The P versus NP problem
Is perhaps the biggest open problem in computer science (and mathematics!) today.
(Even featured in the TV show NUMB3RS)

But what is the P-NP problem?
Sudoku

3 x 3 x 3

4 x 4 x 4

Sudoku

Suppose it takes you $S(n)$ to solve $n \times n \times n$

$V(n)$ time to verify the solution

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

Question: is there some constant $c$ such that $S(n) \leq n^c$?

P vs NP problem

Does there exist an algorithm for $n \times n \times n$ Sudoku that runs in time $p(n)$ for some polynomial $p(\ )$?

The P versus NP problem (informally)

Is proving a theorem much more difficult than checking the proof of a theorem?
Let’s start at the beginning…

**Hamilton Cycle**
Given a graph $G = (V,E)$, a cycle that visits all the nodes exactly once

**The Problem “HAM”**
Input: Graph $G = (V,E)$
Output: YES if $G$ has a Hamilton cycle
NO if $G$ has no Hamilton cycle

**The Set “HAM”**
$HAM = \{ \text{graph } G \mid G \text{ has a Hamilton cycle} \}$

**Circuit-Satisfiability**
Input: A circuit $C$ with one output
Output: YES if $C$ is satisfiable
NO if $C$ is not satisfiable

**The Set “SAT”**
$SAT = \{ \text{all satisfiable circuits } C \}$

**Bipartite Matching**
Input: A bipartite graph $G = (U,V,E)$
Output: YES if $G$ has a perfect matching
NO if $G$ does not
The Set “BI-MATCH”

BI-MATCH = \{ all bipartite graphs that have a perfect matching \}

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 \}

Decision Versus Search Problems

Decision Problem
YES/NO answers
Does G have a Hamilton cycle?
Can G be 3-colored?

Search Problem
Find a Hamilton cycle in G if one exists, else return NO
Find a 3-coloring of G if one exists, else return NO

Reducing Search to Decision

Given an algorithm for decision Sudoku, devise an algorithm to find a solution

Idea:
Fill in one-by-one and use decision algorithm

Reducing Search to Decision

Given an algorithm for decision HAM, devise an algorithm to find a solution

Idea:
Find the edges of the cycle one by one

Decision/Search Problems

We’ll study decision problems because they are almost the same (asymptotically) as their search counterparts
Polynomial Time and The Class “P” of Decision Problems

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
non-polynomial time

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

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

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.

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

The question is: can we achieve even this for 3-coloring?
- SAT?
- Sudoku?
- HAM?

The Class P

We say a set $L \subseteq \Sigma^*$ is in P if there is a program $A$ and a polynomial $p(\cdot)$ such that for any $x$ in $\Sigma^*$, $A(x)$ runs for at most $p(|x|)$ time and answers question “is $x$ in $L$?” correctly.

The class of all sets $L$ that can be recognized in polynomial time.

The class of all decision problems that can be decided in polynomial time.
Why are we looking only at sets $\subseteq \Sigma^*$?
What if we want to work with graphs or boolean formulas?

Languages/Functions in P?
Example 1:
$\text{CONN} = \{(\text{graph } G: G \text{ is a connected graph})\}$
Algorithm $A_1$:
If $G$ has $n$ nodes, then run depth first search from any node, and count number of distinct nodes you see. If you see $n$ nodes, $G \in \text{CONN}$, else not.
Time: $p_1(|x|) = \Theta(|x|)$.

Languages/Functions in P?
HAM, SUDOKU, SAT are not known to be in P
$\text{CO-HAM} = \{G \mid G \text{ does NOT have a Hamilton cycle}\}$
$\text{CO-HAM} \in P$ if and only if $\text{HAM} \in P$

Onto the new class, NP

Verifying Membership
Is there a short “proof” I can give you for:
$G \in \text{HAM}?$
$G \in \text{BI-MATCH}?$
$C \in \text{SAT}?$
$G \in \text{CO-HAM}?$

NP
A set $L \in \text{NP}$
if there exists an algorithm $A$ and a polynomial $p(\cdot)$
For all $x \in L$
there exists $y$ with $|y| \leq p(|x|)$
such that $A(x,y) = \text{YES}$
in $p(|x|)$ time
For all $x' \not\in L$
For all $y'$ with $|y'| \leq p(|x'|)$
we have $A(x',y') = \text{NO}$
in $p(|x|)$ time
Recall the Class P

We say a set $L \subseteq \Sigma^*$ is in $P$ if there is a program $A$ and a polynomial $p(\cdot)$ such that for any $x$ in $\Sigma^*$,

- $A(x)$ runs for at most $p(|x|)$ time
- and answers question “is $x$ in $L$?” correctly.

We can think of $A$ as “proving” that $x$ is in $L$.

<table>
<thead>
<tr>
<th>$L \in P$</th>
<th>$L \in NP$</th>
</tr>
</thead>
<tbody>
<tr>
<td>if there exists an algorithm $A$ and a polynomial $p(\cdot)$</td>
<td></td>
</tr>
<tr>
<td>$A(x)$ runs for at most $p(</td>
<td>x</td>
</tr>
<tr>
<td>and answers question “is $x$ in $L$?” correctly.</td>
<td></td>
</tr>
<tr>
<td>For all $x \in L$</td>
<td></td>
</tr>
<tr>
<td>there exists a $y$ with $</td>
<td>y</td>
</tr>
<tr>
<td>such that $A(x,y) = YES$</td>
<td></td>
</tr>
<tr>
<td>in $p(</td>
<td>x</td>
</tr>
<tr>
<td>For all $x' \notin L$</td>
<td></td>
</tr>
<tr>
<td>For all $y'$ with $</td>
<td>y'</td>
</tr>
<tr>
<td>Such that $A(x',y') = NO$</td>
<td></td>
</tr>
<tr>
<td>in $p(</td>
<td>x</td>
</tr>
</tbody>
</table>

Example: $HAM \in NP$

Let $A(x,y)$ be a program that takes two strings $x$ and $y$, and returns YES if the following conditions hold otherwise it returns NO.

- $y$ is a representation of a labeled graph
- $x$ is a representation of a cycle with the same labeled vertices as $y$
- every edge of the cycle $x$ is in the graph $y$

(All of these conditions can be easily checked in linear time)

By our definition, this proves $HAM \in NP$.

The Class NP

The class of sets $L$ for which there exist “short” proofs of membership (of polynomial length) that can be “quickly” verified (in polynomial time).

Recall: $A$ doesn’t have to find these proofs $y$; it just needs to be able to verify that $y$ is a “correct” proof.

<table>
<thead>
<tr>
<th>$P \subseteq NP$</th>
</tr>
</thead>
<tbody>
<tr>
<td>For any $L$ in $P$, we can just take $y$ to be the empty string and satisfy the requirements.</td>
</tr>
<tr>
<td>Hence, every language in $P$ is also in $NP$.</td>
</tr>
</tbody>
</table>

Languages/Functions in NP?

- $G \in HAM$? (Yes, already saw)
- $G \in BI-MATCH$? (is in $P$)
- $G \in SAT$? (Yes, explain it)
- $G \in CO-HAM$? (not clear)

Proof that something is in $NP$ is often trivial.
Summary: P versus NP

Set L is in P if membership in L can be decided in poly-time.

Set L is in NP if each x in L has a short “proof of membership” that can be verified in poly-time.

Fact: \( P \subseteq NP \)

Question: Does \( NP \subseteq P \) ?

Why Care?

Classroom Scheduling
 Packing objects into bins
 Scheduling jobs on machines
 Finding cheap tours visiting a subset of cities
 Allocating variables to registers
 Finding good packet routings in networks
 Decryption
 ...

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

OK, OK, I care...

But where do I begin if I want to reason about the \( P=NP \) problem?

How can we prove that \( NP \subseteq P \)?

I 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?

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” Sets in NP

Sudoku

Sudoku has a polynomial time algorithm if and only if \( P = NP \)

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

“Poly-time reducible to each other”

Reducing problem Y to problem X in poly-time

F is poly-time computable

Oracle for problem Y

Oracle for problem X

Theorem [Cook/Levin]:

SAT is one language in NP, such that if we can show SAT is in P, then we have shown \( NP \subseteq P \).

SAT is a language in NP that can capture all other languages in NP.

We say SAT is NP-complete.
Last lecture...

SAT and 3COLOR: Two problems that seem quite different, but are substantially the same.

Also substantially the same as CLIQUE and INDEPENDENT SET.

If you get a polynomial-time algorithm for one, you get a polynomial-time algorithm for ALL.

Any language in NP

SAT

3COLOR

Definition of P and NP
Definition of problems
SAT, 3-COLOR, HAM, SUDOKU, BI-MATCH
SAT, 3-COLOR, HAM, SUDOKU all essentially equivalent.

Solve any one in poly-time
⇒ solve all of them in poly-time

Here’s What You Need to Know...

Definition of P and NP
Definition of problems
SAT, 3-COLOR, HAM, SUDOKU, BI-MATCH
SAT, 3-COLOR, HAM, SUDOKU all essentially equivalent.

Solve any one in poly-time
⇒ solve all of them in poly-time