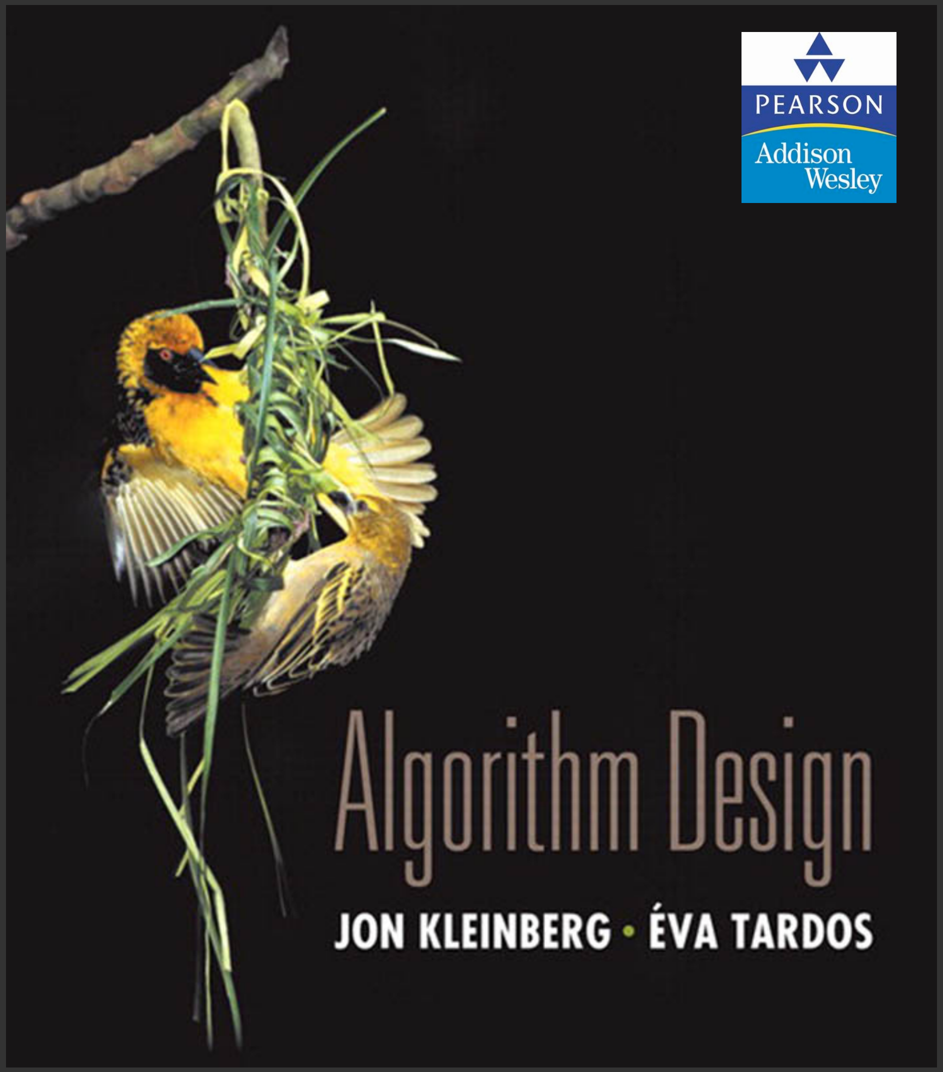


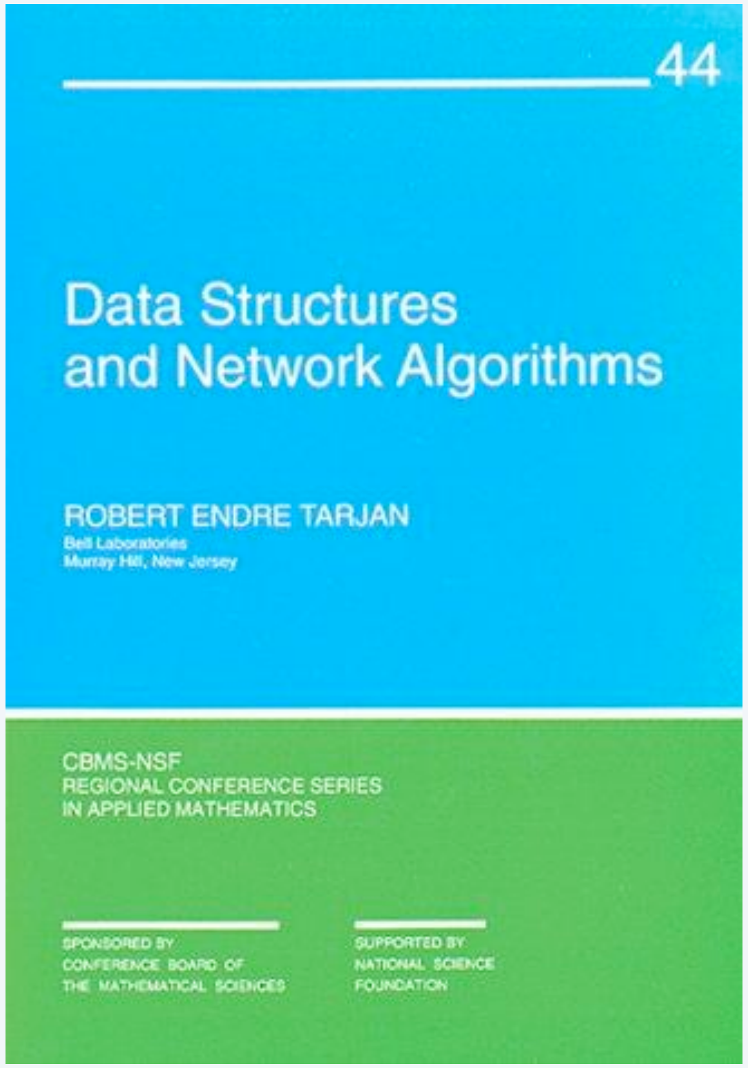
4. Greedy Algorithms II

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

Lecture slides by Kevin Wayne

http://www.cs.princeton.edu/~wayne/kleinberg-tardos

Last updated on 1/15/20 9:35 AM



SECTION 6.1

4. GREEDY ALGORITHMS II

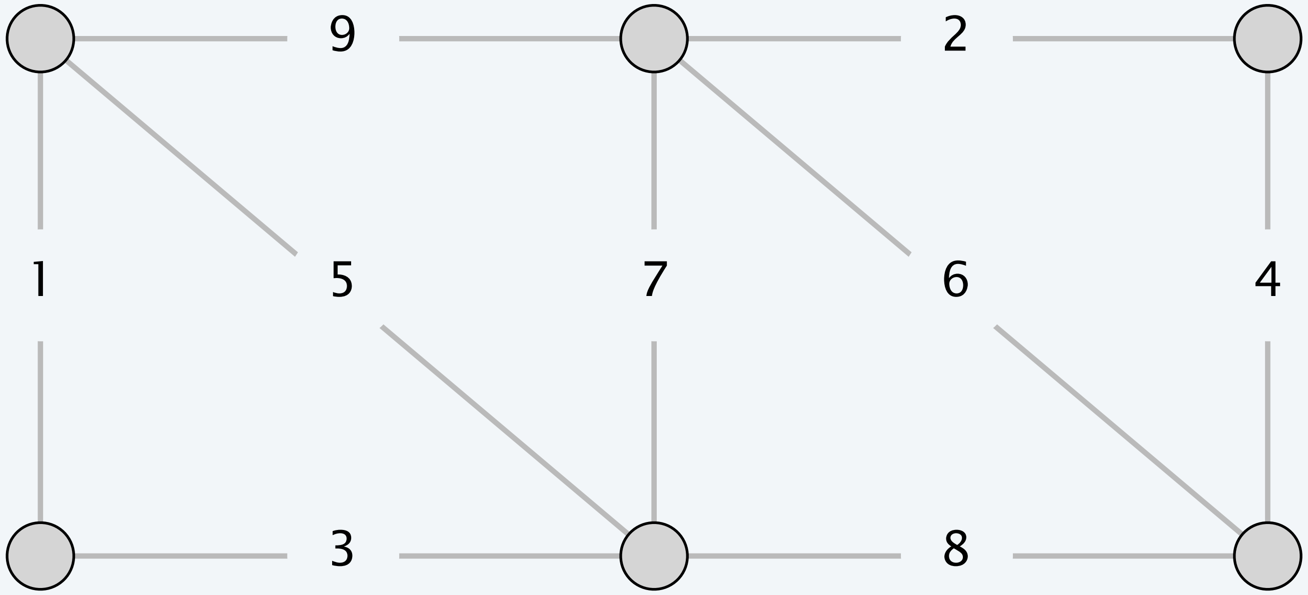
- *red-rule blue-rule demo*
- *Prim's algorithm demo*
- *Kruskal's algorithm demo*
- *reverse-delete algorithm demo*
- *Boruvka's algorithm demo*

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.

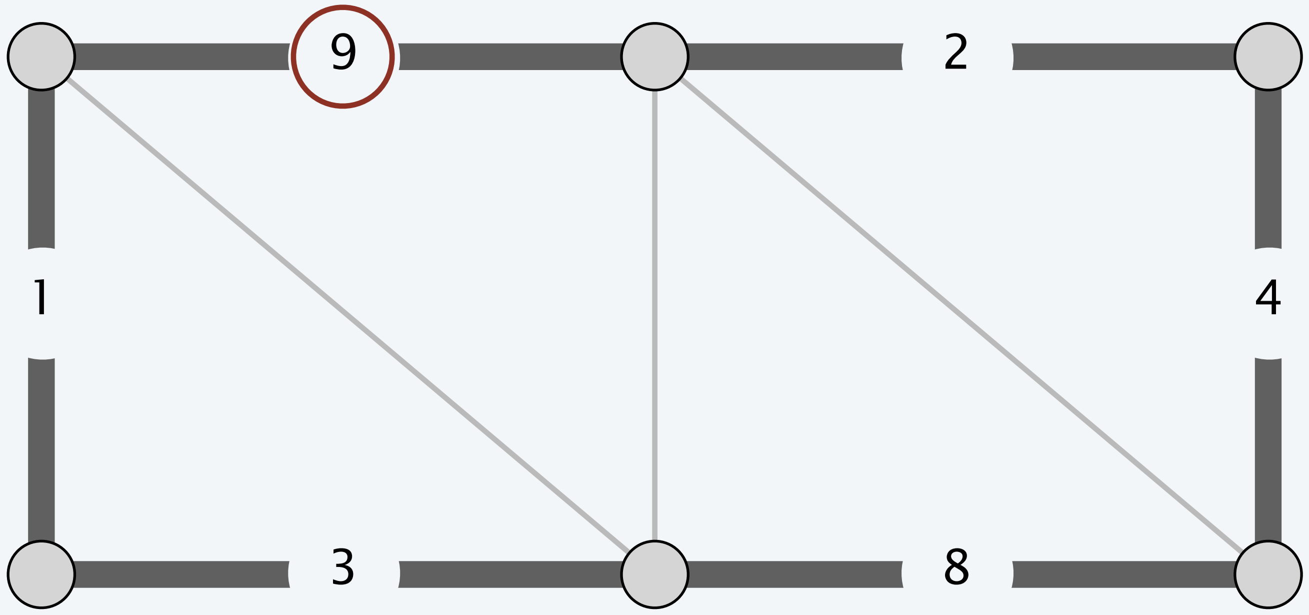
the input graph



Red-rule blue-rule demo

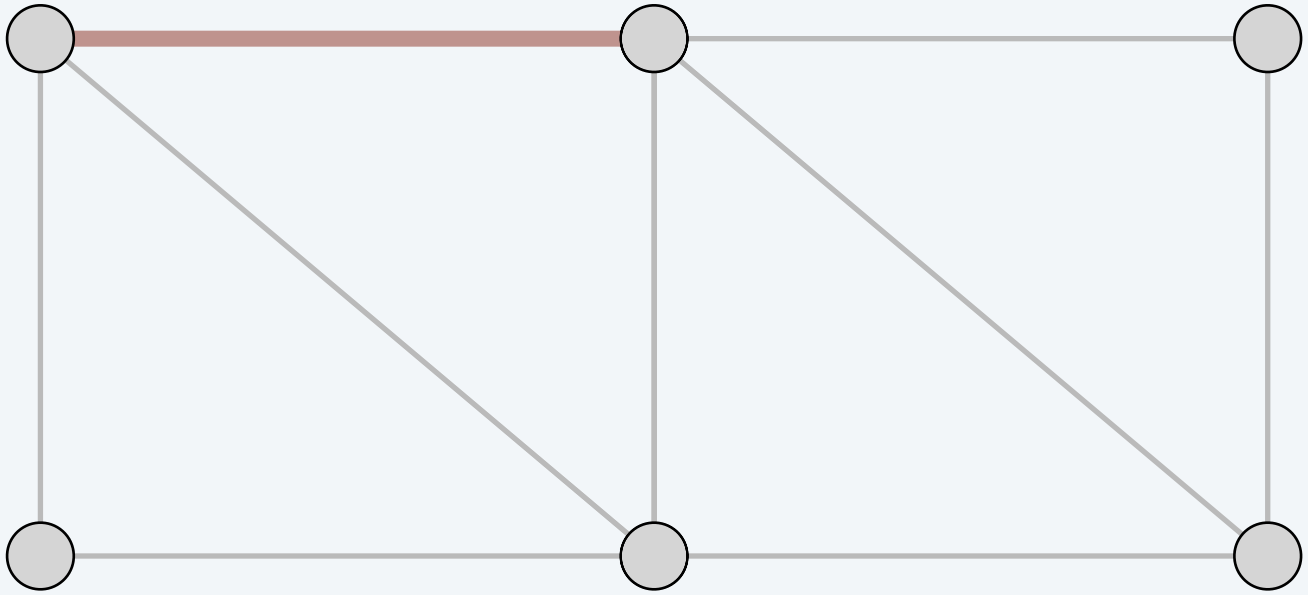
Red rule. Let C be a cycle with no red edges. Select an uncolored edge of C of max weight and color it red.

apply the red rule to the cycle



Red-rule blue-rule demo

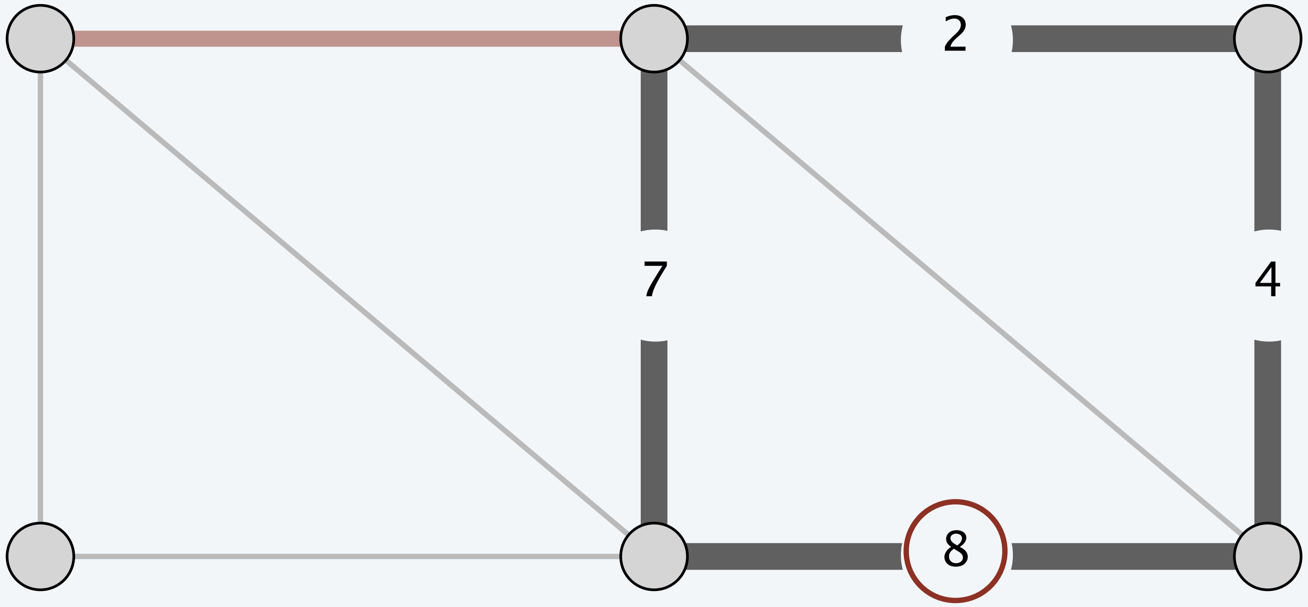
current set of red and blue edges



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.

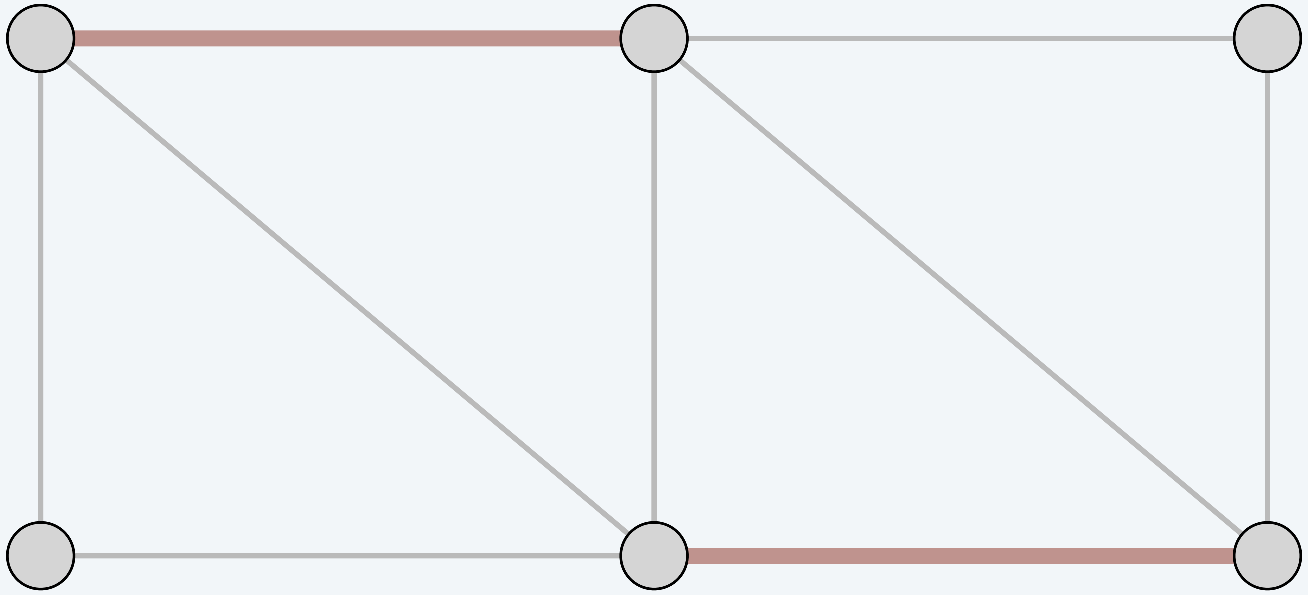
apply the red rule to the cycle



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.

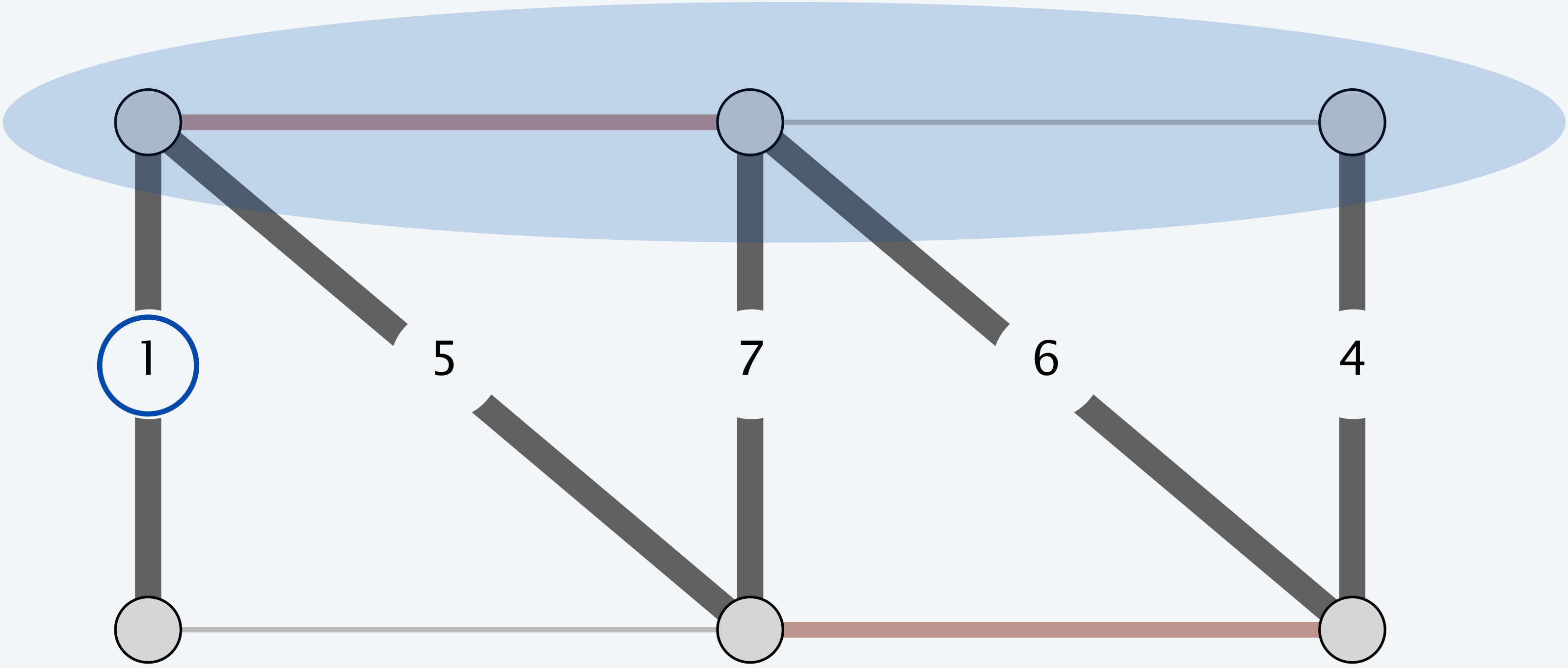
current set of red and blue edges



Red-rule blue-rule demo

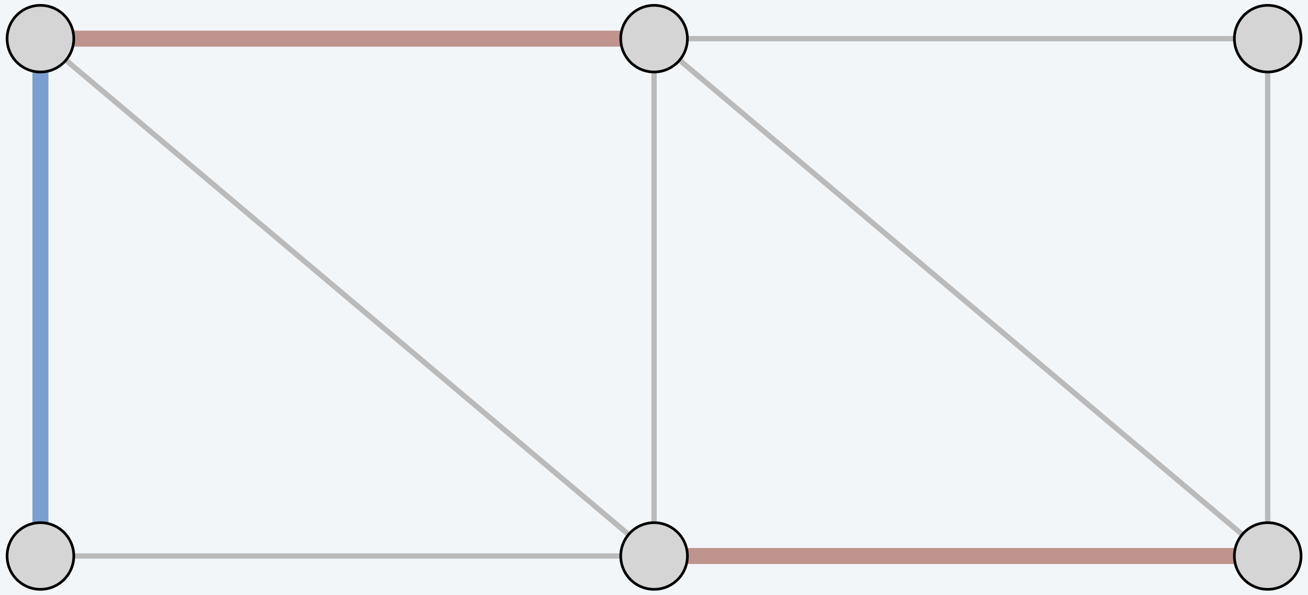
Blue rule. Let D be a cutset with no blue edges. Select an uncolored edge in D of min weight and color it blue.

apply the blue rule to the cutset



Red-rule blue-rule demo

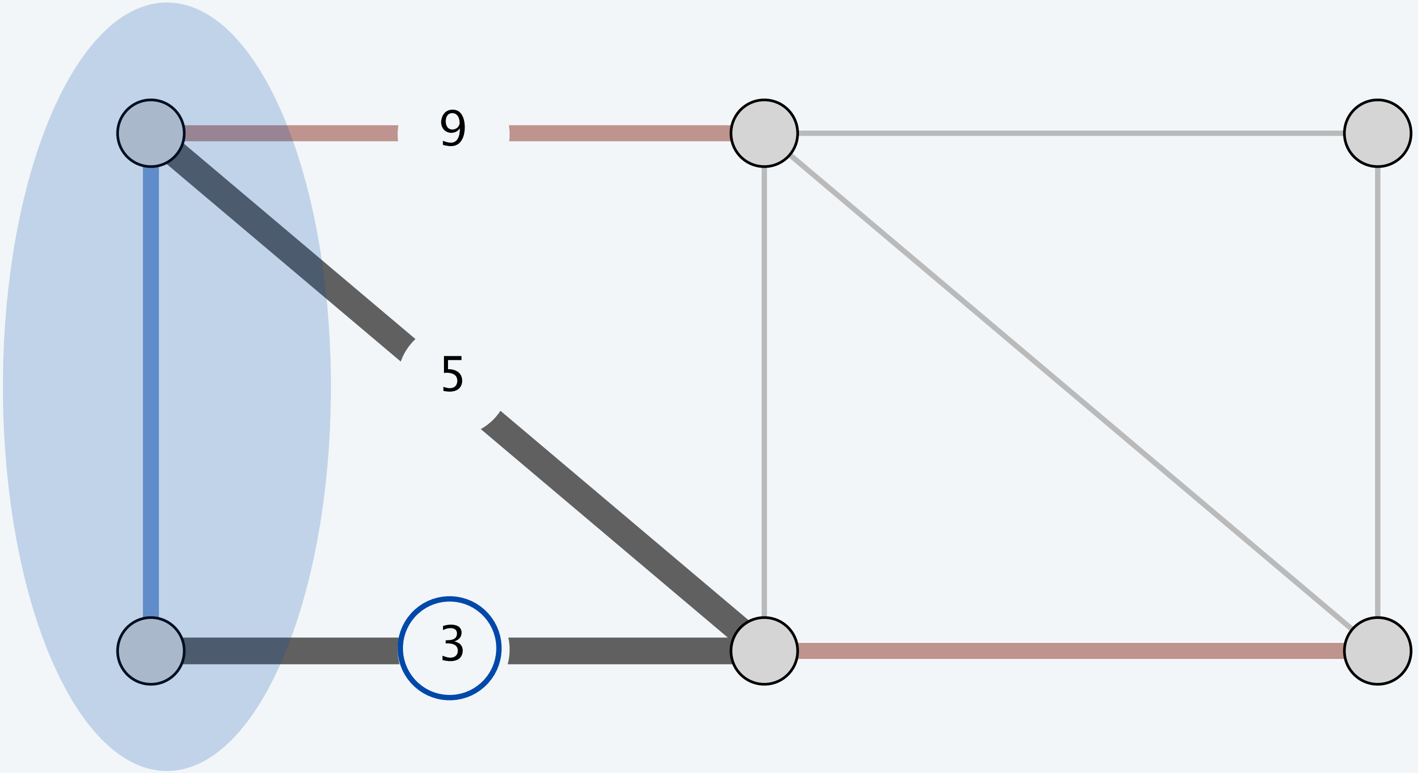
current set of red and blue edges



Red-rule blue-rule demo

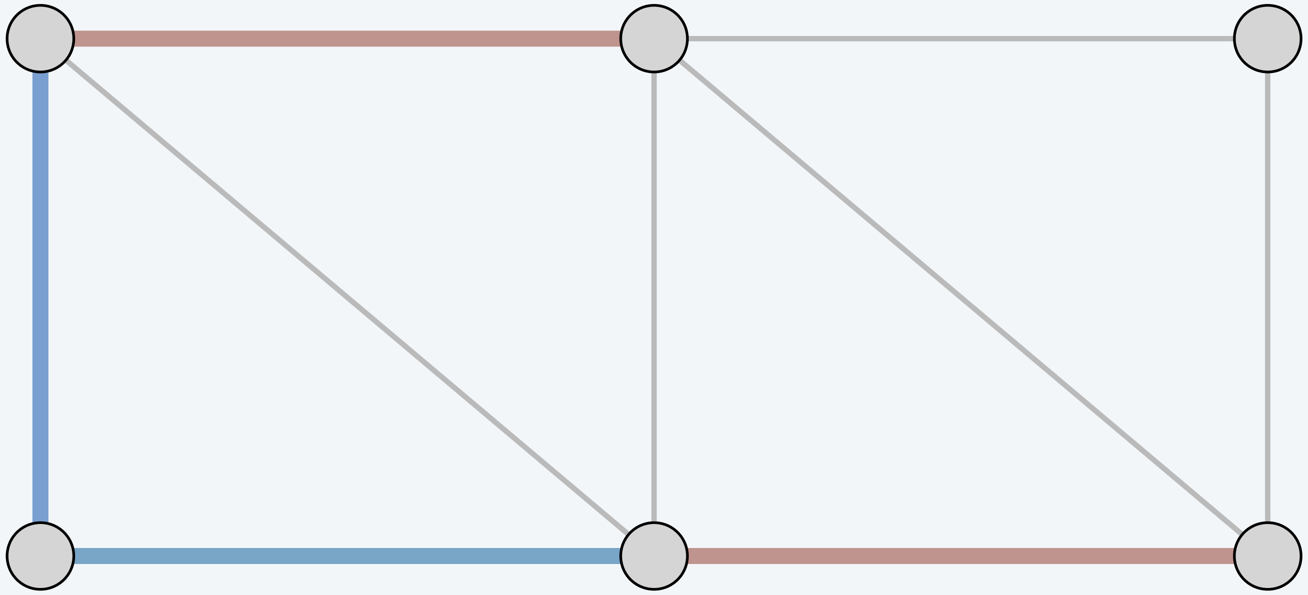
Blue rule. Let D be a cutset with no blue edges. Select an uncolored edge in D of min weight and color it blue.

apply the blue rule to the cutset



Red-rule blue-rule demo

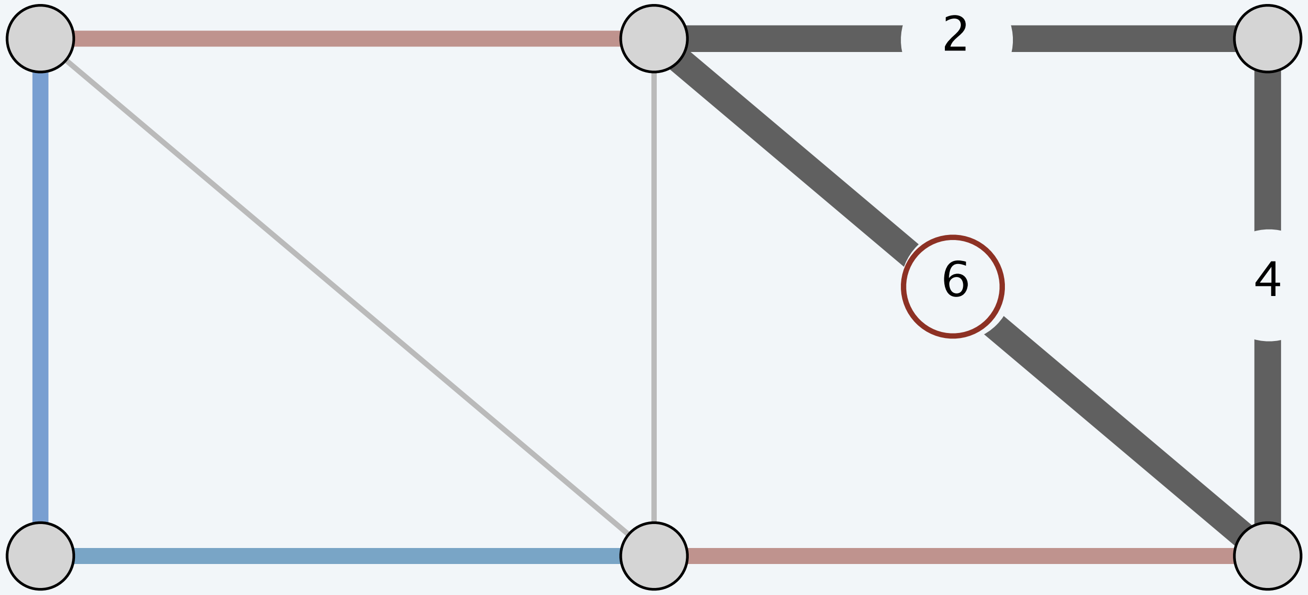
current set of red and blue edges



Red-rule blue-rule demo

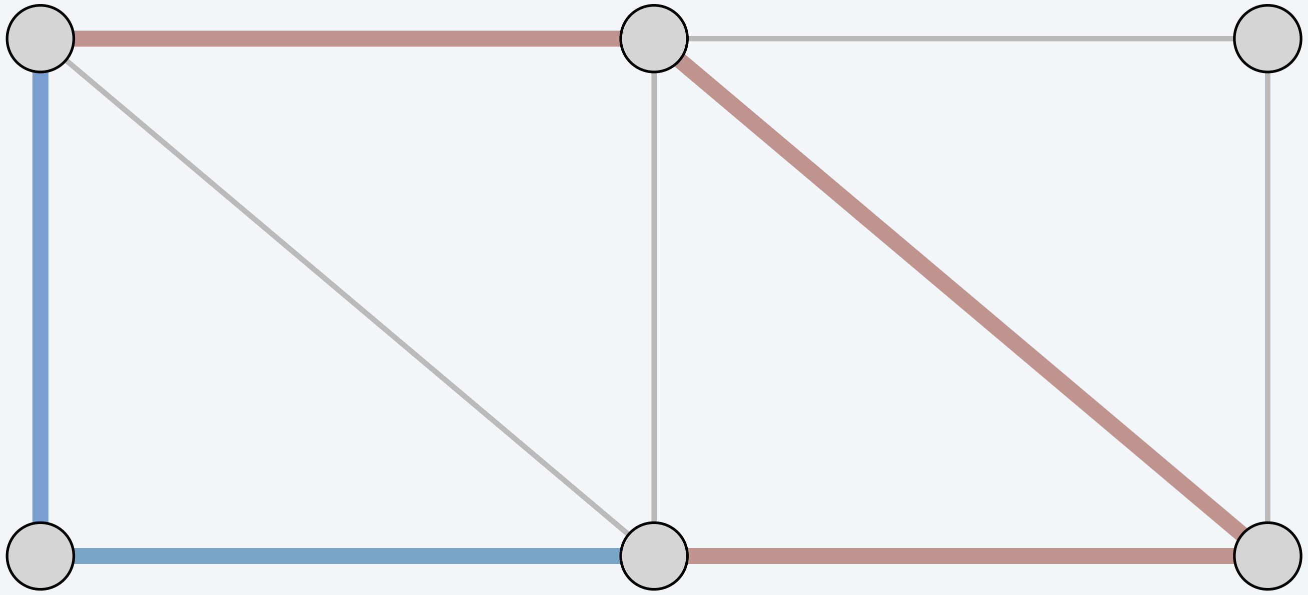
Red rule. Let C be a cycle with no red edges. Select an uncolored edge of C of max weight and color it red.

apply the red rule to the cycle



Red-rule blue-rule demo

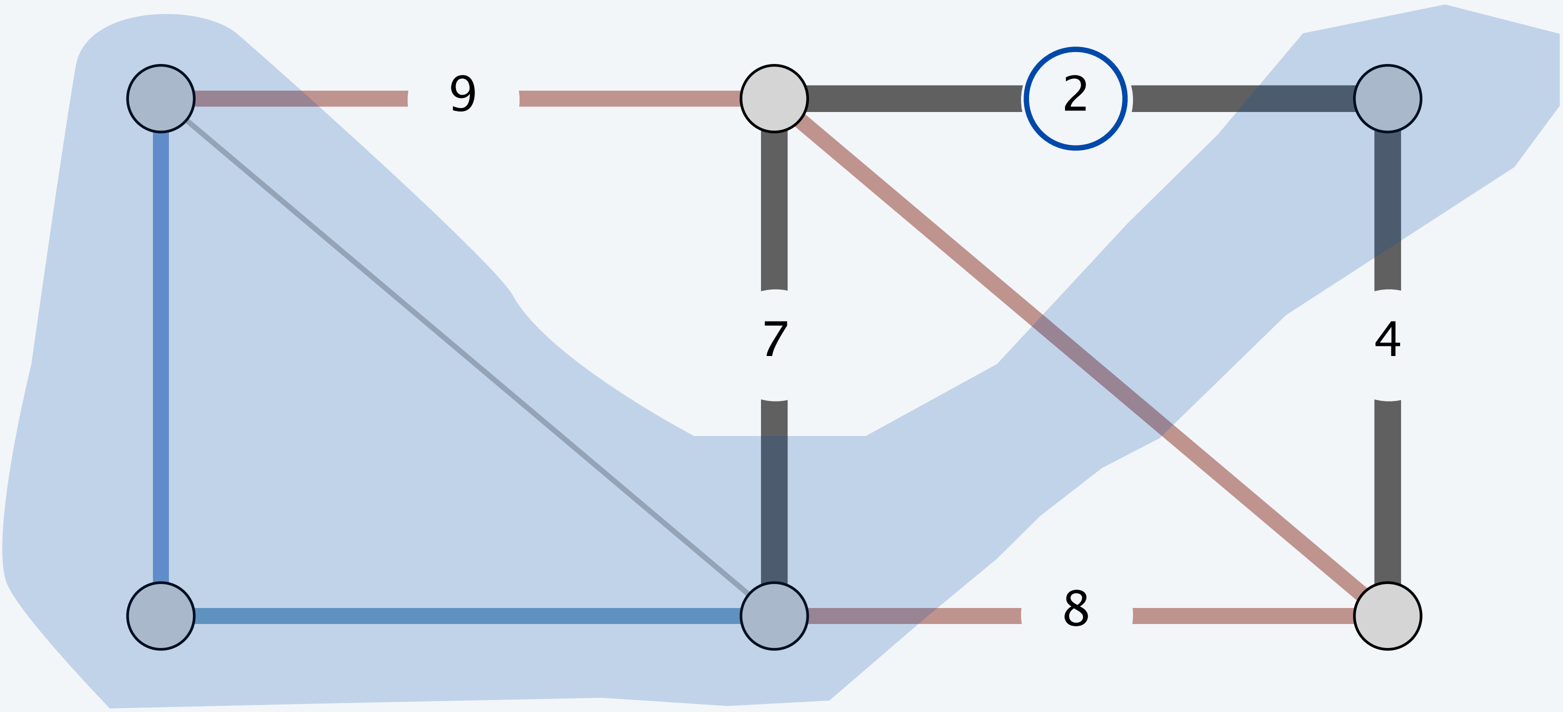
current set of red and blue edges



Red-rule blue-rule demo

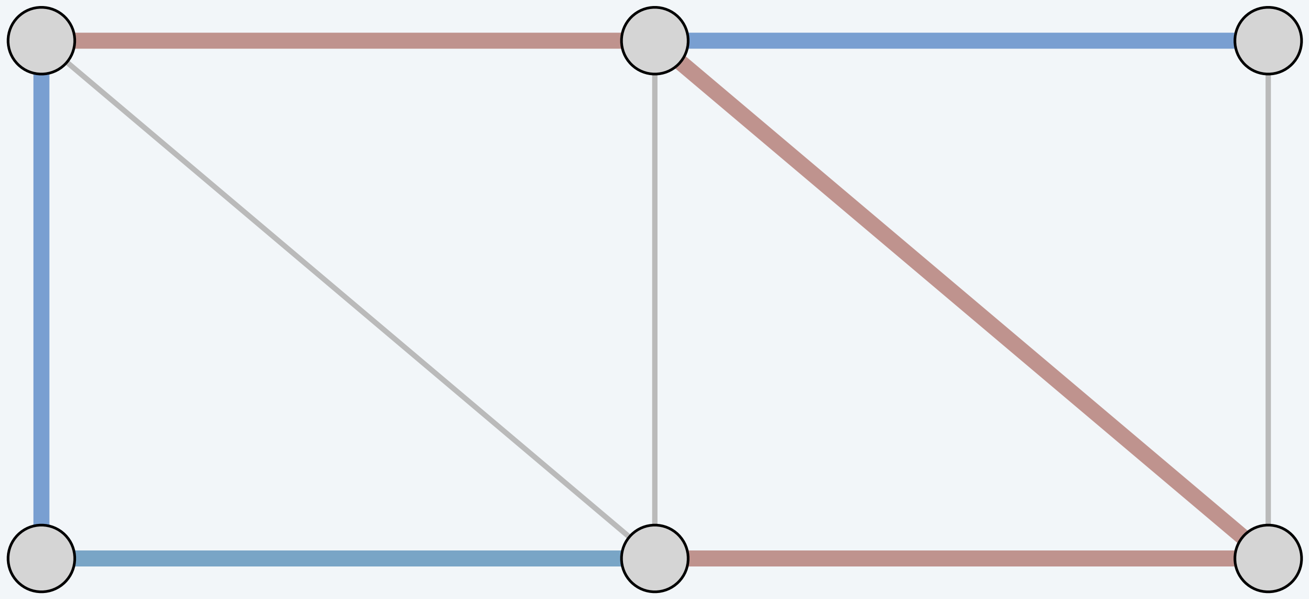
Blue rule. Let D be a cutset with no blue edges. Select an uncolored edge in D of min weight and color it blue.

apply the blue rule to the cutset



Red-rule blue-rule demo

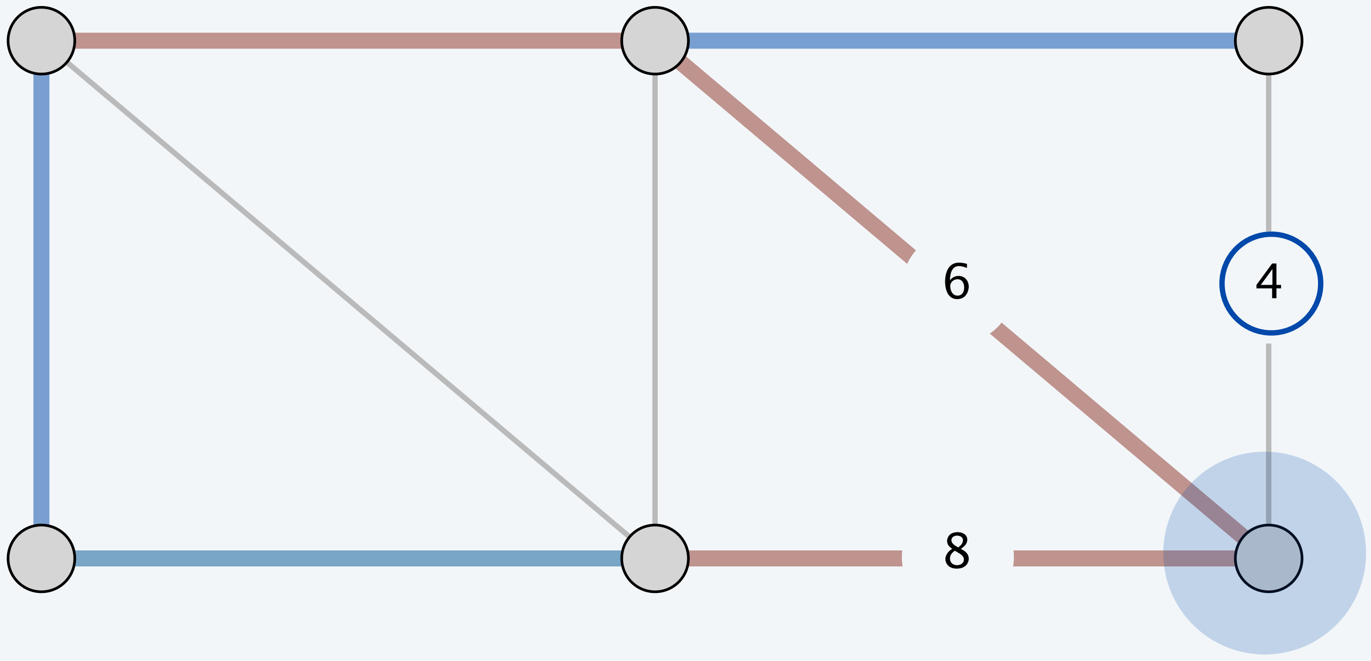
current set of red and blue edges



Red-rule blue-rule demo

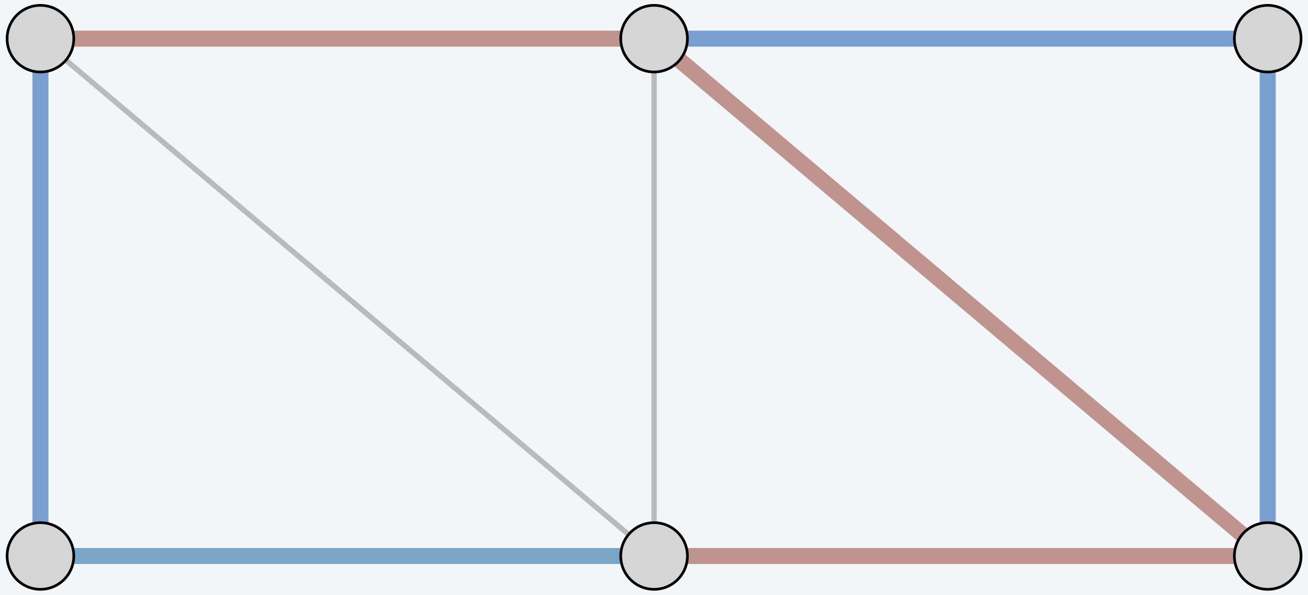
Blue rule. Let D be a cutset with no blue edges. Select an uncolored edge in D of min weight and color it blue.

apply the blue rule to the cutset



Red-rule blue-rule demo

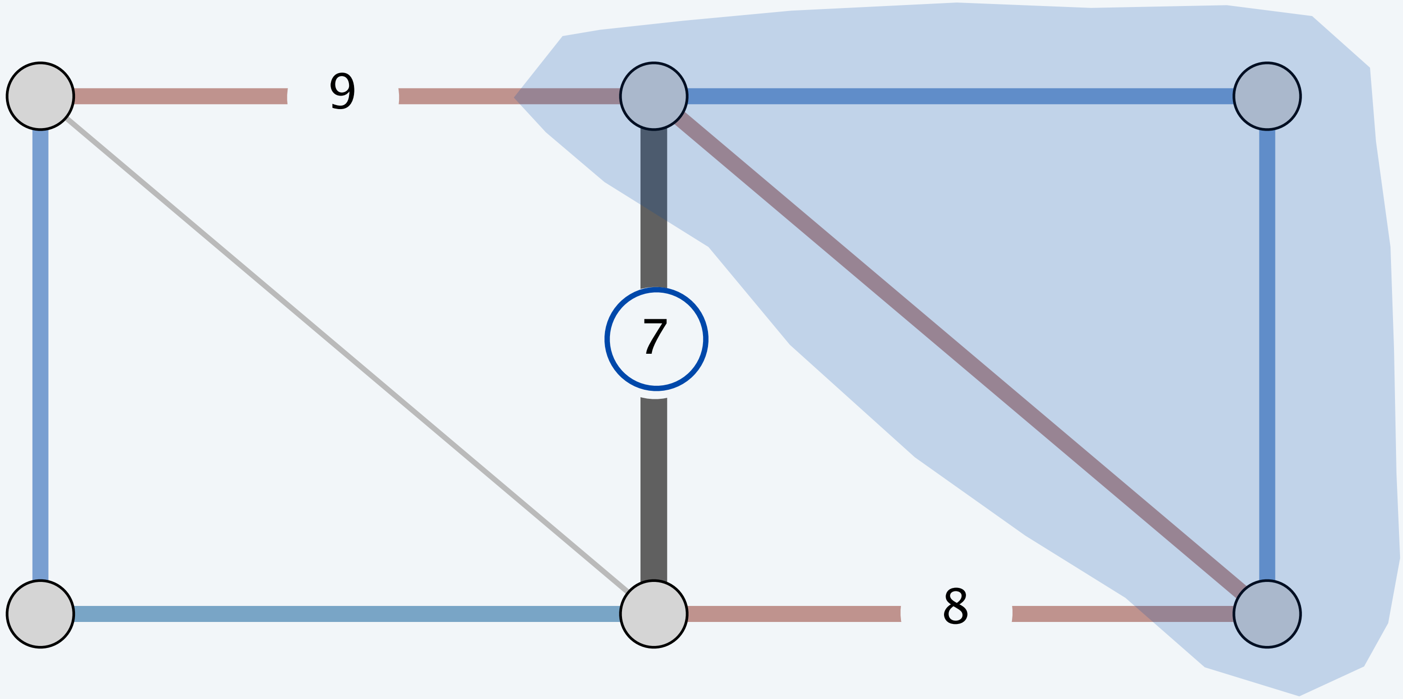
current set of red and blue edges



Red-rule blue-rule demo

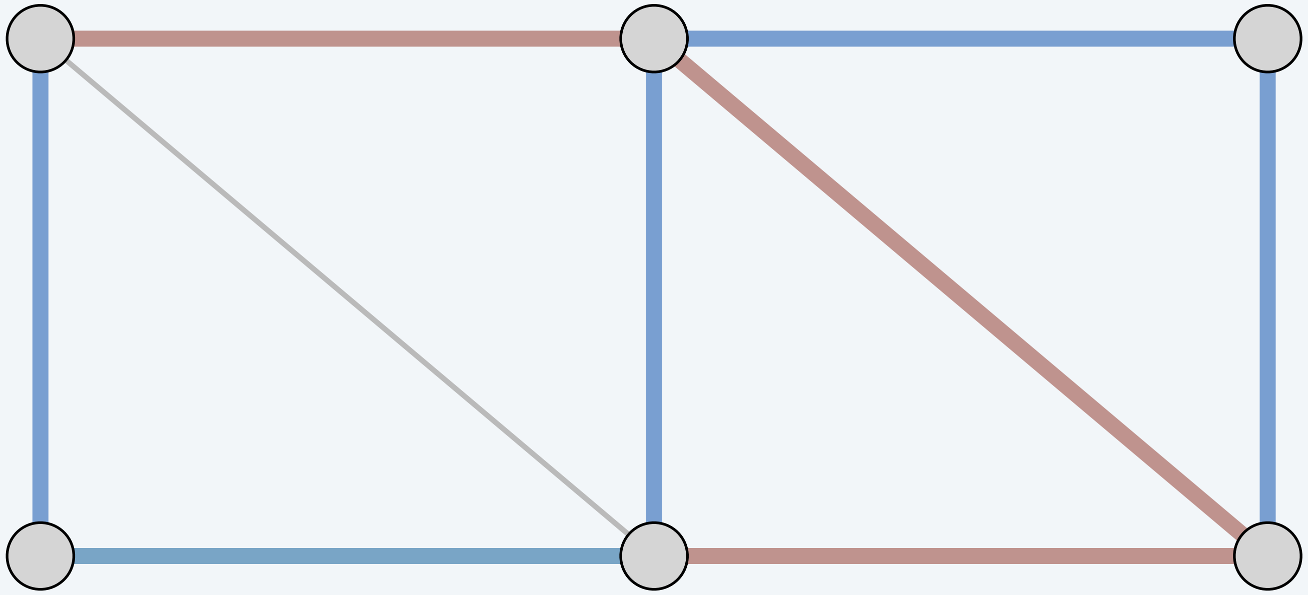
Blue rule. Let D be a cutset with no blue edges. Select an uncolored edge in D of min weight and color it blue.

apply the blue rule to the cutset



Red-rule blue-rule demo

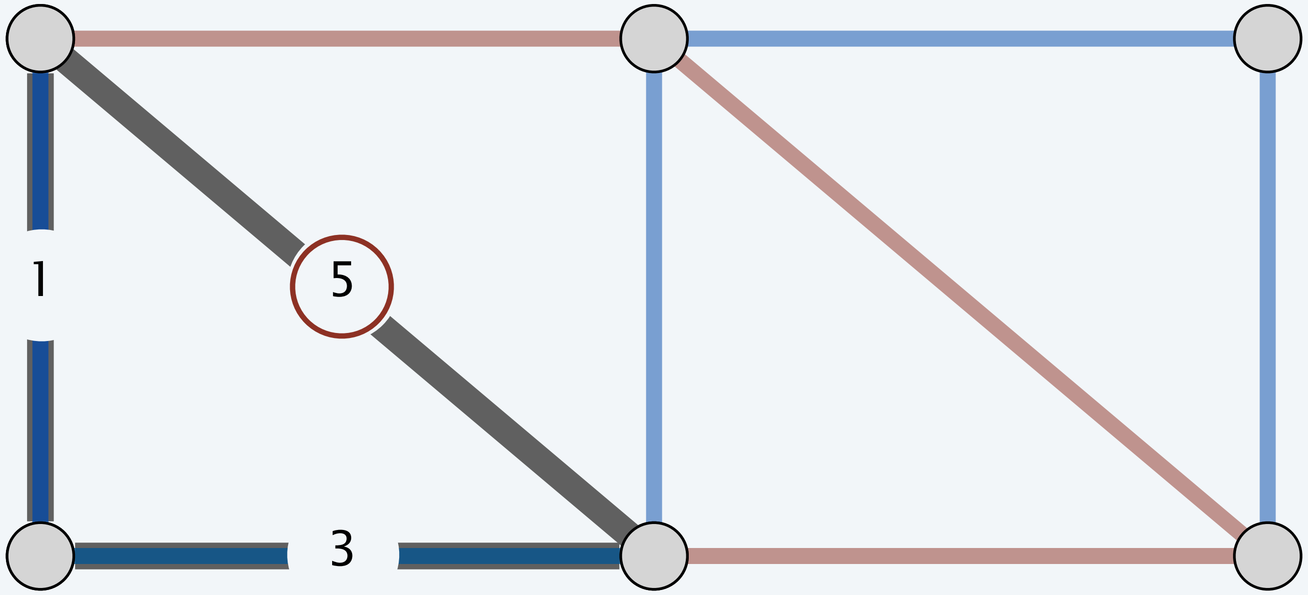
current set of red and blue edges



Red-rule blue-rule demo

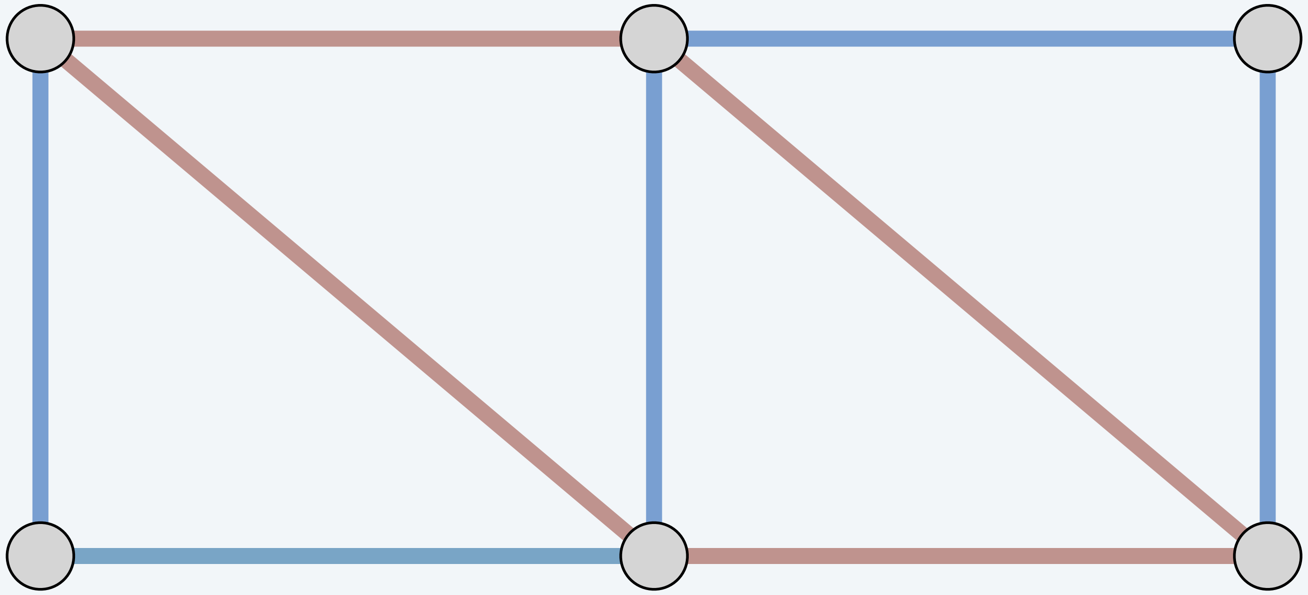
Blue rule. Let D be a cutset with no blue edges. Select an uncolored edge in D of min weight and color it blue.

apply the red rule to the cycle



Red-rule blue-rule demo

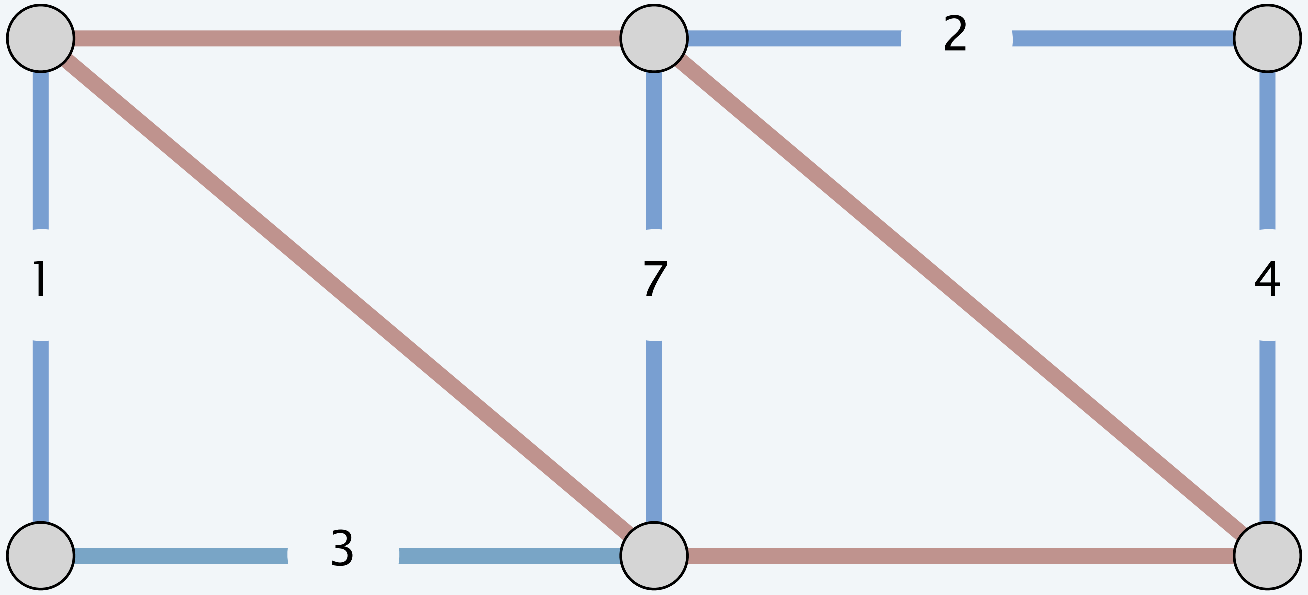
current set of red and blue edges

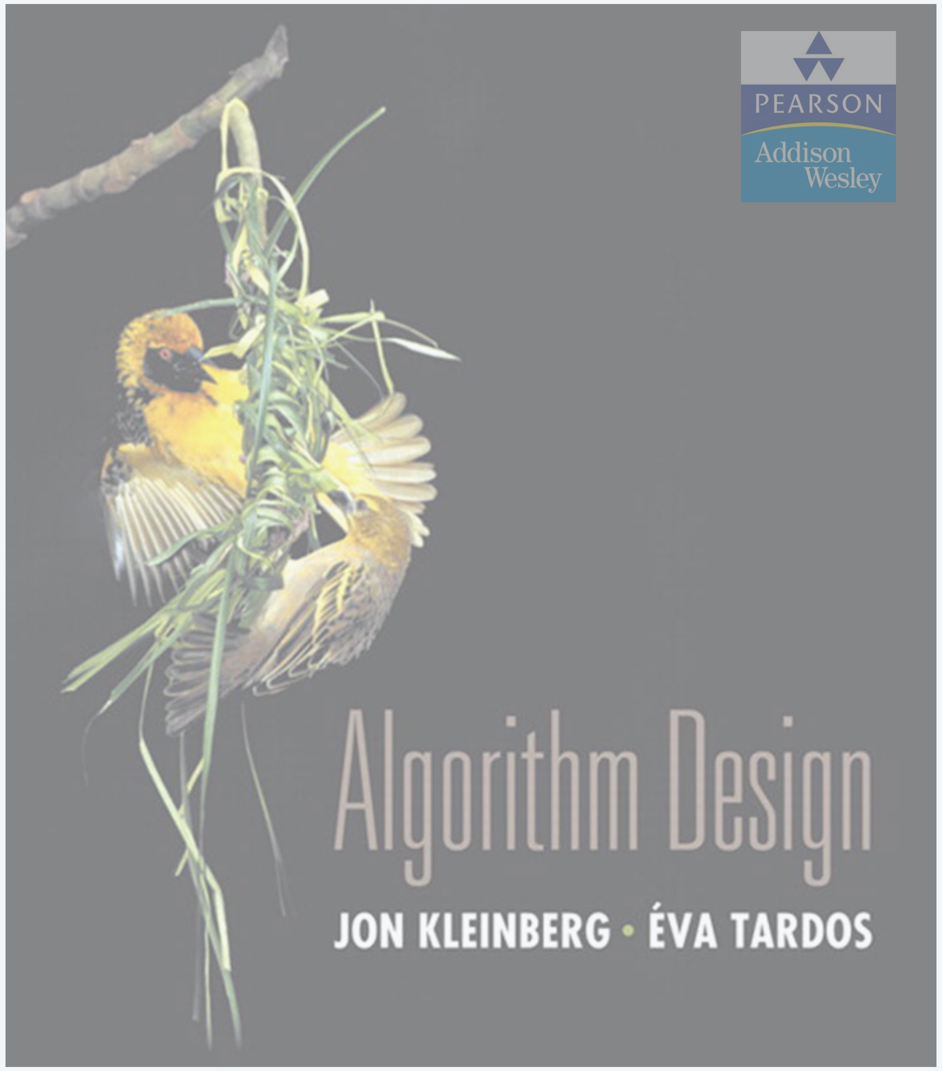


Red-rule blue-rule demo

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

a minimum spanning tree





SECTION 4.5

4. GREEDY ALGORITHMS II

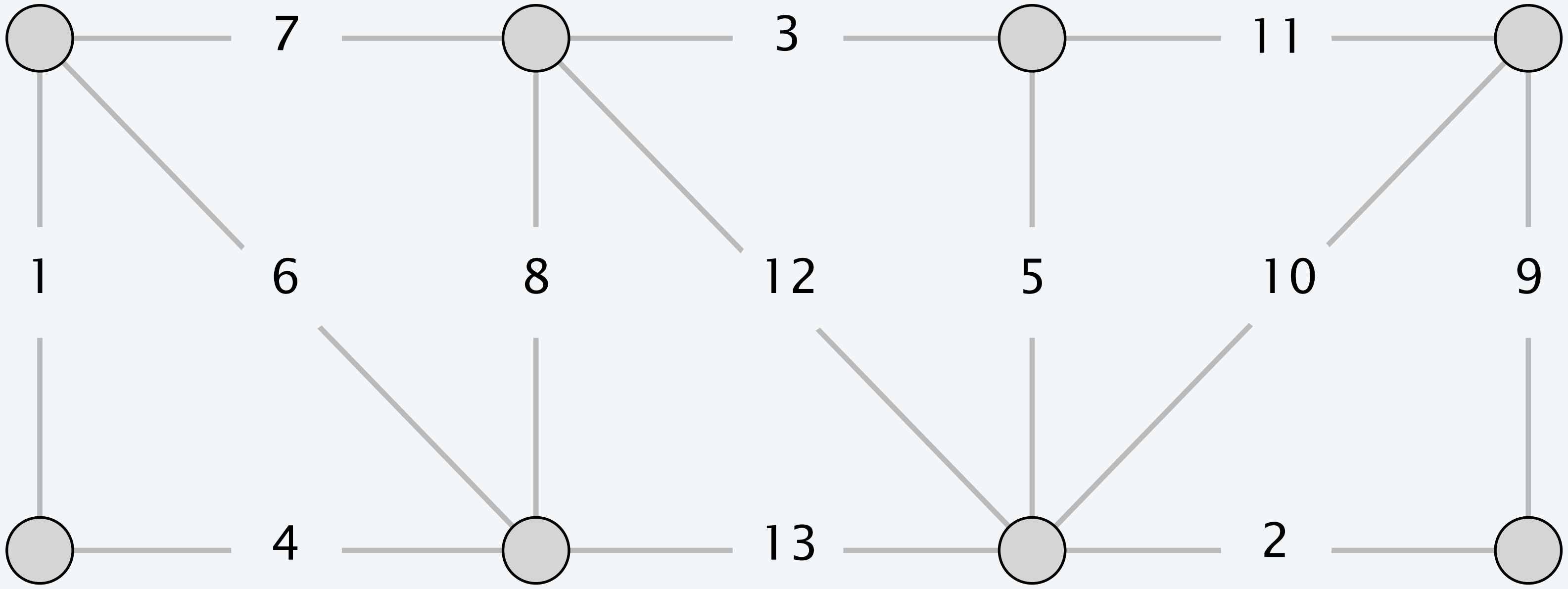
- *red-rule blue-rule demo*
- *Prim's algorithm demo*
- *Kruskal's algorithm demo*
- *reverse-delete algorithm demo*
- *Boruvka's algorithm demo*

Prim's algorithm demo

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

Repeat $n - 1$ times:

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

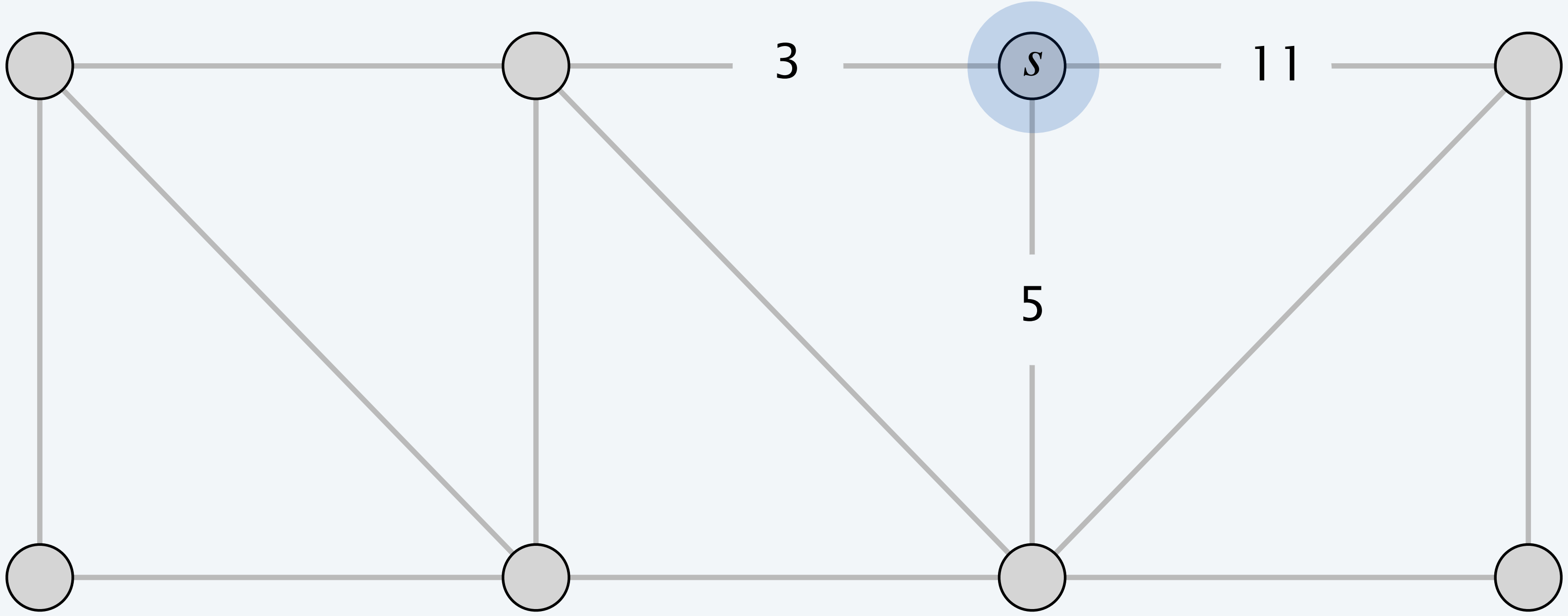


Prim's algorithm demo

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

Repeat $n - 1$ times:

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

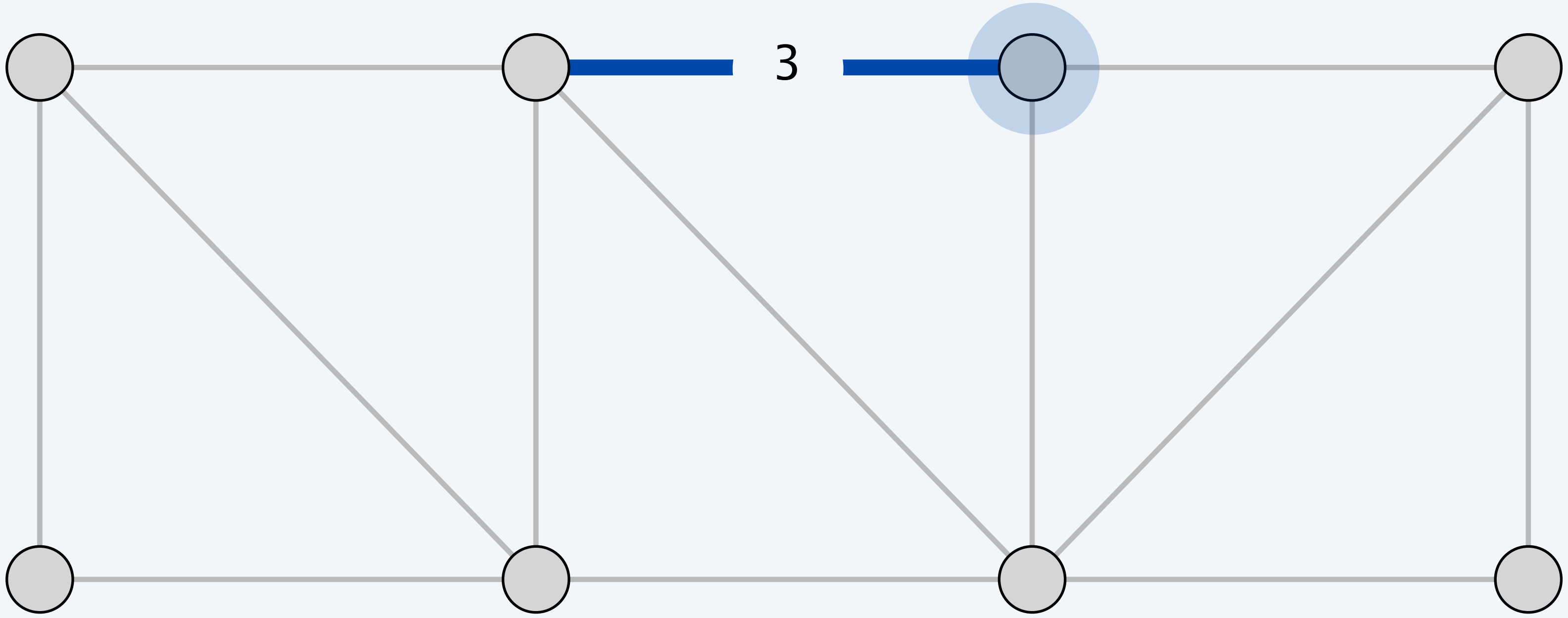


Prim's algorithm demo

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

Repeat $n - 1$ times:

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

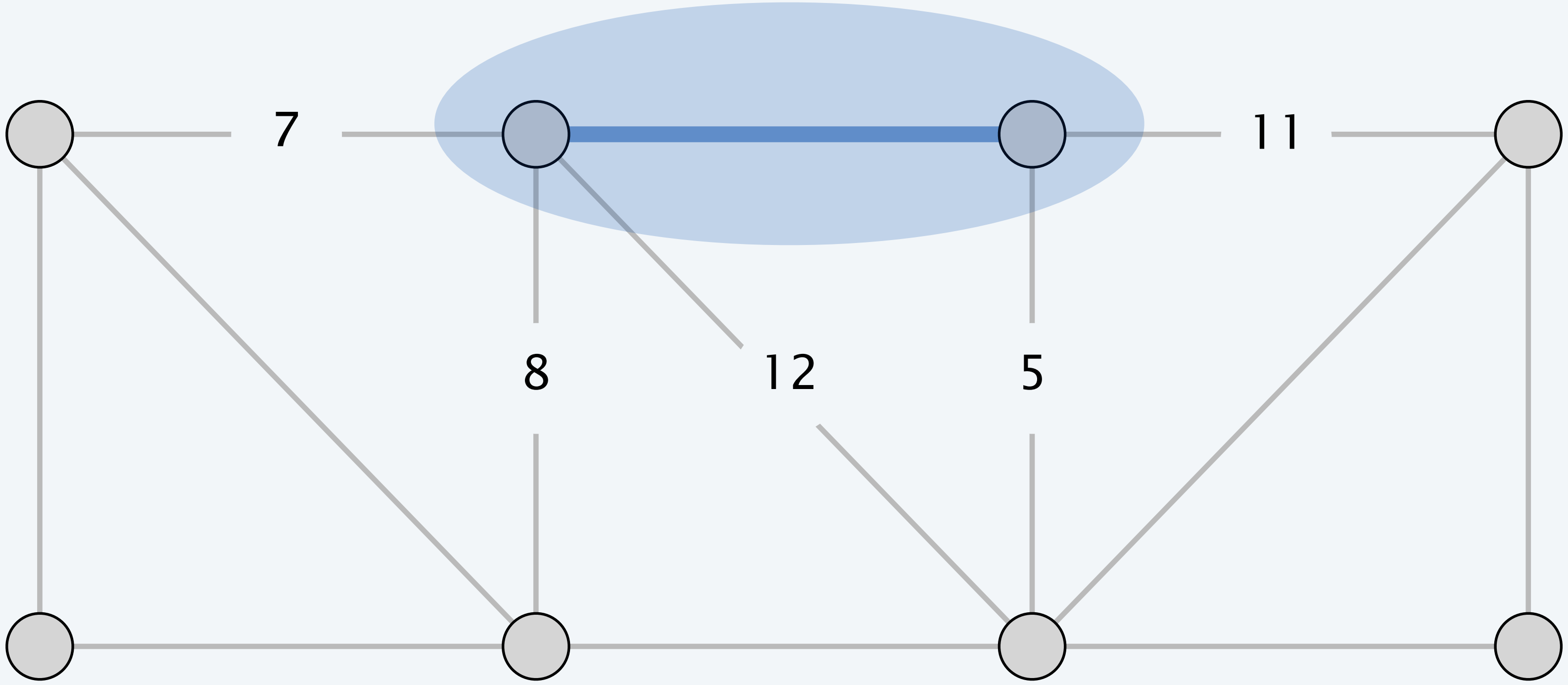


Prim's algorithm demo

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

Repeat $n - 1$ times:

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

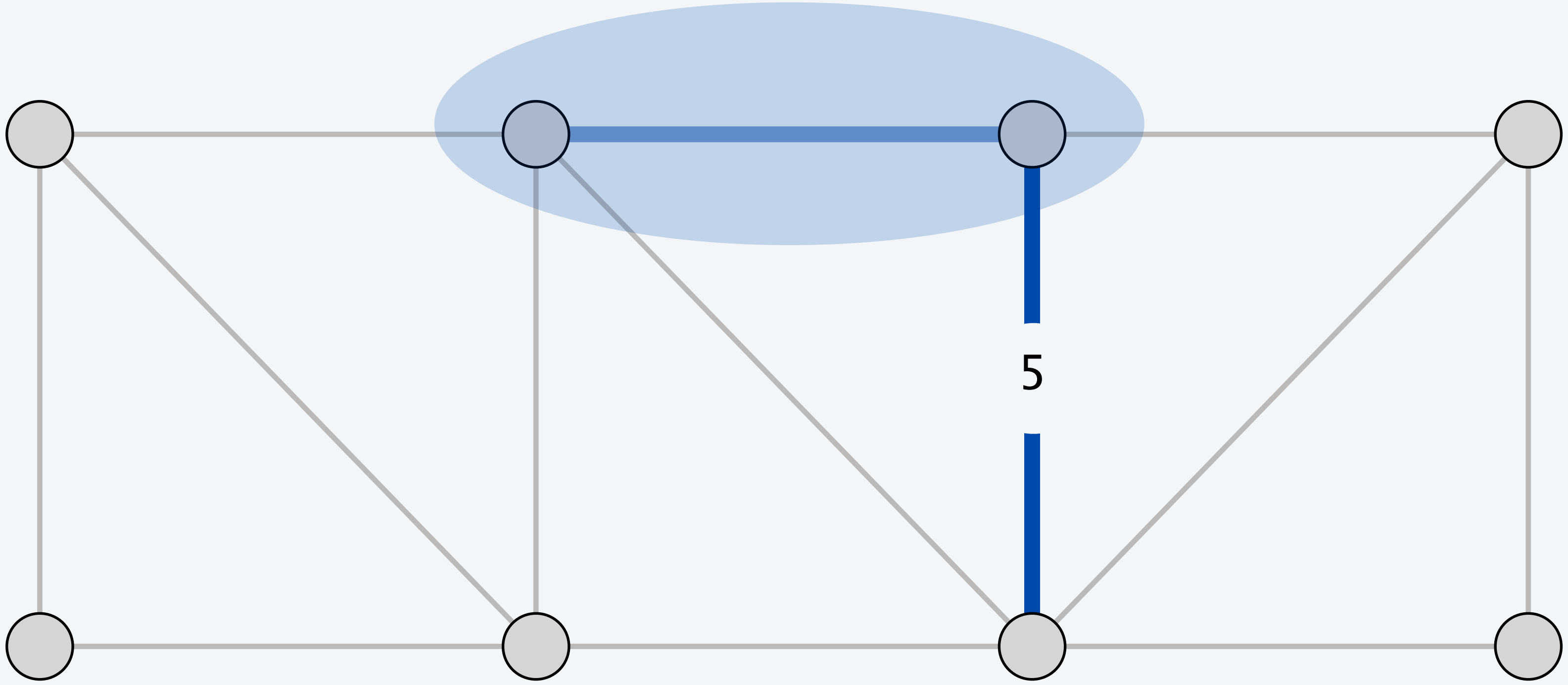


Prim's algorithm demo

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

Repeat $n - 1$ times:

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

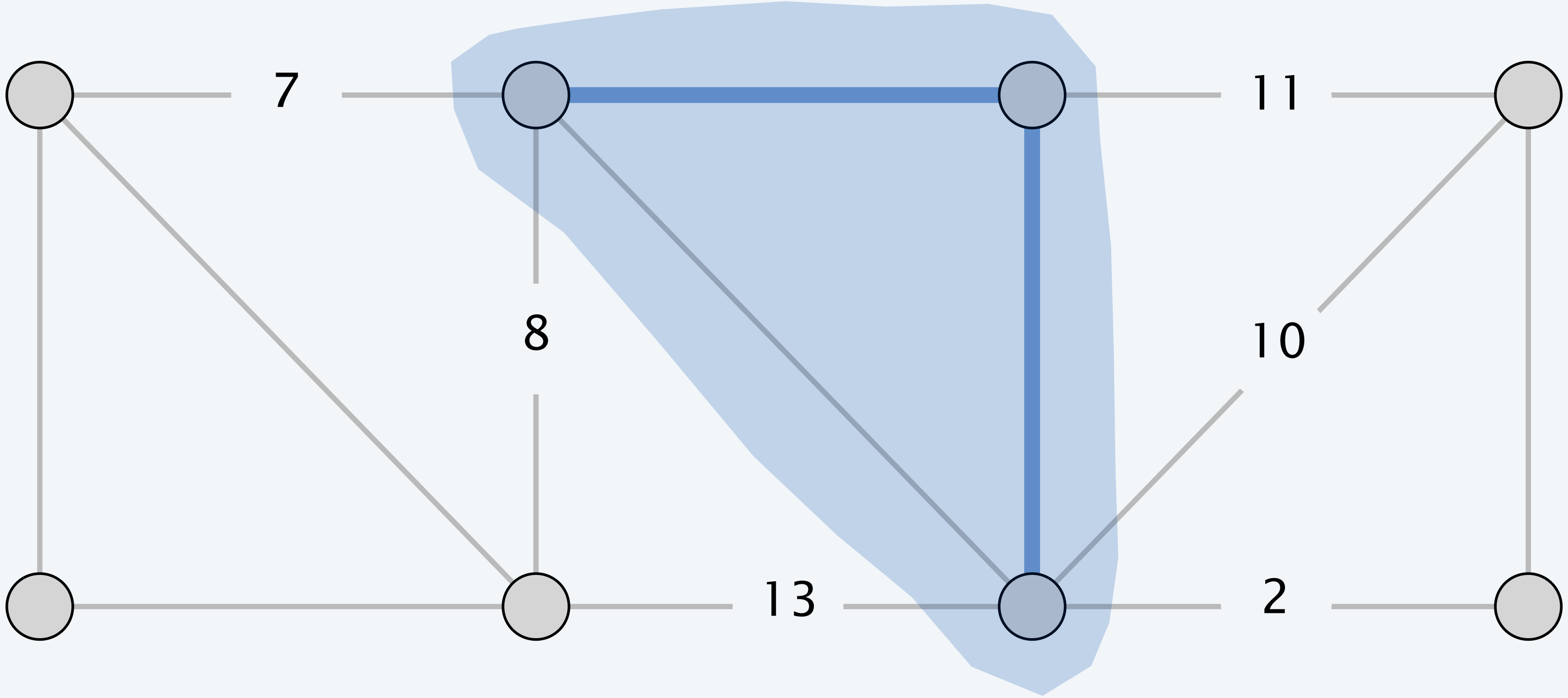


Prim's algorithm demo

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

Repeat $n - 1$ times:

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

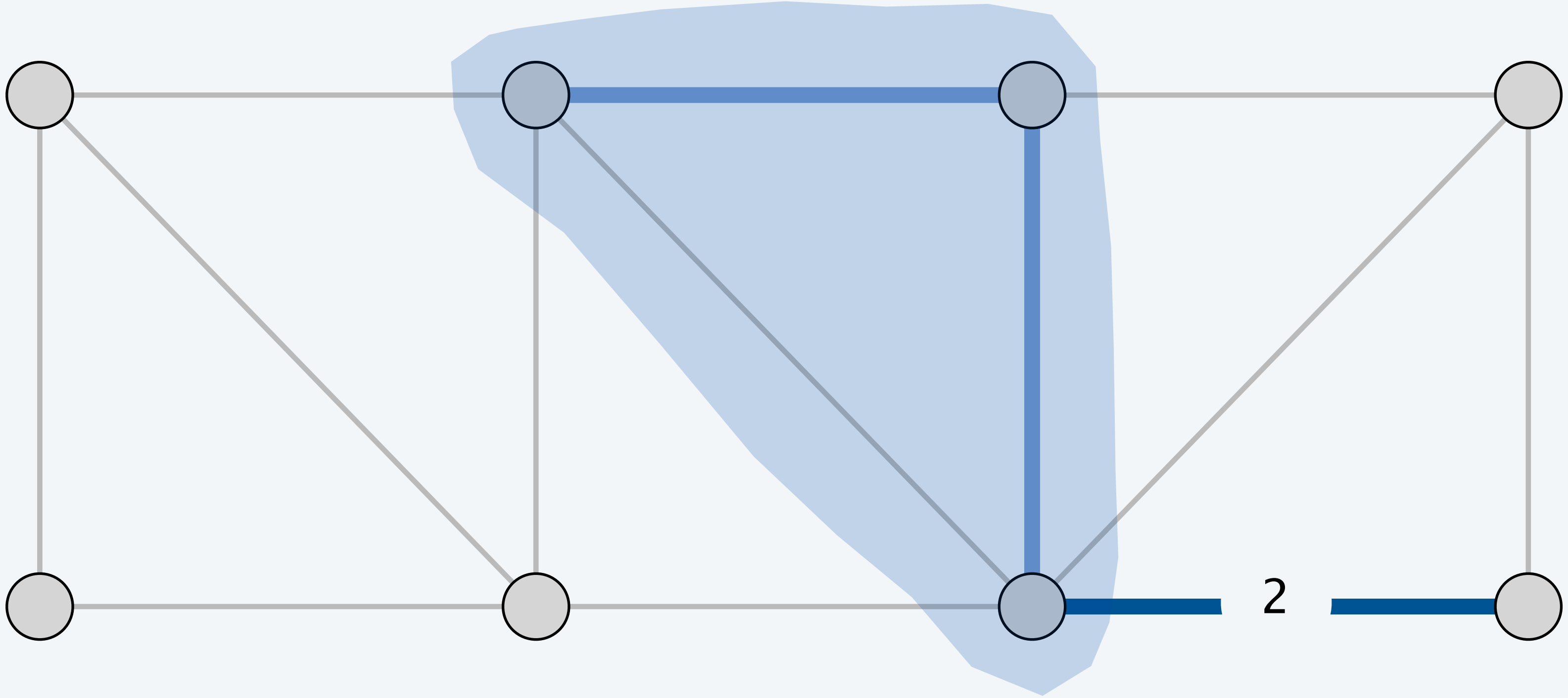


Prim's algorithm demo

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

Repeat $n - 1$ times:

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

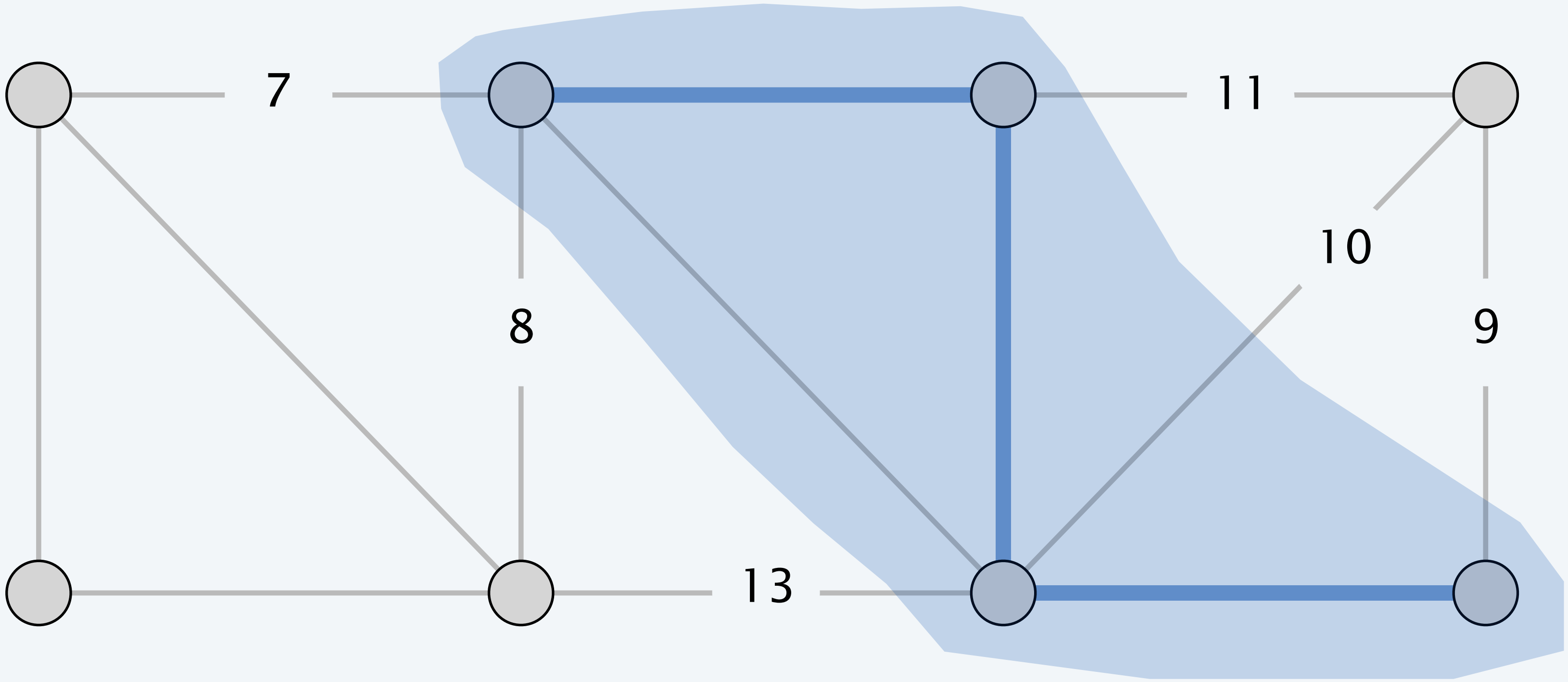


Prim's algorithm demo

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

Repeat $n - 1$ times:

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

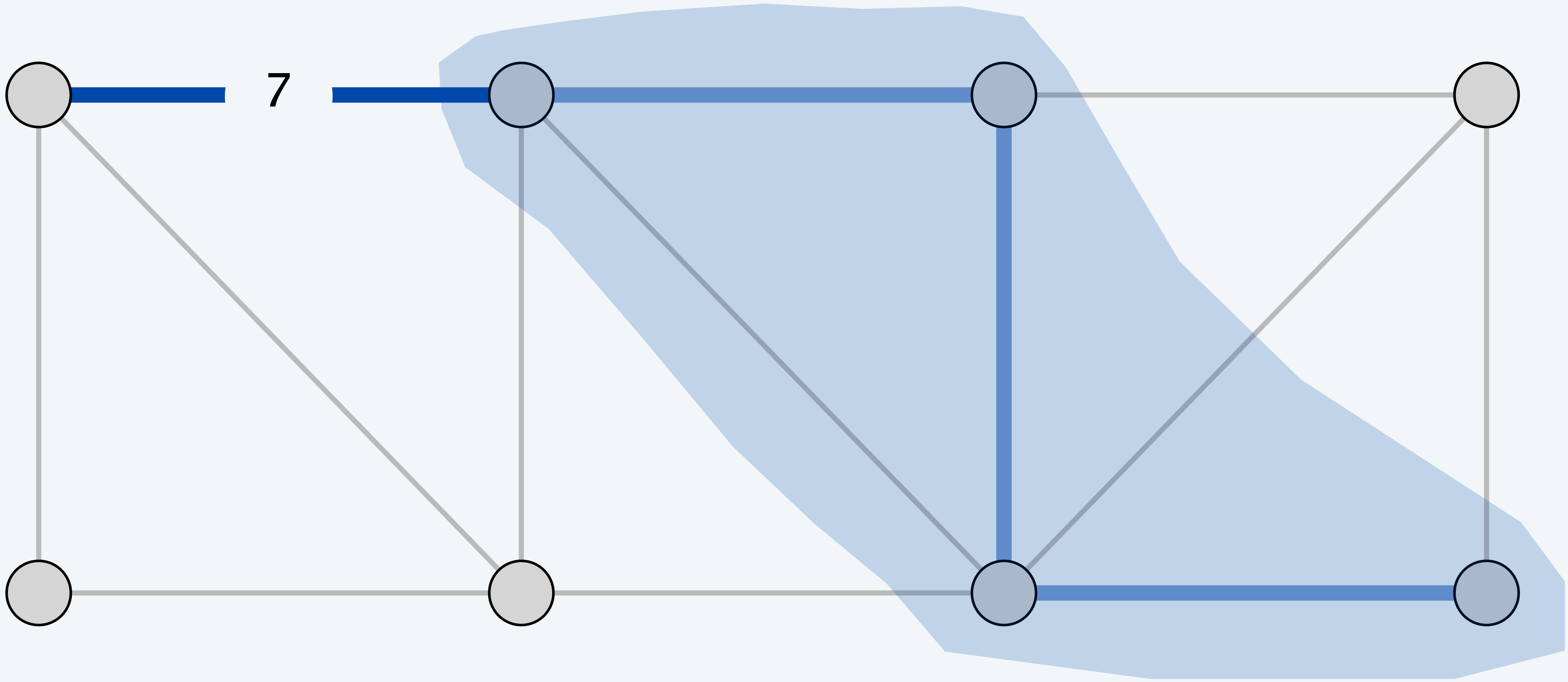


Prim's algorithm demo

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

Repeat $n - 1$ times:

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

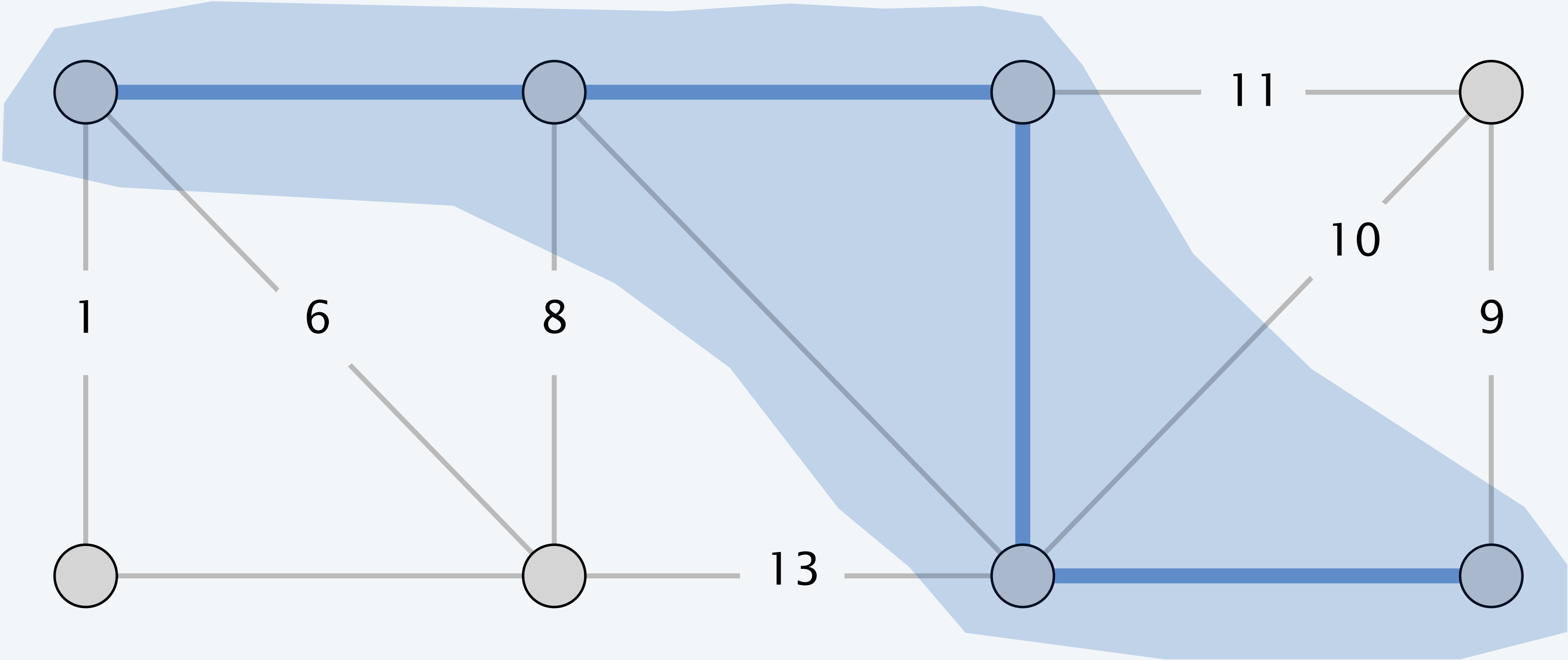


Prim's algorithm demo

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

Repeat $n - 1$ times:

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

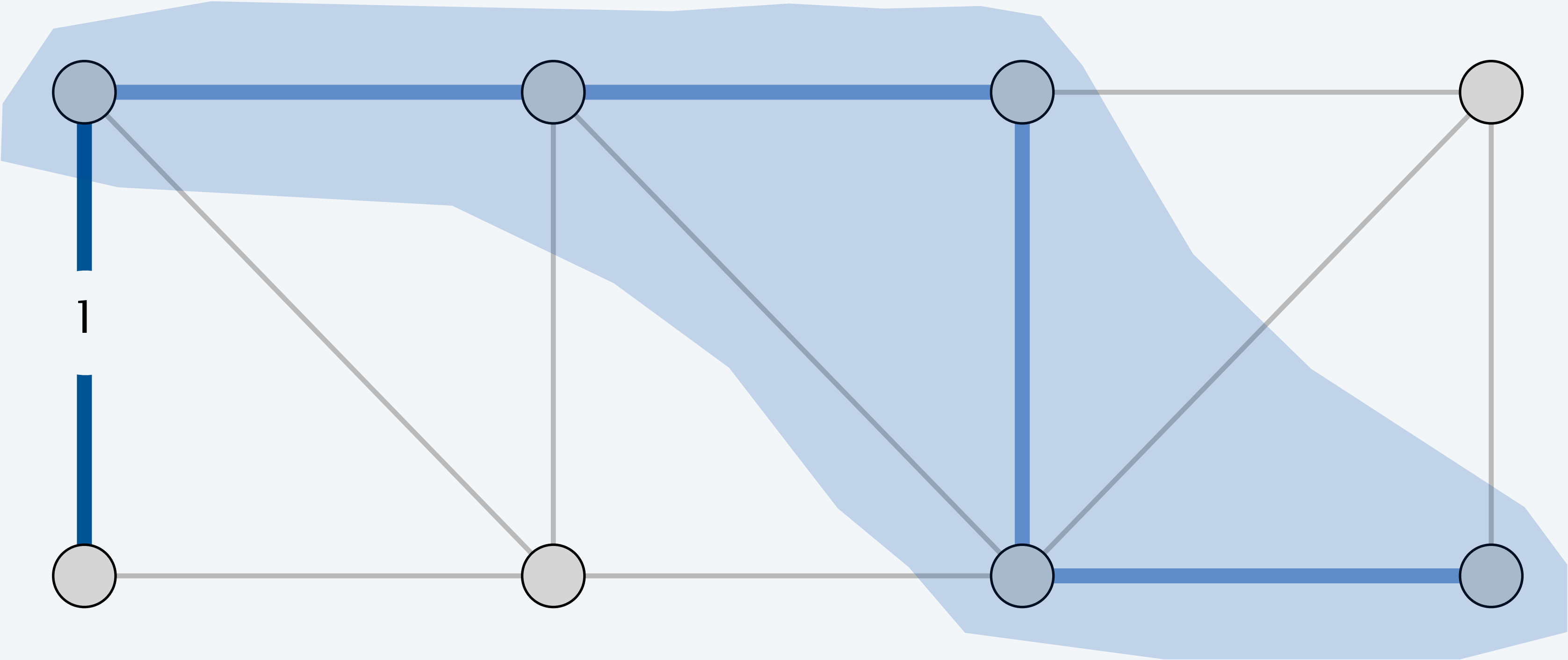


Prim's algorithm demo

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

Repeat $n - 1$ times:

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

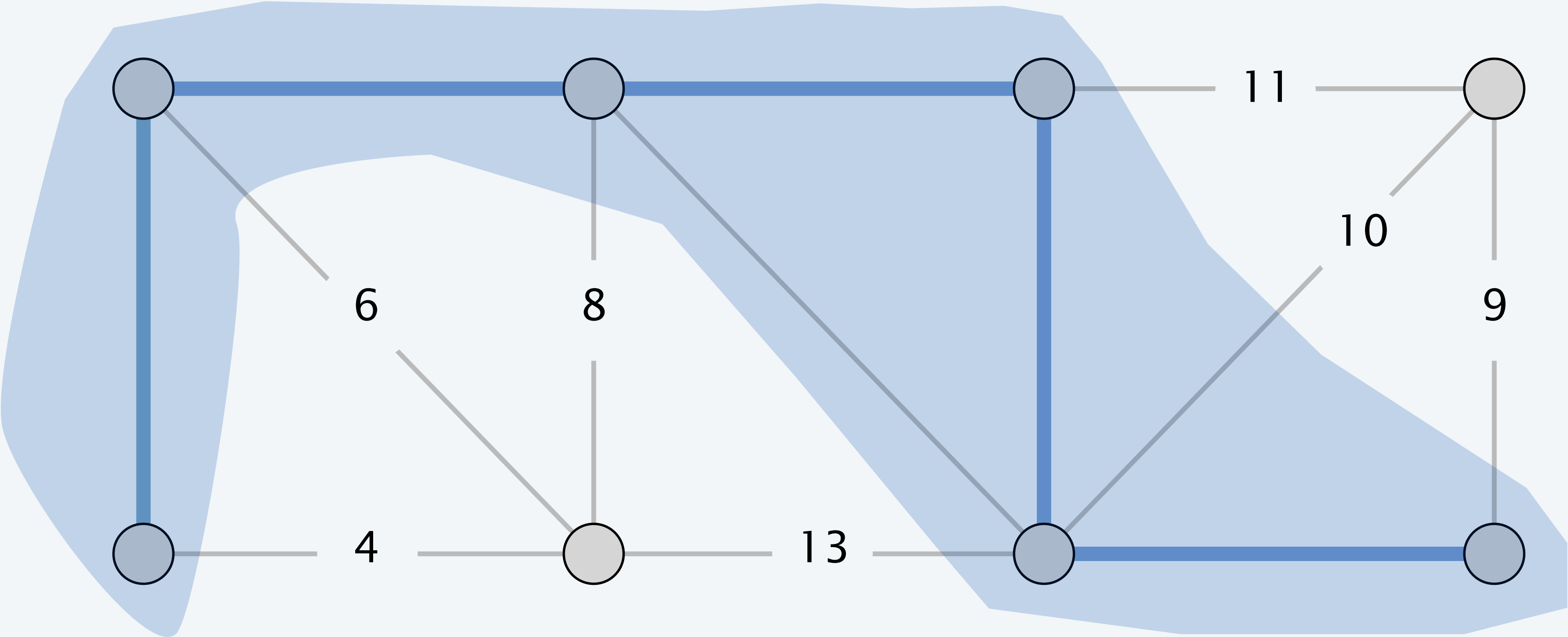


Prim's algorithm demo

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

Repeat $n - 1$ times:

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

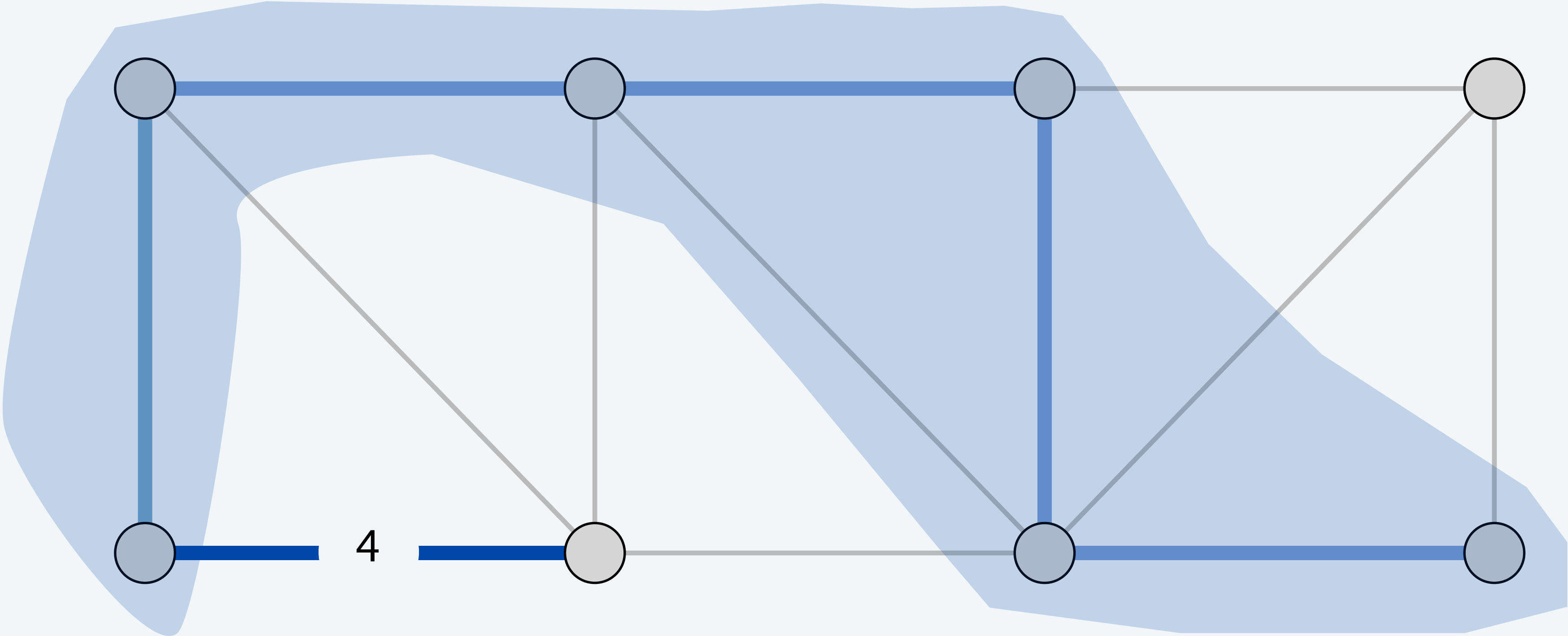


Prim's algorithm demo

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

Repeat $n - 1$ times:

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

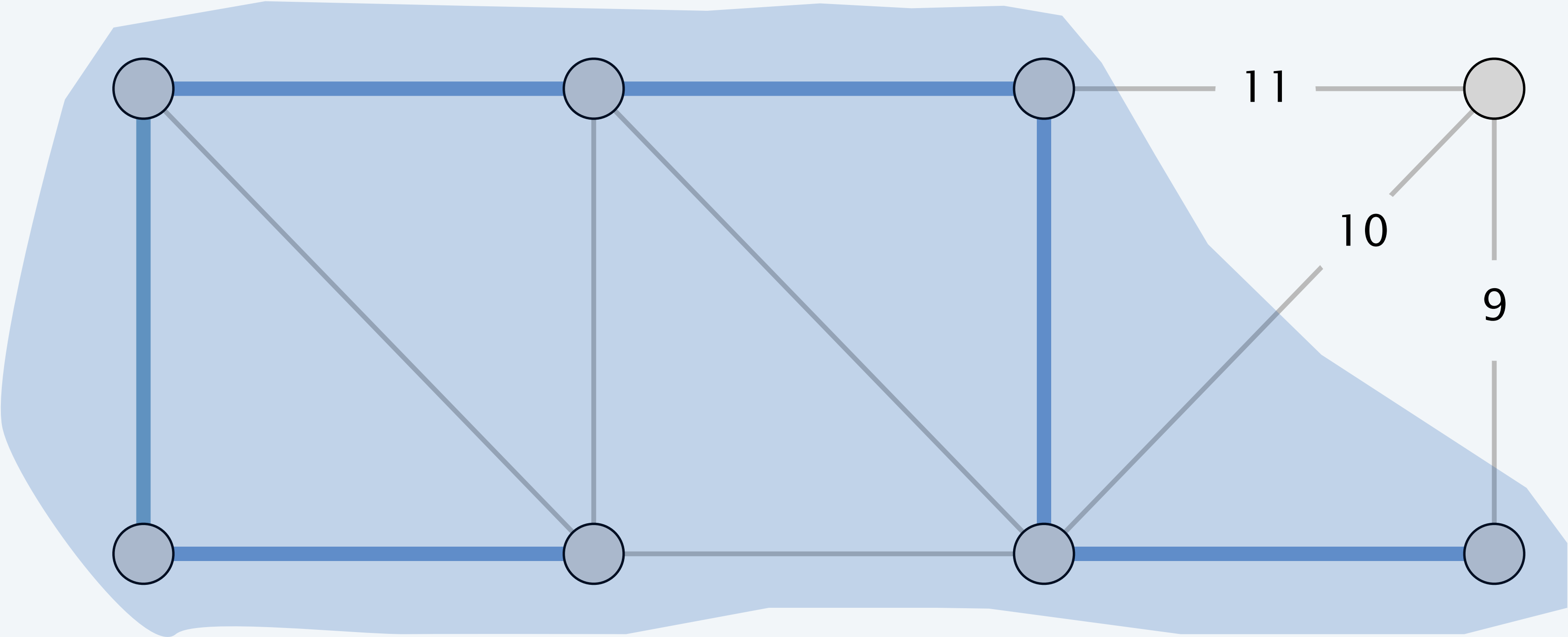


Prim's algorithm demo

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

Repeat $n - 1$ times:

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

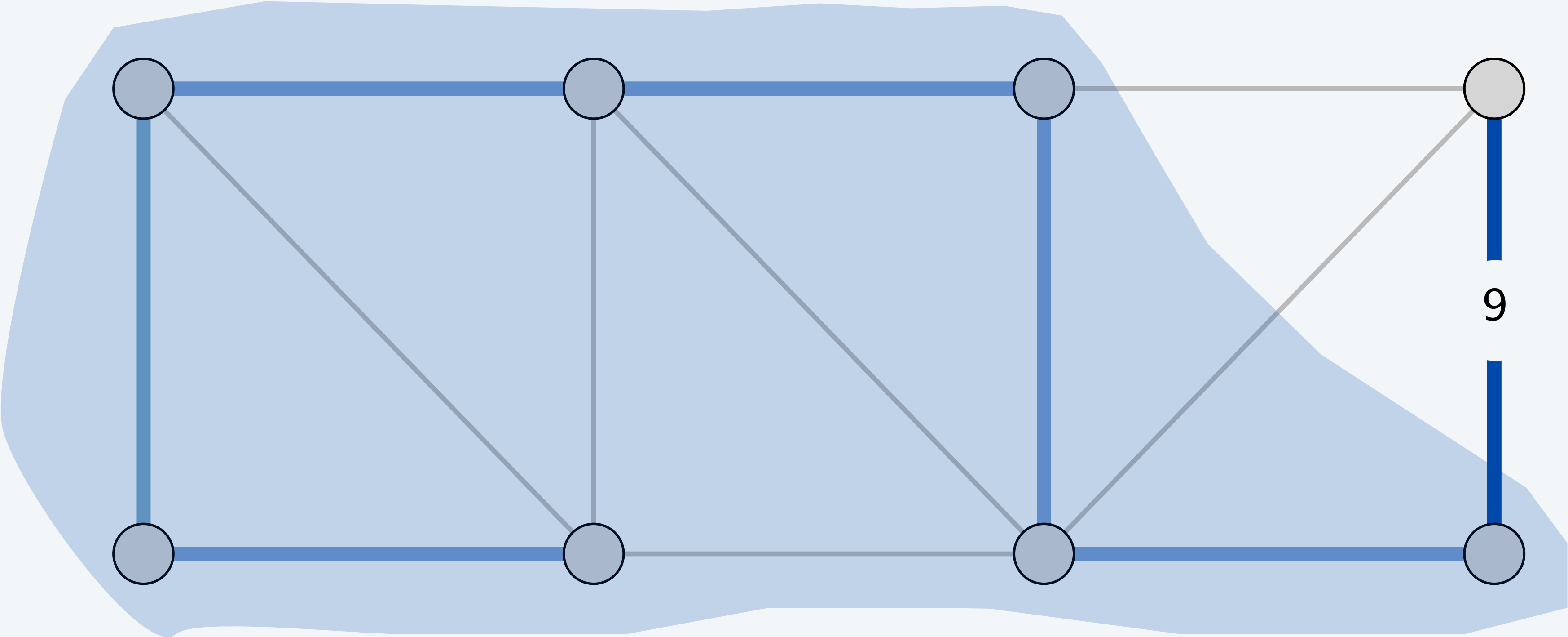


Prim's algorithm demo

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

Repeat $n - 1$ times:

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

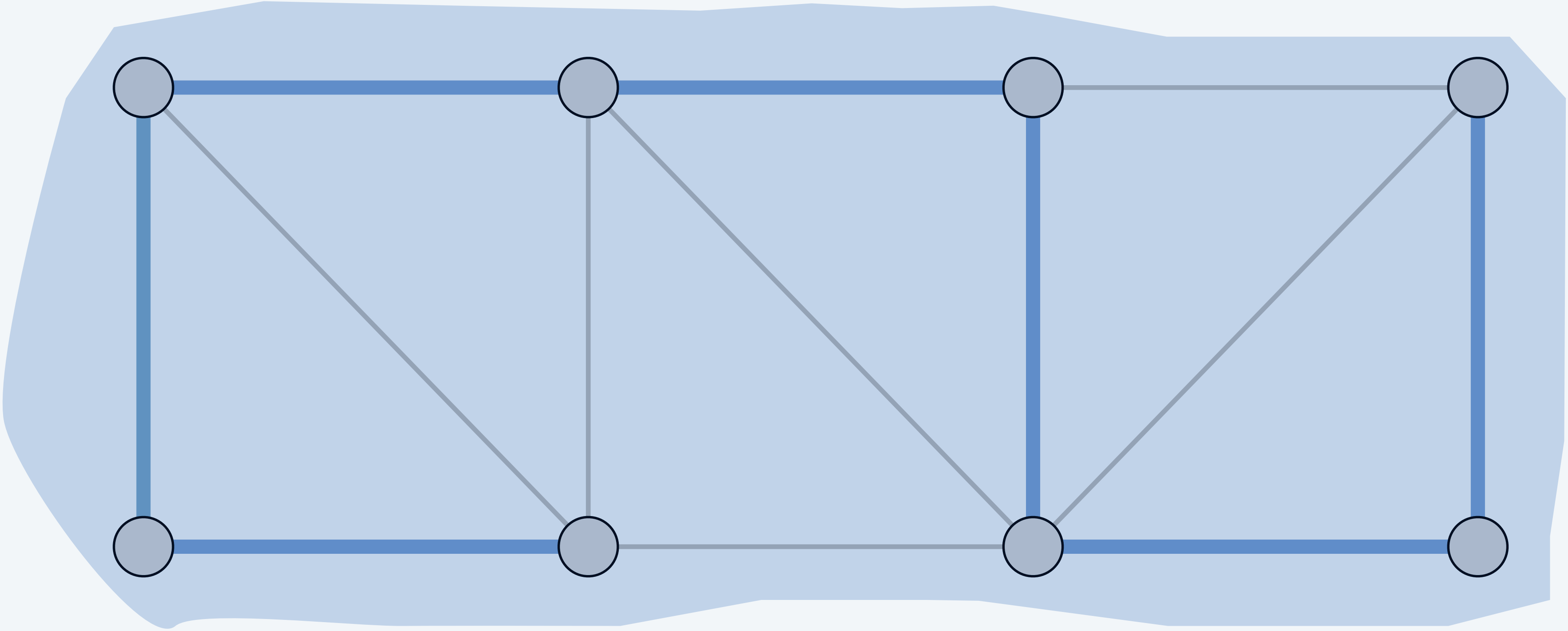


Prim's algorithm demo

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

Repeat $n - 1$ times:

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

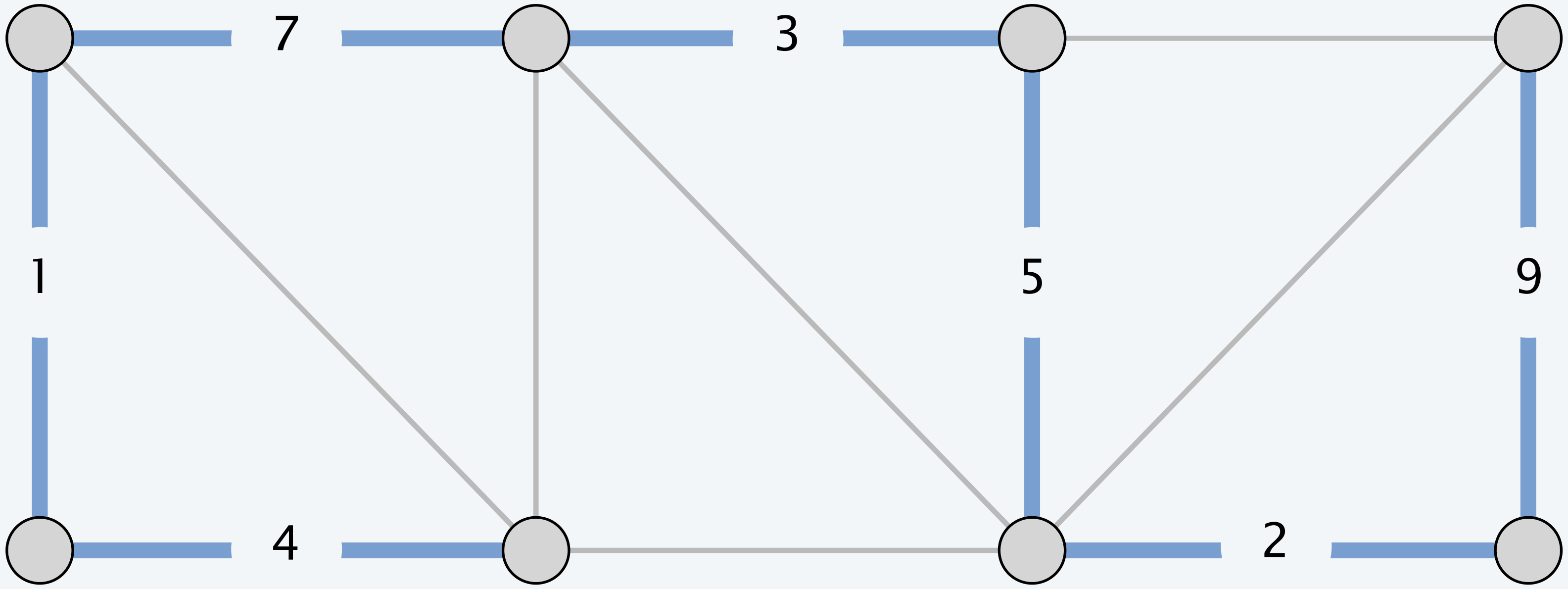


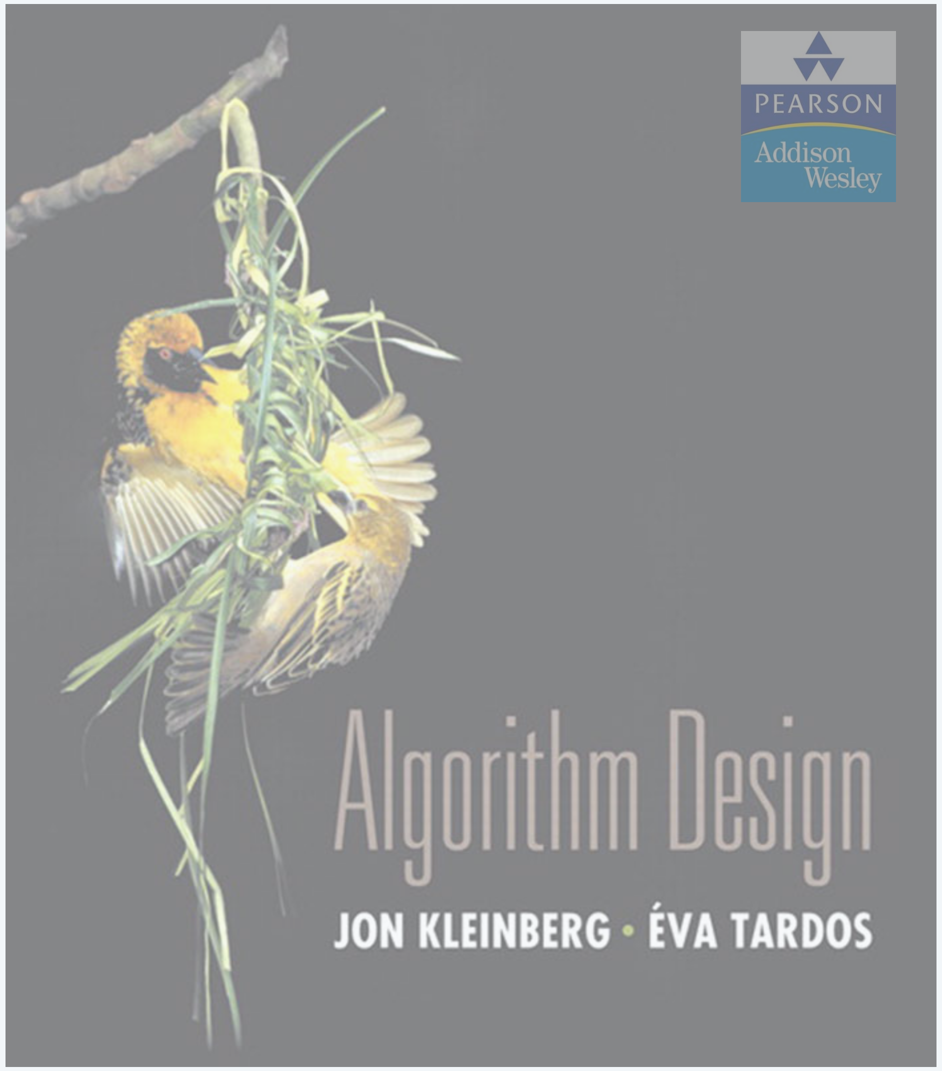
Prim's algorithm demo

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

Repeat $n - 1$ times:

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





Section 4.5

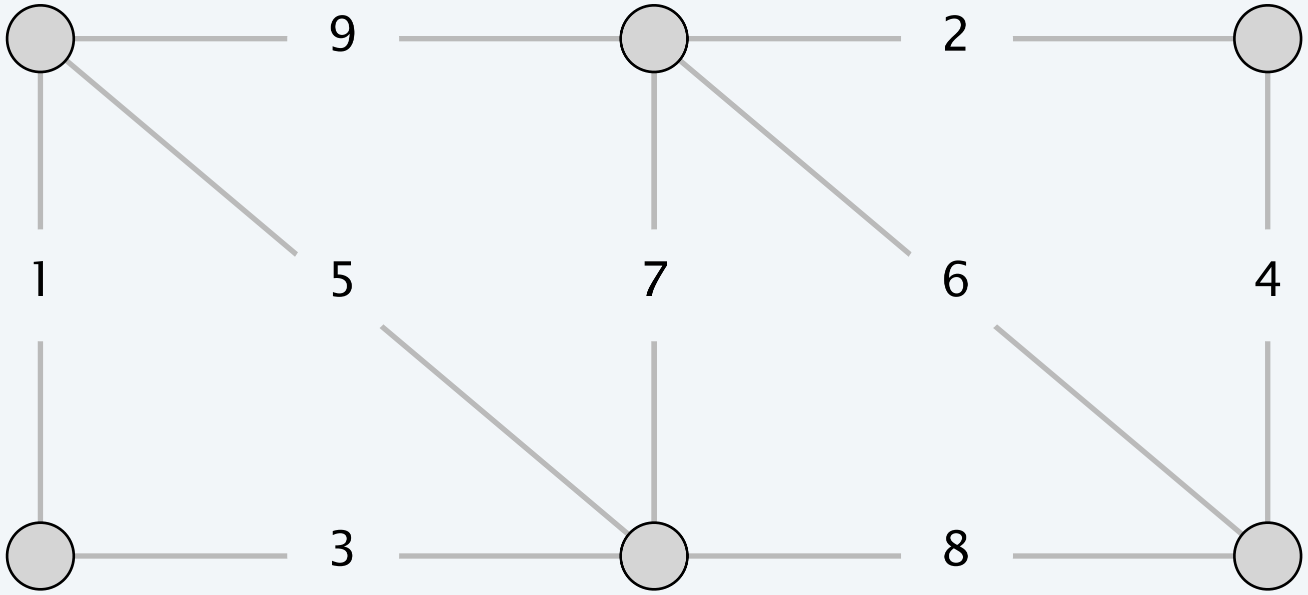
4. Greedy Algorithms II

- 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:

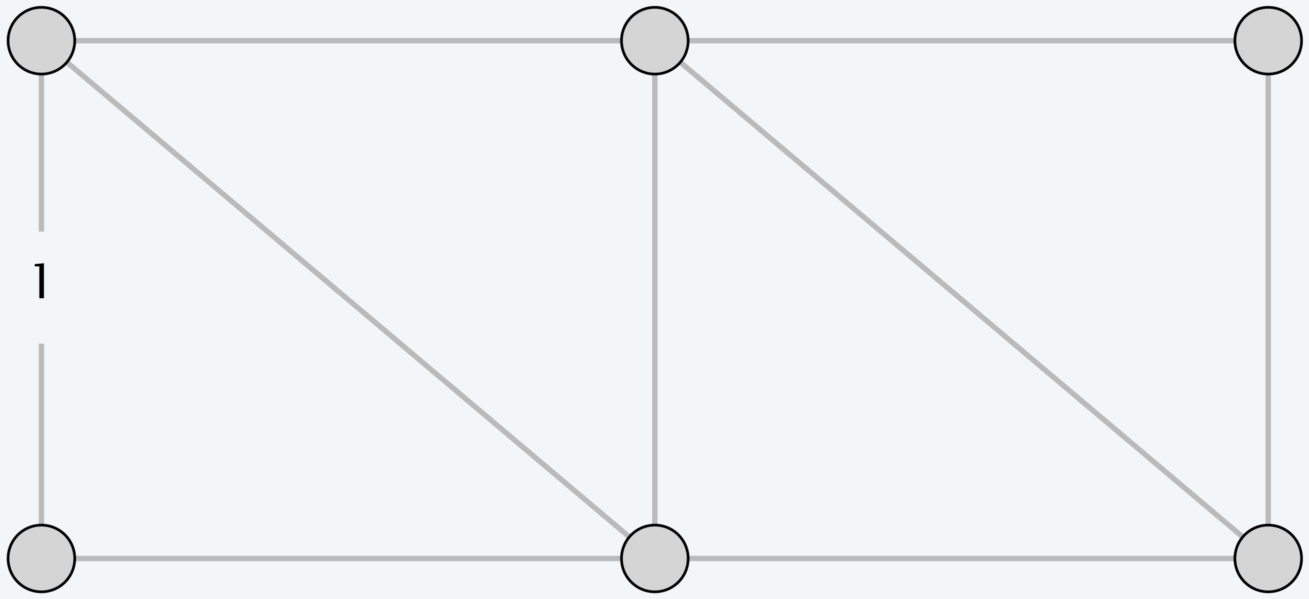
- Add to T unless it would create a cycle.



Kruskal’s algorithm demo

Consider edges in ascending order of weight:

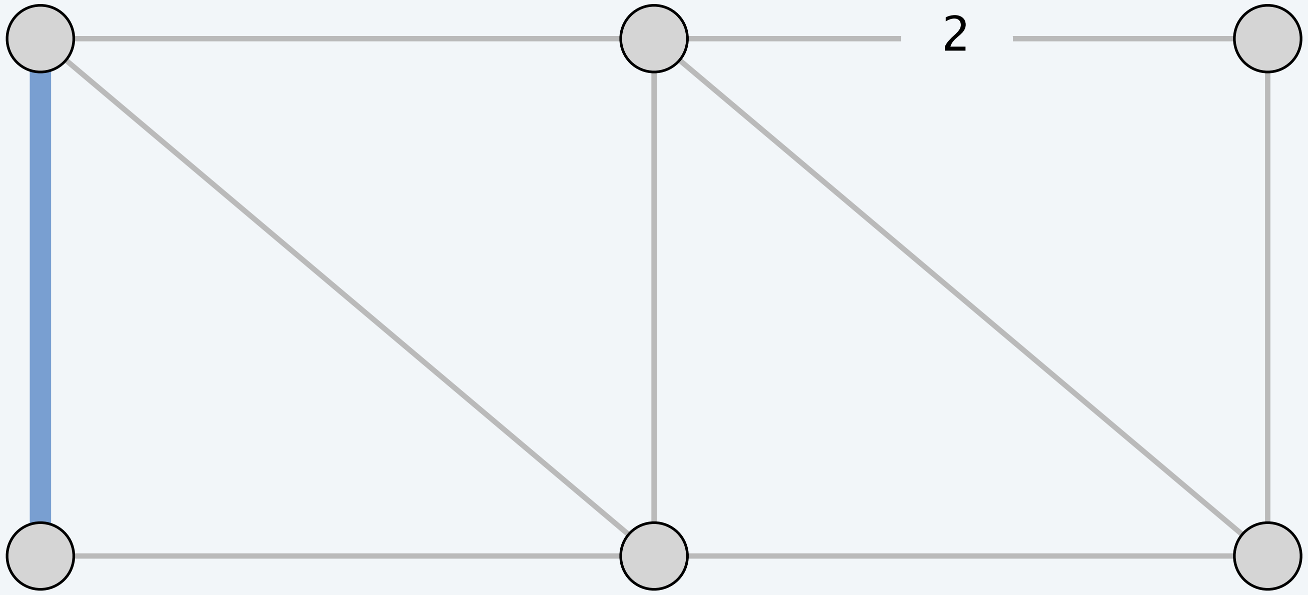
- Add to T unless it would create a cycle.



Kruskal's algorithm demo

Consider edges in ascending order of weight:

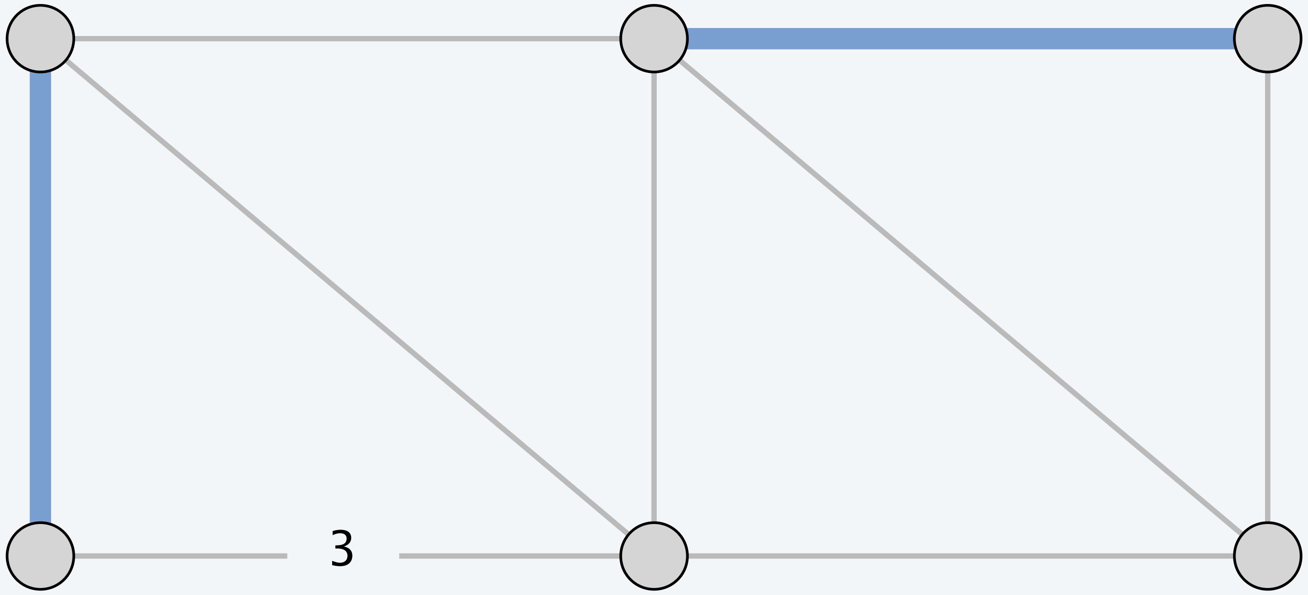
- Add to T unless it would create a cycle.



Kruskal's algorithm demo

Consider edges in ascending order of weight:

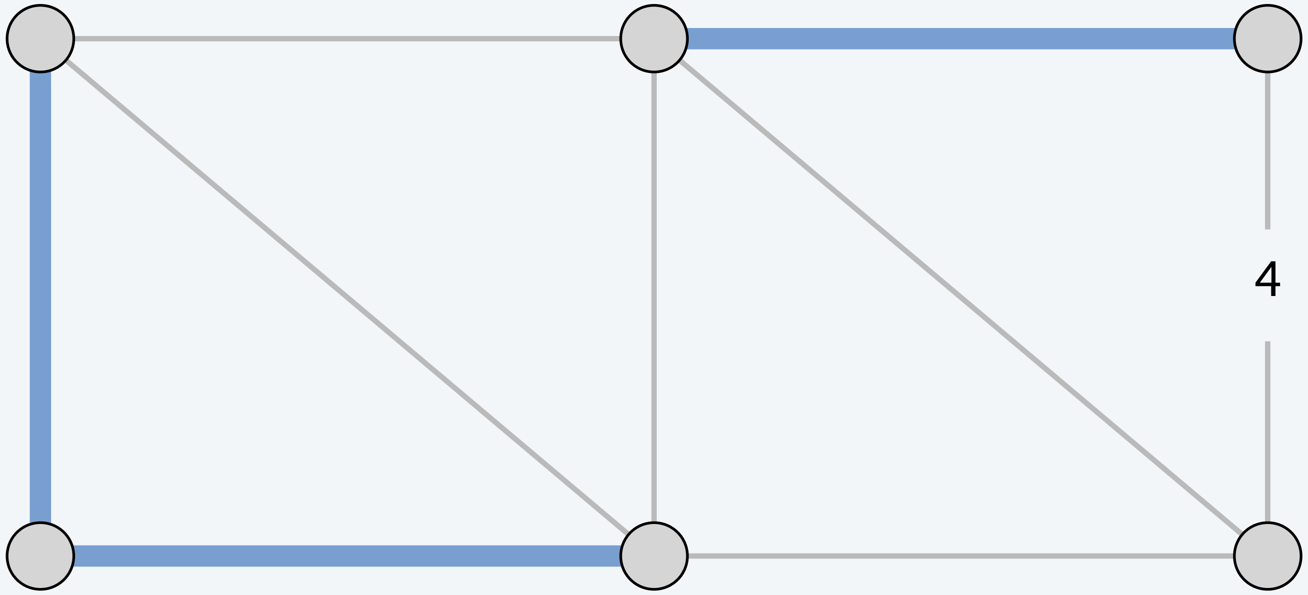
- Add to T unless it would create a cycle.



Kruskal's algorithm demo

Consider edges in ascending order of weight:

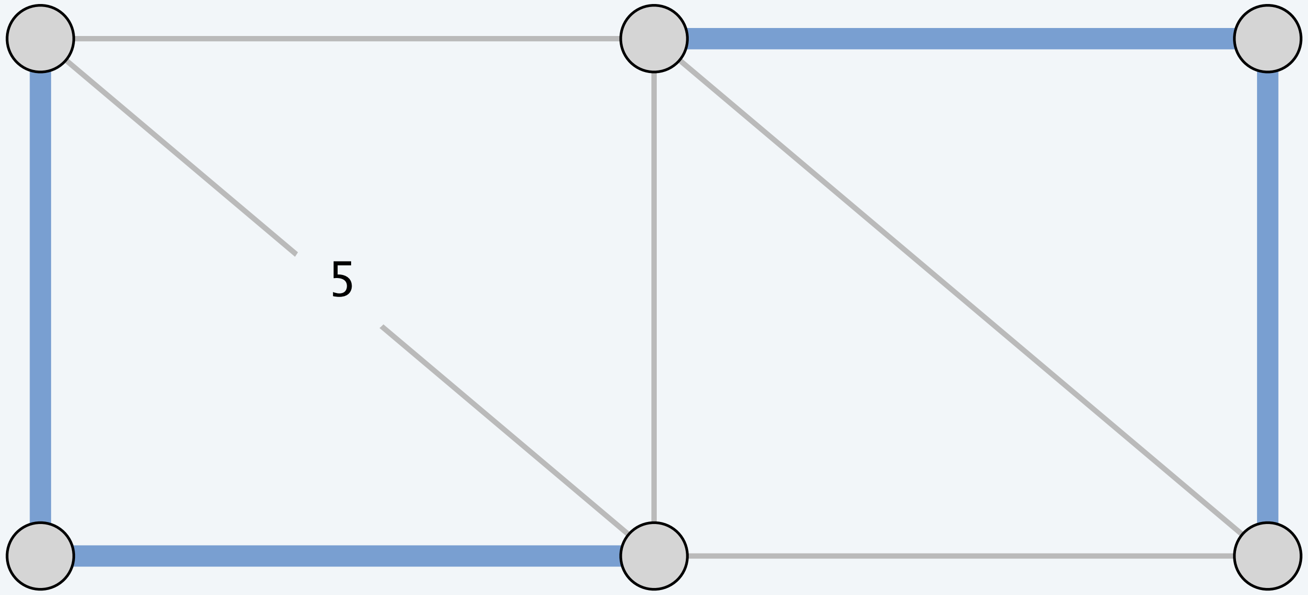
- Add to T unless it would create a cycle.



Kruskal's algorithm demo

Consider edges in ascending order of weight:

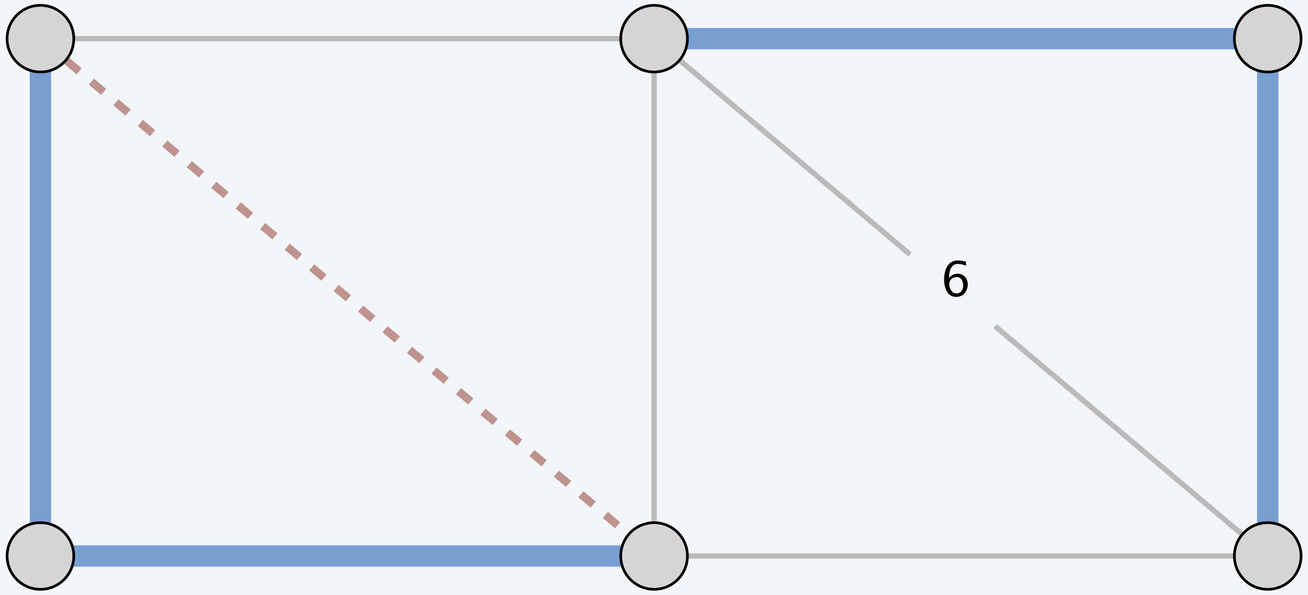
- Add to T unless it would create a cycle.



Kruskal's algorithm demo

Consider edges in ascending order of weight:

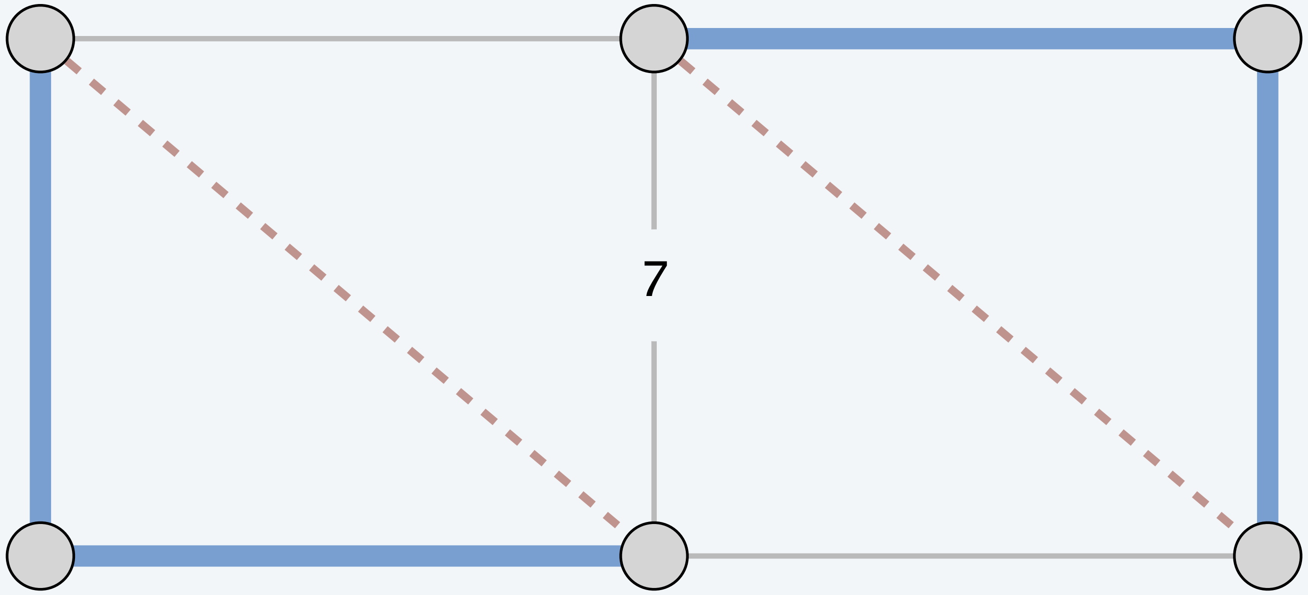
- Add to T unless it would create a cycle.



Kruskal's algorithm demo

Consider edges in ascending order of weight:

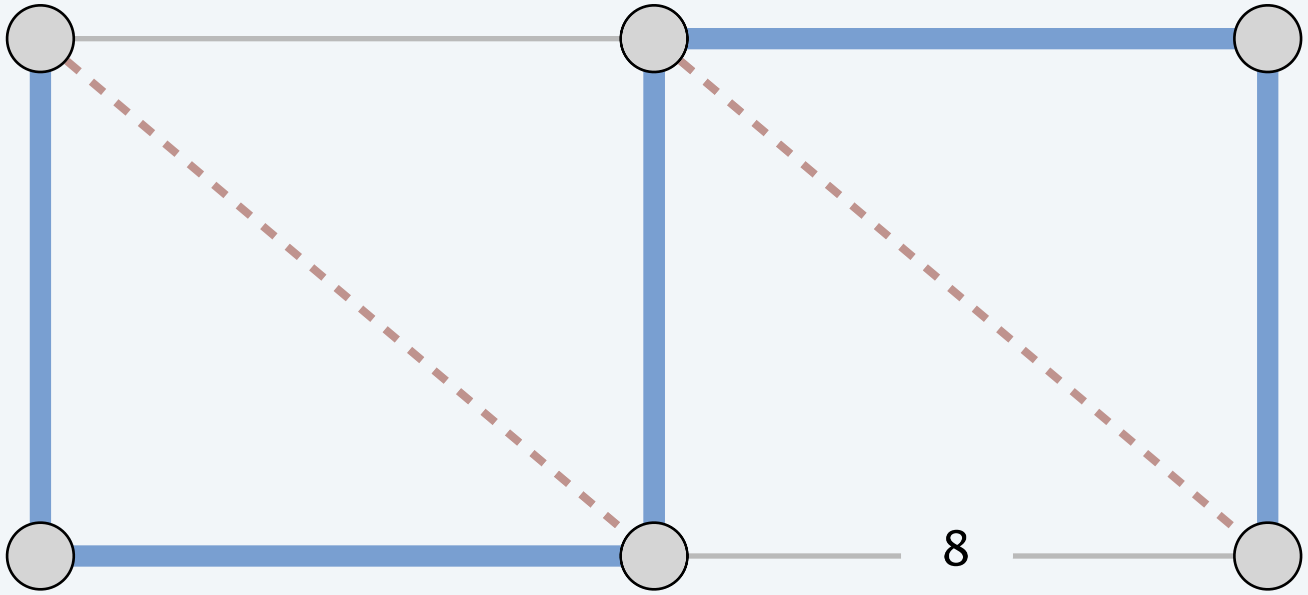
- Add to T unless it would create a cycle.



Kruskal's algorithm demo

Consider edges in ascending order of weight:

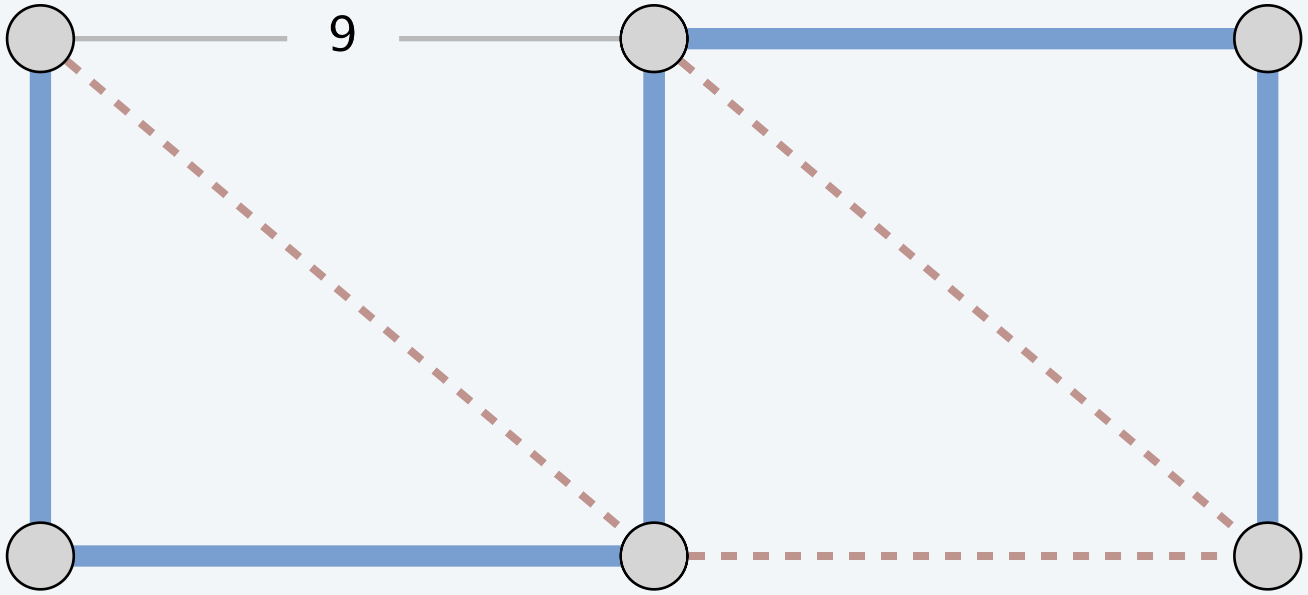
- Add to T unless it would create a cycle.



Kruskal's algorithm demo

Consider edges in ascending order of weight:

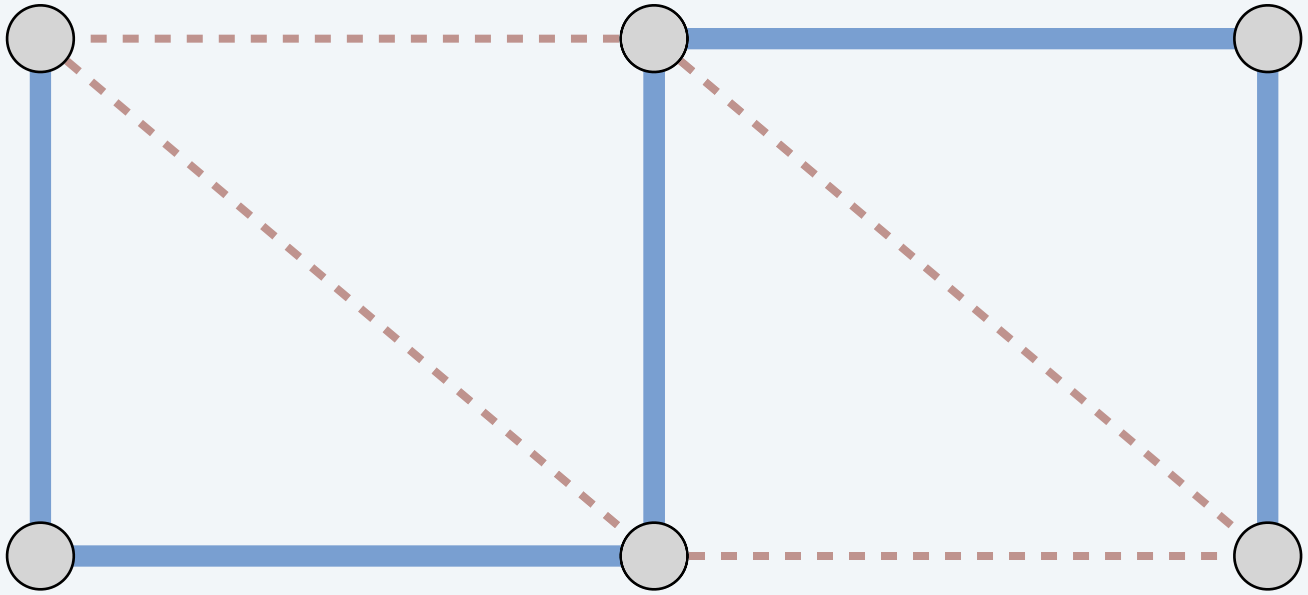
- Add to T unless it would create a cycle.



Kruskal's algorithm demo

Consider edges in ascending order of weight:

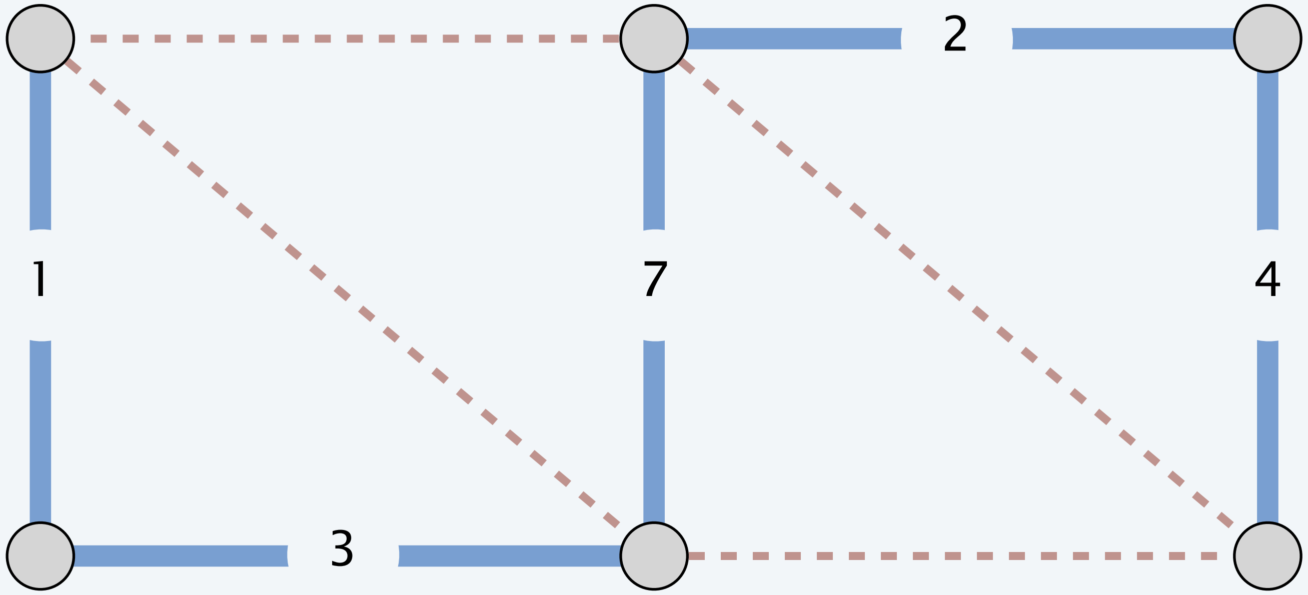
- Add to T unless it would create a cycle.

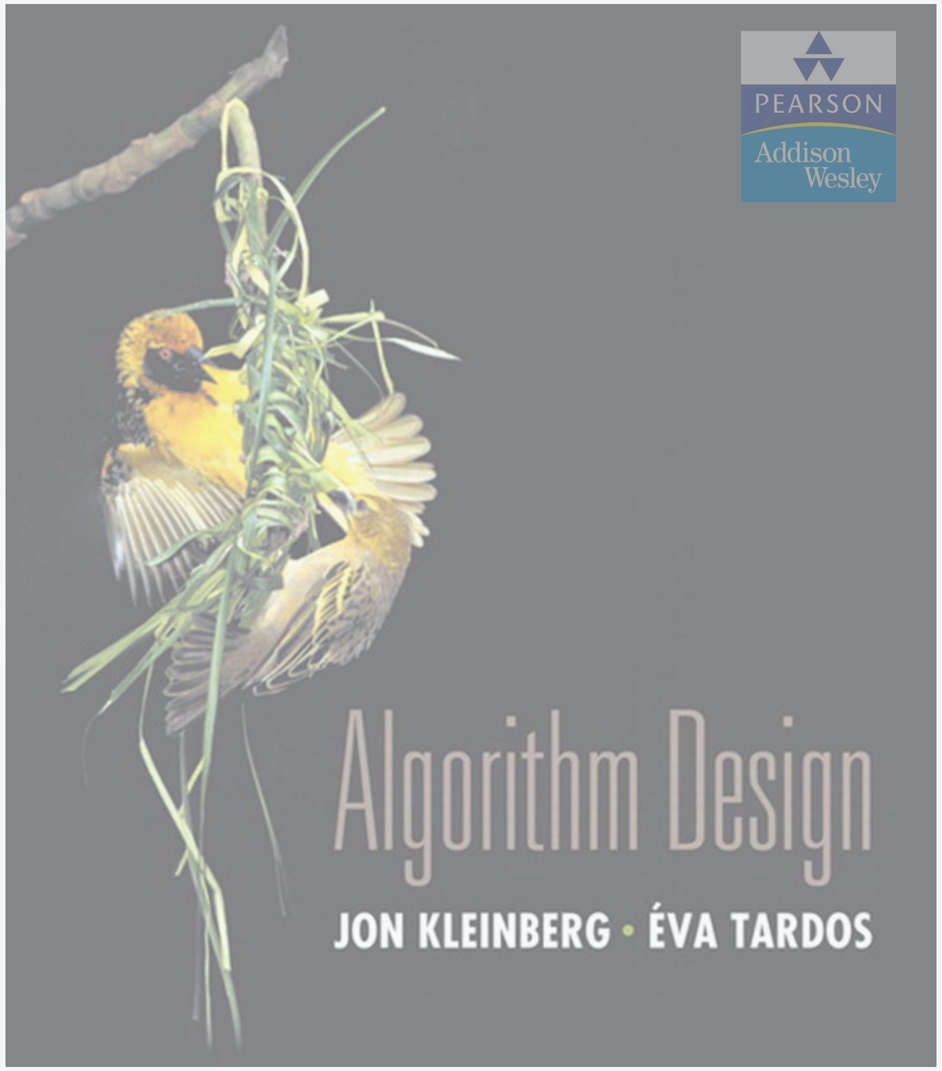


Kruskal's algorithm demo

Consider edges in ascending order of weight:

- Add to T unless it would create a cycle.





SECTION 4.5

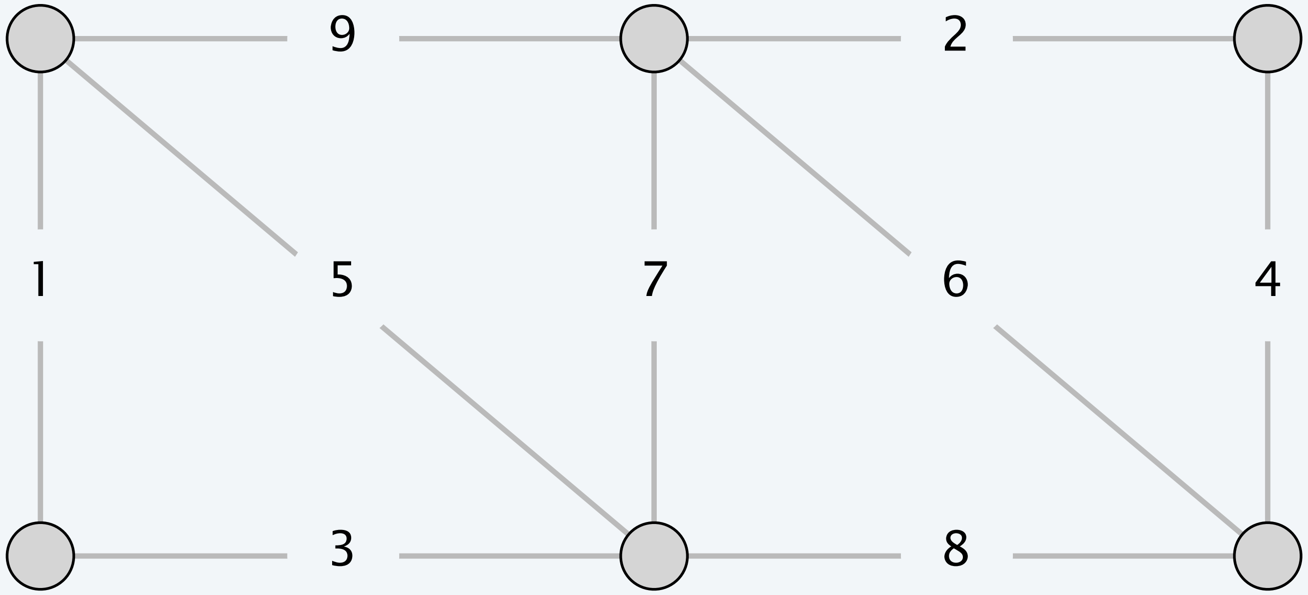
4. GREEDY ALGORITHMS II

- *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:

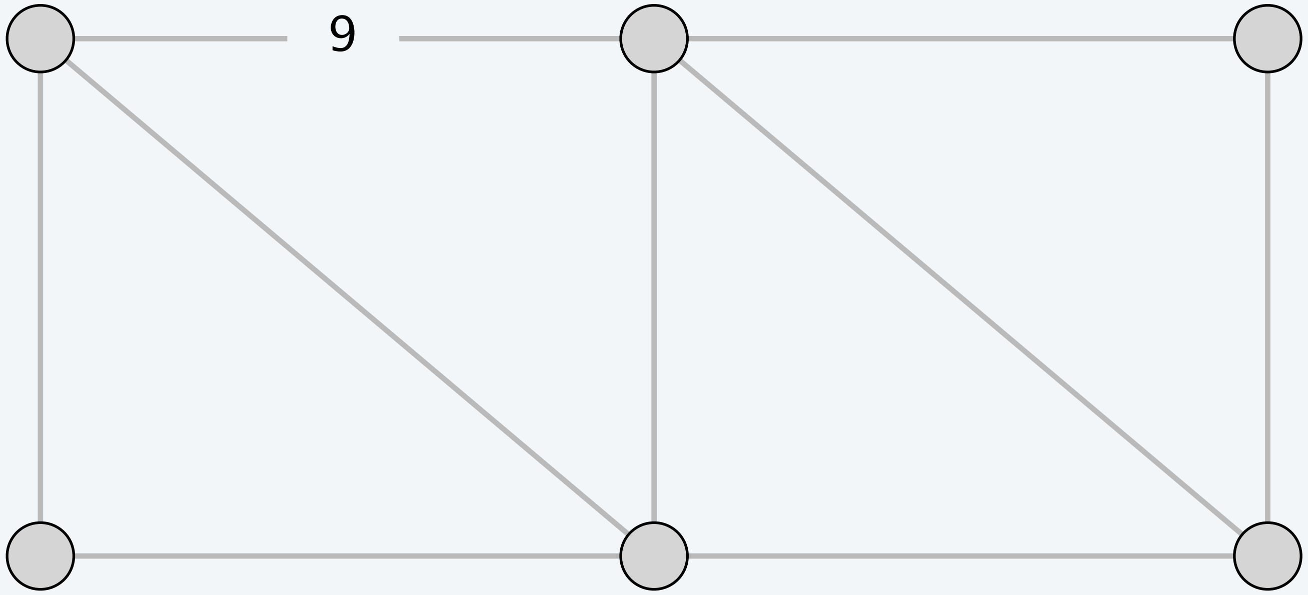
- Delete edge from T unless it would disconnect T.



Reverse-delete algorithm

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

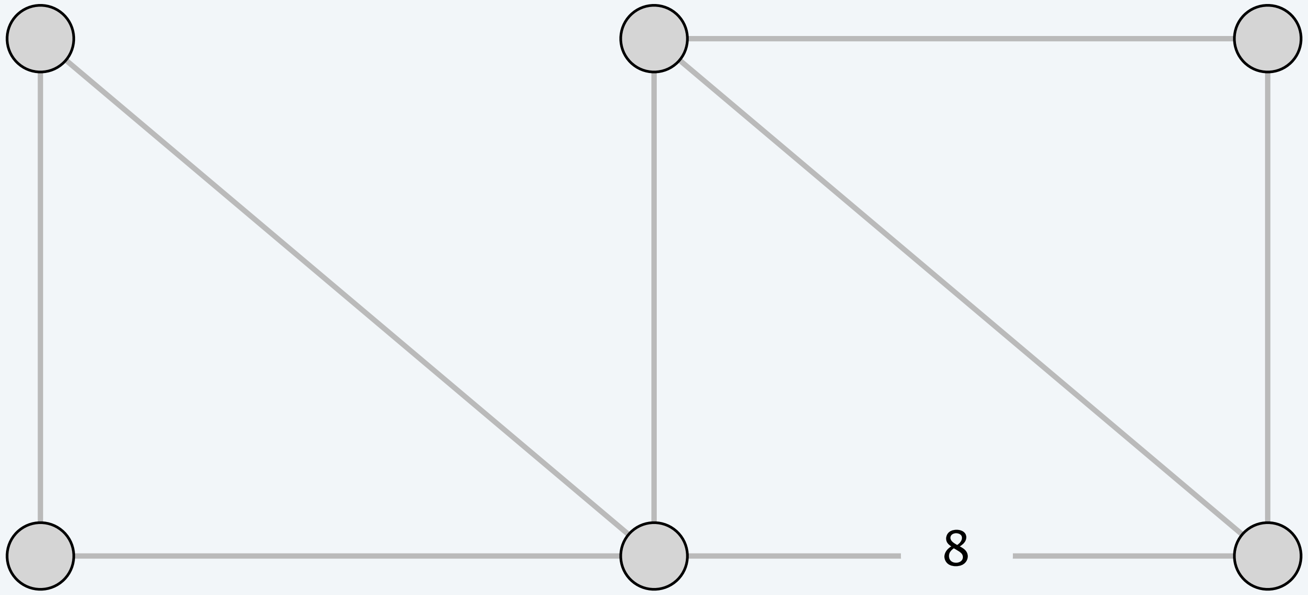
- Delete edge from T unless it would disconnect T.



Reverse-delete algorithm

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

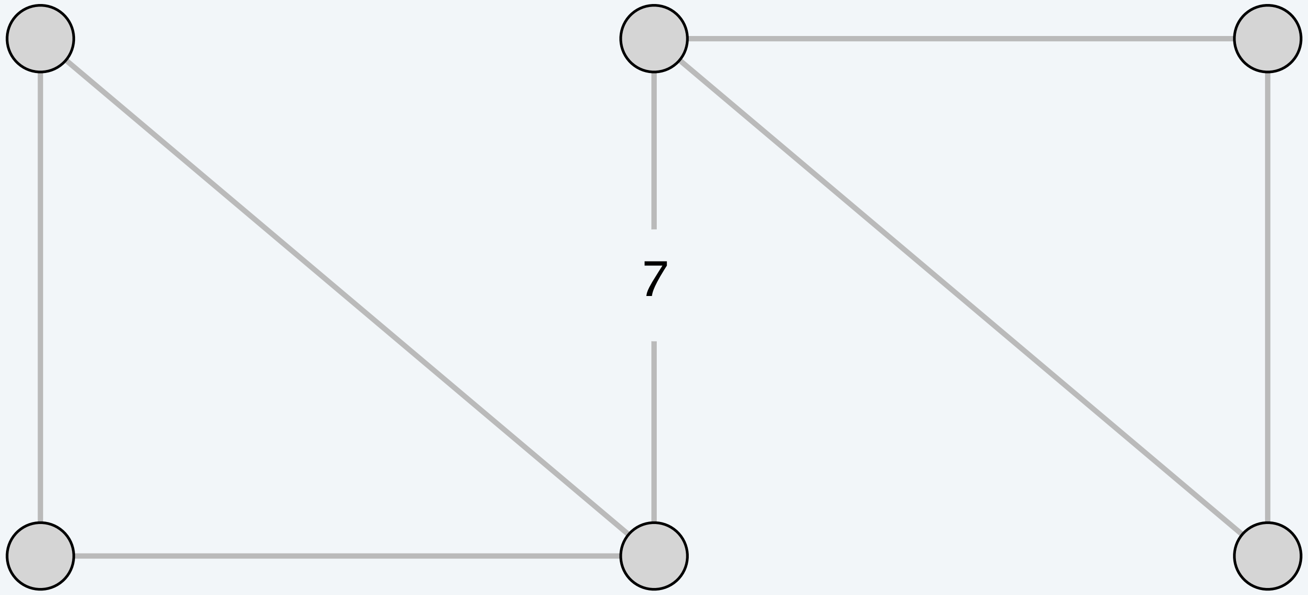
- Delete edge from T unless it would disconnect T.



Reverse-delete algorithm

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

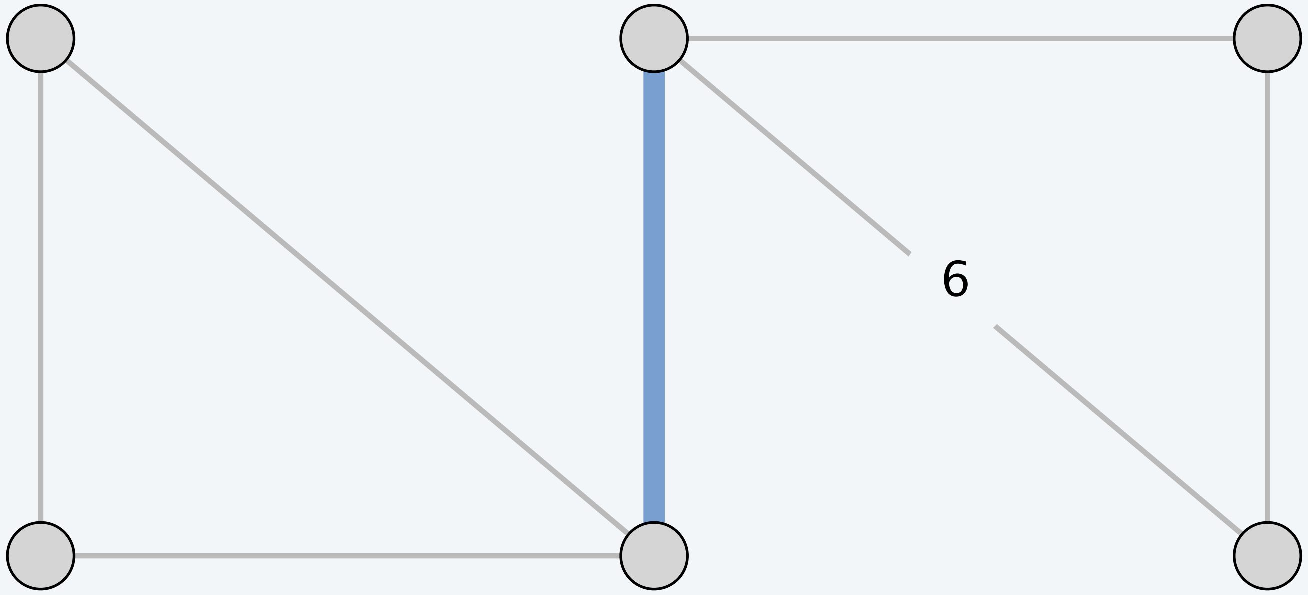
- Delete edge from T unless it would disconnect T.



Reverse-delete algorithm

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

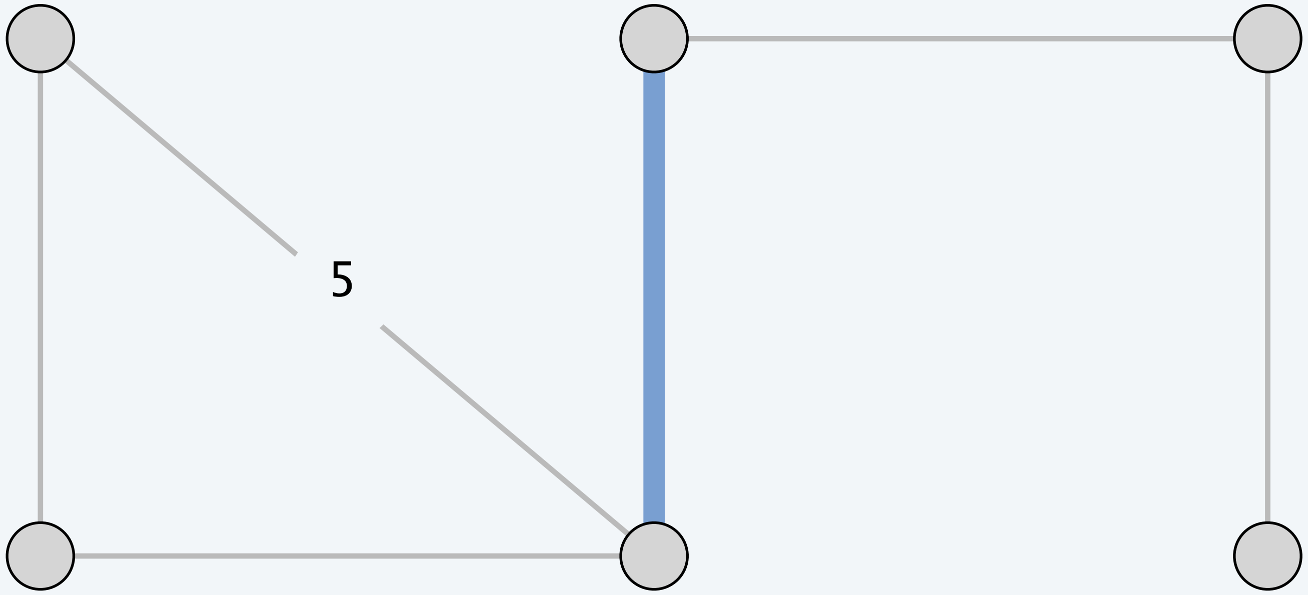
- Delete edge from T unless it would disconnect T.



Reverse-delete algorithm

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

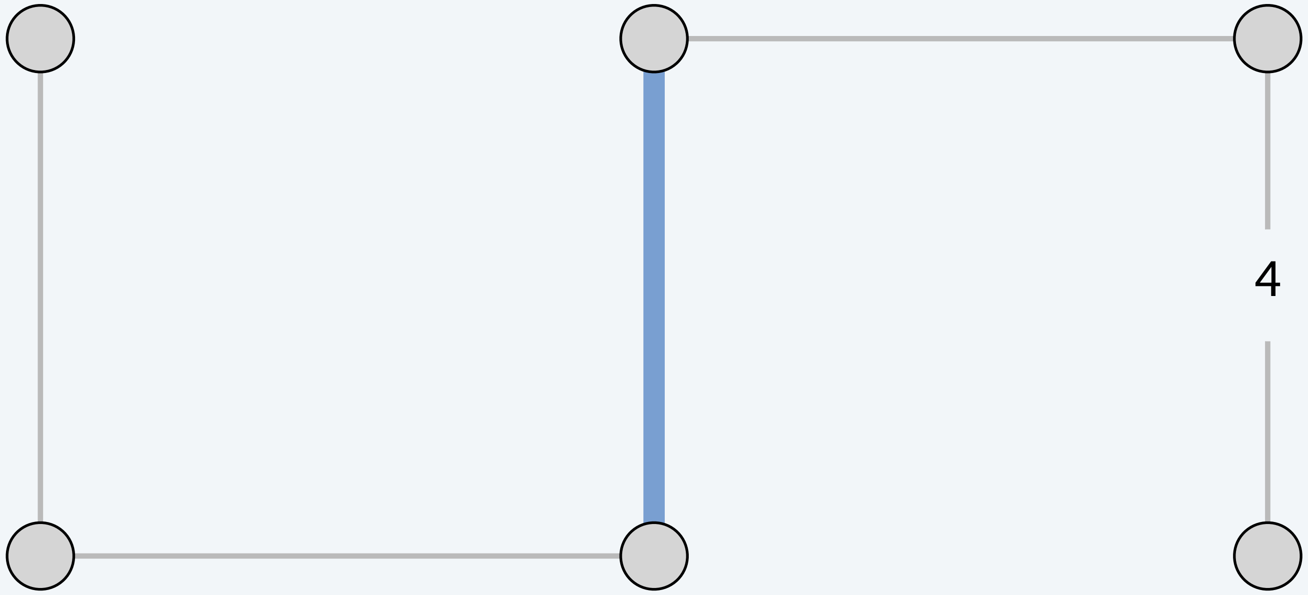
- Delete edge from T unless it would disconnect T.



Reverse-delete algorithm

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

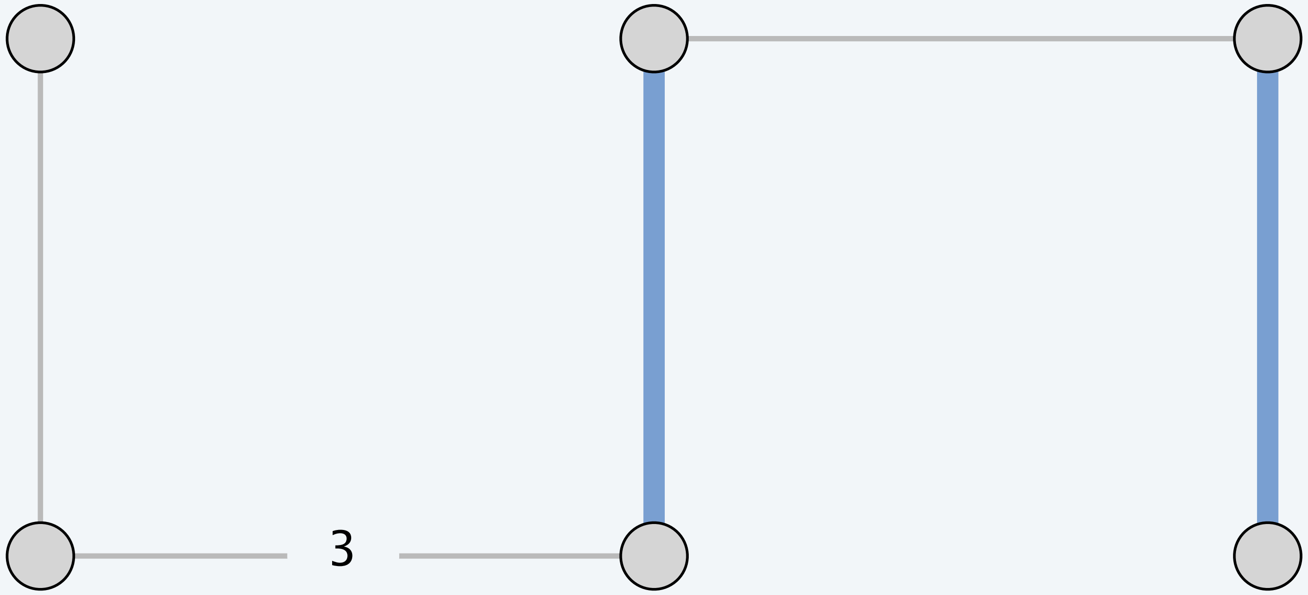
- Delete edge from T unless it would disconnect T.



Reverse-delete algorithm

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

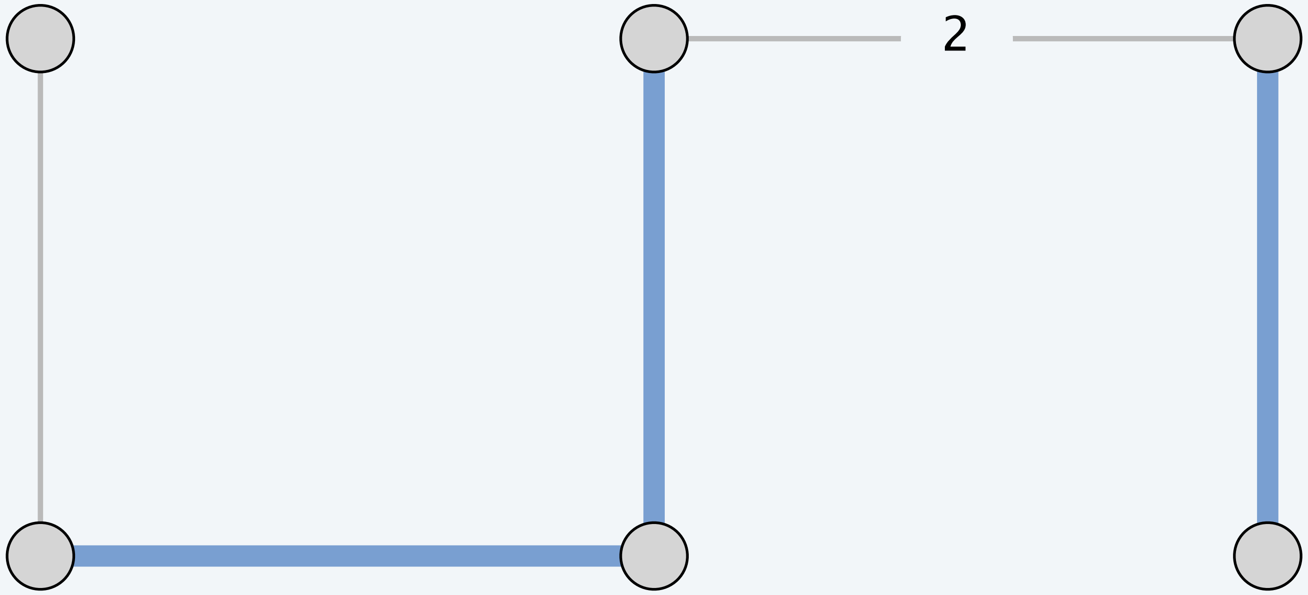
- Delete edge from T unless it would disconnect T.



Reverse-delete algorithm

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

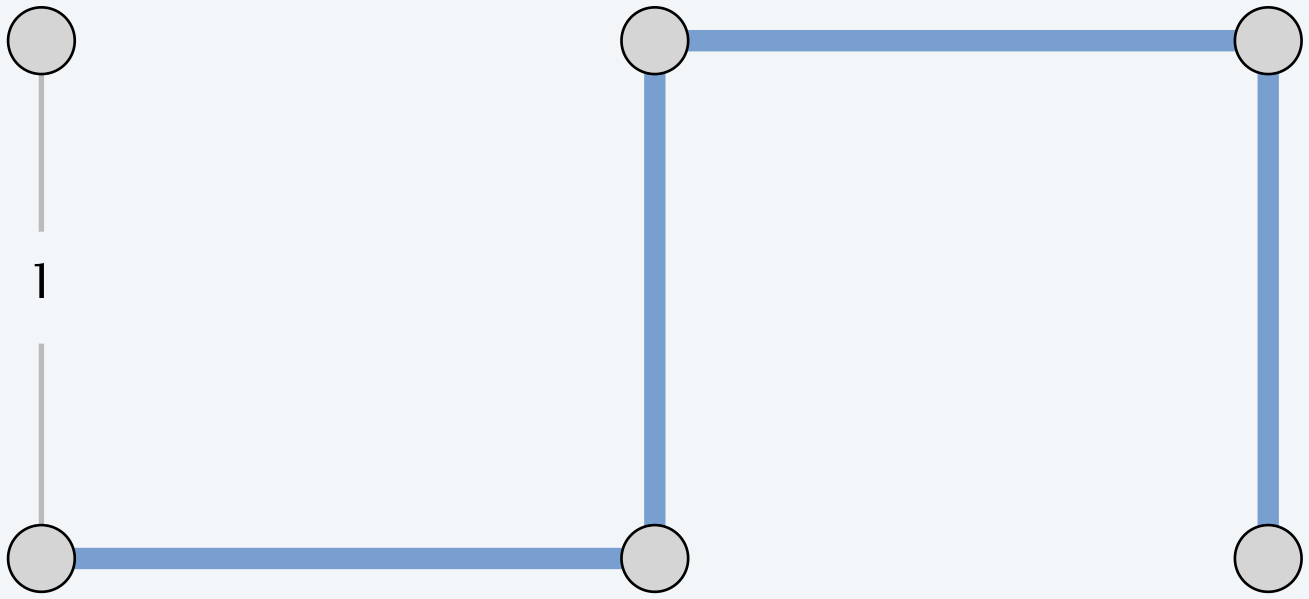
- Delete edge from T unless it would disconnect T.



Reverse-delete algorithm

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

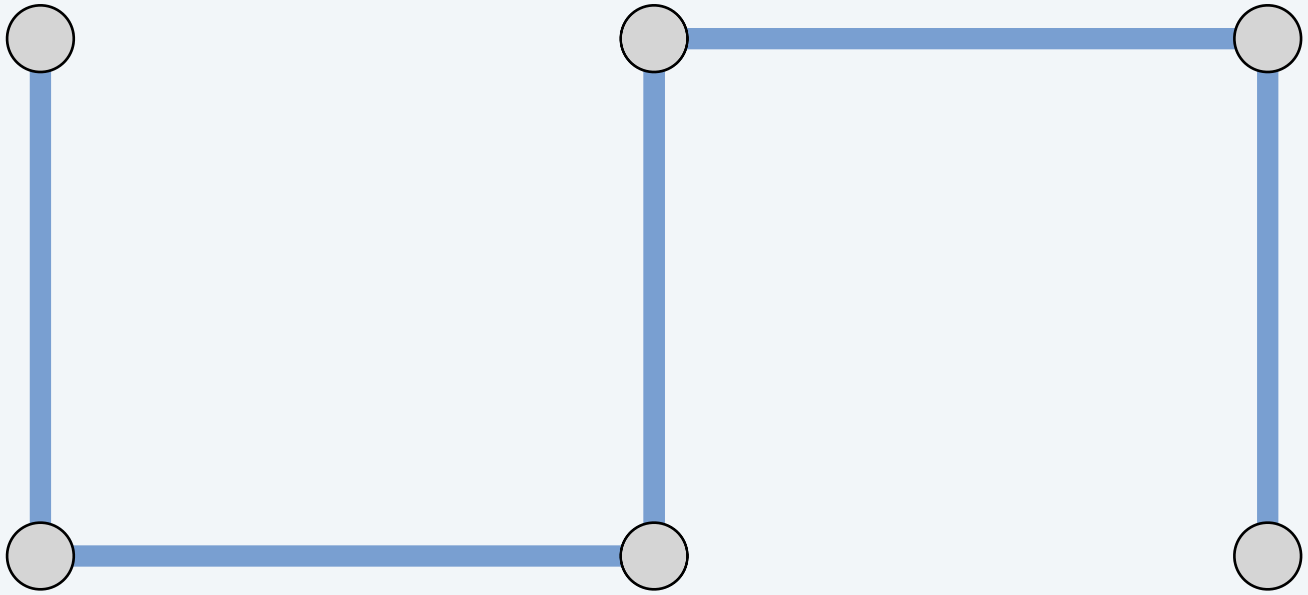
- Delete edge from T unless it would disconnect T.



Reverse-delete algorithm

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

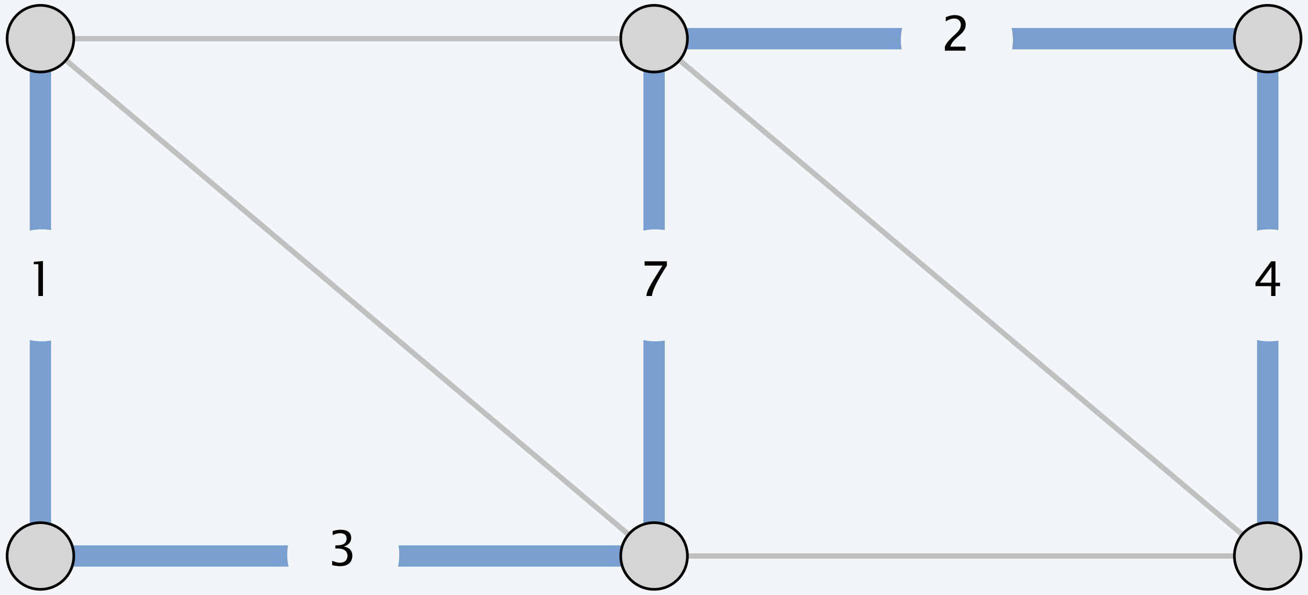
- Delete edge from T unless it would disconnect T.

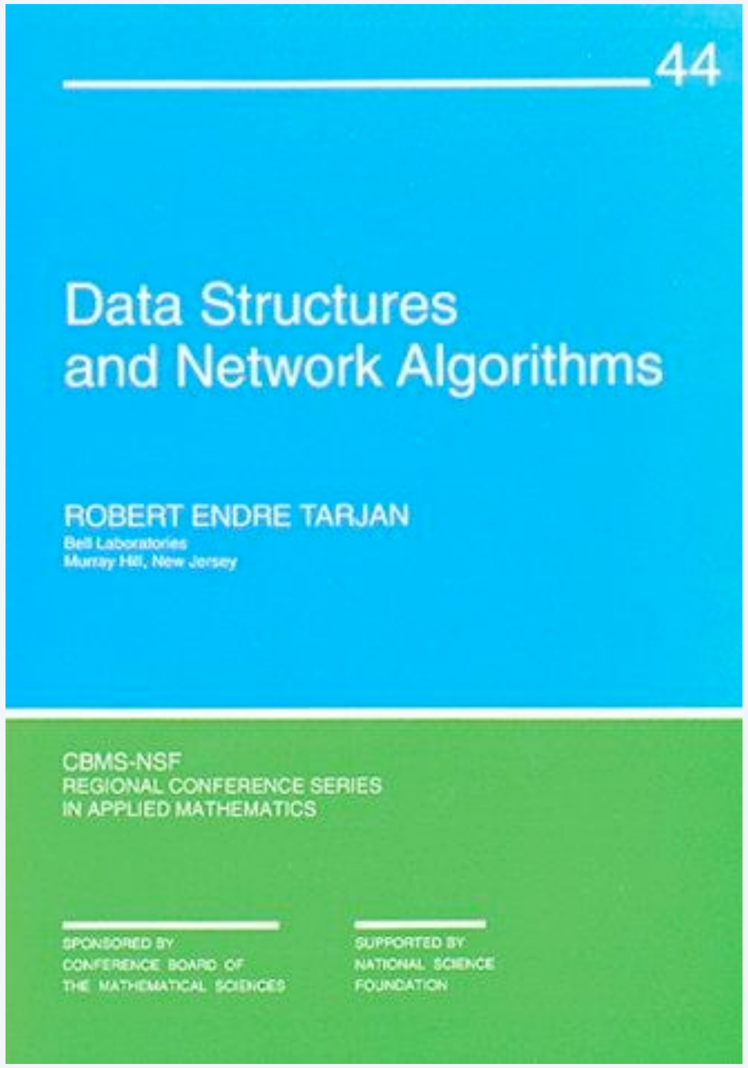


Reverse-delete algorithm

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

- Delete edge from T unless it would disconnect T.





44

Data Structures
and Network Algorithms

ROBERT ENDRE TARJAN
Bell Laboratories
Murray Hill, New Jersey

CBMS-NSF
REGIONAL CONFERENCE SERIES
IN APPLIED MATHEMATICS

SPONSORED BY
CONFERENCE BOARD OF
THE MATHEMATICAL SCIENCES

SUPPORTED BY
NATIONAL SCIENCE
FOUNDATION

SECTION 6.2

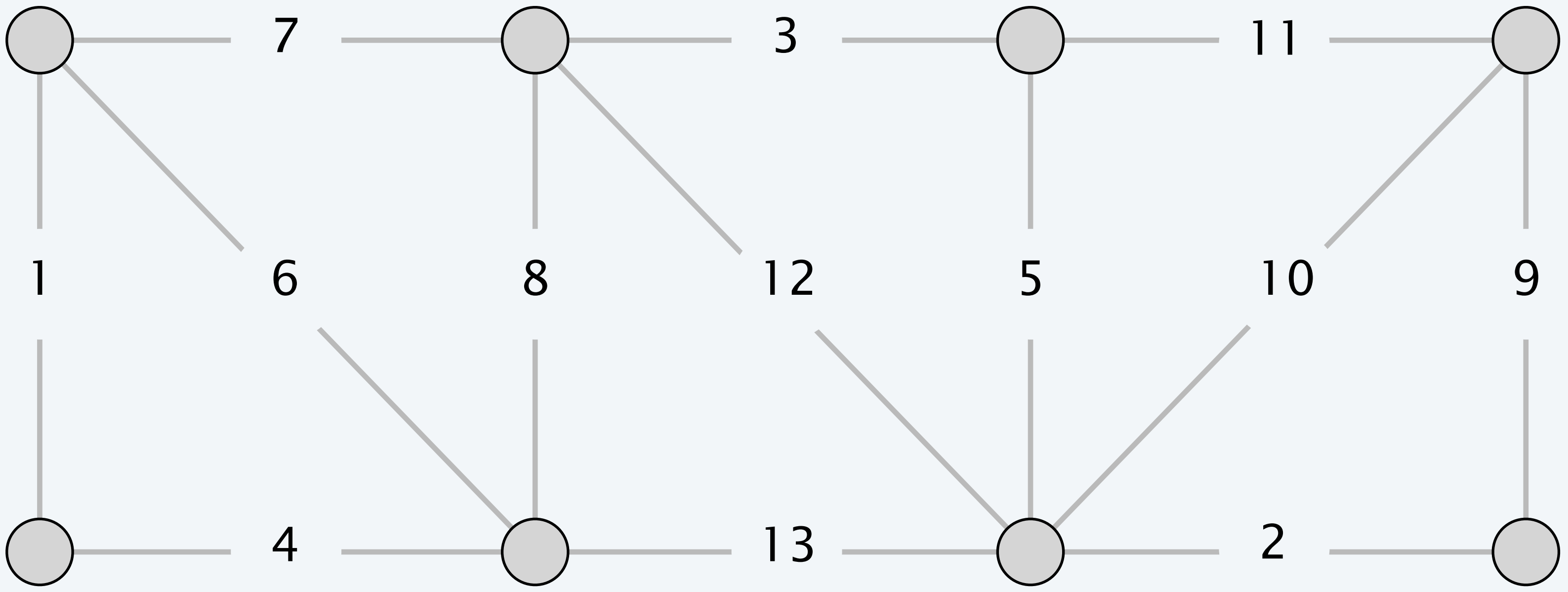
4. GREEDY ALGORITHMS II

- *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.

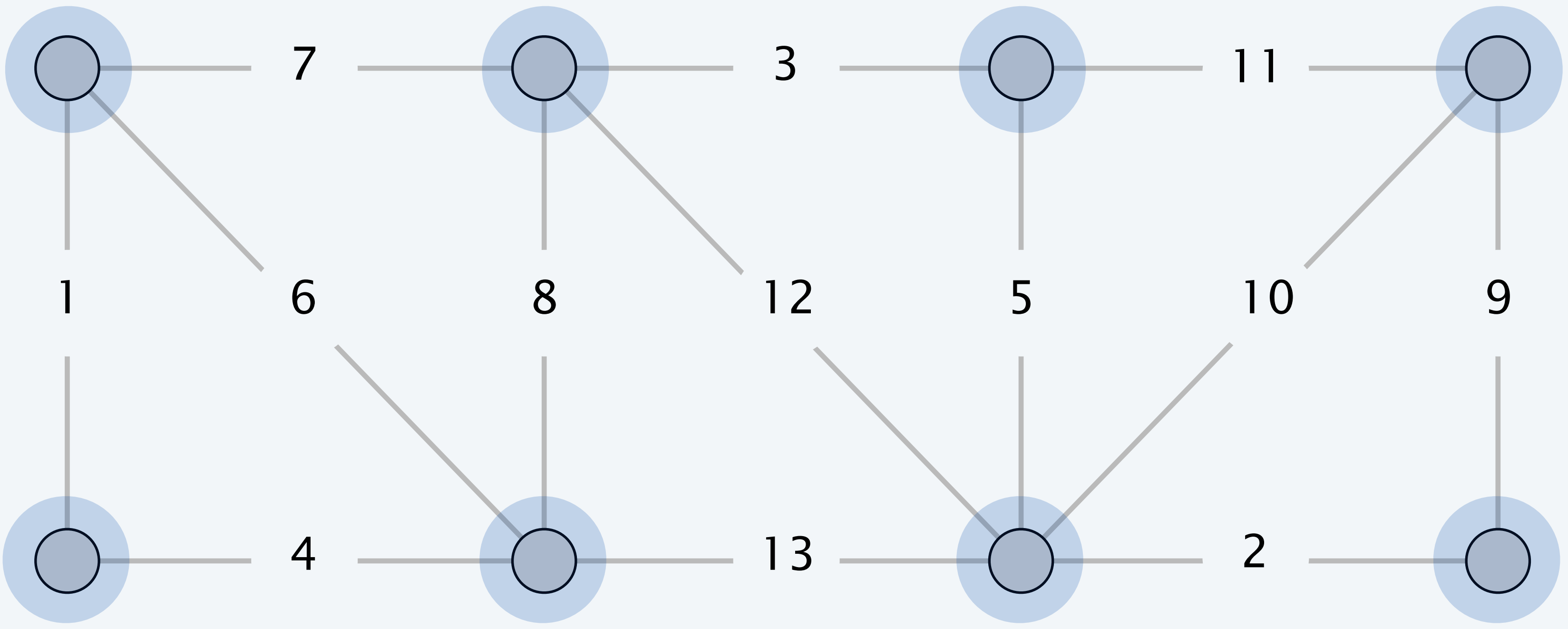
- Apply blue rule to cutset corresponding to each blue tree.
- Color all selected edges blue.



Borůvka's algorithm demo

Repeat until only one tree.

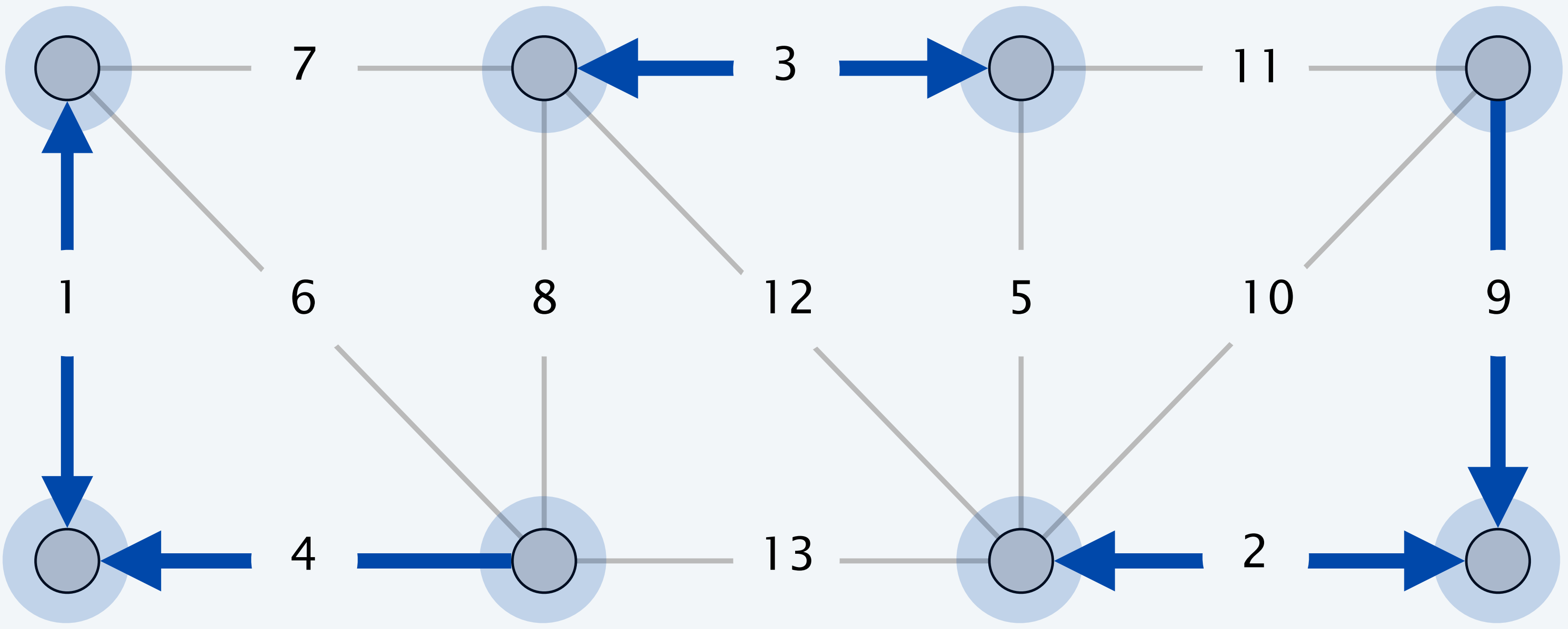
- Apply blue rule to cutset corresponding to each blue tree.
- Color all selected edges blue.



Borůvka's algorithm demo

Repeat until only one tree.

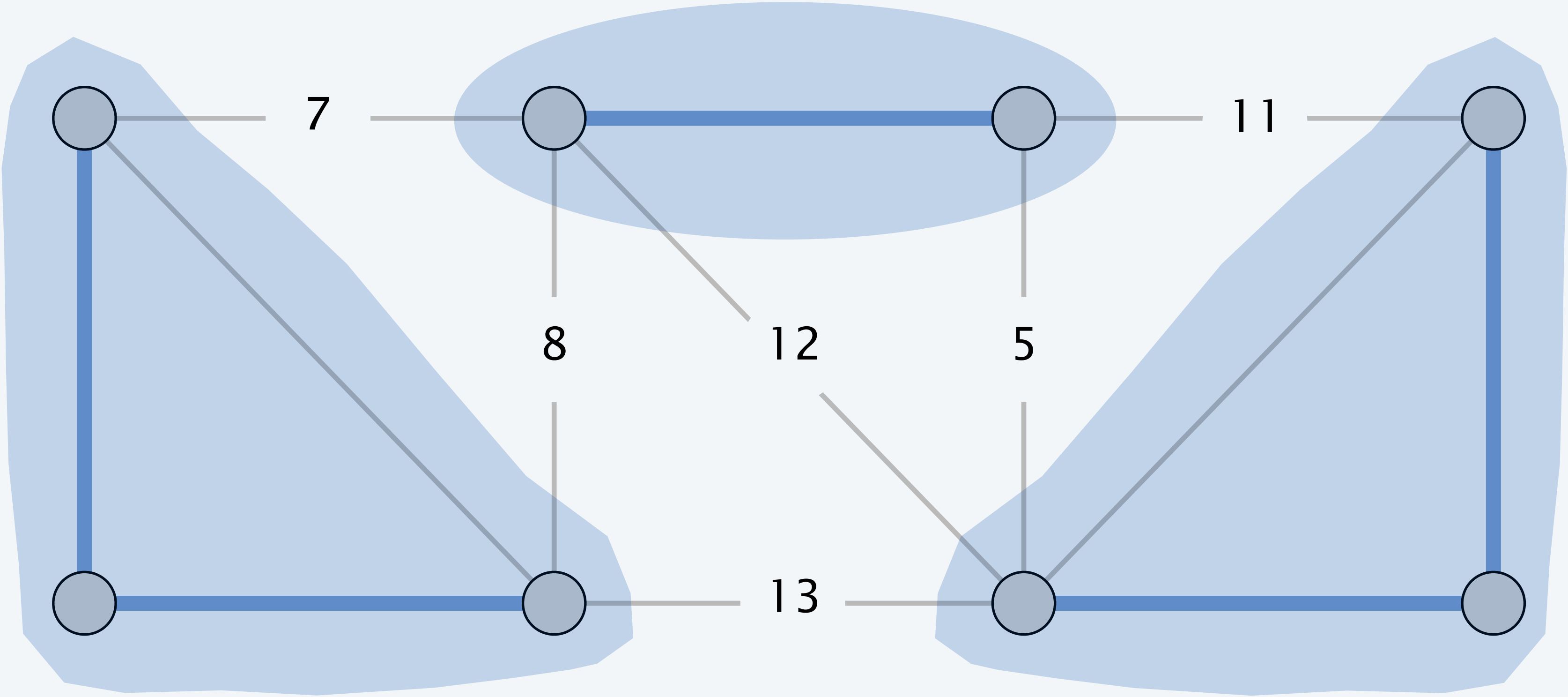
- Apply blue rule to cutset corresponding to each blue tree.
- Color all selected edges blue.



Borůvka's algorithm demo

Repeat until only one tree.

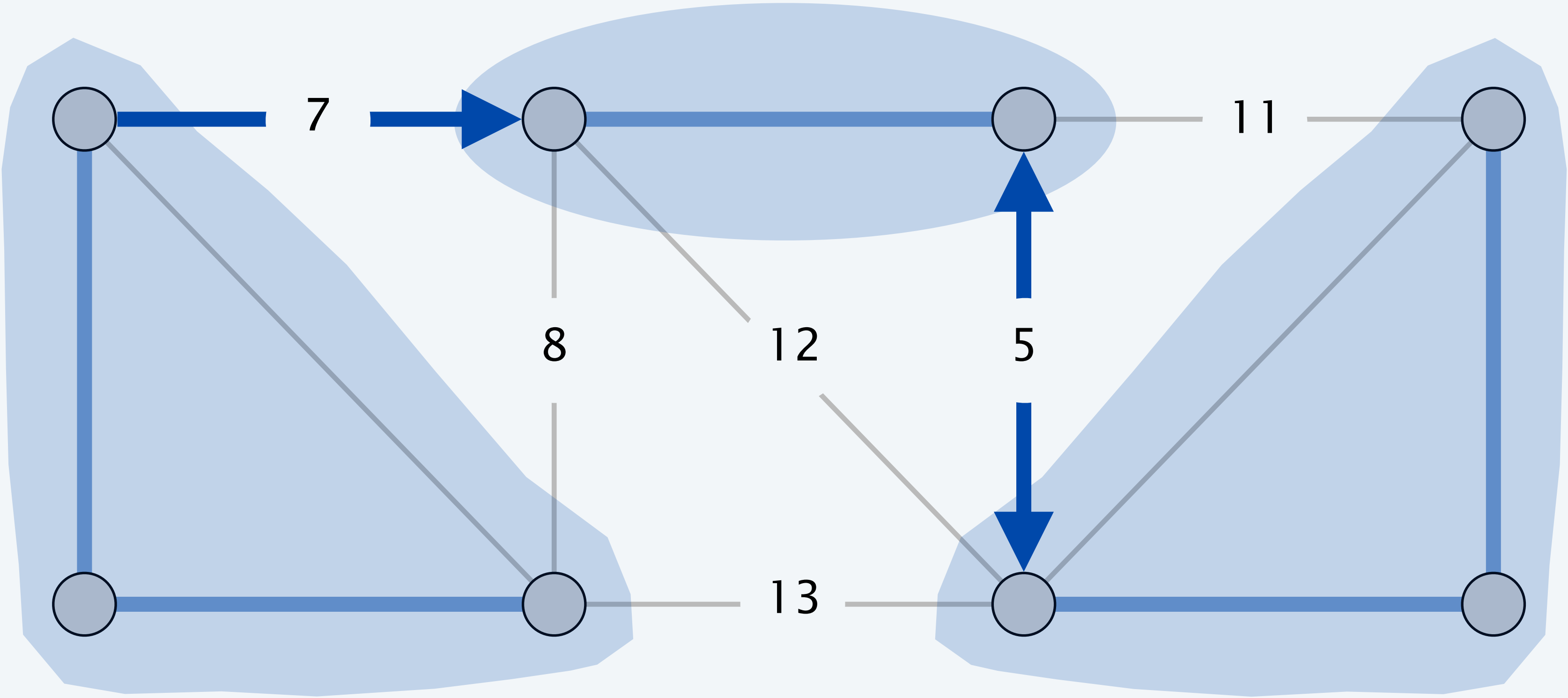
- Apply blue rule to cutset corresponding to each blue tree.
- Color all selected edges blue.



Borůvka's algorithm demo

Repeat until only one tree.

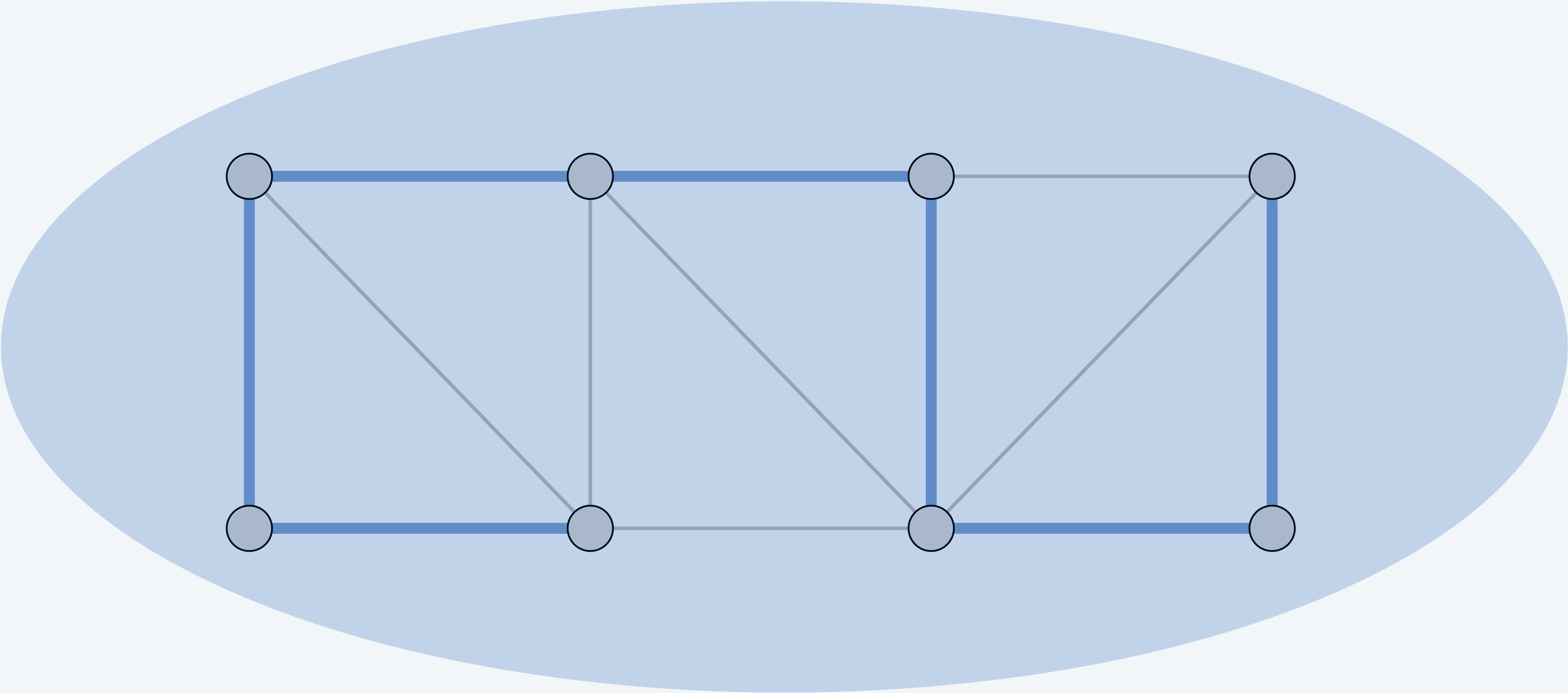
- Apply blue rule to cutset corresponding to each blue tree.
- Color all selected edges blue.



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.