set

집합

집합의 특징

- 중복 허용x
- 순서가 없음(하지만 현재 상황에서는 id가 순차적으로 할당되기에 상관없다고 판단)

구조

- {대표 id : {id집합}}으로 관리
- {1 : {1, 51, 121...}, 2 : {2, 11, 35}, 3 : {3, 24, 76}, ...}
- 키를 호출해 특정 집합에 즉시 접근 가능

```
final_id = 2  # 대표 ID (딕셔너리 키)
id_set = id_clusters[final_id]  # {2, 11, 35} (한번의 조회로 모든 관계 파악)
```

다른 표현에서의 id 탐색 과정

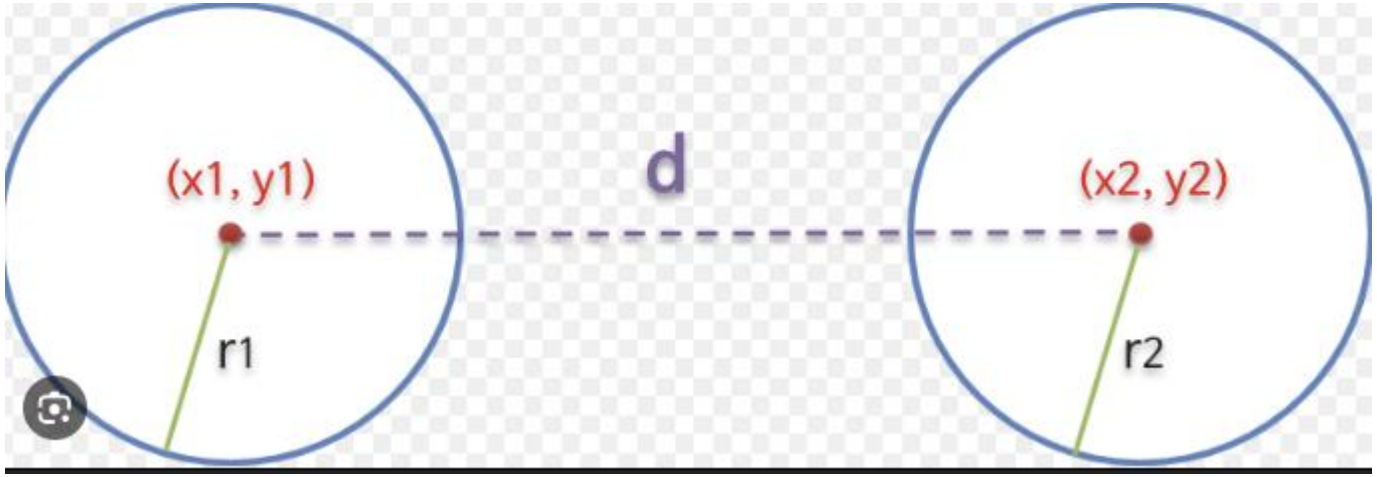
- Id 2의 최종 id를 찾으려면 체인을 따라 모두 탐색해야 함.

```
id_transitions = {
    2: 11,   # 이전 ID: 새 ID
    11: 35
}
```

```
current_id = 2
while current_id in id_transitions:
    current_id = id_transitions[current_id]  # 2->11->35 (체인 따라 여러 단계 탐색)
final_id = current_id  # 35
```

거리 기반 병합

- 중심점 계산: x, y좌표의 차이 계산
- 거리 임계값(distance_threshold, 현재 15)설정하여 이 거리 내에 있는 객체들은 동일 객체로 간주
- 각 집합마다 하나의 대표 id(일반적으로 가장 작은 값)를 유지



관심 영역(ROI) 설정 이유

- 현재 지정한 영역에 폐사체가 2개가 있는데 폐사체에 대한 탐지가 풀렸다가 잡혀도 id가 잘 유지되는지 확인하기 위함

병합 판단 과정

1. 이미 같은 집합에 속한 id쌍은 건너뜀
2. 두 객체 쌍 거리가 임계값보다 작으면 동일 객체로 판단
3. 두 id가 서로 다른 집합에 속한 경우 두 집합을 하나로 병합
4. 하나의 id만 집합에 속한 경우 한 id를 해당 집합에 추가
5. 병합 후 대표 id는 가장 작은 값을 사용

개선점

- 교차 상황에서 두 객체를 한 집합으로 표현
- 관심영역 안에서 탐지된 id가 생각보다 많음
- 밀집되있는 상황은 예외처리
- 모든 객체 쌍을 비교해서 건너뛰는 작업도 결국 **모든 쌍에 대해 확인이 필요** -> 이전 프레임의 id목록과 현재 프레임의 id목록을 비교해서 **사라진 id목록**과 **새로 생긴 id목록**만 비교
- O(n^2)(n은 총 객체 수) -> O(m*k)(m은 사라진 ID 수, k는 새로 생긴 ID 수, 일반적으로 m,k << n)