CIS 606 Analysis of Algorithms

Complexity: P, NP and NPC

RATIONALE

- All the algorithms we have studied thus far have been polynomial-time algorithms: on input of size n, their worst-case running time is $O(n^k)$.
- Can all problems be solved in polynomial time?

OBJECTIVES

- Understand P, NP, NPC definitions.

PRIOR KNOWLEDGE

- Sets
- Automata and formal language

OPTIMIZATION PROBLEMS

- Optimization Problem: the answer of the problem is a feasible solution with the best (minimum or maximum) value.
  - Knapsack problem
  - Single-source shortest path
  - Maximum flow
- Dynamic programming
- Divide-and-conquer
- Prune-and-search

DECISION PROBLEMS

- Decision Problem: the problem of determining an answer to a class of yes/no questions.
  - Given a graph G, vertices u and v, an integer k, does a path exist from u to v consisting of at most k edges?
  - Dose a graph have a path that goes through every node exactly once?
  - Is the number x prime?
- A solution to a decision problem is given by an algorithm (e.g., a turing machine automata) that answers yes or no.

ENCODING INSTANCES TO A SET OF BINARY STRINGS

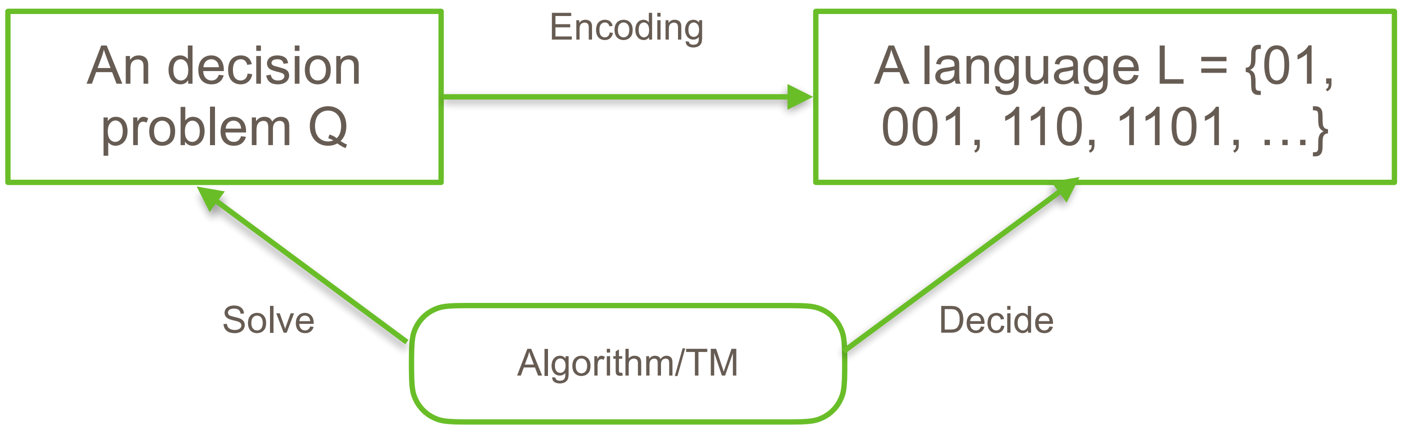
- An instance of a problem is the input to a particular problem
  - E.g., a particular graph G, particular vertices u and v of G, and a particular integer k for the decision problem Path: whether there is a path from u to v of at most k edges.
- An encoding of a set S of abstract objects is a mapping from S to the set of binary strings.

AN ALGORITHM

An algorithm for solving a decision problem is a Turing Machine for deciding the corresponding language.

One Turing machine decides a language L = {01001, 001, 001, 100}

- The Turing machine outputs 1 for every binary string in L, i.e., accepts L
- It outputs 0 for every one not in L, i.e., rejects any binary string not L.

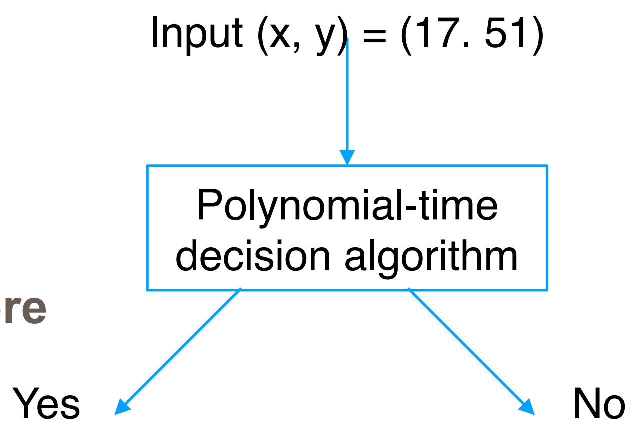


ISSUES FOR DECIDING A LANGUAGE

- Computability Issue:
  - Does it have an algorithm at all?
  - Question the existence of an algorithm (Turing machine)
- Complexity Issue:
  - Does it have an efficient solution (algorithm)?
  - Is there algorithms with the running time scales well with the input size?

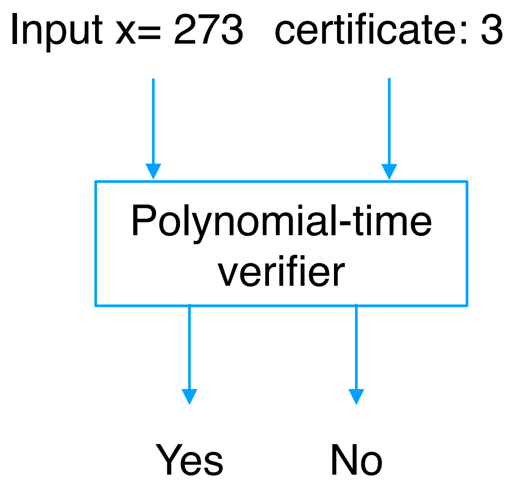
P AND NP

- P: the set of languages decided by an algorithm in polynomial time, i.e., decided in polynomial time on a deterministic Turing machine (deterministic algorithms).
  - Multiple: is the integer y a multiple of x ?
    - Yes: (x, y) = (17, 51)
  - Given integers x1, x2, …, xn, is the median value < M?
    - No: (M, x1, x2, x3, x4, x5) = (17, 82, 5, 104, 22, 10)
  - Given a graph G, two vertices u, v, and an integer k, is there
  - a path between u and v of no more than k edges?



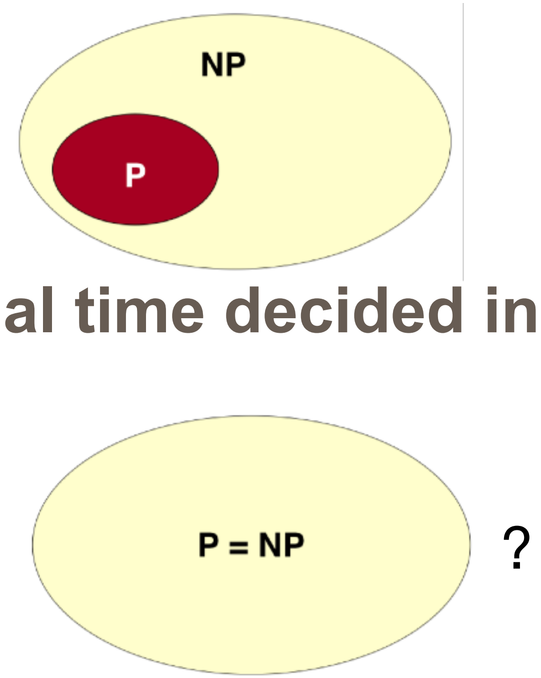
P AND NP(CONT)

- NP (not mean “not polynomial”): the set of languages that can be verified by a polynomial-time algorithms.
  - Given a certificate of a solution of a decision problem, an algorithm verifies this solution in polynomial time.
    - E.g., decision problem: given integer x, is x composite?
    - Given an integer x = 273 and a certificate k=3, deciding whether x=273 is composed of k=3 can be done in polynomial-time verification algorithm.
- Or the set of all decision problems solvable in polynomial time on a nondeterministic Turing machine.
  - A nondeterministic TM is the one that can explore many, many paths of computation in parallel.



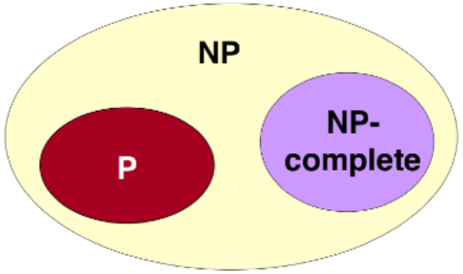
OPEN PROBLEM: P = NP?

- $P \subseteq NP$:
  - A language that can be decided in polynomial time can be verified in polynomial time.
- How about $NP \subseteq P$?
  - Is a language that can be verified in polynomial time decided in polynomial time?

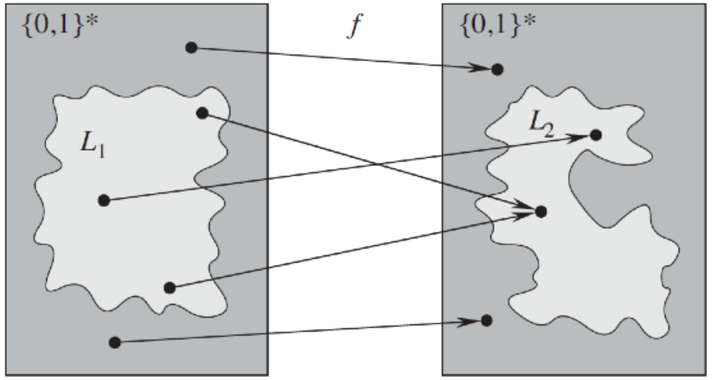


NP-COMPLETENESS AND NP-HARD

- Reducibility:
- $L_1 \leq_p L_2$: the reduction function f maps any instance x of the decision problem represented by L1 to an instance f(x) of the decision problem represented by L2.
- A language L is **NP-Complete** (NPC) if
  - L is in NP, and
  - Every language L' in NP is polynomial-time reducible to L, i.e., $L' \leq_p L$
- A language L is **NP-Hard** if $L' \leq_p L$ for every L' in NP.

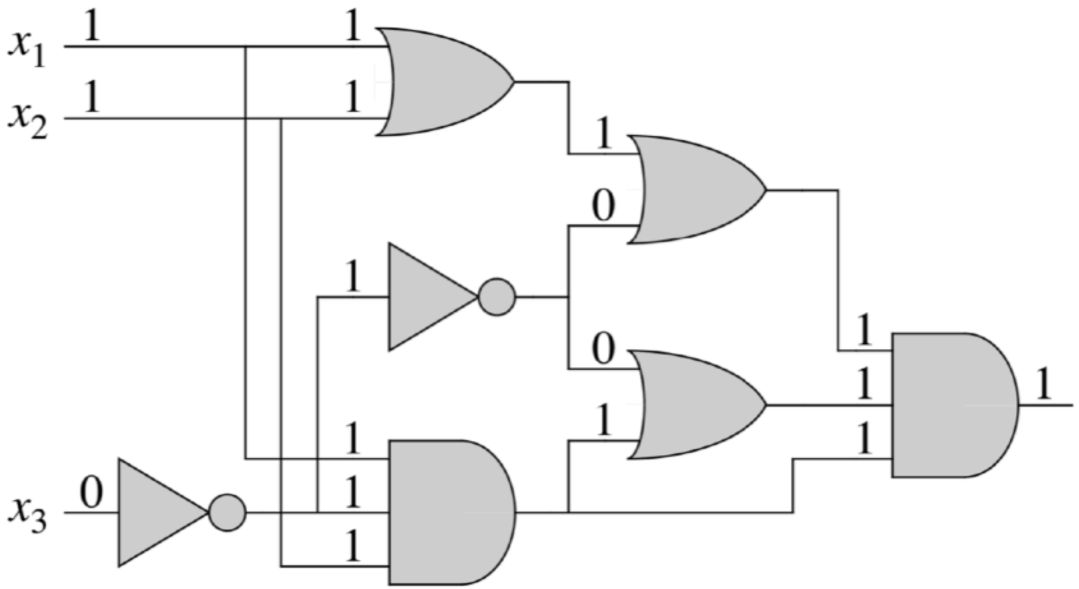


If P ≠ NP



THE FIRST NPC Problem

- Circuit Boolean Satisfiability Problem (SAT)
  - An instance is a boolean combinatorial circuit.
- Question: is there a satisfying assignment, i.e., an assignment of inputs, to the circuit that satisfies it (makes its output 1) ?



BOOLEAN SATISFIABILITY PROBLEM (SAT)

- The given is
  - A Boolean Formula $F(x_1, x_2, \ldots, x_n)$ in conjunctive normal form (CNF)
- Question: does the given formula have a satisfying assignment?
  - E.g., $F = (x_1 \vee x_4 \vee x_6 \vee \neg x_n) \wedge (\neg x_1 \vee x_2 \vee \neg x_4 \vee x_8 \vee \neg x_n)$

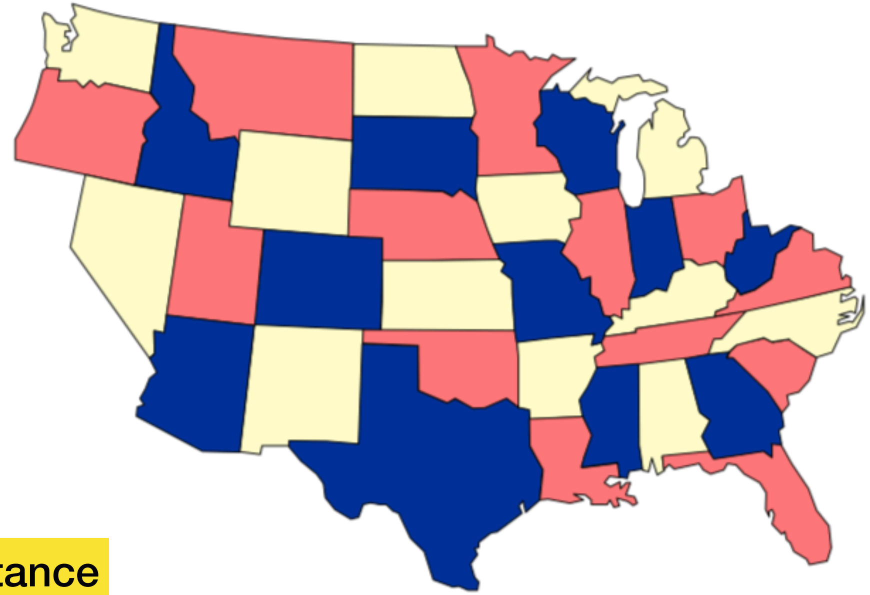
    $\wedge (\neg x_3 \vee x_9 \vee \neg x_{13} \vee x_{24} \vee \neg x_{n-1}) \ldots$

3-CNF-SAT

- Given:
  - A Boolean Formula $F(x_1, x_2, \ldots, x_n)$ in conjunctive normal form (CNF) and each clause has 3 variables.
- Question: does the given formula has a satisfying assignment?
  - E.g., $F = (x_1 \vee x_4 \vee x_6) \wedge (\neg x_1 \vee x_8 \vee \neg x_n) \wedge (\neg x_3 \vee x_9 \vee \neg x_{13}) \ldots$

GRAPH 3-COLOR

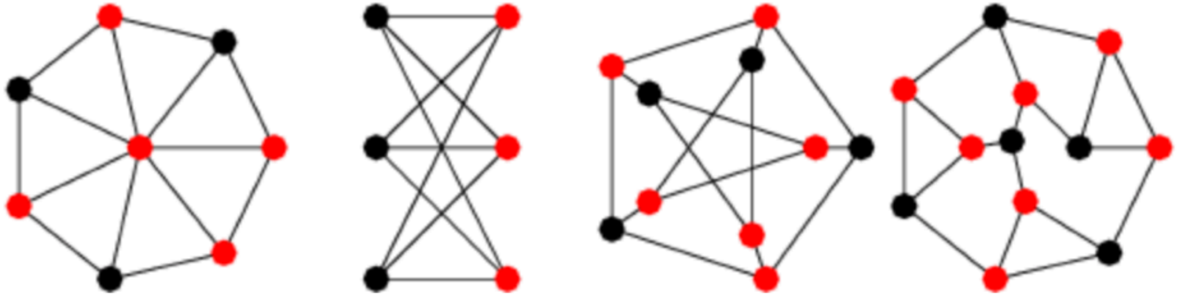
- Given a planar map, can it be colored using 3 colors so that no adjacent regions have the same color?



Yes instance

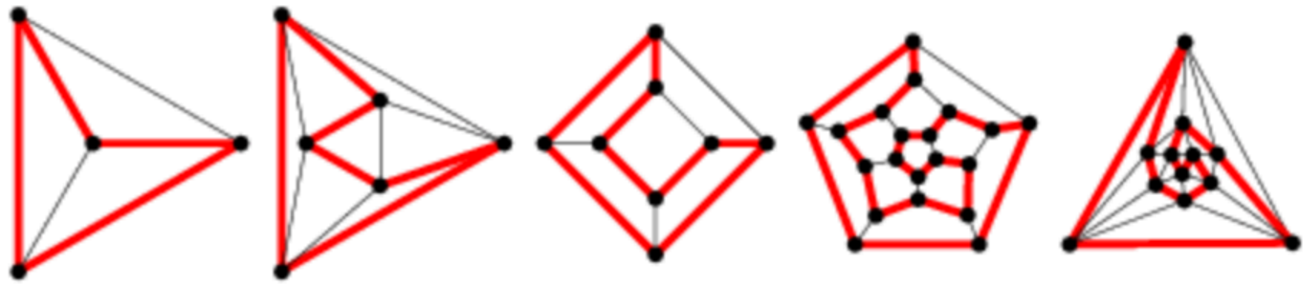
VERTEX COVER

- A vertex cover of a graph is a set of vertices such that each edge is incident to at least one vertex of this set.
- The NP-complete problem:
- Given a graph G(V,E) and a positive integer k, the problem is to find whether there is a vertex cover of size at most k.



HAMILTONIAN CYCLE

- Hamiltonian cycle in an undirected graph is a graph cycle that visits each vertex exactly once.
- The problem:
- Given any undirected graph, is there a hamiltonian cycle in this graph?



NPC PROOF

- To prove that a problem B is NPC:
  - B is in NP
  - Choose some known NPC problem A, define a polynomial transformation from A to an instance B to show that $A \leq_p B$

SUMMARY

- P is a set of decision problems that can be solved in polynomial time.
- NP is a set of decision problems that can be verified in polynomial time.
- NPC is a subset of NP and as hard as other problems in NP
- NP-Hard is the set of problems that every NP problem can be reduced to one of it.