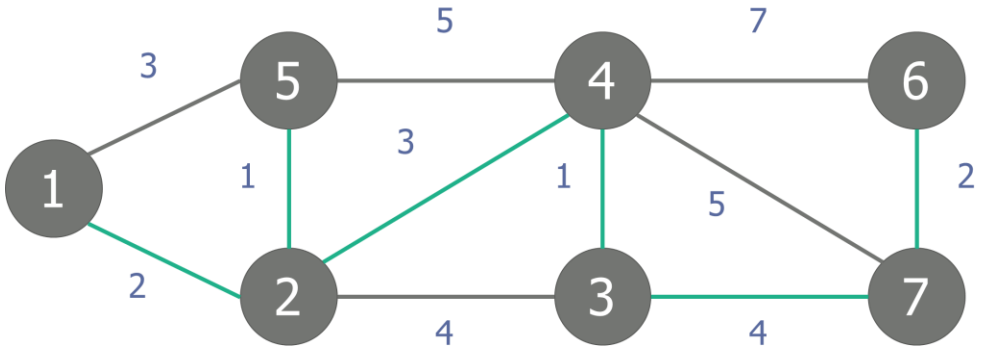
최소 스패닝 트리

Minimum Spanning Tree

- 스패닝 트리 : 그래프에서 일부 간선을 선택해서 만든 트리
- 최소 스패닝 트리 : 스패닝 트리 중에 선택한 간선의 가중치의 합이 최소인 트리
- 최소 비용 : 2(1 ~ 2) + 1(5 ~ 2) + 3(2 ~ 4) + 1(4 ~ 3) + 4(3 ~ 7) + 2(6 ~ 7)



- 프림(Prim) : MST의 정점을 점점 더 확장해 나아가는 방법
- 크루스칼(Kruskal) : MST의 간선을 점점 더 확장해 나아가는 방법

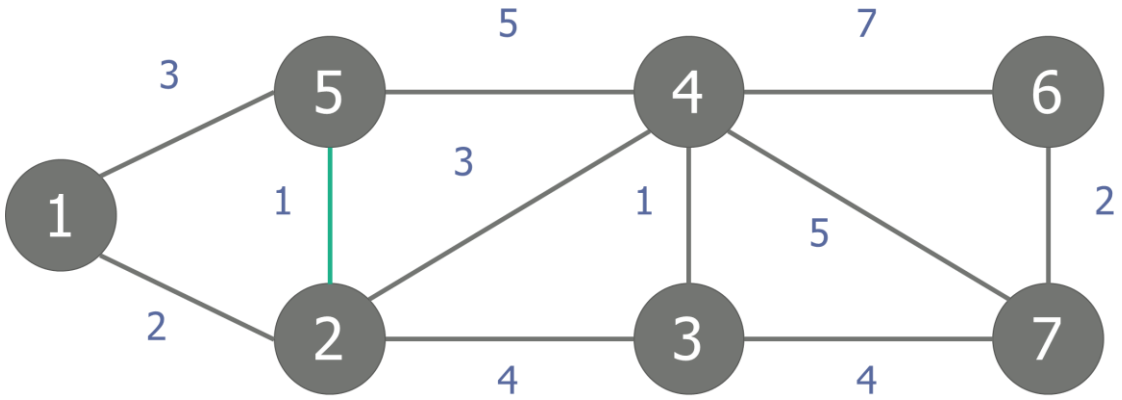
최소 스패닝 트리 - 크루스칼

Minimum Spanning Tree

- 가중치가 작은 Edge 부터 순서대로 살펴 본다. 먼저 가중치를 오름차순으로 정렬한다.
- Edge e 정보가 (u, v, w) 일 때, u와 v가 다른 집합이면 간선 e를 MST에 추가한다.
- 같은 집합 / 다른 집합을 판별할 때에는 유니온 파인드를 사용한다.

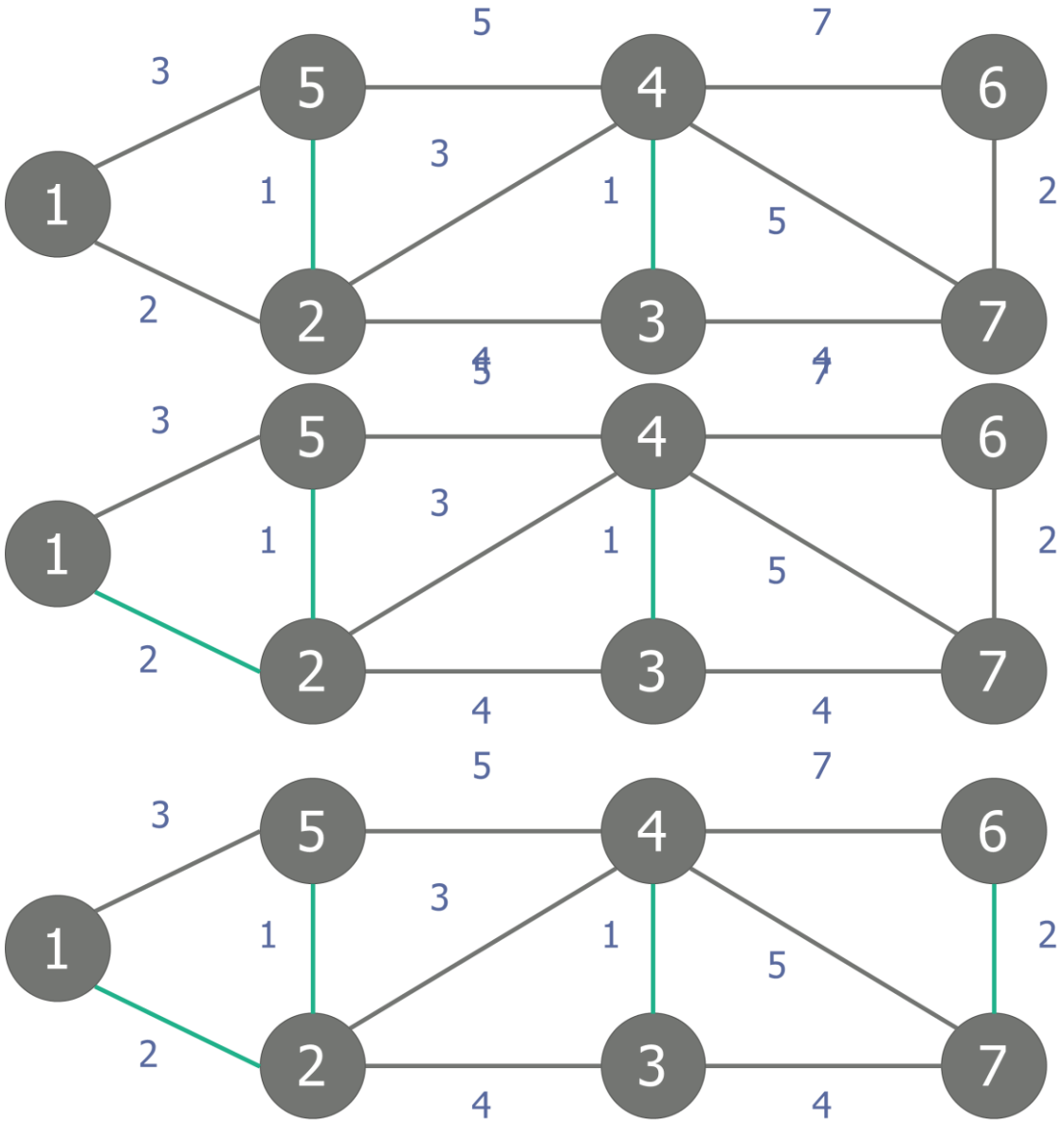
최소 간선 1을 추가한다.

간선을 추가할 때에는 u, v 정점이 같은 집합인지 확인



최소 스패닝 트리 - 크루스칼

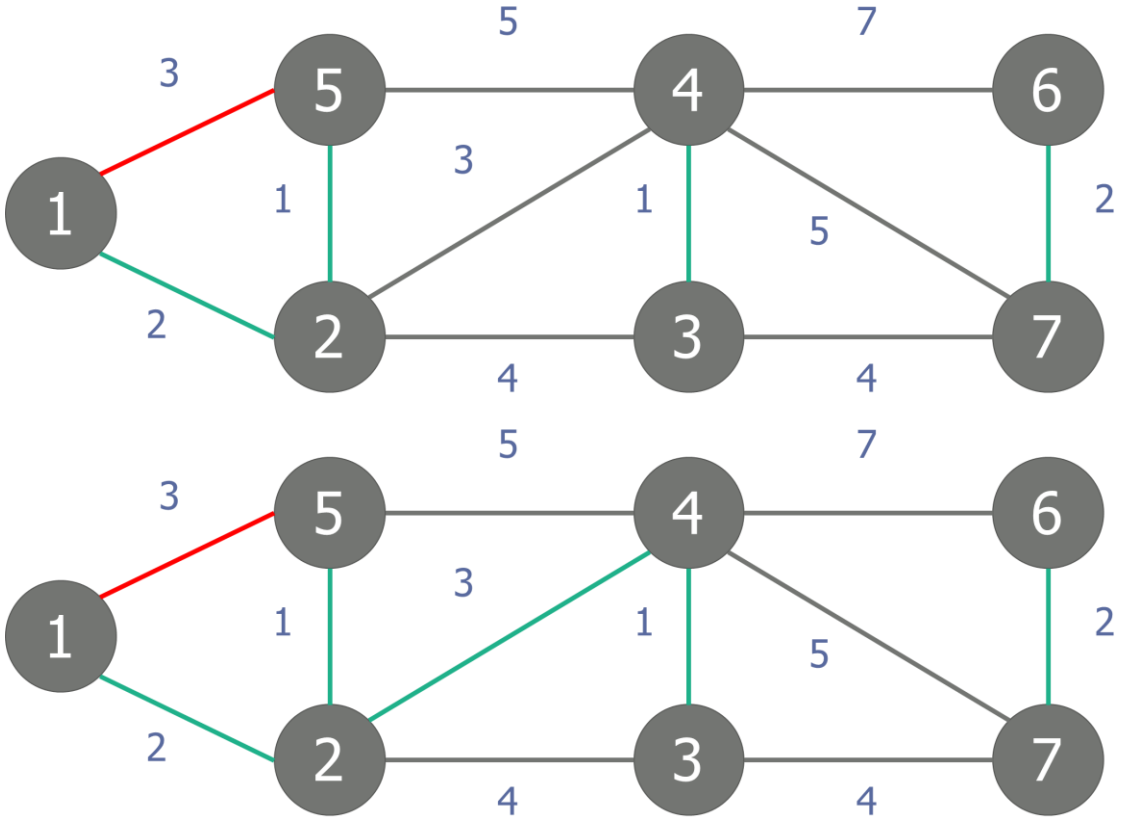
[Minimum Spanning Tree]



최소 스패닝 트리 - 크루스칼

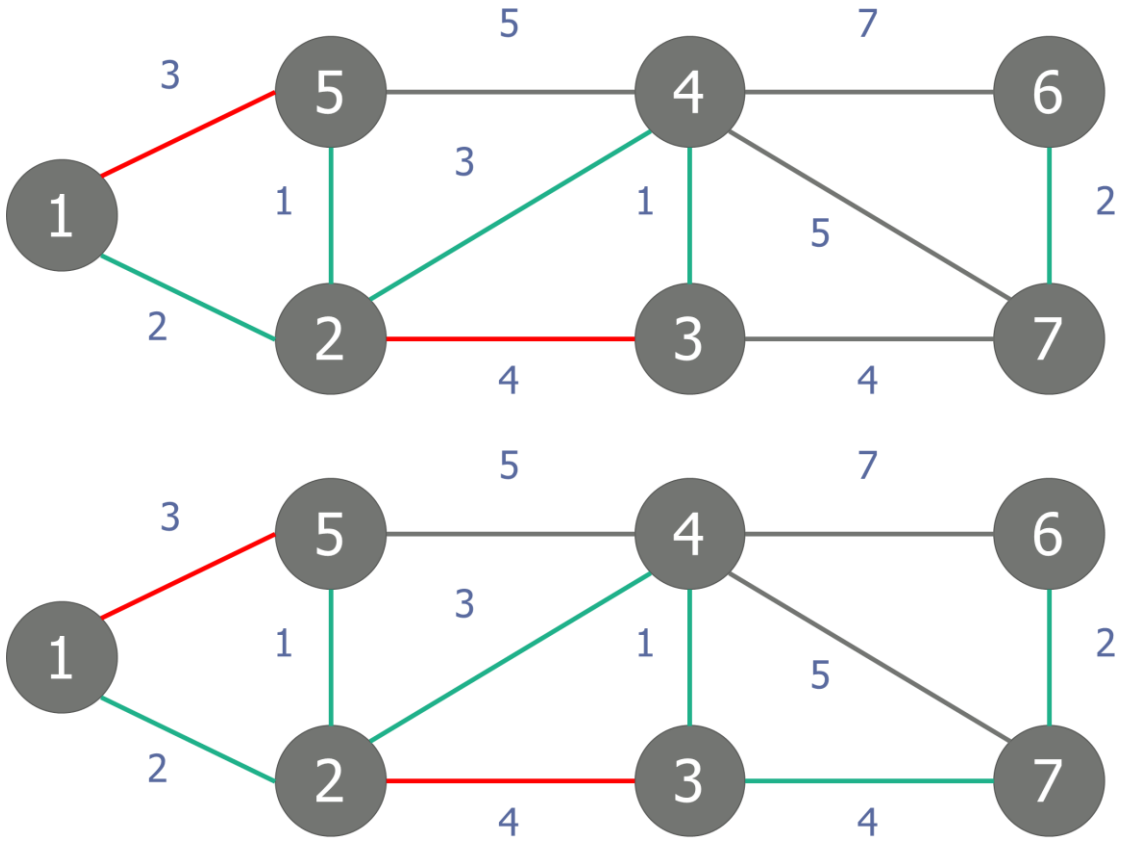
[Minimum Spanning Tree]

정점 1의 집합과 정점 5의 집합은 같은 집합이므로 간선을 선택하지 않는다.



최소 스패닝 트리 - 크루스칼

[Minimum Spanning Tree]



최소 스패닝 트리 - 크루스칼

[Minimum Spanning Tree]

Union - Find

```
vi p;
int Find(int x) {
        if (x == p[x]) {
                return x;
        }
        else {
                return p[x] = Find(p[x]);
        }
}
void Union(int x, int y) {
        x = Find(x);
        y = Find(y);
        p[x] = y;
}
```

Kruskal 간선 선택

```
sort(A.begin(), A.end());

int ans = 0;
for (int i = 0; i<M; i++) {
        Edge e = A[i];
        int x = Find(e.start);
        int y = Find(e.end);
        if (x != y) {
                Union(e.start, e.end);
                ans += e.cost;
        }
}
```