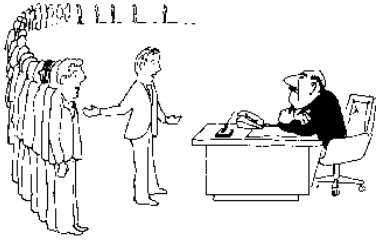


**15-251**  
**Great Ideas in**  
**Theoretical Computer Science**

Lecture 16:  
NP and NP-completeness I



March 8th, 2018

I can't find an efficient algorithm, but neither can all these famous people.

---

---

---

---

---

---

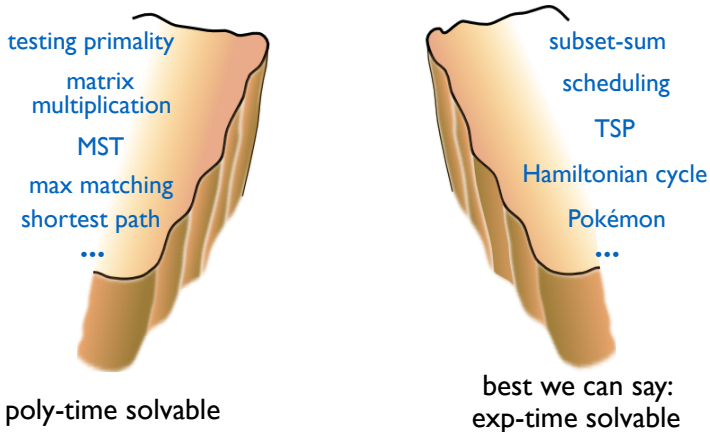
---

---

---

---

**The chasm between poly-time and exp-time.**



---

---

---

---

---

---

---

---

---

---

**Exponential running time examples**

**Subset Sum Problem**

**Theorem Proving Problem**

**Traveling Salesperson Problem (TSP)**

**Satisfiability Problem (SAT)**

**Circuit Satisfiability Problem (Circuit-SAT)**

**Sudoku Problem**

---

---

---

---

---

---

---

---

---

---

In our quest to understand efficient computation,  
we come across:

**P vs NP problem**

Biggest open problem in all of Computer Science.  
One of the biggest open problems in all of Mathematics.

---

---

---

---

---

---

---

---

---

---

So what is the **P vs NP** question?

The **P vs NP** question is the following:

---

---

---

---

---

---

---

---

---

---

An important goal for a computer scientist

Identifying and dealing with intractable problems

After decades of research and billions of dollars of funding,  
**no** poly-time algs for:

*Subset Sum, SAT, Theorem Proving, TSP, Sudoku, ...*

Can we prove there is no poly-time alg?

---

---

---

---

---

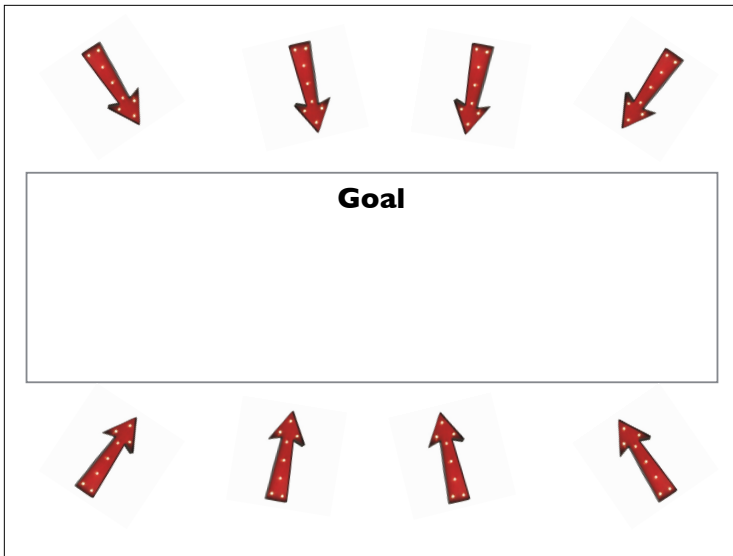
---

---

---

---

---



---

---

---

---

---

---

---

---

---

---

**Revisiting reductions**

A central concept for comparing the “difficulty” of problems.

↓  
differs based on context

Right now we are interested in **poly-time decidability vs not poly-time decidability**

**Want to define:**  $A \leq B$  ( $B$  is at least as hard as  $A$  w.r.t. poly-time decidability.)

---

---

---

---

---

---

---

---

---

---

**Revisiting reductions**

---

---

---

---

---

---

---

---

---

---

## Revisiting reductions

### Example

**A:**

Given a graph and an integer  $k$ , does there exist at least  $k$  pairs of vertices connected to each other?  
(by a path)

**B:**

Given a graph and a pair of vertices  $(s,t)$ ,  
is  $s$  and  $t$  connected?

---

---

---

---

---

---

---

---

---

---

## Revisiting reductions

### The 2 sides of reductions

1. Expand the landscape of **tractable** problems.

---

---

---

---

---

---

---

---

---

---

## Revisiting reductions

### The 2 sides of reductions

2. Expand the landscape of **intractable** problems.

---

---

---

---

---

---

---

---

---

---

## Gathering evidence for intractability

including some that we think should not be in **P**



If we can show  $L \leq_T^P A$  for **many**  $L$

then that would be good evidence that  $A \notin \mathbf{P}$ .

---

---

---

---

---

---

---

---

---

---

## Definition of **C**-hard

---

---

---

---

---

---

---

---

---

---

## Definition of **C**-complete

---

---

---

---

---

---

---

---

---

---

## Definitions of **C**-hard and **C**-complete

### **Observation:**

Suppose  $A$  is **C**-complete.

---

**2 possible worlds**

## Recall the goal

Good evidence for  $A \notin \mathbf{P}$ :

- $A$  is **C**-complete for a really rich/large set **C**  
(a set **C** such that we believe  $\mathbf{C} \neq \mathbf{P}$ )

So what is a good choice for **C**?

(if we want to show *SAT*, *Theorem Proving*, *TSP*, ... are **C**-complete?)



**Main Goal Reduces to:**



## Finding the right complexity class **C**

**Try 1:**

**Try 2:**

---

---

---

---

---

---

---

---

---

---

## A complexity class for BFS?

What would be a reasonable definition for:  
“class of problems decidable using BFS” ?

What is common about  
*SAT, Theorem Proving, TSP, Sudoku, etc...?*

---

---

---

---

---

---

---

---

---

---

## The complexity class **NP**

**Informally:**

---

---

---

---

---

---

---

---

---

---

## Poll: Test your intuition

Which of these are in **NP**?

- Subset Sum
- TSP
- SAT
- Circuit-SAT
- Sudoku
- HALTS
- $\{0^k 1^k : k \in \mathbb{N}\}$

---

---

---

---

---

---

---

---

---

---

## Formal definition of **NP**

---

---

---

---

---

---

---

---

---

---

## Examples of languages in **NP**

### CLIQUE

**Input:**  $\langle G, c \rangle$  where  $G$  is a graph and  $c$  is a positive int.

**Output:** Yes iff  $G$  contains a clique of size  $c$ .

**Fact:** CLIQUE is in **NP**.

---

---

---

---

---

---

---

---

---

---



## Examples of languages in **NP**

**Proof:** We need to show a verifier TM  $V$  exists as specified in the definition of **NP**.

**def**  $V(x, u)$  :

## Examples of languages in **NP**

**Proof (continued):**

Need to show:

- 1.
- 2.
- 3.

## Examples of languages in **NP**

**Proof (continued):**

Need to show:

1. if  $x \in \text{CLIQUE}$ , there is some proof  $u$  (of poly-length) that makes  $V$  **ACCEPT**.

## Examples of languages in **NP**

### **Proof (continued):**

Need to show:

2. if  $x \notin \text{CLIQUE}$ , no matter what  $u$  is,  $V$  **REJECTS**.



---

---

---

---

---

---

---

---

---

---

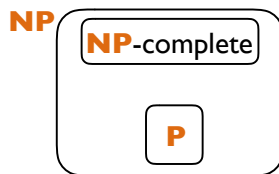
## The complexity class **NP**

### **2 Observations:**

1. Every decision problem in **NP** can be solved using BFS.

2. This is a big class!

Contains everything in **P**.



People expect **NP** contains much more than **P**.

---

---

---

---

---

---

---

---

---

---

## Coming back to our main goal

Could it be true that one of  
*SAT, Theorem Proving, TSP, Sudoku, etc.*  
is **NP**-complete?

---

---

---

---

---

---

---

---

---

---

Is there **any** language that is **NP**-complete??

## The Cook-Levin Theorem



### Theorem (Cook 1971 - Levin 1973):

---

---

---

---

---

---

---

---

---

---

## Karp's 21 NP-complete problems

### 1972: "Reducibility Among Combinatorial Problems"

0-1 Integer Programming

Clique

Set Packing

Vertex Cover

Set Covering

Feedback Node Set

Feedback Arc Set

Directed Hamiltonian Cycle

Undirected Hamiltonian Cycle

3SAT

Partition

Clique Cover

Exact Cover

Hitting Set

Knapsack

Steiner Tree

3-Dimensional Matching

Job Sequencing

Max Cut

Chromatic Number



---

---

---

---

---

---

---

---

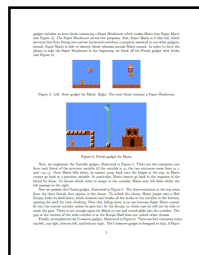
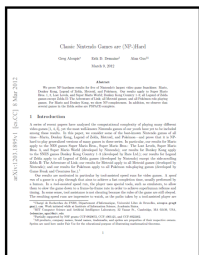
---

---

## Some other "interesting" examples

### Super Mario Bros

Given a Super Mario Bros level, is it completable?



### Tetris

Given a sequence of Tetris pieces, and a number  $k$ , can you clear more than  $k$  lines?

---

---

---

---

---

---

---

---

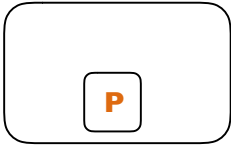
---

---

## How do you show a language is **NP**-complete?

How did Cook and Levin do it ??!

**NP**



$\leq_T^P$  SAT

How did Karp do it ??!

**IMPORTANT NOTE:**

---

---

---

---

---

---

---

---

---

---

## How do you show a language is **NP**-complete?

It is similar to showing undecidability.

- need an initial direct proof that a language is **NP**-hard. (Cook-Levin Theorem)
- to show other languages are **NP**-hard, use poly-time reductions.

**These are the topics of next 2 lectures.**

---

---

---

---

---

---

---

---

---

---

**The P vs NP Question**

---

---

---

---

---

---

---

---

---

---

## Good evidence for intractability?

If  $A$  is **NP**-hard,  
that seems to be good evidence that  $A \notin \mathbf{P} \dots$

if you believe  $\mathbf{P} \neq \mathbf{NP}$

But is  $\mathbf{P} \neq \mathbf{NP}???$

---

---

---

---

---

---

---

---

---

---

## The two possible worlds



---

---

---

---

---

---

---

---

---

---

## What do experts think?

Two polls from **2002** and **2012**

# respondents in **2002**: 100

# respondents in **2012**: 152

	$\mathbf{P} \neq \mathbf{NP}$	$\mathbf{P} = \mathbf{NP}$	Ind	DC	DK
2002	61(61%)	9(9%)	4(4%)	1(1%)	22(22%)
2012	126 (83%)	12 (9%)	5 (3%)	5 (3%)	1(0.6%)

---

---

---

---

---

---

---

---

---

---

What does **NP** stand for anyway?

---

---

---

---

---

---

---

---

---

---

Next 2 Lectures

How did Cook-Levin show SAT is **NP**-complete?

How do you show other problems are **NP**-complete?

---

---

---

---

---

---

---

---

---

---