최소 스패닝 트리

Minimum Spanning Tree

스패닝 트리란?

- 무향 연결 그래프에서 간선들을 부분적으로 뽑아서 만들 수 있는 그래프의 정점 개수와 같은 정점 개수를 가지는 트리

- 트리니까 당연히 연결 그래프

최소 스패닝 트리란?

- 스패닝 트리를 구성하는 모든 간선의 가중치의 합이 가장 작은 스패닝 트리

그래프에서 MST 구하기!

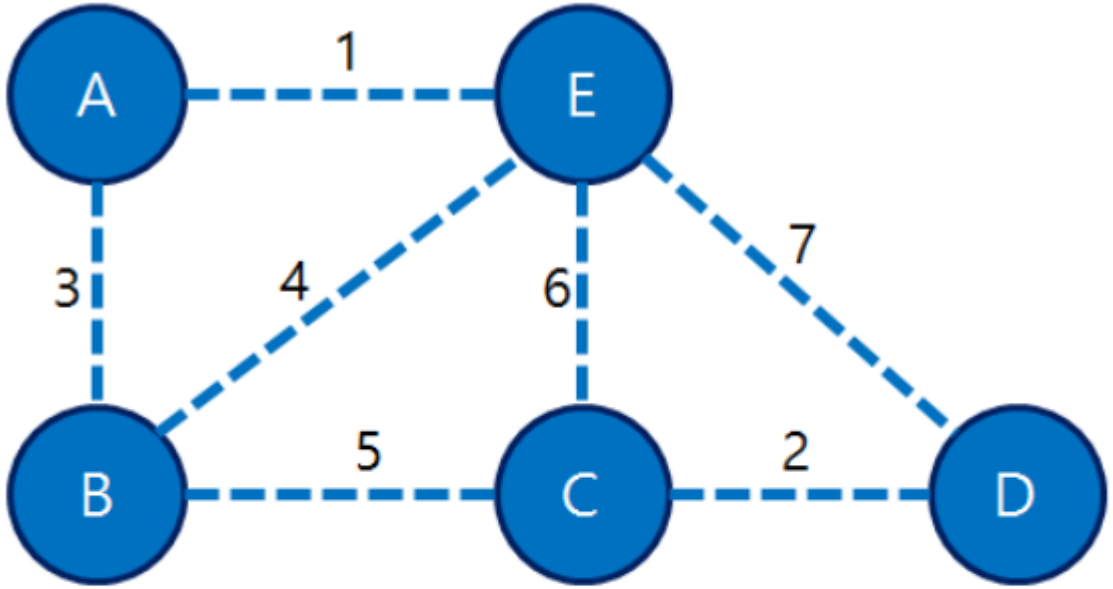
- 프림 알고리즘(Prim`s algorithm)
  - 다익스트라와 유사함
- 크루스칼 알고리즘(Kruskal`s algorithm)
  - 편함, 보통 이걸로 구현

크루스칼 알고리즘

- 간선들을 가중치 순으로 오름차순 정렬
- 간선들을 훑으면서 양쪽 정점이 연결되어 있지 않으면 있으면 연결
- 간선 v-1개가 뽑혔을 때 그 간선들과 정점들이 이루는 그래프가 MST

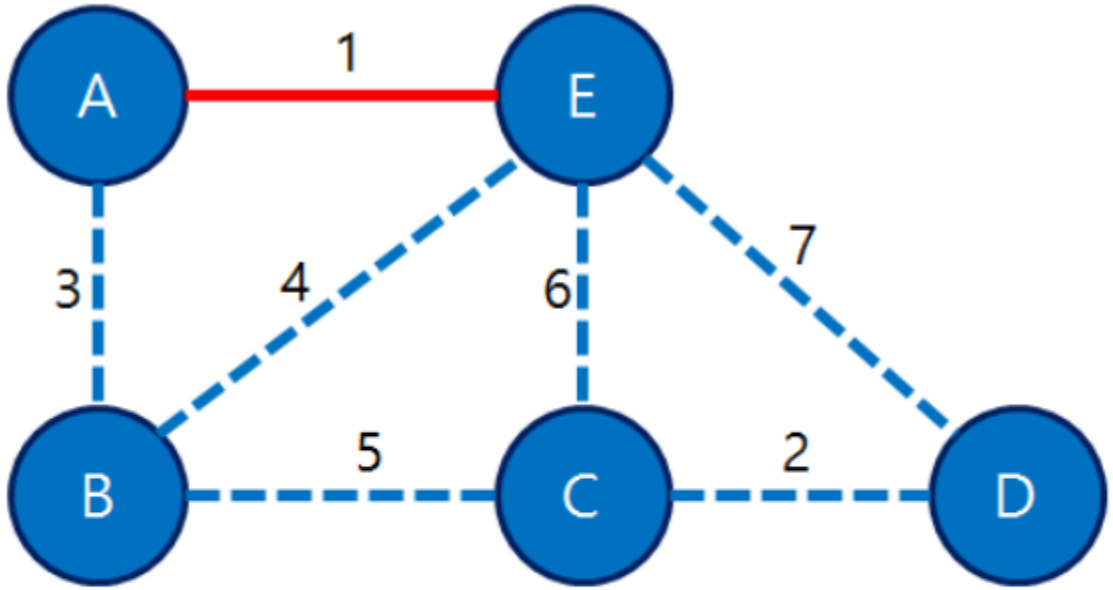
* 간선을 가중치 순으로 정렬함 -> 가중치가 같은 간선이 여러 개일 경우 MST가 여러 개 나올 수도 있음

크루스칼 알고리즘



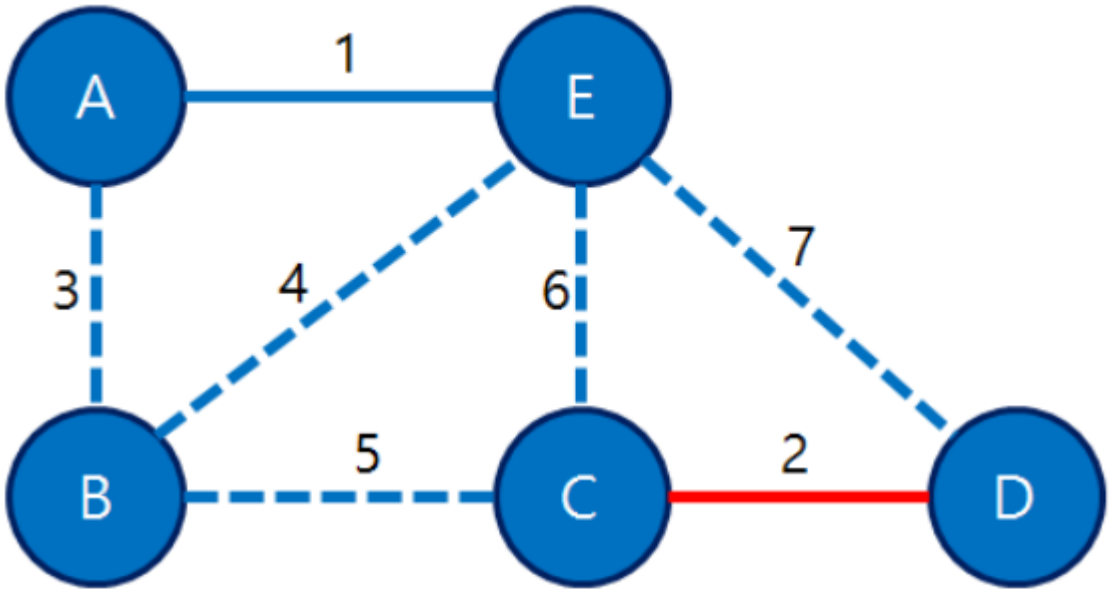
AE	CD	AB	BE	BC	CE	DE
1	2	3	4	5	6	7

크루스칼 알고리즘



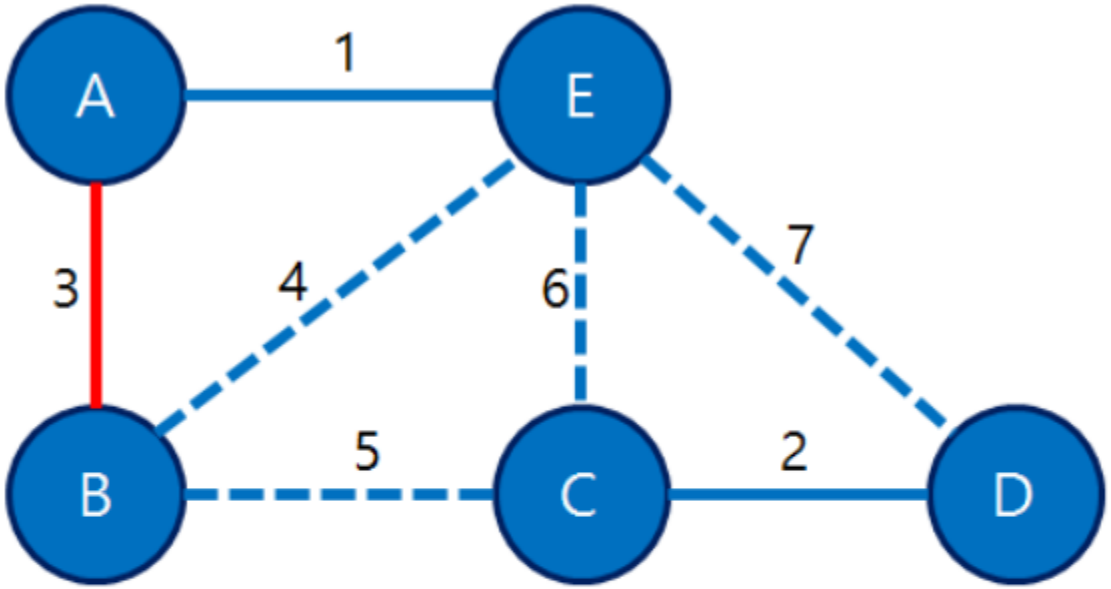
AE	CD	AB	BE	BC	CE	DE
1	2	3	4	5	6	7

크루스칼 알고리즘



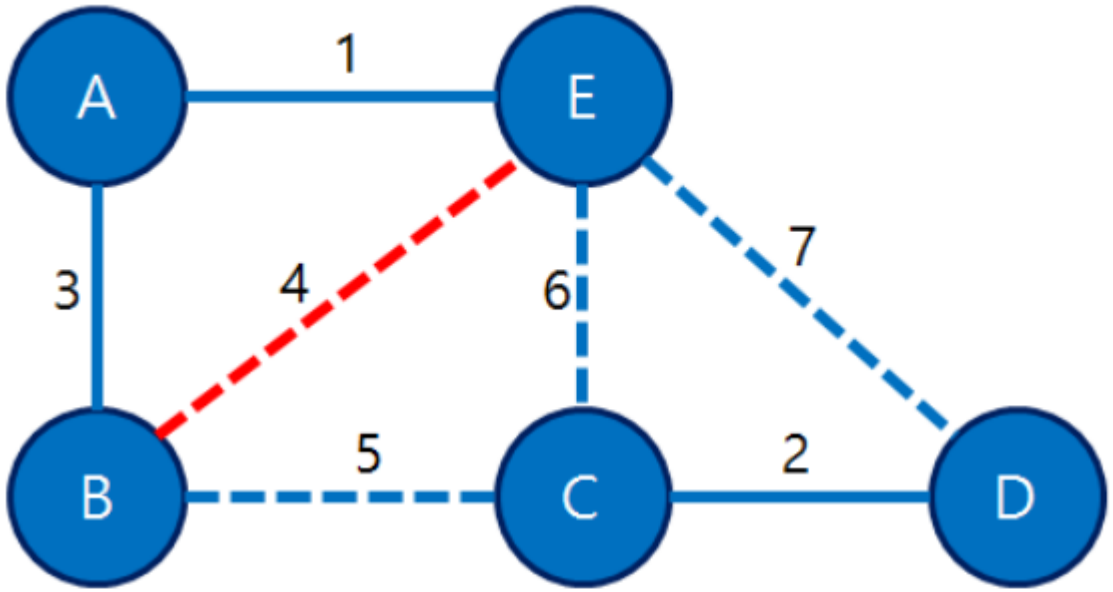
AE	CD	AB	BE	BC	CE	DE
1	2	3	4	5	6	7

크루스칼 알고리즘



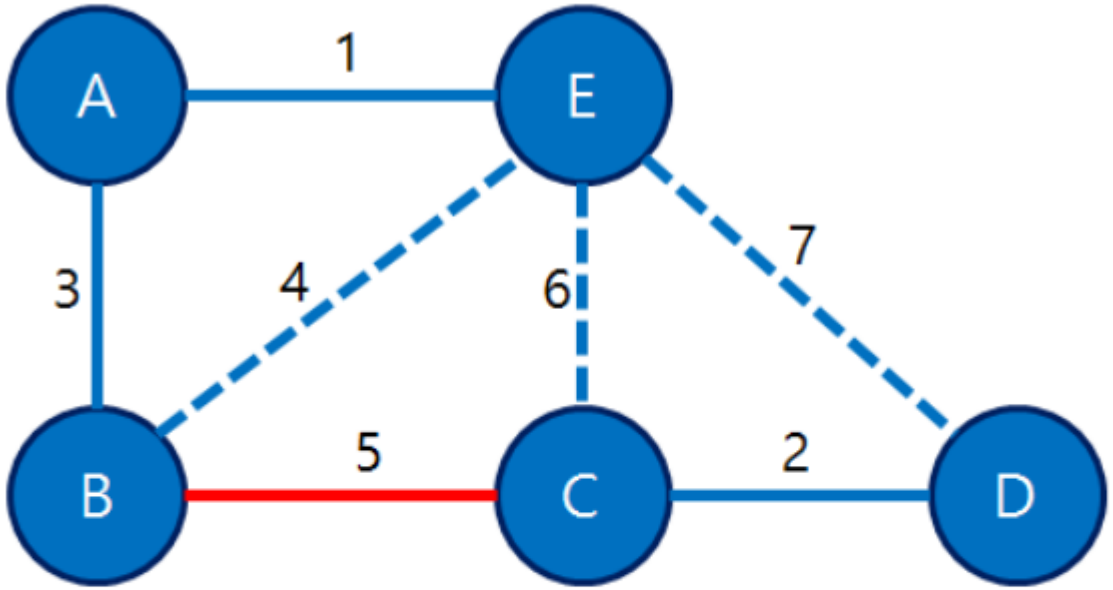
AE	CD	AB	BE	BC	CE	DE
1	2	3	4	5	6	7

크루스칼 알고리즘



AE	CD	AB	BE	BC	CE	DE
1	2	3	4	5	6	7

크루스칼 알고리즘



AE	CD	AB	BE	BC	CE	DE
1	2	3	4	5	6	7

크루스칼 알고리즘

- 4개의 간선을 뽑았고 MST의 가중치는 1 + 2 + 3 + 5 = 11

- 11보다 작은 가중치를 가지는 스패닝 트리는 존재하지 않음

- 무조건 가중치가 최선인 간선을 뽑고 절대 결정을 되돌리지 않음
  -> 무슨 알고리즘 ?

크루스칼 알고리즘

- 간선의 개수가 E일 때 간선을 가중치의 오름차순으로 정렬
  => O(ElogE)
- 간선이 연결하고 있는 두개의 정점이 연결되어 있는지 판단
  => Find
- 연결되어 있지 않으면 두 정점을 연결
  => Union

  * 유니온-파인드를 구현할 줄 알아야 함

1922_네트워크 연결

- 컴퓨터 수 N개, 간선의 수 M개
- M개의 a, b, c가 주어지고 a와 b를 잇는 가중치c의 간선이 존재

예제 입력 1 복사

```
6
9
1 2 5
1 3 4
2 3 2
2 4 7
3 4 6
3 5 11
4 5 3
4 6 8
5 6 8
```

1922_네트워크 연결

예제 입력 1 복사

```
6
9
1 2 5
1 3 4
2 3 2
2 4 7
3 4 6
3 5 11
4 5 3
4 6 8
5 6 8
```

1922_네트워크 연결

```
12  int n, m, ans;
13  vector<pair<int, pair<int, int> > > edges;
14  int p[1001];
```

```
34      cin >> n >> m;
35
36      for (int i = 1; i <= n; i++)
37          p[i] = i;
38
39      for (int i = 0; i < m; i++) {
40          int a, b, c;
41          cin >> a >> b >> c;
42          edges.push_back({ c, {a, b} });
43      }
44      sort(edges.begin(), edges.end());
45
46      for (auto i : edges) {
47          int cur_cost = i.first;
48          int v1 = i.second.first;
49          int v2 = i.second.second;
50          if (Find(v1) != Find(v2)) {
51              Union(v1, v2);
52              ans += cur_cost;
53          }
54      }
55      cout << ans;
56      return 0;
```

p배열의 의미?

유니온 파인드

```
16  int Find(int x) {
17      if (x == p[x])
18          return x;
19      return Find(p[x]);
20  }
```

유니온 파인드

```
int Find(int x) {
    if (x == p[x])
        return x;
    return p[x] = Find(p[x]);
}
```

유니온 파인드

```
22  void Union(int a, int b) {
23      int aa = Find(a);
24      int bb = Find(b);
25      if (aa != bb) {
26          p[aa] = bb;
27      }
28  }
```