CS141: Intermediate Data Structures and Algorithms

NP-Completeness

Amr Magdy
Why Studying NP-Completeness?

Two reasons:

1. In almost all cases, if we can show a problem to be **NP-complete** or **NP-hard**, the best we can achieve (NOW) is mostly exponential algorithms.
   - This means we cannot solve large problem sizes efficiently

2. If we can solve only one NP-complete problem efficiently, we can solve ALL NP problems efficiently (major breakthrough)

More details come on what does these mean
Topic Outline

1. Background
   › Decision vs. Optimization Problems
   › Models of Computation
   › Input Encoding

2. Complexity Classes
   › P
   › NP
     › Polynomial Verification
     › Examples

3. NP-hardness
   › Polynomial Reductions

4. NP-Complete Problems
   › Definition and Examples
   › Weak vs. Strong NP-Complete Problems
Decision vs Optimization Problems

- Decision problem: a problem expressed as a yes/no question
Decision vs Optimization Problems

- Decision problem: a problem expressed as a yes/no question

  - Examples:
    - Is graph G connected?
    - Is path $P:u \rightarrow v$ shortest?
Decision vs Optimization Problems

- Decision problem: a problem expressed as a yes/no question
  - Examples:
    - Is graph G connected?
    - Is path P:u→v shortest?

- Optimization problem: finding the best solution from all feasible solutions
  - In continuous optimization, the answer is real valued objective function (either min or max)
Decision vs Optimization Problems

- Decision problem: a problem expressed as a yes/no question
  - Examples:
    - Is graph G connected?
    - Is path P:u→v shortest?

- Optimization problem: finding the best solution from all feasible solutions
  - In continuous optimization, the answer is real valued objective function (either min or max)
  - Examples:
    - Find a maximum fully-connected subgraph (clique) size in a graph.
    - Find the least cost of multiplying a chain of matrices.
Decision vs Optimization Problems

- Decision problem: a problem expressed as a yes/no question
  - Examples:
    - Is graph G connected?
    - Is path P:u→v shortest?

- Optimization problem: finding the best solution from all feasible solutions
  - In continuous optimization, the answer is real valued objective function (either min or max)
  - Examples:
    - Find a maximum fully-connected subgraph (clique) size in a graph.
    - Find the least cost of multiplying a chain of matrices.

- Converting optimization problem → decision problem?
  - Put a bound on the objective function.
Decision vs Optimization Problems

- Decision problem: a problem expressed as a yes/no question
  - Examples:
    - Is graph G connected?
    - Is path P:u→v shortest?

- Optimization problem: finding the best solution from all feasible solutions
  - In continuous optimization, the answer is real valued objective function (either min or max)
  - Examples:
    - Find a maximum fully-connected subgraph (clique) size in a graph.
    - Find the least cost of multiplying a chain of matrices.

- Converting optimization problem → decision problem?
  - Put a bound on the objective function.
  - Does G have a clique of size k? for k= 3, 4, 5,…(finding max clique)
Take Home Messages

(1) Computation theory focuses on decision problems
Models of Computation

- MoC: informally a theoretic description of a way to compute
Models of Computation

› MoC: informally a theoretic description of a way to compute
› Example: mask model

Mask Model (on paper)  Mask Realization (fabric instance)

www.firstpalette.com
Models of Computation

› At a low level:
  › Finite State Automata (FSA)
  › Pushdown Automata (PDA)
  › Turing Machine (TM)
  › ..... 

Focus of other courses (e.g., Theory of Computation, Compilers Design, etc.)

› At a high level:
  › RAM (Random Access Machine)
  › Pointer Machine
  › .....
Models of Computation

- A model of computation determines two things:
  - What are the possible operations
  - What is the cost of each operation
Models of Computation

- A model of computation determines two things:
  - What are the possible operations
  - What is the cost of each operation

- w-Random Access Machine (w-RAM) MoC:
  - The one we used throughout the course
  - Possible operations in $\Theta(1)$:
    - Access any memory word at random
    - Read variable
    - Write variable
    - Basic mathematical operations (add, multiply, assign,…etc)
    - Single-command output operations (print, return, …etc)
    - ….
A model of computation determines two things:
- What are the possible operations
- What is the cost of each operation

**w-Random Access Machine (w-RAM) MoC:**
- The one we used throughout the course
- Possible operations in $\Theta(1)$:
  - Access any memory word at random
  - Read variable
  - Write variable
  - Basic mathematical operations (add, multiply, assign, … etc)
  - Single-command output operations (print, return, … etc)
- ….

What the cost of appending to a list in w-RAM model? Sorting? Finding maximum?
Models of Computation

- A model of computation determines two things:
  - What are the possible operations
  - What is the cost of each operation

- Pointer Machine (PM) MoC:
  - A machine with only dynamic allocated memory through pointers
  - Possible operations in $\Theta(1)$:
    - Follow pointer (no random memory anymore)
    - Read pointed variable
    - Write pointed location
    - ....
Models of Computation

- A model of computation determines two things:
  - What are the possible operations
  - What is the cost of each operation

- Pointer Machine (PM) MoC:
  - A machine with only dynamic allocated memory through pointers
  - Possible operations in $\Theta(1)$:
    - Follow pointer (no random memory anymore)
    - Read pointed variable
    - Write pointed location
    - ....

- What the cost of accessing any memory location in PM model? Sorting? Finding maximum?
  - Function of the basic operations
Take Home Messages

(1) Computation theory focuses on decision problems

(2) Algorithm complexity is affected by the computation model
Input / Output Encoding

› Assume multiplying two decimal integers

› 2 * 2 = 4

(basic operation, single digit op)

› 12 * 12 = (1 * 10 + 2) * (1 * 10 + 2)

= 1 * 10 * 1 * 10 + 1 * 10 * 2 + 2 * 1 * 10 + 2 * 2

(4 mult ops, 4 add ops, 4 shift ops)

› $O(n^2)$ operations for n-digit number
Input / Output Encoding

› Assume multiplying two decimal integers
  › 2 * 2 = 4
      (basic operation, single digit op)
  › 12 * 12 = (1*10+2)*(1*10+2)
     = 1*10*1*10+1*10*2+2*1*10+2*2
     (4 mult ops, 4 add ops, 4 shift ops)
  › O(n^2) operations for n-digit number

› Assume multiplying two binary integers
  › (10)_b * (10)_b = (1*2+0)*(1*2+0)
     = 1*2*1*2+1*2*0+0*1*2+0*0
     (4 mult ops, 4 add ops, 4 shift ops)
  › O(n^2) operations for n-digit number
Input / Output Encoding

- Assume multiplying two decimal integers
  - $2 \times 2 = 4$
    - (basic operation, single digit op)
  - $12 \times 12 = (1 \times 10 + 2)(1 \times 10 + 2)$
    - $= 1 \times 10 \times 1 \times 10 + 1 \times 10 \times 2 + 2 \times 1 \times 10 + 2 \times 2$
    - (4 mult ops, 4 add ops, 4 shift ops)
  - $O(n^2)$ operations for n-digit number

- Assume multiplying two binary integers
  - $(10)_b \times (10)_b = (1 \times 2 + 0)(1 \times 2 + 0)$
    - $= 1 \times 2 \times 1 \times 2 + 1 \times 2 \times 0 + 0 \times 1 \times 2 + 0 \times 0$
    - (4 mult ops, 4 add ops, 4 shift ops)
  - $O(n^2)$ operations for n-digit number

Same input (2x2), different encoding
Input / Output Encoding

- Assume multiplying two decimal integers
  - $2 \times 2 = 4$
    (basic operation, single digit op)
  - $12 \times 12 = (1\times 10 + 2) \times (1\times 10 + 2)$
    $= 1\times 10 \times 1\times 10 + 1\times 10 \times 2 + 2\times 1\times 10 + 2 \times 2$
    (4 mult ops, 4 add ops, 4 shift ops)
  - $O(n^2)$ operations for $n$-digit number

- Assume multiplying two binary integers
  - $(10)_b \times (10)_b = (1\times 2 + 0) \times (1\times 2 + 0)$
    $= 1\times 2 \times 1\times 2 + 1\times 2 \times 0 + 0 \times 1\times 2 + 0 \times 0$
    (4 mult ops, 4 add ops, 4 shift ops)
  - $O(n^2)$ operations for $n$-digit number

- Input representation (encoding) affects the amount of computations for same input
Exercise

- design a divide & conquer algorithm to multiply two n-bits integers in $O(n^2)$

Note:
- Multiplying by $2^n$ for binary numbers is shifting by $n$ bits $\Rightarrow \Theta(n)$
- Multiplying by $10^n$ for decimal numbers is shifting by $n$ digits $\Rightarrow \Theta(n)$
Take Home Messages

(1) Computation theory focuses on decision problems

(2) Algorithm complexity is affected by the computation model

(3) Algorithm complexity is affected by the input encoding/length
Take Home Messages

(1) Computation theory focuses on decision problems

(2) Algorithm complexity is affected by:
(a) the computation model
(b) the input encoding/length
Encoding Examples in Binary Strings

- Binary strings are the standard encoding for computing now
Encoding Examples in Binary Strings

- Binary strings are the standard encoding for computing now
- Integer $n \rightarrow$ binary number (in $\log_2(n)$ bits)
  - Example: $999 \rightarrow 01111100111$
Encoding Examples in Binary Strings

- Binary strings are the standard encoding for computing now.
- Integer $n \rightarrow$ binary number (in $\log_2(n)$ bits)
  - Example: $999 \rightarrow 01111100111$
- Array of $n$ integers $\rightarrow$ sequence of integers (in $n \cdot \log_2(n)$ bits)
  - Example: $9,15,3 \rightarrow 1001,1111,0011$
Encoding Examples in Binary Strings

- Binary strings are the standard encoding for computing now.
- Integer n \(\rightarrow\) binary number (in \(\log_2(n)\) bits)
  - Example: 999 \(\rightarrow\) 01111100111
- Array of n integers \(\rightarrow\) sequence of integers (in \(n\times\log_2(n)\) bits)
  - Example: 9,15,3 \(\rightarrow\) 1001,1111,0011
- String of n chars \(\rightarrow\) sequence of integer codes (in \(n\times\log_2(n)\) bits), e.g., ASCII codes
  - Example: Amr \(\rightarrow\) 1000001,1101101,1110010
Encoding Examples in Binary Strings

- Binary strings are the standard encoding for computing now.
- Integer $n \rightarrow$ binary number (in $\log_2(n)$ bits)
  - Example: $999 \rightarrow 01111100111$
- Array of $n$ integers $\rightarrow$ sequence of integers (in $n \cdot \log_2(n)$ bits)
  - Example: $9,15,3 \rightarrow 1001,1111,0011$
- String of $n$ chars $\rightarrow$ sequence of integer codes (in $n \cdot \log_2(n)$ bits), e.g., ASCII codes
  - Example: Amr $\rightarrow$ $1000001,1101101,1110010$
- Graph $G$ of $n$ vertices and $m$ edges:
  - Each vertex with integer id $\rightarrow$ $n$ integers
  - Each edge with integer id and weight $\rightarrow$ $m$ integers + $m$ floats
  - $m$ is maximum of $n^2/2$, i.e., $m=O(n^2)$
  - Example: $0110101000011101101111001000011101011110010000111001101010000111011011110010000111010111111001001…$
Encoding Examples in Binary Strings

- Binary strings are the standard encoding for computing now
- Integer
  - Example: 999 → 01111100111
- Array of n integers
  - Example: 9,15,3 → 1001,1111,0011
- String of n chars
  - Example: Amr → 1000001,1101101,1110010
- Graph G of n vertices and m edges:
  - Example: 0110101000001110110111100100001110101111001001…
Take Home Messages

(1) Computation theory focuses on decision problems

(2) Algorithm complexity is affected by:
   (a) the computation model
   (b) the input encoding/length

(3) Binary input string (concrete input) is different in length than the algorithm abstract input
Complexity Class

- Complexity class:
  - A set of problems that share some complexity characteristics
    - Either in time complexity
    - Or in space complexity
Complexity Class

- Complexity class:
  A set of problems that share some complexity characteristics
  - Either in time complexity
  - Or in space complexity

- In this course, our discussion is limited to only two time complexity classes: P and NP
  - Other courses cover more content (e.g., Theory of Computation course)
P

- P is a complexity class of problems that are *decidable* in *polynomial-time* of input string length, i.e., $O(m^k)$
  - where $m$ the input string length and $k$ is constant

- For simplicity, P is the set of problems that are *solvable* in polynomial time
  - i.e., has $O(m^k)$ algorithm to find a solution
P

- P is a complexity class of problems that are *decidable* in *polynomial-time* of input string length, i.e., $O(m^k)$
  - where $m$ the input string length and $k$ is constant

- For simplicity, P is the set of problems that are *solvable* in polynomial time
  - i.e., has $O(m^k)$ algorithm to find a solution

- Examples:
  - Shortest paths in graph
  - Matrix chain multiplication
  - Activity scheduling problem
  - ….
NP is a complexity class of problems that are **verifiable** in **polynomial-time** of input string length.

For simplicity, given a solution of an NP problem, we can verify in polynomial time $O(m^k)$ if this solution is correct.
Is $P \subset NP$?
Is $P \subseteq NP$?

- Yes
- What does this mean?
Is $P \subseteq NP$?

- Yes
- What does this mean?
  - Every problem that is solvable in polynomial time is verifiable in polynomial time as well
Is $P \subseteq NP$? or Is $P = NP$?

What does this mean?
Is $P \subseteq NP$? or Is $P = NP$?

- What does this mean?
  - There are polynomial time algorithms to solve NP problems
Is P \subseteq NP? or Is P = NP?

- What does this mean?
  - There are polynomial time algorithms to solve NP problems

- Nobody yet knows
  - The question posed in 1971
Is $P \subseteq NP$? or Is $P = NP$?

- What does this mean?
  - There are polynomial time algorithms to solve NP problems

- Nobody yet knows
  - The question posed in 1971
  - You think it is old?
    - Check Alhazen’s problem then
Is $P \subseteq NP$? or Is $P = NP$?

- What does this mean?
  - There are polynomial time algorithms to solve NP problems

- Nobody yet knows
  - The question posed in 1971
  - You think it is old?
    - Check Alhazen’s problem then

- Computer Science theoreticians “thinks” $P \neq NP$, but no proof
Is $P \subseteq NP$? or Is $P = NP$?

» What does this mean?
» There are polynomial time algorithms to solve NP problems

» Nobody yet knows
» The question posed in 1971
» You think it is old?
  » Check Alhazen’s problem then

» Computer Science theoreticians “thinks” $P \neq NP$, but no proof
NP Problems

Example: Travelling Salesman Problem

Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city and returns to the origin city?
NP Problems

- Example: **Travelling Salesman Problem**
  Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city and returns to the origin city?

- How to solve this problem?
NP Problems

Example: **Travelling Salesman Problem**
Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city and returns to the origin city?

How to solve this problem?
- Brute force: $O(n!)$
NP Problems

Example: **Travelling Salesman Problem**
Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city and returns to the origin city?

How to solve this problem?
- Brute force: $O(n!)$
- Dynamic programming: $O(n2^n)$
Travelling Salesman Movie

https://www.youtube.com/watch?v=6ybd5rbQ5rU
NP Problems

Example: SAT Problem
Given a Boolean circuit $S$, is there a satisfying assignment for $S$? (i.e., variable assignment that outputs 1)
NP Problems

Example: SAT Problem

Given a Boolean circuit $S$, is there a satisfying assignment for $S$? (i.e., variable assignment that outputs 1)
NP Problems

Example: **3-CNF Problem**
Given a Boolean circuit S in 3-CNF form, is there a satisfying assignment for S? (i.e., variable assignment that outputs 1)

**3-CNF formula**: a set ANDed Boolean clauses, each with 3 ORed literals (Boolean variables)

Example: \( \lor = \text{OR}, \land = \text{AND}, \neg = \text{NOT} \)
\((x_1 \lor \neg x_2 \lor \neg x_3) \land (\neg x_1 \lor x_2 \lor x_3) \land (x_1 \lor x_2 \lor x_3)\)
NP Problems

- Example: **3-CNF Problem**
  Given a Boolean circuit S in 3-CNF form, is there a satisfying assignment for S? (i.e., variable assignment that outputs 1)

- 3-CNF formula: a set ANDed Boolean clauses, each with 3 ORed literals (Boolean variables)

- Example: \( \lor = \text{OR}, \land = \text{AND}, \neg = \text{NOT} \)
  \((x_1 \lor \neg x_2 \lor \neg x_3) \land (\neg x_1 \lor x_2 \lor x_3) \land (x_1 \lor x_2 \lor x_3)\)

- Solution: \(O(k2^n)\) for k clauses and n variables
NP Problems

Example: (Max) Clique Problem
Given a graph $G=(V,E)$, find the clique of maximum size. Clique: fully connected subgraph.
NP Problems

Example: (Max) Clique Problem

Given a graph $G=(V,E)$ of $n$ vertices, find the clique of maximum size.

Clique: fully connected subgraph.

Solution:

- Assume max clique size $k$ and $|V| = n$
- Brute force: $O(n 2^n)$
- Combinations of $k$: $O(n^k k^2)$
  - Try for $k=3, 4, 5, …$
  - $k$ is not constant, so this is not polynomial
NP Problems: Polynomial Verification

- Given a solution, can I verify if it is correct in polynomial time?
- TSP Problem: Yes (the decision version)
  - Is there a tour with weight $W$?
- SAT Problem: Yes
- 3-CNF Problem: Yes
- Max Clique Problem: Yes (the decision version)
  - Is there a clique of size $k$?
NP-hard Problems

Informally:
an NP-hard problem B is a problem that is at least as hard as the hardest problems in NP class

Formally:
B is NP-hard if \( \forall A \in \text{NP}, A \leq_p B \)
(i.e., A is polynomial reducible to B)
Polynomial Reductions

- Polynomial reduction $A \leq_P B$ is converting an instance of $A$ into an instance of $B$ in polynomial time.
Polynomial Reductions

- Polynomial reduction \( A \leq_P B \) is converting an instance of \( A \) into an instance of \( B \) in polynomial time.

- How to solve \( A \) given a solver to \( B \)?
Polynomial Reductions

- Polynomial reduction $A \leq_P B$ is converting an instance of $A$ into an instance of $B$ in polynomial time.

- How to solve $A$ given a solver to $B$?

```
instance $\alpha$ of $A$ → polynomial-time reduction algorithm → instance $\beta$ of $B$ → polynomial-time algorithm to decide $B$ → yes

 polynomial-time algorithm to decide $A$ → yes → no
```
Polynomial Reductions: Example

- Reduce 3-CNF to k-size Clique
- Example: three 3-CNF clauses
  \[(x_1 \lor \neg x_2 \lor \neg x_3) \land (\neg x_1 \lor x_2 \lor x_3) \land (x_1 \lor x_2 \lor x_3)\]
Polynomial Reductions: Example

- Reduce 3-CNF to k-size Clique
- Example: three 3-CNF clauses
  \[(x_1 \lor \neg x_2 \lor \neg x_3) \land (\neg x_1 \lor x_2 \lor x_3) \land (x_1 \lor x_2 \lor x_3)\]

\[C_1 = x_1 \lor \neg x_2 \lor \neg x_3\]
\[C_2 = \neg x_1 \lor x_2 \lor x_3\]
\[C_3 = x_1 \lor x_2 \lor x_3\]
Polynomial Reductions: Example

- Reduce 3-CNF to k-size Clique
- Example: three 3-CNF clauses
  \((x_1 \lor \neg x_2 \lor \neg x_3) \land (\neg x_1 \lor x_2 \lor x_3) \land (x_1 \lor x_2 \lor x_3)\)

\[C_1 = x_1 \lor \neg x_2 \lor \neg x_3\]

\[C_2 = \neg x_1 \lor x_2 \lor x_3\]

\[C_3 = x_1 \lor x_2 \lor x_3\]
Polynomial Reductions: Example

- Reduce 3-CNF to k-size Clique
- Example: three 3-CNF clauses
  \((x_1 \lor \neg x_2 \lor \neg x_3) \land (\neg x_1 \lor x_2 \lor x_3) \land (x_1 \lor x_2 \lor x_3)\)

- Given: S: k-clause 3-CNF formula
- Reduction Algorithm:
  - Compose a graph G of k sets of vertices, each set has three vertices
  - Connect all pairs of vertices \((u,v)\) such that:
    - \(u\) and \(v\) belong to two different sets
    - If \(u=x_i\), then \(v \neq \neg x_i\)
  - If there is k-size clique in G, there is a satisfying assignment to S (assign 1 to each vertex in the clique).
NP-hard Proofs

To prove B an NP-hard problem:
  Show a polynomial time reduction algorithm from B to ONE of the existing NP-hard problems.
NP-Complete Problems

B is NP-complete problem if:
1. $B \in \text{NP}$
2. B is NP-hard
NP-Complete Problems

NP-Hard

NP-Complete

NP

P

P ≠ NP

Complexity

NP-Hard

P = NP = NP-Complete

P = NP

70
NP-Complete Problems: Examples

CIRCUIT-SAT

SAT

3-CNF-SAT

CLIQUE

VERTEX-COVER

HAM-CYCLE

TSP

SUBSET-SUM
NP-Complete Problems: Examples

- Hamiltonian Cycle Problem: Given an undirected or directed graph $G$, is there a cycle in $G$ that visits each vertex exactly once?
Take Home Messages: Remember?

(1) Computation theory focuses on decision problems.

(2) Algorithm complexity is affected by:
   (a) the computation model
   (b) the input encoding/length

(3) Binary input string (concrete input) is different in length than the algorithm (abstract input).
Strong vs Weak NP-Completeness

- Abstract input vs Concrete input:
  - Input array of $n$ integers:
    - Abstract input size: $a = n$ (# of integers)
    - Concrete input size in binary: $b = n \log n$ (# of bits of the array)

- Weak NP-complete problem:
  - An NP-complete problem that has a known polynomial solution in terms of the abstract input size.

- Strong NP-complete problem:
  - An NP-complete problem that does not have a known polynomial solution in terms of either abstract or concrete input size.
Weak NP-Completeness: Examples

- **Subset-Sum Problem:**
  - Given set $S$ of $n$ integers and integer $T$
  - Dynamic Programming solution: $O(nT)$
  - Abstract input: $a_1 = n$ (integers of $S$), $a_2 = 1$ (integer $T$)
  - Concrete input: $b_1 = n \log n$, $b_2 = \log T$
  - $O(nT) = O(b_1 \cdot 2^{b_2})$ \(\Rightarrow\) exponential in concrete input but polynomial in abstract input \(\Rightarrow\) weak NP-complete

- **Partition Problem:**
  - Given set $S$ of $n$ integers, divide $S$ into two disjoint subsets of equal sum
  - Same solution (and complexity) as Subset-Sum

- **0-1 Knapsack Problem**
  - Similar solution to subset-sum ($O(nW)$ for knapsack of weight $W$)
Weak NP-Completeness

- For weak NP-complete problems, we are able to solve many instances in practical input sizes.
Book Readings

› Ch. 34