COMP319 Algorithms 1
Lecture 15
Minimum Spanning Trees

Instructor: Gil-Jin Jang

School of Electronics Engineering, Kyungpook National University

Textbook Chapters 25 and 26

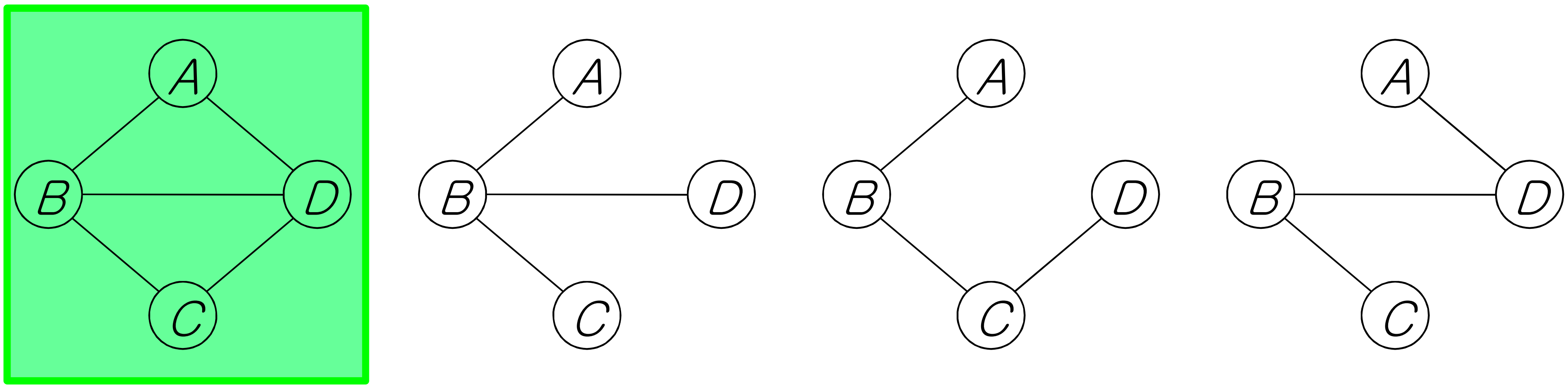
Slide credits: 홍석원, 명지대학교; 김한준, 서울시립대학교; J. Lillis, UIC; David Luebke, CS332, Virginia University

Table of Contents

- Minimum Spanning Trees (MST)
  - Definition of minimum spanning tree
  - Kruskal’s algorithm
  - Prim’s algorithm

신장 트리(Spanning Tree)

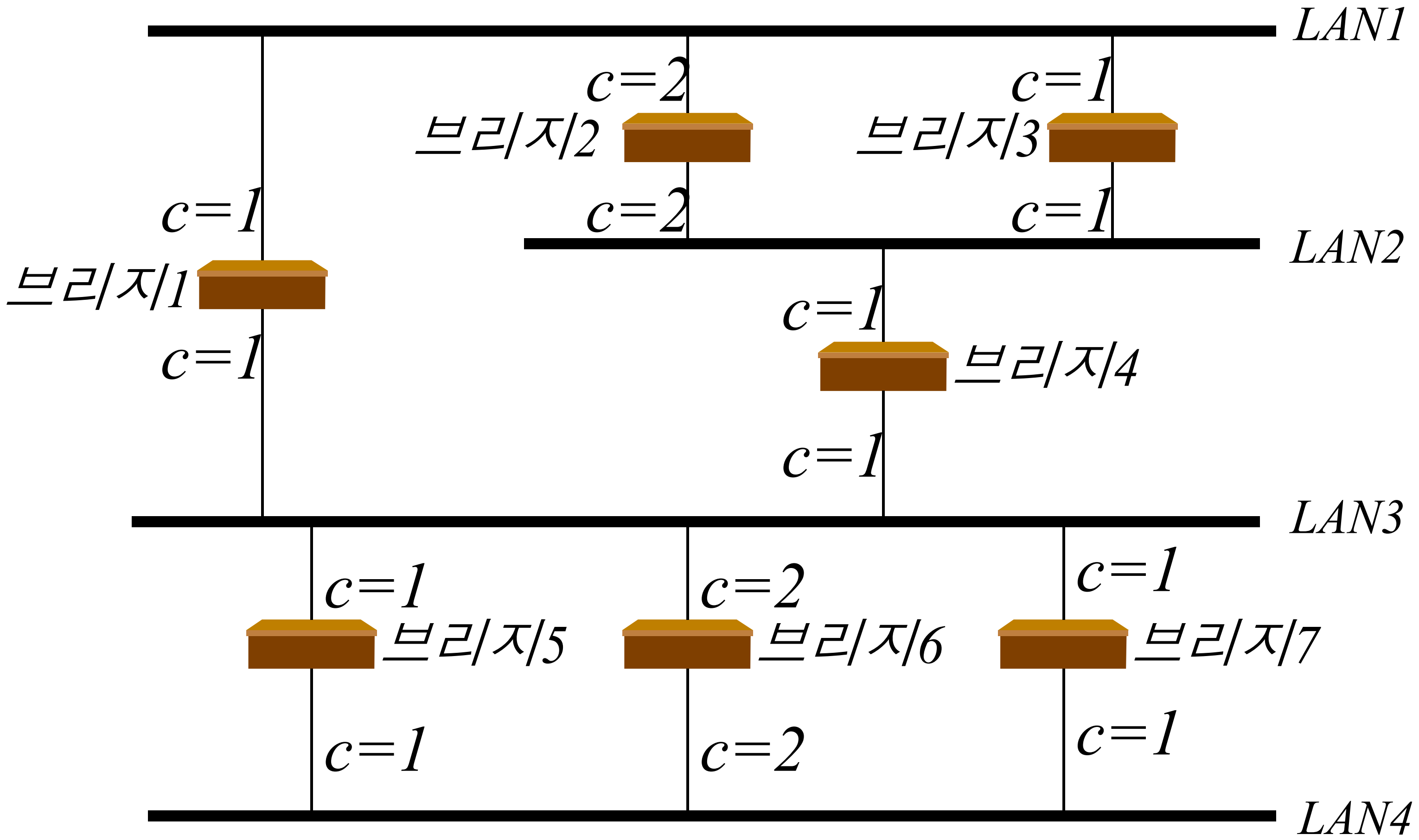
- 그래프 $G=\{V,E\}$에서 V의 모든 정점을 포함하면서 순환(cycle)이 존재하지 않는 부분 그래프(subgraph)를 신장 트리라고 한다.
  - n개의 정점으로 이루어진 **무방향 연결 그래프**에서, n개의 정점 전부와 n-1개의 간선으로 만들어진 트리
    - 신장 트리는 최소 갯수의 간선으로 그래프의 모든 정점들이 연결되도록 함



Slide credit: 홍석원, 명지대학교; 김한준, 서울시립대학교; J. Lillis, UIC

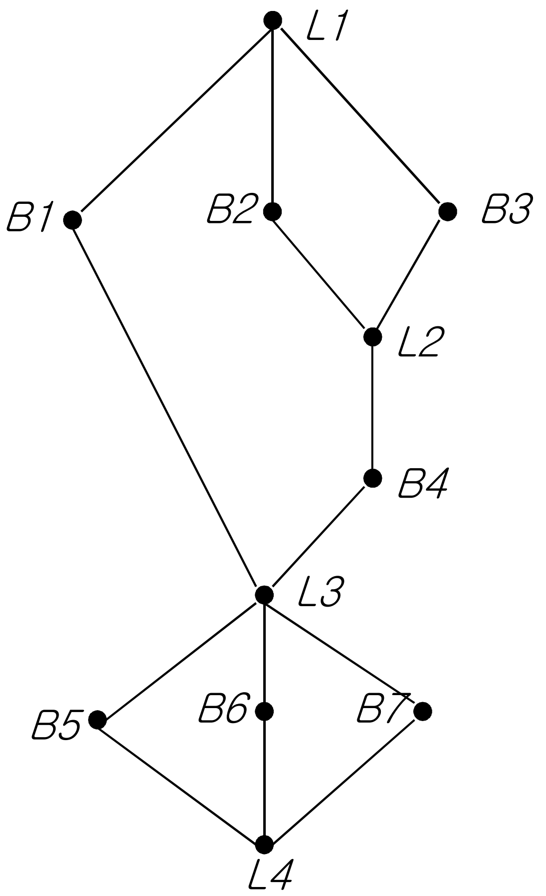
신장 트리의 응용 예

중복되는 연결선들이 많은 네트워크

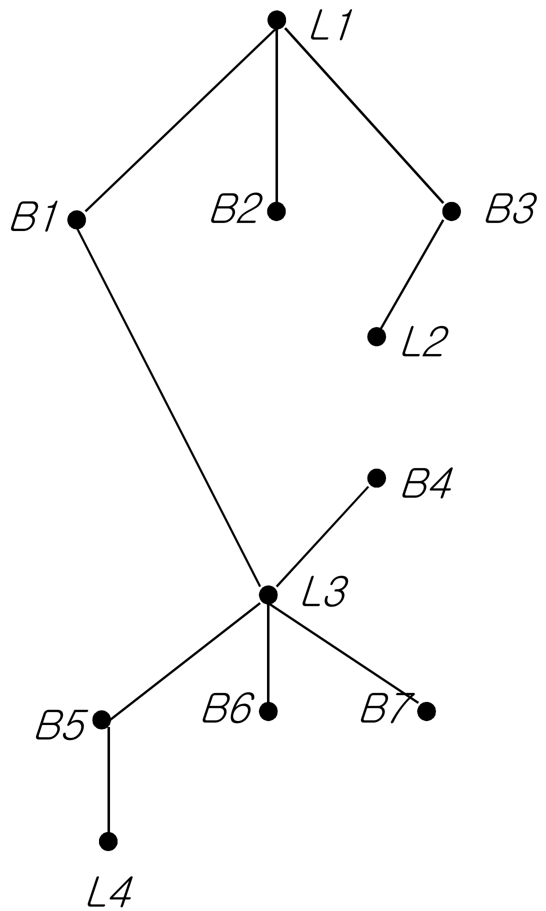


Slide credit: 홍석원, 명지대학교; 김한준, 서울시립대학교; J. Lillis, UIC

그래프로 표현한 네트워크



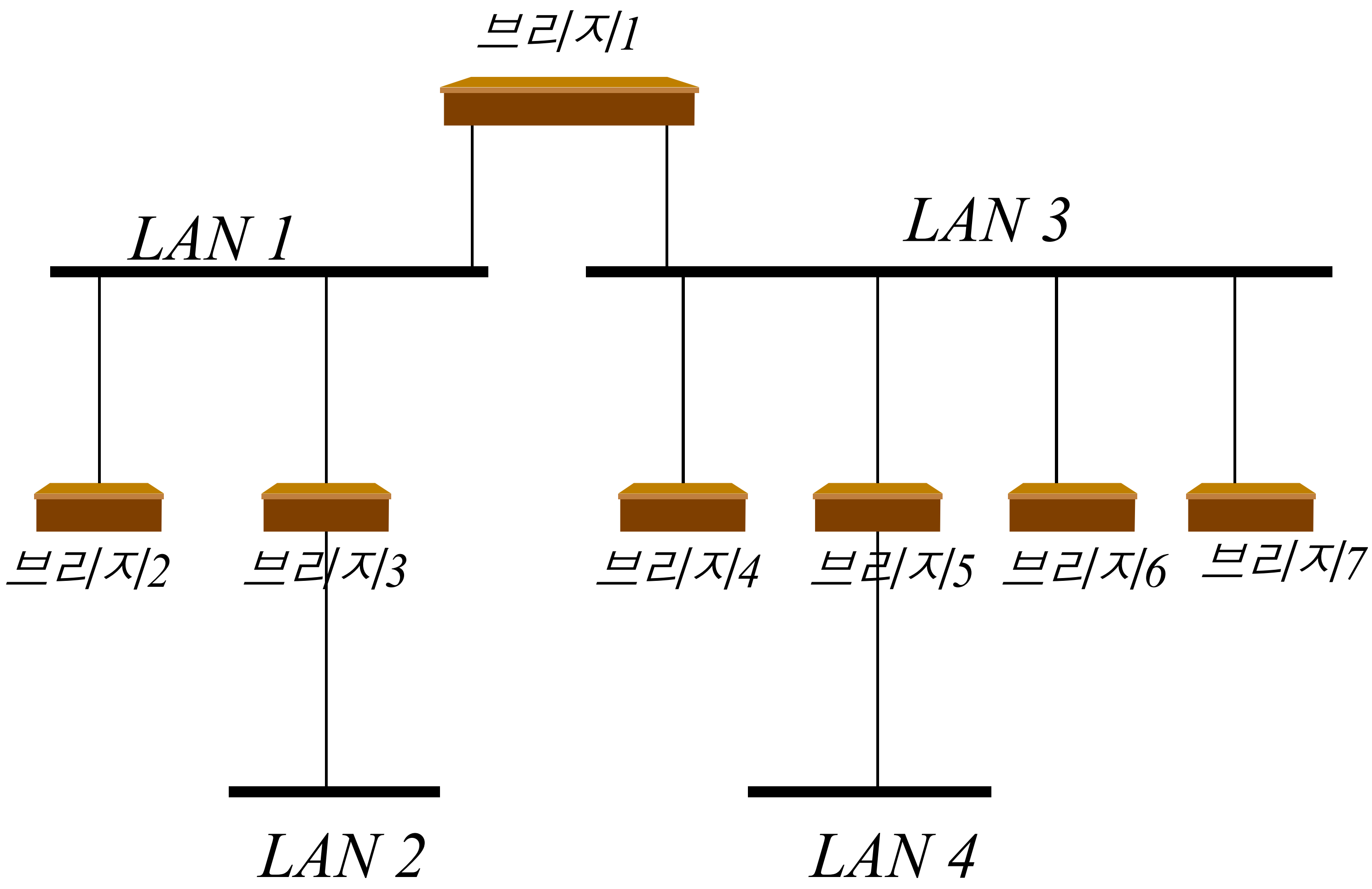
LAN의 그래프 G



G의 신장 트리

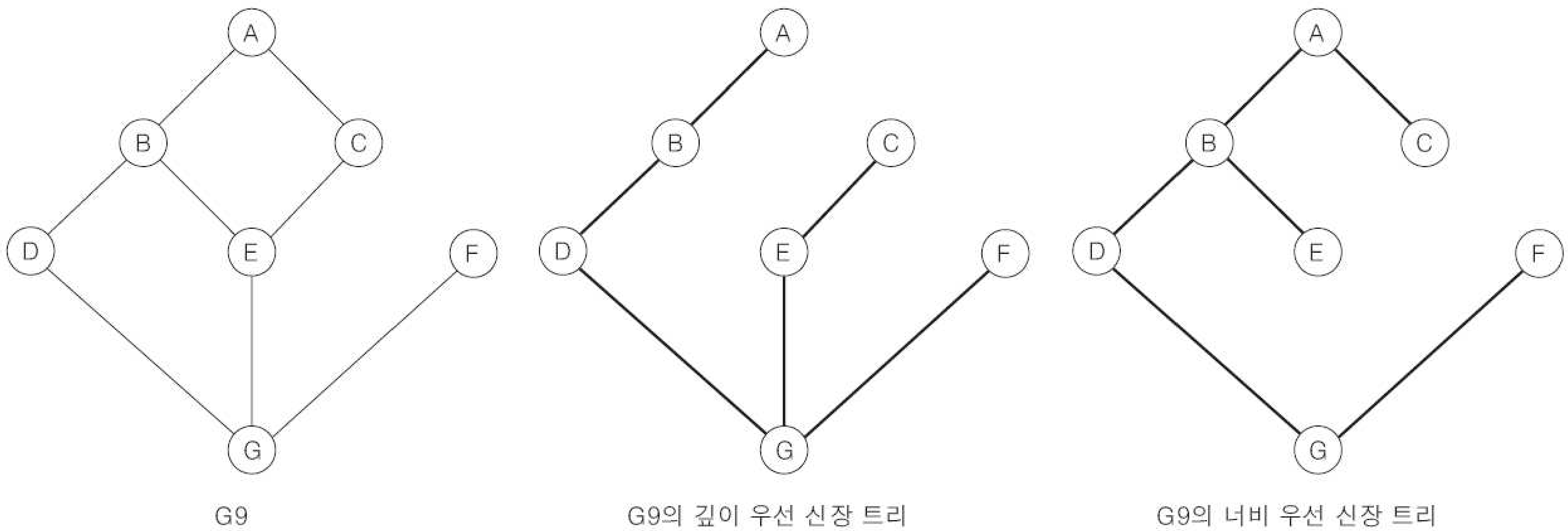
Slide credit: 홍석원, 명지대학교; 김한준, 서울시립대학교; J. Lillis, UIC

신장 트리에 의한 LAN의 구성



신장 트리 생성방법

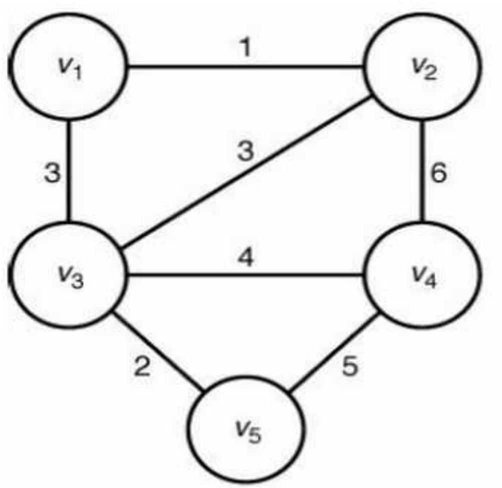
- 깊이 우선 신장 트리(depth first spanning tree)
- 너비 우선 신장 트리(breadth first spanning tree)
- 그래프 G9의 깊이 우선 신장 트리와 너비 우선 신장 트리 비교



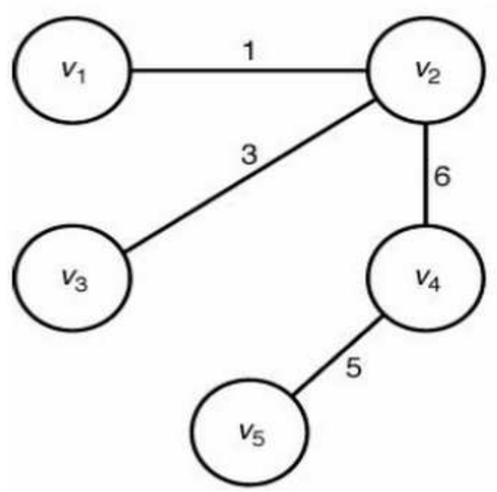
Slide credit: 홍석원, 명지대학교; 김한준, 서울시립대학교; J. Lillis, UIC

Minimum Cost Spanning Tree

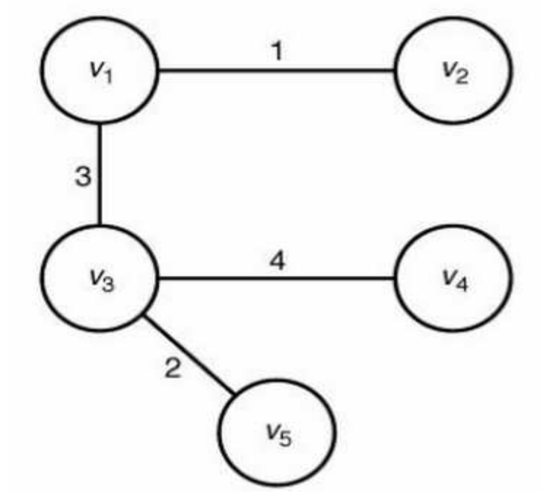
- **무방향 가중치 연결 그래프(undirected, weighted, connected graph)**의 신장 트리들 중, 간선들의 가중치 합이 최소인 것
  - 가중치 그래프의 간선에 주어진 가중치는 두 정점 사이의 비용이나 거리, 시간을 의미하는 값



연결된 가중치
무방향 그래프



신장트리



최소비용신장트리

Slide credit: 홍석원, 명지대학교; 김한준, 서울시립대학교; J. Lillis, UIC

Minimum Cost Spanning Tree

- MST properties
  - |V|-1 개의 edge 들로 이루어진다
  - 모든 신장트리가 최소비용신장트리는 아니다.
  - 최소비용신장트리는 1개 이상 존재할 수 있다.
- MST Applications
  - 도로건설: 도시들을 모두 연결하면서 도로의 길이가 최소가 되도록 하는 문제
  - 통신(telecommunications): 전화선의 길이가 최소가 되도록 전화 케이블 망을 구성하는 문제
  - 배관(plumbing): 파이프의 총 길이가 최소가 되도록 연결하는 문제

Slide credit: 홍석원, 명지대학교; 김한준, 서울시립대학교; J. Lillis, UIC

Brute-Force Algorithm for MST

- Exhaustive search
  - 모든 가능한 신장트리를 다 생성하고 비용을 계산
    - 전체 edge에서 |V|-1 개를 선택하는 문제

$$ {}_eP_v = \frac{e!}{(e-v)!}, \qquad e = |E|,\ v = |V| $$

  - 그 중에서 최소비용이 드는 것을 신장트리를 고른다.
  - 완전 연결의 경우가 최악의 경우

$$ {}_{v^2}P_v = \frac{v^2!}{(v^2-v)!}, \qquad v^2 \simeq |E|,\ v = |V| $$

Slide credit: 홍석원, 명지대학교; 김한준, 서울시립대학교; J. Lillis, UIC

Edge elimination

KRUSKAL’S ALGORITHM FOR MST

Slide credit: 홍석원, 명지대학교; 김한준, 서울시립대학교; J. Lillis, UIC

Overview of Kruskal’s Algorithm

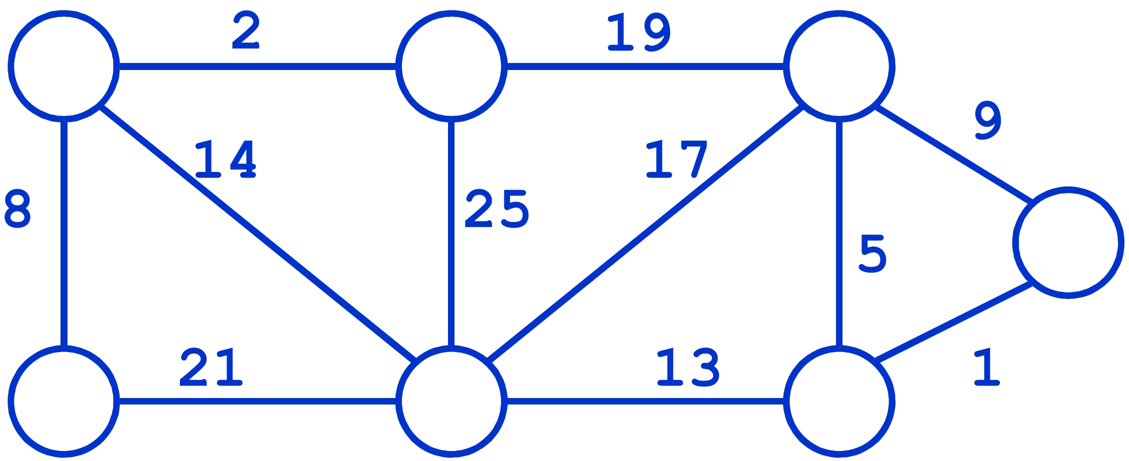
1. 그래프의 각 꼭짓점이 각각 하나의 tree가 되도록 하는 숲(forest) F를 만든다
2. 모든 변을 원소로 갖는 집합 S 를 만든다
3. S 가 비어있지 않는 동안
   1. 가장 작은 가중치의 변을 S 에서 하나 빼낸다
   2. 그 변이 어떤 두 개의 나무를 연결한다면 두 나무를 연결하여 하나의 나무로 만든다
   3. 그렇지 않다면 그 변은 버린다
4. 알고리즘이 종료됐을 때 숲 F는 하나의 최소 비용 신장 부분 그래프만을 가지게 된다.

Source: Wikipedia.org

Slide credit: 홍석원, 명지대학교; 김한준, 서울시립대학교; J. Lillis, UIC

Kruskal's Algorithm

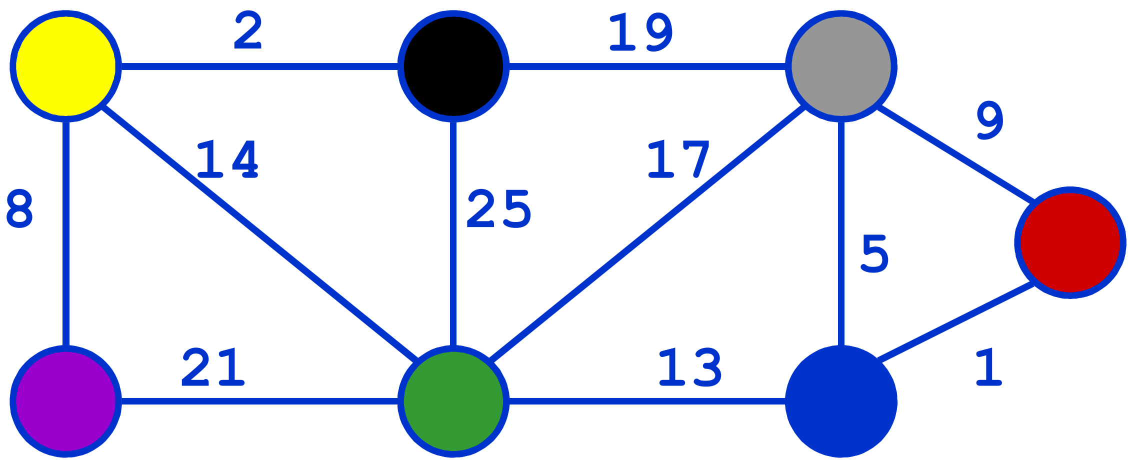
Run the algorithm:




```
Kruskal()
{
    T = ∅;
    for each v ∈ V
        MakeSet(v);
    sort E by increasing edge weight w
    for each (u,v) ∈ E (in sorted order)
        if FindSet(u) ≠ FindSet(v)
            T = T U {{u,v}};
            Union(FindSet(u), FindSet(v));
}
```

Kruskal's Algorithm

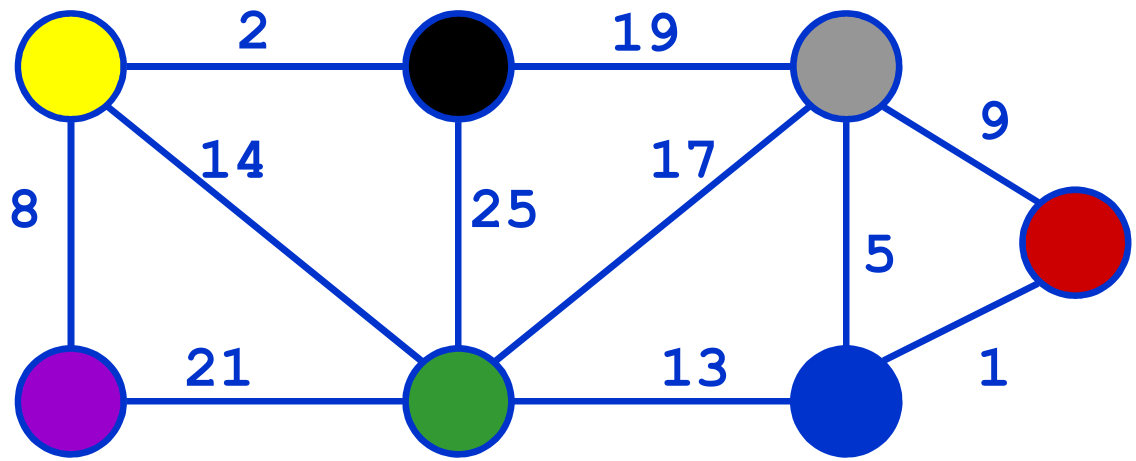
Run the algorithm:




```
Kruskal()
{
   T = ∅;
   for each v ∈ V
      MakeSet(v);
   sort E by increasing edge weight w
   for each (u,v) ∈ E (in sorted order)
      if FindSet(u) ≠ FindSet(v)
         T = T U {{u,v}};
         Union(FindSet(u), FindSet(v));
}
```

Kruskal's Algorithm

Run the algorithm:

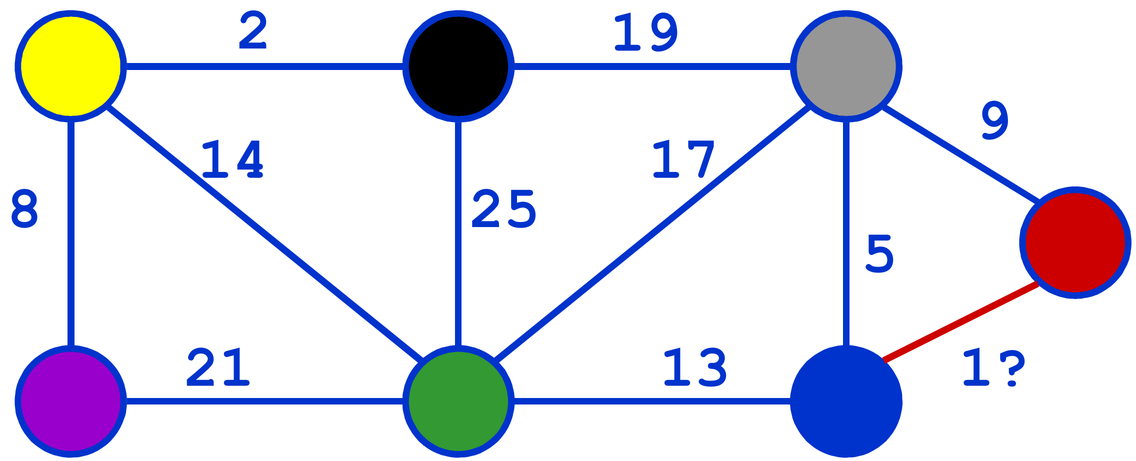



```
Kruskal()
{
    T = ∅;
    for each v ∈ V
        MakeSet(v);
    sort E by increasing edge weight w
    for each (u,v) ∈ E (in sorted order)
        if FindSet(u) ≠ FindSet(v)
            T = T U {{u,v}};
            Union(FindSet(u), FindSet(v));
}
```

Slide credit: 홍석원, 명지대학교; 김한준, 서울시립대학교; J. Lillis, UIC

Kruskal's Algorithm

Run the algorithm:

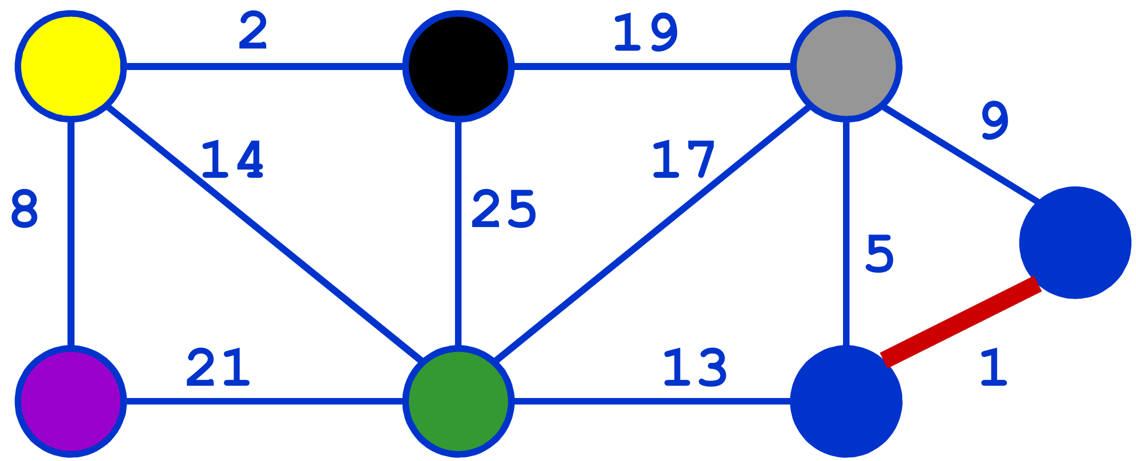



```
Kruskal()
{
    T = ∅;
    for each v ∈ V
        MakeSet(v);
    sort E by increasing edge weight w
    for each (u,v) ∈ E (in sorted order)
        if FindSet(u) ≠ FindSet(v)
            T = T U {{u,v}};
            Union(FindSet(u), FindSet(v));
}
```

Slide credit: 홍석원, 명지대학교; 김한준, 서울시립대학교; J. Lillis, UIC

Kruskal's Algorithm

Run the algorithm:

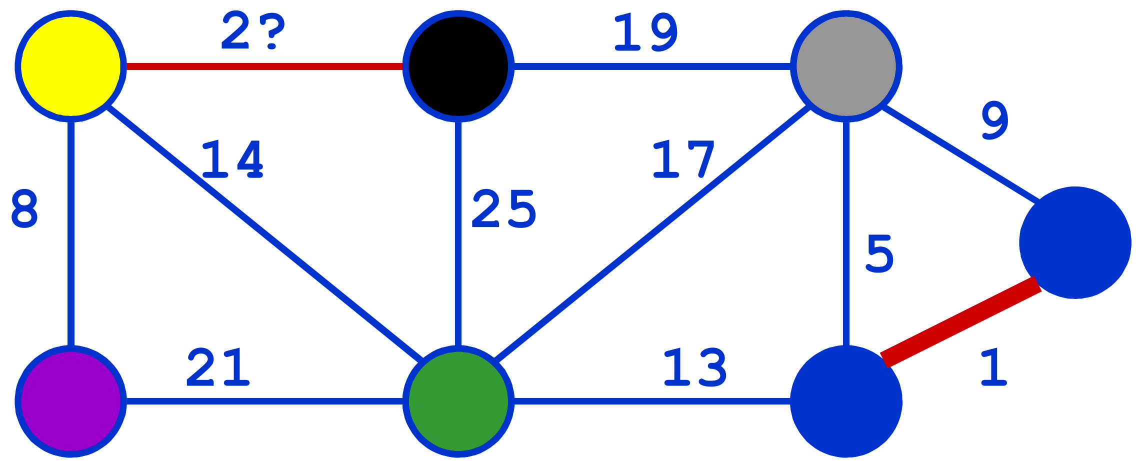



```
Kruskal()
{
    T = ∅;
    for each v ∈ V
        MakeSet(v);
    sort E by increasing edge weight w
    for each (u,v) ∈ E (in sorted order)
        if FindSet(u) ≠ FindSet(v)
            T = T U {{u,v}};
            Union(FindSet(u), FindSet(v));
}
```

Slide credit: 홍석원, 명지대학교; 김한준, 서울시립대학교; J. Lillis, UIC

Kruskal's Algorithm

Run the algorithm:



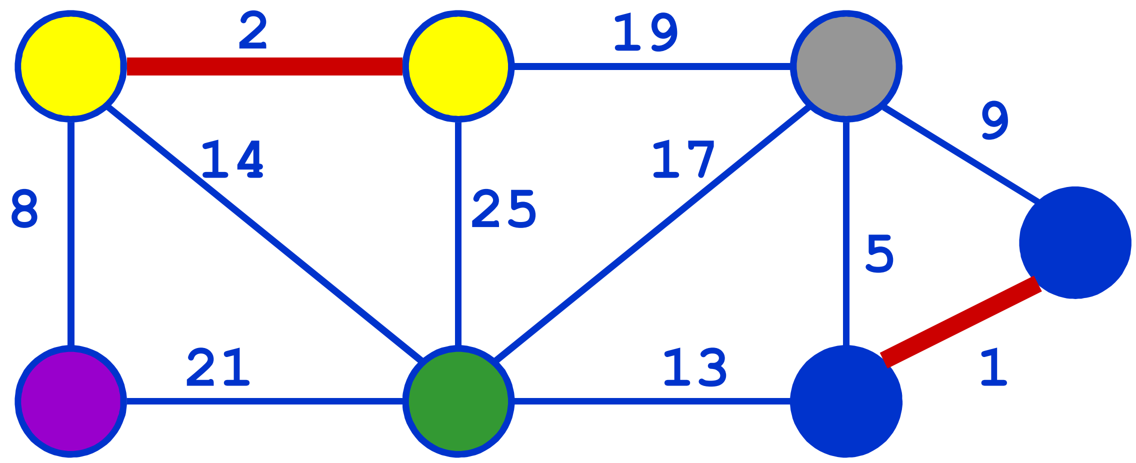

```
Kruskal()
{
    T = ∅;
    for each v ∈ V
        MakeSet(v);
    sort E by increasing edge weight w
    for each (u,v) ∈ E (in sorted order)
        if FindSet(u) ≠ FindSet(v)
            T = T U {{u,v}};
            Union(FindSet(u), FindSet(v));
}
```

Slide credit: 홍석원, 명지대학교; 김한준, 서울시립대학교; J. Lillis, UIC

Kruskal's Algorithm

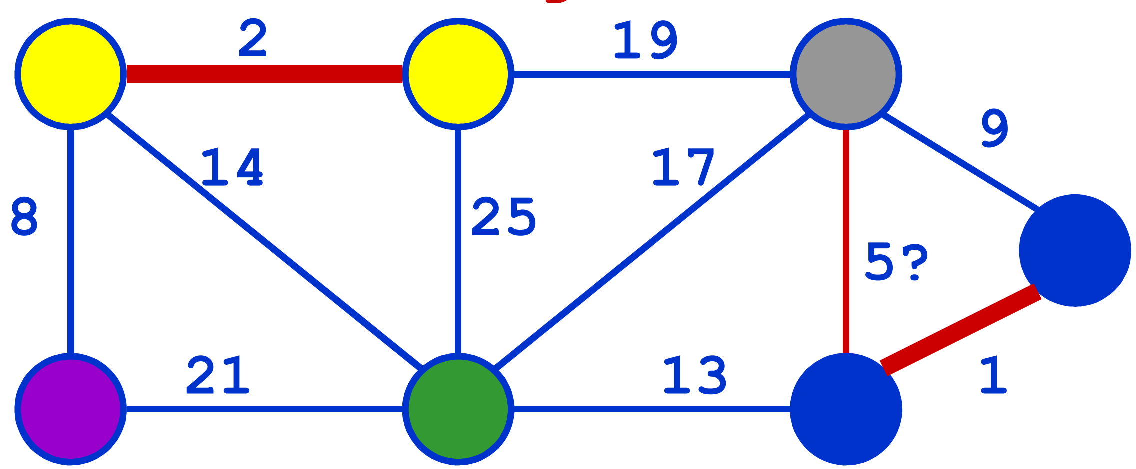
```
Kruskal()
{
    T = ∅;
    for each v ∈ V
        MakeSet(v);
    sort E by increasing edge weight w
    for each (u,v) ∈ E (in sorted order)
        if FindSet(u) ≠ FindSet(v)
            T = T U {{u,v}};
            Union(FindSet(u), FindSet(v));
}
```

Run the algorithm:



Kruskal's Algorithm

Run the algorithm:

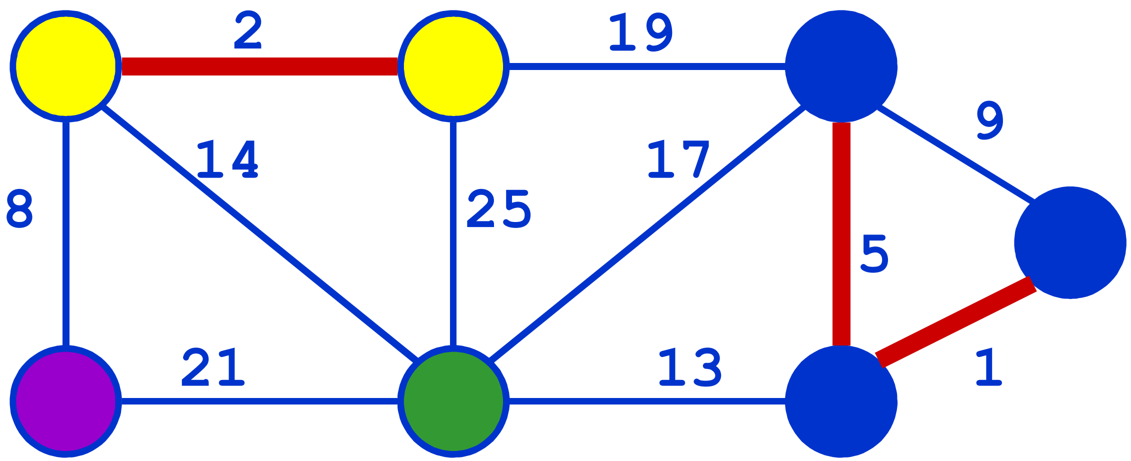



```
Kruskal()
{
    T = ∅;
    for each v ∈ V
        MakeSet(v);
    sort E by increasing edge weight w
    for each (u,v) ∈ E (in sorted order)
        if FindSet(u) ≠ FindSet(v)
            T = T U {{u,v}};
            Union(FindSet(u), FindSet(v));
}
```

Slide credit: 홍석원, 명지대학교; 김한준, 서울시립대학교; J. Lillis, UIC

Kruskal's Algorithm

Run the algorithm:

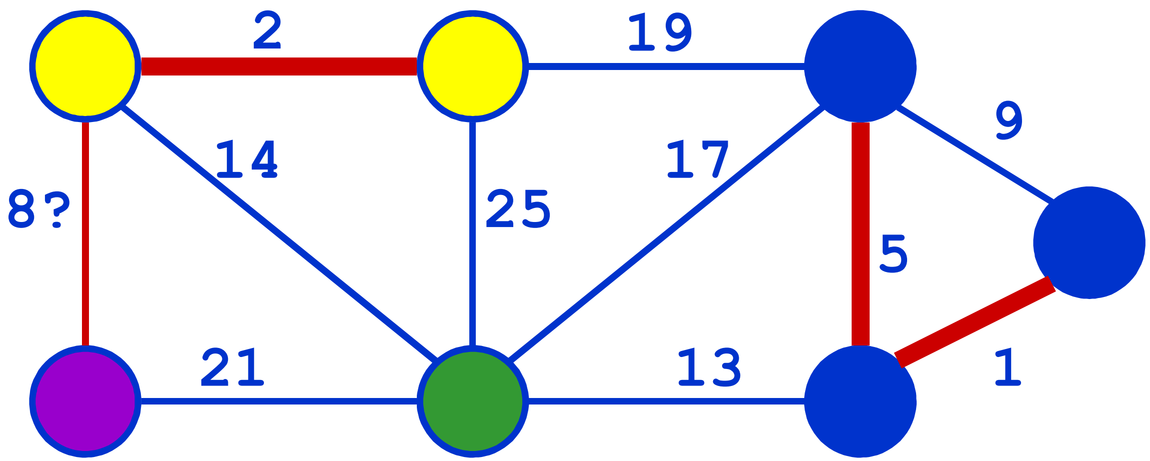



```
Kruskal()
{
    T = ∅;
    for each v ∈ V
        MakeSet(v);
    sort E by increasing edge weight w
    for each (u,v) ∈ E (in sorted order)
        if FindSet(u) ≠ FindSet(v)
            T = T U {{u,v}};
            Union(FindSet(u), FindSet(v));
}
```

Slide credit: 홍석원, 명지대학교; 김한준, 서울시립대학교; J. Lillis, UIC

Kruskal's Algorithm

Run the algorithm:

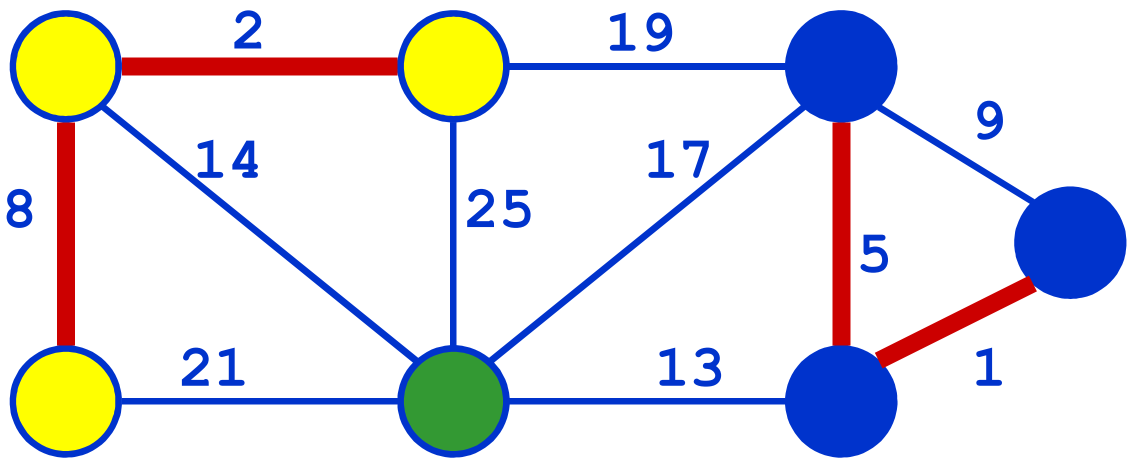



```
Kruskal()
{
    T = ∅;
    for each v ∈ V
        MakeSet(v);
    sort E by increasing edge weight w
    for each (u,v) ∈ E (in sorted order)
        if FindSet(u) ≠ FindSet(v)
            T = T U {{u,v}};
            Union(FindSet(u), FindSet(v));
}
```

Slide credit: 홍석원, 명지대학교; 김한준, 서울시립대학교; J. Lillis, UIC

Kruskal's Algorithm

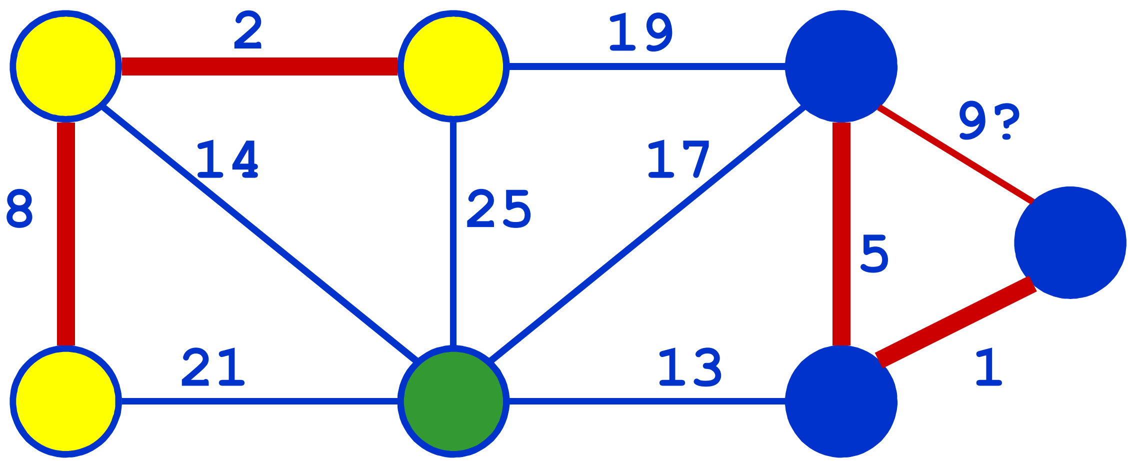
Run the algorithm:




```
Kruskal()
{
    T = ∅;
    for each v ∈ V
        MakeSet(v);
    sort E by increasing edge weight w
    for each (u,v) ∈ E (in sorted order)
        if FindSet(u) ≠ FindSet(v)
            T = T U {{u,v}};
            Union(FindSet(u), FindSet(v));
}
```

Kruskal's Algorithm

Run the algorithm:

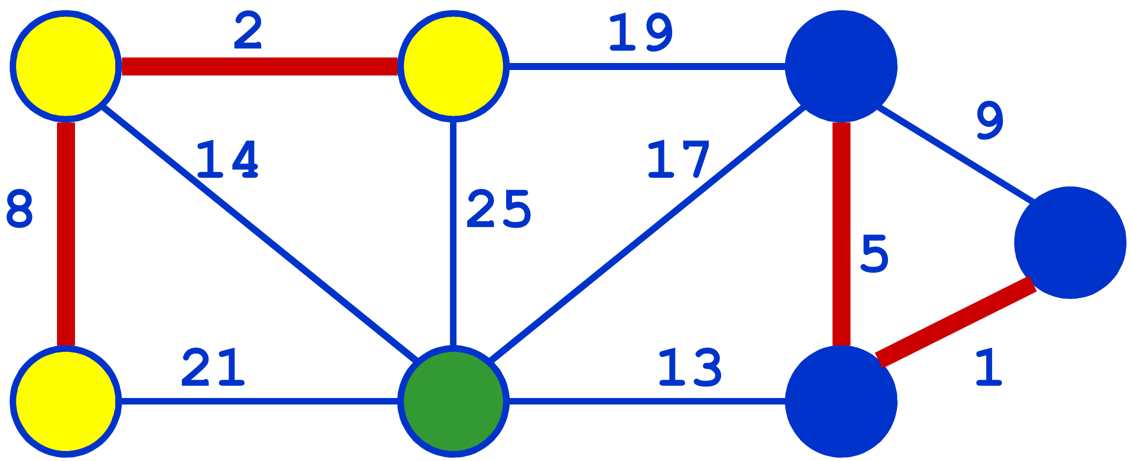



```
Kruskal()
{
    T = ∅;
    for each v ∈ V
        MakeSet(v);
    sort E by increasing edge weight w
    for each (u,v) ∈ E (in sorted order)
        if FindSet(u) ≠ FindSet(v)
            T = T U {{u,v}};
            Union(FindSet(u), FindSet(v));
}
```

Slide credit: 홍석원, 명지대학교; 김한준, 서울시립대학교; J. Lillis, UIC

Kruskal's Algorithm

Run the algorithm:

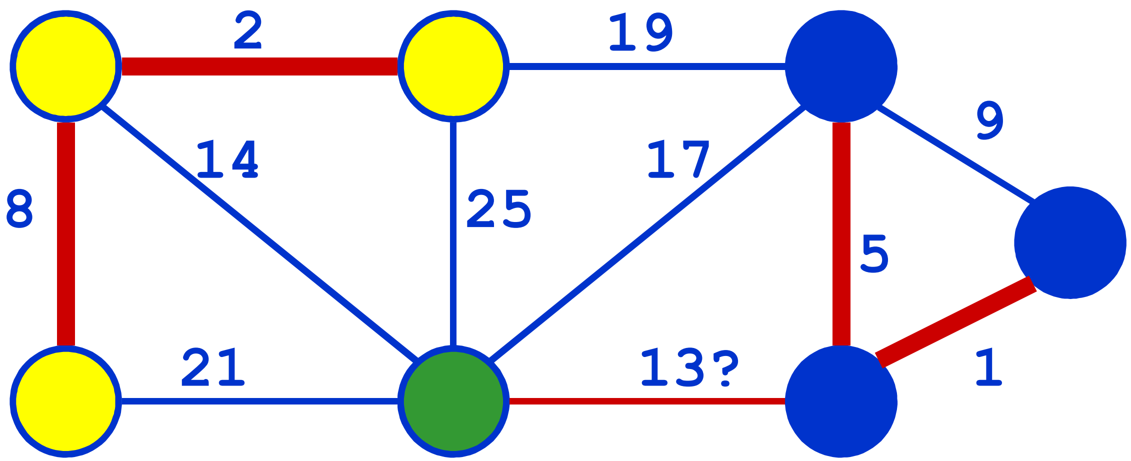



```
Kruskal()
{
    T = ∅;
    for each v ∈ V
        MakeSet(v);
    sort E by increasing edge weight w
    for each (u,v) ∈ E (in sorted order)
        if FindSet(u) ≠ FindSet(v)
            T = T U {{u,v}};
            Union(FindSet(u), FindSet(v));
}
```

Slide credit: 홍석원, 명지대학교; 김한준, 서울시립대학교; J. Lillis, UIC

Kruskal's Algorithm

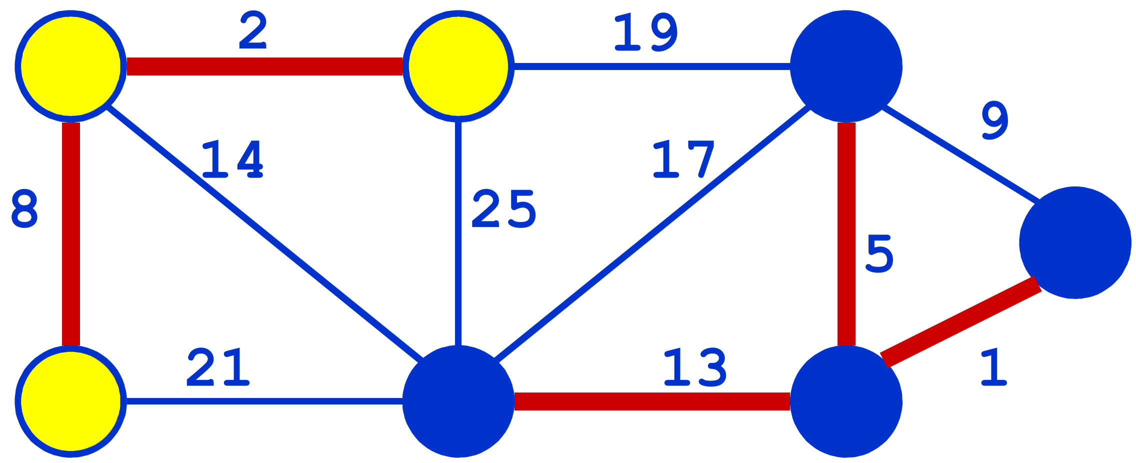
Run the algorithm:




```
Kruskal()
{
    T = ∅;
    for each v ∈ V
        MakeSet(v);
    sort E by increasing edge weight w
    for each (u,v) ∈ E (in sorted order)
        if FindSet(u) ≠ FindSet(v)
            T = T U {{u,v}};
            Union(FindSet(u), FindSet(v));
}
```

Kruskal's Algorithm

Run the algorithm:

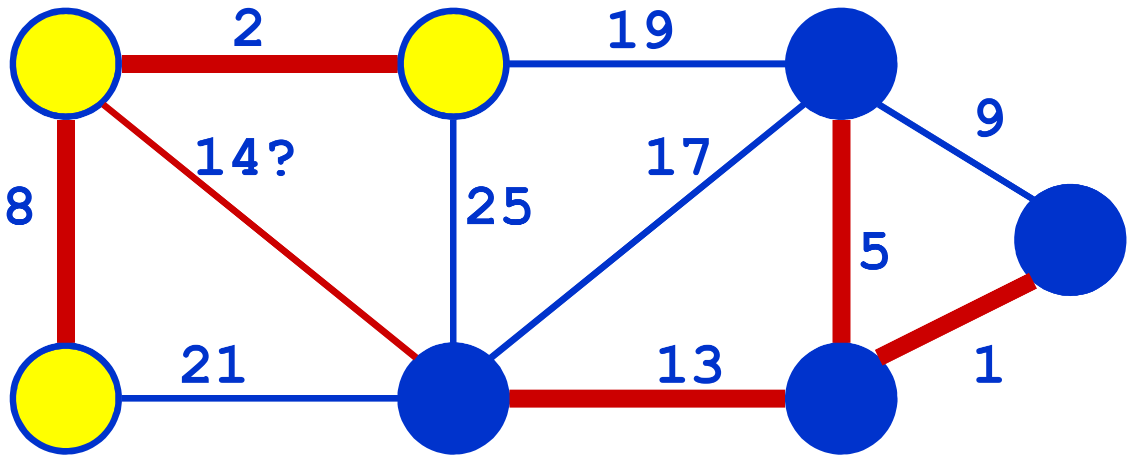



```
Kruskal()
{
    T = ∅;
    for each v ∈ V
        MakeSet(v);
    sort E by increasing edge weight w
    for each (u,v) ∈ E (in sorted order)
        if FindSet(u) ≠ FindSet(v)
            T = T U {{u,v}};
            Union(FindSet(u), FindSet(v));
}
```

Slide credit: 홍석원, 명지대학교; 김한준, 서울시립대학교; J. Lillis, UIC

Kruskal's Algorithm

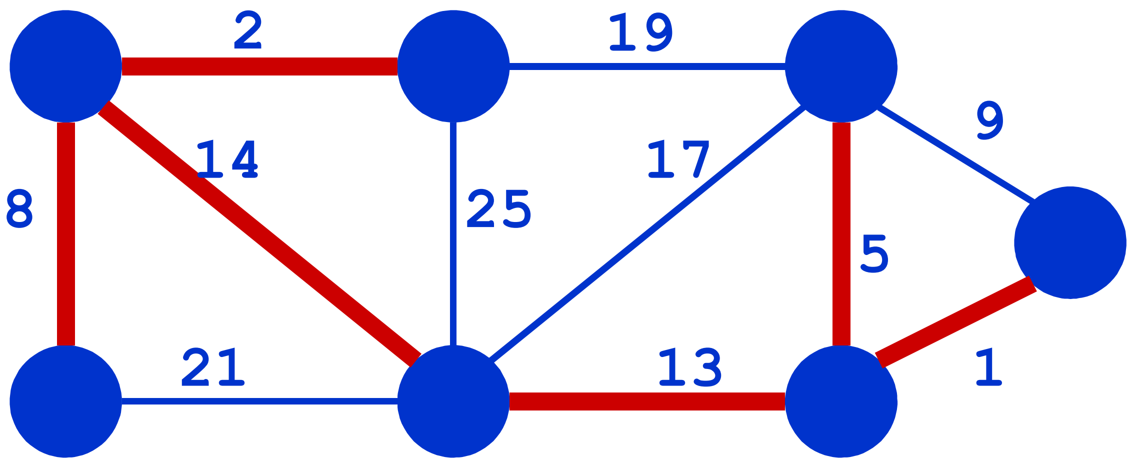
Run the algorithm:




```
Kruskal()
{
    T = ∅;
    for each v ∈ V
        MakeSet(v);
    sort E by increasing edge weight w
    for each (u,v) ∈ E (in sorted order)
        if FindSet(u) ≠ FindSet(v)
            T = T U {{u,v}};
            Union(FindSet(u), FindSet(v));
}
```

Kruskal's Algorithm

Run the algorithm:

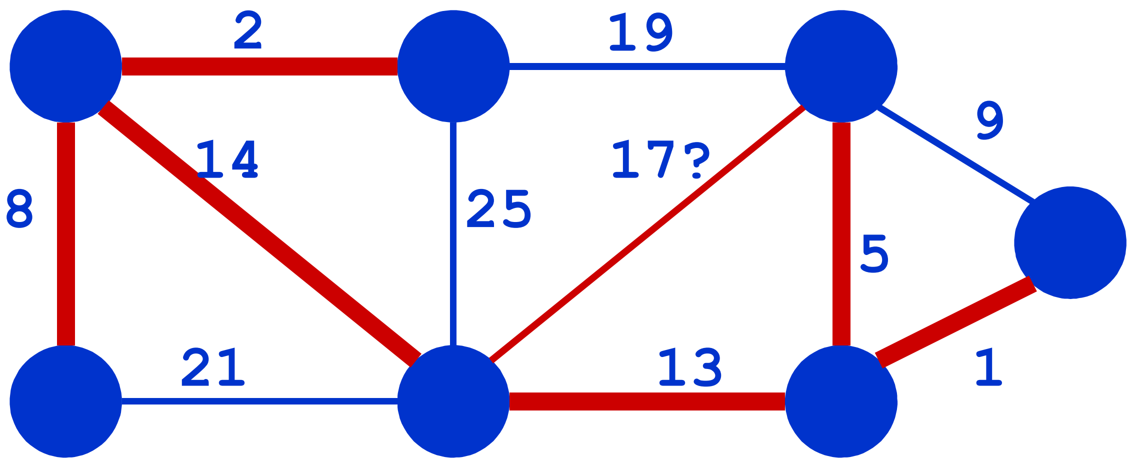



```
Kruskal()
{
    T = ∅;
    for each v ∈ V
        MakeSet(v);
    sort E by increasing edge weight w
    for each (u,v) ∈ E (in sorted order)
        if FindSet(u) ≠ FindSet(v)
            T = T U {{u,v}};
            Union(FindSet(u), FindSet(v));
}
```

Slide credit: 홍석원, 명지대학교; 김한준, 서울시립대학교; J. Lillis, UIC

Kruskal's Algorithm

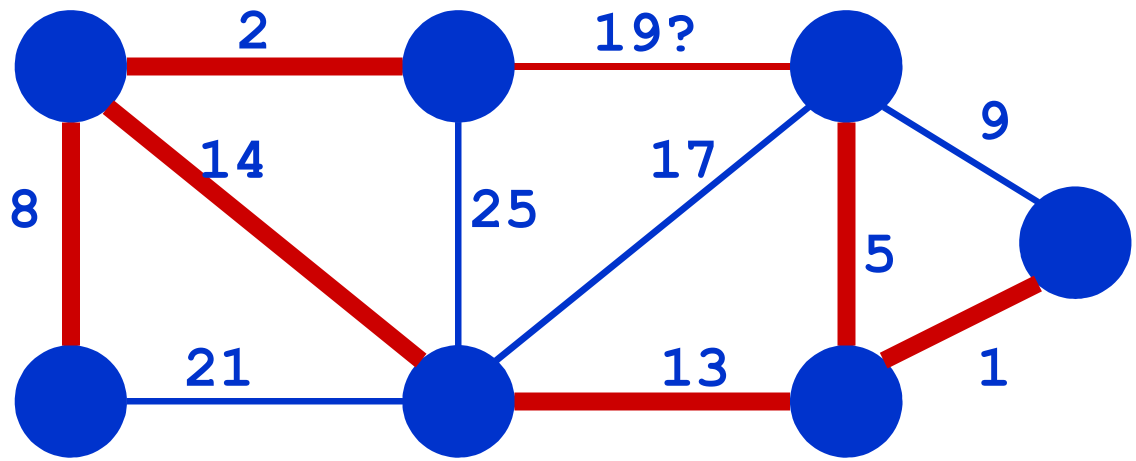
Run the algorithm:




```
Kruskal()
{
    T = ∅;
    for each v ∈ V
        MakeSet(v);
    sort E by increasing edge weight w
    for each (u,v) ∈ E (in sorted order)
        if FindSet(u) ≠ FindSet(v)
            T = T U {{u,v}};
            Union(FindSet(u), FindSet(v));
}
```

Kruskal's Algorithm

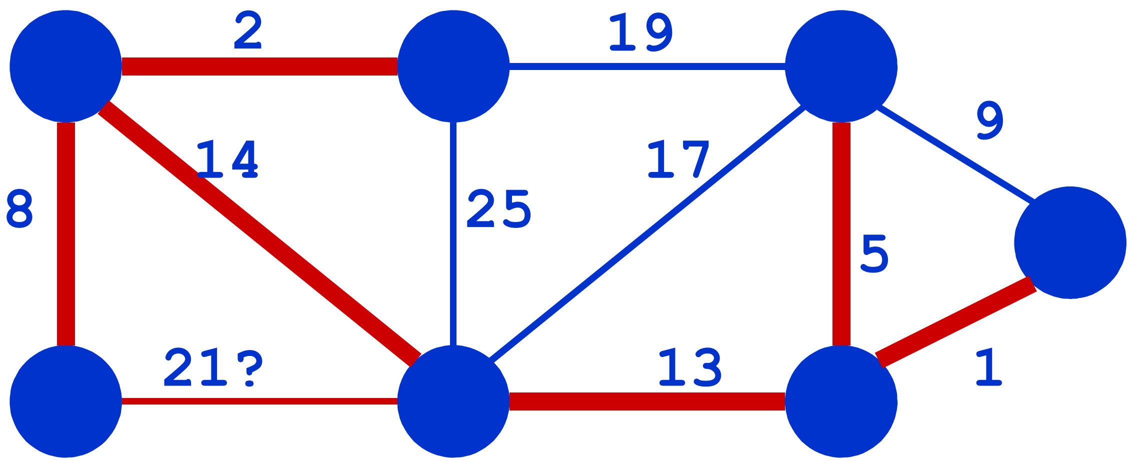
Run the algorithm:




```
Kruskal()
{
    T = ∅;
    for each v ∈ V
        MakeSet(v);
    sort E by increasing edge weight w
    for each (u,v) ∈ E (in sorted order)
        if FindSet(u) ≠ FindSet(v)
            T = T U {{u,v}};
            Union(FindSet(u), FindSet(v));
}
```

Kruskal's Algorithm

Run the algorithm:

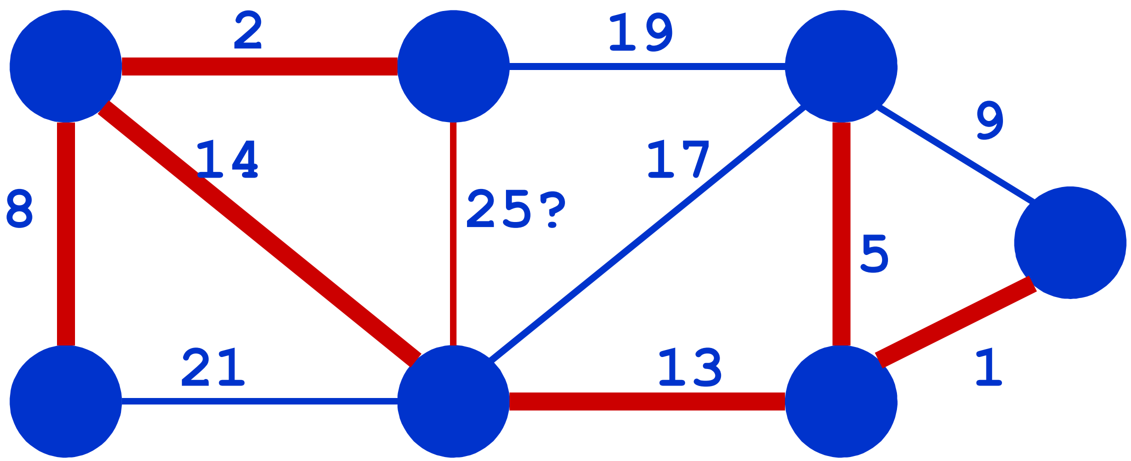



```
Kruskal()
{
    T = ∅;
    for each v ∈ V
        MakeSet(v);
    sort E by increasing edge weight w
    for each (u,v) ∈ E (in sorted order)
        if FindSet(u) ≠ FindSet(v)
            T = T U {{u,v}};
            Union(FindSet(u), FindSet(v));
}
```

Slide credit: 홍석원, 명지대학교; 김한준, 서울시립대학교; J. Lillis, UIC

Kruskal's Algorithm

Run the algorithm:

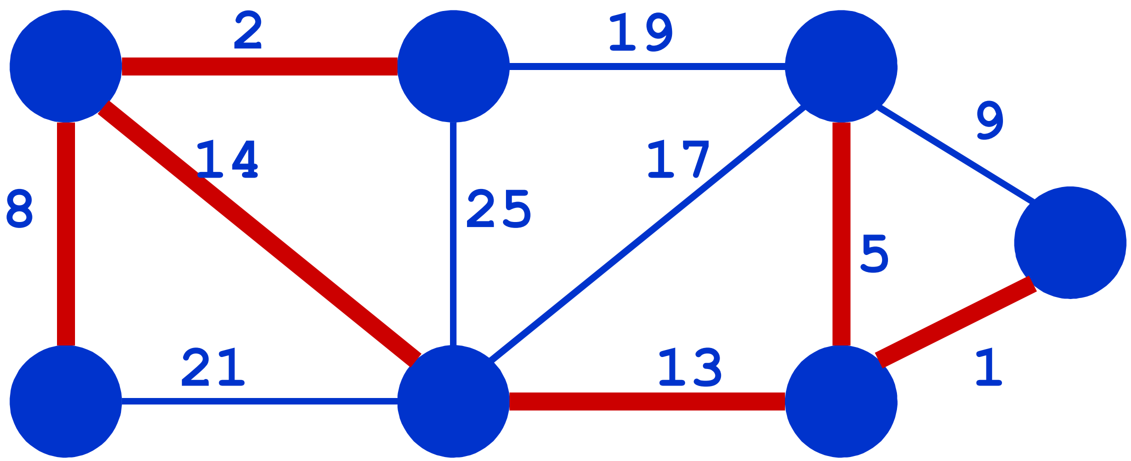



```
Kruskal()
{
    T = ∅;
    for each v ∈ V
        MakeSet(v);
    sort E by increasing edge weight w
    for each (u,v) ∈ E (in sorted order)
        if FindSet(u) ≠ FindSet(v)
            T = T U {{u,v}};
            Union(FindSet(u), FindSet(v));
}
```

Slide credit: 홍석원, 명지대학교; 김한준, 서울시립대학교; J. Lillis, UIC

Kruskal's Algorithm

Run the algorithm:

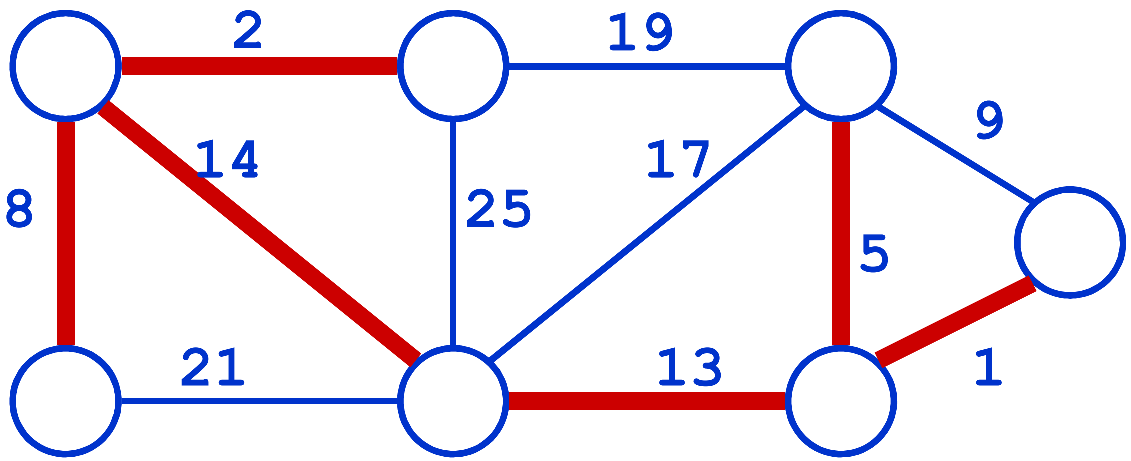



```
Kruskal()
{
    T = ∅;
    for each v ∈ V
        MakeSet(v);
    sort E by increasing edge weight w
    for each (u,v) ∈ E (in sorted order)
        if FindSet(u) ≠ FindSet(v)
            T = T U {{u,v}};
            Union(FindSet(u), FindSet(v));
}
```

Slide credit: 홍석원, 명지대학교; 김한준, 서울시립대학교; J. Lillis, UIC

Kruskal's Algorithm

Run the algorithm:




```
Kruskal()
{
    T = ∅;
    for each v ∈ V
        MakeSet(v);
    sort E by increasing edge weight w
    for each (u,v) ∈ E (in sorted order)
        if FindSet(u) ≠ FindSet(v)
            T = T U {{u,v}};
            Union(FindSet(u), FindSet(v));
}
```

Slide credit: 홍석원, 명지대학교; 김한준, 서울시립대학교; J. Lillis, UIC

Correctness Of Kruskal's Algorithm

- Sketch of a proof that this algorithm produces an MST for *T*:
  - Assume algorithm is wrong: result is not an MST
  - Then algorithm adds a wrong edge at some point
  - If it adds a wrong edge, there must be a lower weight edge (cut and paste argument)
  - But algorithm chooses lowest weight edge at each step. Contradiction
- Again, important to be comfortable with cut and paste arguments

Kruskal's Algorithm

```
Kruskal()
{
    T = ∅;
    for each v ∈ V
        MakeSet(v);
    sort E by increasing edge weight w
    for each (u,v) ∈ E (in sorted order)
        if FindSet(u) ≠ FindSet(v)
            T = T U {{u,v}};
            Union(FindSet(u), FindSet(v));
}
```

What will affect the running time?

1 Sort

O(V) MakeSet() calls

O(E) FindSet() calls

O(V) Union() calls

(*Exactly how many Union()s?*)

Slide credit: 홍석원, 명지대학교; 김한준, 서울시립대학교; J. Lillis, UIC

Kruskal's Algorithm: Running Time

- To summarize:
  - Sort edges: O(E lg E)
  - O(V) MakeSet()'s
  - O(E) FindSet()'s
  - O(V) Union()'s
- Upshot:
  - Best disjoint-set union algorithm makes above 3 operations take $O(E \cdot \alpha(V))$, α almost constant function
  - Overall thus O(E lg E), almost linear w/o sorting

Minimum cost edge selection

PRIM'S ALGORITHM FOR MST

Slide credit: 홍석원, 명지대학교; 김한준, 서울시립대학교; J. Lillis, UIC

Overview of Prim’s Algorithm

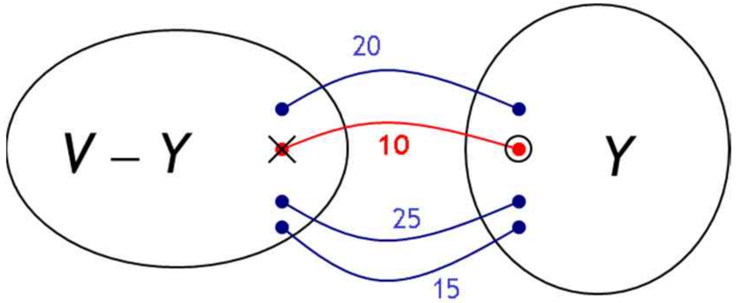
1. 그래프에서 하나의 꼭지점을 시작점으로 선택한다
2. 그래프의 모든 변이 들어 있는 집합을 만든다
3. 모든 꼭지점이 트리에 포함되어 있지 않은 동안
   - 트리와 연결된 변 가운데 트리 속의 두 꼭지점을 연결하지 않는 *가장 가중치가 작은* 변을 트리에 추가한다
   - 알고리즘이 종료됐을 때 만들어진 트리는 최소 비용 신장트리가 된다.

Source: Wikipedia.org

Slide credit: 홍석원, 명지대학교; 김한준, 서울시립대학교; J. Lillis, UIC

Prim's Algorithm

- 문제: 비방향성 그래프 G = (V,E)가 주어졌을 때, F⊆E를 만족하면서, T=(V,F)가 G의 MST가 되는 F를 찾는 문제.

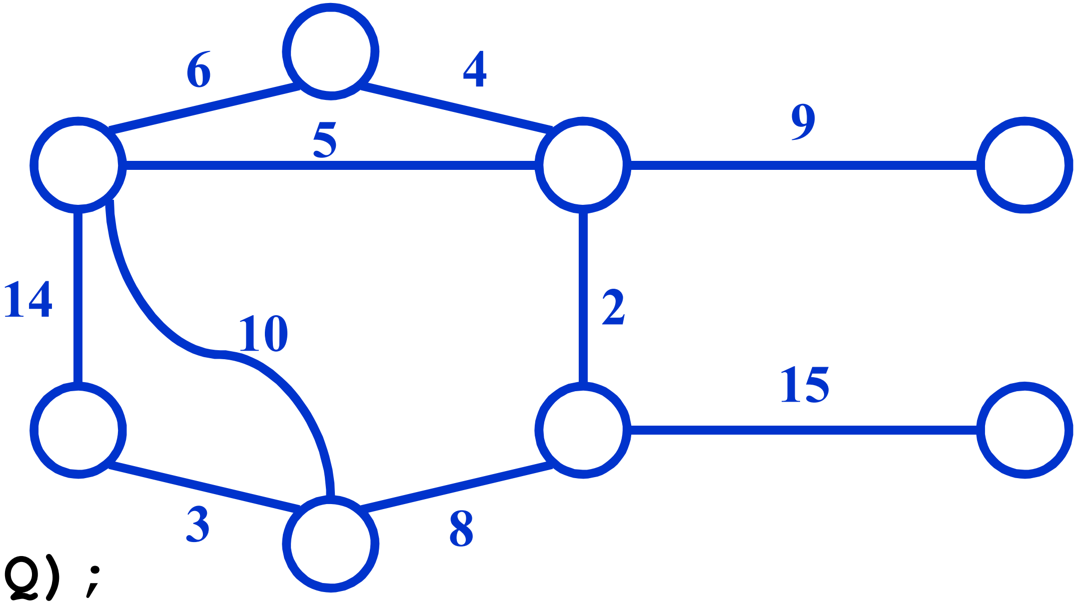


1. F := 0;
2. Y := {v0}; //정점들의 부분 집합
3. 최종해답을 얻을 때까지 다음 절차를 계속 반복하라
   - (**<u>ExtractMIN</u>**) 선정 절차/적정성 점검: Y 에 속한 임의의 정점과 가장 가까운 (즉, 가중치가 가장 낮은) V-Y 에 속한 정점 하나를 선정한다. (자연스럽게 순환은 생기지 않는다.)
   - 두 정점을 연결하는 이음선을 F에 추가한다.
   - 선정한 정점을 Y에 추가한다.
   - 해답 점검: Y = V가 되면, T = (V,F)가 최소비용 신장트리이다.

Slide credit: 홍석원, 명지대학교; 김한준, 서울시립대학교; J. Lillis, UIC

Prim's Algorithm

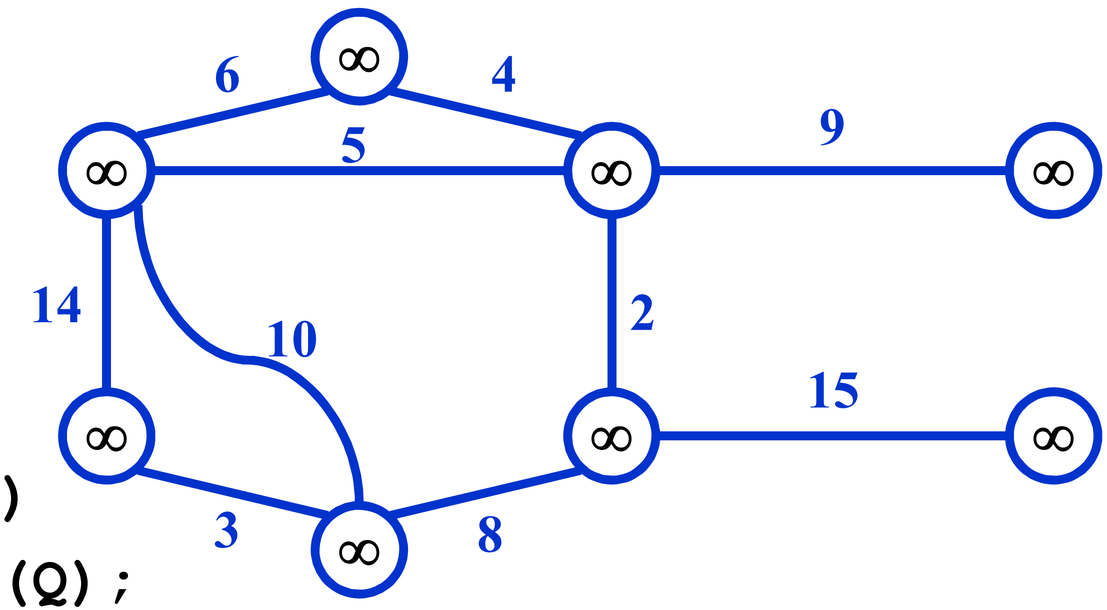
```
MST-Prim(G, w, r)
    Q = V[G];
    for each u ∈ Q
        key[u] = ∞;
    key[r] = 0;
    p[r] = NULL;
    while (Q not empty)
        u = ExtractMin(Q);
        for each v ∈ Adj[u]
            if (v ∈ Q and w(u,v) < key[v])
                p[v] = u;
                key[v] = w(u,v);
```



Run on example graph

Prim's Algorithm

```
MST-Prim(G, w, r)
    Q = V[G];
    for each u ∈ Q
        key[u] = ∞;
    key[r] = 0;
    p[r] = NULL;
    while (Q not empty)
        u = ExtractMin(Q);
        for each v ∈ Adj[u]
            if (v ∈ Q and w(u,v) < key[v])
                p[v] = u;
                key[v] = w(u,v);
```

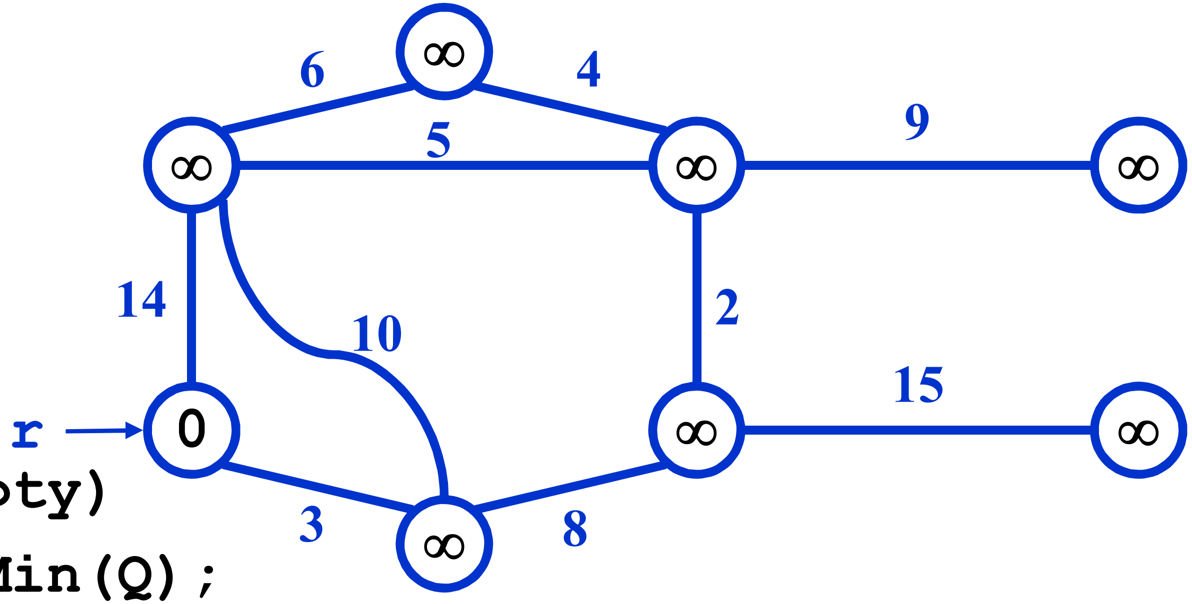


Run on example graph

Slide credit: 홍석원, 명지대학교; 김한준, 서울시립대학교; J. Lillis, UIC

Prim's Algorithm

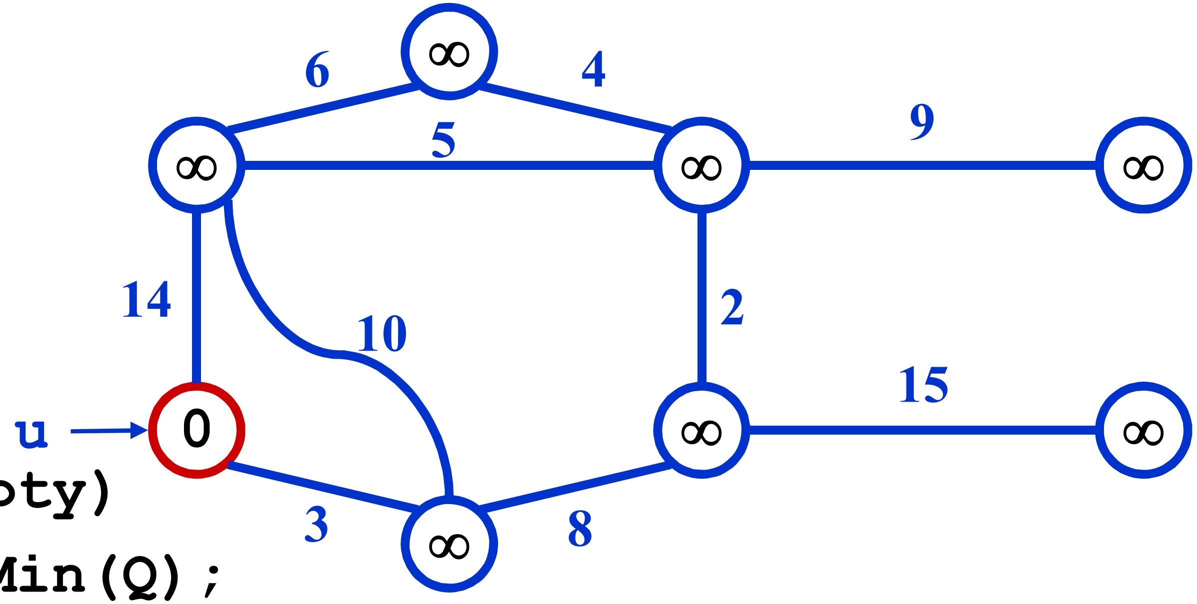
```
MST-Prim(G, w, r)
    Q = V[G];
    for each u ∈ Q
        key[u] = ∞;
    key[r] = 0;
    p[r] = NULL;
    while (Q not empty)
        u = ExtractMin(Q);
        for each v ∈ Adj[u]
            if (v ∈ Q and w(u,v) < key[v])
                p[v] = u;
                key[v] = w(u,v);
```



Pick a start vertex r

Prim’s Algorithm

```
MST-Prim(G, w, r)
    Q = V[G];
    for each u ∈ Q
        key[u] = ∞;
    key[r] = 0;
    p[r] = NULL;
    while (Q not empty)
        u = ExtractMin(Q);
        for each v ∈ Adj[u]
            if (v ∈ Q and w(u,v) < key[v])
                p[v] = u;
                key[v] = w(u,v);
```

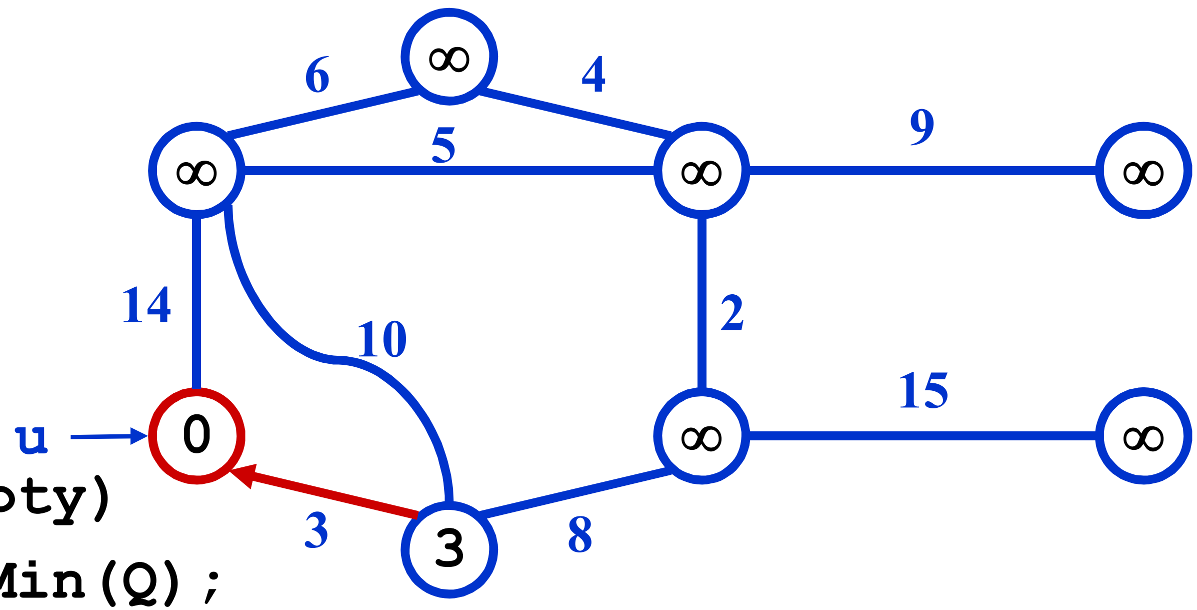


Red vertices have been removed from Q

Slide credit: 홍석원, 명지대학교; 김한준, 서울시립대학교; J. Lillis, UIC

Prim's Algorithm

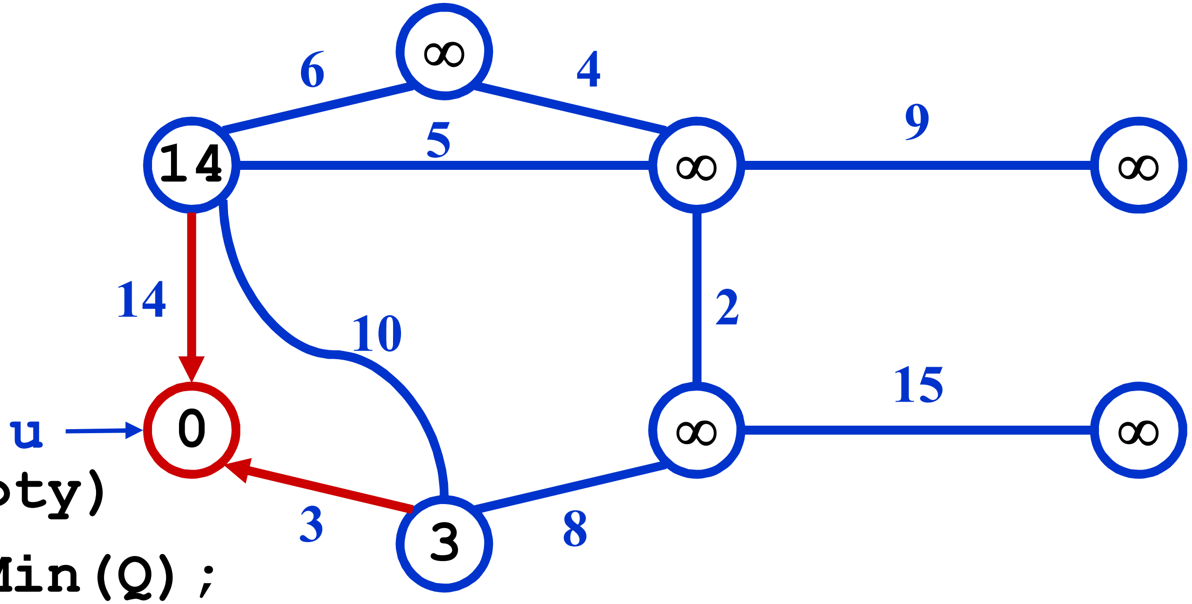
```
MST-Prim(G, w, r)
    Q = V[G];
    for each u ∈ Q
        key[u] = ∞;
    key[r] = 0;
    p[r] = NULL;
    while (Q not empty)
        u = ExtractMin(Q);
        for each v ∈ Adj[u]
            if (v ∈ Q and w(u,v) < key[v])
                p[v] = u;
                key[v] = w(u,v);
```



Red arrows indicate parent pointers

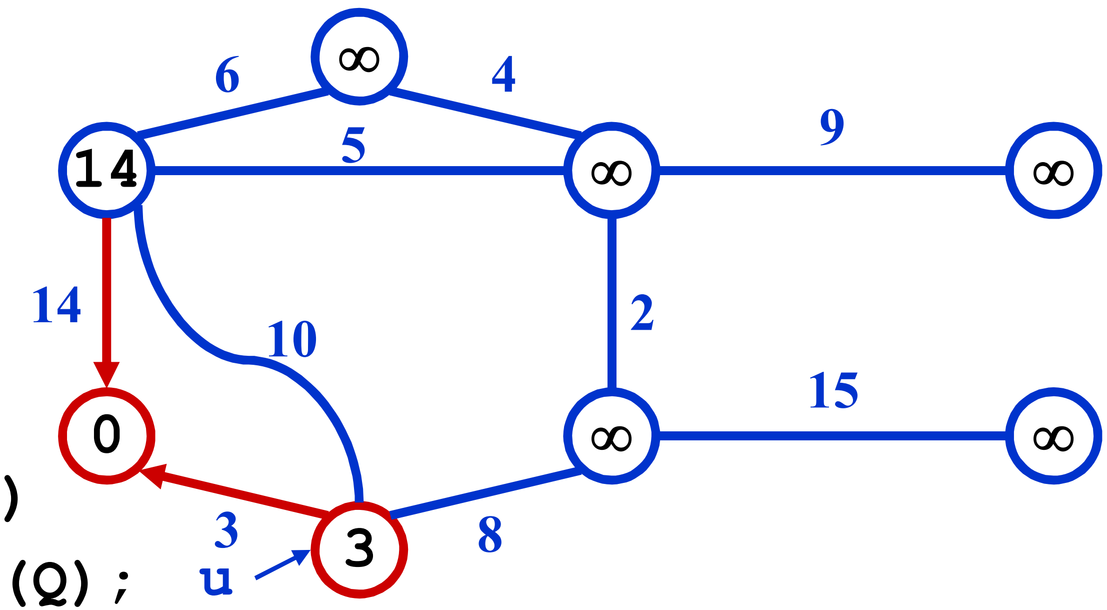
Prim's Algorithm

```
MST-Prim(G, w, r)
    Q = V[G];
    for each u ∈ Q
        key[u] = ∞;
    key[r] = 0;
    p[r] = NULL;
    while (Q not empty)
        u = ExtractMin(Q);
        for each v ∈ Adj[u]
            if (v ∈ Q and w(u,v) < key[v])
                p[v] = u;
                key[v] = w(u,v);
```



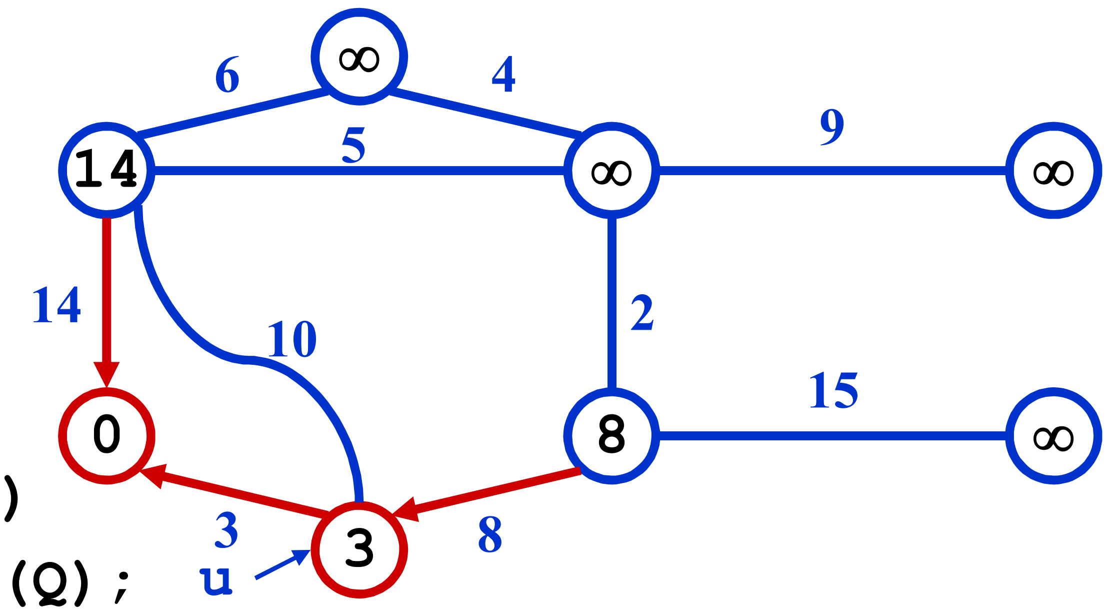
Prim's Algorithm

```
MST-Prim(G, w, r)
    Q = V[G];
    for each u ∈ Q
        key[u] = ∞;
    key[r] = 0;
    p[r] = NULL;
    while (Q not empty)
        u = ExtractMin(Q);
        for each v ∈ Adj[u]
            if (v ∈ Q and w(u,v) < key[v])
                p[v] = u;
                key[v] = w(u,v);
```



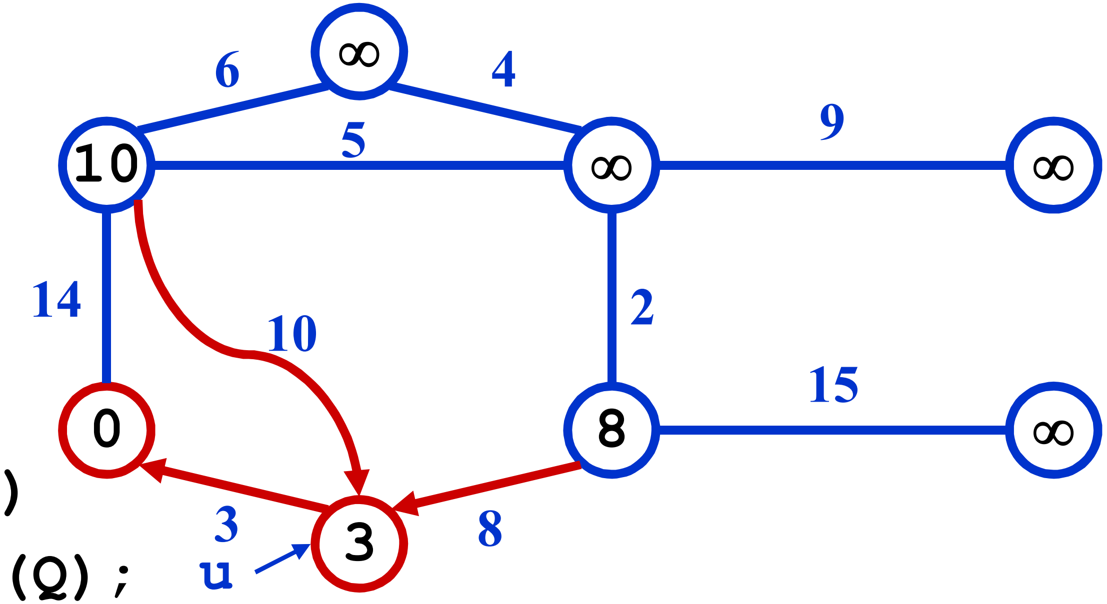
Prim's Algorithm

```
MST-Prim(G, w, r)
    Q = V[G];
    for each u ∈ Q
        key[u] = ∞;
    key[r] = 0;
    p[r] = NULL;
    while (Q not empty)
        u = ExtractMin(Q);
        for each v ∈ Adj[u]
            if (v ∈ Q and w(u,v) < key[v])
                p[v] = u;
                key[v] = w(u,v);
```



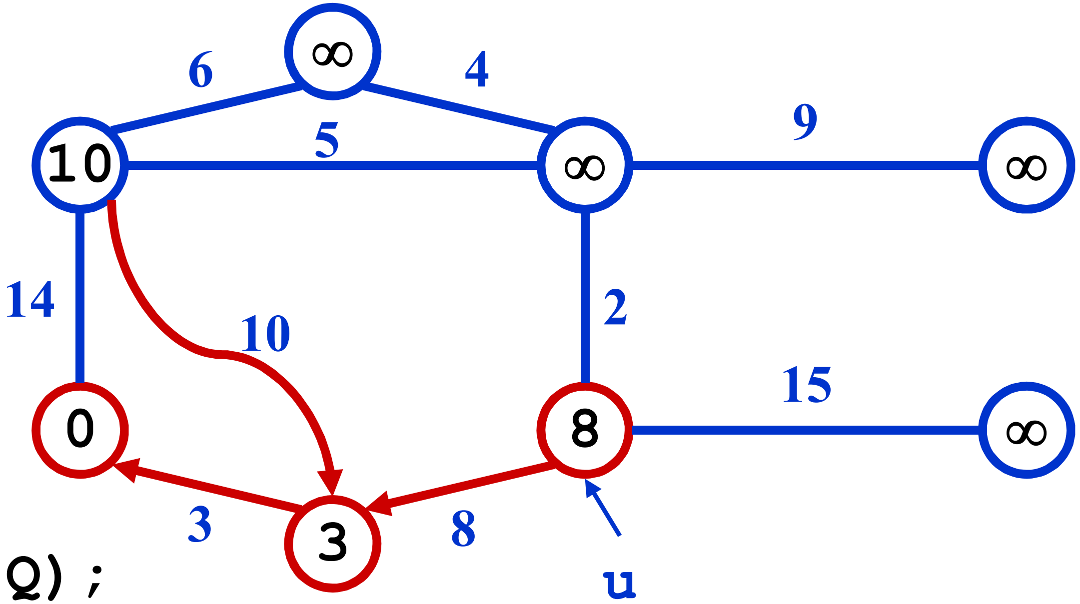
Prim's Algorithm

```
MST-Prim(G, w, r)
    Q = V[G];
    for each u ∈ Q
        key[u] = ∞;
    key[r] = 0;
    p[r] = NULL;
    while (Q not empty)
        u = ExtractMin(Q);
        for each v ∈ Adj[u]
            if (v ∈ Q and w(u,v) < key[v])
                p[v] = u;
                key[v] = w(u,v);
```



Prim's Algorithm

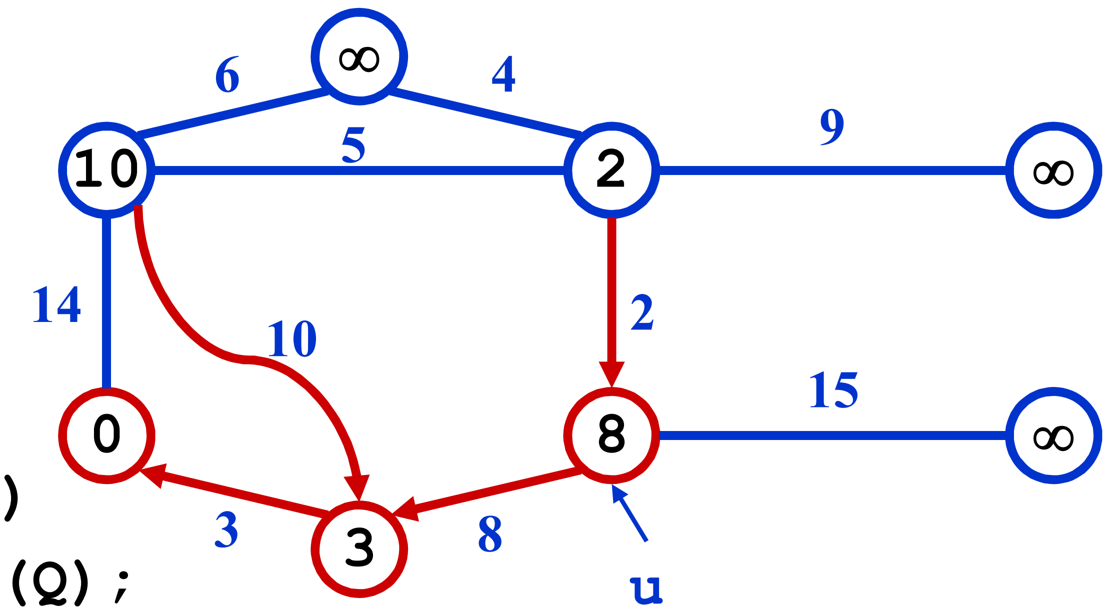
```
MST-Prim(G, w, r)
    Q = V[G];
    for each u ∈ Q
        key[u] = ∞;
    key[r] = 0;
    p[r] = NULL;
    while (Q not empty)
        u = ExtractMin(Q);
        for each v ∈ Adj[u]
            if (v ∈ Q and w(u,v) < key[v])
                p[v] = u;
                key[v] = w(u,v);
```



Slide credit: 홍석원, 명지대학교; 김한준, 서울시립대학교; J. Lillis, UIC

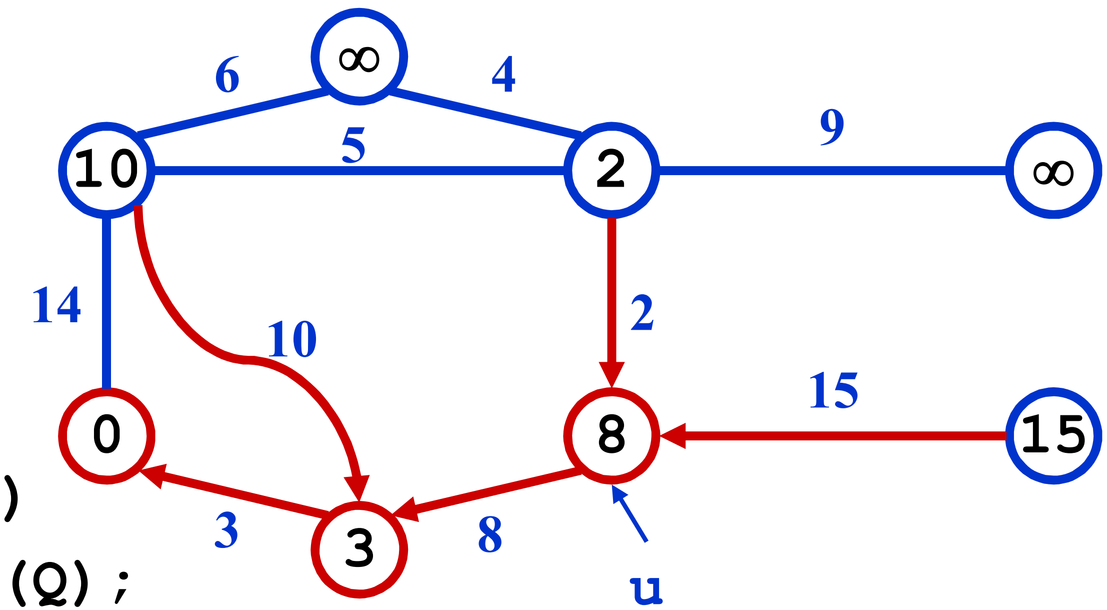
Prim's Algorithm

```
MST-Prim(G, w, r)
    Q = V[G];
    for each u ∈ Q
        key[u] = ∞;
    key[r] = 0;
    p[r] = NULL;
    while (Q not empty)
        u = ExtractMin(Q);
        for each v ∈ Adj[u]
            if (v ∈ Q and w(u,v) < key[v])
                p[v] = u;
                key[v] = w(u,v);
```



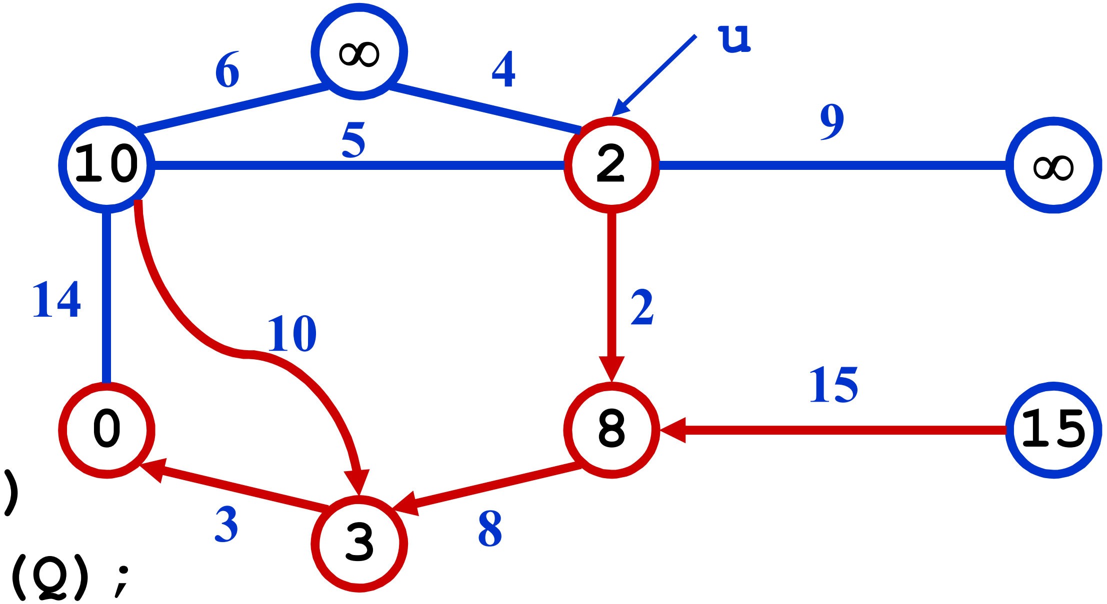
Prim's Algorithm

```
MST-Prim(G, w, r)
    Q = V[G];
    for each u ∈ Q
        key[u] = ∞;
    key[r] = 0;
    p[r] = NULL;
    while (Q not empty)
        u = ExtractMin(Q);
        for each v ∈ Adj[u]
            if (v ∈ Q and w(u,v) < key[v])
                p[v] = u;
                key[v] = w(u,v);
```



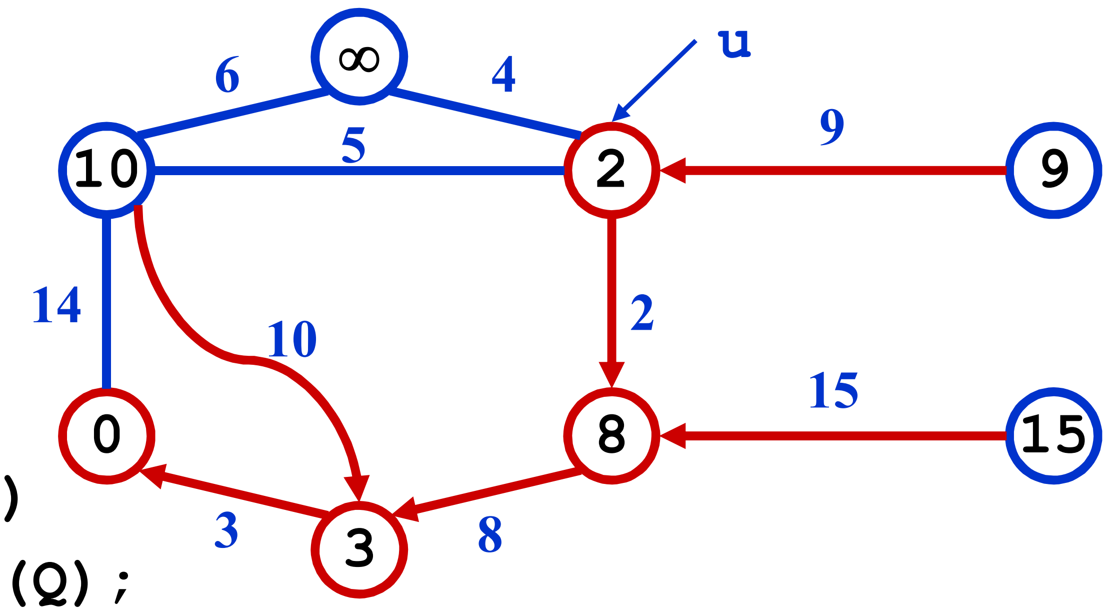
Prim's Algorithm

```
MST-Prim(G, w, r)
    Q = V[G];
    for each u ∈ Q
        key[u] = ∞;
    key[r] = 0;
    p[r] = NULL;
    while (Q not empty)
        u = ExtractMin(Q);
        for each v ∈ Adj[u]
            if (v ∈ Q and w(u,v) < key[v])
                p[v] = u;
                key[v] = w(u,v);
```



Prim's Algorithm

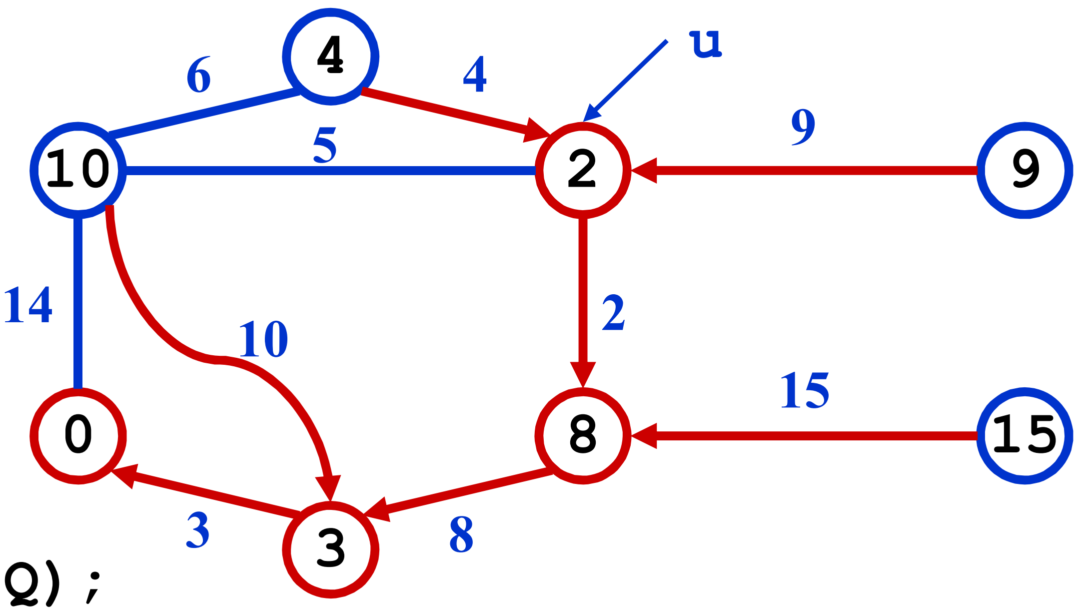
```
MST-Prim(G, w, r)
    Q = V[G];
    for each u ∈ Q
        key[u] = ∞;
    key[r] = 0;
    p[r] = NULL;
    while (Q not empty)
        u = ExtractMin(Q);
        for each v ∈ Adj[u]
            if (v ∈ Q and w(u,v) < key[v])
                p[v] = u;
                key[v] = w(u,v);
```



Slide credit: 홍석원, 명지대학교; 김한준, 서울시립대학교; J. Lillis, UIC

Prim's Algorithm

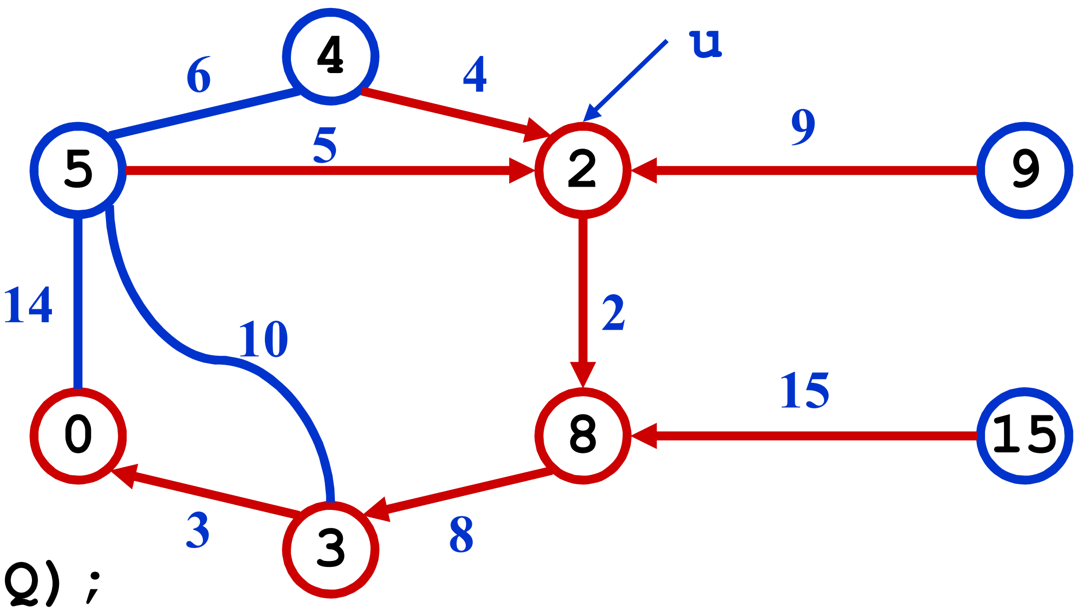
```
MST-Prim(G, w, r)
    Q = V[G];
    for each u ∈ Q
        key[u] = ∞;
    key[r] = 0;
    p[r] = NULL;
    while (Q not empty)
        u = ExtractMin(Q);
        for each v ∈ Adj[u]
            if (v ∈ Q and w(u,v) < key[v])
                p[v] = u;
                key[v] = w(u,v);
```



Slide credit: 홍석원, 명지대학교; 김한준, 서울시립대학교; J. Lillis, UIC

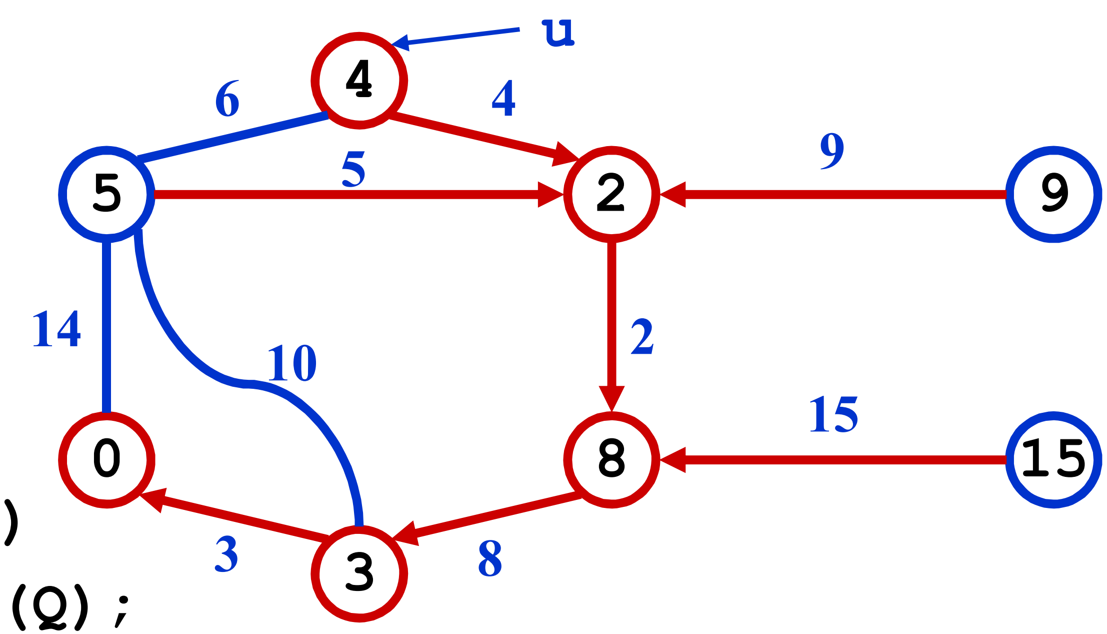
Prim's Algorithm

```
MST-Prim(G, w, r)
    Q = V[G];
    for each u ∈ Q
        key[u] = ∞;
    key[r] = 0;
    p[r] = NULL;
    while (Q not empty)
        u = ExtractMin(Q);
        for each v ∈ Adj[u]
            if (v ∈ Q and w(u,v) < key[v])
                p[v] = u;
                key[v] = w(u,v);
```



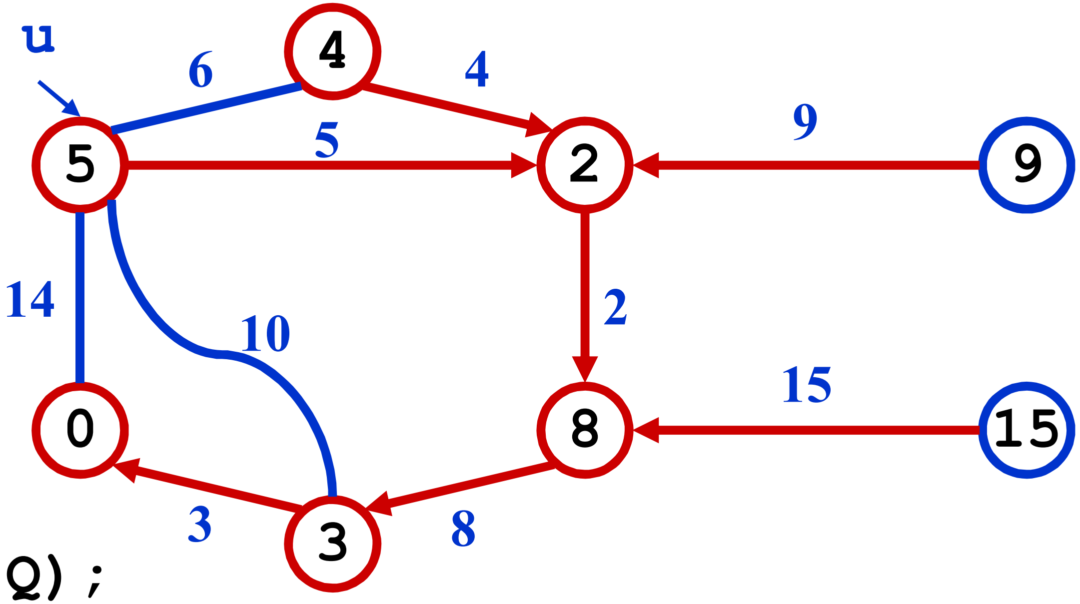
Prim's Algorithm

```
MST-Prim(G, w, r)
    Q = V[G];
    for each u ∈ Q
        key[u] = ∞;
    key[r] = 0;
    p[r] = NULL;
    while (Q not empty)
        u = ExtractMin(Q);
        for each v ∈ Adj[u]
            if (v ∈ Q and w(u,v) < key[v])
                p[v] = u;
                key[v] = w(u,v);
```



Prim's Algorithm

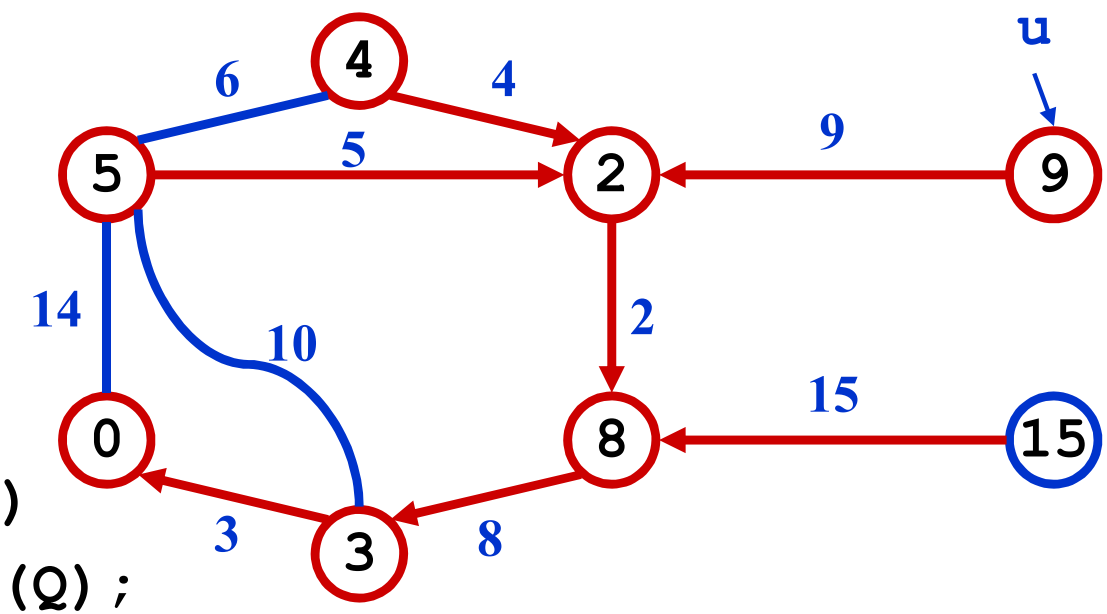
```
MST-Prim(G, w, r)
    Q = V[G];
    for each u ∈ Q
        key[u] = ∞;
    key[r] = 0;
    p[r] = NULL;
    while (Q not empty)
        u = ExtractMin(Q);
        for each v ∈ Adj[u]
            if (v ∈ Q and w(u,v) < key[v])
                p[v] = u;
                key[v] = w(u,v);
```



Slide credit: 홍석원, 명지대학교; 김한준, 서울시립대학교; J. Lillis, UIC

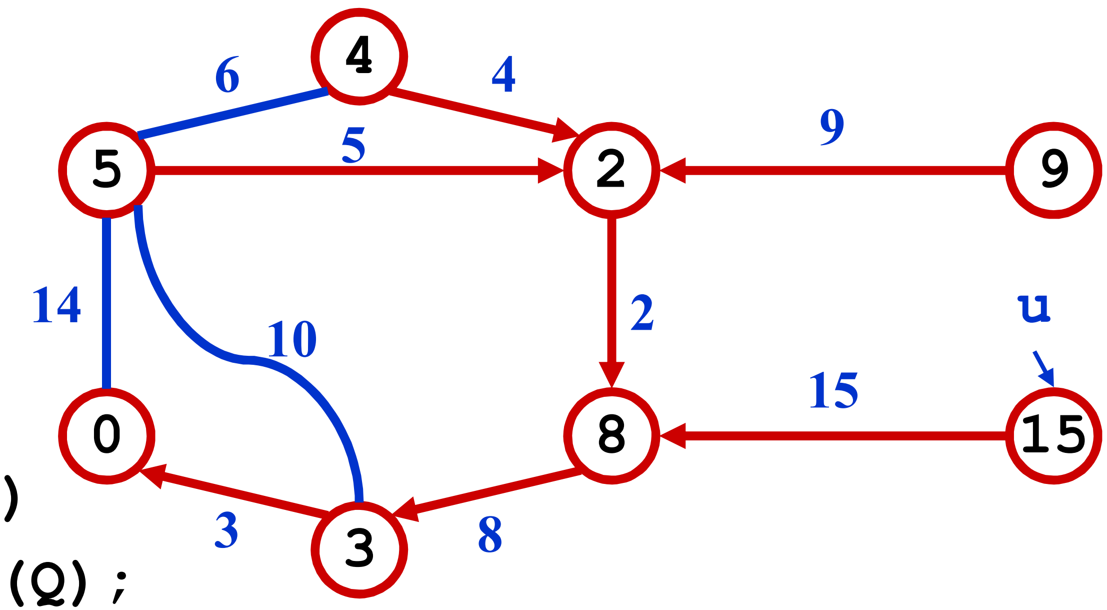
Prim's Algorithm

```
MST-Prim(G, w, r)
    Q = V[G];
    for each u ∈ Q
        key[u] = ∞;
    key[r] = 0;
    p[r] = NULL;
    while (Q not empty)
        u = ExtractMin(Q);
        for each v ∈ Adj[u]
            if (v ∈ Q and w(u,v) < key[v])
                p[v] = u;
                key[v] = w(u,v);
```



Prim's Algorithm

```
MST-Prim(G, w, r)
    Q = V[G];
    for each u ∈ Q
        key[u] = ∞;
    key[r] = 0;
    p[r] = NULL;
    while (Q not empty)
        u = ExtractMin(Q);
        for each v ∈ Adj[u]
            if (v ∈ Q and w(u,v) < key[v])
                p[v] = u;
                key[v] = w(u,v);
```



Analysis of Prim's Algorithm

```
MST-Prim(G, w, r)
  Q = V[G];
  for each u ∈ Q
    key[u] = ∞;
  key[r] = 0;
  p[r] = NULL;
  while (Q not empty)
    u = ExtractMin(Q);
    for each v ∈ Adj[u]
      if (v ∈ Q and
       w(u,v) < key[v])
         p[v] = u;
         key[v] = w(u,v);
```

- 단위연산:
  - while-루프 안에 있는 for-루프 내부에 있는 명령문(비교문 및 지정문)
- 입력크기: 노드의 개수, n
- 모든 경우 분석:
  - while-루프가 n-1번 반복되고,
  - 각 루프마다 for-루프가 n-1번씩 수행되므로 (모든 경우의) 시간복잡도는 다음과 같다.
- $T(n) = 2(n-1)(n-1) \in O(n^2)$

Slide credit: 홍석원, 명지대학교; 김한준, 서울시립대학교; J. Lillis, UIC

Various Implementations of Prim's

- ExtractMin 의 구현방법에 따라 결정됨
  - Y 에 속한 임의의 정점과 가장 가까운 (즉, 가중치가 가장 낮은) V-Y 에 속한 정점 하나를 선정한다.
- $\Theta(V^2)$
  - 인접 행렬과 최소 비용값들이 들어가 있는 배열 내에서 최소 비용값을 찾는 검색을 사용할 경우
- $\Theta(E\log V)$
  - 이진 힙 자료 구조(binary heap)와 인접 리스트(adjacency list) 표현을 사용
- $\Theta(E+V\log V)$, $\Theta(V\log V)$ for dense graph
  - 피보나치 힙(Fibonacci heap)을 사용

Slide credit: 홍석원, 명지대학교; 김한준, 서울시립대학교; J. Lillis, UIC

Optimal Substructure for MST

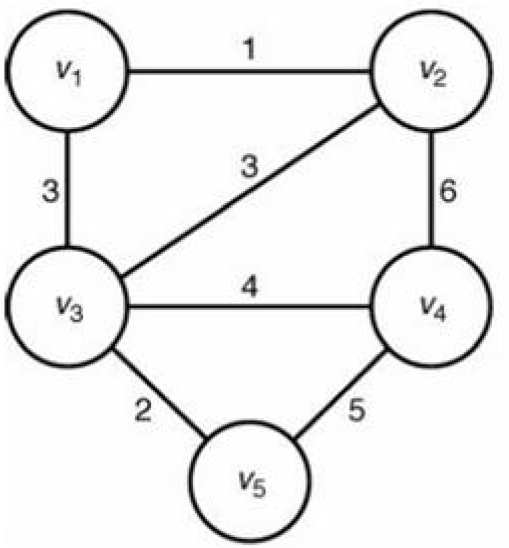
- 수학적 귀납법을 통해 다음 명제를 증명하여 보인다.
  - "프림 알고리즘 각 단계에서 얻는 그래프의 간선들을 원소로 하는 집합을 포함한 최소비용 신장트리가 항상 존재한다"
- MSTs satisfy the *optimal substructure* property: an optimal tree is composed of optimal subtrees
  - Let T be an MST of G with an edge (*u,v*) in the middle
  - Removing (*u,v*) partitions T into two trees T_1 and T_2
  - Claim: T_1 is an MST of $G_1 = (V_1,E_1)$, and T_2 is an MST of $G_2 = (V_2,E_2)$
    (*Do V_1 and V_2 share vertices? Why?*)
  - Proof: $w(T) = w(u,v) + w(T_1) + w(T_2)$
    (There can't be a better tree than T_1 or T_2, or T would be suboptimal)

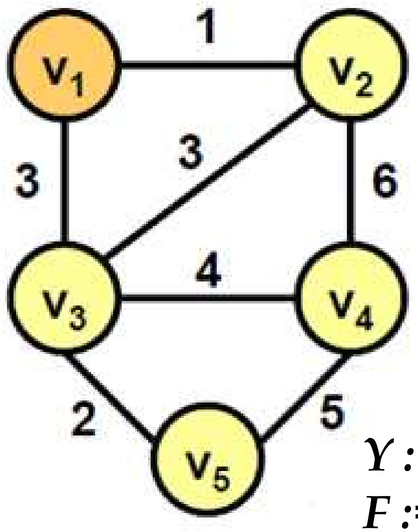
Slide credit: 홍석원, 명지대학교; 김한준, 서울시립대학교; J. Lillis, UIC

Proof of Prim's Algorithm

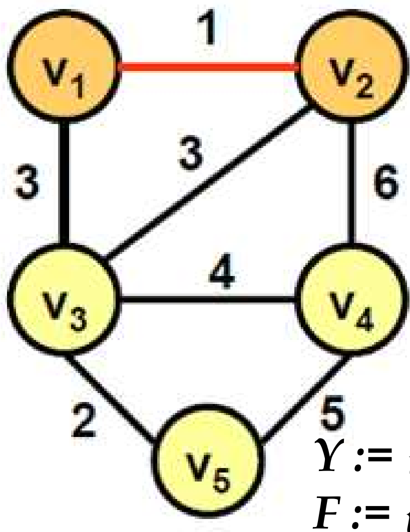
- 프림 알고리즘으로 얻은 그래프 P는 V개의 꼭지점을 연결하고 있고 V-1개의 변을 가지므로 트리이며 모든 점 V개를 가지므로 신장트리이다.
- P가 최소비용임은 optimal substructure 명제와 귀류법(모순에 의한 증명, proof by contradiction)을 이용하여 증명 가능하다.
  1. 현재 V-1개의 꼭지점으로 되어 있는 MST가 있다고 가정하자
  2. 1개의 꼭지점을 추가한다. ExtractMin()에 의해 Prim's algorithm은 최소의 weight (w1)를 가진 간선을 선택하여 V개의 꼭지점으로 되어 있는 Spanning Tree를 생성한다.
  3. 만약 Prim's algorithm의 최적이 아니라면 ExtractMin이 선택하지 않은 다른 간선(w2)이 MST를 만들게 된다.
  4. 그런데 ExtractMin()의 정의에 따라 w1 < w2 ➔ Weight(V-1)+w1 < Weight(V-1)+w2 --- !!contradiction!!

MST – Prim 알고리즘 (추상적)

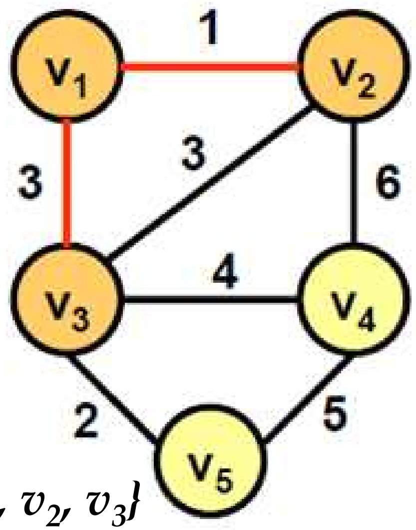




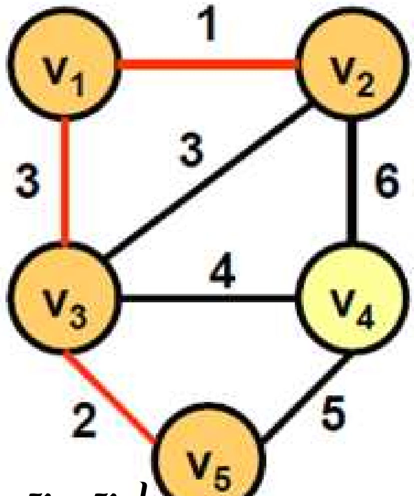
$Y := \{v_1\}$
$F := \{\}$



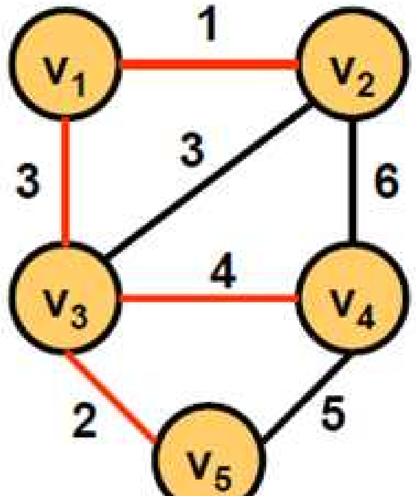
$Y := \{v_1, v_2\}$
$F := \{(v_1, v_2)\}$



$Y := \{v_1, v_2, v_3\}$
$F := \{(v_1, v_2), (v_1, v_3)\}$



$Y := \{v_1, v_2, v_3, v_5\}$
$F := \{(v_1, v_2), (v_1, v_3), (v_3, v_5)\}$



$Y := \{v_1, v_2, v_3, v_5, v_4\}$
$F := \{(v_1, v_2), (v_1, v_3), (v_3, v_5), (v_3, v_4)\}$

Slide credit: 홍석원, 명지대학교; 김한준, 서울시립대학교; J. Lillis, UIC

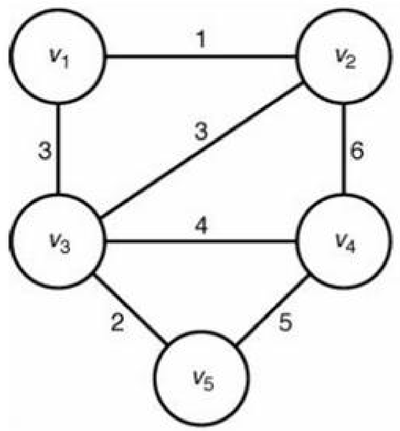
MST – Prim 알고리즘의 최적 여부 검증 (1/3)

- *Prim의 알고리즘이 찾아낸 신장트리가 최소비용(minimal)인지를 검증해야 한다. 다시 말하면, Prim의 알고리즘이 최적(optimal)인지를 보여야 한다.*

- *탐욕적 방법의 문제점은 이것이다. 즉, 알고리즘을 개발하기는 비교적 용이하나, 개발한 알고리즘의 최적성을 보이는 작업이 어렵다.*

MST – Prim 알고리즘의 최적 여부 검증 (2/3)

- ***정의*** 4.1: ***비방향성 그래프*** $G = (V,E)$***가 주어졌을 때, 만약*** E***의 부분집합*** F***에 최소비용 신장트리가 되도록 이음선을 추가할 수 있으면 (즉,*** F***에 이음선들을 추가하여*** MST***가 되면)*** F***는*** ***<u>유망하다(promising)</u>라고 한다.***
  - ***옆의 그림에서*** $F_1=\{(v1, v2), (v1, v3)\}$***는 유망 하고*** $F_2=\{(v2, v4)\}$***는 유망하지 않다.***



  - ***"유망" 의 의미는 "지금까지 구성한 집합을 사용하여 최적의 솔루션을 구성할 수 있음" 을 말한다.***
  - Prim***의 알고리즘에서 구성되는 각 단계의*** F***들이 유망함을 보이면, 최적임을 보일 수 있게 된다.***

Slide credit: 홍석원, 명지대학교; 김한준, 서울시립대학교; J. Lillis, UIC

MST – Prim 알고리즘의 최적 여부 검증 (3/3)

- **보조정리** 4.1: G = (V,E)는 가중치 포함 비방향성 연결 그래프라고 하자. E의 부분집합인 F는 유망하며, Y는 F 안에 있는 이음선 들에 의해서 연결이 되어 있는 정점의 집합이라고 가정 하자. 이때, Y에 있는 어떤 정점과 V - Y에 있는 어떤 정점을 잇는 이음선 중에서 가중치가 가장 작은 이음선을 e라고 하면, F ∪ {e}는 유망하다. (즉, Prim의 방법을 사용한 F ∪ {e}는 유망하다.)
  - 증명 생략

- **정리** 4.1: <u>Prim 알고리즘은 항상 MST를 만들어 낸다</u>.
  - 개략적인 증명
    - ✓ 귀납법을 사용하여 repeat 루프가 매번 수행 후에 집합 F가 유망하다는 것을 증명함
  - 구체적 증명 생략

Next topic: NP-Completeness

END OF LECTURE 16