CSE373: Data Structures & Algorithms

Lecture 24: The P vs. NP question, NP-Completeness

Nicki Dell

Spring 2014

Admin

- Homework 5 due TONIGHT at 11pm!

- Homework 6 is posted
  - Due one week from today, June 4^{th} at 11pm
  - No partners

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	

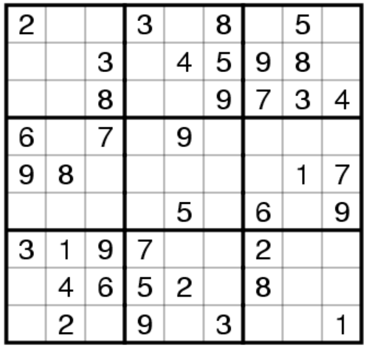
4x4x4

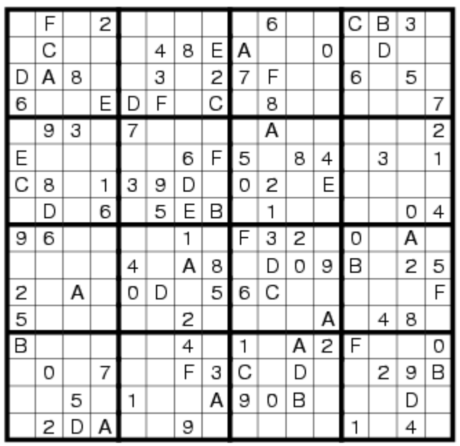
Sudoku

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

4x4x4

Sudoku





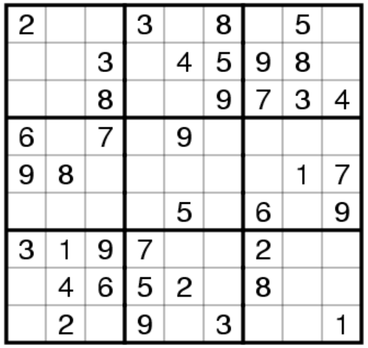
n x n x n

Suppose you have an algorithm S(n) to solve n x n x n

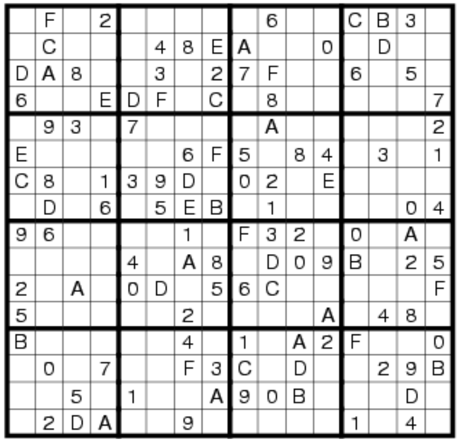
V(n) time to verify the solution

Fact: $V(n) = O(n^2 \times n^2)$

Question: is there some constant such that $S(n) = O(n^{constant})$?



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)

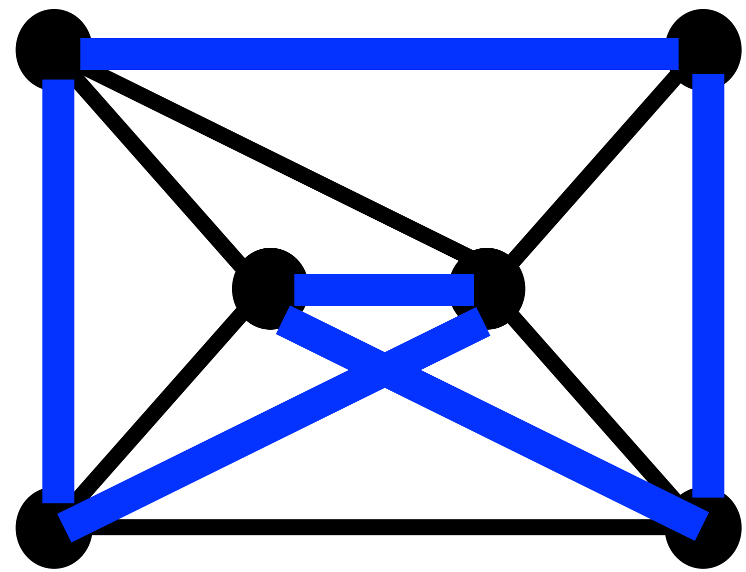
Is finding an answer to a problem *much* more difficult than verifying an answer to a problem?

Hamilton Cycle

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

YES if G has a Hamilton cycle

NO if G has no Hamilton cycle



The Set "HAM"

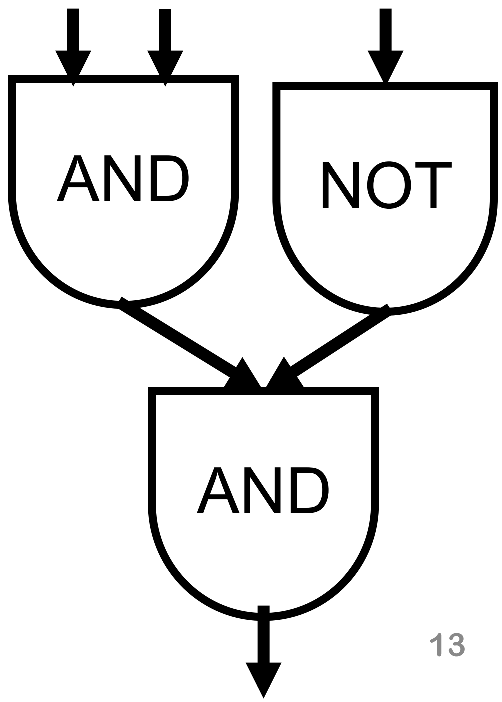
HAM = { graph G | G has a Hamilton cycle }

Circuit-Satisfiability

Input: A circuit C with one output

Output: YES if C is satisfiable

NO if C is not satisfiable



The Set "SAT"

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 $O(n \log n)$?

$O(n^2)$?

$O(n^{10})$?

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

$O(2^n)$?

$O(n!)$?

polynomial time

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

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.

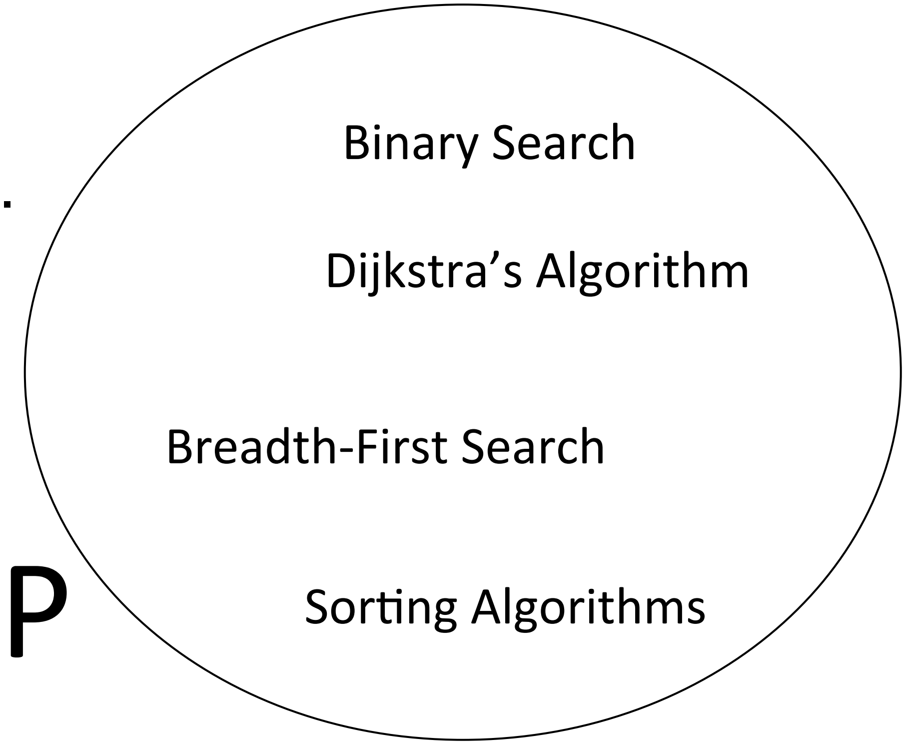
And hence a necessary condition for an algorithm to be efficient is that it should run in poly-time.

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?

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?
$G \in$ SAT?

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

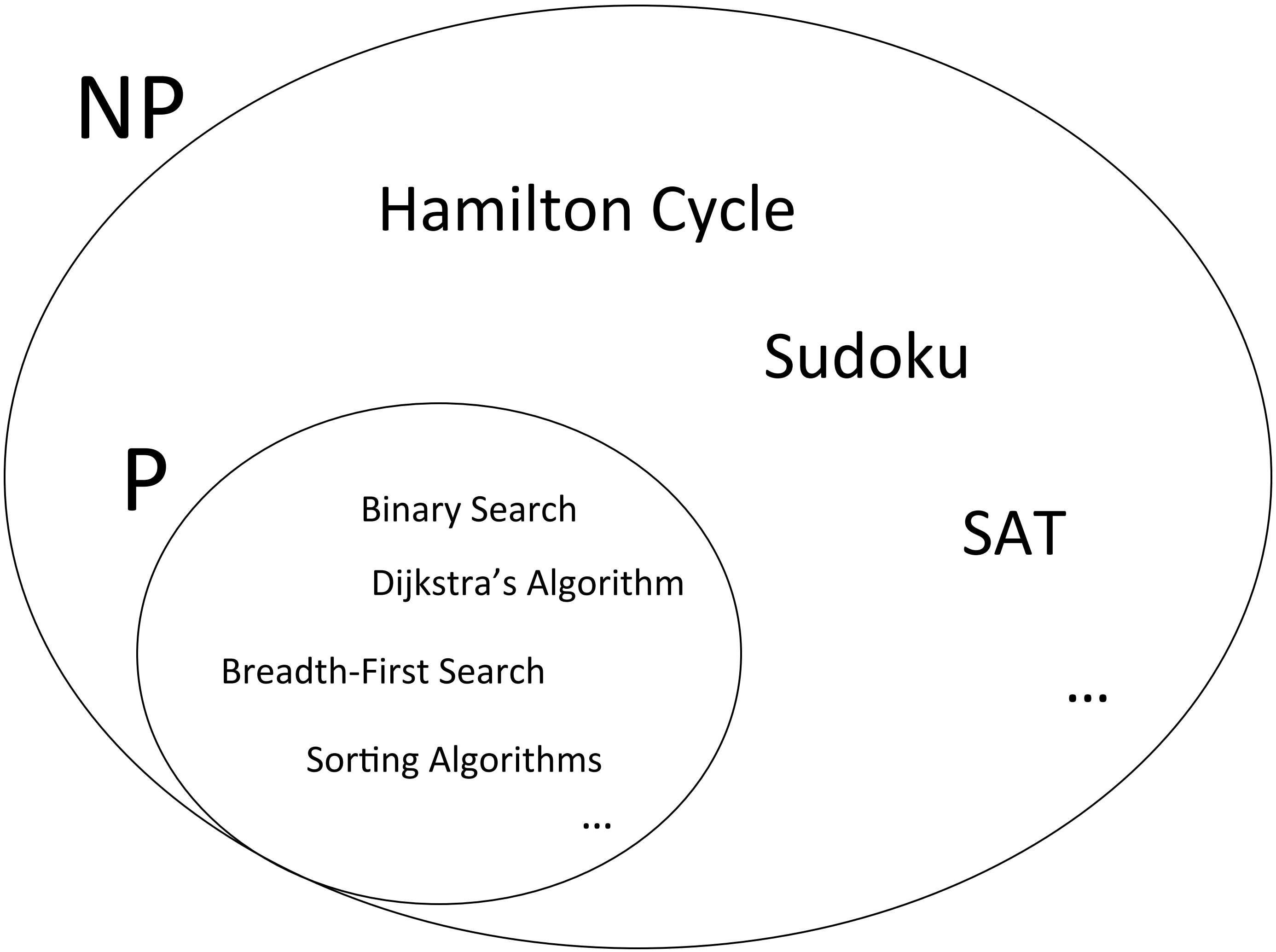
The Class NP

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

Fact: P $\subseteq$ NP

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

P ⊆ NP


NP
Hamilton Cycle
Sudoku
SAT
...
P
Binary Search
Dijkstra's Algorithm
Breadth-First Search
Sorting Algorithms
...

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

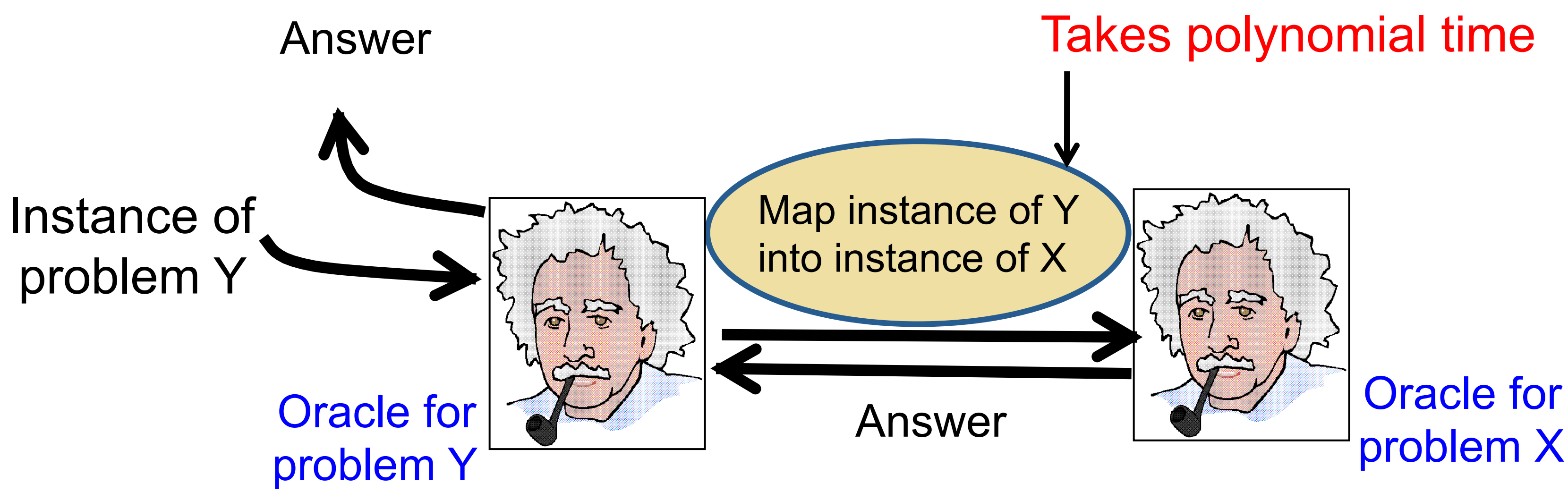
Theorem [Cook/Levin]

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

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



Any problem in NP → can be reduced (in polytime to) an instance of → SAT → can be reduced (in polytime to) an instance of → Sudoku

hence SAT is NP-complete

hence Sudoku is NP-complete

NP-complete: The "Hardest" problems in NP

Sudoku

Clique

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.)