Minimum Spanning Trees

Hengfeng Wei

Institute of Computer Software
Nanjing University

December 12, 2016

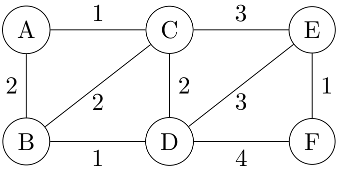
Minimum Spanning Trees

1. The MST Problem
2. The Generic MST Algorithm
3. Kruskal's and Prim's Algorithms

Minimum Spanning Trees

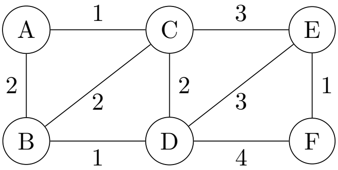
Minimum Spanning Tree

$G = (V, E)$: connected, undirected, weighted graph $(w(e))$



Minimum Spanning Tree

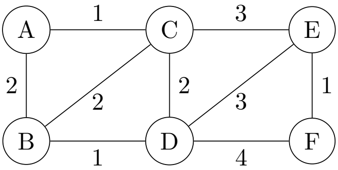
$G = (V, E)$: connected, undirected, weighted graph $(w(e))$



Spanning tree $T = (V, E' \subseteq E)$: connected, acyclic

Minimum Spanning Tree

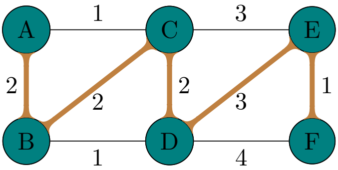
$G = (V, E)$: connected, undirected, weighted graph $(w(e))$



Spanning tree $T = (V, E' \subseteq E)$: connected, acyclic ($\Rightarrow n - 1$ edges)

Minimum Spanning Tree

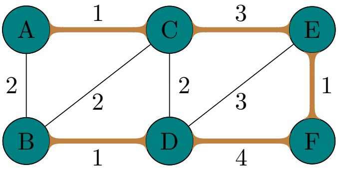
$G = (V, E)$: connected, undirected, weighted graph ($w(e)$)



Spanning tree $T = (V, E' \subseteq E)$: connected, acyclic ($\Rightarrow n - 1$ edges)

Minimum Spanning Tree

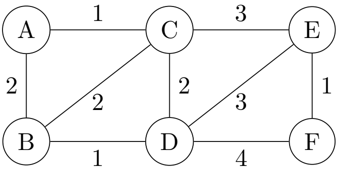
$G = (V, E)$: connected, undirected, weighted graph $(w(e))$



Spanning tree $T = (V, E' \subseteq E)$: connected, acyclic ($\Rightarrow n - 1$ edges)

Minimum Spanning Tree

$G = (V, E)$: connected, undirected, weighted graph $(w(e))$

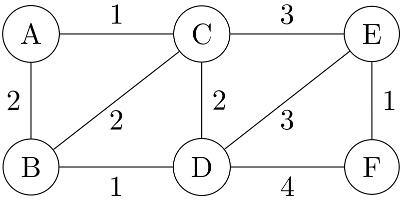


Spanning tree $T = (V, E' \subseteq E)$: connected, acyclic ($\Rightarrow n - 1$ edges)

$$w(T) = \sum_{e \in E'} w(e)$$

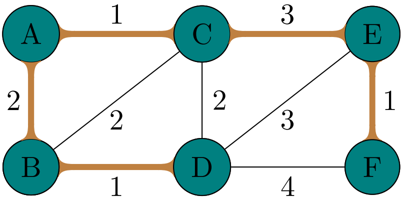
Minimum Spanning Tree

$$\text{MST:} \quad \arg\min_{T} \; w(T)$$



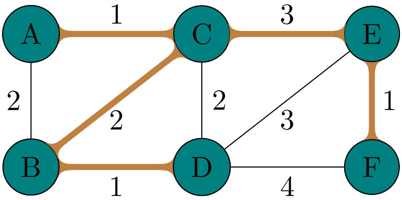
Minimum Spanning Tree

$$\text{MST:} \quad \arg\min_{T} \; w(T)$$



Minimum Spanning Tree

$$\text{MST:} \quad \arg\min_{T} \; w(T)$$



A Wrong Algorithm

Wrong divide-and-conquer algorithm for MST

Input: $G = (V, E, w)$

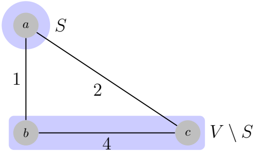
Divide: $V = (S, V \setminus S); \; ||S| - |V \setminus S|| \leq 1$

A Wrong Algorithm

Wrong divide-and-conquer algorithm for MST

Input: $G = (V, E, w)$

Divide: $V = (S, V \setminus S)$; $||S| - |V \setminus S|| \leq 1$ (Cut)



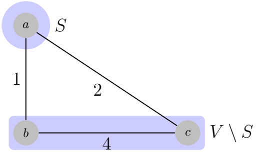
A Wrong Algorithm

Wrong divide-and-conquer algorithm for MST

Input: $G = (V, E, w)$

Divide: $V = (S, V \setminus S)$; $||S| - |V \setminus S|| \leq 1$ (Cut)

Conquer: T_1: an MST of S; T_2: an MST of $V \setminus S$



A Wrong Algorithm

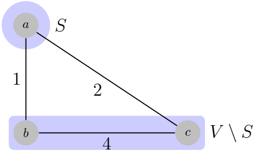
Wrong divide-and-conquer algorithm for MST

Input: $G = (V, E, w)$

Divide: $V = (S, V \setminus S)$; $||S| - |V \setminus S|| \leq 1$ (Cut)

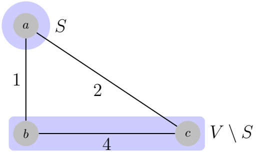
Conquer: T_1: an MST of S; T_2: an MST of $V \setminus S$

Combine: $T_1 + T_2 + \{e\}$: e is a **lightest** edge across $(S, V \setminus S)$



A Wrong Algorithm

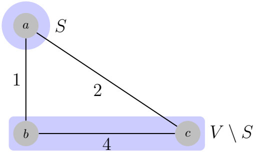
What is wrong?



The edges *bc* and *ad* do **not** belong to any MST.

A Wrong Algorithm

What is wrong?



The edges bc and ad do **not** belong to any MST.

What if:

Invariant: Manages a set of edges X which is a subset of **some** MST.

Minimum Spanning Trees

The Generic MST Algorithm

Overview: Grow the MST one edge at a time

The Generic MST Algorithm

Overview: Grow the MST one edge at a time

State: Manage a set of edges X

The Generic MST Algorithm

Overview: Grow the MST one edge at a time

State: Manage a set of edges X

Invariant: X is a subset of some MST

The Generic MST Algorithm

Overview: Grow the MST one edge at a time

State: Manage a set of edges X

Invariant: X is a subset of some MST

Init: $X = \emptyset$

The Generic MST Algorithm

Overview: Grow the MST one edge at a time

State: Manage a set of edges X

Invariant: X is a subset of some MST

Init: $X = \emptyset$

Iteration: Find an edge e *s.t.*
$X \cup \{e\}$ is also a subset of some MST

The Generic MST Algorithm

Overview: Grow the MST one edge at a time

State: Manage a set of edges X

Invariant: X is a subset of some MST

Init: $X = \emptyset$

Iteration: Find a **safe** edge e *s.t.*
$X \cup \{e\}$ is also a subset of some MST

The Generic MST Algorithm

Overview: Grow the MST one edge at a time

State: Manage a set of edges X

Invariant: X is a subset of some MST

Init: $X = \emptyset$

Iteration: Find a **safe** edge e *s.t.*
$X \cup \{e\}$ is also a subset of some MST

Termination: $(n-1)$ iterations

The Generic MST Algorithm

Overview: Grow the MST one edge at a time

State: Manage a set of edges X

Invariant: X is a subset of some MST

Init: $X = \emptyset$

Iteration: Find a **safe** edge e *s.t.*
$X \cup \{e\}$ is also a subset of some MST

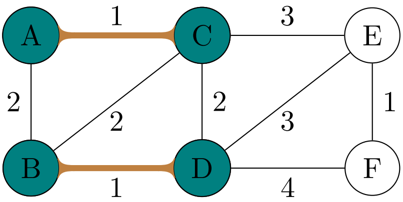
Termination: $(n-1)$ iterations

How to find a safe e for X in each iteration?

The Cut Property

Given that X is part of some MST T:

Then, $X + \{e\}$ is also a part of some MST T'.

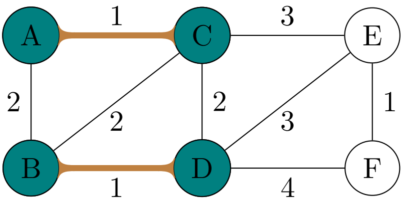


The Cut Property

Given that X is part of some MST T:

- A cut $(S, V \setminus S)$ **respecting** X (X does not cross $(S, V \setminus S)$)

Then, $X + \{e\}$ is also a part of some MST T'.

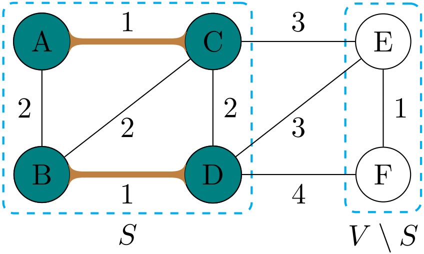


The Cut Property

Given that X is part of some MST T:

- A cut $(S, V \setminus S)$ **respecting** X (X does not cross $(S, V \setminus S)$)

Then, $X + \{e\}$ is also a part of some MST T'.

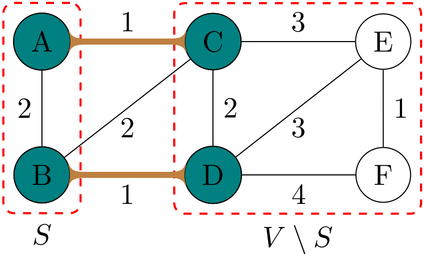


The Cut Property

Given that X is part of some MST T:

- A cut $(S, V \setminus S)$ **respecting** X (X does not cross $(S, V \setminus S)$)

Then, $X + \{e\}$ is also a part of some MST T'.

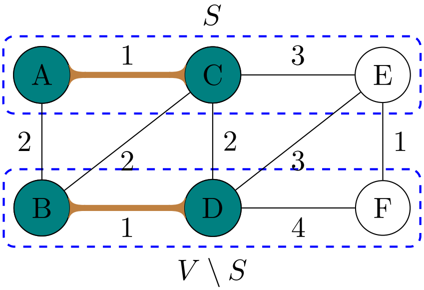


The Cut Property

Given that X is part of some MST T:

- A cut $(S, V \setminus S)$ **respecting** X (X does not cross $(S, V \setminus S)$)

Then, $X + \{e\}$ is also a part of some MST T'.

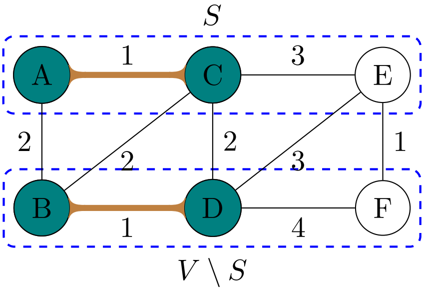


The Cut Property

Given that X is part of some MST T:

- A cut $(S, V \setminus S)$ **respecting** X (X does not cross $(S, V \setminus S)$)
- e: a lightest edge across $(S, V \setminus S)$

Then, $X + \{e\}$ is also a part of some MST T'.

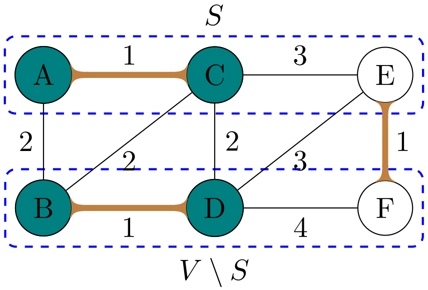


The Cut Property

Given that X is part of some MST T:

- A cut $(S, V \setminus S)$ **respecting** X (X does not cross $(S, V \setminus S)$)
- e: a lightest edge across $(S, V \setminus S)$

Then, $X + \{e\}$ is also a part of some MST T'.



The Cut Property

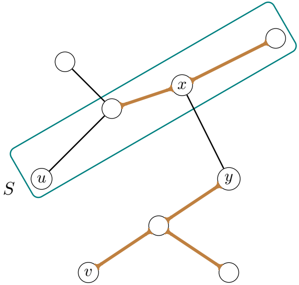
1. $X \subseteq T, e \in T \Rightarrow X + \{e\} \subseteq T$

The Cut Property

1. $X \subseteq T, e \in T \Rightarrow X + \{e\} \subseteq T$
2. $X \subseteq T, e \notin T \Rightarrow X + \{e\} \subseteq T'$.

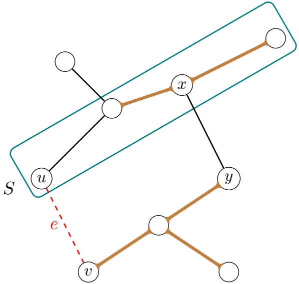
The Cut Property

1. $X \subseteq T, e \in T \Rightarrow X + \{e\} \subseteq T$
2. $X \subseteq T, e \notin T \Rightarrow X + \{e\} \subseteq T'$.



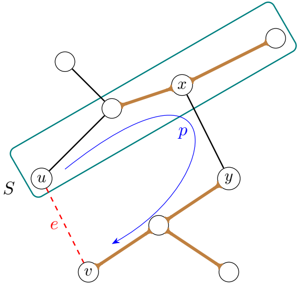
The Cut Property

1. $X \subseteq T, e \in T \Rightarrow X + \{e\} \subseteq T$
2. $X \subseteq T, e \notin T \Rightarrow X + \{e\} \subseteq T'$.



The Cut Property

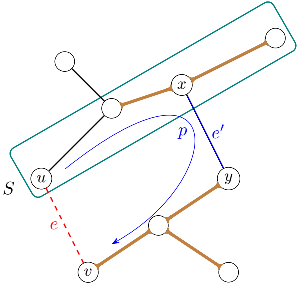
1. $X \subseteq T, e \in T \Rightarrow X + \{e\} \subseteq T$
2. $X \subseteq T, e \notin T \Rightarrow X + \{e\} \subseteq T'$.



- $T + \{e\} \Longrightarrow$ cycle C $(uv + P_{u \rightsquigarrow v})$

The Cut Property

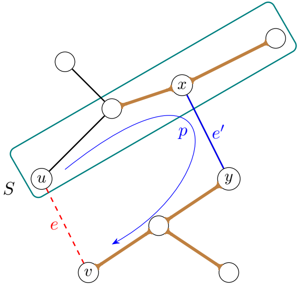
1. $X \subseteq T, e \in T \Rightarrow X + \{e\} \subseteq T$
2. $X \subseteq T, e \notin T \Rightarrow X + \{e\} \subseteq T'$.



- $T + \{e\} \Longrightarrow$ cycle C $(uv + P_{u \rightsquigarrow v})$
- $\exists e' \in P_{u \rightsquigarrow v}$: e' across the cut; $w(e') \geq w(e)$

The Cut Property

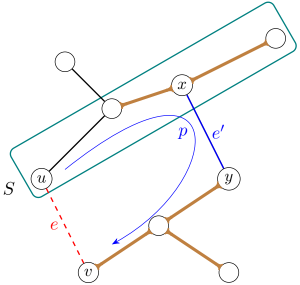
1. $X \subseteq T, e \in T \Rightarrow X + \{e\} \subseteq T$
2. $X \subseteq T, e \notin T \Rightarrow X + \{e\} \subseteq T'$.



- $T + \{e\} \Longrightarrow$ cycle C $(uv + P_{u \rightsquigarrow v})$
- $\exists e' \in P_{u \rightsquigarrow v}$: e' across the cut; $w(e') \geq w(e)$
- $T' = T + \{e\} - \{e'\}$ is an ST

The Cut Property

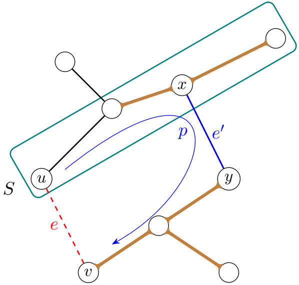
1. $X \subseteq T, e \in T \Rightarrow X + \{e\} \subseteq T$
2. $X \subseteq T, e \notin T \Rightarrow X + \{e\} \subseteq T'$.



- $T + \{e\} \Longrightarrow$ cycle C $(uv + P_{u \rightsquigarrow v})$
- $\exists e' \in P_{u \rightsquigarrow v}$: e' across the cut; $w(e') \geq w(e)$
- $T' = T + \{e\} - \{e'\}$ is an ST
- $w(T') \leq w(T) \Rightarrow T'$ is an MST

The Cut Property

1. $X \subseteq T, e \in T \Rightarrow X + \{e\} \subseteq T$
2. $X \subseteq T, e \notin T \Rightarrow X + \{e\} \subseteq T'$.



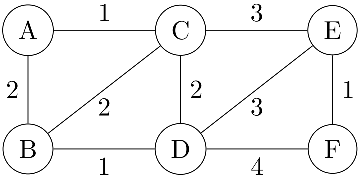
- $T + \{e\} \implies$ cycle C $(uv + P_{u \rightsquigarrow v})$
- $\exists e' \in P_{u \rightsquigarrow v}$: e' across the cut; $w(e') \geq w(e)$
- $T' = T + \{e\} - \{e'\}$ is an ST
- $w(T') \leq w(T) \Rightarrow T'$ is an MST
- $e' \notin X$(**respect**) $\Rightarrow X + \{e\} \subseteq T'$

Minimum Spanning Trees

Kruskal's Algorithm

sort (non-descreasingly) the edges E

$X = \emptyset$
for $e \in E$ in non-descreasing order
  if $X \cup \{e\}$ does not produce cycle
    $X \leftarrow X \cup \{e\}$

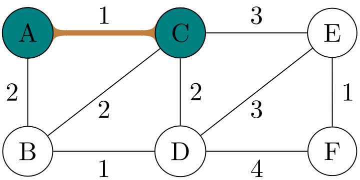


Kruskal's Algorithm

---

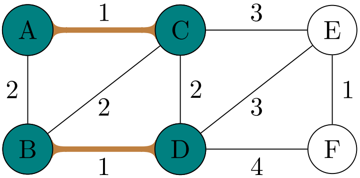
sort (non-descreasingly) the edges E

$X = \emptyset$
for $e \in E$ in non-descreasing order
  if $X \cup \{e\}$ does not produce cycle
    $X \leftarrow X \cup \{e\}$



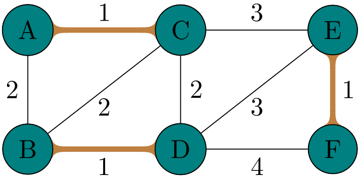
Kruskal's Algorithm

```
sort (non-descreasingly) the edges E

X = ∅
for e ∈ E in non-descreasing order
  if X ∪ {e} does not produce cycle
     X ← X ∪ {e}
```



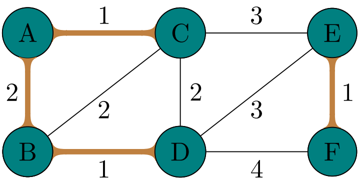
Kruskal's Algorithm

```
sort (non-descreasingly) the edges E

X = ∅
for e ∈ E in non-descreasing order
  if X ∪ {e} does not produce cycle
     X ← X ∪ {e}
```



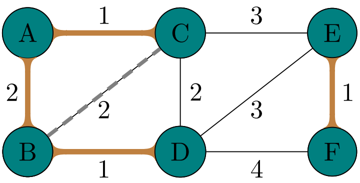
Kruskal's Algorithm

```
sort (non-descreasingly) the edges E

X = ∅
for e ∈ E in non-descreasing order
  if X ∪ {e} does not produce cycle
    X ← X ∪ {e}
```



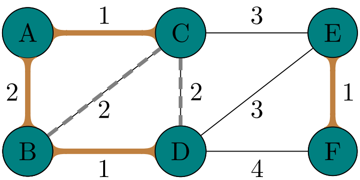
Kruskal's Algorithm

```
sort (non-descreasingly) the edges E

X = ∅
for e ∈ E in non-descreasing order
  if X ∪ {e} does not produce cycle
    X ← X ∪ {e}
```



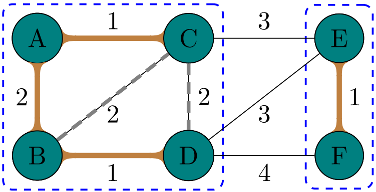
Kruskal's Algorithm

```
sort (non-descreasingly) the edges E

X = ∅
for e ∈ E in non-descreasing order
  if X ∪ {e} does not produce cycle
     X ← X ∪ {e}
```



Kruskal's Algorithm

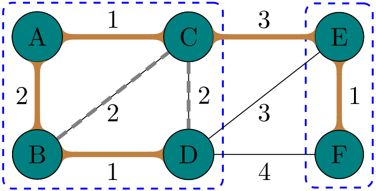
```
sort (non-descreasingly) the edges E

X = ∅
for e ∈ E in non-descreasing order
  if X ∪ {e} does not produce cycle
     X ← X ∪ {e}
```



Kruskal's Algorithm

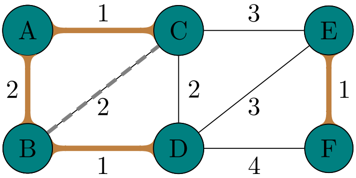
`sort (non-descreasingly) the edges` E

$X = \emptyset$
for $e \in E$ `in non-descreasing order`
  if $X \cup \{e\}$ `does not produce cycle`
    $X \leftarrow X \cup \{e\}$



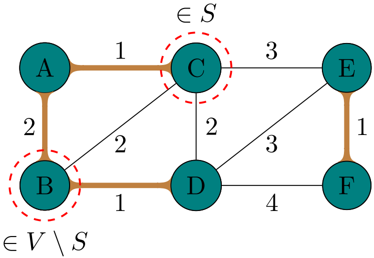
Kruskal's Algorithm

```
sort (non-descreasingly) the edges E

X = ∅
for e ∈ E in non-descreasing order
  if X ∪ {e} does not produce cycle
    X ← X ∪ {e}
```



Kruskal's Algorithm

```
sort (non-descreasingly) the edges E

X = ∅
for e ∈ E in non-descreasing order
  if X ∪ {e} does not produce cycle
     X ← X ∪ {e}
```



Kruskal's Algorithm

State: forest $\triangleq$ a collection of connected components

Ops: on connected components

- cycle detection
- union two CCs

Kruskal's Algorithm

State: forest $\triangleq$ a collection of connected components

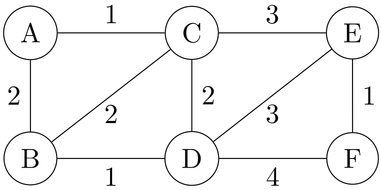
Ops: on connected components

- cycle detection
- union two CCs

Using the **disjoint-set** data structure.

Prim's Algorithm

$X = \emptyset$
$S = \{s\}$ `// pick any` $s \in V$
$R = V \setminus S$
while $R \neq \emptyset$
  $e = (u, v) \leftarrow$ `a lightest edge across` (S, R)
  $X \leftarrow X \cup \{e\}$
  $S \leftarrow S \cup \{u\} \quad R \leftarrow R \setminus \{v\}$



Prim's Algorithm

$X = \emptyset$

$S = \{s\}$ `// pick any` $s \in V$

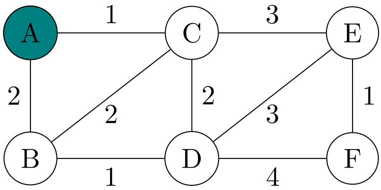
$R = V \setminus S$

while $R \neq \emptyset$

  $e = (u, v) \leftarrow$ `a lightest edge across` (S, R)

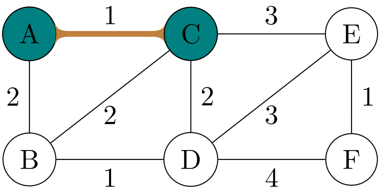
  $X \leftarrow X \cup \{e\}$

  $S \leftarrow S \cup \{u\} \quad R \leftarrow R \setminus \{v\}$



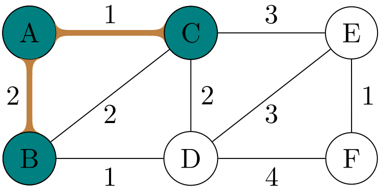
Prim's Algorithm

$X = \emptyset$
$S = \{s\}$ // pick any $s \in V$
$R = V \setminus S$
while $R \neq \emptyset$
    $e = (u, v) \leftarrow$ a lightest edge across (S, R)
    $X \leftarrow X \cup \{e\}$
    $S \leftarrow S \cup \{u\} \quad R \leftarrow R \setminus \{v\}$



Prim's Algorithm

$X = \emptyset$
$S = \{s\}$ `// pick any` $s \in V$
$R = V \setminus S$
while $R \neq \emptyset$
  $e = (u, v) \leftarrow$ `a lightest edge across` (S, R)
  $X \leftarrow X \cup \{e\}$
  $S \leftarrow S \cup \{u\} \quad R \leftarrow R \setminus \{v\}$



Prim's Algorithm

$X = \emptyset$

$S = \{s\}$ `// pick any` $s \in V$

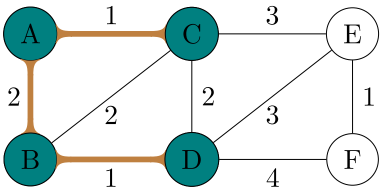
$R = V \setminus S$

while $R \neq \emptyset$

  $e = (u, v) \leftarrow$ `a lightest edge across` (S, R)

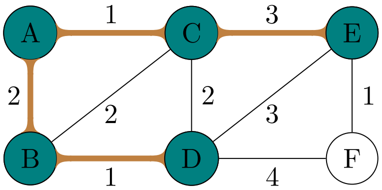
  $X \leftarrow X \cup \{e\}$

  $S \leftarrow S \cup \{u\} \quad R \leftarrow R \setminus \{v\}$



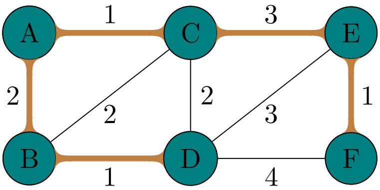
Prim's Algorithm

$X = \emptyset$
$S = \{s\}$ `// pick any` $s \in V$
$R = V \setminus S$
while $R \neq \emptyset$
    $e = (u, v) \leftarrow$ `a lightest edge across` (S, R)
    $X \leftarrow X \cup \{e\}$
    $S \leftarrow S \cup \{u\} \quad R \leftarrow R \setminus \{v\}$



Prim's Algorithm

$X = \emptyset$
$S = \{s\}$ `// pick any` $s \in V$
$R = V \setminus S$
while $R \neq \emptyset$
    $e = (u, v) \leftarrow$ `a lightest edge across` (S, R)
    $X \leftarrow X \cup \{e\}$
    $S \leftarrow S \cup \{u\} \quad R \leftarrow R \setminus \{v\}$



Prim's Algorithm

State: a growing tree (CC)

Op: identifying a lightest edge

Prim's Algorithm

State: a growing tree (CC)

Op: identifying a lightest edge

Using the **priority-queue (min-heap)** data structure.