The chasm between poly-time and exp-time.

The best we can say:
exp-time solvable

Exponential running time examples

Subset Sum Problem
Theorem Proving Problem
Traveling Salesperon 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.)
Given a graph and a pair of vertices (s,t), are s and t connected?

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), are s and t connected?

The 2 sides of reductions

1. Expand the landscape of tractable problems.

The 2 sides of reductions

2. Expand the landscape of intractable problems.
Gathering evidence for intractability

If we can show \( L \leq_p A \) for many \( L \)
then that would be good evidence that \( A \notin P \).

Definition of \( C \)-hard

Definition of \( C \)-complete
Definitions of $\mathcal{C}$-hard and $\mathcal{C}$-complete

**Observation:**
Suppose $A$ is $\mathcal{C}$-complete.

---

2 possible worlds

---

Recall the goal

Good evidence for $A \not\in \mathcal{P}$:
- $A$ is $\mathcal{C}$-complete for a really rich/large set $\mathcal{C}$
  (a set $\mathcal{C}$ such that we believe $\mathcal{C} \neq \mathcal{P}$)

So what is a good choice for $\mathcal{C}$?
(if we want to show SAT, Theorem Proving, TSP, ... are $\mathcal{C}$-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 $\text{NP}$?

- Subset Sum
- TSP
- SAT
- Circuit-SAT
- Sudoku
- HALTS
- $\{0^k1^k : k \in \mathbb{N}\}$

Formal definition of $\text{NP}$

Examples of languages in $\text{NP}$

CLIQUE

**Input:** $(G, c)$ 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 $\text{NP}$.
### Examples of languages in NP

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

```python
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 $x \in$ 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**.

**NP**

**NP-complete**

**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 comletable?

**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 \textbf{NP}-complete?

How did Cook and Levin do it?!?

\[
\begin{array}{c}
\text{NP} \\
\leq^P_T \text{SAT}
\end{array}
\]

How did Karp do it?!?

**IMPORTANT NOTE:**

---

How do you show a language is \textbf{NP}-complete?

It is similar to showing undecidability.

- need an initial direct proof that a language is \textbf{NP}-hard. (Cook-Levin Theorem)

- to show other languages are \textbf{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 \( \text{NP} \)-hard, that seems to be good evidence that \( A \not\in \text{P} \) …

if you believe \( \text{P} \neq \text{NP} \)

But is \( \text{P} \neq \text{NP} \)?

The two possible worlds

What do experts think?

Two polls from 2002 and 2012

\# respondents in 2002: 100
\# respondents in 2012: 152

<table>
<thead>
<tr>
<th></th>
<th>( \text{P} \neq \text{NP} )</th>
<th>( \text{P} = \text{NP} )</th>
<th>Ind</th>
<th>DC</th>
<th>DK</th>
</tr>
</thead>
<tbody>
<tr>
<td>2002</td>
<td>61 (61%)</td>
<td>9 (9%)</td>
<td>4 (4%)</td>
<td>1 (1%)</td>
<td>22 (22%)</td>
</tr>
<tr>
<td>2012</td>
<td>126 (83%)</td>
<td>12 (9%)</td>
<td>5 (3%)</td>
<td>5 (3%)</td>
<td>1 (0.6%)</td>
</tr>
</tbody>
</table>
What does **NP** stand for anyway?