Algorithm Design and Analysis
Victor Adamchik CS 15-451 Spring 2014
Lecture 33 Apr 09, 2014 Carnegie Mellon University

Smale's third problem

P = NP?

Millennium Prize Problems

Seven famous problems in math stated in 2000 by the Clay Foundation $1,000,000 prize for solving any of them

One of the problems: **P** vs. **NP**

Millennium Prize Problems

Keith Devlin

If one is solved in the next few years, it'll probably be P vs. NP

If, in the year 3000, one of them is unsolved, it will be P vs. NP

Laszlo Lovasz

Polynomial Time Complexity

Is there a fixed constant c and an algorithm A such that A solves the decision problem in time $O(n^c)$?

Verifying solutions

In some problems (like SUDOKU), verifying the solution can be done efficiently

NP = Decision problems whose solutions can be verified in polynomial time in their input size

The N in NP stands for "nondeterministically"

Here's how P vs. NP is usually (informally) stated:

Let L be an algorithmic task.

Suppose there is an efficient algorithm for verifying solutions to L. "L∈NP"

Is there always also an efficient algorithm for finding solutions to L? "L∈P"

Definition of P

An input is encoded as a binary string.

P = {L ⊆ {0, 1}* | ∃ polynomial time algorithm
for deciding L}

Definition of NP

NP =
{
L ⊆ {0, 1}* | ∃ polynomial time <u>verifier</u>
R(x, y) = true, where x∈L and |y| ≤ O(|x|^c)
}

Definition of NP-hard

NP-hard = { L ⊆ {0, 1}* | ∀X ∈ NP and X $\le_p$ L}

To reduce problem X to problem L (we write X $\le_p$ L)
we want a function f that maps X to L such that:

1) f is a polynomial time computable
2) x∈ X if and only if f(x)∈ L.

In short. We need to convert X into L.

Lemma. If A $\le_p$ B and B ∈ P then A ∈ P.

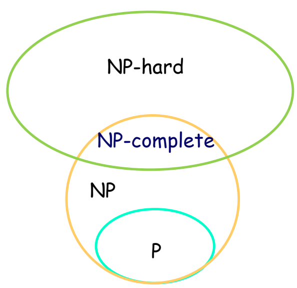
Definition of NP-complete

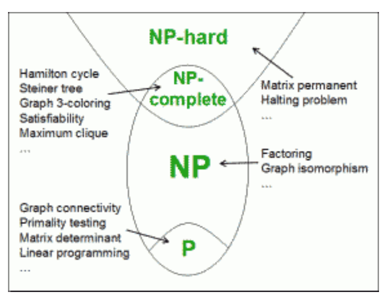
L is NP-complete iff

1) L ⊆ NP
2) L ⊆ NP-hard

2) For all Y ⊆ NP, Y $\le_p$ L

Venn Diagram (P ≠ NP)





NP-complete Reduction

A recipe for proving any $L \in$ NP-complete:

1) Prove $L \in NP$

2) Choose $A \in$ NPC and reduce it to L

2.1) Describe mapping $f: A \to L$

2.2) Prove $x \in A$ iff $f(x) \in L$

2.3) Prove f is polynomial

Conjunctive Normal Form

Let X_k denote variables.
We define literals as either X_k or $!X_k$.

The conjunctive normal form (CNF) is an AND of OR clauses. For example,

$(X_1 \vee X_2 \vee !X_3) \wedge (X_1 \vee !X_2 \vee X_4) \wedge \dots$

SAT Problem: is there exist a set of variables that satisfy a given CNF?

Cook-Levin Theorem (1971)

SAT is NP-complete

No proof, see Kozen's textbook.

3-CNF problem (or 3-SAT)

Each clause has a most 3 literals.

Question: Is there such a set of input variables that 3-cnf is true?

Theorem. 3-CNF is NP-complete

Proof.
3-CNF $\subseteq$ NP
We need to show CNF $\leq_p$ 3-CNF.

CNF $\leq_p$ 3-CNF

We need to convert any CNF into 3-CNF…

Claim:

$(a \vee b \vee c \vee d)$ is true iff
$(a \vee b \vee x) \wedge (!x \vee c \vee d)$ is true

$(a \vee b \vee c \vee d \vee e)$ converts to
$(a \vee b \vee x) \wedge (!x \vee c \vee y) \wedge (!y \vee d \vee e)$

The rest of the proof is left as an exercise to a reader.

Clique is NP-complete

1) Clique is in NP

2) We will show that SAT $\leq_p$ Clique

Create a vertex for each variable in a clause, assume k-clauses.

Two vertices (from different clauses) are connected if one is NOT negation of other.

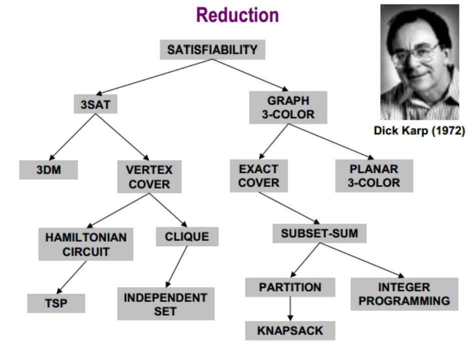
A CNF is satisfiable if at least one literal in each clause is true. Thus those literals create a k-clique.

Sudoku

Theorem (2002)

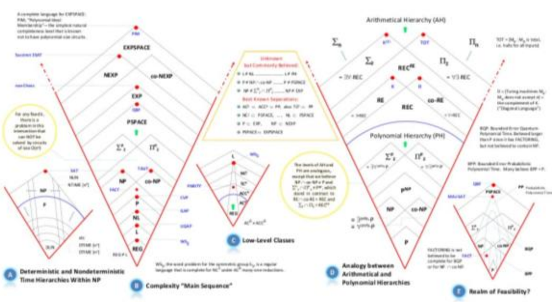
There is a polynomial reduction from 3-coloring to sudoku.

Reduction



All of these problems poly-reduce to one another!

Complexity universe



TRAVELLING SALESMAN

Travelling Salesman is an intellectual thriller about four mathematicians hired by the U.S. government to solve the most elusive problem in computer science history — P vs. NP.

The four have jointly created a "system" which could be the next major advancement for our civilization or destroy the fabric of humanity.

www.travellingsalesmanmovie.com/