

## 4. GREEDY ALGORITHMS II

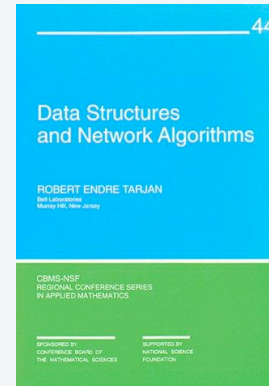
- ▶ *red-rule blue-rule demo*
- ▶ *Prim's algorithm demo*
- ▶ *Kruskal's algorithm demo*
- ▶ *reverse-delete algorithm demo*
- ▶ *Boruvka's algorithm demo*

Lecture slides by Kevin Wayne

Copyright © 2005 Pearson-Addison Wesley

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

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



SECTION 6.1

## 4. GREEDY ALGORITHMS II

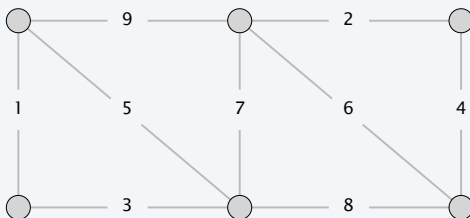
- ▶ *red-rule blue-rule demo*
- ▶ *Prim's algorithm demo*
- ▶ *Kruskal's algorithm demo*
- ▶ *reverse-delete algorithm demo*
- ▶ *Boruvka's algorithm demo*

### 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

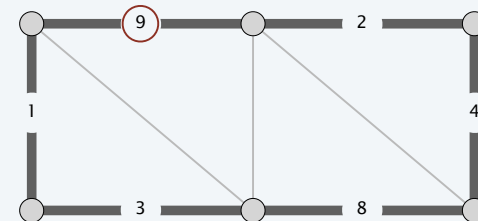


3

### 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.

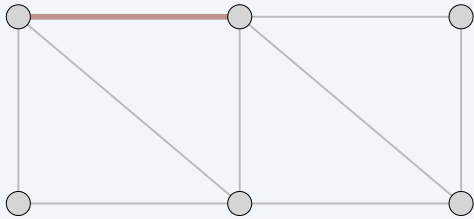
apply the red rule to the cycle



4

## Red-rule blue-rule demo

current set of red and blue edges

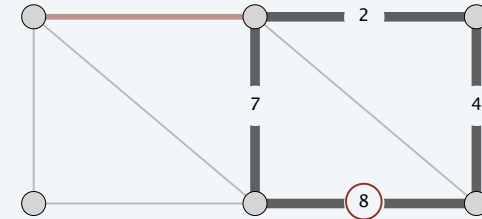


5

## 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

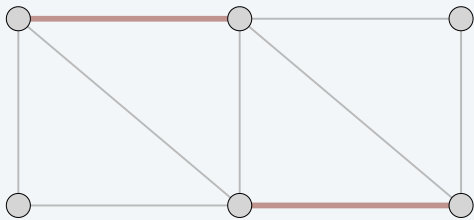


6

## 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

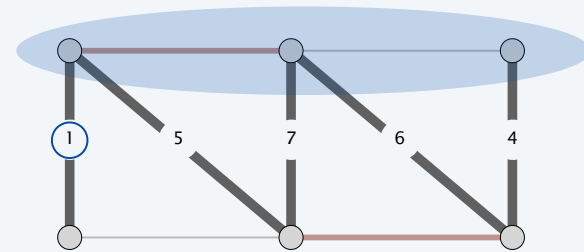


7

## 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

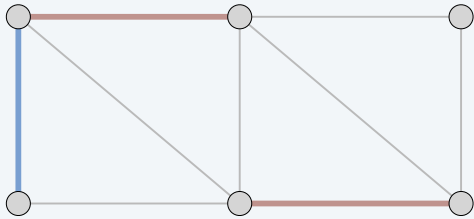


8

## Red-rule blue-rule demo

---

current set of red and blue edges



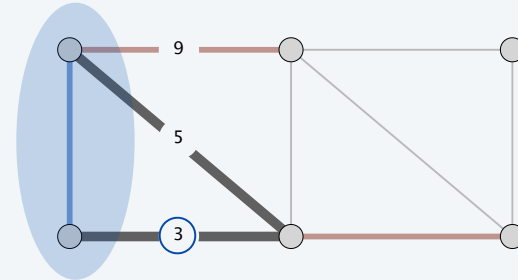
9

## 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

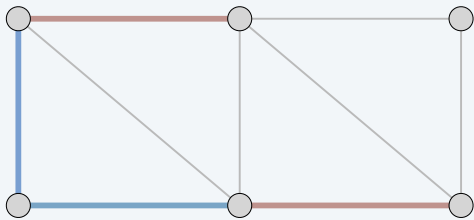


10

## Red-rule blue-rule demo

---

current set of red and blue edges



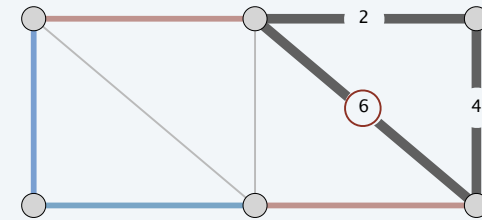
11

## 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

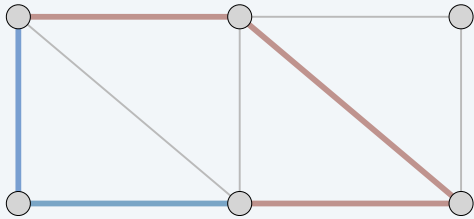


12

## Red-rule blue-rule demo

---

current set of red and blue edges



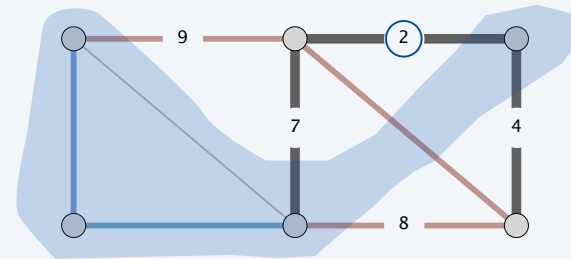
13

## 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

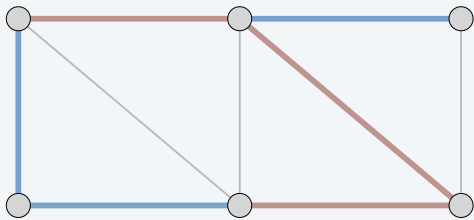


14

## Red-rule blue-rule demo

---

current set of red and blue edges



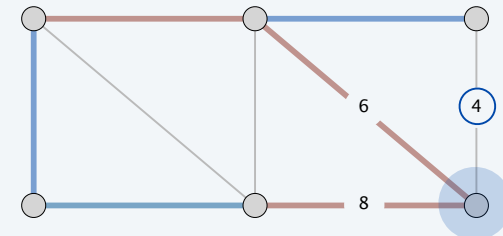
15

## 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

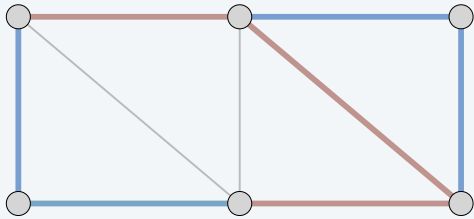


16

## Red-rule blue-rule demo

---

current set of red and blue edges



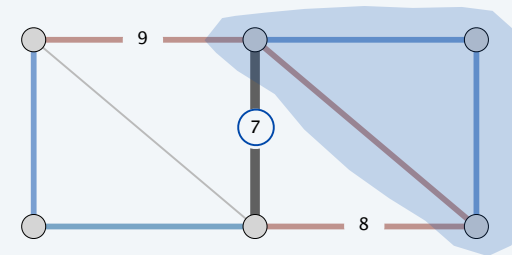
17

## 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

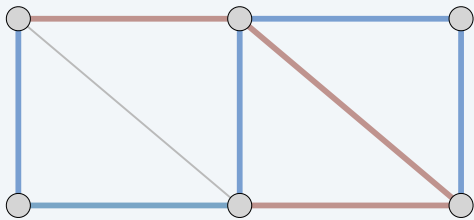


18

## Red-rule blue-rule demo

---

current set of red and blue edges



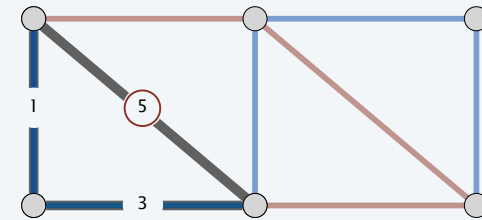
19

## 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.

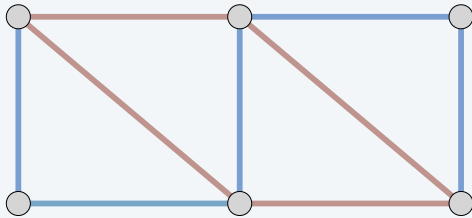
apply the red rule to the cycle



20

## Red-rule blue-rule demo

current set of red and blue edges

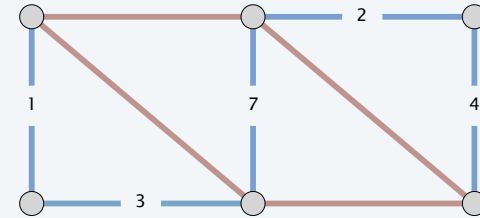


21

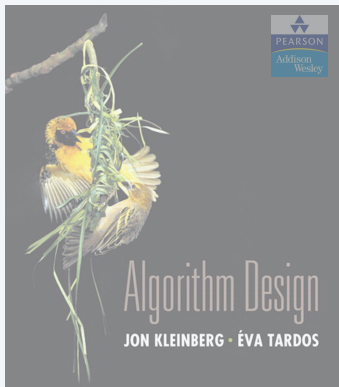
## Red-rule blue-rule demo

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

a minimum spanning tree



22



SECTION 4.5

## 4. GREEDY ALGORITHMS II

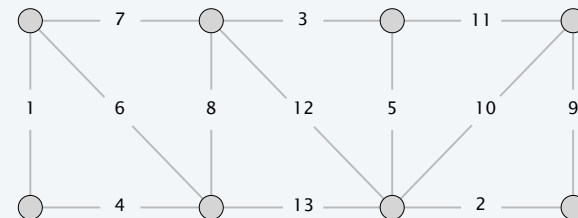
- ▶ *red-rule blue-rule demo*
- ▶ *Prim's algorithm demo*
- ▶ *Kruskal's algorithm demo*
- ▶ *reverse-delete algorithm demo*
- ▶ *Boruvka's algorithm demo*

## Prim's algorithm demo

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

Repeat  $n - 1$  times:

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



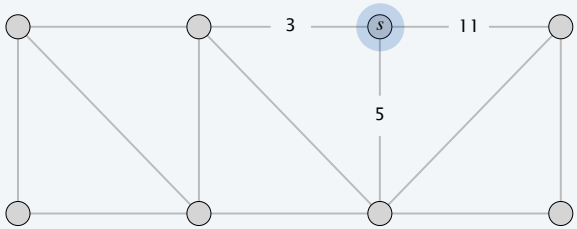
24

## 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  $S$ .



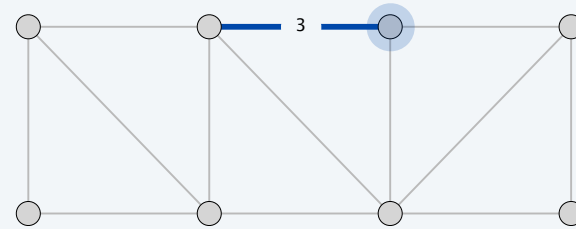
25

## 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  $S$ .



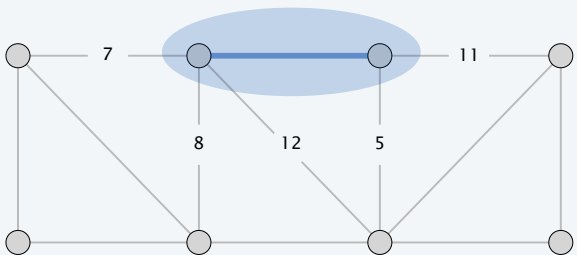
26

## 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  $S$ .



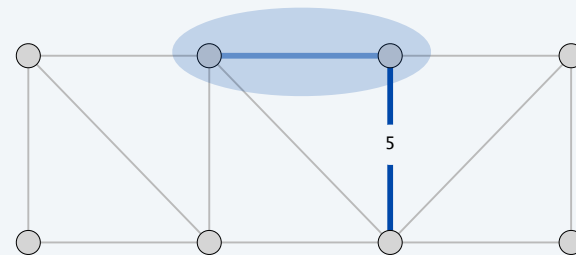
27

## 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  $S$ .



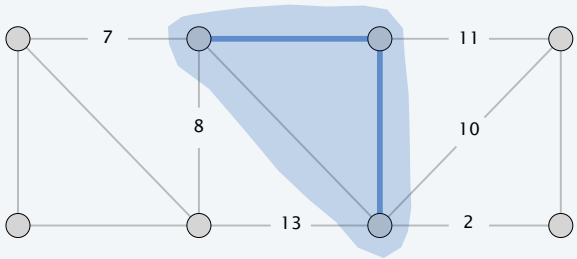
28

## 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  $S$ .



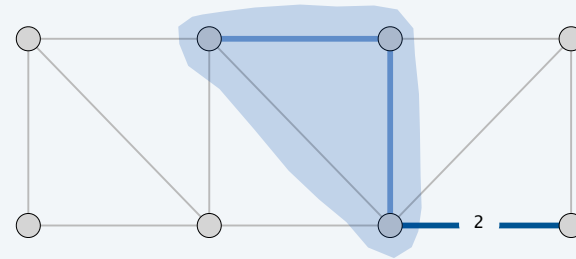
29

## 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  $S$ .



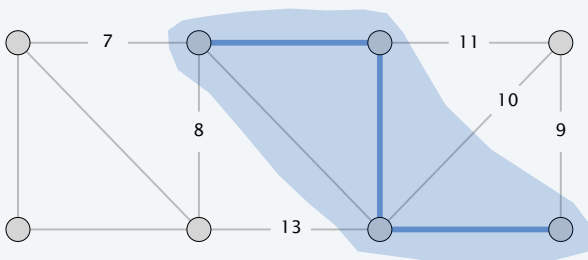
30

## 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  $S$ .



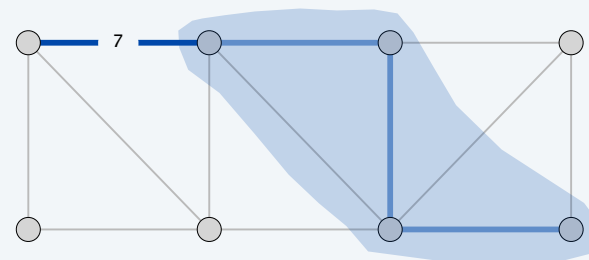
31

## 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  $S$ .



32

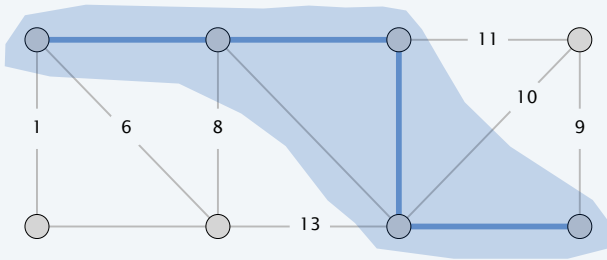


## Prim's algorithm demo

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  $S$ .



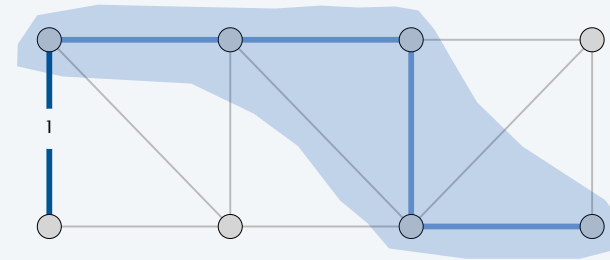
33

## 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  $S$ .



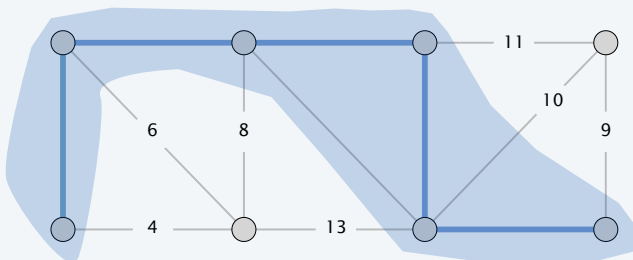
34

## 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  $S$ .



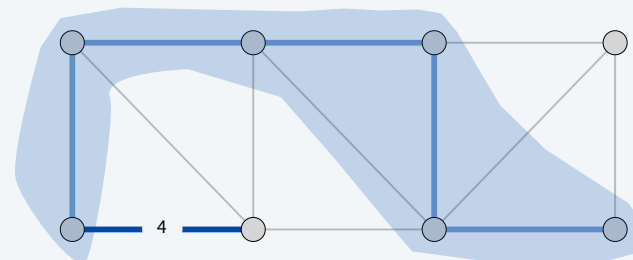
35

## 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  $S$ .



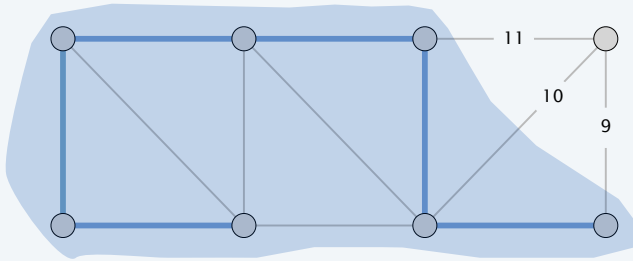
36

## 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  $S$ .



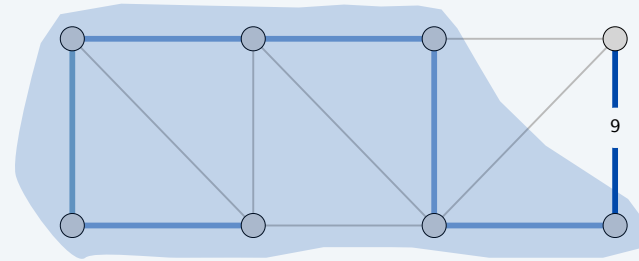
37

## 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  $S$ .



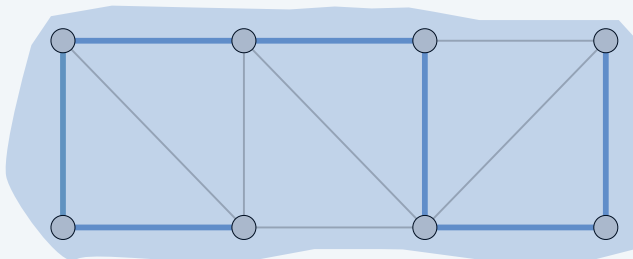
38

## 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  $S$ .



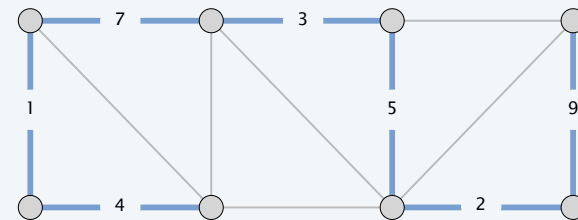
39

## Prim's algorithm demo

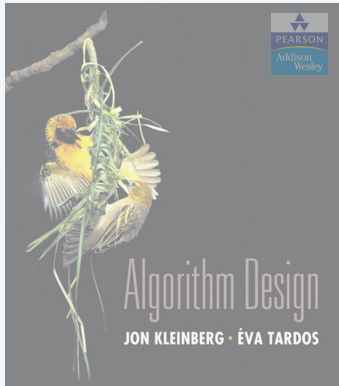
Initialize  $S = \{s\}$  for any node  $s$ ,  $T = \emptyset$ .

Repeat  $n - 1$  times:

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



40



SECTION 4.5

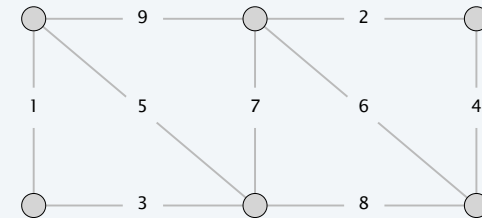
## 4. GREEDY ALGORITHMS II

- ▶ *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.

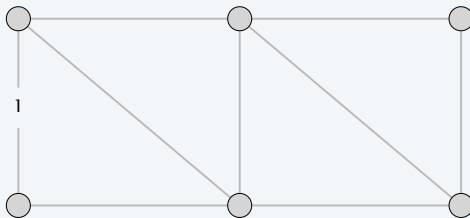


42

### Kruskal's algorithm demo

Consider edges in ascending order of weight:

- Add to  $T$  unless it would create a cycle.

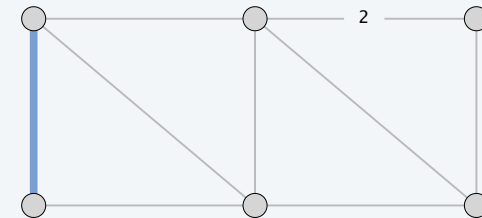


43

### Kruskal's algorithm demo

Consider edges in ascending order of weight:

- Add to  $T$  unless it would create a cycle.



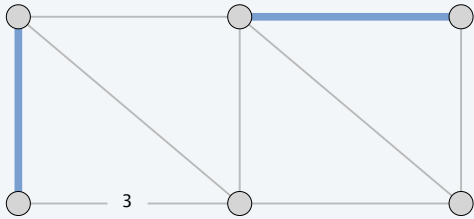
44

## Kruskal's algorithm demo

---

Consider edges in ascending order of weight:

- Add to  $T$  unless it would create a cycle.



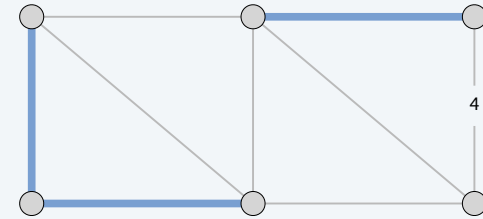
45

## Kruskal's algorithm demo

---

Consider edges in ascending order of weight:

- Add to  $T$  unless it would create a cycle.



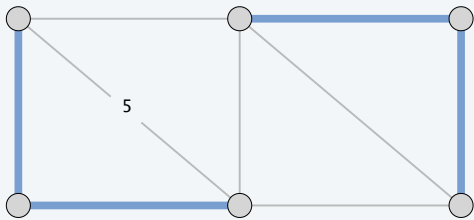
46

## Kruskal's algorithm demo

---

Consider edges in ascending order of weight:

- Add to  $T$  unless it would create a cycle.



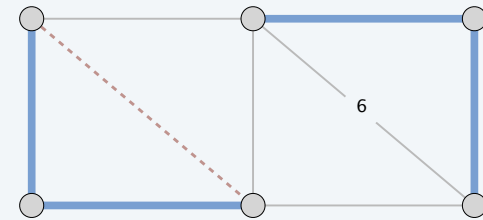
47

## Kruskal's algorithm demo

---

Consider edges in ascending order of weight:

- Add to  $T$  unless it would create a cycle.



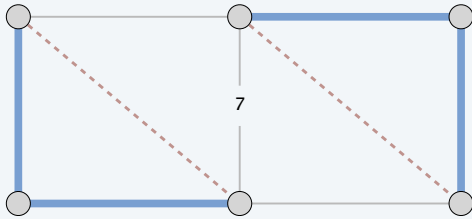
48

## Kruskal's algorithm demo

---

Consider edges in ascending order of weight:

- Add to  $T$  unless it would create a cycle.



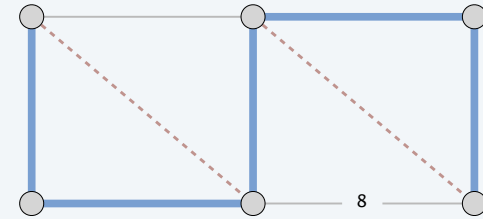
49

## Kruskal's algorithm demo

---

Consider edges in ascending order of weight:

- Add to  $T$  unless it would create a cycle.



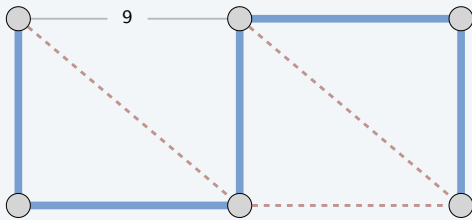
50

## Kruskal's algorithm demo

---

Consider edges in ascending order of weight:

- Add to  $T$  unless it would create a cycle.



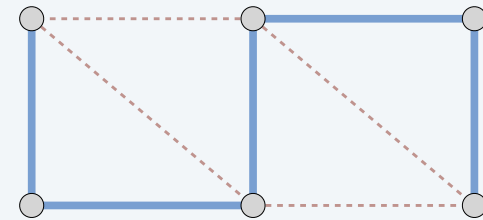
51

## Kruskal's algorithm demo

---

Consider edges in ascending order of weight:

- Add to  $T$  unless it would create a cycle.

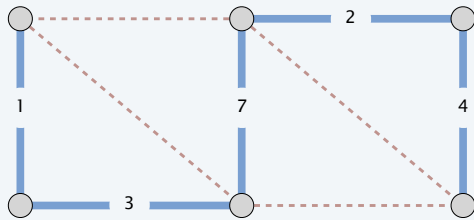


52

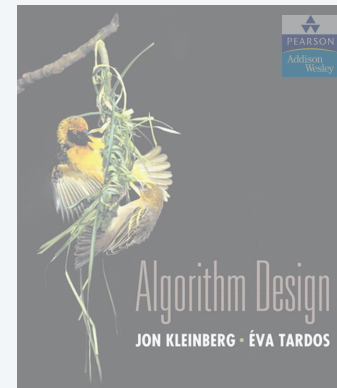
## Kruskal's algorithm demo

Consider edges in ascending order of weight:

- Add to  $T$  unless it would create a cycle.



53



SECTION 4.5

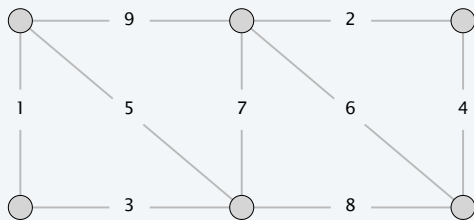
## 4. GREEDY ALGORITHMS II

- ▶ *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  $T$ .

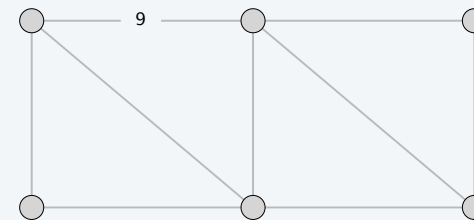


55

## 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  $T$ .



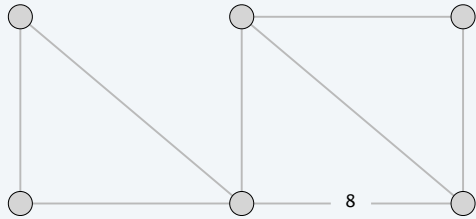
56

## 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  $T$ .



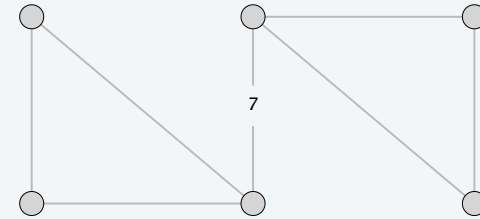
57

## 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  $T$ .



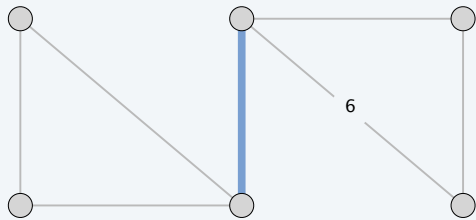
58

## 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  $T$ .



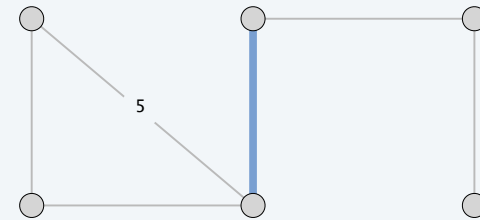
59

## 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  $T$ .



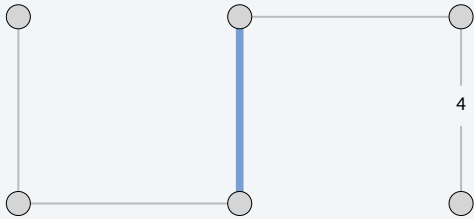
60

## 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  $T$ .



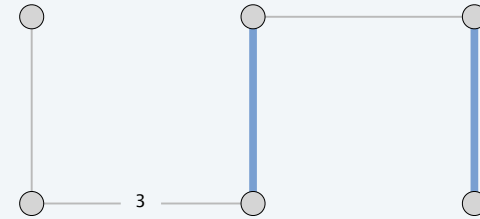
61

## 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  $T$ .



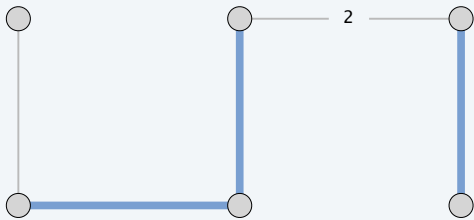
62

## 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  $T$ .



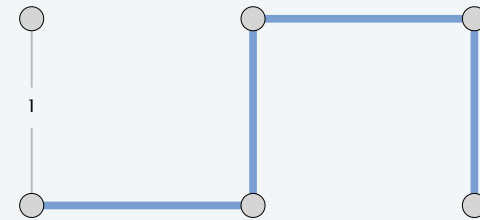
63

## 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  $T$ .



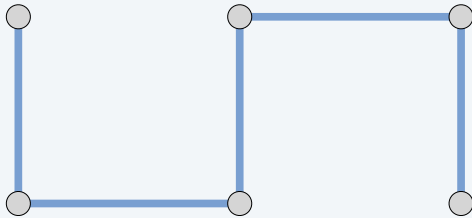
64



## 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  $T$ .

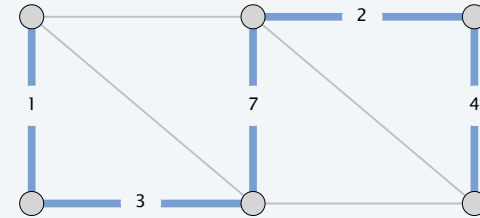


65

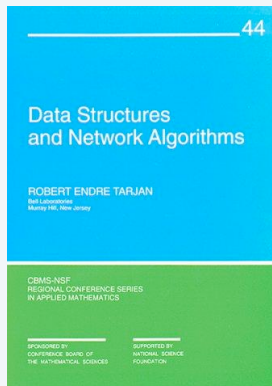
## Reverse-delete algorithm

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

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



66



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  
COMMISSION ON  
MATHEMATICAL SCIENCES

SPONSORED BY  
NATIONAL SCIENCE  
FOUNDATION

SECTION 6.2

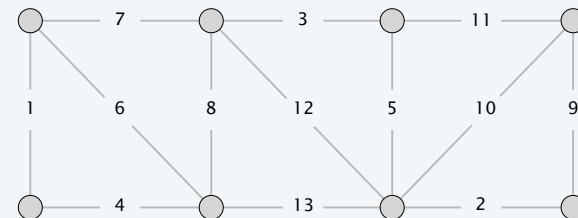
## 4. GREEDY ALGORITHMS II

- ▶ *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.

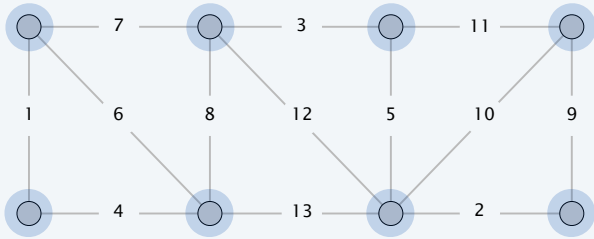


68

## Borůvka's algorithm demo

Repeat until only one tree.

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

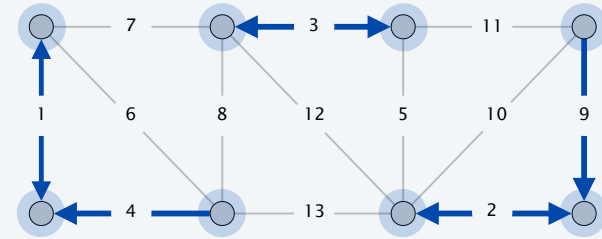


69

## Borůvka's algorithm demo

Repeat until only one tree.

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

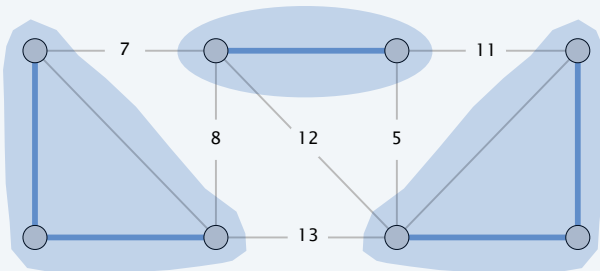


70

## Borůvka's algorithm demo

Repeat until only one tree.

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

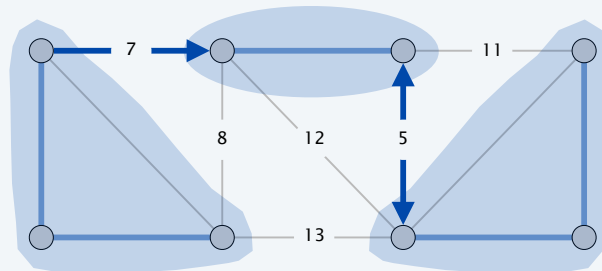


71

## Borůvka's algorithm demo

Repeat until only one tree.

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



72

