집합에 포함되어 있는지를 확인하는 연산이다. a와 b는 n 이하의 자연수 또는 0이며 같을 수도 있다.

출력

1로 시작하는 입력에 대해서 한 줄에 하나씩 YES/NO로 결과를 출력한다. (yes/no 를 출력해도 된다)

예제 입력 1

```
7 8
0 1 3
1 1 7
0 7 6
1 7 1
0 3 7
0 4 2
0 1 1
1 1 1
```

1, 2, 3, 4, 5, 6, 7
1-3, 2, 4, 5, 6, 7
NO
1-3, 2, 4, 5, 6-7
NO
1-3-6-7, 2, 4, 5
1-3-6-7, 2-4, 5
1-3-6-7, 2-4, 5
YES

예제 출력 1

```
NO
NO
YES
```

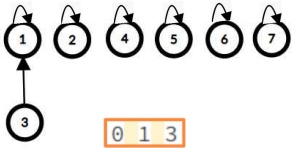
예제 입력 1

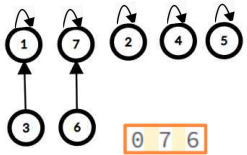
```
7 8
0 1 3
1 1 7
0 7 6
1 7 1
0 3 7
0 4 2
0 1 1
1 1 1
```

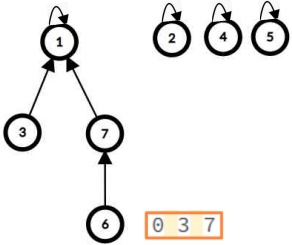
1, 2, 3, 4, 5, 6, 7
1-3, 2, 4, 5, 6, 7
NO
1-3, 2, 4, 5, 6-7
NO
1-3-6-7, 2, 4, 5
1-3-6-7, 2-4, 5
1-3-6-7, 2-4, 5
YES

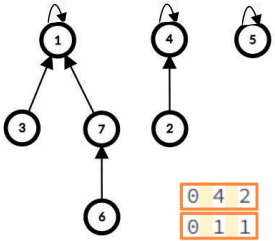
예제 출력 1

```
NO
NO
YES
```





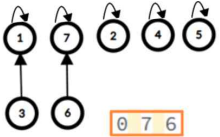


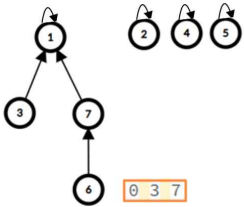


```cpp
#include <iostream>
using namespace std;

int par[1000001], height[1000001];
int find(int num){
    if(num==par[num]) return num;
    return par[num] = find(par[num]); //경로압축
}
void unite(int a, int b){
    a=find(a), b=find(b);
    if(height[a] < height[b]) swap(a,b);
    // 높이가 높은 트리 a에 높이 낮은 b를 붙이기
    par[b]=a;

    if(height[a] == height[b])
        //높이가 a==b였다면 a의 높이는 1 증가
        height[a]++;
}
```






```cpp
int main(){
    ios_base::sync_with_stdio(false); cin.tie(NULL);

    int n,m; cin>> n >> m;
    for(int i=0; i<=n; i++) par[i]=i; // self roof라고 초기화

    for(int i=0; i<m; i++){
        int k,a,b; cin>> k >> a >> b;

        if(k==0) unite(a,b); // 합집합 연산
        else{
            if(find(a)==find(b)) cout<<"YES\n"; // 같은 집합인지 확인 연산
            else cout<<"NO\n";
        }
    }
    return 0;
}
```

로봇 조립

성공

골드 IV

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
4 초 (추가 시간 없음)	1024 MB	2199	667	491	27.961%

문제

성규는 로봇을 조립해야 한다. 상자 안에는 여러 로봇의 부품들이 섞여 있다. 그런데 어떤 부품이 어느 로봇의 부품인지 표시가 되어있지 않다. 호재는 전자과라서 두 부품을 보면 같은 로봇의 부품인지 알 수 있다. 그래서 성규는 호재의 지시에 따라 부품들을 정리하기로 하였다.

부품들은 1부터 10^6까지의 정수로 표현된다. 그리고 부품 i가 속한 로봇은 robot(i)라고도 표현한다. 예를 들어, 부품 11과 부품 22가 로봇 A의 부품이라고 알고 있는 경우, robot(11)은 로봇 A를 의미하고, robot(22)도 로봇 A를 의미한다.

서로 다른 로봇은 공통 부품을 가지지 않는다. 즉 어떤 부품이 로봇 A의 부품이라면, 로봇 B의 부품은 될 수 없다.

호재는 2가지 지시를 한다.

- 서로 다른 부품 2개를 말해주며, 두 부품은 같은 로봇의 부품이라는 정보를 알려준다. **union 연산!**
- 부품 i에 대해서, 지금까지 알아낸 robot(i)의 부품이 몇 개냐고 물어본다. **부품 i가 속한 트리의 노드 개수?**

초기에는 부품에 대한 정보가 존재하지 않는다.

입력

첫 번째 줄에 호재의 지시 횟수 N이 들어온다. ($1 \le N \le 10^6$)

다음 줄부터 N개의 지시가 들어온다.

부품 2개가 같은 로봇의 부품인지 알려줄 때에는 $I\ a\ b$ 의 형태로 들어온다. 부품 a와 부품 b는 같은 로봇의 부품이라는 의미이다. ($1 \le a, b \le 10^6$, $a \ne b$, a, b는 정수)

어떤 로봇의 부품이 몇 개인지 물어볼 때에는 $Q\ c$ 의 형태로 들어온다. 지금까지 알아낸 robot(c)의 부품이 몇 개냐는 의미이다. ($1 \le c \le 10^6$, c는 정수)

입력으로 $Q\ c$의 형태가 적어도 한 번 들어온다.

출력

Q로 시작하는 입력에 대해서 한 줄에 하나씩, 지금까지 알아낸 해당 로봇의 부품 개수를 출력한다.

예제 입력 1 복사

```
4
I 1 2
I 3 2
Q 1
Q 4
```

예제 출력 1 복사

```
3
1
```

예제 입력 1

```
4
I 1 2
I 3 2
Q 1
Q 4
```

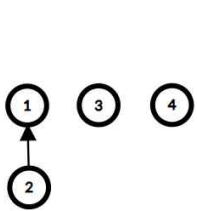
예제 출력 1

```
3
1
```

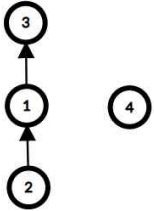
```
if(c=='I'){ // 부품 2개가 같은 로봇의 부품인지 알려줄 때
    cin>>a>>b;
    unite(a,b);
}
else{ // 어떤 로봇의 부품이 몇 개인지 물어볼 때
    cin>>a;
    cout<<cnt[find(a)]<<'\n';
}
```

```
void unite(int a, int b){
    a=find(a), b=find(b);
    if(a!=b){ // 서로 다른 트리에 속한다면
        par[b]=a; // 트리 b를 트리 a에 붙이고
        cnt[a]+=cnt[b]; // 트리 a에 속한 노드의 개수 갱신
    }
}
```

idx	1	2	3	4
cnt	1	1	1	1



idx	1	2	3	4
cnt	2	1	1	1



idx	1	2	3	4
cnt	2	1	3	1

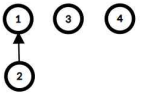
```
4
I 1 2
I 3 2
Q 1
Q 4
```

```
3
1
```

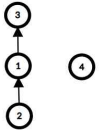
```cpp
#include <iostream>
using namespace std;

int par[1000001],cnt[1000001];
// cnt[i] : i번 노드가 루트노드인 트리의 노드 개수

int find(int num){
    if(num==par[num]) return num;
    return par[num] = find(par[num]); //경로압축
}
void unite(int a, int b){
    a=find(a), b=find(b);
    if(a!=b){ // 서로 다른 트리에 속한다면
        par[b]=a; // 트리 b를 트리 a에 붙이고
        cnt[a]+=cnt[b]; // 트리 a에 속한 노드의 개수 갱신
    }
}
```



idx	1	2	3	4
cnt	2	1	1	1



idx	1	2	3	4
cnt	2	1	3	1

```cpp
int main(){
    ios_base::sync_with_stdio(false); cin.tie(NULL);

    int n; cin>>n;
    for(int i=0; i<1000001; i++) par[i]=i, cnt[i]=1;

    for(int i=0; i<n; i++){
        char c;
        int a,b;
        cin>>c;
        if(c=='I'){ // 부품 2개가 같은 로봇의 부품인지 알려줄 때
            cin>>a>>b;
            unite(a,b);
        }
        else{ // 어떤 로봇의 부품이 몇 개인지 물어볼 때
            cin>>a;
            cout<<cnt[find(a)]<<'\n';
        }
    }
    return 0;
}
```

1	2	3	4
idx	1	2	3	4
cnt	1	1	1	1

감사합니다

- 필수 문제
  - 1717. 집합의 표현
  - 18116. 로봇 조립
- 연습 문제
  - 24391. 귀찮은 해강이
  - 1976. 여행 가자
  - 1043. 거짓말
  - 4803. 트리
  - 16562. 친구비
  - 11805. 군사 이동
  - 20040. 사이클 게임
  - 13904. 과제
  - 10775. 공항
- 11월 23일(수요일) 저녁 6시 T702