the right majors, minors &
ucation?
for students' academic and career success
ging for many students - Many students change their
during college!
e prediction of student success in MMC could
dual students
d the right MMC
chieve their academic goals

Computational Intractability: P, NP & NP-Complete Problems

Prof. Young Park

✓ Computational Intractability

Intractable (Hard) Problems

- A problem in computer science is called intractable if a computer has difficulty solving it or an efficient (polynomial time) algorithm is not possible.
- For a problem to be intractable,
  - There must be NO polynomial-time algorithm that solves it.
- **Hard Problems**

Problems and Intractability

- **Intractability is a property of a problem!**
  - It is not a property of any one algorithm for that problem.
- There are **3 general categories** into which problems can be grouped as far as intractability is concerned!

✓ Three General Problem Categories

Three General Problem Categories

1. Problems for which **polynomial-time algorithms have been found** (i.e. **tractable**).
2. Problems that have been proven to be **intractable.**
3. Problems that have **not been proven** to be intractable, but for which polynomial-time algorithms have **never been found**.

1. Problems for Which Polynomial-Time Algorithms Have Been Found

- Any problem for which we have found a polynomial-time algorithm falls into this category.
  - The list of these types of algorithms is endless.

2. Problems That Have Been Proven to Be Intractable

- Two types of problems in this category:
  a. The first type is problems that require a non-polynomial amount of output.
  b. The second type of intractability occurs when our results are reasonable, and we can prove that a problem cannot be solved in polynomial time.

a. Problems That Have Been Proven to Be Intractable

- The first type is problems that require a non-polynomial amount of output.
  - **The problem of determining all Hamiltonian Circuits**
  - **The Towers of Hanoi Problem**

b. Problems That Have Been Proven to Be Intractable

- The second type of intractability occurs when our results are reasonable, and we can prove that a problem cannot be solved in polynomial time.
  - There are relatively few such problems.
  - **The Halting Problem**
    - **Undecidable!**
    - **Unsolvable!**
    - **The Program Termination Problem**

▶QUIZ?

- What is the Halting problem?
- Can we solve the Halting Problem?

3. Problems That Have Not Been Proven to Be Intractable but for Which Polynomial-Time Algorithms Have Never Been Found

- **Many** problems such as
  - 0-1 Knapsack,
  - Traveling Salesperson,
  - m-Coloring,
  - Hamiltonian Circuits
  - …
- We have found branch-and-bound algorithms, backtracking algorithms for these problems that are efficient in many instances.

Problems and Intractability

- **Most** problems seem to fall into either 1 or 3 category!

▶QUIZ?

- Three general problem categories?

Decision Problems & Optimization Problems

- Decision problems
  - The output of a decision problem is a simple “yes” or “no” answer.
- Optimization problems
  - The output is an optimal solution.
  - Optimization problems can be transformed into decision problems.
  - **Optimization problems are at least as hard as the associated decision problem!**

Decision Problems & Optimization Problems

- Example 9.2 TSP
- Example 9.3 0-1 Knapsack Problem
- Example 9.4 Graph-Coloring Problem

▶QUIZ?

- Given a problem, is there a polynomial-time algorithm that solves the problem?
  - Possible answers:
    - yes
    - no
    - unknown

✓ The Class P

The Class P

- **P** = The set of all decision problems that can be solved by deterministic polynomial-time algorithms.
  - **Tractable problems!**
  - **Easy-to-find!**

The Class P

- All decision problems for which we have found poly-time algorithms are certainly in *P*.
  - Searching
  - Sorting
  - MST
  - Single-source shortest path
  - Fractional Knapsack
  - …

▶QUIZ?

- What is **P**?
- Example problems in **P**?

✓ Nondeterministic Algorithm

Nondeterministic Algorithm

- A **nondeterministic algorithm** is composed of two separate stages:
  1. Guessing (nondeterministic) Stage
  2. Verification (deterministic) Stage

Nondeterministic Algorithm

- **Guessing (nondeterministic) Stage**: Given an instance of a problem, this stage simply produces some string S.
  - The string can be thought of as a guess at the solution.
  - The guessing state is nondeterministic because unique, step-by-step instructions are not specified for it.

Nondeterministic Algorithm

- **Verification (deterministic) Stage**: The instance and the string *S* are the input to this stage.
  - This stage then proceeds in an ordinary deterministic manner either halting with an output of “true,” or halting with an output of “false,”.

Polynomial-Time Nondeterministic Algorithm

- A **polynomial-time nondeterministic algorithm** is a nondeterministic algorithm whose verification stage is a polynomial-time algorithm.
  - **Polynominal-time verifiability!**

✓ The Class NP

The Class NP

- **NP** = The set of all decision problems that can be solved by polynomial-time nondeterministic algorithms.
  - The **NP** stands for **Nondeterministic Polynomial.**
  - **Easy-to-check!**
  - The **NP** does not stand for **Non-Polynomial**

The CNF-Satisfiability Problem

- Example 9.6
  - A logical (Boolean) variable is a variable that can have one of two values: true or false.
  - A literal is a logical variable or the negation of a logical variable.
  - A clause is a sequence of literals separated by the logical or the operator (V).
  - A logical expression in conjunctive normal form (CNF) is a sequence of clauses separated by the logical and operator (^).

$$(P \vee \neg Q \vee W) \wedge (\neg P \vee T) \wedge (P \vee R \vee \neg T) \wedge (\neg R)$$

CNF-SAT is NP

- Example 9.7
  - It is easy to write a polynomial-time algorithm that takes as input a logical expression in CNF and a set of truth assignments to the variables and verifies whether the expression is true for that assignment.
  - Therefore, the CNF-Satisfiability Problem is in ***NP***.

The Class NP

- There are **many** problems proven to be in NP because polynomial-time nondeterministic algorithms have been developed for them.

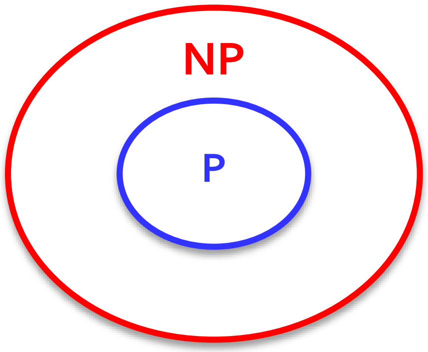
▶QUIZ?

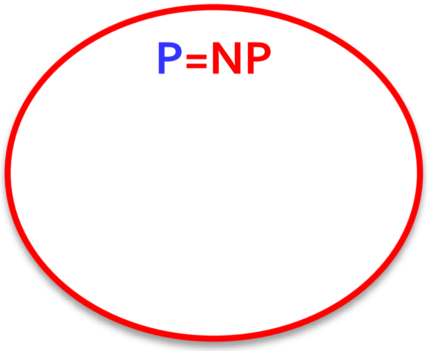
- What is **NP**?
- If a problem is in **P**, then is it in **NP**?

✓ Relationship Between *P* and *NP*

Relationship Between *P* and *NP*

- P is a subset of NP.
- **P $\subseteq$ NP !**





P ≠ NP?

P = NP?

Relationship Between *P* and *NP*

- P ≠ NP?
  - To show that *P* ≠ *NP*, we would have to find a problem in *NP* that is not in *P*.
- P= NP?
  - To show that *P* = *NP*, we would have to find a polynomial-time algorithm for each problem in *NP*.
- **We don't know!**
- **P vs NP is an open problem!**

P versus NP Problem

- A major unsolved problem in computer science.
- One of the seven Millennium US$1 Million Prize Problems!
- Most researchers doubt that *P* equals *NP*!

▶QUIZ?

- Is P = NP?
  - T or F - Unknown?

✓ Reduction

Reduction

- Transforming one problem into another problem.
- A reduction shows that the second problem is at least as difficult as the first.

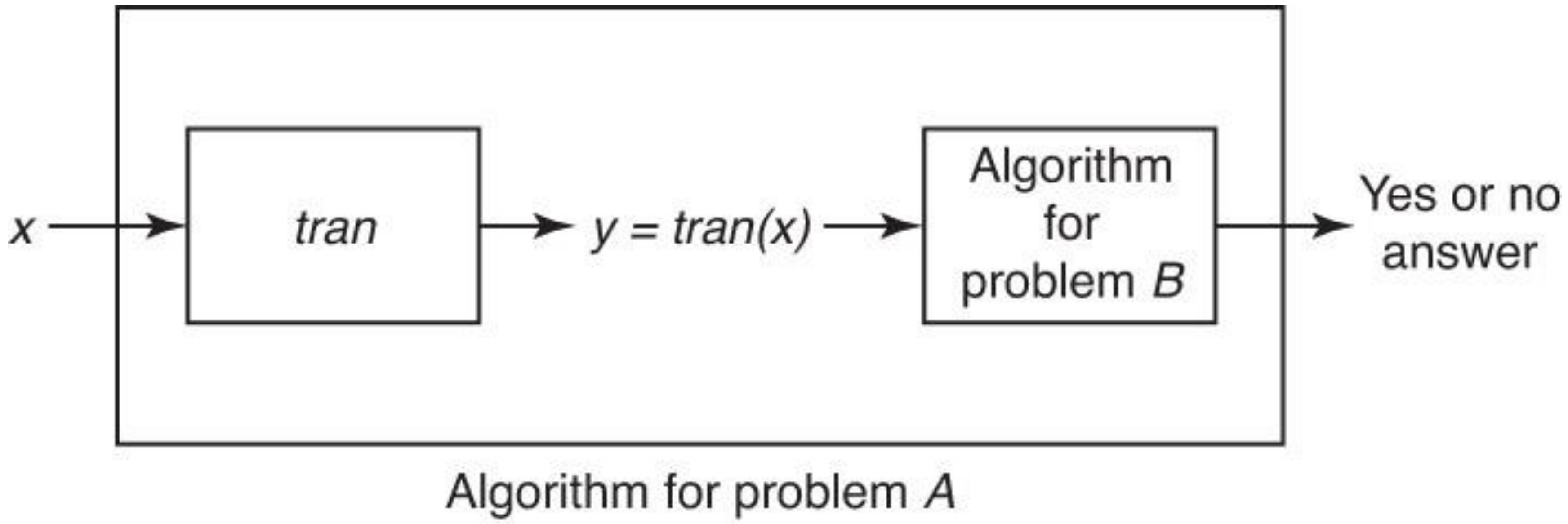
“Transform-and-Conquer”

Transformation Algorithm: A to B

- Suppose we want to solve decision problem A, and we have an algorithm that solves decision problem B.
- Suppose further that we can write an algorithm called a **transformation algorithm** that creates an instance *y* of Problem B from every instance *x* of Problem A such that an algorithm for Problem B answers "yes" for *x*.
- The transformation algorithm with an algorithm for B yields an algorithm for A!

Reduction/Transformation: A to B

Figure 9.4



Polynomial-Time Many-One Reduction : A to B

- **Polynomial-Time Many-One Reduction**:
  - If there exists a polynomial-time transformation algorithm from decision problem A to decision problem B,
    - **A is polynomial-time many-one reducible to B.**
    - **A reduces to B**
    - **A $\leq_p$ B**

B is at least as hard as A

Polynomial-Time Many-One Reduction

- If $A \leq_p B$ and B is in P, then A is in P.
  - If there is a polynomial time algorithm for B, then there is a polynomial time algorithm for A.

Polynomial-Time Many-One Reduction

- If $A \leq_p B$ and $B \leq_p C$, then $A \leq_p C$.
- **Transitivity of Reductions**

▶QUIZ?

- What is Polynomial-time Many-one Reduction?
- When is it used?

✓ NP-Complete Problems

NP-Complete Problems

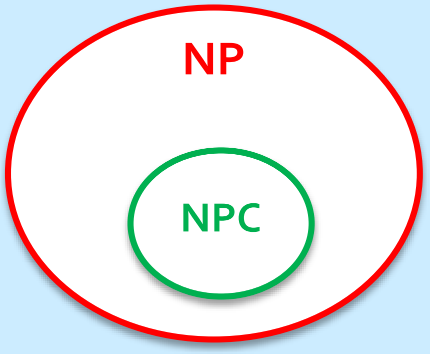
- The Traveling Salesperson decision problem and many other problems are equally hard
  - In the sense that if we had an efficient algorithm for any one of them, we would have efficient algorithms for all of them.
- This kind of problem is called **NP-complete** problems.
  - **complete?**
    - **Solve one, Solve all!**

✓ NP-Complete Problems

- NP-complete problems:
  - The hardest problems in NP and in NP.
  - The toughest problems in NP in the sense that they are the ones most likely not to be in P.
- If we could show that any NP-complete problem is in P, we could conclude that P=NP!
  - If one NP-Complete problem can be solved in polynomial time, then all NP problems can solved in polynomial time.

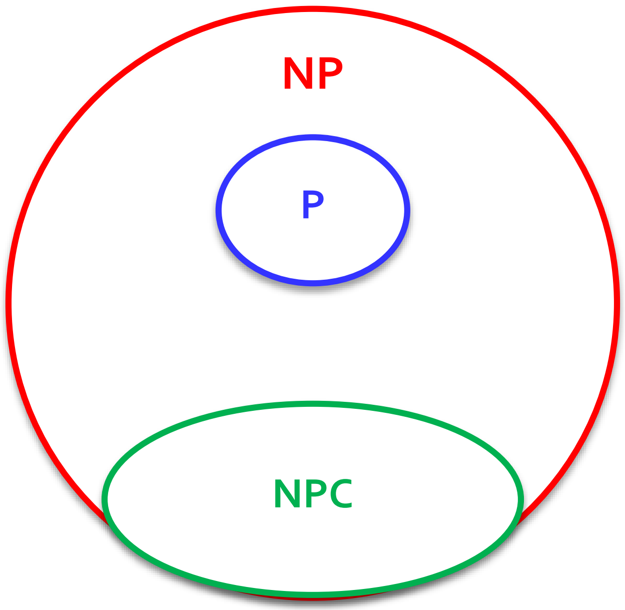
NP-Complete Problems

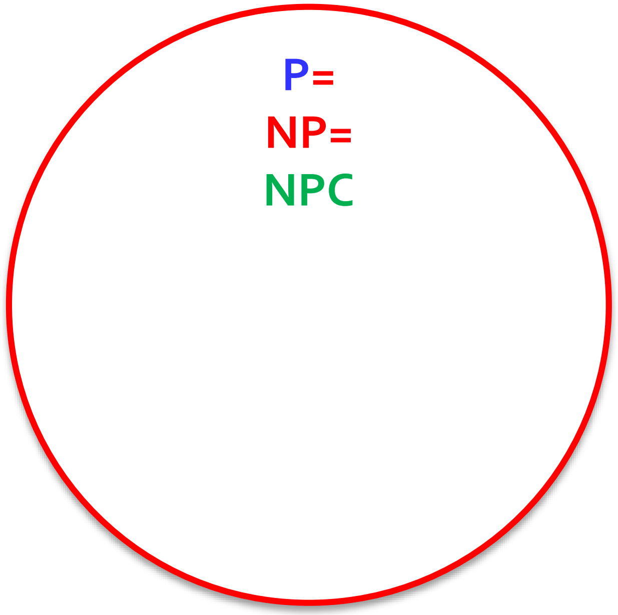
- A problem B is called **NP-complete (NPC)** if
  1. **B is in NP.**
  2. **For every other problem A in NP, A $\leq_p$ B.**



B is at least as hard as every problem in NP.

✓ Relationship Between *P*, *NP* and *NPC*





P ≠ NP?

P = NP?

CNF-SAT: The First NP-Complete Problem

The Satisfiability (SAT) Problem:
"Given a Boolean expression, is it satisfiable?"

CNF-SAT = { *w* : *w* is a wff in Boolean logic, *w* is in conjunctive normal form, & *w* is satisfiable}

$(P \vee \neg Q \vee W) \wedge (\neg P \vee T)$

- **The first problem proved as NP-Complete!**

CNF-SAT is NP-Complete

- Theorem 9.2 **The Cook-Levin (Cook's) Theorem**
  - The CNF-Satisfiability (SAT) Problem is NP-complete.

3-SAT is NP-Complete

The 3-SAT Problem is NP-complete.

3-SAT = {<*w*> : *w* is a wff in Boolean logic, *w* is in 3-conjunctive normal form and *w* is satisfiable}.

$$(P \lor R \lor \neg T) \land (S \lor \neg R \lor W)$$

NP-Complete Problems

- Hundreds of problems have been shown to be NP-Complete!
  - **0-1 Knapsack**
  - **TSP**
  - **Graph coloring**
  - **Hamiltonian path**
  - **…**

✓ How to Show New Problems are NP-Complete Problems?

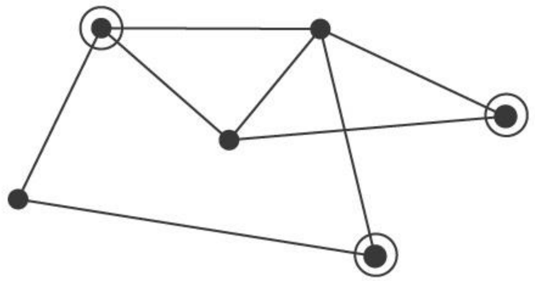
How to Show NP-Complete Problems

- **Theorem 9.3**
- We can **show that a new problem A is NP-complete** by showing that
  1. **A is in NP**
  2. **Choose some NP-complete problem C & Reduce C to A, i.e., C $\leq_p$ A.**

Example: INDEPENDENT-SET is NP-Complete?

INDEPENDENT-SET = {<*G*, *k*> : *G* is an undirected graph and *G* contains an independent set of *at least k* vertices}.

An *independent set* is a set of vertices *no two of which are adjacent.*



QUESTION: Is INDEPENDENT-SET NP-Complete?

Example: INDEPENDENT-SET is NP-Complete?

Proof Idea:

- ✓ INDEPENDENT-SET is in NP and
- ✓ INDEPENDENT-SET is NP-hard by

  3-SAT $\leq_P$ INDEPENDENT-SET

▶QUIZ?

- What is **NP-Complete (NPC)**?
- Example NPC Problems?
- NP-complete problems are the hardest problems in NP?

▶QUIZ?

- How to show that a problem A is NP-complete?
  - By showing that
    1. A is in NP
    2. Some NP-complete problem C reduces to A, i.e., $C \leq_p A$.

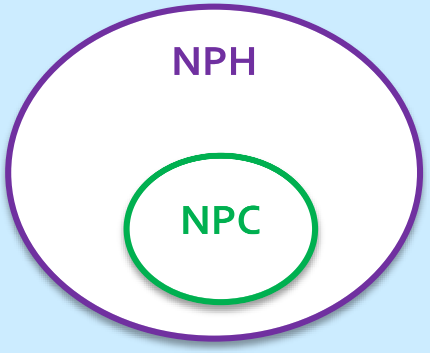
▶QUIZ?

- If anyone finds a polynomial-time algorithm for even one NP-complete problem, then that would imply a polynomial-time algorithm for every NP-complete problem?

✓ NP-Hard Problems

NP-Hard Problems

- **NP-hard**:
  - At least as hard as the hardest problems in NP meaning all **NP**-problems can be reduced to them, but need not be in NP; indeed, they may not even be decision problems.
  - NP-complete is a subset of NP Hard.
  - **NP-C $\subseteq$ NP-Hard**



NP-Hard Problems

- A problem B is called **NP-hard (NPH)** if for every other problem A in NP, A $\leq_p$ B.

NP-Hard Problems

- **All** NP-Complete problems are NP-Hard.

NP-Hard Problems

- Not all **NP-hard** problems are in **NP**, meaning not all of them have solutions verifiable in polynomial time.

- Very often, the NP-hard problems require exponential time or even worse!

  - Example: **The Towers of Hanoi Problem**

▶QUIZ?

- What is **NP-Hard (NPH)**?
- Is NP-complete (NPC) a subset of NP-hard (NPH)?
- NP-complete (NPC) vs. NP-hard (NPH)?

✓ Handling *NP*-Complete/Hard Problems

- In the absence of polynomial-time algorithms for problems known to be *NP*-hard, what can we do about solving such problems?

1. Handling *NP*-Hard Problems

- **One way** is the **backtracking and branch-and-bound algorithms**,
  - Which are all worst-case non-polynomial-time.
  - However, they are often efficient for many large instances.

2. Handling *NP*-Hard Problems

- **Another approach** is to find an algorithm that is efficient for a **subclass of instances** of an *NP*-hard problem.

3. Handling *NP*-Hard Problems

- **A third approach** is to develop **approximation algorithms.**
  - An algorithm that is not guaranteed to give optimal solutions, but rather yields solutions that are reasonably close to optimal.

✓ P vs. NP-Complete: Similar Problems

1. Circuit problems
2. Path problems
3. Map coloring problems

1. Two Similar Circuit Problems

- EULERIAN-CIRCUIT, in which we check that there is a circuit that *visits every edge exactly once*, is in P.
  - A connected graph possess an Eulerian circuit iff all its vertices have even degree.
- HAMILTONIAN-CIRCUIT, in which we check that there is a circuit that *visits every vertex exactly once*, is NP-complete.

2. Two Similar Path Problems

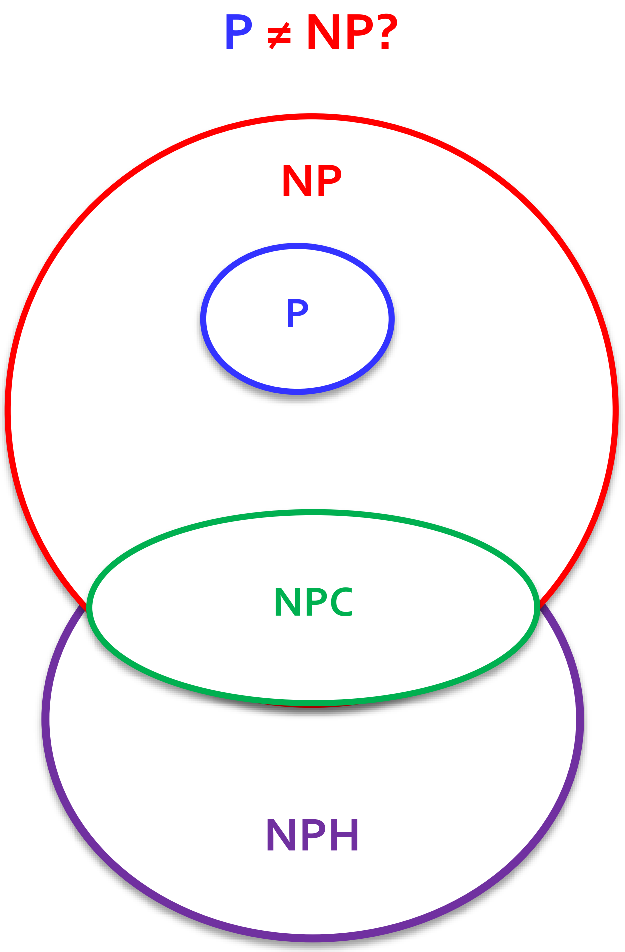
- SHORTEST-PATH = {<*G*, *u*, *v*, *k*>: *G* is an undirected graph, *u* and *v* are vertices in *G*, $k \geq 0$, and there exists a path from *u* to *v* whose length is at most *k*} is in P.

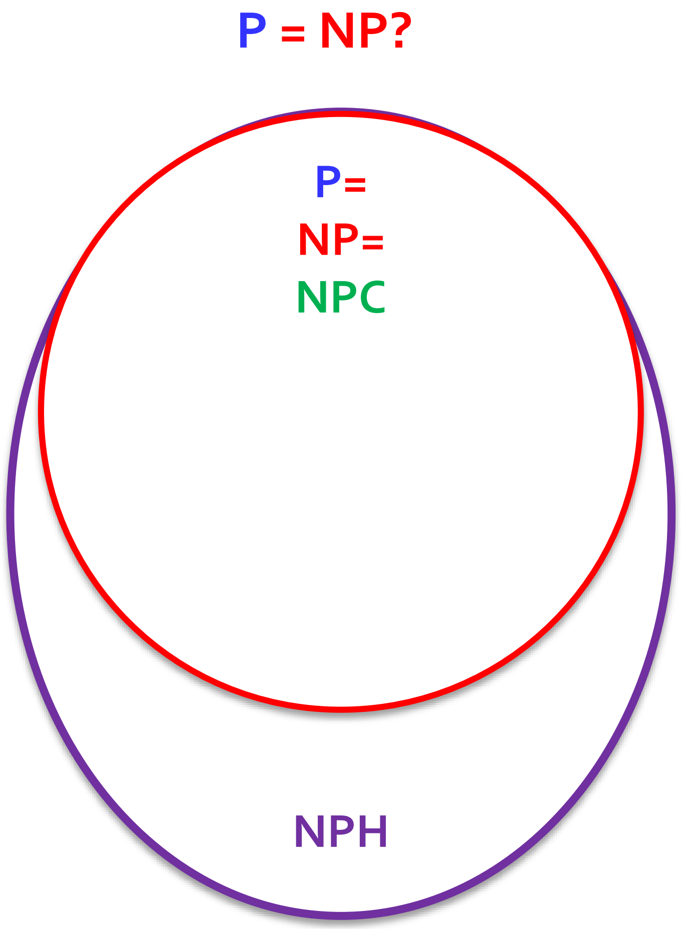
- LONGEST-PATH = {<*G*, *u*, *v*, *k*>: *G* is an undirected graph, *u* and *v* are vertices in *G*, $k \geq 0$, and there exists a path with no repeated edges from *u* to *v* whose length is at least *k*} is NP-complete.

3. Two Similar Map Coloring Problems

- 2-COLORABLE = {<m> : m can be colored with 2 colors} is in P.
- 3-COLORABLE = {<m> : m can be colored with 3 colors} is NP-complete.

▶QUIZ? P, NP, NPC and NPH





✓ Intractable Problems Summary

- 3 General Problem Categories
- P
- NP
- NP-complete
- Reduction
- P=NP?
- NP-hard
- Handling NP-hard problems

Homework Assignment

▶ Homework Assignment?

- Chapter 9: Exercise #1, #6 & #15.

✓ Textbook Readings

- Chapter 9:
  - 9.1
  - 9.3
  - 9.4

the right majors, minors & concentr
ucation?
or students' academic and career success
ing for many students - *Many students change their*
uring college!

e prediction of student success in MMC could
dual students
d their right MMC
chieve their academic goals

Y. PARK • DEPT OF ESBIS, BRADLEY UNIVERSITY
1/2

END

Prof. Young Park