Ch.08 집합의 처리(SET)

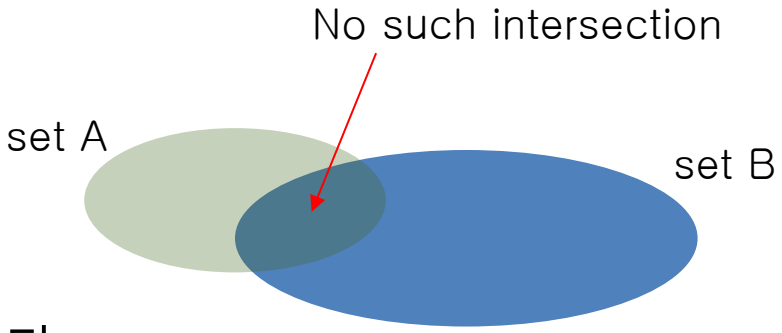
학습목표

- ✓ 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 장의 제한

- 상호배타적 집합만을 대상으로 함. 따라서 교집합은 없다



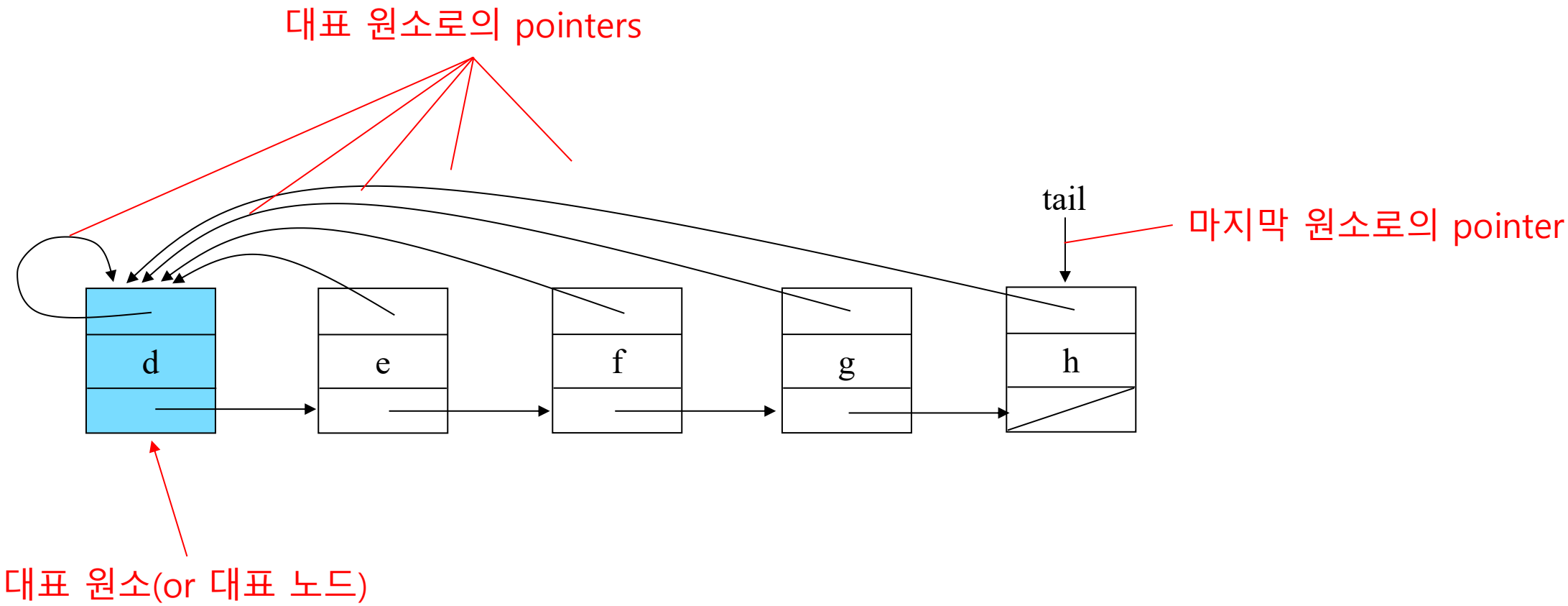
지원할 연산

- Make-Set(x): 원소 x로만 이루어진 집합을 만든다
- Find-Set(x): 원소 x를 가지고 있는 집합을 알아낸다
- Union(x, y): 원소 x를 가진 집합과 원소 y를 가진 집합을 하나로 합친다.

연결 리스트를 이용하는 방법과 트리를 이용하는 방법 소개

연결 리스트를 이용한 집합의 처리

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



연결 리스트를 이용한 집합의 처리

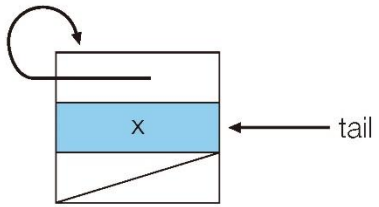
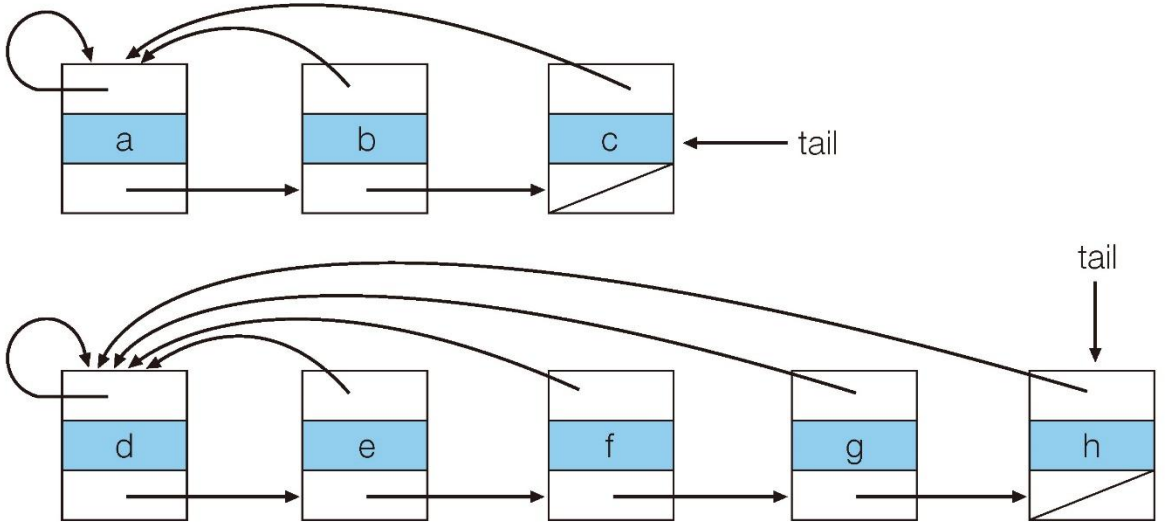
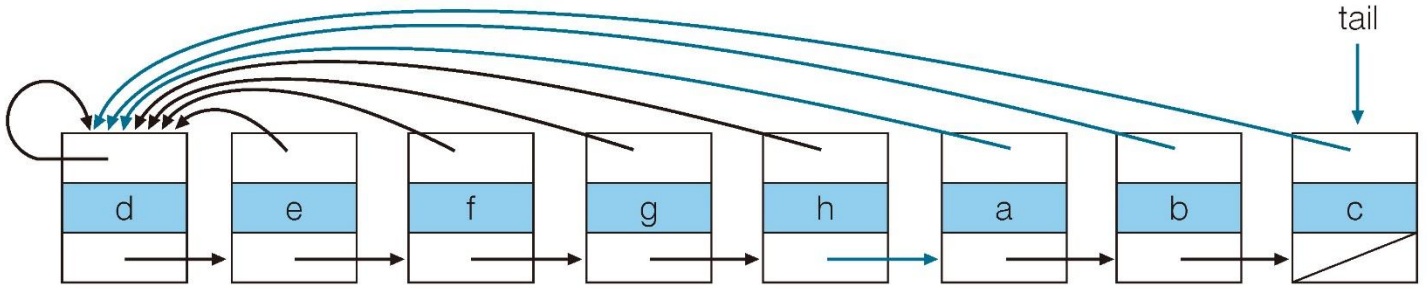


그림 8–1 연결 리스트를 이용하는 표현에서 하나의 원소로 구성된 집합



(a) 합치고자 하는 두 집합



(b) 두 집합을 합친 결과

그림 8–2 연결 리스트를 이용하는 집합의 표현에서 두 집합을 합치는 작업

연결 리스트를 이용한 집합의 처리 수행 시간

정리 8-1

연결 리스트를 이용해 표현되는 배타적 집합에서 무게를 고려한 Union을 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

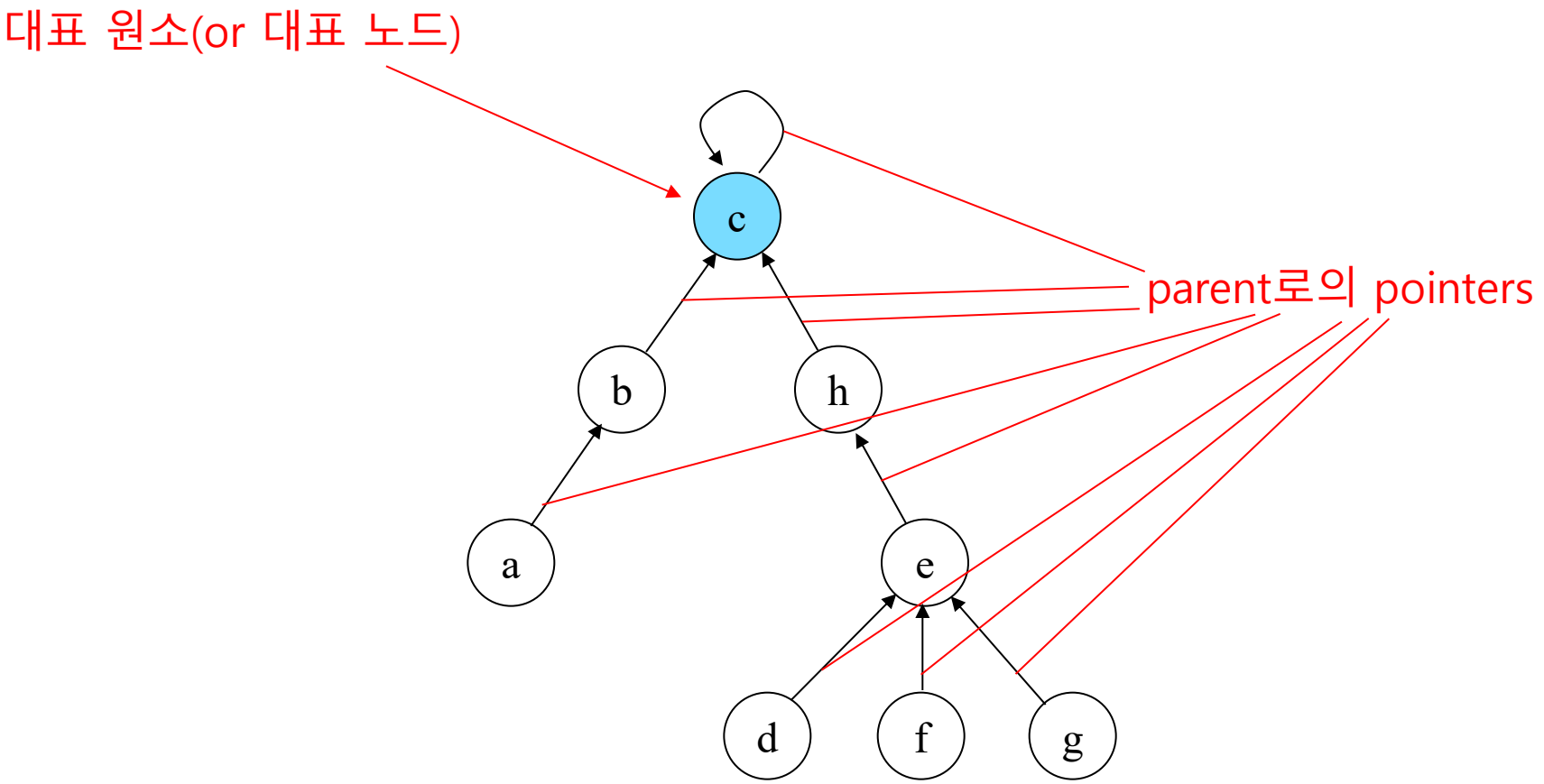


그림 8–3 트리를 이용한 집합 표현의 예

트리를 이용한 집합의 처리

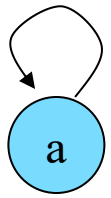
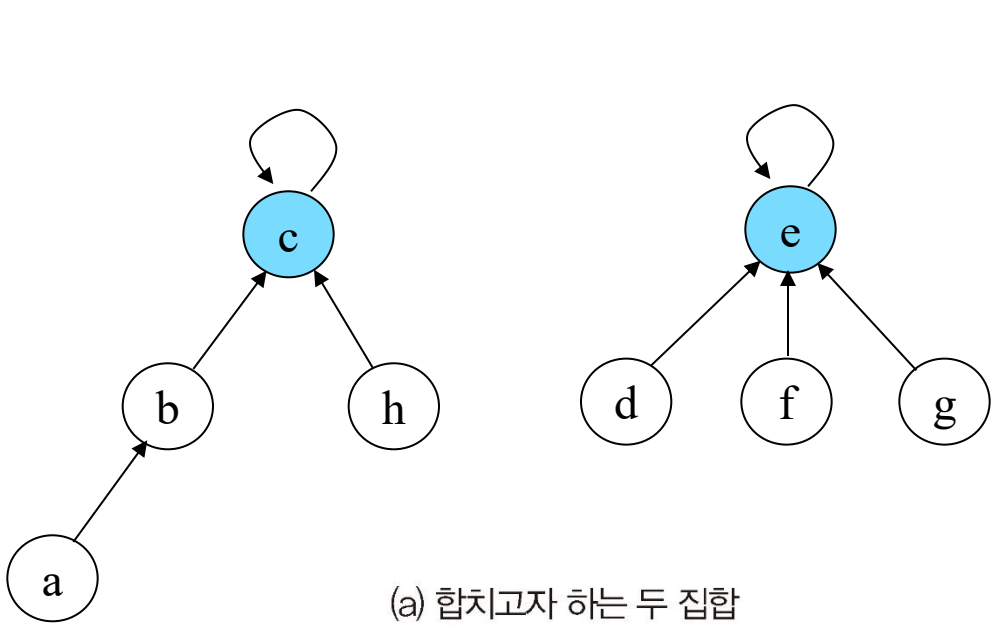
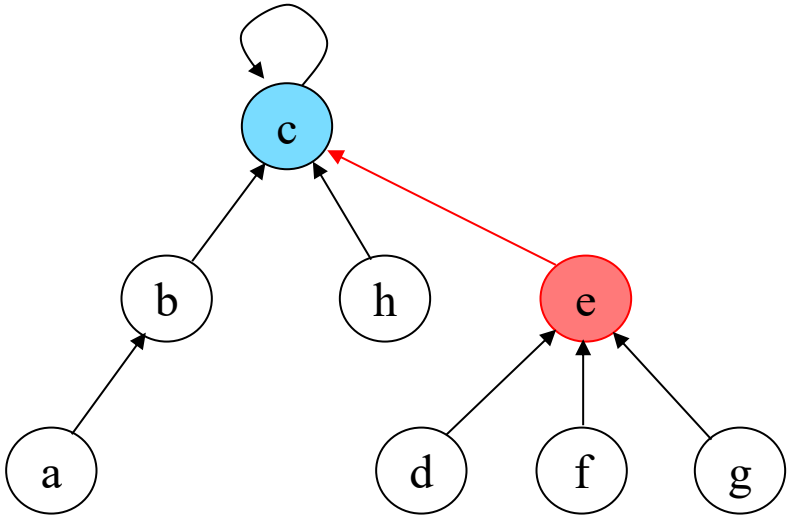


그림 8–5 트리를 이용하는 집합 표현에서 하나의 원소로 구성된 집합



(a) 합치고자 하는 두 집합



(b) 두 집합을 합친 결과

그림 8–4 트리를 이용한 표현에서 두 집합을 합치는 예

트리를 이용한 집합 처리 알고리즘

알고리즘 8-1 트리를 이용한 집합의 처리에서 Make_Set, Union, Find_Set

Make_Set(x):

▷ 노드 x를 유일한 원소로 하는 집합을 만든다.

$x.parent \leftarrow x$

Union(x, y):

▷ 노드 x가 속한 집합과 노드 y가 속한 집합을 합친다.

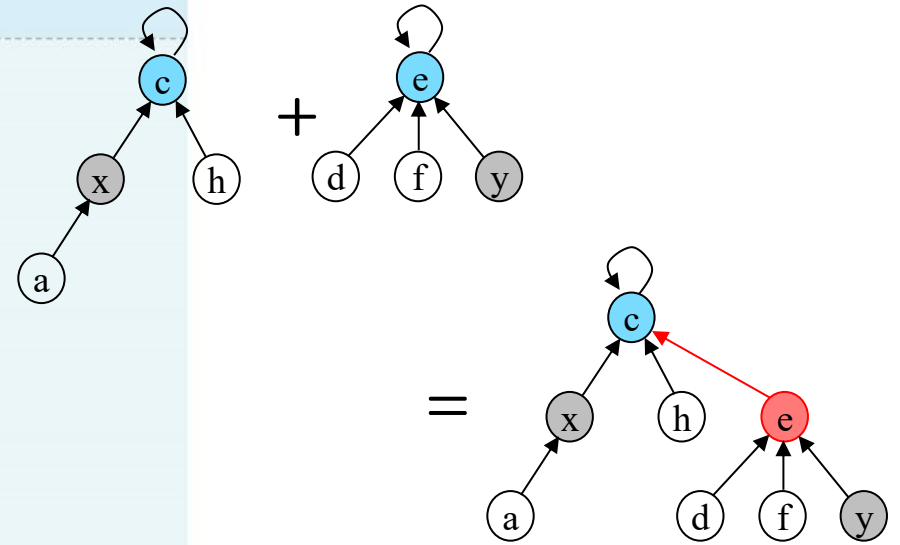
Find_Set(y).$parent$ ← Find_Set(x)

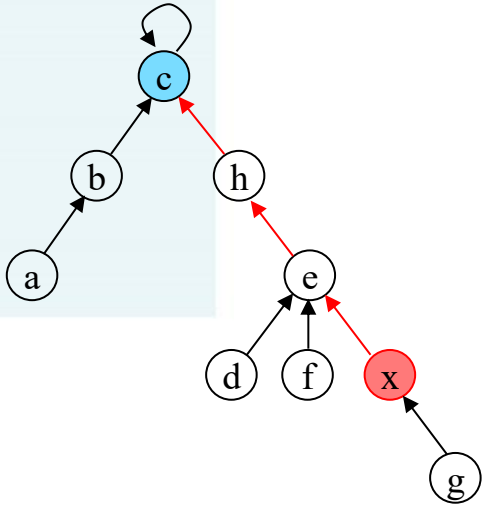
Find_Set(x):

▷ 노드 x가 속한 집합을 알아낸다. 노드 x가 속한 트리의 루트 노드를 리턴한다.

if ($x = x.parent$) return x

else return Find_Set($x.parent$)





트리를 이용한 집합 처리 알고리즘

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

- 경로 압축
  - Find-Set을 행하는 과정에서 만나는 모든 노드들이 직접 루트를 가리키도록 포인터를 바꾸어 준다

트리를 이용한 집합 처리

랭크를 이용한 Union의 예

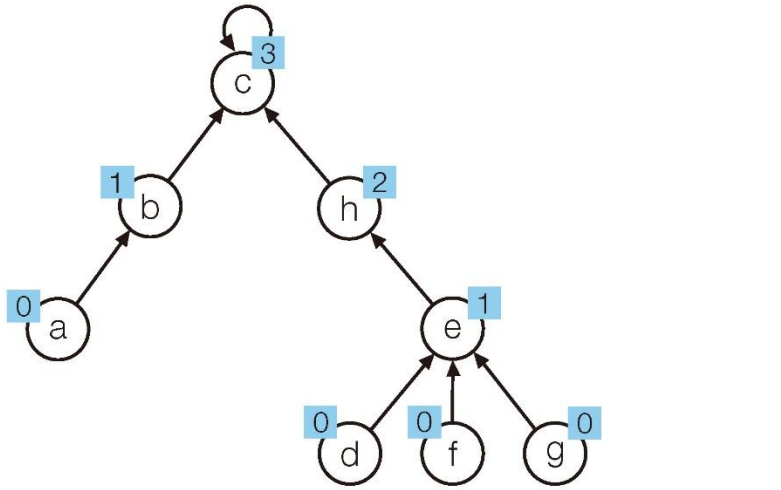
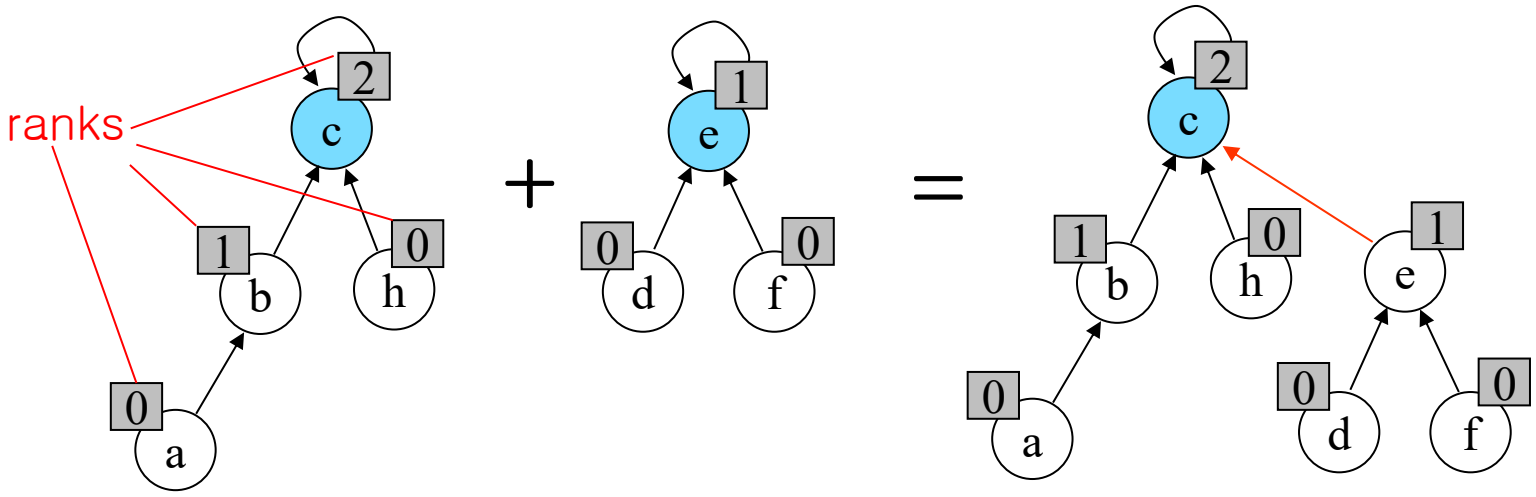


그림 8-6 집합을 표현한 트리에서 각 노드의 랭크가 표시된 예



(a) 합치고자 하는 두 집합

(b) 두 집합을 합친 결과

그림 8-7 랭크를 이용한 Union의 예

트리를 이용한 집합 처리

경로 압축의 예

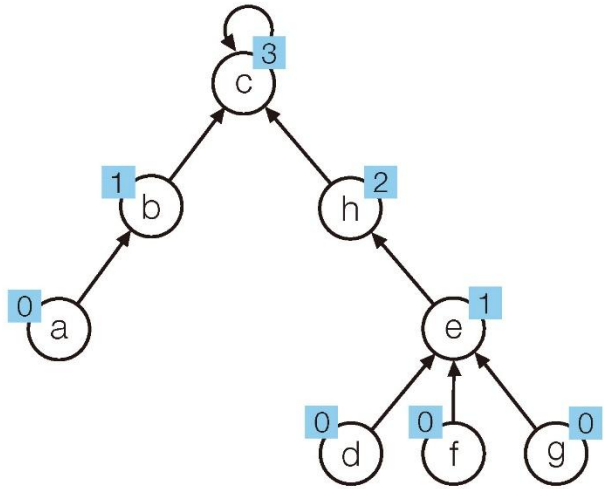


그림 8-6 집합을 표현한 트리에서 각 노드의 랭크가 표시된 예

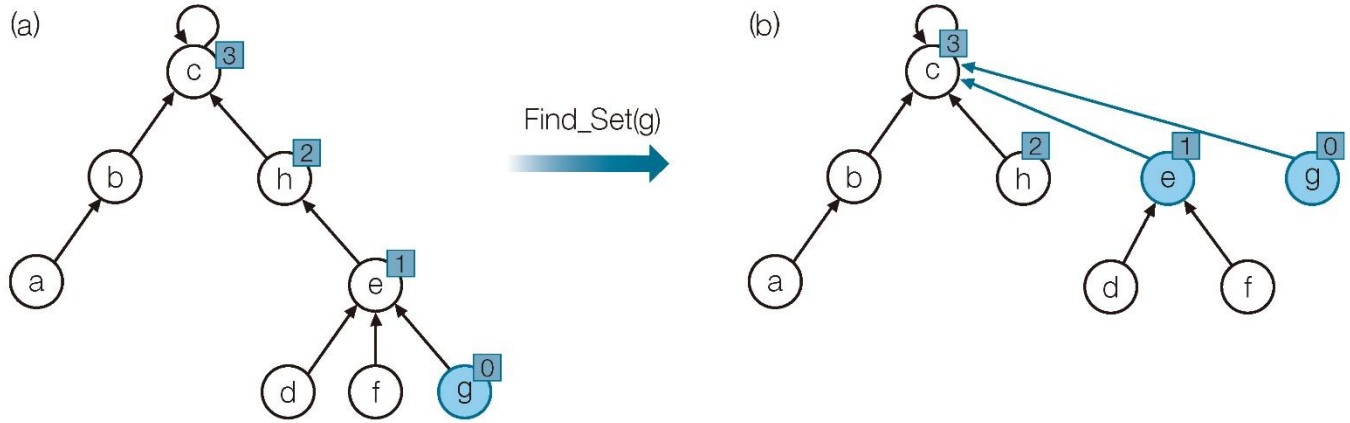


그림 8-8 트리를 이용한 표현에서 경로 압축을 사용하는 Find_Set(g) 직후의 모양

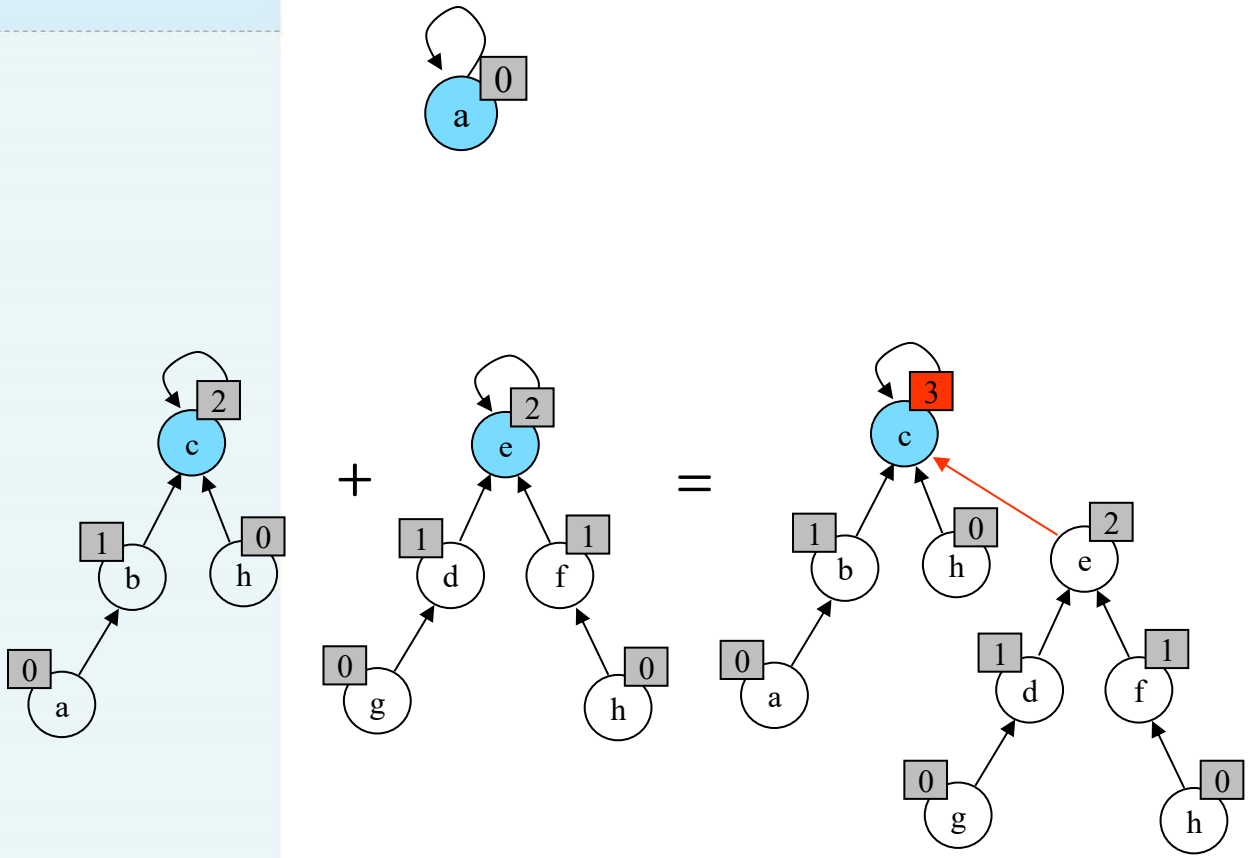
트리를 이용한 집합 처리 알고리즘

랭크를 이용한 Union과 Make-Set

알고리즘 8-2 랭크를 이용한 Union과 Make_Set

```
Make_Set(x):
▷ 노드 x를 유일한 원소로 하는 집합을 만든다.
    x.parent ← x
    x.rank ← 0

Union(x, y):
▷ 노드 x가 속한 집합과 노드 y가 속한 집합을 합친다.
    x' ← Find_Set(x)
    y' ← Find_Set(y)
    if (x'.rank > y'.rank)
        y'.parent ← x'
    else
        x'.parent ← y'
        if (x'.rank = y'.rank) y'.rank ← y'.rank+1
```

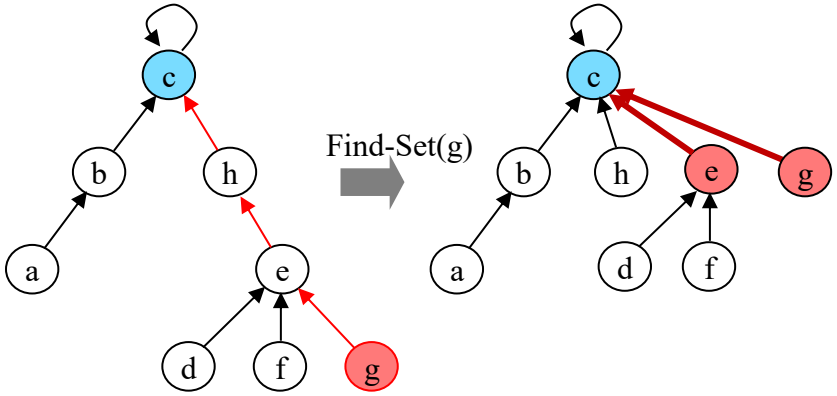


트리를 이용한 집합 처리 알고리즘

경로 압축을 이용한 Find-Set

알고리즘 8-3 경로 압축을 이용한 Find_Set

```
Find_Set(x):
▷ 노드 x가 포함된 트리의 루트를 리턴한다.
    if (x≠x.parent) x.parent ← Find_Set(x.parent)
    return x.parent
```



트리를 이용한 집합 처리 수행 시간

정리 8-2

랭크를 이용한 Union을 사용하면, 랭크가 k인 노드를 대표로 하는 집합의 원소 수는 최소 한 2^k개다.

정리 8-3

랭크를 이용한 Union을 사용하면, 원소 수가 n인 집합을 표현하는 트리에서 모든 노드의 랭크는 $O(\log n)$이다.

트리를 이용한 집합 처리 수행 시간

정리 8-4

트리를 이용해 표현되는 배타적 집합에서 랭크를 이용한 Union을 사용할 때, m번의 Make_Set, Union, Find_Set 중 n번이 Make_Set이라면, 이들의 총 수행 시간은 $O(m\log n)$이다.

정리 8-5

트리를 이용해 표현되는 배타적 집합에서 랭크를 이용한 Union과 경로 압축을 이용한 Find_Set을 동시에 사용하면, m번의 Make_Set, Union, Find_Set 중 n번이 Make_Set일 때 이들의 수행 시간은 $O(m\log^* n)$이다.