Input Size and Time Complexity

- Time complexity of algorithms:
  - Polynomial time (efficient) vs. Exponential time (inefficient)

$f(n)$	$n = 10$	30	50
n	0.00001 sec	0.00003 sec	0.00005 sec
n^5	0.1 sec	24.3 sec	5.2 mins
2^n	0.001 sec	17.9 mins	35.7 yrs

Intractability

- Dictionary Definition of intractable: “difficult to treat or work.”
- Computer Science: problem is intractable if a computer has difficulty solving it

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
- $P(n) = a_n n^k + \ldots + a_1 n + a_0$ where k is a constant
- $P(n)$ is $\theta(n^k)$
- $n\lg n$ not a polynomial
  - $n\lg n < n^2$ bound by a polynomial

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

Polynomial-time Algorithms

- $\Theta(n \lg n)$ for sorting
- $\Theta(\lg n)$ for searching
- $\Theta(n^3)$ for chained-matrix multiplication

Not proven to be intractable no existing polynomial time algorithm

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

Define

- Decision problems
- The class P
- Nondeterministic algorithms
- The class NP
- Polynomial transformations
- The class of NP-Complete

Decision problem

- Problem where the output is a simple “yes” or “no”
- Theory of NP-completeness is developed by restricting problems to decision problems
- Optimization problems can be transformed into decision problems
- Optimization problems are at least as hard as the associated decision problem
- If polynomial-time algorithm for the optimization problem is found, we would have a polynomial-time algorithm for the corresponding decision problem

Decision Problems

- Traveling Salesperson
  - For a given positive number d, is there a tour having length <= d?
- 0-1 Knapsack
  - For a given profit P, is it possible to load the knapsack such that total weight <= W?

Class P

- The set of all decision problems that can be solved by polynomial-time algorithms
- Decision versions of searching, shortest path, spanning tree, etc. 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

Nondeterministic Algorithms – consist of 2 phases

1. Nondeterministic phase – Guessing Phase: given an instance of a problem, a solution is guessed (represented by some string s); We call it nondeterminisitic because unique step-by-step instructions are not specified
2. Deterministic phase – Verification Phase

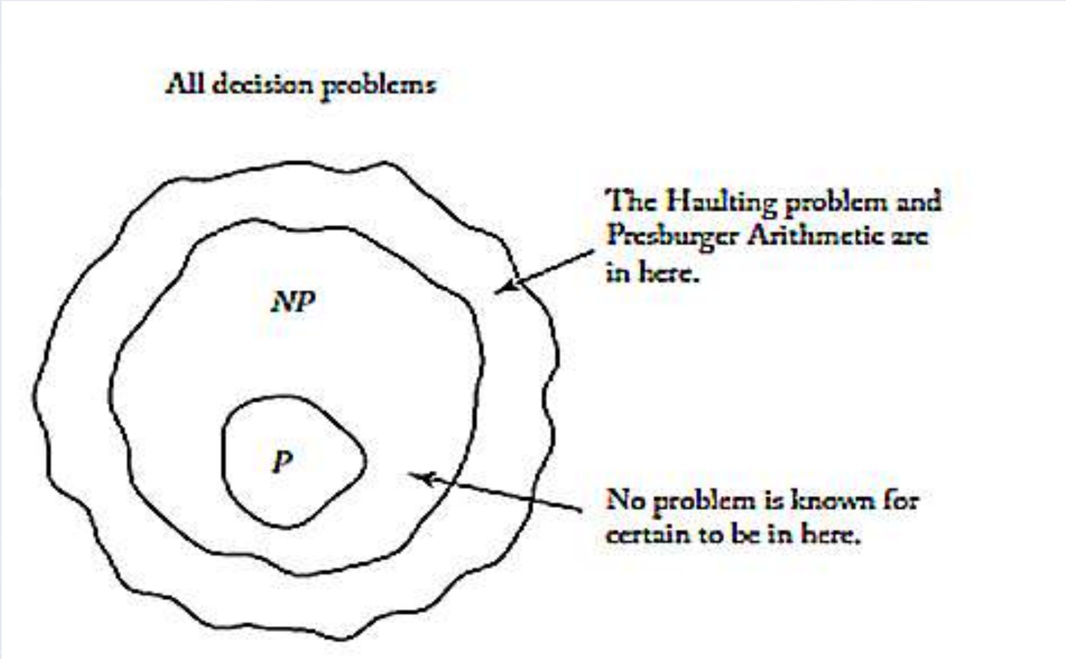
Polynomial-time Nondeterministic Algorithm (NDA)

- A nondeterministic algorithm whose verification stage is a polynomial-time algorithm

Class NP

- The set of all decision problems that can be solved by polynomial-time nondeterministic algorithms
- Nondeterministic polynomial
- For a problem to be in NP, there must be an algorithm that does the verification in polynomial time
- Traveling salesperson decision problem belongs to NP
  - Show a guess, s, length polynomial bounded
  - Yes answer verified in a polynomial number of steps

Figure 9.3


All decision problems
NP
P
The Haulting problem and Presburger Arithmetic are in here.
No problem is known for certain to be in here.

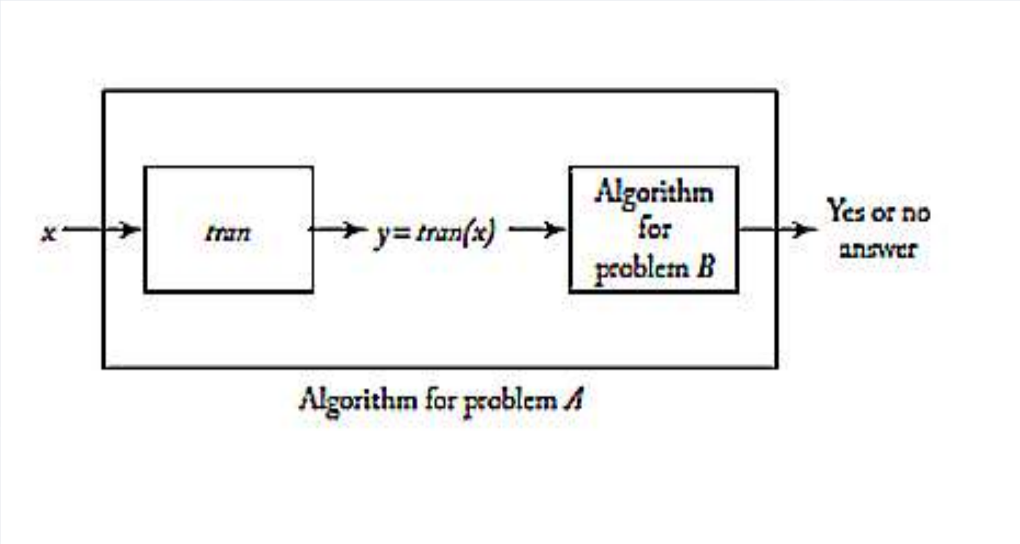
Polynomial-time Reducibility

- Want to solve decision problem A
- Have an algorithm to solve decision problem B
- 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

Polynomial-time Reducibility

- Transformation algorithm
  - Function that maps every instance of problem A to an instance of problem B
  - y = trans(x)
- Transformation algorithm + algorithm for problem B yields an algorithm for problem A

Figure 9.4


x
tran
y = tran(x)
Algorithm for problem B
Yes or no answer
Algorithm for problem A

Polynomial-time many-one reducible

- If there exists a polynomial-time transformation algorithm from decision problem A to decision problem B, problem A is polynomial-time many-one reducible to problem B
- $A \propto B$
- Many-one: transformation algorithm is a function that may map many instances of problem A to one instance of problem B
- If the transformation algorithm is polynomial-time and the algorithm for problem B is polynomial, The algorithm for A must be polynomial

Theorem 9.1

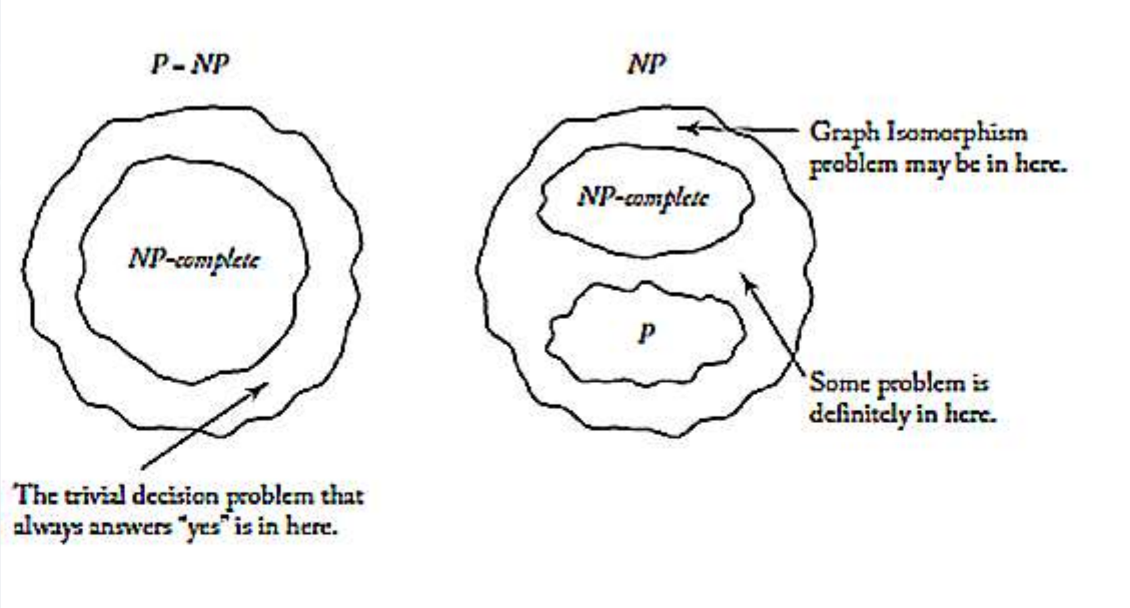
- If decision problem B is in P and A $\propto$ B, then decision problem A is in P

NP-Complete

- A problem B is called NP-complete if both the following are true:

1. B is in NP
2. For every other problem A in NP, $A \propto B$

Figure 9.7


P = NP
NP-complete
The trivial decision problem that always answers "yes" is in here.
NP
Graph Isomorphism problem may be in here.
NP-complete
P
Some problem is definitely in here.