CSE 221: Algorithms

Graph Algorithms

Mumit Khan

Computer Science and Engineering
BRAC University

References

1. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Second Edition*. The MIT Press, September 2001.
2. Jon Kleinberg and Éva Tardos, *Algorithm Design*. Pearson Education, 2006.
3. M. Goodrich and R. Tamassia, *Algorithm Design*. John-Wiley and Sons. 2002.

Last modified: July 21, 2009

Introduction to graph algorithms

- All about weighted graphs.

Introduction to graph algorithms

- All about weighted graphs.
- Minimum-cost Spanning Tree algorithms.

Introduction to graph algorithms

- All about weighted graphs.
- Minimum-cost Spanning Tree algorithms.
- Shortest Path algorithms.

Introduction to graph algorithms

- All about weighted graphs.
- Minimum-cost Spanning Tree algorithms.
- Shortest Path algorithms.
- Computing transitive closure.

Introduction to graph algorithms

- All about weighted graphs.
- Minimum-cost Spanning Tree algorithms.
- Shortest Path algorithms.
- Computing transitive closure.
- ...

Introduction to graph algorithms

- All about weighted graphs.
- Minimum-cost Spanning Tree algorithms.
- Shortest Path algorithms.
- Computing transitive closure.
- . . .
- Excellent applications of Greedy and Dynamic Programming strategies.

Contents

Spanning trees

Definition

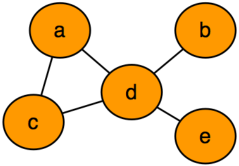
A subgraph T of a undirected graph $G = (V, E)$ is a spanning tree of G if it is a tree and contains every vertex of G.

Spanning trees

Definition

A subgraph T of a undirected graph $G = (V, E)$ is a spanning tree of G if it is a tree and contains every vertex of G.

Given the following graph:

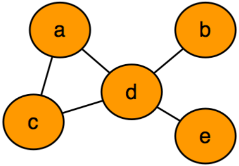


Spanning trees

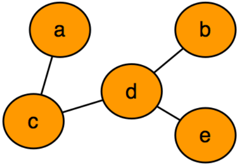
Definition

A subgraph T of a undirected graph $G = (V, E)$ is a spanning tree of G if it is a tree and contains every vertex of G.

Given the following graph:



The spanning trees are:

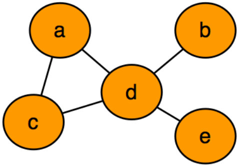


Spanning trees

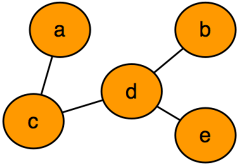
Definition

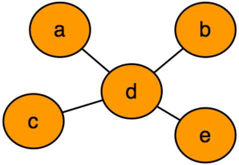
A subgraph T of a undirected graph $G = (V, E)$ is a spanning tree of G if it is a tree and contains every vertex of G.

Given the following graph:



The spanning trees are:



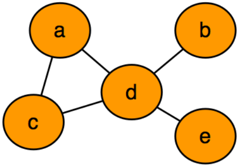


Spanning trees

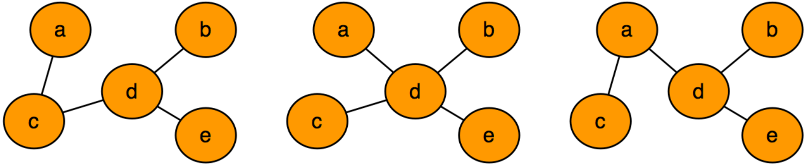
Definition

A subgraph T of a undirected graph $G = (V, E)$ is a spanning tree of G if it is a tree and contains every vertex of G.

Given the following graph:

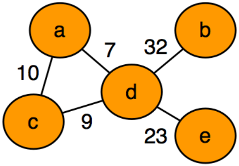


The spanning trees are:



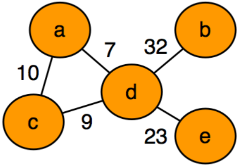
Spanning trees of weighted graphs

Given the following graph:

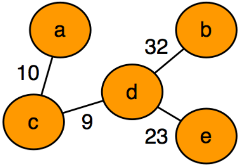


Spanning trees of weighted graphs

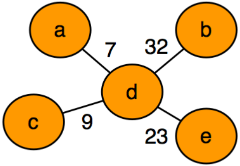
Given the following graph:



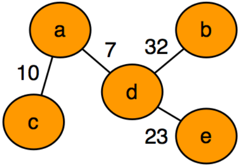
The spanning trees (with associated total weights) are:



$\sum_{e \in T} w(e) = 74$



$\sum_{e \in T} w(e) = 71$



$\sum_{e \in T} w(e) = 72$

Minimum-cost Spanning Tree (MST)

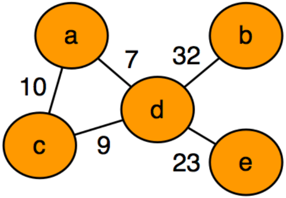
Definition

The minimum-cost spanning tree of a graph A spanning tree T of a undirected graph $G = (V, E)$ is a minimum-cost spanning tree of G if the total weight $w(T) = \sum_{(u,v) \in T} w(u, v)$ is minimized.

Minimum-cost Spanning Tree (MST)

Definition

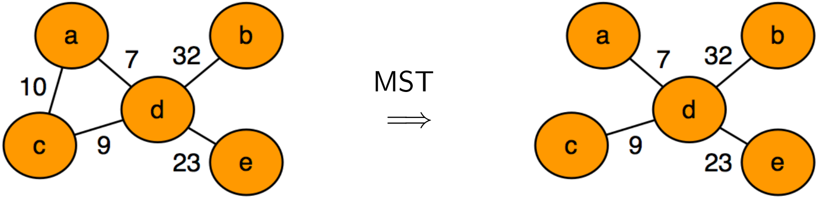
The minimum-cost spanning tree of a graph A spanning tree T of a undirected graph $G = (V, E)$ is a minimum-cost spanning tree of G if the total weight $w(T) = \sum_{(u,v)\in T} w(u, v)$ is minimized.



Minimum-cost Spanning Tree (MST)

Definition

The minimum-cost spanning tree of a graph A spanning tree T of a undirected graph $G = (V, E)$ is a minimum-cost spanning tree of G if the total weight $w(T) = \sum_{(u,v) \in T} w(u, v)$ is minimized.



MST $\Longrightarrow$

Minimum-cost Spanning Tree (continued)

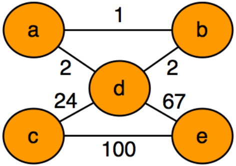
Uniqueness of MST

The minimum-cost spanning tree may not be unique!

Minimum-cost Spanning Tree (continued)

Uniqueness of MST

The minimum-cost spanning tree may not be unique!

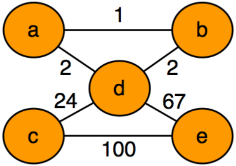


Graph G

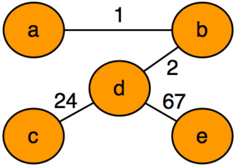
Minimum-cost Spanning Tree (continued)

Uniqueness of MST

The minimum-cost spanning tree may not be unique!



Graph G

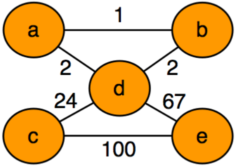


$\sum w_T = 94$

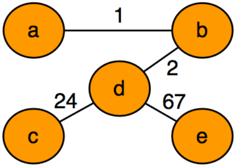
Minimum-cost Spanning Tree (continued)

Uniqueness of MST

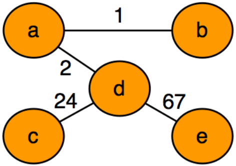
The minimum-cost spanning tree may not be unique!



Graph G



$\sum w_T = 94$

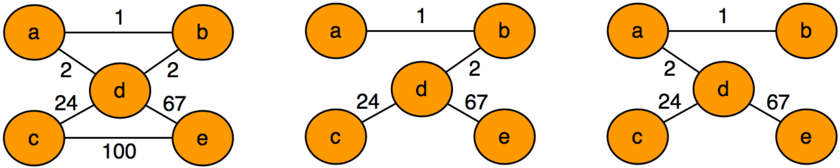


$\sum w_T = 94$

Minimum-cost Spanning Tree (continued)

Uniqueness of MST

The minimum-cost spanning tree may not be unique!



Graph G $\quad \sum w_T = 94$ $\quad \sum w_T = 94$

Key observation

However, if the weights are all distinct (i.e., $w(u_i, v_j) \neq w(u_k, v_l)$ unless $i = k$ and $j = l$), then it is indeed unique.

Computing an MST

- We grow the tree one edge at a time, starting with a graph $G' = (V, \emptyset)$.
- At each step, add a new safe edge, ensuring that it does not create a cycle (why?).
- If adding an edge guarantees that the tree after each step is a subset of some MST, then the final result will be an MST.

Computing an MST

- We grow the tree one edge at a time, starting with a graph $G' = (V, \emptyset)$.
- At each step, add a new safe edge, ensuring that it does not create a cycle (why?).
- If adding an edge guarantees that the tree after each step is a subset of some MST, then the final result will be an MST.

Key question

How do we pick the next safe edge?

Computing an MST

- We grow the tree one edge at a time, starting with a graph $G' = (V, \emptyset)$.
- At each step, add a new safe edge, ensuring that it does not create a cycle (why?).
- If adding an edge guarantees that the tree after each step is a subset of some MST, then the final result will be an MST.

Key question

How do we pick the next safe edge?
Which algorithm design strategy does this question remind you of?

Prim's algorithm to compute an MST

$\text{MST-PRIM}(G, w, r)$

for each $u \in V[G]$
    do $key[u] \leftarrow \infty$
        $\pi[u] \leftarrow \text{NIL}$
$key[r] \leftarrow 0$
$Q \leftarrow V[G]$
while $Q \neq \emptyset$
    do $u \leftarrow \text{EXTRACT-MIN}(Q)$
        for each $v \in Adj[u]$
            do if $v \in Q$ and $w(u, v) < key[v]$
                then $\pi[v] \leftarrow u$
                    $key[v] \leftarrow w(u, v)$

Prim's algorithm to compute an MST

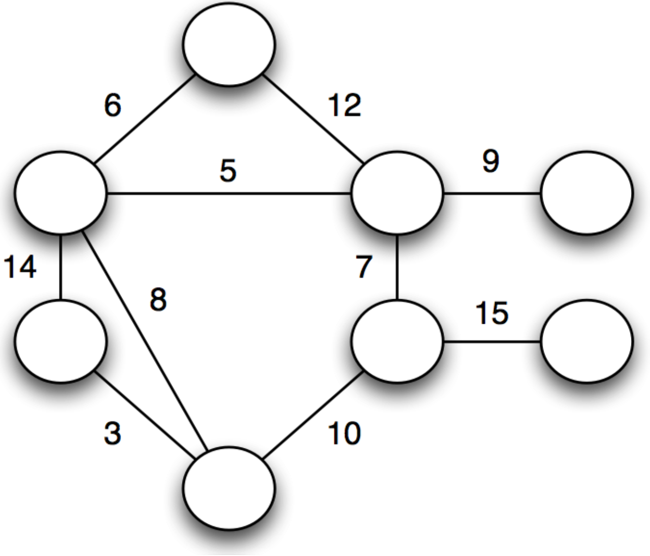
$\text{MST-PRIM}(G, w, r)$

```
for each u ∈ V[G]
        do key[u] ← ∞
            π[u] ← NIL
key[r] ← 0
Q ← V[G]
while Q ≠ ∅
        do u ← EXTRACT-MIN(Q)
            for each v ∈ Adj[u]
                  do if v ∈ Q and w(u, v) < key[v]
                        then π[v] ← u
                              key[v] ← w(u, v)
```

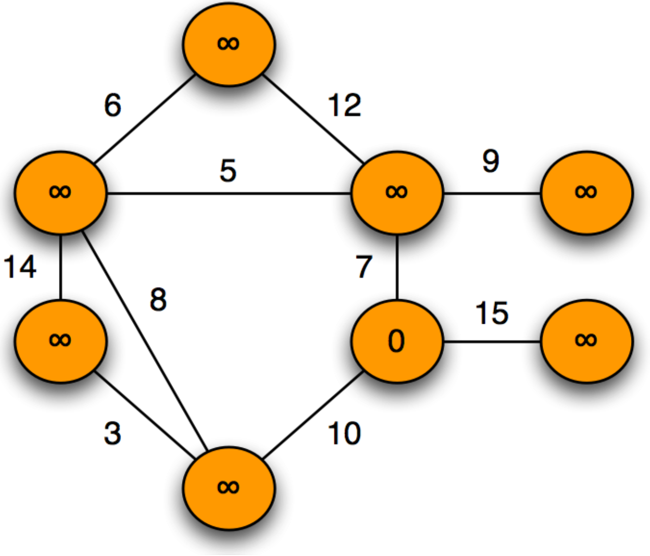
Running time

$O(V \lg V + E \lg V) = O(E \lg V)$

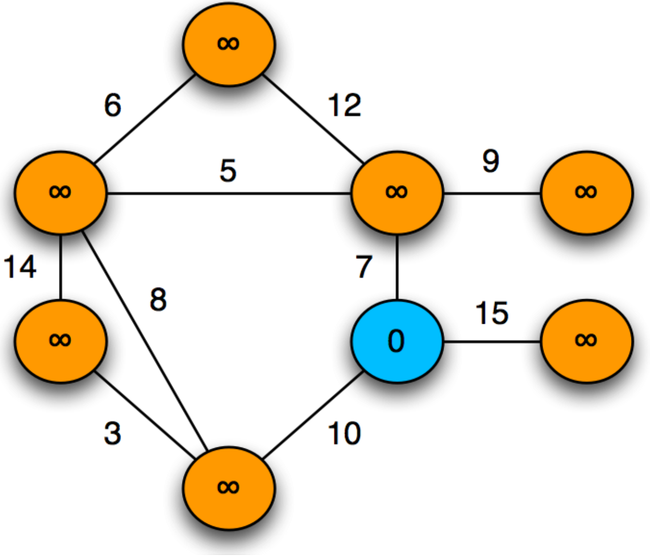
Prim's algorithm in action



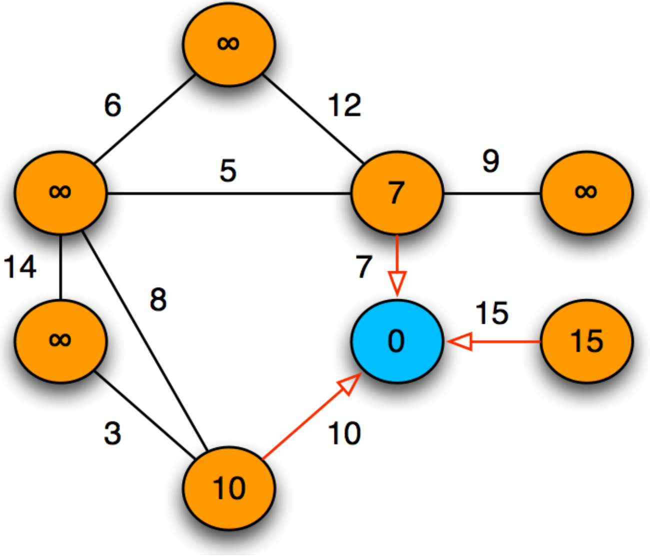
Prim's algorithm in action



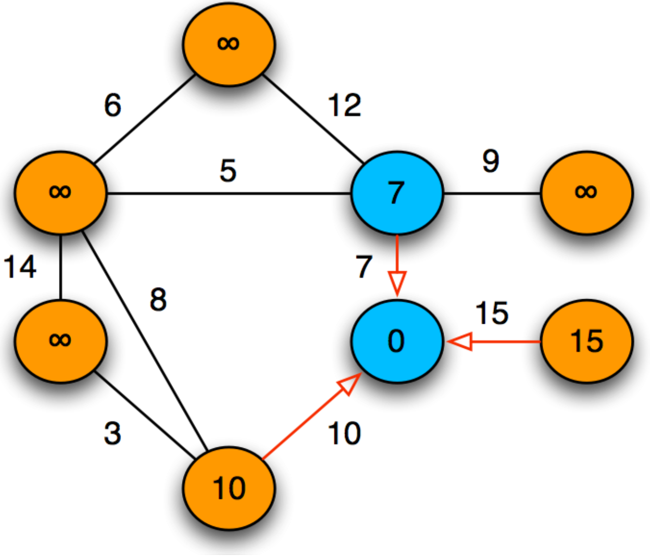
Prim's algorithm in action



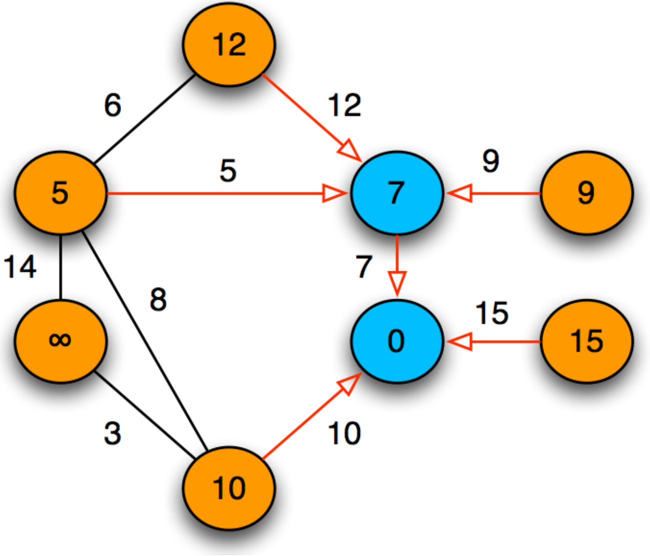
Prim's algorithm in action



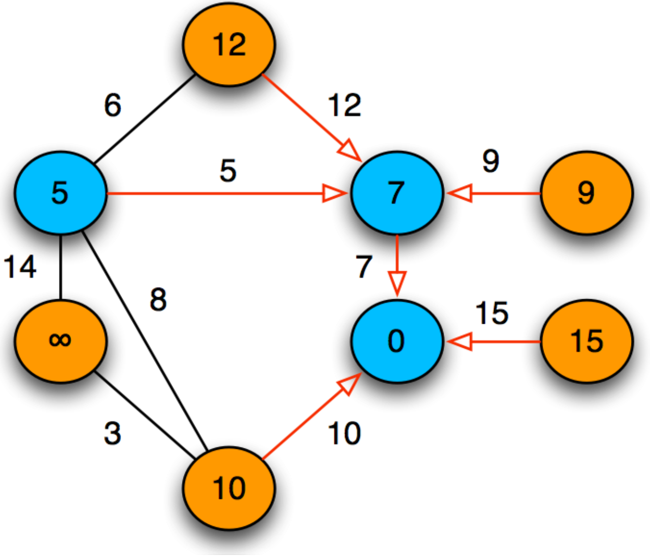
Prim's algorithm in action



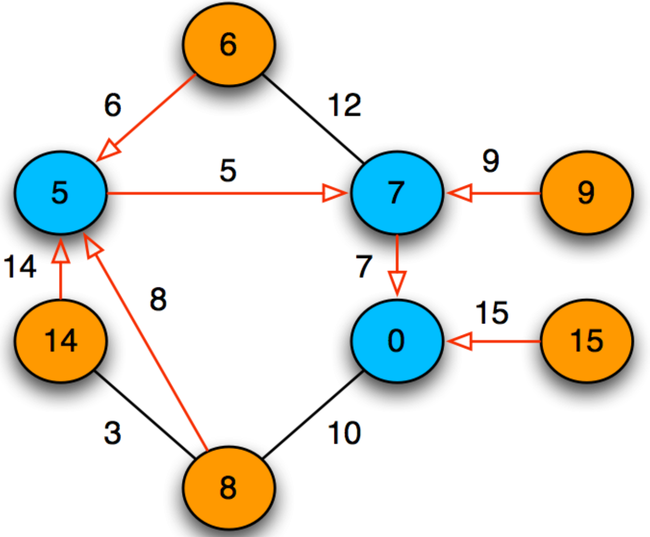
Prim's algorithm in action



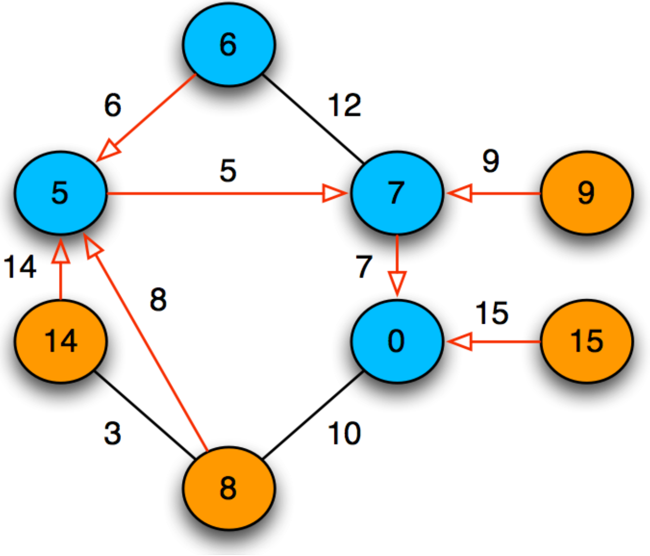
Prim's algorithm in action



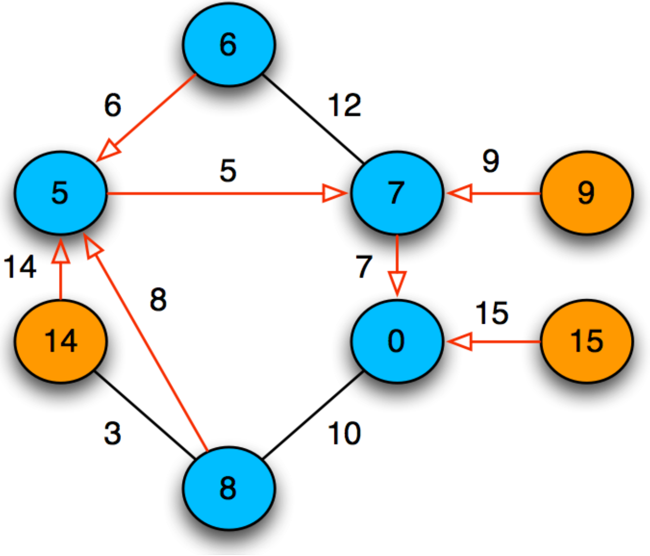
Prim's algorithm in action



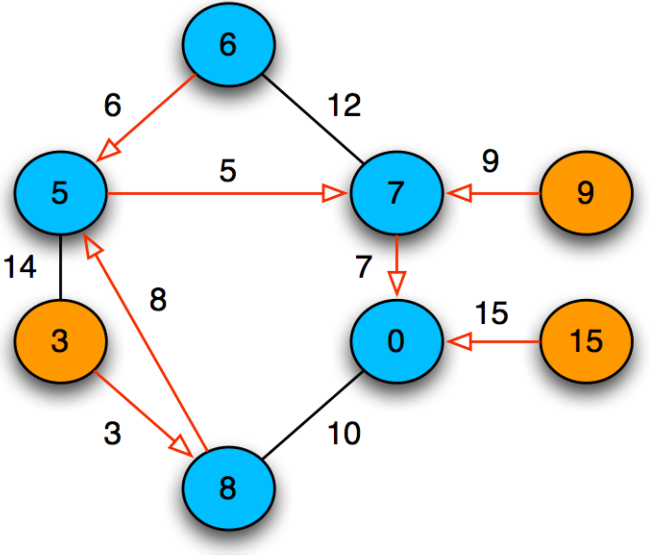
Prim's algorithm in action



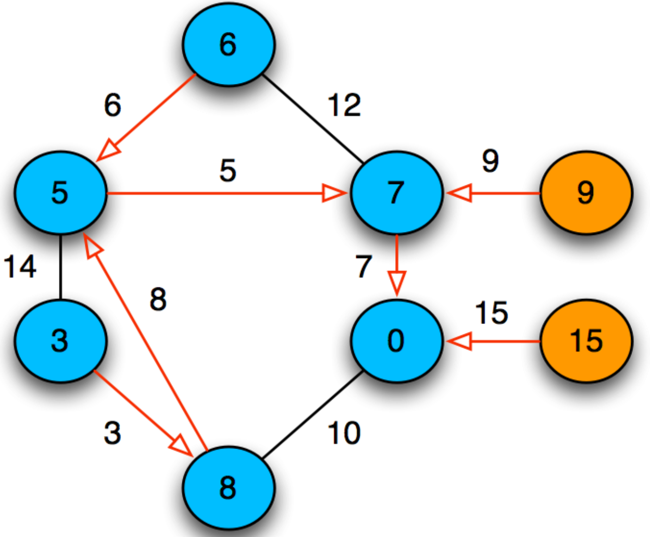
Prim's algorithm in action



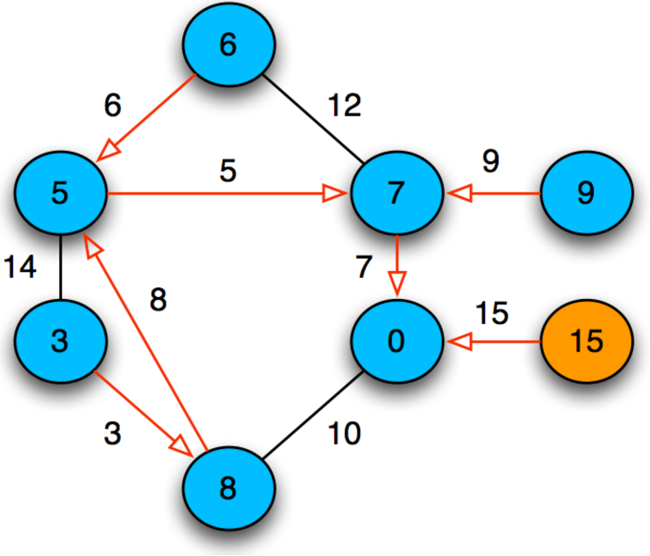
Prim's algorithm in action



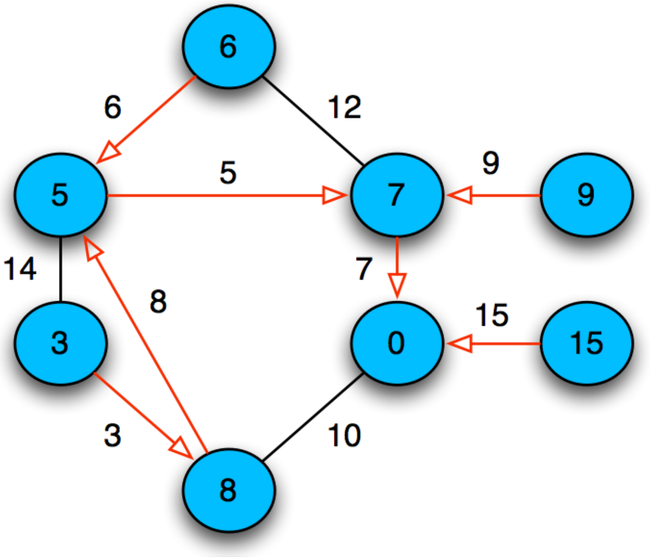
Prim's algorithm in action



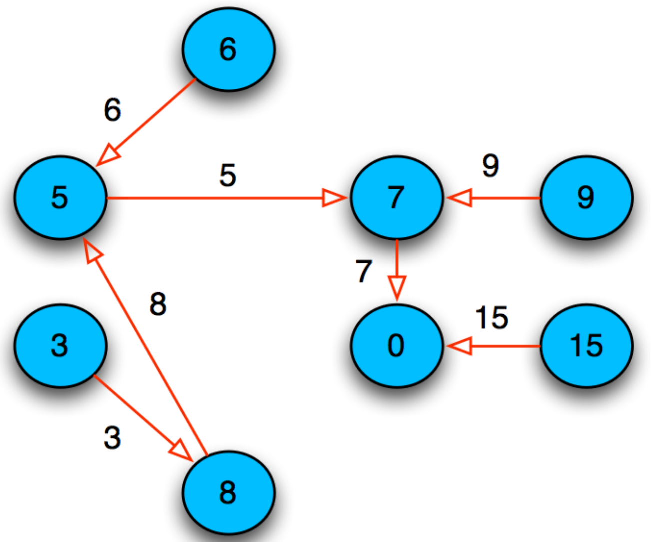
Prim's algorithm in action



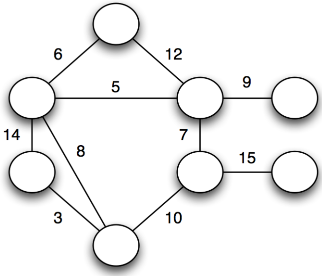
Prim's algorithm in action



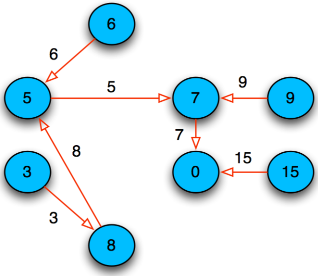
Prim's algorithm in action



Prim's algorithm in action



$\Longrightarrow$



Kruskal's algorithm to compute an MST

```
MST-KRUSKAL(G, w)
A ← ∅
for each vertex v ∈ V[G]
        do MAKE-SET(v)
sort the edges of E into non-decreasing order by weight w
for each edge (u, v) ∈ E, taken in non-decreasing order by weight
        do if FIND-SET(u) ≠ FIND-SET(v)
              then A ← A ∪ {(u, v)}
                   UNION(u, v)
return A
```

Kruskal's algorithm to compute an MST

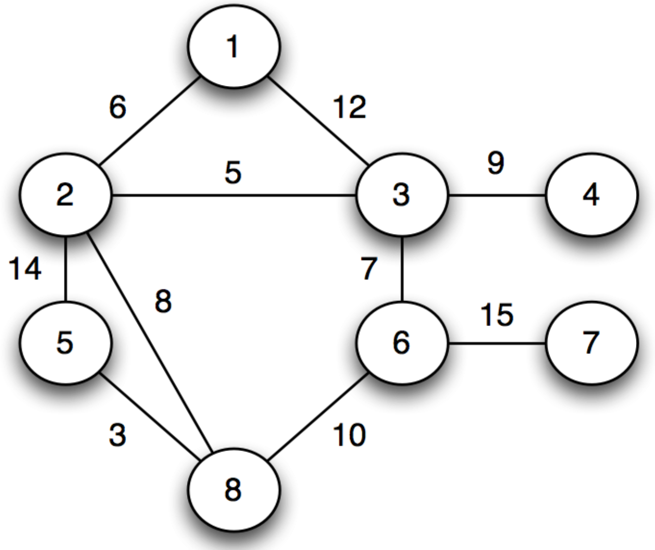
MST-KRUSKAL(G, w)

$A \leftarrow \emptyset$
for each vertex $v \in V[G]$
    do MAKE-SET(v)
sort the edges of E into non-decreasing order by weight w
for each edge $(u, v) \in E$, taken in non-decreasing order by weight
    do if FIND-SET$(u) \neq$ FIND-SET(v)
        then $A \leftarrow A \cup \{(u, v)\}$
            UNION(u, v)
return A

Running time

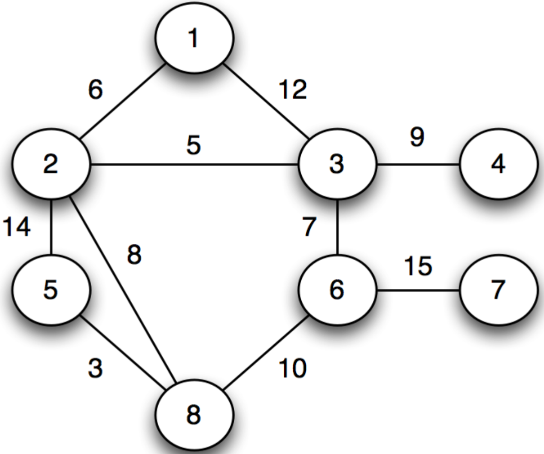
$O(E \lg E)$

Kruskal's algorithm in action



Kruskal's algorithm in action

$w(u,v)$	(u,v)
3	(5,8)
5	(2,3)
6	(1,2)
7	(3,6)
8	(2,8)
9	(3,4)
10	(6,8)
12	(1,3)
14	(2,5)
15	(6,7)



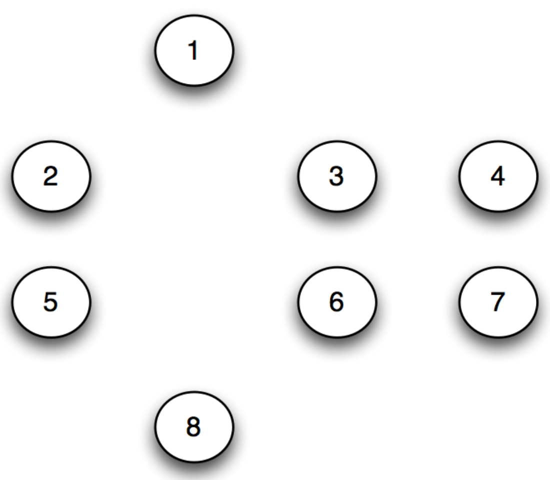
$|V| = 8$

$|E| = 10$

$|T| = 0$

Kruskal's algorithm in action

$w(u,v)$	(u,v)
3	(5,8)
5	(2,3)
6	(1,2)
7	(3,6)
8	(2,8)
9	(3,4)
10	(6,8)
12	(1,3)
14	(2,5)
15	(6,7)



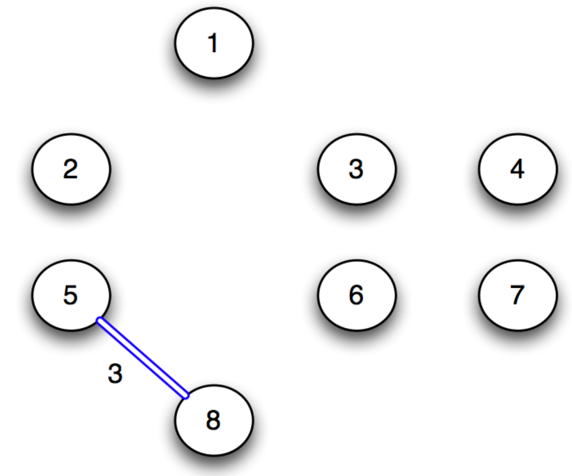
$$|V| = 8$$
$$|E| = 10$$
$$|T| = 0$$

Vertex sets:

$\{1\},\{2\},\{3\},\{4\},\{5\},\{6\},\{7\},\{8\}$

Kruskal's algorithm in action

$w(u, v)$	(u, v)
3	(5,8)
5	(2,3)
6	(1,2)
7	(3,6)
8	(2,8)
9	(3,4)
10	(6,8)
12	(1,3)
14	(2,5)
15	(6,7)



$|V| = 8$

$|E| = 10$

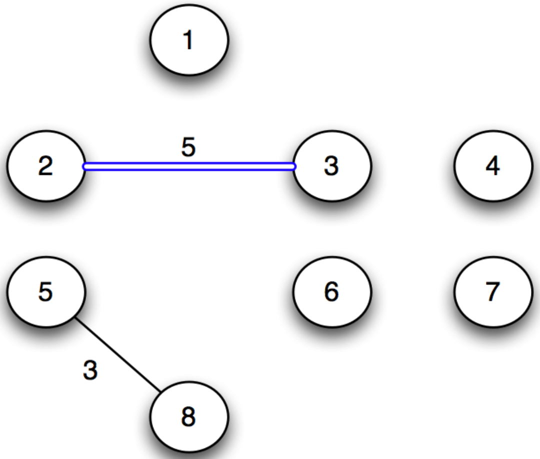
$|T| = \boxed{1}$

Vertex sets:

$\{1\},\{2\},\{3\},\{4\},\{5\},\{6\},\{7\},\{8\} \Longrightarrow \{1\},\{2\},\{3\},\{4\},\boxed{\{5, 8\}},\{6\},\{7\}$

Kruskal's algorithm in action

$w(u, v)$	(u, v)
✓ 3	(5,8)
5	(2,3)
6	(1,2)
7	(3,6)
8	(2,8)
9	(3,4)
10	(6,8)
12	(1,3)
14	(2,5)
15	(6,7)



$|V| = 8$

$|E| = 10$

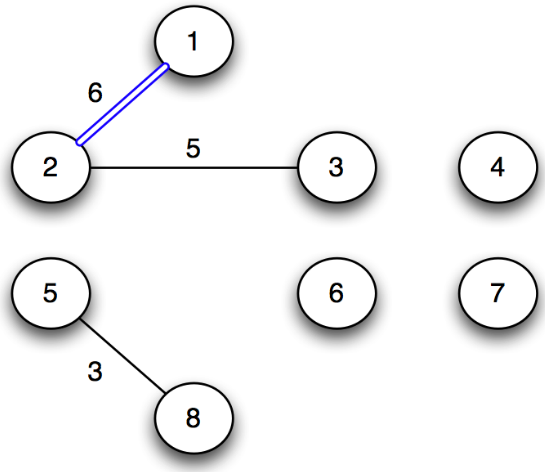
$|T| = \boxed{2}$

Vertex sets:

$\{1\},\{2\},\{3\},\{4\},\{5, 8\},\{6\},\{7\} \Longrightarrow \{1\},\boxed{\{2, 3\}},\{4\},\{5, 8\},\{6\},\{7\}$

Kruskal's algorithm in action

$w(u,v)$	(u,v)
✓ 3	(5,8)
✓ 5	(2,3)
6	(1,2)
7	(3,6)
8	(2,8)
9	(3,4)
10	(6,8)
12	(1,3)
14	(2,5)
15	(6,7)



$|V| = 8$

$|E| = 10$

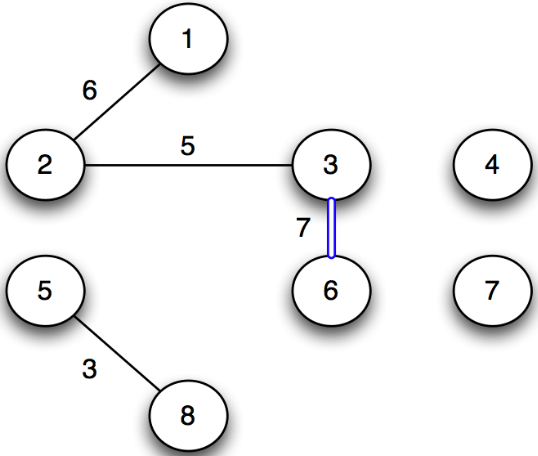
$|T| = \boxed{3}$

Vertex sets:

$\{1\},\{2,3\},\{4\},\{5,8\},\{6\},\{7\} \Longrightarrow \boxed{\{1,2,3\}},\{4\},\{5,8\},\{6\},\{7\}$

Kruskal's algorithm in action

$w(u,v)$	(u,v)
✓ 3	(5,8)
✓ 5	(2,3)
✓ 6	(1,2)
7	(3,6)
8	(2,8)
9	(3,4)
10	(6,8)
12	(1,3)
14	(2,5)
15	(6,7)



$|V| = 8$

$|E| = 10$

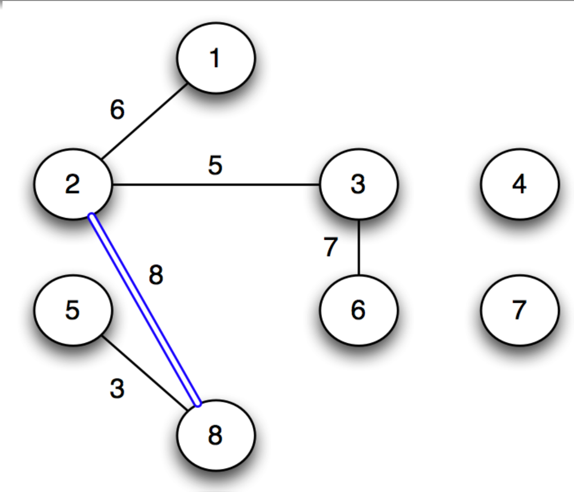
$|T| = \boxed{4}$

Vertex sets:

$\{1, 2, 3\}, \{4\}, \{5, 8\}, \{6\}, \{7\} \Longrightarrow \boxed{\{1, 2, 3, 6\}}, \{4\}, \{5, 8\}, \{7\}$

Kruskal's algorithm in action

$w(u,v)$	(u,v)
✓ 3	(5,8)
✓ 5	(2,3)
✓ 6	(1,2)
✓ 7	(3,6)
8	(2,8)
9	(3,4)
10	(6,8)
12	(1,3)
14	(2,5)
15	(6,7)



$|V| = 8$

$|E| = 10$

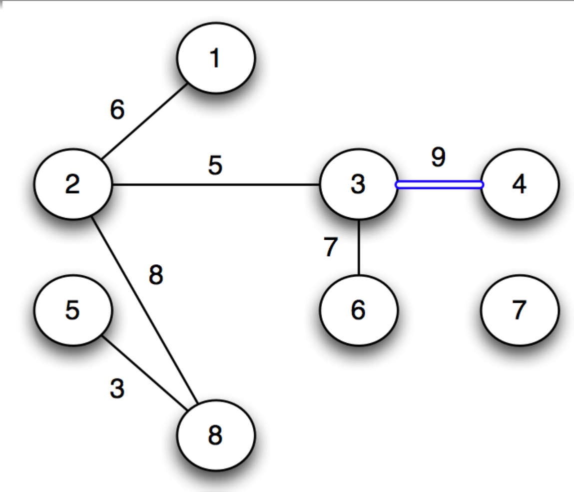
$|T| = \boxed{5}$

Vertex sets:

$\{1, 2, 3, 6\}, \{4\}, \{5, 8\}, \{7\} \Longrightarrow \boxed{\{1, 2, 3, 5, 6, 8\}}, \{4\}, \{7\}$

Kruskal's algorithm in action

$w(u,v)$	(u,v)
✓ 3	(5,8)
✓ 5	(2,3)
✓ 6	(1,2)
✓ 7	(3,6)
✓ 8	(2,8)
9	(3,4)
10	(6,8)
12	(1,3)
14	(2,5)
15	(6,7)



$|V| = 8$

$|E| = 10$

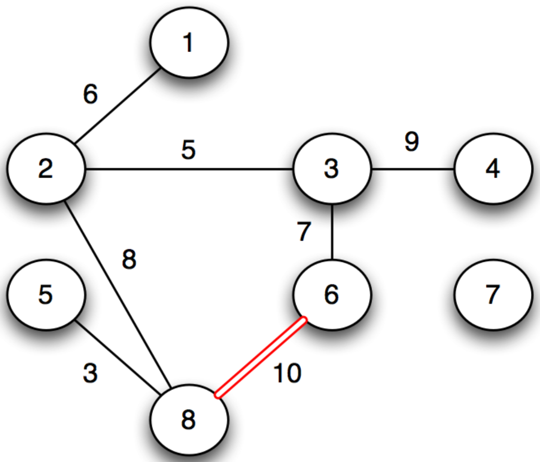
$|T| = \boxed{6}$

Vertex sets:

$\{1, 2, 3, 5, 6, 8\}, \{4\}, \{7\} \Longrightarrow \boxed{\{1, 2, 3, 4, 5, 6, 8\}}, \{7\}$

Kruskal's algorithm in action

$w(u,v)$	(u,v)
✓ 3	(5,8)
✓ 5	(2,3)
✓ 6	(1,2)
✓ 7	(3,6)
✓ 8	(2,8)
✓ 9	(3,4)
10	(6,8)
12	(1,3)
14	(2,5)
15	(6,7)



$$|V| = 8$$

$$|E| = 10$$

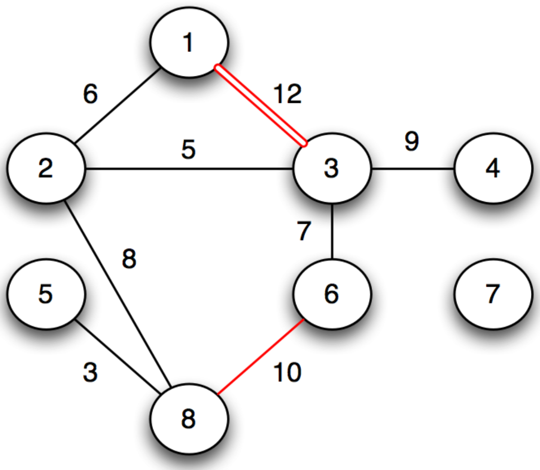
$$|T| = \boxed{6}$$

Vertex sets:

$\{1, 2, 3, 4, 5, 6, 8\},\{7\} \Longrightarrow \{1, 2, 3, 4, 5, 6, 8\},\{7\}$

Kruskal's algorithm in action

$w(u,v)$	(u,v)
✓ 3	(5,8)
✓ 5	(2,3)
✓ 6	(1,2)
✓ 7	(3,6)
✓ 8	(2,8)
✓ 9	(3,4)
× 10	(6,8)
12	(1,3)
14	(2,5)
15	(6,7)



$|V| = 8$

$|E| = 10$

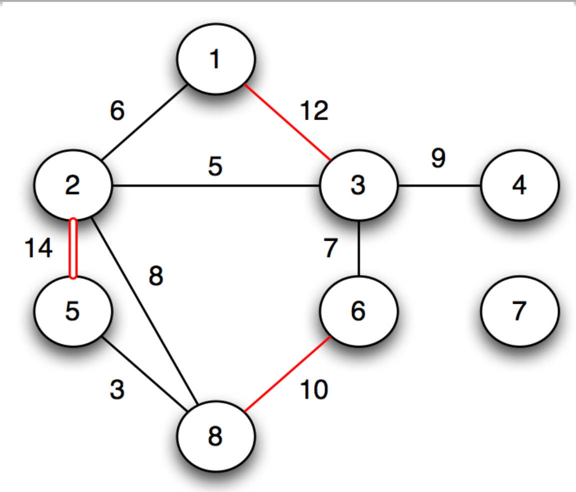
$|T| = \boxed{6}$

Vertex sets:

$\{1, 2, 3, 4, 5, 6, 8\}, \{7\} \implies \{1, 2, 3, 4, 5, 6, 8\}, \{7\}$

Kruskal's algorithm in action

$w(u,v)$	(u,v)
✓ 3	(5,8)
✓ 5	(2,3)
✓ 6	(1,2)
✓ 7	(3,6)
✓ 8	(2,8)
✓ 9	(3,4)
× 10	(6,8)
× 12	(1,3)
14	(2,5)
15	(6,7)



$|V| = 8$

$|E| = 10$

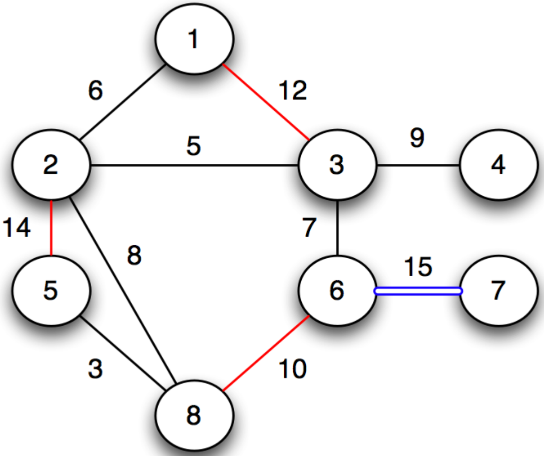
$|T| = \boxed{6}$

Vertex sets:

$\{1, 2, 3, 4, 5, 6, 8\}, \{7\} \Longrightarrow \{1, 2, 3, 4, 5, 6, 8\}, \{7\}$

Kruskal's algorithm in action

$w(u,v)$	(u,v)
✓ 3	(5,8)
✓ 5	(2,3)
✓ 6	(1,2)
✓ 7	(3,6)
✓ 8	(2,8)
✓ 9	(3,4)
× 10	(6,8)
× 12	(1,3)
× 14	(2,5)
15	(6,7)



$$|V| = 8$$

$$|E| = 10$$

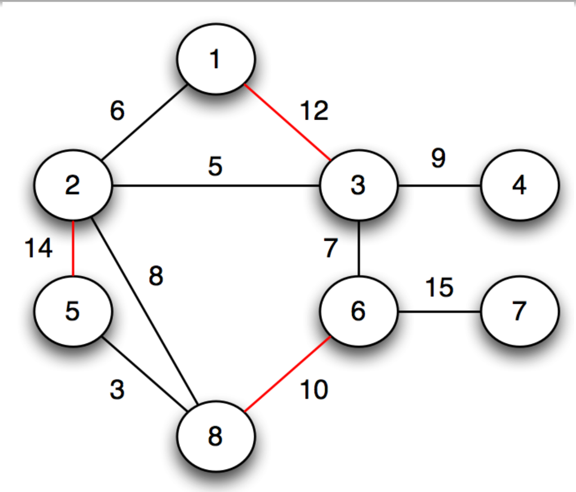
$$|T| = \boxed{7}$$

Vertex sets:

$\{1, 2, 3, 4, 5, 6, 8\}, \{7\} \Longrightarrow \boxed{\{1, 2, 3, 4, 5, 6, 7, 8\}}$

Kruskal's algorithm in action

$w(u,v)$	(u,v)
✓ 3	(5,8)
✓ 5	(2,3)
✓ 6	(1,2)
✓ 7	(3,6)
✓ 8	(2,8)
✓ 9	(3,4)
× 10	(6,8)
× 12	(1,3)
× 14	(2,5)
✓ 15	(6,7)



$|V| = 8$

$|E| = 10$

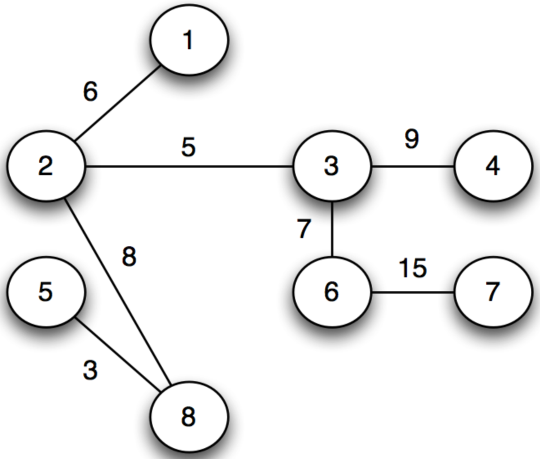
$|T| = \boxed{7}$

Vertex sets:

$\{1, 2, 3, 4, 5, 6, 7, 8\}$

Kruskal's algorithm in action

$w(u,v)$	(u,v)
✓ 3	(5,8)
✓ 5	(2,3)
✓ 6	(1,2)
✓ 7	(3,6)
✓ 8	(2,8)
✓ 9	(3,4)
× 10	(6,8)
× 12	(1,3)
× 14	(2,5)
✓ 15	(6,7)



$|V| = 8$

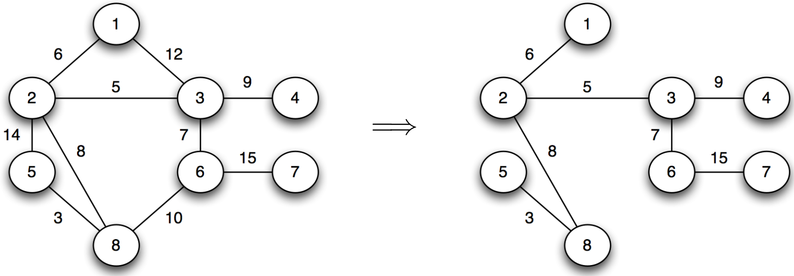
$|E| = 10$

$|T| = \boxed{7}$

Vertex sets:

$\{1, 2, 3, 4, 5, 6, 7, 8\}$

Kruskal's algorithm in action



Contents

Dijkstra's algorithm for SSSP

$\textsc{Dijkstra}(G, s)$

for each $v \in V[G]$
    do $d[v] \leftarrow \infty$
        $\pi[v] \leftarrow \mathrm{NIL}$
$d[s] \leftarrow 0$
$S \leftarrow \emptyset$
$Q \leftarrow V[G]$
while $Q \neq \emptyset$
    do $u \leftarrow \textsc{extract-min}(Q)$
        $S \leftarrow S \cup \{u\}$
        for each vertex $v \in Adj[u]$
            do if $d[v] > d[u] + w(u, v)$
                then $d[v] \leftarrow d[u] + w(u, v)$
                      $\pi[v] \leftarrow u$

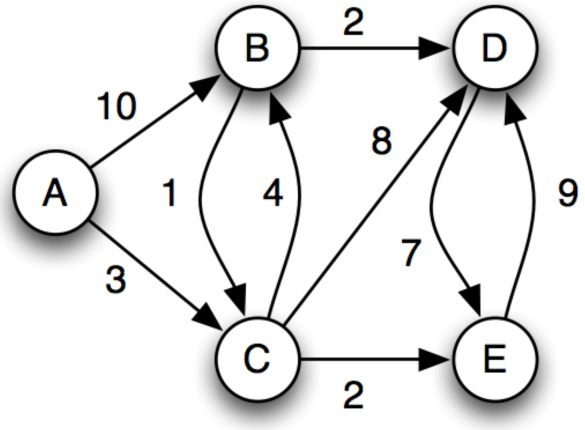
Dijkstra's algorithm for SSSP

```
DIJKSTRA(G, s)
for each v ∈ V[G]
        do d[v] ← ∞
           π[v] ← NIL
d[s] ← 0
S ← ∅
Q ← V[G]
while Q ≠ ∅
        do u ← EXTRACT-MIN(Q)
           S ← S ∪ {u}
           for each vertex v ∈ Adj[u]
                do if d[v] > d[u] + w(u, v)
                      then d[v] ← d[u] + w(u, v)
                           π[v] ← u
```

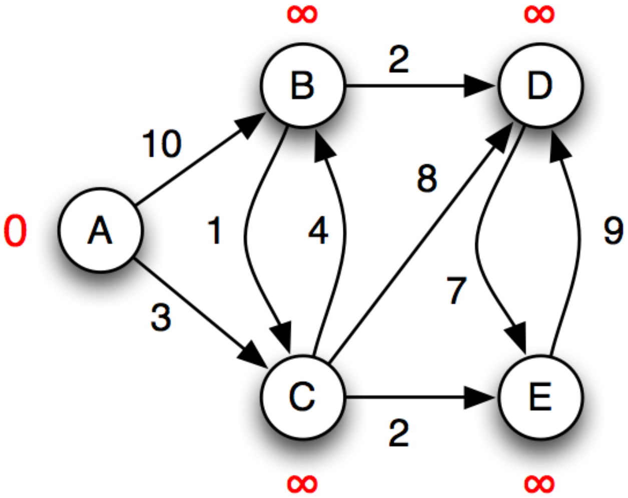
Running time

$O((V + E)\lg V)$

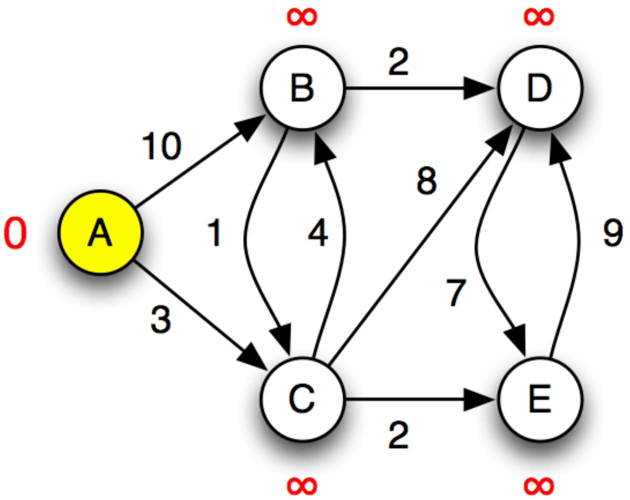
Dijkstra's SSSP algorithm in action



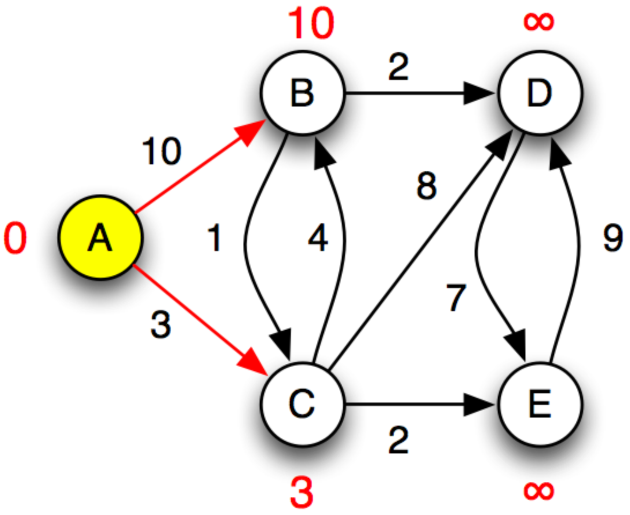
Dijkstra's SSSP algorithm in action



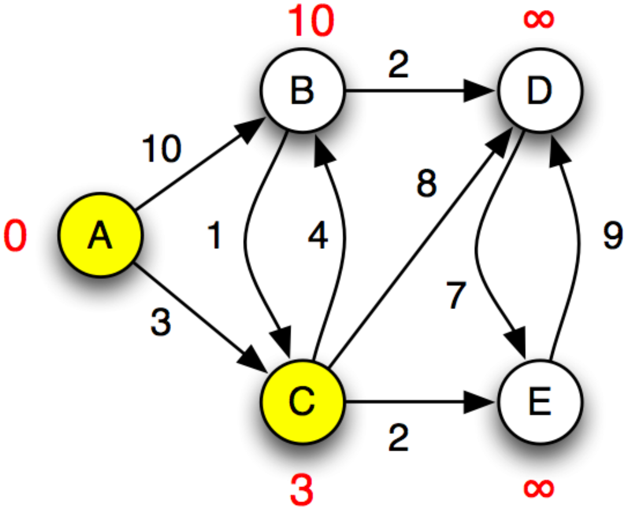
Dijkstra's SSSP algorithm in action



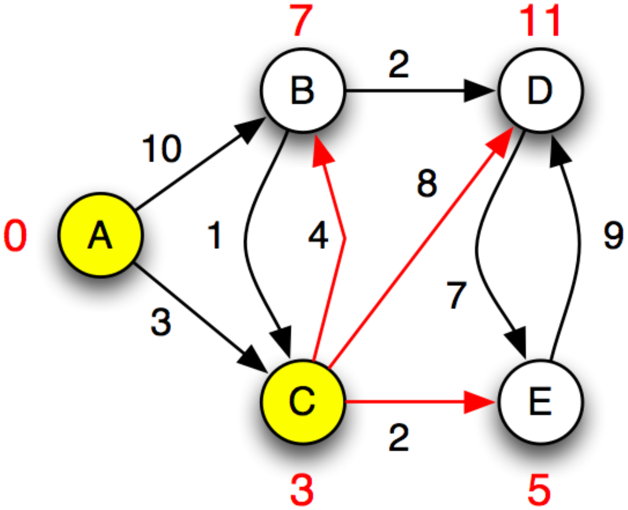
Dijkstra's SSSP algorithm in action



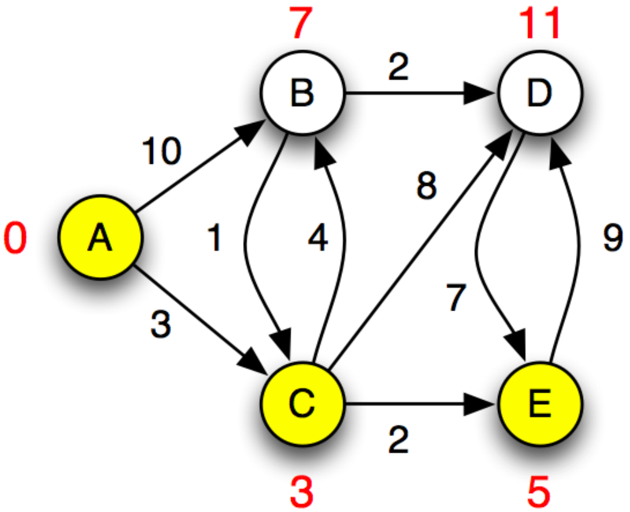
Dijkstra's SSSP algorithm in action



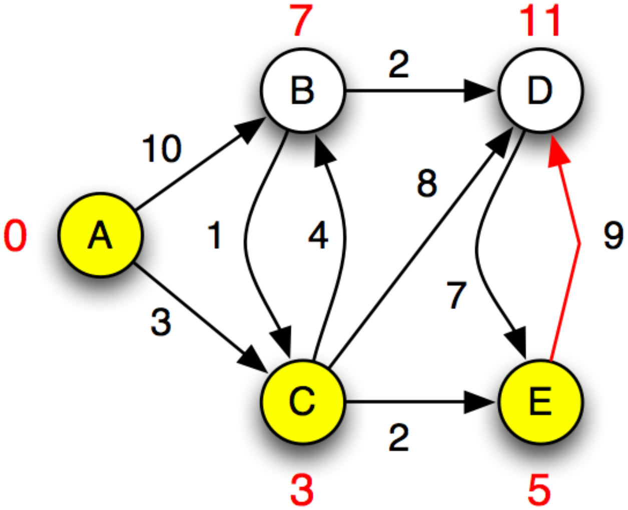
Dijkstra's SSSP algorithm in action



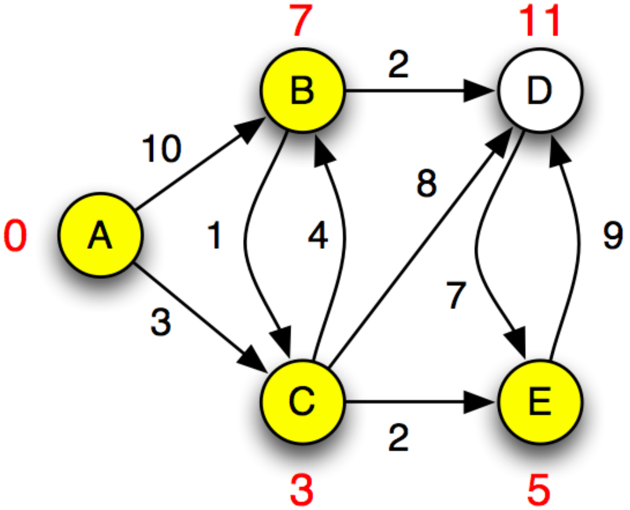
Dijkstra's SSSP algorithm in action



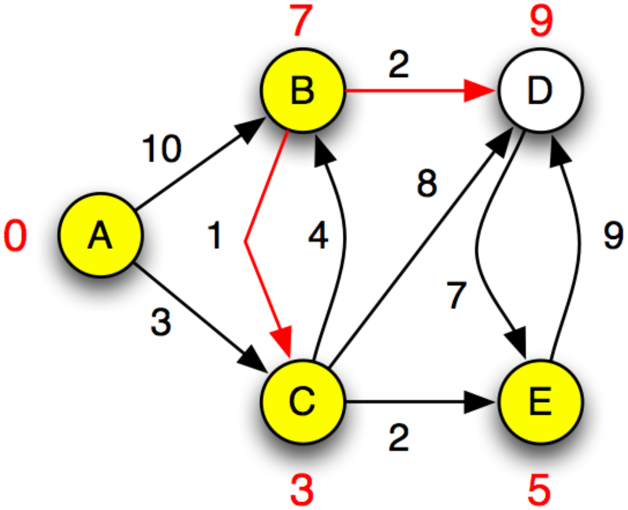
Dijkstra's SSSP algorithm in action



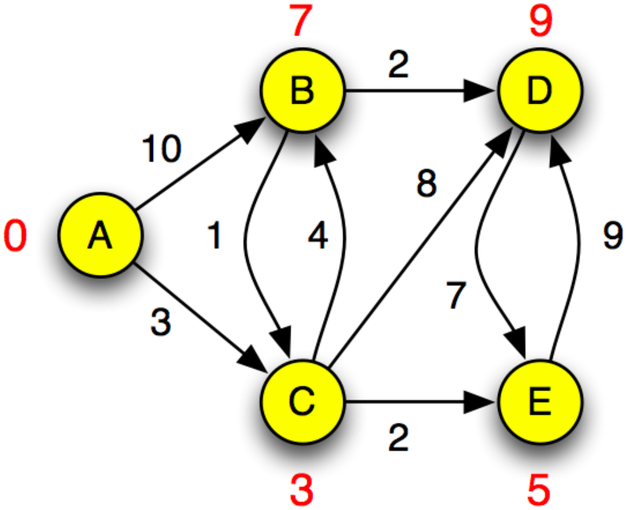
Dijkstra's SSSP algorithm in action



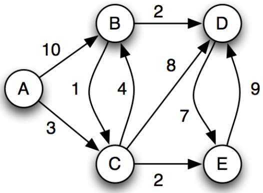
Dijkstra's SSSP algorithm in action



Dijkstra's SSSP algorithm in action



Dijkstra's SSSP algorithm in action



$\Longrightarrow$