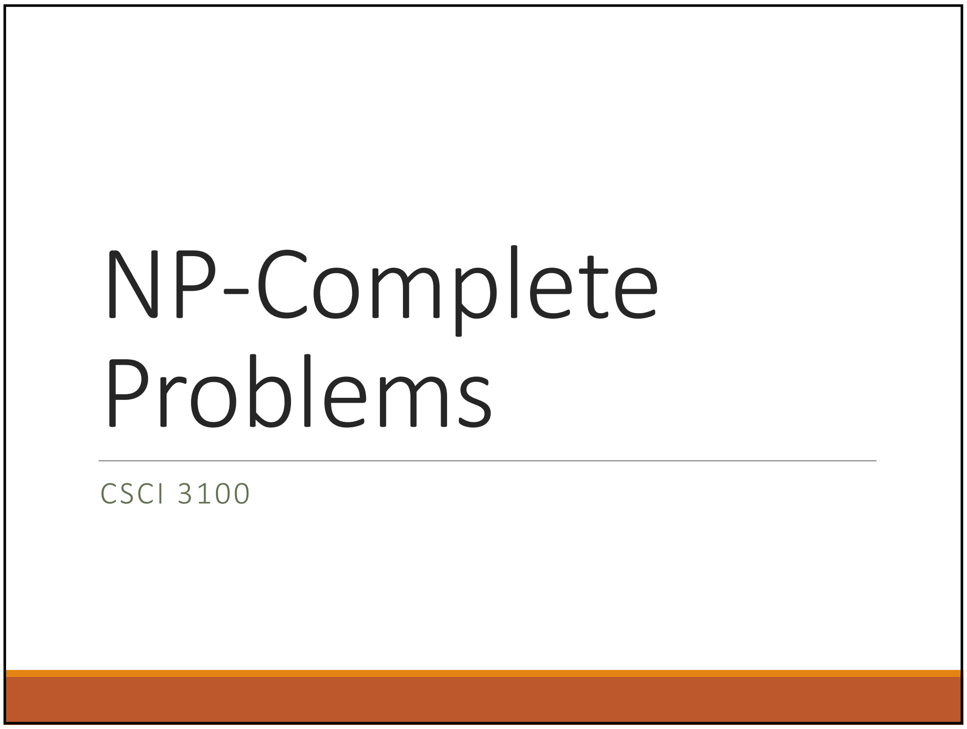
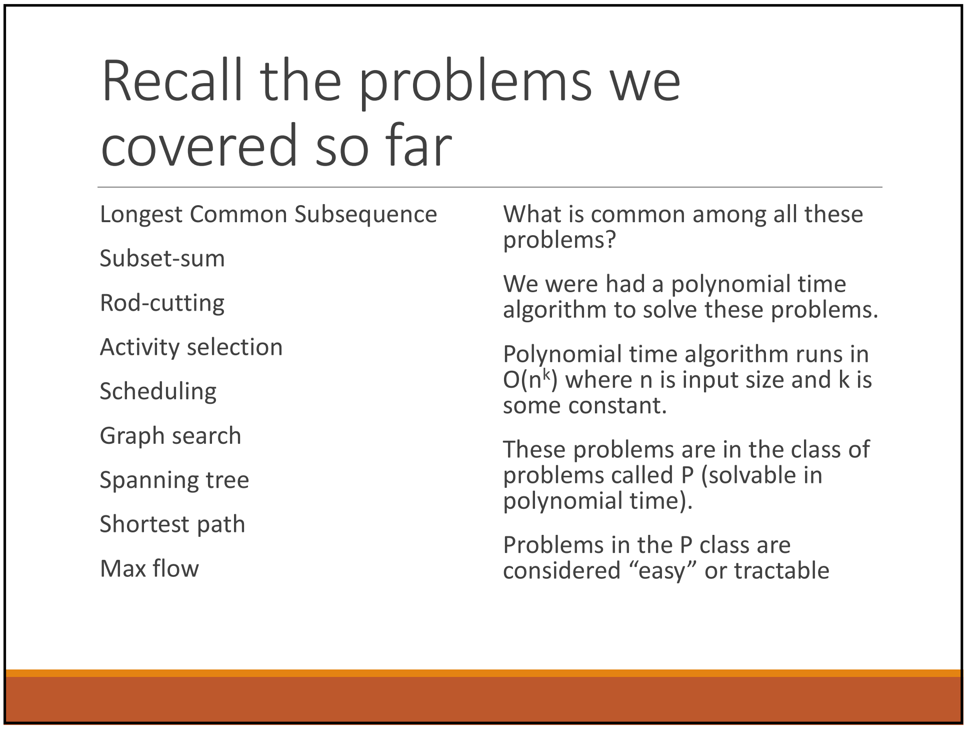


NP-Complete Problems

CSCI 3100



Recall the problems we covered so far

Longest Common Subsequence

Subset-sum

Rod-cutting

Activity selection

Scheduling

Graph search

Spanning tree

Shortest path

Max flow

What is common among all these problems?

We were had a polynomial time algorithm to solve these problems.

Polynomial time algorithm runs in $O(n^k)$ where n is input size and k is some constant.

These problems are in the class of problems called P (solvable in polynomial time).

Problems in the P class are considered “easy” or tractable

Suppose we have a problem with input size m and several algorithms for this problem with different run times shown below. Which one is NOT a polynomial time algorithm?

A. m^2+m

B. $m^3 + 3^m$

C. m lg (m)

D. m+C (C is a constant)

E. m^{1000}

Some problems do NOT have a solution, no matter how much time we allow

Turing’s HALTING PROBLEM

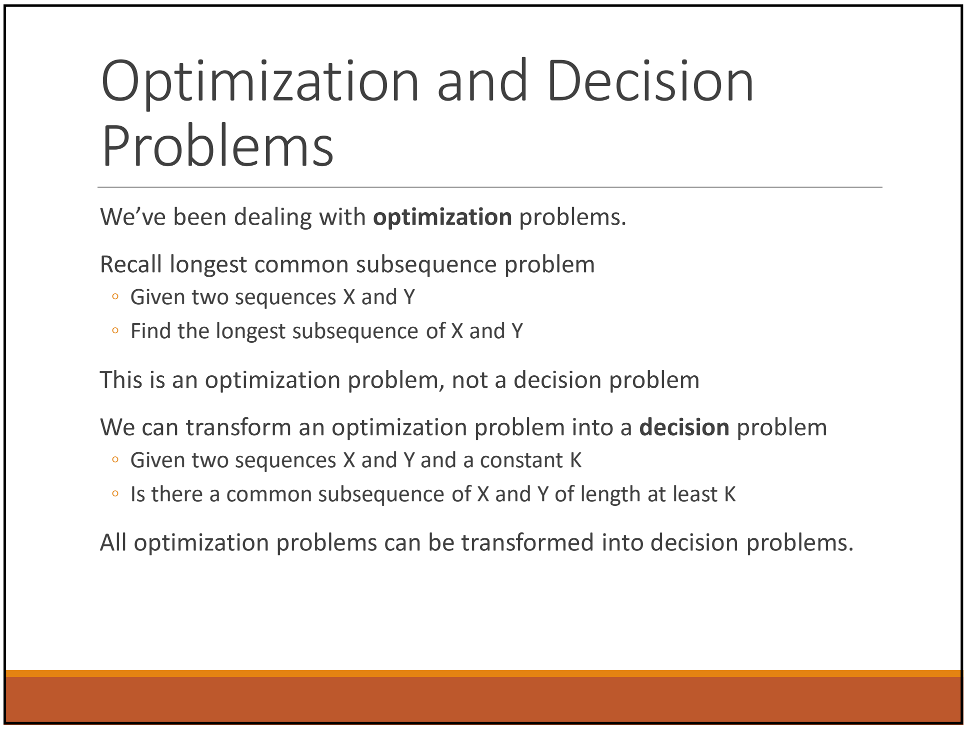
- Given arbitrary computer program and input to the program
- Determine whether the program will terminate (halt) or continue to run forever

This is a decision problem: given the input, provide a yes/no output.

Alan Turing proved that there cannot be an algorithm that can solve the HALTING PROBLEM for all possible inputs.

So the HALTING PROBLEM is very hard (especially, as compared to the problems in the class P)

The name for such problems is *undecidable*



Optimization and Decision Problems

We've been dealing with **optimization** problems.

Recall longest common subsequence problem

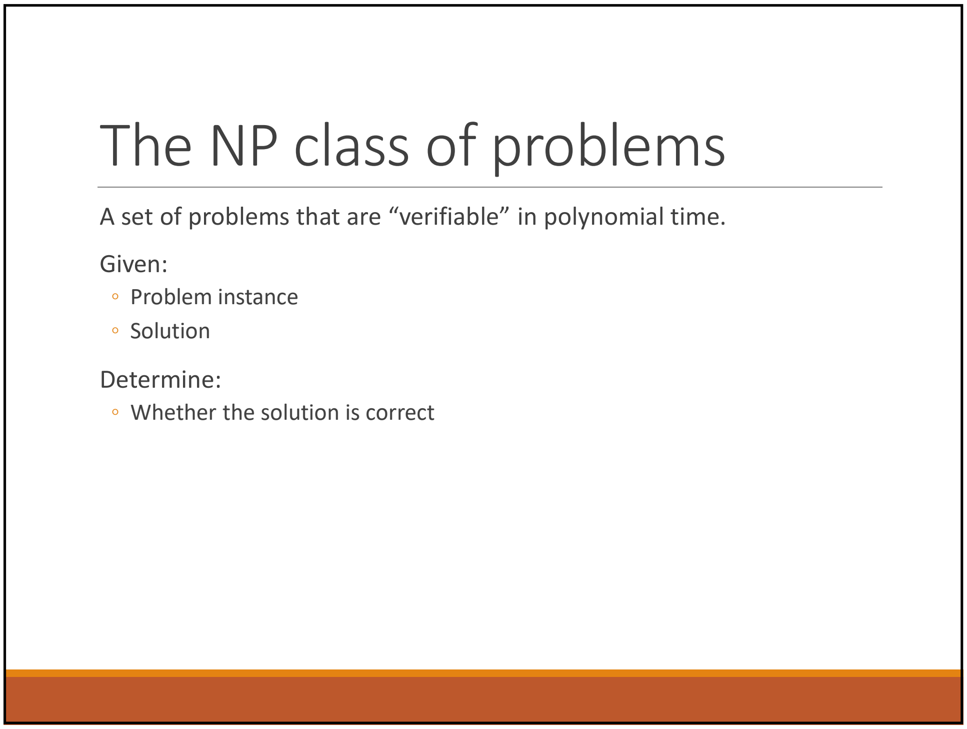
- Given two sequences X and Y
- Find the longest subsequence of X and Y

This is an optimization problem, not a decision problem

We can transform an optimization problem into a **decision** problem

- Given two sequences X and Y and a constant K
- Is there a common subsequence of X and Y of length at least K

All optimization problems can be transformed into decision problems.



The NP class of problems

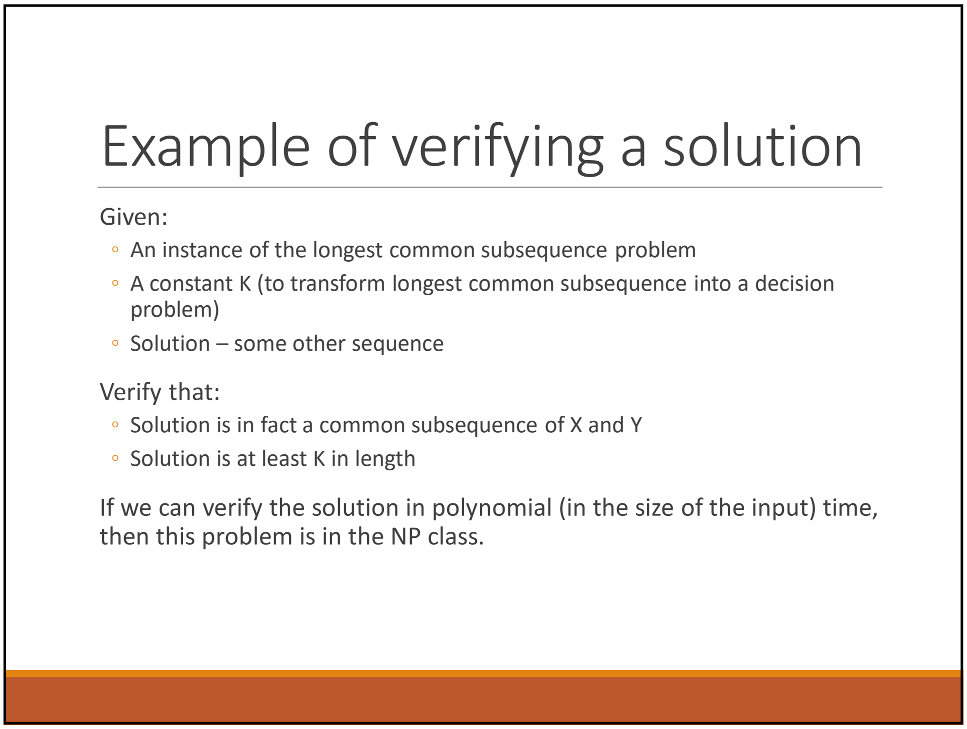
A set of problems that are "verifiable" in polynomial time.

Given:

- Problem instance
- Solution

Determine:

- Whether the solution is correct



Example of verifying a solution

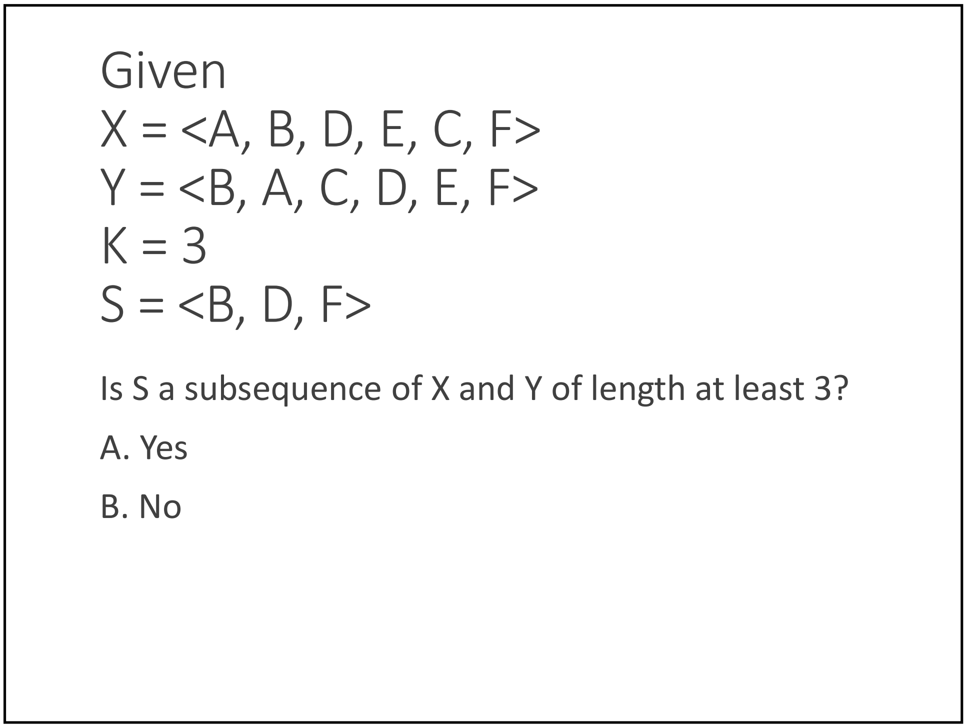
Given:

- An instance of the longest common subsequence problem
- A constant K (to transform longest common subsequence into a decision problem)
- Solution – some other sequence

Verify that:

- Solution is in fact a common subsequence of X and Y
- Solution is at least K in length

If we can verify the solution in polynomial (in the size of the input) time, then this problem is in the NP class.



Given

X = <A, B, D, E, C, F>

Y = <B, A, C, D, E, F>

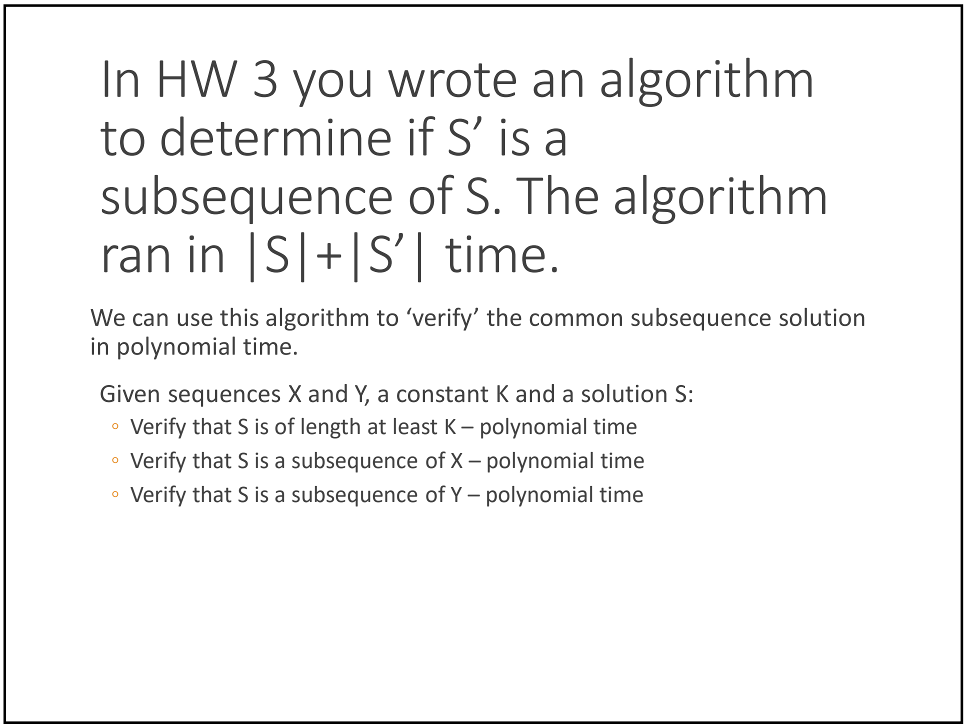
K = 3

S = <B, D, F>

Is S a subsequence of X and Y of length at least 3?

A. Yes

B. No

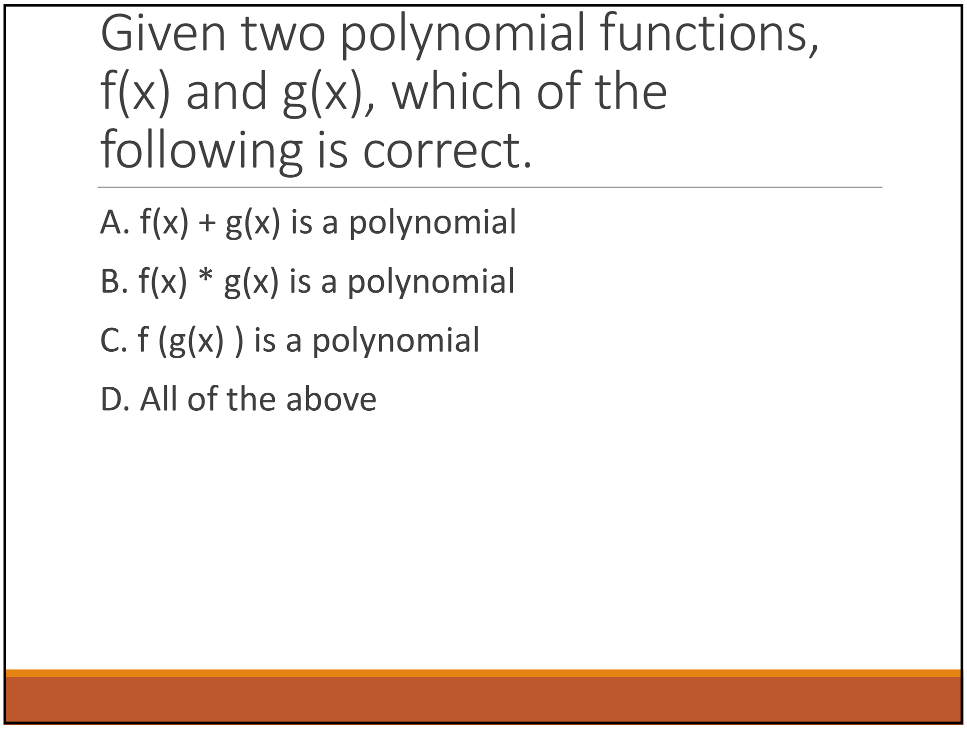


In HW 3 you wrote an algorithm to determine if S’ is a subsequence of S. The algorithm ran in |S|+|S’| time.

We can use this algorithm to ‘verify’ the common subsequence solution in polynomial time.

Given sequences X and Y, a constant K and a solution S:

- Verify that S is of length at least K – polynomial time
- Verify that S is a subsequence of X – polynomial time
- Verify that S is a subsequence of Y – polynomial time



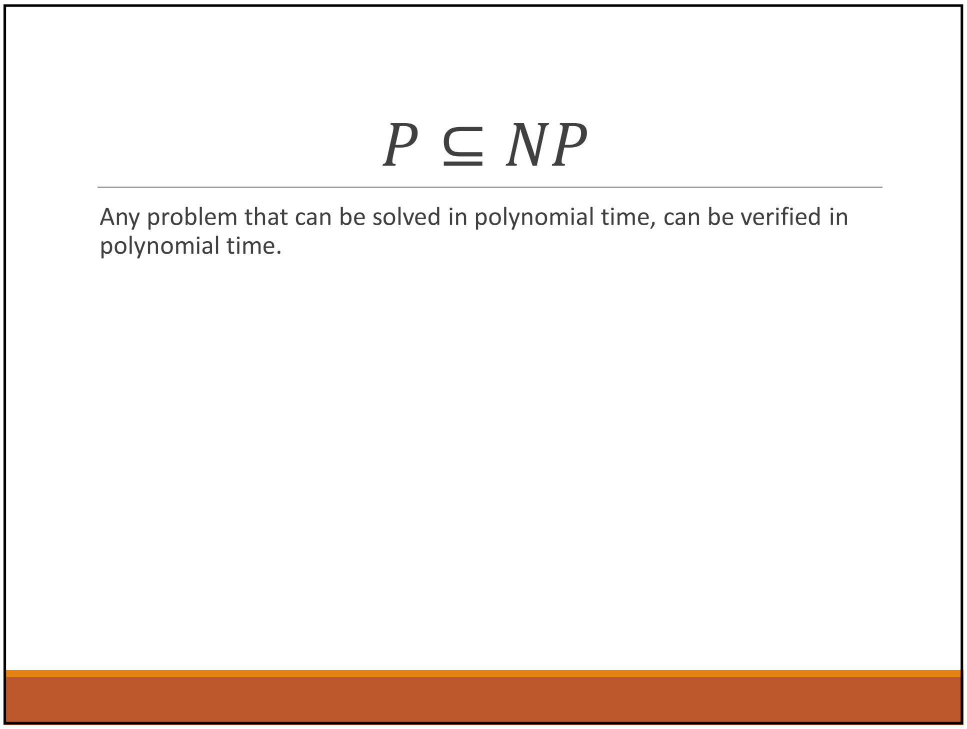
Given two polynomial functions, f(x) and g(x), which of the following is correct.

A. f(x) + g(x) is a polynomial

B. f(x) * g(x) is a polynomial

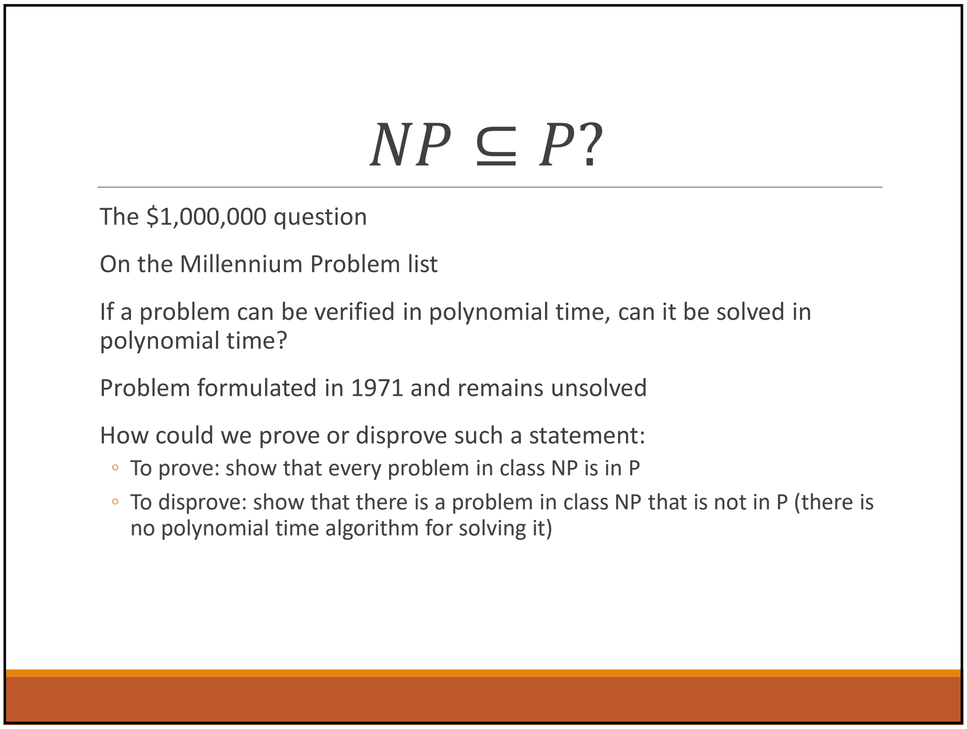
C. f (g(x)) is a polynomial

D. All of the above



$P \subseteq NP$

Any problem that can be solved in polynomial time, can be verified in polynomial time.



$NP \subseteq P?$

The $1,000,000 question

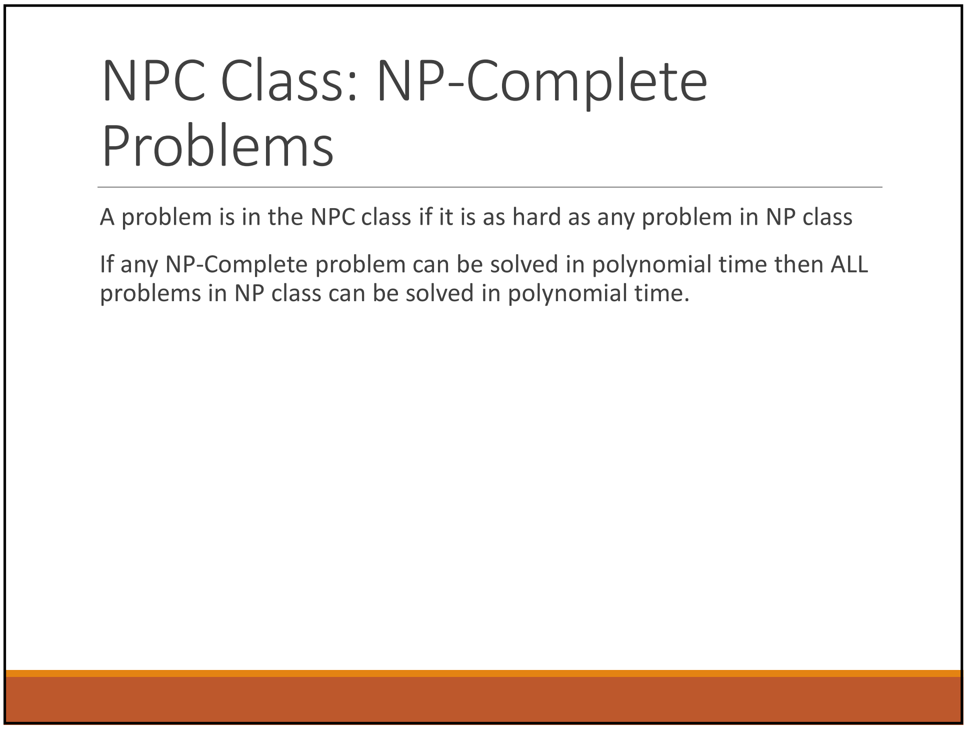
On the Millennium Problem list

If a problem can be verified in polynomial time, can it be solved in polynomial time?

Problem formulated in 1971 and remains unsolved

How could we prove or disprove such a statement:

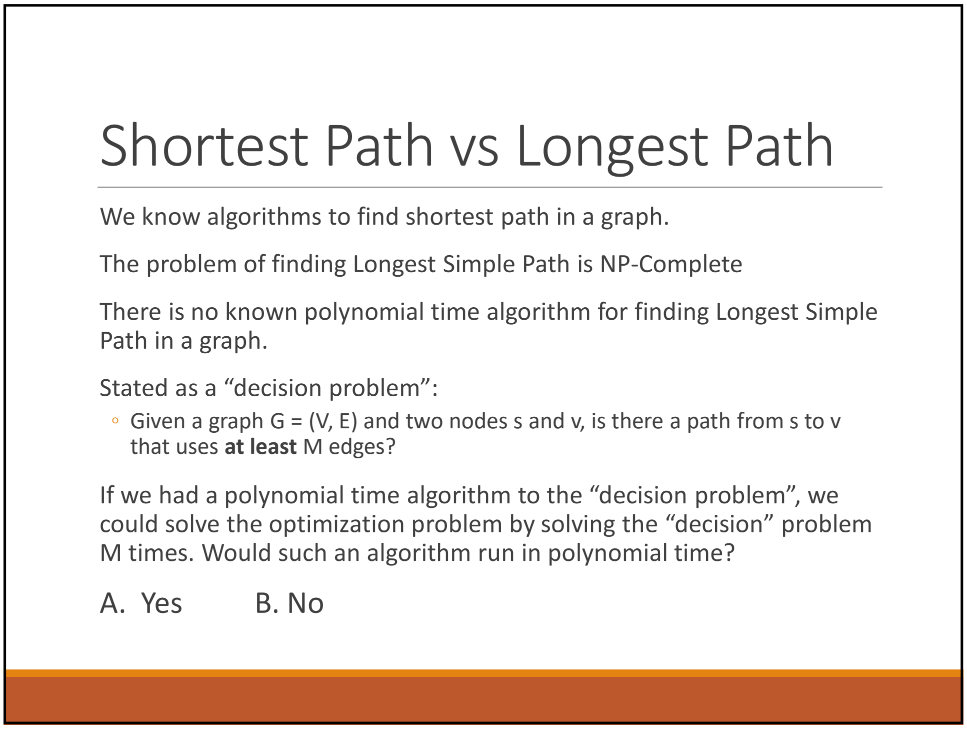
- To prove: show that every problem in class NP is in P
- To disprove: show that there is a problem in class NP that is not in P (there is no polynomial time algorithm for solving it)



NPC Class: NP-Complete Problems

A problem is in the NPC class if it is as hard as any problem in NP class

If any NP-Complete problem can be solved in polynomial time then ALL problems in NP class can be solved in polynomial time.



Shortest Path vs Longest Path

We know algorithms to find shortest path in a graph.

The problem of finding Longest Simple Path is NP-Complete

There is no known polynomial time algorithm for finding Longest Simple Path in a graph.

Stated as a “decision problem”:

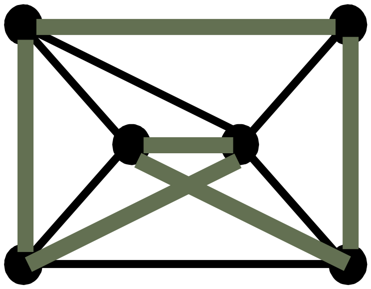
- Given a graph G = (V, E) and two nodes s and v, is there a path from s to v that uses **at least** M edges?

If we had a polynomial time algorithm to the “decision problem”, we could solve the optimization problem by solving the “decision” problem M times. Would such an algorithm run in polynomial time?

A. Yes B. No

Hamilton Cycle

Given a graph G = (V,E), a cycle that visits all the nodes exactly once



Euler Tour vs Hamiltonian Cycle

Euler Tour of a strongly connected directed graph G = (V, E) is a cycle that traverses each edge of G exactly once (each vertex can be visited more than once).

- The problem of determining if a graph has an Euler Tour is in P.

Hamiltonian Cycle of a directed graph G = (V, E) is a simple cycle that contains each vertex in V.

- The problem of determining if a graph has a Hamiltonian Cycle is NP-Complete.

If there is a polynomial time algorithm for solving Hamiltonian Cycle, there is a polynomial time algorithm for finding longest path in a graph.