

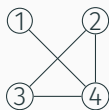
Nondeterministic Polynomial Time

CSCI 3130 Formal Languages and Automata Theory

Siu On CHAN

Fall 2022

Chinese University of Hong Kong



Graph G

A **clique** is a subset of vertices that are **pairwise adjacent**

$\{1, 4\}$, $\{2, 3, 4\}$, $\{1\}$ are cliques

An **independent set** is a subset of vertices that are **pairwise non-adjacent**

$\{1, 2\}$, $\{1, 3\}$, $\{4\}$ are independent sets

A **vertex cover** is a set of vertices that **touches (covers) all edges**

$\{2, 4\}$, $\{3, 4\}$, $\{1, 2, 3\}$ are vertex covers

These problems

CLIQUE = $\{\langle G, k \rangle \mid \text{Graph } G \text{ has a clique of } k \text{ vertices}\}$

INDEPENDENT-SET = $\{\langle G, k \rangle \mid \text{Graph } G \text{ has}$
an independent set of k vertices}

VERTEX-COVER = $\{\langle G, k \rangle \mid \text{Graph } G \text{ has}$
a vertex cover of k vertices}

What do these problems **have in common?**

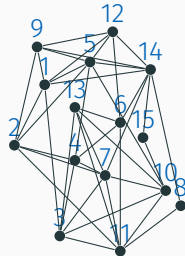
1. Given a candidate solution, we can quickly check if it is valid
2. We don't know how to solve these problems quickly

Checking solutions quickly

If someone told us a candidate solution, we can quickly **verify** if it is valid

Example: Is $\langle G, 5 \rangle \in \text{CLIQUE?}$

Candidate solution: $\{1, 5, 9, 12, 14\}$



The class NP

A **verifier** for L is a Turing machine V such that

$$x \in L \iff V \text{ accepts } \langle x, s \rangle \text{ for some } s$$

s is a **candidate solution** for x

We say V runs in **polynomial time** if on every input x , it runs in time polynomial in $|x|$ (for every s)

NP is the class of all languages that have polynomial-time verifiers

Example

$V =$ On input $\langle G, k, C \rangle$, where C is a set of vertices in G

If C has size k and all edges between vertices C are present in G
accept

Otherwise reject

Running time: $O(k^2)$

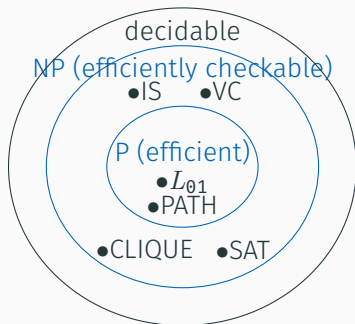
Therefore CLIQUE is in NP

P versus NP

P is contained in NP

because the verifier can ignore the candidate solution

Intuitively, **finding** solutions can only be harder than **verifying** them



IS = INDEPENDENT-SET

VC = VERTEX-COVER

We will talk about SAT in the next lecture

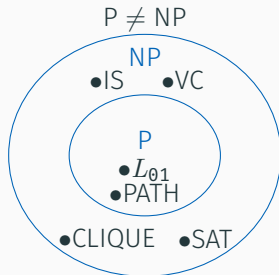
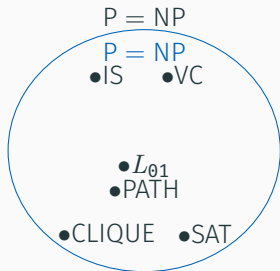
Millennium prize problems

In 2000, Clay Math Institute gave 7 problems for the 21st century

- P versus NP (computer science)
- Hodge conjecture
- Poincaré conjecture (Perelman 2006)
- Riemann hypothesis (Hilbert's 8th problem)
- Yang–Mills existence and mass gap
- Navier–Stokes existence and smoothness
- Birch and Swinnerton-Dyer conjecture

P versus NP

Is P equal to NP?



We don't know. But one reason to believe $P \neq NP$ is that intuitively, **searching** for a solution is harder than verifying its correctness

For example, solving homework problems (searching for solutions) is harder than **grading** (verifying the candidate solution is correct)

Searching versus verifying

Mathematician:

Given a mathematical claim, come up with a proof for it

Scientist:

Given a collection of data on some phenomenon, find a theory explaining it

Engineer:

Given a set of constraints (on cost, physical laws, etc), come up with a design (of an engine, bridge, etc) which meets them

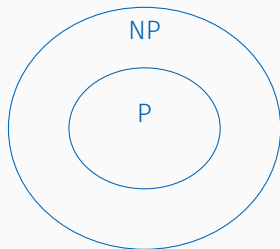
Detective:

Given the crime scene, find “who’s done it”

P and NP

P = languages that can be decided on TM in **polynomial time** (admit efficient algorithms)

NP = languages whose solutions can be **verified** on a TM in polynomial time (solutions can be **checked** efficiently)



We believe $P \neq NP$, but we are not sure

Evidence that NP is bigger than P

CLIQUE = $\{\langle G, k \rangle \mid G \text{ is a graph having a clique of } k \text{ vertices}\}$

IS = $\{\langle G, k \rangle \mid G \text{ is a graph having}$
an independent set of k vertices}

VC = $\{\langle G, k \rangle \mid G \text{ is a graph having}$
a vertex cover of k vertices}

What do they have in common?

- These (and many others) are in NP
- No efficient algorithms are known for solving any of them

Naive algorithm for solving CLIQUE

CLIQUE = $\{\langle G, k \rangle \mid G \text{ is a graph having a clique of } k \text{ vertices}\}$

Turing machine M : On input $\langle G, k \rangle$

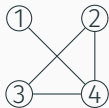
For all subsets S of vertices of size k

If every pair $u \in S, v \in S, u \neq v$ are adjacent

accept

else reject

Example:



Graph G

input:	$\langle G, 3 \rangle$			
subsets:	$\{123\}$	$\{124\}$	$\{134\}$	$\{234\}$
All edges in S ?	No	No	No	Yes

Running time analysis

CLIQUE = $\{\langle G, k \rangle \mid G \text{ is a graph having a clique of } k \text{ vertices}\}$

M : On input $\langle G, k \rangle$:

For all subsets S of vertices of size k $\binom{n}{k}$ subsets

 If every pair $u, v \in S$ are adjacent k^2 pairs

 accept

 else reject

running time: $k^2 \binom{n}{k}$
 $\geq 2^n$ when $k = n/2$

Equivalence of certain NP languages

We strongly suspect that problems like CLIQUE, SAT, etc require roughly 2^n time to solve

We do not know how to prove this, but we can prove that

If **any one** of them can be solved efficiently,
then **all of them** can be solved efficiently

Equivalence of some NP languages

Next lecture:

All problems such as CLIQUE, SAT, IS, VC are **as hard as** one another

Moreover, they are at least as hard as any problem in NP