# COS 126 Lecture 24: NP-completeness

Some Important (But Hard) Problems

* TRAVELING SALESPERSON
  A salesperson needs to visit N cities.
  Find a route that minimizes travel distance.

* SCHEDULING
  A set of jobs of varying length need to be done on two identical machines before a certain deadline. Can the jobs be arranged so that the deadline is met?

* SEQUENCING
  A set of four-character fragments have been obtained by breaking up a long string into overlapping pieces. Can the fragments be reconstituted into the long string?

* SATISFIABILITY
  Is there a way to assign truth values to a given logical formula that makes it true?

# Properties of Algorithms

Step-by-step set of instructions
that can be applied in the same way
to all instances of a problem.

* FINITE
  guaranteed to terminate
* DETERMINISTIC
  always produces the same answer
  given the same inputs

Algorithms are problem-solving methods suitable for use on computers. Example: multiplication.

A given problem can be solved by many different algorithms, but some algorithms are far more efficient than others.

EFFICIENT:
"polynomial" time (ex: N^2) for all inputs

INEFFICIENT:
"exponential" time (ex: 10^N) for some inputs

## Properties of Computers

Modern computers have varying characteristics

1970’s "mainframe"

1980’s "personal computer"

1990’s "microprocessor"

Supercomputer

Networks of computers

From a theoretical standpoint,
they’re all the same!

1930’s Turing Machine

For example, none of the machines can solve
the traveling salesperson problem for 1000 cities.

## Some Numbers

| | |
|---|---|
| 10^8 | PC instructions/second |
| 10^12 | supercomputer instructions/second |
| 10^9 | seconds/year |
| 10^13 | age of universe in years (estimated) |
| 10^79 | number of electrons in the universe (estimated) |

Exponential growth dwarfs technological change:

Suppose each electron in the universe had the computing power of today's supercomputers. If they worked together for the estimated life of the universe, they couldn't solve the traveling salesperson problem on 1000 cities.

$$2^{1000} > 10^{300} \gg 10^{79} * 10^{13} * 10^{9} * 10^{12}$$

# Polynomial Time Algorithms

P:

The set of all problems solvable by
deterministic algorithms in polynomial time.

Covers virtually all programs
running on actual computers.
(Doesn’t matter which computer.)

NP:

The set of all problems solvable by
non-deterministic algorithms in polynomial time.

For a problem in NP,
a machine can efficiently VERIFY that
a given solution is correct

If a machine can guess (and is lucky),
it can solve a problem in NP quickly.
Actual computers can simulate Lucky Guessing,
in exponential time, by trying every possibility.

## The Main Question

Is P=NP?

If not, then some problem must be
in NP
but not in P

(every problem in P is in NP).

Nondeterminism (Lucky Guessing) seems powerful, but no one has been able to PROVE that it helps for any particular problem.

Many important problems are in NP, but no efficient solutions for them have been found. (Only exponential algorithms are known).

Should we abandon hope of finding efficient solutions?

# NP-Completeness

NP-Complete

A problem with the property that
if it can be solved efficiently,
then P=NP.

(Lucky Guessing doesn't help.)

For specific problems A and B, we can often show:
If A can be solved efficiently, then so can B.
Thus, if B is NP-complete, so is A.

Thousands of problems have been shown to be NP-complete (including traveling salesperson) in this way.

If any one of these important problems can be solved efficiently, they all can.
(Moreover, so can any problem in NP).

But how was the first problem shown to be NP-complete?

# Cook's Theorem

SATISFIABILITY is NP-complete.

Outline of proof:

Non-deterministic Turing machines can
solve problems in NP.

Logical formulas can describe Turing machines
including non-deterministic ones

Given any problem in NP
establish a correspondence
with some instance of SATISFIABILITY

SATISFIABILITY solution
gives simulation of Turing machine
solving the corresponding problem

If SATISFIABILITY can be solved quickly,
then so can any problem in NP.

Any problem reducible to SATISFIABILITY
is NP-complete, etc.

# Implications of NP–Completeness

Either:

Conventional machines can do as well as machines capable of Lucky Guessing, but we don't know how to make them do so. (P=NP)

Or:

Lucky Guessing DOES help, but is a fiction on conventional machines, since none of the NP–complete problems can be solved in polynomial time. (P!=NP)

Not many people believe that P=NP

...but it's possible.

Proof that a problem is NP–complete is usually taken as a signal to abandon hope of finding an efficient solution.

## Coping With NP-Completness

* Hope that the worst case doesn't occur
  (try to simulate Lucky Guessing)

* Change the problem
  (try for an approximate solution)

* Exploit NP-completeness
  (example: cryptography)

* Keep trying to prove that P=NP!