Design and Analysis of Algorithm

Lecture-30:
P and NP problem

Contents

Optimization and Decision Problem

Optimization Problems

– An optimization problem is one which asks,

"What is the optimal solution to problem X?"

– ***Examples:***

- 0-1 Knapsack
- Fractional Knapsack
- Minimum Spanning Tree

Decision Problems

– An decision problem is one with yes/no answer

– **Examples:**

Does a graph G have a MST of weight $\leq$W?

Optimization/Decision Problems

- An optimization problem tries to find an optimal solution
- A decision problem tries to answer a yes/no question

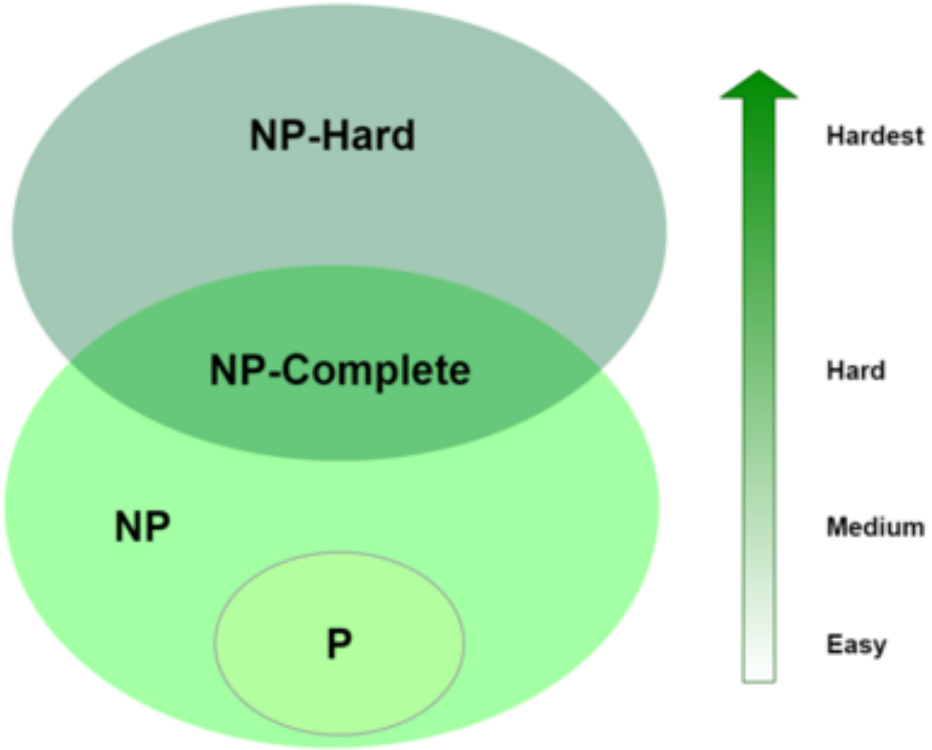
Many problems will have decision and optimization versions

Example

Traveling salesman problem

- **Optimization**: find Hamiltonian cycle of minimum weight
- **Decision**: is there a Hamiltonian cycle of weight $\leq k$

Class of Problems



- $Easy \rightarrow P$
- $Medium \rightarrow NP$
- $Hard \rightarrow NP\ Complete$
- $Hardest \rightarrow NP\ Hard$

Polynomial Algorithms

The first set of problems are polynomial algorithms that we can solve in polynomial time, such as $\log n, n^2, n^3, nlog\ n$

In general the time complexity of the algorithm belonging to this category is defined as:

$$T(n) = O(Cn^k): c > 0\ and\ n > k \geq 0$$

NP Algorithms

The second set of problems cannot be solved in polynomial time. However, they can be verified (or certified) in polynomial time. The complexity will be like $n^n, 2^n$

In general the time complexity of the algorithm belonging to this category is defined as:

$$T(n) = O(C_1 k^{C_2 n}) : c_1 > 0\,, C_2 > 0 and\ k > 0$$

NP-Complete Algorithms

All of these all belong to NP, but are among the hardest in the set. What makes them different from other problems is a useful distinction called completeness.

Completeness means that there exists a polynomial-time algorithm that can transform the problem into any other NP-complete problem. This transformation requirement is also called reduction.

There are numerous NP problems proven to be complete. Among them are:

- **Traveling Salesman**
- **Knapsack**, and
- **Graph Coloring**

NP-Hard Algorithms

They are not only hard to solve but are hard to verify as well. In fact, some of these problems aren't even decidable.

Some of the NP Hard Problems are:

K-means Clustering
Traveling Salesman Problem, and
Graph Coloring

Question

NP Complete

NP Hard

_____ problems are also quick to verify, slow to solve and can be reduced to any other problem called as _____

P

_____ problems are quick to solve

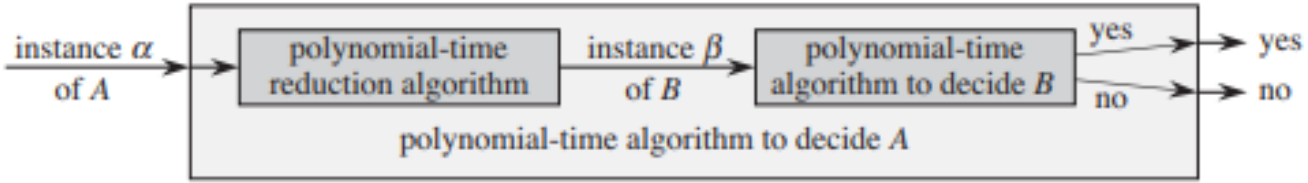
NP

_____ are quick to verify but slow to solve

Polynomial-time reduction algorithm

Let us consider a decision problem A, which we would like to solve in polynomial time.

Now suppose that we already know how to solve a different decision problem B in polynomial time.



1. Given an instance α of problem A, use a polynomial-time reduction algorithm to transform it to an instance β of problem B.
2. Run the polynomial-time decision algorithm for B on the instance β.
3. Use the answer for β as the answer for α .

NP Completeness Proof

NP-completeness is about showing how hard a problem is rather than how easy it is.

Suppose we have a decision problem A for which we already know that no polynomial-time algorithm can exist.

Suppose further that we have a polynomial-time reduction transforming instances of A to instances of B

We can use a simple proof by contradiction to show that no polynomial time algorithm can exist for B.

First NP-complete problem

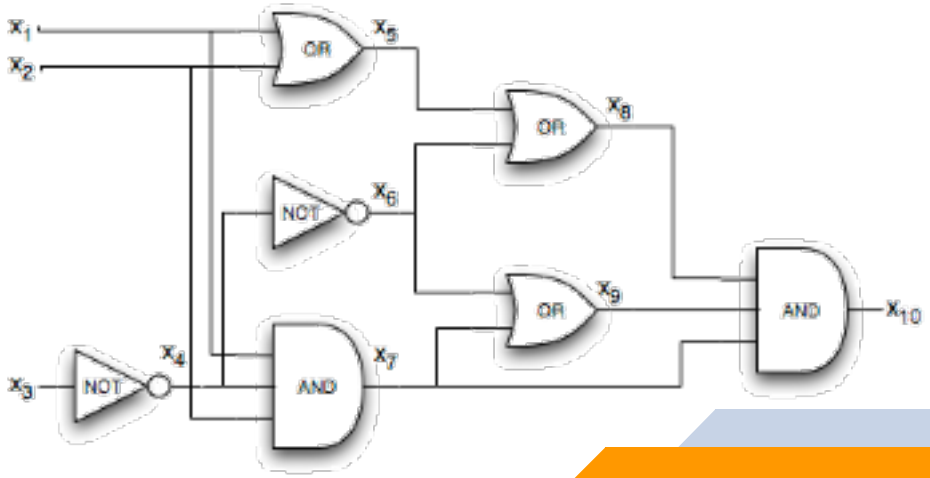
technique of reduction relies on having a problem already known to be NP-complete in order to prove a different problem NP-complete, we need a “first” NP-complete problem.

The problem we shall use is the circuit-satisfiability problem

Circuit Satisfiability problem

Given a Boolean combinational circuit composed of AND, OR, and NOT gates, is it satisfiable?

The circuit-satisfiability problem arises in the area of computer-aided hardware optimization. If a subcircuit always produces 0, that subcircuit is unnecessary; the designer can replace it by a simpler subcircuit that omits all logic gates and provides the constant 0 value as its output.



Question

Consider two decision problems Q1,Q2 such that Q1 reduces in polynomial time to 3-SAT and 3-SAT reduces in polynomial time to Q2. Then which one of the following is consistent with the above statement?

- Q_1 is NP and Q_2 is NP Hard
- Q_2 is NP and Q_1 is NP Hard
- Both Q_1 and Q_2 are NP
- Both Q_1 and Q_2 are NP Hard