COMP36111: Advanced Algorithms I
Lecture 8: Graph-theoretic problems (again)

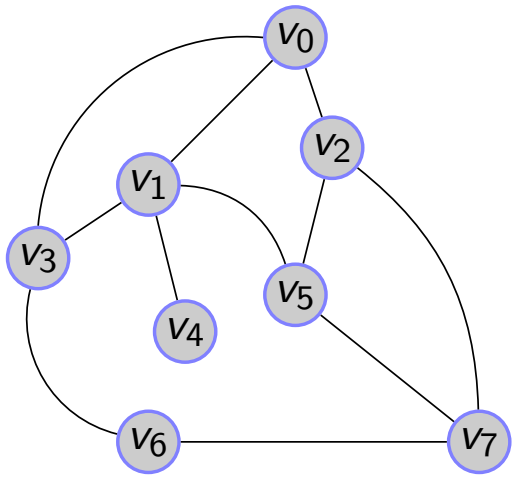
Ian Pratt-Hartmann

Room KB2.38: email: ipratt@cs.man.ac.uk

2017–18

- Reading for this lecture:
  - Sipser: Chapter 7.

- A *graph* is a pair $G = (V, E)$, where V is a finite set and E a set of unordered pairs of (distinct) elements of V.
- Graphs are typically displayed–you guessed it—graphically:

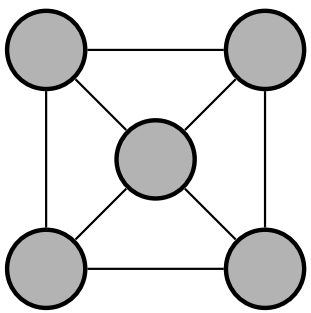


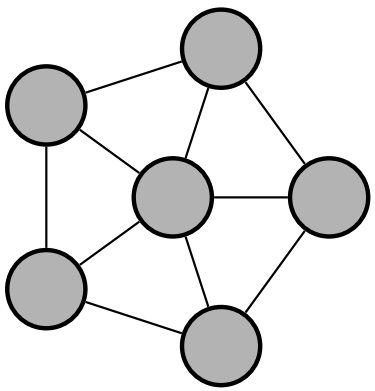
- Graphs can be encoded as strings over finite alphabets. For instance:

$$(n; (u_1, v_1), \ldots (u_m, v_m)),$$

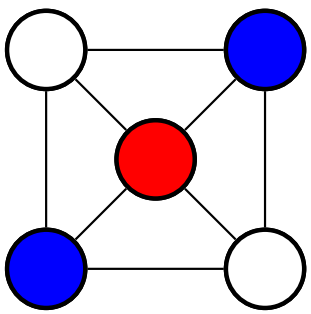
where the u_i and v_i are integers in the range $[0, n-1]$, encodes the obvious graph over the vertices $\{0, \ldots n-1\}$.

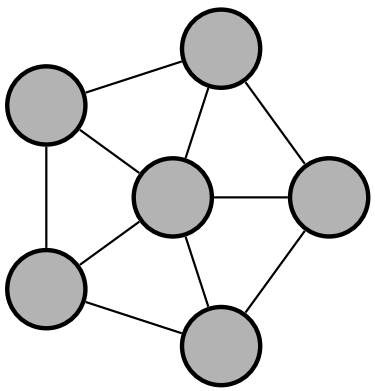
- Let $G = (V, E)$ be a graph. A k-colouring of G is a function $f : V \to \{0, 1, \ldots, k-1\}$ such that, for any edge $(u, v) \in E$, $f(u) \neq f(v)$.
- We say that G is k-colourable if there exists a k-colouring for G.
- Of the two graphs below, one is 3-colourable and the other not. Can you tell which?



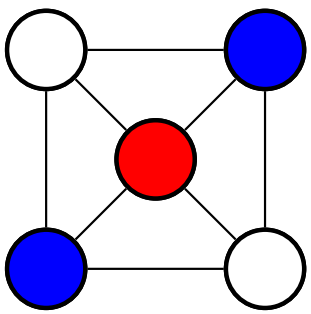


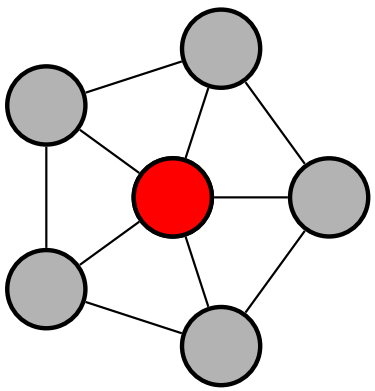
- Let $G = (V, E)$ be a graph. A k-colouring of G is a function $f : V \to \{0, 1, \ldots, k-1\}$ such that, for any edge $(u, v) \in E$, $f(u) \neq f(v)$.
- We say that G is k-colourable if there exists a k-colouring for G.
- Of the two graphs below, one is 3-colourable and the other not. Can you tell which?



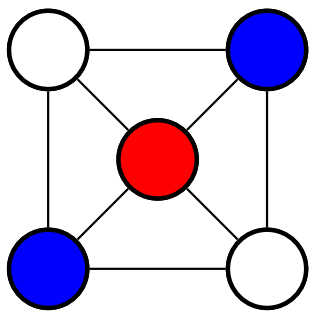


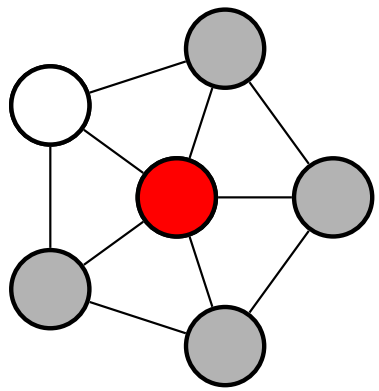
- Let $G = (V, E)$ be a graph. A k-colouring of G is a function $f : V \to \{0, 1, \ldots, k-1\}$ such that, for any edge $(u, v) \in E$, $f(u) \neq f(v)$.
- We say that G is k-colourable if there exists a k-colouring for G.
- Of the two graphs below, one is 3-colourable and the other not. Can you tell which?



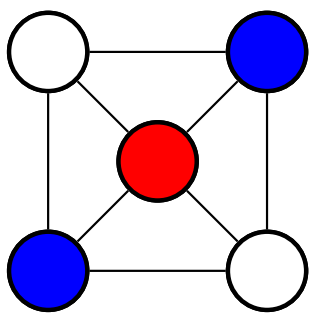


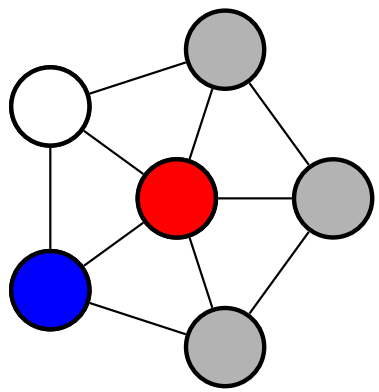
- Let $G = (V, E)$ be a graph. A k-colouring of G is a function $f : V \to \{0, 1, \ldots, k-1\}$ such that, for any edge $(u, v) \in E$, $f(u) \neq f(v)$.
- We say that G is k-colourable if there exists a k-colouring for G.
- Of the two graphs below, one is 3-colourable and the other not. Can you tell which?



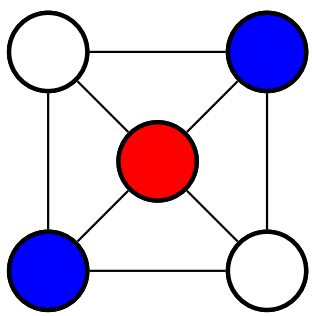


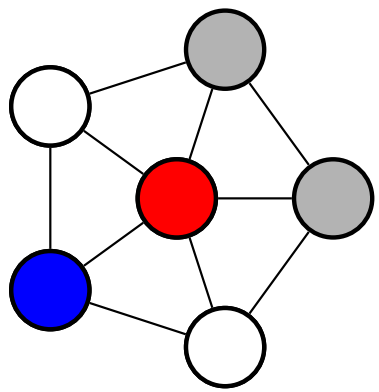
- Let $G = (V, E)$ be a graph. A k-colouring of G is a function $f : V \to \{0, 1, \ldots, k-1\}$ such that, for any edge $(u, v) \in E$, $f(u) \neq f(v)$.
- We say that G is k-colourable if there exists a k-colouring for G.
- Of the two graphs below, one is 3-colourable and the other not. Can you tell which?



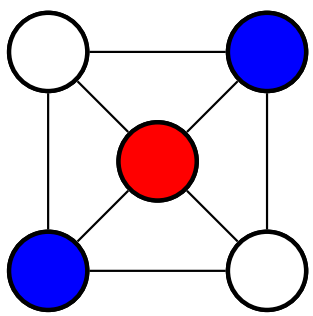


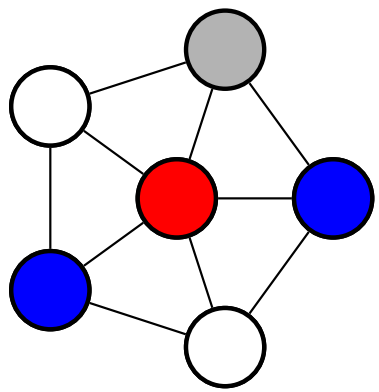
- Let $G = (V, E)$ be a graph. A k-colouring of G is a function $f : V \to \{0, 1, \ldots, k-1\}$ such that, for any edge $(u, v) \in E$, $f(u) \neq f(v)$.
- We say that G is k-colourable if there exists a k-colouring for G.
- Of the two graphs below, one is 3-colourable and the other not. Can you tell which?





- Let $G = (V, E)$ be a graph. A k-colouring of G is a function $f : V \to \{0, 1, \ldots, k-1\}$ such that, for any edge $(u, v) \in E$, $f(u) \neq f(v)$.
- We say that G is k-colourable if there exists a k-colouring for G.
- Of the two graphs below, one is 3-colourable and the other not. Can you tell which?



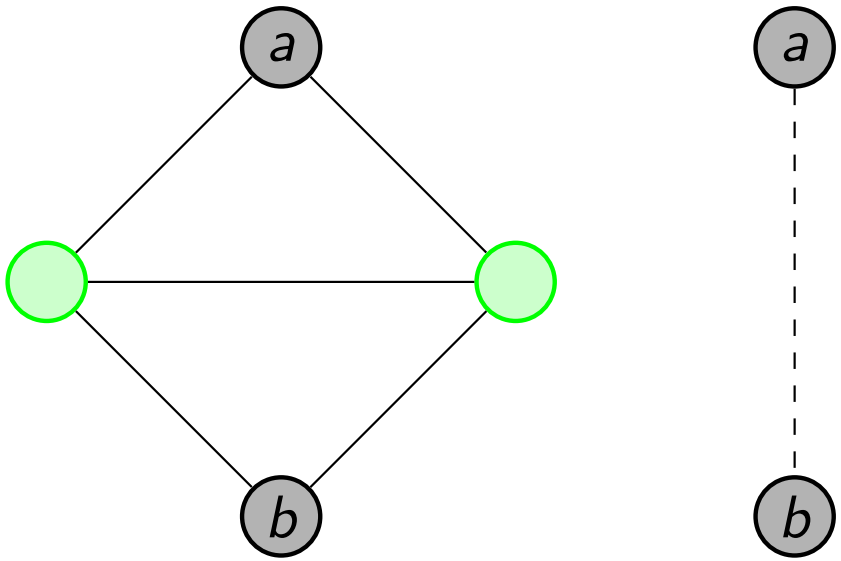


- This gives us a natural problem:

  k-COLOURABILITY
  Given: A graph G.
  Return: Yes if G is k-colourable, and No otherwise.

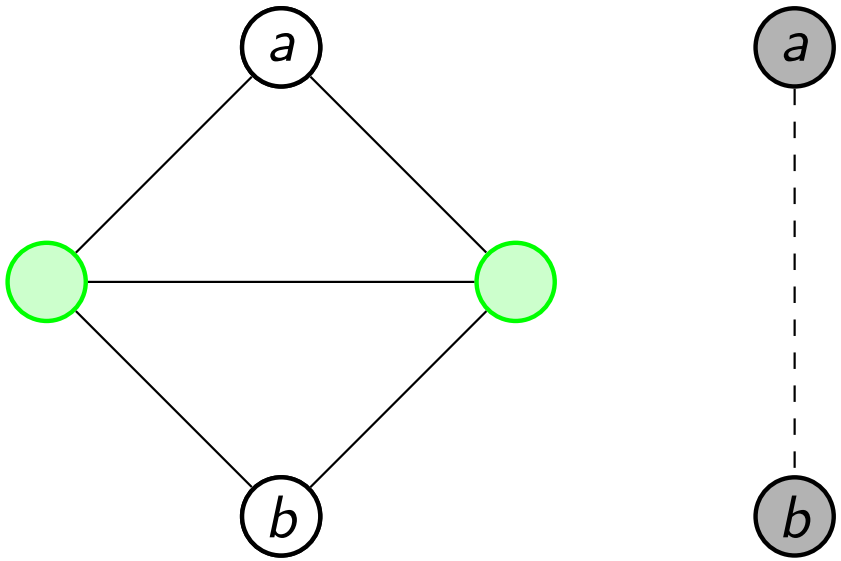
- We are now going to show that 3-colourability is NPTIME-hard.
- We proceed by reduction from 3-Sat: from a set of 3-literal clauses Γ, we compute (in logarithmic space) a graph G_Γ and show that Γ is satisfiable iff G_Γ is 3-colourable.
- We build the graph using a series of gadgets...

- Here is the first gadget:



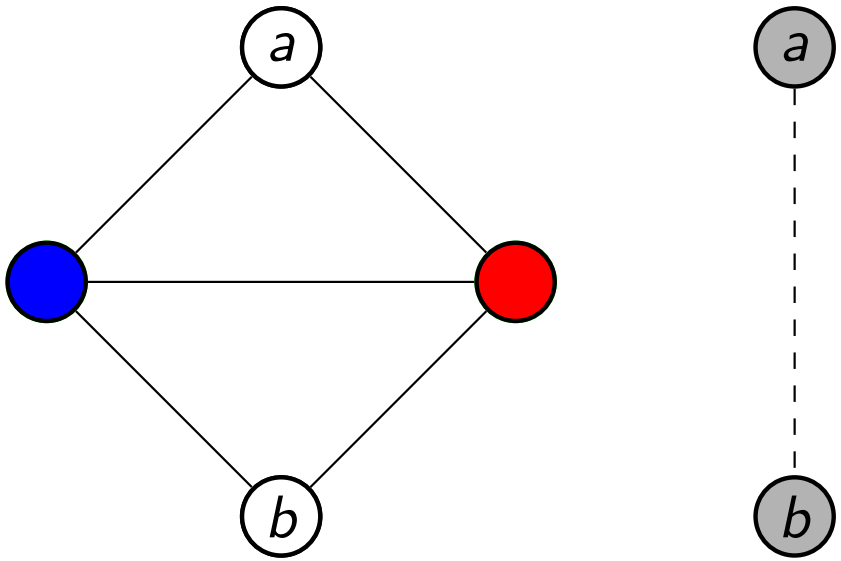
- The green nodes are internal to the gadget: the grey nodes will interface to other nodes in the graph.
- It is obvious that, in any colouring, a and b will have the same colour.
- Conversely, if a and b have the same colour, we can always extend to a three-colouring of the whole gadget.

- Here is the first gadget:



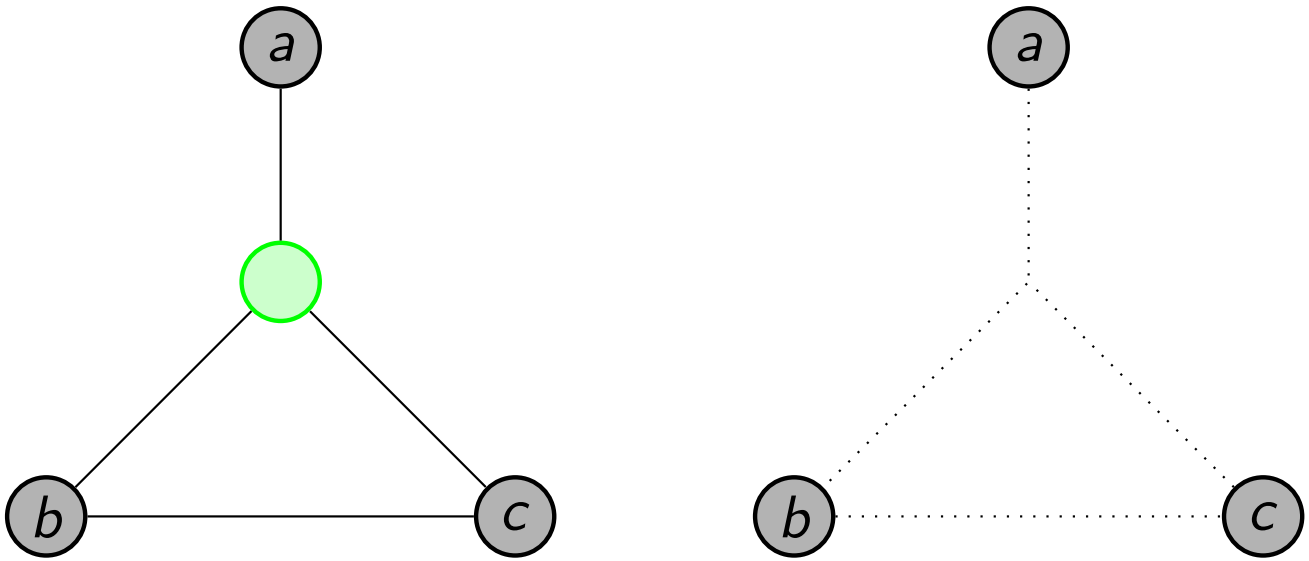
- The green nodes are internal to the gadget: the grey nodes will interface to other nodes in the graph.
- It is obvious that, in any colouring, a and b will have the same colour.
- Conversely, if a and b have the same colour, we can always extend to a three-colouring of the whole gadget.

- Here is the first gadget:



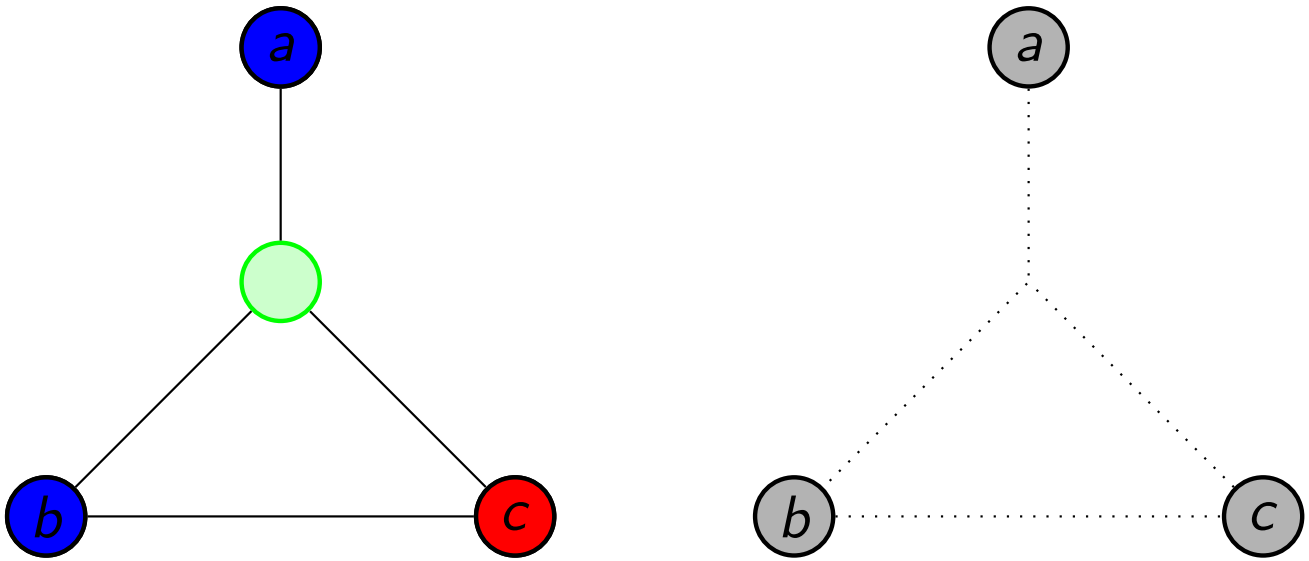
- The green nodes are internal to the gadget: the grey nodes will interface to other nodes in the graph.
- It is obvious that, in any colouring, a and b will have the same colour.
- Conversely, if a and b have the same colour, we can always extend to a three-colouring of the whole gadget.

- Here is the second gadget:



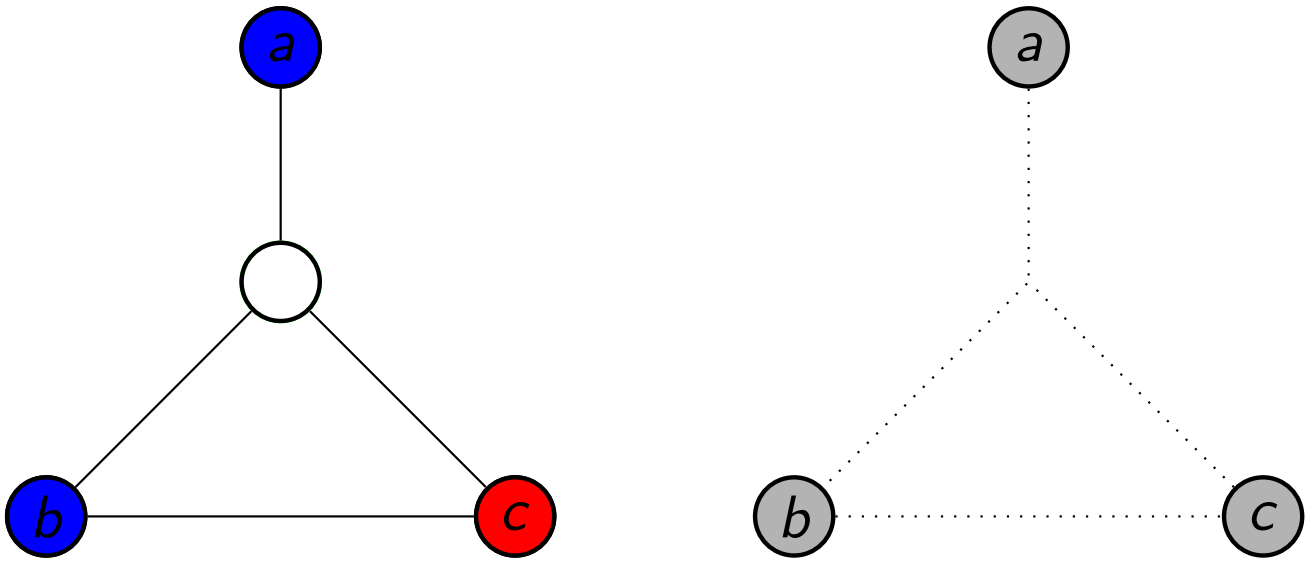
- The green nodes are internal to the gadget: the grey nodes will interface to other nodes in the graph.
- It is obvious that, in any colouring, a will have the same colour as exactly one of b and c.
- Moreover, if a has the same colour as exactly one of b and c, we can extend to a three-colouring.

- Here is the second gadget:



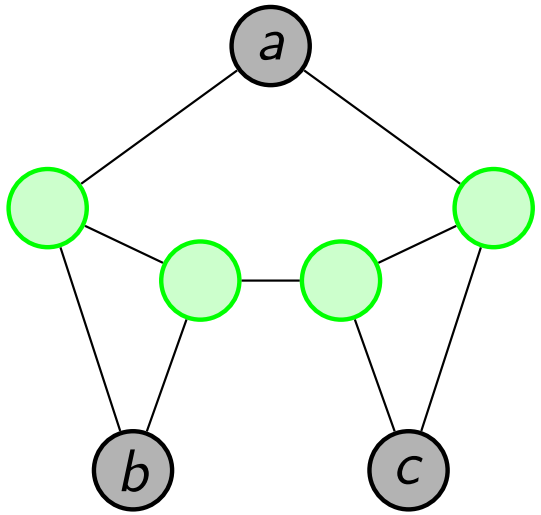
- The green nodes are internal to the gadget: the grey nodes will interface to other nodes in the graph.
- It is obvious that, in any colouring, a will have the same colour as exactly one of b and c.
- Moreover, if a has the same colour as exactly one of b and c, we can extend to a three-colouring.

- Here is the second gadget:



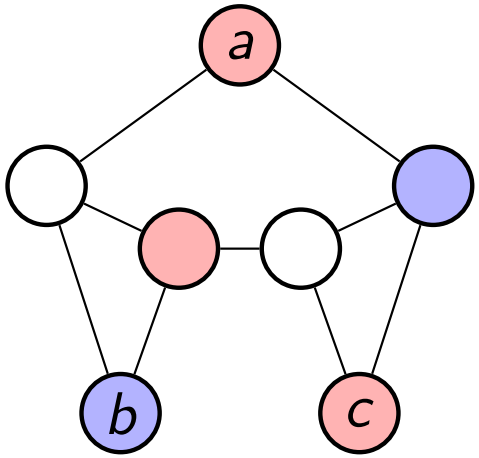
- The green nodes are internal to the gadget: the grey nodes will interface to other nodes in the graph.
- It is obvious that, in any colouring, a will have the same colour as exactly one of b and c.
- Moreover, if a has the same colour as exactly one of b and c, we can extend to a three-colouring.

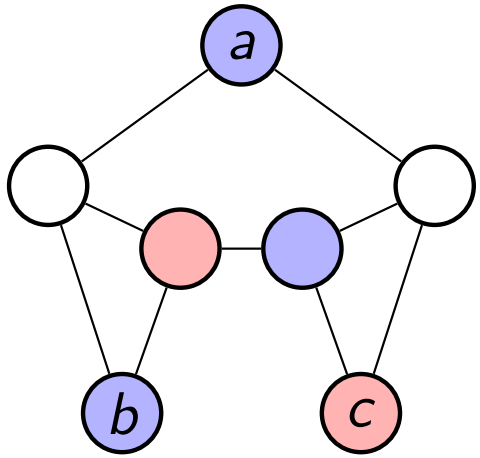
- Here is the third gadget:



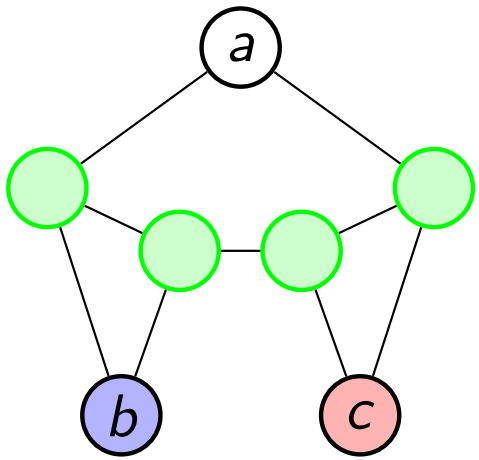
- The green nodes are internal to the gadget: the grey nodes will interface to other nodes in the graph.

- Suppose b and c are differently coloured (blue and red)
- Then a can be given either of these colours:

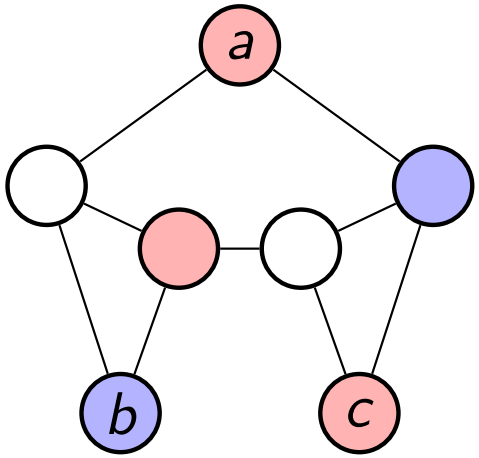


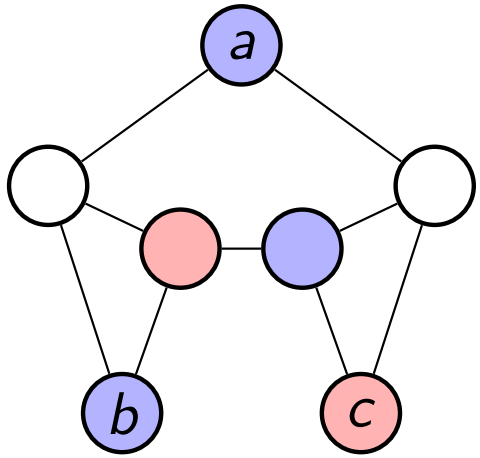


- On the other hand, if a has a different colour, we get stuck:

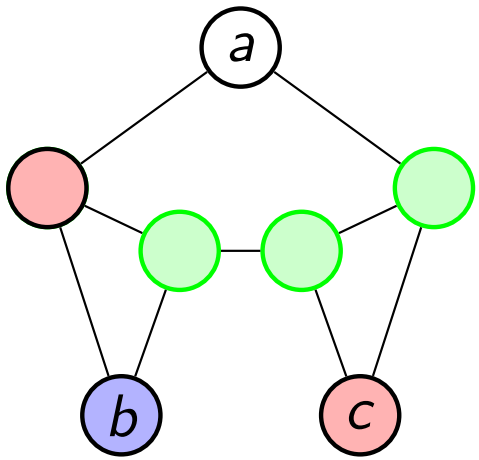


- Suppose b and c are differently coloured (blue and red)
- Then a can be given either of these colours:

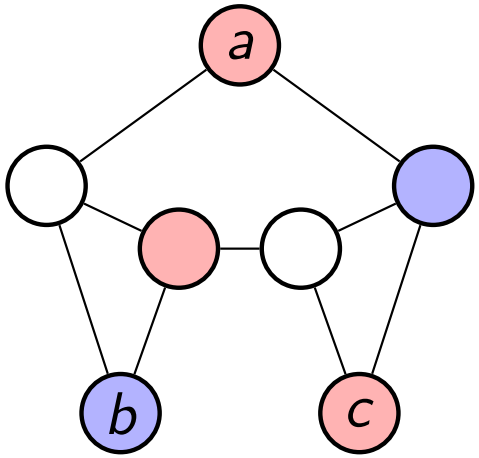


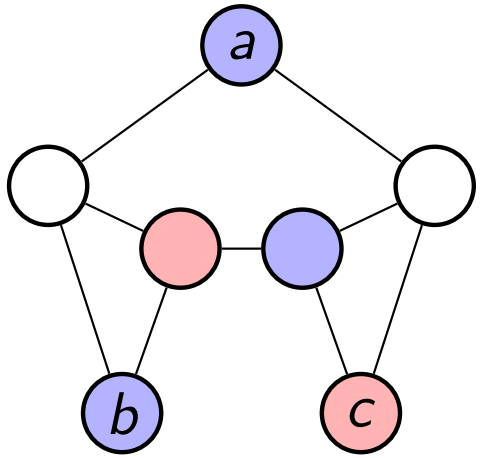


- On the other hand, if a has a different colour, we get stuck:

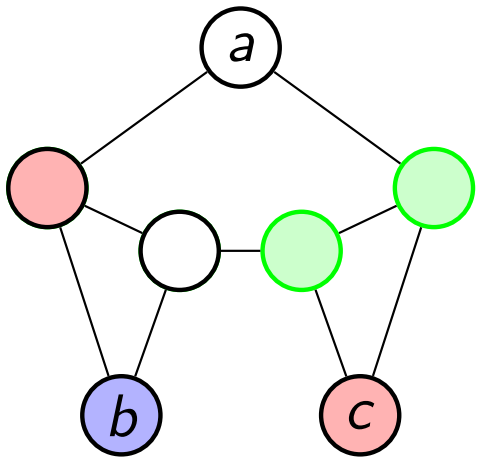


- Suppose b and c are differently coloured (blue and red)
- Then a can be given either of these colours:

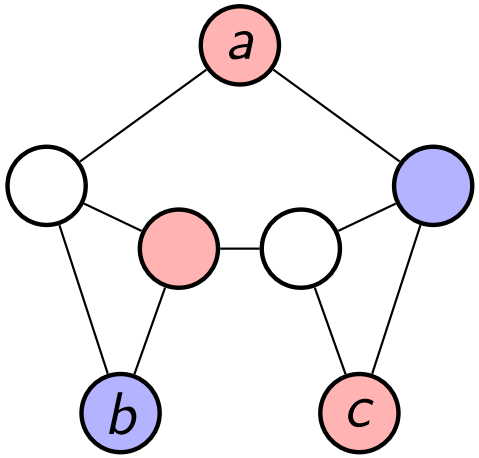


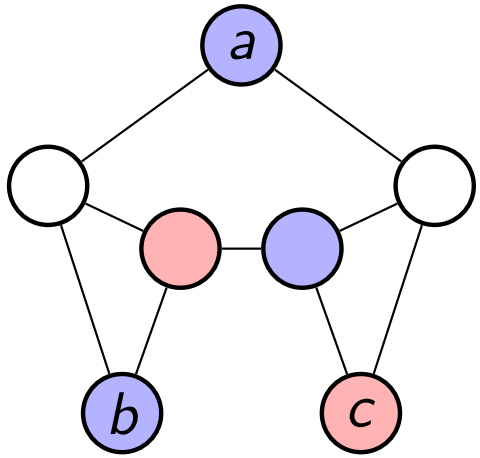


- On the other hand, if a has a different colour, we get stuck:

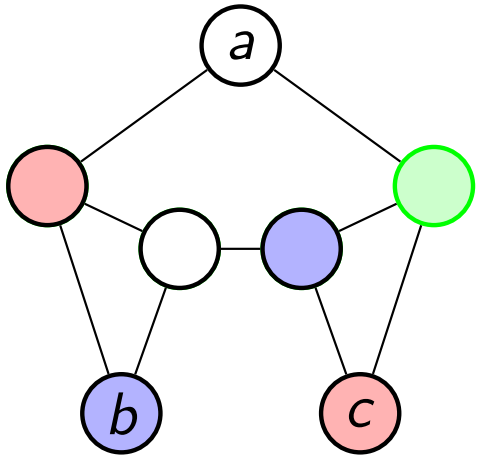


- Suppose b and c are differently coloured (blue and red)
- Then a can be given either of these colours:

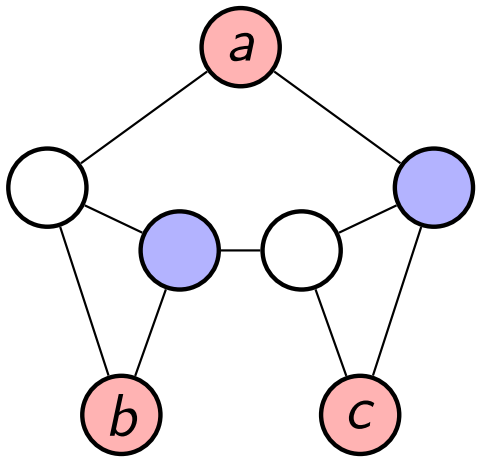




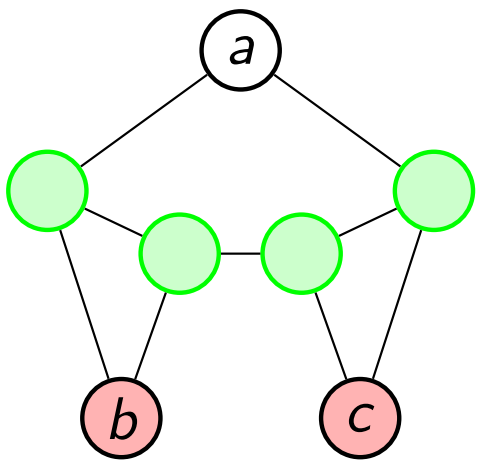
- On the other hand, if a has a different colour, we get stuck:



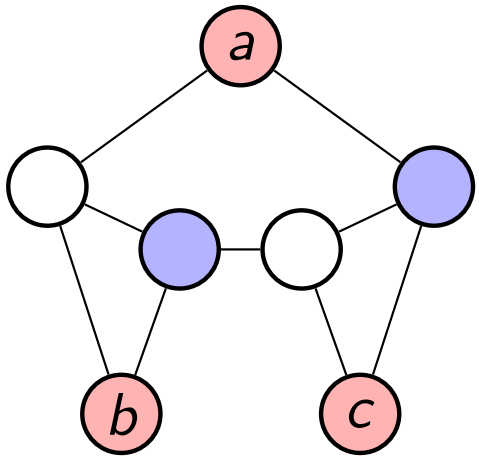
- Suppose b and c have the same colour (red)
- Then a can be given this colour too:



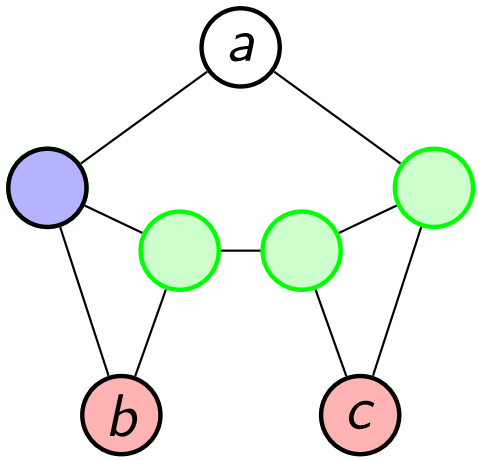
- On the other hand, if a has a different colour (white), we get stuck:



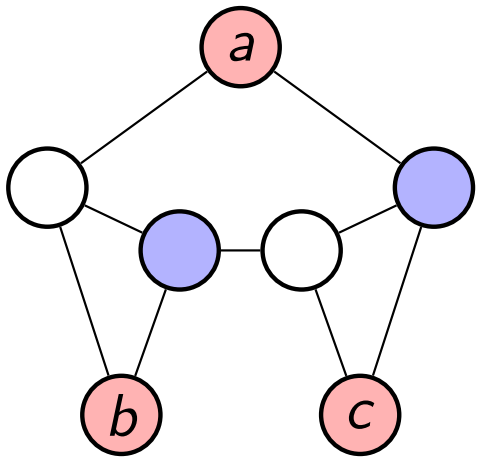
- Suppose b and c have the same colour (red)
- Then a can be given this colour too:



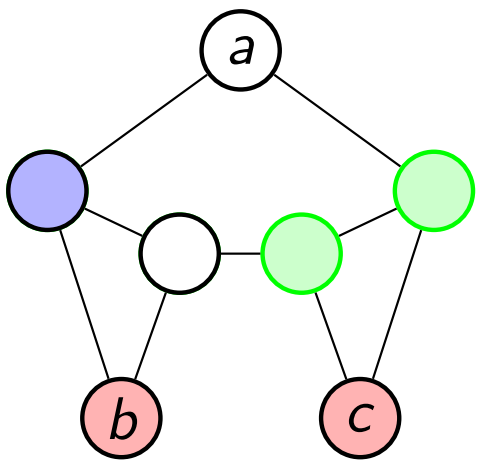
- On the other hand, if a has a different colour (white), we get stuck:



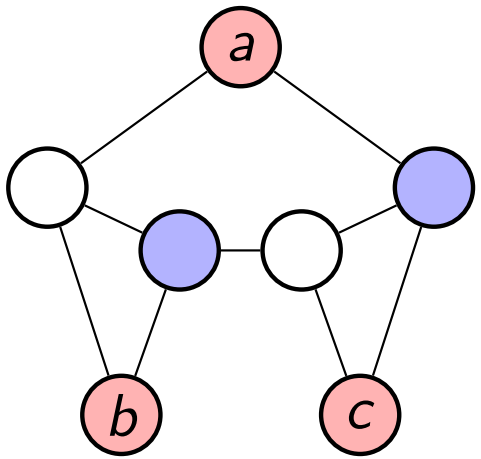
- Suppose b and c have the same colour (red)
- Then a can be given this colour too:



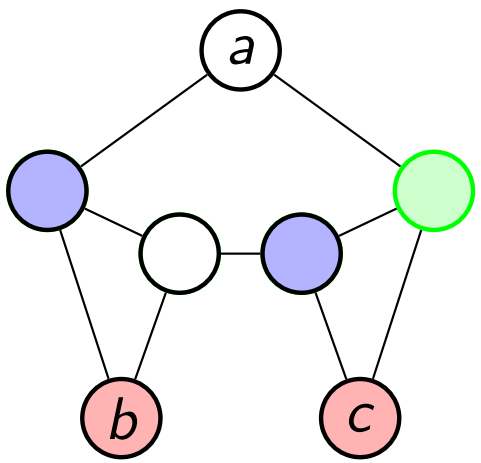
- On the other hand, if a has a different colour (white), we get stuck:



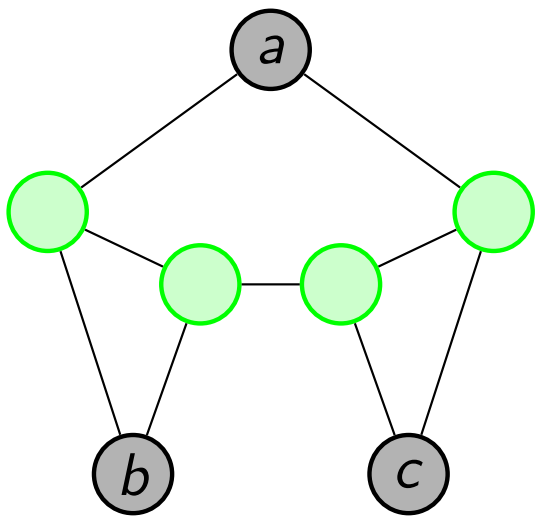
- Suppose b and c have the same colour (red)
- Then a can be given this colour too:

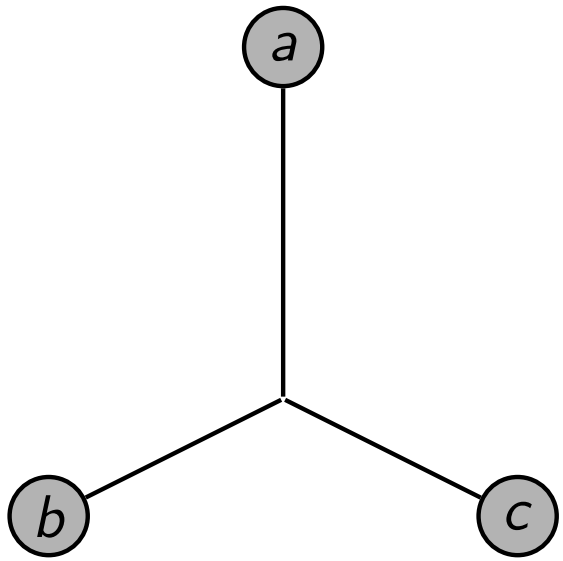


- On the other hand, if a has a different colour (white), we get stuck:

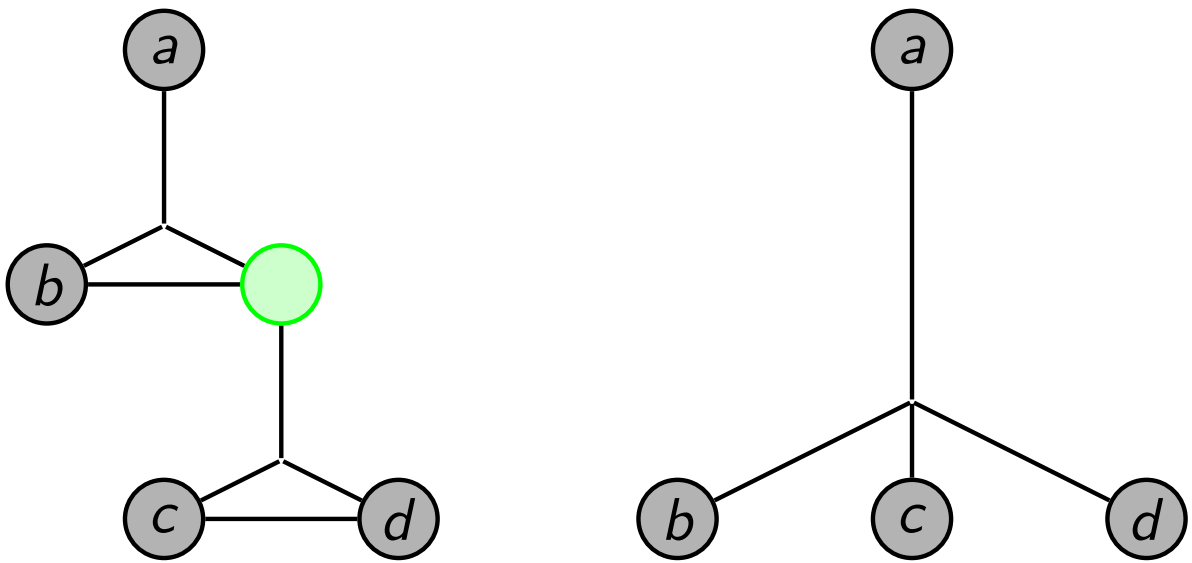


- Thus, in this gadget, node a has the same values as either node b or node c (or both)





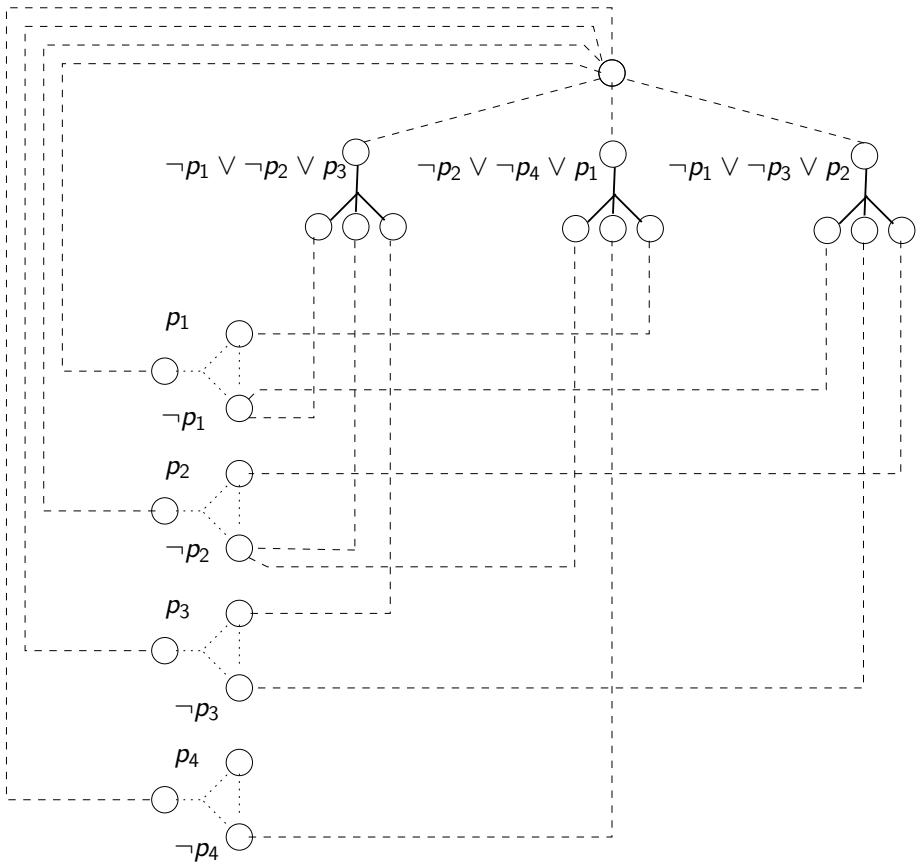
- We can stick two of these gadgets together to make a fourth gadget, in which a has to have the same colour as any of b, c or d (with all such colourings possible)



Theorem

The problem 3-COLOURABILITY is NPTIME*-hard.*

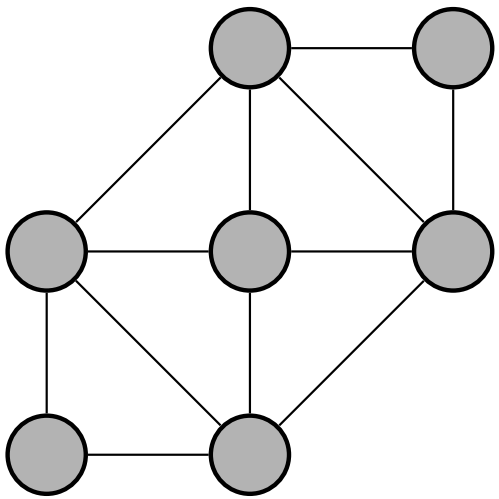
Proof.

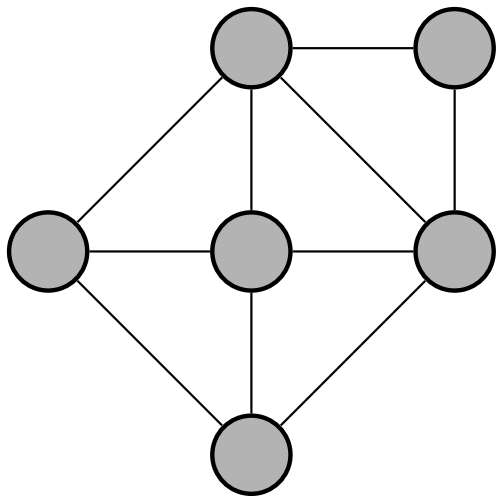


Encoding of $\{(\neg p_1 \vee \neg p_2 \vee p_3), (\neg p_2 \vee p_4 \vee p_1), (\neg p_1 \vee \neg p_3 \vee p_2)\}$

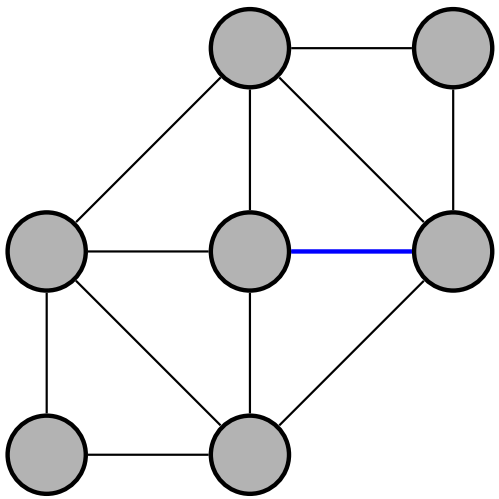
- Let $G = (V, E)$ be a connected graph.
  - A *circuit* of G is a sequence of nodes $v_0, \ldots v_{n-1}$ such that, for all i $(0 \leq i < n)$ $(v_i, v_{(i+1) \mod n}) \in E$.
  - An *Eulerian circuit* of G is a circuit of G in which each edge of G is traversed exactly once.
  - A *Hamiltonian circuit* of G is a circuit of G in which each node is encountered exactly once.

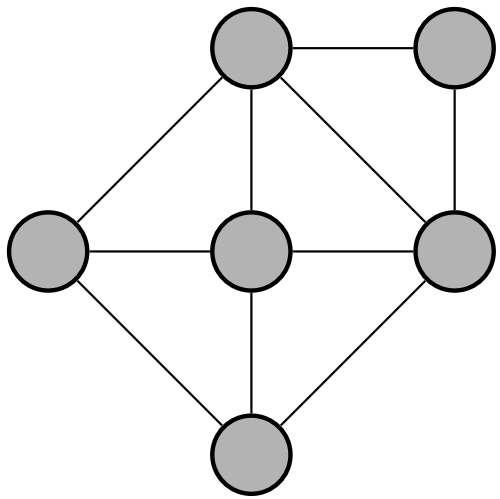
- Exactly one of the following graphs has an Eulerian circuit:



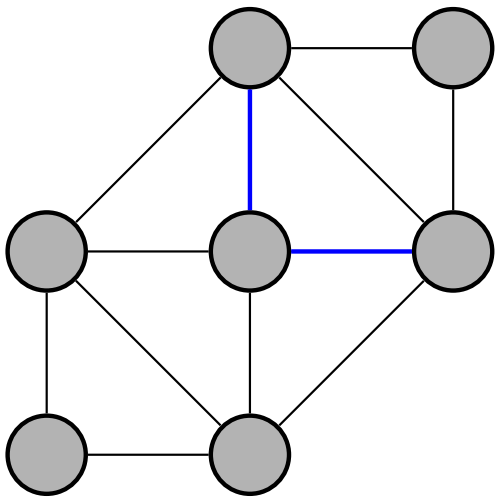


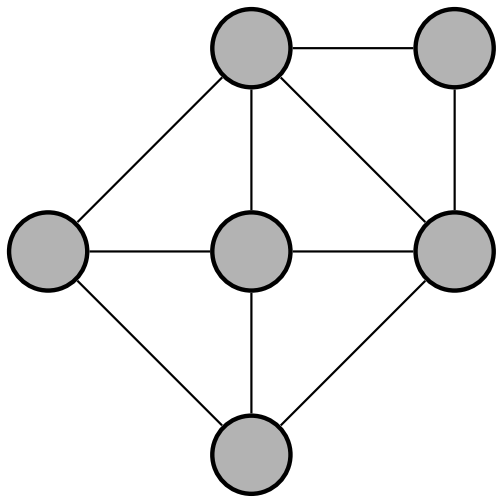
- Exactly one of the following graphs has an Eulerian circuit:



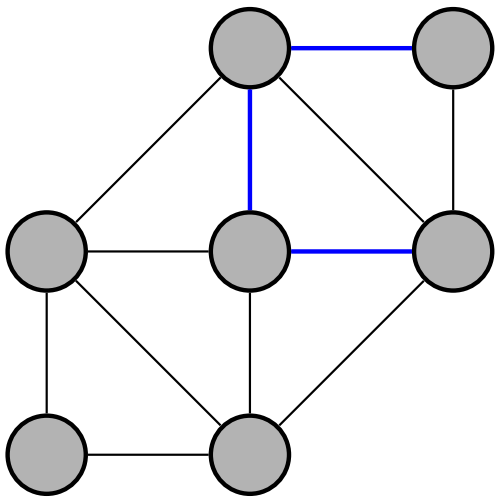


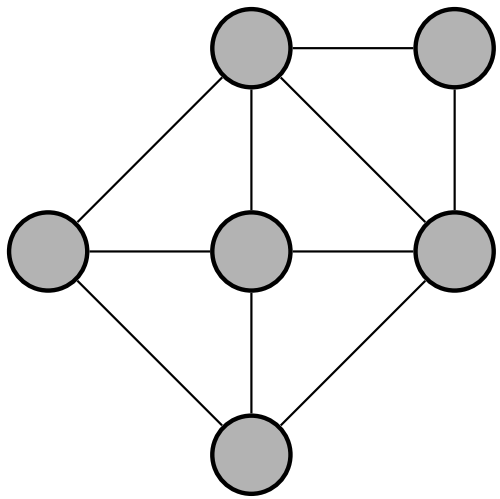
- Exactly one of the following graphs has an Eulerian circuit:



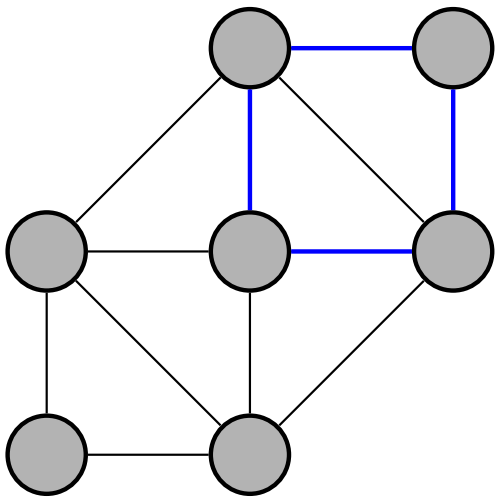


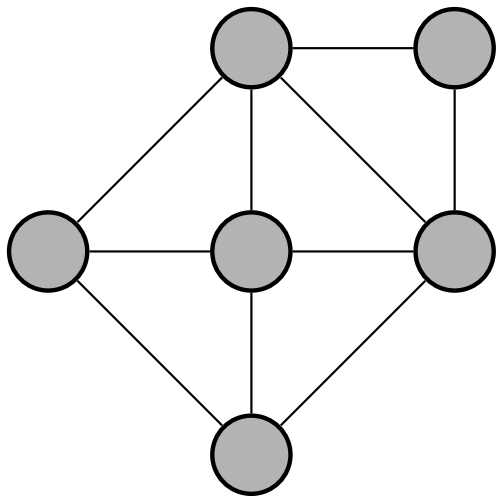
- Exactly one of the following graphs has an Eulerian circuit:



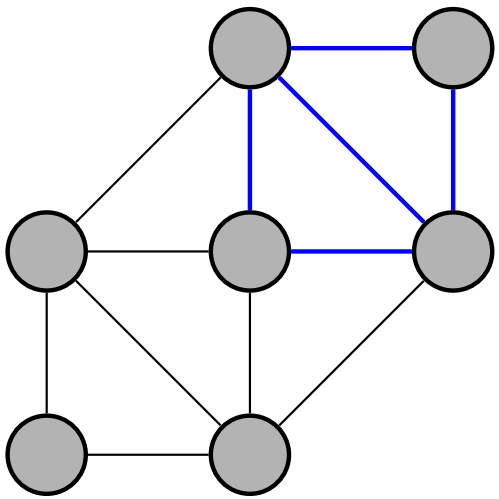


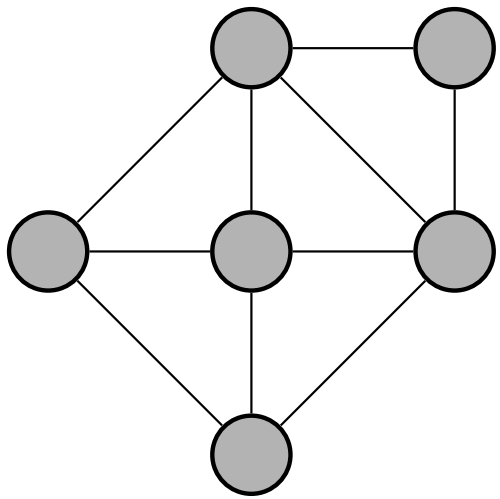
- Exactly one of the following graphs has an Eulerian circuit:



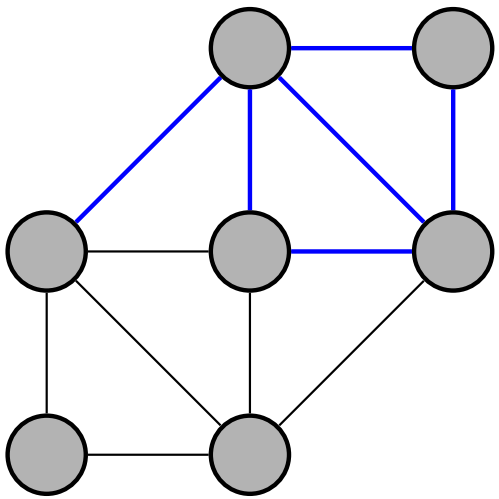


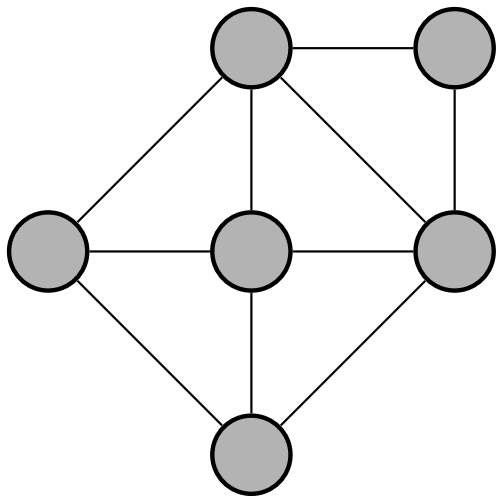
- Exactly one of the following graphs has an Eulerian circuit:



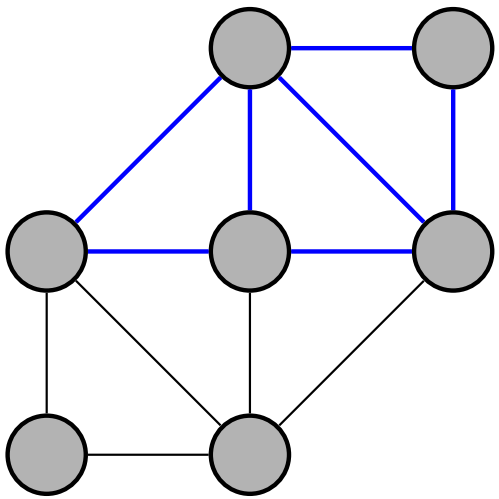


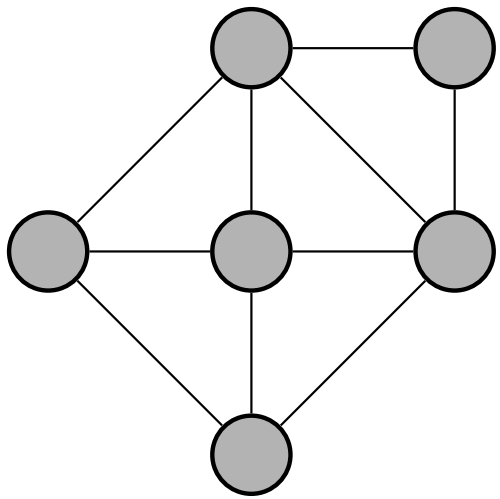
- Exactly one of the following graphs has an Eulerian circuit:



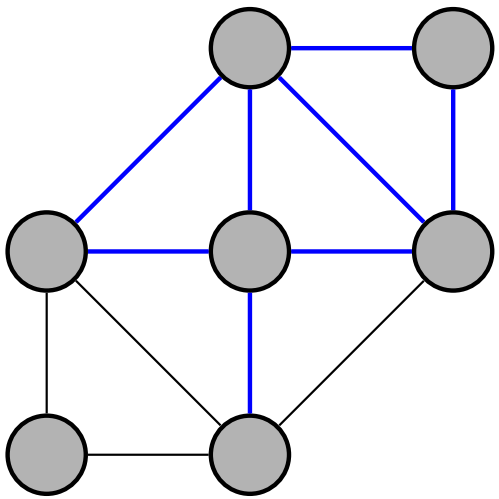


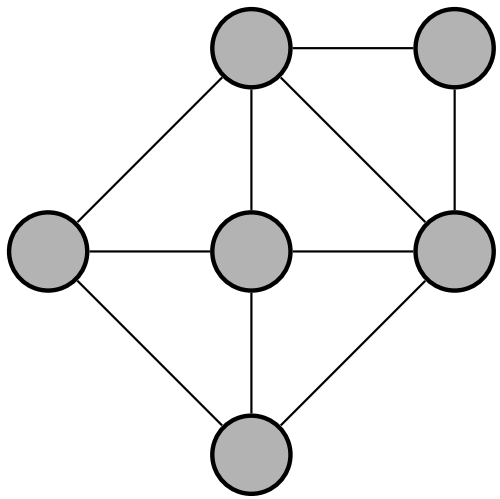
- Exactly one of the following graphs has an Eulerian circuit:



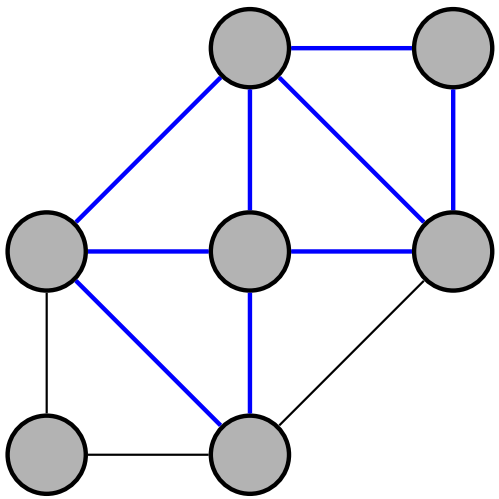


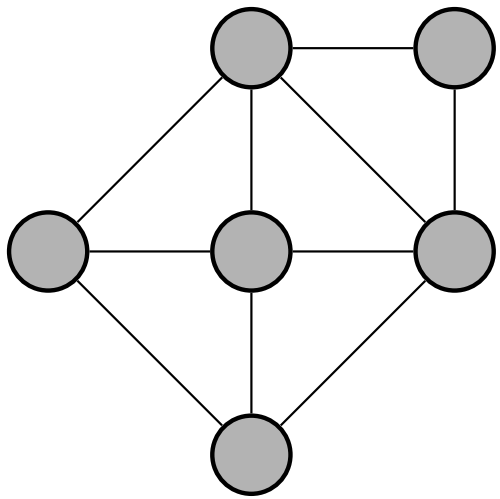
- Exactly one of the following graphs has an Eulerian circuit:



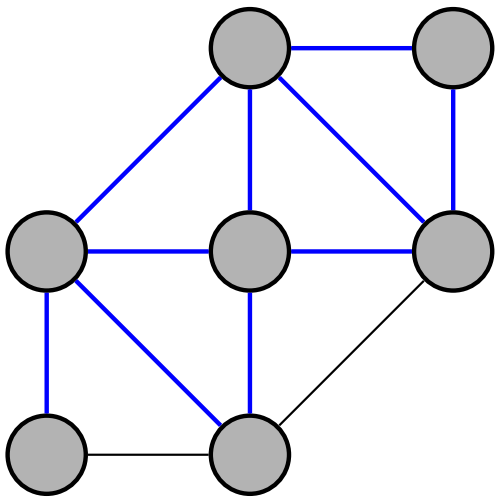


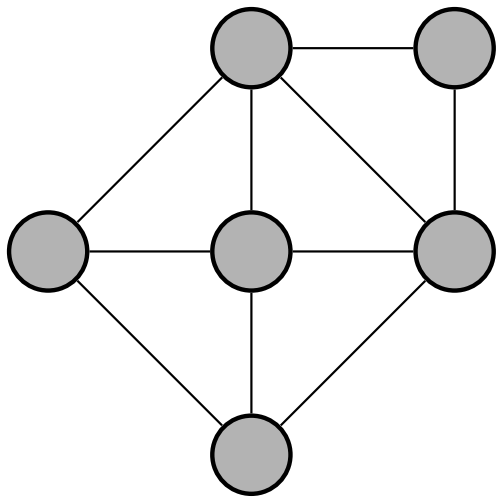
- Exactly one of the following graphs has an Eulerian circuit:



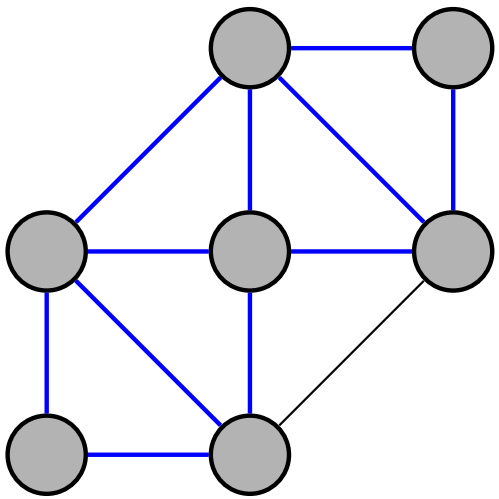


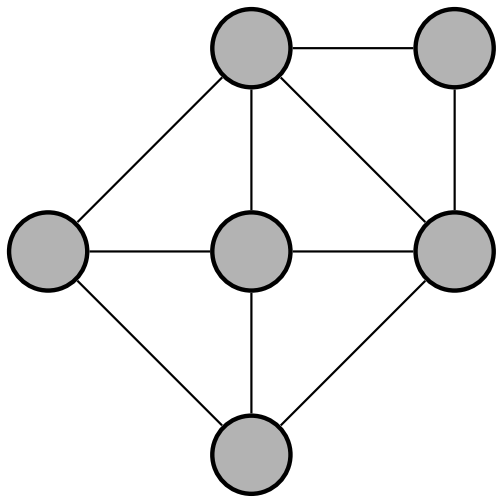
- Exactly one of the following graphs has an Eulerian circuit:



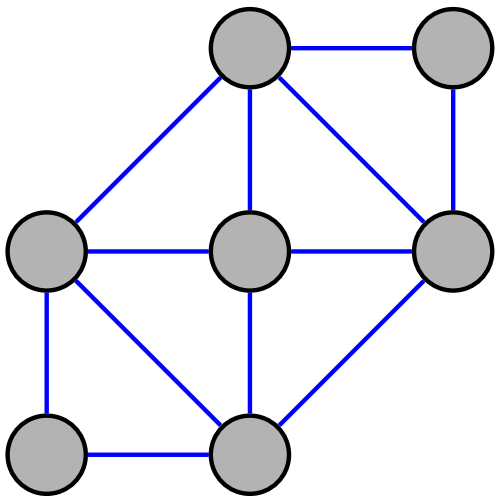


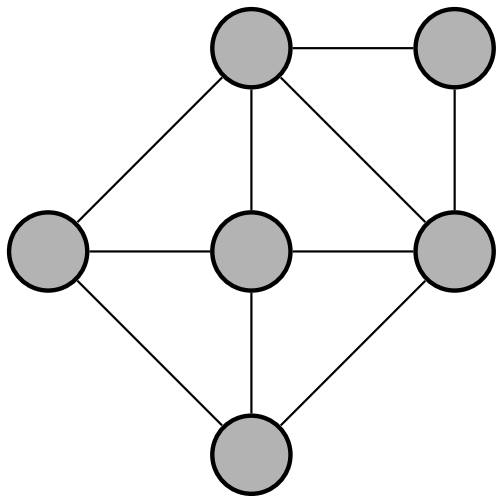
- Exactly one of the following graphs has an Eulerian circuit:



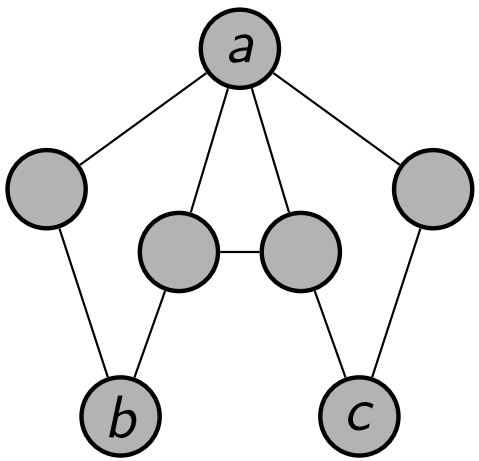


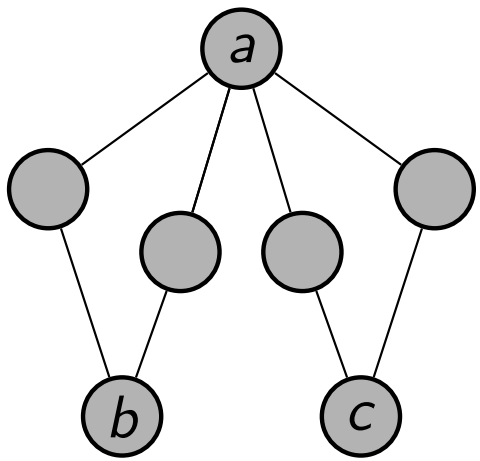
- Exactly one of the following graphs has an Eulerian circuit:



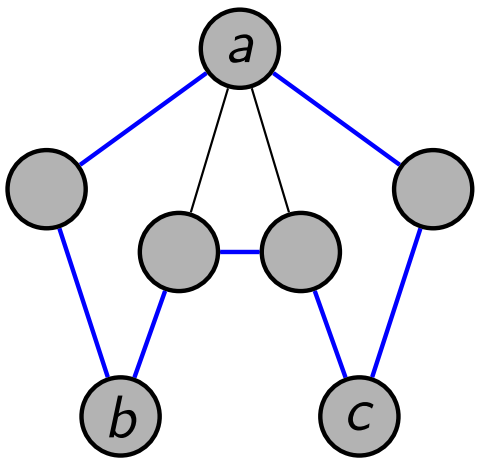


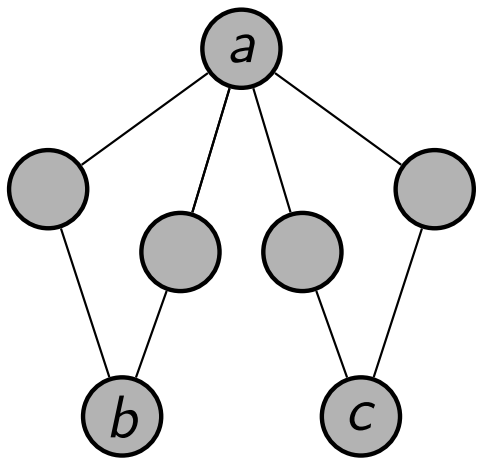
- Exactly one of the following graphs has an Hamiltonian circuit:





- Exactly one of the following graphs has an Hamiltonian circuit:





- We thus have the problems:

EULERIAN-CIRCUIT
Given: A Graph G
Return: Yes if G has an Eulerian circuit, and No otherwise.

HAMILTONIAN-CIRCUIT
Given: A Graph G
Return: Yes if G has an Hamiltonian circuit, and No otherwise.

Theorem (Euler)

A connected graph has an Eulerian circuit if and only if every node has even degree.

Proof.

The only-if-direction is obvious. For the if-direction, let

$$v_0, \ldots, v_m$$

be the longest walk in G in which no edge is traversed more than once. Since every node has even degree, $v_m = v_0$. Suppose $e \in E$ is not in this walk. By connectedness of G let e be (v_i, u), where $0 \leq i < m$. Then

$$v_i, \ldots, v_m (= v_0) v_1, \ldots, v_i u$$

is a longer walk in G in which no edge is traversed more than once. Contradiction. □

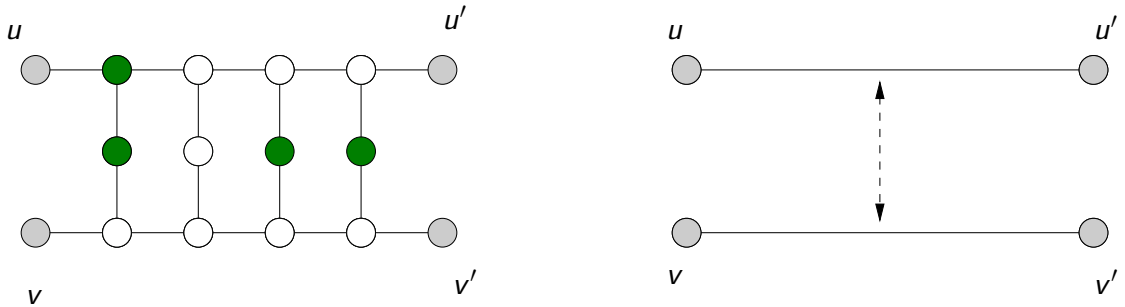
Corollary

The problem EULERIAN-CIRCUIT is in PTIME.

Proof.

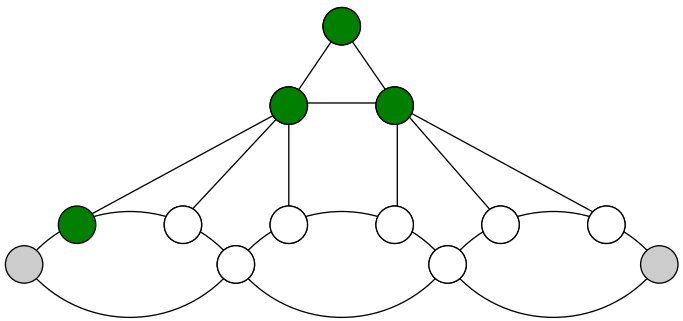
The condition that every node has even degree can obviously be tested in time $O(n^2)$. □

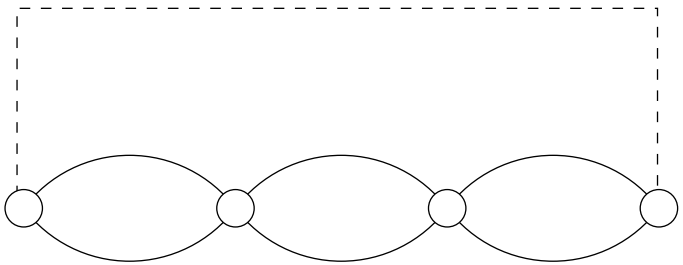
- By contrast, HAMILTONIAN-CIRCUIT is $\mathrm{NP}\textsc{Time}$-complete.
- Again, we use a series of gadgets:
- The first gadget ensures that any Hamiltonian circuit traverses exactly one of the 'edges' (u, u') or (v, v'),



with both possibilities realizable

- The second gadget ensures that any Hamiltonian circuit traverses at least one of the lower three edges,





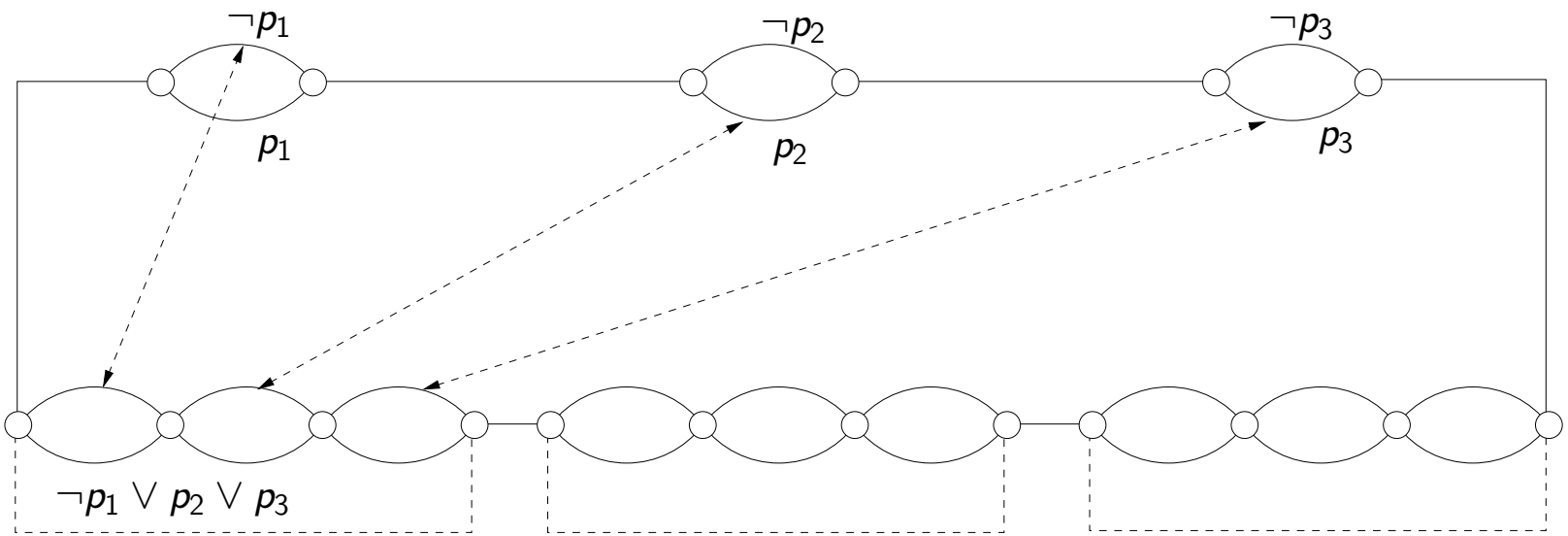
with all combinations possible.

Theorem

HAMILTONIAN-CIRCUIT is NPTIME*-complete.*

Proof.

We use the above gadgets to encode 3-SAT, along the following lines



- The problem HAMILTONIAN-CIRCUIT is closely related to the following 'problem'
  - Imagine a salesman given the task of touring n cities, which we identify as $1, \ldots, n$.
  - Take the distance between the cities i and j to be $c_{i,j}$.
  - In what order should the salesman tour the cities so as to minimize his total round trip?
- We call this 'problem' the travelling salesman problem (TSP).
- It has many practical applications, and been widely studied.

- More formally:

TSP	
Given:	an $n \times n$ symmetric matrix $\{c_{i,j}\}$ over $\mathbb{N}$
Return:	a tour though this matrix of minimal cost.

- This is not a problem in our technical sense (it does not have Yes/No answers); but the following is:

TSP-FEASIBILITY	
Given:	an $n \times n$ 'cost' matrix $\{c_{i,j}\}$ over $\mathbb{N}$, and some $k \in \mathbb{N}$
Return:	Yes, if there is a tour of $\{c_{i,j}\}$ with total cost $< k$; and No otherwise.

- In some sense, this is just as good ....

Theorem

TSP-FEASIBILITY is NPTIME*-complete.*

Proof.

Clearly, TSP-FEASIBILITY is in NP.

Given $G = (V, E)$, where $V = \{1, \ldots, n\}$, set $c_{i,j}$ to be 1 if $(v_i, v_j) \in E$, and 2 otherwise. Set $k = n + 1$. Thus we have a function f from instances of HAMILTONIAN-CIRCUIT to instances of TSP-FEASIBILITY, computable in logarithmic space.

Suppose there is a Hamiltonian circuit of G. Then that is a tour in $\{c_{i,j}\}$ of length n; and $n < k$.

Suppose there is a tour in $\{c_{i,j}\}$ of length $\leq n$ (hence exactly n). Then successive nodes on the tour must be joined by an edge in G, so this tour is is a Hamiltonian circuit in G.

Hence, f is a reduction, and so TSP-FEASIBILITY is NP-hard. □