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

Section 6.1
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
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

![Diagram of a cycle with edge labeled 9 colored red]
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

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

![Diagram](image-url)
Red-rule blue-rule demo

current set of red and blue edges

![Current set of red and blue edges](image-url)
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
**Blue rule.** Let $D$ be a cutset with no blue edges. Select an uncolored edge in $D$ of min weight and color it blue.
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
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

![Graph showing a minimum spanning tree with nodes and edges labeled with weights.]
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 $S$ = any node, $T = \emptyset$.
Repeat $n – 1$ times:
  • Add to $T$ a min-weight edge with one endpoint in $S$.
  • Add new node to $S$. 

```
1  6  8  12  5  10  9
  |  |  |   |   |   |
  4  7  3  11  2  9  9
```

Prim’s algorithm demo

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

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

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

Repeat $n - 1$ times:
  • Add to $T$ a min-weight edge with one endpoint in $S$.
  • Add new node to $S$. 

![Graph diagram with Prim's algorithm visualization]

- Edge weight 3
Prim’s algorithm demo

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

Repeat $n - 1$ times:

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

Initialize $S =$ any node, $T = \emptyset$.
Repeat $n - 1$ times:
  • Add to $T$ a min-weight edge with one endpoint in $S$.
  • Add new node to $S$. 

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

Initialize $S =$ any node, $T = \emptyset$.
Repeat $n - 1$ times:
  • Add to $T$ a min-weight edge with one endpoint in $S$.
  • Add new node to $S$. 

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

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

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

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

Repeat $n - 1$ times:

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

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

Repeat $n - 1$ times:

- Add to $T$ a min-weight edge with one endpoint in $S$.
- Add new node to $S$. 

![Graph with Prim's algorithm applied](image-url)
Prim’s algorithm demo

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

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

Repeat $n - 1$ times:

- Add to $T$ a min-weight edge with one endpoint in $S$.
- Add new node to $S$. 

![Graph with Prim's algorithm example]
Prim’s algorithm demo

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

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

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

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

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

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

Initialize $S =$ any node, $T = \emptyset$.
Repeat $n - 1$ times:
  • Add to $T$ a min-weight edge with one endpoint in $S$.
  • Add new node 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
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.
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.
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

Section 4.5
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]

numbers:

- 1
- 2
- 3
- 5
- 6
- 7
- 8
- 9
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 $T$. 

![Graph diagram with node connections and edge weights]
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 labeled 5](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 and weights]
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 $T$. 
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 with nodes and edges labeled with weights 1, 2, 3, 4, 7.](attachment:graph.png)
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 6.2**
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.
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.