4. **Greedy Algorithms II**

- red-rule blue-rule demo
- Prim’s algorithm demo
- Kruskal’s algorithm demo
- reverse-delete algorithm demo
- Boruvka’s algorithm demo
4. **Greedy Algorithms II**

- red-rule blue-rule demo
- Prim’s algorithm demo
- Kruskal’s algorithm demo
- reverse-delete algorithm demo
- Boruvka’s algorithm demo
Red-rule blue-rule demo

Red rule. Let $C$ be a cycle with no red edges. Select an uncolored edge of $C$ of max weight and color it red.

Blue rule. Let $D$ be a cutset with no blue edges. Select an uncolored edge in $D$ of min weight and color it blue.

the input graph

![Graph Image]
Red-rule blue-rule demo

Red rule. Let $C$ be a cycle with no red edges. Select an uncolored edge of $C$ of max weight and color it red.

apply the red rule to the cycle
Red-rule blue-rule demo

current set of red and blue edges
Red-rule blue-rule demo

**Red rule.** Let $C$ be a cycle with no red edges. Select an uncolored edge of $C$ of max weight and color it red.

apply the red rule to the cycle
Red-rule blue-rule demo

Red rule. Let $C$ be a cycle with no red edges. Select an uncolored edge of $C$ of max weight and color it red.
Red-rule blue-rule demo

Blue rule. Let $D$ be a cutset with no blue edges. Select an uncolored edge in $D$ of min weight and color it blue.

apply the blue rule to the cutset

![Diagram showing the application of the blue rule to a cutset with specified edges and weights.]

1  5  7  6  4
Red-rule blue-rule demo

current set of red and blue edges
**Blue rule.** Let $D$ be a cutset with no blue edges. Select an uncolored edge in $D$ of min weight and color it blue.

apply the blue rule to the cutset
Red-rule blue-rule demo

current set of red and blue edges
Red-rule blue-rule demo

Red rule. Let $C$ be a cycle with no red edges. Select an uncolored edge of $C$ of max weight and color it red.

apply the red rule to the cycle
Red-rule blue-rule demo

current set of red and blue edges
Red-rule blue-rule demo

Blue rule. Let $D$ be a cutset with no blue edges. Select an uncolored edge in $D$ of min weight and color it blue.

apply the blue rule to the cutset
Red-rule blue-rule demo

current set of red and blue edges
Red-rule blue-rule demo

Blue rule. Let $D$ be a cutset with no blue edges. Select an uncolored edge in $D$ of min weight and color it blue.

apply the blue rule to the cutset
Red-rule blue-rule demo

current set of red and blue edges
Red-rule blue-rule demo

Blue rule. Let $D$ be a cutset with no blue edges. Select an uncolored edge in $D$ of min weight and color it blue.

apply the blue rule to the cutset
Red-rule blue-rule demo

current set of red and blue edges
Red-rule blue-rule demo

Blue rule. Let $D$ be a cutset with no blue edges. Select an uncolored edge in $D$ of min weight and color it blue.

apply the red rule to the cycle
Red-rule blue-rule demo

current set of red and blue edges
Red-rule blue-rule demo

Greedy algorithm. Upon termination, the blue edges form a MST.

a minimum spanning tree
4. Greedy Algorithms II

- red-rule blue-rule demo
- Prim’s algorithm demo
- Kruskal’s algorithm demo
- reverse-delete algorithm demo
- Boruvka’s algorithm demo
Prim’s algorithm demo

Initialize $S = \{ s \}$ for any node $s$, $T = \emptyset$.
Repeat $n - 1$ times:
  - Add to $T$ a min-weight edge with exactly one endpoint in $S$.
  - Add the other endpoint to $S$. 
Prim’s algorithm demo

Initialize $S = \{ s \}$ for any node $s$, $T = \emptyset$.

Repeat $n - 1$ times:

- Add to $T$ a min-weight edge with exactly one endpoint in $S$.
- Add the other endpoint to $S$. 
Prim’s algorithm demo

Initialize $S = \{ s \}$ for any node $s$, $T = \emptyset$.
Repeat $n - 1$ times:
  • Add to $T$ a min-weight edge with exactly one endpoint in $S$.
  • Add the other endpoint to $S$. 
Prim’s algorithm demo

Initialize $S = \{ s \}$ for any node $s$, $T = \emptyset$.

Repeat $n - 1$ times:

- Add to $T$ a min-weight edge with exactly one endpoint in $S$.
- Add the other endpoint to $S$. 
Prim’s algorithm demo

Initialize $S = \{ s \}$ for any node $s$, $T = \emptyset$.
Repeat $n - 1$ times:
- Add to $T$ a min-weight edge with exactly one endpoint in $S$.
- Add the other endpoint to $S$. 
Prim’s algorithm demo

Initialize $S = \{ s \}$ for any node $s$, $T = \emptyset$.

Repeat $n - 1$ times:

- Add to $T$ a min-weight edge with exactly one endpoint in $S$.
- Add the other endpoint to $S$. 
Prim’s algorithm demo

Initialize $S = \{ s \}$ for any node $s$, $T = \emptyset$.

Repeat $n - 1$ times:
  - Add to $T$ a min-weight edge with exactly one endpoint in $S$.
  - Add the other endpoint to $S$. 
Prim’s algorithm demo

Initialize $S = \{s\}$ for any node $s$, $T = \emptyset$.

Repeat $n - 1$ times:

- Add to $T$ a min-weight edge with exactly one endpoint in $S$.
- Add the other endpoint to $S$.
Prim’s algorithm demo

Initialize $S = \{ s \}$ for any node $s$, $T = \emptyset$.

Repeat $n - 1$ times:

- Add to $T$ a min-weight edge with exactly one endpoint in $S$.
- Add the other endpoint to $S$. 
Prim’s algorithm demo

Initialize $S = \{ s \}$ for any node $s$, $T = \emptyset$.

Repeat $n - 1$ times:
- Add to $T$ a min-weight edge with exactly one endpoint in $S$.
- Add the other endpoint to $S$. 

[Diagram of a graph with labeled edges and nodes, showing the process of Prim's algorithm.]
Prim’s algorithm demo

Initialize $S = \{ s \}$ for any node $s$, $T = \emptyset$.

Repeat $n - 1$ times:

- Add to $T$ a min-weight edge with exactly one endpoint in $S$.
- Add the other endpoint to $S$. 
Prim’s algorithm demo

Initialize $S = \{ s \}$ for any node $s$, $T = \emptyset$.

Repeat $n - 1$ times:
- Add to $T$ a min-weight edge with exactly one endpoint in $S$.
- Add the other endpoint to $S$. 
Prim’s algorithm demo

Initialize $S = \{ s \}$ for any node $s$, $T = \emptyset$.

Repeat $n - 1$ times:

- Add to $T$ a min-weight edge with exactly one endpoint in $S$.
- Add the other endpoint to $S$. 
Prim’s algorithm demo

Initialize $S = \{ s \}$ for any node $s$, $T = \emptyset$.
Repeat $n - 1$ times:
  • Add to $T$ a min-weight edge with exactly one endpoint in $S$.
  • Add the other endpoint to $S$. 

![Diagram](image)
Prim’s algorithm demo

Initialize $S = \{ s \}$ for any node $s$, $T = \emptyset$.

Repeat $n - 1$ times:

- Add to $T$ a min-weight edge with exactly one endpoint in $S$.
- Add the other endpoint to $S$. 
Prim’s algorithm demo

Initialize $S = \{ s \}$ for any node $s$, $T = \emptyset$.
Repeat $n - 1$ times:

- Add to $T$ a min-weight edge with exactly one endpoint in $S$.
- Add the other endpoint to $S$. 

![Diagram of Prim's algorithm](image-url)
Prim’s algorithm demo

Initialize $S = \{ s \}$ for any node $s$, $T = \emptyset$.

Repeat $n - 1$ times:

- Add to $T$ a min-weight edge with exactly one endpoint in $S$.
- Add the other endpoint to $S$. 
4. **Greedy Algorithms II**

- red-rule blue-rule demo
- Prim’s algorithm demo
- Kruskal’s algorithm demo
- reverse-delete algorithm demo
- Boruvka’s algorithm demo

**Section 4.5**
Kruskal’s algorithm demo

Consider edges in ascending order of weight:

- Add to $T$ unless it would create a cycle.
Consider edges in ascending order of weight:
  - Add to $T$ unless it would create a cycle.
Kruskal’s algorithm demo

Consider edges in ascending order of weight:

- Add to $T$ unless it would create a cycle.
Kruskal’s algorithm demo

Consider edges in ascending order of weight:
  • Add to $T$ unless it would create a cycle.
Kruskal’s algorithm demo

Consider edges in ascending order of weight:
  • Add to $T$ unless it would create a cycle.
Kruskal’s algorithm demo

Consider edges in ascending order of weight:
• Add to $T$ unless it would create a cycle.
Consider edges in ascending order of weight:

- Add to $T$ unless it would create a cycle.
Kruskal’s algorithm demo

Consider edges in ascending order of weight:

- Add to $T$ unless it would create a cycle.
Kruskal’s algorithm demo

Consider edges in ascending order of weight:

- Add to $T$ unless it would create a cycle.
Kruskal’s algorithm demo

Consider edges in ascending order of weight:
  • Add to $T$ unless it would create a cycle.
Kruskal’s algorithm demo

Consider edges in ascending order of weight:
  • Add to $T$ unless it would create a cycle.
Kruskal’s algorithm demo

Consider edges in ascending order of weight:
• Add to $T$ unless it would create a cycle.
4. Greedy Algorithms II

- red-rule blue-rule demo
- Prim’s algorithm demo
- Kruskal’s algorithm demo
- reverse-delete algorithm demo
- Boruvka’s algorithm demo
Reverse-delete algorithm demo

Start with all edges in $T$ and consider them in descending order of weight:
- Delete edge from $T$ unless it would disconnect $T$. 

![Diagram of a graph with weights on edges and nodes labeled 1 to 8.]
Reverse-delete algorithm

Start with all edges in $T$ and consider them in descending order of weight:

- Delete edge from $T$ unless it would disconnect $T$. 

![Graph diagram](56)
Reverse-delete algorithm

Start with all edges in $T$ and consider them in descending order of weight:

- Delete edge from $T$ unless it would disconnect $T$. 

![Diagram of a graph network representing the reverse-delete algorithm](image-url)
Reverse-delete algorithm

Start with all edges in $T$ and consider them in descending order of weight:
  - Delete edge from $T$ unless it would disconnect $T$. 

\[ 7 \]
Reverse-delete algorithm

Start with all edges in $T$ and consider them in descending order of weight:

- Delete edge from $T$ unless it would disconnect $T$. 
Reverse-delete algorithm

Start with all edges in $T$ and consider them in descending order of weight:

- Delete edge from $T$ unless it would disconnect $T$. 

![Graph Diagram](image)
Reverse-delete algorithm

Start with all edges in $T$ and consider them in descending order of weight:

- Delete edge from $T$ unless it would disconnect $T$. 
Reverse-delete algorithm

Start with all edges in $T$ and consider them in descending order of weight:

- Delete edge from $T$ unless it would disconnect $T$. 
Reverse-delete algorithm

Start with all edges in $T$ and consider them in descending order of weight:
- Delete edge from $T$ unless it would disconnect $T$. 

![Diagram of a graph with edges and weights](image-url)
Reverse-delete algorithm

Start with all edges in $T$ and consider them in descending order of weight:

- Delete edge from $T$ unless it would disconnect $T$. 

![Graph diagram](image-url)
Reverse-delete algorithm

Start with all edges in $T$ and consider them in descending order of weight:

- Delete edge from $T$ unless it would disconnect $T$. 

![Diagram of a graph with edges highlighted.](attachment:image.png)
Reverse-delete algorithm

Start with all edges in $T$ and consider them in descending order of weight:

- Delete edge from $T$ unless it would disconnect $T$. 

![Graph Diagram]
4. **Greedy Algorithms II**

- red-rule blue-rule demo
- Prim’s algorithm demo
- Kruskal’s algorithm demo
- reverse-delete algorithm demo
- **Boruvka’s algorithm demo**
Borůvka’s algorithm demo

Repeat until only one tree.

- Apply blue rule to cutset corresponding to each blue tree.
- Color all selected edges blue.
Borůvka’s algorithm demo

Repeat until only one tree.

- Apply blue rule to cutset corresponding to each blue tree.
- Color all selected edges blue.
Borůvka’s algorithm demo

Repeat until only one tree.

- Apply blue rule to cutset corresponding to each blue tree.
- Color all selected edges blue.
Borůvka’s algorithm demo

Repeat until only one tree.

- Apply blue rule to cutset corresponding to each blue tree.
- Color all selected edges blue.
Borůvka’s algorithm demo

Repeat until only one tree.

- Apply blue rule to cutset corresponding to each blue tree.
- Color all selected edges blue.
Borůvka’s algorithm demo

Repeat until only one tree.

• Apply blue rule to cutset corresponding to each blue tree.
• Color all selected edges blue.
Borůvka’s algorithm demo

Repeat until only one tree.

- Apply blue rule to cutset corresponding to each blue tree.
- Color all selected edges blue.