Cash-Flow Management Problem – Modelling as LP

cashflow.lp

```
Maximize
  wealth:  v
Subject To
  Jan:  x1 + y1 - z1 = 150
  Feb:  x2 + y2 - 1.01 x1 + 1.003 z1 - z2 = 100
  Mar:  x3 + y3 - 1.01 x2 + 1.003 z2 - z3 = -200
  Apr:  x4 - 1.02 y1 - 1.01 x3 + 1.003 z3 - z4 = 200
  May:  x5 - 1.02 y2 - 1.01 x4 + 1.003 z4 - z5 = -50
  Jun:  - 1.02 y3 - 1.01 x5 + 1.003 z5 - v = -300
Bounds
  0 <= x1 <= 100
  0 <= x2 <= 100
  0 <= x3 <= 100
  0 <= x4 <= 100
  0 <= x5 <= 100
  -Inf <= v <= Inf
End
```

Cash-Flow Management Problem – Modelling as LP

Gurobi Output

```
Solved in 5 iterations and 0.00 seconds
Optimal objective 9.249694915e+01
v 92.4969491525
x1 0.0
y1 150.0
z1 0.0
x2 0.0
y2 100.0
z2 0.0
x3 0.0
y3 151.944167498
z3 351.944167498
x4 0.0
z4 0.0
x5 52.0
z5 0.0

Obj:  92.4969491525
```

Optimal Investment Strategy:

Jan: Issue commercial paper for £150k.

Cash-Flow Management Problem – Modelling as LP

Gurobi Output

```
Solved in 5 iterations and 0.00 seconds
Optimal objective 9.249694915e+01
v 92.4969491525
x1 0.0
y1 150.0
z1 0.0
x2 0.0
y2 100.0
z2 0.0
x3 0.0
y3 151.944167498
z3 351.944167498
x4 0.0
z4 0.0
x5 52.0
z5 0.0

Obj:  92.4969491525
```

Optimal Investment Strategy:

Jan: Issue commercial paper for £150k.

Feb: Issue commercial paper for £100k.

Cash-Flow Management Problem – Modelling as LP

Gurobi Output

```
Solved in 5 iterations and 0.00 seconds
Optimal objective 9.249694915e+01
v 92.4969491525
x1 0.0
y1 150.0
z1 0.0
x2 0.0
y2 100.0
z2 0.0
x3 0.0
y3 151.944167498
z3 351.944167498
x4 0.0
z4 0.0
x5 52.0
z5 0.0

Obj:  92.4969491525
```

Optimal Investment Strategy:

- Jan: Issue commercial paper for £150k.
- Feb: Issue commercial paper for £100k.
- Mar: Issue paper for ≈ £152k and invest ≈ £352k.

Cash-Flow Management Problem – Modelling as LP

Gurobi Output

```
Solved in 5 iterations and 0.00 seconds
Optimal objective 9.249694915e+01
v 92.4969491525
x1 0.0
y1 150.0
z1 0.0
x2 0.0
y2 100.0
z2 0.0
x3 0.0
y3 151.944167498
z3 351.944167498
x4 0.0
z4 0.0
x5 52.0
z5 0.0

Obj:  92.4969491525
```

Optimal Investment Strategy:

- Jan: Issue commercial paper for £150k.
- Feb: Issue commercial paper for £100k.
- Mar: Issue paper for ≈ £152k and invest ≈ £352k.
- Apr: Take excess to pay outgoing cashflow.

Cash-Flow Management Problem – Modelling as LP

Gurobi Output

```
Solved in 5 iterations and 0.00 seconds
Optimal objective 9.249694915e+01
v 92.4969491525
x1 0.0
y1 150.0
z1 0.0
x2 0.0
y2 100.0
z2 0.0
x3 0.0
y3 151.944167498
z3 351.944167498
x4 0.0
z4 0.0
x5 52.0
z5 0.0

Obj:  92.4969491525
```

Optimal Investment Strategy:

Jan: Issue commercial paper for £150k.

Feb: Issue commercial paper for £100k.

Mar: Issue paper for ≈ £152k and invest ≈ £352k.

Apr: Take excess to pay outgoing cashflow.

May: Take a credit of £52k

Cash-Flow Management Problem – Modelling as LP

Gurobi Output

```
Solved in 5 iterations and 0.00 seconds
Optimal objective 9.249694915e+01
v 92.4969491525
x1 0.0
y1 150.0
z1 0.0
x2 0.0
y2 100.0
z2 0.0
x3 0.0
y3 151.944167498
z3 351.944167498
x4 0.0
z4 0.0
x5 52.0
z5 0.0

Obj:  92.4969491525
```

Optimal Investment Strategy:

- **Jan:** Issue commercial paper for £150k.
- **Feb:** Issue commercial paper for £100k.
- **Mar:** Issue paper for $\approx$ £152k and invest $\approx$ £352k.
- **Apr:** Take excess to pay outgoing cashflow.
- **May:** Take a credit of £52k
- **Jun:** wealth $\approx$ £92k

COMP331/557

Chapter 6: Optimal Trees and Paths

(Cook, Cunningham, Pulleyblank & Schrijver, Chapter 2)

Trees and Forests

Definition 6.1.

- i An undirected graph having no circuit is called a forest.
- ii A connected forest is called a tree.

Trees and Forests

Definition 6.1.

- i An undirected graph having no circuit is called a forest.
- ii A connected forest is called a tree.

Theorem 6.2.

Let $G = (V, E)$ be an undirected graph on $n = |V|$ nodes. Then, the following statements are equivalent:

- i G is a tree.

Trees and Forests

Definition 6.1.

- i An undirected graph having no circuit is called a forest.
- ii A connected forest is called a tree.

Theorem 6.2.

Let $G = (V, E)$ be an undirected graph on $n = |V|$ nodes. Then, the following statements are equivalent:

- i G is a tree.
- ii G has $n - 1$ edges and no circuit.

Trees and Forests

Definition 6.1.

- i An undirected graph having no circuit is called a forest.
- ii A connected forest is called a tree.

Theorem 6.2.

Let $G = (V, E)$ be an undirected graph on $n = |V|$ nodes. Then, the following statements are equivalent:

- i G is a tree.
- ii G has $n - 1$ edges and no circuit.
- iii G has $n - 1$ edges and is connected.

Trees and Forests

Definition 6.1.

1. An undirected graph having no circuit is called a forest.
2. A connected forest is called a tree.

Theorem 6.2.

Let $G = (V, E)$ be an undirected graph on $n = |V|$ nodes. Then, the following statements are equivalent:

1. G is a tree.
2. G has $n - 1$ edges and no circuit.
3. G has $n - 1$ edges and is connected.
4. G is connected. If an arbitrary edge is removed, the resulting subgraph is disconnected.

Trees and Forests

Definition 6.1.

- i An undirected graph having no circuit is called a forest.
- ii A connected forest is called a tree.

Theorem 6.2.

Let $G = (V, E)$ be an undirected graph on $n = |V|$ nodes. Then, the following statements are equivalent:

- i G is a tree.
- ii G has $n - 1$ edges and no circuit.
- iii G has $n - 1$ edges and is connected.
- iv G is connected. If an arbitrary edge is removed, the resulting subgraph is disconnected.
- v G has no circuit. Adding an arbitrary edge to G creates a circuit.

Trees and Forests

Definition 6.1.

i. An undirected graph having no circuit is called a forest.
ii. A connected forest is called a tree.

Theorem 6.2.

Let $G = (V, E)$ be an undirected graph on $n = |V|$ nodes. Then, the following statements are equivalent:

i. G is a tree.
ii. G has $n - 1$ edges and no circuit.
iii. G has $n - 1$ edges and is connected.
iv. G is connected. If an arbitrary edge is removed, the resulting subgraph is disconnected.
v. G has no circuit. Adding an arbitrary edge to G creates a circuit.
vi. G contains a unique path between any pair of nodes.

Kruskal's Algorithm

Minimum Spanning Tree (MST) Problem

Given: connected graph $G = (V, E)$, cost function $c : E \to \mathbb{R}$.

Task: find spanning tree $T = (V, F)$ of G with minimum cost $\sum_{e \in F} c(e)$.

Kruskal's Algorithm

Minimum Spanning Tree (MST) Problem

Given: connected graph $G = (V, E)$, cost function $c : E \to \mathbb{R}$.

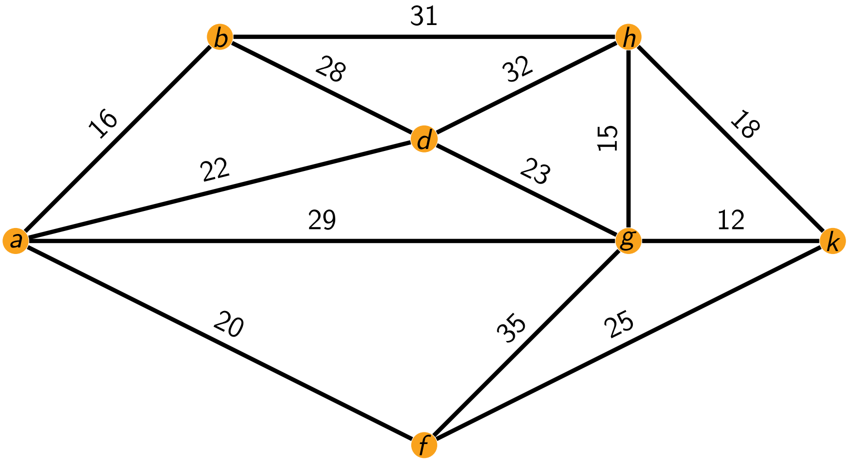
Task: find spanning tree $T = (V, F)$ of G with minimum cost $\sum_{e \in F} c(e)$.

Kruskal's Algorithm for MST

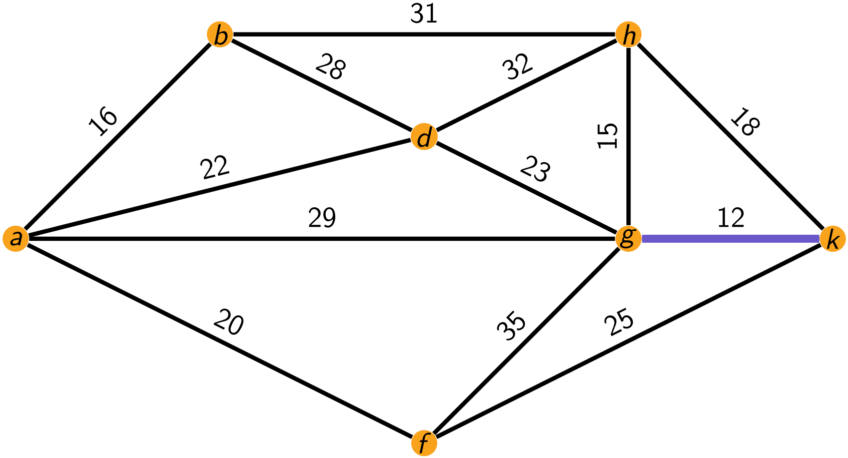
1. Sort the edges in E such that $c(e_1) \leq c(e_2) \leq \cdots \leq c(e_m)$.
2. Set $T := (V, \emptyset)$. ← empty set
3. For $i := 1$ to m do:

    If adding e_i to T does not create a circuit, then add e_i to T.

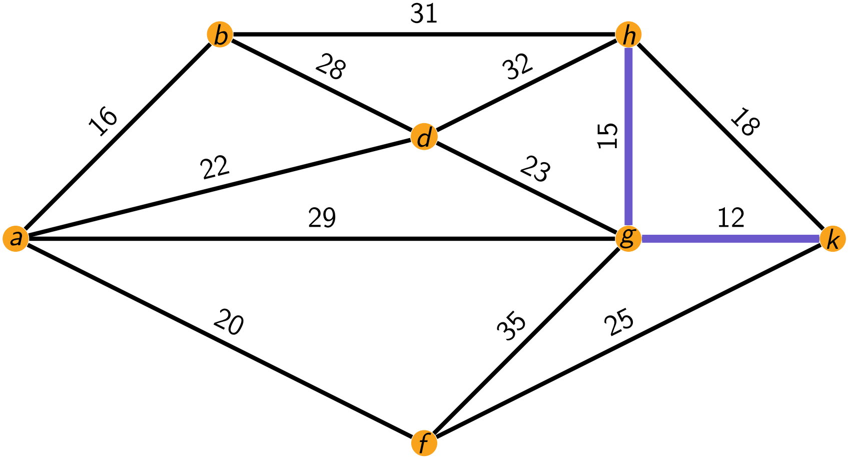
Example for Kruskal's Algorithm



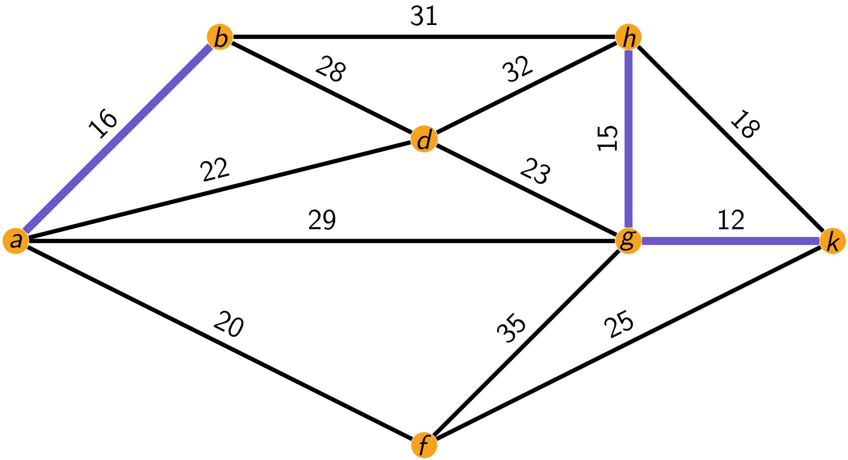
Example for Kruskal's Algorithm



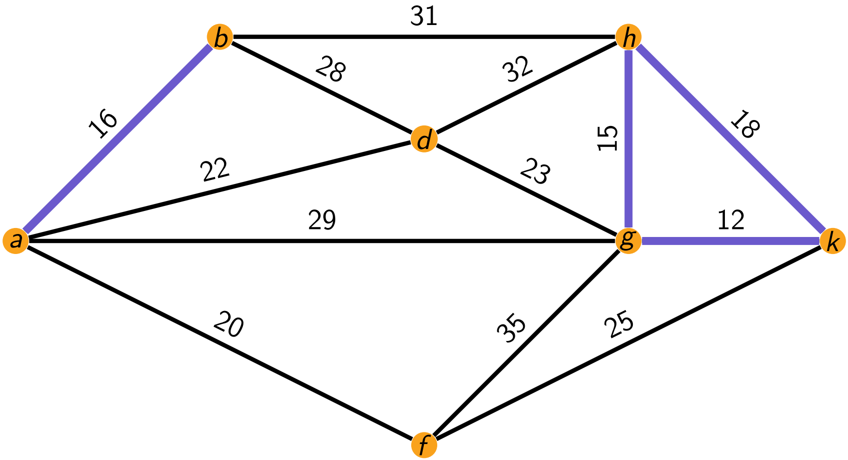
Example for Kruskal's Algorithm



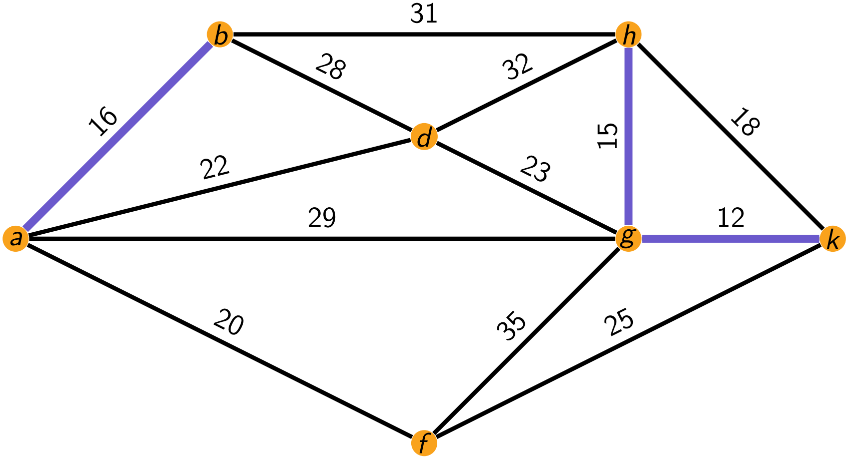
Example for Kruskal's Algorithm



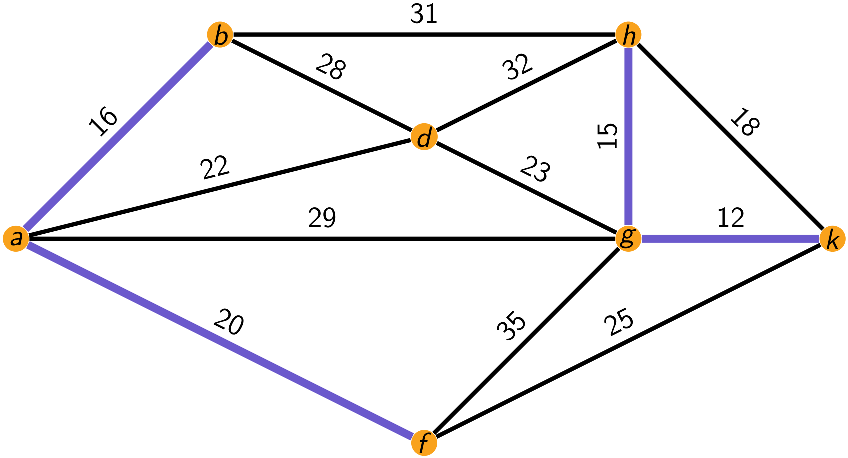
Example for Kruskal's Algorithm



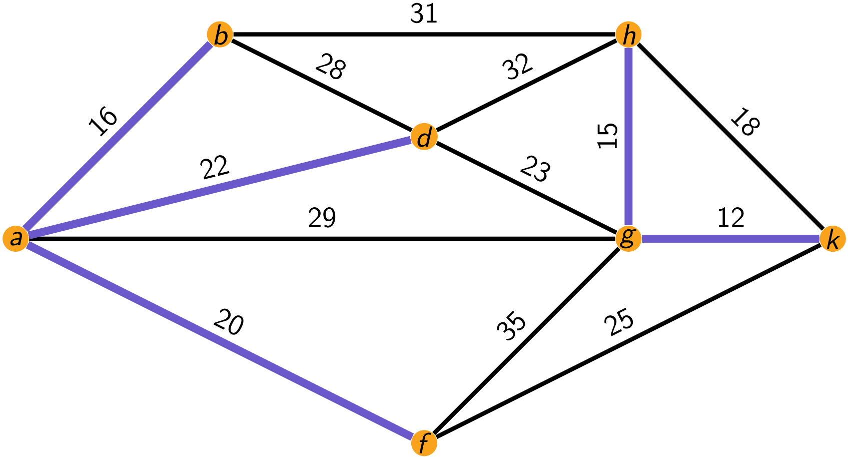
Example for Kruskal's Algorithm



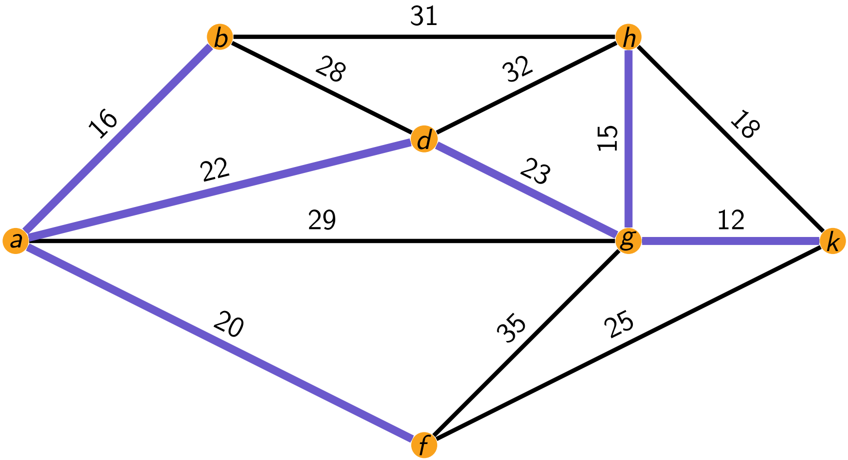
Example for Kruskal's Algorithm



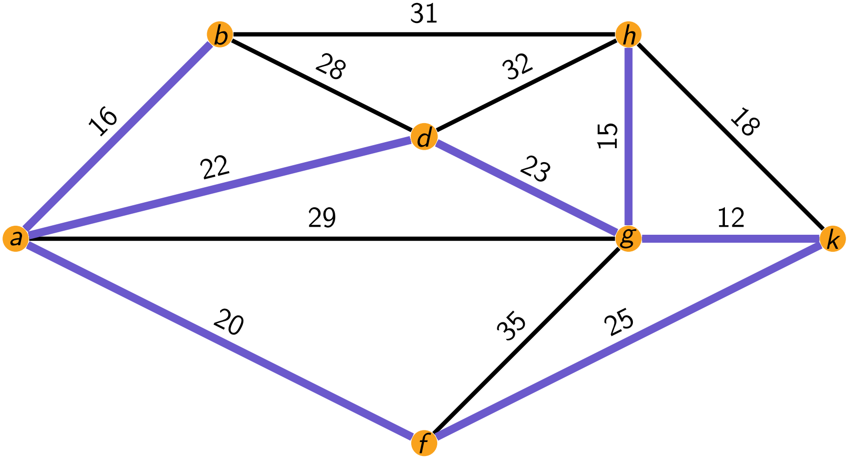
Example for Kruskal's Algorithm



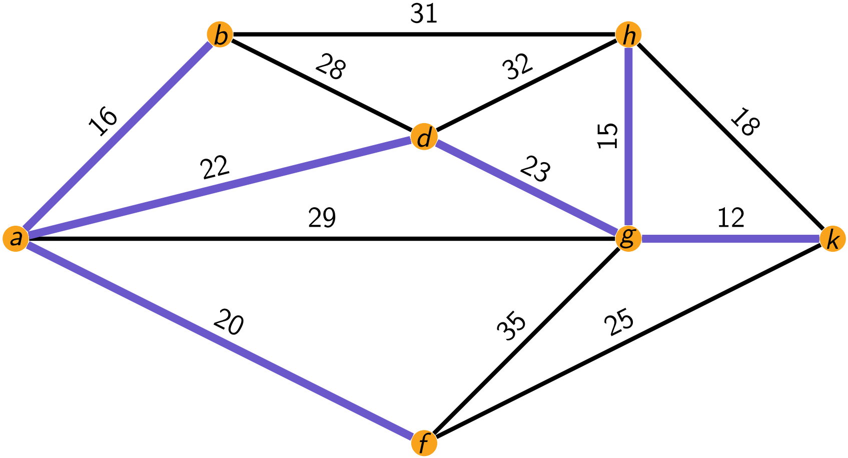
Example for Kruskal's Algorithm



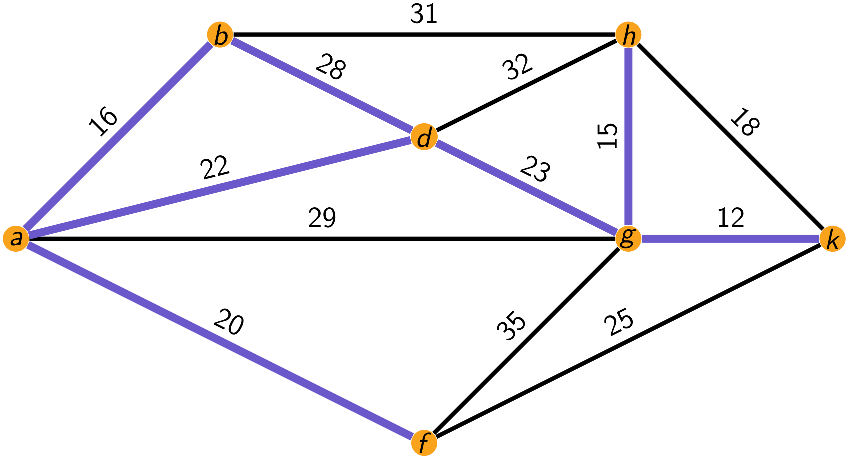
Example for Kruskal's Algorithm



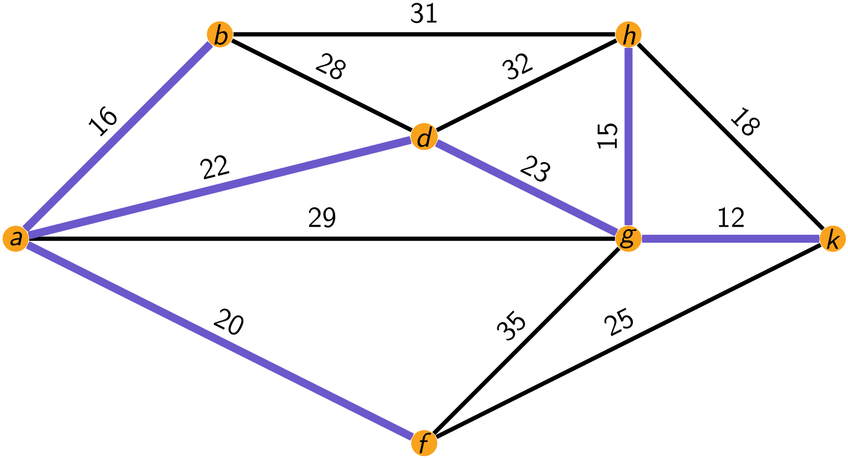
Example for Kruskal's Algorithm



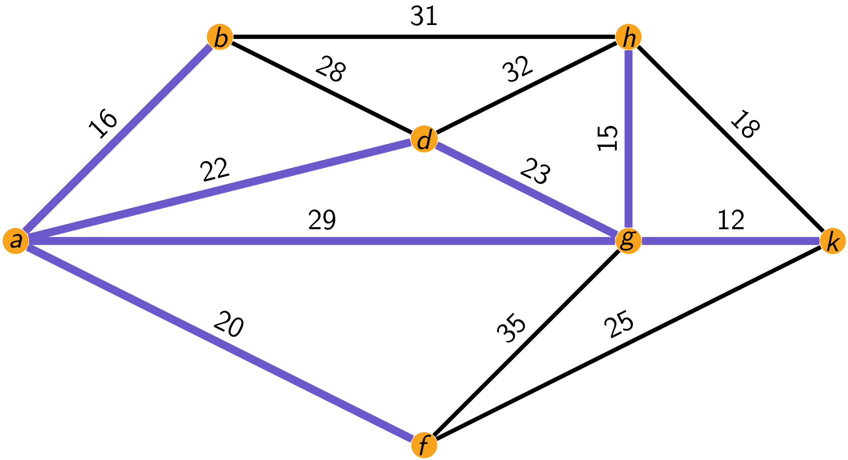
Example for Kruskal's Algorithm



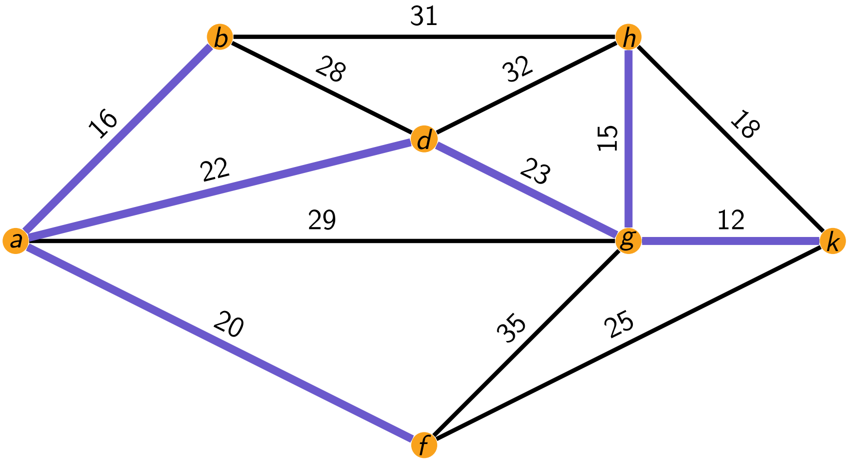
Example for Kruskal's Algorithm



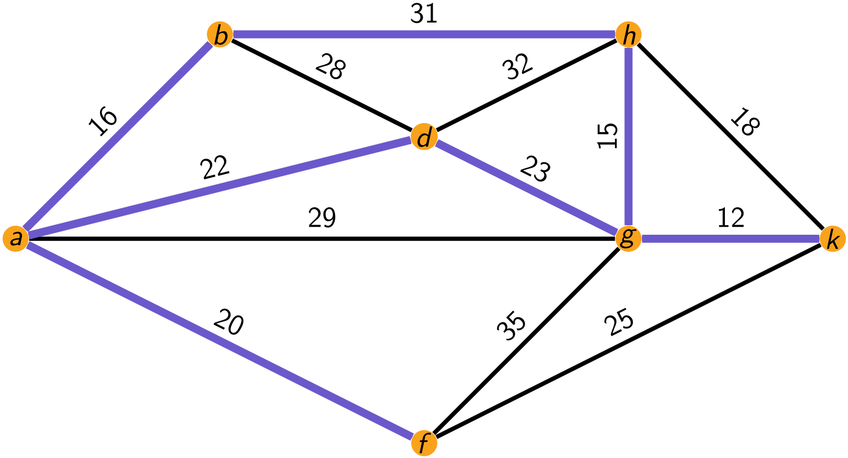
Example for Kruskal's Algorithm



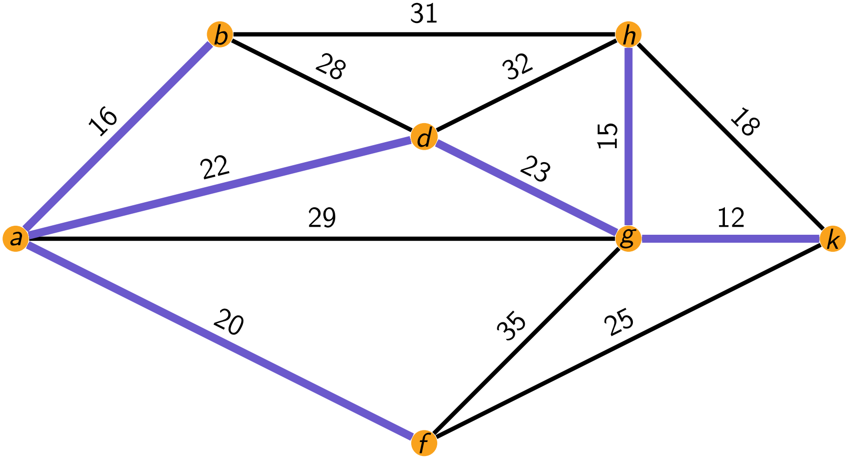
Example for Kruskal's Algorithm



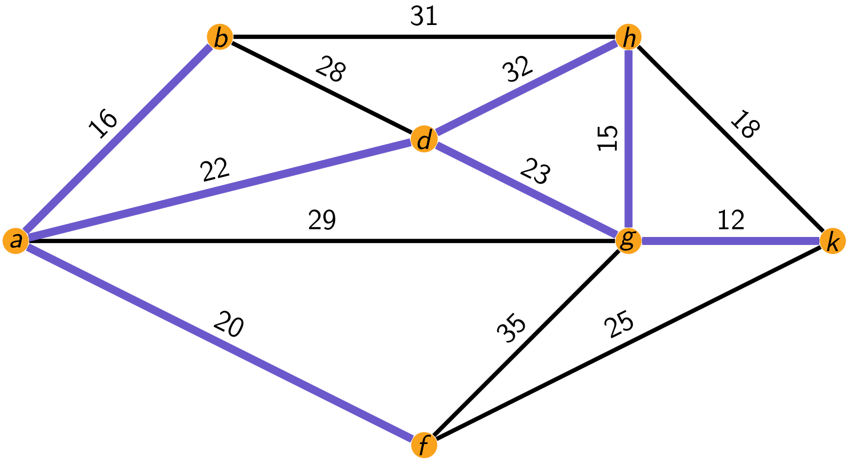
Example for Kruskal's Algorithm



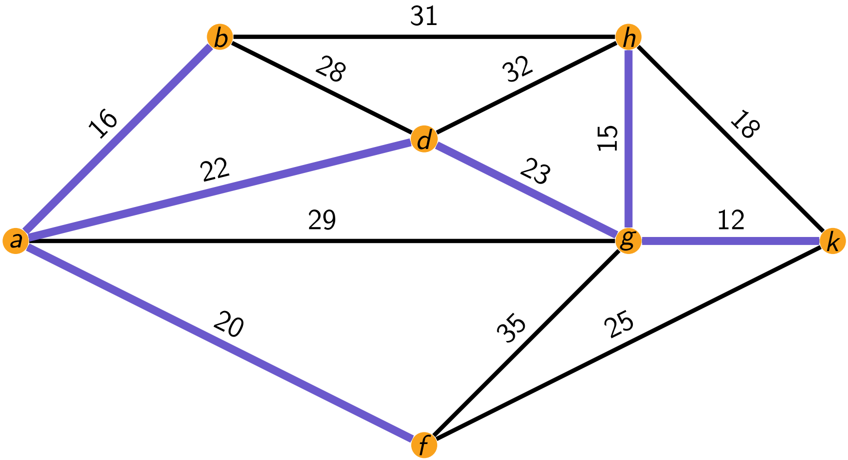
Example for Kruskal's Algorithm



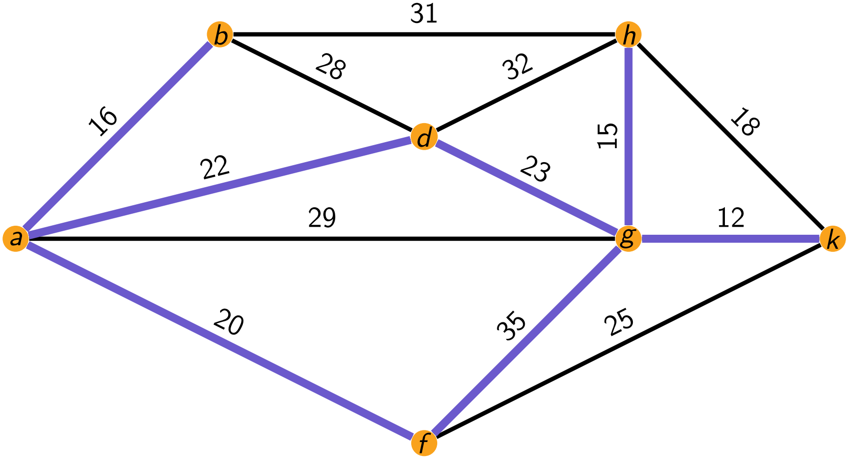
Example for Kruskal's Algorithm



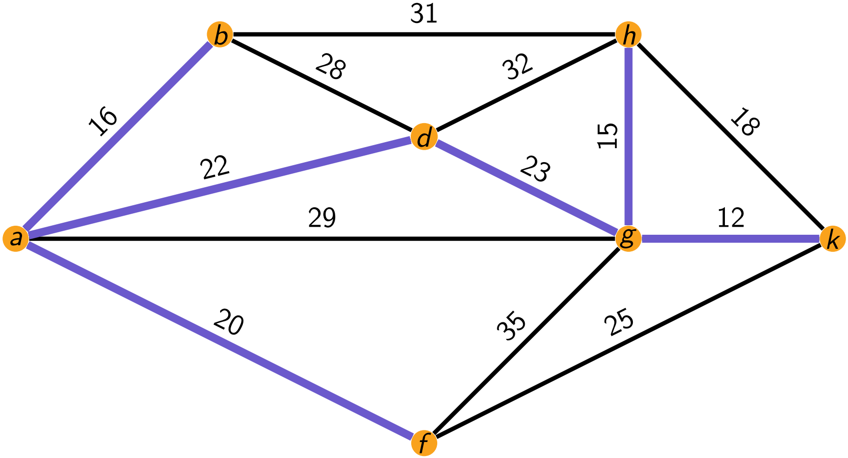
Example for Kruskal's Algorithm



Example for Kruskal's Algorithm



Example for Kruskal's Algorithm

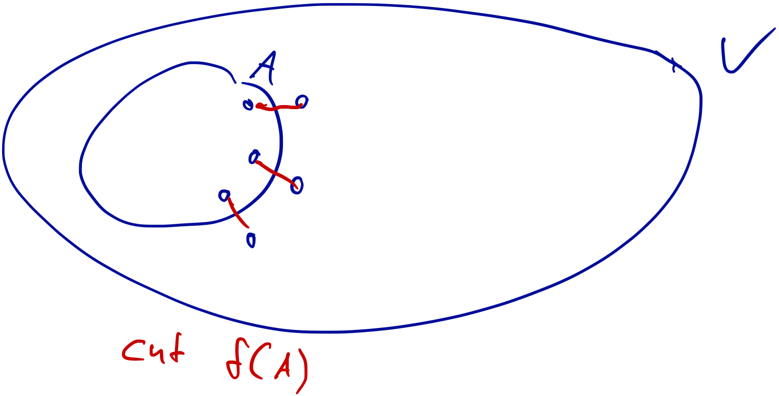


Prim's Algorithm

Notation: For a graph $G = (V, E)$ and $A \subseteq V$ let

$$\delta(A) := \{e = \{v, w\} \in E \mid v \in A \text{ and } w \in V \setminus A\} \ .$$

We call $\delta(A)$ the cut induced by A.



Prim's Algorithm

Notation: For a graph $G = (V, E)$ and $A \subseteq V$ let

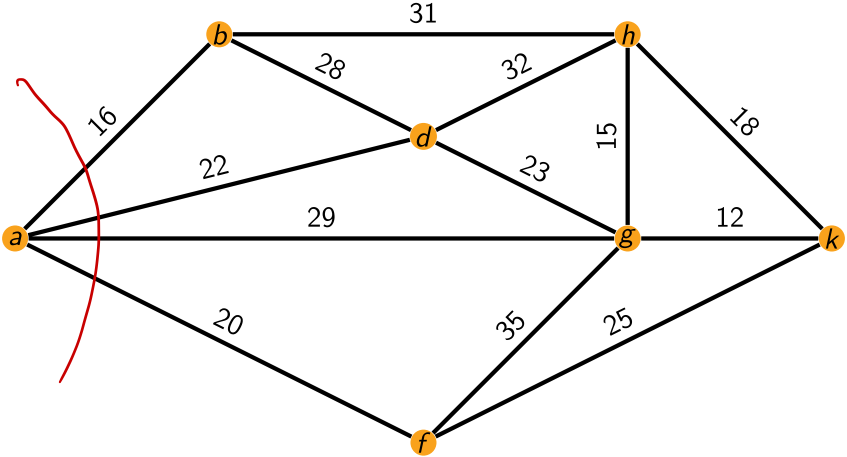
$$\delta(A) := \{e = \{v, w\} \in E \mid v \in A \text{ and } w \in V \setminus A\} \ .$$

We call $\delta(A)$ the cut induced by A.

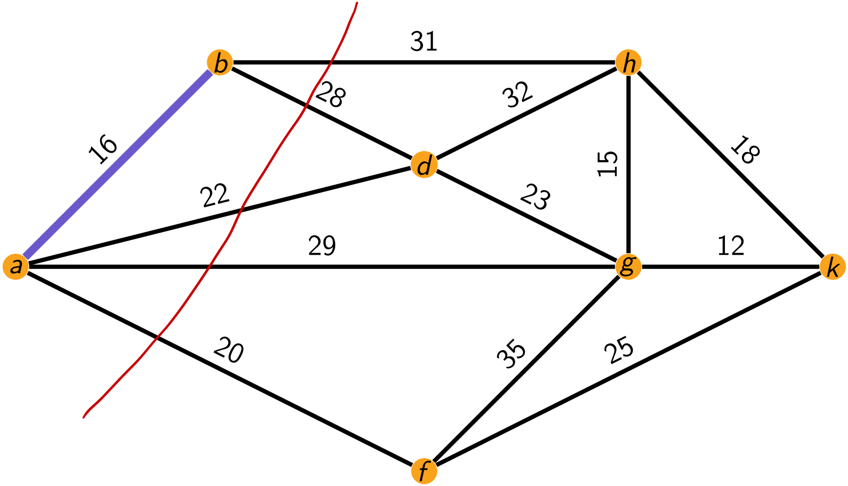
Prim's Algorithm for MST

1. Set $U := \{r\}$ for some node $r \in V$ and $F := \emptyset$; set $T := (U, F)$.
2. While $U \neq V$, determine a minimum cost edge $e \in \delta(U)$.
3. Set $F := F \cup \{e\}$ and $U := U \cup \{w\}$ with $e = \{v, w\}$, $w \in V \setminus U$.

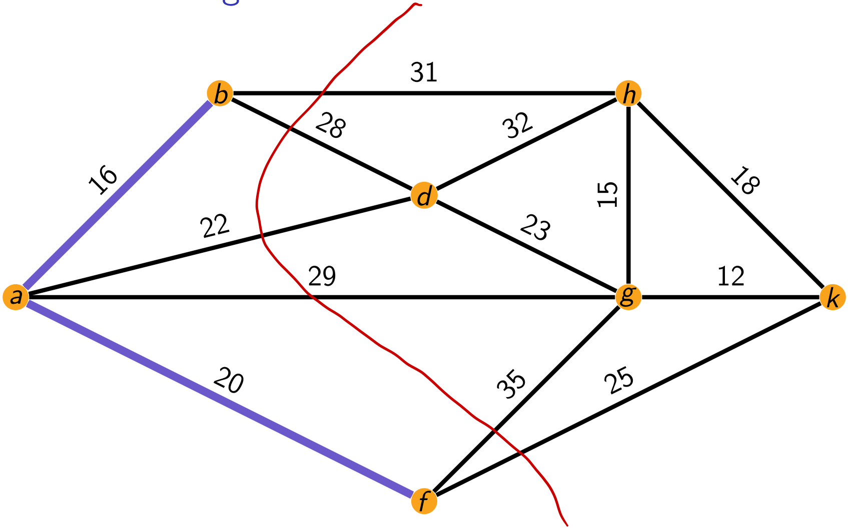
Example for Prim's Algorithm



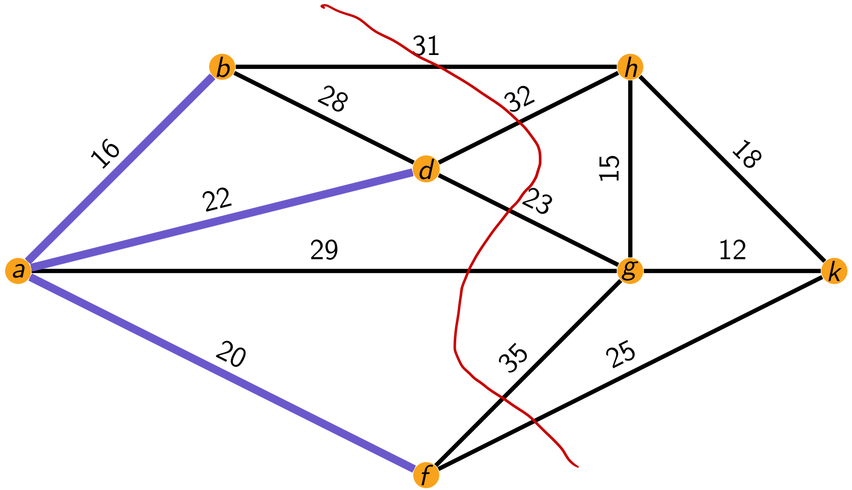
Example for Prim's Algorithm



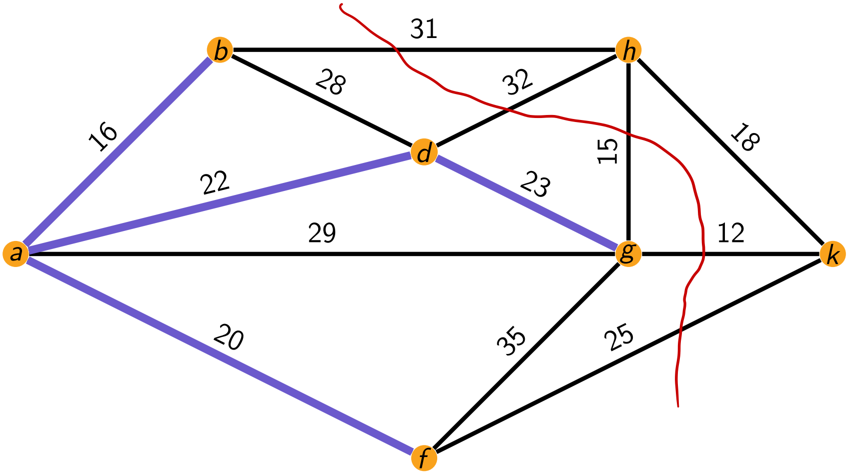
Example for Prim's Algorithm



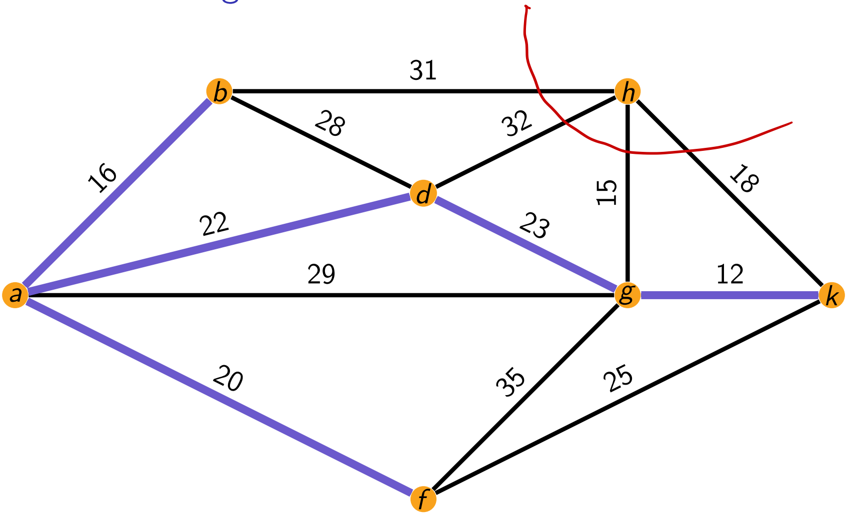
Example for Prim's Algorithm



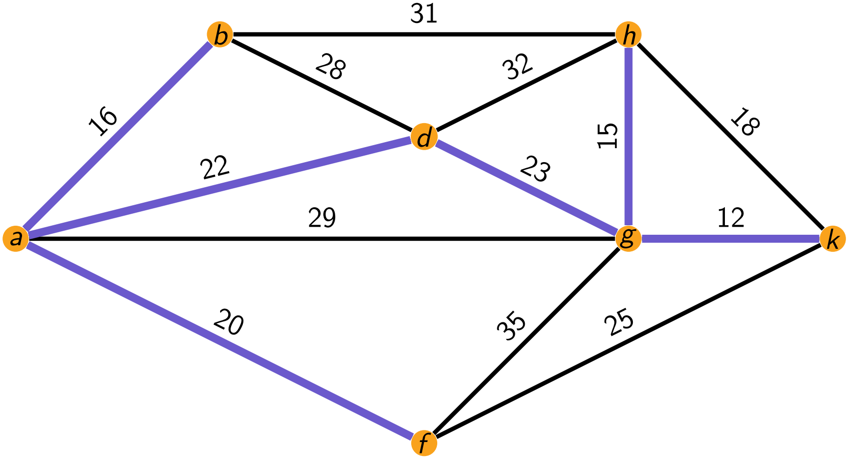
Example for Prim's Algorithm



Example for Prim's Algorithm



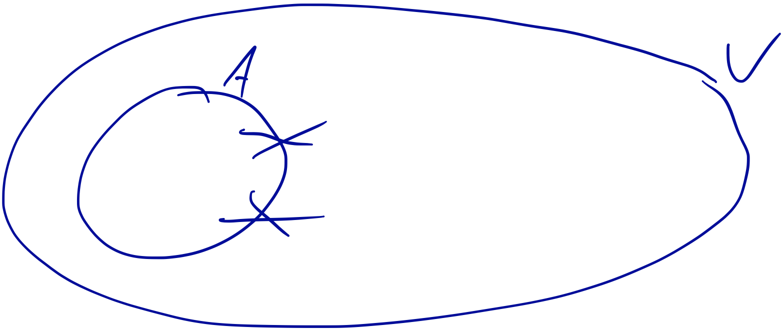
Example for Prim's Algorithm



Correctness of the MST Algorithms

Lemma 6.3.

A graph $G = (V, E)$ is connected if and only if there is no set $A \subseteq V$, $\emptyset \neq A \neq V$, with $\delta(A) = \emptyset$.



Correctness of the MST Algorithms

Lemma 6.3.

A graph $G = (V, E)$ is connected if and only if there is no set $A \subseteq V$, $\emptyset \neq A \neq V$, with $\delta(A) = \emptyset$.

Notation: We say that $B \subseteq E$ is extendible to an MST if B is contained in the edge-set of some MST of G.

Correctness of the MST Algorithms

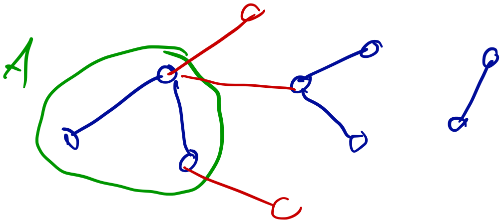
Lemma 6.3.

A graph $G = (V, E)$ is connected if and only if there is no set $A \subseteq V$, $\emptyset \neq A \neq V$, with $\delta(A) = \emptyset$.

Notation: We say that $B \subseteq E$ is extendible to an MST if B is contained in the edge-set of some MST of G.

Theorem 6.4.

Let $B \subseteq E$ be extendible to an MST and $\emptyset \neq A \subsetneq V$ with $B \cap \delta(A) = \emptyset$. If e is a min-cost edge in $\delta(A)$, then $B \cup \{e\}$ is extendible to an MST.



Correctness of the MST Algorithms

Lemma 6.3.

A graph $G = (V, E)$ is connected if and only if there is no set $A \subseteq V$, $\emptyset \neq A \neq V$, with $\delta(A) = \emptyset$.

Notation: We say that $B \subseteq E$ is extendible to an MST if B is contained in the edge-set of some MST of G.

Theorem 6.4.

Let $B \subseteq E$ be extendible to an MST and $\emptyset \neq A \subsetneq V$ with $B \cap \delta(A) = \emptyset$.
If e is a min-cost edge in $\delta(A)$, then $B \cup \{e\}$ is extendible to an MST.

- Correctness of Prim's Algorithm immediately follows.

Correctness of the MST Algorithms

Lemma 6.3.

A graph $G = (V, E)$ is connected if and only if there is no set $A \subseteq V$, $\emptyset \neq A \neq V$, with $\delta(A) = \emptyset$.

Notation: We say that $B \subseteq E$ is extendible to an MST if B is contained in the edge-set of some MST of G.

Theorem 6.4.

Let $B \subseteq E$ be extendible to an MST and $\emptyset \neq A \subsetneq V$ with $B \cap \delta(A) = \emptyset$.
If e is a min-cost edge in $\delta(A)$, then $B \cup \{e\}$ is extendible to an MST.

- Correctness of Prim's Algorithm immediately follows.
- Kruskal: Whenever an edge $e = \{v, w\}$ is added, it is cheapest edge in cut induced by subset of nodes currently reachable from v.

Efficiency of Prim's Algorithm

Prim's Algorithm for MST

1. Set $U := \{r\}$ for some node $r \in V$ and $F := \emptyset$; set $T := (U, F)$.
2. While $U \neq V$, determine a minimum cost edge $e \in \delta(U)$.
3. Set $F := F \cup \{e\}$ and $U := U \cup \{w\}$ with $e = \{v, w\}$, $w \in V \setminus U$.

- Straightforward implementation achieves running time $O(nm)$ where, as usual, $n := |V|$ and $m := |E|$:
  - the while-loop has $n - 1$ iterations;
  - a min-cost edge $e \in \delta(U)$ can be found in $O(m)$ time.

Efficiency of Prim's Algorithm

Prim's Algorithm for MST

1. Set $U := \{r\}$ for some node $r \in V$ and $F := \emptyset$; set $T := (U, F)$.
2. While $U \neq V$, determine a minimum cost edge $e \in \delta(U)$.
3. Set $F := F \cup \{e\}$ and $U := U \cup \{w\}$ with $e = \{v, w\}$, $w \in V \setminus U$.

- Straightforward implementation achieves running time $O(nm)$ where, as usual, $n := |V|$ and $m := |E|$:
  - the while-loop has $n - 1$ iterations;
  - a min-cost edge $e \in \delta(U)$ can be found in $O(m)$ time.
- Best known running time is $O(m + n \log n)$ (uses Fibonacci heaps).

Efficiency of Kruskal's Algorithm

Kruskal's Algorithm for MST

1. Sort the edges in E such that $c(e_1) \leq c(e_2) \leq \cdots \leq c(e_m)$.
2. Set $T := (V, \emptyset)$.
3. For $i := 1$ to m do:
   If adding e_i to T does not create a circuit, then add e_i to T.

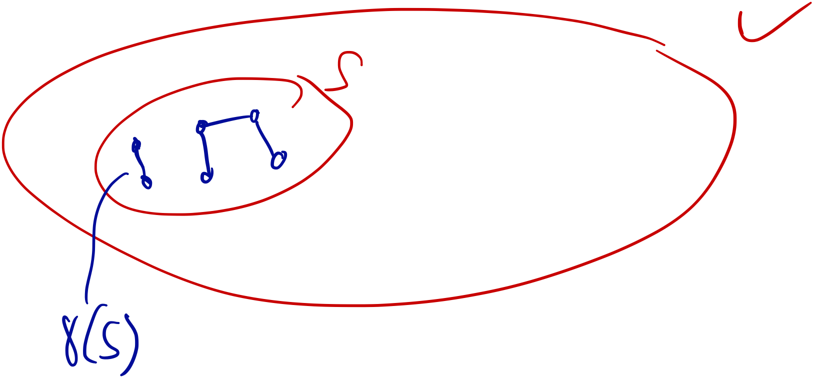
Theorem 6.5.

Kruskal's Algorithm can be implemented to run in $O(m \log m)$ time.

Minimum Spanning Trees and Linear Programming

Notation:

- For $S \subseteq V$ let $\gamma(S) := \{e = \{v, w\} \in E \mid v, w \in S\}$.



Minimum Spanning Trees and Linear Programming

Notation:

- For $S \subseteq V$ let $\gamma(S) := \big\{e = \{v, w\} \in E \mid v, w \in S\big\}$.
- For a vector $x \in \mathbb{R}^E$ and a subset $B \subseteq E$ let $x(B) := \sum_{e \in B} x_e$.