11. Minimum spanning tree & Shortest Path

Div. 3 알고리즘 스터디 / 임지환

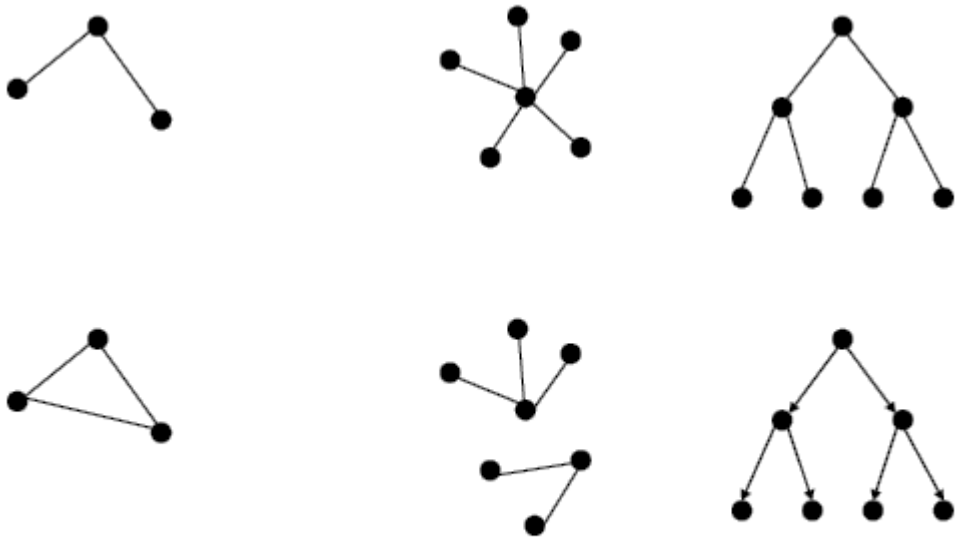
마지막 수업……

1. 더 이상의 강의는 없을 것 같습니다..아마도.
2. 이후 스터디 방식
   - ACM - 스터디원들이 자체적으로 진행. "멘토" 역할만 합니다.
   - SGCC – 모르겠습니다.
3. With raararaara 그룹은 폭파 예정.
4. 강의자료는 각 학회 및 동아리 드라이브에 공유 예정.

Review : Tree

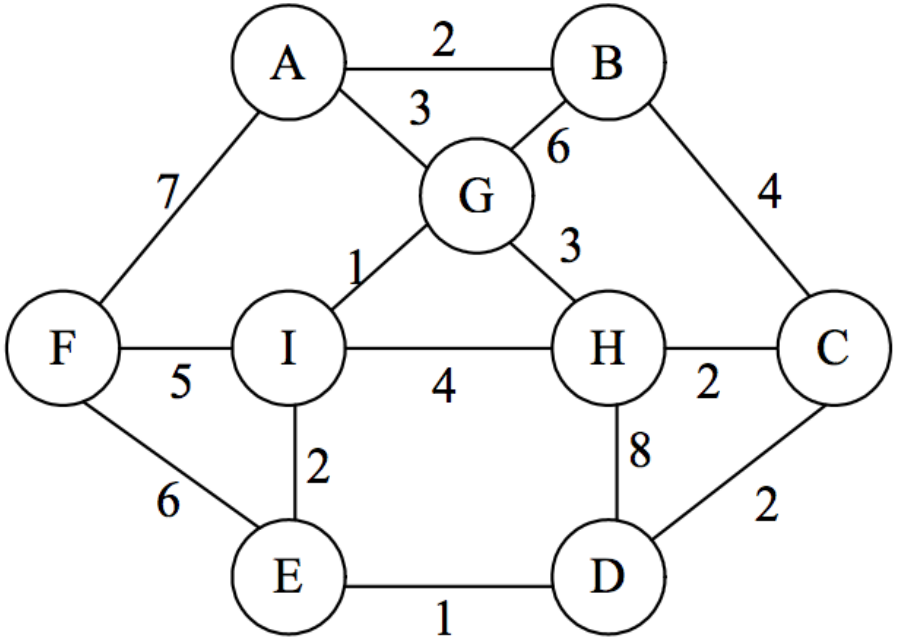
- 그래프 이론에서의 "루트 없는 트리"

Connected, Acyclic, undirected graph



Weighted graph

- 그래프인데 가중치가 있는 그래프.



Weighted Edge Implementation

```cpp
#include <vector>
#include <utility>
using namespace std;
using ii = pair<int, int>;

vector<ii> adj[MXN];

void input(int u, int v, int w) {
    adj[u].push_back(ii(v, w));
    adj[v].push_back(ii(u, w));
}
```

Weighted Edge Implementation

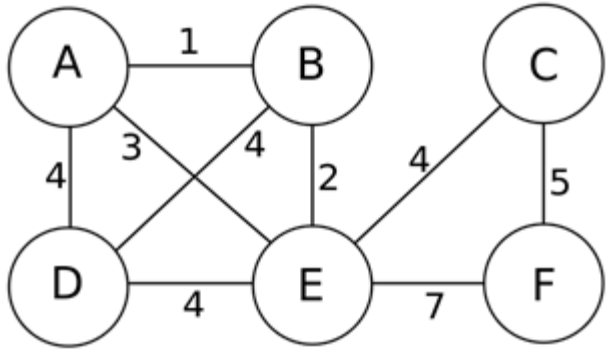
```cpp
#include <vector>
using namespace std;

struct Edge {
    int u, v, w;
};
vector<Edge> E;
void input(int u, int v, int w) {
    E.push_back({ u, v, w });
}
```

Weighted Edge Implementation

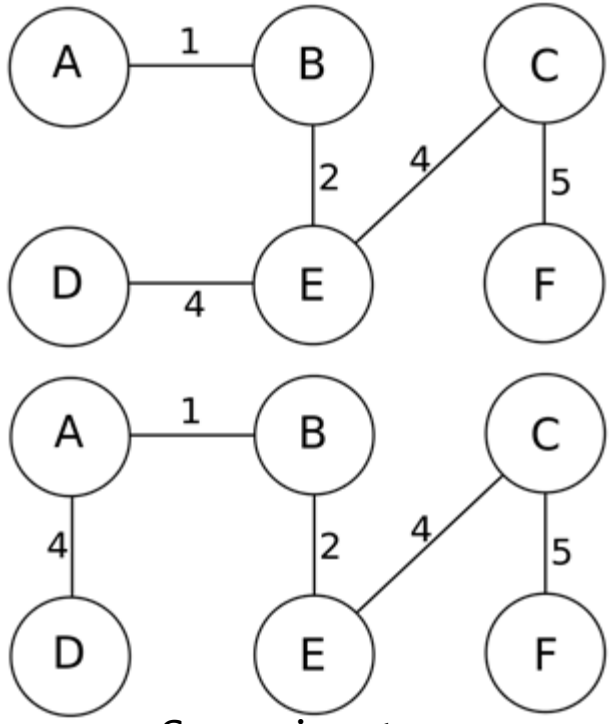
```
struct Edge {
    int u, v, w;
} E[MXN];

int idx;
void input(int u, int v, int w) {
    E[idx++] = { u, v, w };
}
```

Spanning tree

- 그래프를 구성하는 정점 집합 전부와 간선 집합의 부분집합으로 구성



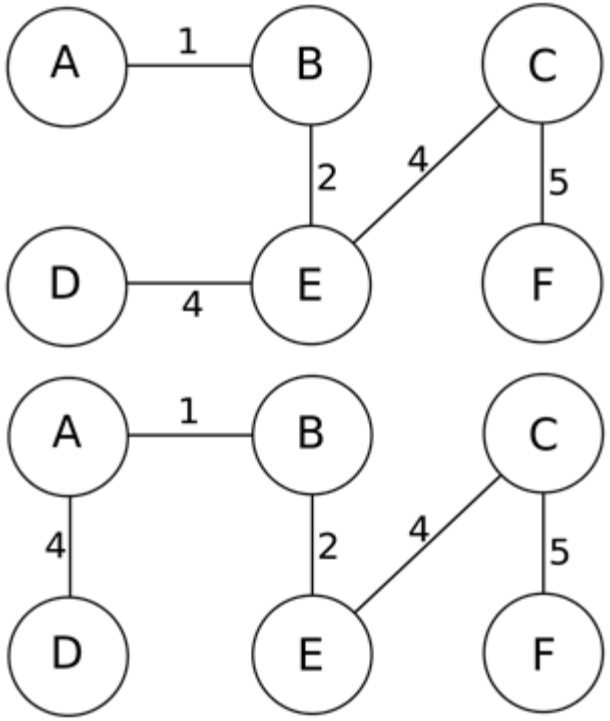
Graph



Spanning tree

Minimum spanning tree

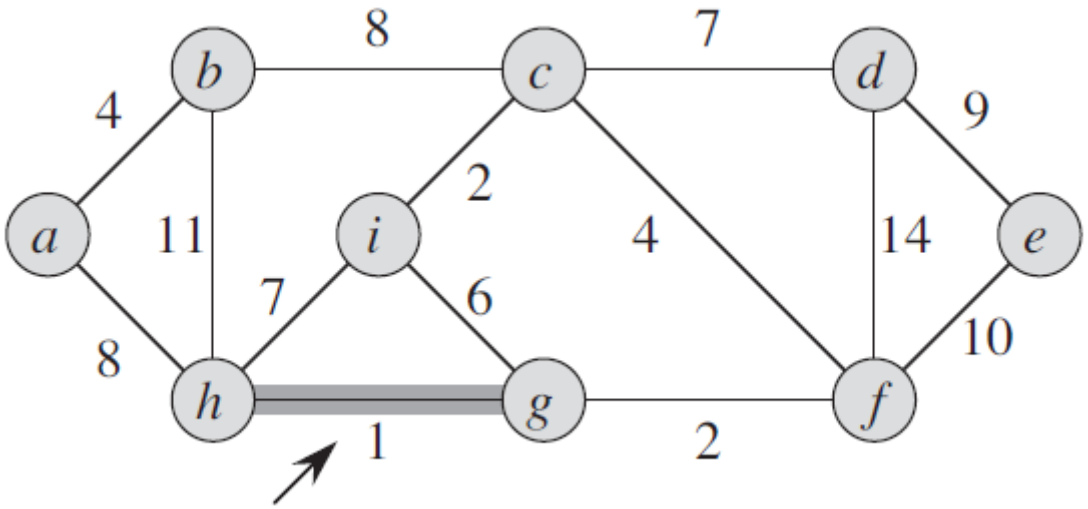
- 가중치의 합이 최소인 spanning tree



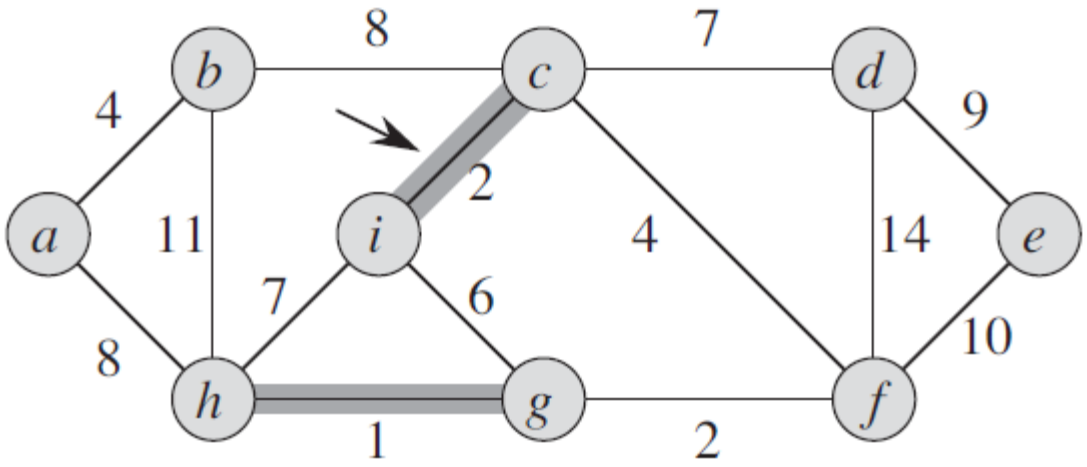
Kruskal's Algorithm

- 가중치가 작은 간선들부터 선택.
- 단 cycle은 생기지 않도록 간선을 선택.
- 정점 개수를 |V|라 하면 |V| - 1만큼 반복.

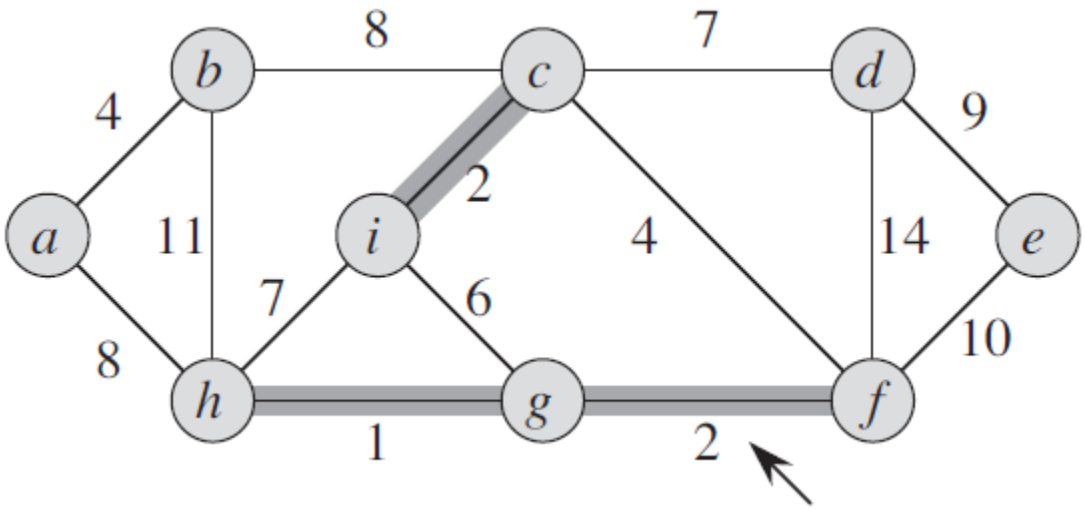
Kruskal's Algorithm



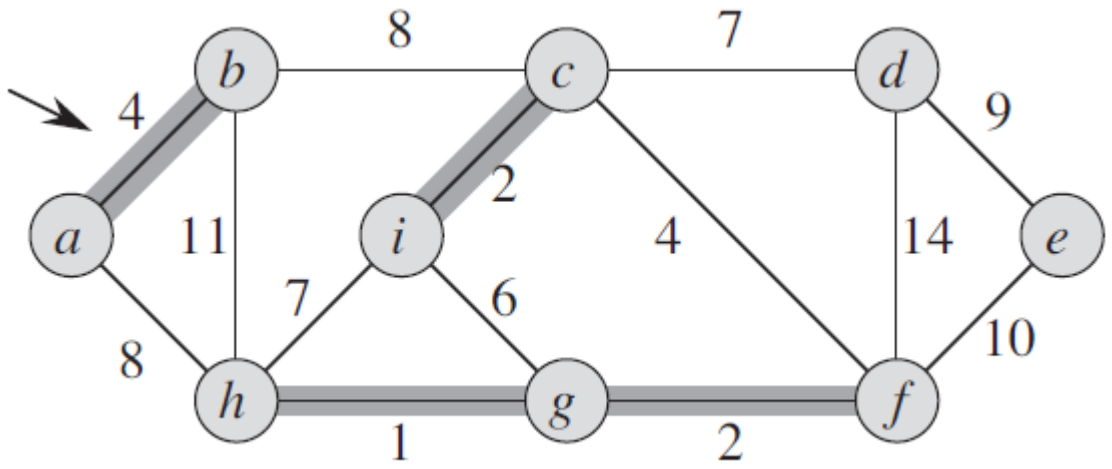
Kruskal's Algorithm



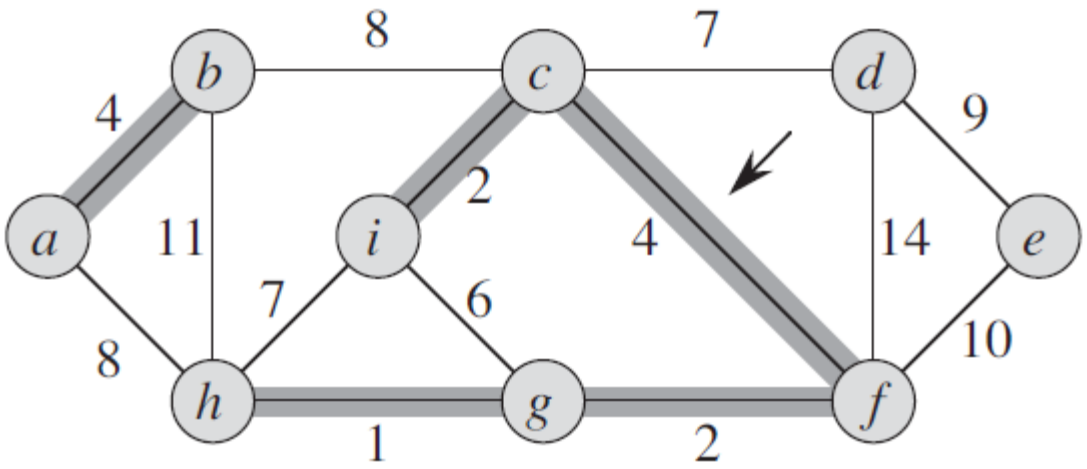
Kruskal's Algorithm



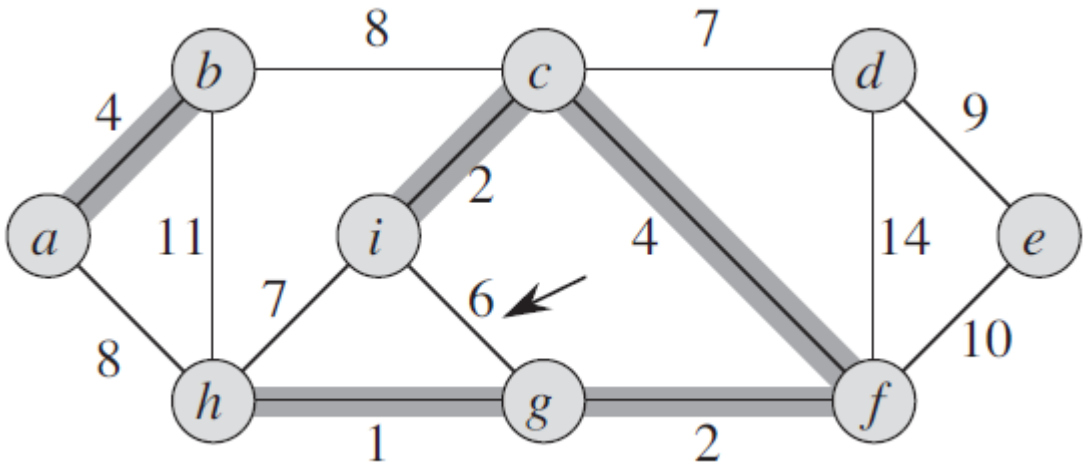
Kruskal's Algorithm



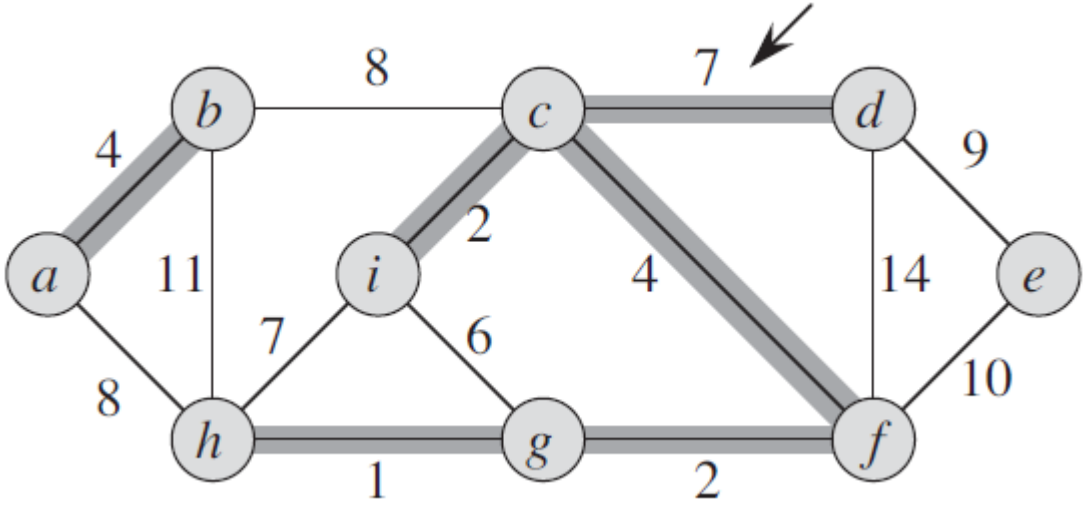
Kruskal's Algorithm



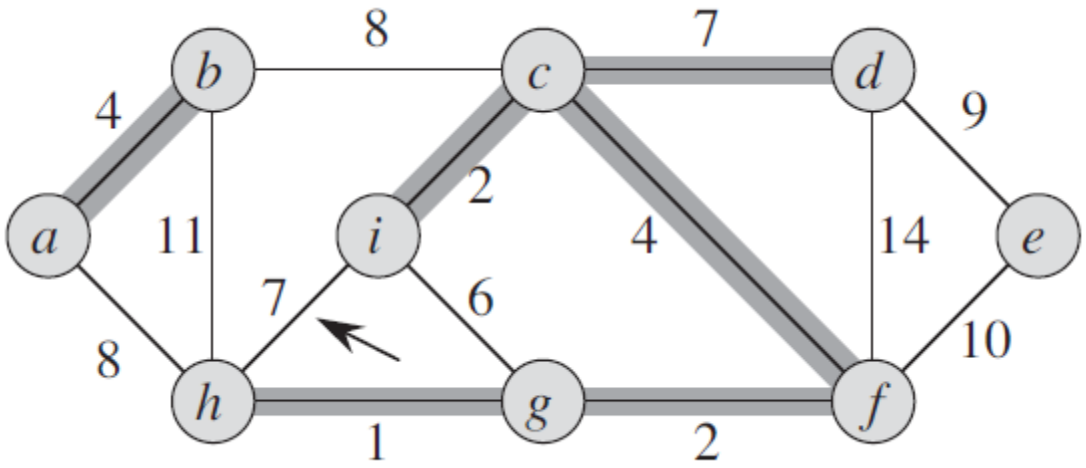
Kruskal's Algorithm



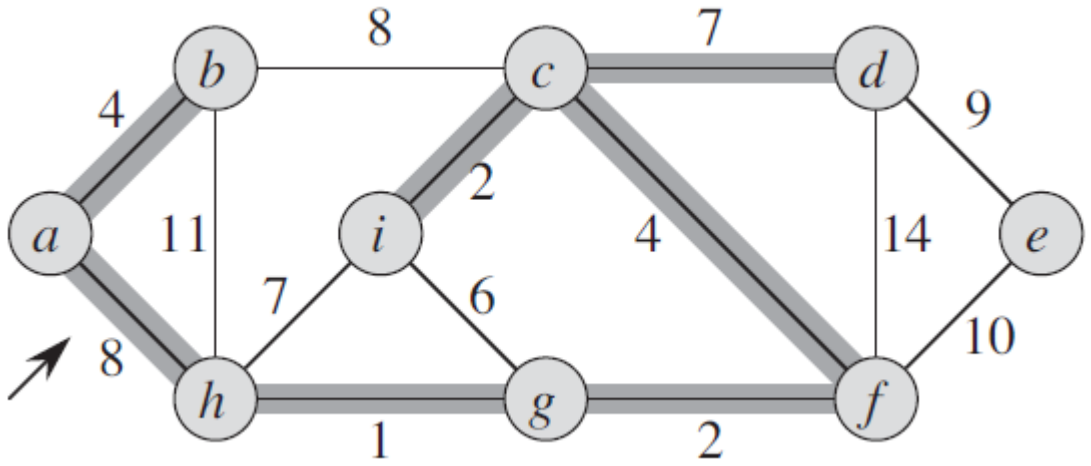
Kruskal's Algorithm



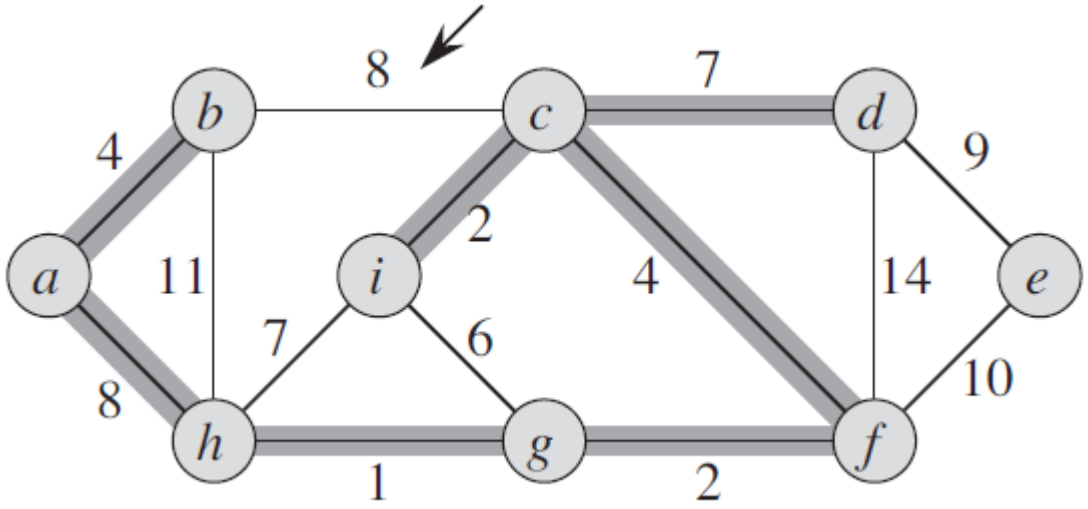
Kruskal's Algorithm



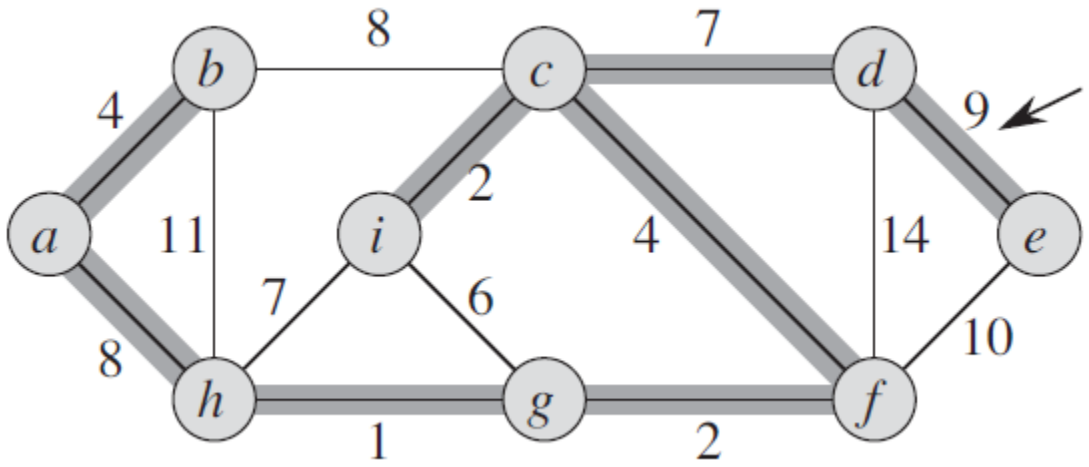
Kruskal's Algorithm



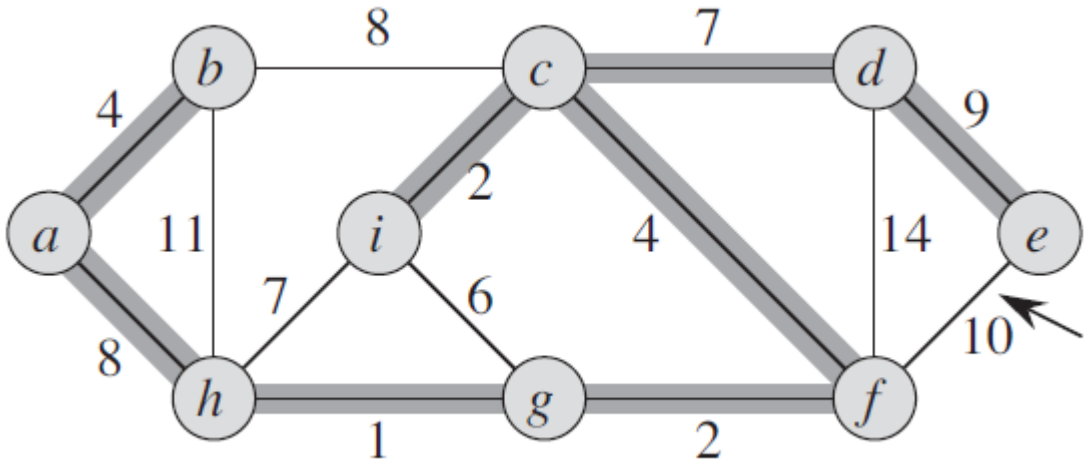
Kruskal's Algorithm



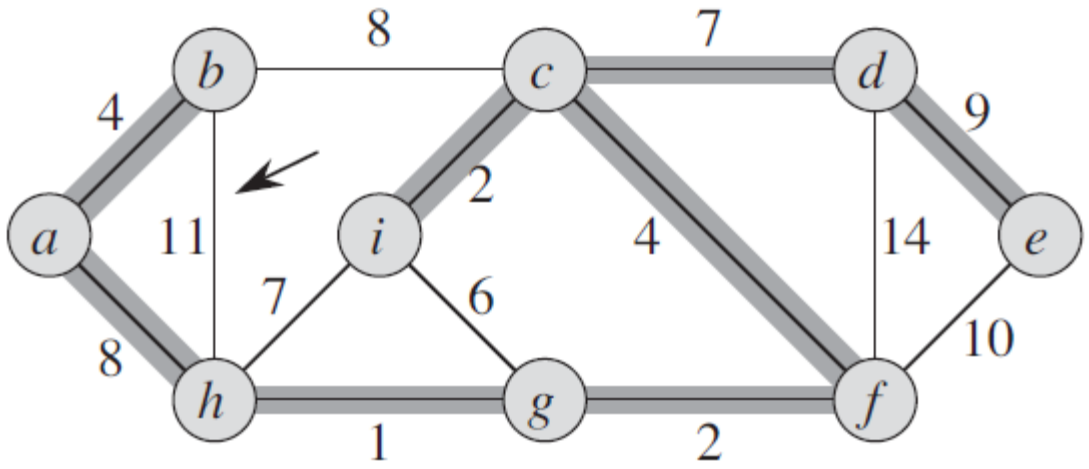
Kruskal's Algorithm



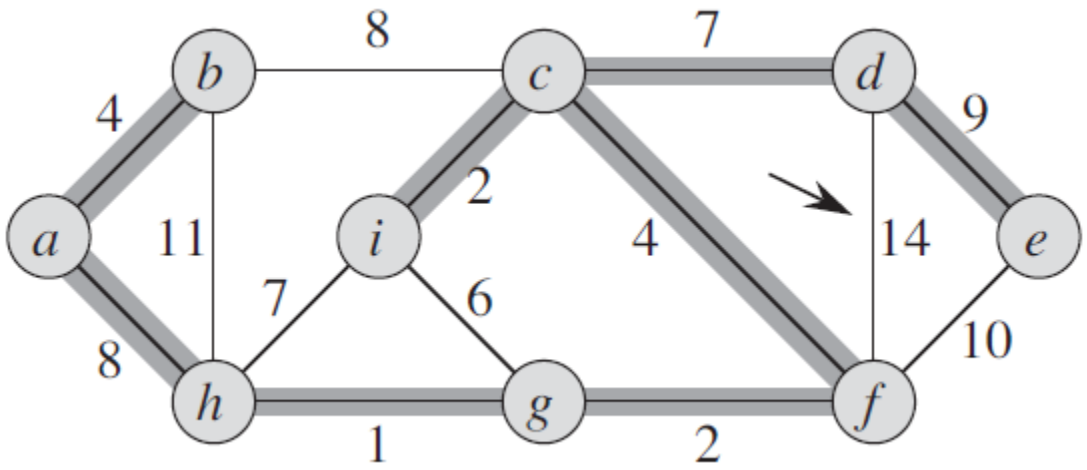
Kruskal's Algorithm



Kruskal's Algorithm



Kruskal's Algorithm



어떻게 한다고?

1. 가중치가 작은 간선부터 선택 ➡ sort
2. Cycle이 생기지 않도록 선택 ➡ by Union Find
3. |V| - 1번 만큼 반복

#1922 네트워크 연결

- 컴퓨터끼리 모두 연결하는 네트워크를 구축
- 컴퓨터를 연결하는데 필요한 최소 비용 구하기

#1922 네트워크 연결

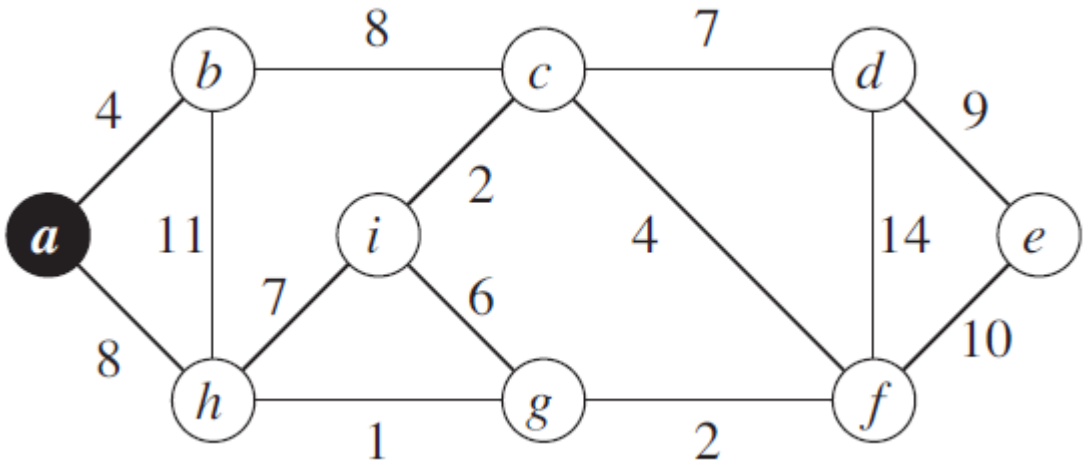
```
int find(int);
bool merge(int, int);

input Edge data
sort Edges

for each edge <u,v> in Edges:
    if successfully merge(u,v) then
        total_w += w(u,v)
        num(edg)++
    if num(edg) == |V|-1 then break
```

Time Complexity : $O(|E|\log|V|)$

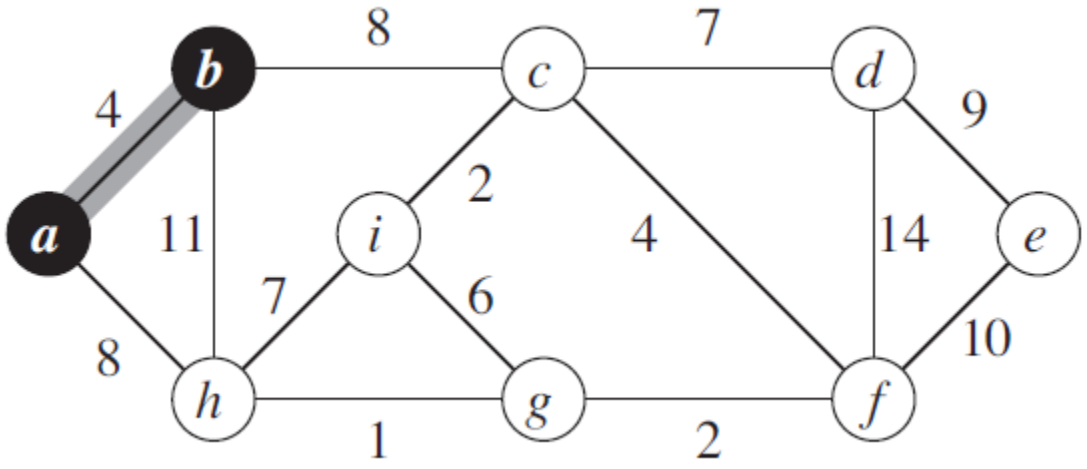
Prim's Algorithm

- 아무 정점부터 시작, Tree에 추가
- Tree에 있는 모든 정점에 대하여, 해당 정점에 incident한 간선들 중 가중치가 가장 낮은 간선 선택
- 물론 Cycle이 생기지는 않도록.

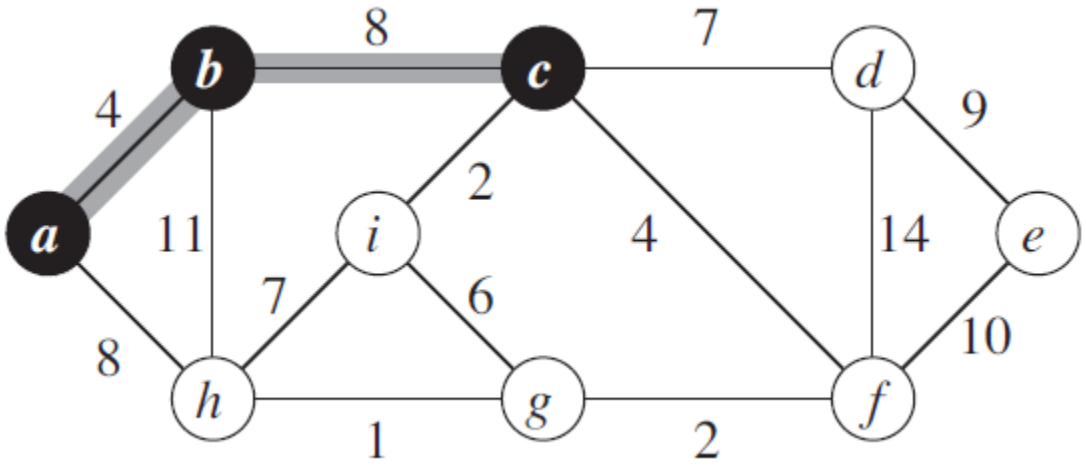
Prim's Algorithm



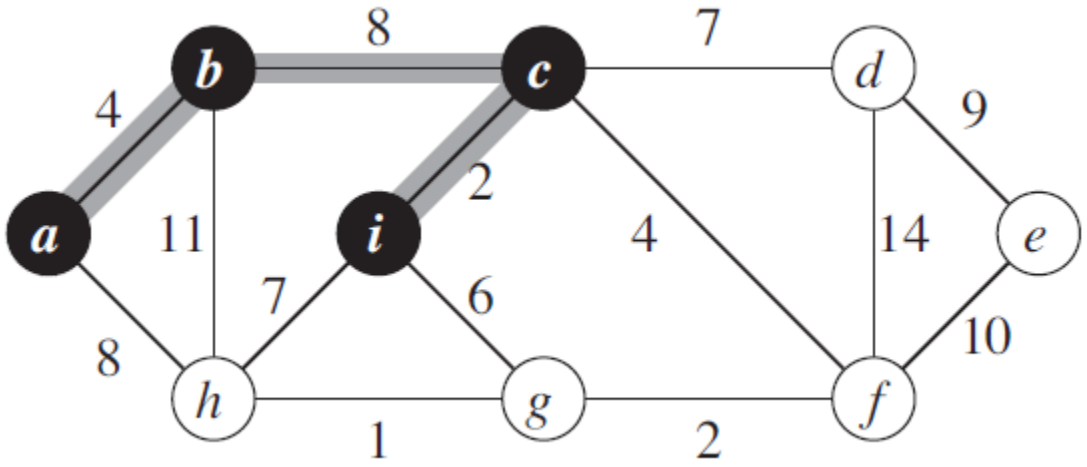
Prim's Algorithm



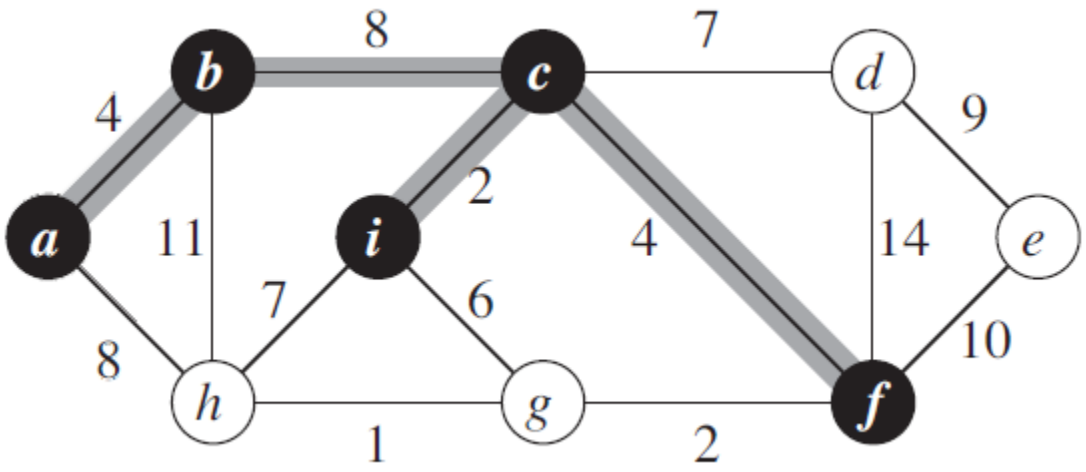
Prim's Algorithm



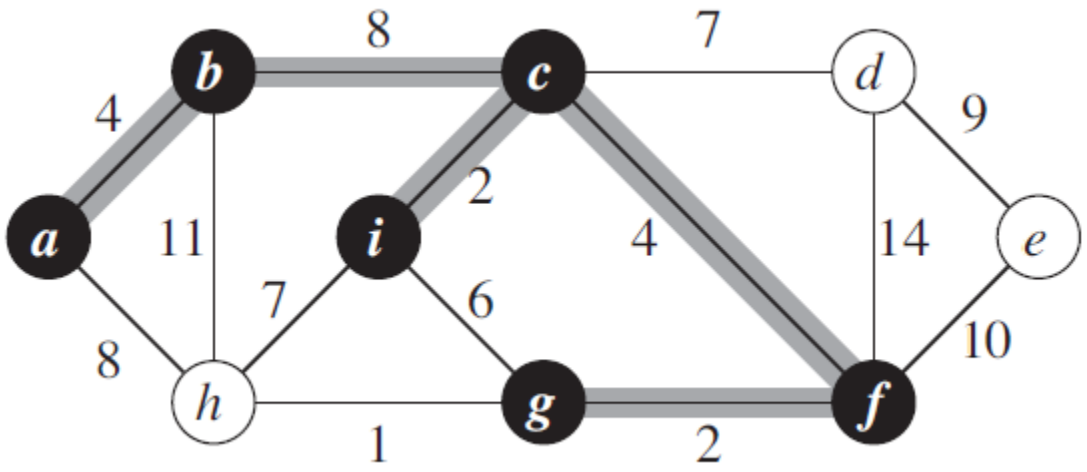
Prim's Algorithm



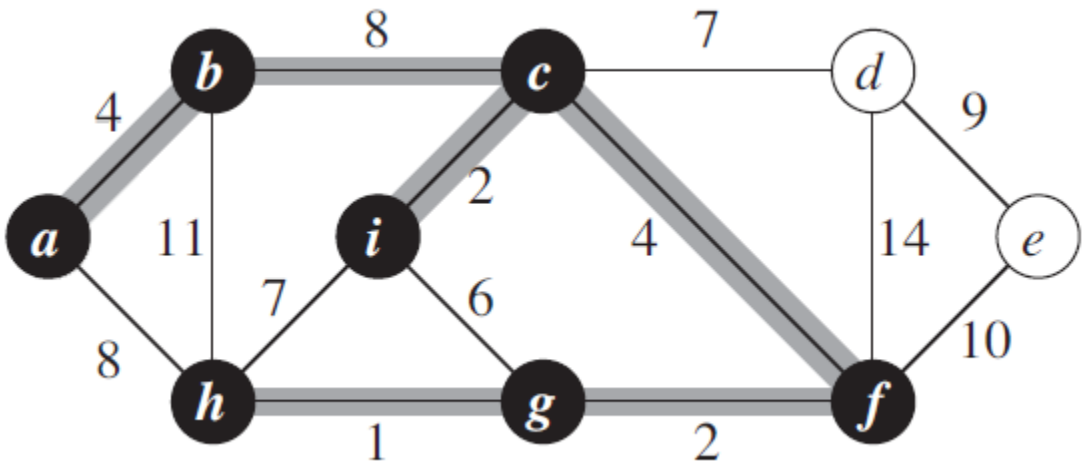
Prim's Algorithm



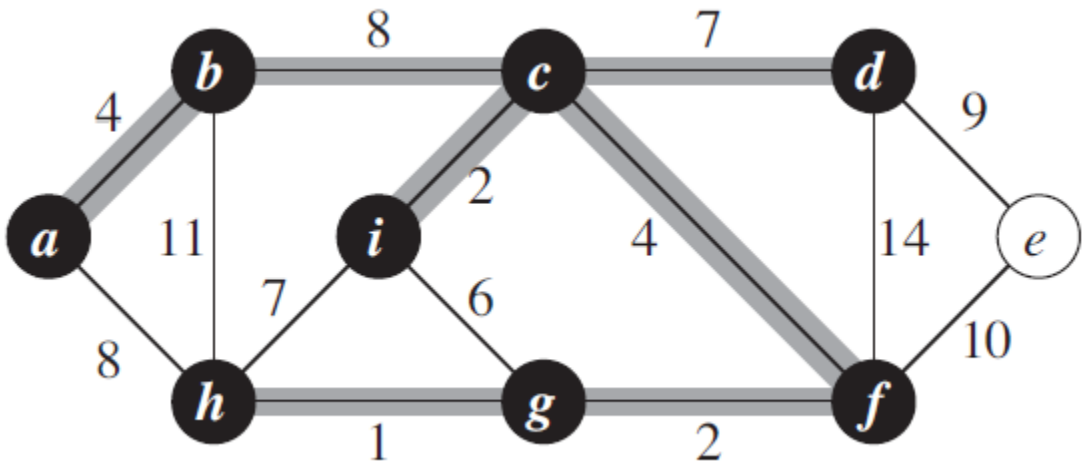
Prim's Algorithm



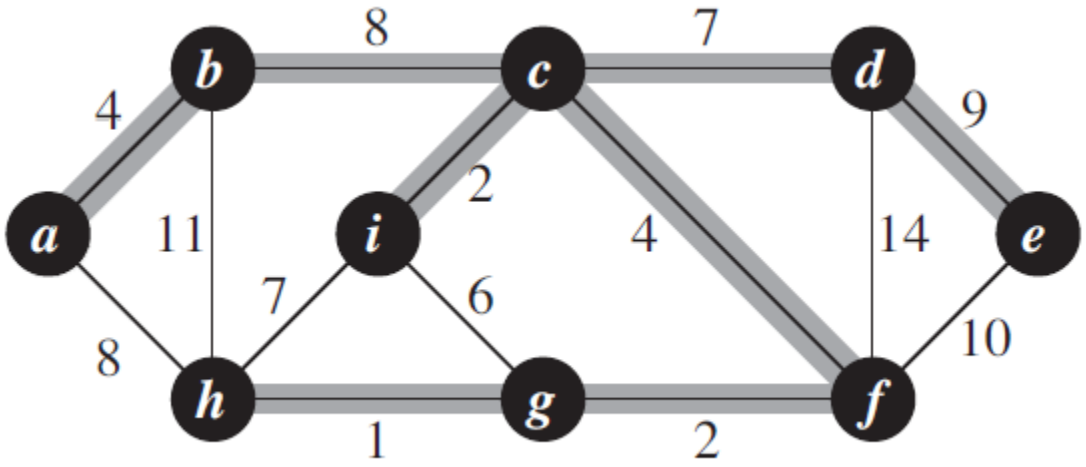
Prim's Algorithm



Prim's Algorithm



Prim's Algorithm



#1922 네트워크 연결

```
define vis[mxv], cnt = 0

input Edge

add (0,v0) in priority queue
while pq is not empty:
    if cnt == |V| then break
    while vis[pq.top()->v]
        pop pq
    u := pq's top, vis[u->v] = true, cnt++
    total_w += u->w
    for each neighbor v` of u:
        if !vis[v`] then insert(w(u,v`),v`) in pq
```

Time Complexity : $O(|E|\log|V|)$

Problem Set of MST

- 1922 네트워크 연결
- 1197 최소 스패닝 트리
- 6497 전력난
- 1647 도시 분할 계획
- 4386 별자리 만들기
- 4343 Arctic Network(*)
- 2887 행성 터널(*)
- 9372 상근이의 여행
- 9373 복도 뚫기(*)
- 1944 복제 로봇

최단경로

- 최단경로 분류
    1. Single-Source Shortest Path
    2. All-Pairs Shortest Path

최단경로

- 최단경로 분류
    1. **Single-Source Shortest Path**
    2. All-Pairs Shortest Path

최단경로

- 최단경로 분류
    1. **Single-Source Shortest Path**
    2. All-Pairs Shortest Path

최단경로

- 최단경로 분류
  1. **Single-Source Shortest Path**
     - Dijkstra's Algorithm
     - Bellman-Ford's Algorithm
  2. All-Pairs Shortest Path

최단경로

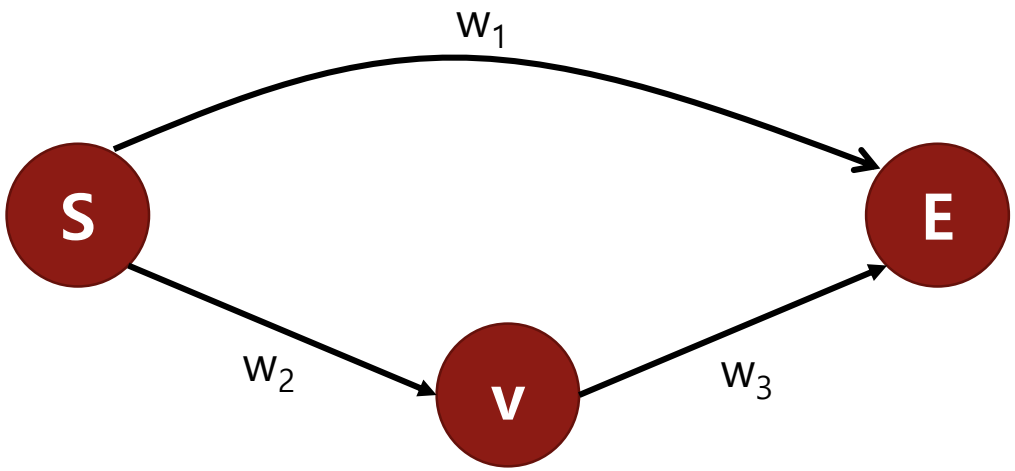
- 최단경로 분류
    1. **Single-Source Shortest Path**
        - Dijkstra's Algorithm
        - Bellman-Ford's Algorithm
    2. All-Pairs Shortest Path

BFS로는 안되나요?

BFS로는 안되나요?

- 예. 그때는 간선의 가중치가 없었습니다.

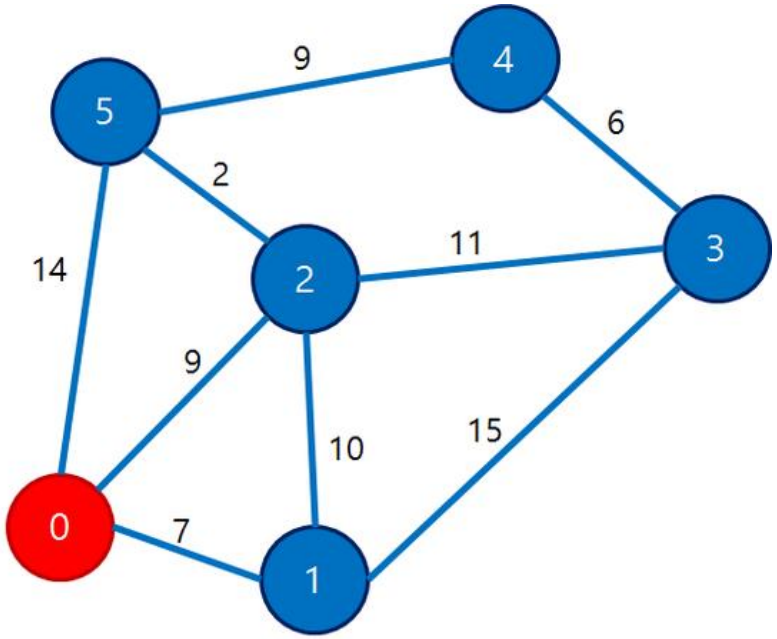
what if $w > w_2 + w_3$?



Dijkstra's Algorithm

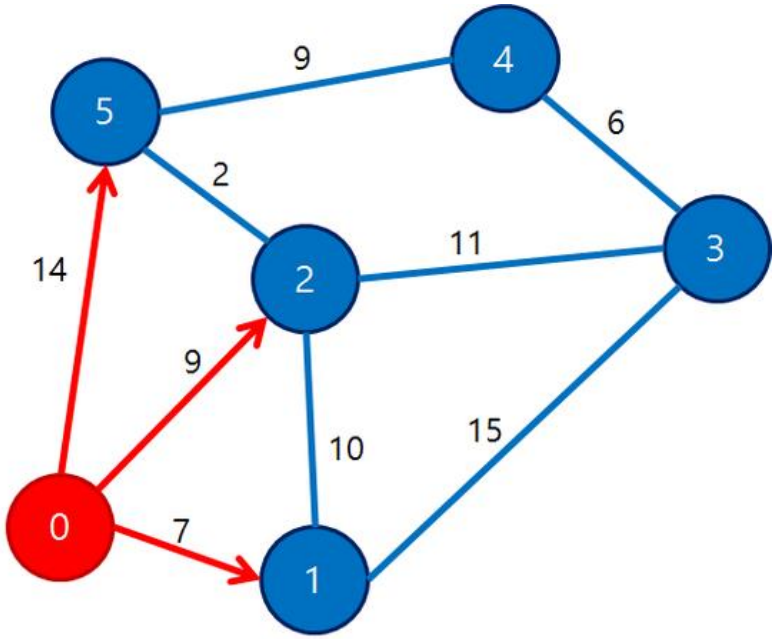
- 시작 지점 설정(u)
- 선택한 정점과 인접한 정점(v) 중 방문하지 않은 정점 중 가장 거리가 작은 정점을 선택 후 방문
- v에 대해 위의 과정을 반복

Dijkstra's Algorithm



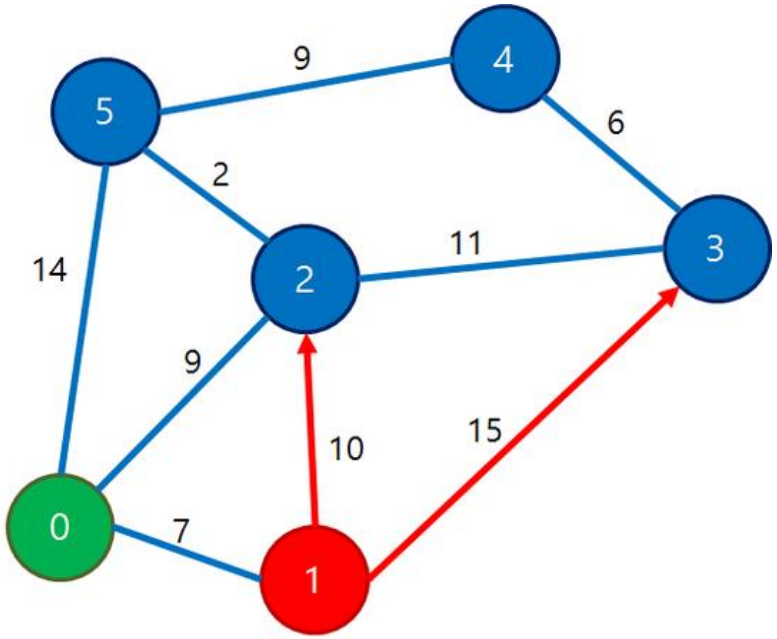
0	1	2	3	4	5
0	∞	∞	∞	∞	∞

Dijkstra's Algorithm



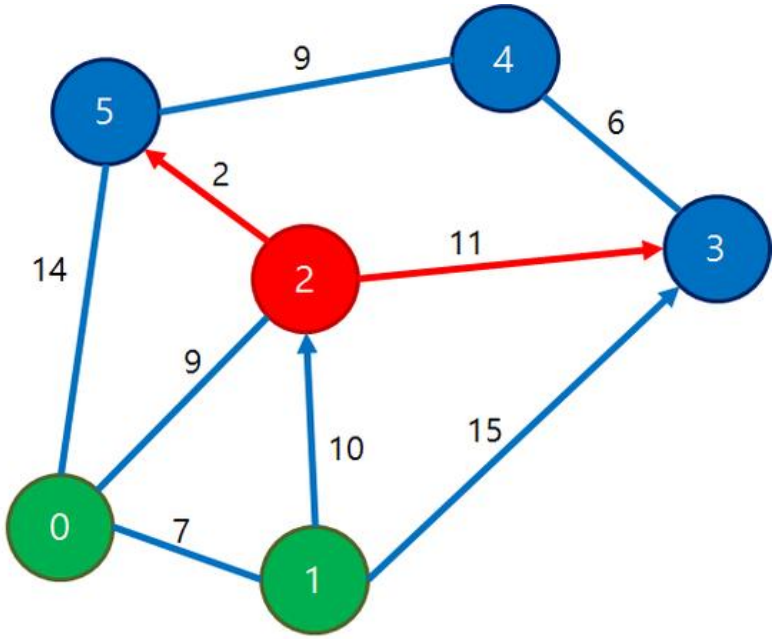
0	1	2	3	4	5
0	7	9	∞	∞	14

Dijkstra's Algorithm



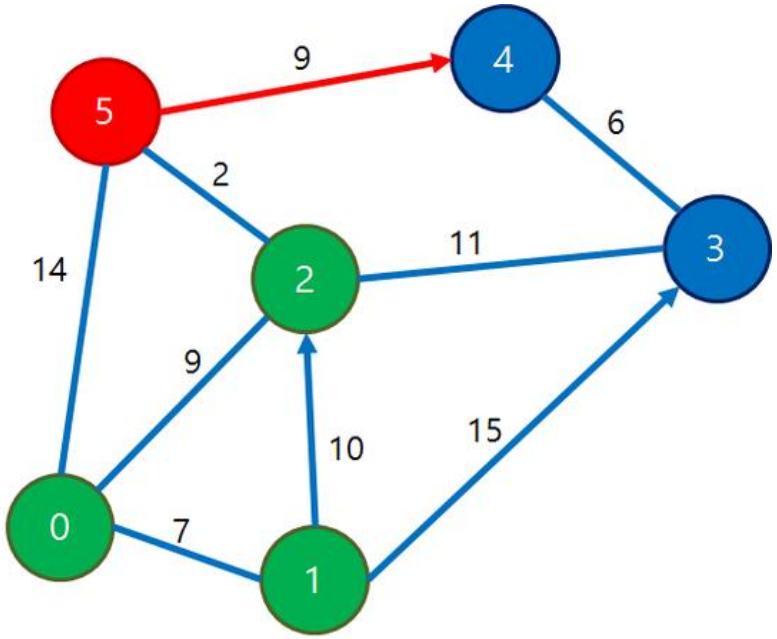
0	1	2	3	4	5
0	7	9	22	∞	14

Dijkstra's Algorithm



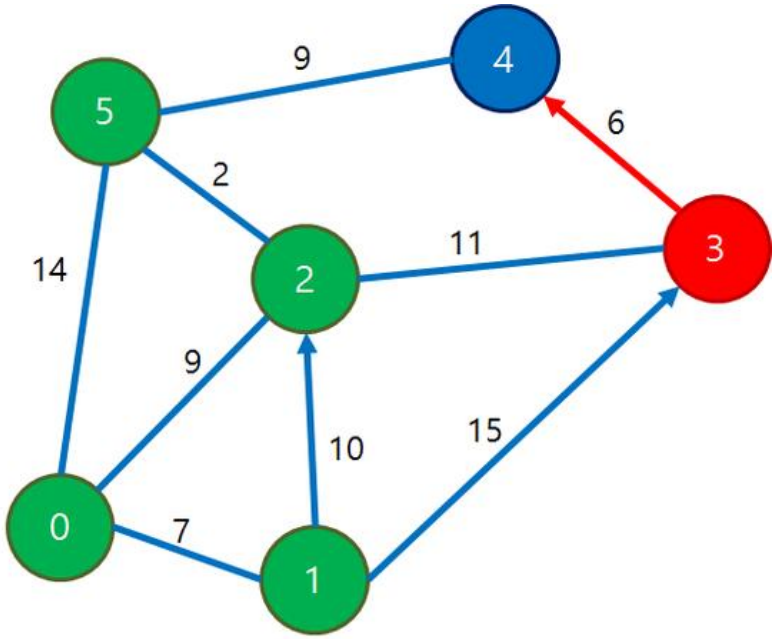
0	1	2	3	4	5
0	7	9	20	∞	11

Dijkstra's Algorithm



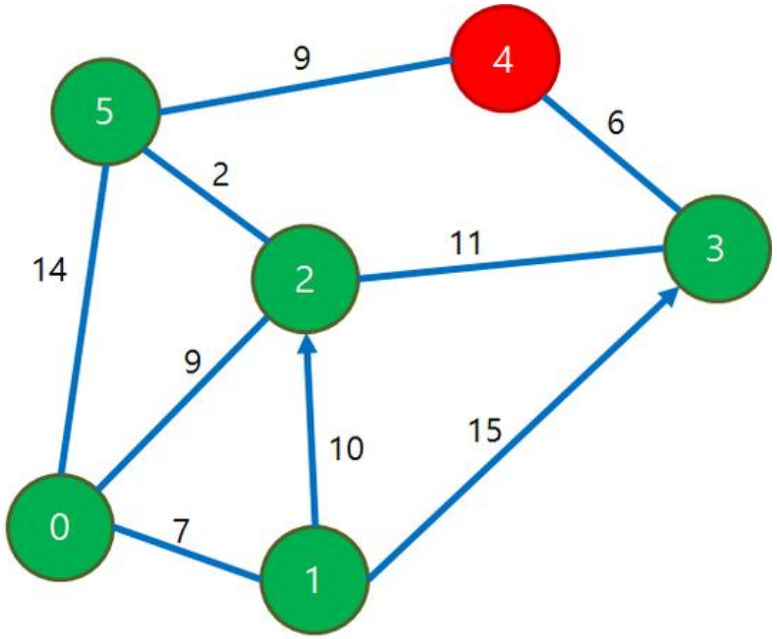
0	1	2	3	4	5
0	7	9	20	20	11

Dijkstra's Algorithm



0	1	2	3	4	5
0	7	9	20	20	11

Dijkstra's Algorithm



0	1	2	3	4	5
0	7	9	20	20	11

Dijkstra's Algorithm

```
define dist[mxv] = {∞}, vis[mxv] = {false}, start

dist[start] = 0
add(0, start) in pq
while pq is not empty:
    while pq is not empty:
        cur := pq.top, delete pq's top
        if vis[cur] == true then break
    if vis[cur] == true then break
    vis[cur] = true
    for each neighbor v of cur:
        if(dist[v] > dist[cur] + w(cur,v) then
            dist[v] = dist[cur] + w(cur,v)
            add(dist[v],v)
```

Prim's Algorithm

```
define vis[mxv], cnt = 0

input Edge

add (0,v0) in priority queue
while pq is not empty:
    if cnt == |V| then break
    while vis[pq.top()->v]
        pop pq
    u := pq's top, vis[u->v] = true, cnt++
    total_w += u->w
    for each neighbor v` of u:
        if !vis[v`] then insert(w(u,v`),v`) in pq
```

Problem Set of Dijkstra

- 1753 최단경로
- 1916 최소비용 구하기
- 1238 파티
- 1504 특정한 최단경로
- 4485 녹색 옷 입은 애가 젤다지?
- 10217 KCM Travel(*)
- 5719 거의 최단경로(*)
- 1854 K번째 최단경로 찾기(*)