What is P

- The set of all problems that can be “solved” in polynomial time
- Technically, an algorithm with a running time of $O(n^{100})$ belongs to P
- However, most of the algorithms in P could be solved within a very short time complexity (for example, all the problems discussed so far were not more than $O(n^2)$)

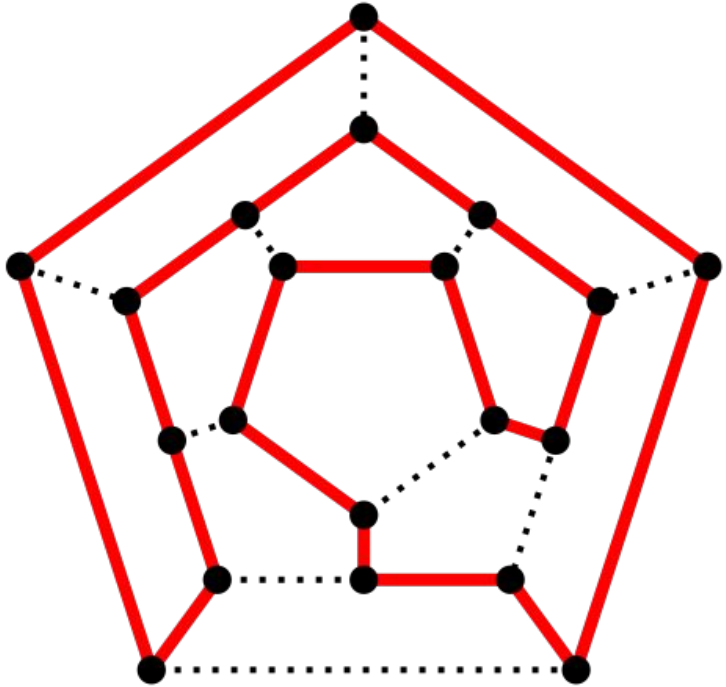
What is NP

- The set of all problems that can be “verified” in polynomial time

- If we have a certificate of a solution, that certificate can be verified in polynomial time

Certificate for Hamiltonian cycle problem:
Given a graph G=(V, E), a certificate would be a sequence of vertices

Hamiltonian Cycle



Decision problem vs Optimization problem

Optimization problem tries to find the maximum or minimum value, whereas the answer to decision problem is simply yes or no

A decision problem related to SHORTEST-PATH is PATH:
Given a directed graph G, vertices u and v, and an integer k, does a path exist from u to v consisting of at most k edges?

Instance

We call the input to a particular problem an instance of that problem

In PATH, an instance would be a particular graph G, particular vertices u and v of G, and a particular integer k

Reduction

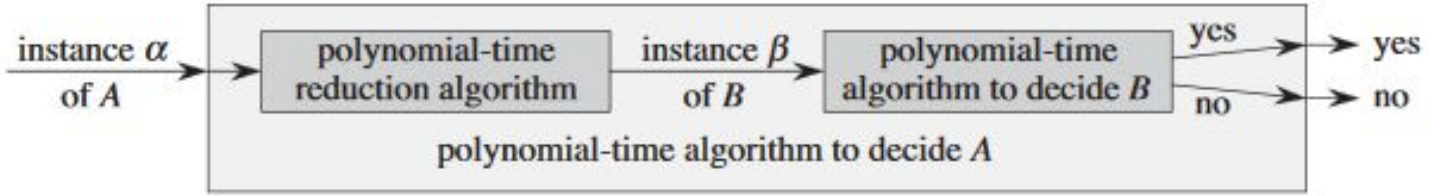


Figure 34.1 How to use a polynomial-time reduction algorithm to solve a decision problem A in polynomial time, given a polynomial-time decision algorithm for another problem B. In polynomial time, we transform an instance α of A into an instance β of B, we solve B in polynomial time, and we use the answer for β as the answer for α.

What is NP-Complete

A decision problem L is NP-complete if:

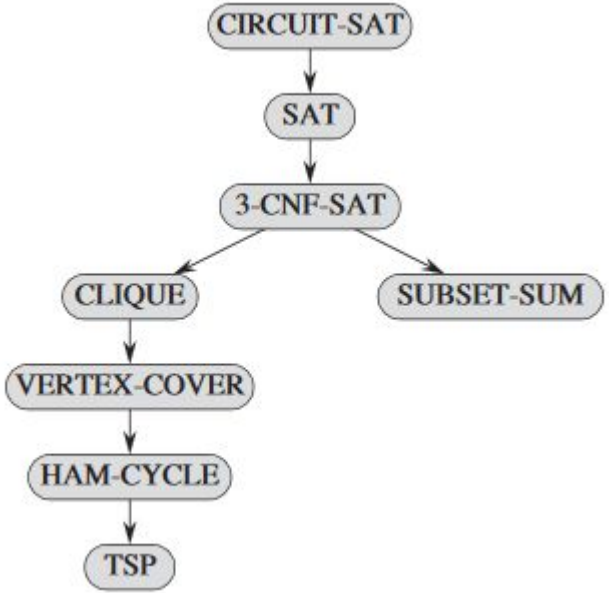
1. L is in NP

2. Every problem in NP is reducible to L in polynomial time

From this definition, it appears impossible to prove that a problem L is NP-complete

The idea is to take a first known NP-complete problem and polynomially reduce it to L

A first NP complete problem



Karp's 21 NP complete problems

- **Satisfiability**: the boolean satisfiability problem for formulas in conjunctive normal form (often referred to as SAT)
  - **0–1 integer programming** (A variation in which only the restrictions must be satisfied, with no optimization)
  - **Clique** (see also independent set problem)
    - **Set packing**
    - **Vertex cover**
      - **Set covering**
      - **Feedback node set**
      - **Feedback arc set**
      - **Directed Hamilton circuit** (Karp's name, now usually called **Directed Hamiltonian cycle**)
        - **Undirected Hamilton circuit** (Karp's name, now usually called **Undirected Hamiltonian cycle**)
  - **Satisfiability with at most 3 literals per clause** (equivalent to 3-SAT)
    - **Chromatic number** (also called the Graph Coloring Problem)
      - **Clique cover**
      - **Exact cover**
        - **Hitting set**
        - **Steiner tree**
        - **3-dimensional matching**
        - **Knapsack** (Karp's definition of Knapsack is closer to Subset sum)
          - **Job sequencing**
          - **Partition**
            - **Max cut**

NP-completeness proof (vertex cover problem)

- Maximum clique problem is used to show that minimum vertex cover problem in NP-complete (via method of reduction)
- A graph has a clique of size k if and only if its complement graph has a vertex cover of size |V|-k
- Proof: theorem 34.12

Approximation Algorithm

Since optimal solution requires exponential time, we try to find approximate solution

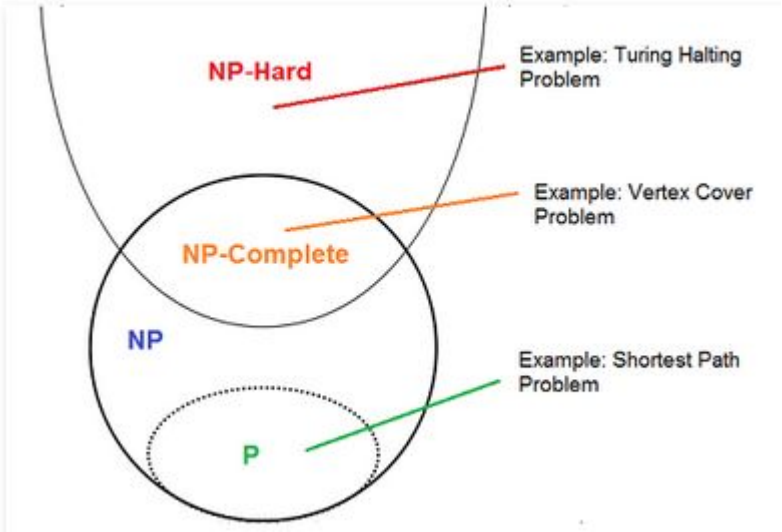
For vertex cover, the approximation algorithm is very simple

2-approximation algorithm (Figure 35.1)

Proof: theorem 35.1

What is NP-hard

A problem is NP-hard if it follows property 2 for NP-complete, but not necessarily follow property 1



Why should the engineers also care

Suppose you are asked to write an efficient algorithm to solve an extremely important problem for your company and after a lot of thinking, you can only come up with an exponential solution

What difference does it make if you know about NP-completeness and if you don’t?