2025 겨울방학 알고리즘 스터디

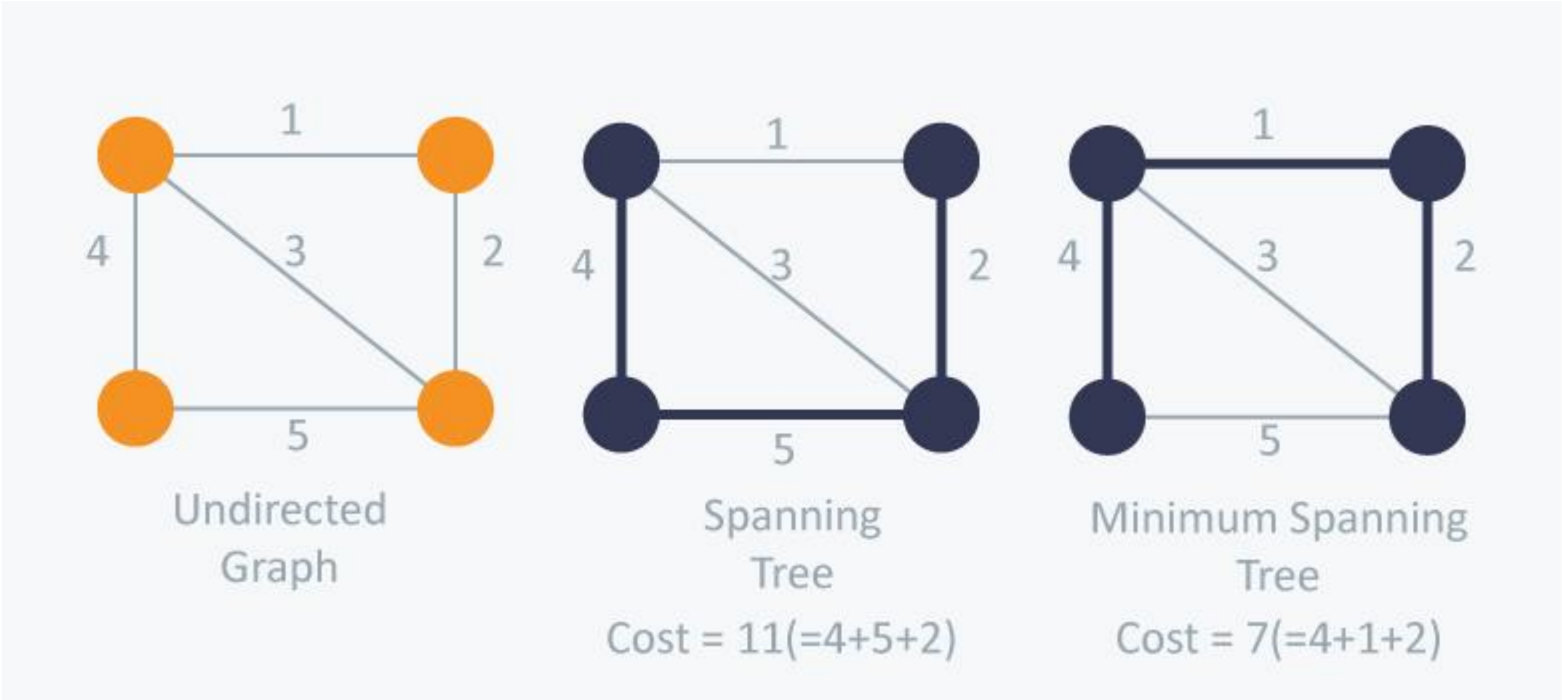
최소 신장 트리

컴퓨터 공학과 20230546 서보경

목차

최소 신장 트리란?

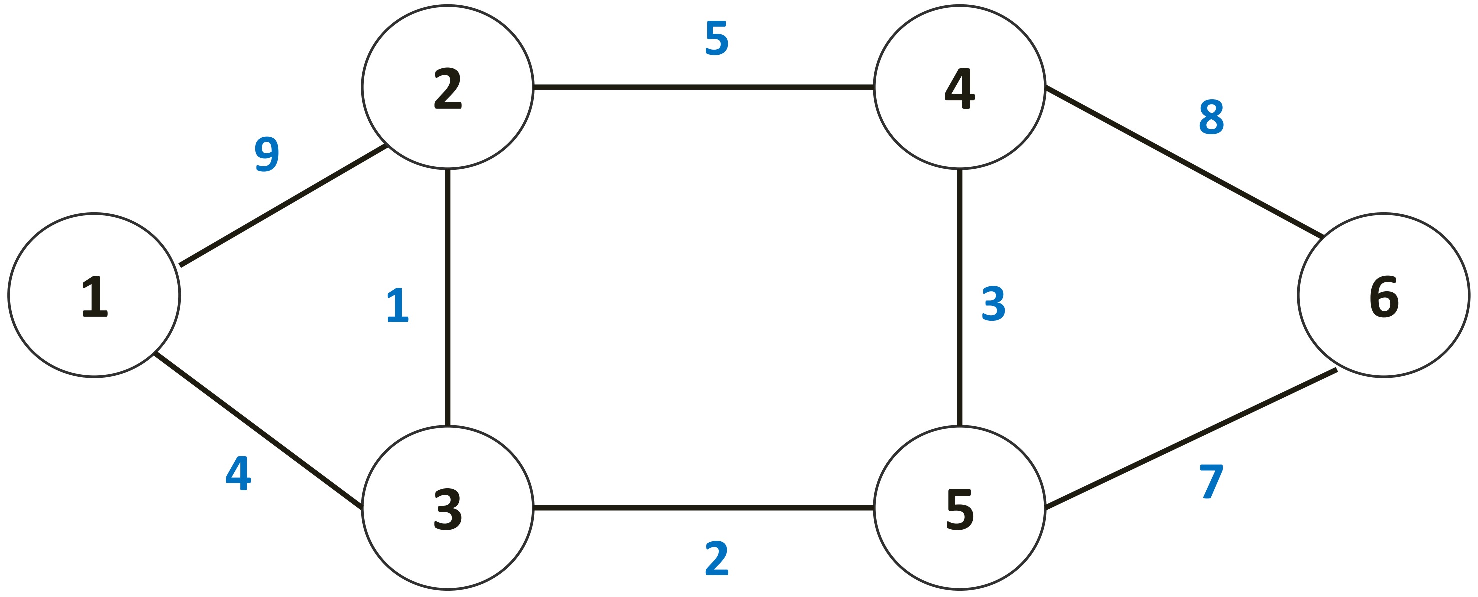
최소 신장 트리(Minimum Spanning Tree; MST) 는 특정 순서에 따라 데이터를 정렬하여 연결을 구성하는 그래프 기반의 구조이다. MST의 가장 큰 특징은 모든 노드를 연결하되, 간선 가중치의 합이 최소가 되는 구조를 빠르게 찾을 수 있다는 점이다. 이러한 최소 비용의 연결은 항상 그래프의 특정 규칙(사이클이 없는 구조, 최소 비용 간선 우선 선택) 을 유지한다.



일반적 신장 트리는 사이클 없이 간선 개수 최소화,
최소 신장 트리는 간선 개수를 최소화 하면서
가중치의 합이 최소가 되어야 한다.

크루스칼 알고리즘

edge_count : 0 | sum : 0



정점	1	2	3	4	5	6
부모	1	2	3	4	5	6

간선	2 – 3	3 – 5	4 – 5	1 – 3	2 – 4	5 – 6	4 – 6	1 - 2
가중치	1	2	3	4	5	7	8	9

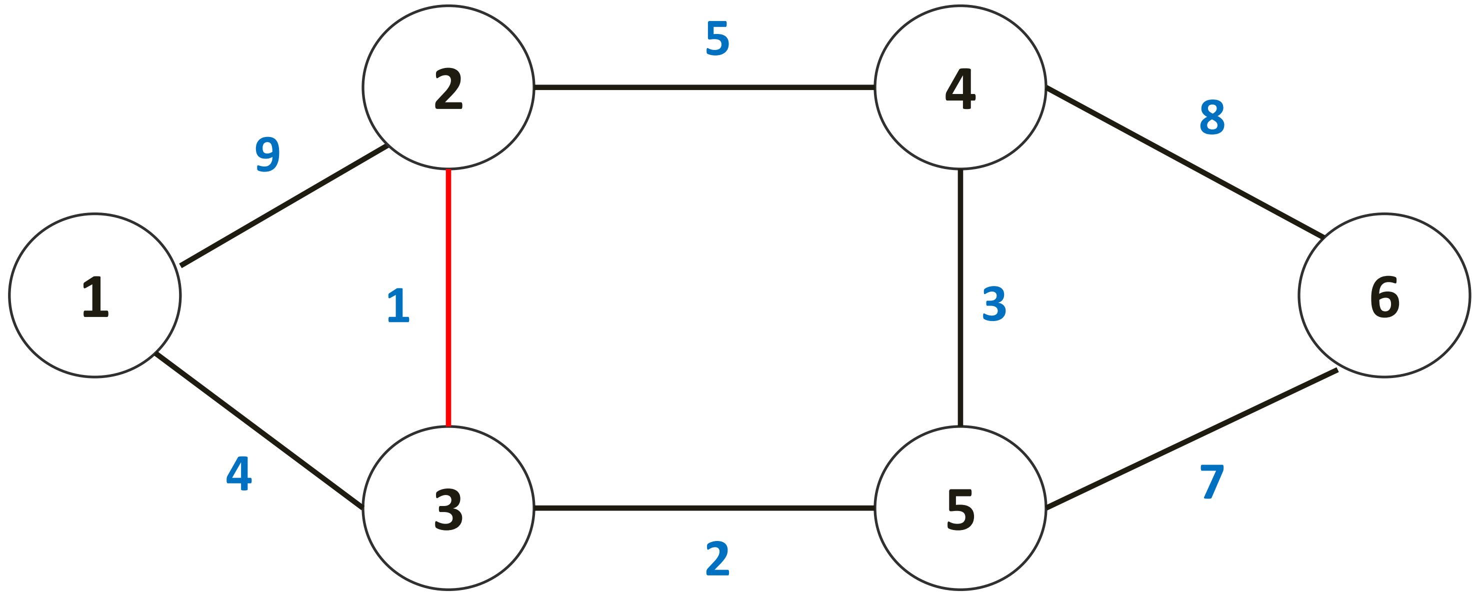
간선을 기반으로 하며, 분리집합을 이용하는 크루스칼 알고리즘을 다뤄보자.

크루스칼 알고리즘은 가장 가중치가 작은 간선부터 골라나가는 방식이다.

이때, 사이클이 없어야 하므로 확인을 위해 분리집합을 사용해 줄 것 이다.

크루스칼 알고리즘

edge_count : 0 | sum : 0



정점	1	2	3	4	5	6
부모	1	2	3	4	5	6

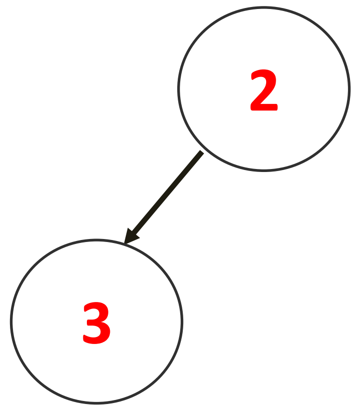
간선	2 – 3	3 – 5	4 – 5	1 – 3	2 – 4	5 – 6	4 – 6	1 - 2
가중치	1	2	3	4	5	7	8	9

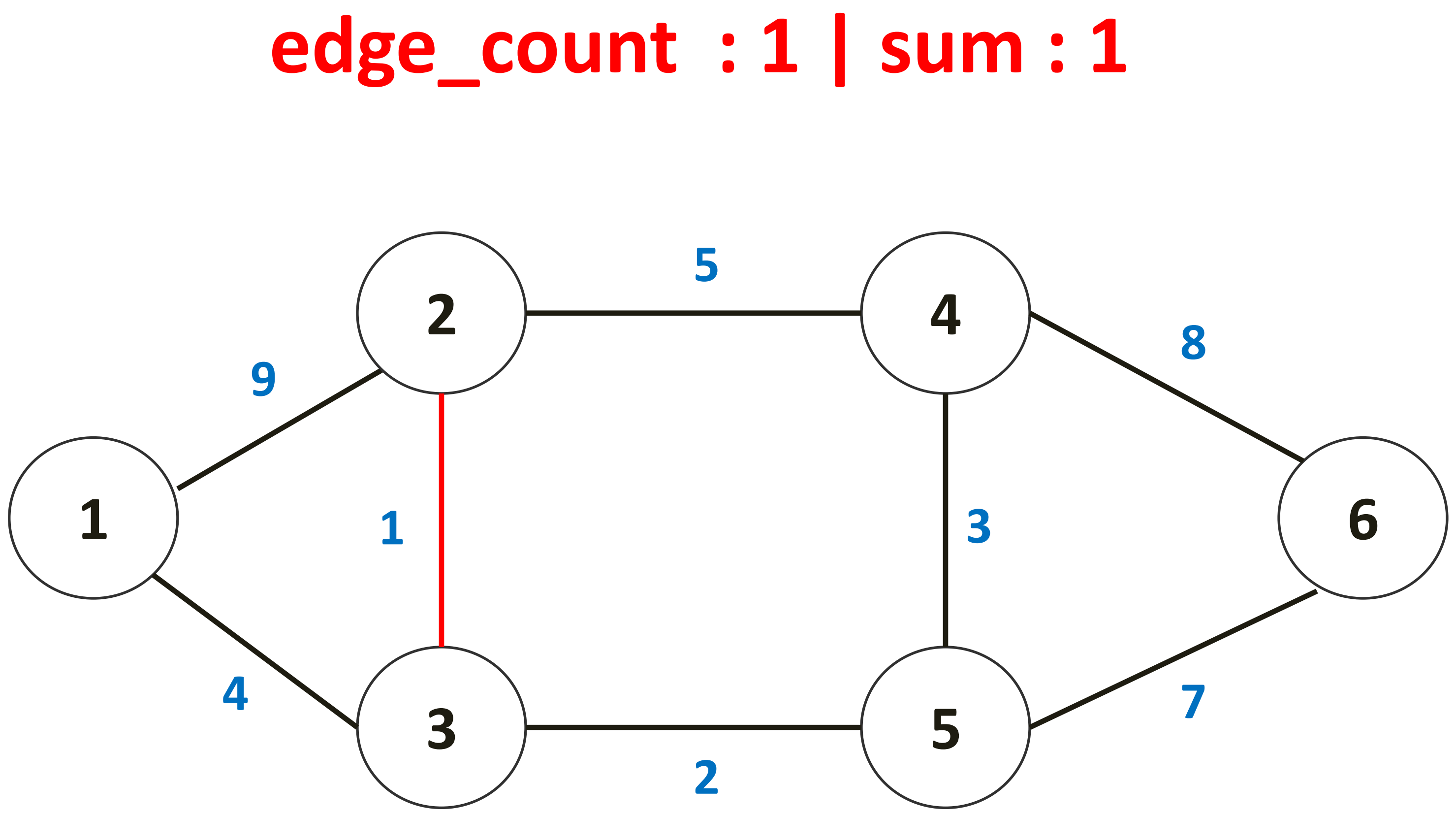
선택 : 2 – 3 간선

2번 정점과 3번 정점을 잇는 가중치 1의 간선을 추가하려고 한다.

이때 분리집합을 이용해서 두 집합이 어떤 컴포넌트에 있는지 알아 보자.

크루스칼 알고리즘





정점	1	2	3	4	5	6
부모	1	2	2	4	5	6

간선	2 – 3	3 – 5	4 – 5	1 – 3	2 – 4	5 – 6	4 – 6	1 - 2
가중치	1	2	3	4	5	7	8	9

선택 : 2 – 3 간선

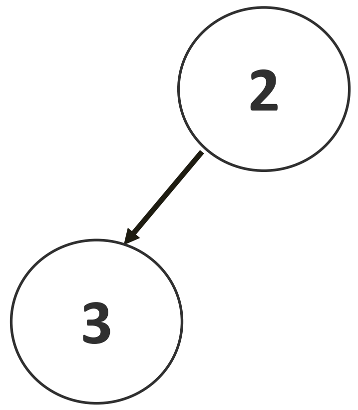
2번과 3번 정점은 같은 컴포넌트 내에 존재 하지 않으므로 사이클을 형성 하지 않는다.

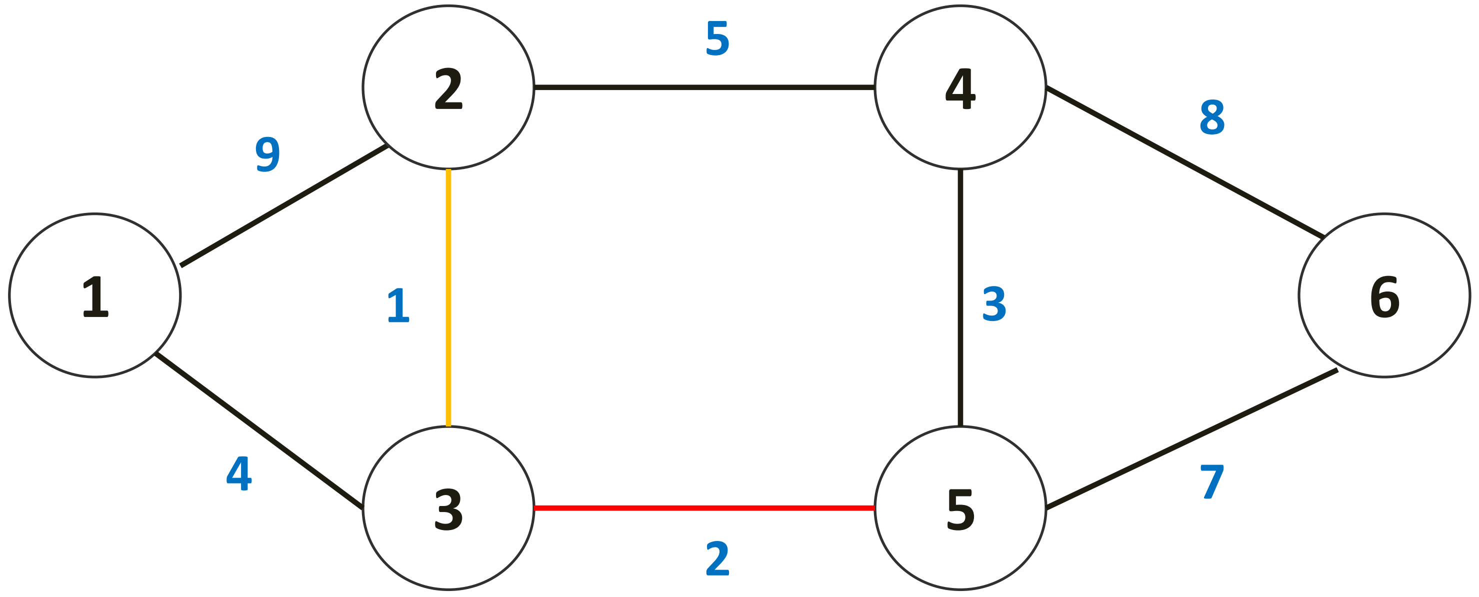
분리집합으로 정점을 병합한 후에 현재 간선을 택해준다.

현재 선택한 간선은 1개, 합계는 1이다.

크루스칼 알고리즘

edge_count : 1 | sum : 1





정점	1	2	3	4	5	6
부모	1	2	2	4	5	6

간선	2 – 3	3 – 5	4 – 5	1 – 3	2 – 4	5 – 6	4 – 6	1 - 2
가중치	1	2	3	4	5	7	8	9

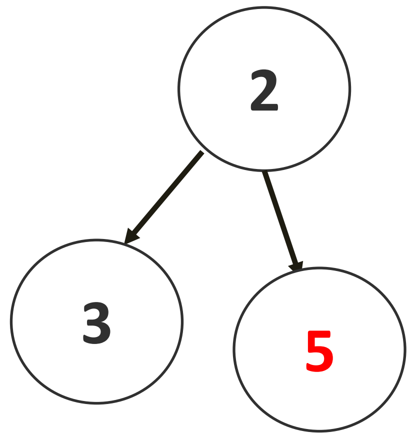
선택 : 3 – 5 간선

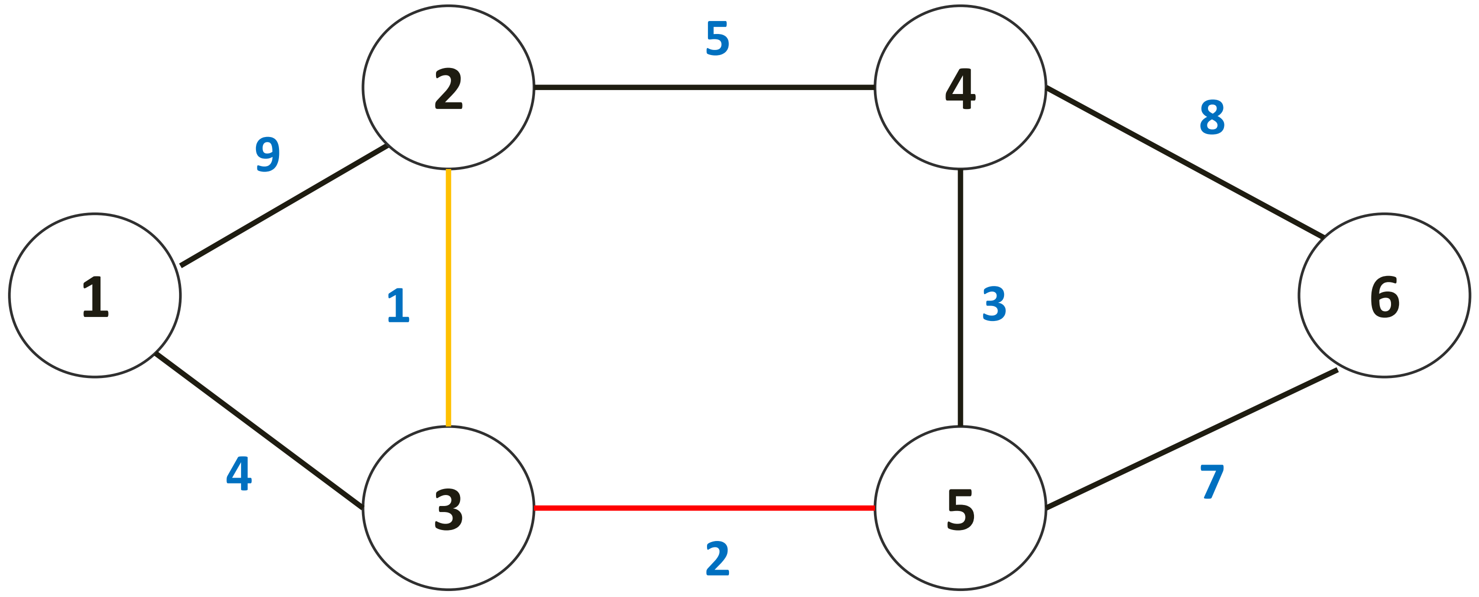
3번 정점과 5번 정점을 잇는 가중치 2의 간선을 추가하려고 한다.

이때 분리집합을 이용해서 두 집합이 어떤 컴포넌트에 있는지 알아 보자.

크루스칼 알고리즘

edge_count : 2 | sum : 3





정점	1	2	3	4	5	6
부모	1	2	2	4	2	6

간선	2 – 3	3 – 5	4 – 5	1 – 3	2 – 4	5 – 6	4 – 6	1 - 2
가중치	1	2	3	4	5	7	8	9

선택 : 3 – 5 간선

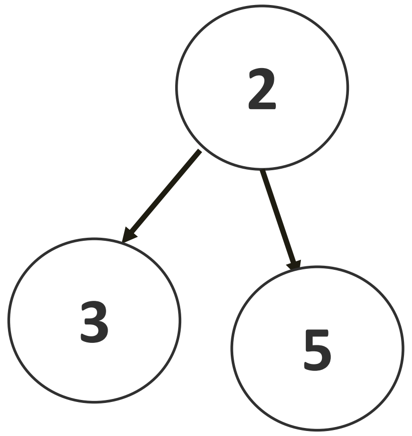
3번과 5번 정점은 같은 컴포넌트 내에 존재 하지 않으므로 사이클을 형성 하지 않는다.

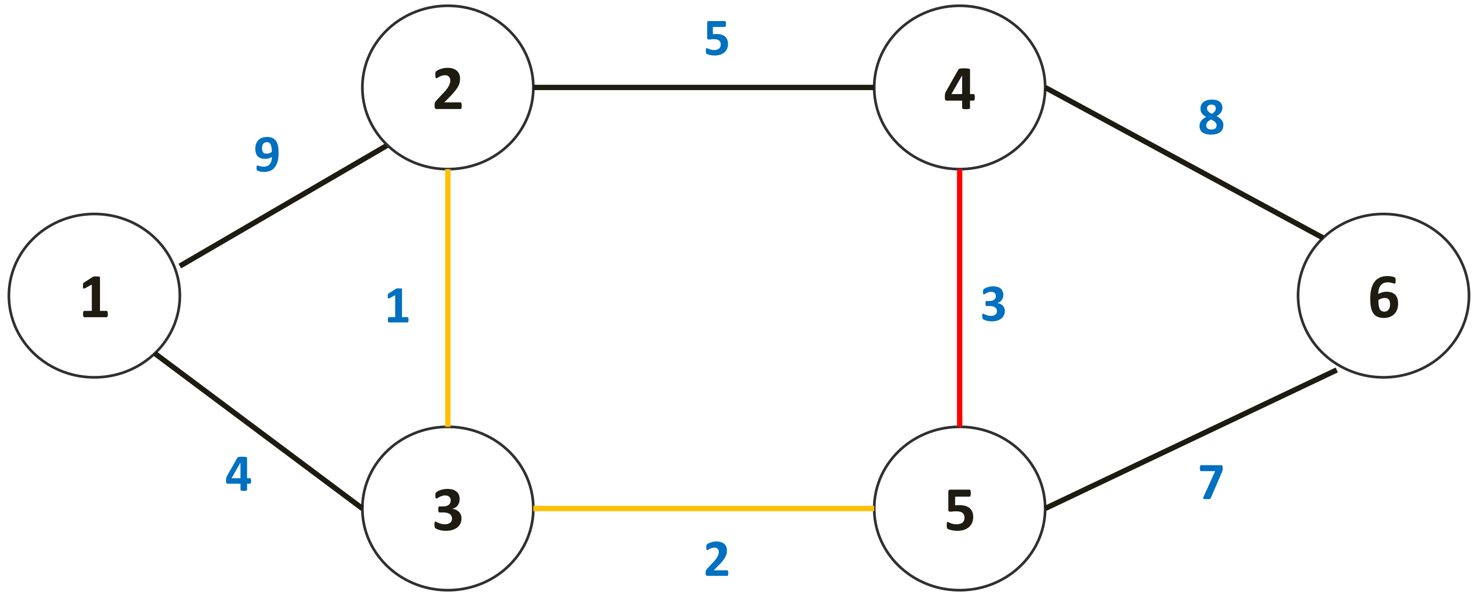
분리집합으로 정점을 병합한 후에 현재 간선을 택해준다.

현재 선택한 간선은 2개, 합계는 3이다.

크루스칼 알고리즘

edge_count : 2 | sum : 3





정점	1	2	3	4	5	6
부모	1	2	2	4	2	6

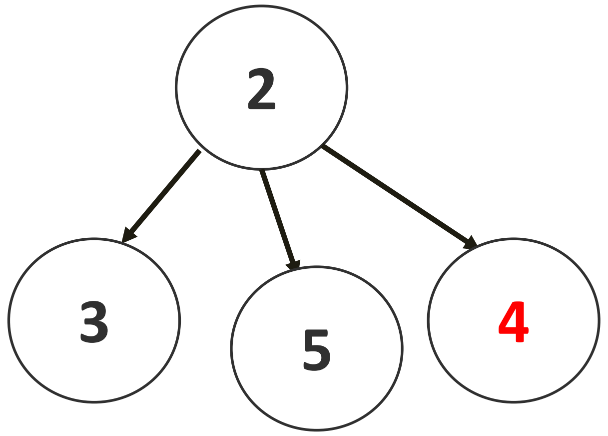
간선	2 – 3	3 – 5	4 – 5	1 – 3	2 – 4	5 – 6	4 – 6	1 - 2
가중치	1	2	3	4	5	7	8	9

선택 : 4 – 5 간선

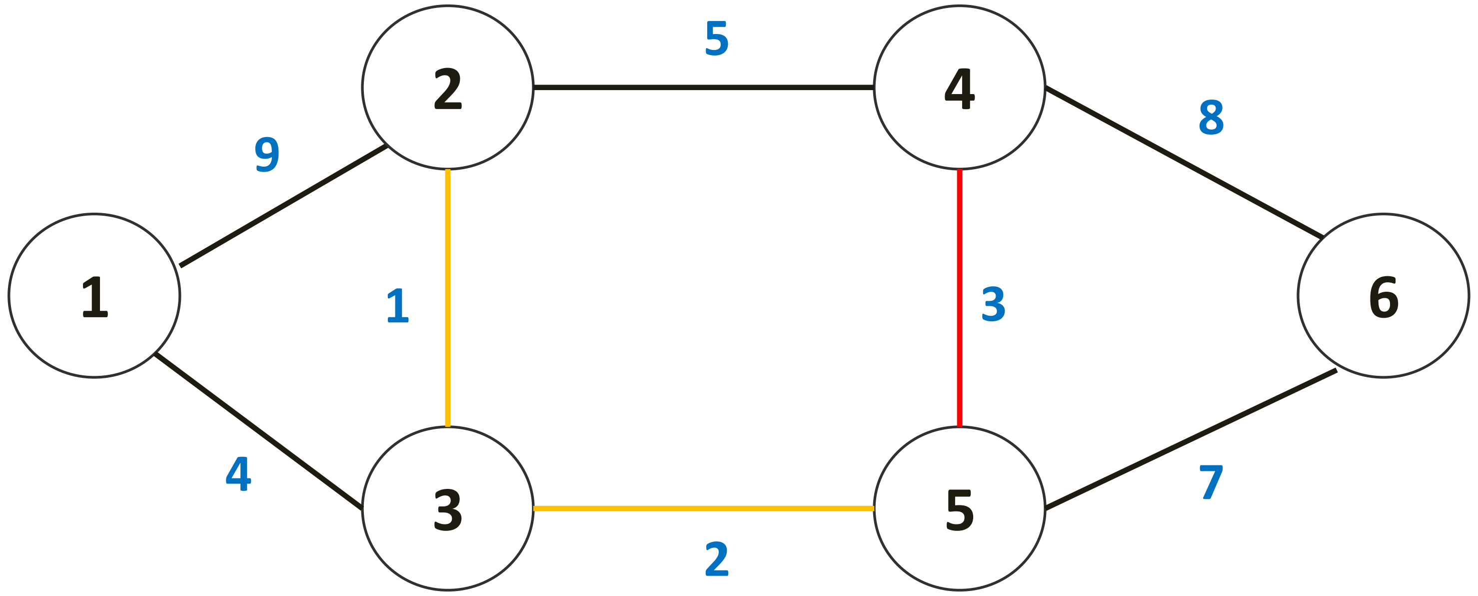
4번 정점과 5번 정점을 잇는 가중치 3의 간선을 추가하려고 한다.

이때 분리집합을 이용해서 두 집합이 어떤 컴포넌트에 있는지 알아 보자.

크루스칼 알고리즘



edge_count : 3 | sum : 6



정점	1	2	3	4	5	6
부모	1	2	2	2	2	6

간선	2 – 3	3 – 5	4 – 5	1 – 3	2 – 4	5 – 6	4 – 6	1 - 2
가중치	1	2	3	4	5	7	8	9

선택 : 4 – 5 간선

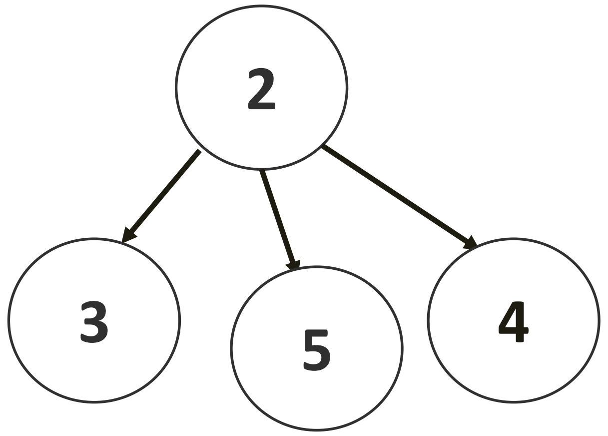
4번과 5번 정점은 같은 컴포넌트 내에 존재 하지 않으므로 사이클을 형성 하지 않는다.

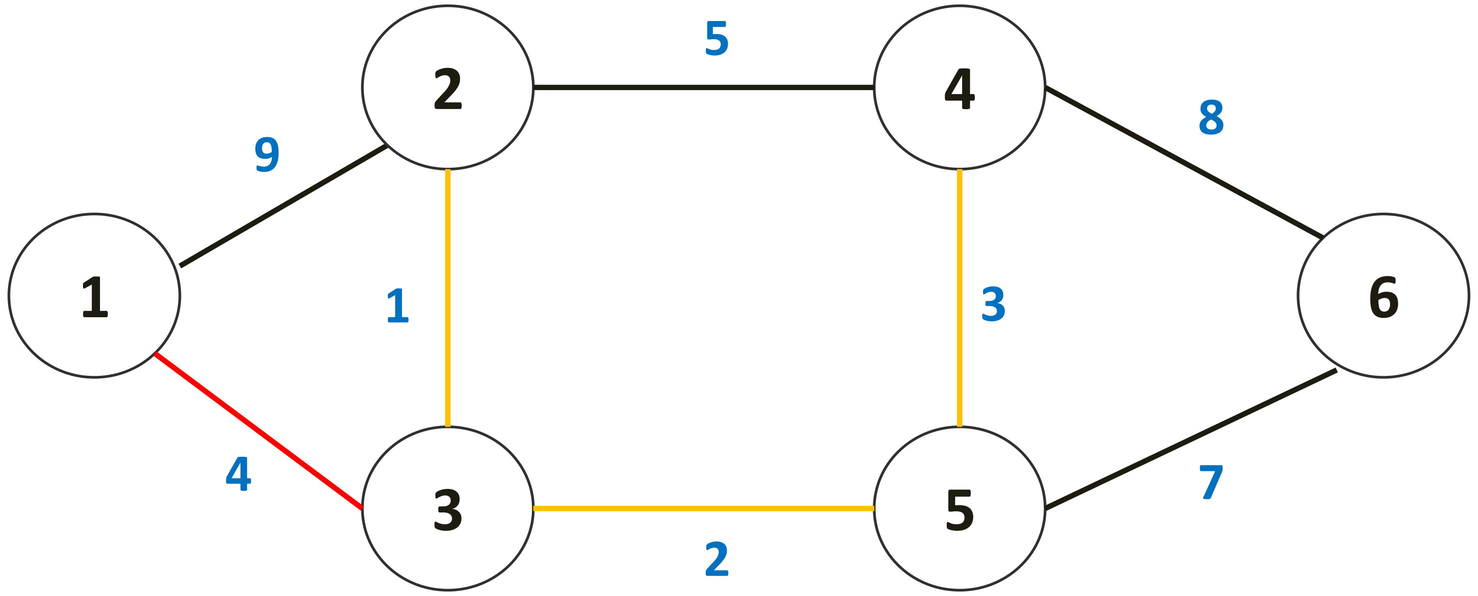
분리집합으로 정점을 병합한 후에 현재 간선을 택해준다.

현재 선택한 간선은 3개, 합계는 6이다.

크루스칼 알고리즘

edge_count : 3 | sum : 6





정점	1	2	3	4	5	6
부모	1	2	2	2	2	6

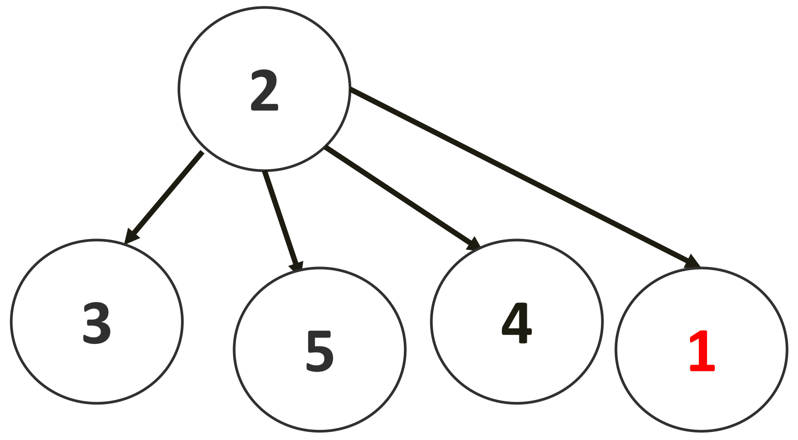
간선	2 – 3	3 – 5	4 – 5	1 – 3	2 – 4	5 – 6	4 – 6	1 - 2
가중치	1	2	3	4	5	7	8	9

선택 : 1 – 3 간선

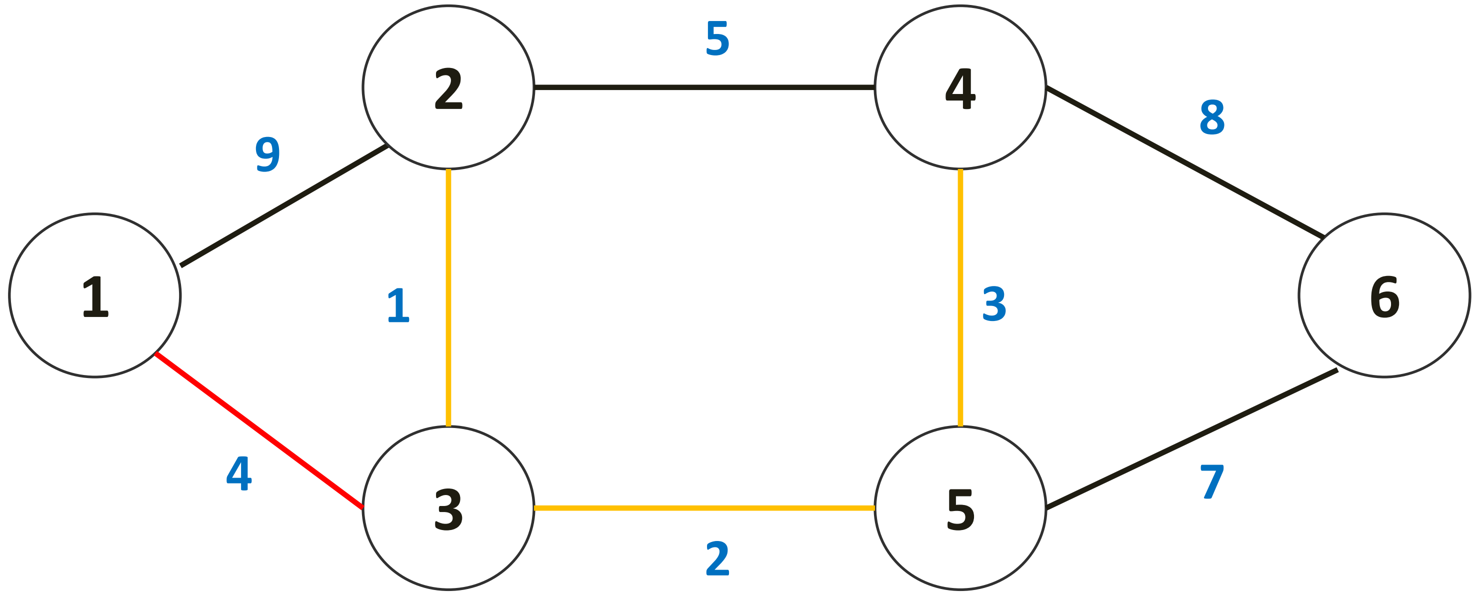
1번 정점과 3번 정점을 잇는 가중치 4의 간선을 추가하려고 한다.

이때 분리집합을 이용해서 두 집합이 어떤 컴포넌트에 있는지 알아 보자.

크루스칼 알고리즘



edge_count : 4 | sum : 10



정점	1	2	3	4	5	6
부모	2	2	2	2	2	6

간선	2 – 3	3 – 5	4 – 5	1 – 3	2 – 4	5 – 6	4 – 6	1 - 2
가중치	1	2	3	4	5	7	8	9

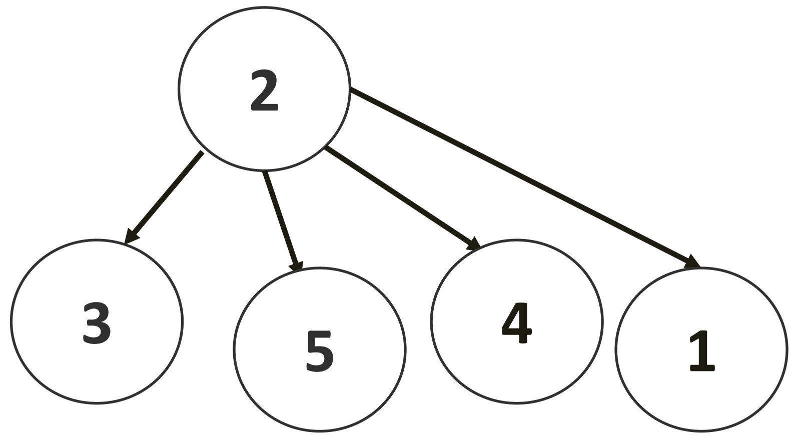
선택 : 1 – 3 간선

1번과 3번 정점은 같은 컴포넌트 내에 존재 하지 않으므로 사이클을 형성 하지 않는다.

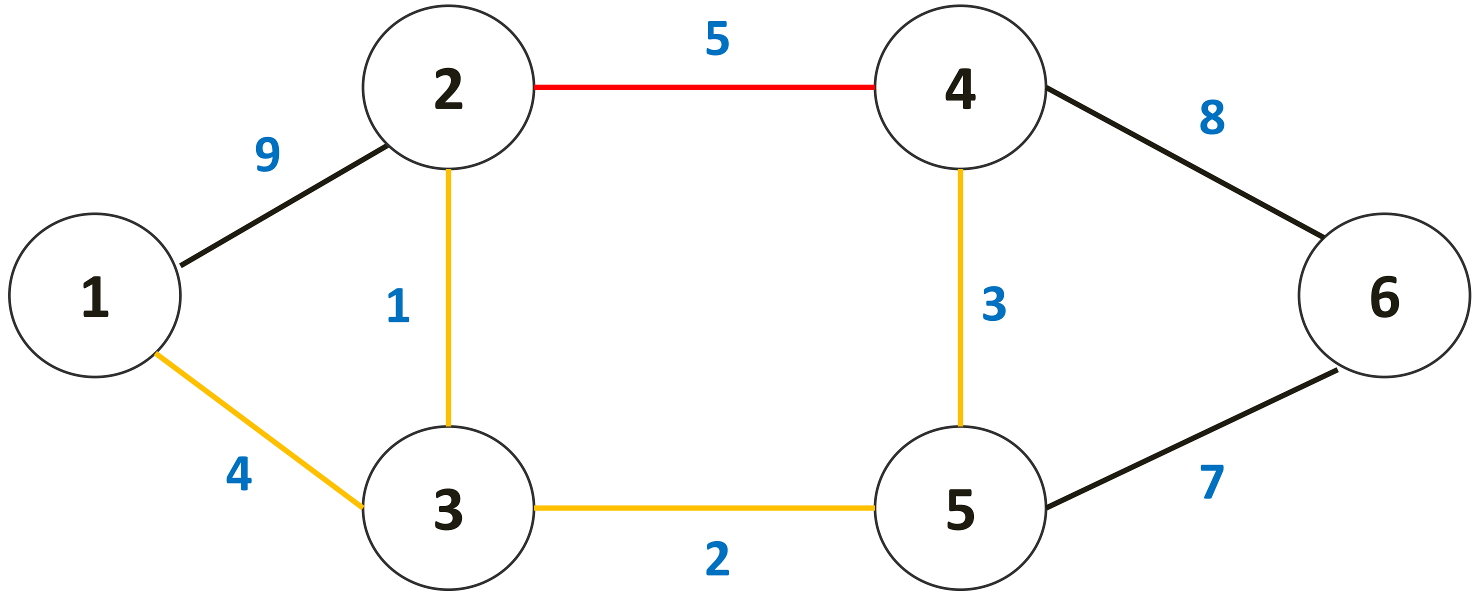
분리집합으로 정점을 병합한 후에 현재 간선을 택해준다.

현재 선택한 간선은 4개, 합계는 10이다.

크루스칼 알고리즘



edge_count : 4 | sum : 10



정점	1	2	3	4	5	6
부모	2	2	2	2	2	6

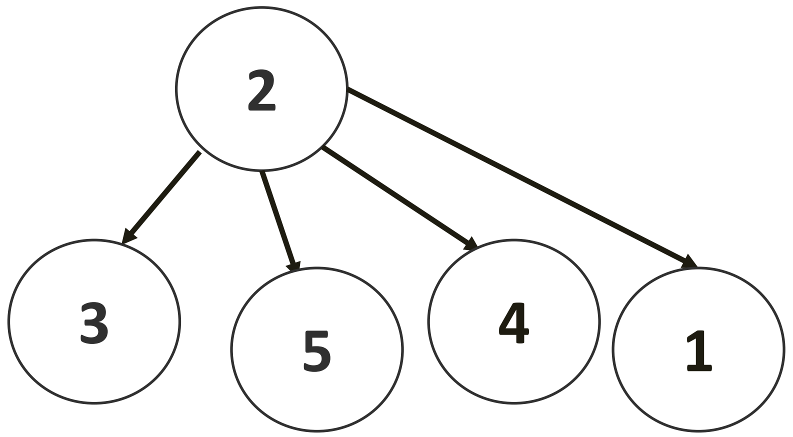
간선	2 – 3	3 – 5	4 – 5	1 – 3	2 – 4	5 – 6	4 – 6	1 - 2
가중치	1	2	3	4	5	7	8	9

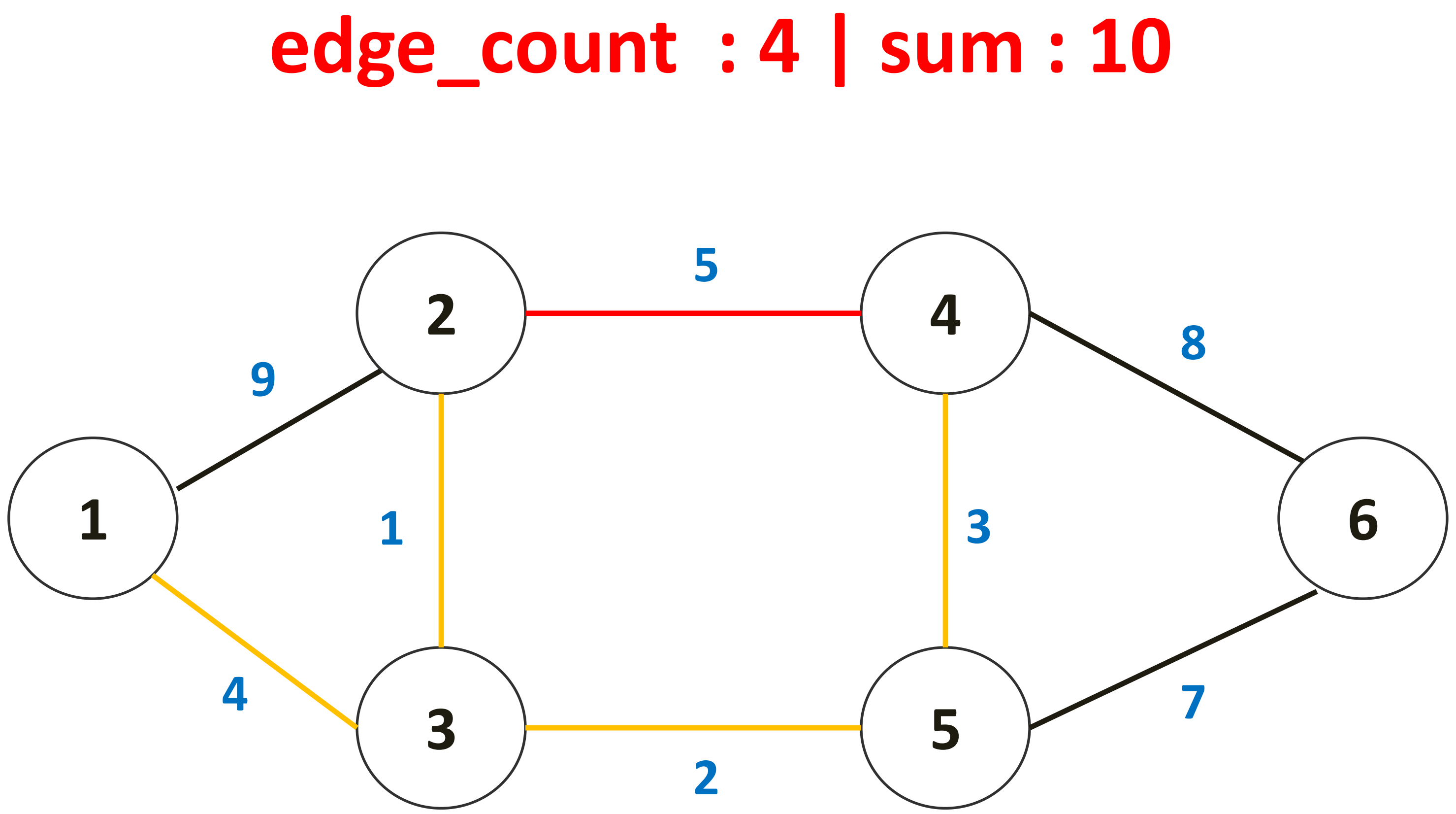
선택 : 2 – 4 간선

2번 정점과 4번 정점을 잇는 가중치 5의 간선을 추가하려고 한다.

이때 분리집합을 이용해서 두 집합이 어떤 컴포넌트에 있는지 알아 보자.

크루스칼 알고리즘





정점	1	2	3	4	5	6
부모	2	2	2	2	2	6

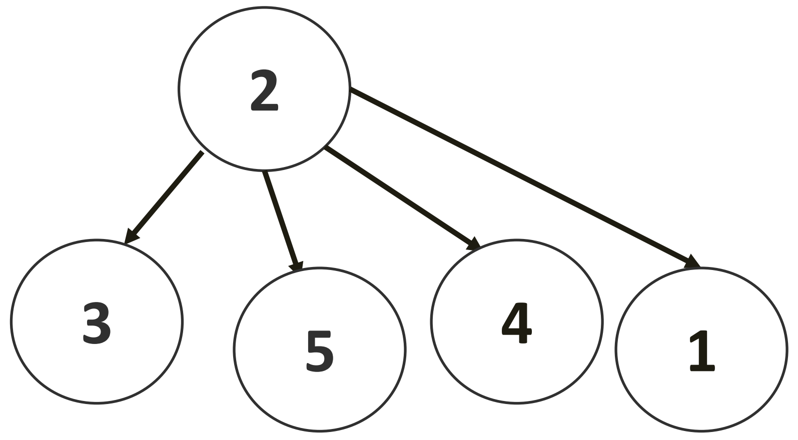
간선	2 – 3	3 – 5	4 – 5	1 – 3	2 – 4	5 – 6	4 – 6	1 - 2
가중치	1	2	3	4	5	7	8	9

선택 : 2 – 4 간선

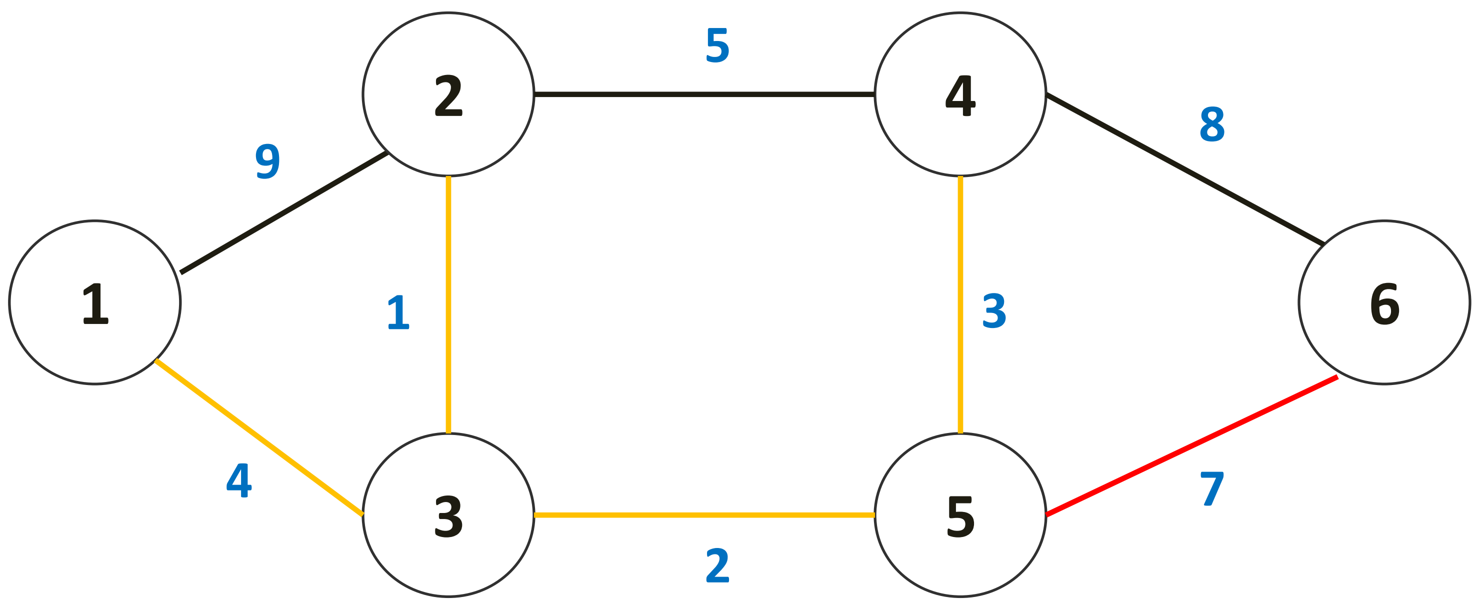
2번과 4번은 같은 컴포넌트 내에 존재 한다.

신장 트리 자체가 사이클을 허용 하지 않는 구조이므로 뒤에 있는 간선들 보다 가중치가 작지만 이 간선은 고를 수가 없다.
고로 무시하고 다음으로 넘어간다.

크루스칼 알고리즘



edge_count : 4 | sum : 10



정점	1	2	3	4	5	6
부모	2	2	2	2	2	6

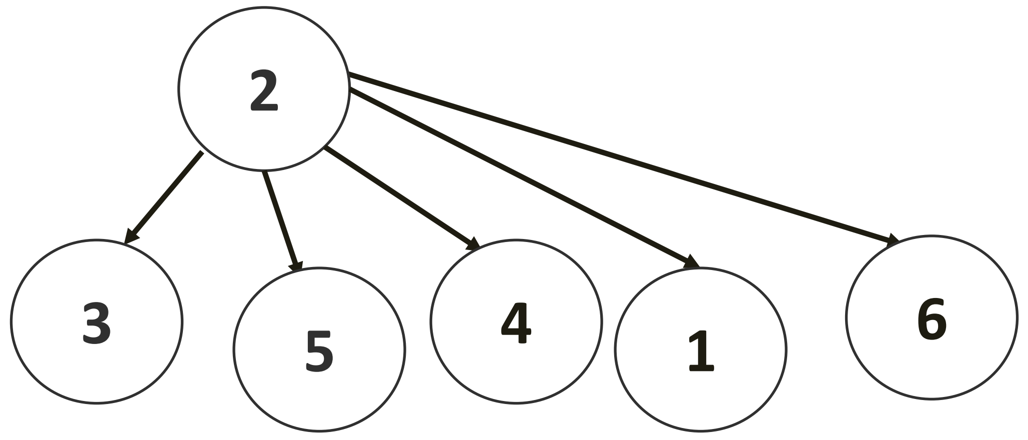
간선	2 – 3	3 – 5	4 – 5	1 – 3	2 – 4	5 – 6	4 – 6	1 - 2
가중치	1	2	3	4	5	7	8	9

선택 : 5 – 6 간선

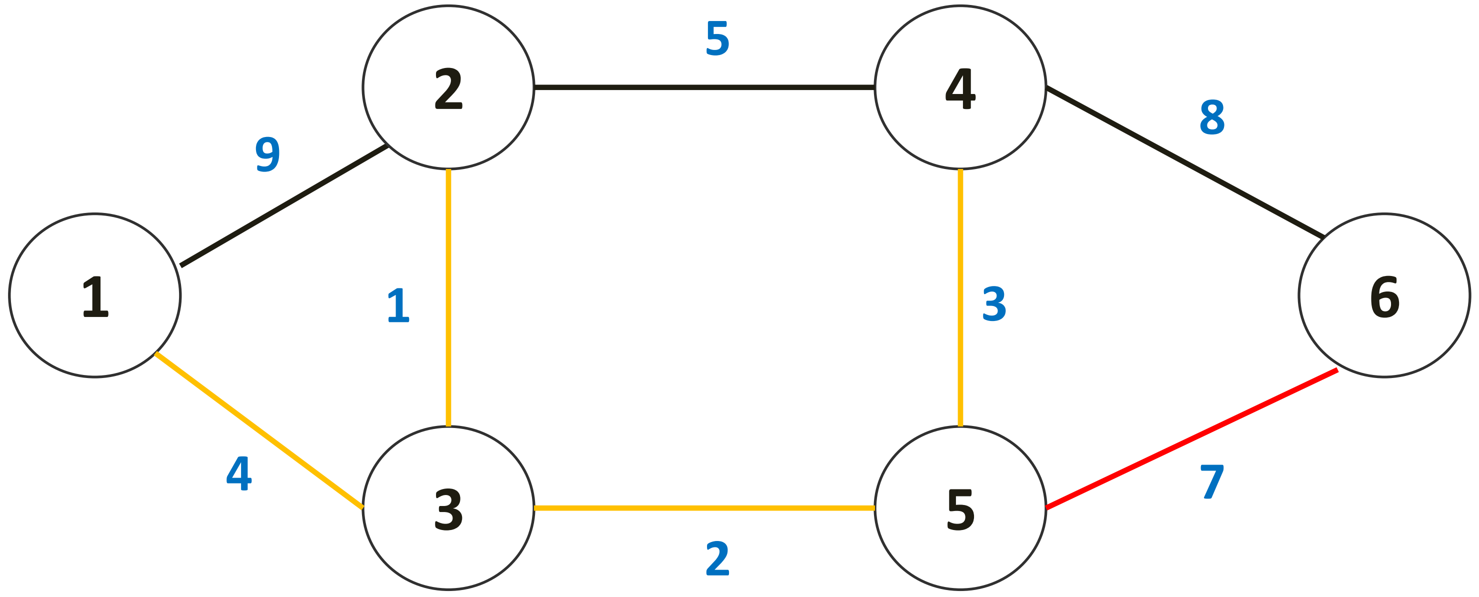
5번 정점과 6번 정점을 잇는 가중치 7의 간선을 추가하려고 한다.

이때 분리집합을 이용해서 두 집합이 어떤 컴포넌트에 있는지 알아 보자.

크루스칼 알고리즘



edge_count : 5 | sum : 17



정점	1	2	3	4	5	6
부모	2	2	2	2	2	2

간선	2 – 3	3 – 5	4 – 5	1 – 3	2 – 4	5 – 6	4 – 6	1 - 2
가중치	1	2	3	4	5	7	8	9

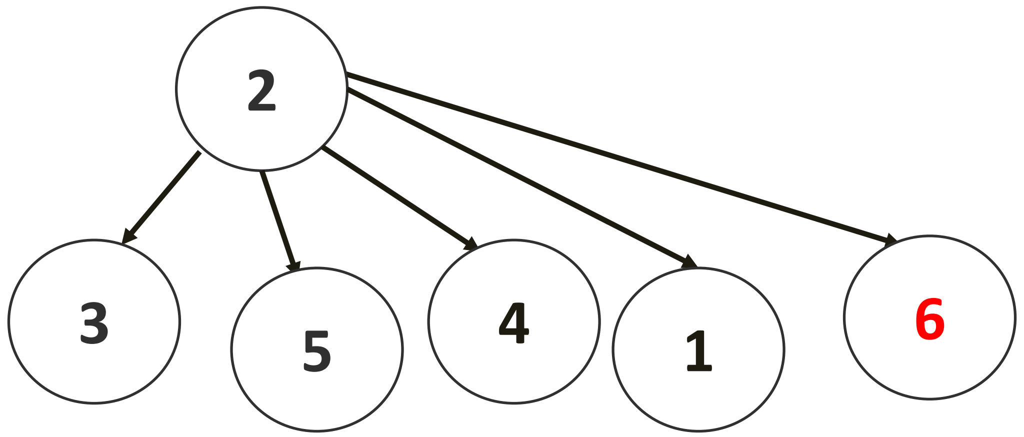
선택 : 5 – 6 간선

5번과 6번 정점은 같은 컴포넌트 내에 존재 하지 않으므로 사이클을 형성 하지 않는다.

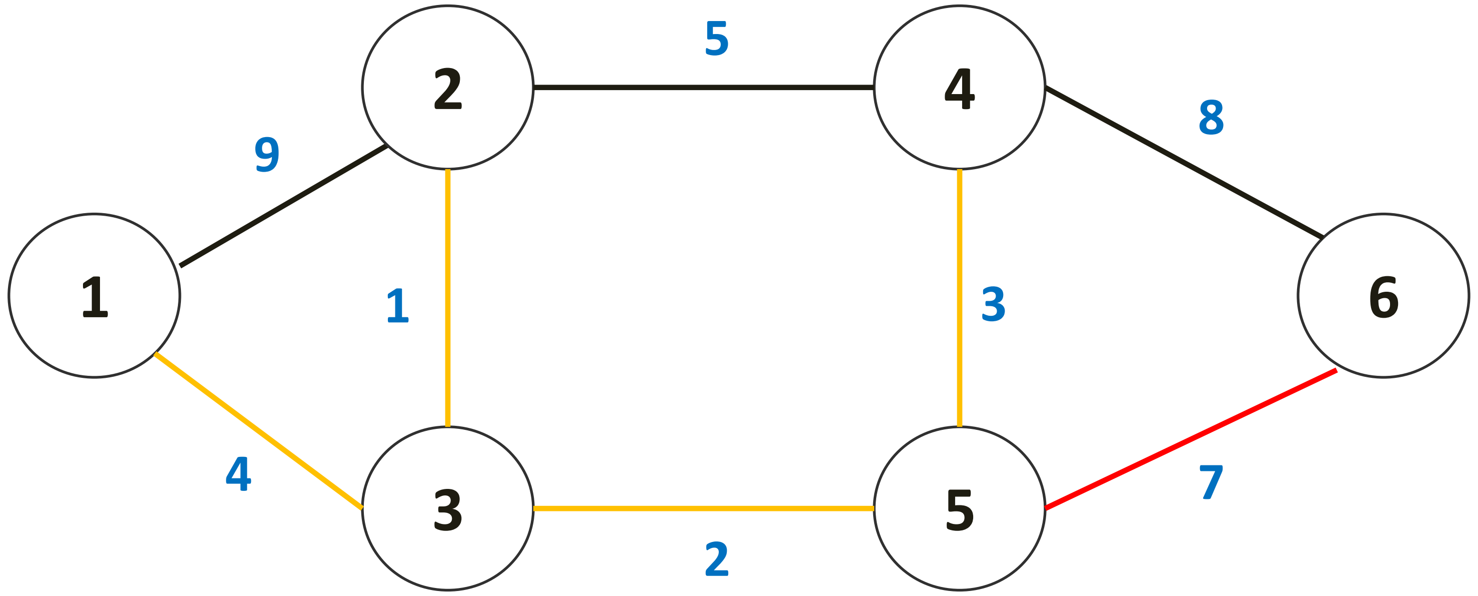
분리집합으로 정점을 병합한 후에 현재 간선을 택해준다.

현재 선택한 간선은 5개, 합계는 17이다.

크루스칼 알고리즘



edge_count : 5 | sum : 17



정점	1	2	3	4	5	6
부모	2	2	2	2	2	2

간선	2 – 3	3 – 5	4 – 5	1 – 3	2 – 4	5 – 6	4 – 6	1 - 2
가중치	1	2	3	4	5	7	8	9

선택 : 5 – 6 간선

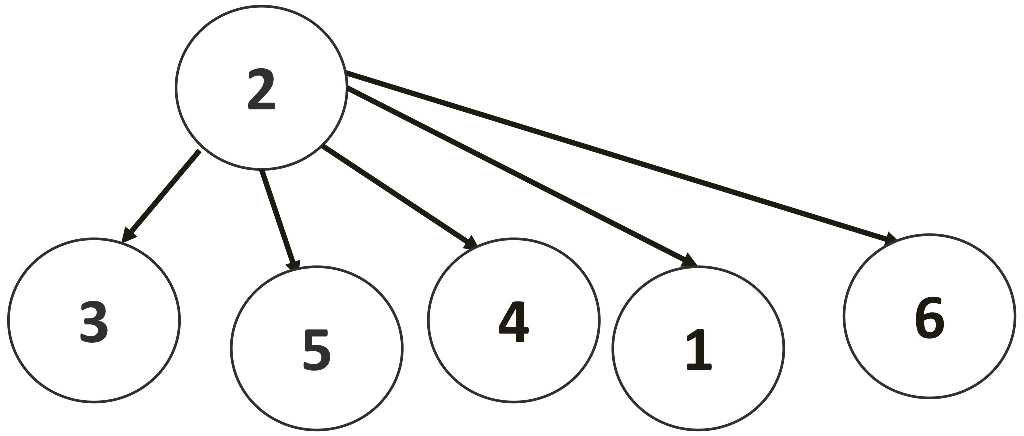
5번과 6번 정점은 같은 컴포넌트 내에 존재 하지 않으므로 사이클을 형성 하지 않는다.

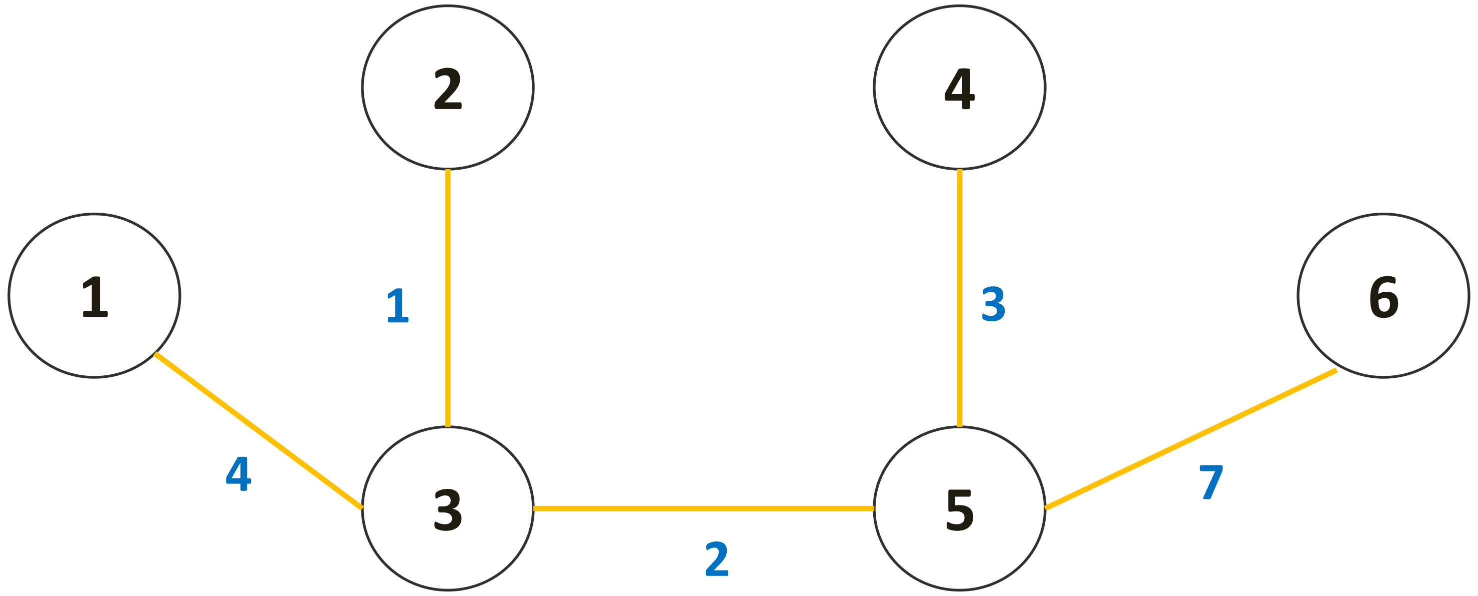
분리집합으로 정점을 병합한 후에 현재 간선을 택해준다.

현재 선택한 간선은 5개, 합계는 17이다.

크루스칼 알고리즘

edge_count : 5 | sum : 17





정점	1	2	3	4	5	6
부모	2	2	2	2	2	2

간선	2 – 3	3 – 5	4 – 5	1 – 3	2 – 4	5 – 6	4 – 6	1 - 2
가중치	1	2	3	4	5	7	8	9

간선(5개)이 정점(6개) – 1개가 되었으므로 종료한다.

신장 트리는 위와 같이 형성 되었으며, 가중치의 합은 17로 최소이다.

그런데 왜 간선을 V-1개만 골라도 되는지, 가중치 작은 것 부터 골라도 되는지 궁금하지 않은가?

신장 트리의 간선은 항상 V - 1개인가?

가정 : 신장 트리의 간선 수가 V - 1개가 아닐 수가 있다.

Case#1 : 간선 수가 V -1개 보다 적다.

- 트리는 사이클이 없는 연결 그래프이다. 트리의 정점 수가 V일 때 간선 수(E)는 V - 1개로, 이는 트리의 기본 성질 중 하나이다.
- 트리의 간선 수가 V – 1개보다 적으면, 모든 정점을 연결하기에 충분한 간선이 부족하다. 따라서 트리가 연결 그래프의 조건을 만족하지 못하게 된다.
- 신장 트리는 원본 그래프 G의 모든 정점을 포함하는 연결 그래프로, 트리와 비슷한 형태를 가진다. 이때 간선이 V – 1개 보다 적으면 모든 간선을 연결 할 수 없으므로 연결 그래프를 형성할 수 없으며 이는 신장 트리의 정의에 모순된다.

Case#2 : 간선 수가 V -1개 보다 많다.

- 트리는 사이클이 없는 그래프이다. 만약 간선의 개수가 V – 1개를 초과하면 추가된 간선으로 인해 사이클이 형성된다.
- 아까 보았듯이 V개의 정점을 가진 트리는 간선을 최대 V – 1개만을 가질 수 있다. 만약 간선이 하나 더 추가 된다면 V개의 정점 사이에 최소 하나의 사이클이 형성 되는데, 이는 그래프 이론의 기본 정리 중 하나로 사이클의 존재를 보장한다.
- 신장 트리는 사이클이 없는 연결 그래프이다. 하지만 간선 수가 V - 1개 보다 많으면 반드시 사이클이 존재하게 되어 신장 트리의 조건을 위배하게 된다.

신장 트리의 간선 수는 반드시 V – 1개 여야 한다.

최소 가중치 간선만 골라도 되는가?

가정 1 : 크루스칼 알고리즘이 가중치가 작은 간선부터 선택하지 않는다고 가정하자.

가정 2 : 이 경우, 어떤 시점에서 가중치가 큰 간선이 먼저 선택될 수 있다. 이를 통해 생성된 트리를 T1이라고 한다.

가정 3 : 실제 최소 신장 트리를 T2 라고 한다.

- T1은 T2가 아니며, T1의 간선 가중치 합은 T2보다 크거나 같다. T1과 T2는 모두 모든 정점을 연결하므로 최소 하나 이상의 간선을 공유하게 된다.
- T1에는 T2에 없는 간선 E1이 존재한다고 하자. 반대로 T2에는 T1에 없는 간선 E2가 존재한다고 하자. 이때 E1의 가중치는 E2보다 크다고 가정한다.
- T1에 E2를 추가하면 사이클이 형성되는데, 이 사이클 내에서는 반드시 E1과 E2가 존재한다.
- 사이클에서 E1을 제거하면 사이클이 해소 된다. 이 과정에서 그래프는 여전히 연결 상태를 유지한다.
- 이로 인해 새로운 신장 트리 T3가 생성되는데, T3의 간선 가중치의 합은 T1의 간선 가중치보다 작다.
- 정의로 인해 T1의 간선을 바꾸어도 새로 만든 트리가 가중치가 같거나 더 커야 한다. 그러나 새롭게 만든 T3는 T1보다 가중치의 합이 작으므로 T1의 방법으로 만들어진 신장 트리는 절대 MST가 될 수 없다.

따라서, 크루스칼 알고리즘이 가중치가 작은 간선부터 선택하지 않는다면 항상 더 작은 가중치로 대체할 수 있는 간선이 존재하게 되며, 이로 인해 최소 신장 트리를 보장하지 못한다. 크루스칼 알고리즘의 본질은 가중치 순서에 따르는 것에 있다.