Gems of TCS

Easy and Hard Problems

Sasha Golovnev

August 24, 2022

Theoretical Computer Science

Theoretical Computer Science

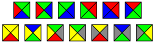
$$P \implies Q$$

Mathematical logic

Theoretical Computer Science

$$P \implies Q$$

Mathematical logic

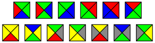


Computability theory

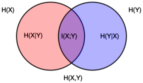
Theoretical Computer Science

$$P \implies Q$$

Mathematical logic



Computability theory

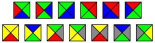


Information theory

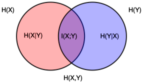
Theoretical Computer Science

$$P \implies Q$$

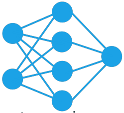
Mathematical logic



Computability theory



Information theory

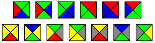


Learning, neural nets

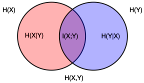
Theoretical Computer Science

$P \implies Q$

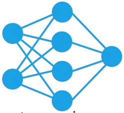
Mathematical logic



Computability theory



Information theory



Learning, neural nets

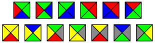
$P = NP?$

Computational complexity

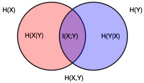
Theoretical Computer Science

$P \implies Q$

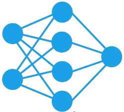
Mathematical logic



Computability theory



Information theory



Learning, neural nets

$P = NP?$

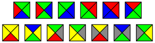
Computational complexity

Cryptography

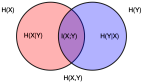
Theoretical Computer Science

$$P \implies Q$$

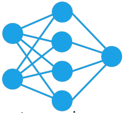
Mathematical
logic



Computability
theory



Information
theory

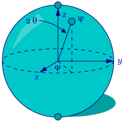


Learning,
neural nets

$$P = NP?$$

Computational
complexity

Cryptography

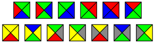


Quantum
Algorithms

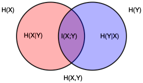
Theoretical Computer Science

$$P \implies Q$$

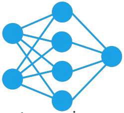
Mathematical logic



Computability theory



Information theory

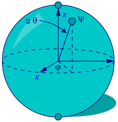


Learning, neural nets

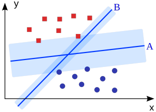
$$P = NP?$$

Computational complexity

Cryptography



Quantum Algorithms

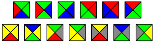


Machine learning

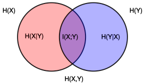
Theoretical Computer Science

$P \implies Q$

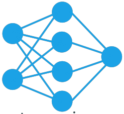
Mathematical logic



Computability theory



Information theory

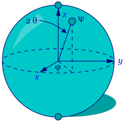


Learning, neural nets

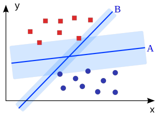
$P = NP?$

Computational complexity

Cryptography



Quantum Algorithms



Machine learning

Data Science

This Course

- Theoretical/Mathematical viewpoint

This Course

- Theoretical/Mathematical viewpoint
- Topic overview

This Course

- Theoretical/Mathematical viewpoint

- Topic overview
    - Algorithms

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, STM 354

Administrative Info

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

Administrative Info

- Classes: MW 12:30pm–1:45pm, Walsh 396
- Office Hours: M 2:00pm–3:00pm, STM 354
- Prerequisites: Algorithms or Theory of Computation, a Programming Language
- Webpage: https://golovnev.org/gradgems

Administrative Info

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

Administrative Info

- Classes: MW 12:30pm–1:45pm, Walsh 396
- Office Hours: M 2:00pm–3:00pm, STM 354
- Prerequisites: Algorithms or Theory of Computation, a Programming Language
- Webpage: https://golovnev.org/gradgems
- Grading: ≈ 5 Problem Sets
- email: `alexgolovnev+gems@gmail.com`

Course Begins

- Running time of an algorithm

Course Begins

- 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 $2^n/100$

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

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 **P** equal to **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=NP, then all NP-problems can be solved in polynomial time.

- If P≠NP, then there exist 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 NP
- If any NP-complete problem can be solved in polynomial time, then all of NP can be solved in polynomial time
- If one NP-complete problem cannot be solved in polynomial time, then all NP-complete problems cannot be solved in polynomial time

NP-complete Problems

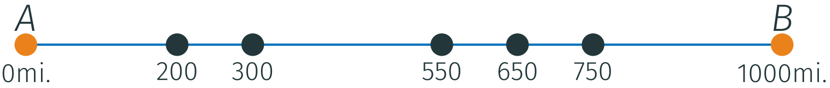
- The "hardest" problems in NP
- If any NP-complete problem can be solved in polynomial time, then all of NP can be solved in polynomial time
- If one NP-complete problem cannot be solved in polynomial time, then all NP-complete problems cannot be solved in polynomial time
- Later we'll show NP-complete problems exist!

Car Fueling

Car Fueling

Distance with full tank 300 mi.

Minimize the number of stops at gas stations

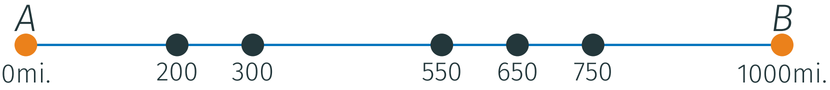


Break http://bit.ly/car-fueling

Example

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)

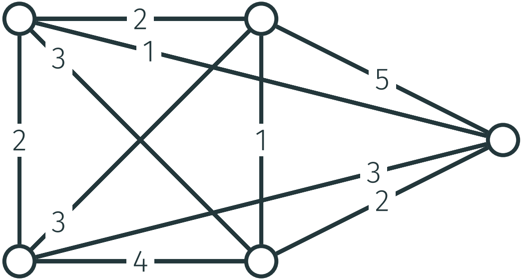
Car Fueling. Solution

- “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)

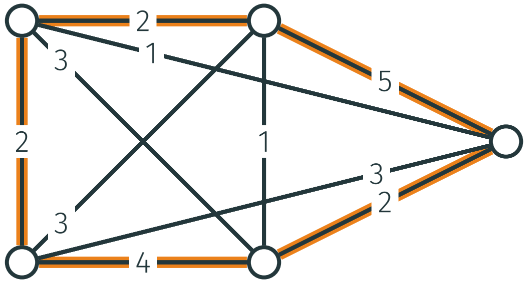
Traveling Salesman Problem

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



Traveling Salesman Problem

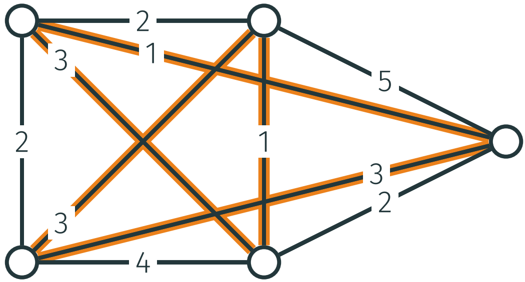
Given a complete weighted graph, find a cycle (or a path) of minimum total weight (length) visiting each node exactly once



length: 15

Traveling Salesman Problem

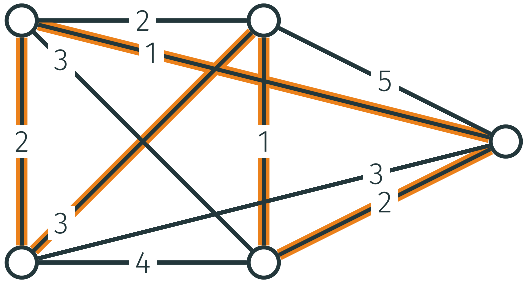
Given a complete weighted graph, find a cycle (or a path) of minimum total weight (length) visiting each node exactly once



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

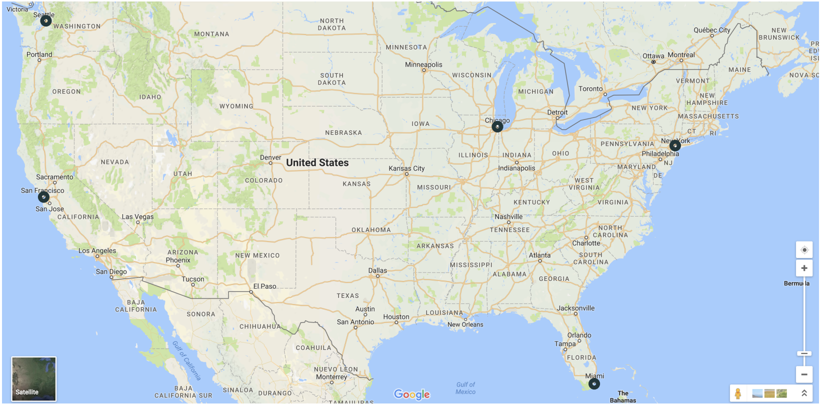
Delivering Goods

Need to visit several points. What is the optimal order of visiting them?

Traveling


Victoria
Seattle
WASHINGTON
Portland
OREGON
IDAHO
MONTANA
NORTH DAKOTA
SOUTH DAKOTA
WYOMING
NEBRASKA
MINNESOTA
Minneapolis
WISCONSIN
MICHIGAN
Toronto
Québec City
Ottawa
Montreal
NEW BRUNSWICK
MAINE
VERMONT
NEW HAMPSHIRE
MASSACHUSETTS
CT
RI
NEW YORK
New York
Detroit
Chicago
IOWA
ILLINOIS
INDIANA
OHIO
PENNSYLVANIA
Philadelphia
NJ
MARYLAND
DE
Indianapolis
WEST VIRGINIA
VIRGINIA
KENTUCKY
Nashville
TENNESSEE
NORTH CAROLINA
Charlotte
SOUTH CAROLINA
Atlanta
GEORGIA
ALABAMA
MISSISSIPPI
ARKANSAS
MISSOURI
Kansas City
KANSAS
OKLAHOMA
United States
Denver
COLORADO
UTAH
NEVADA
Sacramento
San Francisco
San Jose
CALIFORNIA
Las Vegas
Los Angeles
San Diego
ARIZONA
Phoenix
Tucson
NEW MEXICO
El Paso
Dallas
TEXAS
Austin
San Antonio
Houston
LOUISIANA
New Orleans
Jacksonville
Orlando
Tampa
FLORIDA
Miami
The Bahamas
Bermuda
BAJA CALIFORNIA
SONORA
CHIHUAHUA
COAHUILA
NUEVO LEON
Monterrey
BAJA CALIFORNIA SUR
SINALOA
DURANGO
Gulf of California
Gulf of Mexico
Google
Satellite

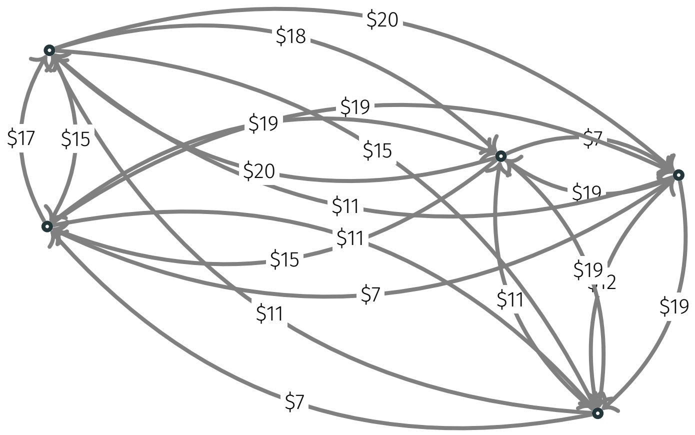
Traveling



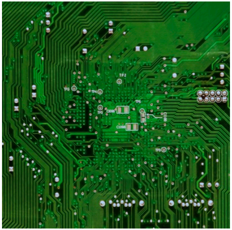
Victoria
Seattle
WASHINGTON
MONTANA
NORTH DAKOTA
MINNESOTA
Québec City
NEW BRUNSWICK
Portland
Minneapolis
WISCONSIN
Ottawa
Montreal
MAINE
NOVA SC
OREGON
IDAHO
SOUTH DAKOTA
MICHIGAN
Toronto
VERMONT
NEW HAMPSHIRE
WYOMING
NEW YORK
Detroit
MASSACHUSETTS
Chicago
IOWA
NEBRASKA
CT
RI
PENNSYLVANIA
New York
NEVADA
Denver
United States
ILLINOIS
INDIANA
OHIO
Philadelphia
NJ
UTAH
Kansas City
Indianapolis
MARYLAND
DE
Sacramento
COLORADO
KANSAS
MISSOURI
WEST VIRGINIA
San Francisco
KENTUCKY
VIRGINIA
San Jose
CALIFORNIA
Las Vegas
Nashville
OKLAHOMA
TENNESSEE
NORTH CAROLINA
Charlotte
ARKANSAS
Los Angeles
ARIZONA
Phoenix
NEW MEXICO
SOUTH CAROLINA
Atlanta
MISSISSIPPI
Dallas
San Diego
ALABAMA
GEORGIA
Tucson
El Paso
BAJA CALIFORNIA
TEXAS
SONORA
Austin
LOUISIANA
Jacksonville
Houston
San Antonio
New Orleans
CHIHUAHUA
Orlando
Tampa
COAHUILA
FLORIDA
NUEVO LEON
Monterrey
Miami
BAJA CALIFORNIA SUR
SINALOA
DURANGO
Gulf of Mexico
The Bahamas
Satellite
Google

Traveling

Traveling

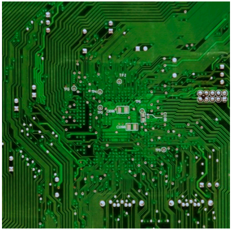


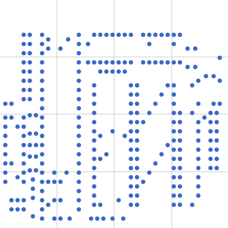
Drilling a Circuit Board



https://developers.google.com/optimization/routing/tsp/tsp

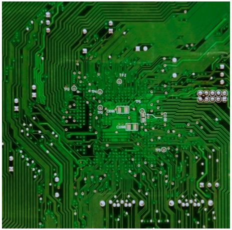
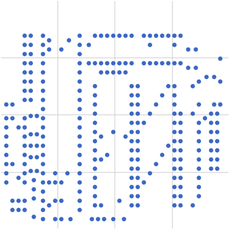
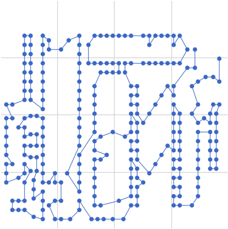
Drilling a Circuit Board





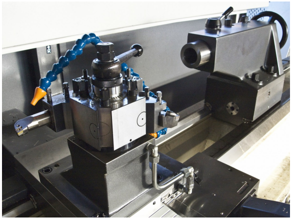
https://developers.google.com/optimization/routing/tsp/tsp

Drilling a Circuit Board

https://developers.google.com/optimization/routing/tsp/tsp

Processing Components



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

Brute Force Search

- 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

n	$n!$
5	120
8	40320
10	3628800
13	6227020800
20	2432902008176640000
30	265252859812191058636308480000000

Satisfiability Problem (SAT)

SAT

$$(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_3) \wedge (x_2 \vee \neg x_3)$$

SAT

$$(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_3) \wedge (x_2 \vee \neg x_3)$$

$$(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_3) \wedge (x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3)$$

Applications of SAT

- Software Engineering
- Chip testing
- Circuit design
- Automatic theorem provers
- Image analysis
- ...

k-SAT

$$
\begin{aligned}
\phi(x_1, \ldots, x_n) = & (x_1 \lor \neg x_2 \lor \ldots \lor x_k) \quad \land \\
& \qquad\qquad \ldots \qquad\qquad\quad \land \\
& (x_2 \lor \neg x_3 \lor \ldots \lor x_8)
\end{aligned}
$$

k-SAT

$$
\begin{aligned}
\phi(x_1, \dots, x_n) = & (x_1 \vee \neg x_2 \vee \dots \vee x_k) \quad \wedge \\
& \qquad\qquad \dots \qquad\qquad\quad \wedge \\
& (x_2 \vee \neg x_3 \vee \dots \vee x_8)
\end{aligned}
$$

ϕ is satisfiable if

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

Otherwise, ϕ is unsatisfiable

k-SAT

$$\begin{aligned}\phi(x_1,\dots,x_n) = &(x_1 \vee \neg x_2 \vee \dots \vee x_k) \quad \wedge \\ &\qquad\qquad \dots \qquad\qquad\qquad \wedge \\ &(x_2 \vee \neg x_3 \vee \dots \vee x_8)\end{aligned}$$

ϕ is satisfiable if

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

Otherwise, ϕ is unsatisfiable

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

k-SAT. Examples

$$(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_3) \wedge (x_2 \vee \neg x_3)$$

k-SAT. Examples

$$(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_3) \wedge (x_2 \vee \neg x_3)$$

$$(x_1) \wedge (\neg x_2) \wedge (x_3) \wedge (\neg x_1)$$

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

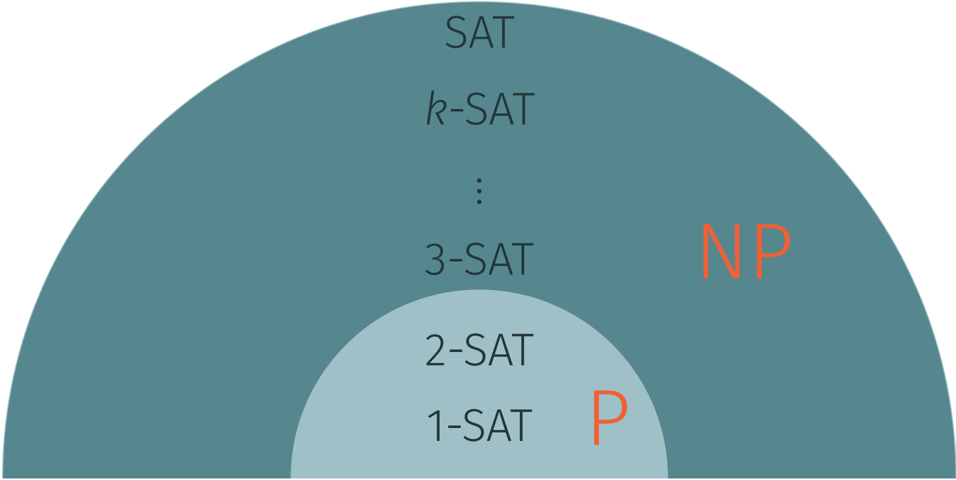
- Cook-Levin Theorem [Coo71, Lev73]: SAT can model non-deterministic Turing machine: SAT is NP-complete
- 3-SAT is NP-complete

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

Complexity of SAT



The SAT game

http://bit.ly/sat-game