*Algorithms and Data Structures Part 4*

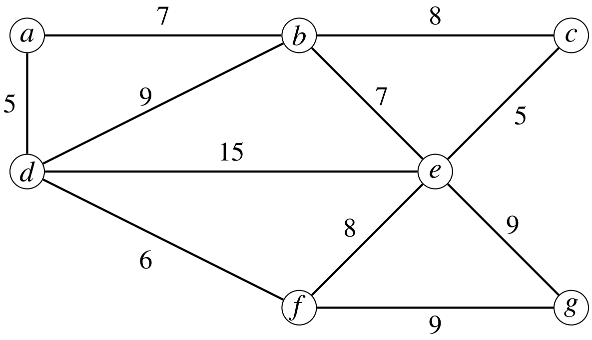
# Lecture 9: Minimum Spanning Trees

George Mertzios

george.mertzios@durham.ac.uk

# Connecting the vertices

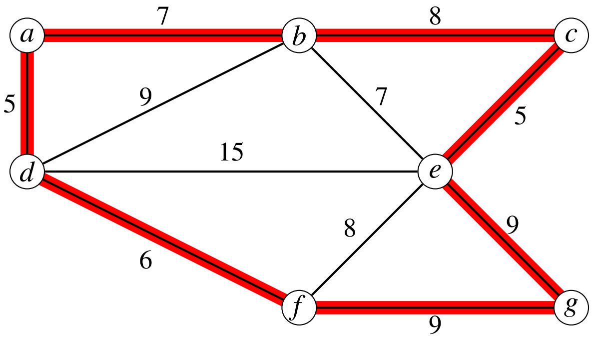
Input: a graph $G = (V, E)$ with a weight (or a cost) $w(u, v)$ for each edge $(u, v)$.




Objective: Choose a subset of the edges that connects the vertices.
Find the solution that costs the least.

# Connecting the vertices

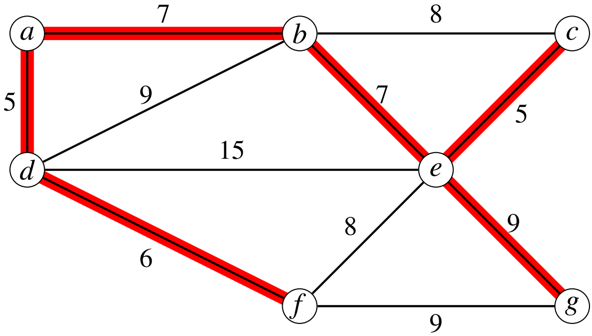
Input: a graph $G = (V, E)$ with a weight (or a cost) $w(u, v)$ for each edge $(u, v)$.




Objective: Choose a subset of the edges that connects the vertices.
Find the solution that costs the least.

# Connecting the vertices

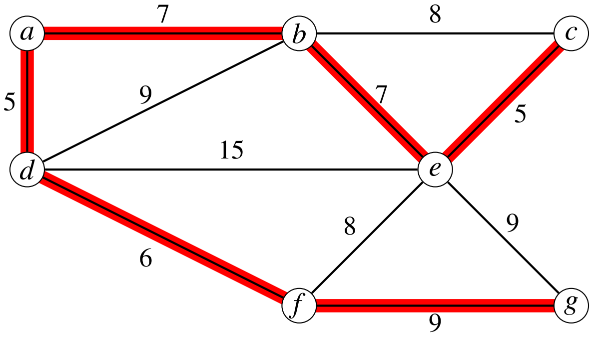
Input: a graph $G = (V, E)$ with a weight (or a cost) $w(u, v)$ for each edge $(u, v)$.




Objective: Choose a subset of the edges that connects the vertices.
Find the solution that costs the least.

# Connecting the vertices

Input: a graph $G = (V, E)$ with a weight (or a cost) $w(u, v)$ for each edge $(u, v)$.




Objective: Choose a subset of the edges that connects the vertices.
Find the solution that costs the least.

# Minimum Spanning Tree Problem

Find a tree that spans the vertices and has minimum cost.

# Minimum Spanning Tree Problem

Find a tree that spans the vertices and has minimum cost.

Basic properties of MSTs:

- have $|V| - 1$ edges;
- have no cycles;
- might not be unique.

# Representations of weighted graphs

$$A = \begin{pmatrix} 0 & 5 & 0 & 4 & 0 & 0 & 0 & 0 & 0 \\ 5 & 0 & 10 & 3 & 9 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & 5 & 7 & 0 & 0 & 0 \\ 4 & 3 & 0 & 0 & 8 & 0 & 7 & 0 & 0 \\ 0 & 9 & 5 & 8 & 0 & 7 & 6 & 7 & 0 \\ 0 & 0 & 7 & 0 & 7 & 0 & 0 & 2 & 4 \\ 0 & 0 & 0 & 7 & 6 & 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 & 7 & 2 & 5 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 & 4 & 0 & 3 & 0 \end{pmatrix}$$

# Representations of weighted graphs

Be careful with the adjacency matrix $A$:

- An entry $A(i,j) = 0$ means "the edge $(v_i, v_j)$ does not exist"; this is not the same as " edge $(v_i, v_j)$ has weight 0".

$$A = \begin{pmatrix} 0 & 5 & 0 & 4 & 0 & 0 & 0 & 0 & 0 \\ 5 & 0 & 10 & 3 & 9 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & 5 & 7 & 0 & 0 & 0 \\ 4 & 3 & 0 & 0 & 8 & 0 & 7 & 0 & 0 \\ 0 & 9 & 5 & 8 & 0 & 7 & 6 & 7 & 0 \\ 0 & 0 & 7 & 0 & 7 & 0 & 0 & 2 & 4 \\ 0 & 0 & 0 & 7 & 6 & 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 & 7 & 2 & 5 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 & 4 & 0 & 3 & 0 \end{pmatrix}$$

# Representations of weighted graphs

Be careful with the adjacency matrix $A$:

- An entry $A(i,j) = 0$ means "the edge $(v_i, v_j)$ does not exist"; this is not the same as " edge $(v_i, v_j)$ has weight 0".
- As we are looking for a minimum-cost spanning tree, we could think of "$A(i,j) = 0$" as "$w(i,j) = \infty$"

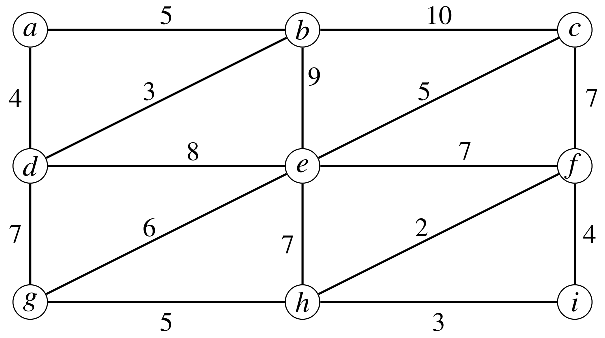
$$A = \begin{pmatrix} 0 & 5 & 0 & 4 & 0 & 0 & 0 & 0 & 0 \\ 5 & 0 & 10 & 3 & 9 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & 5 & 7 & 0 & 0 & 0 \\ 4 & 3 & 0 & 0 & 8 & 0 & 7 & 0 & 0 \\ 0 & 9 & 5 & 8 & 0 & 7 & 6 & 7 & 0 \\ 0 & 0 & 7 & 0 & 7 & 0 & 0 & 2 & 4 \\ 0 & 0 & 0 & 7 & 6 & 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 & 7 & 2 & 5 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 & 4 & 0 & 3 & 0 \end{pmatrix}$$

# Representations of weighted graphs

Be careful with the adjacency matrix $A$:

- An entry $A(i,j) = 0$ means "the edge $(v_i, v_j)$ does not exist"; this is not the same as " edge $(v_i, v_j)$ has weight 0".
- As we are looking for a minimum-cost spanning tree, we could think of "$A(i,j) = 0$" as "$w(i,j) = \infty$"

$$\begin{pmatrix}
\infty & 5 & \infty & 4 & \infty & \infty & \infty & \infty & \infty \\
5 & \infty & 10 & 3 & 9 & \infty & \infty & \infty & \infty \\
\infty & 10 & \infty & \infty & 5 & 7 & \infty & \infty & \infty \\
4 & 3 & \infty & \infty & 8 & \infty & 7 & \infty & \infty \\
\infty & 9 & 5 & 8 & \infty & 7 & 6 & 7 & \infty \\
\infty & \infty & 7 & \infty & 7 & \infty & \infty & 2 & 4 \\
\infty & \infty & \infty & 7 & 6 & \infty & \infty & 5 & \infty \\
\infty & \infty & \infty & \infty & 7 & 2 & 5 & \infty & 3 \\
\infty & \infty & \infty & \infty & \infty & 4 & \infty & 3 & \infty
\end{pmatrix}$$

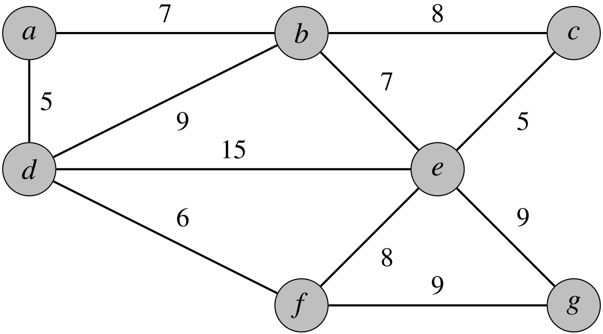

a
b
c
d
e
f
g
h
i
5
10
4
3
9
5
7
8
7
7
6
7
2
4
5
3

# Kruskal's Algorithm

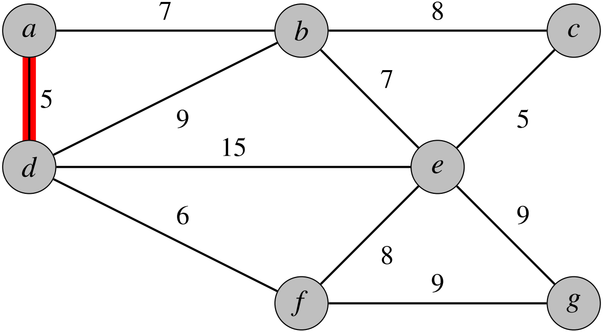
1. Sort the edges by weight.
2. Let $A = \emptyset$.
3. Consider edges in increasing order of weight. For each edge $e$, add $e$ to $A$ unless this would create a cycle.

(Running time is $O(E \log V)$.)

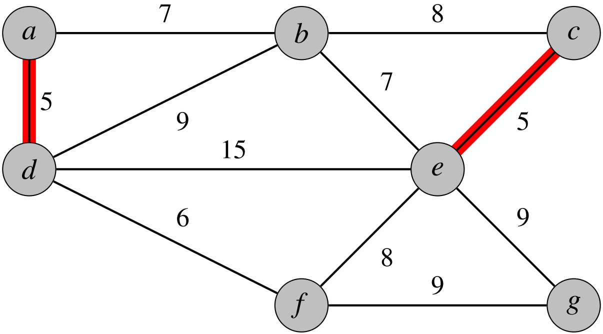
# Kruskal's algorithm



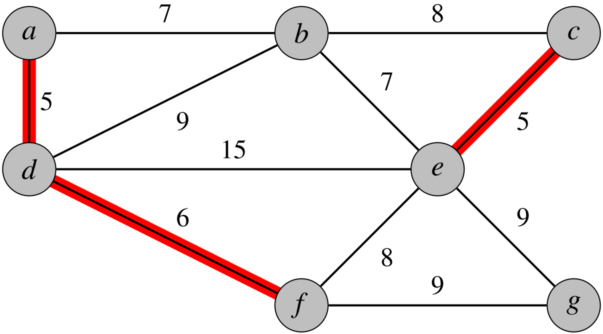
# Kruskal's algorithm



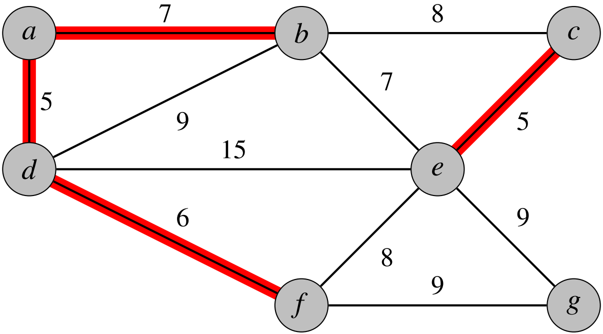
# Kruskal's algorithm



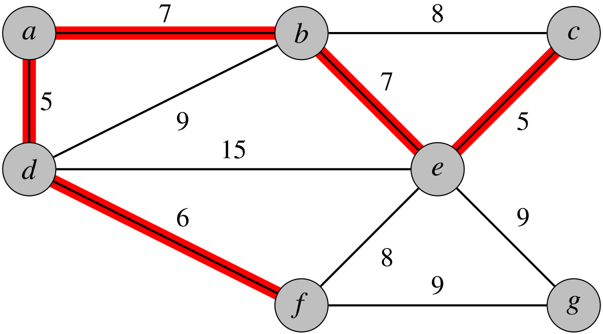
# Kruskal's algorithm



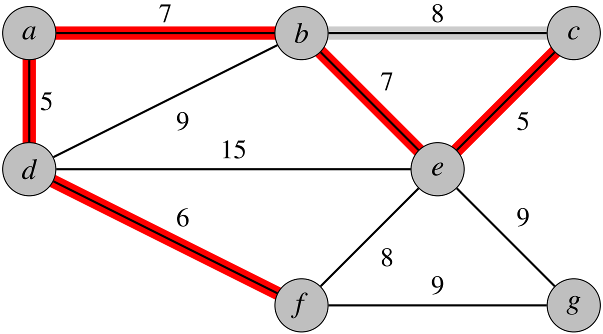
# Kruskal's algorithm



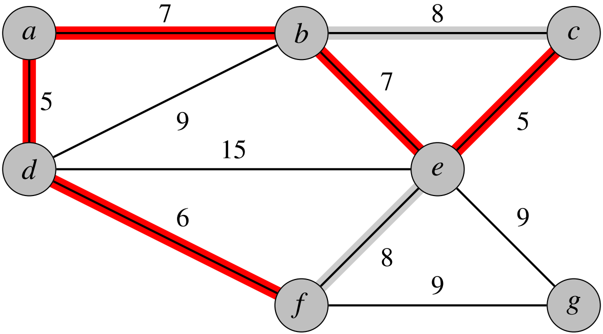
# Kruskal's algorithm



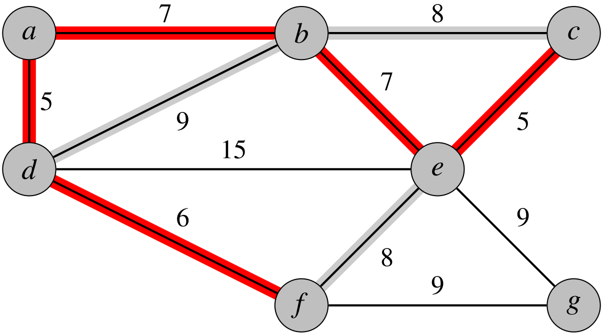
# Kruskal's algorithm



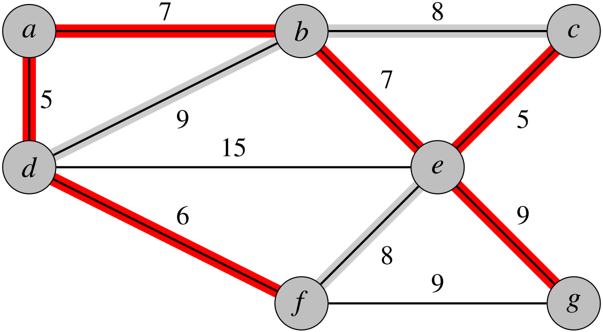
# Kruskal's algorithm



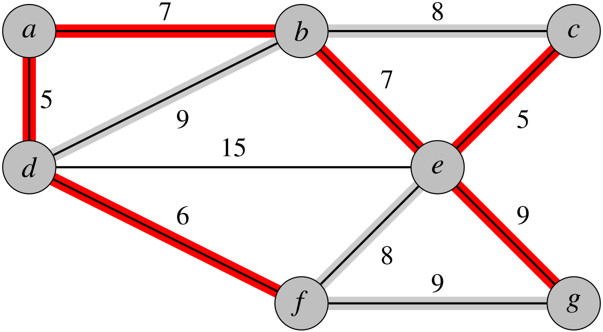
# Kruskal's algorithm



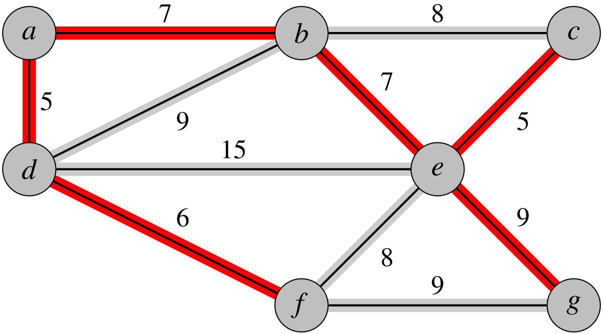
# Kruskal’s algorithm



# Kruskal's algorithm



# Kruskal's algorithm



# Prim's Algorithm

1. Let $U = \{u\}$ where $u$ is some vertex chosen arbitrarily.
2. Let $A = \emptyset$.
3. Until $U$ contains all vertices: find the least-weight edge $e$ that joins a vertex $v$ in $U$ to a vertex $w$ not in $U$ and add $e$ to $A$ and $w$ to $U$.

(Running time is $O(V \log V + E)$.)

# A generic MST Algorithm

Both Kruskal's and Prim's are special cases of a generic (greedy) MST-algorithm:

- We iteratively build a vertex set $A$ such that:
  $A$ is a subset of some minimum spanning tree (MST)

# A generic MST Algorithm

Both Kruskal's and Prim's are special cases of a generic (greedy) MST-algorithm:

- We iteratively build a vertex set $A$ such that:
  $A$ is a subset of some minimum spanning tree (MST)

Some necessary definitions:

- Let $A$ be a subset of an MST. If the set $A \cup \{(u, v)\}$ is also a subset of an MST, then $(u, v)$ is a safe edge for $A$.
- A cut $(S, V - S)$ of $G = (V, E)$ is a partition of $V$.
- An edge $(u, v) \in E$ crosses the cut $(S, V - S)$ if $u \in S$ and $v \in V - S$ (or vice-versa).
- An edge $(u, v) \in E$ is a light edge crossing a cut if its weight is smallest among all edges crossing the cut.
- A cut respects a set $A$ of edges if no edge of $A$ crosses the cut.

# A generic MST Algorithm

Both Kruskal's and Prim's are special cases of a generic (greedy) MST-algorithm:

```
GENERIC-MST(G, w)
A ← Ø
while A does not form a spanning tree
      do find an edge (u, v) that is safe for A
            A ← A ∪ {(u, v)}
return A
```

# A generic MST Algorithm

Both Kruskal's and Prim's are special cases of a generic (greedy) MST-algorithm:

```
GENERIC-MST(G, w)
A ← Ø
while A does not form a spanning tree
    do find an edge (u, v) that is safe for A
        A ← A ∪ {(u, v)}
return A
```

**Theorem**

*Let $G = (V, E)$ be a connected undirected graph with a real-valued weight function $w$ defined on the edges $E$. Let $A \subseteq E$ such that $A$ is included in some MST of $G$, let $(S, V - S)$ be any cut of $G$ that respects $A$, and let $(u, v)$ be a light edge crossing $(S, V - S)$. Then the edge $(u, v)$ is safe for $A$.*

# A generic MST Algorithm

Both Kruskal's and Prim's are special cases of a generic (greedy) MST-algorithm:

```
GENERIC-MST(G, w)
A ← Ø
while A does not form a spanning tree
      do find an edge (u, v) that is safe for A
           A ← A ∪ {(u, v)}
return A
```

**Theorem**

*Let $G = (V, E)$ be a connected undirected graph with a real-valued weight function $w$ defined on the edges $E$. Let $A \subseteq E$ such that $A$ is included in some MST of $G$, let $(S, V - S)$ be any cut of $G$ that respects $A$, and let $(u, v)$ be a light edge crossing $(S, V - S)$. Then the edge $(u, v)$ is safe for $A$.*

- This theorem implies correctness for Prim's algorithm.

# A generic MST Algorithm

## Corollary

*Let $G = (V, E)$ be a connected undirected graph with a real-valued weight function $w$ defined on the edges $E$. Let $A \subseteq E$ such that $A$ is included in some MST of $G$, and let $C = (V_C, E_C)$ be a connected component (tree) in the forest $G_A = (V, A)$. If $(u, v)$ is a light edge that connects $C$ to some other component in $G_A$, then the edge $(u, v)$ is safe for $A$.*

## Proof.

The cut $(V_C, V - V_C)$ respects $A$, and $(u, v)$ is a light edge for this cut. Therefore $(u, v)$ is safe for $A$ by the above theorem. $\square$

# A generic MST Algorithm

## Corollary

*Let $G = (V, E)$ be a connected undirected graph with a real-valued weight function $w$ defined on the edges $E$. Let $A \subseteq E$ such that $A$ is included in some MST of $G$, and let $C = (V_C, E_C)$ be a connected component (tree) in the forest $G_A = (V, A)$. If $(u, v)$ is a light edge that connects $C$ to some other component in $G_A$, then the edge $(u, v)$ is safe for $A$.*

## Proof.

The cut $(V_C, V - V_C)$ respects $A$, and $(u, v)$ is a light edge for this cut. Therefore $(u, v)$ is safe for $A$ by the above theorem. $\square$

- This corollary implies correctness for Kruskal's algorithm.