Outline

1. **Minimum Spanning Tree**
   - Kruskal’s Algorithm
   - Reverse-Kruskal’s Algorithm
   - Prim’s Algorithm

2. **Single Source Shortest Paths**
   - Dijkstra’s Algorithm

3. **Shortest Paths in Graphs with Negative Weights**

4. **All-Pair Shortest Paths and Floyd-Warshall**
Two Methods to Build a MST

1. Start from $F \leftarrow \emptyset$, and add edges to $F$ one by one until we obtain a spanning tree.

Q: Which edge can be safely excluded from the MST?

A: The heaviest non-bridge edge.

Def. A bridge is an edge whose removal disconnects the graph.
Two Methods to Build a MST

1. Start from $F \leftarrow \emptyset$, and add edges to $F$ one by one until we obtain a spanning tree.

2. Start from $F \leftarrow E$, and remove edges from $F$ one by one until we obtain a spanning tree.
Two Methods to Build a MST

1. Start from $F \leftarrow \emptyset$, and add edges to $F$ one by one until we obtain a spanning tree.
2. Start from $F \leftarrow E$, and remove edges from $F$ one by one until we obtain a spanning tree.

Q: Which edge can be safely excluded from the MST?
Two Methods to Build a MST

1. Start from $F \leftarrow \emptyset$, and add edges to $F$ one by one until we obtain a spanning tree.

2. Start from $F \leftarrow E$, and remove edges from $F$ one by one until we obtain a spanning tree.

Q: Which edge can be safely excluded from the MST?

A: The heaviest non-bridge edge.
Two Methods to Build a MST

1. Start from $F \leftarrow \emptyset$, and add edges to $F$ one by one until we obtain a spanning tree.

2. Start from $F \leftarrow E$, and remove edges from $F$ one by one until we obtain a spanning tree.

Q: Which edge can be safely excluded from the MST?

A: The heaviest non-bridge edge.

Def. A bridge is an edge whose removal disconnects the graph.
Lemma  It is safe to exclude the heaviest non-bridge edge: there is an MST that does not contain the heaviest non-bridge edge.
Reverse Kruskal’s Algorithm

**MST-Greedy**\((G, w)\)

1. \( F \leftarrow E \)
2. sort \( E \) in non-increasing order of weights
3. for every \( e \) in this order do
4. \[ \text{if } (V, F \setminus \{e\}) \text{ is connected then} \]
5. \[ F \leftarrow F \setminus \{e\} \]
6. return \( (V, F) \)
Reverse Kruskal’s Algorithm: Example
Reverse Kruskal’s Algorithm: Example
Reverse Kruskal’s Algorithm: Example
Reverse Kruskal’s Algorithm: Example
Reverse Kruskal’s Algorithm: Example
Reverse Kruskal’s Algorithm: Example
Reverse Kruskal’s Algorithm: Example
Reverse Kruskal’s Algorithm: Example
Reverse Kruskal’s Algorithm: Example
Reverse Kruskal’s Algorithm: Example

Diagram of a graph with vertices labeled a, b, c, d, e, f, g, h, and i. The edges and their weights are as follows:
- a to b: 5
- b to c: 8
- c to i: 2
- i to g: 6
- g to f: 3
- f to e: 10
- e to d: 9
Reverse Kruskal’s Algorithm: Example
Reverse Kruskal’s Algorithm: Example
Reverse Kruskal’s Algorithm: Example
Reverse Kruskal’s Algorithm: Example
Reverse Kruskal’s Algorithm: Example
Reverse Kruskal’s Algorithm: Example
Reverse Kruskal’s Algorithm: Example
Reverse Kruskal’s Algorithm: Example
Outline

1. Minimum Spanning Tree
   - Kruskal’s Algorithm
   - Reverse-Kruskal’s Algorithm
   - Prim’s Algorithm

2. Single Source Shortest Paths
   - Dijkstra’s Algorithm

3. Shortest Paths in Graphs with Negative Weights

4. All-Pair Shortest Paths and Floyd-Warshall
Recall the greedy strategy for **Kruskal’s algorithm**: choose the edge with the smallest weight.

![Graph with weights]
Recall the greedy strategy for Kruskal’s algorithm: choose the edge with the smallest weight.

Greedy strategy for Prim’s algorithm: choose the lightest edge incident to \( a \).
Recall the greedy strategy for Kruskal’s algorithm: choose the edge with the smallest weight.

Greedy strategy for Prim’s algorithm: choose the lightest edge incident to $a$. 
**Lemma**  It is safe to include the lightest edge incident to $a$. 

---

**Proof.** Let $T$ be a MST. Consider all components obtained by removing $a$ from $T$. Let $e^\ast$ be the lightest edge incident to $a$ and $e^\ast$ connects $a$ to component $C$. Let $e$ be the edge in $T$ connecting $a$ to $C$. 

$T_0 = T \setminus \{e\} \cup \{e^\ast\}$ is a spanning tree with $w(T_0) \leq w(T)$.
Lemma It is safe to include the lightest edge incident to $a$.

Proof.

- Let $T$ be a MST
- Consider all components obtained by removing $a$ from $T$
**Lemma**  It is safe to include the lightest edge incident to $a$.

**Proof.**

- Let $T$ be a MST
- Consider all components obtained by removing $a$ from $T$
- Let $e^*$ be the lightest edge incident to $a$ and $e^*$ connects $a$ to component $C$
**Lemma** It is safe to include the lightest edge incident to \( a \).

**Proof.**

- Let \( T \) be a MST.
- Consider all components obtained by removing \( a \) from \( T \).
- Let \( e^* \) be the lightest edge incident to \( a \) and \( e^* \) connects \( a \) to component \( C \).
- Let \( e \) be the edge in \( T \) connecting \( a \) to \( C \).
**Lemma**  It is safe to include the lightest edge incident to \( a \).

**Proof.**

- Let \( T \) be a MST.
- Consider all components obtained by removing \( a \) from \( T \).
- Let \( e^* \) be the lightest edge incident to \( a \) and \( e^* \) connects \( a \) to component \( C \).
- Let \( e \) be the edge in \( T \) connecting \( a \) to \( C \).
- \( T' = T \setminus \{e\} \cup \{e^*\} \) is a spanning tree with \( w(T') \leq w(T) \).
Prim’s Algorithm: Example
Prim’s Algorithm: Example
Prim’s Algorithm: Example
Prim’s Algorithm: Example
Prim’s Algorithm: Example
Prim’s Algorithm: Example
Prim’s Algorithm: Example
Prim’s Algorithm: Example
Prim’s Algorithm: Example
Prim’s Algorithm: Example
Prim’s Algorithm: Example
Prim’s Algorithm: Example
Prim’s Algorithm: Example
Prim’s Algorithm: Example
Prim’s Algorithm: Example
Prim’s Algorithm: Example
Prim’s Algorithm: Example
Prim’s Algorithm: Example
MST-Greedy1\((G, w)\)

1: \( S \leftarrow \{s\} \), where \( s \) is arbitrary vertex in \( V \)
2: \( F \leftarrow \emptyset \)
3: \textbf{while} \( S \neq V \) \textbf{do}
4: \((u, v) \leftarrow \text{lightest edge between } S \text{ and } V \setminus S, \text{ where } u \in S \text{ and } v \in V \setminus S\)
5: \( S \leftarrow S \cup \{v\} \)
6: \( F \leftarrow F \cup \{(u, v)\} \)
7: \textbf{return} \((V, F)\)
Greedy Algorithm

MST-Greedy1($G, w$)

1: $S \leftarrow \{s\}$, where $s$ is arbitrary vertex in $V$
2: $F \leftarrow \emptyset$
3: while $S \neq V$ do
4:   $(u, v) \leftarrow$ lightest edge between $S$ and $V \setminus S$, where $u \in S$ and $v \in V \setminus S$
5:   $S \leftarrow S \cup \{v\}$
6:   $F \leftarrow F \cup \{(u, v)\}$
7: return $(V, F)$

- Running time of naive implementation: $O(nm)$
Prim’s Algorithm: Efficient Implementation of Greedy Algorithm

For every \( v \in V \setminus S \) maintain

- \( d[v] = \min_{u \in S: (u,v) \in E} w(u, v) \): the weight of the lightest edge between \( v \) and \( S \)
- \( \pi[v] = \arg\min_{u \in S: (u,v) \in E} w(u, v) \): \((\pi[v], v)\) is the lightest edge between \( v \) and \( S \)
Prim’s Algorithm: Efficient Implementation of Greedy Algorithm

For every $v \in V \setminus S$ maintain

- $d[v] = \min_{u \in S: (u,v) \in E} w(u,v)$:
  
  the weight of the lightest edge between $v$ and $S$

- $\pi[v] = \arg\min_{u \in S: (u,v) \in E} w(u,v)$:

  $(\pi[v], v)$ is the lightest edge between $v$ and $S$

In every iteration

- Pick $u \in V \setminus S$ with the smallest $d[u]$ value
- Add $(\pi[u], u)$ to $F$
- Add $u$ to $S$, update $d$ and $\pi$ values.
Prim’s Algorithm

MST-Prim\((G, w)\)

1: \(s \leftarrow \text{arbitrary vertex in } G\)
2: \(S \leftarrow \emptyset, d(s) \leftarrow 0\) and \(d[v] \leftarrow \infty\) for every \(v \in V \setminus \{s\}\)
3: \textbf{while } S \neq V \textbf{ do}
4: \hspace{1em} u \leftarrow \text{vertex in } V \setminus S \text{ with the minimum } d[u]
5: \hspace{1em} S \leftarrow S \cup \{u\}
6: \hspace{1em} \textbf{for each } v \in V \setminus S \text{ such that } (u, v) \in E \textbf{ do}
7: \hspace{2em} \textbf{if } w(u, v) < d[v] \textbf{ then}
8: \hspace{3em} d[v] \leftarrow w(u, v)
9: \hspace{3em} \pi[v] \leftarrow u
10: \textbf{return } \{(u, \pi[u])|u \in V \setminus \{s\}\}
Example

Diagram:

- Nodes: a, b, c, d, e, f, g, h, i
- Edges with weights:
  - a to b: 5
  - a to h: 12
  - b to c: 8
  - b to i: 11
  - c to d: 13
  - c to i: 2
  - d to e: 9
  - d to f: 14
  - e to f: 10
  - h to g: 7
  - h to i: 6
  - i to g: 4
  - i to f: 1
  - g to f: 3
Example
Example
Example

\begin{itemize}
\item \((5, a)\)
\item \((12, a)\)
\end{itemize}
Example
Example

\[(8, b)\]

\[(11, b)\]
Example

![Graph diagram]

- Node labels: a, b, c, d, e, f, g, h, i
- Edges with labels: 5, 8, 2, 13, 14, 9, 7, 6, 3, 11
- Node pairs with labels: (8, b), (11, b)

The graph contains a highlighted subset with nodes a, b, and h.
Example
Example
Example
Example
Example
Example
Example
Example
Example

\begin{tikzpicture}[node distance=2cm, thick, main node/.style={circle,draw}]% set node distance and style for nodes
  \node[main node] (A) {\textcolor{red}{a}}; % node A with red color
  \node[main node] (B) [right of=A] {\textcolor{red}{b}}; % node B with red color
  \node[main node] (C) [right of=B] {\textcolor{red}{c}}; % node C with red color
  \node[main node] (D) [right of=C] {d}; % node D
  \node[main node] (E) [right of=D] {\textcolor{red}{e}}; % node E with red color
  \node[main node] (F) [below of=C] {\textcolor{red}{f}}; % node F with red color
  \node[main node] (H) [below of=A] {h}; % node H
  \node[main node] (I) [below of=B] {i}; % node I
  \node[main node] (G) [below of=F] {g}; % node G

  \path[->] (A) edge node [above] {5} (B) % edge from A to B with label 5
  \path[->] (B) edge node [above] {8} (C) % edge from B to C with label 8
  \path[->] (C) edge node [above] {13} (D) % edge from C to D with label 13
  \path[->] (D) edge node [above] {9} (E) % edge from D to E with label 9
  \path[->] (E) edge node [above] {(10, f)} (F) % edge from E to F with label (10, f)
  \path[->] (F) edge node [above] {3} (G) % edge from F to G with label 3
  \path[->] (G) edge node [above] {1} (H) % edge from G to H with label 1
  \path[->] (H) edge node [above] {12} (A) % edge from H to A with label 12
  \path[->] (I) edge node [above] {7} (B) % edge from I to B with label 7
  \path[->] (I) edge node [above] {6} (C) % edge from I to C with label 6
  \path[->] (I) edge node [above] {(3, f)} (F) % edge from I to F with label (3, f)
  \path[->] (I) edge node [above] {11} (A) % edge from I to A with label 11
  \path[->] (I) edge node [above] {(7, i)} (H) % edge from I to H with label (7, i)
\end{tikzpicture}
Example
Example
Example

(1, g) → h → i → c → (13, c) → d → e → (10, f)

a → b → 8 → c → 13

d → 9 → (10, f)

e → 14 → f → 3 → g → 6 → i

(1, g) → h → 1 → f → 10

b → 5 → a → 11

c → 2 → i → 7

d → 4 → g → 4

e → 14 → f → 3

Example
Example
Example
Example
Example

Diagram with labeled nodes and edges.
Example

(9, e)
Example