Algorithms

Lecture Topic: NP Completeness (Part 1)

Anxiao (Andrew) Jiang

Roadmap of this lecture:

1. NP Completeness

   1.1 Polynomial-time algorithms.

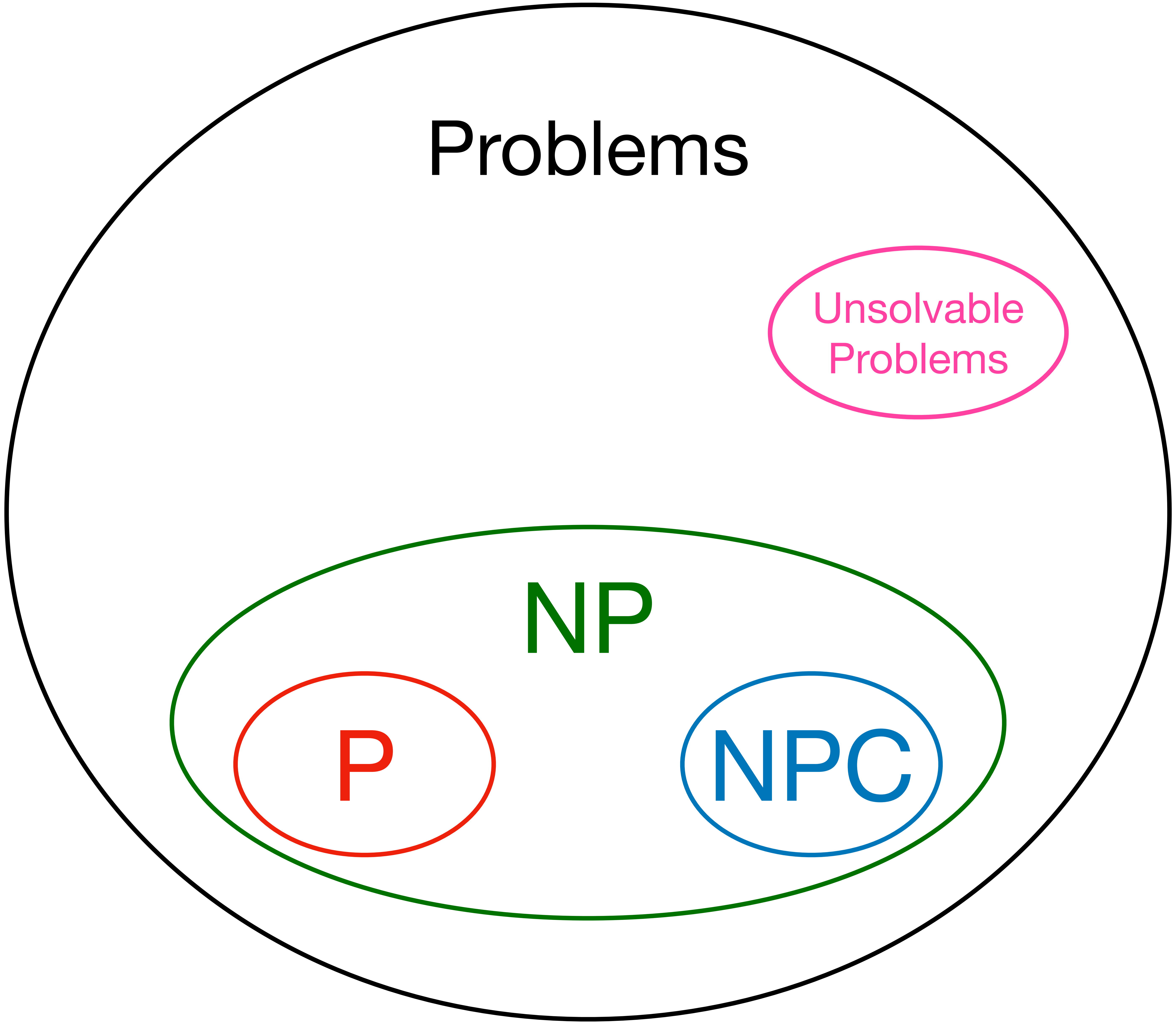
   1.2 Examples of NP-complete problems.

   1.3 P versus NP.

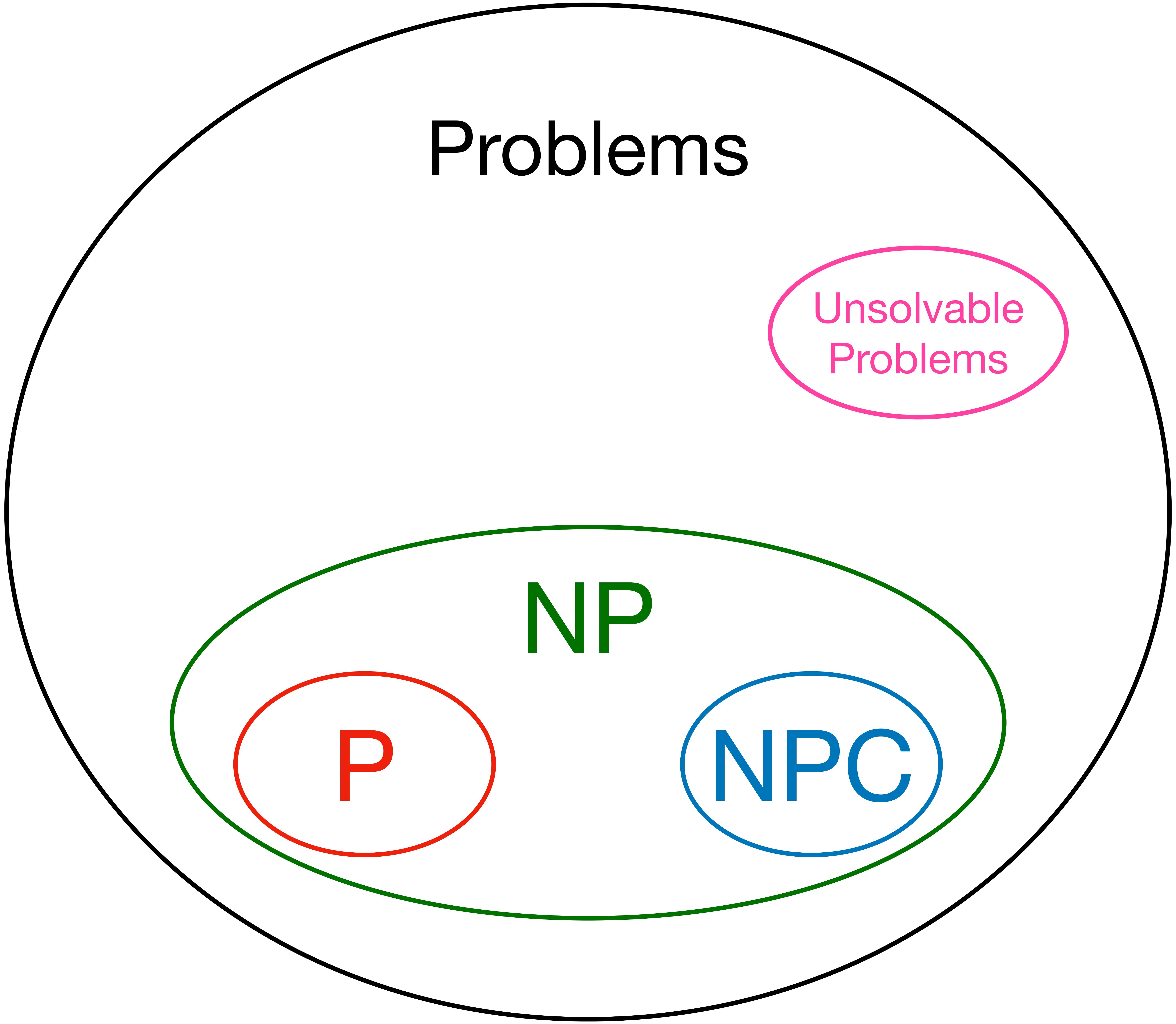
   1.4 Decision problems.

   1.5 Polynomial-time reduction.

NP Completeness



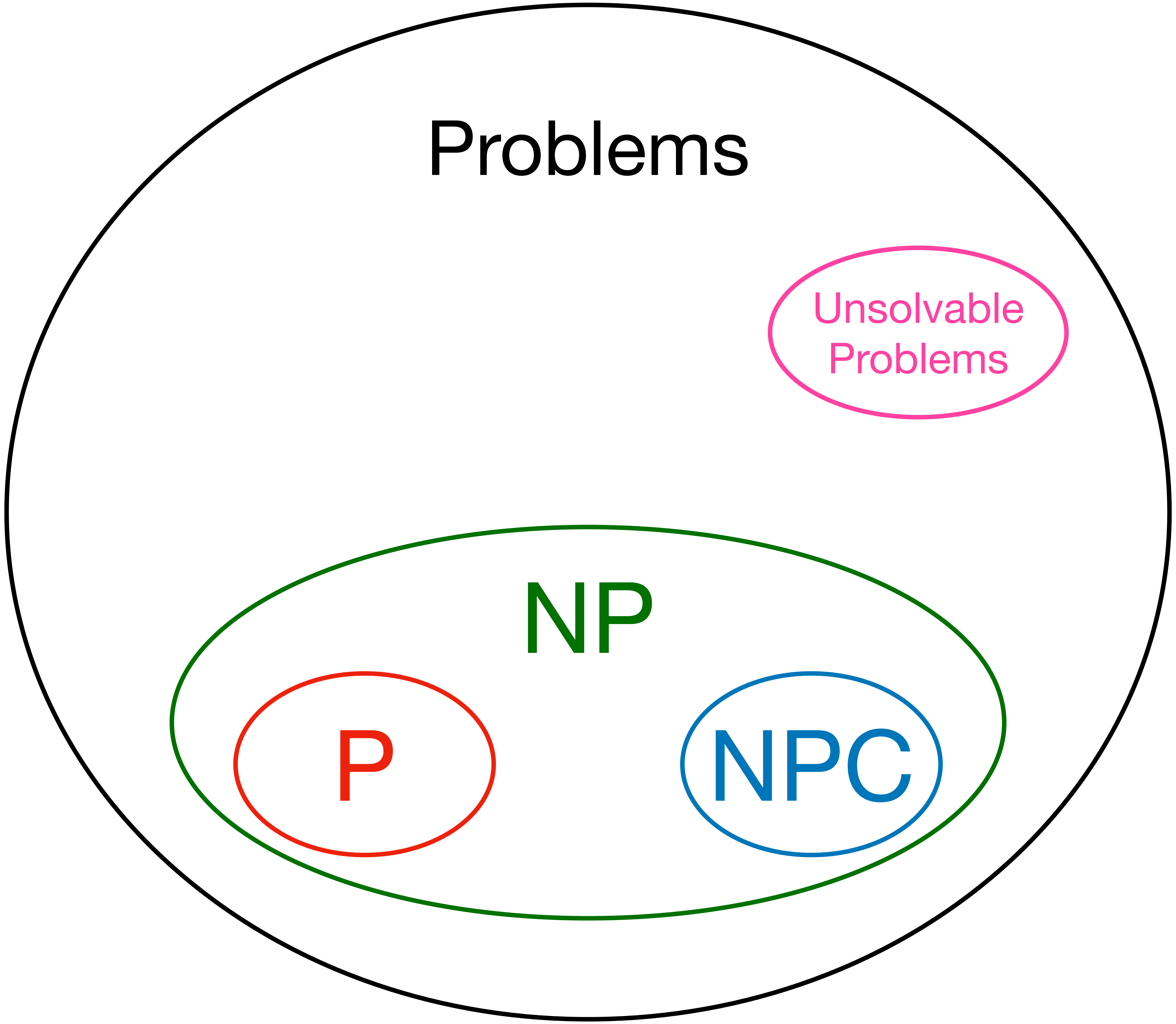
NP Completeness



Example of unsolvable problems:

Turing’s Halting Problem

NP Completeness



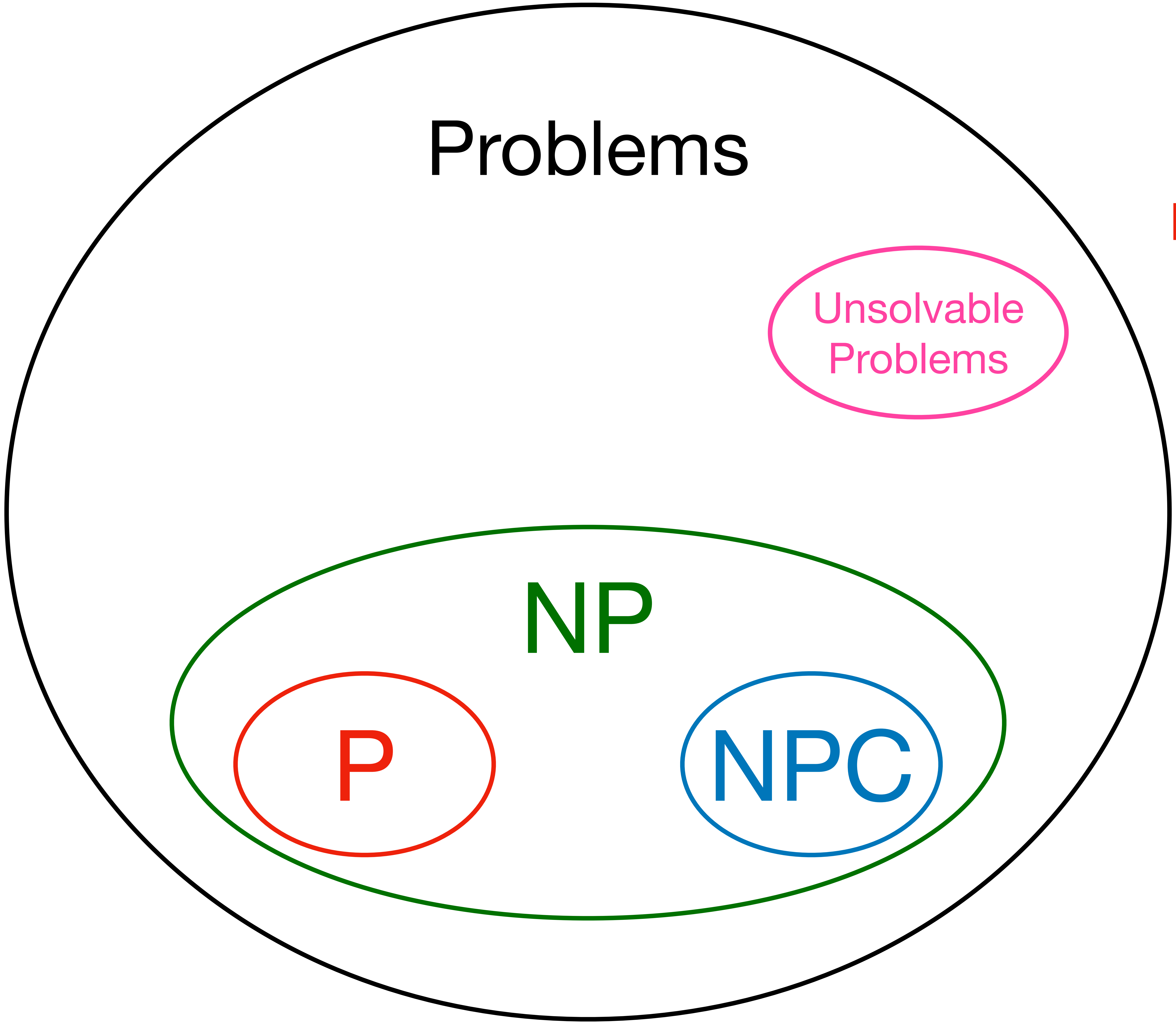
Example of unsolvable problems:

Turing’s Halting Problem

Among solvable problems:

Polynomial-time
v.s.
Super-polynomial time
(Often exponential time)

NP Completeness



Polynomial-time algorithm

Problem's input size: n bits

Polynomial time: $O(n)$, $O(n^2)$, $O(n^c)$

Polynomial-time solvable

P: the set of all problems that can be solved in polynomial time.

Instance of a problem

Worst-case time complexity

Quiz questions:

1. What are polynomial-time solvable problems?
2. Are all problems solvable in polynomial time?

Roadmap of this lecture:

1. NP Completeness

    1.1 Polynomial-time algorithms.

    1.2 Examples of NP-complete problems.

    1.3 P versus NP.

    1.4 Decision problems.

    1.5 Polynomial-time reduction.

NP Completeness

Shortest-Path Problem

Input: A directed graph G=(V,E),

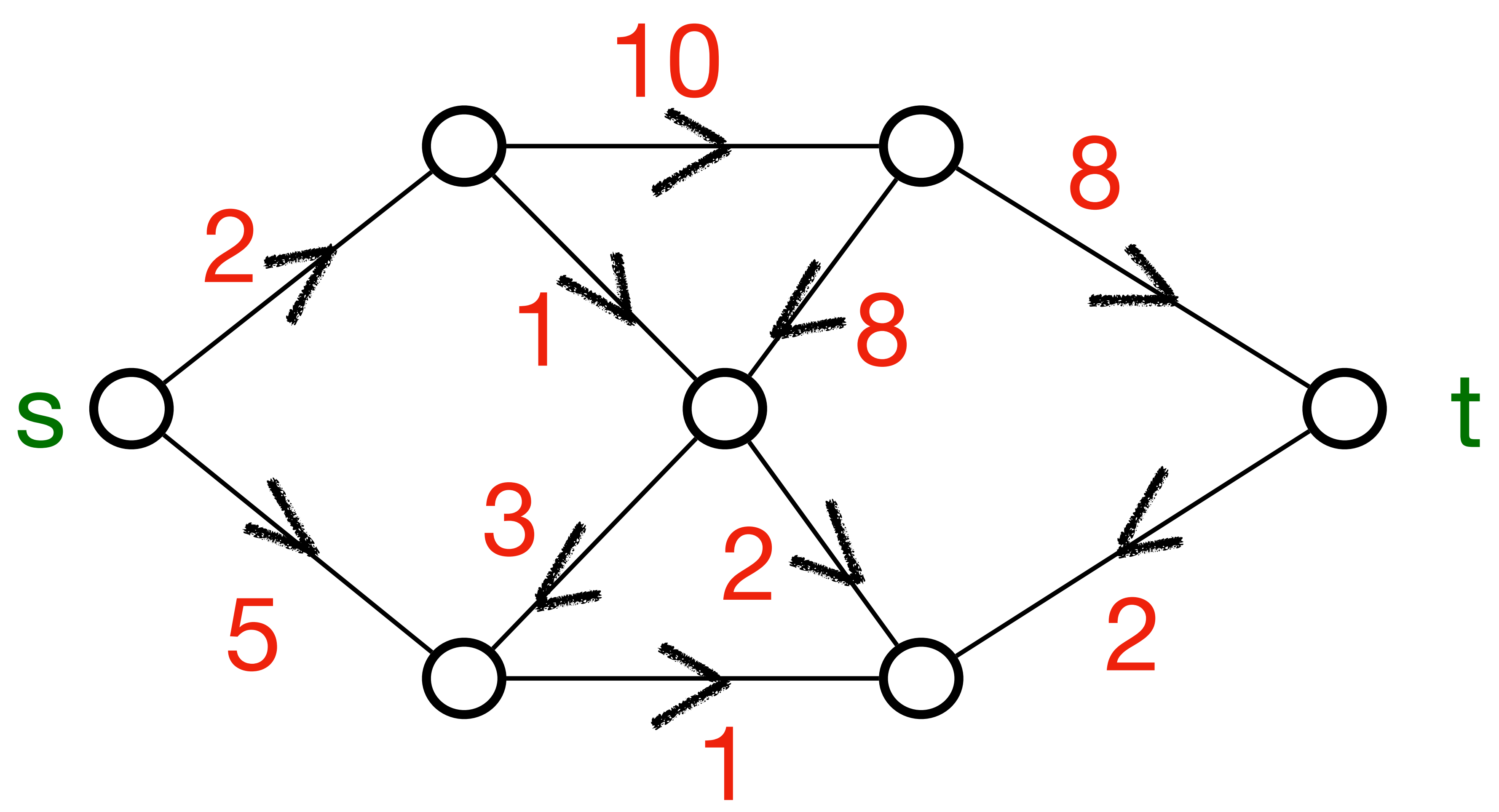
where every edge $e \in E$

has a weight $w(e) > 0$.

Let $s, t \in V$ be two nodes.

Output: A shortest path from s to t.

An instance:

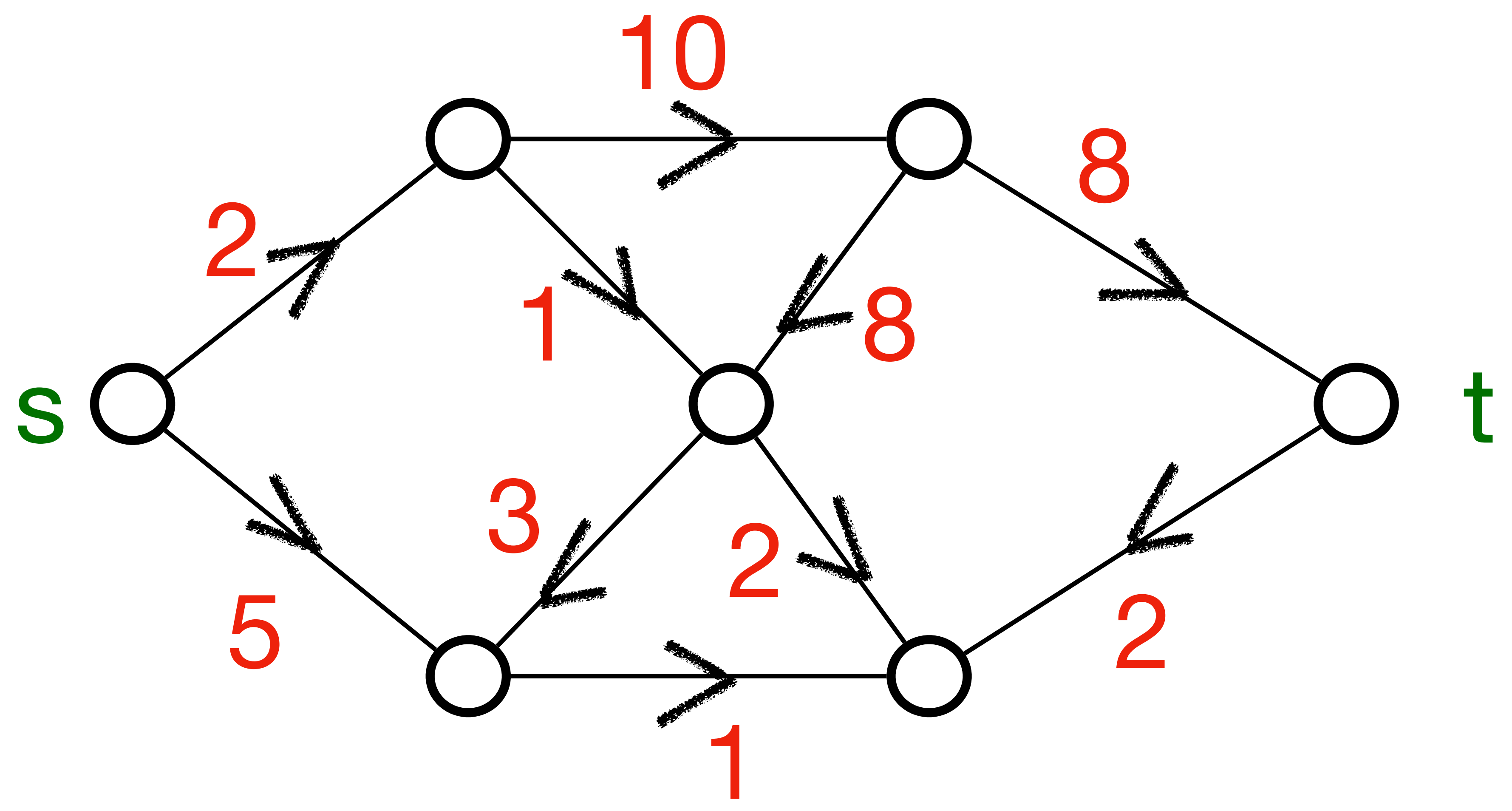


Shortest-Path Problem can be solved in polynomial time. (We have learned it.)

NP Completeness

Shortest-Path Problem

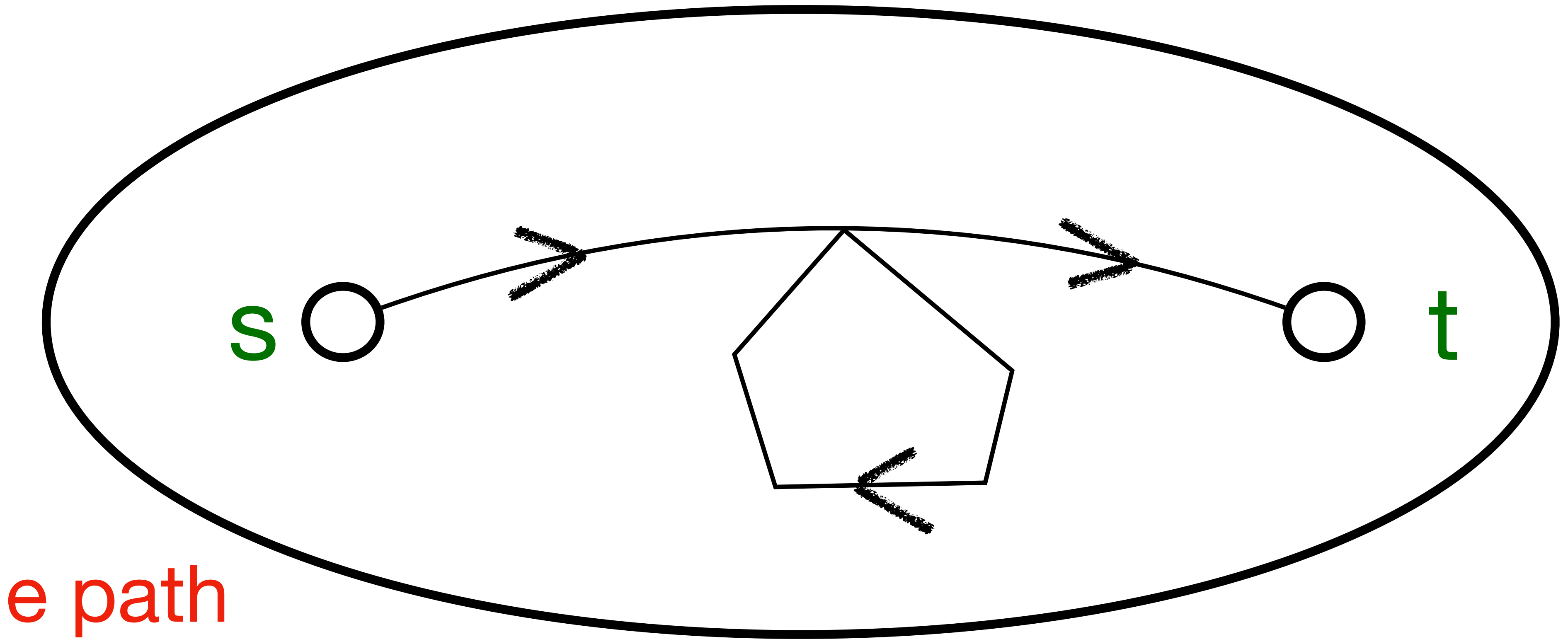
An instance:



Input: A directed graph G=(V,E), where every edge $e \in E$ has a weight $w(e) > 0.$ Let $s, t \in V$ be two nodes.

Output: A shortest path from s to t.

Simple path: a path without cycles.

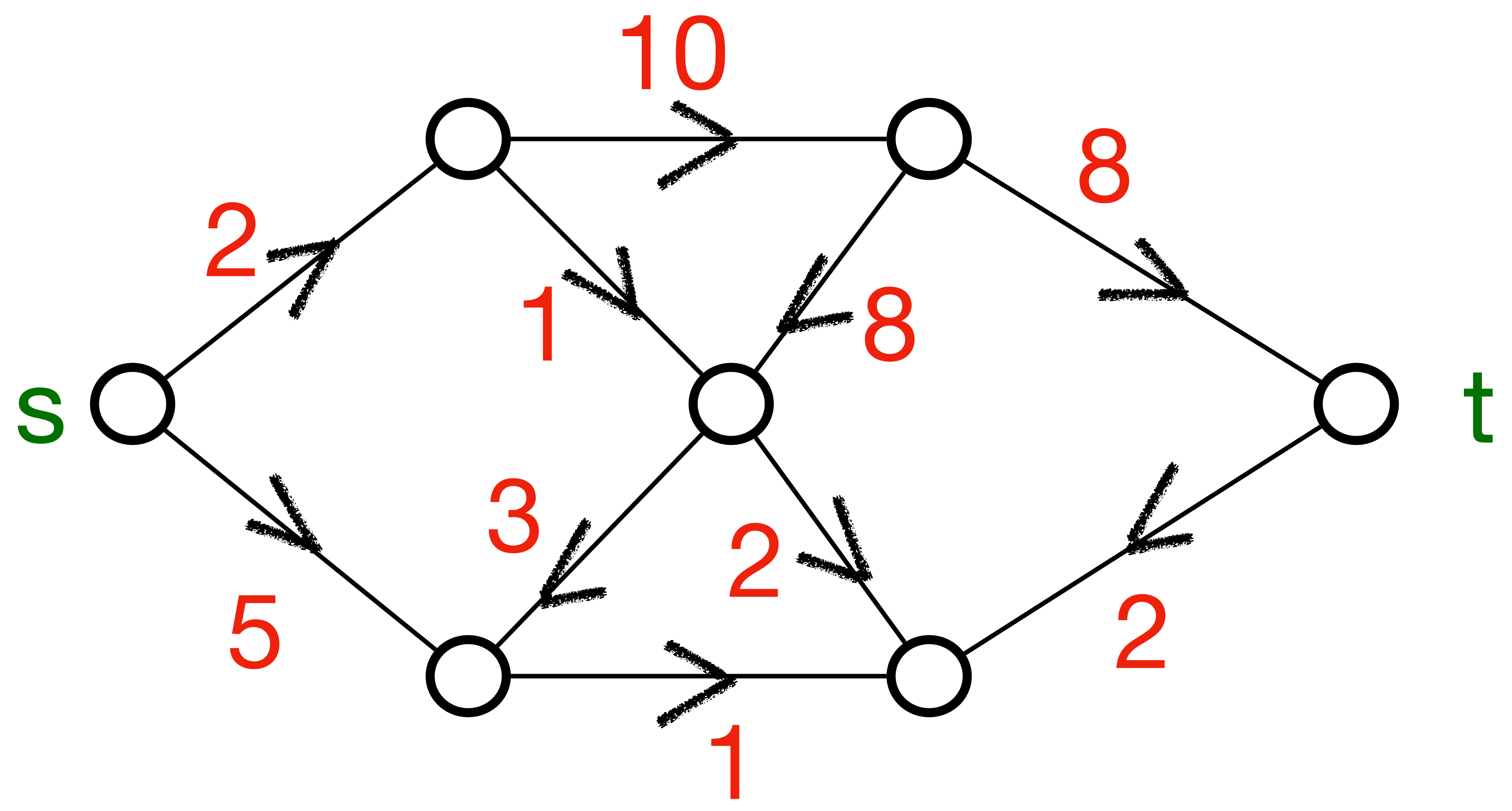


Not simple path

NP Completeness

Shortest-Path Problem

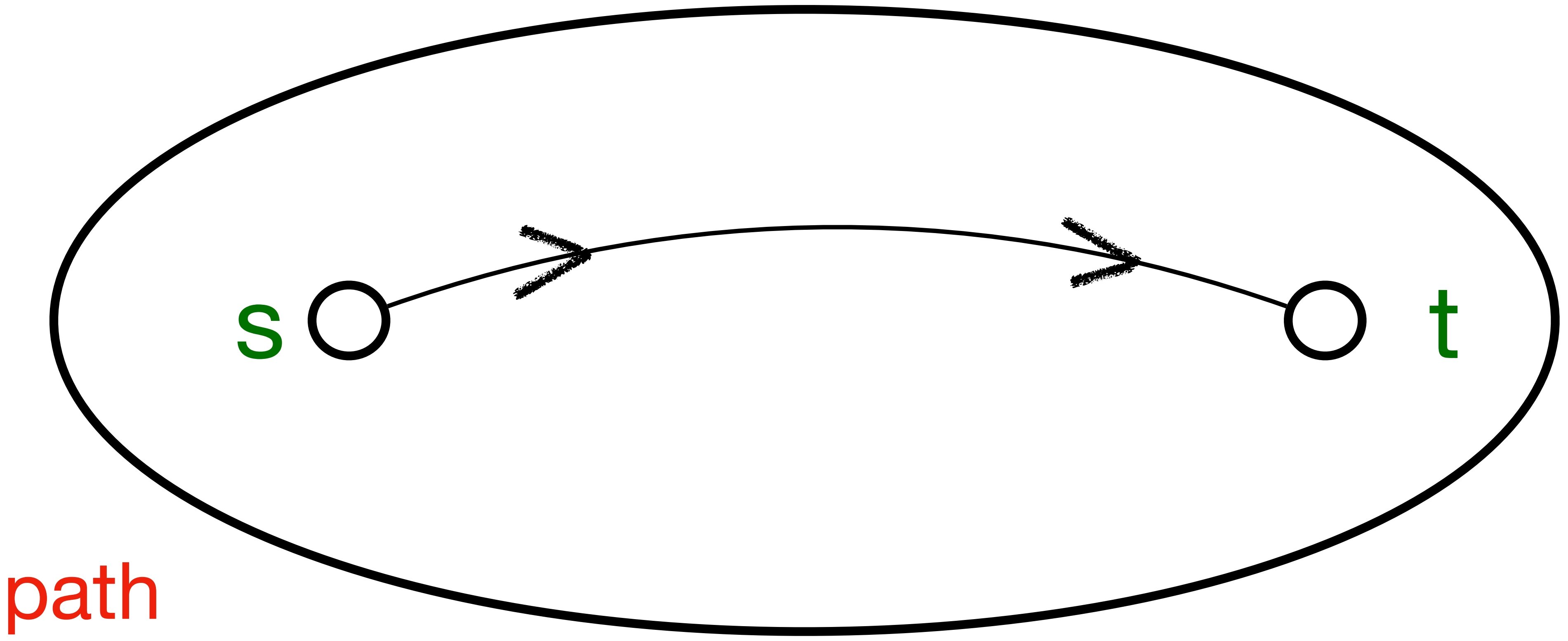
An instance:



Input: A directed graph G=(V,E), where every edge $e \in E$ has a weight $w(e) > 0$. Let $s, t \in V$ be two nodes.

Output: A shortest path from s to t.

Simple path: a path without cycles.



simple path

NP Completeness

Longest
~~Shortest~~-Path Problem

An instance:

Input: A directed graph G=(V,E),

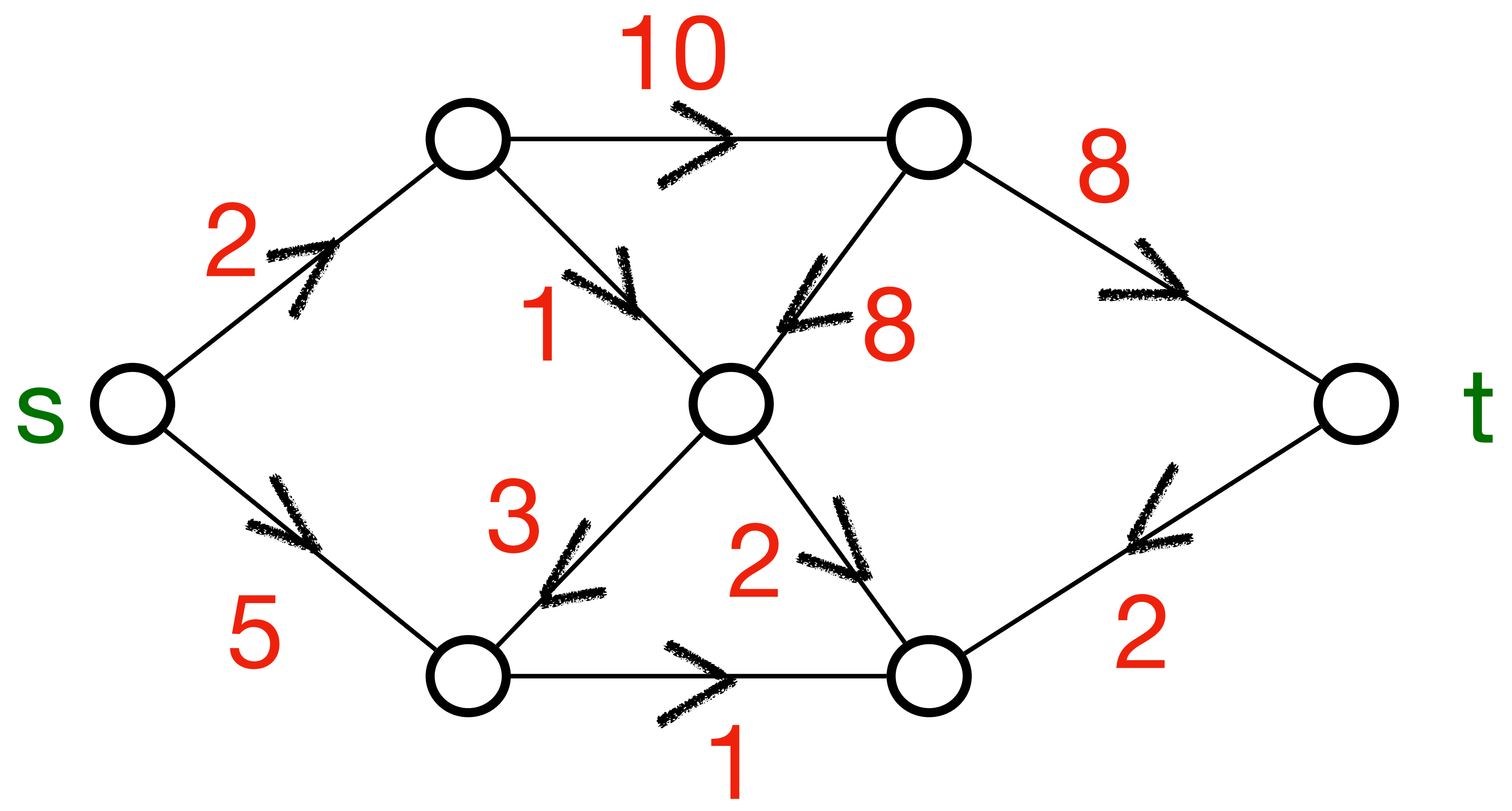
where every edge $e \in E$

has a weight $w(e) > 0$.

Let $s, t \in V$ be two nodes.

Output: A ~~shortest~~ simple path from s to t.

longest



Can the Longest Path Problem be solved in polynomial time?

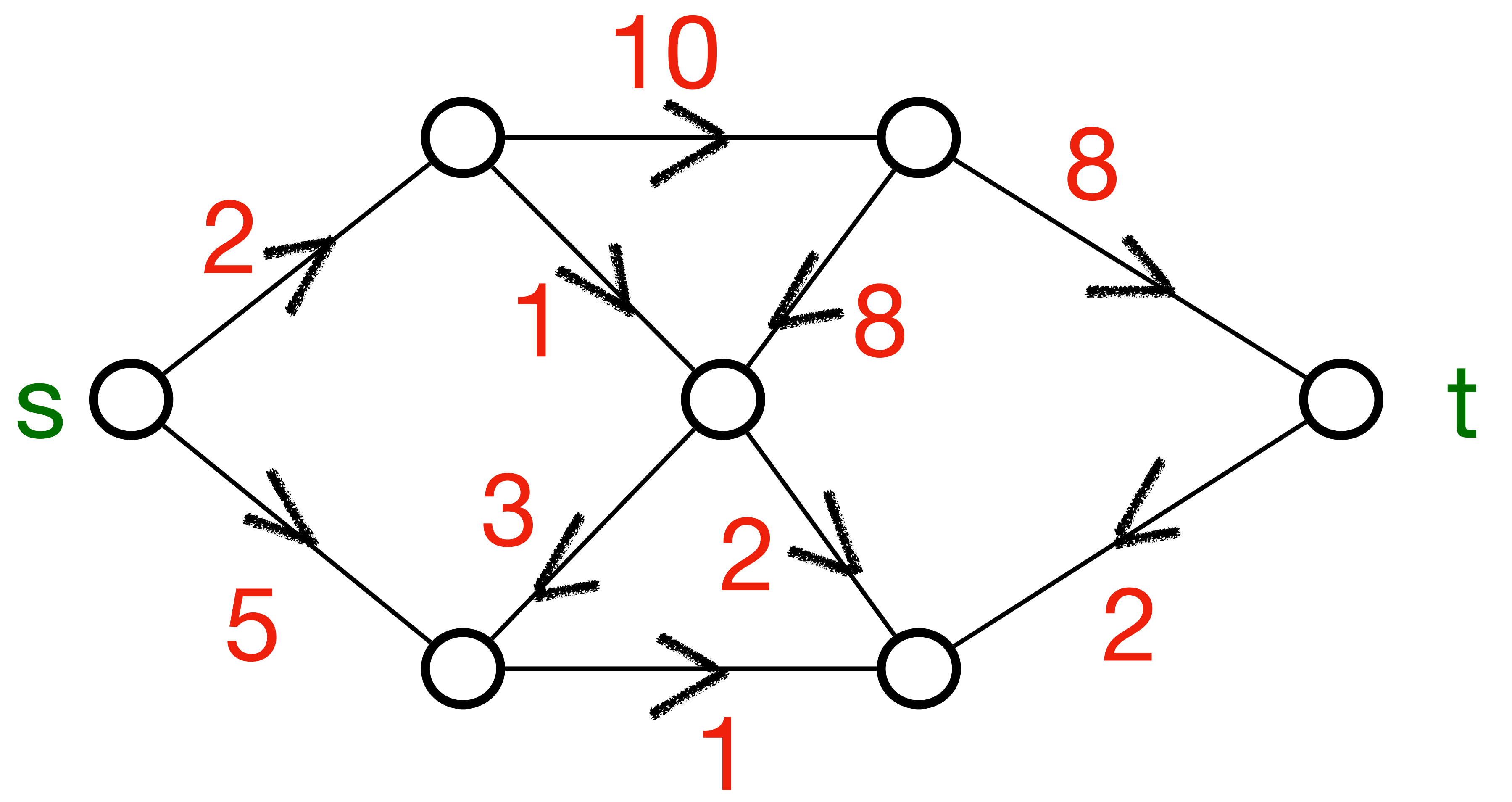
NP Completeness

Longest
~~Shortest~~-Path Problem

Input: A directed graph G=(V,E),
where every edge $e \in E$
has a weight $w(e) > 0$.
Let $s, t \in V$ be two nodes.

Output: A ~~shortest~~ simple path from s to t.
longest

An instance:



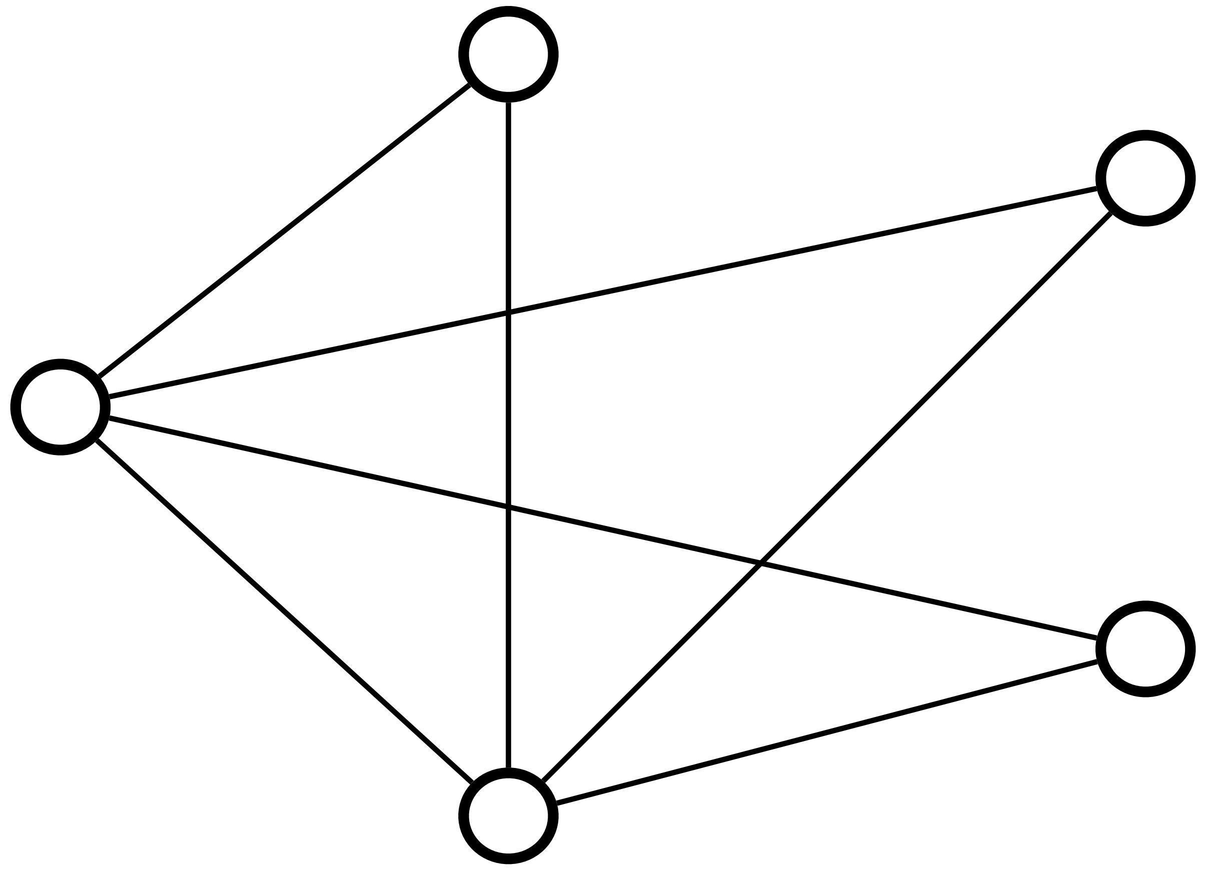
Can the Longest Path Problem be solved in polynomial time?

No one knows. But if you do ……

NP Completeness

Euler Tour:

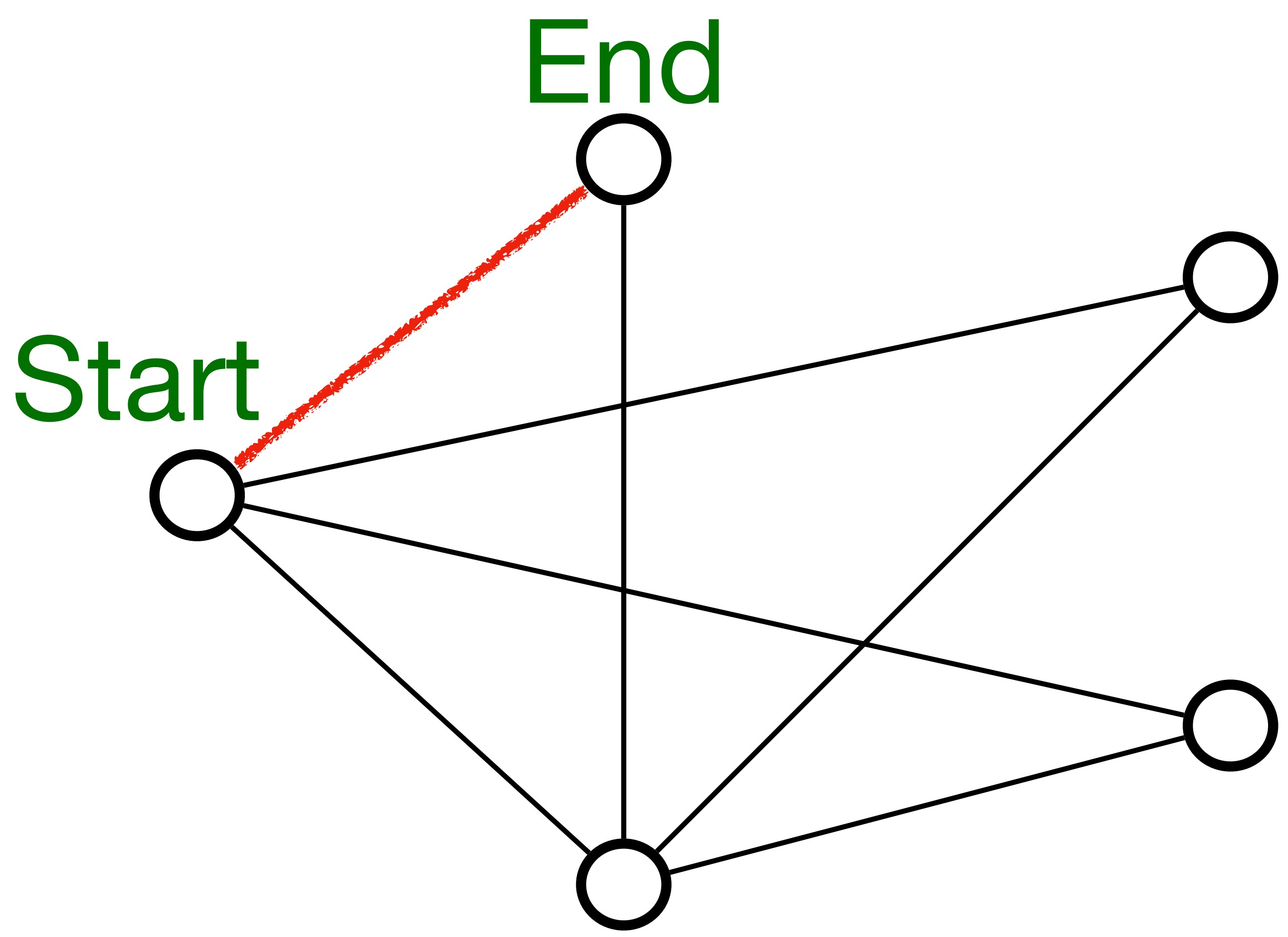
Given a graph G=(V,E), an Euler Tour is a cycle that passes every edge of G exactly once.



NP Completeness

Euler Tour:

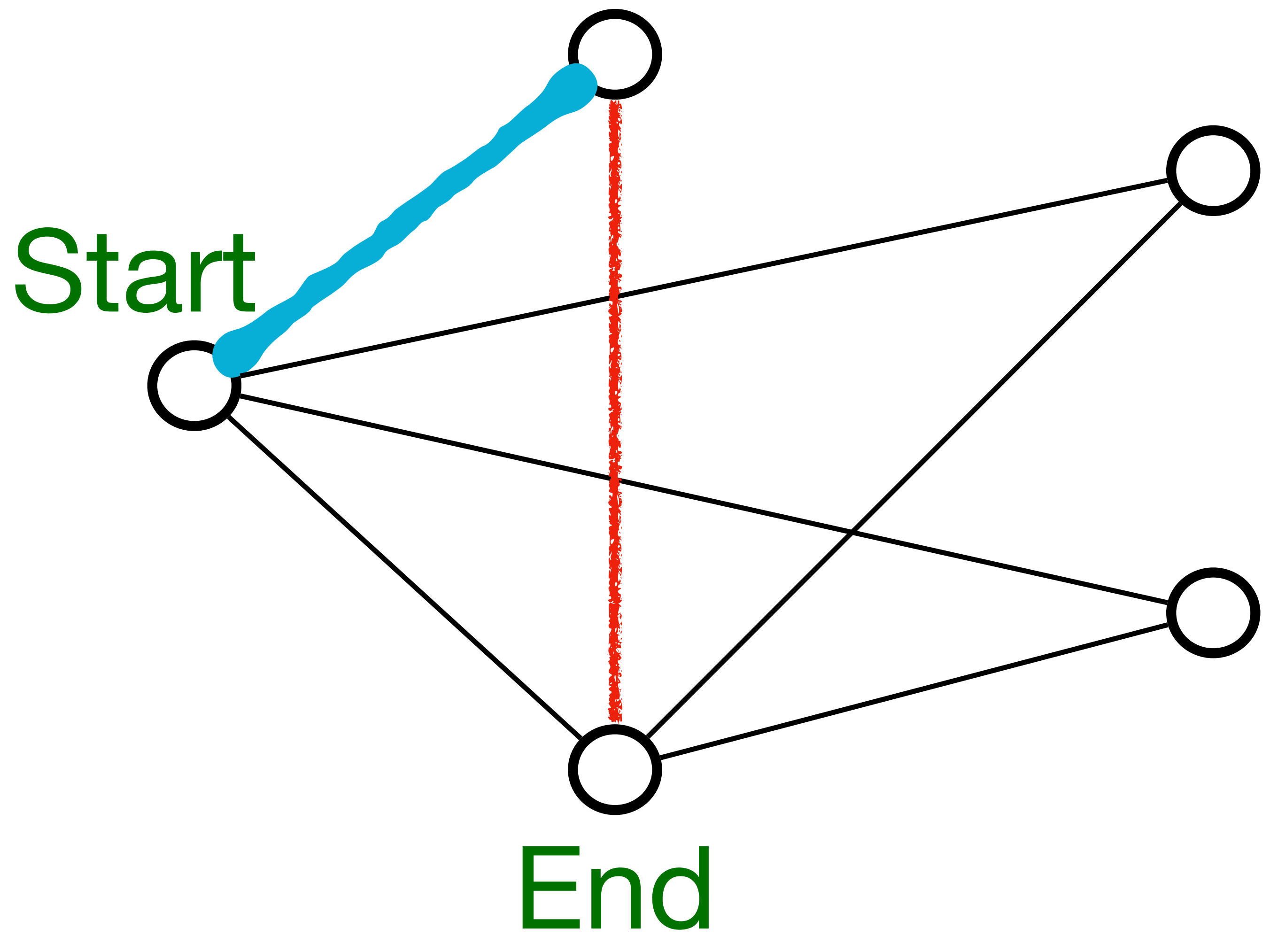
Given a graph G=(V,E), an Euler Tour is a cycle that passes every edge of G exactly once.



NP Completeness

Euler Tour:

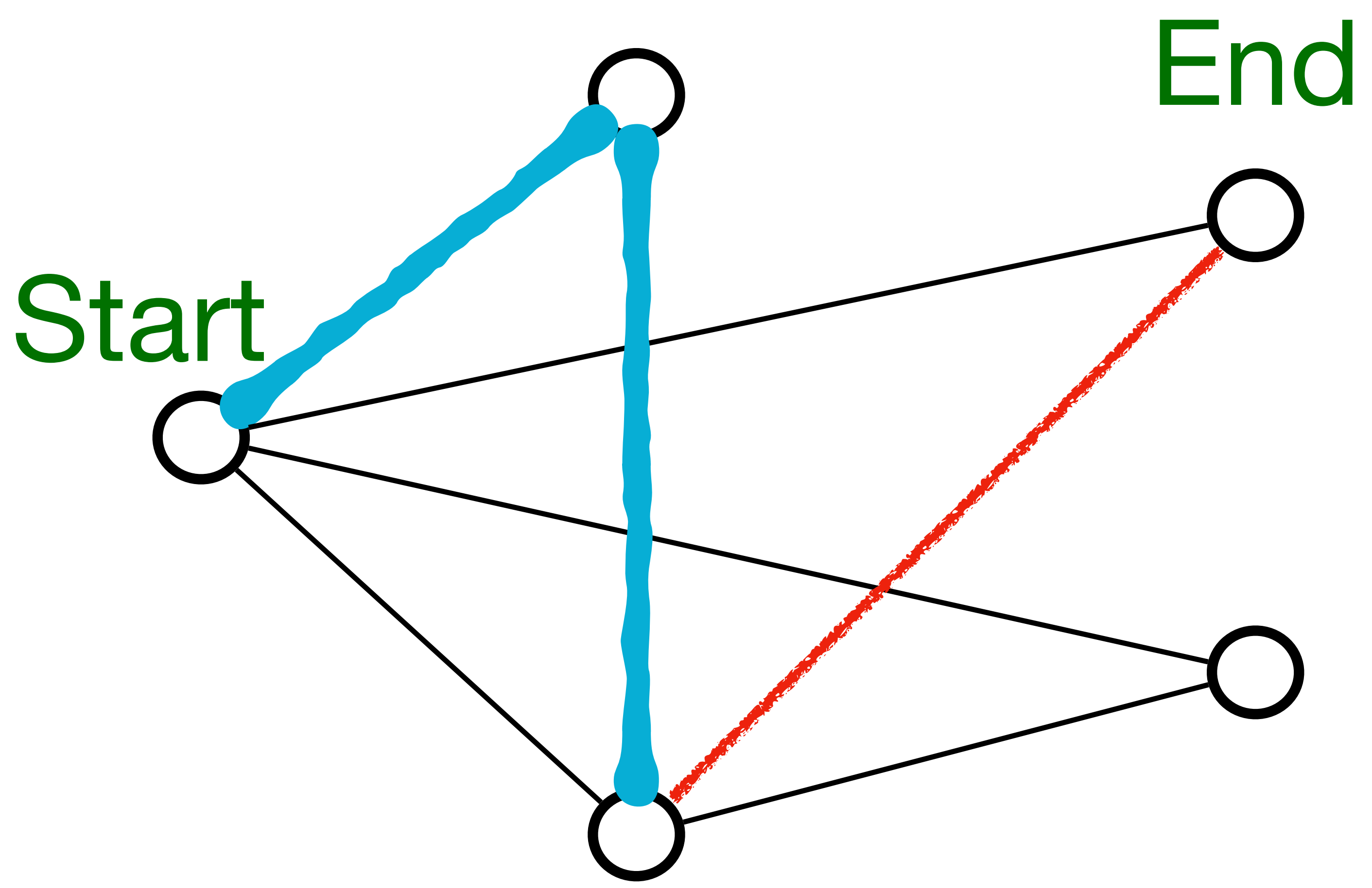
Given a graph G=(V,E), an Euler Tour is a cycle that passes every edge of G exactly once.



NP Completeness

Euler Tour:

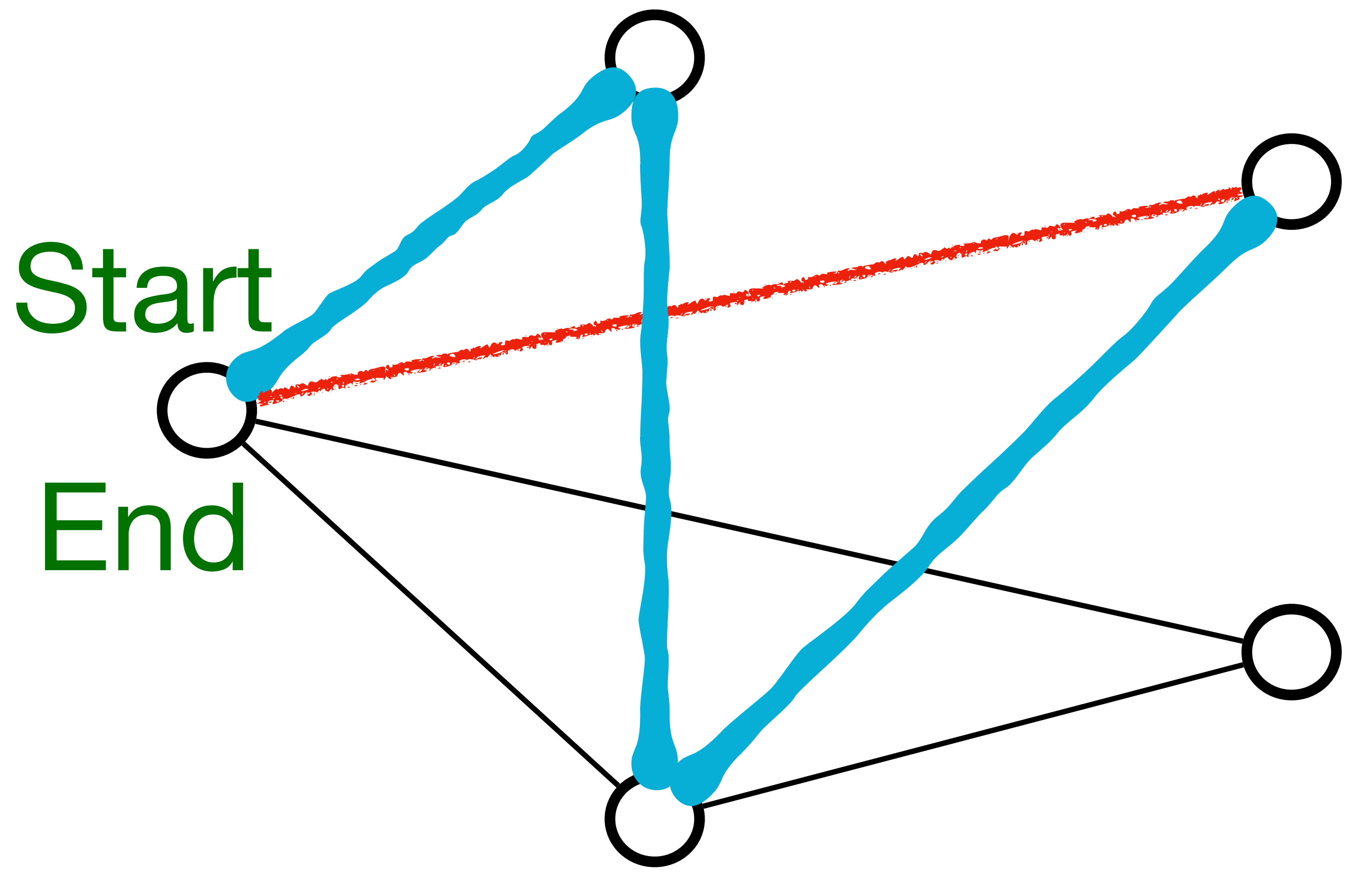
Given a graph G=(V,E), an Euler Tour is a cycle that passes every edge of G exactly once.



NP Completeness

Euler Tour:

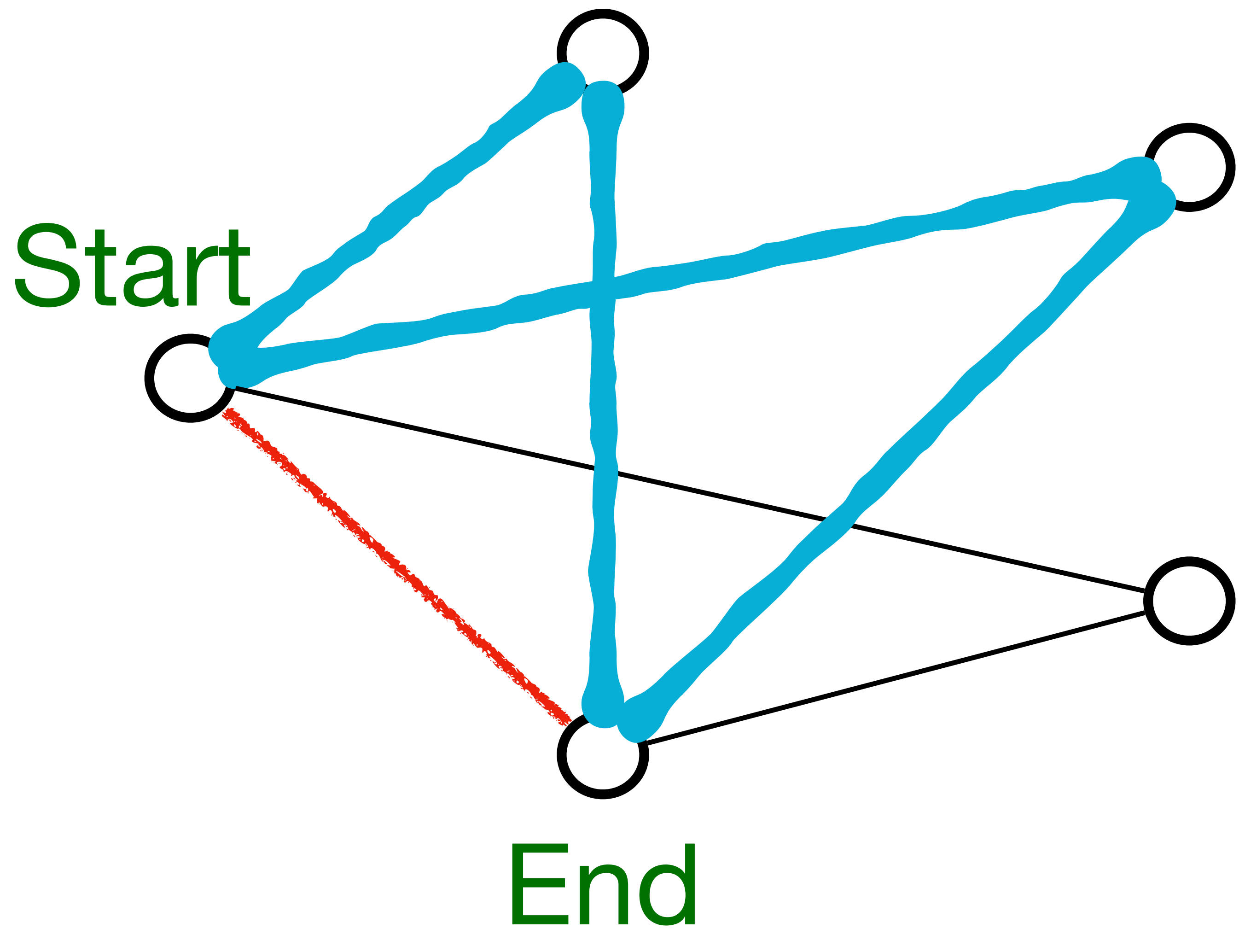
Given a graph G=(V,E), an Euler Tour is a cycle that passes every edge of G exactly once.



NP Completeness

Euler Tour:

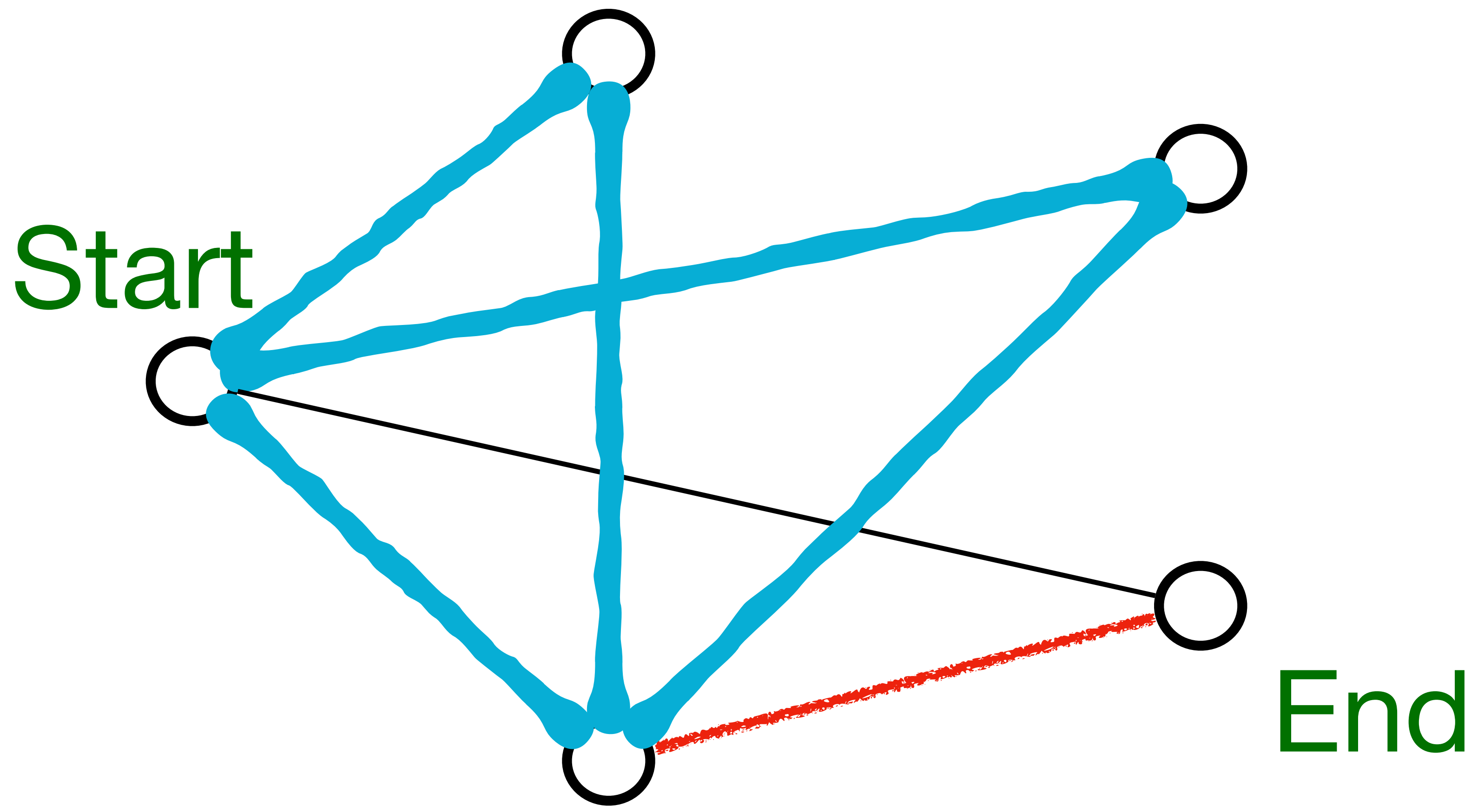
Given a graph G=(V,E), an Euler Tour is a cycle that passes every edge of G exactly once.



NP Completeness

Euler Tour:

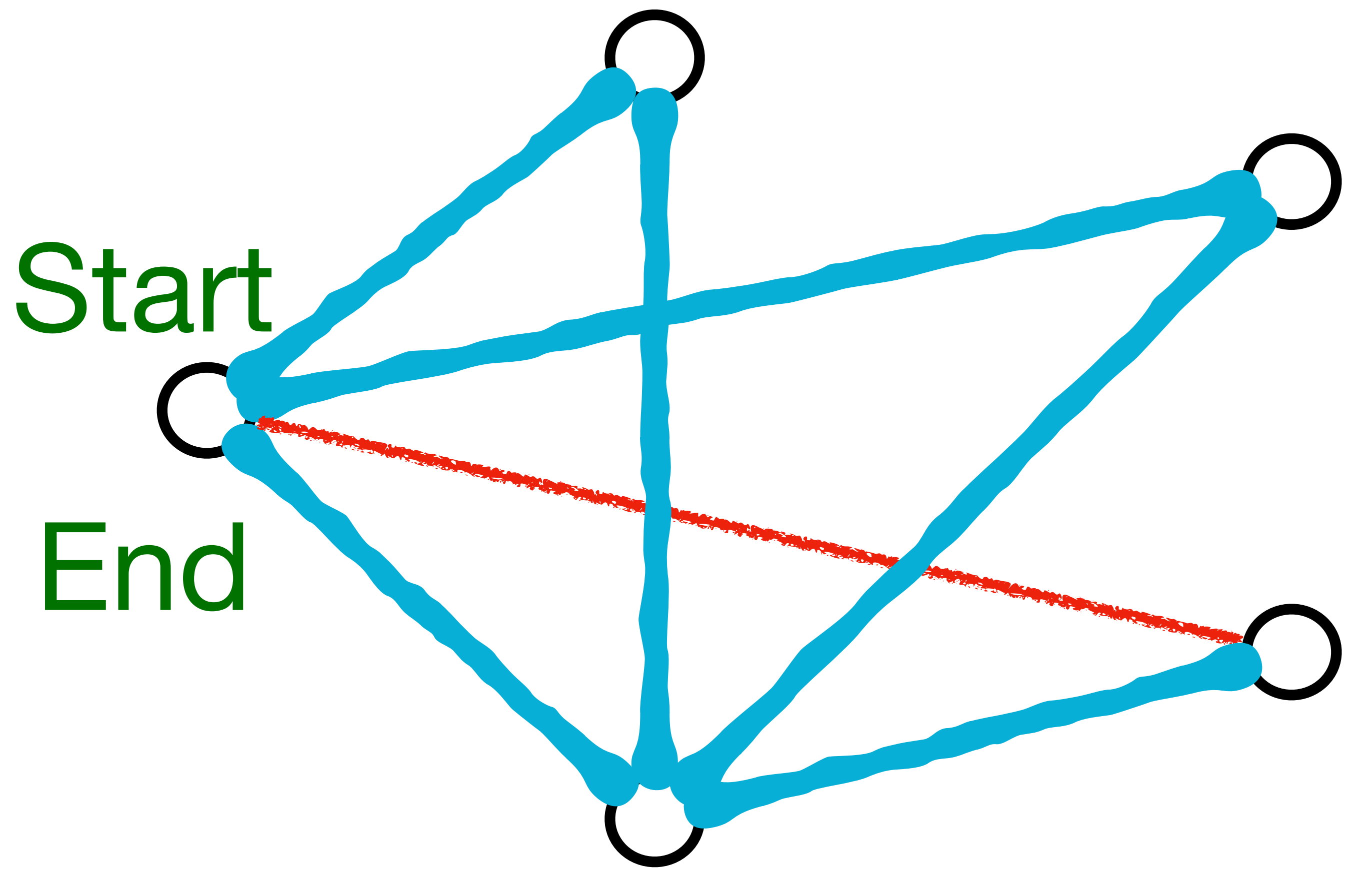
Given a graph G=(V,E), an Euler Tour is a cycle that passes every edge of G exactly once.



NP Completeness

Euler Tour:

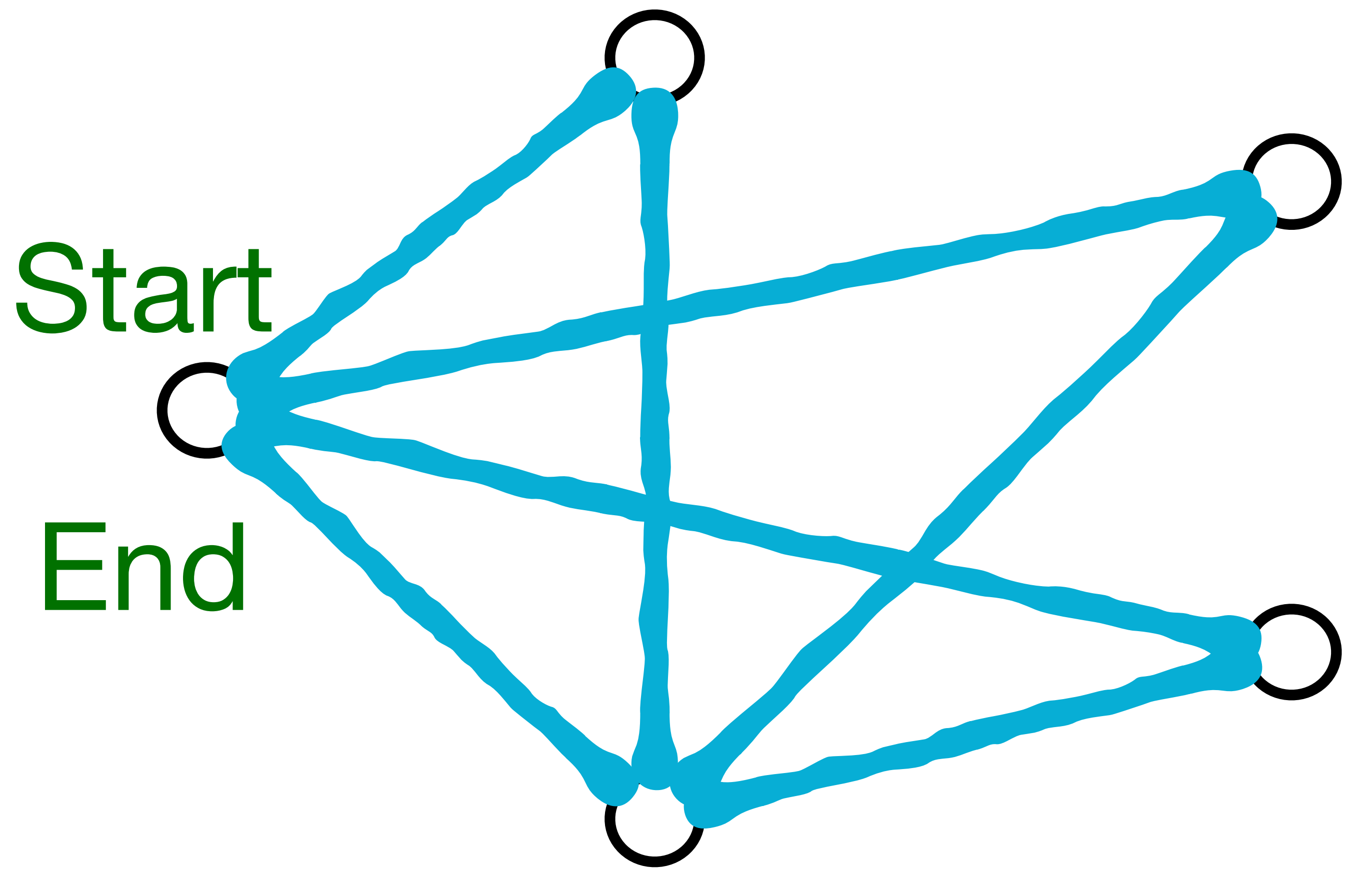
Given a graph G=(V,E), an Euler Tour is a cycle that passes every edge of G exactly once.



NP Completeness

Euler Tour:

Given a graph G=(V,E), an Euler Tour is a cycle that passes every edge of G exactly once.



Euler Tour Problem:

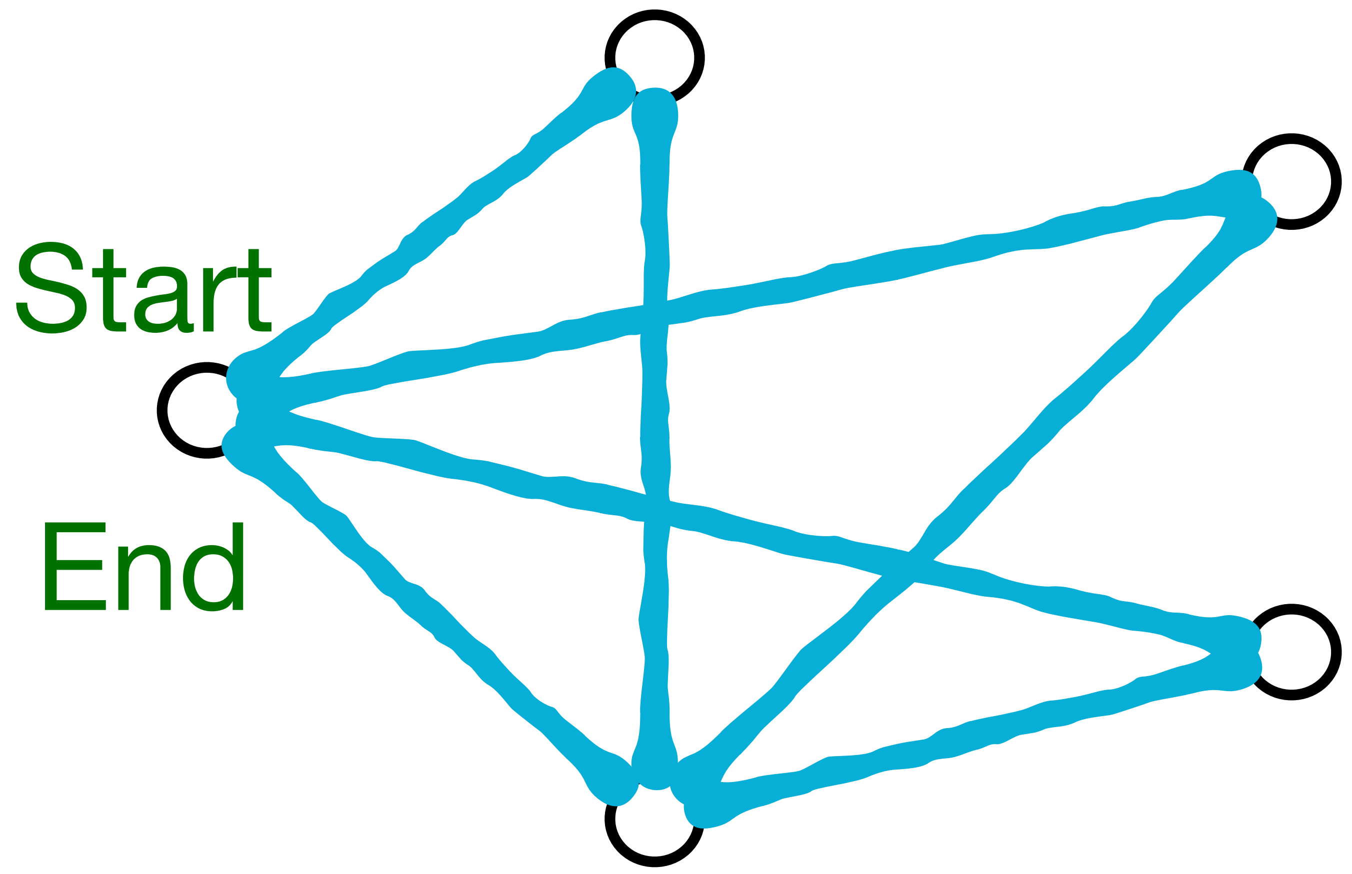
Input: A graph G=(V,E).

Question: Does G have an Euler Tour?

NP Completeness

Euler Tour:

Given a graph G=(V,E), an Euler Tour is a cycle that passes every edge of G exactly once.



Euler Tour Problem:

Input: A graph G=(V,E).

Question: Does G have an Euler Tour?

Theorem: G has an Euler Tour if and only if G is connected and all its nodes have even degrees.

So the Euler Tour Problem is polynomial-time solvable.

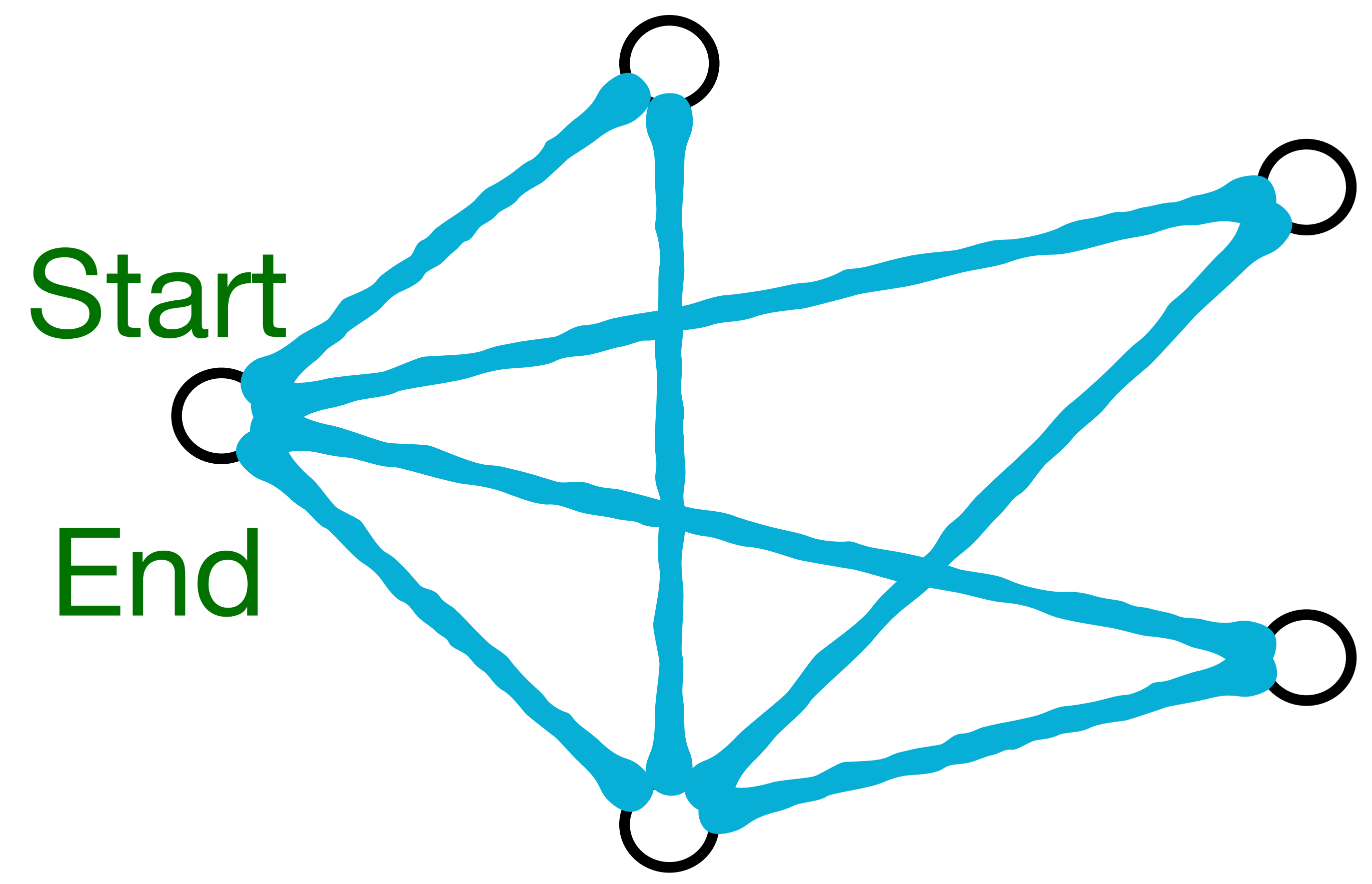
NP Completeness

How about we change "edge" to "node" below"

Hamiltonian cycle:

~~Euler Tour~~:

Given a graph G=(V,E), an ~~Euler Tour~~ Hamiltonian cycle is a cycle that passes every ~~edge~~ node of G exactly once.



Euler Tour Problem:

Input: A graph G=(V,E).

Question: Does G have an Euler Tour?

The Euler Tour Problem is polynomial-time solvable.

NP Completeness

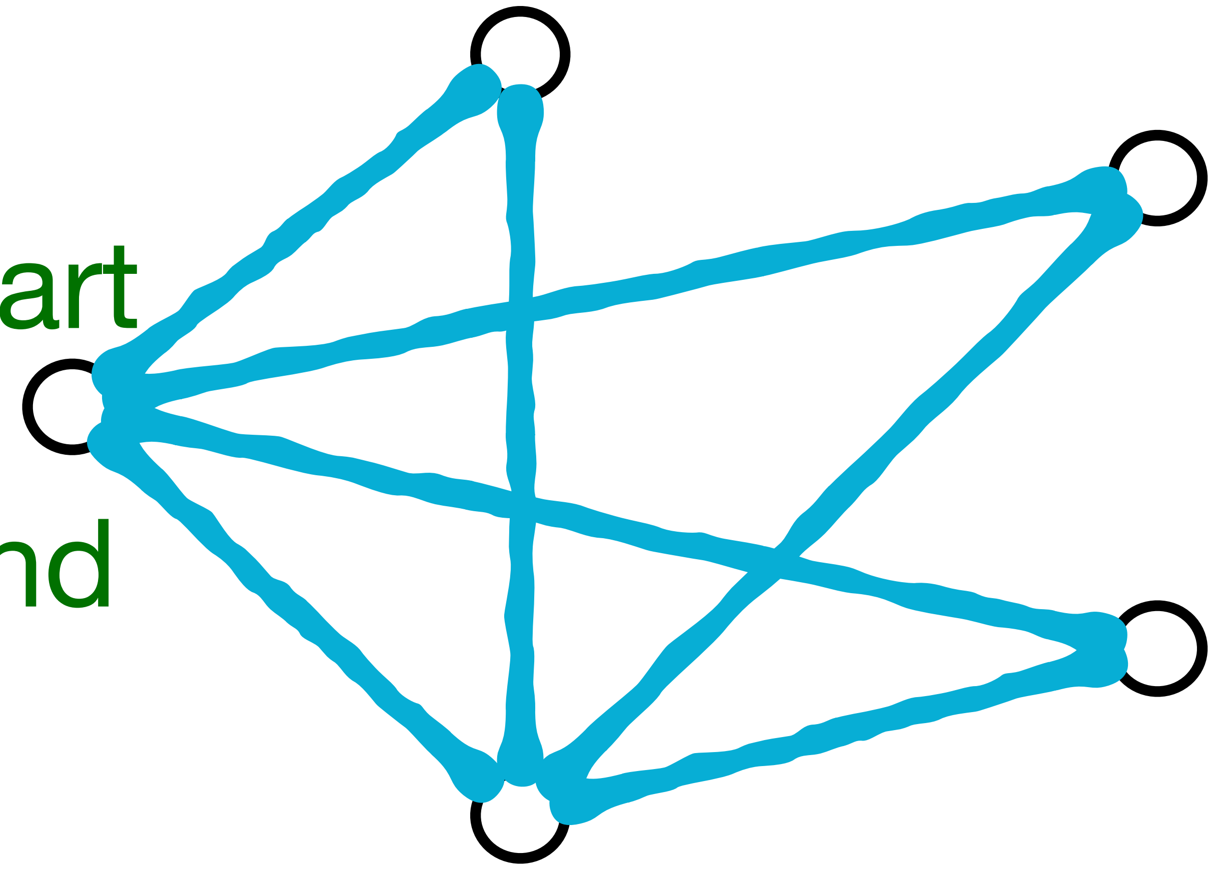
Hamiltonian cycle:

~~Euler Tour~~:

Hamiltonian cycle node

Given a graph G=(V,E), an ~~Euler Tour~~ is a cycle that passes every ~~edge~~ of G exactly once.

Start



End

Hamiltonian cycle

~~Euler Tour~~ Problem:

Input: A graph G=(V,E).

Question: Does G have an ~~Euler Tour~~?

Hamiltonian cycle

The Euler Tour Problem is polynomial-time solvable.

Is the Hamiltonian cycle Problem polynomial-time solvable?

No one knows.

NP Completeness

Boolean logic:

AND operation : $0 \wedge 0 = 0, \quad 0 \wedge 1 = 0, \quad 1 \wedge 0 = 0, \quad 1 \wedge 1 = 1$

OR operation : $0 \vee 0 = 0, \quad 0 \vee 1 = 1, \quad 1 \vee 0 = 1, \quad 1 \vee 1 = 1$

NOT operation : $\bar{0} = 1, \quad \bar{1} = 0$

Boolean variables: $x_1, x_2, \cdots, x_n \in \{0,1\}$

Boolean literal: $x_i, \bar{x}_i$

Boolean formula:

$$\underbrace{(x_1 \vee x_2 \vee x_3)}_{\text{clause}} \wedge \underbrace{(\bar{x}_1 \vee \bar{x}_2 \vee x_4)}_{\text{clause}} \wedge \underbrace{(x_1 \vee \bar{x}_4 \vee x_5)}_{\text{clause}} \wedge \underbrace{(x_2 \vee \bar{x}_3 \vee \bar{x}_5)}_{\text{clause}}$$

CNF: Conjunctive Normal Form

NP Completeness

2-CNF SAT Problem:

Input: A CNF formula with n variables and k clauses, where each clause is the "OR" of 2 literals.

Question: Does there exist a solution to the variables that make the formula be true?

Instance: $(x_1 \vee x_3) \wedge (\bar{x}_2 \vee x_4) \wedge (x_1 \vee x_5) \wedge (\bar{x}_3 \vee \bar{x}_5)$

n=5 variables   k=4 clauses

NP Completeness

2-CNF SAT Problem:

Input: A CNF formula with n variables and k clauses, where each clause is the "OR" of 2 literals.

Question: Does there exist a solution to the variables that make the formula be true?

Satisfied

Instance: $(x_1 \vee x_3) \wedge (\bar{x}_2 \vee x_4) \wedge (x_1 \vee x_5) \wedge (\bar{x}_3 \vee \bar{x}_5)$

A solution: $x_1 = 1,\ x_2 = 0,\ x_3 = 0,\ x_4 = 0,\ x_5 = 0$

2-CNF SAT Problem:

Input: A CNF formula with n variables and k clauses, where each clause is the "OR" of 2 literals.

Question: Does there exist a solution to the variables that make the formula be true?

Satisfied

Instance:

$$\overset{1}{(x_1} \vee \overset{0}{x_3)} \wedge \overset{1}{(\bar{x}_2} \vee \overset{0}{x_4)} \wedge \overset{1}{(x_1} \vee \overset{0}{x_5)} \wedge \overset{1}{(\bar{x}_3} \vee \overset{1}{\bar{x}_5)}$$

A solution: $x_1 = 1,\ x_2 = 0,\ x_3 = 0,\ x_4 = 0,\ x_5 = 0$

2-CNF SAT Problem:

Input: A CNF formula with n variables and k clauses, where each clause is the "OR" of 2 literals.

Question: Does there exist a solution to the variables that make the formula be true?

The 2-CNF SAT Problem can be solved in polynomial time.

Instance: $(x_1 \vee x_3) \wedge (\bar{x_2} \vee x_4) \wedge (x_1 \vee x_5) \wedge (\bar{x_3} \vee \bar{x_5})$

n=5 variables k=4 clauses

NP Completeness

3-CNF SAT Problem:

Input: A CNF formula with n variables and k clauses, where each clause is the "OR" of 3 literals.

Question: Does there exist a solution to the variables that make the formula be true?

The 2-CNF SAT Problem can be solved in polynomial time.

How about 3-CNF SAT Problem? No one knows. But if you do …

Instance: $(x_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_4) \wedge (x_1 \vee \bar{x}_4 \vee x_5) \wedge (x_2 \vee \bar{x}_3 \vee \bar{x}_5)$

clause clause clause clause

n=5 variables k=4 clauses

Quiz question:

1. In the examples above, which problems have known polynomial-time algorithms, and which ones do not?

Roadmap of this lecture:

1. NP Completeness

    1.1 Polynomial-time algorithms.

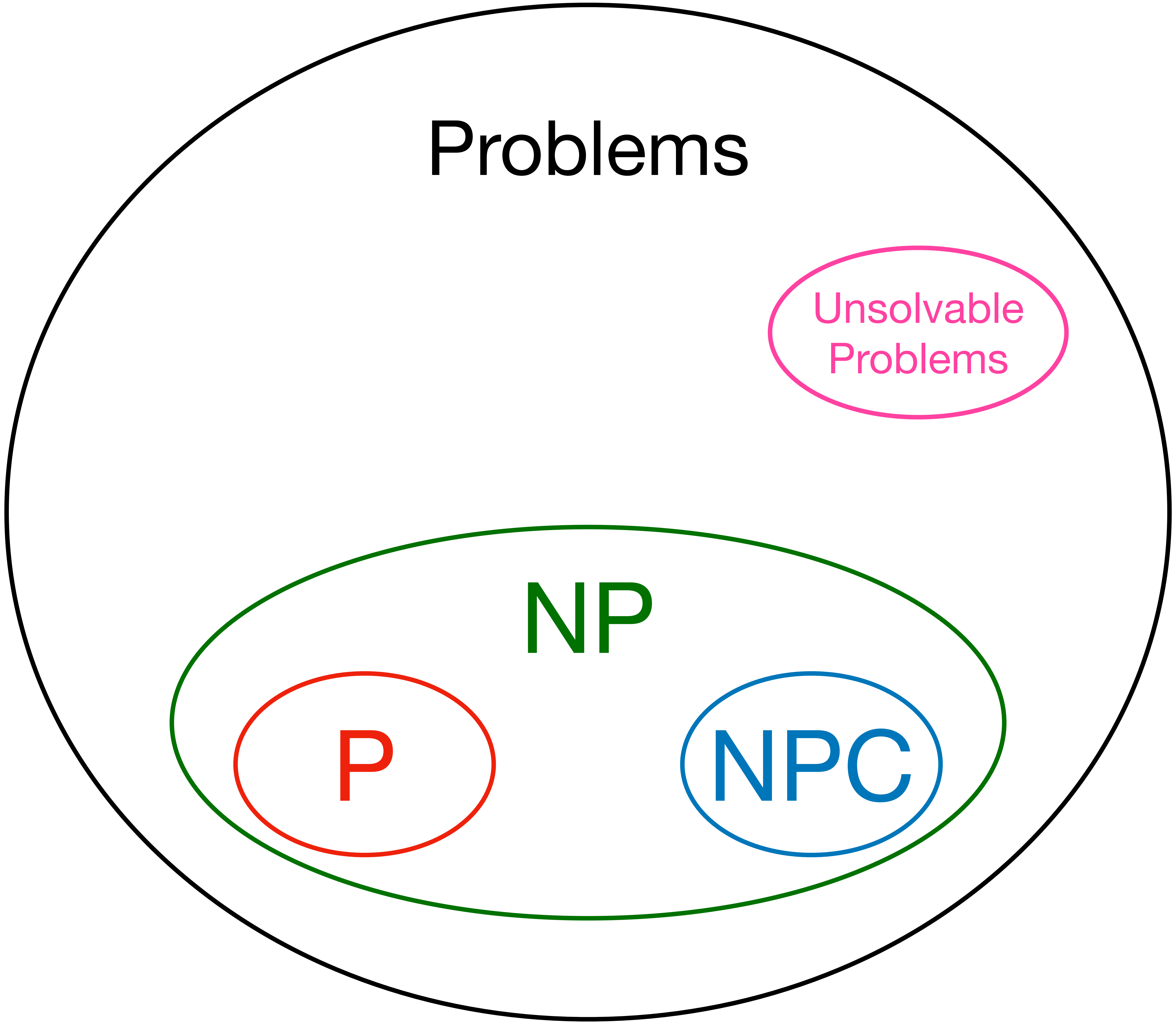
    1.2 Examples of NP-complete problems.

    1.3 P versus NP.

    1.4 Decision problems.

    1.5 Polynomial-time reduction.

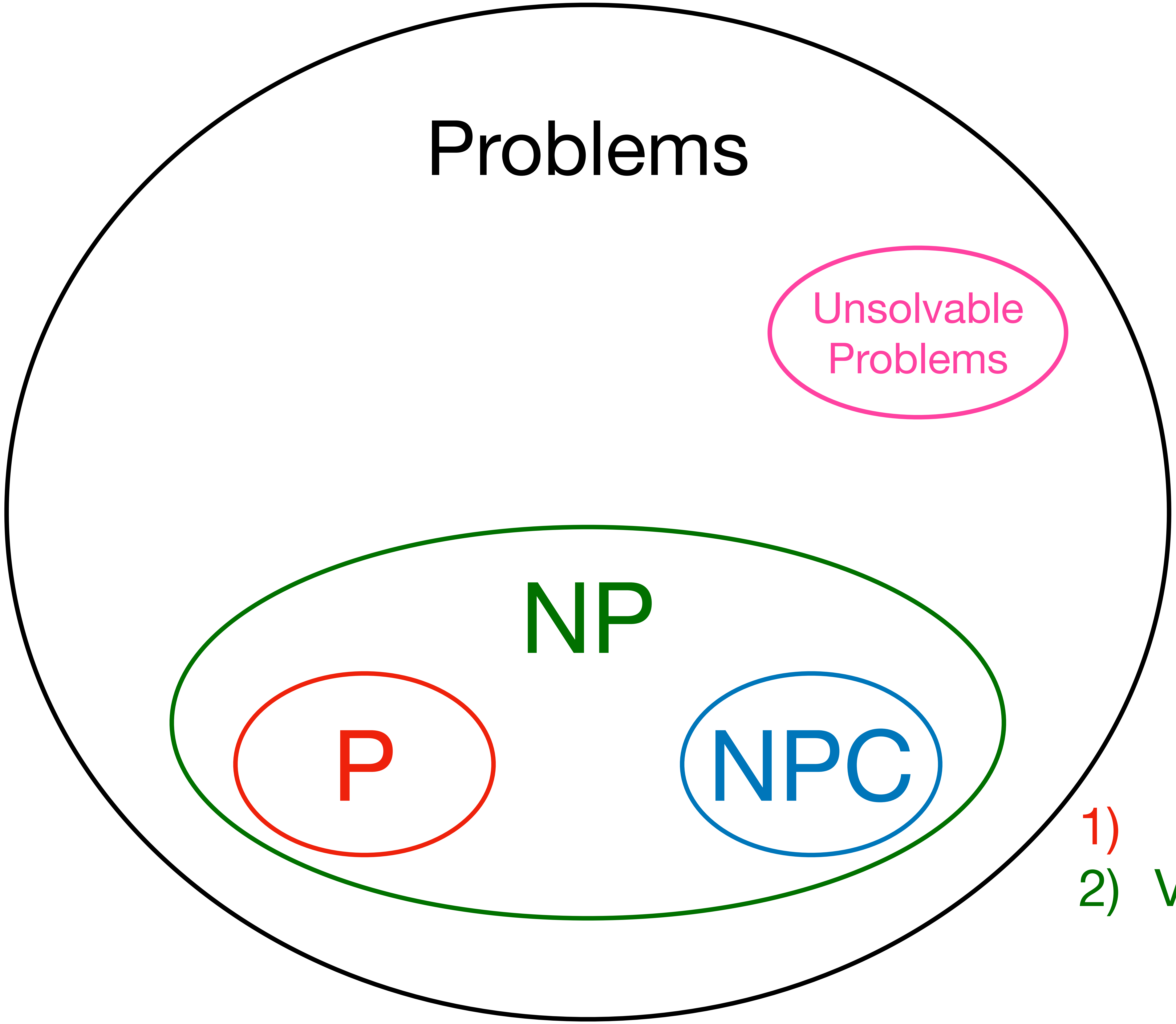
NP Completeness



P: the set of all problems
that can be solved in
polynomial time.

NP: the set of all problems
with this property:
it takes polynomial time
to verify the correctness
of the solution.

NP Completeness



P: the set of all problems that can be solved in polynomial time.

NP: the set of all problems with this property: it takes polynomial time to verify the correctness of the solution.

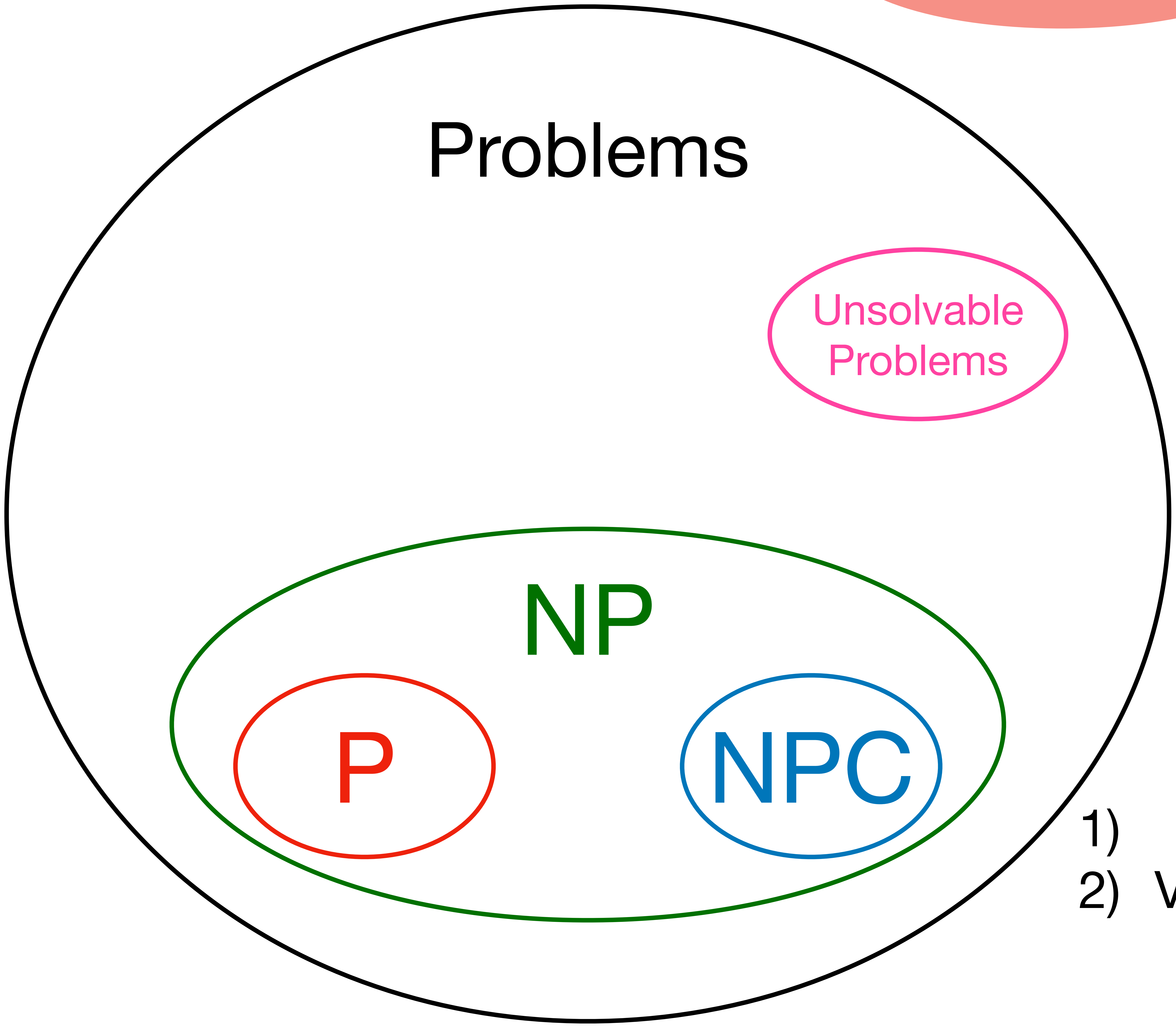
Two elements of an algorithm:

1) The process of finding a solution.
2) Verify the correctness of the solution (so it knows when to end).

NP Completeness

$$P \subseteq NP$$

$$P \overset{?}{\neq} NP$$



P: the set of all problems that can be solved in polynomial time.

NP: the set of all problems with this property: it takes polynomial time to verify the correctness of the solution.

Two elements of an algorithm:

1) The process of finding a solution.
2) Verify the correctness of the solution (so it knows when to end).

NP Completeness

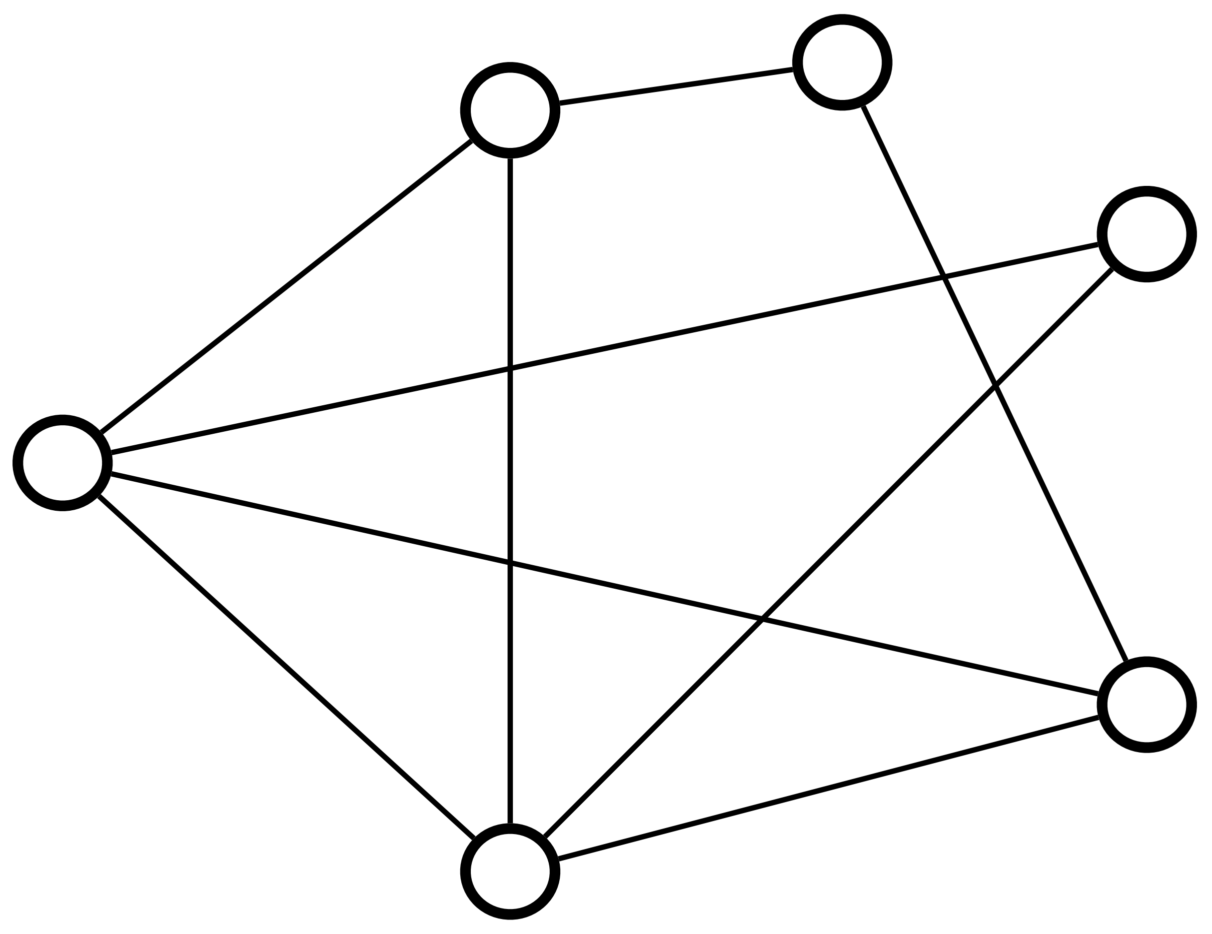
$$P \subseteq NP$$

$$P \overset{?}{=} NP$$

Hamiltonian cycle Problem:

Input: A graph G=(V,E).

Question: Does G have a Hamiltonian cycle?



Finding a Hamiltonian cycle might be hard ……

P: the set of all problems that can be solved in polynomial time.

NP: the set of all problems with this property: it takes polynomial time to verify the correctness of the solution.

Two elements of an algorithm:

1) The process of finding a solution.
2) Verify the correctness of the solution (so it knows when to end).

NP Completeness

$$P \subseteq NP$$

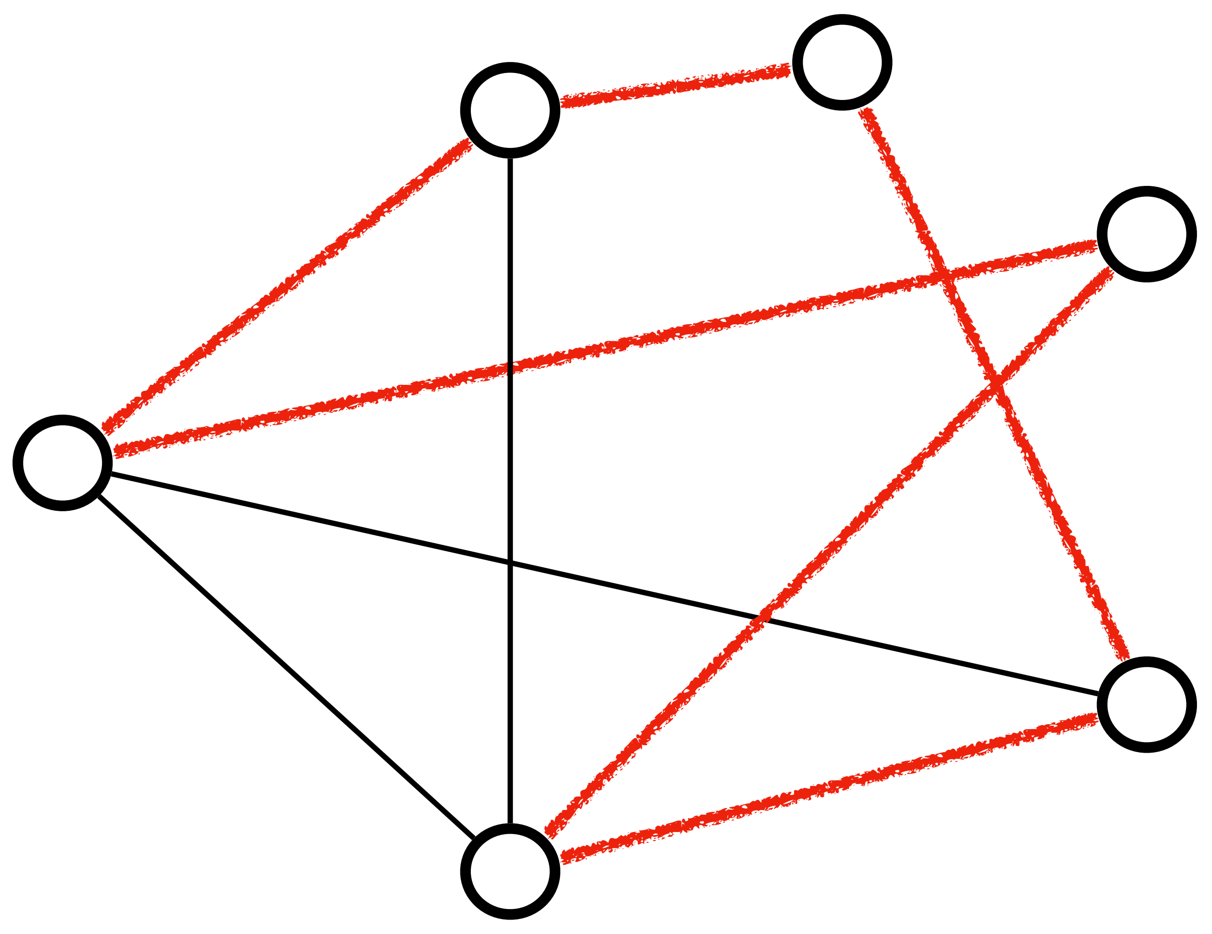
$$P \overset{?}{=} NP$$

Hamiltonian cycle Problem:

Input: A graph G=(V,E).

Question: Does G have a Hamiltonian cycle?

P: the set of all problems that can be solved in polynomial time.



Finding a Hamiltonian cycle might be hard ……

But given a Hamiltonian cycle, it is easy to verify it is indeed A Hamiltonian cycle.

NP: the set of all problems with this property: it takes polynomial time to verify the correctness of the solution.

Two elements of an algorithm:

1) The process of finding a solution.
2) Verify the correctness of the solution (so it knows when to end).

Quiz questions:

1. What is the difference between P and NP, according to their definitions?

2. Do we know if there is really any difference between P and NP?

Roadmap of this lecture:

1. NP Completeness

   1.1 Polynomial-time algorithms.

   1.2 Examples of NP-complete problems.

   1.3 P versus NP.

   1.4 Decision problems.

   1.5 Polynomial-time reduction.

NP Completeness

Optimization Problem: A problem where we look for a solution whose objective value is maximized or minimized.

NP Completeness

Optimization Problem: A problem where we look for a solution whose objective value is maximized or minimized.

Decision Problem: A problem that asks a Yes/No question.

NP Completeness

Optimization Problem: A problem where we look for a solution whose objective value is maximized or minimized.

Decision Problem: A problem that asks a Yes/No question.

Optimization Problem

Shortest-Path Problem

Input: A directed graph G=(V,E), where every edge $e \in E$ has a weight $w(e)$. Let $s, t \in V$ be two nodes.

Output: A shortest path from s to t.

Decision Problem

Shortest-Path Problem

Input: A directed graph G=(V,E), where every edge $e \in E$ has a weight $w(e)$. Let $s, t \in V$ be two nodes. Let k be a real number.

Question: Is there a path from s to t whose length is at most k?

NP Completeness

Optimization Problem: A problem where we look for a solution whose objective value is maximized or minimized.

Decision Problem: A problem that asks a Yes/No question.

Optimization Problem

Shortest-Path Problem

Input: A directed graph G=(V,E), where every edge $e \in E$ has a weight $w(e)$. Let $s, t \in V$ be two nodes.

Output: A shortest path from s to t.

Find the shortest path.

Decision Problem

Is there a sufficiently short path?

Shortest-Path Problem

Input: A directed graph G=(V,E), where every edge $e \in E$ has a weight $w(e)$. Let $s, t \in V$ be two nodes. Let k be a real number.

Question: Is there a path from s to t whose length is at most k?

NP Completeness

Optimization Problem: A problem where we look for a solution whose objective value is maximized or minimized.

Decision Problem: A problem that asks a Yes/No question.

Optimization Problem

Shortest-Path Problem

Input: A directed graph G=(V,E), where every edge $e \in E$ has a weight $w(e)$. Let $s, t \in V$ be two nodes.

Output: A shortest path from s to t.

Find the best solution.

Decision Problem

Is there a sufficiently good solution?

Shortest-Path Problem

Input: A directed graph G=(V,E), where every edge $e \in E$ has a weight $w(e)$. Let $s, t \in V$ be two nodes. Let k be a real number.

Question: Is there a path from s to t whose length is at most k?

NP Completeness

Optimization Problem: A problem where we look for a solution whose objective value is maximized or minimized.

Decision Problem: A problem that asks a Yes/No question.

Optimization Problem

Longest-Path Problem

Input: A directed graph G=(V,E), where every edge $e \in E$ has a weight $w(e)$. Let $s, t \in V$ be two nodes.

Output: A longest simple path from s to t.

Decision Problem

Longest-Path Problem

Input: A directed graph G=(V,E), where every edge $e \in E$ has a weight $w(e)$. Let $s, t \in V$ be two nodes. Let k be a real number.

Question: Is there a path from s to t whose length is at least k?

NP Completeness

Optimization Problem: A problem where we look for a solution whose objective value is maximized or minimized.

Decision Problem: A problem that asks a Yes/No question.

Optimization Problem

Longest-Path Problem

Input: A directed graph G=(V,E), where every edge $e \in E$ has a weight $w(e)$. Let $s, t \in V$ be two nodes.

Output: A longest simple path from s to t.

Find the longest path.

Decision Problem

Is there a sufficiently long path?

Longest-Path Problem

Input: A directed graph G=(V,E), where every edge $e \in E$ has a weight $w(e)$. Let $s, t \in V$ be two nodes. Let k be a real number.

Question: Is there a path from s to t whose length is at least k?

NP Completeness

Optimization Problem: A problem where we look for a solution whose objective value is maximized or minimized.

Decision Problem: A problem that asks a Yes/No question.

Optimization Problem

Longest-Path Problem

Input: A directed graph G=(V,E), where every edge $e \in E$ has a weight $w(e)$. Let $s, t \in V$ be two nodes.

Output: A longest simple path from s to t.

Find the best solution.

Decision Problem

Is there a sufficiently good solution?

Longest-Path Problem

Input: A directed graph G=(V,E), where every edge $e \in E$ has a weight $w(e)$. Let $s, t \in V$ be two nodes. Let k be a real number.

Question: Is there a path from s to t whose length is at least k?

NP Completeness

Optimization Problem: A problem where we look for a solution whose objective value is maximized or minimized.

Decision Problem: A problem that asks a Yes/No question.

Given an optimization problem, there is a corresponding decision problem.

NP Completeness

Optimization Problem: A problem where we look for a solution whose objective value is maximized or minimized.

Decision Problem: A problem that asks a Yes/No question.

Given an optimization problem, there is a corresponding decision problem.

If an optimization problem can be solved in polynomial time, then the corresponding decision problem can also be solved in polynomial time.

NP Completeness

Optimization Problem: A problem where we look for a solution whose objective value is maximized or minimized.

Decision Problem: A problem that asks a Yes/No question.

Given an optimization problem, there is a corresponding decision problem.

If an optimization problem can be solved in polynomial time, then the corresponding decision problem can also be solved in polynomial time.

If an optimization problem is “easy", then the corresponding decision problem is also “easy”.

NP Completeness

Optimization Problem: A problem where we look for a solution whose objective value is maximized or minimized.

Decision Problem: A problem that asks a Yes/No question.

Given an optimization problem, there is a corresponding decision problem.

If an optimization problem can be solved in polynomial time, then the corresponding decision problem can also be solved in polynomial time.

If an optimization problem is “easy", then the corresponding decision problem is also “easy”.

If a decision problem is “hard”, then the corresponding optimization problem is also “hard”.

From now on, we focus on “Decision Problems” only.

Quiz questions:

1. What is the difference between optimization problems and decision problems?
2. If a decision problem can be solved in polynomial time, can its corresponding optimization problem also be solved in polynomial time?

Roadmap of this lecture:

1. NP Completeness

   1.1 Polynomial-time algorithms.

   1.2 Examples of NP-complete problems.

   1.3 P versus NP.

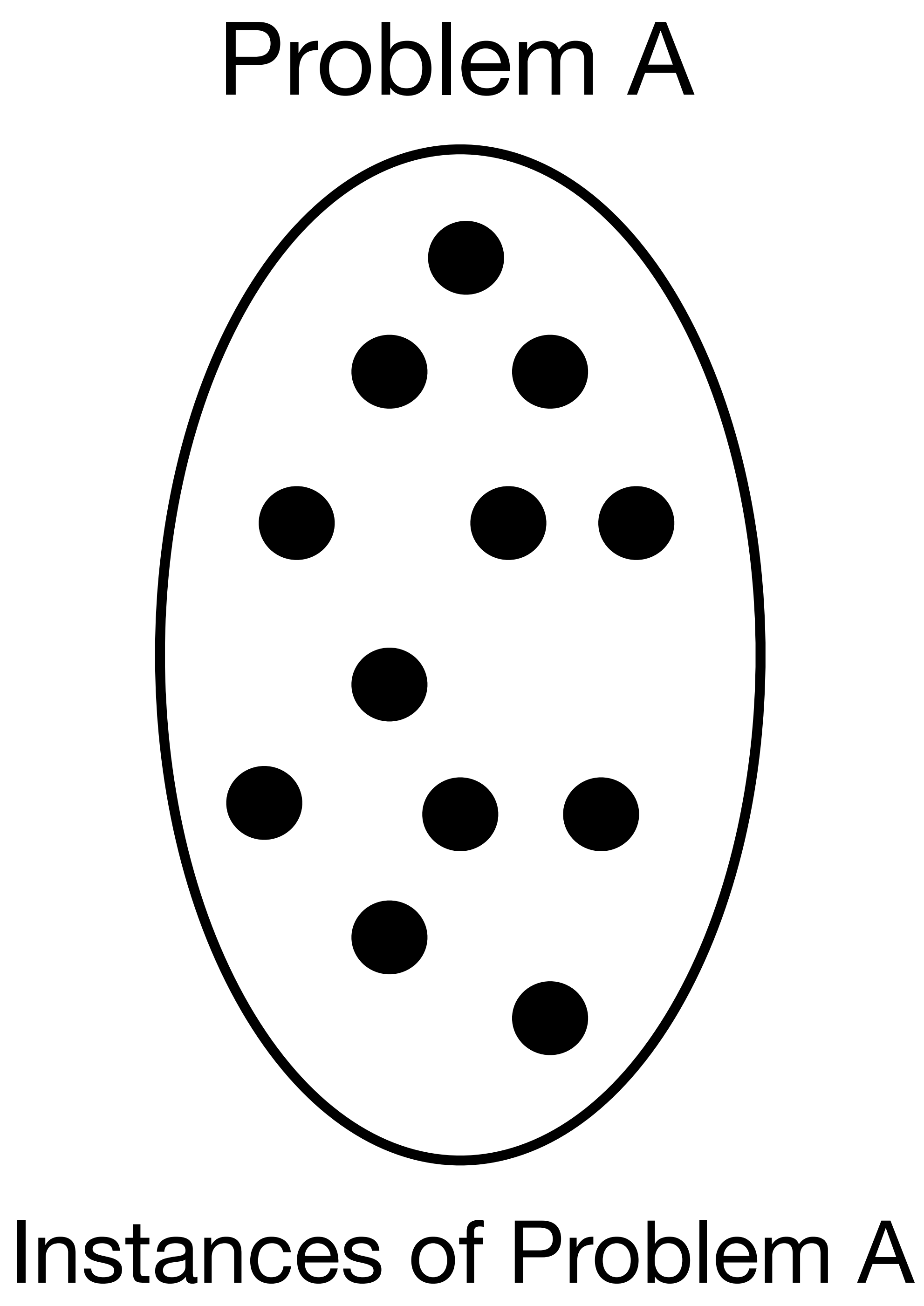
   1.4 Decision problems.

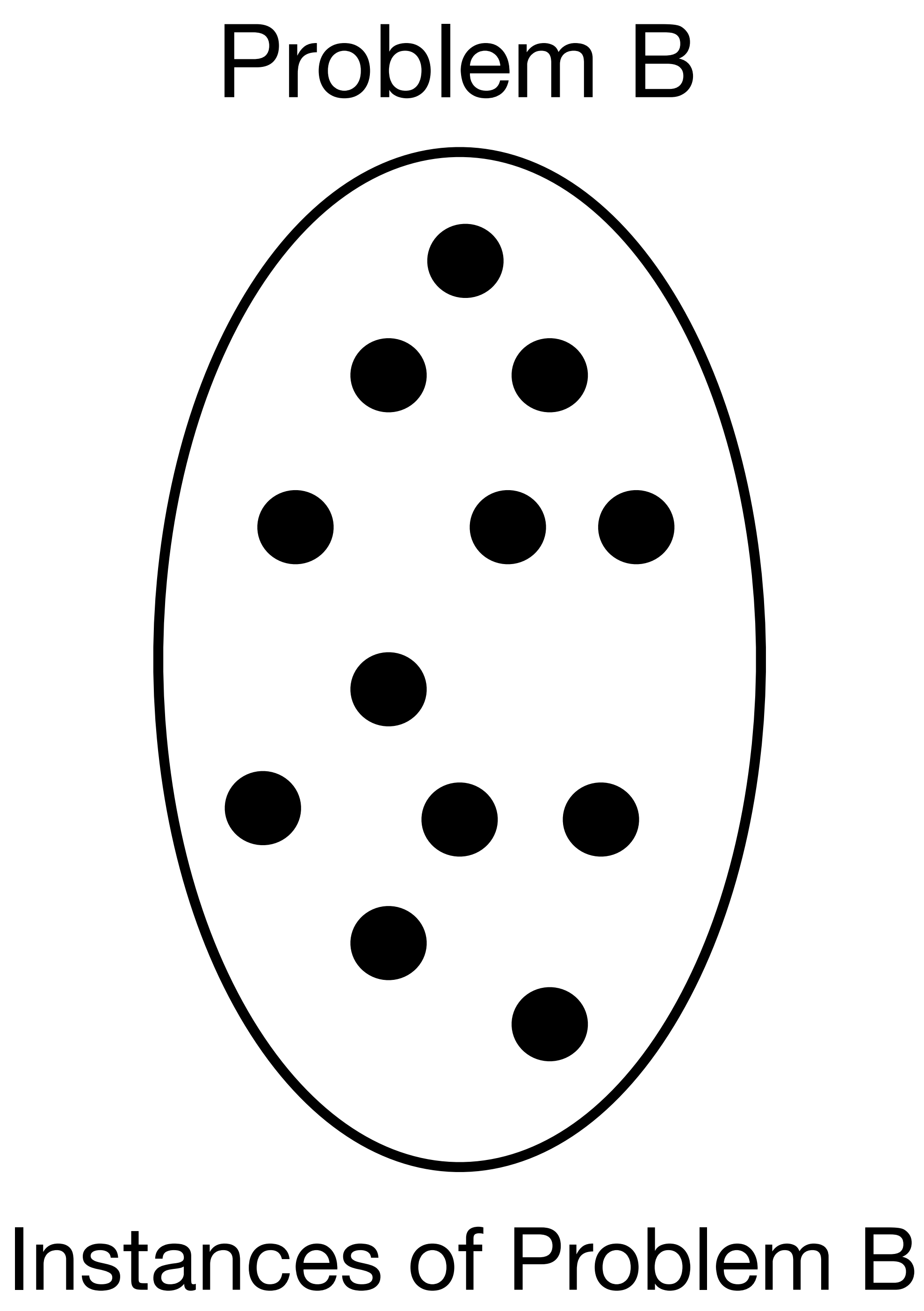
   1.5 Polynomial-time reduction.

NP Completeness

Decision Problem: A problem that asks a Yes/No question.

Reduction

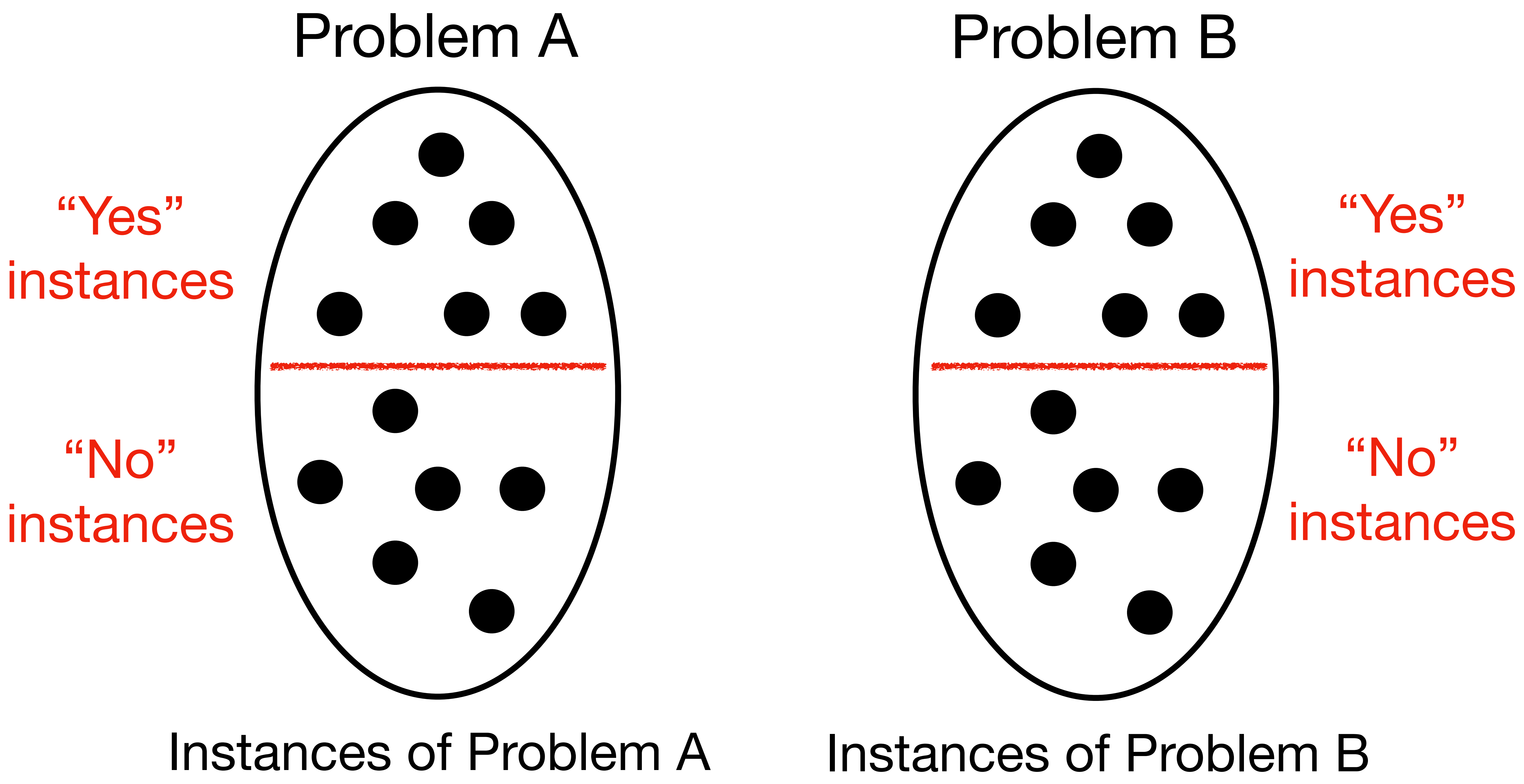




NP Completeness

Decision Problem: A problem that asks a Yes/No question.

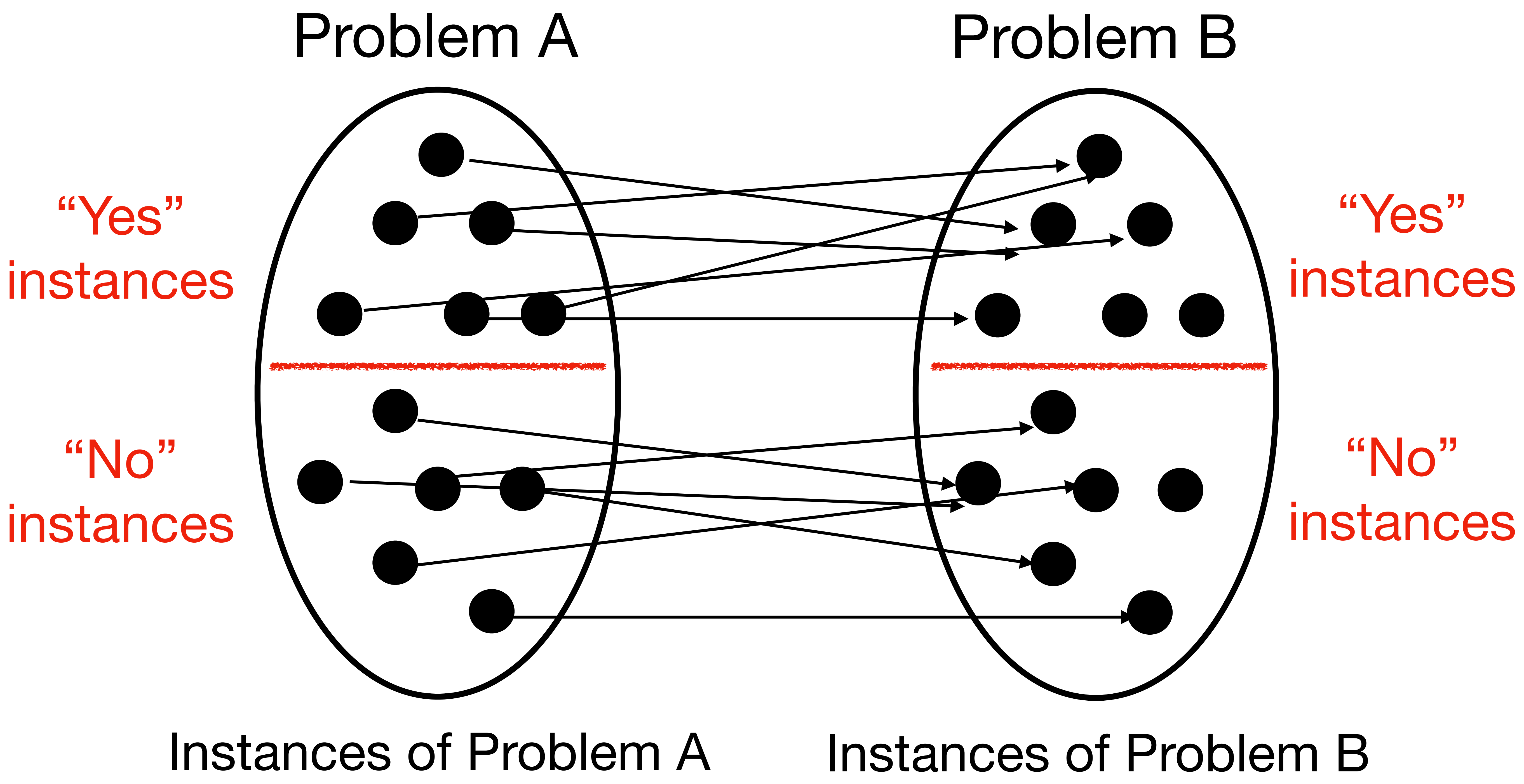
Reduction



NP Completeness

Decision Problem: A problem that asks a Yes/No question.

Reduction



The mapping is from A to B (not B to A)

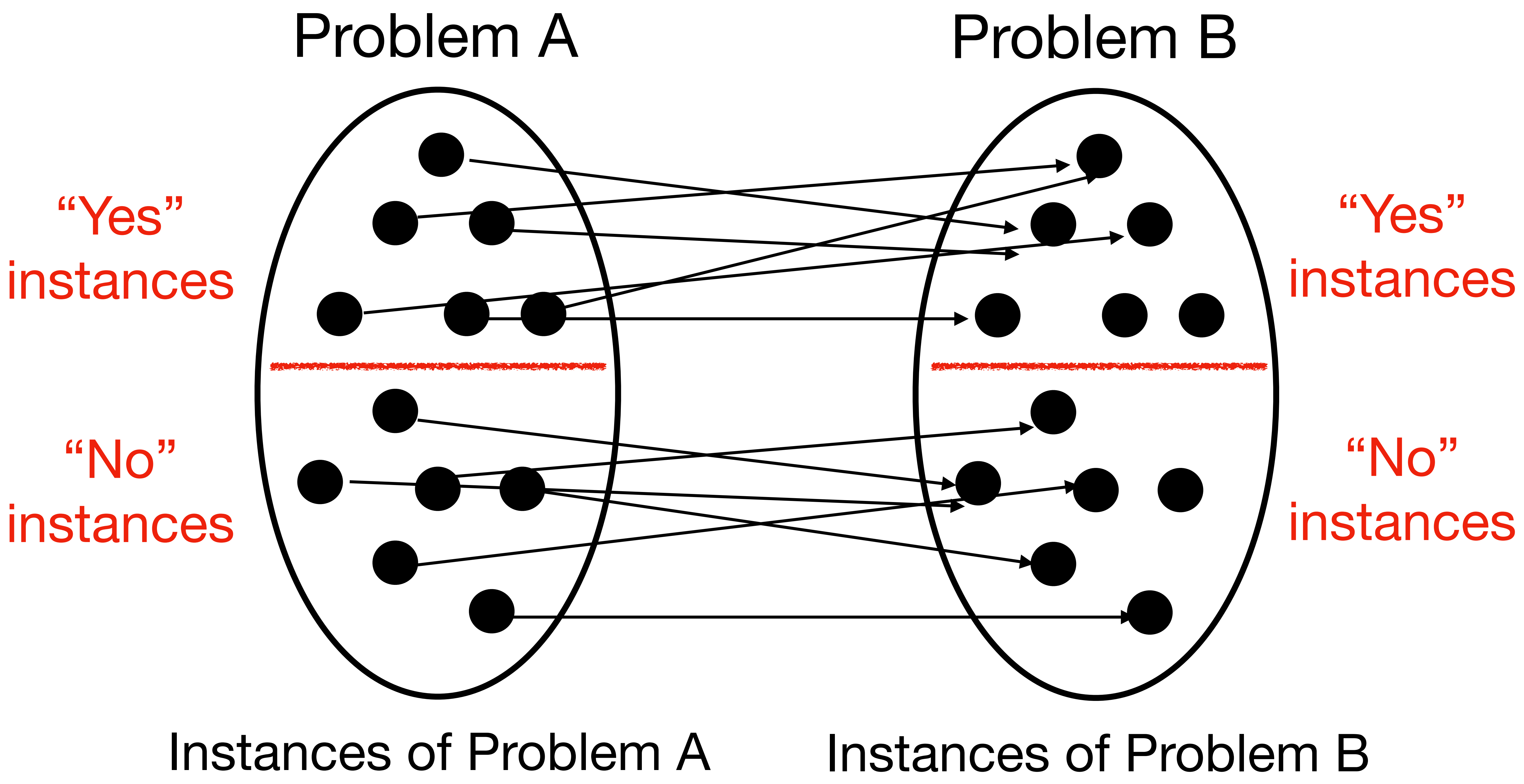
The mapping preserves the "Yes/No" answer.

NP Completeness

Decision Problem: A problem that asks a Yes/No question.

Reduction: If there is a mapping from the instances of Problem A to the instances of Problem B, such that every "Yes" instance of Problem A is mapped to a "Yes" instance of Problem B, and every "No" instance of Problem A is mapped to a "No" instance of Problem B, then the mapping is called a "Reduction" from Problem A to Problem B.

We can also say "Problem A is reduced to Problem B".



The mapping is from A to B (not B to A)

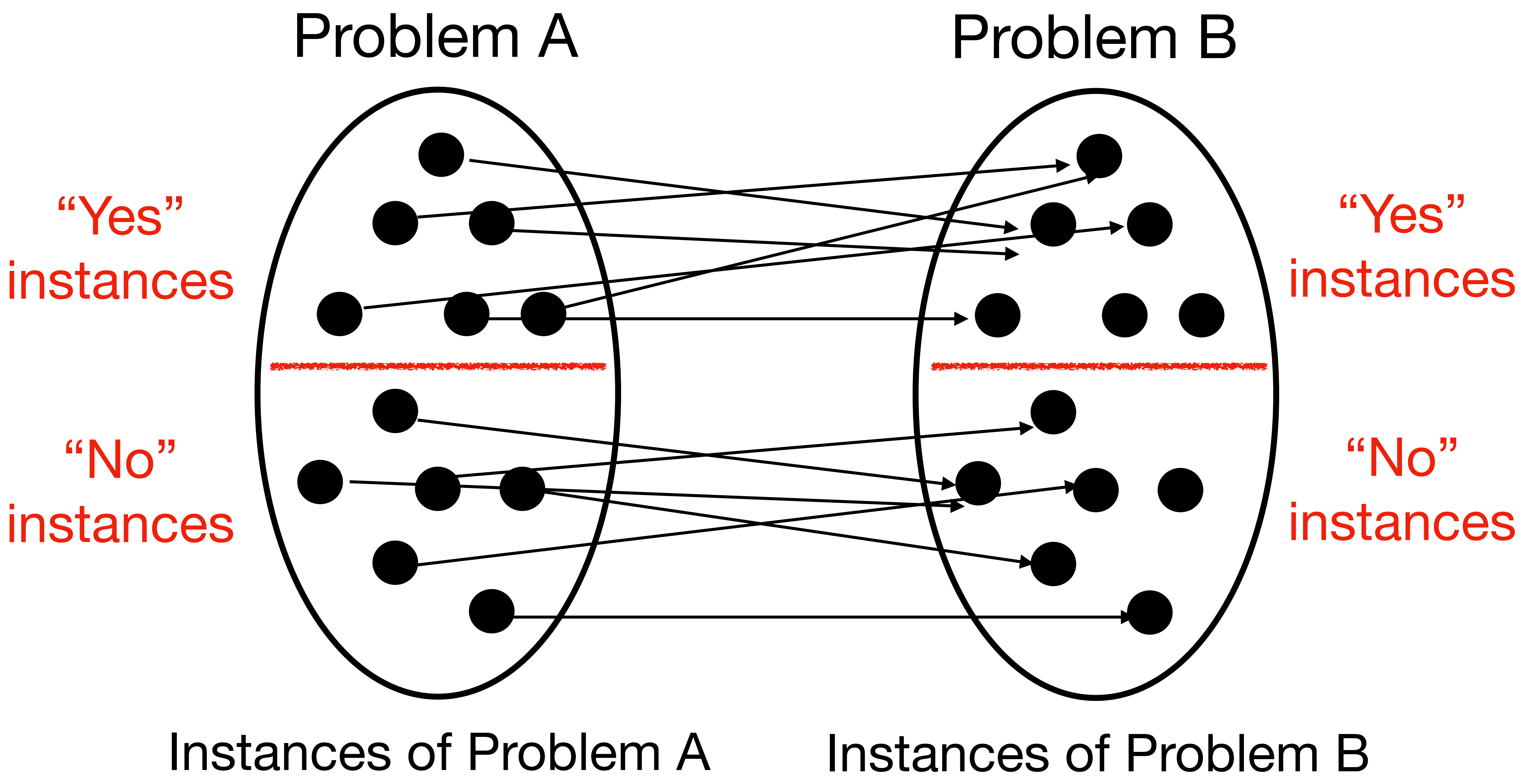
The mapping preserves the "Yes/No" answer.

NP Completeness

Example:

Problem A: Does $ax + b = 0$ have an integer solution?

Problem B: Does $ax^2 + bx + c = 0$ have an integer solution?



The mapping is from A to B (not B to A)

The mapping preserves the "Yes/No" answer.

NP Completeness

Example:

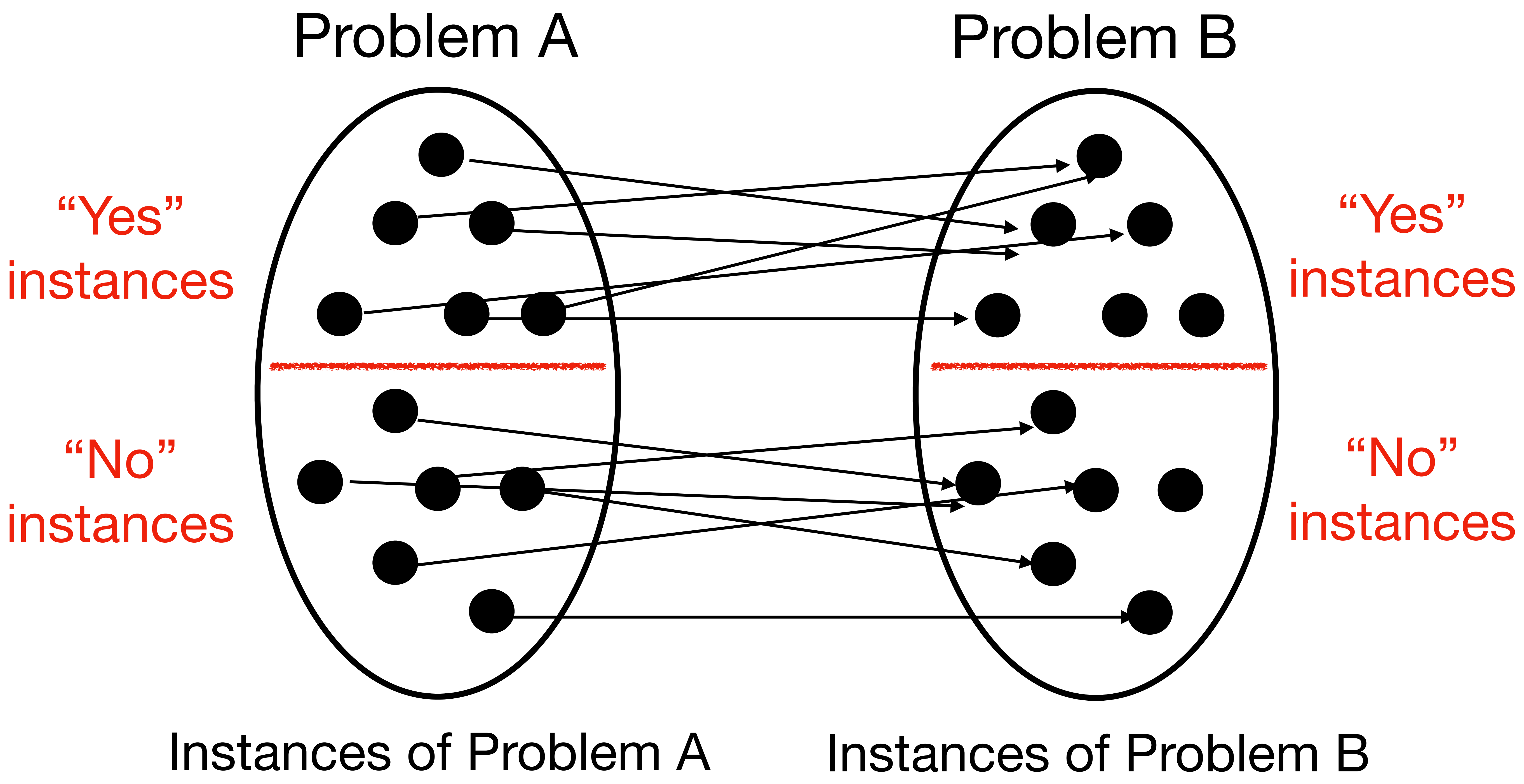
Problem A: Does $ax + b = 0$ have an integer solution?

Problem B: Does $ax^2 + bx + c = 0$ have an integer solution?

Mapping:

Instance of Problem A: $ax + b = 0$

Instance of Problem B: $0x^2 + ax + b = 0$



The mapping is from A to B (not B to A)

The mapping preserves the “Yes/No” answer.

NP Completeness

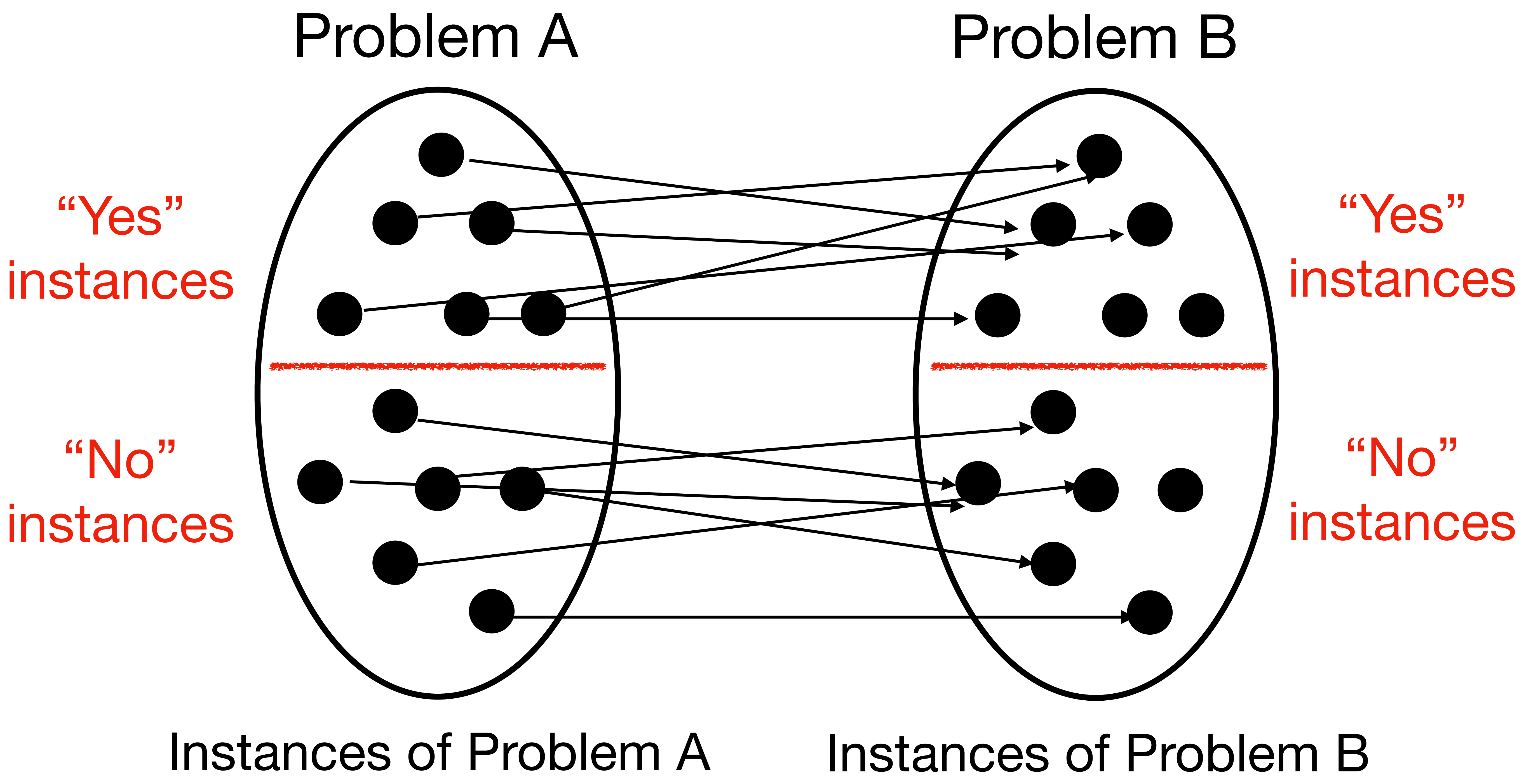
Example:

Problem A: Does $ax + b = 0$

have an integer solution?

Problem B: Does $ax^2 + bx + c = 0$

have an integer solution?



Mapping:

Instance of Problem A: $ax + b = 0$

Instance of Problem B: $0x^2 + ax + b = 0$

The mapping is from A to B (not B to A)

The mapping is a reduction.

The mapping preserves the "Yes/No" answer.

NP Completeness

Example:

Problem A: Does $ax + b = 0$

have an integer solution?

Problem B: Does $ax^2 + bx + c = 0$

have an integer solution?

Mapping:

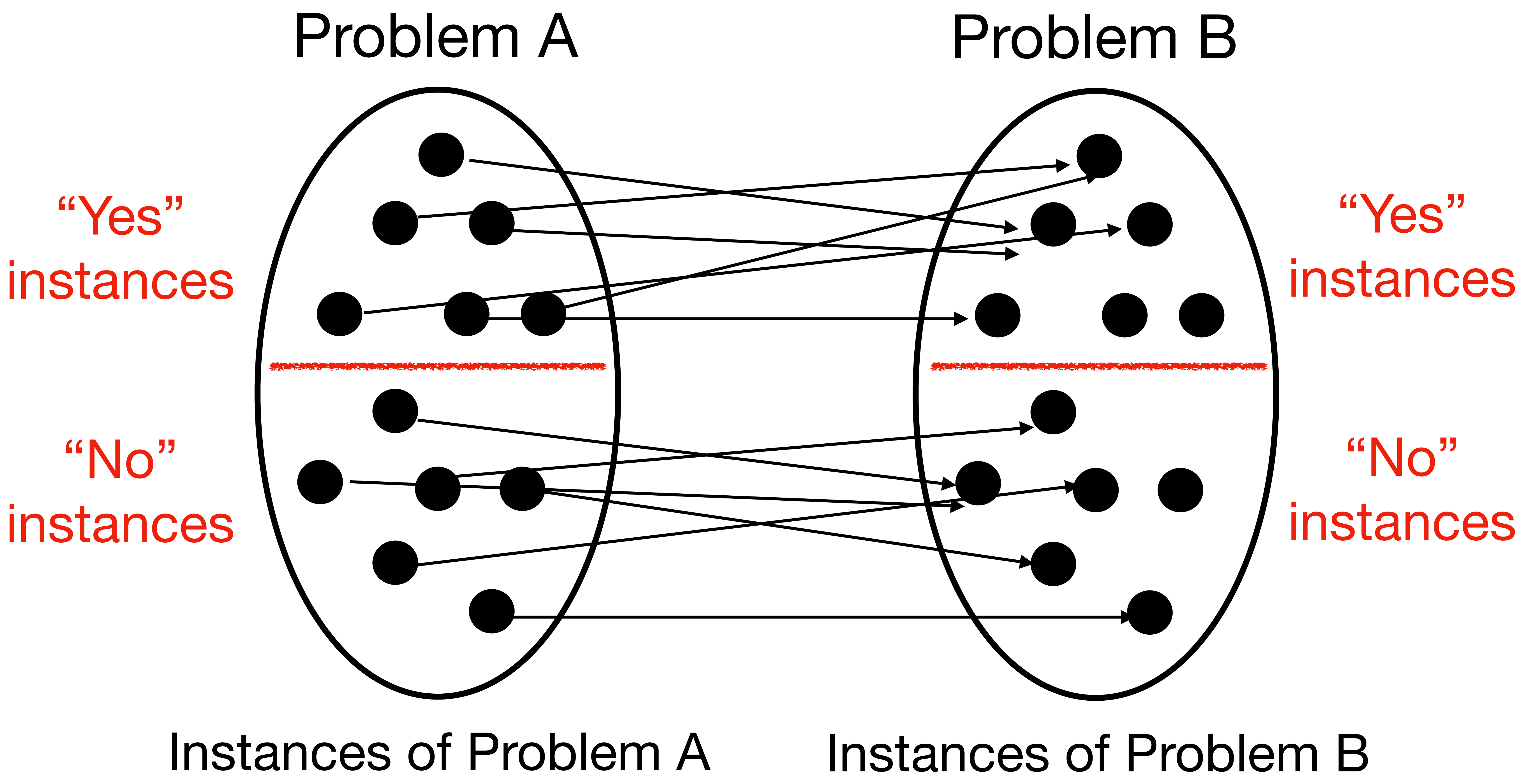
Instance of Problem A: $ax + b = 0$

$\Downarrow$

Instance of Problem B: $0x^2 + ax + b = 0$

The mapping is a reduction.

Polynomial-time reduction:
A reduction that takes polynomial time.



The mapping is from A to B (not B to A)

The mapping preserves the “Yes/No” answer.

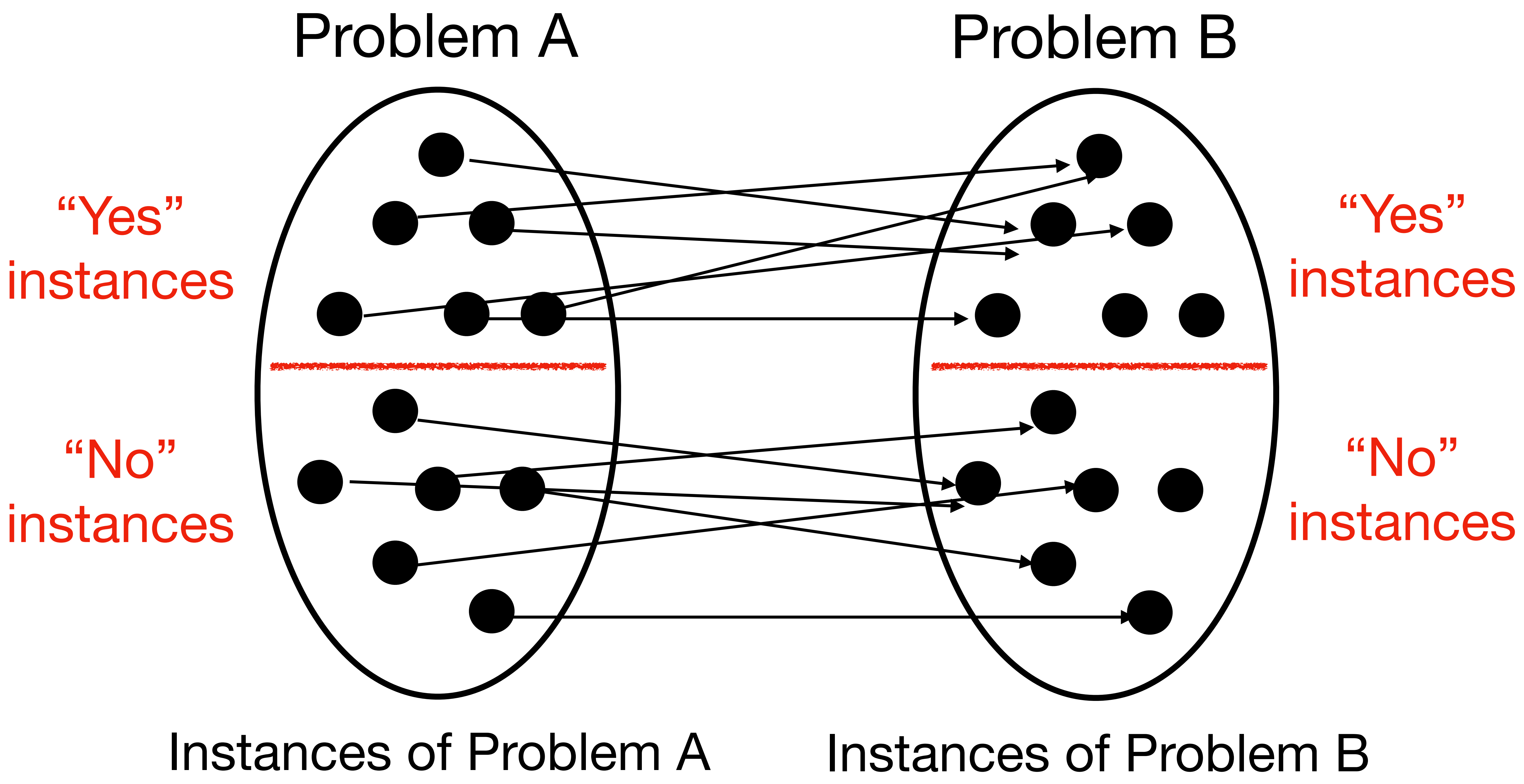
NP Completeness

If B is polynomial-time solvable,

then A is polynomial-time solvable.

Why?

Polynomial-time reduction:

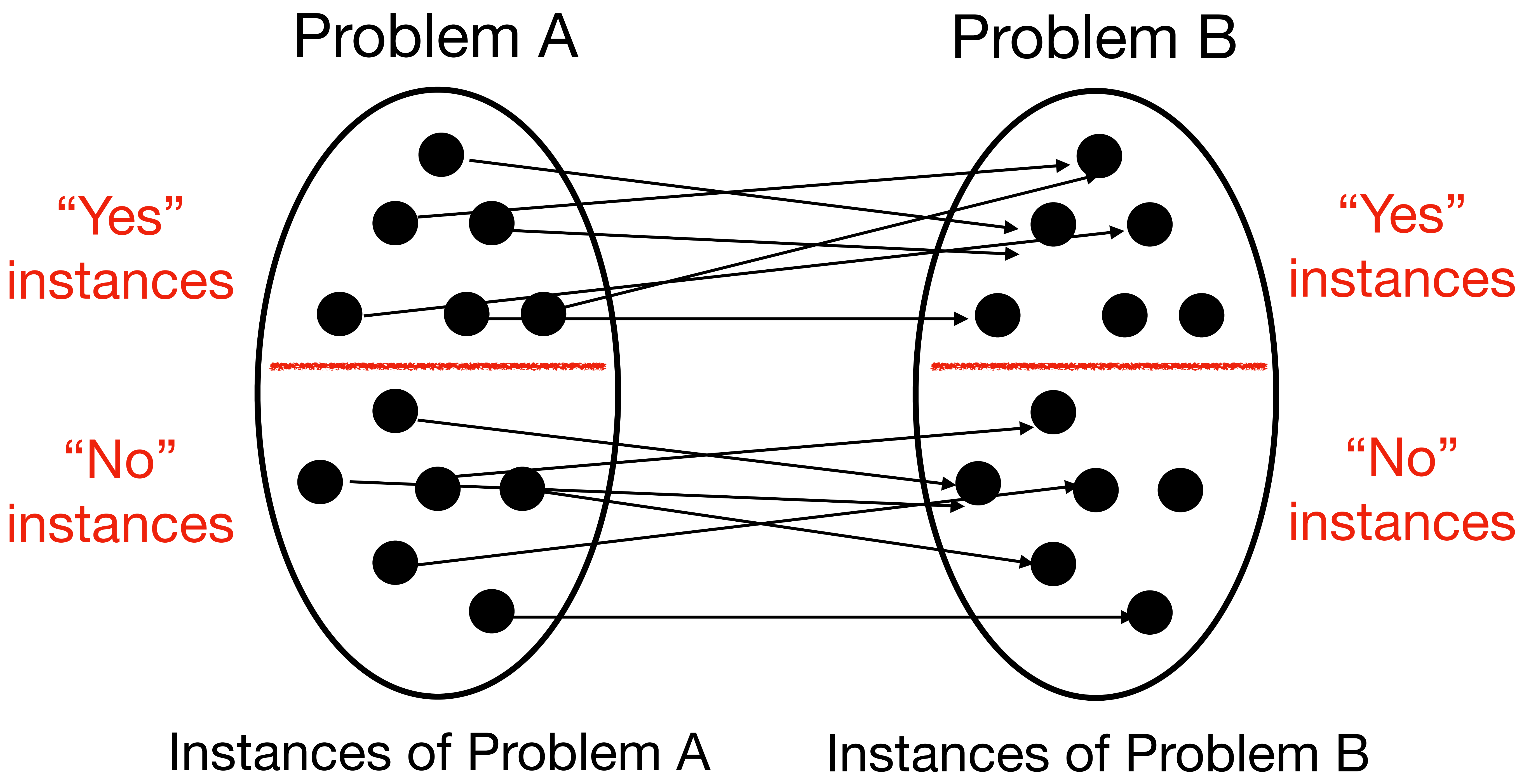


NP Completeness

If B is polynomial-time solvable,

then A is polynomial-time solvable.

Polynomial-time reduction:



In this sense,

Problem A is “easier” (or “no harder”),

Problem B is “harder” (or “no easier”).

We will use this property to prove the “hardness” of problems.

Quiz questions:

1. Is a reduction a one-to-one mapping?

2. How can polynomial-time reduction be used to prove a problem is easy?

3. How can polynomial-time reduction be used to prove a problem is hard?