You come across a problem you cannot solve.
  What do you do?
    Could it be in P?

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

Is there a deep reason why the problem seems to be hard?
Summary so far

- How do you identify *intractable* problems? (problems not in \( \text{P} \)) e.g. SAT, TSP, Subset-Sum, …

- Poly-time reductions \( A \leq^p_T B \) are useful to compare hardness of problems.

- Evidence for intractability of \( A \):
  Show \( L \leq^p_T A \), for all \( L \in \mathbf{C} \), for a large class \( \mathbf{C} \).

\[
\begin{array}{c}
\mathbf{C} \\
\leq^p_A \\
A \text{ is } \mathbf{C}\text{-hard}
\end{array}
\]

Summary so far

\[
\begin{array}{c}
\mathbf{C} \\
A \\
\leq^p_A \\
A \text{ is } \mathbf{C}\text{-complete}
\end{array}
\]

\( \mathbf{C} = \mathbf{P} \iff A \in \mathbf{P} \)

Summary so far

2 possible worlds

\[
\begin{array}{c|c}
\mathbf{C} \\
\mathbf{C}\text{-complete} \\
\mathbf{P} \\
\mathbf{C} = \mathbf{C}\text{-complete} \\
\mathbf{C} = \mathbf{P}
\end{array}
\]
• The complexity class **NP** (take **C = NP**)

---

**Summary so far**

Which languages \( L \) are in **NP**?

1. Every \( x \) in \( L \) has
   (at most) exponentially large “possible solutions space”

2. Easy (poly-time) to verify whether a possible solution is indeed a solution or not.

---

**Summary so far**

• **NP**-hardness, **NP**-completeness

• Cook-Levin Theorem: CIRCUIT-SAT is **NP**-complete

\[
\begin{array}{c}
\text{NP} \\
\xrightarrow{\leq_T} \\
\text{P} \\
\end{array}
\]

CIRCUIT-SAT

• Many other languages are **NP**-complete.

• The **P vs NP** question
Every L in \textbf{NP}

\textbf{Cook-Levin Theorem}

\begin{itemize}
  \item CIRCUIT-SAT
  \item 3SAT
  \item 3COL
  \item SUBSET-SUM
  \item CLIQUE
  \item VERTEX-COVER
  \item IS
  \item HAMILTONIAN-CYCLE
  \item TSP
\end{itemize}

\textbf{First: An important note about reductions}

\textbf{Cook reduction}

\textbf{Cook reductions}: poly-time Turing reductions

\[ A \leq_T^P B \]

\begin{itemize}
  \item \textbf{Yes} or \textbf{No}
  \item \textbf{Yes} or \textbf{No}
\end{itemize}

“\textbf{You can solve }A\textbf{ in poly-time using a blackbox that solves }B\textbf{.}”

\textbf{You can call the blackbox poly(|x|) times.}
**Karp reduction**

NP-hardness is usually defined using Karp reductions. 

**Karp reduction** (polynomial-time many-one reduction):

\[ A \leq_{m}^{P} B \]

Make one call to \( M_B \) and directly use its answer as output. 
We must have:

**Karp reduction picture**

**Karp reduction: Example**

**CLIQUE**

**Input:** \((G, k)\) where \(G\) is a graph and \(k\) is a positive int. 

**Output:** Yes iff \(G\) contains a clique of size \(k\).

**INDEPENDENT-SET (IS)**

**Input:** \((G, k)\) where \(G\) is a graph and \(k\) is a positive int. 

**Output:** Yes iff \(G\) contains an independent set of size \(k\).

**Fact:** CLIQUE \( \leq_{m}^{P} \) IS.
Want:
\( (G, k) \overset{f}{\rightarrow} (G', k') \)

\( G \) has a clique of size \( k \) \iff \( G' \) has an ind. set of size \( k' \)

\( G \)

\( G' \)

**Proof:**

1. Define a map \( f: \Sigma^* \rightarrow \Sigma^*. \)
2. Show \( w \in \text{CLIQUE} \implies f(w) \in \text{IS} \)
3. Show \( w \notin \text{CLIQUE} \implies f(w) \notin \text{IS} \)
   (often easier to argue the contrapositive)
4. Argue \( f \) is computable in polynomial time.

**Proof (continued):**

1. Define a map \( f: \Sigma^* \rightarrow \Sigma^*. \)

\[ \text{def } f(w) : \]
Karp reduction: Example

Proof (continued):
2. Show $w \in \text{CLIQUE} \implies f(w) \in \text{IS}$

---

Karp reduction: Example

Proof (continued):
3. Show $w \notin \text{CLIQUE} \implies f(w) \notin \text{IS}$
   
   (Show the contrapositive.)

---

Karp reduction: Example

Proof (continued):
4. Argue $f$ is computable in polynomial time.

- checking if the input is a valid encoding can be done in polynomial time.
  (for any reasonable encoding scheme)

- creating $E^*$, and therefore $G^*$, can be done in polynomial time.
Can define \textbf{NP}-hardness with respect to $\leq_P^T$.  
(what some courses use for simplicity)

Can define \textbf{NP}-hardness with respect to $\leq_P^m$.  
(what experts use)

These lead to different notions of \textbf{NP}-hardness.

---

**Poll**

Which of the following are true?

- 3\text{COL} \leq_P^m 2\text{COL} \text{ is known to be true.}
- 3\text{COL} \leq_P^m 2\text{COL} \text{ is known to be false.}
- 3\text{COL} \leq_P^m 2\text{COL} \text{ is open.}
- 2\text{COL} \leq_P^m 3\text{COL} \text{ is known to be true.}
- 2\text{COL} \leq_P^m 3\text{COL} \text{ is known to be false.}
- 2\text{COL} \leq_P^m 3\text{COL} \text{ is open.}
- if $A \leq_P^m B$ and $B \in \text{NP}$, then $A \in \text{NP}$.

---

**\text{CLIQUE} is \textbf{NP}-complete**
Want to show:

- CLIQUE is in **NP**.
- CLIQUE is **NP**-hard.
  
  3SAT is **NP**-hard, so show \( 3SAT \leq_P CLIQUE \).

---

### Definition of 3SAT

**3SAT**

**Input**: A Boolean formula in “conjunctive normal form” in which every clause has exactly 3 literals.

**Output**: Yes iff the formula is satisfiable.

**Conjunctive normal form**: AND of clauses.

**e.g.**:

\[
( x_1 \lor \neg x_2 \lor x_3 ) \land ( \neg x_1 \lor x_4 \lor x_5 ) \land ( x_2 \lor \neg x_5 \lor x_6 )
\]

(a clause)  
(literal: a variable or its negation)

**Aside: 3SAT is in **NP****

\[
\varphi = ( x_1 \lor \neg x_2 \lor x_3 ) \land ( \neg x_1 \lor x_4 \lor x_5 ) \land ( x_2 \lor \neg x_5 \lor x_6 )
\]

\( \varphi \) satisfiable

\[
\iff
\]

can pick one literal from each clause and set them to True

\[
\iff
\]

the sequence of literals picked does not contain both a variable and its negation.

What is a good proof that \( \varphi \in 3SAT \)?
CLIQUE is **NP**-complete: High level steps

CLIQUE is in **NP**. ✓

We know 3SAT is **NP**-hard.
So suffices to show 3SAT $\leq^P_{m} \text{CLIQUE}$.

**We need to:**

1. Define a map $f : \Sigma^* \rightarrow \Sigma^*$.
2. Show $w \in 3\text{SAT} \implies f(w) \in \text{CLIQUE}$
3. Show $w \notin 3\text{SAT} \implies f(w) \notin \text{CLIQUE}$
4. Argue $f$ is computable in polynomial time.

---

**3SAT $\leq$ CLIQUE: Defining the map**

1. Define a map $f : \Sigma^* \rightarrow \Sigma^*$.

   - not valid encoding of a 3SAT formula $\mapsto \epsilon$
   - otherwise we have valid 3SAT formula $\varphi$
     (with $m$ clauses).

   $\varphi \mapsto (G, k) \quad \text{(we set } k = m \text{)}$

   Construction demonstrated with an example.

---

**3SAT $\leq$ CLIQUE: Defining the map**

\[ \varphi = (x_1 \lor \neg x_2 \lor x_3) \land (\neg x_1 \lor x_2 \lor x_3) \land (x_1 \lor x_1 \lor \neg x_1) \]
\[ G_\varphi \]

\[
\begin{array}{cccc}
C_1 & \land & C_2 & \land & C_3 \\
\varphi = (x_1 \lor \neg x_2 \lor x_3) & \land & (\neg x_1 \lor x_2 \lor x_3) & \land & (x_1 \lor x_1 \lor \neg x_1) \\
G_{\varphi} & x_1 & \land & \neg x_1 & \land & x_2 & \land & \neg x_2 & \land & x_3 & \land & \neg x_3 & \land & x_1 & \land & \neg x_1 & \land & k = 3
\end{array}
\]
3SAT $\leq$ CLIQUE: Why it works

If $\varphi$ is satisfiable, then $G_\varphi$ contains an $m$-clique:

$\varphi$ is satisfiable $\implies$

$\implies G_\varphi$ contains an $m$-clique.

3SAT $\leq$ CLIQUE: Why it works

If $G_\varphi$ contains an $m$-clique, then $\varphi$ is satisfiable:

$G_\varphi$ has a clique $K$ of size $m$ $\implies$

$\implies \varphi$ is satisfiable.

3SAT $\leq$ CLIQUE: Poly-time reduction?

Creation of $G_\varphi$ is poly-time:

Creating the vertex set:
- there is just one vertex for each literal in each clause.
- scan input formula and create the vertex set.

Creating the edge set:
- there are at most $O(m^2)$ possible edges.
- scan input formula to determine if an edge should be present.
CIRCUIT-SAT is **NP**-complete

**Recall**

**Theorem:** Let \( f : \{0, 1\}^* \rightarrow \{0, 1\} \) be a decision problem which can be decided in time \( O(T(n)) \).
Then it can be computed by a circuit family of size \( O(T(n)^2) \).

Given a TM \( V \), we can create a circuit family that has the same behavior as \( V \).

With this Theorem, it is actually easy to prove that CIRCUIT-SAT is **NP**-hard.

**Proof Sketch**

**WTS:** for an arbitrary \( L \) in **NP**, \( L \leq_{m}^{P} \) CIRCUIT-SAT.