CS 577- Intro to Algorithms

Computational Intractability

Dieter van Melkebeek

November 17, 2020

Outline

Outline

Motivation

Outline

Motivation

- Recognizing infeasible approaches

Outline

Motivation

- Recognizing infeasible approaches
- P vs NP problem

Outline

Motivation

- Recognizing infeasible approaches
- P vs NP problem

Topics

Outline

Motivation

- Recognizing infeasible approaches
- P vs NP problem

Topics

- Classes P and NP

Outline

Motivation

- Recognizing infeasible approaches
- P vs NP problem

Topics

- Classes P and NP
- NP-hardness and NP-completeness

Intractable Problems

Intractable Problems

Independent Set

Input: graph G

Output: independent set S of G such that $|S|$ is maximized

Intractable Problems

Independent Set

Input: graph G

Output: independent set S of G such that $|S|$ is maximized

Satisfiability

Input: Boolean formula φ

Output: satisfying assignment of φ, or report that none exists

More Intractable Problems

Graph coloring

More Intractable Problems

Graph coloring

Input: graph $G = (V, E)$

More Intractable Problems

Graph coloring

Input: graph $G = (V, E)$

Output: $c : V \rightarrow [k]$ such that $(\forall\, (u, v) \in E)\, c(u) \neq c(v))$
and k is minimized

More Intractable Problems

Graph coloring

Input: graph $G = (V, E)$

Output: $c : V \to [k]$ such that $(\forall\, (u, v) \in E)\, c(u) \neq c(v))$
and k is minimized

Traveling salesperson problem (TSP)

More Intractable Problems

Graph coloring

Input: graph $G = (V, E)$

Output: $c : V \to [k]$ such that $(\forall\, (u, v) \in E)\, c(u) \neq c(v))$ and k is minimized

Traveling salesperson problem (TSP)

Input: n cities and direct intercity travel costs for each pair

More Intractable Problems

Graph coloring

Input: graph $G = (V, E)$

Output: $c : V \to [k]$ such that $(\forall (u, v) \in E)\, c(u) \neq c(v))$ and k is minimized

Traveling salesperson problem (TSP)

Input: n cities and direct intercity travel costs for each pair

Output: tour that visits every city once and has minimum total cost

More Intractable Problems

Graph coloring

Input: graph $G = (V, E)$

Output: $c : V \to [k]$ such that $(\forall (u, v) \in E)\, c(u) \neq c(v))$ and k is minimized

Traveling salesperson problem (TSP)

Input: n cities and direct intercity travel costs for each pair

Output: tour that visits every city once and has minimum total cost

Subset sum

More Intractable Problems

Graph coloring

Input: graph $G = (V, E)$

Output: $c : V \to [k]$ such that $(\forall\,(u, v) \in E)\, c(u) \neq c(v))$ and k is minimized

Traveling salesperson problem (TSP)

Input: n cities and direct intercity travel costs for each pair

Output: tour that visits every city once and has minimum total cost

Subset sum

Input: $a_1, a_2, \ldots, a_n \in \mathbb{N}$; $t \in \mathbb{N}$

More Intractable Problems

Graph coloring

Input: graph $G = (V, E)$

Output: $c : V \to [k]$ such that $(\forall (u, v) \in E)\, c(u) \neq c(v))$ and k is minimized

Traveling salesperson problem (TSP)

Input: n cities and direct intercity travel costs for each pair

Output: tour that visits every city once and has minimum total cost

Subset sum

Input: $a_1, a_2, \ldots, a_n \in \mathbb{N}$; $t \in \mathbb{N}$

Output: $I \subseteq [n]$ such that $\sum_{i \in I} a_i = t$, or report impossible

Common Pattern

Common Pattern

On input x of length $n \doteq |x|$:

Common Pattern

On input x of length $n \doteq |x|$:

- Candidate solutions can be described by strings $y \in \{0,1\}^*$ with $|y| = n^c$ for some constant c.

Common Pattern

On input x of length $n \doteq |x|$:

- Candidate solutions can be described by strings $y \in \{0,1\}^*$ with $|y| = n^c$ for some constant c.
- Whether a candidate solution $y \in \{0,1\}^{n^c}$ is valid for input x can be checked in time polynomial in n.

Common Pattern

On input x of length $n \doteq |x|$:

- Candidate solutions can be described by strings $y \in \{0,1\}^*$ with $|y| = n^c$ for some constant c.
- Whether a candidate solution $y \in \{0,1\}^{n^c}$ is valid for input x can be checked in time polynomial in n.

$$V(x,y) = \begin{cases} 1 & \text{if } y \text{ is valid for } x \\ 0 & \text{otherwise} \end{cases}$$

Common Pattern

On input x of length $n \doteq |x|$:

- Candidate solutions can be described by strings $y \in \{0,1\}^*$ with $|y| = n^c$ for some constant c.
- Whether a candidate solution $y \in \{0,1\}^{n^c}$ is valid for input x can be checked in time polynomial in n.

$$V(x,y) = \begin{cases} 1 & \text{if } y \text{ is valid for } x \\ 0 & \text{otherwise} \end{cases}$$

- [in case of optimization problem]
  Objective $f(x,y)$ can be evaluated in time polynomial in n.

NP Decision/Search/Optimization

NP Decision/Search/Optimization

- Parameters:
  - $c \in \mathbb{N}$
  - $V : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ in P
  - $f : \{0,1\}^* \times \{0,1\}^* \to \mathbb{R}$ in P

NP Decision/Search/Optimization

- Parameters:
  - $c \in \mathbb{N}$
  - $V : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ in P
  - $f : \{0,1\}^* \times \{0,1\}^* \to \mathbb{R}$ in P
- Solution set for input $x \in \{0,1\}^n$:

$$S_x \doteq \{y \in \{0,1\}^{n^c} : V(x,y) = 1\}$$

NP Decision/Search/Optimization

- Parameters:
  - $c \in \mathbb{N}$
  - $V : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ in P
  - $f : \{0,1\}^* \times \{0,1\}^* \to \mathbb{R}$ in P
- Solution set for input $x \in \{0,1\}^n$:

$$S_x \doteq \{y \in \{0,1\}^{n^c} : V(x,y) = 1\}$$

- Goal:

NP Decision/Search/Optimization

- Parameters:
  - $c \in \mathbb{N}$
  - $V : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ in P
  - $f : \{0,1\}^* \times \{0,1\}^* \to \mathbb{R}$ in P
- Solution set for input $x \in \{0,1\}^n$:

$$S_x \doteq \{y \in \{0,1\}^{n^c} : V(x,y) = 1\}$$

- Goal:

  Decision Is $S_x \neq \emptyset$?

NP Decision/Search/Optimization

- Parameters:
  - $c \in \mathbb{N}$
  - $V : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ in P
  - $f : \{0,1\}^* \times \{0,1\}^* \to \mathbb{R}$ in P
- Solution set for input $x \in \{0,1\}^n$:

$$S_x \doteq \{y \in \{0,1\}^{n^c} : V(x,y) = 1\}$$

- Goal:

  Decision Is $S_x \neq \emptyset$?

  Search Find $y \in S_x$ or report that no such y exists.

NP Decision/Search/Optimization

- Parameters:
  - $c \in \mathbb{N}$
  - $V : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ in P
  - $f : \{0,1\}^* \times \{0,1\}^* \to \mathbb{R}$ in P
- Solution set for input $x \in \{0,1\}^n$:

$$S_x \doteq \{y \in \{0,1\}^{n^c} : V(x,y) = 1\}$$

- Goal:

Decision	Is $S_x \neq \emptyset$?
Search	Find $y \in S_x$ or report that no such y exists.
Optimization	Find $y^* \in S_x$ such that $f(x,y^*) = \min_{y \in S_x}(f(x,y))$ respectively $f(x,y^*) = \max_{y \in S_x}(f(x,y))$

P vs NP

P vs NP

Definition

P vs NP

Definition

- P: class of decision problems computable in polynomial time

P vs NP

Definition

- P: class of decision problems computable in polynomial time
- NP: class of NP decision problems

P vs NP

Definition

- P: class of decision problems computable in polynomial time
- NP: class of NP decision problems, i.e., decision problems for which yes-instances have certificates that can be verified in polynomial time.

P vs NP

Definition

- P: class of decision problems computable in polynomial time
- NP: class of NP decision problems, i.e., decision problems for which yes-instances have certificates that can be verified in polynomial time.

Fact: $P \subseteq NP$

P vs NP

Definition

- P: class of decision problems computable in polynomial time
- NP: class of NP decision problems, i.e., decision problems for which yes-instances have certificates that can be verified in polynomial time.

Fact: $\mathrm{P} \subseteq \mathrm{NP}$

Open: $\mathrm{P} = \mathrm{NP}$

P vs NP

Definition

- P: class of decision problems computable in polynomial time
- NP: class of NP decision problems, i.e., decision problems for which yes-instances have certificates that can be verified in polynomial time.

Fact: $\mathrm{P} \subseteq \mathrm{NP}$

Open: $\mathrm{P} = \mathrm{NP}$

Conjecture: $\mathrm{P} \neq \mathrm{NP}$

P vs NP

Definition

- P: class of decision problems computable in polynomial time
- NP: class of NP decision problems, i.e., decision problems for which yes-instances have certificates that can be verified in polynomial time.

Fact: P $\subseteq$ NP

Open: P $=$ NP

Conjecture: P $\neq$ NP

Assuming P $\neq$ NP, the "hardest" problems in NP cannot be solved in polynomial time (but some problems in NP can).

NP-Hardness and NP-Completeness

NP-Hardness and NP-Completeness

Definition

B is NP-hard if $(\forall\, A \in \mathrm{NP})\, A \leq^p B$.

NP-Hardness and NP-Completeness

Definition

B is NP-hard if $(\forall A \in \mathrm{NP})\, A \leq^p B$.

Definition

B is NP-complete if B is NP-hard and $B \in \mathrm{NP}$.

NP-Hardness and NP-Completeness

Definition

B is NP-hard if $(\forall A \in \mathrm{NP})\, A \leq^p B$.

Definition

B is NP-complete if B is NP-hard and $B \in \mathrm{NP}$.

Proposition

Suppose B is NP-complete. Then $B \in \mathrm{P} \Leftrightarrow \mathrm{P} = \mathrm{NP}$.

NP-Hardness and NP-Completeness

Definition

B is NP-hard if $(\forall\, A \in \mathrm{NP})\, A \leq^p B$.

Definition

B is NP-complete if B is NP-hard and $B \in \mathrm{NP}$.

Proposition

Suppose B is NP-complete. Then $B \in \mathrm{P} \Leftrightarrow \mathrm{P} = \mathrm{NP}$.

Proof

NP-Hardness and NP-Completeness

Definition

B is NP-hard if $(\forall A \in \mathrm{NP})\, A \leq^p B$.

Definition

B is NP-complete if B is NP-hard and $B \in \mathrm{NP}$.

Proposition

Suppose B is NP-complete. Then $B \in \mathrm{P} \Leftrightarrow \mathrm{P} = \mathrm{NP}$.

Proof

$\Leftarrow$ $B \in \mathrm{NP}$ and $\mathrm{P} = \mathrm{NP}$ implies $B \in \mathrm{P}$.

NP-Hardness and NP-Completeness

Definition

B is NP-hard if $(\forall\, A \in \mathrm{NP})\, A \leq^p B$.

Definition

B is NP-complete if B is NP-hard and $B \in \mathrm{NP}$.

Proposition

Suppose B is NP-complete. Then $B \in \mathrm{P} \Leftrightarrow \mathrm{P} = \mathrm{NP}$.

Proof

- $\Leftarrow$ $B \in \mathrm{NP}$ and $\mathrm{P} = \mathrm{NP}$ implies $B \in \mathrm{P}$.
- $\Rightarrow$ Consider any $A \in \mathrm{NP}$.

NP-Hardness and NP-Completeness

Definition
B is NP-hard if $(\forall A \in \mathrm{NP})\, A \leq^p B$.

Definition
B is NP-complete if B is NP-hard and $B \in \mathrm{NP}$.

Proposition
Suppose B is NP-complete. Then $B \in \mathrm{P} \Leftrightarrow \mathrm{P} = \mathrm{NP}$.

Proof
$\Leftarrow$ $B \in \mathrm{NP}$ and $\mathrm{P} = \mathrm{NP}$ implies $B \in \mathrm{P}$.

$\Rightarrow$ Consider any $A \in \mathrm{NP}$.
$A \leq^p B$ and $B \in \mathrm{P}$ implies $A \in \mathrm{P}$.

NP-Hardness and NP-Completeness

Definition

B is NP-hard if $(\forall A \in \mathrm{NP})\, A \leq^p B$.

Definition

B is NP-complete if B is NP-hard and $B \in \mathrm{NP}$.

Proposition

Suppose B is NP-complete. Then $B \in \mathrm{P} \Leftrightarrow \mathrm{P} = \mathrm{NP}$.

Proof

- $\Leftarrow$ $B \in \mathrm{NP}$ and $\mathrm{P} = \mathrm{NP}$ implies $B \in \mathrm{P}$.
- $\Rightarrow$ Consider any $A \in \mathrm{NP}$.
  $A \leq^p B$ and $B \in \mathrm{P}$ implies $A \in \mathrm{P}$.

Corollary

Assume $\mathrm{P} \neq \mathrm{NP}$. If B is NP-hard then $B \notin \mathrm{P}$.

Existence of NP-Complete Problems

Existence of NP-Complete Problems

Theorem (next lecture)

CNF-SAT is NP-hard.

Existence of NP-Complete Problems

Theorem (next lecture)

CNF-SAT is NP-hard.

Corollary

Independent Set is NP-hard.

Existence of NP-Complete Problems

Theorem (next lecture)

CNF-SAT is NP-hard.

Corollary

Independent Set is NP-hard.

Proof

Consider any $A \in \mathrm{NP}$.

Existence of NP-Complete Problems

Theorem (next lecture)

CNF-SAT is NP-hard.

Corollary

Independent Set is NP-hard.

Proof

Consider any $A \in \mathrm{NP}$.

- By the NP-hardness of CNF-SAT, $A \leq^p$ CNF-SAT.

Existence of NP-Complete Problems

Theorem (next lecture)

CNF-SAT is NP-hard.

Corollary

Independent Set is NP-hard.

Proof

Consider any $A \in \mathrm{NP}$.

- By the NP-hardness of CNF-SAT, $A \leq^p$ CNF-SAT.
- By previous lecture, CNF-SAT $\leq^p$ Independent Set.

Existence of NP-Complete Problems

Theorem (next lecture)

CNF-SAT is NP-hard.

Corollary

Independent Set is NP-hard.

Proof

Consider any $A \in \mathrm{NP}$.

- By the NP-hardness of CNF-SAT, $A \leq^p$ CNF-SAT.
- By previous lecture, CNF-SAT $\leq^p$ Independent Set.
- By transitivity, $A \leq^p$ Independent Set.

Proving NP-Hardness

Proving NP-Hardness

Strategy

Proving NP-Hardness

Strategy

To show a new problem C is NP-hard:

Proving NP-Hardness

Strategy

To show a new problem C is NP-hard:

- Find a known NP-complete problem B.

Proving NP-Hardness

Strategy

To show a new problem C is NP-hard:

- Find a known NP-complete problem B.
- Show that $B \leq^p C$.

Proving NP-Hardness

Strategy

To show a new problem C is NP-hard:

- Find a known NP-complete problem B.
- Show that $B \leq^p C$.

Motivation

Proving NP-Hardness

Strategy

To show a new problem C is NP-hard:

- Find a known NP-complete problem B.
- Show that $B \leq^p C$.

Motivation

- Recognizing infeasible approaches.

Proving NP-Hardness

Strategy

To show a new problem C is NP-hard:

- Find a known NP-complete problem B.
- Show that $B \leq^p C$.

Motivation

- Recognizing infeasible approaches.
- Convincing your boss

Proving NP-Hardness

Strategy

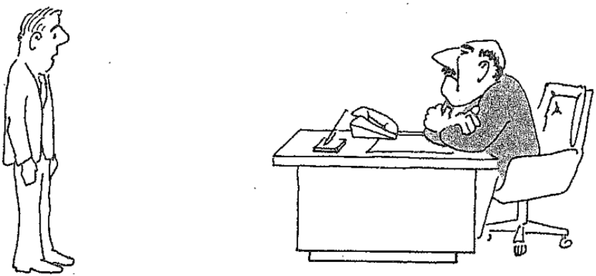
To show a new problem C is NP-hard:

- Find a known NP-complete problem B.
- Show that $B \leq^p C$.

Motivation

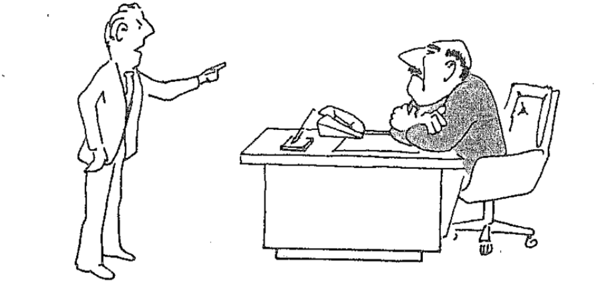
- Recognizing infeasible approaches.
- Convincing your boss [Garey and Johnson, Computers and Intractability – A guide to the Theory of NP-Completeness]

Motivation for Proving NP-Hardness



"I can't find an efficient algorithm, I guess I'm just too dumb."

Motivation for Proving NP-Hardness



“I can’t find an efficient algorithm, because no such algorithm is possible!”

Motivation for Proving NP-Hardness

"I can't find an efficient algorithm, but neither can all these famous people."

Prevalence of NP-completeness

Prevalence of NP-completeness

- Thousands of problems from all areas of science and engineering have been shown to be NP-complete.

Prevalence of NP-completeness

- Thousands of problems from all areas of science and engineering have been shown to be NP-complete.
- Considered strong evidence that $P \neq NP$.

Prevalence of NP-completeness

- Thousands of problems from all areas of science and engineering have been shown to be NP-complete.
- Considered strong evidence that $P \neq NP$.
- In fact, almost all of the known problems in NP that are not known to be in P, have been shown to be NP-hard.

Prevalence of NP-completeness

- Thousands of problems from all areas of science and engineering have been shown to be NP-complete.
- Considered strong evidence that $\mathrm{P} \neq \mathrm{NP}$.
- In fact, almost all of the known problems in NP that are not known to be in P, have been shown to be NP-hard.
- Notorious exceptions include: graph isomorphism, factoring integers.