알고리즘2 (2024-2)

6. 분리 집합의 표현

국립금오공과대학교 컴퓨터공학과

김 경 수

학습 목표

① 분리 집합의 개념과 트리를 이용하여 분리 집합을 표현하는 방법을 이해하고 이를 설명할 수 있다.

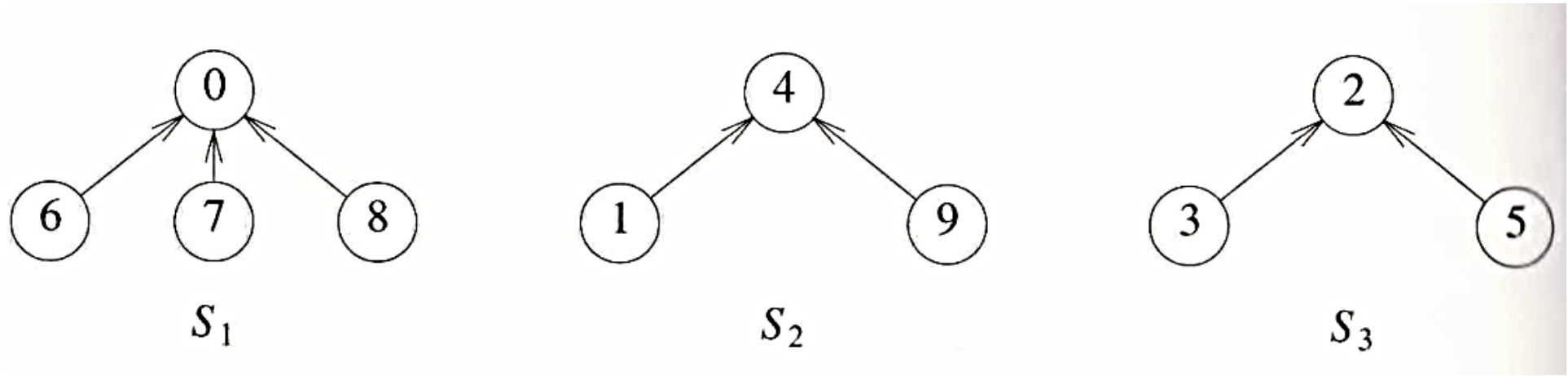
② 트리를 이용하여 분리 집합 표현 시 union과 find 연산의 기능을 이해하고 이를 효율적으로 구현하는 방법을 숙달할 수 있다.

③ 실제 프로그래밍 언어를 이용하여 분리 집합에 대한 union, find 연산 알고리즘을 구현할 수 있다.

분리 집합의 표현 방법

- **분리 집합(disjoint set)**은 서로 어떤 원소도 공통으로 가지지 않는 두 개 이상의 집합을 의미하며, 이들 각각의 집합은 **트리**로 표현할 수 있다.
- 이 때, 각 집합은 해당 집합을 나타내는 트리의 **루트 노드 값**으로 구분한다.
- **분리 집합의 표현 방법의 예**

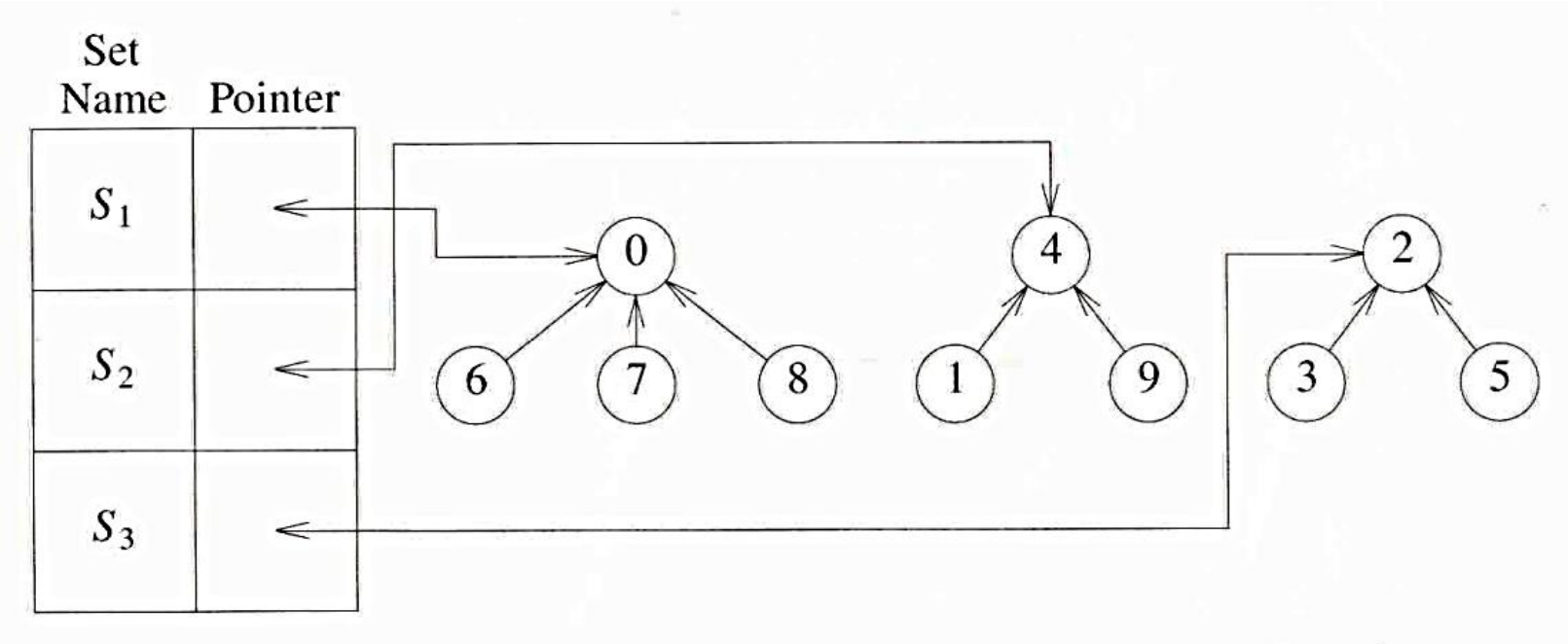
$S_1=\{0, 6, 7, 8\}$, $S_2=\{1, 4, 9\}$, $S_3=\{2, 3, 5\}$



분리 집합의 표현 방법

- **(예)** 분리 집합 S_1, S_2, S_3를 트리로 표현하는 방법

S_1={0, 6, 7, 8}, S_2={1, 4, 9}, S_3={2, 3, 5}

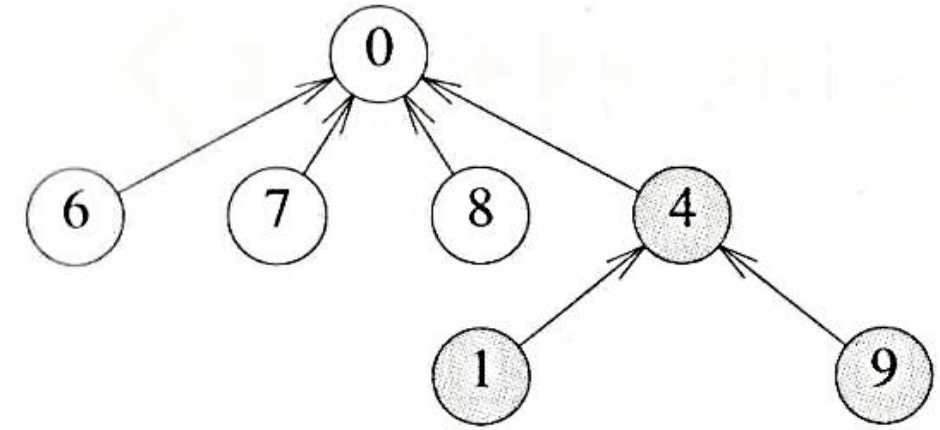


i	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
parent	-1	4	-1	2	-1	2	0	0	0	4

Union 연산

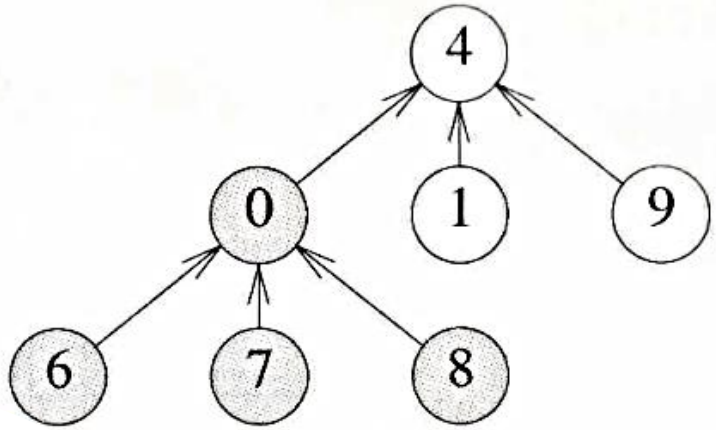
- **union(i,j):** 두 집합 S_i와 S_j의 합집합을 만든다. 이를 위해, 해당 집합을 나타내는 두 트리 중에서 하나를 선택하여 다른 트리의 서브 트리로 만든다.
- **(예)** S_1과 S_2에 대한 합집합 $\mathbf{S_1 \cup S_2}$의 표현 방법

$$\mathbf{S_1=\{0, 6, 7, 8\},\quad S_2=\{1, 4, 9\}}$$



$S_1 \cup S_2$

or

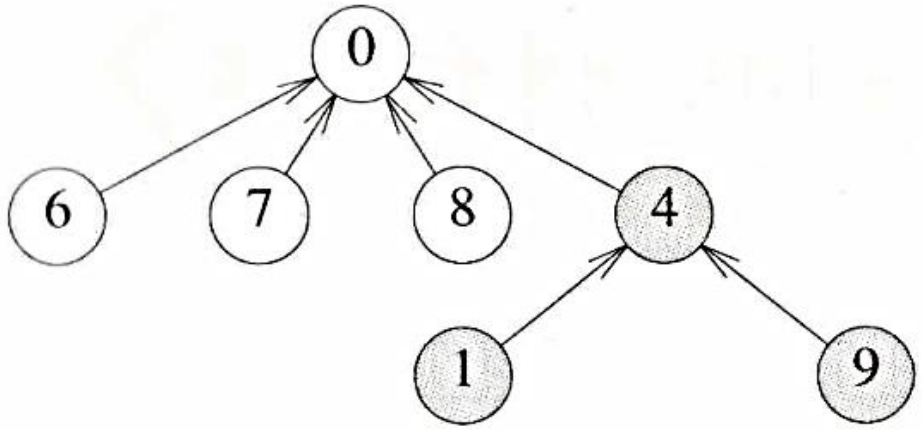


$S_1 \cup S_2$

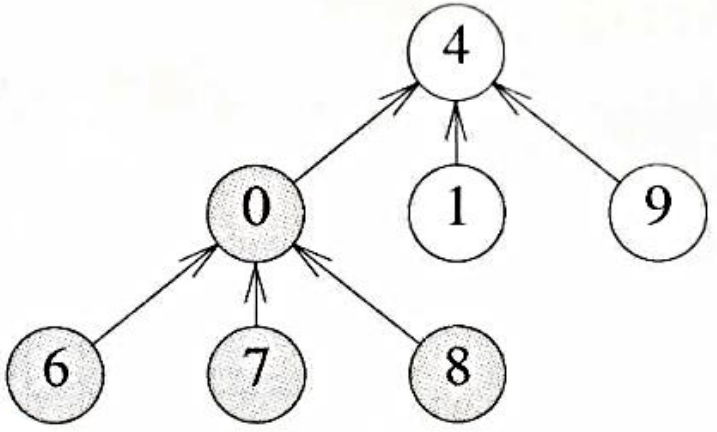
union(0, 4)

Find 연산

- **find(i):** 원소 i를 포함하는 집합을 탐색하고, 해당 집합에 해당하는 트리의 루트 노드를 반환한다.
- **(예)** $S_1 \cup S_2$ 에서 원소 "8"에 대한 **find(8)**의 수행 결과



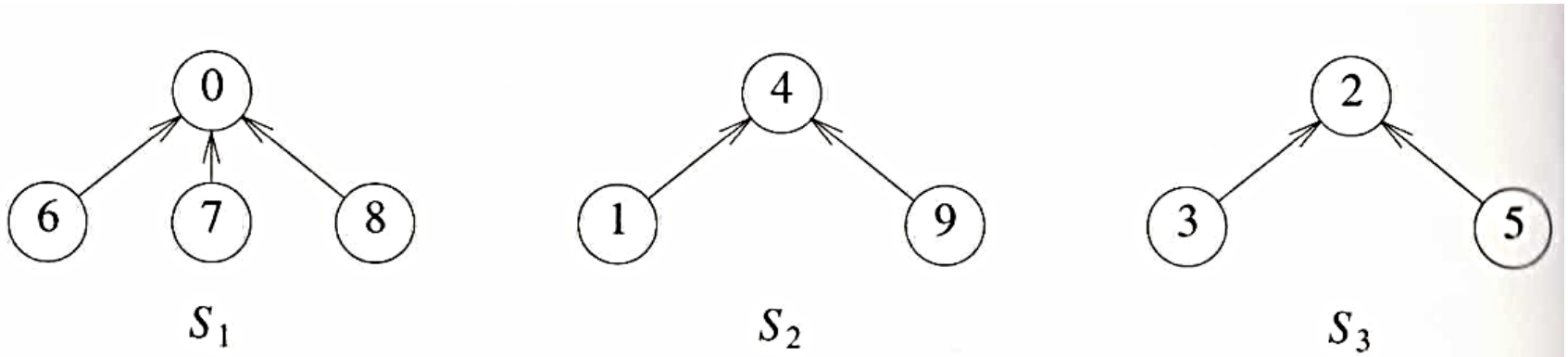
find(8) = 0



find(8) = 4

간단한 Union-Find 연산

```
void union(int i, int j){
      parent[i] = j;
}
int find(int i){
     for( ; parent[i] >= 0; i = parent[i] )  ;
     return i;
}
```

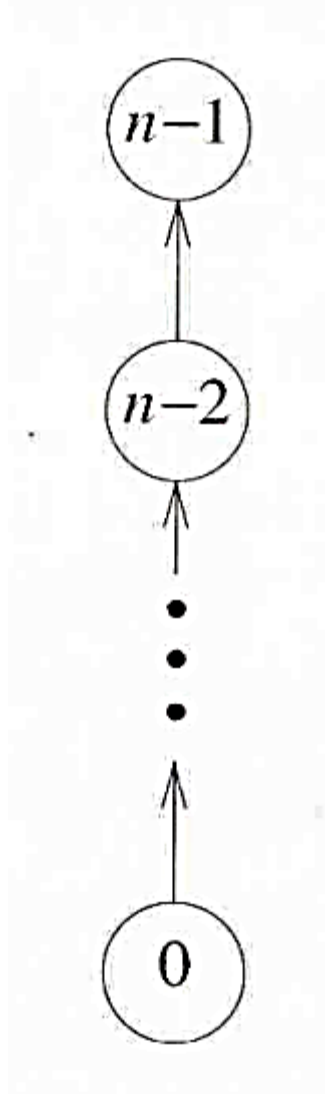


i	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
parent	-1	4	-1	2	-1	2	0	0	0	4

간단한 Union-Find 연산의 문제점

- 앞서 구현한 Union 연산을 연속적으로 수행하면……

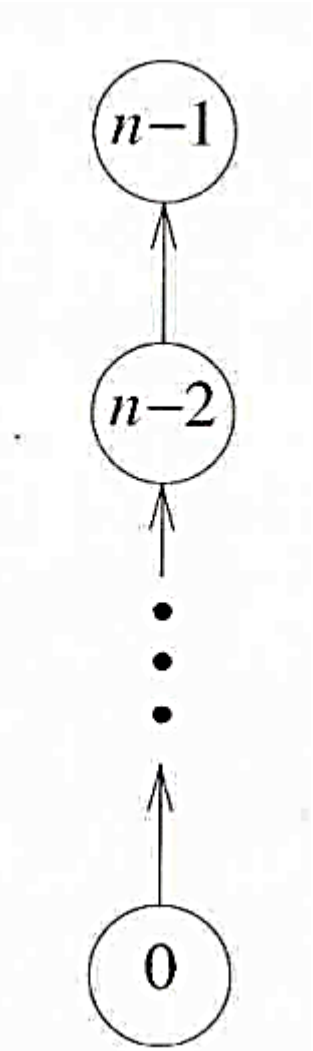
```
union(0, 1)
union(1, 2)
union(2, 3)
union(3, 4)
        ...
union(n-2, n-1)
```



간단한 Union-Find 연산의 문제점

- 앞의 Union 연산을 반복적으로 수행하면 오른쪽 그림과 같은 선형 구조와 같은 트리로 변형되는데 이를 "변질 트리"라 한다.
- 변질 트리에서 n-1번의 find 연산을 수행하는 경우, 시간 복잡도는 $O(n^2)$이다.

```
union(0, 1) 수행 후 find(0)     →   1
union(1, 2) 수행 후 find(0)     →   2
union(2, 3) 수행 후 find(0)     →   3      = n(n-1)/2
            ...
union(n-2, n-1) 수행 후 find(0) → n-1
```



Union 연산의 개선

- **union(i, j) 연산의 개선을 위한 가중 규칙**
  - ➢루트 노드 i를 가진 트리의 노드 수가 루트 노드 j를 가진 트리의 노드 수보다 적으면 노드 j를 노드 i의 부모로 만들고, 그렇지 않으면 노드 i를 노드 j의 부모로 만든다.
  - ➢이를 위해 parent 배열에서 루트 노드에 해당하는 값은 "해당 트리가 포함하는 원소의 총 개수"를 음수로 표현하여 저장한다.

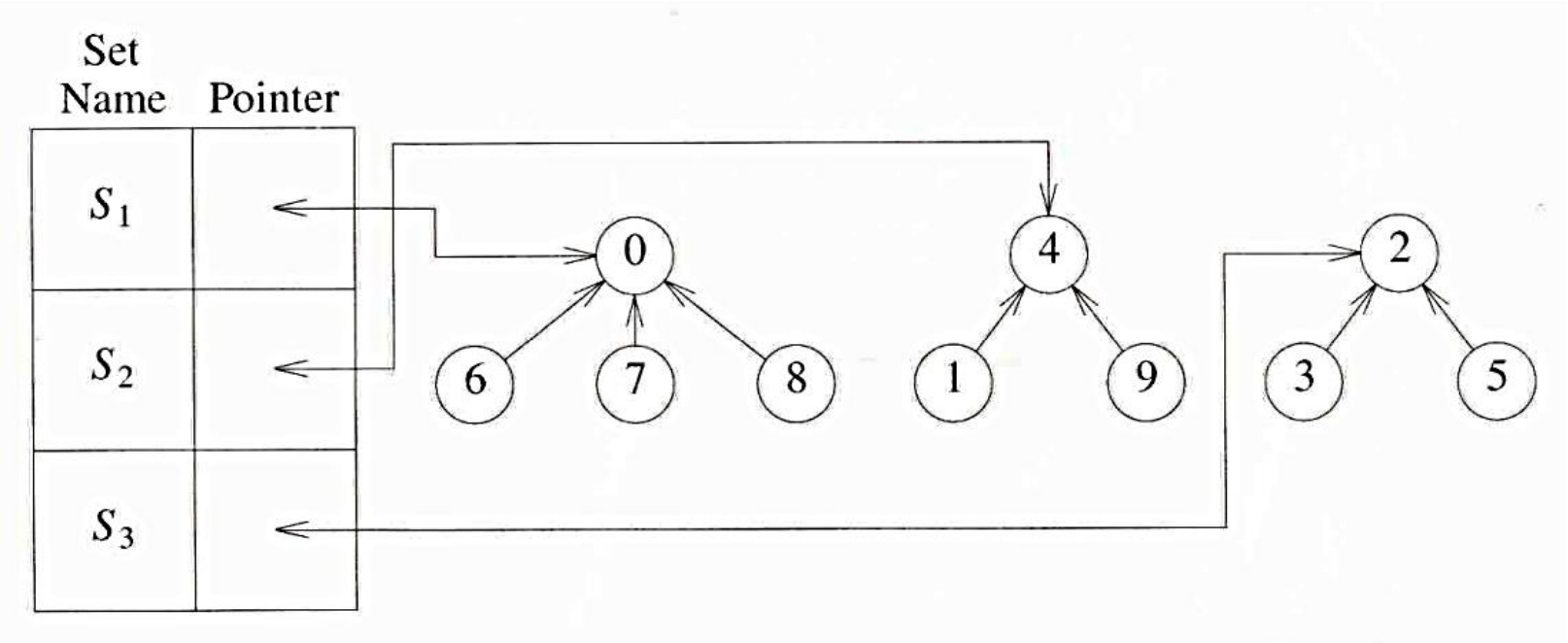
i	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
parent	-1	4	-1	2	-1	2	0	0	0	4

i	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
parent	-4	4	-3	2	-3	2	0	0	0	4

Union 연산의 개선

- **(예제)** 분리 집합 S_1, S_2, S_3에 대한 개선된 데이터 표현 방법

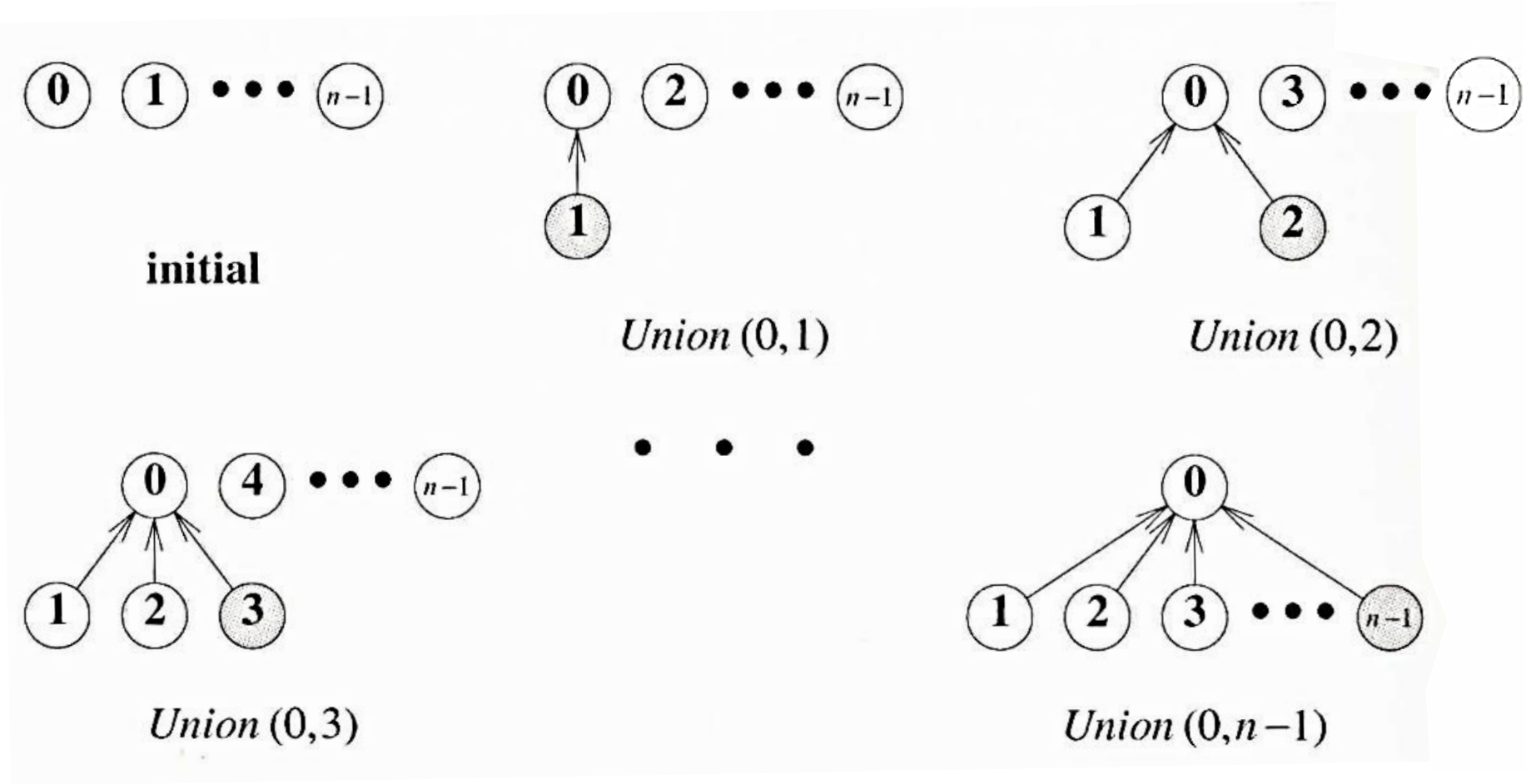
S_1={0, 6, 7, 8}, S_2={1, 4, 9}, S_3={2, 3, 5}



i	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
parent	**-4**	4	**-3**	2	**-3**	2	0	0	0	4

Union 연산의 개선

- 가중 규칙을 적용하였을 때 union 연산 시 구축되는 트리의 예

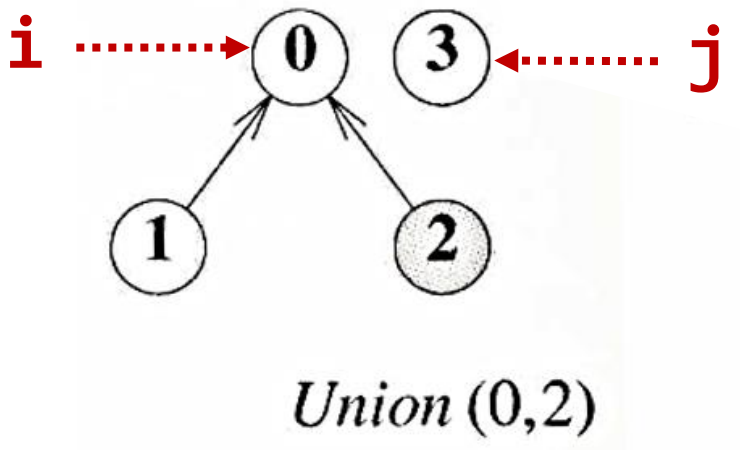


Union 연산의 개선

```
void weighted_union(int i, int j){
        int temp = parent[i] + parent[j];
        if( -parent[i] < -parent[j] ){
                parent[i] = j;
                parent[j] = temp;
         }else{
                parent[j] = i;
                parent[i] = temp;
         }
}
```

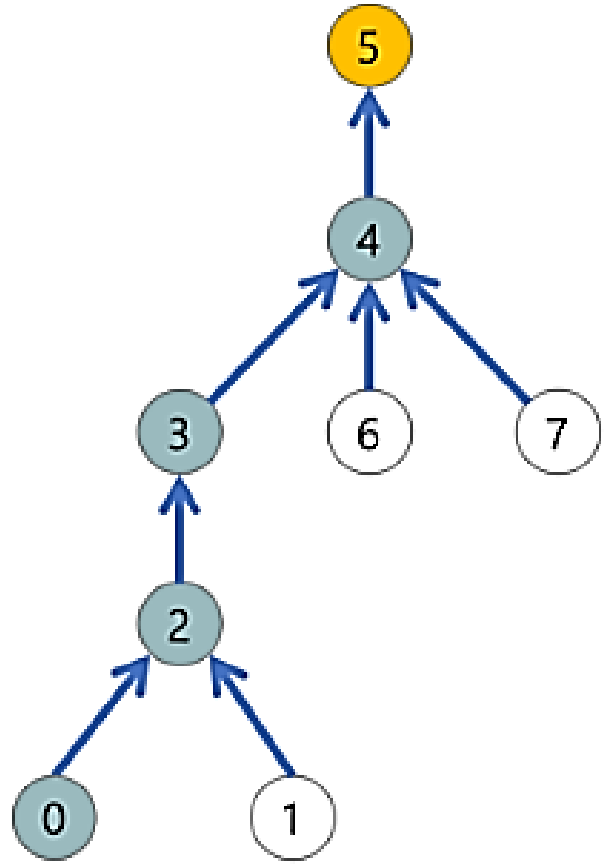
노드 i와 j는 각 집합의 루트 노드를 의미함.

루트 노드 i와 j에 대한 **parent[i]**, **parent[j]**는 각 집합이 포함하는 원소의 총 개수가 **음수**로 저장되어 있음.



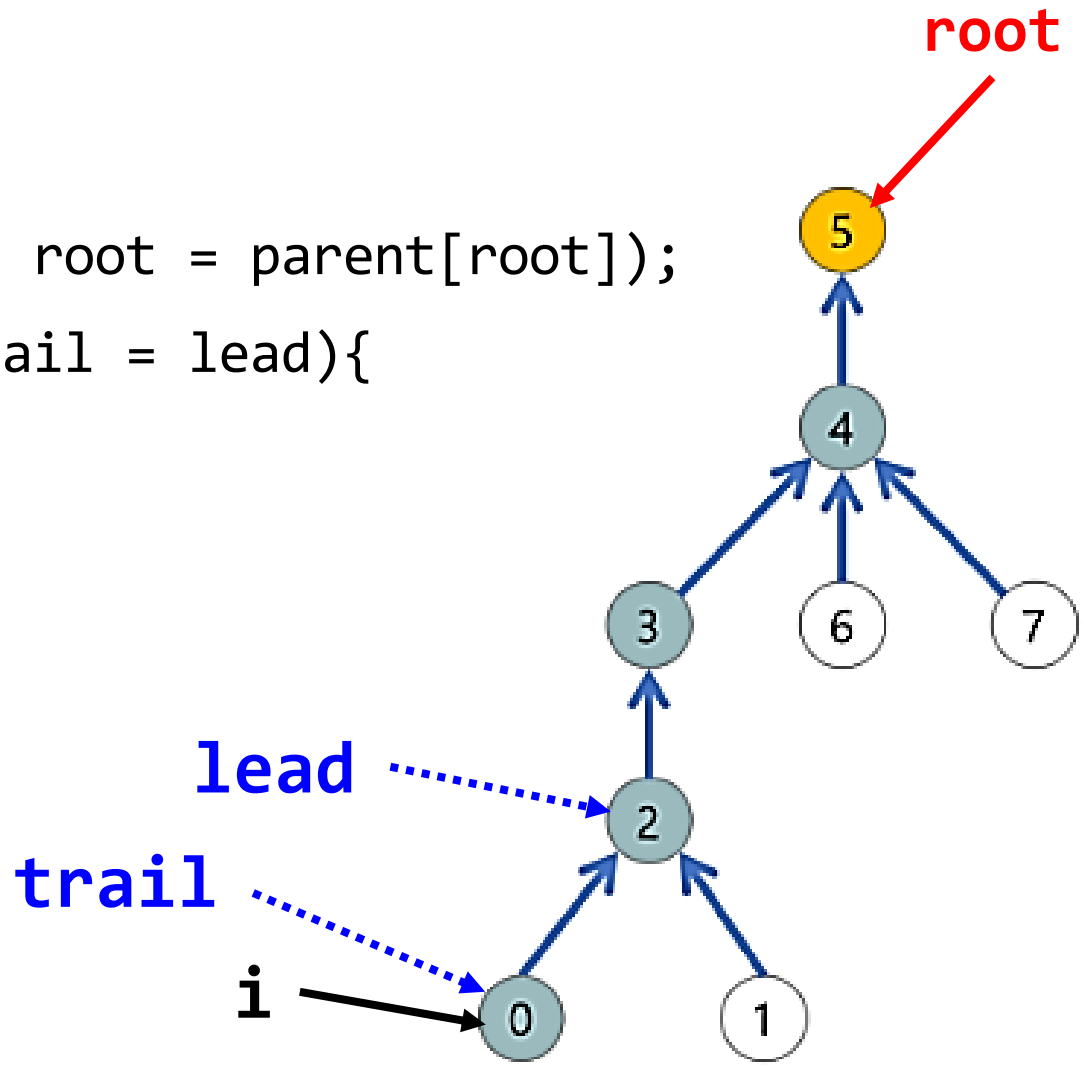
Find 연산의 개선

- **`find(i)`를 위한 붕괴 규칙 (collapsing rule)**
  - 만일 노드 j가 노드 i에서 루트로 가는 경로상에 있으면서 parent[i] != root(i)이면 parent[j]를 root(i)로 지정한다.
- **붕괴 규칙의 역할**
  - **`find(i)`** 연산 수행 시 노드 i의 부모 노드를 루트 노드의 자식 노드로 지정한다.
  - 해당 작업을 노드 i부터 루트 노드까지의 경로상에 있는 모든 노드에 대하여 반복적으로 수행한다.
  - 이에 따라 트리의 높이를 낮춤으로써 변질 트리가 되는 현상을 방지할 수 있다.



Find 연산의 개선

```
int collapsing_find(int i){
      int root, trail, lead;
      for(root = i; parent[root] >= 0; root = parent[root]);
      for(trail = i; trail != root; trail = lead){
            lead = parent[trail];
            parent[trail] = root;
       }
       return root;
}
```



연습 문제 1. 집합의 표현

- **초기에 n+1개의 집합 {0}, {1}, {2}, … , {n}이 있다. 여기에 합집합 연산과, 두 원소가 같은 집합에 포함되어 있는지를 확인하는 연산을 수행하려고 한다. 이를 위한 집합을 표현하는 프로그램을 작성하시오.**

- **입력**
  - 첫째 줄에 n, m이 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.

- **출력**
  - 1로 시작하는 입력에 대해서 a와 b가 같은 집합에 포함되어 있으면 "YES" 또는 "yes"를, 그렇지 않다면 "NO" 또는 "no"를 한 줄에 하나씩 출력한다.

- **제한 조건**
  - $1 \le n \le 1\,000\,000$
  - $1 \le m \le 100\,000$
  - $0 \le a, b \le n$
  - a, b는 정수
  - a와 b는 같을 수도 있다.

문제 출처: https://www.acmicpc.net/problem/1717

연습 문제 1. 집합의 표현

- 예제 입력

```
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
```

- 예제 출력

```
NO
NO
YES
```


문제 출처: https://www.acmicpc.net/problem/1717

Q & A