جامعة الملك سعود
King Saud University

Theory of Computation
CSC 339 – Spring 2021

Chapter-7: part3
The Class NP

King Saud University
Department of Computer Science
Dr. Azzam Alsudais

Introduction

- **For some problems, we haven’t found polynomial algorithms, yet!**

Introduction

- **For some problems, we haven't found polynomial algorithms, yet!**
  - **Do polynomial algorithms exist for those problems?**

Introduction

- **For some problems, we haven't found polynomial algorithms, yet!**
  - **Do polynomial algorithms exist for those problems?**
    - **Maybe and maybe not.**

The Class NP: Example Problem (Hamiltonian Paths)

- **A Hamiltonian path in a directed graph G is a directed path that goes each node exactly once.**

The Class NP: Example Problem (Hamiltonian Paths)

- **A Hamiltonian path in a directed graph G is a directed path that goes each node exactly once.**

$$HAMPTAH = \{\langle G, s, t \rangle \mid G \text{ is a directed graph with a Hamiltonian path from } s \text{ to } t\}$$

The Class NP: Example Problem (Hamiltonian Paths)

- **A Hamiltonian path in a directed graph G is a directed path that goes each node exactly once.**

HAMPTAH = $\{\langle G, s, t\rangle \mid G$ *is a directed graph with a Hamiltonian path from* s *to* $t\}$

- **How can we decide *HAMPATH*?**

The Class NP: Example Problem (Hamiltonian Paths)

- **A Hamiltonian path in a directed graph G is a directed path that goes each node exactly once.**

HAMPTAH = {⟨*G*, *s*, *t* ⟩ | *G is a directed graph with a Hamiltonian path from s to t*}

- **How can we decide *HAMPATH*?**

- **Can we design a polynomial algorithm for *HAMPATH*?**

The Class NP: Example Problem (Hamiltonian Paths)

- **A Hamiltonian path in a directed graph G is a directed path that goes each node exactly once.**

HAMPTAH = {⟨*G*, *s*, *t* ⟩ | *G is a directed graph with a Hamiltonian path from s to t*}

- **How can we decide** *HAMPATH***?**
- **Can we design a polynomial algorithm for** *HAMPATH***?**
- **So far, there has not been any polynomial algorithms that can decide** *HAMPATH***.**
- **However, …**

The Class NP: Example Problem (Hamiltonian Paths)

- **A Hamiltonian path in a directed graph G is a directed path that goes each node exactly once.**

HAMPTAH = $\{\langle G, s, t \rangle \mid G$ *is a directed graph with a Hamiltonian path from s to t*$\}$

- **How can we decide** *HAMPATH***?**

- **Can we design a polynomial algorithm for** *HAMPATH***?**

- **So far, there has not been any polynomial algorithms that can decide** *HAMPATH***.**

The Class NP: Example Problem (Hamiltonian Paths)

- **However, HAMPATH has a feature called <u>polynomial verifiability</u>.**

The Class NP: Example Problem (Hamiltonian Paths)

- **However, HAMPATH has a feature called <u>polynomial verifiability</u>.**
- **What does <u>polynomial verifiability</u> mean?**
  - **Given a path, we can verify whether it's Hamiltonian in polynomial time.**

The Class NP: Example Problem (Compositeness)

- **A natural number is <u>composite</u> if it is the product of two integers greater than 1.**

The Class NP: Example Problem (Compositeness)

- **A natural number is <u>composite</u> if it is the product of two integers greater than 1.**

$$COMPOSITES = \{x \mid x = pq, \text{ for integers } p, q > 1\}$$

The Class NP: Example Problem (Compositeness)

- **A natural number is <u>composite</u> if it is the product of two integers greater than 1.**

$$COMPOSITES = \{x \mid x = pq, \text{ for integers } p, q > 1\}$$

- **There is not a simple polynomial algorithm for testing whether a given number is composite or not.**

The Class NP: Example Problem (Compositeness)

- **A natural number is <u>composite</u> if it is the product of two integers greater than 1.**

$$COMPOSITES = \{x \mid x = pq, \text{ for integers } p, q > 1\}$$

- **There is not a simple polynomial algorithm for testing whether a given number is composite or not.**

- **We can easily verify that a natural number is composite. All we need is a divisor for that number.**

The Class NP: Verifiers

Definition 7.18

A ***verifier*** for a language A is an algorithm V , where

$$A = \{w \mid V \text{ accepts } \langle w, c\rangle \text{ for some string } c\}.$$

We measure the time of a verifier only in terms of the length of w, so a ***polynomial time verifier*** runs in polynomial time in the length of w. A language A is ***polynomially verifiable*** if it has a polynomial time verifier.

The Class NP: Verifiers

Definition 7.18

***c* is a certificate (or proof) of *w*'s membership in A**

here

ring c}.

We measure the time of a verifier only in terms of the length of w, so a ***polynomial time verifier*** runs in polynomial time in the length of w. A language A is ***polynomially verifiable*** if it has a polynomial time verifier.

The Class NP: Verifiers

Definition 7.18

…where

…ring c}.

We measure the time of a verifier only in terms of the length of w, …omial time in the length of w. A language A is *polynomially verifiable* if it has a polynomial …

***c* is a certificate (or proof) of *w*'s membership in A**

In HAMPATH, c is a path from s to t

In COMPOSITES, c is a divisor for the number

The Class NP: Verifiers

- **For *HAMPATH* as well as *COMPOSITES*, we can verify an answer in polynomial time.**

The Class NP: Verifiers

- **For *HAMPATH* as well as *COMPOSITES*, we can verify an answer in polynomial time.**

Definition 7.19

NP is the class of languages (problems) that have polynomial time verifiers.

The Class NP: Example Problem (Hamiltonian Paths)

- **The term NP comes from *nondeterministic polynomial time*.**

The Class NP: Example Problem (Hamiltonian Paths)

- **The term NP comes from *nondeterministic polynomial time*.**

- **This means problems that are members of NP can be decided using nondeterministic TMs (NTM).**

- N_1 **in the following slide can decide** *HAMPATH* **using NTM.**

The Class NP: Example Problem (Hamiltonian Paths)

N_1 = "On input ⟨G, s, t⟩, where G is a directed graph with nodes s and t:

1. Write a list of m numbers, p_1 , …, p_m, where m is the number of nodes in G. Each number in the list is nondeterministically selected to be between 1 and m.
2. Check for repetitions in the list. If any are found, *reject*.
3. Check whether $s = p_1$ and $t = p_m$. If either fail, *reject*.
4. For each i between 1 and $m - 1$, check whether (p_i , p_{i+1}) is an edge of G. If any are not, ***reject*** . Otherwise, all tests have been passed, so ***accept***."

The Class NP

Theorem 7.20

A language is in NP iff it is decided by some nondeterministic polynomial time Turing machine.

The Class NP

Theorem 7.20

A language is in NP iff it is decided by some nondeterministic polynomial time Turing machine.

Does this definition apply to problems that can be decided in polynomial time ($\in$ P)?

The Class NP

Theorem 7.20

A language is in NP iff it is decided by some nondeterministic polynomial time Turing machine.

Does this definition apply to problems that can be decided in polynomial time ($\in$ P)?

P $\subset$ NP

The Class NP

- **NP is insensitive to the choice of a reasonable nondeterministic computation model, because they are polynomially equivalent.**

The Class NP

- **NP is insensitive to the choice of a reasonable nondeterministic computation model, because they are polynomially equivalent.**

- **This means we can decide problems in NP using nondeterminstic FA, TMs, etc.**

The Class NP: Example Problem (CLIQUE)

- **A clique in an undirected graph is a subgraph wherein every two nodes are connected by an edge.**

The Class NP: Example Problem (CLIQUE)

- **A clique in an undirected graph is a subgraph wherein every two nodes are connected by an edge.**

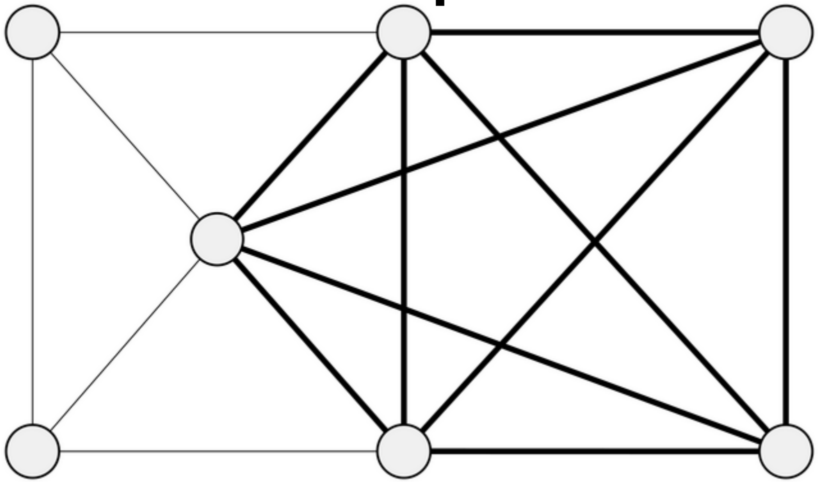
- **A** *k-clique* **is a clique that contains** *k* **nodes.**

The Class NP: Example Problem (CLIQUE)

- **A clique in an undirected graph is a subgraph wherein every two nodes are connected by an edge.**

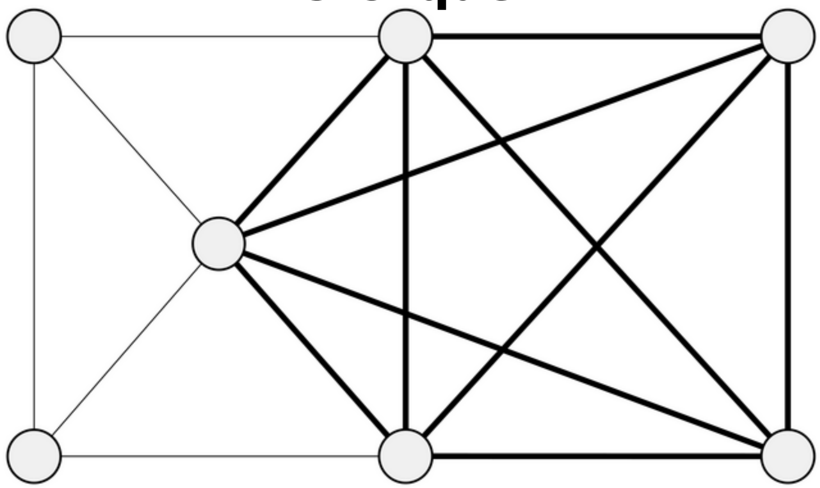
- **A** *k-clique* **is a clique that contains** *k* **nodes.**

5-clique



The Class NP: Example Problem (CLIQUE)

- **A clique in an undirected graph is a subgraph wherein every two nodes are connected by an edge.**
- **A** *k-clique* **is a clique that contains** *k* **nodes.**
- **The clique problem is to determine whether a graph contains a clique of a specified size.**



The Class NP: Example Problem (CLIQUE)

- **A clique in an undirected graph is a subgraph wherein every two nodes are connected by an edge.**

- **A** *k-clique* **is a clique that contains** *k* **nodes.**

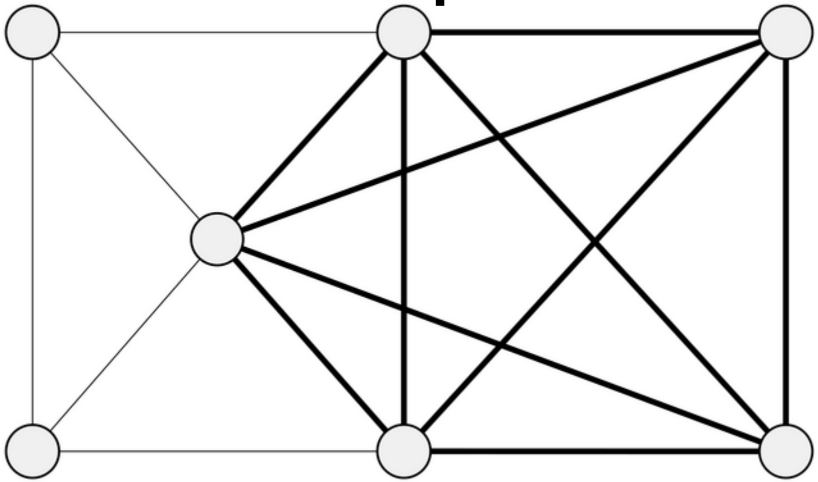
- **The clique problem is to determine whether a graph contains a clique of a specified size.**

CLIQUE = {⟨G, k⟩| G is an undirected graph with a k-clique}.

Theorem 7.24

CLIQUE is in NP

5-clique



The Class NP: Example Problem (CLIQUE)

Decider N for CLIQUE

N = "On input ⟨G, k⟩, where G is a graph:

1. Nondeterministically select a subset c of k nodes of G.
2. Test whether G contains all edges connecting nodes in c.
3. If yes, accept ; otherwise, reject ."

The Class NP: Example Problem (CLIQUE)

Verifier V for CLIQUE

V = "On input $\langle\langle G, k\rangle, c\rangle$:

1. Test whether c is a subgraph with k nodes in G.
2. Test whether G contains all edges connecting nodes in c.
3. If both pass, *accept*; otherwise, *reject*."

The Class NP: Example Problem (CLIQUE)

Verifier V for CLIQUE

c is the clique to be verified

V = "On input $\langle\langle G, k\rangle, c\rangle$:

1. Test whether c is a subgraph with k nodes in G.
2. Test whether G contains all edges connecting nodes in c.
3. If both pass, *accept*; otherwise, *reject*."

The Class NP: coNP

- $\overline{HAMPATH}$ **and** $\overline{CLIQUE}$ **are not in NP.**

The Class NP: coNP

- $\overline{\mathit{HAMPATH}}$ **and** $\overline{\mathit{CLIQUE}}$ **are not in NP. They are in coNP.**

- **Proving something is not present is more difficult than verifying it is present.**

The Class NP: coNP

- $\overline{HAMPATH}$ **and** $\overline{CLIQUE}$ **are not in NP. They are in coNP.**
- **Proving something is not present is more difficult than verifying it is present.**
- **We don’t know if coNP is different from NP!**

The Class NP: P vs. NP

- **P is the class of languages for which membership can be *decided* quickly.**

The Class NP: P vs. NP

- **P is the class of languages for which membership can be *decided* quickly.**

- **NP is the class of languages for which membership can be *verified* quickly.**

The Class NP: P vs. NP

- **P is the class of languages for which membership can be *decided* quickly.**

- **NP is the class of languages for which membership can be *verified* quickly.**

- **The best deterministic method currently known for deciding languages in NP uses exponential time.**

The Class NP: P vs. NP

- **P is the class of languages for which membership can be *decided* quickly.**

- **NP is the class of languages for which membership can be *verified* quickly.**

- **The best deterministic method currently known for deciding languages in NP uses exponential time.**

- **We are unable to prove the existence of a single language in NP that is not in P.**

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

The Class NP: P vs. NP

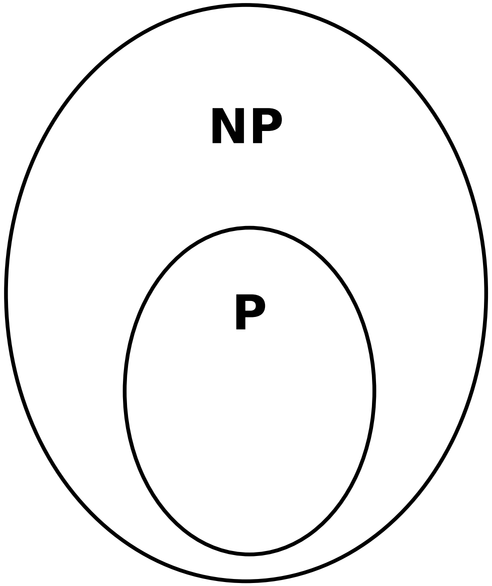
$$\boxed{P = NP?}$$

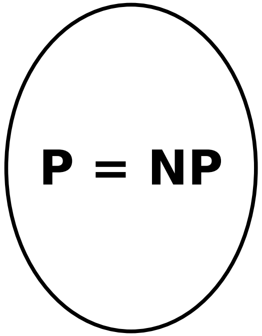
- **Researchers have invested an enormous amount of time to find polynomial algorithms for some problems in NP, but without success.**

- **On the other hand, scientific tools do not lend themselves to showing that there isn’t fast algorithms to replace brute-force search.**

The Class NP: P vs. NP

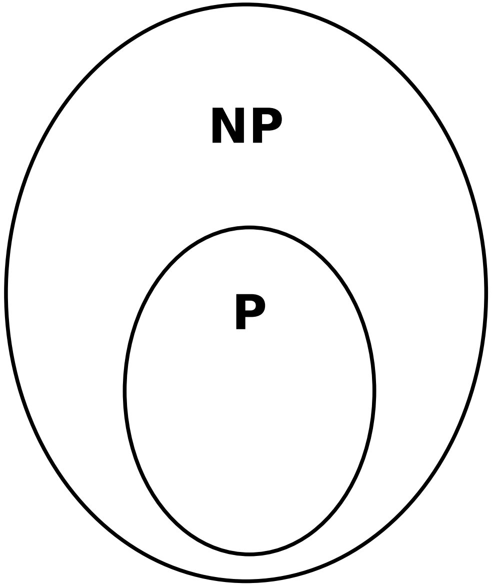
P = NP?





The Class NP: P vs. NP

$P = NP?$



One of these two possibilities is correct. But, we don’t know (yet) which one