GEMS OF TCS

EASY AND HARD PROBLEMS

Sasha Golovnev
August 24, 2021
THEORETICAL COMPUTER SCIENCE
$P \iff Q$

Mathematical logic
$P \implies Q$

- **Mathematical logic**
- **Computability theory**
**THEORETICAL COMPUTER SCIENCE**

\[ P \implies Q \]

- Mathematical logic
- Computability theory
- Information theory
THEORETICAL COMPUTER SCIENCE

\[ P \implies Q \]

- Mathematical logic
- Computability theory
- Information theory

Learning, neural nets
THEORETICAL COMPUTER SCIENCE

\[ P \implies Q \]

Mathematical logic

Computability theory

Information theory

\[ P = NP? \]

Learning, neural nets

Computational complexity

\[ H(X|Y) \cap H(Y|X) \]
THEORETICAL COMPUTER SCIENCE

\[ P \quad \Rightarrow \quad Q \]

Mathematical logic

Computability theory

Learning, neural nets

Computational complexity

Information theory

Cryptography

\[ P = NP? \]
**Theoretical Computer Science**

\[ P \quad \Rightarrow \quad Q \]

- Mathematical logic
- Computability theory
- Information theory
- Learning, neural nets
- Computational complexity
- Quantum Algorithms
- Cryptography

\[ P = NP? \]
**THEORETICAL COMPUTER SCIENCE**

- \( P \Rightarrow Q \)
  - Mathematical logic
- Learning, neural nets
- Quantum Algorithms
- Computability theory
- Information theory
- \( P = NP? \)
  - Computational complexity
- Cryptography
  - Machine learning
THEORETICAL COMPUTER SCIENCE

$P \implies Q$

- Mathematical logic
- Computability theory
- Information theory

- Learning, neural nets
- Computational complexity
- Cryptography

- Quantum Algorithms
- Machine learning
- Data Science

$P = NP$?
THIS COURSE

• Theoretical/Mathematical viewpoint
THIS COURSE

- Theoretical/Mathematical viewpoint
- Topic overview
This Course

- Theoretical/Mathematical viewpoint
- Topic overview
  - Algorithms
- Computational Complexity
- Cryptography
- Learning
This Course

- Theoretical/Mathematical viewpoint

- Topic overview
  - Algorithms
  - Computational Complexity
THIS COURSE

• Theoretical/Mathematical viewpoint

• Topic overview
  • Algorithms
  • Computational Complexity
  • Cryptography
This Course

• Theoretical/Mathematical viewpoint

• Topic overview
  • Algorithms
  • Computational Complexity
  • Cryptography
  • Learning
ADMINISTRATIVE INFO

• Classes: MW 12:30pm–1:45pm, Walsh 396
ADMINISTRATIVE INFO

• Classes: MW 12:30pm–1:45pm, Walsh 396
• Office Hours: M 2:00pm–3:00pm, SMH 354
Administrative Info

- Classes: MW 12:30pm–1:45pm, Walsh 396
- Office Hours: M 2:00pm–3:00pm, SMH 354
- Prerequisites: Algorithms or Theory of Computation, a Programming Language

Webpage: https://golovnev.org/gradgems
Grading: 5-6 Problem Sets
Email: alexgolovnev+gems@gmail.com
ADMINISTRATIVE INFO

• Classes: MW 12:30pm–1:45pm, Walsh 396
• Office Hours: M 2:00pm–3:00pm, SMH 354
• Prerequisites: Algorithms or Theory of Computation, a Programming Language
• Webpage: https://golovnev.org/gradgems
• email: alexgolovnev+gems@gmail.com
ADMINISTRATIVE INFO

• Classes: MW 12:30pm–1:45pm, Walsh 396
• Office Hours: M 2:00pm–3:00pm, SMH 354
• Prerequisites: Algorithms or Theory of Computation, a Programming Language
• Webpage: https://golovnev.org/gradgems
• Grading: 5-6 Problem Sets
ADMINISTRATIVE INFO

• Classes: MW 12:30pm–1:45pm, Walsh 396
• Office Hours: M 2:00pm–3:00pm, SMH 354
• Prerequisites: Algorithms or Theory of Computation, a Programming Language
• Webpage: https://golovnev.org/gradgems
• Grading: 5-6 Problem Sets
• email: alexgolovnev+gems@gmail.com
• Running time of an algorithm
• Running time of an algorithm
  • $100n^2$ vs $n^3/10$
Course Begins

- Running time of an algorithm
  - $100n^2$ vs $n^3/10$
  - $100n^2$ vs $2^n/100$
- Running time of an algorithm
  - $100n^2$ vs $n^3/10$
  - $100n^2$ vs $2^n/100$
- Complexity class $\mathbf{P}$: Problems whose solution can be found efficiently
Course Begins

- Running time of an algorithm
  - $100n^2$ vs $n^3/10$
  - $100n^2$ vs $2^n/100$
- Complexity class **P**: Problems whose solution can be found efficiently
- Complexity class **NP**: Problems whose solution can be verified efficiently
The main open problem in Computer Science

Is $P$ equal to $NP$?
The main open problem in Computer Science

Is $\text{P}$ equal to $\text{NP}$?

Millenium Prize Problem

Clay Mathematics Institute: $1M prize for solving the problem
• If $P=NP$, then all $NP$-problems can be solved in polynomial time.

• If $P \neq NP$, then there exist $NP$-problems that cannot be solved in polynomial time.
• If $\mathbf{P}=\mathbf{NP}$, then all $\mathbf{NP}$-problems can be solved in polynomial time.

• If $\mathbf{P}\neq \mathbf{NP}$, then there exist $\mathbf{NP}$-problems that cannot be solved in polynomial time.
NP-complete Problems

• The “hardest” problems in NP
NP-complete Problems

• The “hardest” problems in NP

• If any NP-complete problem can be solved in polynomial time, then all of NP can be solved in polynomial time
**NP-complete Problems**

- The “hardest” problems in $\text{NP}$
- If any $\text{NP}$-complete problem can be solved in polynomial time, then all of $\text{NP}$ can be solved in polynomial time
- If one $\text{NP}$-complete problem cannot be solved in polynomial time, then all $\text{NP}$-complete problems cannot be solved in polynomial time
NP-complete Problems

- The “hardest” problems in \( \text{NP} \)
- If any \( \text{NP} \)-complete problem can be solved in polynomial time, then all of \( \text{NP} \) can be solved in polynomial time
- If one \( \text{NP} \)-complete problem cannot be solved in polynomial time, then all \( \text{NP} \)-complete problems cannot be solved in polynomial time
- Later we’ll show \( \text{NP} \)-complete problems exist!
Car Fueling
CAR FUELING

Distance with full tank 300 mi.
Minimize the number of stops at gas stations
Distance with full tank 300 mi.

Minimize the number of stops at gas stations
CAR FUELING. SOLUTION

- “Greedy” algorithm
Car Fueling. Solution

- “Greedy” algorithm

- Runs in linear time $O(n)$, where $n$ is the size of the input (# of gas stations)
CAR FUELING. SOLUTION

• “Greedy” algorithm

• Runs in linear time $O(n)$, where $n$ is the size of the input (# of gas stations)

• Easy problem
Traveling Salesman Problem (TSP)
Given a complete weighted graph, find a cycle (or a path) of minimum total weight (length) visiting each node exactly once.
Traveling Salesman Problem

Given a complete weighted graph, find a cycle (or a path) of minimum total weight (length) visiting each node exactly once.
TRAVELING SALESMAN PROBLEM

Given a complete weighted graph, find a cycle (or a path) of minimum total weight (length) visiting each node exactly once.

length: 11
TRAVELING SALESMAN PROBLEM

Given a complete weighted graph, find a cycle (or a path) of minimum total weight (length) visiting each node exactly once.

length: 9
STATUS

• Classical optimization problem with countless number of real life applications (we’ll see soon)
Status

- Classical optimization problem with countless number of real life applications (we’ll see soon)
- No polynomial time algorithms known
STATUS

• Classical optimization problem with countless number of real life applications (we’ll see soon)
• No polynomial time algorithms known
• The best known algorithm runs in time $2^n$
DELIVERING GOODS

Need to visit several points. What is the optimal order of visiting them?
TRAVELING
TRAVELING
TRAVELING
Drilling a Circuit Board

https://developers.google.com/optimization/routing/tsp/tsp
DRILLING A CIRCUIT BOARD

https://developers.google.com/optimization/routing/tsp/tsp
There are $n$ mechanical components to be processed on a complex machine. After processing the $i$-th component, it takes $t_{ij}$ units of time to reconfigure the machine so that it is able to process the $j$-th component. What is the minimum processing cost?
Euclidean TSP

- Euclidean TSP: instead of a complete graph, the input consists of \( n \) points 
  \( p_1 = (x_1, y_1), \ldots, p_n = (x_n, y_n) \) on the plane
Euclidean TSP

- Euclidean TSP: instead of a complete graph, the input consists of \( n \) points \( p_1 = (x_1, y_1), \ldots, p_n = (x_n, y_n) \) on the plane.
- Weights are given implicitly:

\[
d(p_i, p_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}
\]
Euclidean TSP

- **Euclidean TSP**: instead of a complete graph, the input consists of \( n \) points \( p_1 = (x_1, y_1), \ldots, p_n = (x_n, y_n) \) on the plane
- Weights are given implicitly:
  \[
  d(p_i, p_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}
  \]
- Weights are symmetric: \( d(p_i, p_j) = d(p_j, p_i) \)
Euclidean TSP

- **Euclidean TSP**: instead of a complete graph, the input consists of \( n \) points \( p_1 = (x_1, y_1), \ldots, p_n = (x_n, y_n) \) on the plane.

- Weights are given implicitly:

\[
d(p_i, p_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}
\]

- Weights are symmetric: \( d(p_i, p_j) = d(p_j, p_i) \)

- Weights satisfy the triangle inequality: \( d(p_i, p_j) \leq d(p_i, p_k) + d(p_k, p_j) \)
Brute Force Search

• Finding the best permutation is easy: simply iterate through all of them and select the best one
Finding the best permutation is easy: simply iterate through all of them and select the best one.

But the number of permutations of $n$ objects is $n!$. 
$n!$: GROWTH RATE

<table>
<thead>
<tr>
<th>$n$</th>
<th>$n!$</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>120</td>
</tr>
<tr>
<td>8</td>
<td>40320</td>
</tr>
<tr>
<td>10</td>
<td>3628800</td>
</tr>
<tr>
<td>13</td>
<td>6227020800</td>
</tr>
<tr>
<td>20</td>
<td>24329020081766400000</td>
</tr>
<tr>
<td>30</td>
<td>2652528598121910586363084800000000</td>
</tr>
</tbody>
</table>
Satisfiability Problem (SAT)
SAT

\((x_1 \lor x_2 \lor x_3) \land (x_1 \lor \neg x_2) \land (\neg x_1 \lor x_3) \land (x_2 \lor \neg x_3)\)
\[(x_1 \lor x_2 \lor x_3) \land (x_1 \lor \neg x_2) \land (\neg x_1 \lor x_3) \land (x_2 \lor \neg x_3) \land (x_1 \lor x_2 \lor x_3) \land (x_1 \lor \neg x_2) \land (\neg x_1 \lor x_3) \land (x_2 \lor \neg x_3) \land (\neg x_1 \lor \neg x_2 \lor \neg x_3)\]
APPLICATIONS OF SAT

• Software Engineering
• Chip testing
• Circuit design
• Automatic theorem provers
• Image analysis
• ...
$k$-SAT

\[ \phi(x_1, \ldots, x_n) = (x_1 \lor \neg x_2 \lor \ldots \lor x_k) \land \ldots \land (x_2 \lor \neg x_3 \lor \ldots \lor x_8) \]
$k$-SAT

$$
\phi(x_1, \ldots, x_n) = (x_1 \lor \neg x_2 \lor \ldots \lor x_k) \land \\
\ldots \land \\
(x_2 \lor \neg x_3 \lor \ldots \lor x_8)
$$

$\phi$ is *satisfiable* if

$$
\exists x \in \{0, 1\}^n : \phi(x) = 1.
$$

Otherwise, $\phi$ is *unsatisfiable*
$k$-SAT

\[ \phi(x_1, \ldots, x_n) = (x_1 \lor \neg x_2 \lor \ldots \lor x_k) \land \ldots \land (x_2 \lor \neg x_3 \lor \ldots \lor x_8) \]

$\phi$ is satisfiable if

\[ \exists x \in \{0, 1\}^n : \phi(x) = 1 . \]

Otherwise, $\phi$ is unsatisfiable

$k$-SAT is SAT where clause length $\leq k$
$k$-SAT. EXAMPLES

$$(x_1 \lor x_2 \lor x_3) \land (x_1 \lor \neg x_2) \land (\neg x_1 \lor x_3) \land (x_2 \lor \neg x_3)$$
$k$-SAT. EXAMPLES

$$(x_1 \lor x_2 \lor x_3) \land (x_1 \lor \neg x_2) \land (\neg x_1 \lor x_3) \land (x_2 \lor \neg x_3)$$

$$(x_1) \land (\neg x_2) \land (x_3) \land (\neg x_1)$$
Queen of NP-complete Problems

- Cook-Levin Theorem [Coo71, Lev73]: SAT can model non-deterministic Turing machine:
  SAT is NP-complete
QUEEN OF NP-COMPLETE PROBLEMS

• Cook-Levin Theorem [Coo71, Lev73]: SAT can model non-deterministic Turing machine: SAT is NP-complete

• 3-SAT is NP-complete
QUEEN OF NP-COMPLETE PROBLEMS

• Cook-Levin Theorem [Coo71, Lev73]: SAT can model non-deterministic Turing machine:
  SAT is NP-complete

• 3-SAT is NP-complete

• 2-SAT is in P
Complexity of SAT

- 1-SAT
- 2-SAT
- 3-SAT...
- k-SAT
Complexity of SAT

SAT

$k$-SAT

:\

3-SAT

2-SAT

1-SAT

P

NP
The SAT game