Business Processes Modelling

MPB (6 cfu, 295AA)

Roberto Bruni

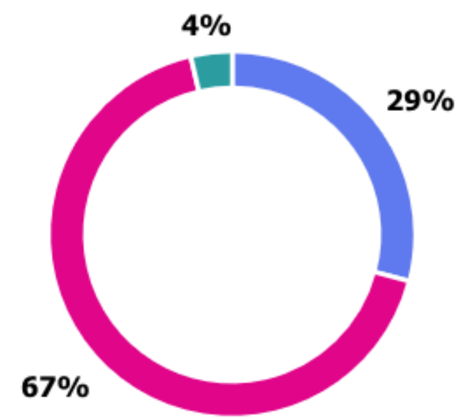
http://www.di.unipi.it/~bruni

17 aux - P and NP problems

Complexity classes

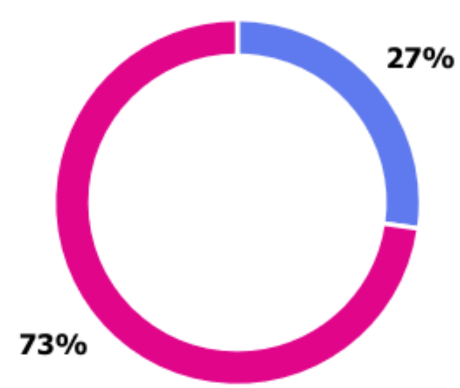
23. Do you know what are the algorithmic time complexity classes **P** and **NP**? Più dettagli

Yes	16
No	37
I do not understand the question	2



24. Do you know what is a **NP-complete** problem? Più dettagli

Yes	15
No	40



Problems and instances

A **problem** defines a family of related questions

For example, the factorization problem is:
“given a number n, return all its prime factors”

A **problem instance** is one such question

An instance of the factorization problem is:
“return all prime factors of 18”

Decision problem

A **decision problem** requires just a **boolean answer**

For example: *“given a number n, is n prime?”*

And an instance: *“is 18 prime?”*

Computational Complexity Theory

Computational complexity theory deals with the resources needed to solve a problem

how many basic operations (time)
or how much memory (space)
it takes to solve a problem

P

The complexity class **P** is the set of decision problems that can be solved by a deterministic algorithm in a **P**olynomial number of steps (time) w.r.t. input size

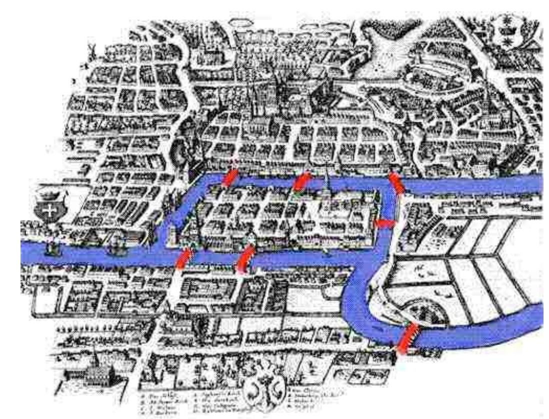
Problems in **P** can be (checked and) **solved effectively**

Eulerian circuit (P)

Given a graph G,
is it possible to draw an Eulerian circuit over it?
(i.e. a circuit that traverses each edge exactly once)

We have seen that it is the same problem as:

Given a graph G,
is the degree of each vertex even?



The problem can be (checked and) solved effectively
(linear time w.r.t. number of arcs)!

NP

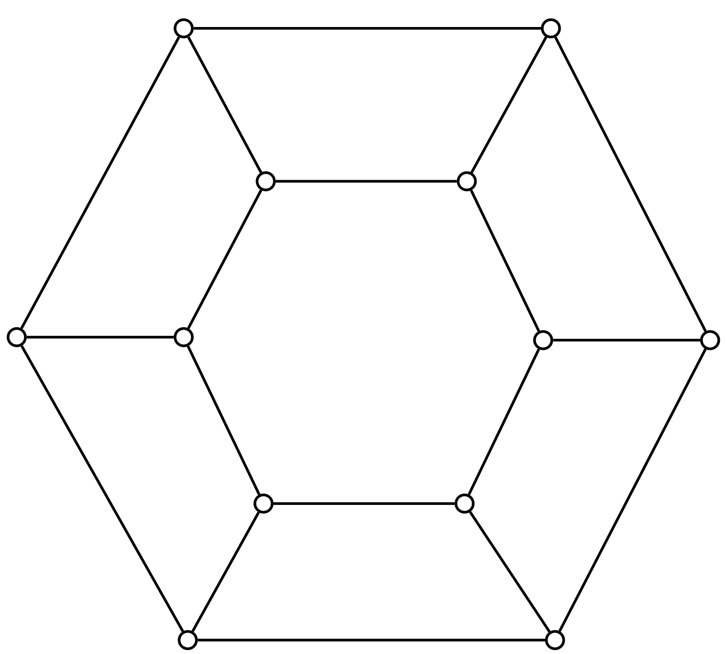
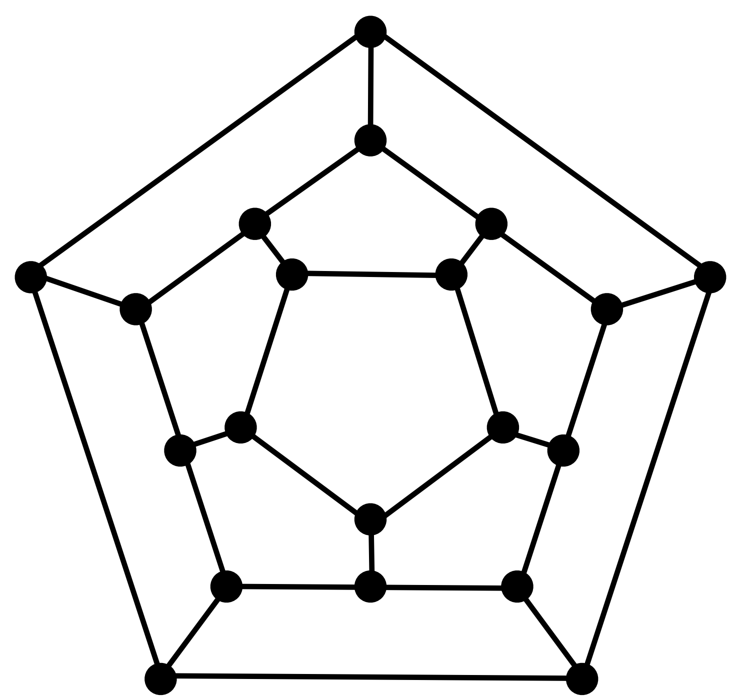
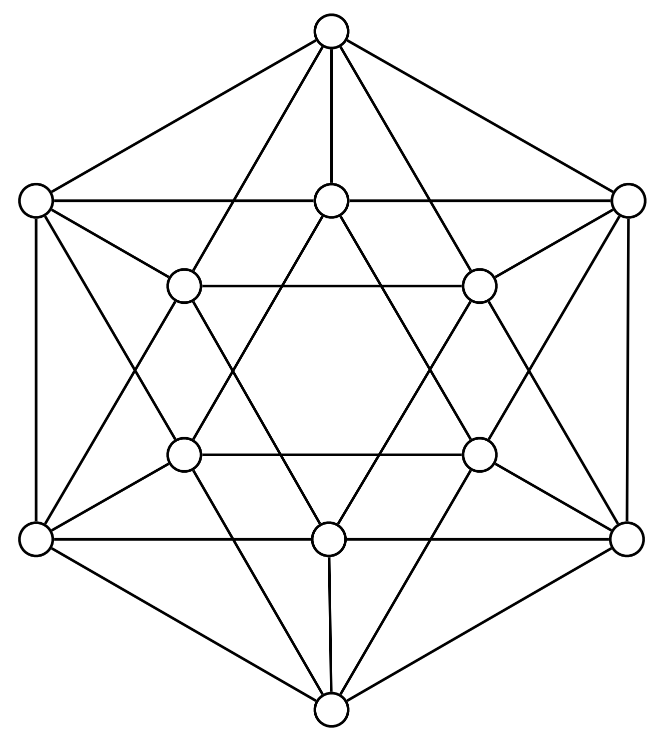
The complexity class **NP** is the set of decision problems that can be **solved** by a **N**on-deterministic algorithm in a **P**olynomial number of steps (time)

Equivalently **NP** is the set of decision problems whose solutions can be **checked** by a deterministic algorithm in a polynomial number of steps (time)

Solutions of problems in **NP** can be **checked effectively**

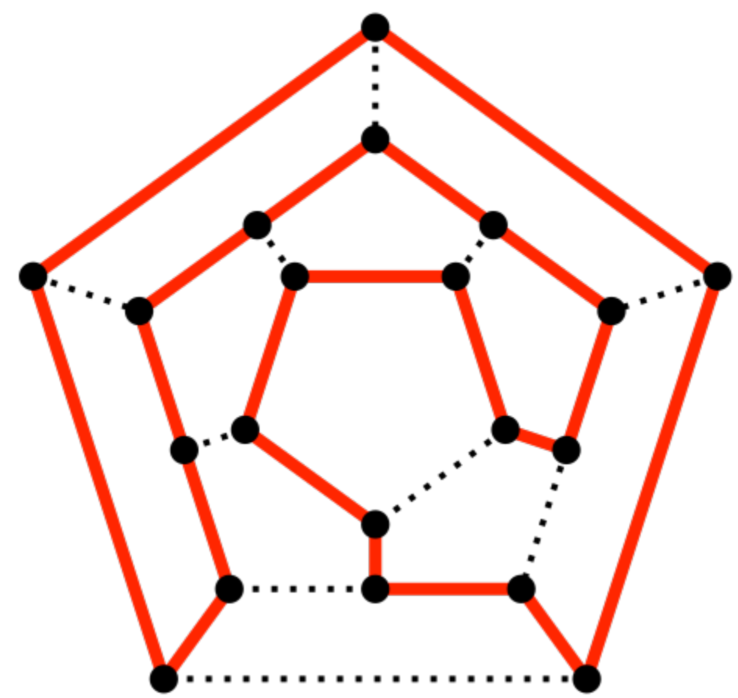
Hamiltonian circuit

Given a graph G,
is it possible to draw an Hamiltonian circuit over it?
(i.e. a circuit that visits each vertex exactly once)

Hamiltonian circuit (NP)

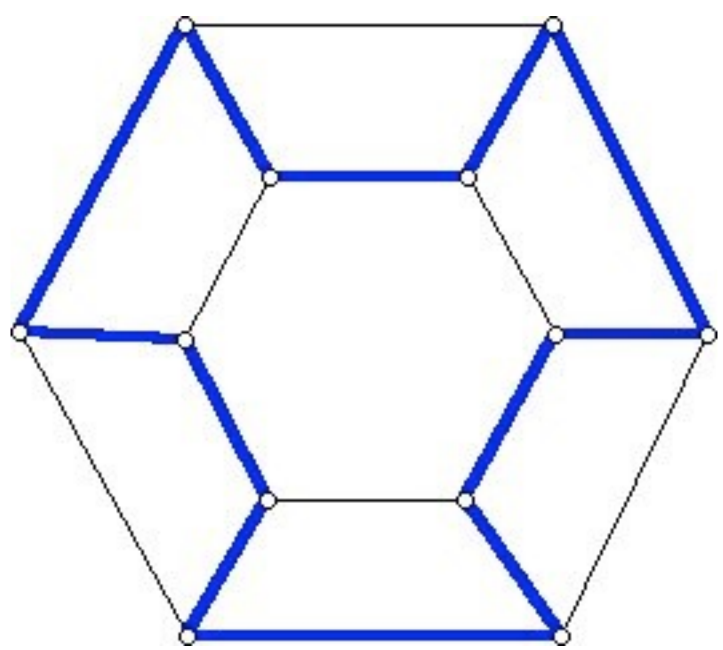
Given a graph G,
is it possible to draw an Hamiltonian circuit over it?
(i.e. a circuit that visits each vertex exactly once)



The problem can be checked effectively!

Hamiltonian circuit (NP)

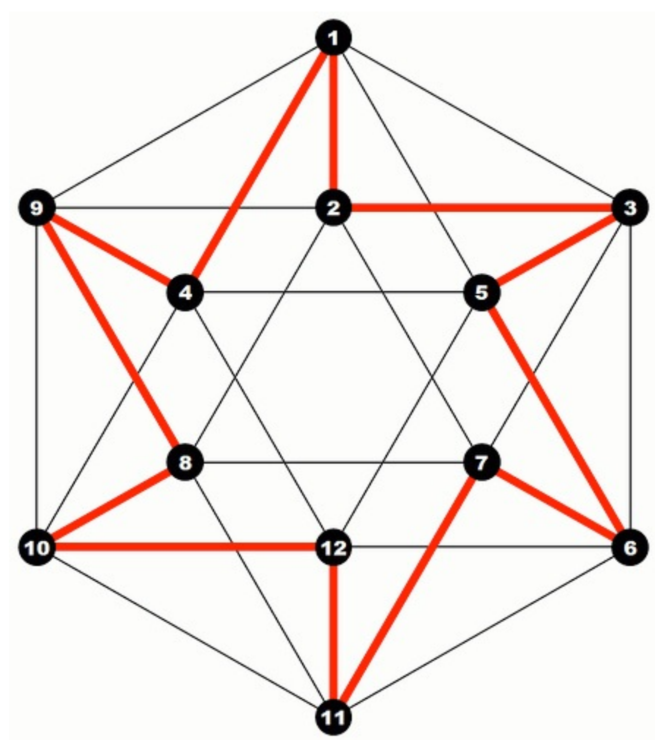
Given a graph G,
is it possible to draw an Hamiltonian circuit over it?
(i.e. a circuit that visits each vertex exactly once)



The problem can be checked effectively!

Hamiltonian circuit (NP)

Given a graph G,
is it possible to draw an Hamiltonian circuit over it?
(i.e. a circuit that visits each vertex exactly once)

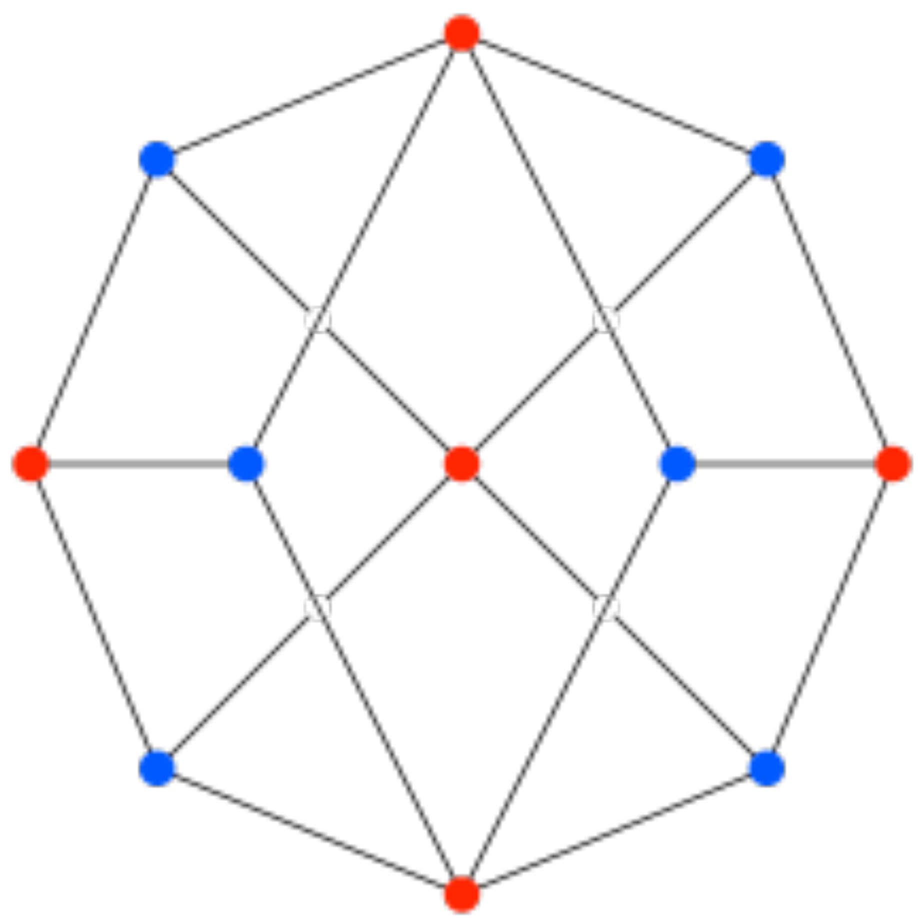


The problem can be checked effectively!

Hamiltonian circuit (NP)

Given a graph G,
is it possible to draw an Hamiltonian circuit over it?
(i.e. a circuit that visits each vertex exactly once)

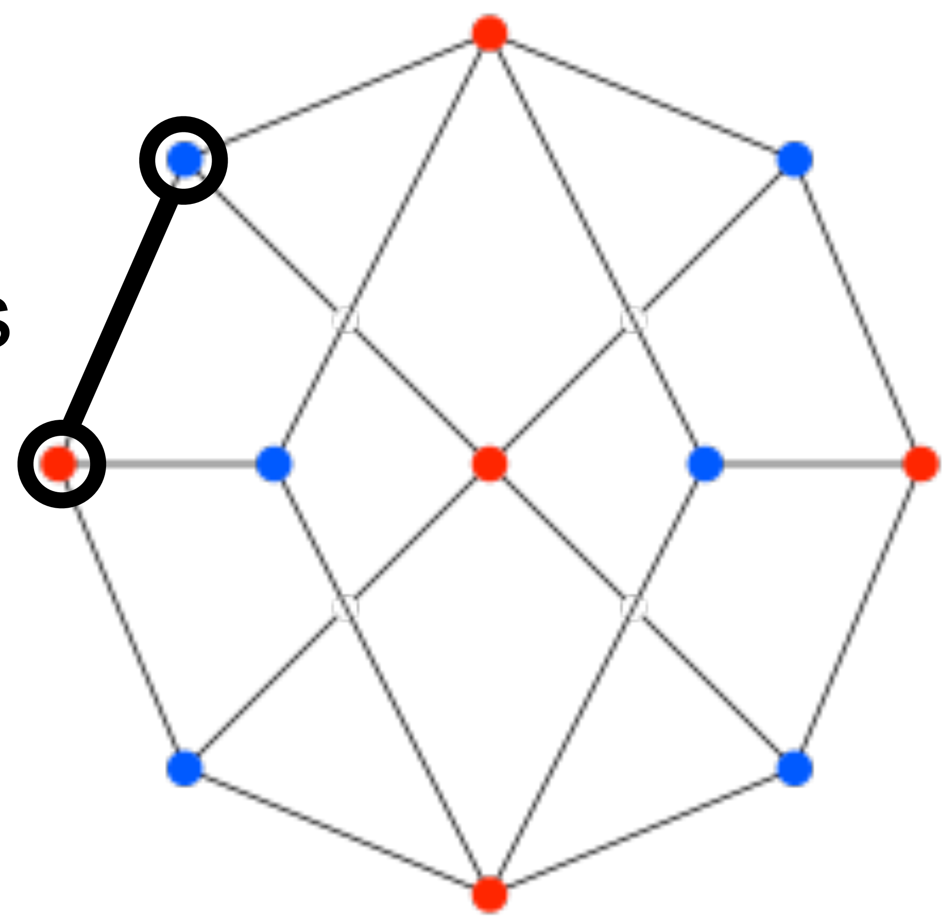
The problem looks difficult to solve



Hamiltonian circuit (NP)

Given a graph G,
is it possible to draw an Hamiltonian circuit over it?
(i.e. a circuit that visits each vertex exactly once)

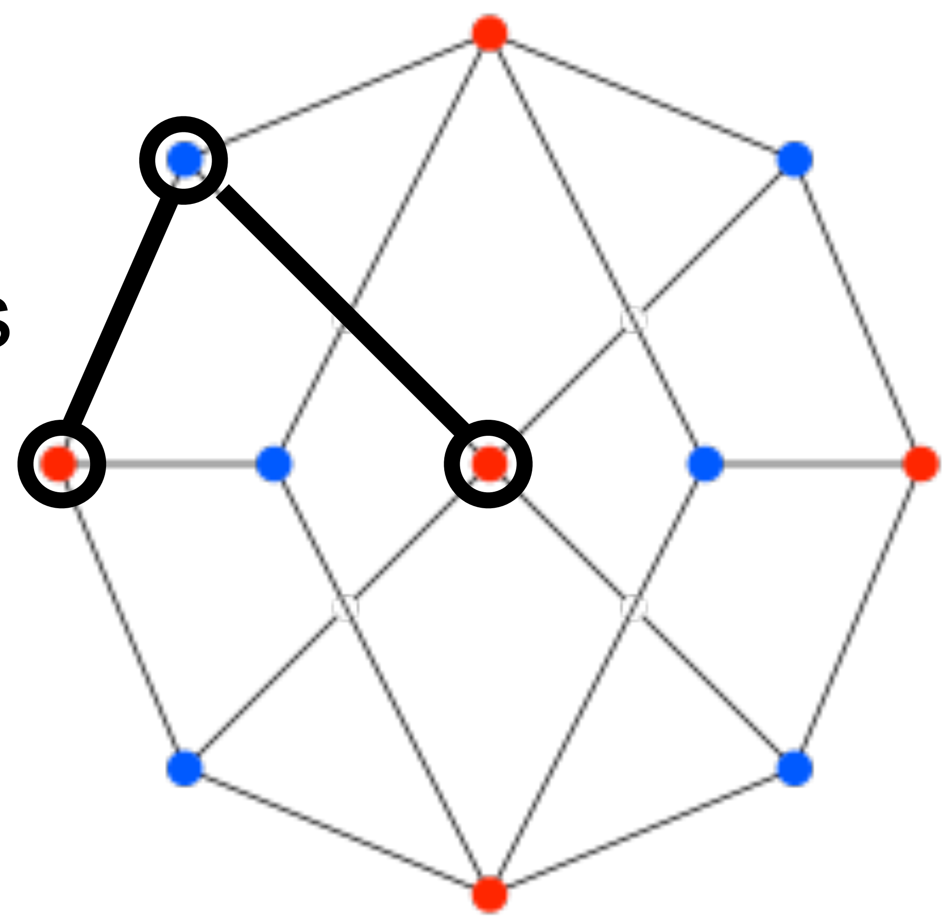
The problem looks
difficult to solve



Hamiltonian circuit (NP)

Given a graph G,
is it possible to draw an Hamiltonian circuit over it?
(i.e. a circuit that visits each vertex exactly once)

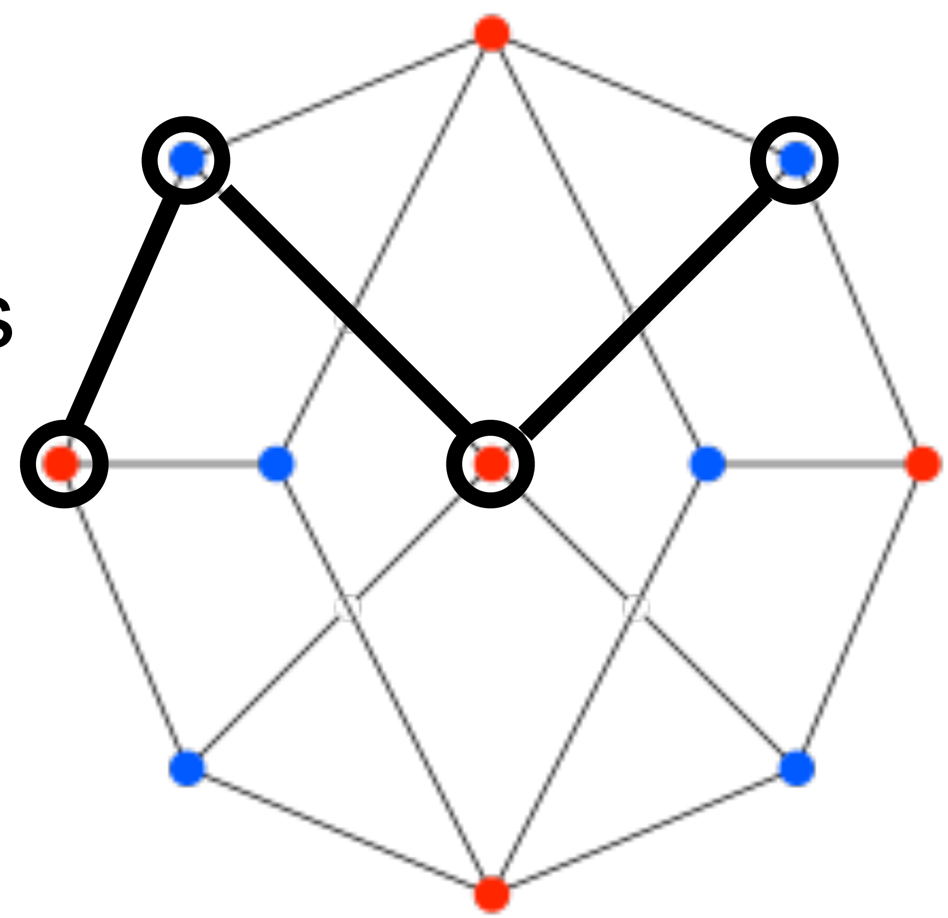
The problem looks difficult to solve



Hamiltonian circuit (NP)

Given a graph G,
is it possible to draw an Hamiltonian circuit over it?
(i.e. a circuit that visits each vertex exactly once)

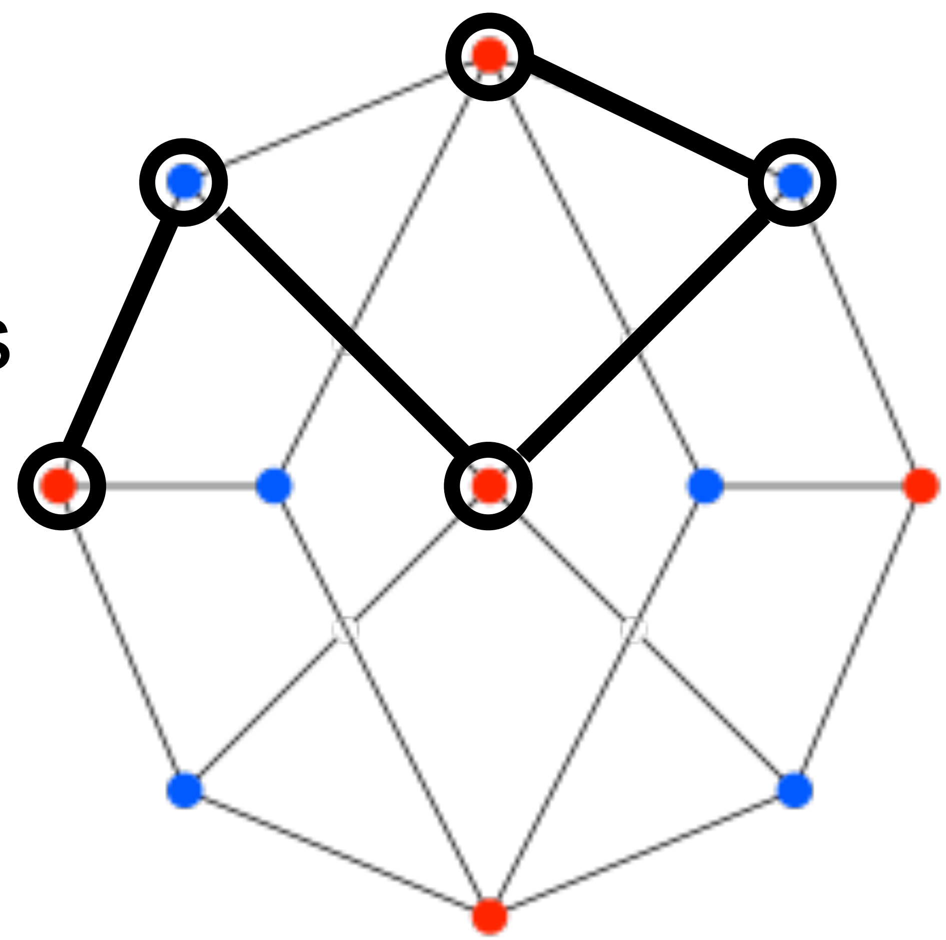
The problem looks difficult to solve



Hamiltonian circuit (NP)

Given a graph G,
is it possible to draw an Hamiltonian circuit over it?
(i.e. a circuit that visits each vertex exactly once)

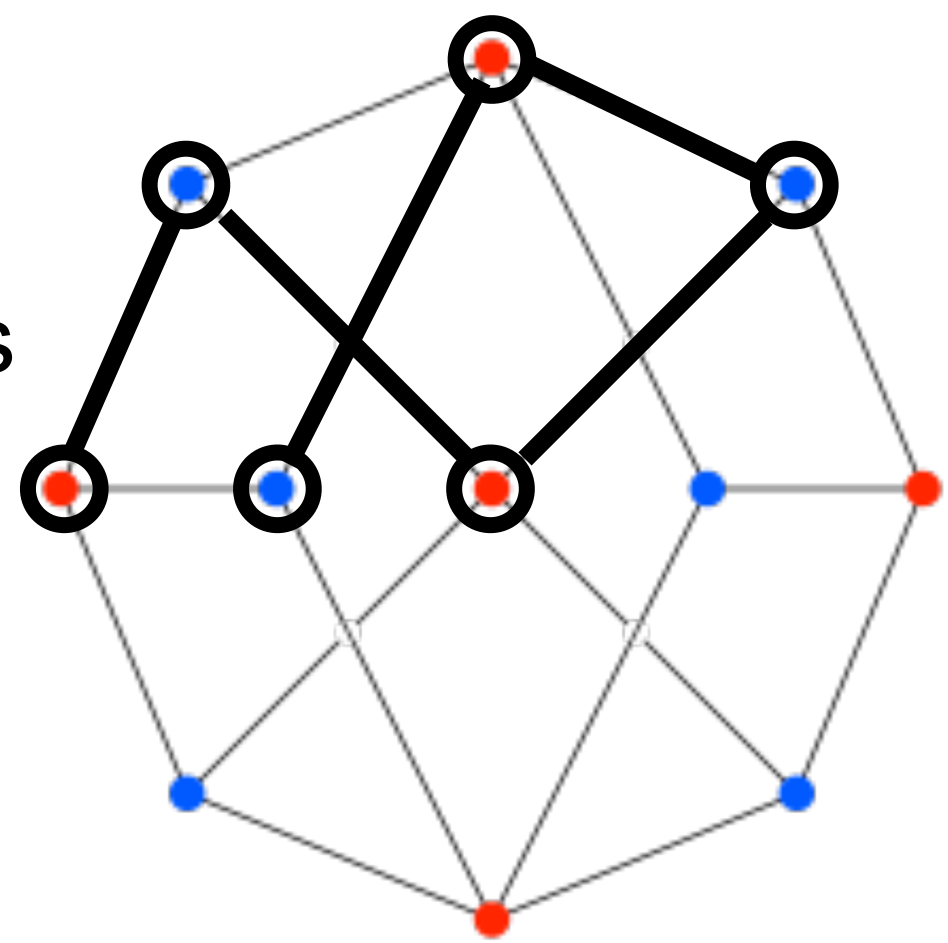
The problem looks difficult to solve



Hamiltonian circuit (NP)

Given a graph G,
is it possible to draw an Hamiltonian circuit over it?
(i.e. a circuit that visits each vertex exactly once)

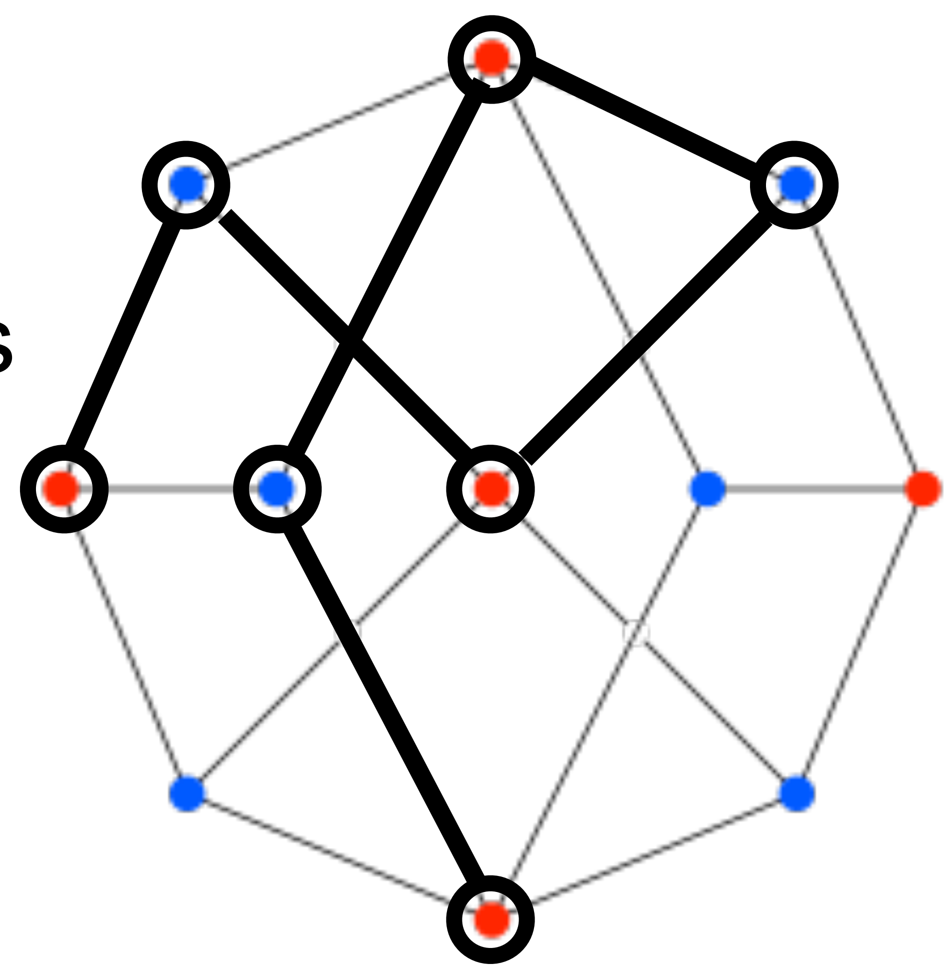
The problem looks difficult to solve



Hamiltonian circuit (NP)

Given a graph G,
is it possible to draw an Hamiltonian circuit over it?
(i.e. a circuit that visits each vertex exactly once)

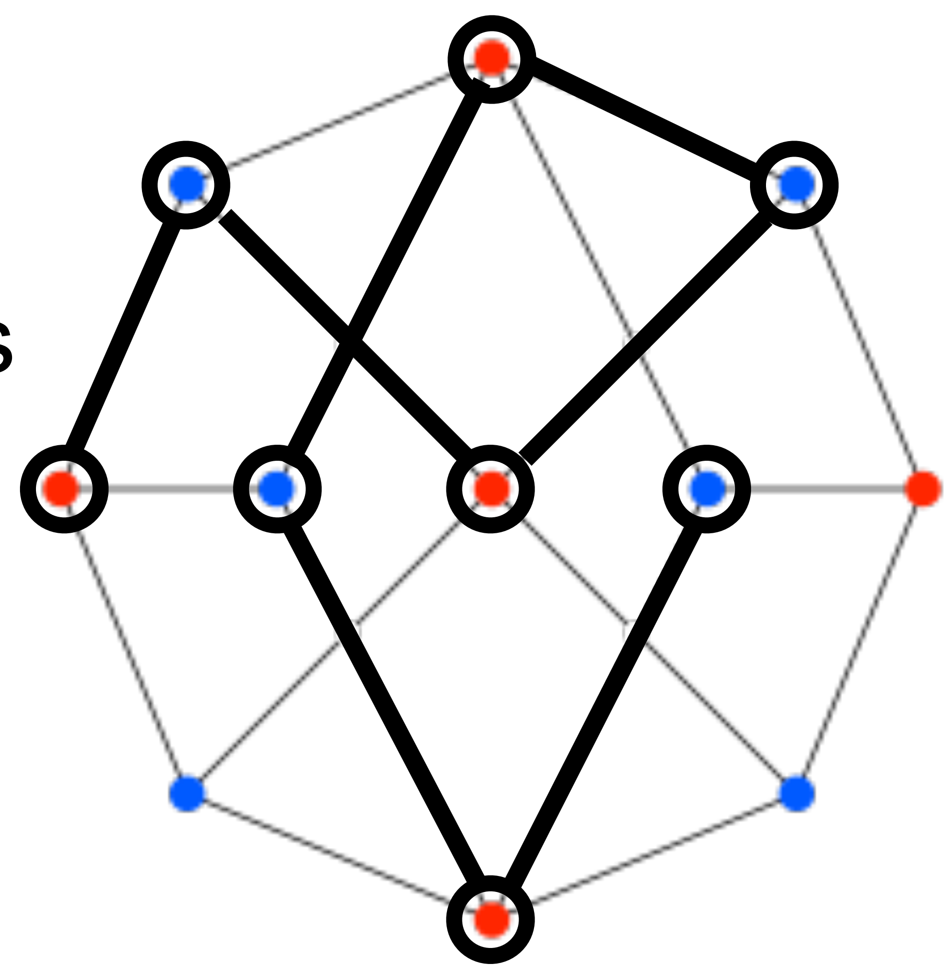
The problem looks difficult to solve



Hamiltonian circuit (NP)

Given a graph G,
is it possible to draw an Hamiltonian circuit over it?
(i.e. a circuit that visits each vertex exactly once)

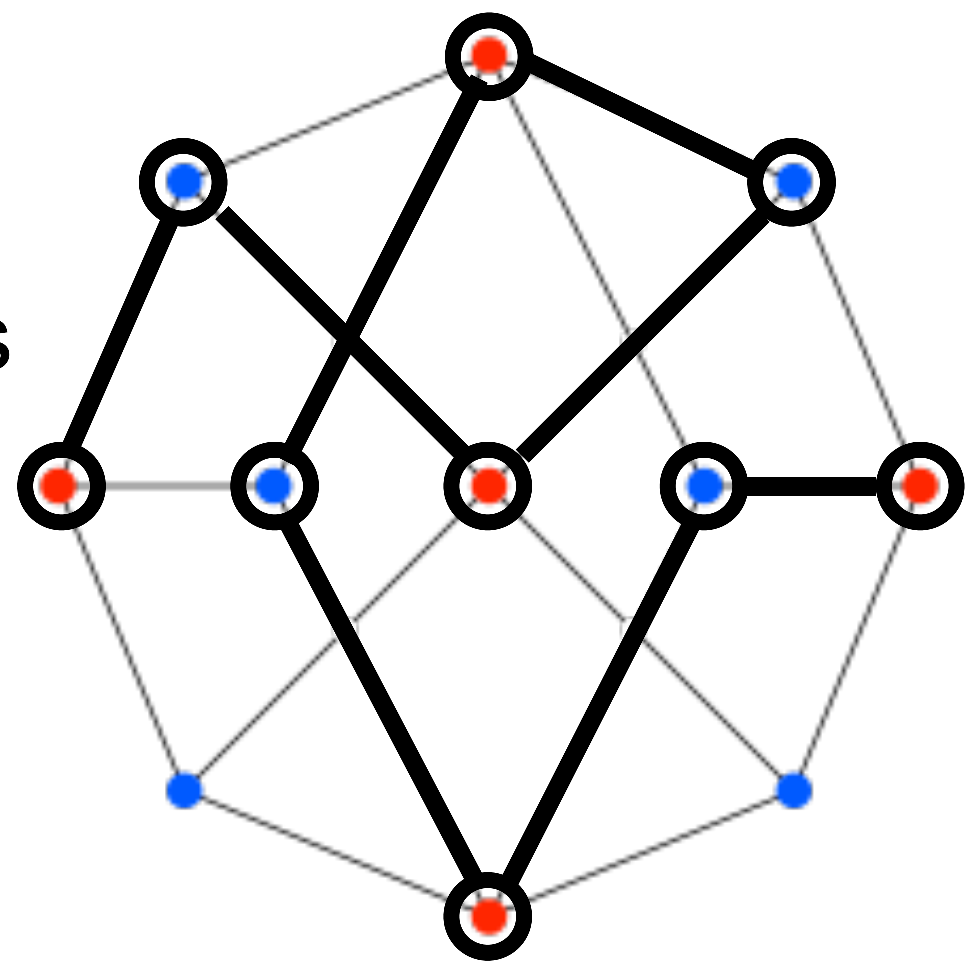
The problem looks difficult to solve



Hamiltonian circuit (NP)

Given a graph G,
is it possible to draw an Hamiltonian circuit over it?
(i.e. a circuit that visits each vertex exactly once)

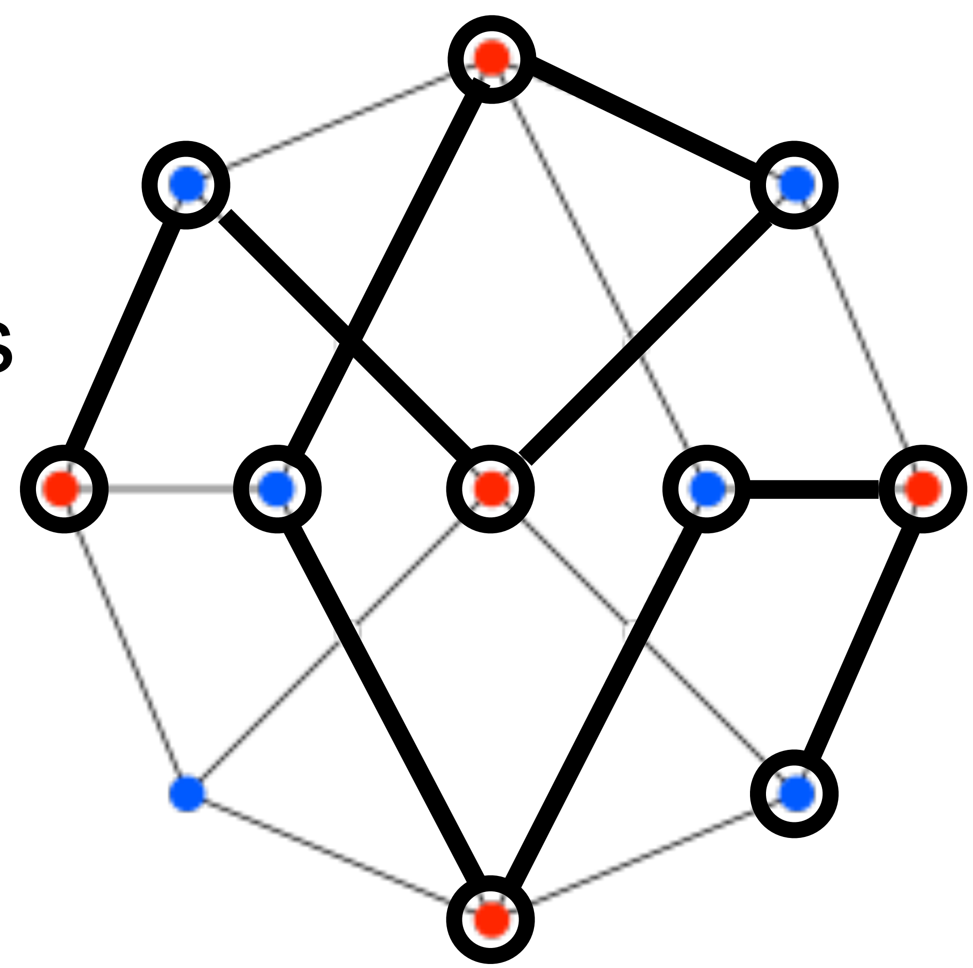
The problem looks difficult to solve



Hamiltonian circuit (NP)

Given a graph G,
is it possible to draw an Hamiltonian circuit over it?
(i.e. a circuit that visits each vertex exactly once)

The problem looks difficult to solve



P vs NP

The question of whether **P** is the same set as **NP** is the most important open question in computer science

Intuitively, it is much harder to solve a problem than to check the correctness of a solution

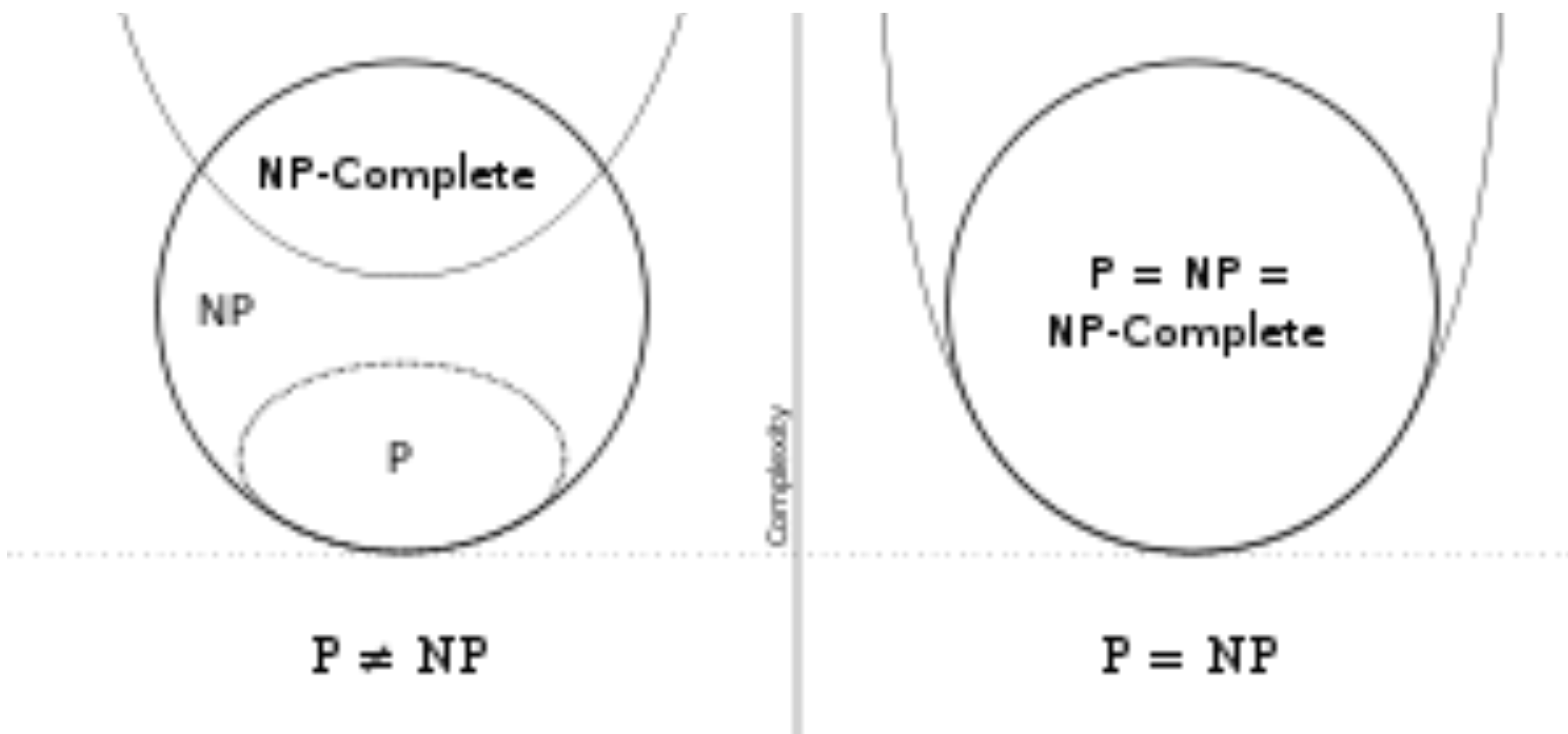
A fact supported by our daily experience, which leads us to conjecture **P** ≠ **NP**

What if “solving” is not really harder than “checking”? what if **P** = **NP**?

NP-completeness

A problem Q in **NP** is **NP-complete** if every other problem in **NP** can be reduced to Q (in polynomial time)

(finding an effective way to solve such a problem Q would allow to solve effectively any other problem in **NP**)



SAT decision problem (is NP-complete)

Variables: $x_1, x_2, ..., x_n$

Literals: $x_1, \bar{x}_1, x_2, \bar{x}_2, ..., x_n, \bar{x}_n$

Clause: disjunction of literals

Formula: conjunction of clauses

Example: $\phi = (x_1 \vee \bar{x_3}) \wedge (x_1 \vee \bar{x}_2 \vee x_3) \wedge (x_2 \vee \bar{x}_3)$

Is there an assignment of boolean values to the variables such that $\phi = true$?

$$x_1 = true, x_2 = true, x_3 = true$$

(NP-complete) SAT decision problem

Try yourself and play the SAT game:

online SAT game by Olivier Roussel

(just 9 variables, but 5 difficulty levels)