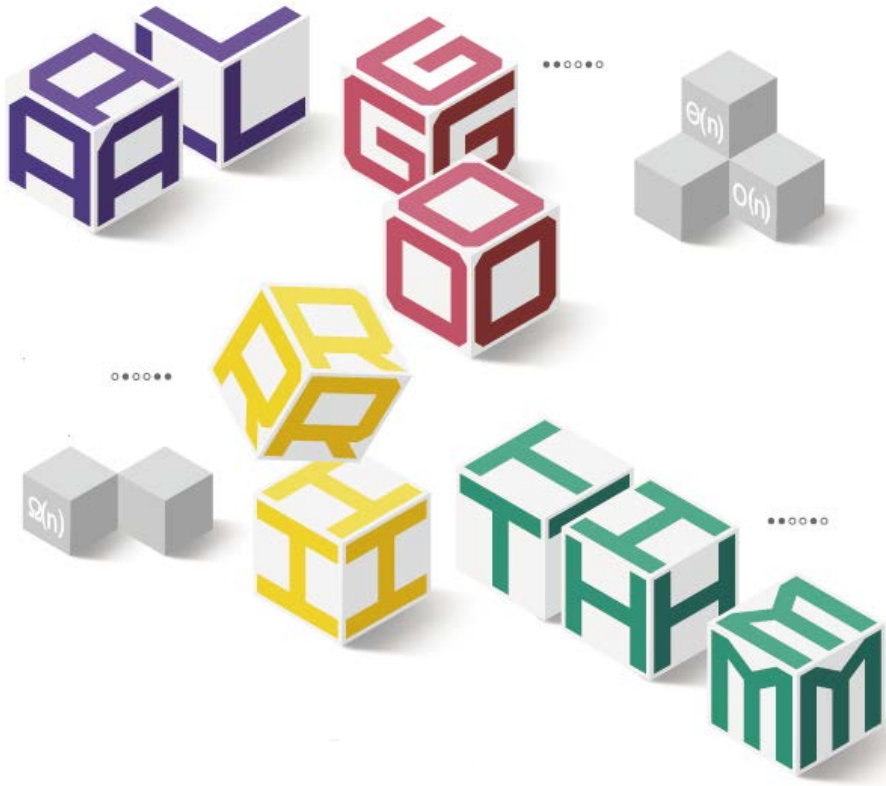


관계 중심의 사고법

쉽게 배우는 알고리즘

8장. 상호 배타적 집합의 처리

8장. 상호 배타적 집합의 처리

얼마 전 나는 학술회의 준비차 『다윈』을 읽었다.
읽으면서 그동안 읽지 않기를 잘했다는 생각이 들었다.
다른 때 『다윈』을 읽었더라면
여전히 이해할 수 없었을 것이기 때문이다.
… 결국, 읽을 준비가 되었을 때 읽어야 한다는 것이다.

-로저 생크

학습목표

- 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.

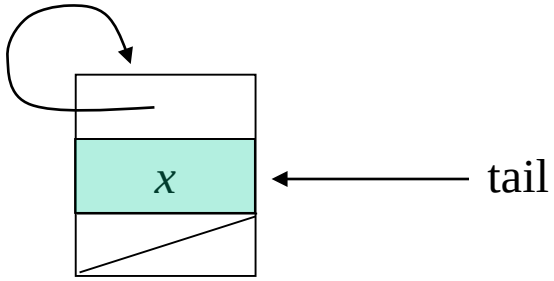
집합의 처리

- 이 장에서는 상호배타적 집합만을 대상으로 한다
- 그러므로 교집합은 없다
- 지원할 연산
  - Make-Set(x): 원소 x로만 이루어진 집합을 만든다
  - Find-Set(x): 원소 x를 가지고 있는 집합의 대표원소를 return 한다
  - Union(x, y): 원소 x를 가진 집합과 원소 y를 가진 집합을 합한다
- 연결 리스트를 이용하는 방법과 트리를 이용하는 방법을 소개한다

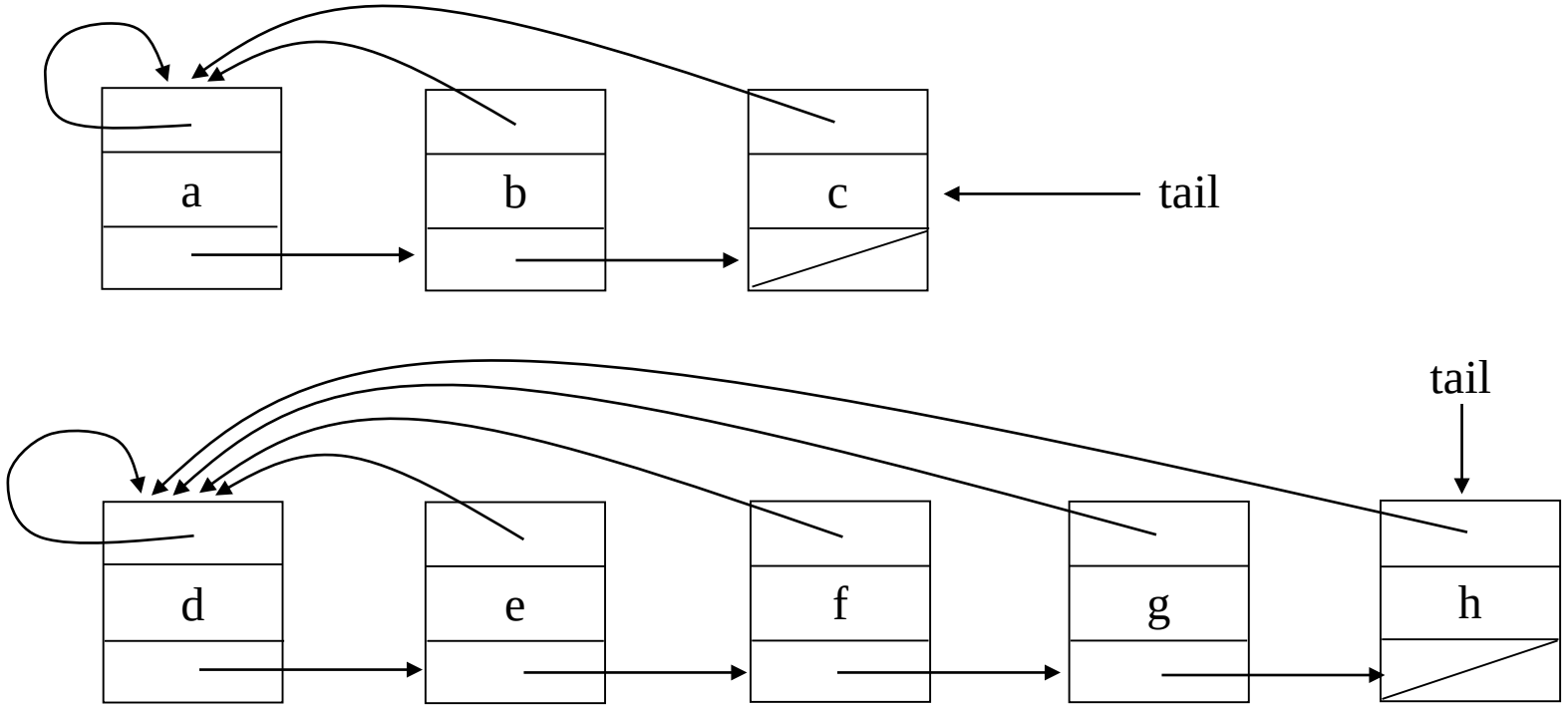
연결 리스트를 이용한 처리

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

하나의 원소로 이루어진 집합

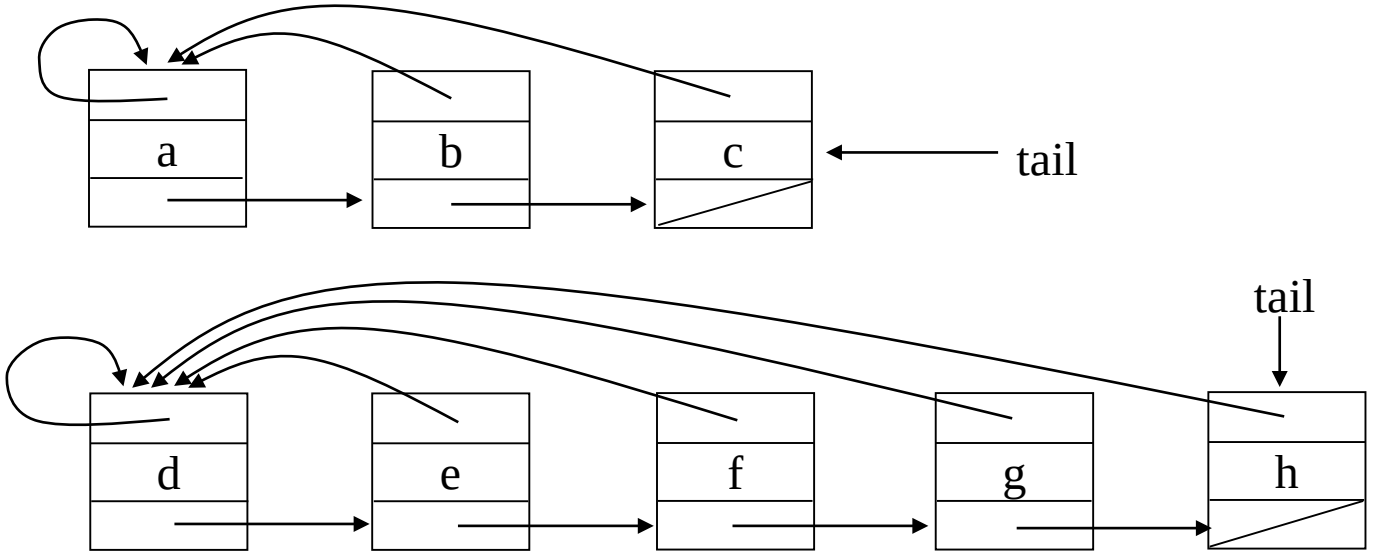


연결 리스트로 된 두 집합

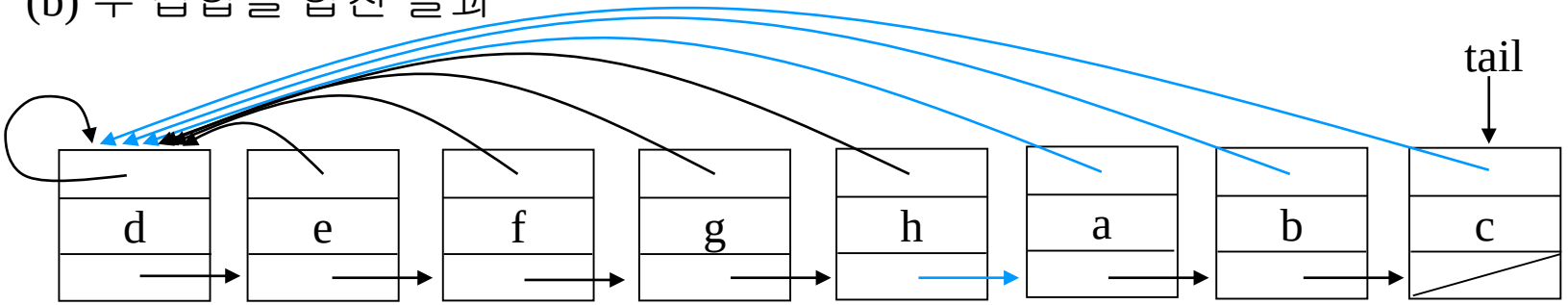


합집합을 만드는 예

(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

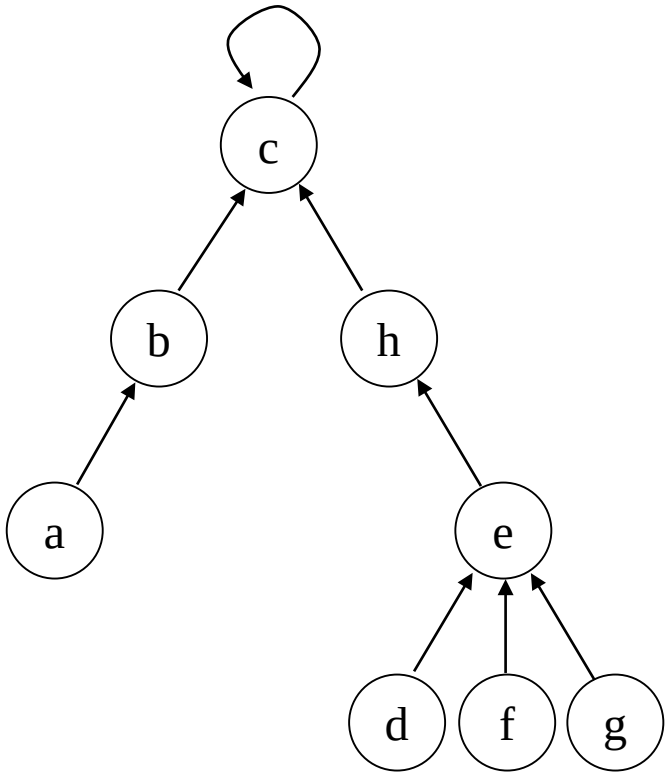
[정리 1]

연결 리스트를 이용해 표현되는 배타적 집합에서 **무게를 고려한 Union**을 사용할 때, m번의 Make-Set, Union, Find-Set 중 n번이 Make-Set이라면 이들의 총 수행 시간은 $O(m + n\log n)$이다.

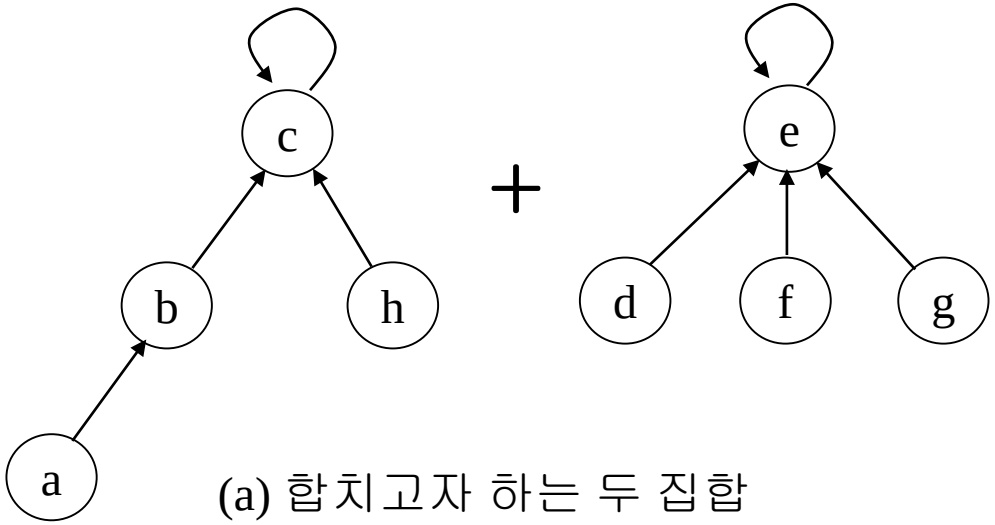
트리를 이용한 집합의 처리

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

트리를 이용한 집합 표현의 예

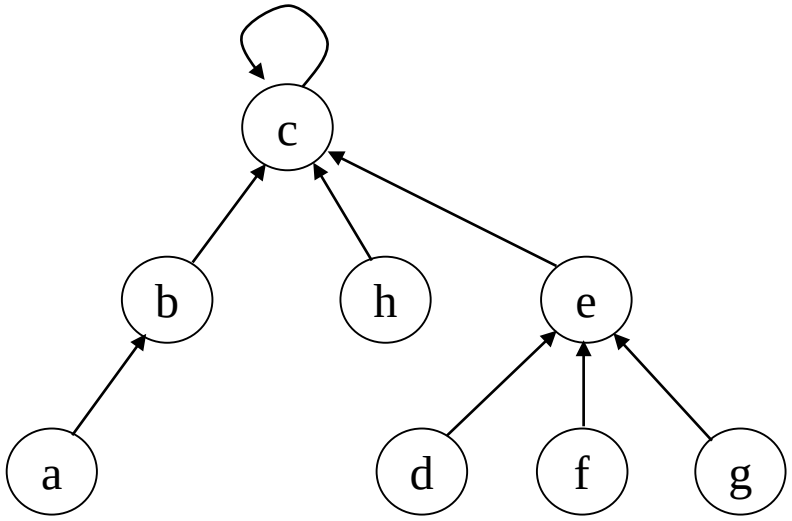


두 집합의 합집합



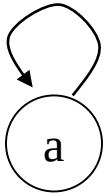
(a) 합치고자 하는 두 집합

=



(b) 두 집합을 합친 결과

하나의 원소로 이루어진 집합



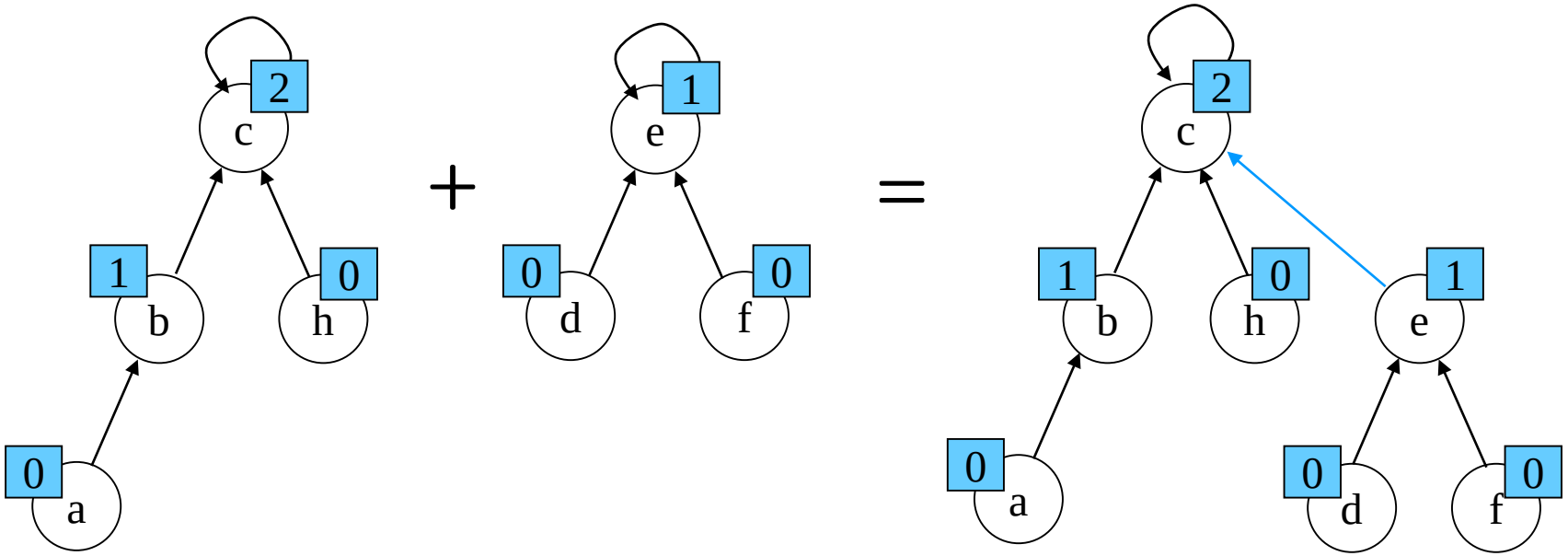
트리를 이용한 집합 처리 알고리즘

```
Make-Set(x)          ▷ 노드 x를 유일한 원소로 하는 집합을 만든다.
{
    p[x] ← x ;
}

Union(x, y)          ▷ 노드 x가 속한 집합과 노드 y가 속한 집합을 합친다
{
    p[Find-Set(y)] ← Find-Set(x) ;
}

Find-Set(x)          ▷ 노드 x가 속한 집합을 알아낸다.
                       노드 x가 속한 트리의 루트 노드를 리턴한다.
{
    if (x  = p[x])
        then return x ;
        else return Find-Set(p[x]) ;
}
```

연산의 효율을 높이는 방법

- 랭크를 이용한 Union
  - 각 노드는 자신을 루트로 하는 서브트리의 높이를 랭크Rank라는 이름으로 저장한다
  - 두 집합을 합칠 때 랭크가 낮은 집합을 랭크가 높은 집합에 붙인다
- 경로압축
  - Find-Set을 행하는 과정에서 만나는 모든 노드들이 직접 루트를 가리키도록 포인터를 바꾸어 준다

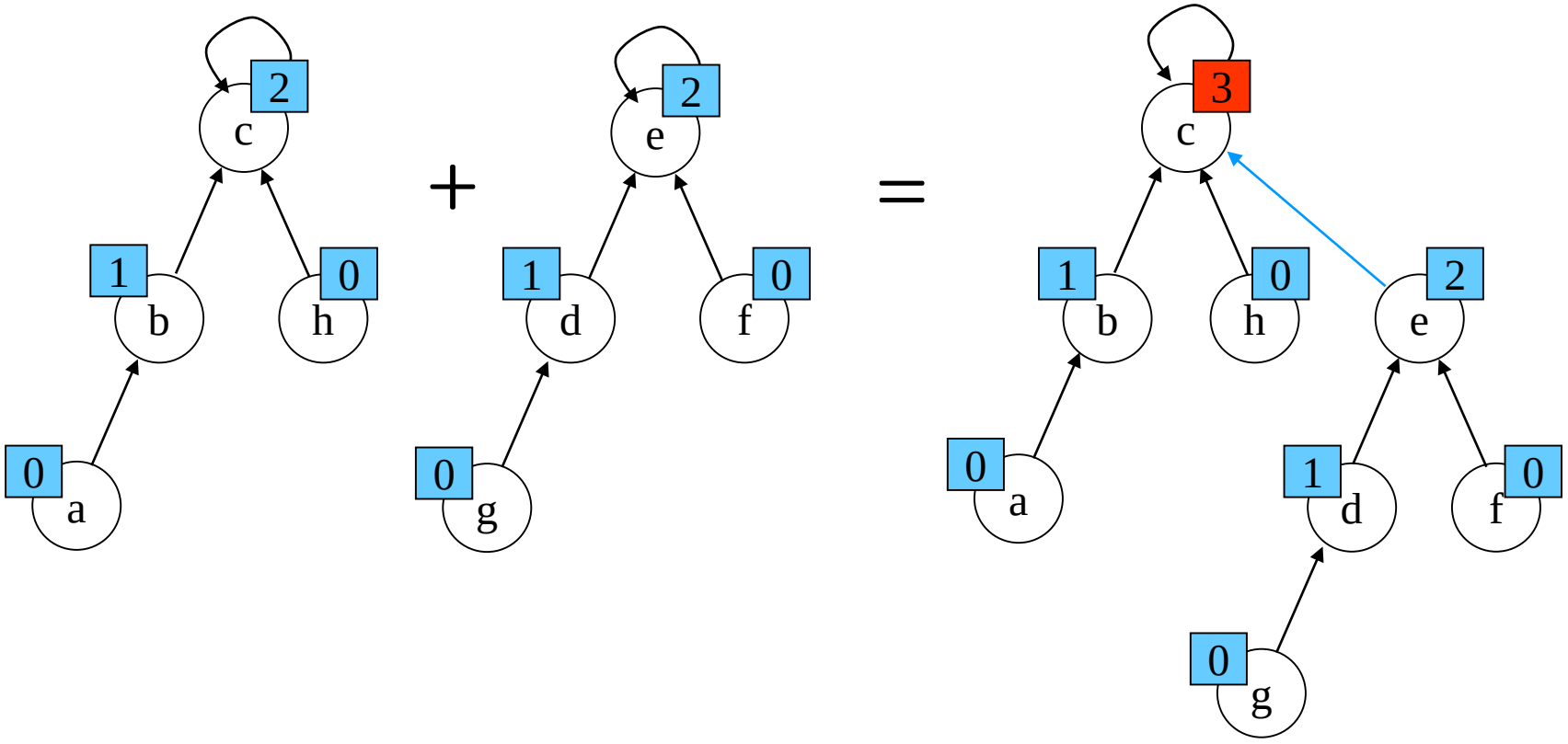
랭크를 이용한 Union의 예



(a) 합치고자 하는 두 집합

(b) 두 집합을 합친 결과

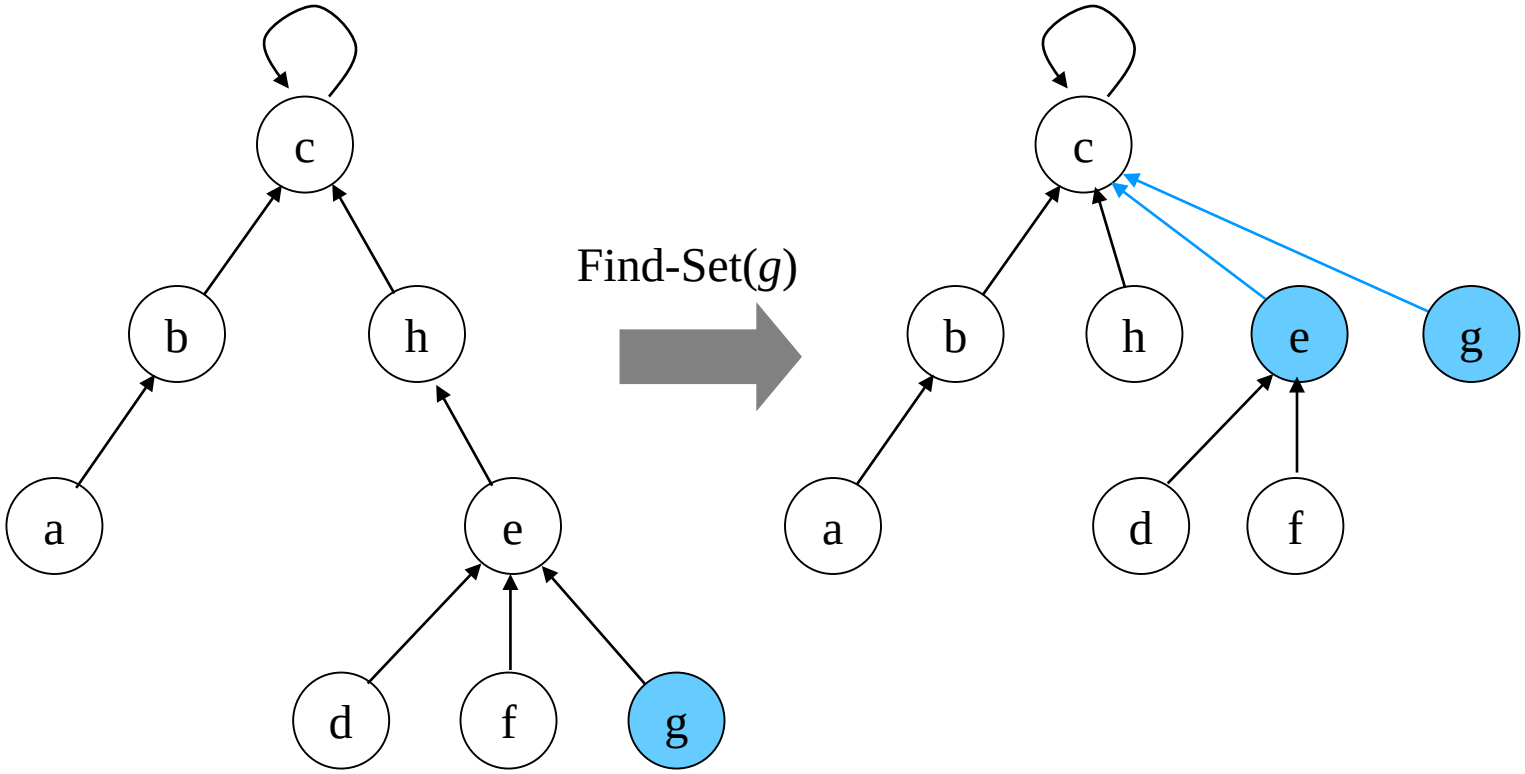
랭크를 이용한 Union에서 랭크가 증가하는 예



(a) 합치고자 하는 두 집합

(b) 두 집합을 합친 결과

경로압축의 예



랭크를 이용한 Union과 Make-Set

```
Make-Set(x)          ▷ 노드 x를 유일한 원소로 하는 집합을 만든다.
{
    p[x] ← x;
    rank[x] ← 0;
}

Union(x, y)          ▷ 노드 x가 속한 집합과 노드 y가 속한 집합을 합한다
{
    x′ ← Find-Set(x);
    y′ ← Find-Set(y);
    if (rank[x′] > rank[y′])
        then p[y′] ← x′ ;
        else {
            p[x′] ← y′ ;
            if (rank[x′] = rank[y′]) then rank[y′] ← rank[y′] + 1;
        }
}
```

경로압축을 이용한 Find-Set

```
Find-Set(x)
▷ 노드 x가 포함된 트리의 루트를 리턴한다.
{
    if (p[x] ≠ x) then p[x] ← Find-Set(p[x]);
    return p[x];
}
```

[정리 5]

트리를 이용해 표현되는 배타적 집합에서 **랭크를 이용한 Union**과 **경로압축을 이용한 Find-Set**을 동시에 사용하면, m번의 Make-Set, Union, Find-Set 중 n번이 Make-Set일 때 이들의 수행 시간은 $O(m \log^* n)$이다.

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

사실상 선형시간임

Thank you