Recall

Summary of Lecture 18

We will study which problems (seemingly) cannot be solved in polynomial time.

\[ P = \text{the class of decision problems that have polynomial time algorithms} \]

\[ X \leq_P Y, \text{ for problems } X, Y, \text{ “}X \text{ reduces to } Y \text{ in polynomial time”}, \text{ means: we can use a polynomial time algorithm for } Y \text{ to make a polynomial time algorithm for } X. \]
A few decision problems in NP:

- Hamiltonian path/cycle
- Travelling Salesman Problem
- Independent Set

Common feature: if the answer is YES, then there is some succinct information (a certificate) to verify that the answer is YES.

**Example: Independent Set.** Given graph $G$, and number $k$, does $G$ have an independent set of size $\geq k$?

How can I convince you that Yes, there is an independent set of size $\geq 5$?

How can I convince you that No, there is no independent set of size $\geq 7$?
A verification algorithm takes input + certificate and checks it. Formally:

**Definition.** Algorithm A is a verification algorithm for the decision problem X if

- A takes two inputs $x$, $y$ and outputs YES or NO 
- for every input $x$ for problem X, $x$ is a YES for X iff there exists a $y$ (a certificate) such that $A(x,y)$ outputs YES

Furthermore, A is a polynomial time verification algorithm if

- A runs in polynomial time
- there is a polynomial bound on the size of the certificate $y$

We say X “can be verified in polynomial time” if there is a poly time verification algorithm for X.

**Definition.**

NP = the class of decision problems that can be verified in polynomial time

NP = Non-deterministic Polynomial time — because the certificate is like a non-deterministic guess
Examples

Subset Sum $\in$ NP

Given numbers $w_1, \ldots, w_n, W$ is there a subset $S \subseteq \{1, \ldots, n\}$ such that $\sum_{i \in S} w_i = W$

Certificate: the set $S$
Verification: check that $\sum_{i \in S} w_i = W$
This takes poly. time.

TSP (decision version) $\in$ NP

Given a graph $G$, weights on edges, number $k$, does $G$ have a TSP tour of length $\leq k$

Certificate: a permutation of the vertices
Verification: check it's a permutation, check edges exist to make a cycle, check sum of weights of edges in cycle is $\leq k$. This takes poly. time.
Examples that don’t seem to be in NP

Unique Subset Sum

Given numbers $w_1, \ldots, w_n, W$ is there a unique subset $S \subseteq \{1, \ldots, n\}$ such that $\sum_{i \in S} w_i = W$

You can verify that a given $S$ is a solution. But how can you verify that $S$ is the only solution?

Steiner tree in the plane

Given points in the plane, can you connect them (using extra points) with a tree of Euclidean length $\leq k$

Two difficulties:
- the coordinates of the extra points (are they rational?)
- checking sum of Euclidean lengths $\leq k$ is not known in poly. time because of $\sqrt{5}$. 

Two difficulties:
- the coordinates of the extra points (are they rational?)
- checking sum of Euclidean lengths $\leq k$ is not known in poly. time because of $\sqrt{5}$. 

Two difficulties:
- the coordinates of the extra points (are they rational?)
- checking sum of Euclidean lengths $\leq k$ is not known in poly. time because of $\sqrt{5}$. 

Two difficulties:
- the coordinates of the extra points (are they rational?)
- checking sum of Euclidean lengths $\leq k$ is not known in poly. time because of $\sqrt{5}$. 

Two difficulties:
- the coordinates of the extra points (are they rational?)
- checking sum of Euclidean lengths $\leq k$ is not known in poly. time because of $\sqrt{5}$. 

Two difficulties:
- the coordinates of the extra points (are they rational?)
- checking sum of Euclidean lengths $\leq k$ is not known in poly. time because of $\sqrt{5}$. 

Two difficulties:
- the coordinates of the extra points (are they rational?)
- checking sum of Euclidean lengths $\leq k$ is not known in poly. time because of $\sqrt{5}$. 

Two difficulties:
- the coordinates of the extra points (are they rational?)
- checking sum of Euclidean lengths $\leq k$ is not known in poly. time because of $\sqrt{5}$. 

Two difficulties:
- the coordinates of the extra points (are they rational?)
- checking sum of Euclidean lengths $\leq k$ is not known in poly. time because of $\sqrt{5}$. 

Two difficulties:
- the coordinates of the extra points (are they rational?)
- checking sum of Euclidean lengths $\leq k$ is not known in poly. time because of $\sqrt{5}$. 

Two difficulties:
- the coordinates of the extra points (are they rational?)
- checking sum of Euclidean lengths $\leq k$ is not known in poly. time because of $\sqrt{5}$. 

Two difficulties:
- the coordinates of the extra points (are they rational?)
- checking sum of Euclidean lengths $\leq k$ is not known in poly. time because of $\sqrt{5}$. 

Two difficulties:
- the coordinates of the extra points (are they rational?)
- checking sum of Euclidean lengths $\leq k$ is not known in poly. time because of $\sqrt{5}$.

Two difficulties:
- the coordinates of the extra points (are they rational?)
- checking sum of Euclidean lengths $\leq k$ is not known in poly. time because of $\sqrt{5}$.

Two difficulties:
- the coordinates of the extra points (are they rational?)
- checking sum of Euclidean lengths $\leq k$ is not known in poly. time because of $\sqrt{5}$.

Two difficulties:
- the coordinates of the extra points (are they rational?)
- checking sum of Euclidean lengths $\leq k$ is not known in poly. time because of $\sqrt{5}$.

Two difficulties:
- the coordinates of the extra points (are they rational?)
- checking sum of Euclidean lengths $\leq k$ is not known in poly. time because of $\sqrt{5}$.

Two difficulties:
- the coordinates of the extra points (are they rational?)
- checking sum of Euclidean lengths $\leq k$ is not known in poly. time because of $\sqrt{5}$.

Two difficulties:
- the coordinates of the extra points (are they rational?)
- checking sum of Euclidean lengths $\leq k$ is not known in poly. time because of $\sqrt{5}$.

Two difficulties:
- the coordinates of the extra points (are they rational?)
- checking sum of Euclidean lengths $\leq k$ is not known in poly. time because of $\sqrt{5}$.

Two difficulties:
- the coordinates of the extra points (are they rational?)
- checking sum of Euclidean lengths $\leq k$ is not known in poly. time because of $\sqrt{5}$.

Two difficulties:
- the coordinates of the extra points (are they rational?)
- checking sum of Euclidean lengths $\leq k$ is not known in poly. time because of $\sqrt{5}$.

Two difficulties:
- the coordinates of the extra points (are they rational?)
- checking sum of Euclidean lengths $\leq k$ is not known in poly. time because of $\sqrt{5}$.

Two difficulties:
- the coordinates of the extra points (are they rational?)
- checking sum of Euclidean lengths $\leq k$ is not known in poly. time because of $\sqrt{5}$.

Two difficulties:
- the coordinates of the extra points (are they rational?)
- checking sum of Euclidean lengths $\leq k$ is not known in poly. time because of $\sqrt{5}$.

Two difficulties:
- the coordinates of the extra points (are they rational?)
- checking sum of Euclidean lengths $\leq k$ is not known in poly. time because of $\sqrt{5}$.

Two difficulties:
- the coordinates of the extra points (are they rational?)
- checking sum of Euclidean lengths $\leq k$ is not known in poly. time because of $\sqrt{5}$.

Two difficulties:
- the coordinates of the extra points (are they rational?)
- checking sum of Euclidean lengths $\leq k$ is not known in poly. time because of $\sqrt{5}$.

Two difficulties:
- the coordinates of the extra points (are they rational?)
- checking sum of Euclidean lengths $\leq k$ is not known in poly. time because of $\sqrt{5}$.

Two difficulties:
- the coordinates of the extra points (are they rational?)
- checking sum of Euclidean lengths $\leq k$ is not known in poly. time because of $\sqrt{5}$.

Two difficulties:
- the coordinates of the extra points (are they rational?)
- checking sum of Euclidean lengths $\leq k$ is not known in poly. time because of $\sqrt{5}$.

Two difficulties:
- the coordinates of the extra points (are they rational?)
- checking sum of Euclidean lengths $\leq k$ is not known in poly. time because of $\sqrt{5}$.

Two difficulties:
- the coordinates of the extra points (are they rational?)
- checking sum of Euclidean lengths $\leq k$ is not known in poly. time because of $\sqrt{5}$.

Two difficulties:
- the coordinates of the extra points (are they rational?)
- checking sum of Euclidean lengths $\leq k$ is not known in poly. time because of $\sqrt{5}$.

Two difficulties:
- the coordinates of the extra points (are they rational?)
- checking sum of Euclidean lengths $\leq k$ is not known in poly. time because of $\sqrt{5}$.

Two difficulties:
- the coordinates of the extra points (are they rational?)
- checking sum of Euclidean lengths $\leq k$ is not known in poly. time because of $\sqrt{5}$.

Two difficulties:
- the coordinates of the extra points (are they rational?)
- checking sum of Euclidean lengths $\leq k$ is not known in poly. time because of $\sqrt{5}$. 

Two difficulties:
- the coordinates of the extra points (are they rational?)
- checking sum of Euclidean lengths $\leq k$ is not known in poly. time because of $\sqrt{5}$. 

Two difficulties:
- the coordinates of the extra points (are they rational?)
- checking sum of Euclidean lengths $\leq k$ is not known in poly. time because of $\sqrt{5}$. 

Two difficulties:
- the coordinates of the extra points (are they rational?)
- checking sum of Euclidean lengths $\leq k$ is not known in poly. time because of $\sqrt{5}$. 

Two difficulties:
- the coordinates of the extra points (are they rational?)
- checking sum of Euclidean lengths $\leq k$ is not known in poly. time because of $\sqrt{5}$. 

Two difficulties:
- the coordinates of the extra points (are they rational?)
- checking sum of Euclidean lengths $\leq k$ is not known in poly. time because of $\sqrt{5}$. 

Two difficulties:
- the coordinates of the extra points (are they rational?)
- checking sum of Euclidean lengths $\leq k$ is not known in poly. time because of $\sqrt{5}$. 

Two difficulties:
- the coordinates of the extra points (are they rational?)
- checking sum of Euclidean lengths $\leq k$ is not known in poly. time because of $\sqrt{5}$. 

Two difficulties:
- the coordinates of the extra points (are they rational?)
- checking sum of Euclidean lengths $\leq k$ is not known in poly. time because of $\sqrt{5}$. 

Two difficulties:
- the coordinates of the extra points (are they rational?)
- checking sum of Euclidean lengths $\leq k$ is not known in poly. time because of $\sqrt{5}$. 

Two difficulties:
- the coordinates of the extra points (are they rational?)
- checking sum of Euclidean lengths $\leq k$ is not known in poly. time because of $\sqrt{5}$. 

Two difficulties:
- the coordinates of the extra points (are they rational?)
- checking sum of Euclidean lengths $\leq k$ is not known in poly. time because of $\sqrt{5}$. 

Two difficulties:
- the coordinates of the extra points (are they rational?)
- checking sum of Euclidean lengths $\leq k$ is not known in poly. time because of $\sqrt{5}$. 

Two difficulties:
- the coordinates of the extra points (are they rational?)
- checking sum of Euclidean lengths $\leq k$ is not known in poly. time because of $\sqrt{5}$. 

Two difficulties:
- the coordinates of the extra points (are they rational?)
- checking sum of Euclidean lengths $\leq k$ is not known in poly. time because of $\sqrt{5}$. 

Two difficulties:
- the coordinates of the extra points (are they rational?)
- checking sum of Euclidean lengths $\leq k$ is not known in poly. time because of $\sqrt{5}$.
**Claim.** $P \subseteq NP$, i.e. if $X$ is in $P$ then $X$ is in $NP$.

**Proof.** The certificate is empty and the verification algorithm is just the poly time algorithm for $X$.

---

**Definition.**

\[\text{coNP} = \text{the class of decision problems where the NO instances can be verified in polynomial time}\]

---

Example. **Primes:** Given a number $n$, is it prime?

Primes $\in$ coNP

To verify that $n$ is NOT prime, the certificate is numbers $a, b \in \mathbb{N}$, $a, b \geq 2$ and verify $a \cdot b = n$

In fact, Primes $\in P$. A poly time algorithm was found in 2002.

OPEN QUESTIONS

1. $P = \? NP$ 
   worth $1$ million (Millenium Prize)

2. $NP = \? coNP$

3. $P = \? NP \cap coNP$
OPEN QUESTIONS

1. P =? NP

2. NP =? coNP

3. P =? NP \cap coNP

Properties

1. P \subseteq NP, P \subseteq coNP

2. Any problem in NP can be solved in time \(O(2^n)\) by trying all certificates one by one
Summary of Lecture 19, Part 1

- classes \( \text{NP, coNP} \)

What you should know from Lecture 19, Part 1:

- how to prove that a problem is in \( \text{NP} \) (certificate, verification)

Next:

- \( \text{NP-complete} \) problems
**Definition.** A decision problem $X$ is **NP-complete** if

- $X \in \text{NP}$
- for every $Y$ in NP, $Y \leq_P X$

i.e. $X$ is [one of] the hardest problem in NP.

Two important implications of $X$ being NP-complete

- if $X$ can be solved in polynomial time then so can every problem in NP  
  (if $X \in \text{P}$ then $\text{P} = \text{NP}$)

- if $X$ cannot be solved in polynomial time then no NP-complete problem can be solved in polynomial time

- if $X \in \text{co-NP}$ then $\text{NP} = \text{coNP}$ (this needs proof)
The first NP-completeness proof is difficult — must show that every problem \( Y \in \text{NP} \) reduces to \( X \).

Subsequent NP-completeness proofs are easier because \( \leq_p \) is transitive:

**Claim.** If \( Y \leq_p X \) and \( X \leq_p Z \) then \( Y \leq_p Z \).

So to prove \( Z \) is NP-complete, we just need to prove \( X \leq_p Z \) where \( X \) is a known NP-complete problem.
Summary: to prove a decision problem $Z$ is NP-complete

1. prove $Z$ in NP
2. prove $X \leq_P Z$ for some known NP-complete problem $X$.

Our first NP-complete problem: Circuit Satisfiability
[definition and proof later]

second NP-complete problem: Satisfiability
[proof later, but definition now]

Satisfiability (SAT)
Input: a Boolean formula made of Boolean variables, and logical operands $\land$ "and", $\lor$ "or", $\neg$ "not"

e.g. $\neg (x_1 \land x_2) \lor (x_3 \land (x_5 \lor \neg x_4))$

Question: Is there an assignment of True/False to the variables to make the formula True?

 e.g. $x_1 = \text{False}$ and others arbitrary makes the above formula True

Exercise. Prove that Satisfiability is in NP.
SAT is NP-complete, even the special case of “CNF” — Conjunctive Normal Form

**Definition** of CNF

formula is $\land$ of *clauses*; clause is $\lor$ of *literals*; literal is $x$ or $\neg x$

\[
(x_1 \lor \neg x_2 \lor x_3) \land (\neg x_1 \lor x_4) \land (x_3 \lor x_4 \lor \neg x_5)
\]

clause $\uparrow$ $\uparrow$

literals

In fact, SAT is still NP-complete when all clauses have 3 literals — this is called 3-SAT

**3-SAT**

**Input:** A Boolean formula that is an $\land$ of clauses, each clause an $\lor$ of 3 literals, each literal a variable or negation of a variable.

**Question:** Is there an assignment of True/False to the variables to make the formula True?

**Theorem.** 3-SAT is NP-complete [proof later]

but 2-SAT is in P

There is a linear time algorithm for 2-SAT that uses strong connectivity of a directed graph.

Summary of Lecture 19, Part 2

definition of NP-complete, the first NP-complete problems: SAT, 3-SAT

What you should know from Lecture 19, Part 2:

- what are the two steps to proving a problem is NP-complete

Next:

- examples of NP-completeness proofs
**Independent Set**
**Input:** Graph $G = (V,E)$, number $k$.
**Question:** Does $G$ have an independent set of size $\geq k$?

**Theorem.** Independent Set is NP-complete.

**Proof.**

1. Independent Set is in NP — we already saw the idea of this in Part 1.
2. $3\text{-SAT} \leq_P$ Independent Set
Independent Set

Input: Graph G = (V,E), number k.
Question: Does G have an independent set of size ≥ k?

Theorem. Independent Set is NP-complete.

Proof.

1. Independent Set is in NP — we already saw the idea of this in Part 1.
2. 3-SAT ≤ₚ Independent Set

Suppose we have a polynomial time algorithm for Independent Set.
Give a polynomial time algorithm for 3-SAT.

Input: A 3-SAT formula F with clauses C₁ . . . Cₘ on variables x₁ . . . xₙ
Output: Is F satisfiable?

Idea: - construct a graph G and choose a number k such that
  \[ G \text{ has an independent set of size } \geq k \text{ iff } F \text{ is satisfiable} \]
  - run the Independent Set algorithm on G, k
  - return its answer

This is a many-one ("one-shot") reduction. To prove correctness, just prove ★
To prove poly time, just prove that G can be constructed in poly time (in size of F).
Proof. continued

Input: A 3-SAT formula $F$ with clauses $C_1 \ldots C_m$ on variables $x_1 \ldots x_n$

Output: Is $F$ satisfiable?

Idea: - construct a graph $G$ and choose a number $k$ such that
  - $G$ has an independent set of size $\geq k$ iff $F$ is satisfiable
- run the Independent Set algorithm on $G$, $k$
- return its answer
**Input:** A 3-SAT formula $F$ with clauses $C_1 \ldots C_m$ on variables $x_1 \ldots x_n$

**Output:** Is $F$ satisfiable?

**Idea:**
- construct a graph $G$ and choose a number $k$ such that $G$ has an independent set of size $\geq k$ iff $F$ is satisfiable
- run the Independent Set algorithm on $G$, $k$
- return its answer

**Construction:**
- For each clause $C_i$ with literals $l_1, l_2, l_3$, make 3 vertices joined by 3 edges
- if two literals are opposite, join them with an edge.
- $k := m$

**Runtime:** Prove that $G$ can be constructed in poly time (in the size of $F$). $G$ has $3m$ vertices and can be constructed in time polynomial in $m$ and $n$

**Correctness:** prove $G$ has an independent set of size $\geq k$ iff $F$ is satisfiable
- if $F$ is satisfiable then each clause has (at least) one True literal. Choose the corresponding $m$ vertices of $G$. They are independent.
- if $G$ has an independent set of size $\geq m$ there must be one in each triangle. Set the corresponding literals True. This is valid, and satisfies $F$.

This completes the proof that Independent Set is NP-complete.
Definition. Problem X reduces to problem Y, written $X \leq Y$, if an algorithm for Y can be used to make an algorithm for X.

Definition. A many one reduction $X \leq Y$ uses the algorithm for Y once and outputs its answer.

mnemonic: many-one = “one-shot”

“many-one” is a standard name; one-shot is not

The form of a polynomial time many-one reduction $X \leq_P Y$:

Assume we have an algorithm A for Y

Algorithm for X:
- take input $x$ and construct an input $y$ for problem Y
- run A on $y$
- return the answer

For correctness we just need to prove:

the answer for $x$ is YES iff the answer for $y$ is YES

For poly time we just need to prove:

the construction of $y$ takes polynomial time.
How to prove that a decision problem $Z$ is NP-complete

1. prove $Z$ in NP
2. prove $X \leq_P Z$ for some known NP-complete problem $X$.
   
   Use a *many-one* reduction.
Summary of Lecture 19

- definition of NP-complete, first NP-completeness proofs

What you should know from Lecture 19 (and Lecture 20)

- how to prove a problem is NP-complete using a polynomial time many-one reduction

Next:

- more examples of NP-completeness proofs