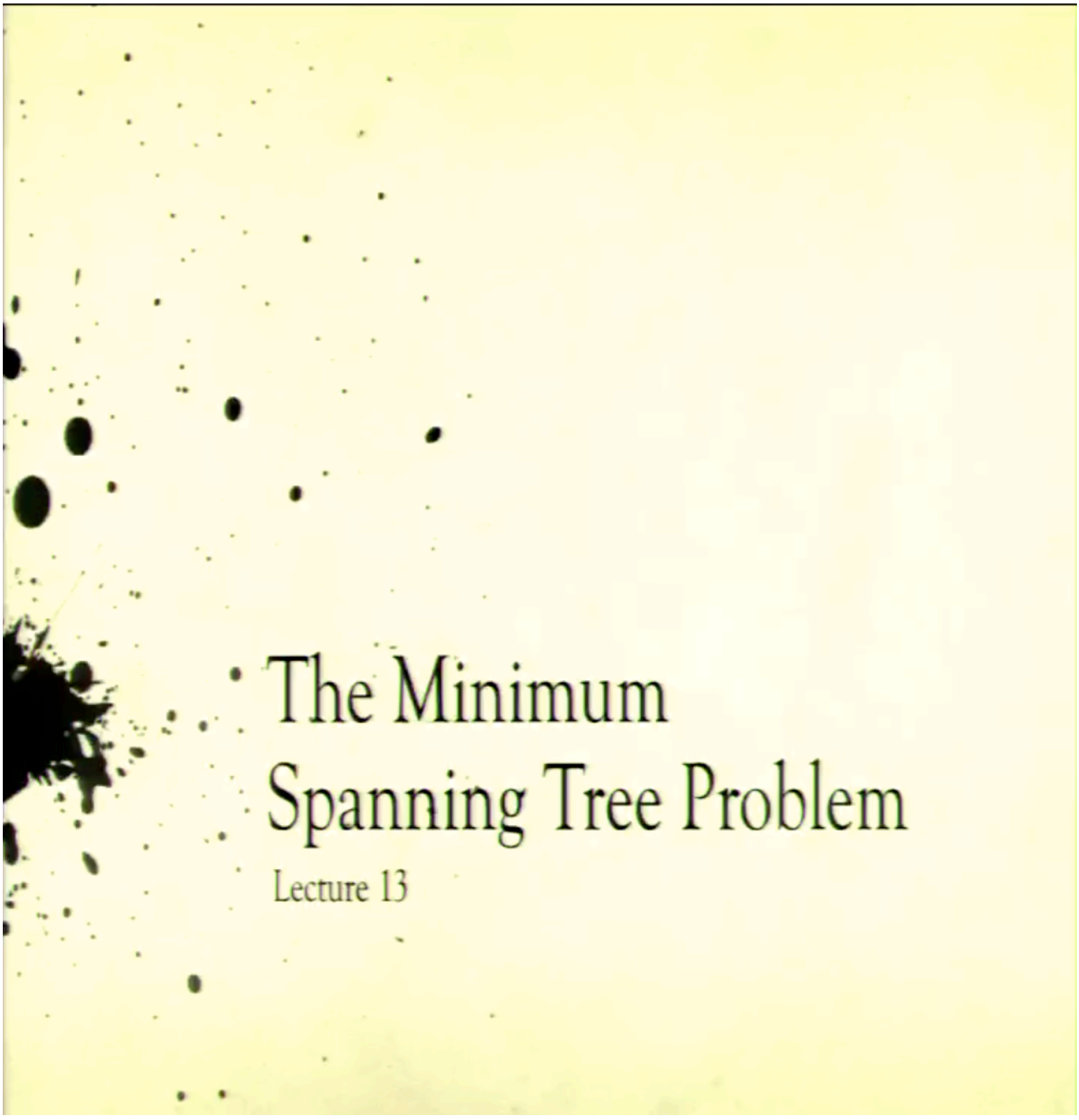
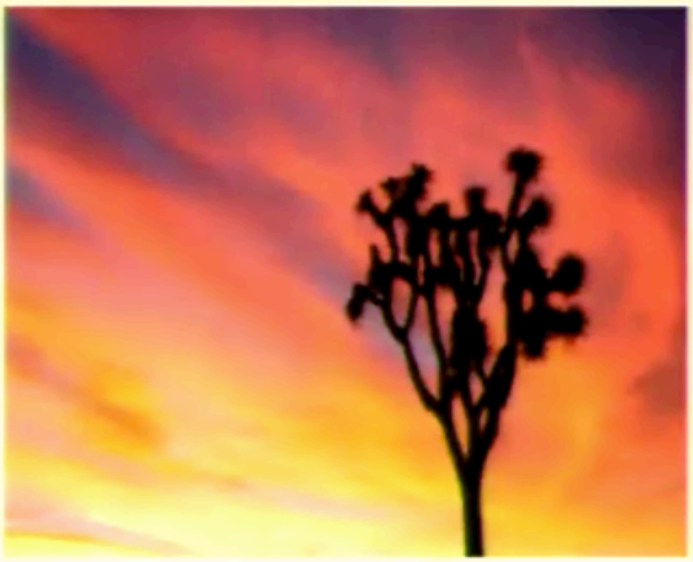


The Minimum Spanning Tree Problem

Lecture 13

The Minimum Spanning Tree Problem



- Given a graph G=(V,E) where each edge e has a non-negative cost c_e.
  - For convenience, we will assume all edge costs are distinct.
- The **cost** of a tree $T \subseteq E(G)$ is: $c(T) = \sum_{e \in T} c_e$
- A fundamental problem in network design is then:

Minimum Spanning Tree (MST) Problem. Find a spanning tree T in G with the minimum cost.

The MST Problem is a.k.a. the problem where (almost) every greedy algorithm works!

Kruskal's Algorithm

Kruskal's Algorithm [Kruskal 1956]

Sort the edges $\{e_1, e_2, \ldots, e_m\}$ by cost $c_1 < c_2 < \cdots < c_m$

Set $T = \emptyset$

For $i = \{1, 2, \ldots, m\}$

Let $e_i = (u, v)$

If u and v are in different components of T then set $T \leftarrow T \cup e_i$

Kruskal's Algorithm

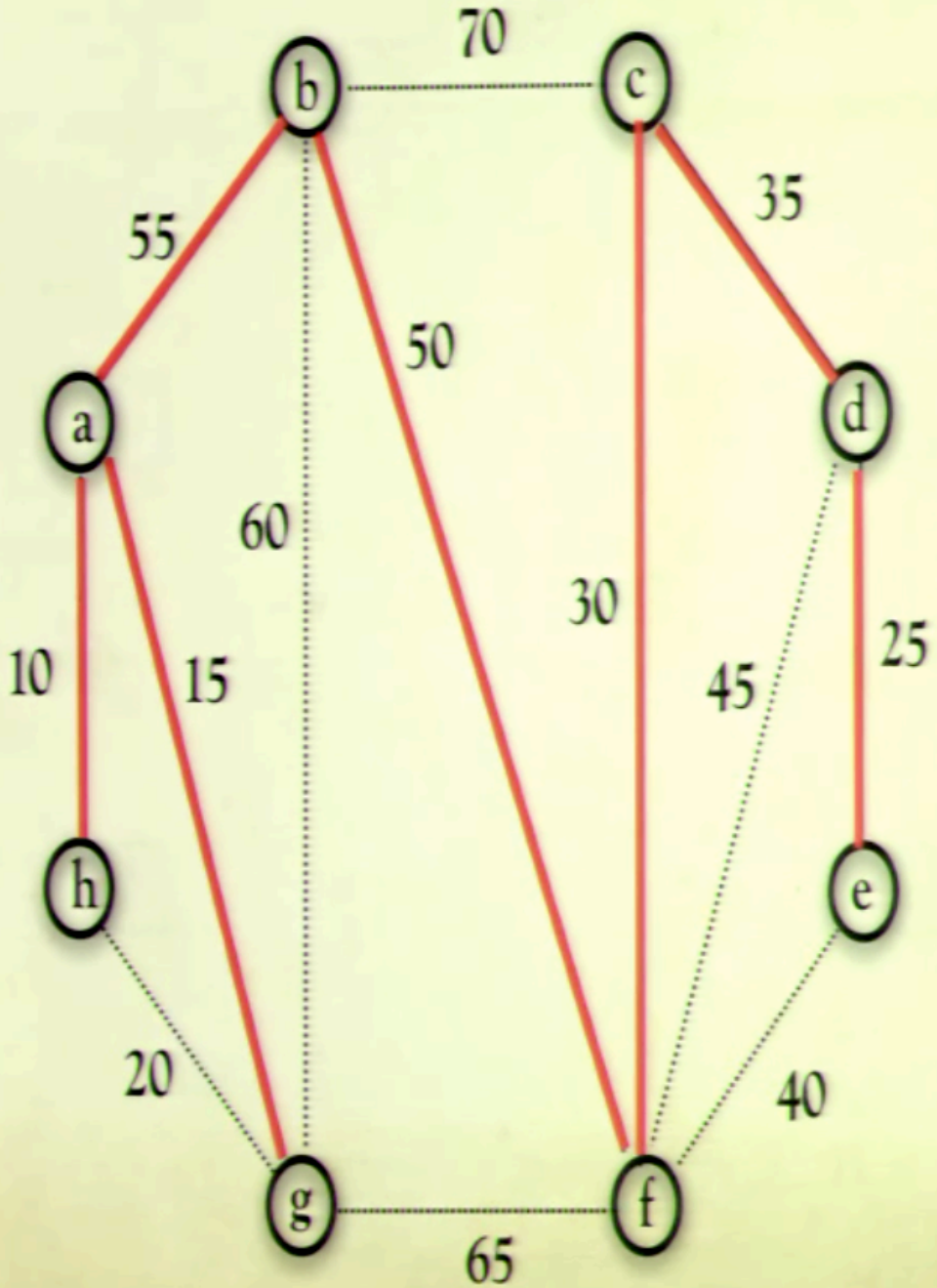
Sort the edges $c_1 < c_2 < \cdots < c_m$

Set $T = \emptyset$

For $i = \{1, 2, \ldots, m\}$

Let $e_i = (u, v)$

If u and v are in different components set $T \leftarrow T \cup e_i$



Prim's Algorithm

Prim's Algorithm [Jarnik 1929, Prim 1957]

Set $T = \{a\}$

If $V(T) \neq V$ **then**

Let e be the minimum cost edge in $\delta(T)$

Set $T \leftarrow T \cup e$

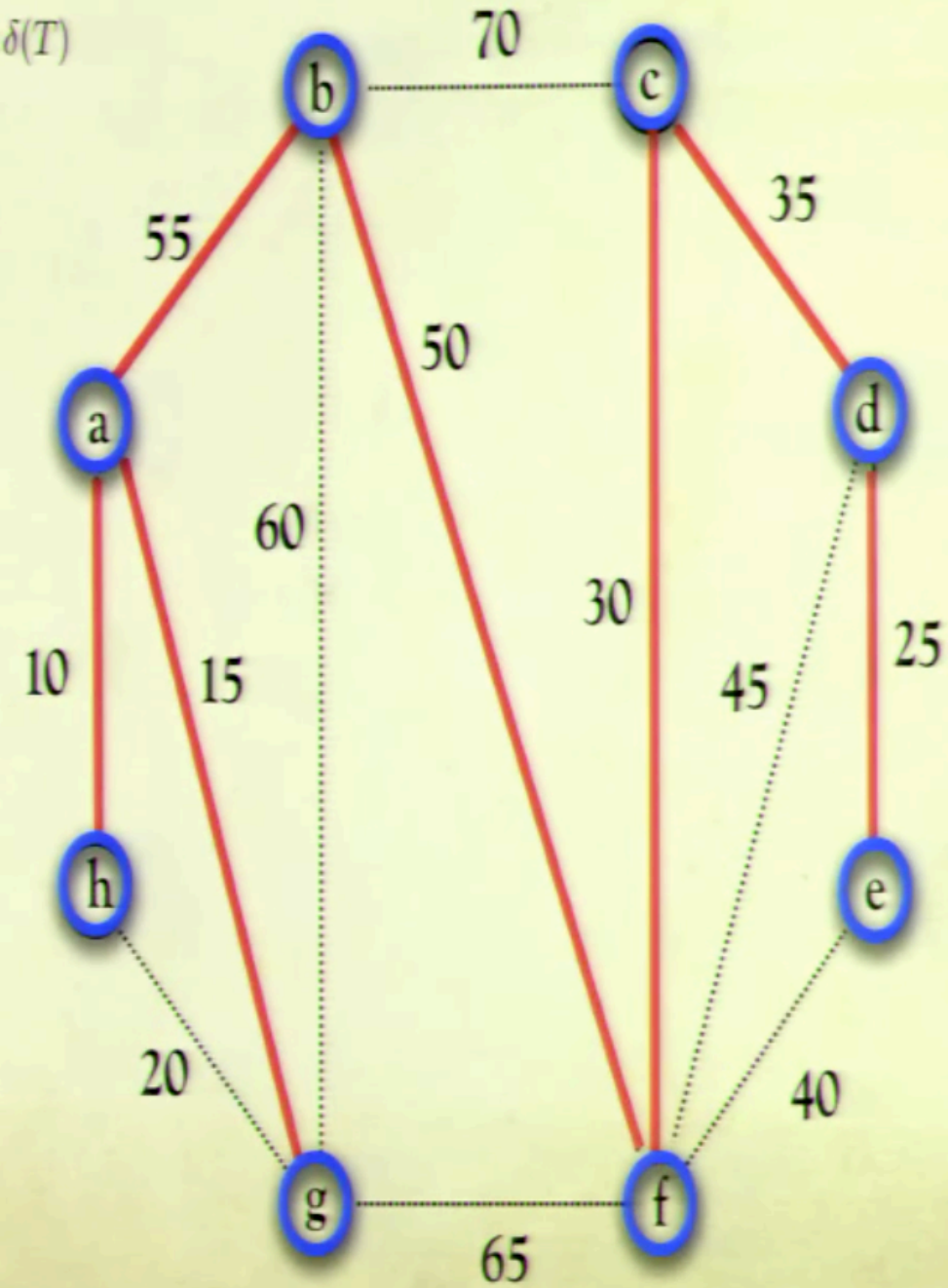
Prim's Algorithm

Set $T = \{a\}$

$V(T) \neq V$

Let e be the minimum cost edge in $\delta(T)$

Set $T \leftarrow T \cup e$



Borůvka's Algorithm

Borůvka's Algorithm [Borůvka 1926]

Set $T = \emptyset$

If T has more than one component $\{S_1, S_2, \ldots, S_\ell\}$ then

For $i = \{1, 2, \ldots, \ell\}$, let e_i be the minimum cost edge in $\delta(S_i)$

Set $T \leftarrow T \cup e_1 \cup e_2 \cup \cdots \cup e_\ell$

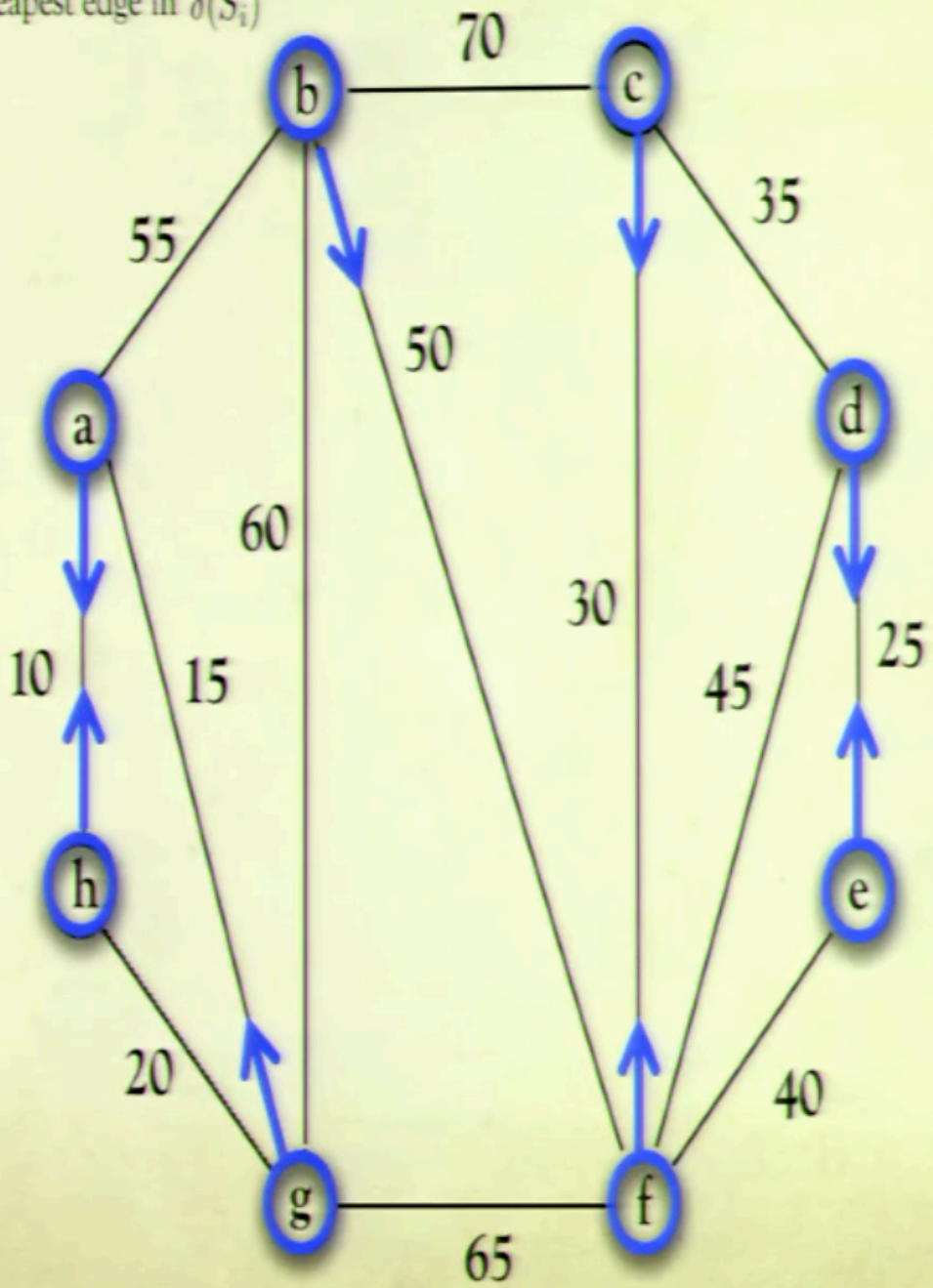
Borůvka's Algorithm

Set $T = \emptyset$

If T has more than one component $\{S_1, S_2, \ldots, S_\ell\}$ then

For $i = \{1, 2, \ldots, \ell\}$, let e_i be the cheapest edge in $\delta(S_i)$

Set $T \leftarrow T \cup e_1 \cup e_2 \cup \cdots \cup e_\ell$



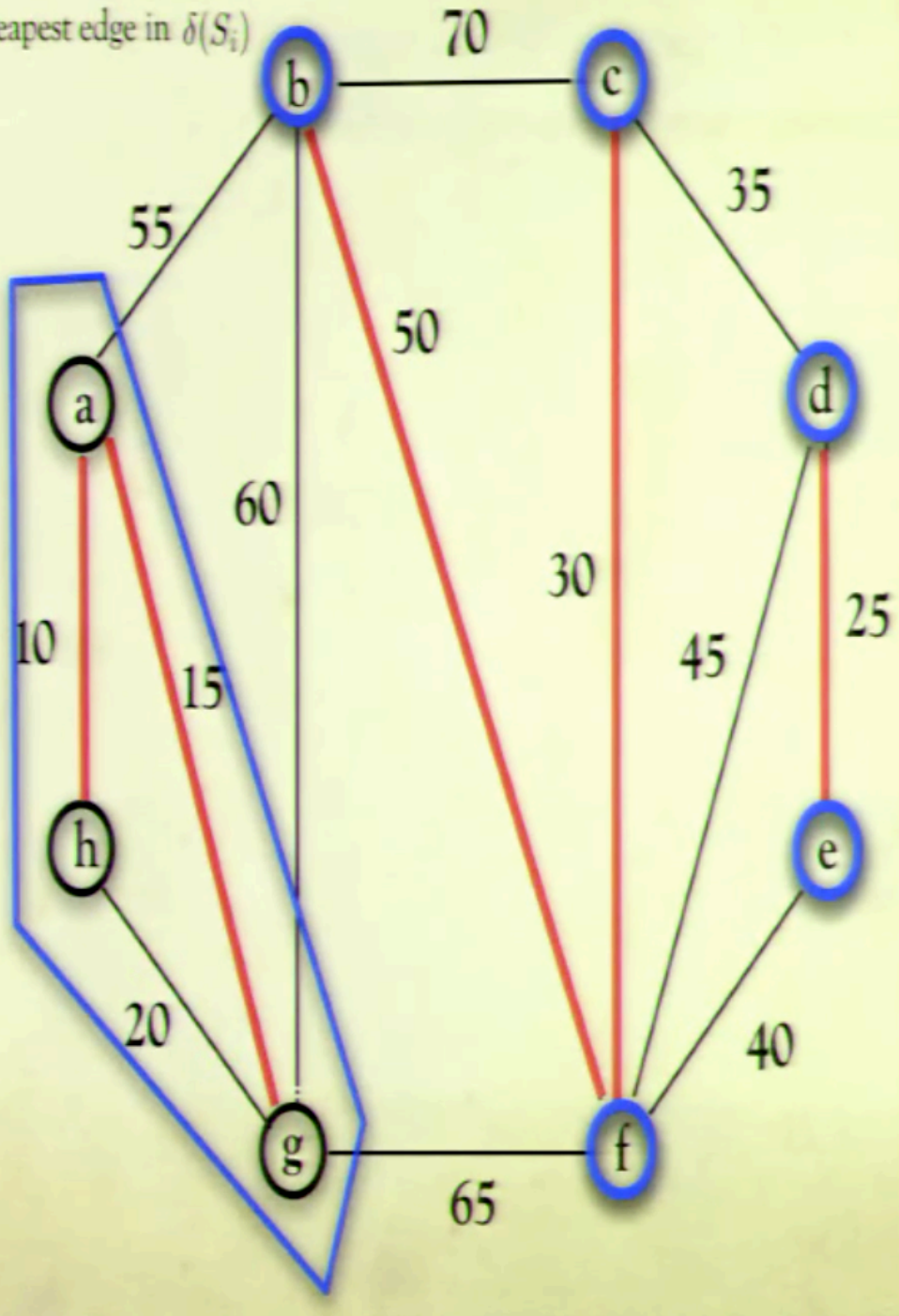
Borůvka's Algorithm

Set $T = \emptyset$

If T has more than one component $\{S_1, S_2, \ldots, S_\ell\}$ then

For $i = \{1, 2, \ldots, \ell\}$, let e_i be the cheapest edge in $\delta(S_i)$

Set $T \leftarrow T \cup e_1 \cup e_2 \cup \cdots \cup e_\ell$



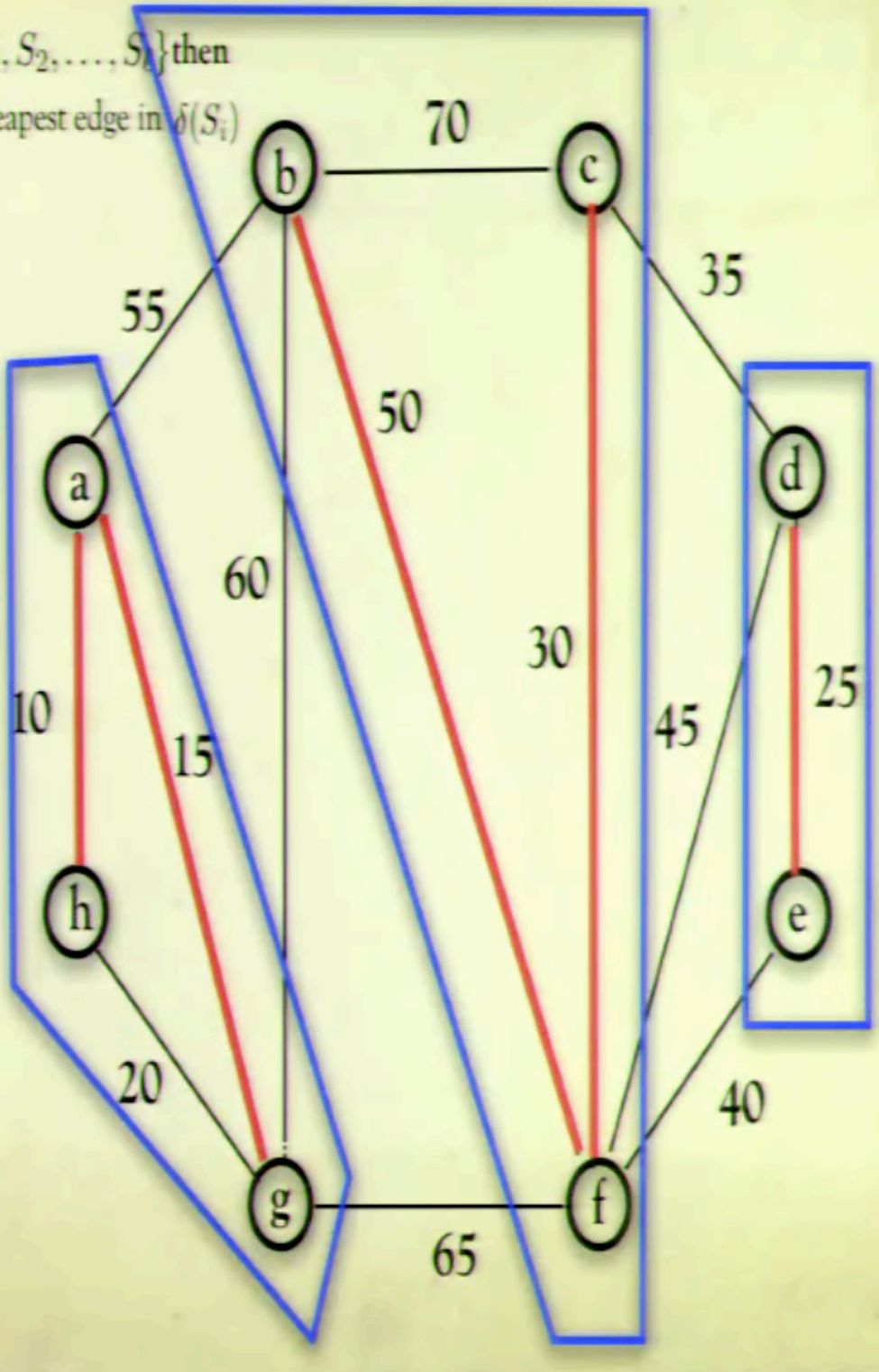
Borůvka's Algorithm

Set $T = \emptyset$

If T has more than one component $\{S_1, S_2, \ldots, S_\ell\}$ then

For $i = \{1, 2, \ldots, \ell\}$, let e_i be the cheapest edge in $\delta(S_i)$

Set $T \leftarrow T \cup e_1 \cup e_2 \cup \cdots \cup e_\ell$



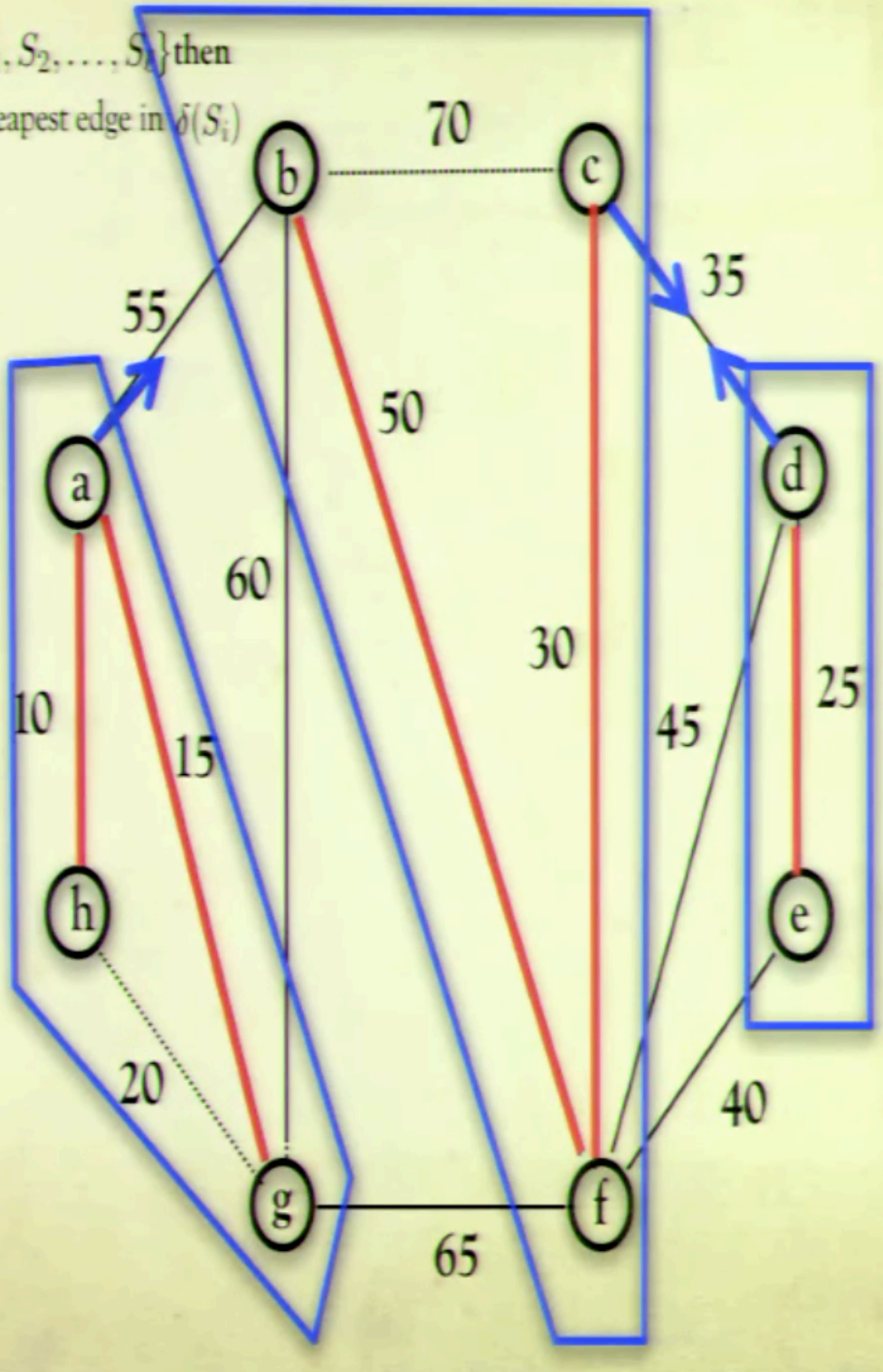
Borůvka's Algorithm

Set $T = \emptyset$

If T has more than one component $\{S_1, S_2, \ldots, S_\ell\}$ then

For $i = \{1, 2, \ldots, \ell\}$, let e_i be the cheapest edge in $\delta(S_i)$

Set $T \leftarrow T \cup e_1 \cup e_2 \cup \cdots \cup e_\ell$



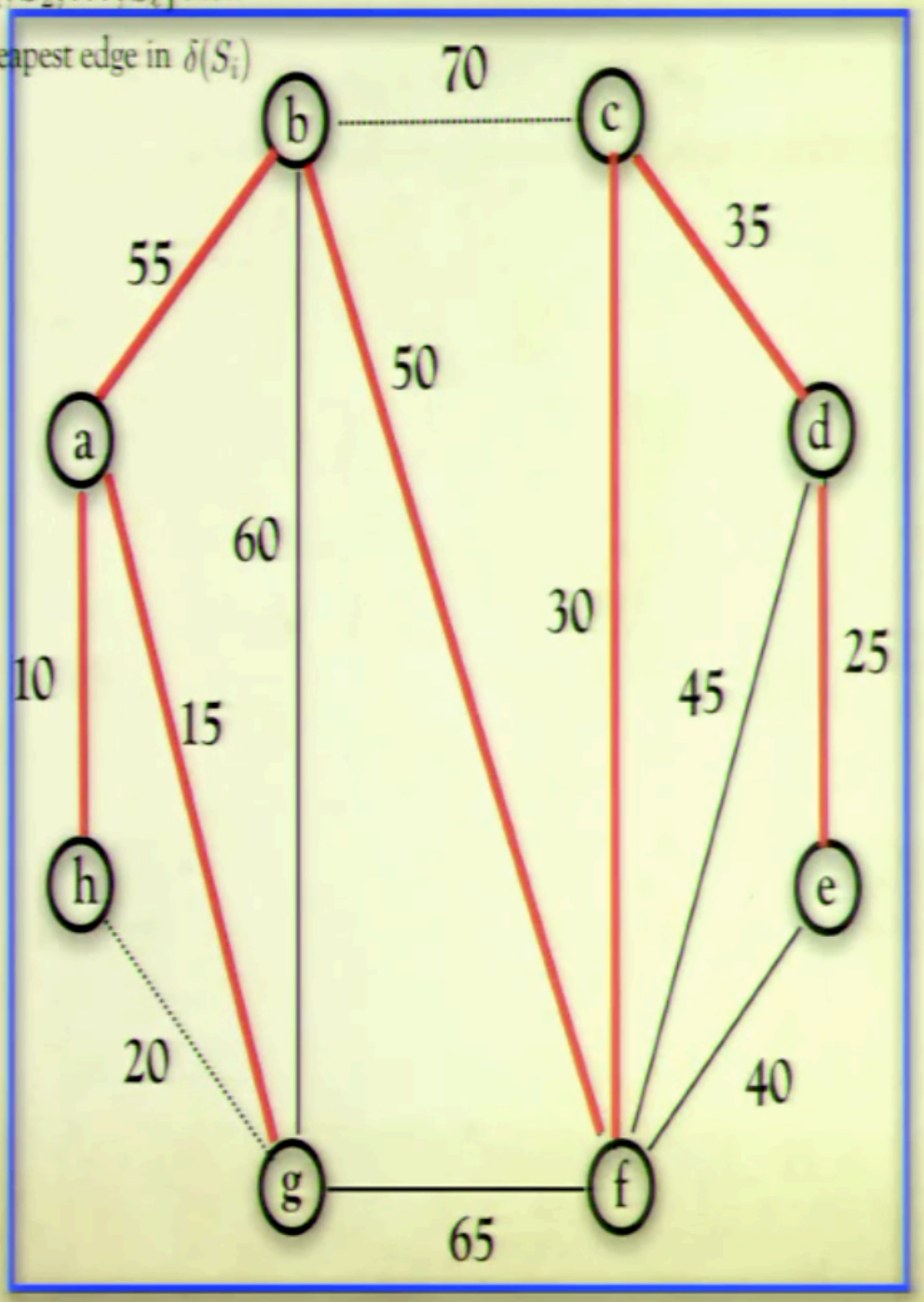
Borůvka's Algorithm

Set $T = \emptyset$

If T has more than one component $\{S_1, S_2, \ldots, S_\ell\}$ then

For $i = \{1, 2, \ldots, \ell\}$, let e_i be the cheapest edge in $\delta(S_i)$

Set $T \leftarrow T \cup e_1 \cup e_2 \cup \cdots \cup e_\ell$



Kruskal's Algorithm

Sort time is $O(m \cdot \log m)$

Kruskal's Algorithm [Kruskal 1956]

Sort the edges $\{e_1, e_2, \ldots, e_m\}$ by cost $c_1 < c_2 < \cdots < c_m$

Set $T = \emptyset$

m iterations

For $i = \{1, 2, \ldots, m\}$

Let $e_i = (u, v)$

Test this, by tree searching, in time $O(n)$

If u and v are in different components of T then set $T \leftarrow T \cup e_i$

$$\Longrightarrow \quad \text{Runtime} = O(m \cdot \log m + m \cdot n) = O(m \cdot n)$$

Prim's Algorithm

Prim's Algorithm [Jarnik 1929, Prim 1957]

Set $T = \{a\}$

n iterations

If $V(T) \neq V$ then

  Let e be the minimum cost edge in $\delta(T)$

  Set $T \leftarrow T \cup e$

Find this edge, by exhaustive search, in time $O(m)$

$\Longrightarrow$ Runtime $= O(m \cdot n)$

Borůvka's Algorithm

Borůvka's Algorithm [Borůvka 1927]

Set $T = \emptyset$

If T has more than one component $\{S_1, S_2, \ldots, S_\ell\}$ then

For $i = \{1, 2, \ldots, \ell\}$, let e_i be the minimum cost edge in $\delta(S_i)$

Set $T \leftarrow T \cup e_1 \cup e_2 \cup \cdots \cup e_\ell$

log n iterations

At most n components

Find this edge, by exhaustive search, in time $O(m)$

$$\Longrightarrow \quad \text{Runtime} = O(m \cdot n \cdot \log n)$$

Improving the Running Times

- In fact, later in the course, we will see how to data structures to improve the running times of these algorithms.

Data Structure for Disjoint Sets.

  - **Kruskal's Algorithm:** $O(m \cdot \log n + m \cdot \log^* n)$

Improving the Running Times

- In fact, later in the course, we will see how to data structures to improve the running times of these algorithms.

  - **Kruskal's Algorithm:** $O(m \cdot \log n + m \cdot \log^* n)$
  - **Prims's Algorithm:** $O(m + n \cdot \log n)$ — *Fibonacci Heaps.*

Improving the Running Times

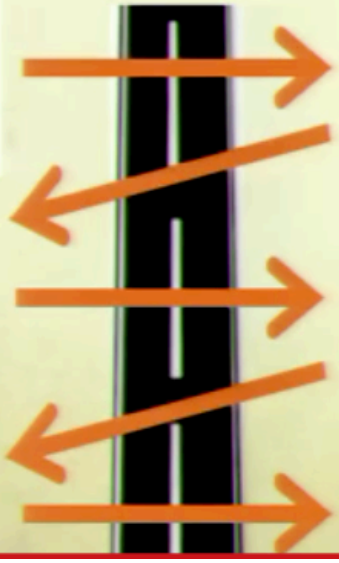
- In fact, later in the course, we will see how to data structures to improve the running times of these algorithms.
  - **Kruskal's Algorithm:** $O(m \cdot \log n + m \cdot \log^* n)$
  - **Prims's Algorithm:** $O(m + n \cdot \log n)$
  - **Borůvka's Algorithm:** $O(m \cdot \log^* n)$
- Here: $\log^* n = \min\{k : \underbrace{\log\log \cdot \log n}_{k \text{ times}} \leq 1\}$

e.g. $\log^*(2^{65536}) = 5$

Three Proofs for the Price of One

Question: *How many times does a Chicken cross the Road?*

Answer. *An odd number of times.*



The Cut Property of MSTs

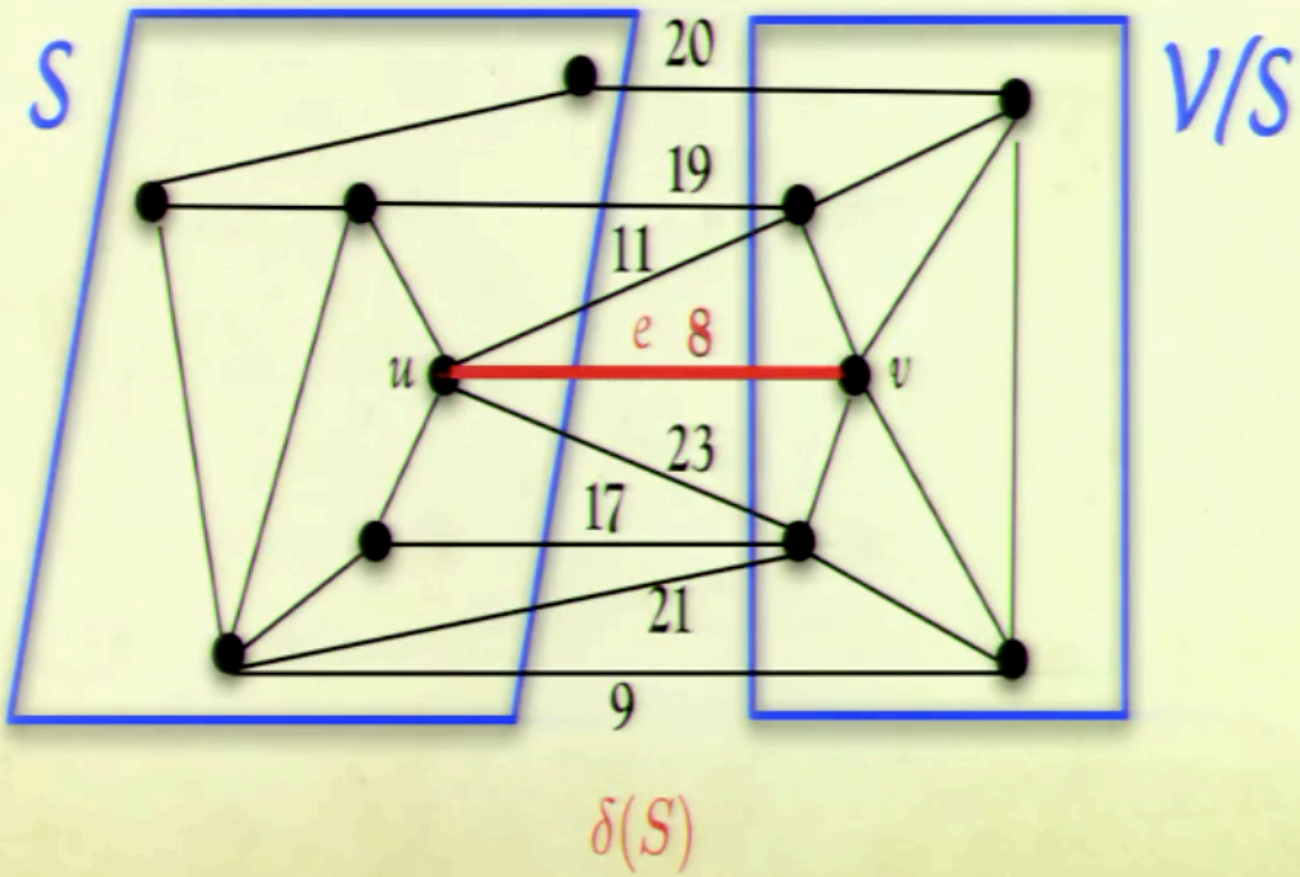
- The prove these algorithms work we must investigate the structure of Minimum Spanning Trees.
- One important property of MST is the *Cut Property*:

The Cut Property of Minimum Spanning Trees. Assume the edge costs are distinct. If e is the cheapest edge in some cut $\delta(S)$ then e is in the MST.

The Cut Property of Minimum Spanning Trees. Assume the edge costs are distinct. If e is the cheapest edge in some cut $\delta(S)$ then e is in the MST.

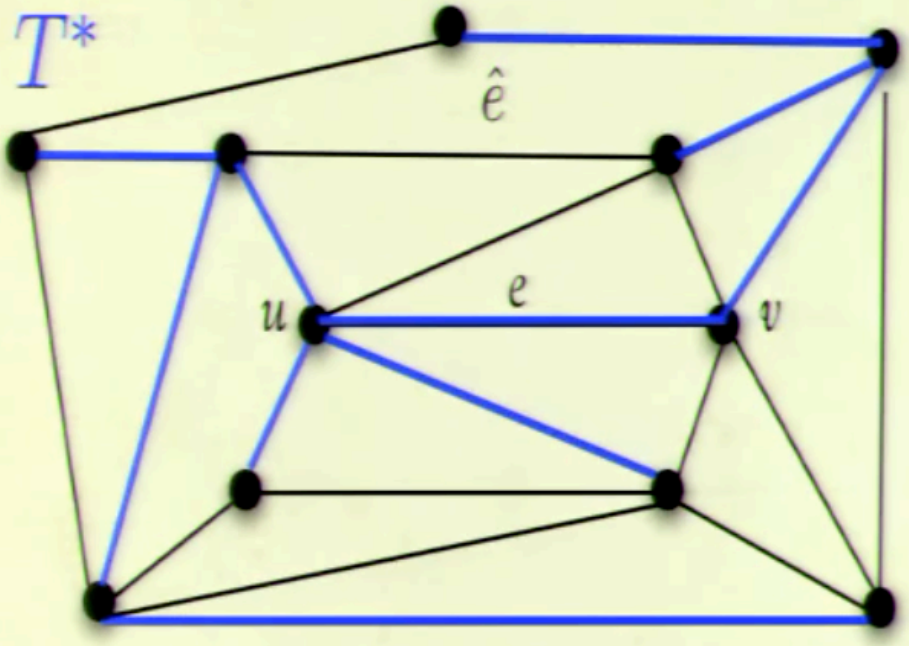
Proof.

- Let $e = (u, v)$ be the cheapest edge in a cut $\delta(S)$.



Proof [cont.]

- Let T^* be a minimum spanning tree, and assume $e \notin T^*$
- Since T^* is a spanning tree there is a unique path P in T^* from u to v.



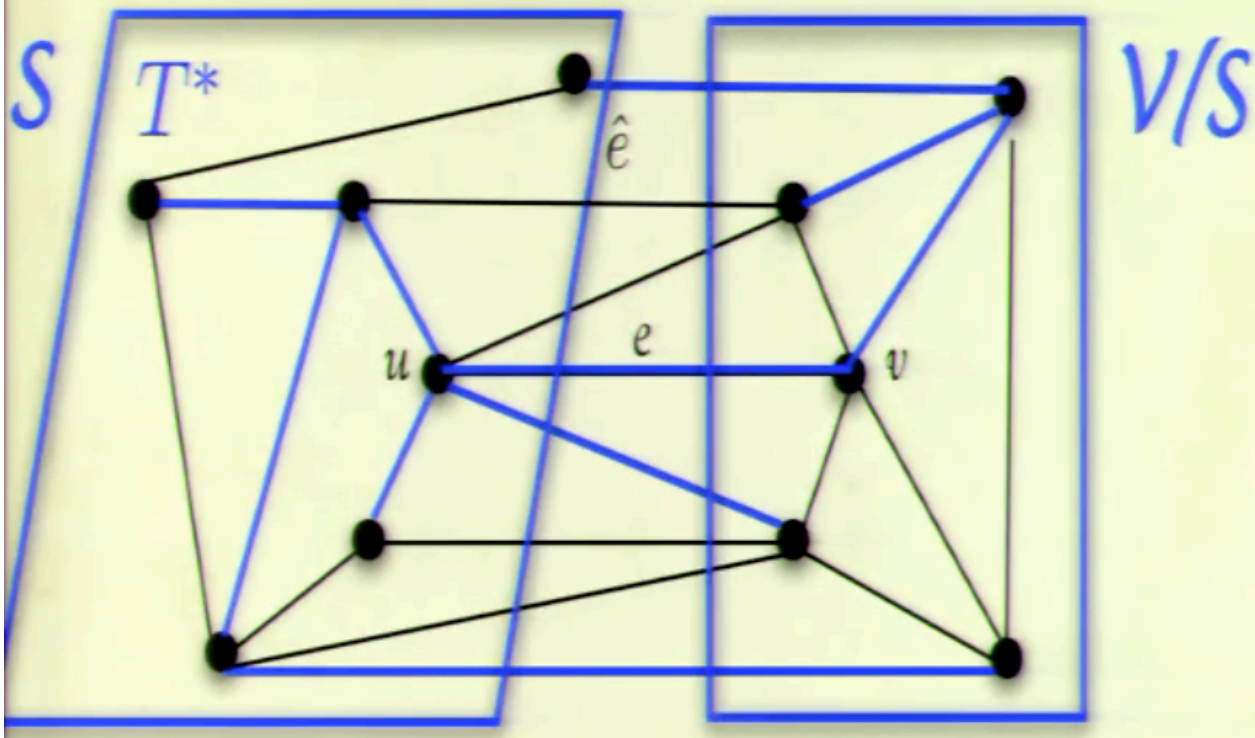
Observation 1. If $\hat{e} \in P$ then $(T^* \setminus \hat{e}) \cup e$ is a spanning tree.

- A chicken walking along P from u to v must cross the cut $\delta(S)$. So

Observation 2. There is at least one edge $\hat{e} \in P \cap \delta(S)$.

Proof [cont.]

- Let T^* be a minimum spanning tree, and assume $e \notin T^*$
- Since T^* is a spanning tree there is a unique path P in T^* from u to v.



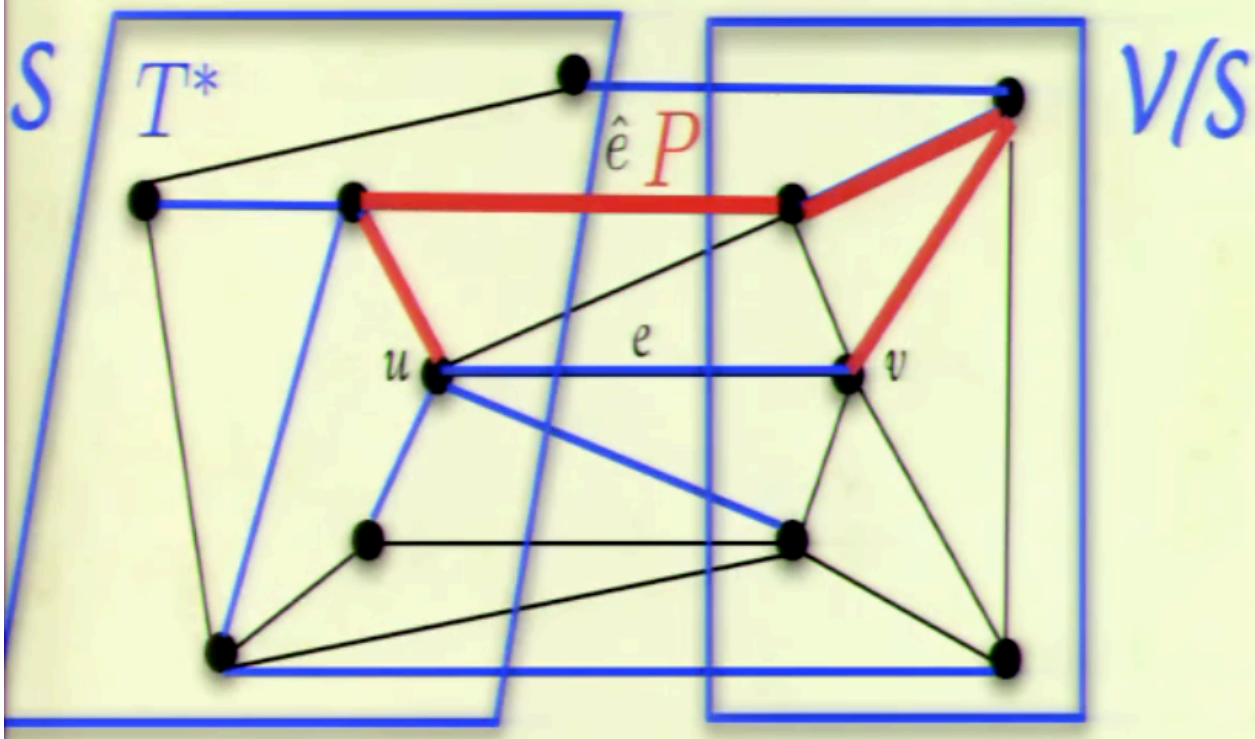
Observation 1. If $\hat{e} \in P$ then $(T^* \setminus \hat{e}) \cup e$ is a spanning tree.

- A chicken walking along P from u to v must cross the cut $\delta(S)$. So

Observation 2. There is at least one edge $\hat{e} \in P \cap \delta(S)$.

Proof [cont.]

- Let T^* be a minimum spanning tree, and assume $e \notin T^*$
- Since T^* is a spanning tree there is a unique path P in T^* from u to v.



Observation 1. If $\hat{e} \in P$ then $(T^* \setminus \hat{e}) \cup e$ is a spanning tree.

- A chicken walking along P from u to v must cross the cut $\delta(S)$. So

Observation 2. There is at least one edge $\hat{e} \in P \cap \delta(S)$.

Proof [cont.]

Observation 1. If $\hat{e} \in P$ then $(T^* \setminus \hat{e}) \cup e$ is a spanning tree.

Observation 2. There is at least one edge $\hat{e} \in P \cap \delta(S)$.

$\Longrightarrow \quad c_{\hat{e}} > c_e$

$\Longrightarrow \quad (T^* \setminus \hat{e}) \cup e$ is a cheaper spanning tree than T^*.

- This gives the desired contraction. □

We can use the Cut Property of MSTs to prove that all three of our greedy algorithms work...

Prim's Algorithm

Prim's Algorithm [Jarnik 1929, Prim 1957]

Set $T = \{a\}$

If $V(T) \neq V$ then

Let e be the minimum cost edge in $\delta(T)$

Set $T \leftarrow T \cup e$

So every chosen edge is in the MST.

Borůvka's Algorithm

Borůvka's Algorithm [Borůvka 1927]

Set $T = \emptyset$

If T has more than one component $\{S_1, S_2, \ldots, S_\ell\}$ then

For $i = \{1, 2, \ldots, \ell\}$, let e_i be the minimum cost edge in $\delta(S_i)$

Set $T \leftarrow T \cup e_1 \cup e_2 \cup \cdots \cup e_\ell$

So every chosen edge is in the MST.

- This proves that Borůvka's Algorithm cannot output a cycle!
  - This was not obvious from the description of the algorithm.

Kruskal's Algorithm

Kruskal's Algorithm [Kruskal 1956]

Sort the edges $\{e_1, e_2, \ldots, e_m\}$ by cost $c_1 < c_2 < \cdots < c_m$

Set $T = \emptyset$

For $i = \{1, 2, \ldots, m\}$

Let $e_i = (u, v)$

If u and v are in different components of T then set $T \leftarrow T \cup e_i$

Let S be one of these two components.

Then e_i is the cheapest edge in $\delta(S)$ by the cost ordering.