

Union Find (Disjoint Set)

(상호 독립 집합을 찾아 합치기 알고리즘)

(주)한컴에듀케이션 이주현

1. Union Find Algorithm

❖ 집합의 처리

- 상호 배타적 집합(서로소 집합)만을 대상으로 한다. 따라서 교집합은 없다.
- 상호 배타적인 독립 집합(disjoint set)을 찾아 합친다.

❖ 지원할 연산

- $\text{Make}(x)$: 원소 x 로만 이루어진 집합을 만든다.
- $\text{Find}(x)$: 원소 x 를 포함하는 집합을 알아낸다.
- $\text{Union}(x, y)$: 원소 x 를 포함하는 집합과 원소 y 를 포함하는 집합을 합한다.

❖ 참조

- ✓ 쉽게 배우는 알고리즘 - 문병로
- ✓ wiki

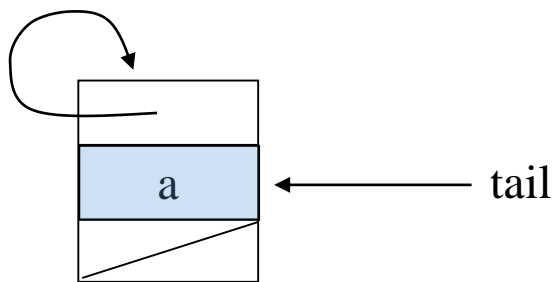
학습 목표

- ❖ 연결 리스트를 이용한 상호 배타적 집합의 처리 방법을 이해한다.
- ❖ 연결 리스트를 이용해 집합을 처리하는 연산들의 수행 시간을 분석할 수 있도록 한다.
- ❖ 트리를 이용한 상호 배타적 집합의 처리 방법을 이해한다.
- ❖ 트리를 이용해 집합을 처리하는 연산들의 수행 시간을 기본적인 수준에서 분석할 수 있도록 한다.

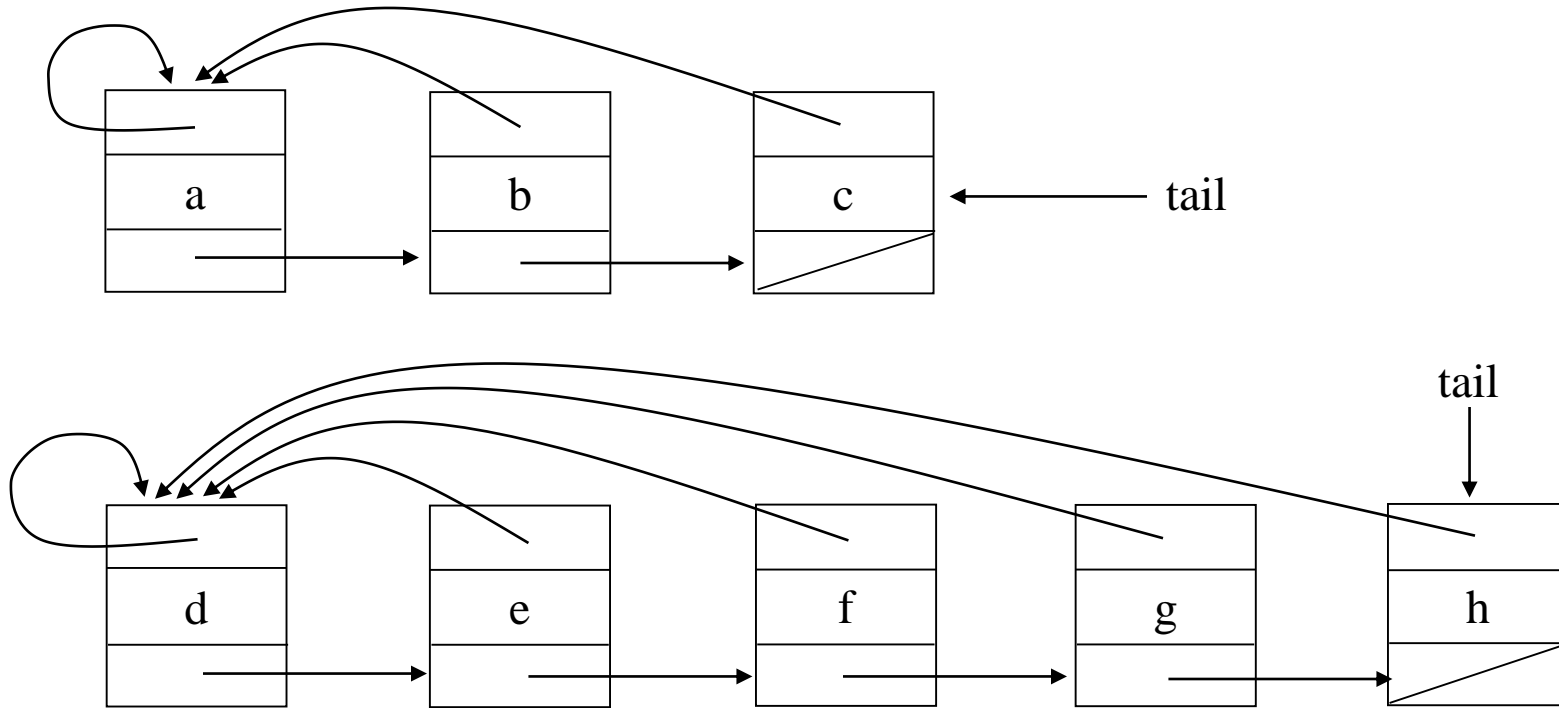
2. 연결 리스트를 이용한 처리

- ❖ 같은 집합의 원소들은 하나의 연결 리스트로 관리한다
- ❖ 연결 리스트의 맨 앞의 원소를 집합의 대표 원소로 삼는다

하나의 원소로 이루어진 집합

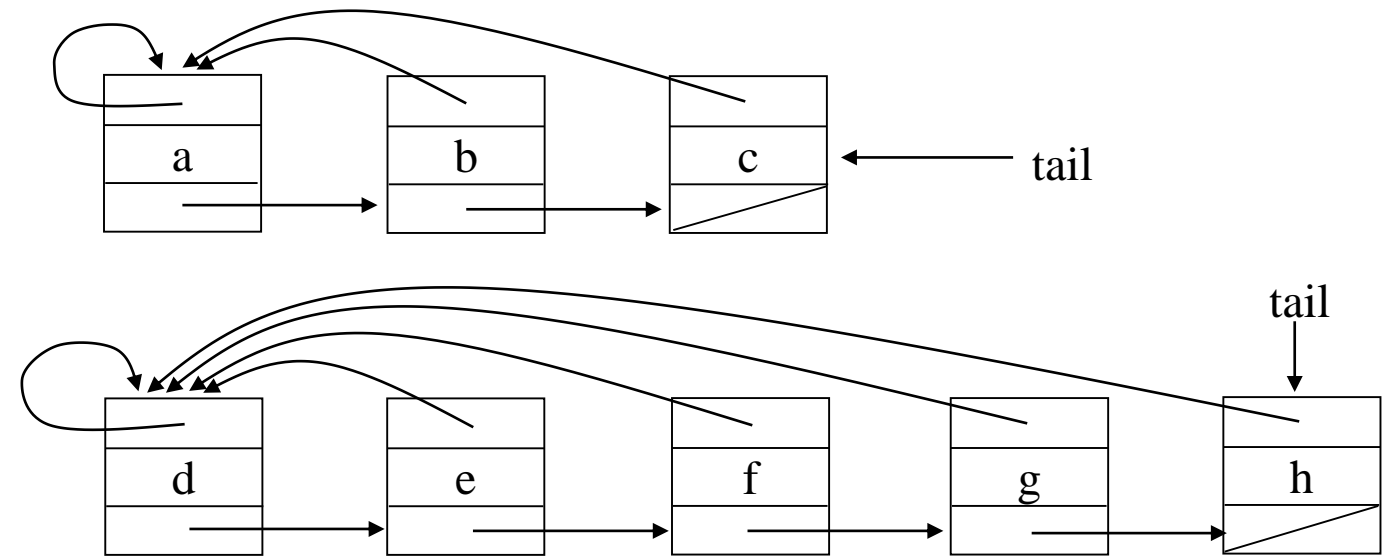


연결리스트로 이루어진 두 집합

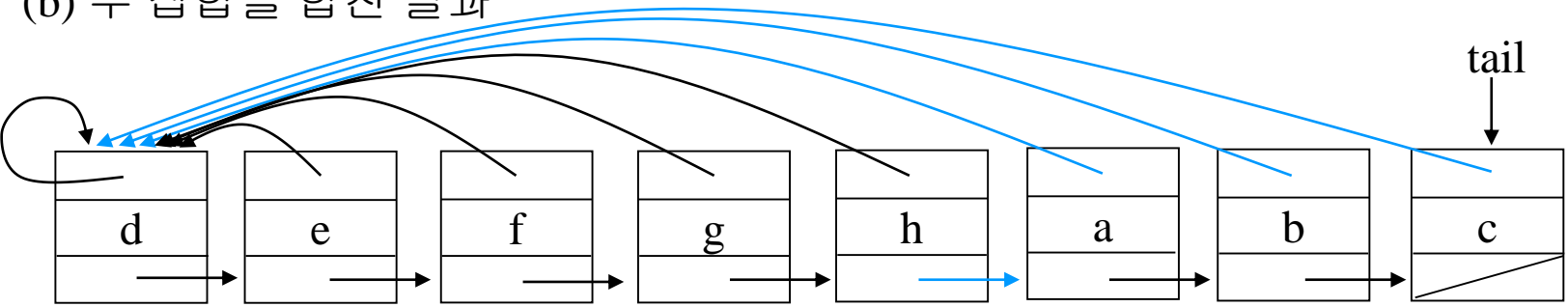


합집합을 만드는 예

(a) 합치고자 하는 두 집합



(b) 두 집합을 합친 결과



무게를 고려한 Union

- ❖ 연결 리스트로 된 두 집합을 합칠 때
작은 집합을 큰 집합의 뒤에 붙인다
 - 대표 원소를 가리키는 포인터 갱신 작업을 최소화하기 위한 것

수행시간

[정리 1]

연결 리스트를 이용해 표현되는 배타적 집합에서

무게를 고려한 **Union**을 사용할 때,

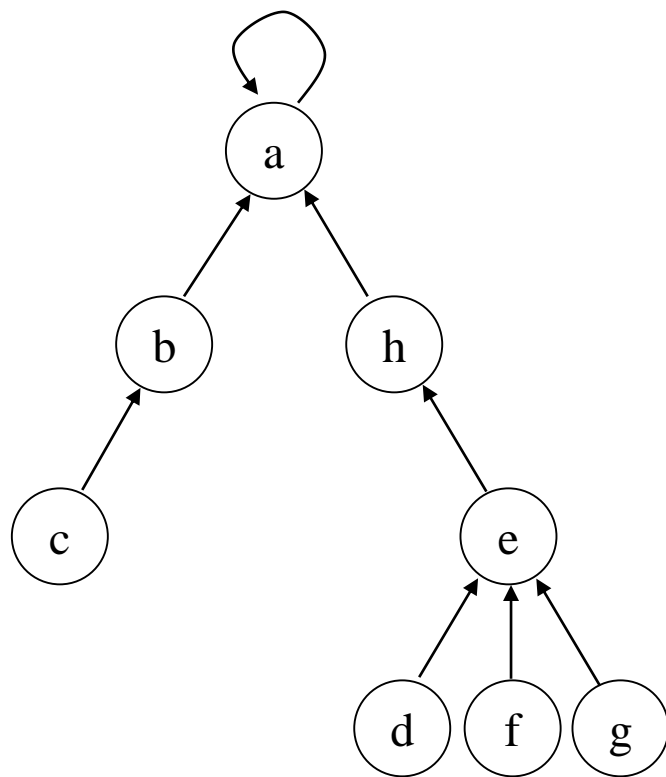
m번의 Make, Union, Find 중 n번이 **Union** 이라면

이들의 총 수행 시간은 $O(m + n \log n)$ 이다.

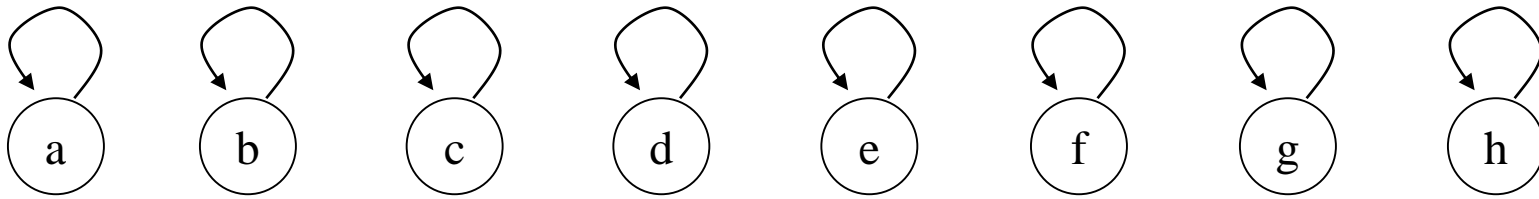
3. 트리를 이용한 집합의 처리

- ❖ 같은 집합의 원소들은 하나의 트리로 관리한다
 - 자식 노드가 부모 노드를 가리킨다
- ❖ 트리의 루트를 집합의 대표 원소로 삼는다

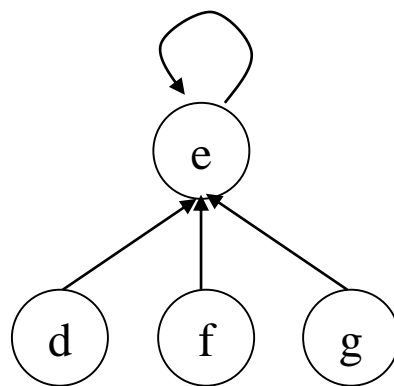
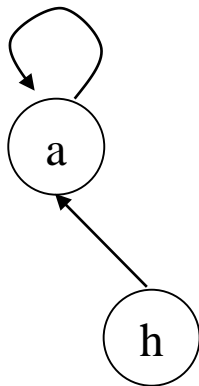
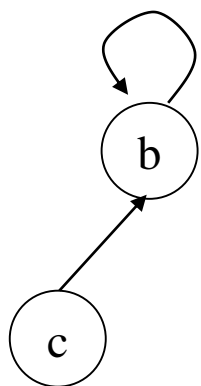
트리를 이용한 집합의 표현 예



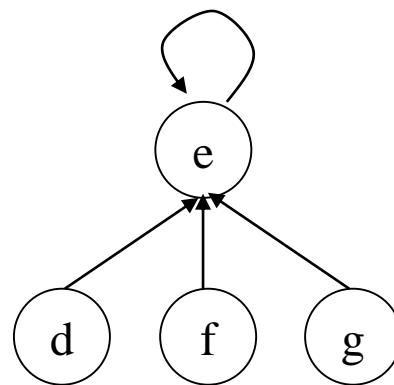
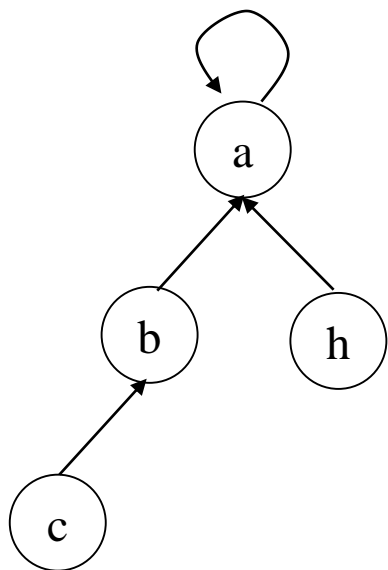
하나의 원소로 이루어진 8개의 독립 집합



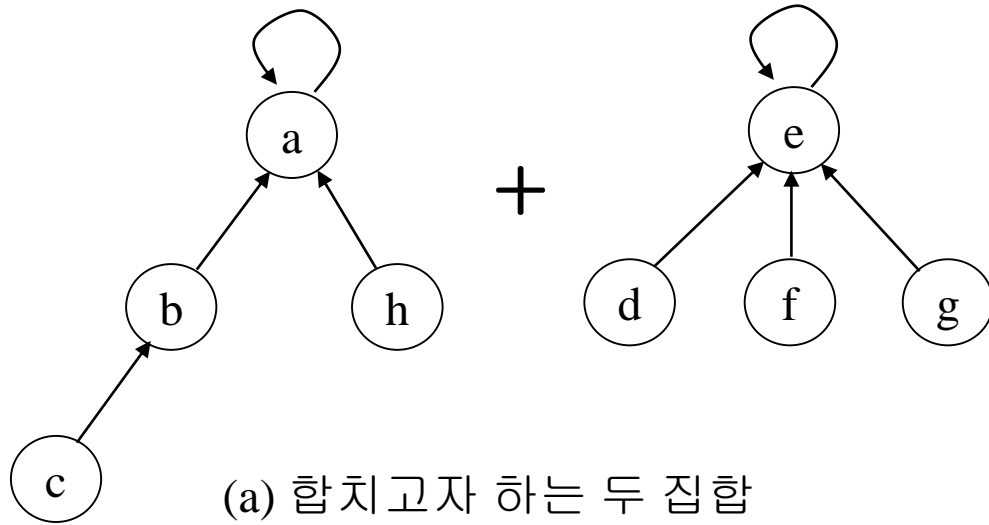
일부를 **union** 하여 세 개의 독립 집합으로



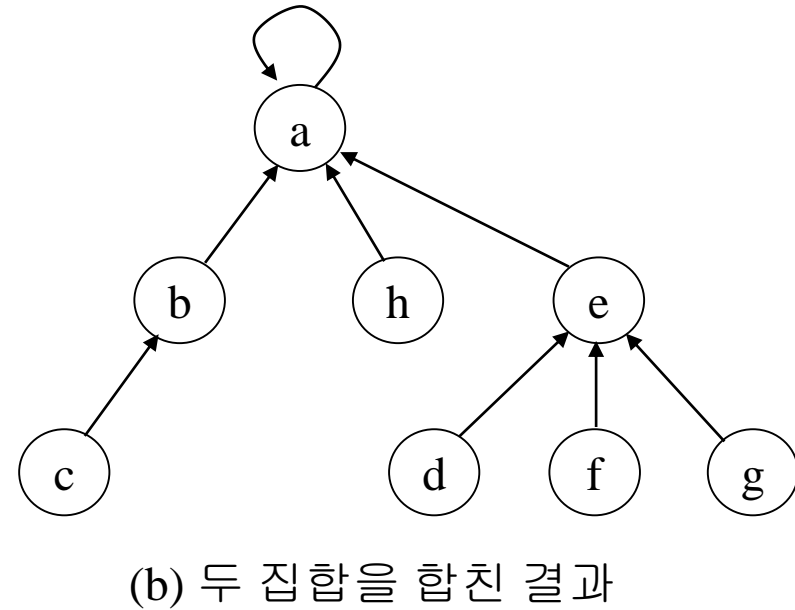
일부를 **union** 하여 두 개의 독립 집합으로



두 독립 집합의 합집합 만들기



=



연산의 효율을 높이는 방법

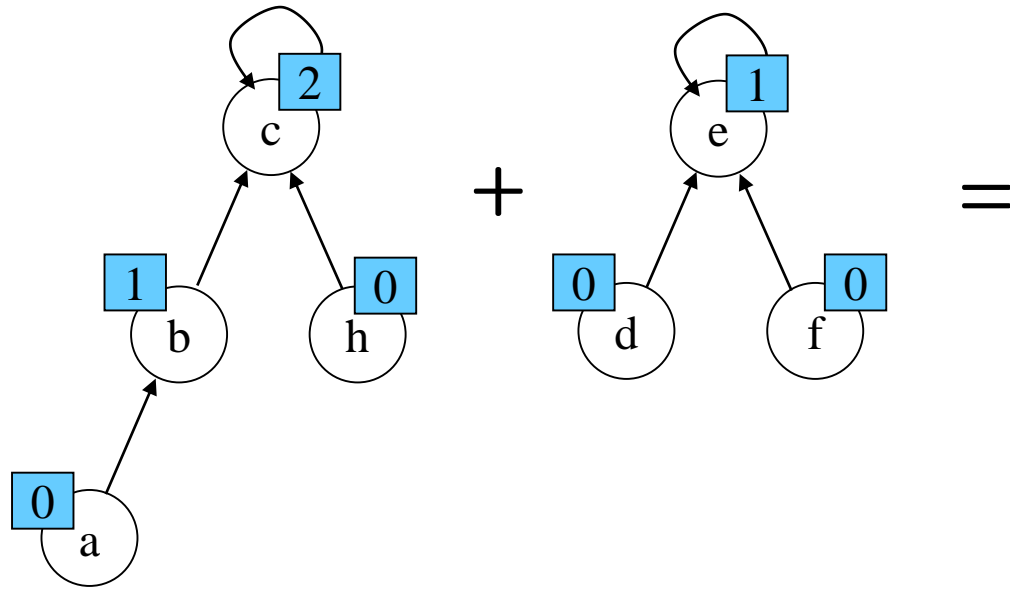
❖ 랭크를 이용한 Union

- 각 노드는 자신을 루트로 하는 서브 트리의 높이를 랭크(Rank)라는 이름으로 저장한다.
- 두 집합을 합칠 때 랭크가 낮은 집합을 랭크가 높은 집합에 붙인다.

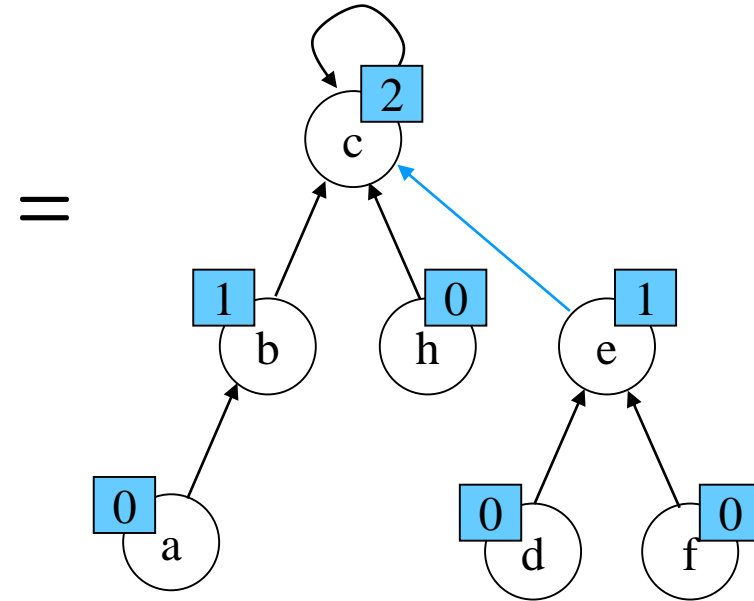
❖ 경로 압축

- Find를 행하는 과정에서 만나는 모든 노드들이 직접 루트를 가리키도록 포인터(부모 노드)를 바꾸어 준다

랭크를 이용한 Union의 예

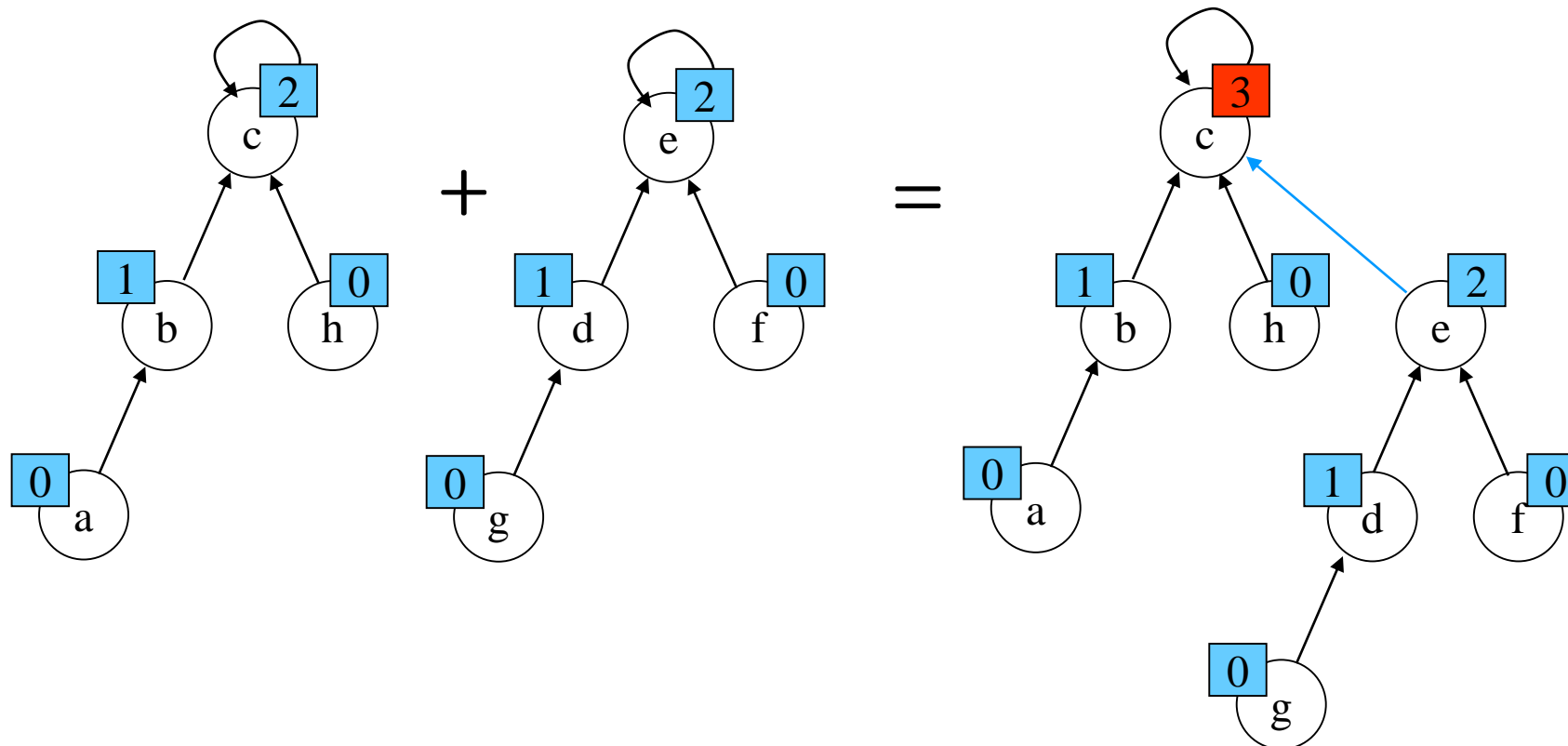


(a) 합치고자 하는 두 집합



(b) 두 집합을 합친 결과

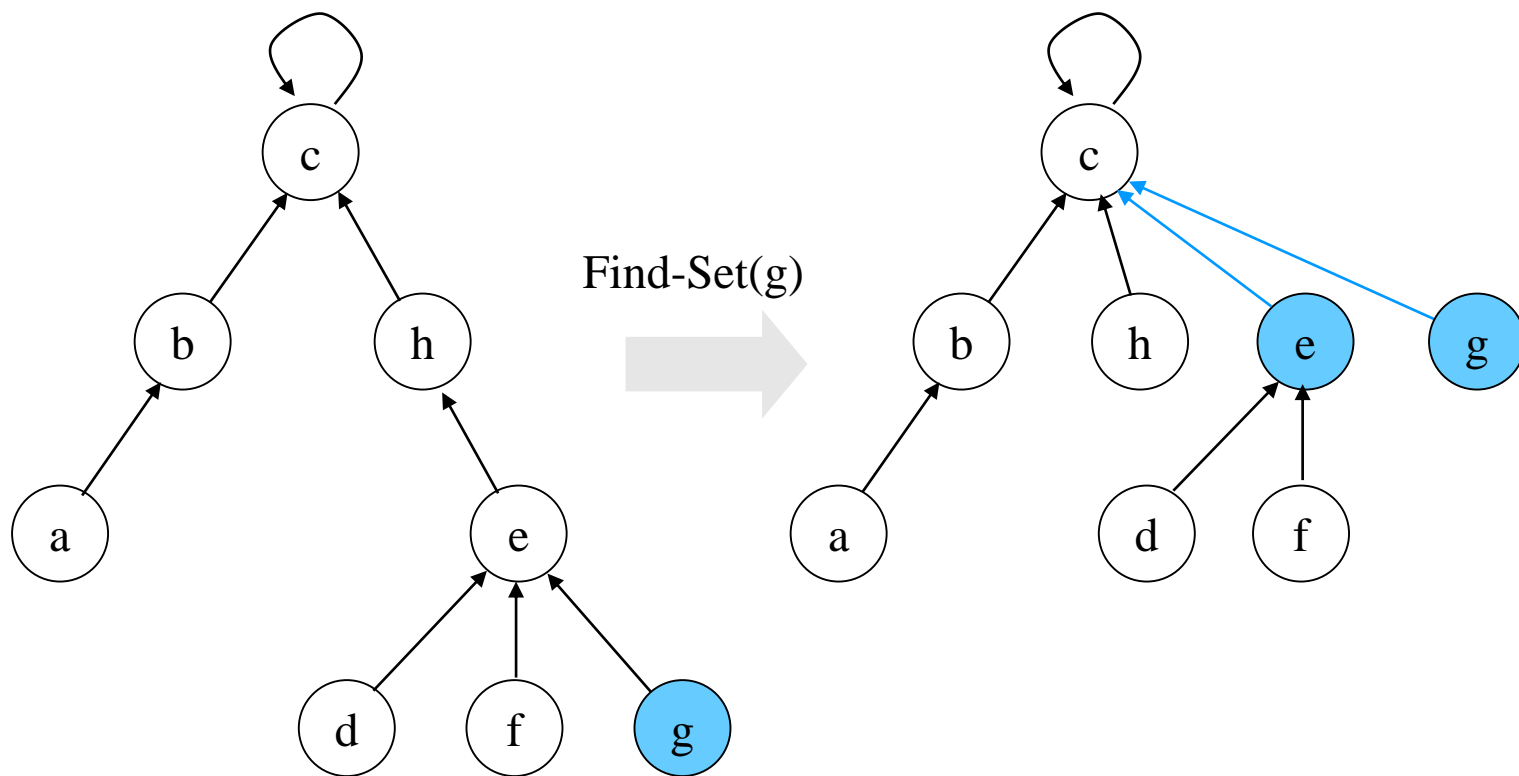
랭크를 이용한 Union에서 랭크가 증가하는 예



(a) 합치고자 하는 두 집합

(b) 두 집합을 합친 결과

경로압축의 예



랭크를 이용한 Union과 경로 압축을 이용한 Find

```
Find(x) {                                     // 노드 x가 속한 트리의 루트 노드를 반환한다.
    if (group [x]  $\neq$  x) then group [x]  $\leftarrow$  Find (group [x]);
    return group[x];
}
```

```
Union(x, y) {                                 // 노드 x가 속한 집합과 노드 y가 속한 집합을 합친다
    x  $\leftarrow$  Find (x);  y  $\leftarrow$  Find (y);
    if (x == y) return;
    if (rank[x] > rank[y]) then group [y]  $\leftarrow$  x ;
    else {
        group [x]  $\leftarrow$  y ;
        if (rank[x] == rank[y]) then rank[y]  $\leftarrow$  rank[y] + 1;
    }
}
```

수행시간

[정리 2]

트리를 이용해 표현되는 배타적 집합에서
랭크를 이용한 **Union**과 경로 압축을 이용한 **Find**를
동시에 사용하면, m번의 Make, Union, Find 중
n번이 **Union** 일 때 이들의 수행 시간은 $O(m + n \log^* n)$ 이다.

*** Quiz ***

$\log^* n$ 의
의미는?

수행시간

[정리 2]

트리를 이용해 표현되는 배타적 집합에서

랭크를 이용한 **Union**과 경로 압축을 이용한 **Find**를

동시에 사용하면, m번의 Make, Union, Find 중

n번이 **Union** 일 때 이들의 수행 시간은 $O(m + n \log^* n)$ 이다.

$$\log^* n = \min \{k : \underbrace{\log \log \dots \log n}_{k\text{개}} \leq 1\}$$

사실상 상수시간임

어떤 수가 1이하가 될
때까지 반복적으로 log를
취한 횟수

4. 관련 문제

- ❖ [jungol_1060 : 최소비용신장트리](#)
- ❖ [jungol_1863 : 종교](#)
- ❖ [jungol_2467 : 비용](#)
- ❖ [jungol_2997 : 트리\(중등\)](#)
- ❖ [jungol_3000 : 트리\(고등\)](#)

감사합니다.^^