Computational Complexity and Intractability

By: Aminul Islam

Based on Chapter 9 of Foundations of Algorithms

Objectives

Objectives

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

Tractable

Tractable

- A problem is tractable if there exists a polynomial-bound algorithm that solves it.

Tractable

- A problem is tractable if there exists a polynomial-bound algorithm that solves it.
- $P(n) = a_n n^k + \ldots a_1 n + a_0$ where k is a constant

Tractable

- A problem is tractable if there exists a polynomial-bound algorithm that solves it.
- $P(n) = a_n n^k + \ldots a_1 n + a_0$ where k is a constant
- $P(n) \in \Theta(n^k)$

Tractable

- A problem is tractable if there exists a polynomial-bound algorithm that solves it.
- $P(n) = a_n n^k + \ldots a_1 n + a_0$ where k is a constant
- $P(n) \in \Theta(n^k)$
- $n \lg n$ is not a ploynomial

Tractable

- A problem is tractable if there exists a polynomial-bound algorithm that solves it.
- $P(n) = a_n n^k + \ldots a_1 n + a_0$ where k is a constant
- $P(n) \in \Theta(n^k)$
- $n \lg n$ is not a ploynomial
  - $n \lg n < n^2$

Tractable

- A problem is tractable if there exists a polynomial-bound algorithm that solves it.
- $P(n) = a_n n^k + \ldots a_1 n + a_0$ where k is a constant
- $P(n) \in \Theta(n^k)$
- $n \lg n$ is not a ploynomial
  - $n \lg n < n^2$
    bound by a polynomial

Intractable

Intractable

- Means "difficult to treat or work"

Intractable

- Means “difficult to treat or work”
- A problem is intractable if a computer has difficulty solving it

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

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

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, $c^{0.01n}$, $n^{\lg n}$, $n!$

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, $c^{0.01n}$, $n^{\lg n}$, $n!$
- It is the property of the problem not the algorithm

Three General Categories of Problems

Three General Categories of Problems

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

Three General Categories of Problems

1. Problems for which polynomial-time algorithms have been found
2. Problems that have been proven to be intractable

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

Not proven to be intractable no existing polynomial time algorithm

Not proven to be intractable no existing polynomial time algorithm

- Traveling salesperson

Not proven to be intractable no existing polynomial time algorithm

- Traveling salesperson
- 0-1 Knapsack

Not proven to be intractable no existing polynomial time algorithm

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

Not proven to be intractable no existing polynomial time algorithm

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

Not proven to be intractable no existing polynomial time algorithm

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

Define

Define

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

Decision problem

Decision problem

- Problem where the output is a simple "yes" or "no"

Decision problem

- Problem where the output is a simple “yes” or “no”
- Theory of NP-completeness is developed by restricting problems to decision problems

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

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

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

Decision Problems

- Traveling Salesperson

Decision Problems

- Traveling Salesperson
  - For a given positive number d, is there a tour having length $<= d$?

Decision Problems

- Traveling Salesperson
  - For a given positive number d, is there a tour having length $<= d$?
- 0-1 Knapsack

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

Class P

- The set of all decision problems that can be solved by polynomial-time algorithms

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

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?

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

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 <u>must be proven</u> it is not possible to develop a polynomial-time algorithm to solve it

Class NP

Class NP

- For a problem to be in NP, there must be an algorithm that does the verification of results in polynomial time

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

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

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 Traveling salesperson decision problem belongs to NP

Is P = NP?

Is P = NP?

- It has not been proven that there is a problem in NP that is not in P

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

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? One of the most important questions in CS

Is P = NP?

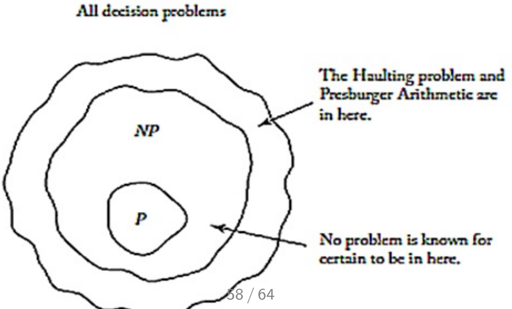
- It has not been proven that there is a problem in NP that is not in P
- NP-P may be empty
- P=NP? One of the most important questions in CS
- To show P!=NP, find a problem in NP that is not in P

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? One of the most important questions in CS
- To show P!=NP, find a problem in NP that is not in P
- To show P = NP, find polynomial-time algorithm for each problem in NP

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? One of the most important questions in CS
- To show P!=NP, find a problem in NP that is not in P
- To show P = NP, find polynomial-time algorithm for each problem in NP



NP-Complete Problems

NP-Complete Problems

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

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?

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?

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

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