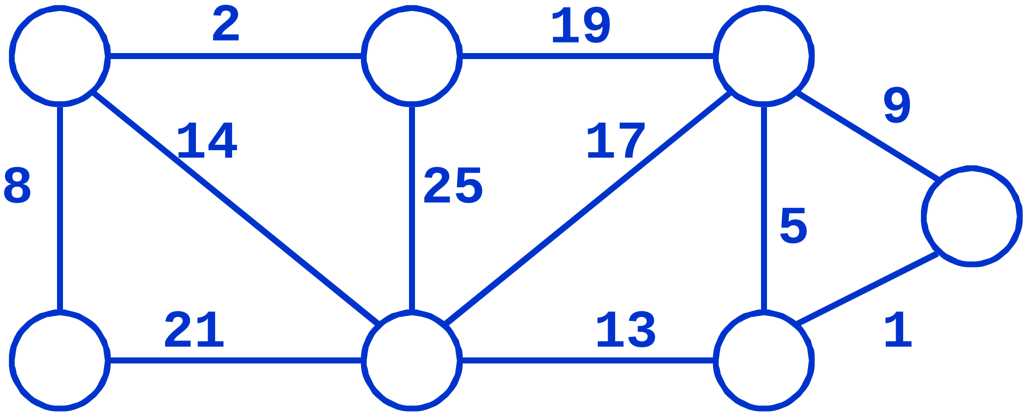
Disjoint-Set Union Problem

- Want a data structure to support disjoint sets
  - Collection of disjoint sets $S = \mathbf{U}_i \{S_i\}$, $S_i \cap S_j = \varnothing$
- Need to support following operations:
  - MakeSet(x): $S = S \mathbf{U} \{\{x\}\}$
  - Union(S_i, S_j): $S = S - \{S_i, S_j\} \mathbf{U} \{S_i \mathbf{U} S_j\}$
  - FindSet(x): return $S_i \in S$ such that $x \in S_i$
- Before discussing implementation details, we look at example application: MSTs

Kruskal’s Algorithm

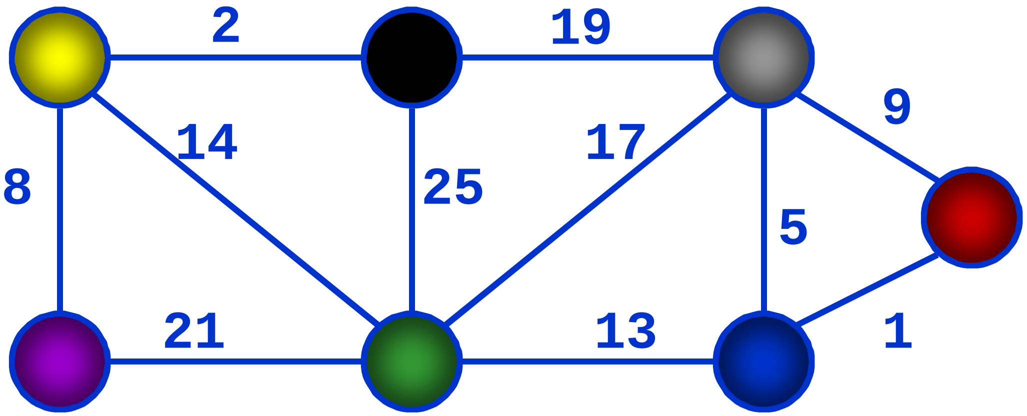
```
Kruskal()
{
   T = ∅;
   for each v ∈ V
      MakeSet(v);
   sort E into nondecreasing order by weight w
   for each (u,v) ∈ E (in sorted order)
      if FindSet(u) ≠ FindSet(v)
         T = T U {{u,v}};
         Union(FindSet(u), FindSet(v));
}
```

Kruskal’s Algorithm



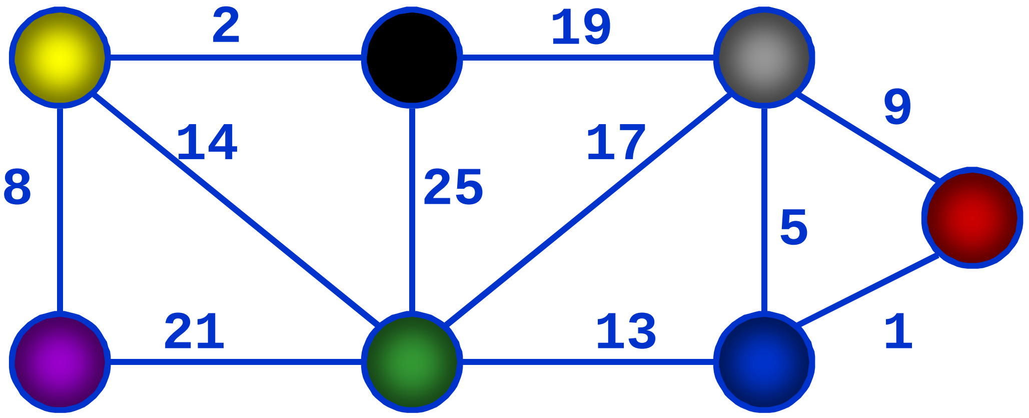
```
Kruskal()
{
    T = ∅;
    for each v ∈ V
        MakeSet(v);
    sort E into nondecreasing order by weight w
    for each (u,v) ∈ E (in sorted order)
        if FindSet(u) ≠ FindSet(v)
            T = T U {{u,v}};
            Union(FindSet(u), FindSet(v));
}
```

Kruskal’s Algorithm



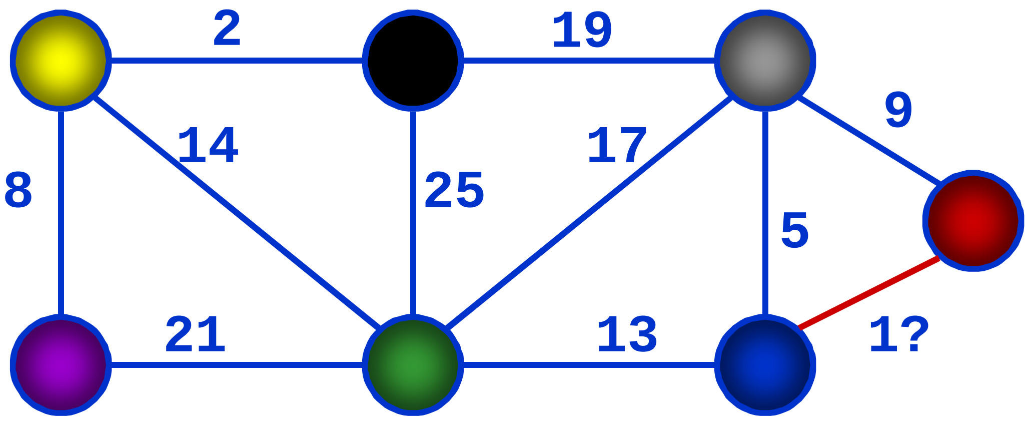
```
Kruskal()
{
    T = ∅;
    for each v ∈ V
        MakeSet(v);
    sort E into nondecreasing order by weight w
    for each (u,v) ∈ E (in sorted order)
        if FindSet(u) ≠ FindSet(v)
            T = T U {{u,v}};
            Union(FindSet(u), FindSet(v));
}
```

Kruskal’s Algorithm



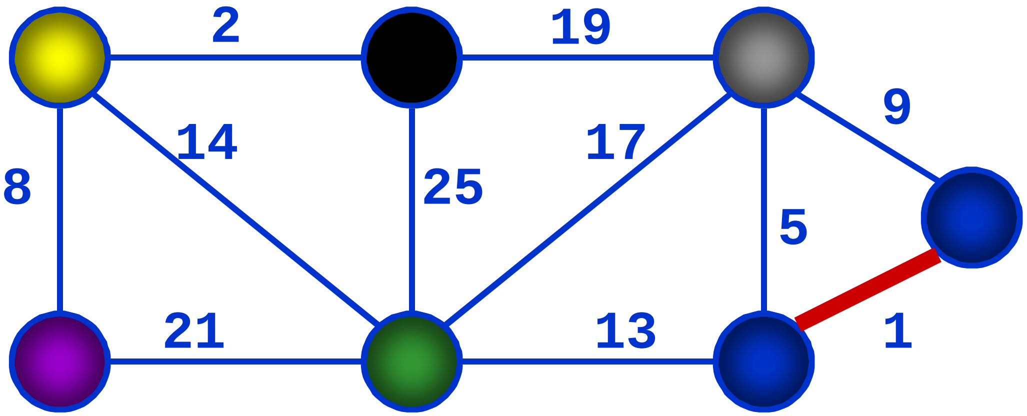
```
Kruskal()
{
    T = $\emptyset$;
    for each v $\in$ V
        MakeSet(v);
  { sort E into nondecreasing order by weight w
    for each (u,v) $\in$ E (in sorted order)
        if FindSet(u) $\neq$ FindSet(v)
            T = T U {{u,v}};
            Union(FindSet(u), FindSet(v));
}
```

Kruskal's Algorithm



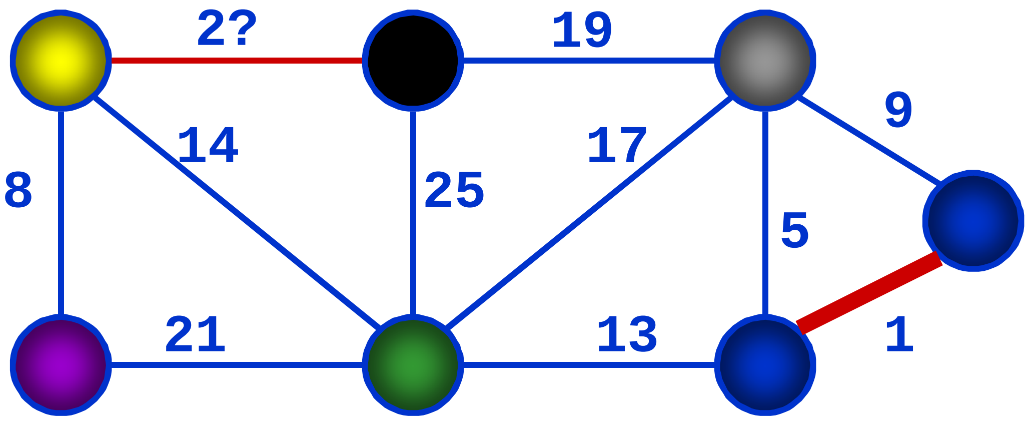
```
Kruskal()
{
    T = ∅;
    for each v ∈ V
        MakeSet(v);
    sort E into nondecreasing order by weight w
    for each (u,v) ∈ E (in sorted order)
        if FindSet(u) ≠ FindSet(v)
            T = T U {{u,v}};
            Union(FindSet(u), FindSet(v));
}
```

Kruskal’s Algorithm



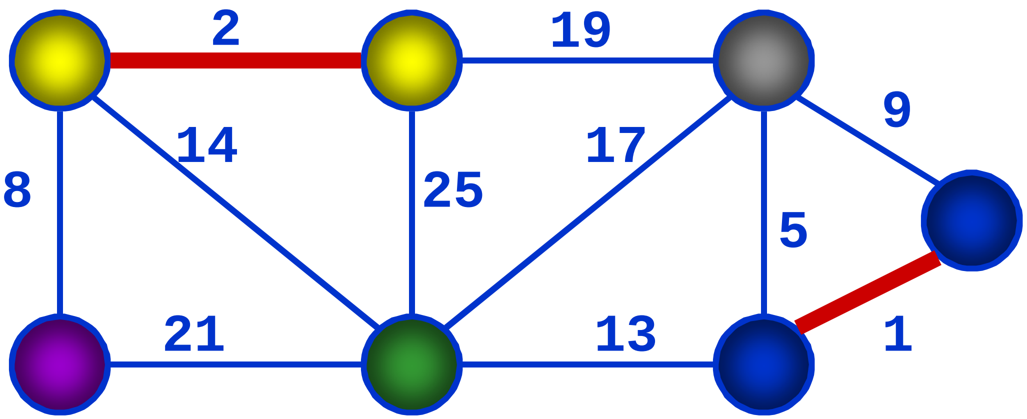
```
Kruskal()
{
    T = ∅;
    for each v ∈ V
        MakeSet(v);
    sort E into nondecreasing order by weight w
    for each (u,v) ∈ E (in sorted order)
        if FindSet(u) ≠ FindSet(v)
            T = T U {{u,v}};
            Union(FindSet(u), FindSet(v));
}
```

Kruskal’s Algorithm



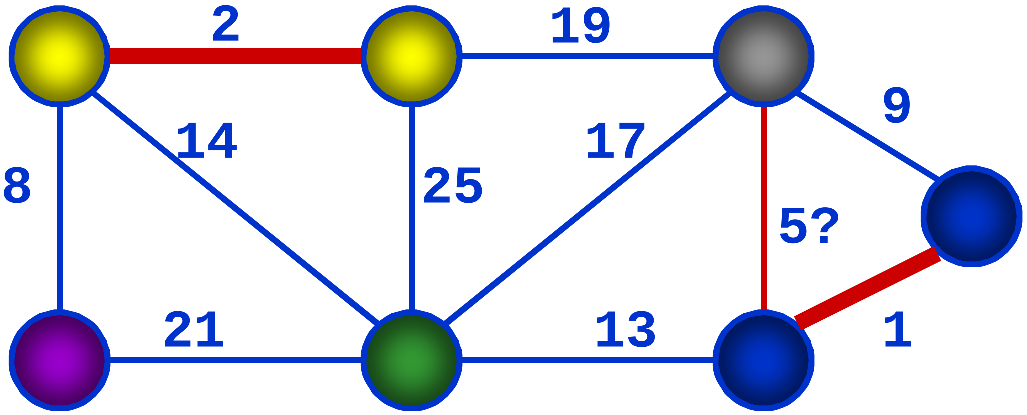
```
Kruskal()
{
    T = ∅;
    for each v ∈ V
        MakeSet(v);
    sort E into nondecreasing order by weight w
    for each (u,v) ∈ E (in sorted order)
        if FindSet(u) ≠ FindSet(v)
            T = T U {{u,v}};
            Union(FindSet(u), FindSet(v));
}
```

Kruskal's Algorithm



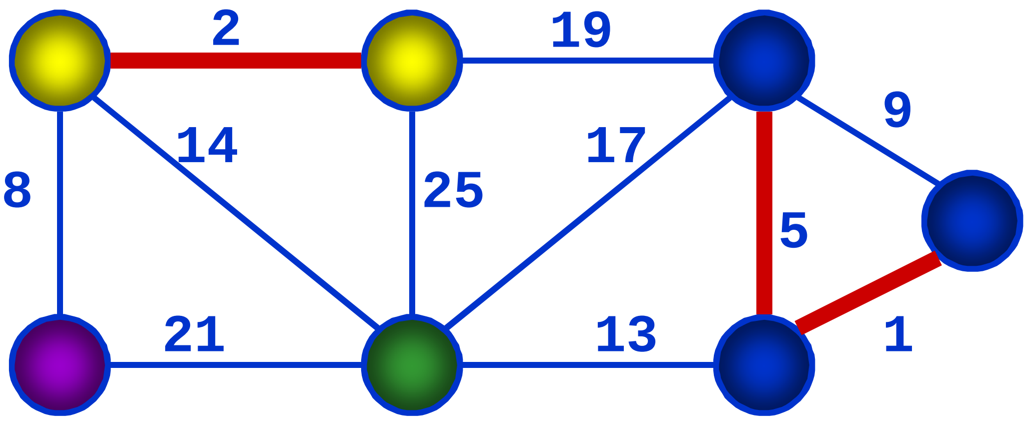
```
Kruskal()
{
    T = ∅;
    for each v ∈ V
        MakeSet(v);
    sort E into nondecreasing order by weight w
    for each (u,v) ∈ E (in sorted order)
        if FindSet(u) ≠ FindSet(v)
            T = T U {{u,v}};
            Union(FindSet(u), FindSet(v));
}
```

Kruskal’s Algorithm



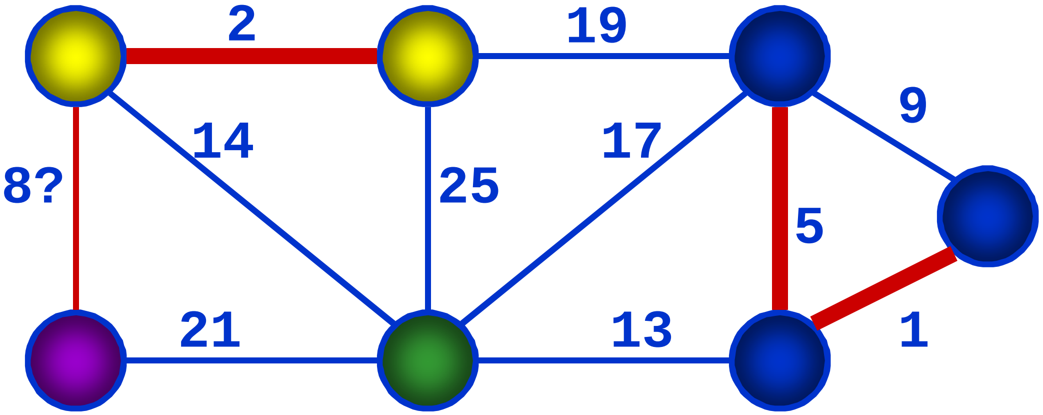
```
Kruskal()
{
   T = ∅;
   for each v ∈ V
      MakeSet(v);
   sort E into nondecreasing order by weight w
   for each (u,v) ∈ E (in sorted order)
      if FindSet(u) ≠ FindSet(v)
         T = T U {{u,v}};
         Union(FindSet(u), FindSet(v));
}
```

Kruskal's Algorithm



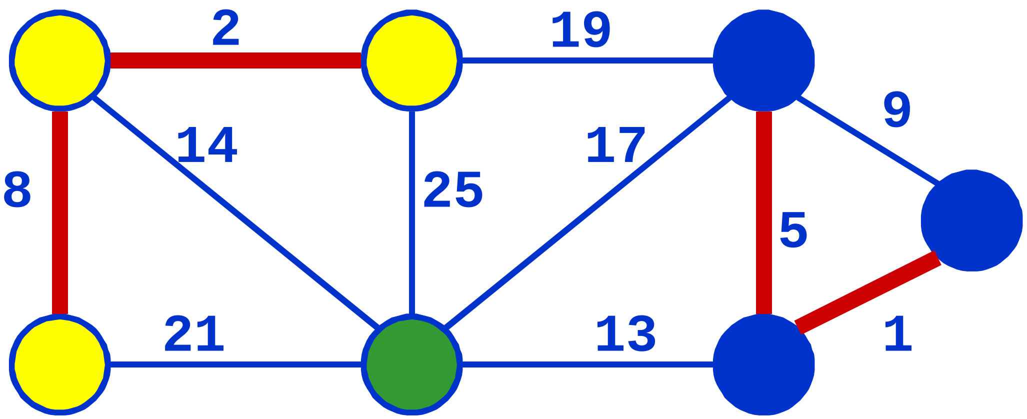
```
Kruskal()
{
    T = ∅;
    for each v ∈ V
        MakeSet(v);
    sort E into nondecreasing order by weight w
    for each (u,v) ∈ E (in sorted order)
        if FindSet(u) ≠ FindSet(v)
            T = T U {{u,v}};
            Union(FindSet(u), FindSet(v));
}
```

Kruskal’s Algorithm



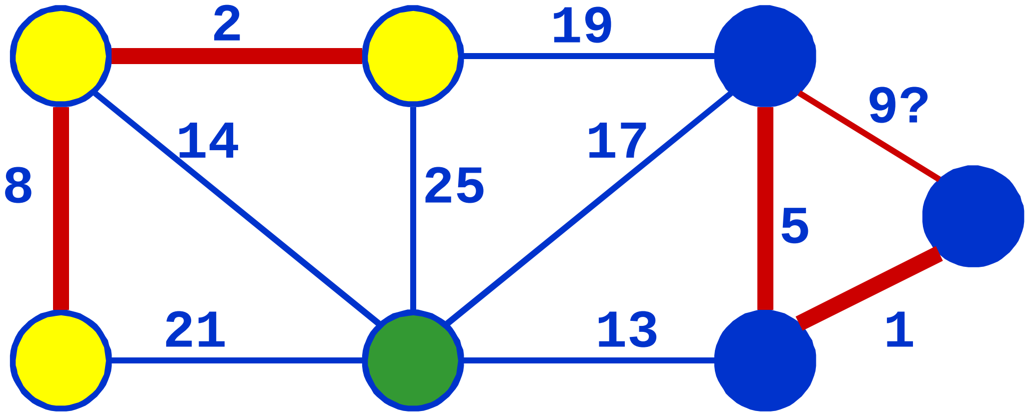
```
Kruskal()
{
    T = ∅;
    for each v ∈ V
        MakeSet(v);
    sort E into nondecreasing order by weight w
    for each (u,v) ∈ E (in sorted order)
        if FindSet(u) ≠ FindSet(v)
            T = T U {{u,v}};
            Union(FindSet(u), FindSet(v));
}
```

Kruskal’s Algorithm



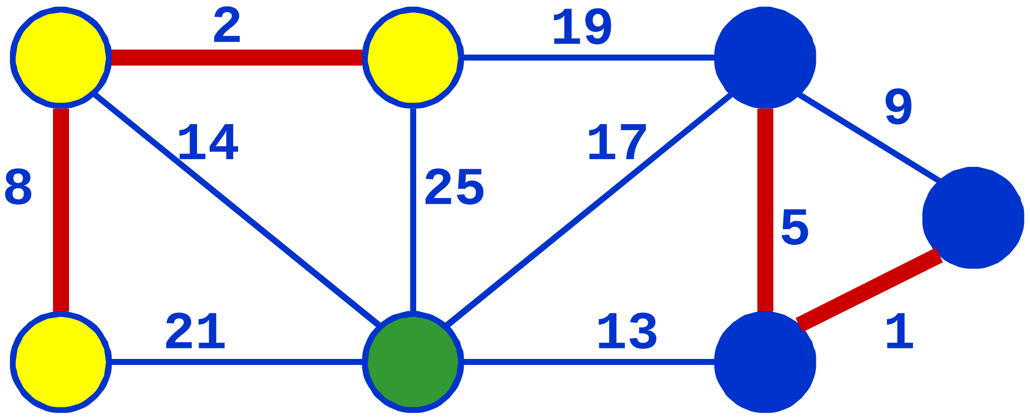
```
Kruskal()
{
    T = ∅;
    for each v ∈ V
        MakeSet(v);
    sort E into nondecreasing order by weight w
    for each (u,v) ∈ E (in sorted order)
        if FindSet(u) ≠ FindSet(v)
            T = T U {{u,v}};
            Union(FindSet(u), FindSet(v));
}
```

Kruskal’s Algorithm



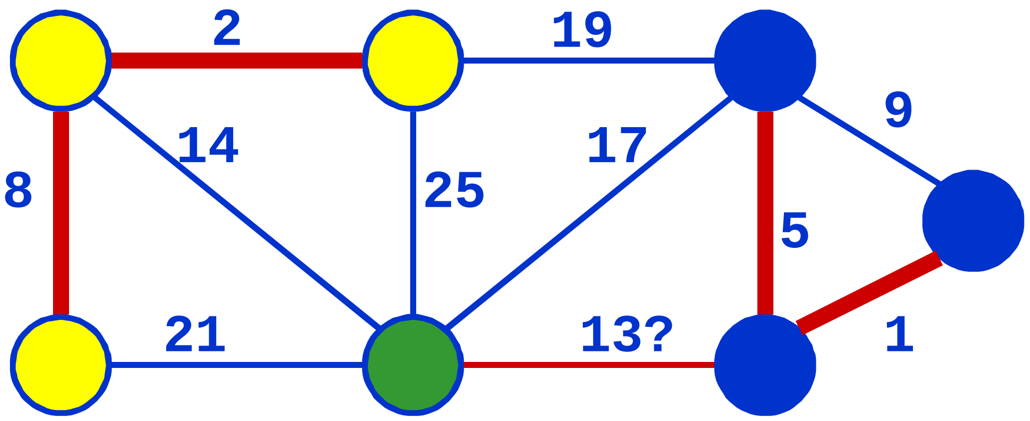
```
Kruskal()
{
    T = ∅;
    for each v ∈ V
        MakeSet(v);
    sort E into nondecreasing order by weight w
    for each (u,v) ∈ E (in sorted order)
        if FindSet(u) ≠ FindSet(v)
            T = T U {{u,v}};
            Union(FindSet(u), FindSet(v));
}
```

Kruskal's Algorithm



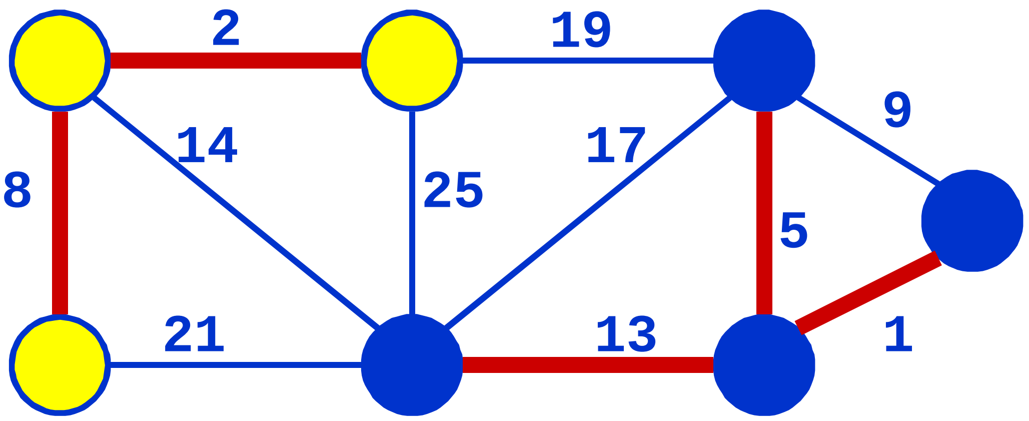
```
Kruskal()
{
    T = ∅;
    for each v ∈ V
        MakeSet(v);
    sort E into nondecreasing order by weight w
    for each (u,v) ∈ E (in sorted order)
        if FindSet(u) ≠ FindSet(v)
            T = T U {{u,v}};
            Union(FindSet(u), FindSet(v));
}
```

Kruskal’s Algorithm



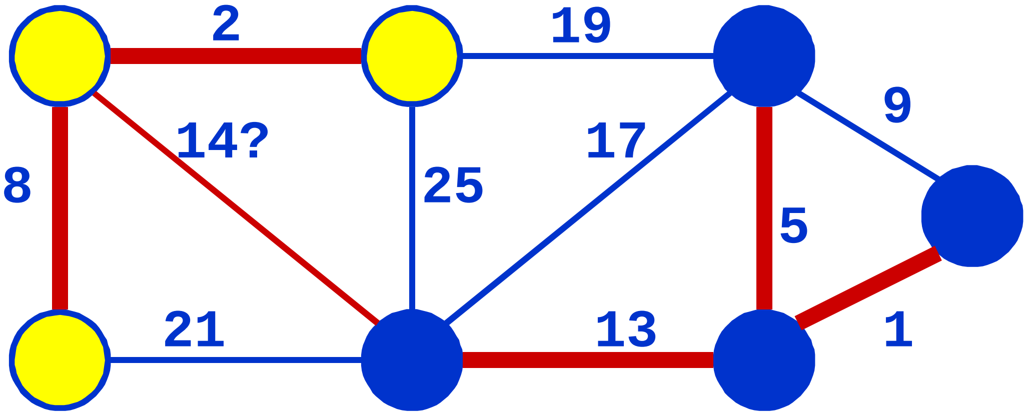
```
Kruskal()
{
    T = ∅;
    for each v ∈ V
        MakeSet(v);
    sort E into nondecreasing order by weight w
    for each (u,v) ∈ E (in sorted order)
        if FindSet(u) ≠ FindSet(v)
            T = T U {{u,v}};
            Union(FindSet(u), FindSet(v));
}
```

Kruskal’s Algorithm



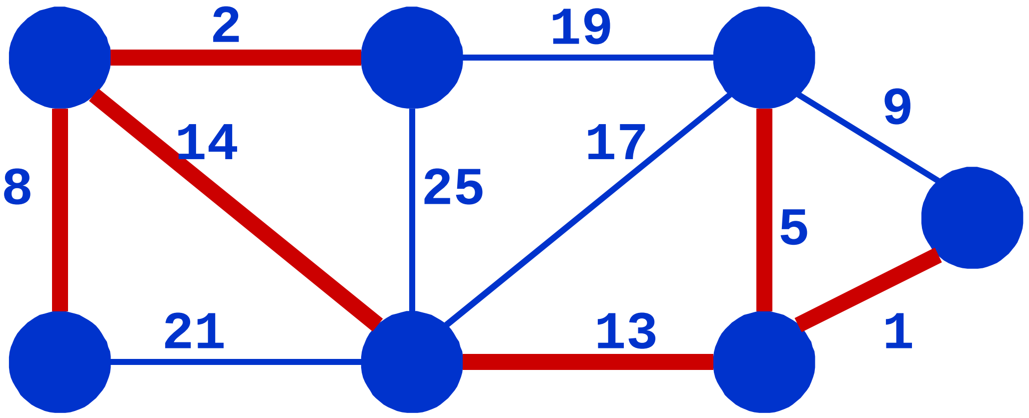
```
Kruskal()
{
    T = ∅;
    for each v ∈ V
        MakeSet(v);
    sort E into nondecreasing order by weight w
    for each (u,v) ∈ E (in sorted order)
        if FindSet(u) ≠ FindSet(v)
            T = T U {{u,v}};
            Union(FindSet(u), FindSet(v));
}
```

Kruskal’s Algorithm



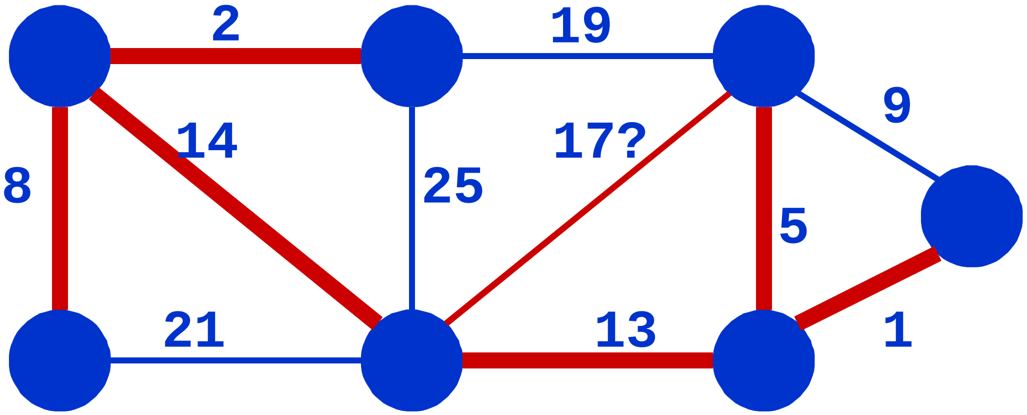
```
Kruskal()
{
    T = ∅;
    for each v ∈ V
        MakeSet(v);
    sort E into nondecreasing order by weight w
    for each (u,v) ∈ E (in sorted order)
        if FindSet(u) ≠ FindSet(v)
            T = T U {{u,v}};
            Union(FindSet(u), FindSet(v));
}
```

Kruskal's Algorithm



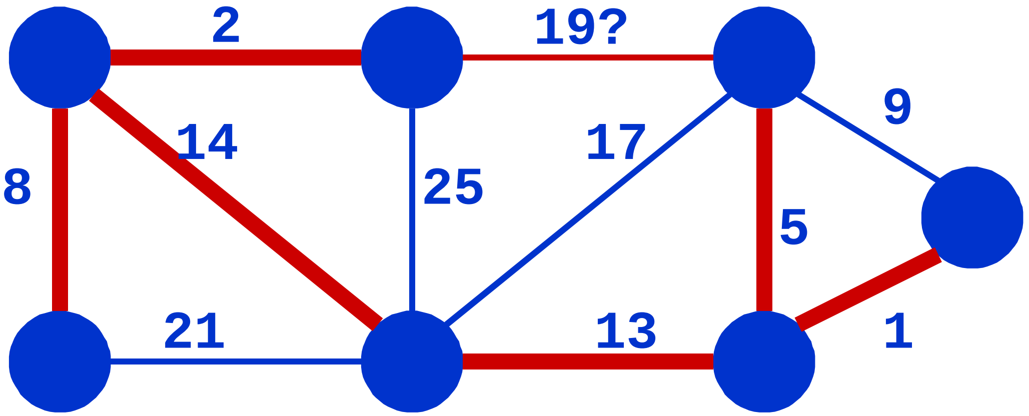
```
Kruskal()
{
   T = ∅;
   for each v ∈ V
      MakeSet(v);
   sort E into nondecreasing order by weight w
   for each (u,v) ∈ E (in sorted order)
      if FindSet(u) ≠ FindSet(v)
         T = T U {{u,v}};
         Union(FindSet(u), FindSet(v));
}
```

Kruskal’s Algorithm



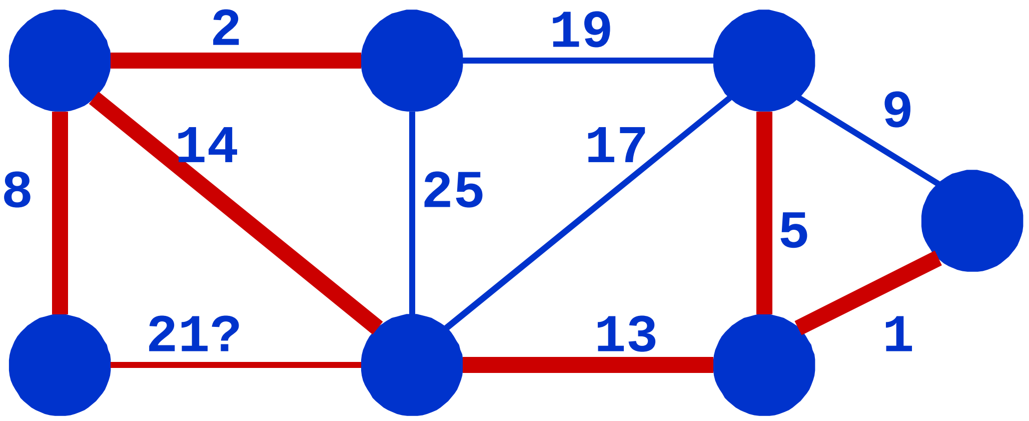
```
Kruskal()
{
    T = ∅;
    for each v ∈ V
        MakeSet(v);
    sort E into nondecreasing order by weight w
    for each (u,v) ∈ E (in sorted order)
        if FindSet(u) ≠ FindSet(v)
            T = T U {{u,v}};
            Union(FindSet(u), FindSet(v));
}
```

Kruskal’s Algorithm



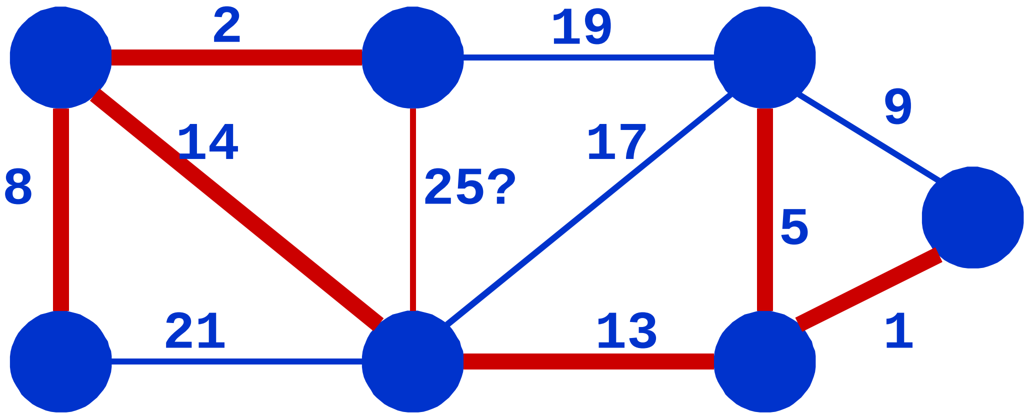
```
Kruskal()
{
   T = ∅;
   for each v ∈ V
      MakeSet(v);
   sort E into nondecreasing order by weight w
   for each (u,v) ∈ E (in sorted order)
      if FindSet(u) ≠ FindSet(v)
         T = T U {{u,v}};
         Union(FindSet(u), FindSet(v));
}
```

Kruskal’s Algorithm



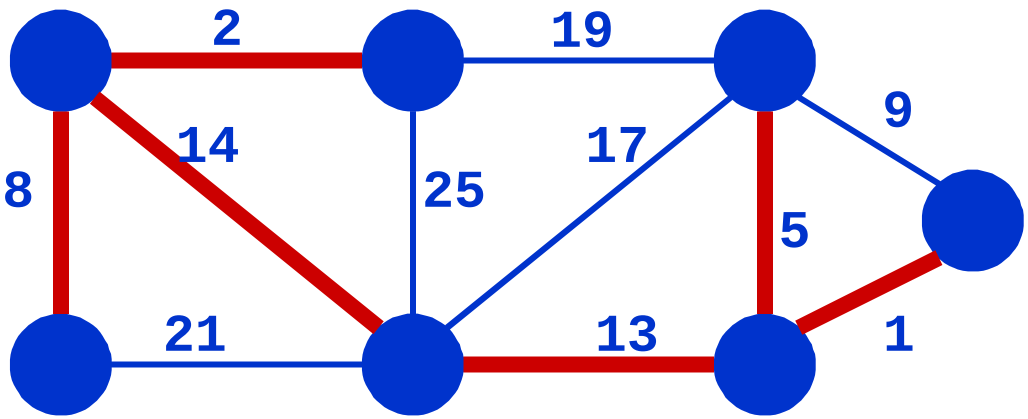
```
Kruskal()
{
    T = ∅;
    for each v ∈ V
        MakeSet(v);
    sort E into nondecreasing order by weight w
    for each (u,v) ∈ E (in sorted order)
        if FindSet(u) ≠ FindSet(v)
            T = T U {{u,v}};
            Union(FindSet(u), FindSet(v));
}
```

Kruskal's Algorithm



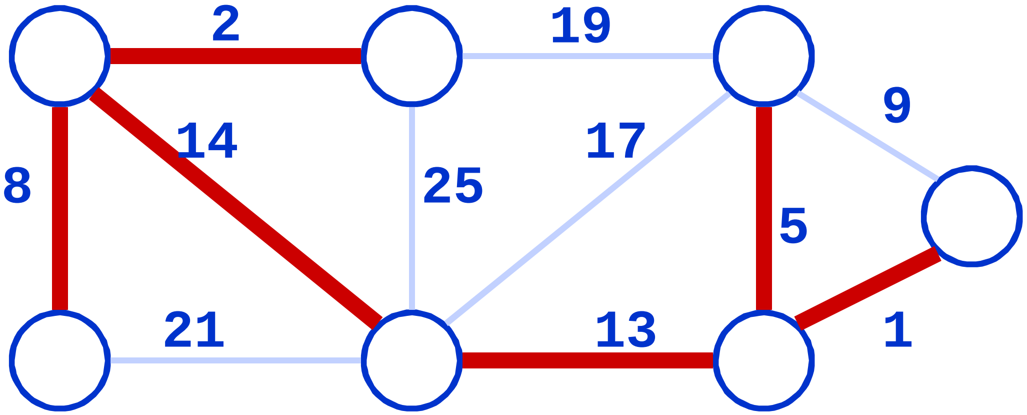
```
Kruskal()
{
    T = ∅;
    for each v ∈ V
        MakeSet(v);
    sort E into nondecreasing order by weight w
    for each (u,v) ∈ E (in sorted order)
        if FindSet(u) ≠ FindSet(v)
            T = T U {{u,v}};
            Union(FindSet(u), FindSet(v));
}
```

Kruskal’s Algorithm



```
Kruskal()
{
    T = ∅;
    for each v ∈ V
        MakeSet(v);
    sort E into nondecreasing order by 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



```
Kruskal()
{
    T = ∅;
    for each v ∈ V
        MakeSet(v);
    sort E into nondecreasing order by 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: Running Time

What will affect the running time?

```
Kruskal()
{
   T = $\emptyset$;
   for each $v \in V$
      MakeSet(v);
   sort E by increasing edge weight w
   for each $(u,v) \in E$ (in sorted order)
      if FindSet(u) $\neq$ FindSet(v)
         T = $T \cup \{\{u,v\}\}$;
         Union(FindSet(u), FindSet(v));
}
```

Kruskal's Algorithm: Running Time

```
Kruskal()
{
    T = $\emptyset$;
    for each $v \in V$
        MakeSet(v);
    sort E by increasing edge weight w
    for each $(u,v) \in E$ (in sorted order)
        if FindSet(u) $\neq$ FindSet(v)
            T = T $\cup$ {{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?*)

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 operation algorithm makes above three operations to take O(E lg E) time.
  - Thus overall time is O(E lg E) = O(E lg V), since $|E| < |V|^2$