Computational Complexity and Intractability Quick Overview

Based on Chapter 9 of Foundations of Algorithms
(CLRS and Neapolitan)

1

1

Objectives

2

- Classify problems as tractable or intractable
- Define decision problems
- Define the class P
- Define the class NP
- Define polynomial transformations
- Define the class of NP-Complete

2

Tractable

- A problem is tractable if there exists a polynomial-bound algorithm that solves it.
- Worst-case growth rate can be bounded by a polynomial
- Function of its input size
- $P(n) = a_n n^k + \dots + a_1 n + a_0$ where k is a constant
- $P(n)\ \varepsilon\ \theta(n^k)$
- n lg n not a polynomial
  - n lg n < n^2 bound by a polynomial

3

3

Intractable

- Means "difficult to treat or work"
- A problem is intractable if a computer has difficulty solving it
- A problem is intractable if it is not tractable
- Any algorithm with a growth rate not bounded by a polynomial
  - c^n, $c^{.01n}$, n^{logn}, n!, etc.
- It is the property of the problem not the algorithm

4

4

Three General Categories of Problems

1. Problems for which polynomial-time algorithms have been found
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

5

5

Not proven to be intractable no existing polynomial time algorithm

- Traveling salesperson
- 0-1 Knapsack
- Graph coloring
- Sum of subsets
- …

6

Classes of Problems

- Decision problems:
  - Problem where the output is a simple “yes” or “no”
  - Example: Searching a number within a list
- Classes of decision problems:
  - The class P
  - The class NP
  - The class of NP-Complete

7

Class P

- The set of all decision problems that can be solved by polynomial-time algorithms
  - Decision versions of searching and spanning tree belong to P
- Do problems such as traveling salesperson and 0-1 Knapsack (no polynomial-time algorithm has been found), etc., belong to P?
  - No one knows
  - To know a decision problem is not in P, it must be proven it is not possible to develop a polynomial-time algorithm to solve it

8

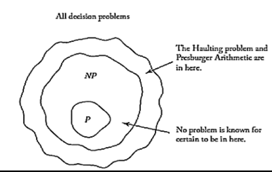
Class NP

- For a problem to be in NP, there must be an algorithm that does the verification of results in polynomial time
  - You cannot solve the problem in polynomial time
  - BUT, if you have a solution, you can verify its correctness
- For instance finding a clique within a graph is an NP problem
  - Given a solution, you can find its correctness (using adjacency matrix) in polynomial time.

9

Is P=NP?

- It has not been proven that there is a problem in NP that is not in P
- NP - P may be empty!
- P=NP? This is one of the most important questions in Computer Science
- To show P = NP, find polynomial-time algorithm for each problem in NP!



10

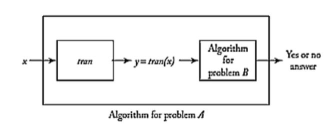
Polynomial-time Reducibility

- Suppose that you want to solve a decision problem A
- You already have an algorithm to solve decision problem B
- If you can write an algorithm that creates instance y of problem B from every instance x of problem A such that:
  - Algorithm for B answers yes for y if the answer to problem A is yes for x

11

Polynomial-time Reducibility (2)

- Transformation algorithm:
  - A function that maps every instance of problem A to an instance of problem B
  - y = trans(x)
  - The transformation algorithm usually are in P class
- Transformation algorithm + algorithm for problem B yields an algorithm for problem A



❖ Combination of a P problem and an NP is NP
  - ❖ That is, if B is NP and trans(x) is P then A will be NP
  - ❖ Also, if A is NP, and trans(x) is P it means that B must be NP

12

Reducibility Examples

1. Solving linear equations using an algorithm for quadratic equations
   - We already have an algorithm to solve $ax^2+bx+c=0$
   - Now we want to solve linear equations of: $bx+c=0$
   - It is reducible! Just you should consider $a=0$

2. Reducing "map coloring" problem to a "graph coloring" problem.

NP-Complete Problems

- **NP-Compete are problems that other NP problems can be reduced to them**

- How to prove that a problem is NP-complete?
  - Suppose that you want to know if problem B is NP-complete or not?
  - You know problem A as an NP-complete problem
  - If you can reduce problem A to B, you have proved that B is also NP-complete

Some Well-known Examples of NP-Complete Problems

- Knapsack problem
- Travelling Salesman problem
- Clique Problem
- Graph Coloring problem
- These famous NP-complete problems can be used to be reduced to the problems we encounter and want to prove that they are NP-complete as well