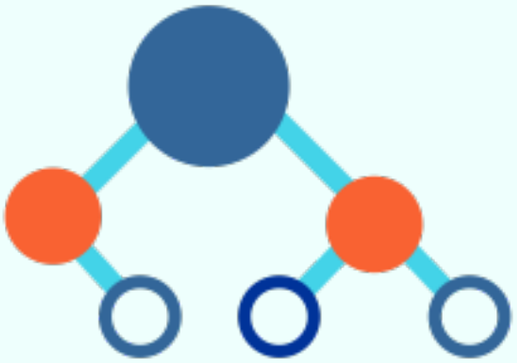
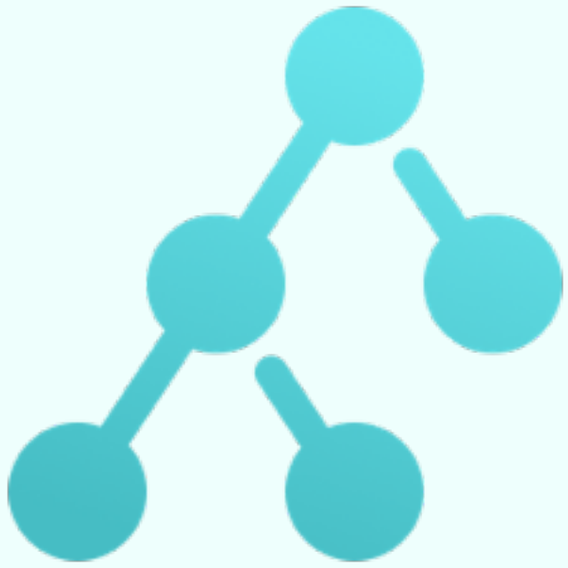
Data Structures

Exercise Session



Marco Vassena

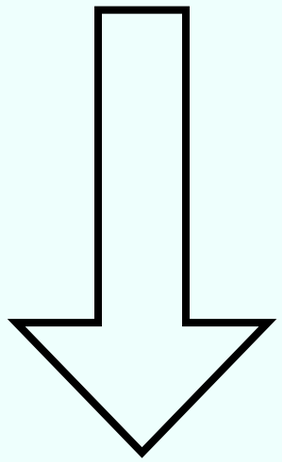


Exercise 9.20

Minimum Spanning Tree

Exercise 9.20

Minimum Spanning Tree



Maximum Spanning Tree

$$\text{spacing } (V_1, V_2)$$

$$=$$

$$\min \{\, w(v_1, v_2) \mid v_1 \in V_1, v_2 \in V_2, (v_1, v_2) \in E \,\}$$

$$\text{spacing}\,(V_1, V_2)$$

$$\min \{\, w(v_1, v_2) \mid v_1 \in V_1,\ v_2 \in V_2,\ (v_1, v_2) \in E \,\}$$

Weight

Vertex

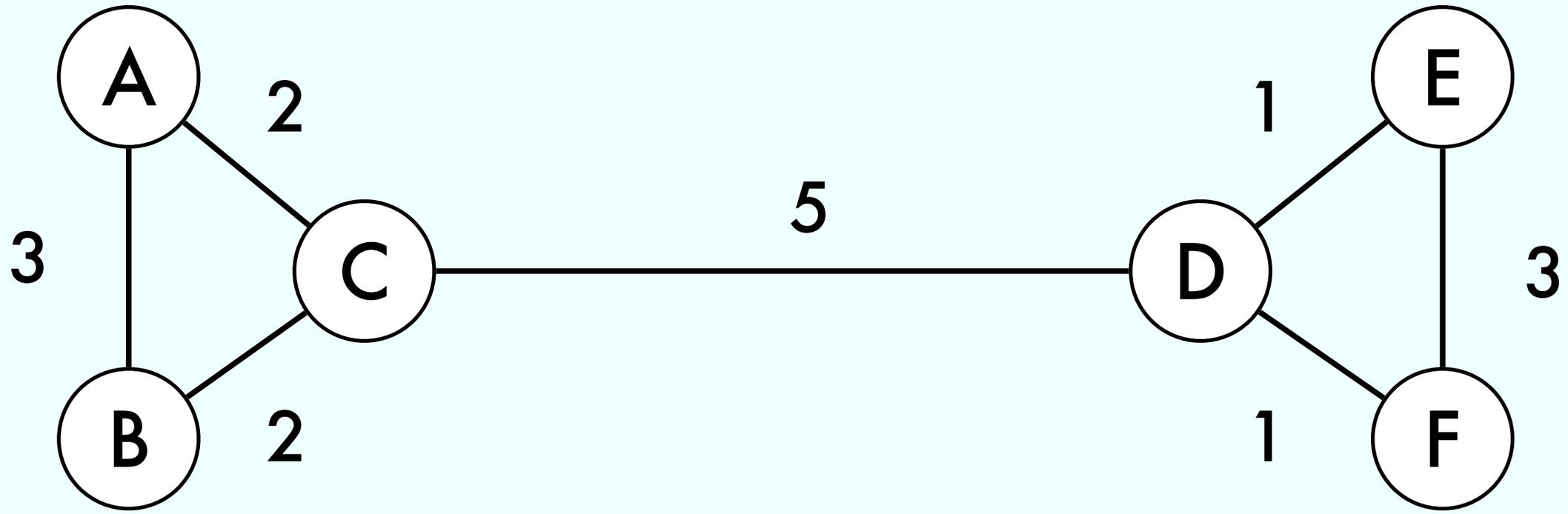
Vertex

Edge

$$\text{spacing}\ (V_1, V_2)$$

Weight — Vertex — Vertex — Edge

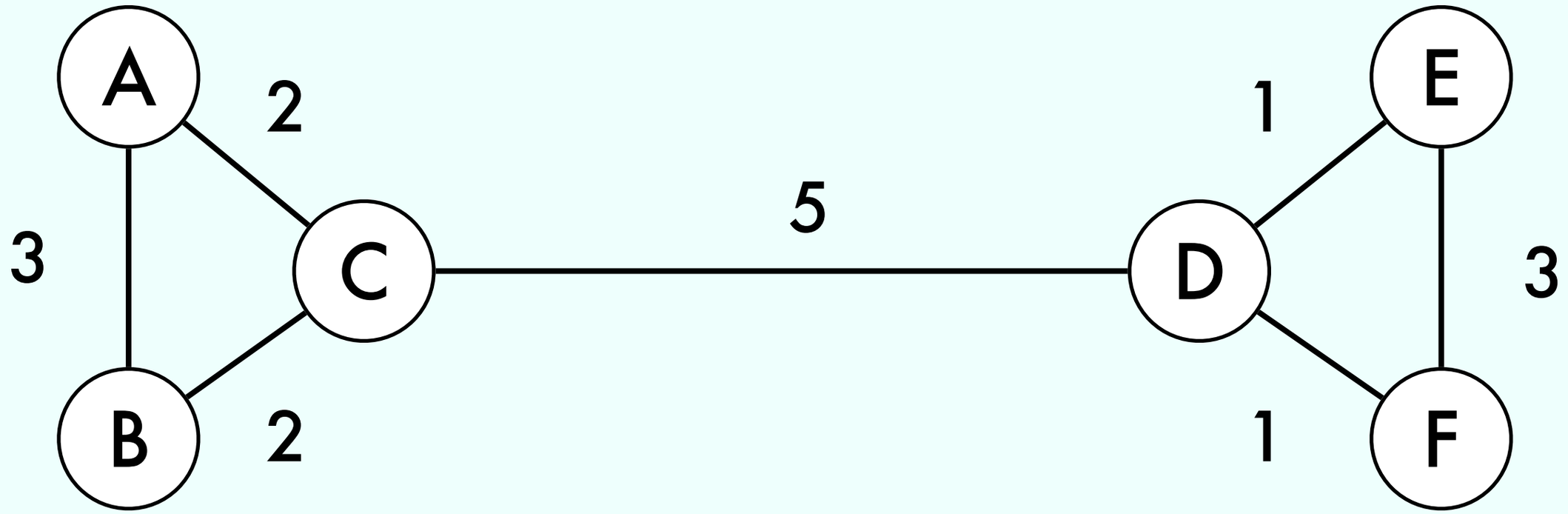
$$\min \{\ w(v_1, v_2) \mid v_1 \in V_1, v_2 \in V_2, (v_1, v_2) \in E\ \}$$



$$\text{spacing}\,(V_1, V_2)$$

Weight — Vertex — Vertex — Edge

$$\min \{\, w(v_1, v_2) \mid v_1 \in V_1, v_2 \in V_2, (v_1, v_2) \in E \,\}$$

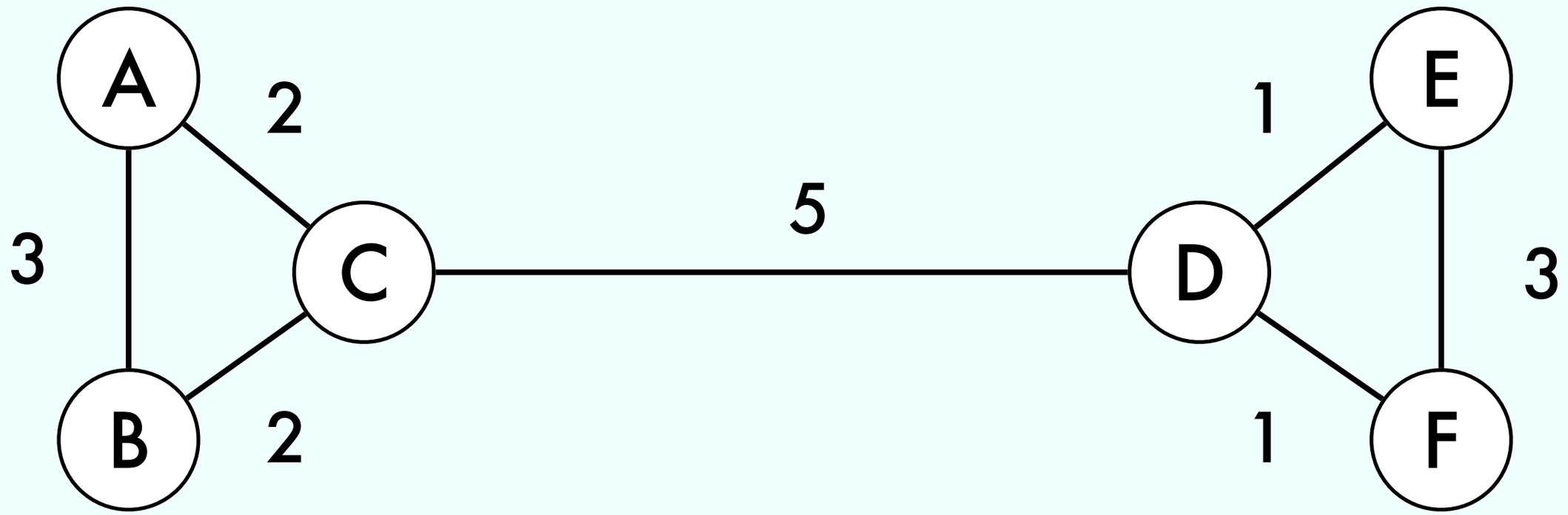


spacing({A,B} , {C,D,E,F}) =

$$\text{spacing}\,(V_1, V_2)$$

$$\min \{\, w(v_1, v_2) \mid v_1 \in V_1, v_2 \in V_2, (v_1, v_2) \in E \,\}$$

Weight — Vertex — Vertex — Edge



spacing({A,B} , {C,D,E,F}) = 2

Compute

$$\max \{ \text{spacing}(V_1, V_2) \mid V_1, V_2 \text{ partitions of } V \}$$

Compute

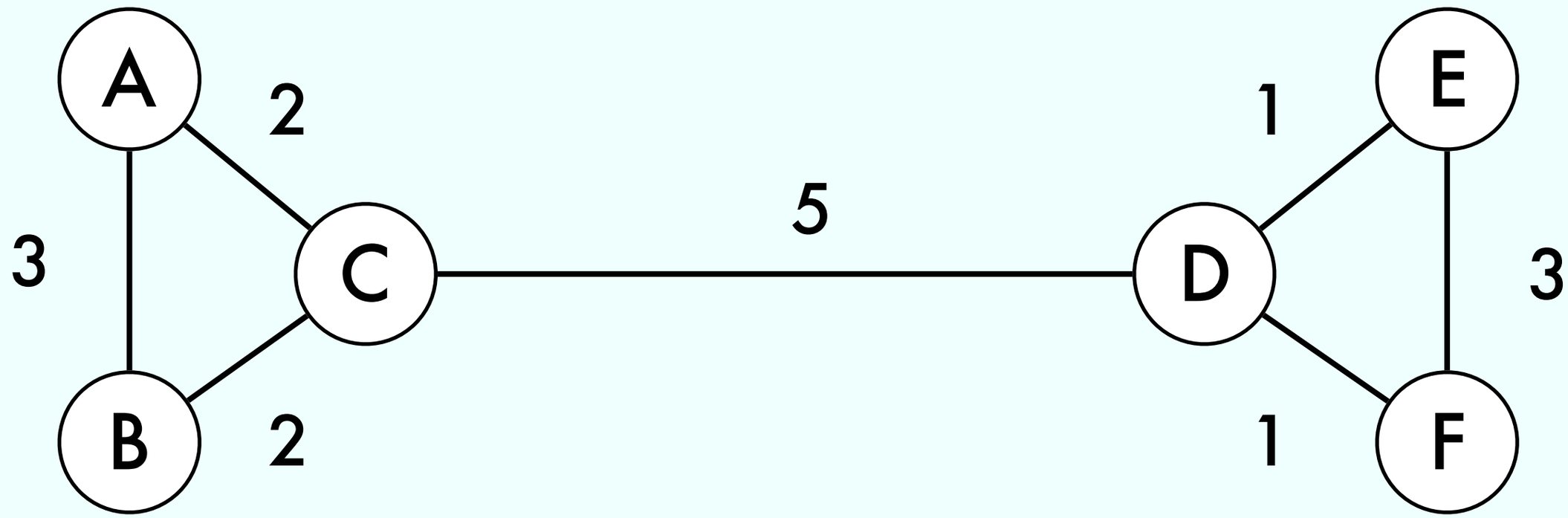
$$\max \{ \text{spacing}(V_1, V_2) \mid V_1, V_2 \text{ partitions of } V \}$$

$$\text{spacing}(V_1, V_2) = \min \{ w(v_1, v_2) \mid v_1 \in V_1, v_2 \in V_2, (v_1, v_2) \in E \}$$

Compute

$$\max \{ \text{spacing}(V_1, V_2) \mid V_1, V_2 \text{ partitions of } V \}$$

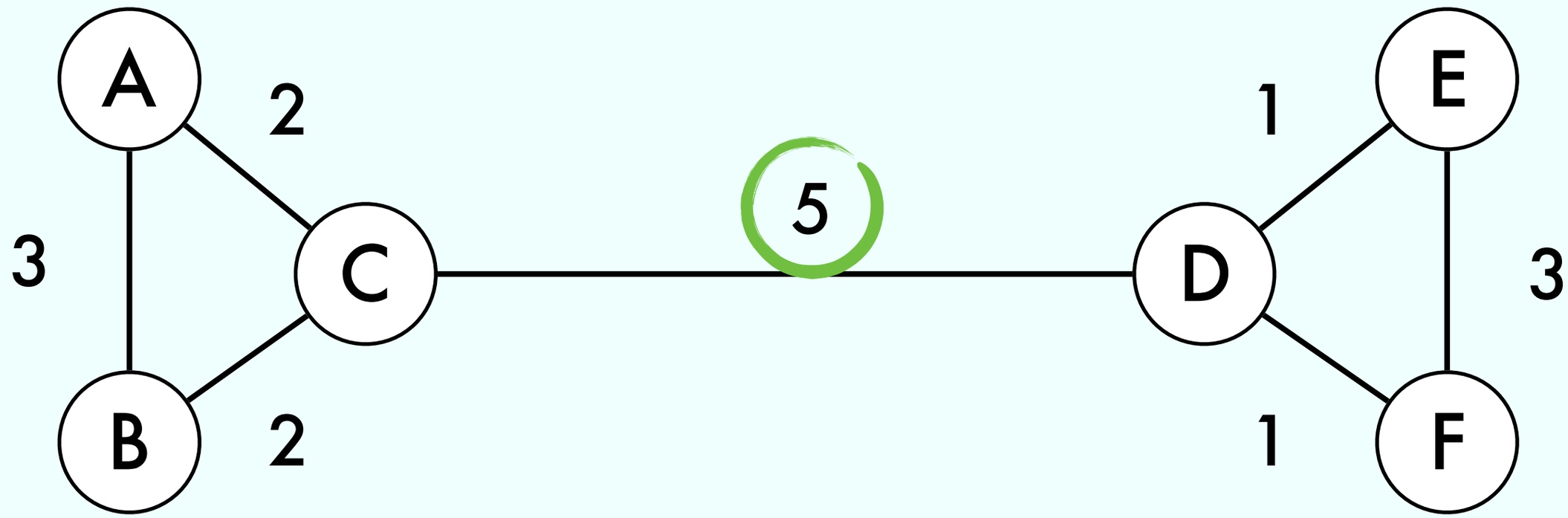
$$\text{spacing}(V_1, V_2) = \min \{ w(v_1, v_2) \mid v_1 \in V_1, v_2 \in V_2, (v_1, v_2) \in E \}$$



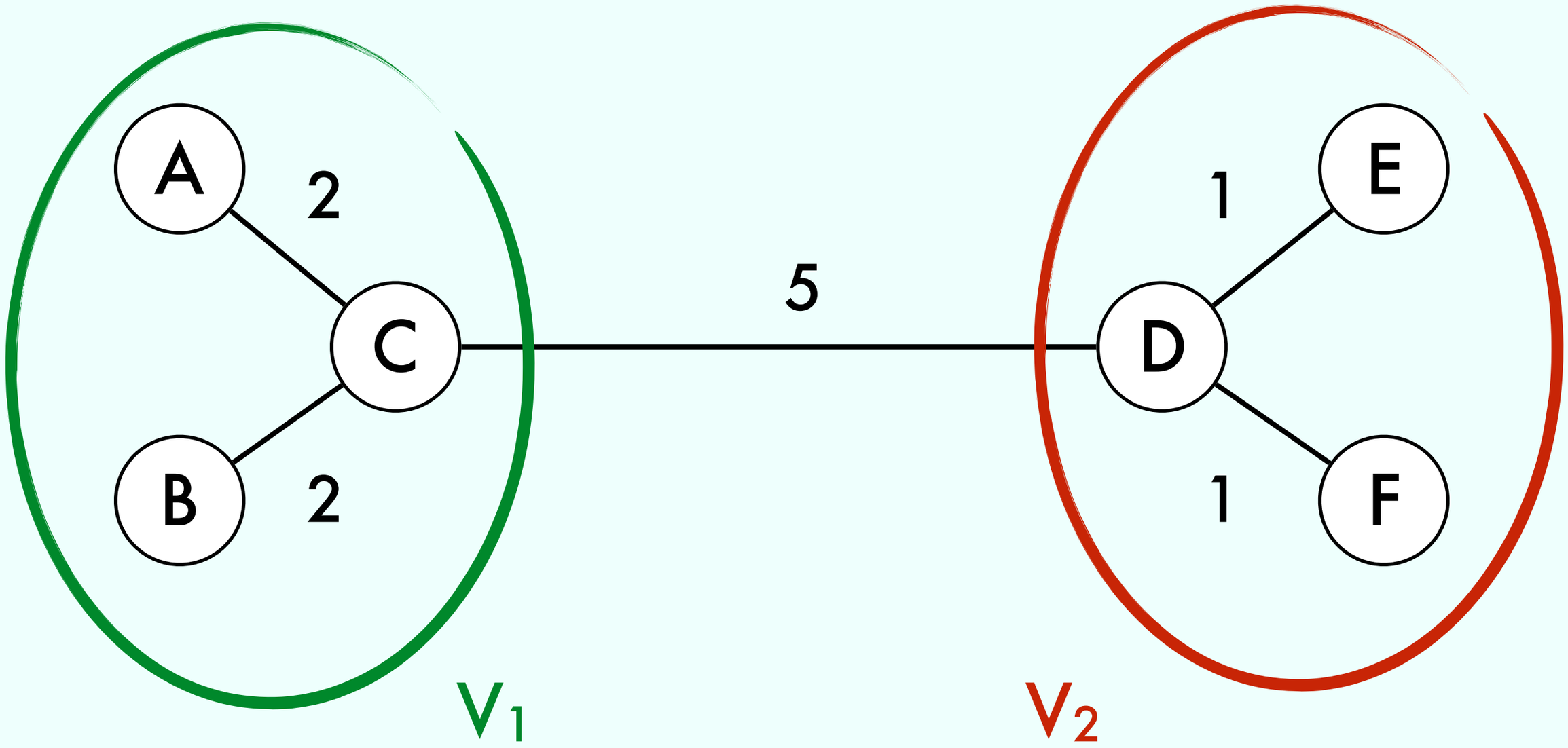
Compute

$$\max \{ \text{spacing}(V_1, V_2) \mid V_1, V_2 \text{ partitions of } V \}$$

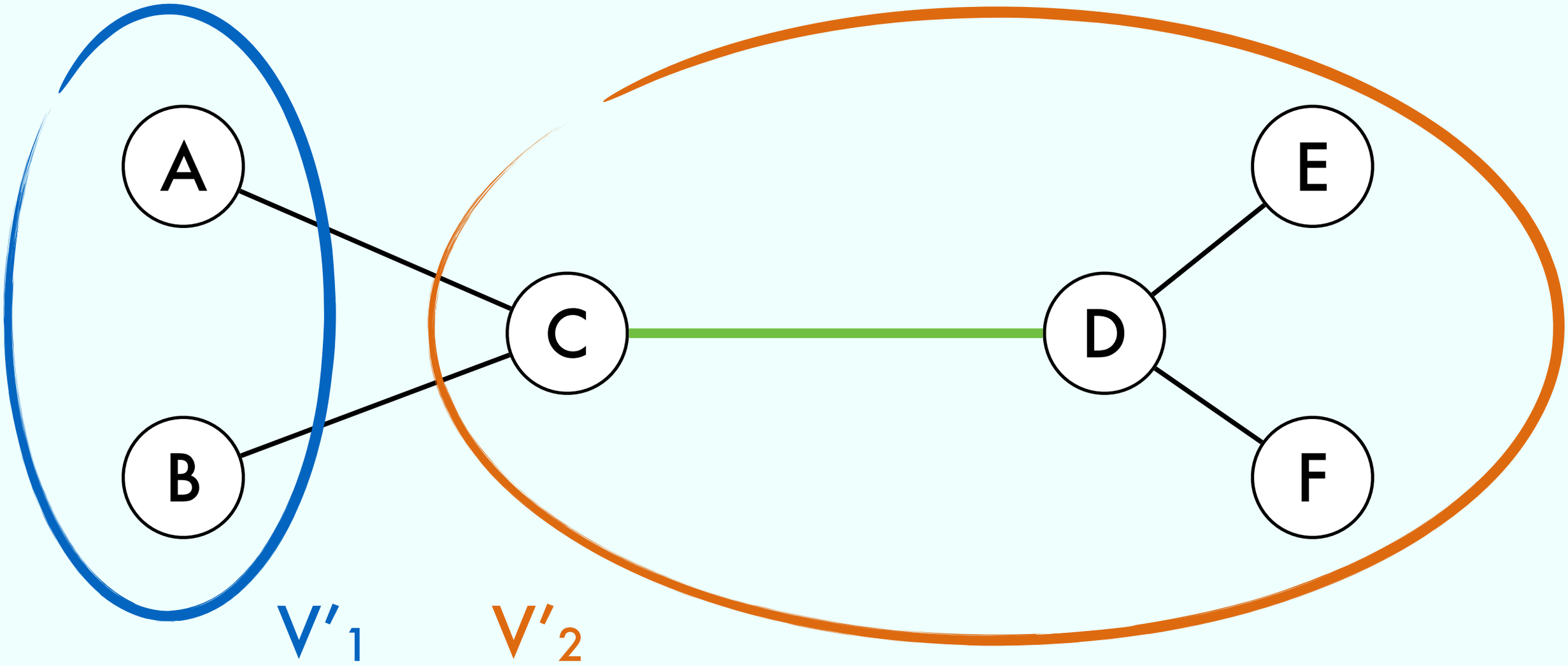
$$\text{spacing}(V_1, V_2) = \min \{ w(v_1, v_2) \mid v_1 \in V_1, v_2 \in V_2, (v_1, v_2) \in E \}$$

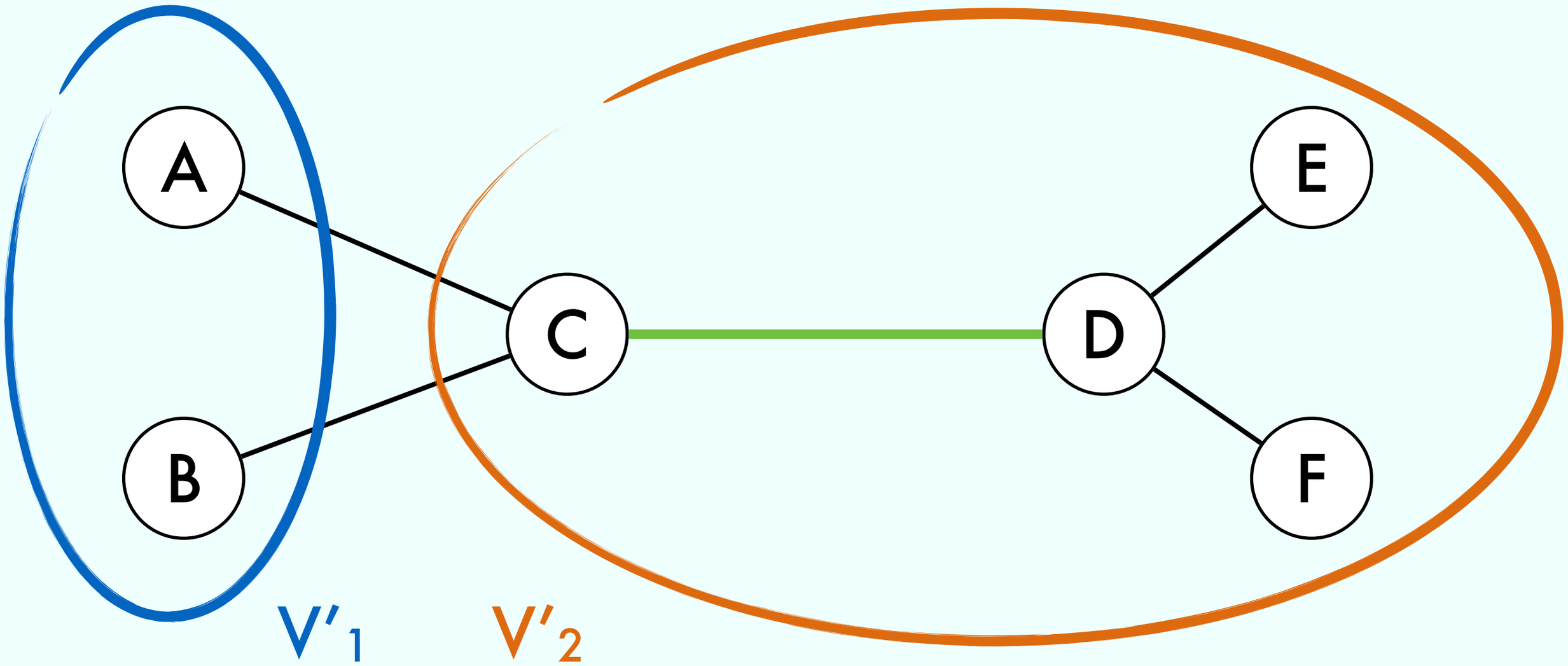


Problem 7

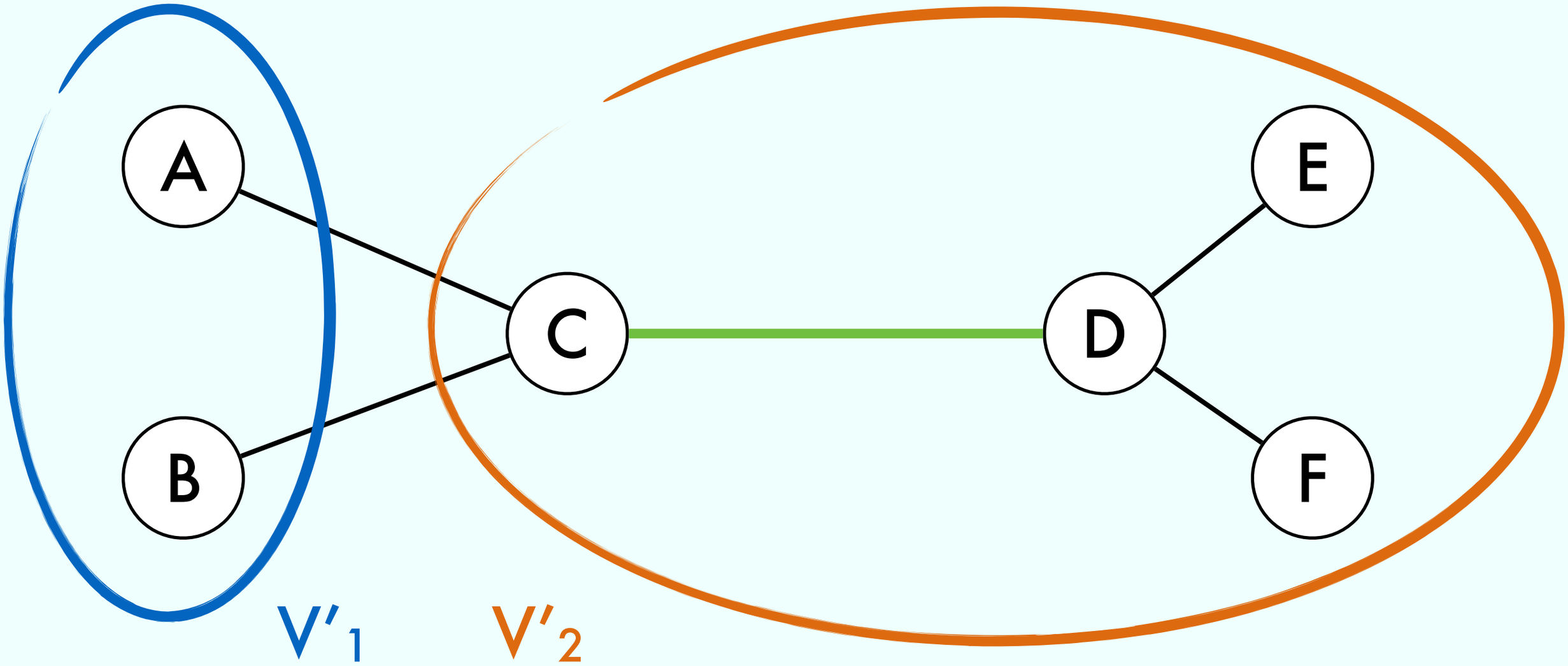


Minimum Spanning Tree


A
B
C
D
E
F
V′1
V′2

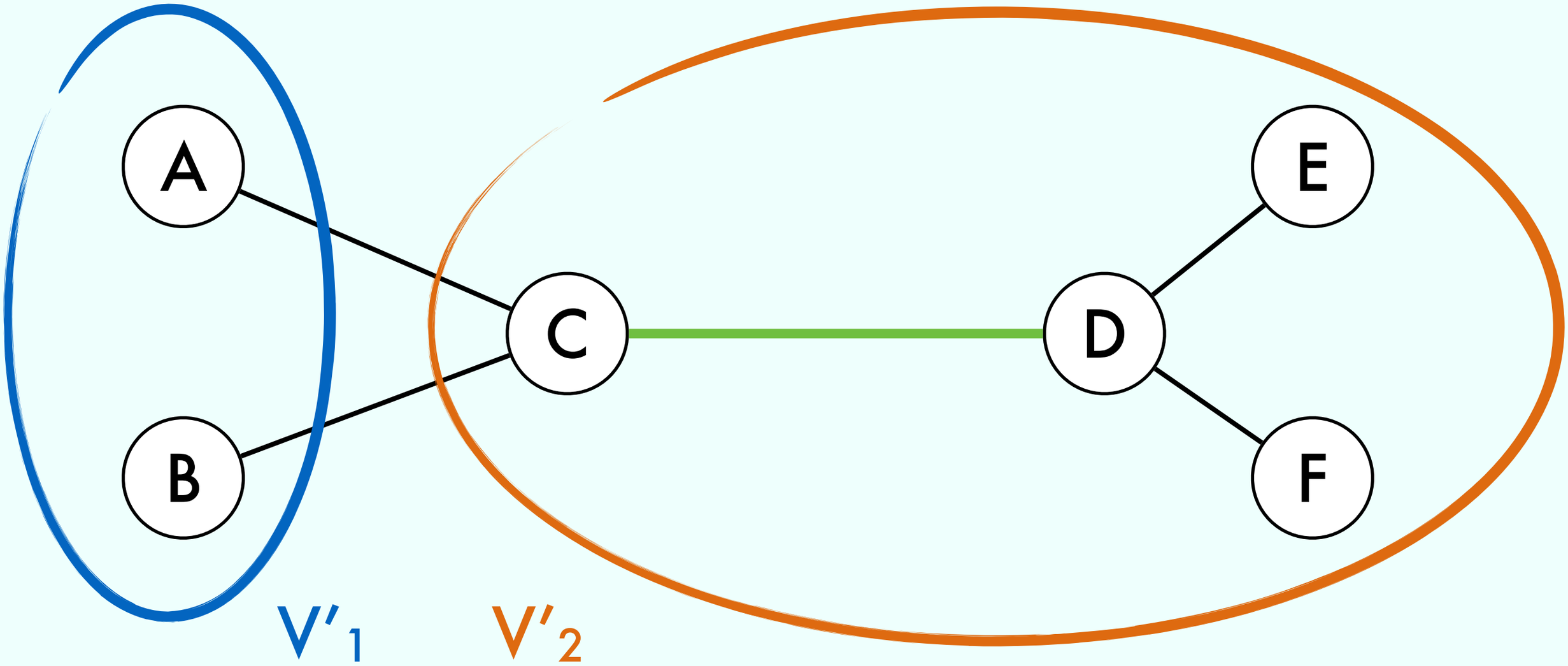


Absurd



Absurd

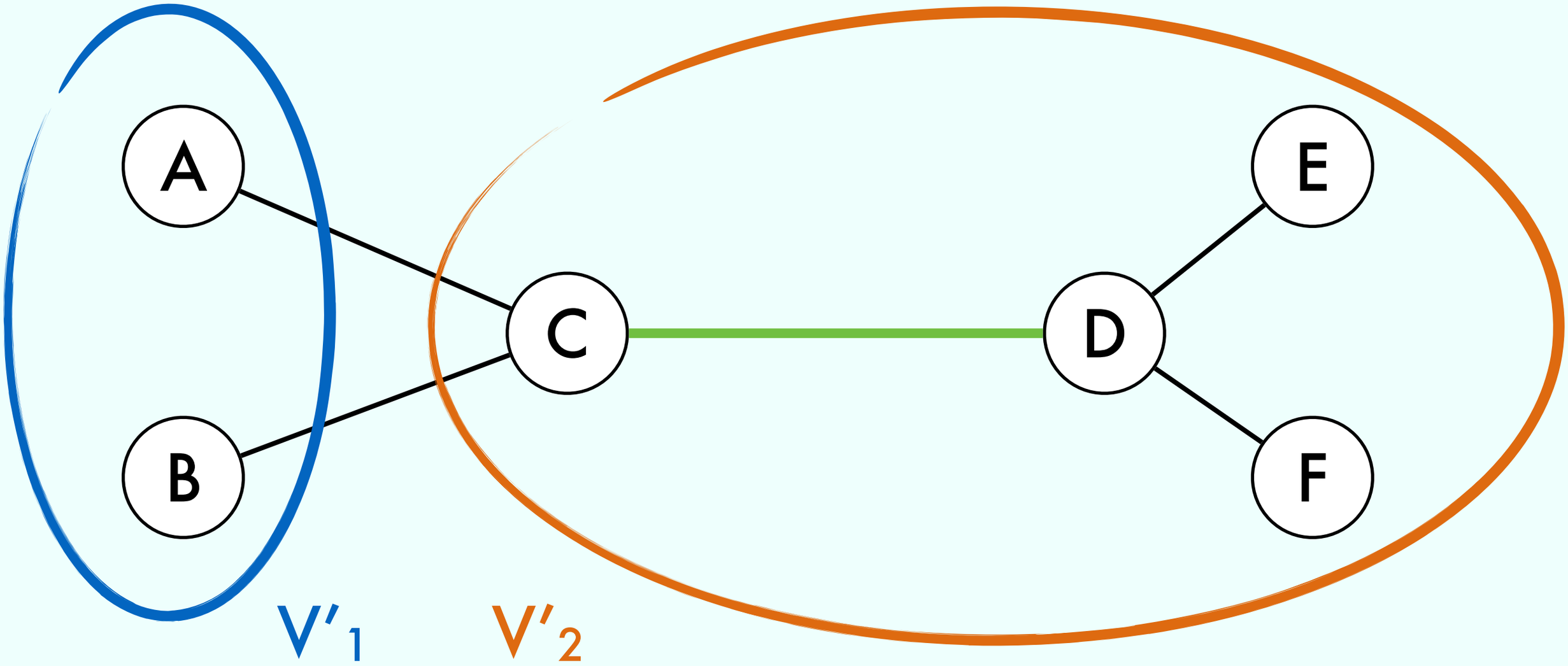
$w(C, D) < w(B, C)$



Absurd

larger spacing

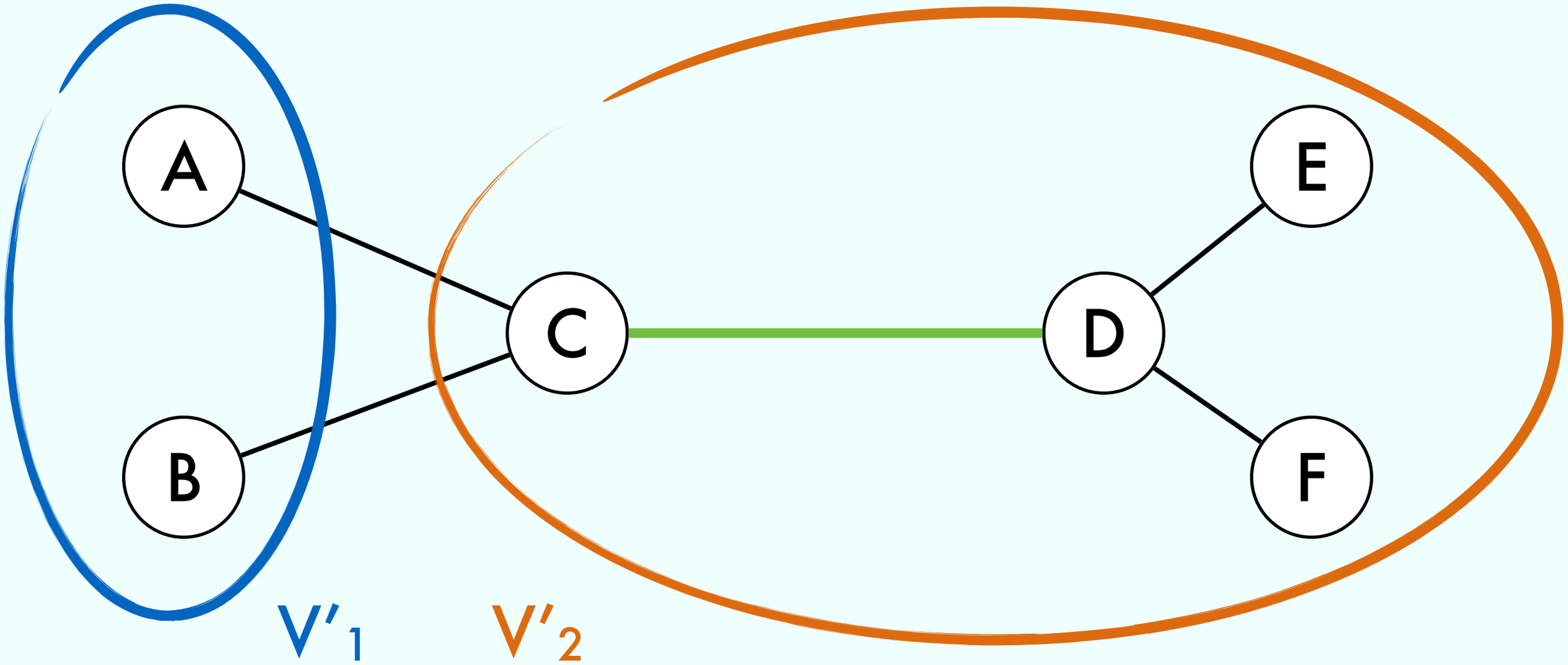
w(C, D) < w(B, C)



Absurd

larger spacing

$w(B, C) \leq w(C, D) < w(B, C)$



Absurd

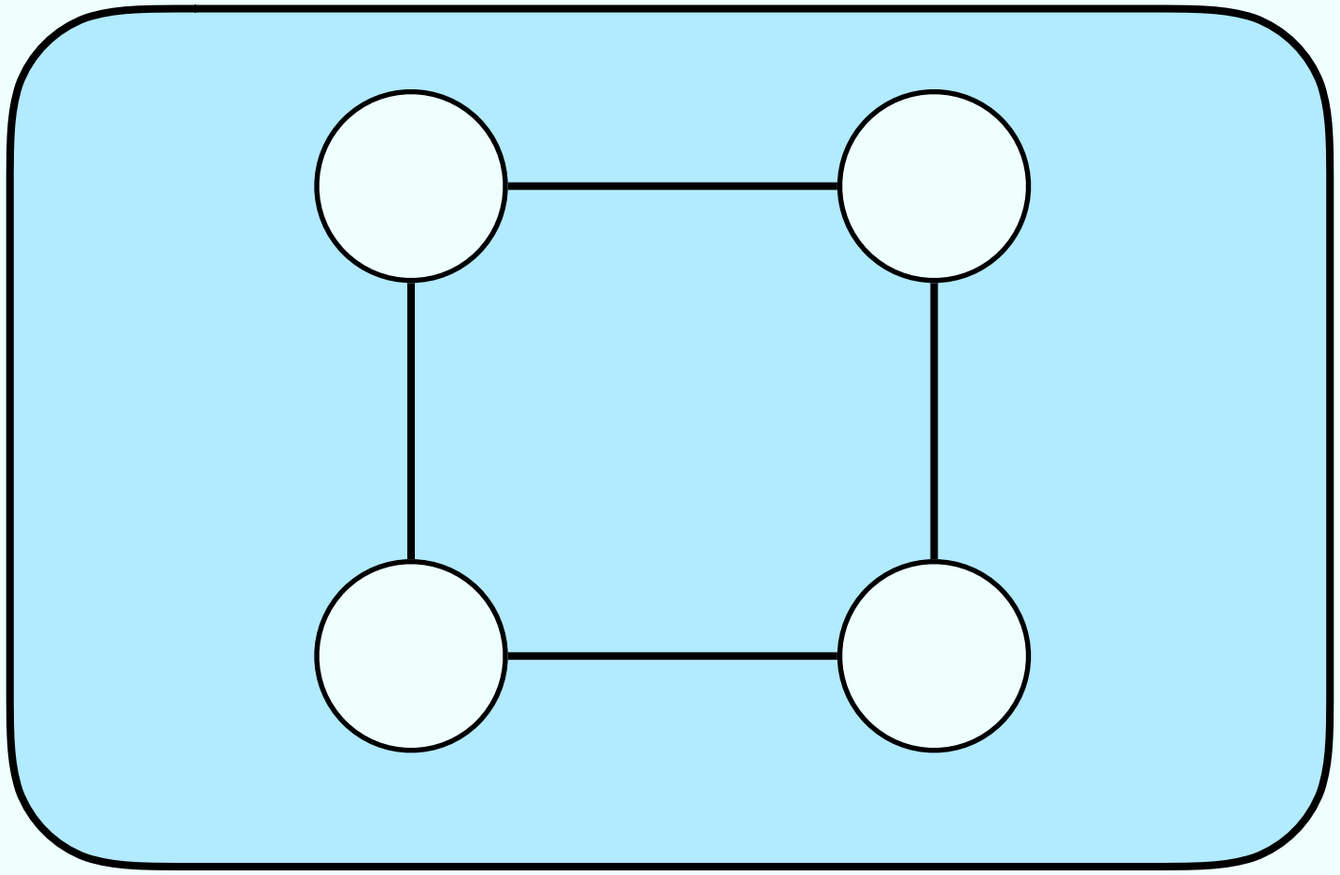
MST

larger spacing

$w(B, C) \leq w(C, D) < w(B, C)$

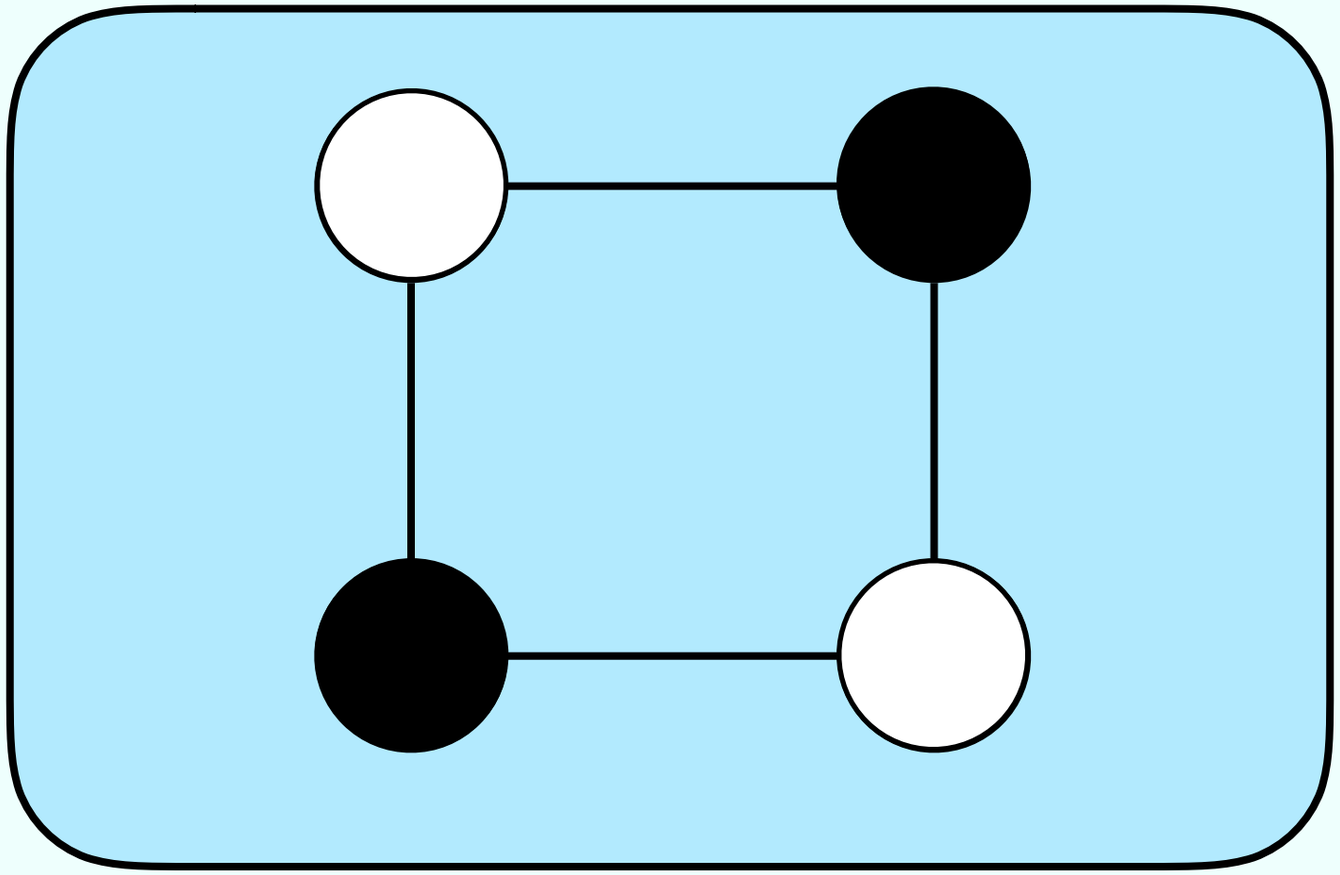
Exercise 6 from 12/04

Colour the graph in black and white.
Neighbours nodes must have different colours



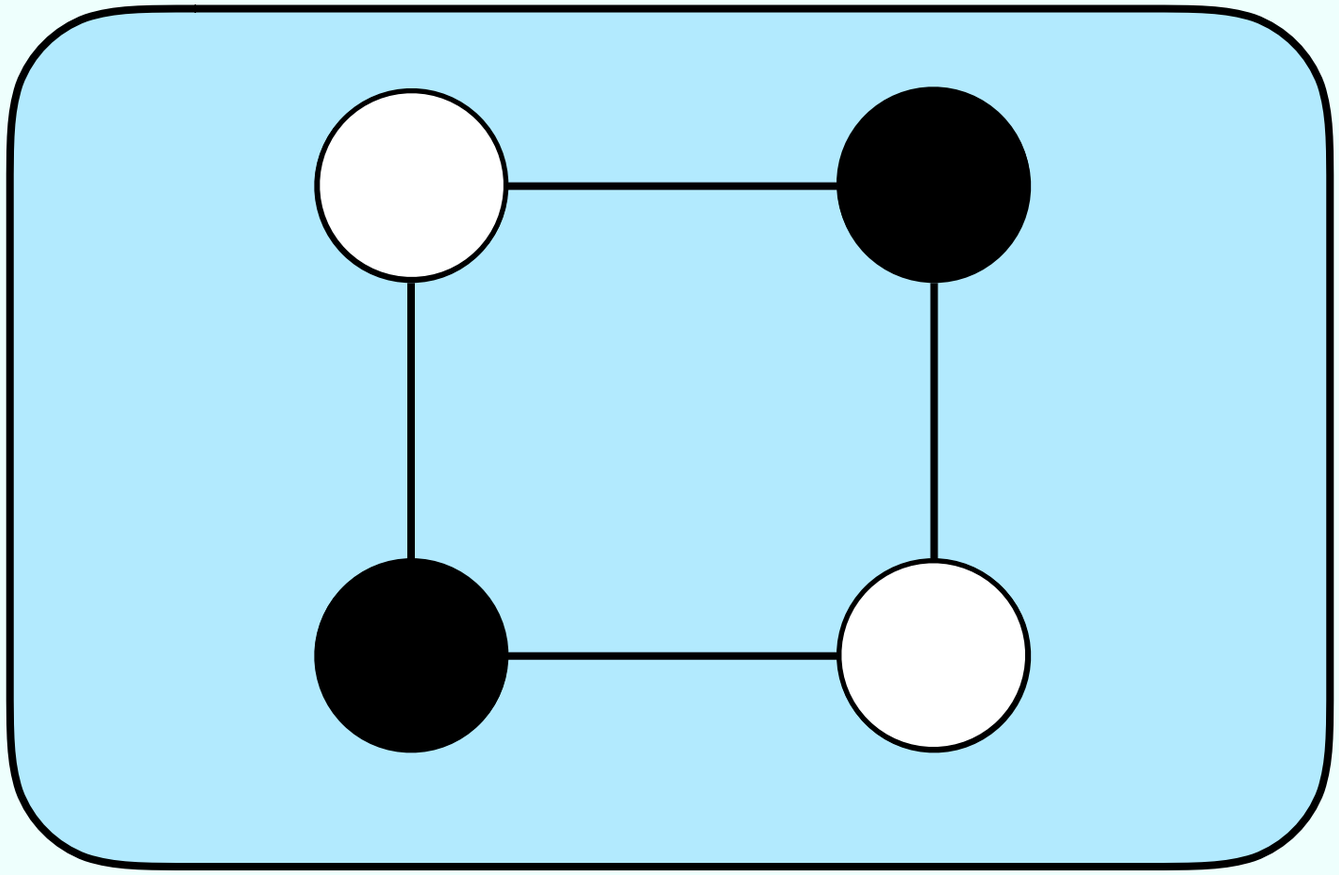
Exercise 6 from 12/04

Colour the graph in black and white.
Neighbours nodes must have different colours



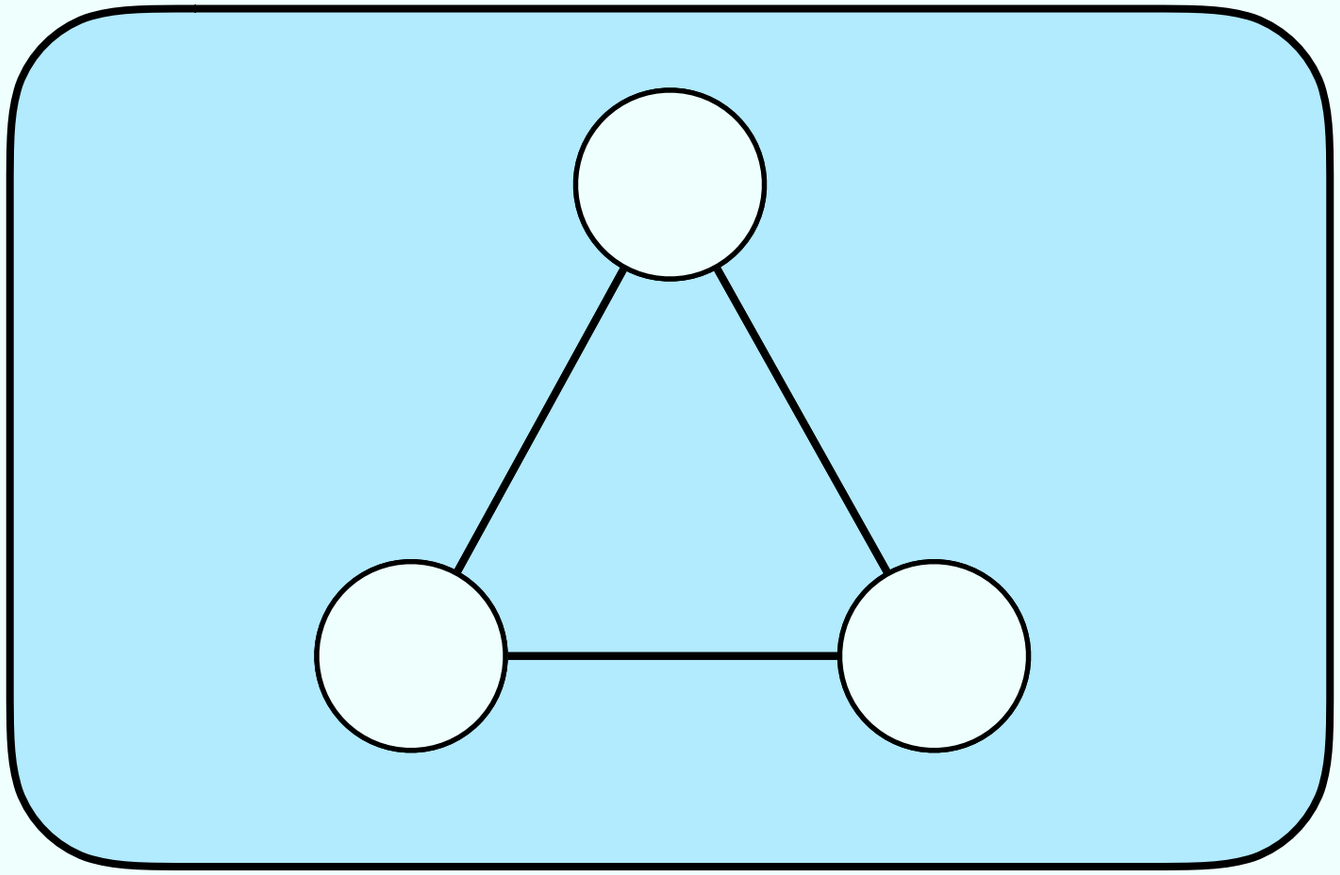
Exercise 6 from 12/04

Colour the graph in black and white.
Neighbours nodes must have different colours



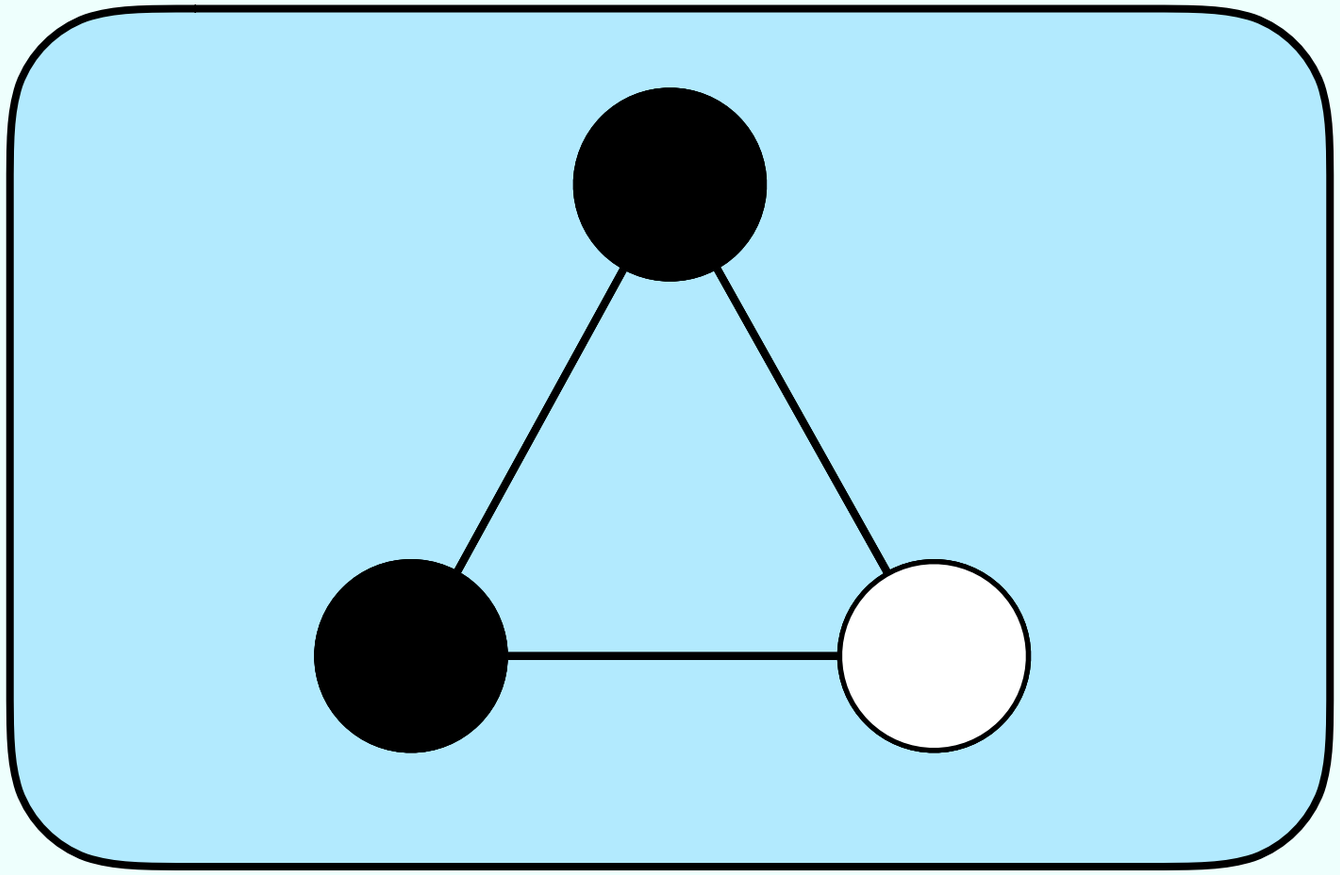
Exercise 6 from 12/04

Colour the graph in black and white.
Neighbours nodes must have different colours



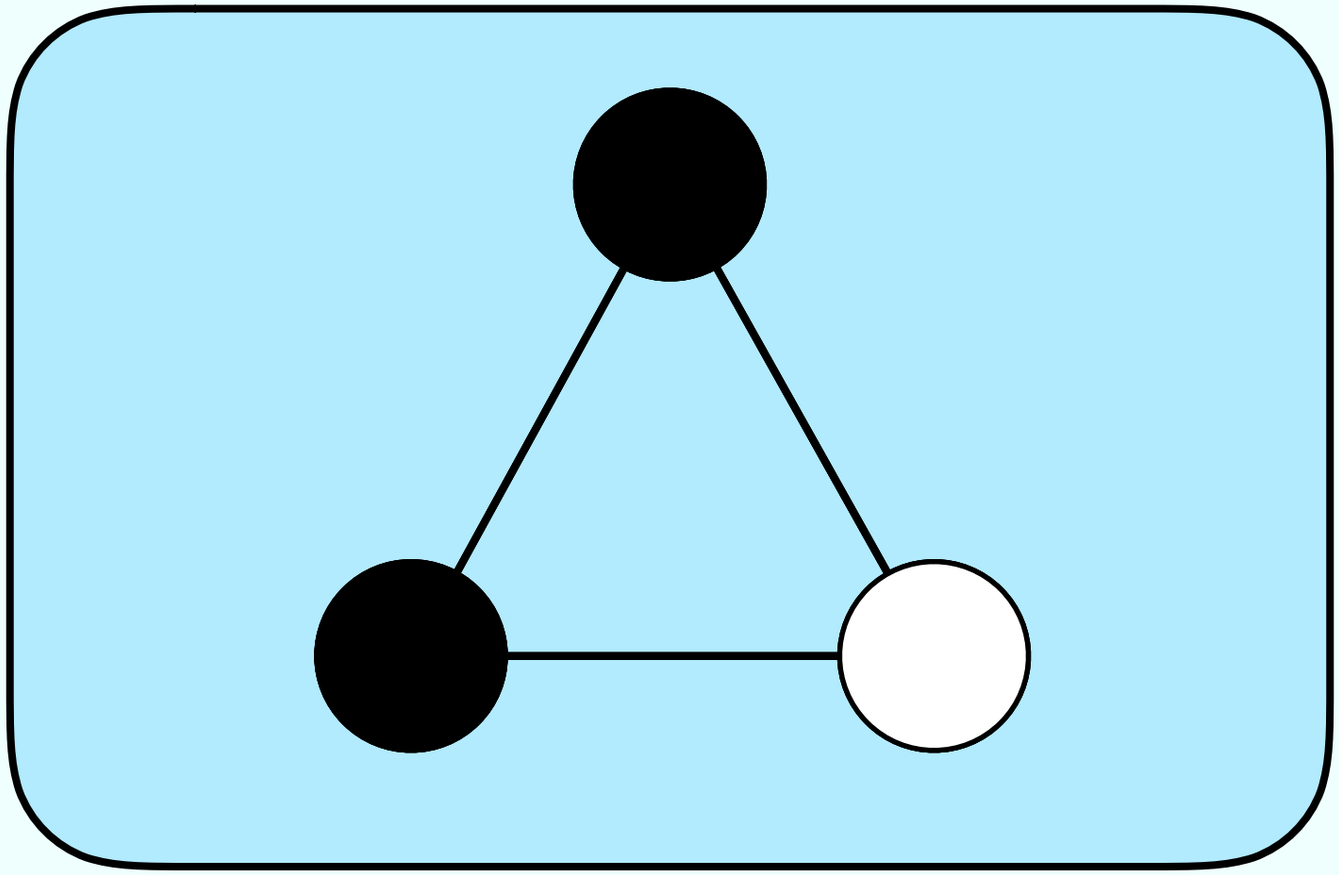
Exercise 6 from 12/04

Colour the graph in black and white.
Neighbours nodes must have different colours



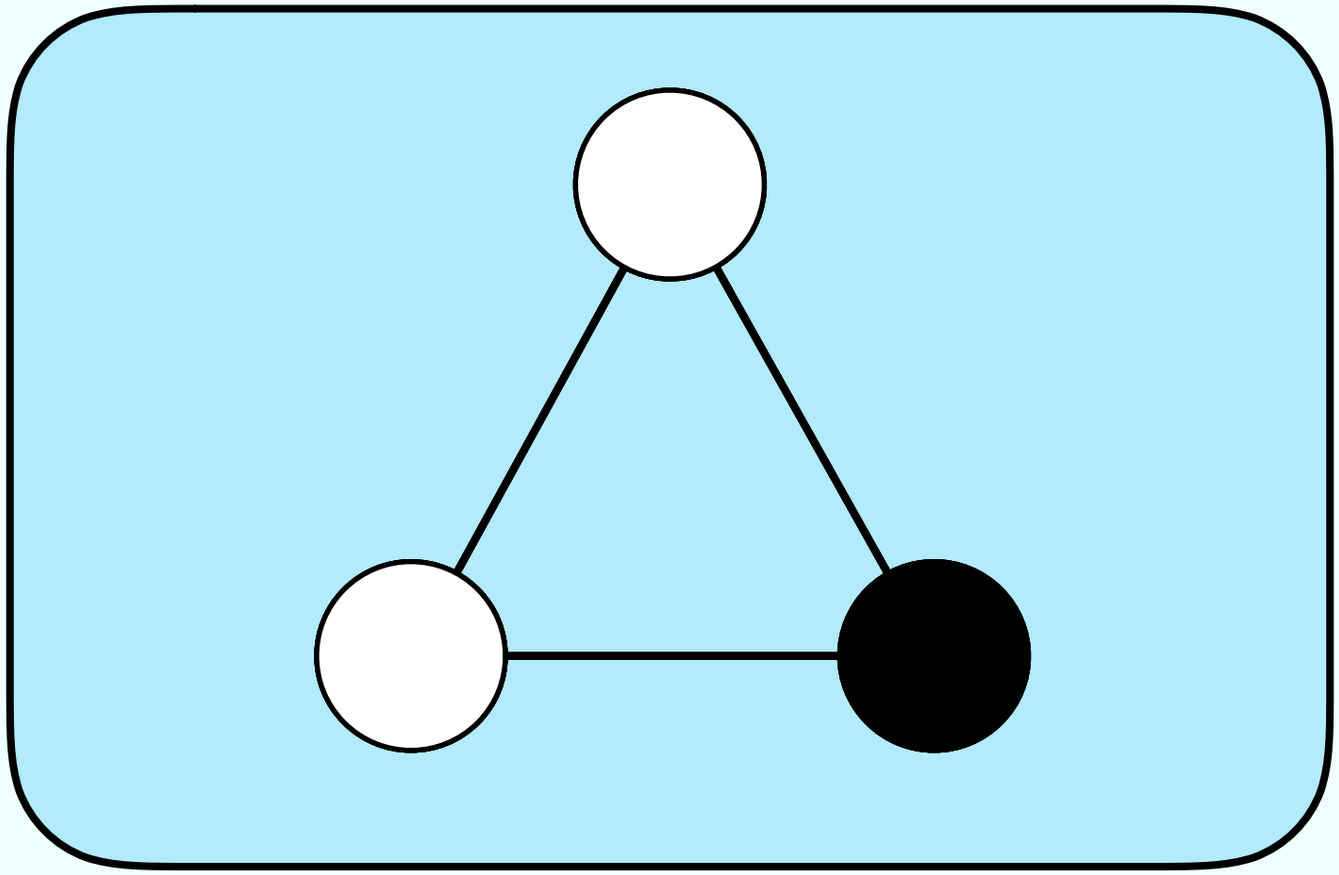
Exercise 6 from 12/04

Colour the graph in black and white.
Neighbours nodes must have different colours



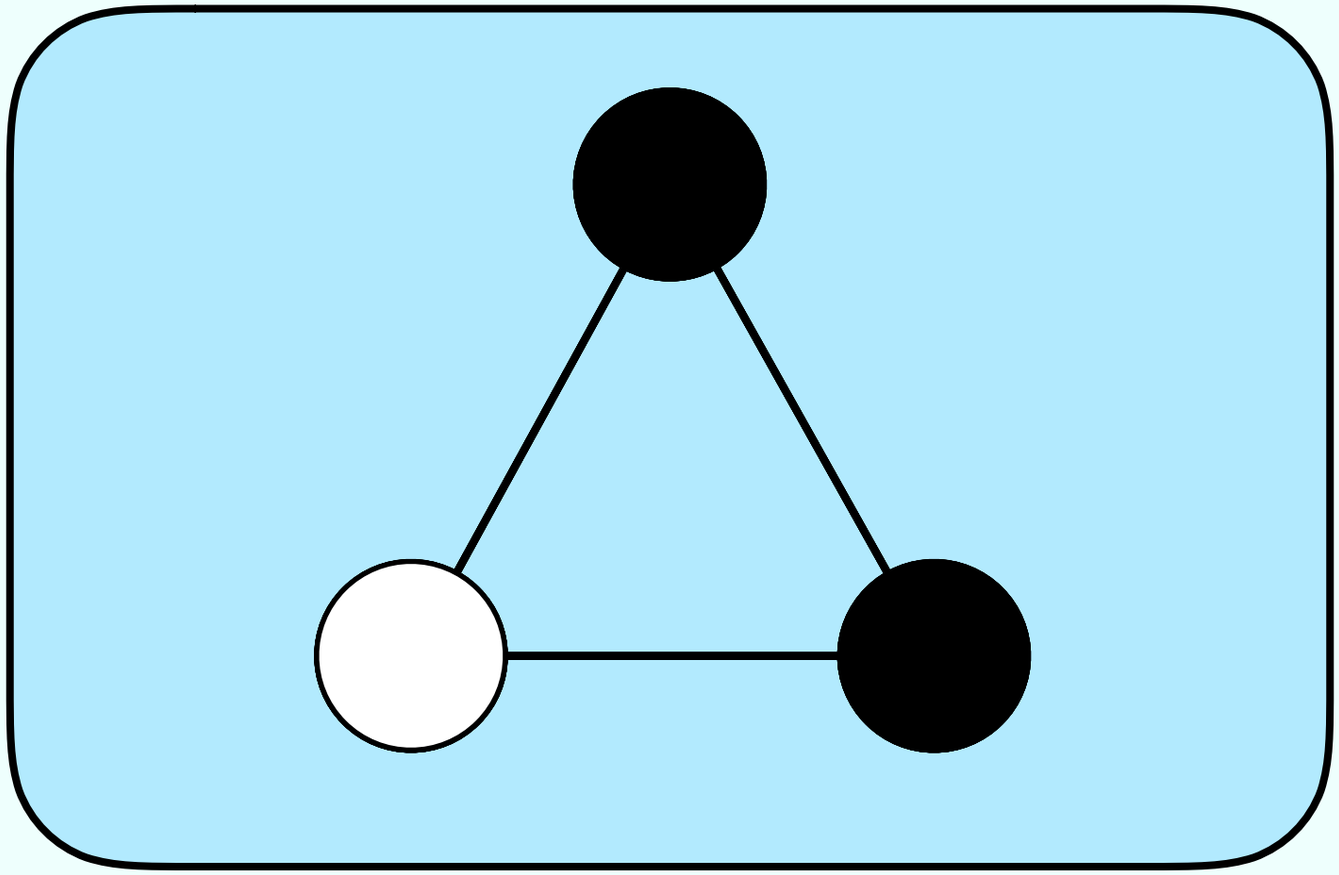
Exercise 6 from 12/04

Colour the graph in black and white.
Neighbours nodes must have different colours



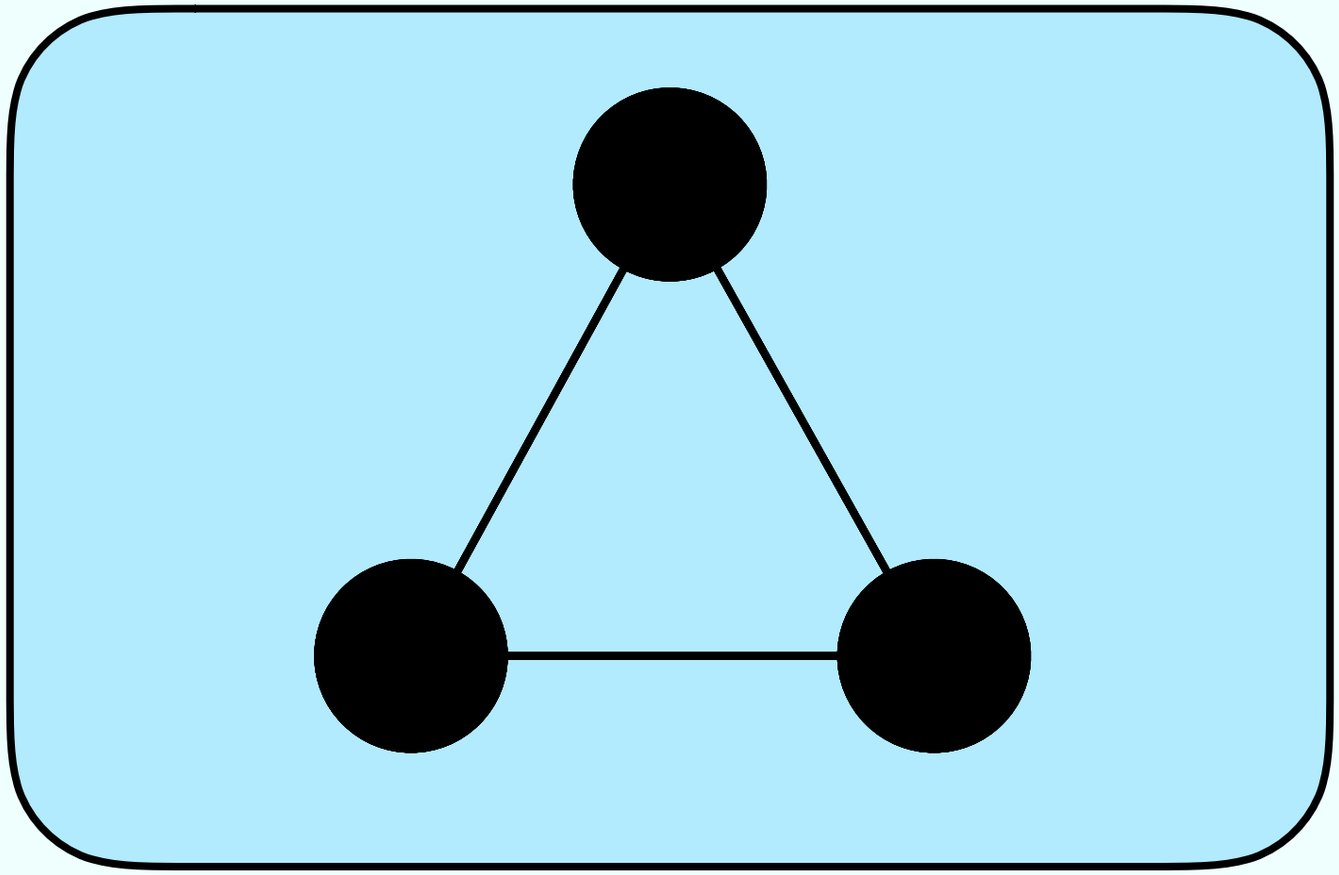
Exercise 6 from 12/04

Colour the graph in black and white.
Neighbours nodes must have different colours



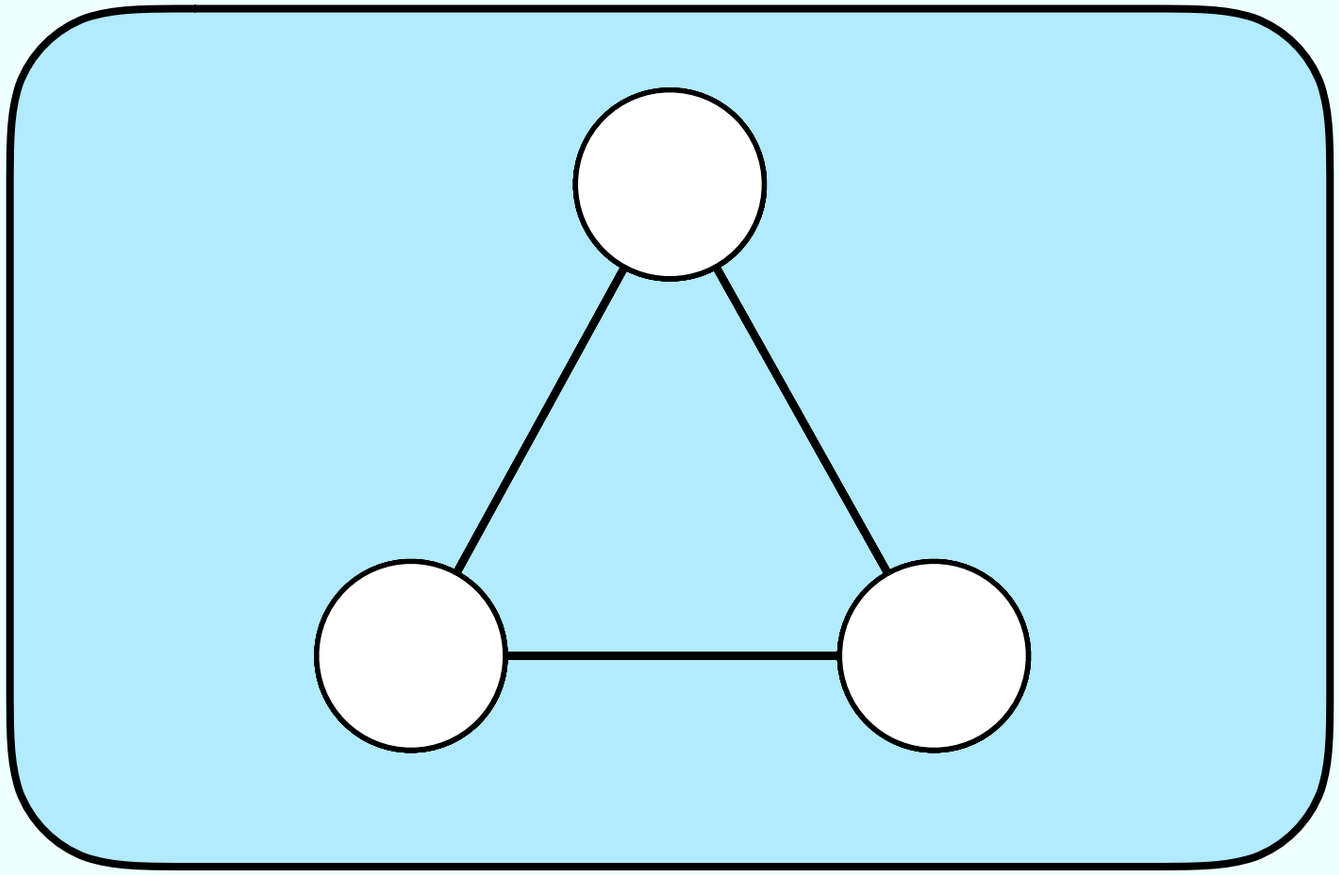
Exercise 6 from 12/04

Colour the graph in black and white.
Neighbours nodes must have different colours



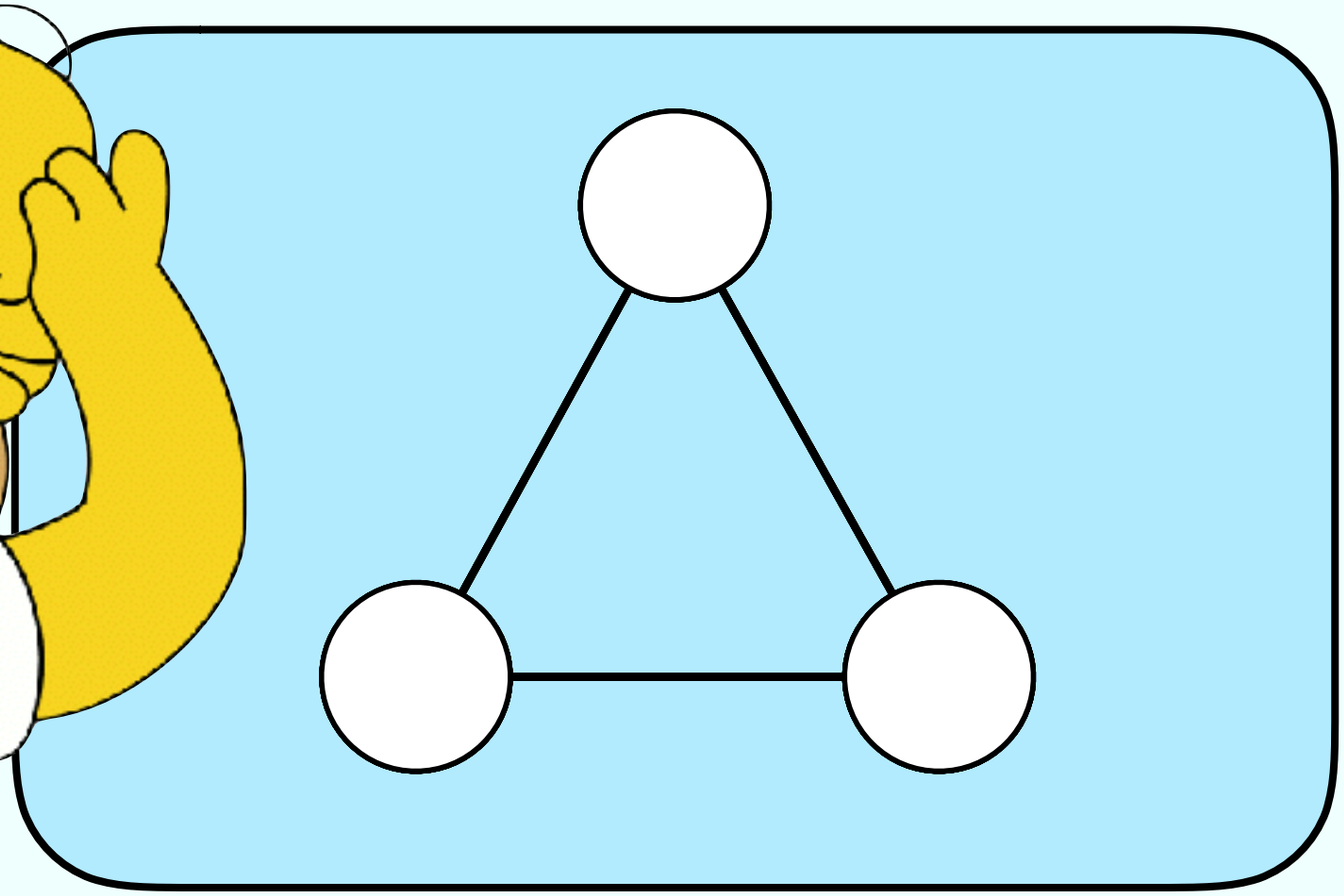
Exercise 6 from 12/04

Colour the graph in black and white.
Neighbours nodes must have different colours



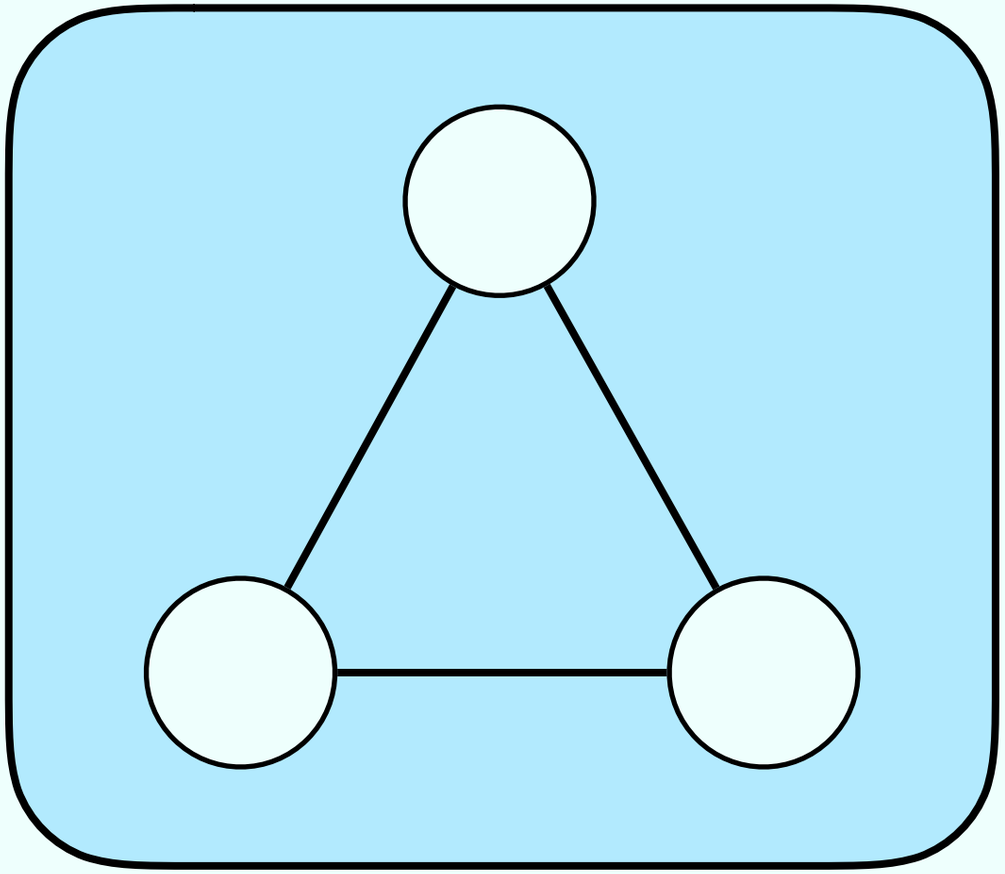
Exercise 6 from 12/04

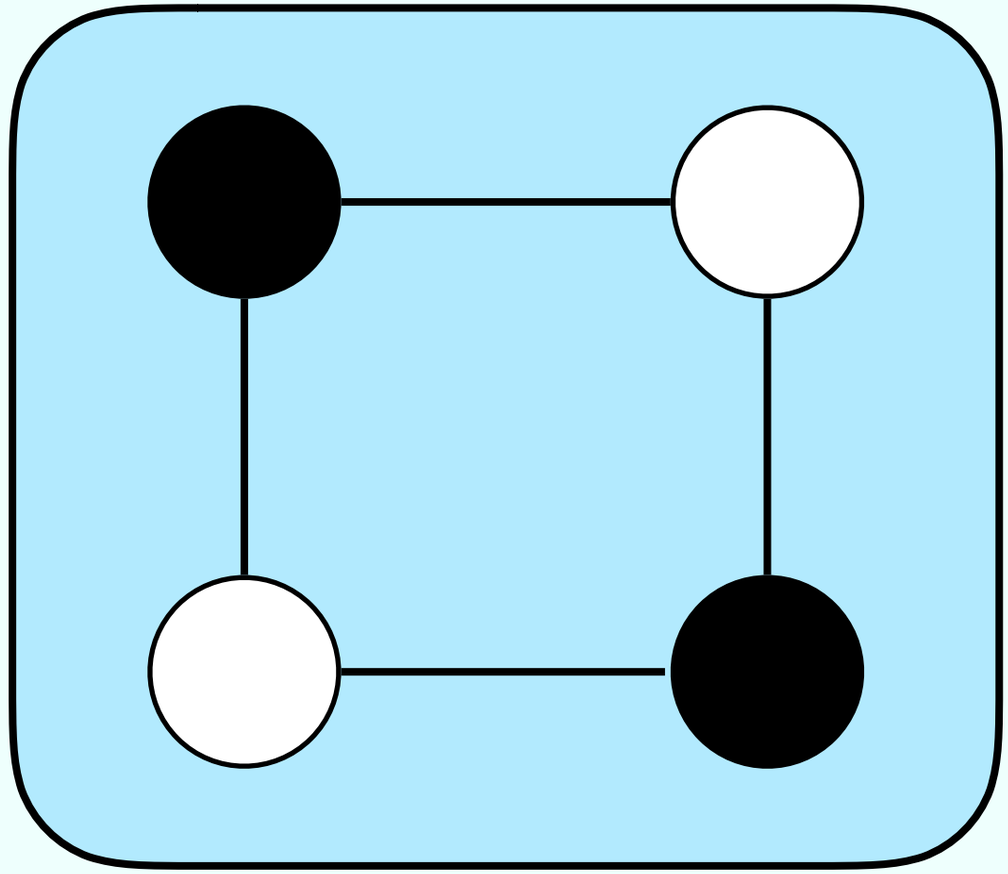
Colour the graph in black and white.
Neighbours nodes must have different colours



Better Question

Can we colour a graph in black and white,
such that neighbours nodes have different colours?

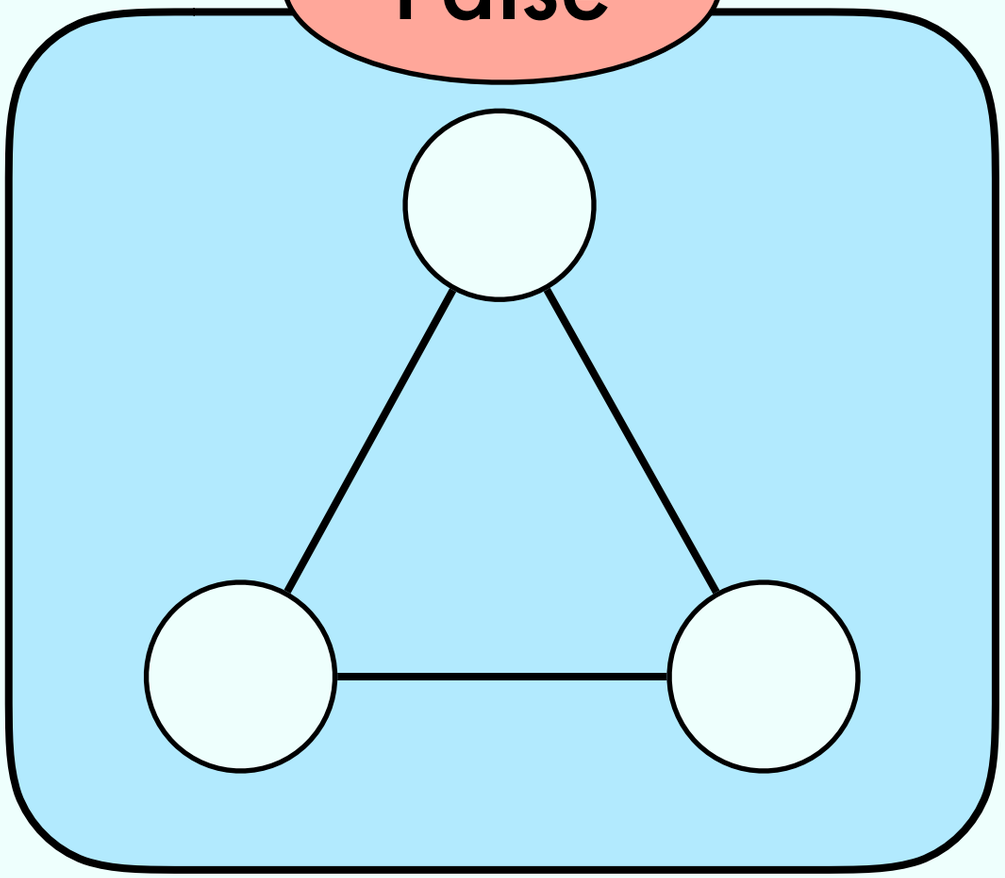


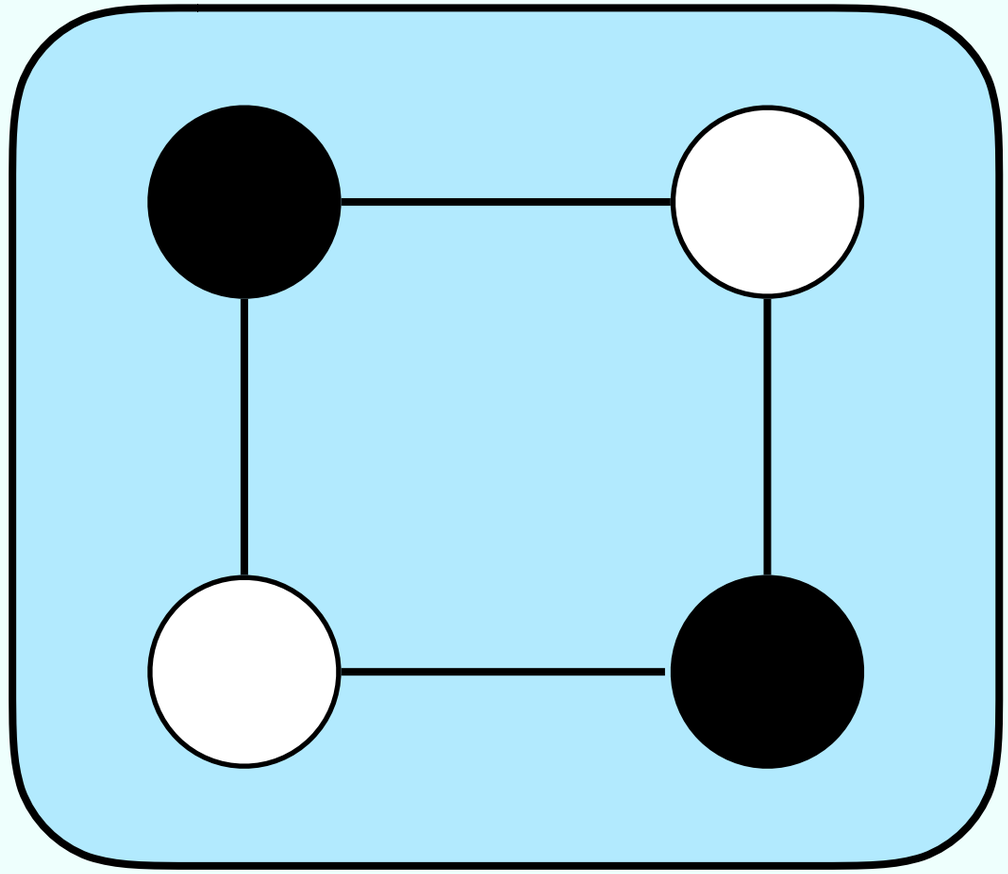


Better Question

Can we colour a graph in black and white,
such that neighbours nodes have different colours?

False

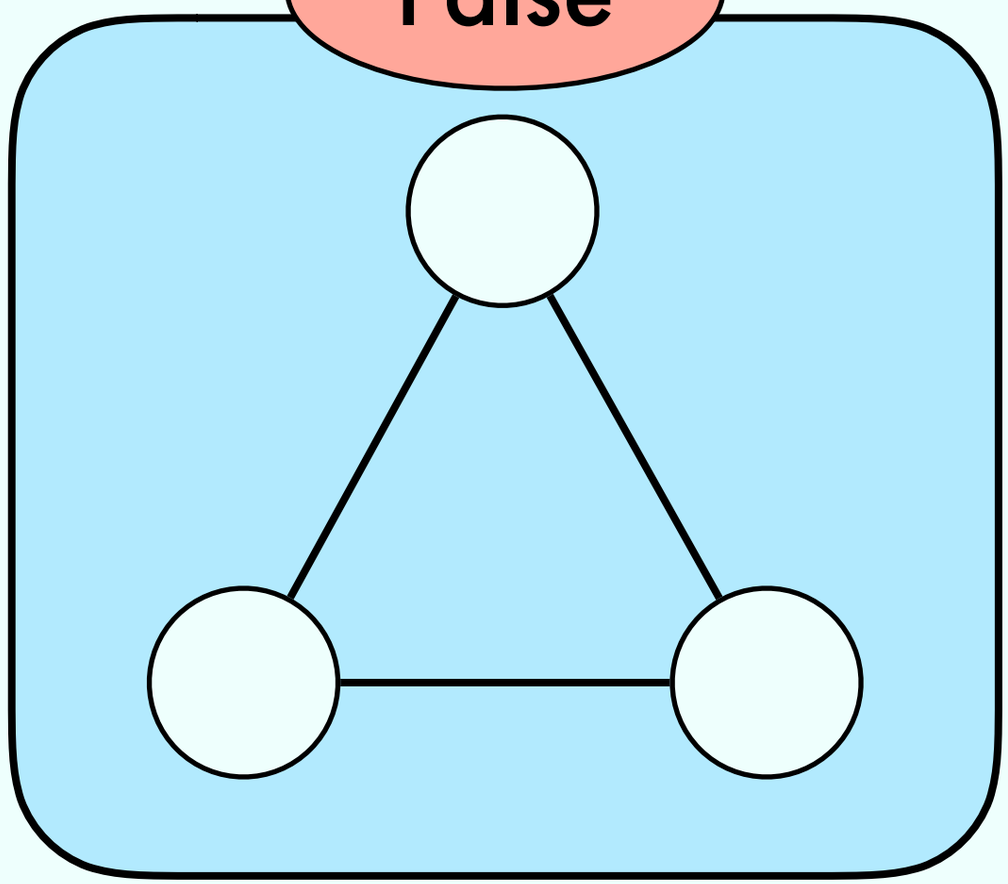




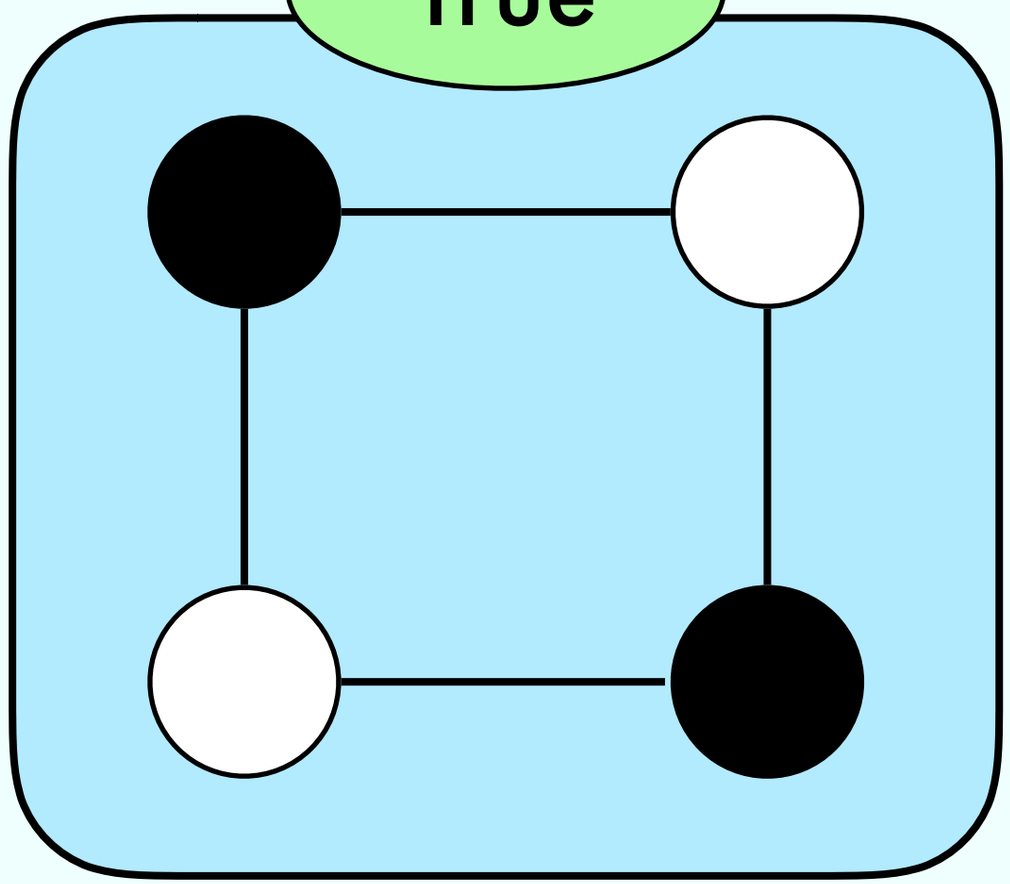
Better Question

Can we colour a graph in black and white,
such that neighbours nodes have different colours?

False



True



Exercise 9.42

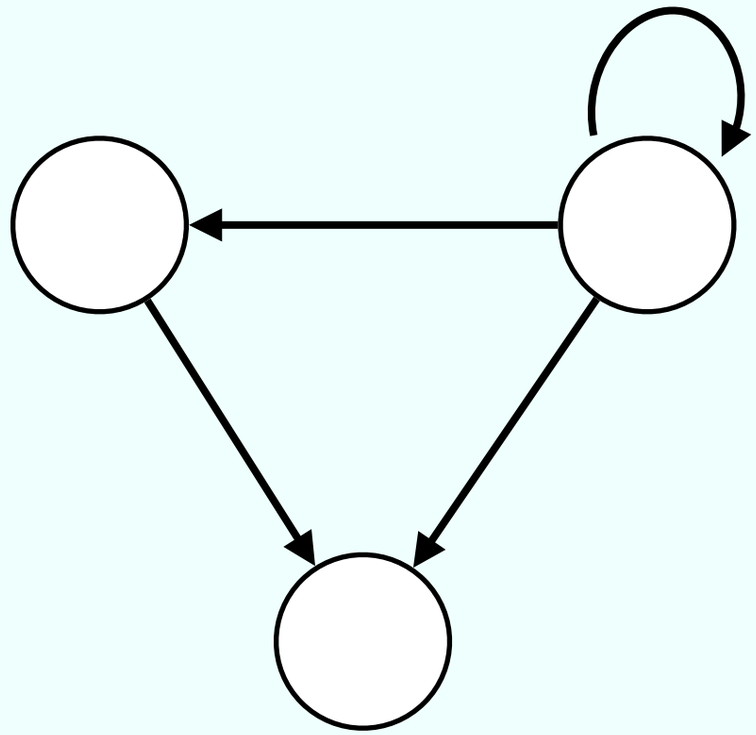
A sink node:

- Has incoming edges from any other node
- Has no outgoing edges

Exercise 9.42

A sink node:

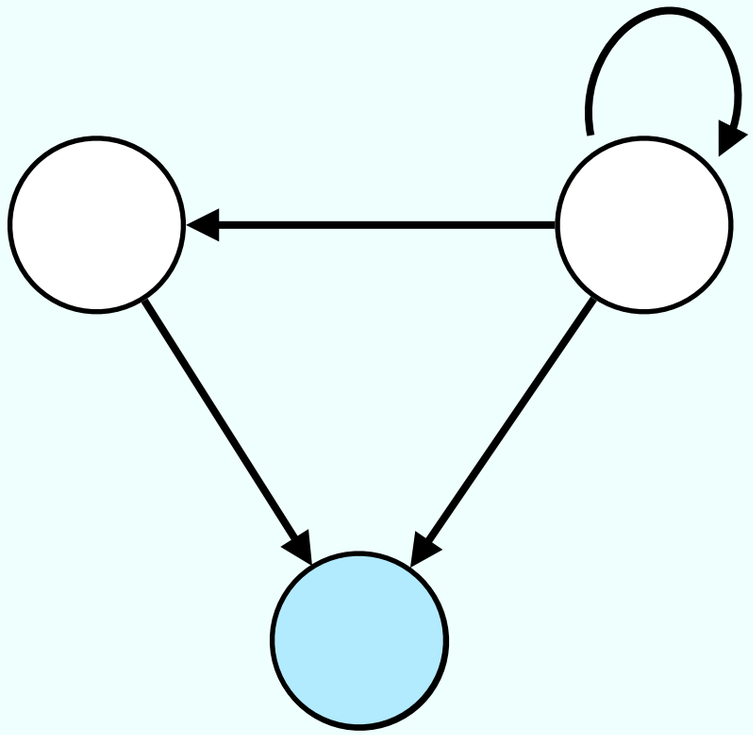
- Has incoming edges from any other node
- Has no outgoing edges

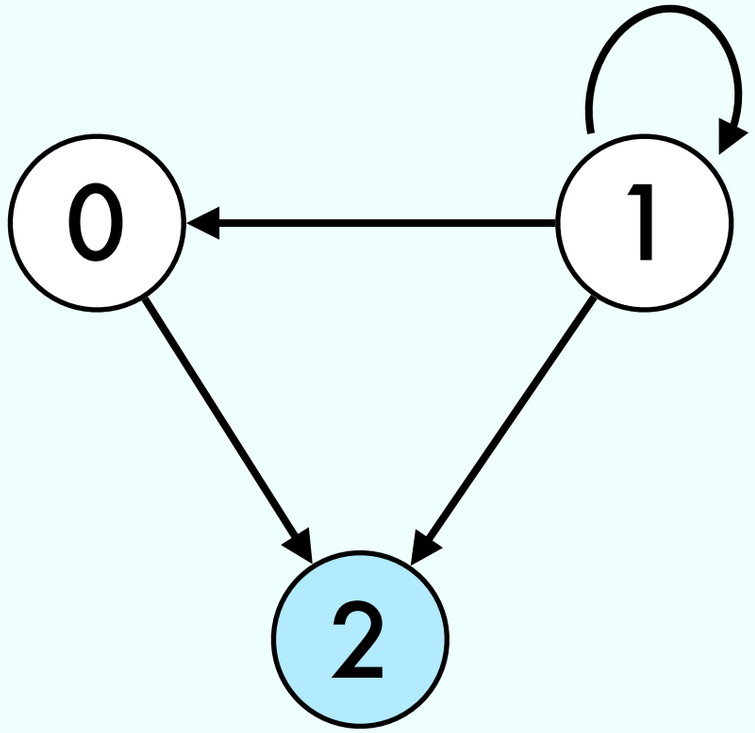


Exercise 9.42

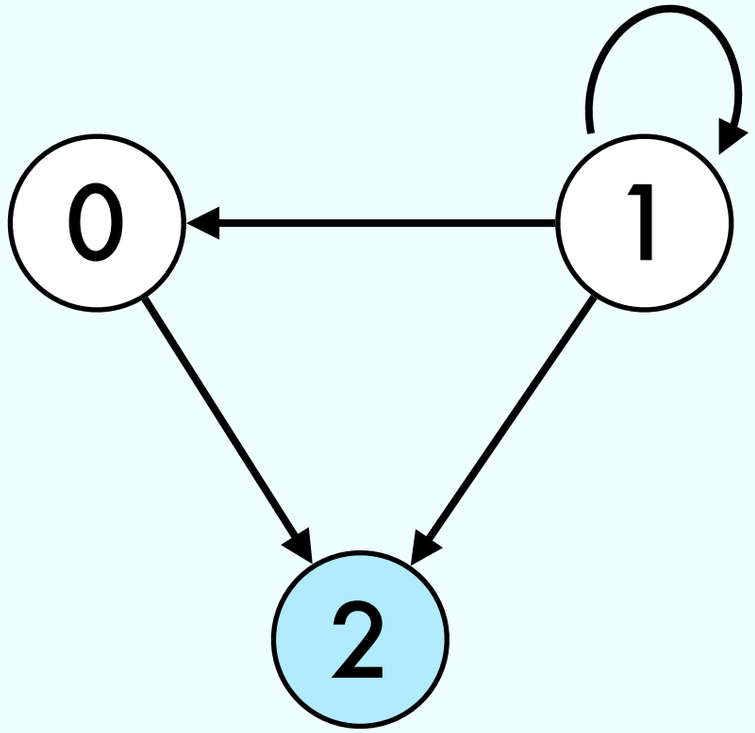
A sink node:

- Has incoming edges from any other node
- Has no outgoing edges

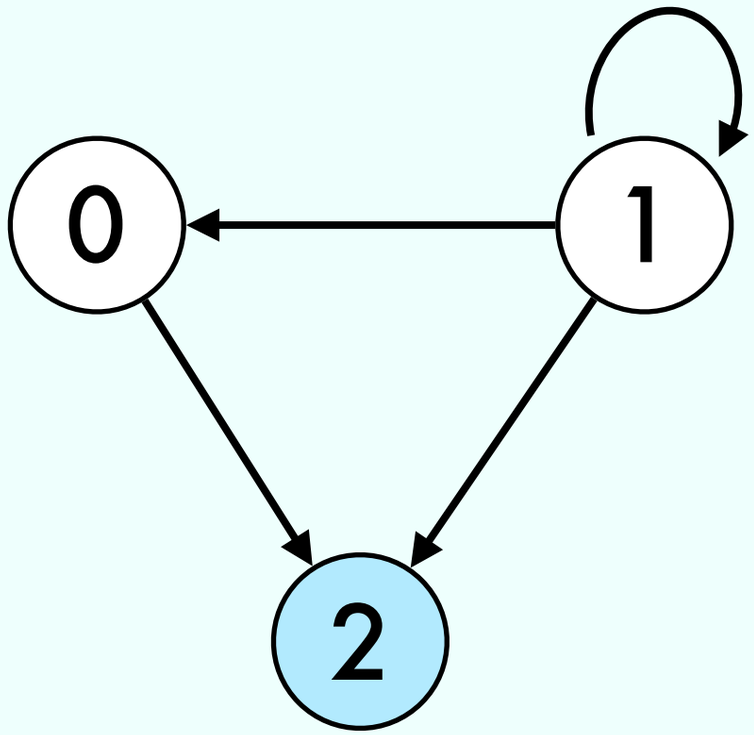




0	0	1
1	1	1
0	0	0



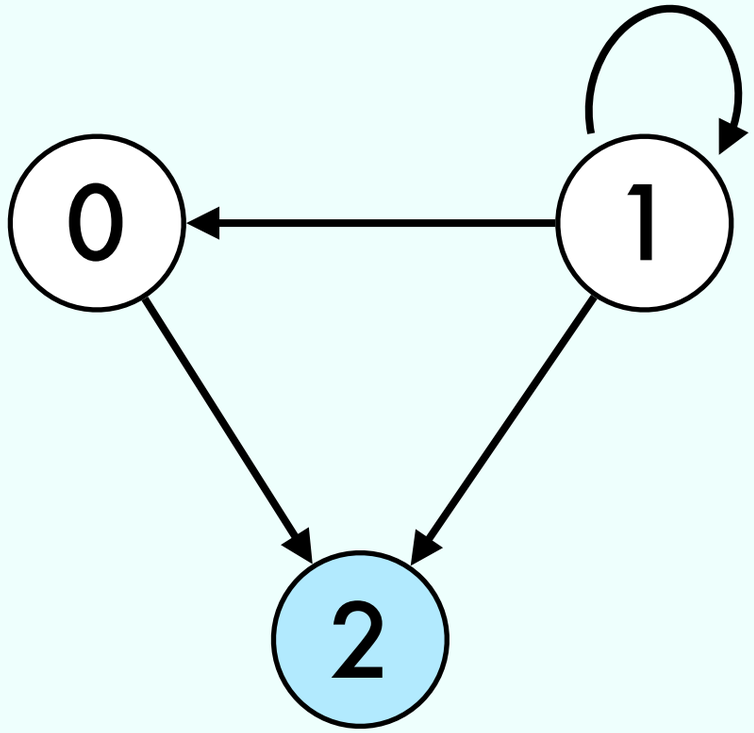
0	0	1
1	1	1
0	0	0



0	0	1
1	1	1
0	0	0

Implement Algorithm

Does a graph have a sink?



0	0	1
1	1	1
0	0	0

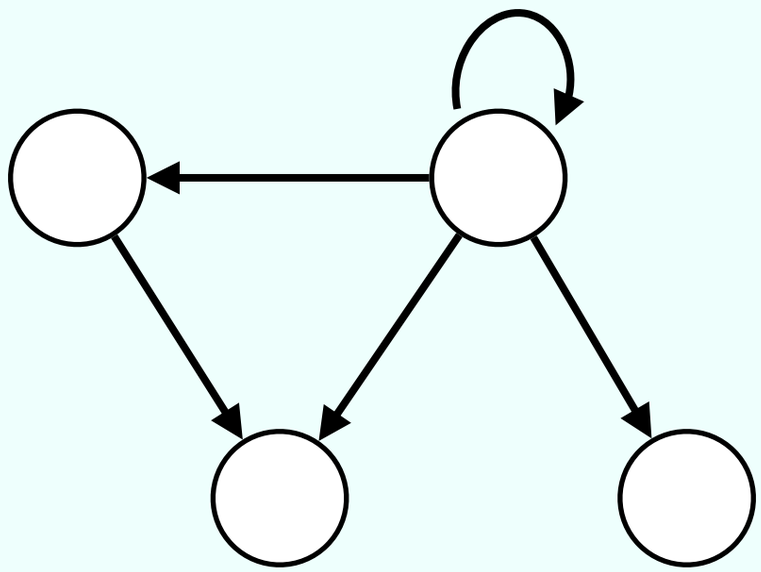
Implement Algorithm
Does a graph have a sink?

O(V)

A sink node:

- Has incoming edges from any other node
- Has no outgoing edges

How many sinks can a graph have?



Sketch Solution

Find Candidate

O(V)

Is Candidate a Sink?

O(V)

Find Candidate

```
row = 0
col = 1

while (col < N)
   if M[row][col]
      row = col
     col = col + 1
   else
      col = col + 1
```

Find Candidate

```
row = 0
col = 1

while (col < N)
  if M[row][col]
    row = col
    col = col + 1
  else
    col = col + 1
```

row is not a sink

Find Candidate

```
row = 0
col = 1

while (col < N)
  if M[row][col]
    row = col
    col = col + 1
  else
    col = col + 1
```

row is not a sink

col is not a sink

Find Candidate

```
row = 0
col = 1

while (col < N)
   if M[row][col]
      row = col
     col = col + 1
   else
      col = col + 1
```

row is not a sink

col is not a sink

row is the candidate

Is candidate row a sink?

```
for i in {0 … N–1}
  if row ≠ i ∧ M[row][i] ∨ ¬ M[i][row]
    return false
return true
```

Is candidate row a sink?

```
for i in {0 … N−1}
   if row ≠ i ∧ M[row][i] ∨ ¬ M[i][row]
      return false
return true
```

Is candidate row a sink?

```
for i in {0 … N–1}
  if row ≠ i ∧ M[row][i] ∨ ¬ M[i][row]
    return false
return true
```

Is candidate row a sink?

```
for i in {0 … N–1}
  if row ≠ i ∧ M[row][i] ∨ ¬ M[i][row]
    return false
return true
```

Is candidate row a sink?

```
for i in {0 … N–1}
  if row ≠ i ∧ M[row][i] ∨ ¬ M[i][row]
    return false
return true
```

Is candidate row a sink?

```
for i in {0 … N–1}
  if row ≠ i ∧ M[row][i] ∨ ¬ M[i][row]
    return false
return true
```

Is candidate row a sink?

```
for i in {0 … N–1}
  if row ≠ i ∧ M[row][i] ∨ ¬ M[i][row]
    return false
return true
```

Exercise 9.51

	SEK	EUR	USD
SEK	1	2	3
EUR	1/2	1	2
USD	1/3	1/2	1

Exercise 9.51

	SEK	EUR	USD
SEK	1	2	3
EUR	1/2	1	2
USD	1/3	1/2	1

300 USD

Exercise 9.51

	SEK	EUR	USD
SEK	1	2	3
EUR	1/2	1	2
USD	1/3	1/2	1

300 USD $\Rightarrow$ 100 SEK

Exercise 9.51

	SEK	EUR	USD
SEK	1	2	3
EUR	1/2	1	2
USD	1/3	1/2	1

300 USD $\Rightarrow$ 100 SEK $\Rightarrow$ 200 EUR

Exercise 9.51

	SEK	EUR	USD
SEK	1	2	3
EUR	1/2	1	2
USD	1/3	1/2	1

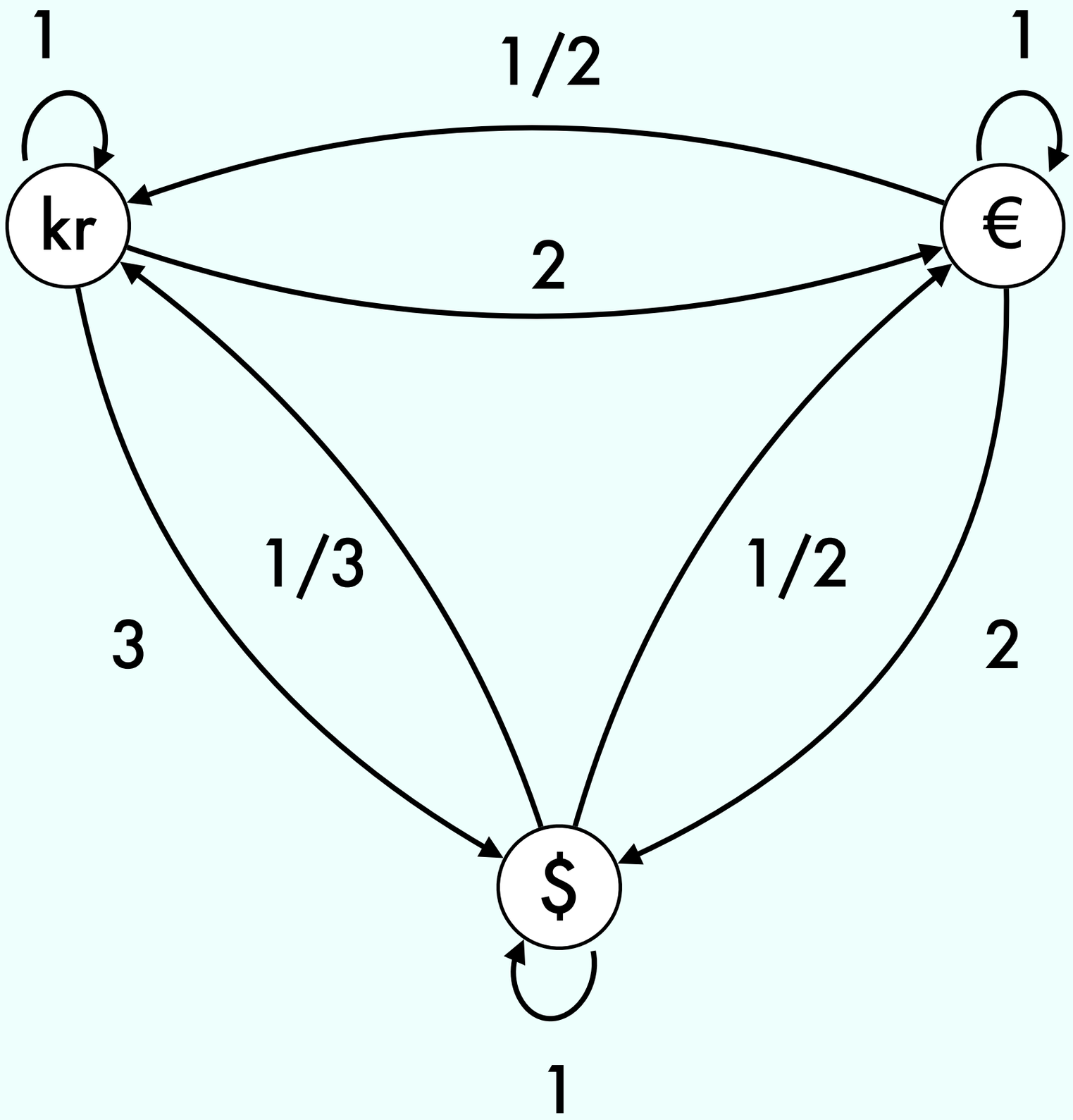
300 USD $\Rightarrow$ 100 SEK $\Rightarrow$ 200 EUR $\Rightarrow$ 400 USD

Exercise 9.51

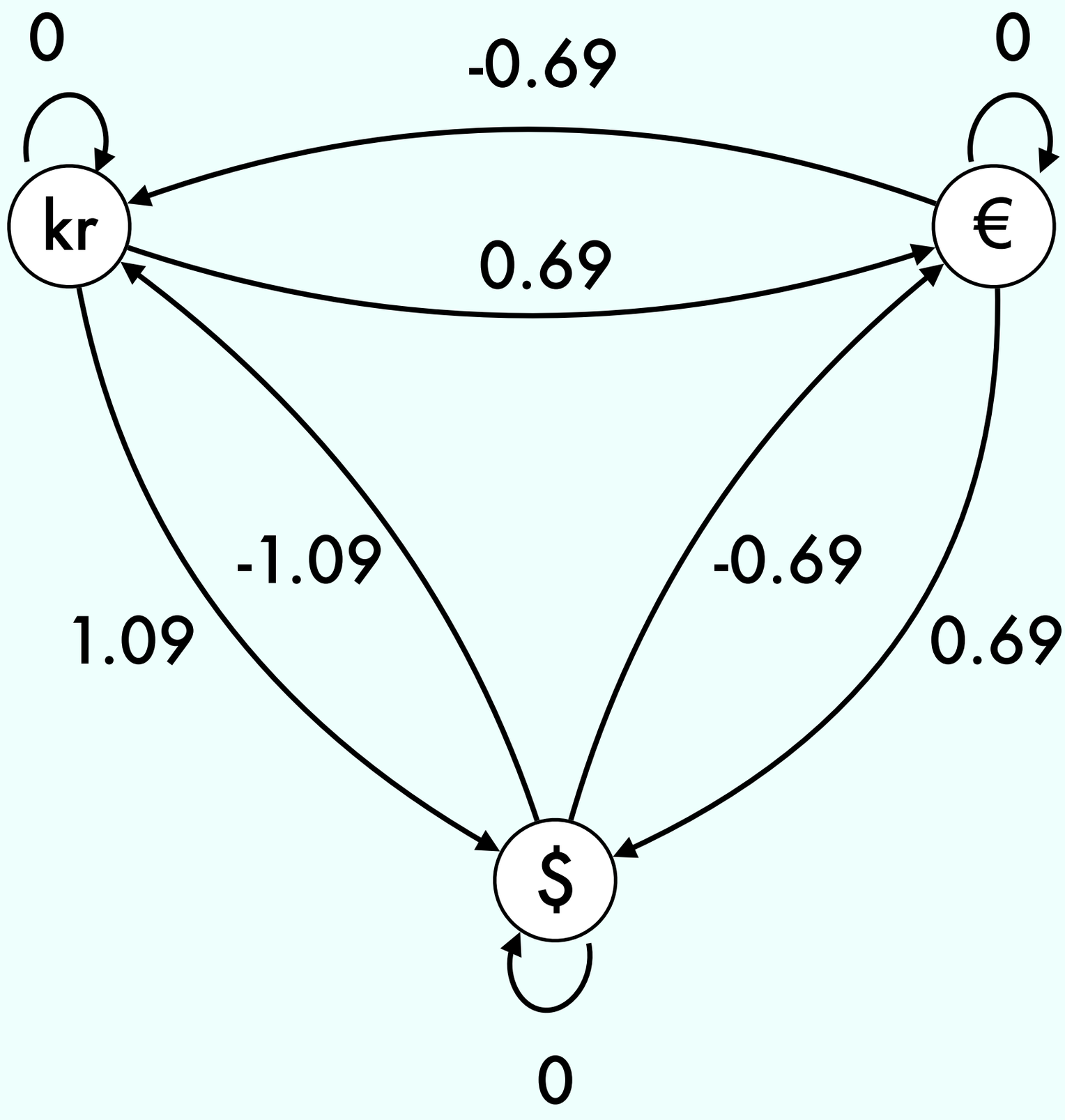
	SEK	EUR	USD
SEK	1	2	3
EUR	1/2	1	2
USD	1/3	1/2	1

300 USD ⇒ 100 SEK ⇒ 200 EUR ⇒ 400 USD

REPEAT for $$$


1
1/2
1
kr
€
2
1/3
1/2
3
2
$
1

Find Shortest Path from kr to kr



Find Shortest Path from kr to kr