Greedy Algorithms: Exchange Property

Algorithm Design & Analysis

Spring 2018
Outline

Greedy Algorithms: Exchange Property

Minimum-Cost Spanning Trees
  Maximum Greed: Kruskal’s Algorithm
  Analysis: Kruskal’s Algorithm
  Moderate Greed: Prim’s Algorithm
  Analysis: Prim’s Algorithm
  Reverse Greed: Reverse-Delete Algorithm
  Allowing Non-Unique Edge Costs
The Exchange Property

**Exchange Property:** Show that an optimal solution can be sequentially transformed into a greedy solution without compromising optimality.
Greedy Algorithms: Exchange Property

Outline

Greedy Algorithms: Exchange Property

Minimum-Cost Spanning Trees
  Maximum Greed: Kruskal’s Algorithm
  Analysis: Kruskal’s Algorithm
  Moderate Greed: Prim’s Algorithm
  Analysis: Prim’s Algorithm
  Reverse Greed: Reverse-Delete Algorithm
  Allowing Non-Unique Edge Costs
Minimum-Cost Spanning Trees

Figure: A Graph $G$ with Positive Edge-Weights
Minimum-Cost Spanning Trees

Figure: A Min-Cost Spanning Tree for $G$
Minimum-Cost Spanning Trees

Computing a minimum-cost spanning tree for a graph has many applications
Minimum-Cost Spanning Trees

Computing a minimum-cost spanning tree for a graph has many applications

- Classic Application: Underground Cable (Power, Telecom, ...)

---
Minimum-Cost Spanning Trees

Computing a minimum-cost spanning tree for a graph has many applications:

- Classic Application: Underground Cable (Power, Telecom, ...)
- Efficient broadcasting on a computer network (Note: different from shortest paths)
Minimum-Cost Spanning Trees

Computing a minimum-cost spanning tree for a graph has many applications

- Classic Application: Underground Cable (Power, Telecom, ...)
- Efficient broadcasting on a computer network (Note: different from shortest paths)
- Taxonomy (mental maps)
Minimum-Cost Spanning Trees

Computing a minimum-cost spanning tree for a graph has many applications

- Classic Application: Underground Cable (Power, Telecom, ...)
- Efficient broadcasting on a computer network (Note: different from shortest paths)
- Taxonomy (mental maps)
- Reliable subnetwork
Computing a minimum-cost spanning tree for a graph has many applications

- Classic Application: Underground Cable (Power, Telecom, ...)
- Efficient broadcasting on a computer network (Note: different from shortest paths)
- Taxonomy (mental maps)
- Reliable subnetwork
- Approximate solutions to harder problems, such as TSP
The Problem
**The Problem**

*Definition*

The *cost* of a subgraph $G' = (V', E')$ of a graph $G = (V, E)$ with edge-costs is the sum of the costs of the edges of $E'$. 
The Problem

**Definition**
The cost of a subgraph $G' = (V', E')$ of a graph $G = (V, E)$ with edge-costs is the sum of the costs of the edges of $E'$.

**Observation**
A minimum-cost connected spanning subgraph for a connected graph $G = (V, E)$ with positive edge costs $c()$ is a tree.
The Problem

Definition
The cost of a subgraph \( G' = (V', E') \) of a graph \( G = (V, E) \) with edge-costs is the sum of the costs of the edges of \( E' \).

Observation
A minimum-cost connected spanning subgraph for a connected graph \( G = (V, E) \) with positive edge costs \( c() \) is a tree.

Definition
A spanning tree \( T \) of a graph \( G = (V, E) \) with edge costs is minimum-cost if no other spanning tree has lower cost.
The Problem

Definition
The cost of a subgraph $G' = (V', E')$ of a graph $G = (V, E)$ with edge-costs is the sum of the costs of the edges of $E'$.

Observation
A minimum-cost connected spanning subgraph for a connected graph $G = (V, E)$ with positive edge costs $c()$ is a tree.

Definition
A spanning tree $T$ of a graph $G = (V, E)$ with edge costs is minimum-cost if no other spanning tree has lower cost.

We will assume that all edge-costs are distinct; we’ll relax this assumption at the end of class.
Cuts and Cut-Edges
Cuts and Cut-Edges

A cut in a graph $G = (V, E)$ is a partition of $V$ into two sets $\{S, V - S\}$. The edges $E(S, V - S)$ with one endpoint in each set are called cut edges.
**Definition**

A *cut* in a graph $G = (V, E)$ is a partition of $V$ into two sets $\{S, V - S\}$. The edges $E(S, V - S)$ with one endpoint in each set are called *cut edges.*
Trees, Cycles, & Cuts

Trees, cycles, and cuts relate to one another in useful ways.
Trees, Cycles, & Cuts

Trees, cycles, and cuts relate to one another in useful ways.

Let $G = (V, E)$ be a graph and $T$ be a spanning tree of $G$.
Trees, Cycles, & Cuts

Trees, cycles, and cuts relate to one another in useful ways.

Let $G = (V, E)$ be a graph and $T$ be a spanning tree of $G$.

Observations:

• Every edge $e$ of $T$ defines a cut in $G$ that has $e$ as a cut edge.

• $S$ and $V - S$ are the vertex sets of the (two) components of $T - \{e\}$.

• Adding an edge $e$ of $G - T$ to $T$ creates a unique cycle in $T + \{e\}$; the cycle contains $e$.

• For any cycle $C$ and cut $\{S, V - S\}$, $|E(C) \setminus E(S, V - S)|$ is even.

• That is, any cycle and any cut share an even number of edges.

• So if a cycle intersects a cut, they share at least two edges.
Trees, Cycles, & Cuts

Trees, cycles, and cuts relate to one another in useful ways.

Let $G = (V, E)$ be a graph and $T$ be a spanning tree of $G$.

Observations:

- Every edge $e$ of $T$ defines a cut in $G$ that has $e$ as a cut edge.
Trees, Cycles, & Cuts

Trees, cycles, and cuts relate to one another in useful ways.

Let $G = (V, E)$ be a graph and $T$ be a spanning tree of $G$

Observations:

• Every edge $e$ of $T$ defines a cut in $G$ that has $e$ as a cut edge
  • $S$ and $V - S$ are the vertex sets of the (two) components of $T - \{e\}$
Trees, Cycles, & Cuts

Trees, cycles, and cuts relate to one another in useful ways.

Let $G = (V, E)$ be a graph and $T$ be a spanning tree of $G$

Observations:

• Every edge $e$ of $T$ defines a cut in $G$ that has $e$ as a cut edge
  • $S$ and $V - S$ are the vertex sets of the (two) components of $T - \{e\}$
  • Adding an edge $e$ of $G - T$ to $T$ creates a unique cycle in $T + \{e\}$; the cycle contains $e$
Trees, Cycles, & Cuts

Trees, cycles, and cuts relate to one another in useful ways.

Let $G = (V, E)$ be a graph and $T$ be a spanning tree of $G$.

Observations:

- Every edge $e$ of $T$ defines a cut in $G$ that has $e$ as a cut edge.
  - $S$ and $V - S$ are the vertex sets of the (two) components of $T - \{e\}$.
- Adding an edge $e$ of $G - T$ to $T$ creates a unique cycle in $T + \{e\}$; the cycle contains $e$.
- For any cycle $C$ and cut $\{S, V - S\}$, $|E(C) \cap E(S, V - S)|$ is even.
Trees, Cycles, & Cuts

Trees, cycles, and cuts relate to one another in useful ways.

Let $G = (V, E)$ be a graph and $T$ be a spanning tree of $G$.

Observations:

- Every edge $e$ of $T$ defines a cut in $G$ that has $e$ as a cut edge.
  - $S$ and $V - S$ are the vertex sets of the (two) components of $T - \{e\}$.
- Adding an edge $e$ of $G - T$ to $T$ creates a unique cycle in $T + \{e\}$; the cycle contains $e$.
- For any cycle $C$ and cut $\{S, V - S\}$, $|E(C) \cap E(S, V - S)|$ is even.
  - That is, any cycle and any cut share an even number of edges.
Trees, Cycles, & Cuts

Trees, cycles, and cuts relate to one another in useful ways.

Let $G = (V, E)$ be a graph and $T$ be a spanning tree of $G$. 

Observations:

- Every edge $e$ of $T$ defines a cut in $G$ that has $e$ as a cut edge.
  - $S$ and $V - S$ are the vertex sets of the (two) components of $T - \{e\}$

- Adding an edge $e$ of $G - T$ to $T$ creates a unique cycle in $T + \{e\}$; the cycle contains $e$.

- For any cycle $C$ and cut $\{S, V - S\}$, $|E(C) \cap E(S, V - S)|$ is even.
  - That is, any cycle and any cut share an even number of edges.
  - So if a cycle intersects a cut, they share at least two edges.
Observations:

• If $T$ is a MCST of $G$ and $e \in E(G) - E(T)$, then $e$ is the highest cost edge on the unique cycle in $T + \{e\}$.

• For any cut in $G$, its lowest-cost edge is in every MCST of $G$.

• For any cycle in $G$, its highest-cost edge is in no MCST of $G$. 
Observations:

- If $T$ is a MCST of $G$ and $e \in E(G) - E(T)$, then $e$ is the highest cost edge on the unique cycle in $T + \{e\}$. 
Observations:

- If $T$ is a MCST of $G$ and $e \in E(G) - E(T)$, then $e$ is the highest cost edge on the unique cycle in $T + \{e\}$.
- For any cut in $G$, its lowest-cost edge is in every MCST of $G$. 
Properties of Min-Cost Spanning Trees

Observations:

- If \( T \) is a MCST of \( G \) and \( e \in E(G) - E(T) \), then \( e \) is the highest cost edge on the unique cycle in \( T + \{ e \} \).
- For any cut in \( G \), its lowest-cost edge is in every MCST of \( G \).
- For any cycle in \( G \), its highest-cost edge is in no MCST of \( G \).
Proof of Cut Property

For any cut in $G$, its lowest-cost edge is in every MCST of $G$.

Proof.
Proof of Cut Property

For any cut in $G$, its lowest-cost edge is in every MCST of $G$.

Proof.

• Let $T$ be any MCST of $G$, let $\{S, V - S\}$ be any cut of $G$, and let $e$ be the cheapest edge of the cut.
Proof of Cut Property

For any cut in $G$, its lowest-cost edge is in every MCST of $G$.

Proof.

- Let $T$ be any MCST of $G$, let $\{S, V - S\}$ be any cut of $G$, and let $e$ be the cheapest edge of the cut.
- If $e \not\in T$, then $T + \{e\}$ contains a unique cycle $C$, and $e \in C$. 
Proof of Cut Property

For any cut in $G$, its lowest-cost edge is in every MCST of $G$.

Proof.

- Let $T$ be any MCST of $G$, let $\{S, V - S\}$ be any cut of $G$, and let $e$ be the cheapest edge of the cut.
- If $e \not\in T$, then $T + \{e\}$ contains a unique cycle $C$, and $e \in C$.
- So $C$ contains another edge $e' \in E(S, V - S)$. 

\[\square\]
Proof of Cut Property

For any cut in $G$, its lowest-cost edge is in every MCST of $G$.

Proof.

- Let $T$ be any MCST of $G$, let $\{S, V - S\}$ be any cut of $G$, and let $e$ be the cheapest edge of the cut.
- If $e \notin T$, then $T + \{e\}$ contains a unique cycle $C$, and $e \in C$.
- So $C$ contains another edge $e' \in E(S, V - S)$.
- But $T + \{e\} - \{e'\}$ is a tree with lower cost than $T \Rightarrow \Leftarrow$.  

Proof of Cycle Property

For any cycle in $G$, its highest-cost edge is in no MCST of $G$.

*Proof.*
Proof of Cycle Property

For any cycle in $G$, its highest-cost edge is in no MCST of $G$.

Proof.

• Suppose tree $T$ contains the highest-cost edge $e$ of cycle $C$. 
Proof of Cycle Property

For any cycle in $G$, its highest-cost edge is in no MCST of $G$.

Proof.

- Suppose tree $T$ contains the highest-cost edge $e$ of cycle $C$.
- Let $\{S, V - S\}$ be the cut obtained by removing $e$ from $T$. 
Proof of Cycle Property

For any cycle in $G$, its highest-cost edge is in no MCST of $G$.

Proof.

- Suppose tree $T$ contains the highest-cost edge $e$ of cycle $C$.
- Let $\{S, V - S\}$ be the cut obtained by removing $e$ from $T$.
- $e \in C \cap E(S, V \setminus S)$, so $|C \cap E(S, V \setminus S)| > 0$.  


Proof of Cycle Property

For any cycle in $G$, its highest-cost edge is in no MCST of $G$.

Proof.

- Suppose tree $T$ contains the highest-cost edge $e$ of cycle $C$.
- Let $\{S, V - S\}$ be the cut obtained by removing $e$ from $T$.
- $e \in C \cap E(S, V \setminus S)$, so $|C \cap E(S, V \setminus S)| > 0$.
- So $C$ contains another cut edge $e'$ of $\{S, V - S\}$.


**Proof of Cycle Property**

For any cycle in $G$, its highest-cost edge is in *no* MCST of $G$.

*Proof.*

- Suppose tree $T$ contains the highest-cost edge $e$ of cycle $C$.
- Let $\{S, V - S\}$ be the cut obtained by removing $e$ from $T$.
- $e \in C \cap E(S, V \setminus S)$, so $|C \cap E(S, V \setminus S)| > 0$.
- So $C$ contains another cut edge $e'$ of $\{S, V - S\}$.
- And $c(e') < c(e)$. 


Proof of Cycle Property

For any cycle in $G$, its highest-cost edge is in no MCST of $G$.

Proof.

- Suppose tree $T$ contains the highest-cost edge $e$ of cycle $C$.
- Let $\{S, V - S\}$ be the cut obtained by removing $e$ from $T$.
- $e \in C \cap E(S, V \setminus S)$, so $|C \cap E(S, V \setminus S)| > 0$.
- So $C$ contains another cut edge $e'$ of $\{S, V - S\}$.
- And $c(e') < c(e)$.
- So, $T - \{e\} + \{e'\}$ is a spanning tree cheaper than $T$. 
Proof of Cycle Property

For any cycle in $G$, its highest-cost edge is in no MCST of $G$.

Proof.

- Suppose tree $T$ contains the highest-cost edge $e$ of cycle $C$.
- Let $\{S, V - S\}$ be the cut obtained by removing $e$ from $T$.
- $e \in C \cap E(S, V \setminus S)$, so $|C \cap E(S, V \setminus S)| > 0$.
- So $C$ contains another cut edge $e'$ of $\{S, V - S\}$.
- And $c(e') < c(e)$.
- So, $T - \{e\} + \{e'\}$ is a spanning tree cheaper than $T$.
- So $T$ is not a MCST of $G$.  

\[\square\]
Proof of Cycle Property

For any cycle in $G$, its highest-cost edge is in no MCST of $G$.

Proof.

- Suppose tree $T$ contains the highest-cost edge $e$ of cycle $C$.
- Let $\{S, V - S\}$ be the cut obtained by removing $e$ from $T$.
- $e \in C \cap E(S, V \setminus S)$, so $|C \cap E(S, V \setminus S)| > 0$.
- So $C$ contains another cut edge $e'$ of $\{S, V - S\}$.
- And $c(e') < c(e)$.
- So, $T - \{e\} + \{e'\}$ is a spanning tree cheaper than $T$.
- So $T$ is not a MCST of $G$.

\[\square\]

Exchange Property!
Maximum Greed: Kruskal’s Algorithm

Idea: Add cheapest remaining edge that don’t create a cycle
Maximum Greed: Kruskal’s Algorithm

Idea: Add cheapest remaining edge that don’t create a cycle

Algorithm 2 Kruskal’s Algorithm

procedure Kruskal(G, c())
    ⊲ G = (V, E) is connected
    T ← (V, ∅) ⊲ The eventual MCST
    F ← E
    while |E(T)| < |V| − 1 do
        Remove cheapest edge e ∈ F from F
        if T + {e} does not contain a cycle then
            Add e to T
    end procedure
Proof of Correctness of Kruskal

Theorem

Kruskal produces a minimum-cost spanning tree of $G$. 
Proof of Correctness of Kruskal

Theorem

Kruskal produces a minimum-cost spanning tree of $G$.

The proof has two parts

1. Show $T$ is a tree by showing it has no cycles and is connected
2. Show $T$ is minimum-cost by showing each of its edges is contained in every MCST

$T$ has no cycles and is connected:

- $T$ is a forest at all times: new edges don't create cycles
- If $T$ is not connected at top of loop, then $|E(T)| < |V| - 1$, so loop repeats
- If $S$ is the vertex set of a connected component of $T$ then $\{S, V - S\}$ is a cut of $G$.
- $G$ is connected, so $E(S, V - S) \neq \emptyset$, so $F \neq \emptyset$
- But $|F|$ decreases at each iteration, so loop must stop repeating, so $T$ is a tree
Proof of Correctness of Kruskal

Theorem

Kruskal produces a minimum-cost spanning tree of $G$.

The proof has two parts

- Show $T$ is a tree by showing it has no cycles and is connected

• If $T$ is not connected at top of loop, then $|E(T)| < |V| - 1$, so loop repeats

• $G$ is connected, so $E(S, V - S) \neq \emptyset$, so $F \neq \emptyset$

• But $|F|$ decreases at each iteration, so loop must stop repeating, so $T$ is a tree
Proof of Correctness of Kruskal

Theorem
Kruskal produces a minimum-cost spanning tree of $G$.

The proof has two parts

- Show $T$ is a tree by showing it has no cycles and is connected
- Show $T$ is minimum-cost by showing each of its edges is contained in every MCST
Proof of Correctness of Kruskal

Theorem
Kruskal produces a minimum-cost spanning tree of $G$.

The proof has two parts
- Show $T$ is a tree by showing it has no cycles and is connected
- Show $T$ is minimum-cost by showing each of its edges is contained in every MCST

$T$ has no cycles and is connected:
Proof of Correctness of Kruskal

Theorem
Kruskal produces a minimum-cost spanning tree of $G$.

The proof has two parts
- Show $T$ is a tree by showing it has no cycles and is connected
- Show $T$ is minimum-cost by showing each of its edges is contained in every MCST

$T$ has no cycles and is connected:
- $T$ is a forest at all times: new edges don’t create cycles
Proof of Correctness of Kruskal

Theorem

Kruskal produces a minimum-cost spanning tree of $G$.

The proof has two parts

- Show $T$ is a tree by showing it has no cycles and is connected
- Show $T$ is minimum-cost by showing each of its edges is contained in every MCST

$T$ has no cycles and is connected:

- $T$ is a forest at all times: new edges don’t create cycles
- If $T$ is not connected at top of loop, then $|E(T)| < |V| - 1$, so loop repeats
Proof of Correctness of Kruskal

Theorem

Kruskal produces a minimum-cost spanning tree of $G$.

The proof has two parts

- Show $T$ is a tree by showing it has no cycles and is connected
- Show $T$ is minimum-cost by showing each of its edges is contained in every MCST

$T$ has no cycles and is connected:

- $T$ is a forest at all times: new edges don’t create cycles
- If $T$ is not connected at top of loop, then $|E(T)| < |V| - 1$, so loop repeats
- If $S$ is the vertex set of a connected component of $T$ then \( \{S, V - S\} \) is a cut of $G$.  

Proof of Correctness of Kruskal

Theorem
Kruskal produces a minimum-cost spanning tree of G.

The proof has two parts
• Show T is a tree by showing it has no cycles and is connected
• Show T is minimum-cost by showing each of its edges is contained in every MCST

T has no cycles and is connected:
• T is a forest at all times: new edges don’t create cycles
• If T is not connected at top of loop, then |E(T)| < |V| − 1, so loop repeats
• If S is the vertex set of a connected component of T then {S, V − S} is a cut of G.
• G is connected, so E(S, V − S) ≠ ∅, so F ≠ ∅
Proof of Correctness of Kruskal

Theorem
Kruskal produces a minimum-cost spanning tree of $G$.

The proof has two parts

- Show $T$ is a tree by showing it has no cycles and is connected
- Show $T$ is minimum-cost by showing each of its edges is contained in every MCST

$T$ has no cycles and is connected:

- $T$ is a forest at all times: new edges don’t create cycles
- If $T$ is not connected at top of loop, then $|E(T)| < |V| - 1$, so loop repeats
- If $S$ is the vertex set of a connected component of $T$ then \{$S, V - S$\} is a cut of $G$.
- $G$ is connected, so $E(S, V - S) \neq \emptyset$, so $F \neq \emptyset$
- But $|F|$ decreases at each iteration, so loop must stop repeating, so $T$ is a tree
Proof of Correctness of Kruskal

$T$ is an MCST:
Proof of Correctness of Kruskal

$T$ is an MCST:

- Let $e = \{u, v\}$ be an edge selected by Kruskal.
**Proof of Correctness of Kruskal**

$T$ is an MCST:

- Let $e = \{u, v\}$ be an edge selected by Kruskal.
- Let $S \subseteq V$ be the set of vertices reachable from $u$ in $T$ just before $e$ was added to $T$. 
Proof of Correctness of Kruskal

$T$ is an MCST:

- Let $e = \{u, v\}$ be an edge selected by Kruskal.
- Let $S \subseteq V$ be the set of vertices reachable from $u$ in $T$ just before $e$ was added to $T$.
- At this point $T$ contains no edge from $S$ to $V - S$
Proof of Correctness of Kruskal

$T$ is an MCST:

- Let $e = \{u, v\}$ be an edge selected by Kruskal.
- Let $S \subset V$ be the set of vertices reachable from $u$ in $T$ just before $e$ was added to $T$.
- At this point $T$ contains no edge from $S$ to $V - S$.
- So, $e$ is the cheapest cut edge of $E(S, V - S)$ in $G$. 
**Proof of Correctness of Kruskal**

$T$ is an MCST:

- Let $e = \{u, v\}$ be an edge selected by Kruskal.
- Let $S \subset V$ be the set of vertices reachable from $u$ in $T$ just before $e$ was added to $T$.
- At this point $T$ contains no edge from $S$ to $V - S$.
- So, $e$ is the cheapest cut edge of $E(S, V - S)$ in $G$.
- So $e$ is part of every MCST of $G$. 
Proof of Correctness of Kruskal

$T$ is an MCST:

- Let $e = \{u, v\}$ be an edge selected by Kruskal.
- Let $S \subseteq V$ be the set of vertices reachable from $u$ in $T$ just before $e$ was added to $T$.
- At this point $T$ contains no edge from $S$ to $V - S$.
- So, $e$ is the cheapest cut edge of $E(S, V - S)$ in $G$.
- So $e$ is part of every MCST of $G$.
- So every edge of $T$ is in every MCST of $G$. 
Proof of Correctness of Kruskal

$T$ is an MCST:

- Let $e = \{u, v\}$ be an edge selected by Kruskal.
- Let $S \subseteq V$ be the set of vertices reachable from $u$ in $T$ just before $e$ was added to $T$.
- At this point $T$ contains no edge from $S$ to $V - S$.
- So, $e$ is the cheapest cut edge of $E(S, V - S)$ in $G$.
- So $e$ is part of every MCST of $G$.
- So every edge of $T$ is in every MCST of $G$.
- So $T$ is the only MCST of $G$!
Proof of Correctness of Kruskal

$T$ is an MCST:
- Let $e = \{u, v\}$ be an edge selected by Kruskal.
- Let $S \subseteq V$ be the set of vertices reachable from $u$ in $T$ just before $e$ was added to $T$.
- At this point $T$ contains no edge from $S$ to $V - S$.
- So, $e$ is the cheapest cut edge of $E(S, V - S)$ in $G$.
- So $e$ is part of every MCST of $G$.
- So every edge of $T$ is in every MCST of $G$.
- So $T$ is the only MCST of $G$!

Corollary

A graph without repeated edge lengths has a unique MCST.
Moderate Greed: Prim’s Algorithm

Here $T$ is at tree at all times—the cheapest tree on the subgraph of $G$ that is spans.
Moderate Greed: Prim’s Algorithm

Here $T$ is at tree at all times—the cheapest tree on the subgraph of $G$ that is spans.
Prim maintains a subtree $T = (V', E')$ of $G$ and adds the cheapest cut edge of $E(V', V - V')$ (in $G$) to $T$. 
Moderate Greed: Prim’s Algorithm

Here \( T \) is at tree at all times—the cheapest tree on the subgraph of \( G \) that is spans.
Prim maintains a subtree \( T = (V', E') \) of \( G \) and adds the cheapest cut edge of \( E(V', V - V') \) (in \( G \)) to \( T \).

Algorithm 5 Prim’s Algorithm

\[
\text{procedure } \text{Prim}(G, c()) \quad \triangleright \ G = (V, E) \text{ is connected}
\]
Select some \( v \in V; \ V' \leftarrow \{v\}; \ T \leftarrow (V', \emptyset) \quad \triangleright \text{The eventual MCST}

\[
\text{while } |E(T)| < |V| - 1 \text{ do}
\]
Select cheapest edge \( e \in E(V', V - V') \)
Add \( e \) to \( T \) \quad \triangleright \text{This adds a new vertex to } T

end procedure
Analysis of Prim’s Algorithm

Theorem
Prim produces a minimum-cost spanning tree of G.

Proof.
Analysis of Prim’s Algorithm

Theorem

Prim produces a minimum-cost spanning tree of $G$.

Proof.

• $T$ is a tree at all times, and $T$ eventually must span $G$, since $G$ is connected.
**Theorem**

*Prim produces a minimum-cost spanning tree of G.*

**Proof.**

- $T$ is a tree at all times, and $T$ eventually must span $G$, since $G$ is connected.
- The next edge added to $T$ is, in $G$, the cheapest cut edge for some cut.
Analysis of Prim’s Algorithm

Theorem

Prim produces a minimum-cost spanning tree of G.

Proof.

- T is a tree at all times, and T eventually must span G, since G is connected.
- The next edge added to T is, in G, the cheapest cut edge for some cut.
- That edge, therefore, must be in every MCST.
Analysis of Prim’s Algorithm

Theorem
Prim produces a minimum-cost spanning tree of G.

Proof.

• \( T \) is a tree at all times, and \( T \) eventually must span \( G \), since \( G \) is connected.
• The next edge added to \( T \) is, in \( G \), the cheapest cut edge for some cut.
• That edge, therefore, must be in every MCST
• As in Kruskal proof, \( T \) is the only MCST of \( G \)
**Reverse-Delete Algorithm**

We can also construct an MCST by throwing away all of the most expensive edges.
We can also construct an MCST by throwing away all of the most expensive edges.

Algorithm 7 Reverse-Delete Algorithm

\begin{algorithm}
\begin{algorithmic}
\Procedure{ReverseDelete}{G, c()} \Comment{G = (V, E) is connected}
\While{|E(G)| > |V| - 1}
\State Select most expensive edge $e \in G$ that does not disconnect $G$
\State Remove $e$ from $G$
\EndWhile
\EndProcedure
\end{algorithmic}
\end{algorithm}
We can also construct an MCST by throwing away all of the most expensive edges.

**Algorithm 8** Reverse-Delete Algorithm

```plaintext
procedure REVERSEDELETE(G, c())

▷ G = (V, E) is connected

while |E(G)| > |V| − 1 do

    Select most expensive edge e ∈ G that does not disconnect G

    Remove e from G

end procedure
```

When might you ever want to use this algorithm?
Relaxing Assumption of Distinct Edge-Costs

Suppose $G$ does not have distinct edge costs.
Relaxing Assumption of Distinct Edge-Costs

Suppose $G$ does not have distinct edge costs.

Idea: Perturbation
Relaxing Assumption of Distinct Edge-Costs

Suppose $G$ does not have distinct edge costs.

Idea: Perturbation

- For each set of edges having identical costs, perturb their costs by distinct positive values.
Relaxing Assumption of Distinct Edge-Costs

Suppose $G$ does not have distinct edge costs.

Idea: Perturbation

- For each set of edges having identical costs, perturb their costs by distinct positive values
- Ensure that the sum of all of the perturbations is tiny compared to the actual edge costs.
Suppose $G$ does not have distinct edge costs. 

Idea: Perturbation

- For each set of edges having identical costs, perturb their costs by distinct positive values
- Ensure that the sum of all of the perturbations is tiny compared to the actual edge costs.
- Every spanning tree $T^*$ of perturbed graph $G^*$ corresponds to a spanning tree $T$ of $G$
Relaxing Assumption of Distinct Edge-Costs

Suppose $G$ does not have distinct edge costs.

Idea: Perturbation

- For each set of edges having identical costs, perturb their costs by distinct positive values.
- Ensure that the sum of all of the perturbations is tiny compared to the actual edge costs.
- Every spanning tree $T^*$ of perturbed graph $G^*$ corresponds to a spanning tree $T$ of $G$.
- Correspondence preserves relative cost: $c(T_1^*) \leq c(T_2^*)$ iff $c(T_1) \leq c(T_2)$.
Suppose $G$ does not have distinct edge costs.

Idea: Perturbation

- For each set of edges having identical costs, perturb their costs by distinct positive values.
- Ensure that the sum of all of the perturbations is tiny compared to the actual edge costs.
- Every spanning tree $T^*$ of perturbed graph $G^*$ corresponds to a spanning tree $T$ of $G$.
- Correspondence preserves relative cost: $c(T_1^*) \leq c(T_2^*)$ iff $c(T_1) \leq c(T_2)$.
- So $T^*$ is an MCST of $G^*$ iff $T$ is an MCST of $G$. 

Relaxing Assumption of Distinct Edge-Costs