CSE373: Data Structures and Algorithms

The P vs. NP question, NP-Completeness

Steve Tanimoto
Autumn 2016

This lecture material represents the work of multiple instructors at the University of Washington. Thank you to all who have contributed!

The $1M question

The Clay Mathematics Institute
Millenium Prize Problems

1. Birch and Swinnerton-Dyer Conjecture
2. Hodge Conjecture
3. Navier-Stokes Equations
4. P vs NP
5. Poincaré Conjecture
6. Riemann Hypothesis
7. Yang-Mills Theory

The P versus NP problem

Is one of the biggest open problems in computer science (and mathematics) today

It's currently unknown whether there exist polynomial time algorithms for NP-complete problems
- That is, does P = NP?
- People generally believe P ≠ NP, but no proof yet

But what is the P-NP problem?

Sudoku

2			3		8		5	
		3		4	5	9	8	
		8			9	7	3	4
6		7		9				
9	8						1	7
				5		6		9
3	1	9	7			2		
	4	6	5	2		8		
	2		9		3			1

3x3x3

Sudoku

2	9	4	3	7	8	1	5	6
1	7	3	6	4	5	9	8	2
5	6	8	2	1	9	7	3	4
6	5	7	1	9	2	3	4	8
9	8	2	4	3	6	5	1	7
4	3	1	8	5	7	6	2	9
3	1	9	7	8	4	2	6	5
7	4	6	5	2	1	8	9	3
8	2	5	9	6	3	4	7	1

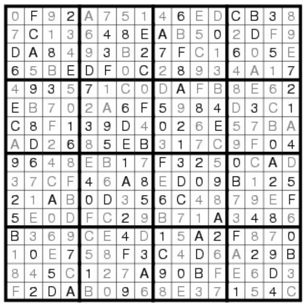
3x3x3

Sudoku

	F		2						6			C	B	3	
	C				4	8	E	A			0		D		
D	A	8			3		2	7	F			6		5	
6			E	D	F		C		8						7
	9	3		7					A						2
E						6	F	5		8	4		3		1
C	8		1	3	9	D		0	2		E				
	D		6		5	E	B		1					0	4
9	6					1		F	3	2		0		A	
				4		A	8		D	0	9	B		2	5
2		A		0	D		5	6	C						F
5						2					A		4	8	
B						4		1		A	2	F			0
	0		7			F	3	C		D			2	9	B
		5		1			A	9	0	B				D	
	2	D	A			9						1		4	

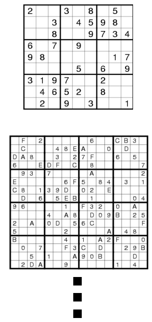
4x4x4

Sudoku



4x4x4

Sudoku



n x n x n

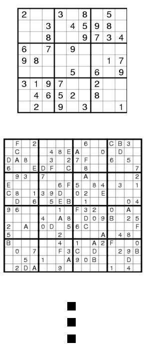
Suppose you have an algorithm S(n) to solve n x n x n, with running time T(n).

V(n): time to verify the solution.

Fact: V(n) $\in \Theta(n^2 \times n^2)$

Question: is there some constant such that T(n) $\in O(n^{constant})$?

Sudoku



n x n x n

P vs NP problem

=

Does there exist an algorithm for solving n x n x n Sudoku that runs in time p(n) for some polynomial p( ) ?

The P versus NP problem (informally)

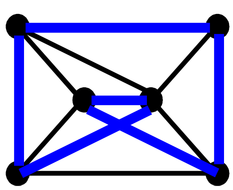
Is **finding** an answer to a problem *much* more difficult than **verifying** an answer to a problem?

Hamiltonian Cycle

Given a graph G = (V,E), is there a cycle that visits all the nodes exactly once?

YES if G has a Hamiltonian cycle

NO if G has no Hamiltonian cycle



The Set "HAM"

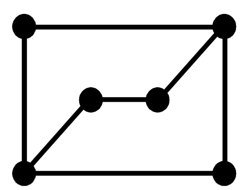
HAM = { graph G | G has a Hamiltonian cycle }

Independent Set

Given a graph G = (V,E), and an integer k, is there a a subset of V with at least k vertices such that no two of them are adjacent?

YES if G has an independent set of size k.

NO if G has no independent set of size k.



The Set "INDEP-SET$_k$"

INDEP-SET$_k$ = { graph G | G has an independent set of size k}

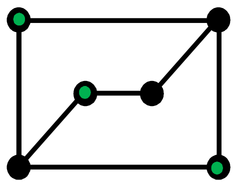
Independent Set

Given a graph G = (V,E), and an integer k, is there a a subset of V with at least k vertices such that no two of them are adjacent?

YES if G has an independent set of size k.

NO if G has no independent set of size k.

Yes for k =3; no for k = 4



The Set "INDEP-SET$_k$"

INDEP-SET$_k$ = { graph G | G has an independent set of size k}

3-Satisfiability

Given a Boolean formula in Conjunctive Normal Form, having at most 3 literals per clause, is it satisfiable?

YES if there exists an assignment of truth values to each variable xi such that the overall formula is true.

NO otherwise

$$(x_0 \vee \neg x_1 \vee x_3) \wedge (\neg x_0 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_0 \vee x_3 \vee \neg x_4)$$

This assignment works: $x_0 = T$; $x_1 = F$; $x_2 = T$; $x_3 = F$; $x_4 = F$;

The Set "3SAT"

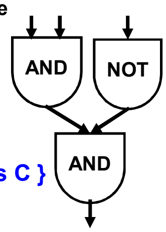
3SAT= { 3CNF formulas F | F is satisfiable}

Circuit-Satisfiability

Input: A circuit C with one output

Output: YES if C is satisfiable

NO if C is not satisfiable



The Set "Circuit-SAT"

Circuit-SAT = { all satisfiable circuits C }

Sudoku

Input: *n* x *n* x *n* sudoku instance

Output: YES if this sudoku has a solution

NO if it does not

The Set "SUDOKU"

SUDOKU = { All solvable Sudoku instances }

Polynomial Time and The Class "P"

What is an efficient algorithm?

Is an O(n) algorithm efficient?

How about O(n log n)?

$O(n^2)$?

$O(n^{10})$?

polynomial time

$O(n^c)$ for some constant c

$O(n^{\log n})$?

$O(2^n)$?

O(n!) ?

non-polynomial time

What is an efficient algorithm?

Does an algorithm running in O(n^{100}) time count as efficient?

Asking for a poly-time algorithm for a problem sets a (very) low bar when asking for efficient algorithms.

We consider non-polynomial time algorithms to be inefficient.

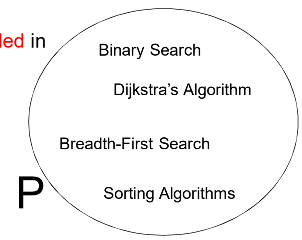
And hence a necessary condition for an algorithm to be efficient is that it should run in poly-time.

The Class P

The class of all sets that can be verified in polynomial time.

AND

The class of all decision problems that can be decided in polynomial time.



The question is: can we achieve even this for

HAM?
Circuit-SAT?
Sudoku?

Onto the new class, NP

(Nondeterministic Polynomial Time)

Verifying Membership

Is there a short "proof" I can give you to verify that:

G $\in$ HAM?
G $\in$ Sudoku?
G $\in$ Circuit-SAT?

Yes: I can just give you the cycle, solution, circuit

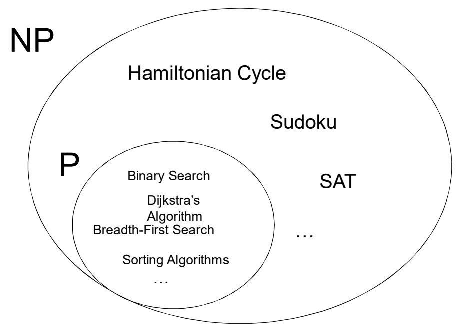
The Class NP

The class of sets for which there exist "short" proofs of membership
(of polynomial length)
that can "quickly" verified
(in polynomial time).

Recall: The algorithm doesn't have to find the proof; it just needs to be able to verify that it is a "correct" proof.

Fact: P $\subseteq$ NP

$P \subseteq NP$



Summary: P versus NP

NP: "proof of membership" in a set can be verified in polynomial time.

P: in NP (membership verified in polynomial time)
AND **membership in a set can be decided in polynomial time.**

Fact: $P \subseteq NP$

Question: Does $NP \subseteq P$?
i.e., *Does P = NP?*
People generally believe $P \neq NP$, but no proof yet

Why Care?

NP Contains Lots of Problems We Don't Know to be in P

Classroom Scheduling
Packing objects into bins
Scheduling jobs on machines
Finding cheap tours visiting a subset of cities
Finding good packet routings in networks
Decryption
…

OK, OK, I care...

How could we prove that NP = P?

We would have to show that every set in NP has a polynomial time algorithm…

How do I do that?
It may take a long time!
Also, what if I forgot one of the sets in NP?

How could we prove that NP = P?

We can describe just one problem L in NP, such that if this problem L is in P, then $NP \subseteq P$.

It is a problem that can capture all other problems in NP.

The "Hardest" Set in NP

We call these problems NP-complete

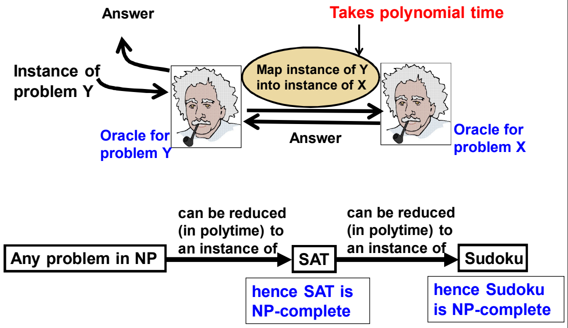
Theorem (Cook/Levin)

Circuit-SAT is one problem in NP, such that if we can show Circuit-SAT is in P, then we have shown NP = P.

Circuit-SAT is a problem in NP that can capture all other languages in NP.

We say SAT is NP-complete.

Poly-time reducible to each other



NP-complete: The "Hardest" problems in NP

Sudoku **Clique** **3SAT**

Circuit-SAT **Independent-Set**

3-Colorability **HAM**

These problems are all "polynomial-time equivalent" i.e., each of these can be reduced to any of the others in polynomial time

If you get a polynomial-time algorithm for one, you get a polynomial-time algorithm for ALL.
(you get millions of dollars, you solve decryption, … etc.)