LECTURE 22: REDUCTIONS

This week: Meaning of "P=NP?"

Today:
1. Some problems we don't know how to solve in polynomial time
2. Reductions between problems

Decision problems:
Problems where goal of algorithm is to compute a binary value.

Example:
\[
O(V) \begin{cases} 
\exists \text{ Short Path} : \text{ Input: } G = (V, E, W); \text{ Output: } \text{YES if } d(s,t) \leq L. 
\end{cases}
\]
\(O(\log n)\) 

```
Distinct: Input = A[1..n]
Output: YES if \(\exists i \neq j \) \( A[i] \neq A[j] \).
```

Nice feature ... solvable in polytime.

Practically: \(O(1)\) factor more resources

\(=\) \(O(n)\) factor larger problems.

**Example:** We understand the problem well
denition

\(P = \text{Class of polynomial time solvable problems}\)

**Problem 1:** Factoring (Gauss)

**Input:** Given \(n\) bit integer \(X\), integer bound \(B (1 \leq B \leq X)\)

**Decide:** Does \(X\) have a factor \(Y\) s.t. \(2 \leq Y \leq B\) ?
**Problem 2:** Divisor (Kilian)

**Input:** Given a bit integer \( X \)  
Bounds: \( A, B \)

**Decide:** Does \( X \) have a factor \( Y \)  
\( A \leq Y \leq B \) ?

**Problem 3:** Common Divisor (Euclid)

**Input:** Two integers \( X_1, X_2 \)  
Also: Bound \( A \)

**Decide:** Does \( \exists Y \) dividing \( X_1, X_2 \)  
\( A \leq Y ? \)
Prob. 4: **Integer Programming** (von Neumann)

**Given**: Integer Matrix $A$, Vector $b$

**Question**: Does there exist a Vector $\mathbf{x}$ such that $A \mathbf{x} \leq b$?

\[ U \leq V \text{ if } \forall i \quad u_i \leq v_i \]

Prob 5: **Clique** (Cook/Karp)

- Subset of vertices $C$ in $G = (V, E)$ is a clique if $\forall u, v \in C \quad (u, v) \in E$.

**Given**: $G$, $k$ - integer

**Decide**: Does $G$ have a clique of size $k$?
Example

Clt-SAT: (Gok/Karp)

Digital Circuit:

(What does this compute?)

\( \overline{x_i} \)
Generally:

\[ \text{Circuit} = n \text{ variables } x_1 \ldots x_n \]

in gates \( g_1 \ldots g_m \)

Each gate \( c \in \{ \text{AND, OR, NOT} \} \)

\( g_i \) takes as input, output of \( g_j \) for smaller \( j \) or input variable

**Problem**: Given circuit \( C \) on \( x_1 \ldots x_n \)

**Decide**: Does \( \exists x_1 \ldots x_n \) s.t.

\[ g_m = 1 \]
Hypothetical Debate

Who is trying to solve a hard problem?

Evolution: GCD solves his problem.

Exits Debate

Others are still fighting it out...

Can they come to some civilized conclusion?
Actually

Gowers ≤ Karp = Cook = von Neumann ≈ Kilián

How does one prove this?

Problem $\mathbb{T}_{1} \leq \mathbb{T}_{2}$ if ....

"no harder than"

Polynomial time Alg. for $\mathbb{T}_{1}$

$\Rightarrow$ Polynomial time Alg. for $\mathbb{T}_{2}$.

How could you prove such an implication without solving either problem?
"Structure of proof"

Build algorithm for \( \Pi_1 \)

using subroutine for \( \Pi_2 \)

Cook-Reduction

We will require very specific algorithms

\[
\{ \\
\text{Solve-}\Pi_1(x) \\
\text{; Blah Blah ...} \\
\text{Construct } y; \\
\text{Return: Solve-}\Pi_2(y) \\
\} 
\]
$\Pi_1 \leq \Pi_2$

If there exist a polynomial time algorithm $f$

such that for every $x$

$" \Pi_1 (x) = \text{YES} " \iff " \Pi_2 (f(x)) = \text{YES} "$

Example:

Factoring $\leq$ Divisor

$(X, B) \xrightarrow{f} (X, 2, B)$
Factoring $\leq$ $C_{\text{kt}} - \text{SAT}$

Given: $(X, B)$

Need to produce

$$C = C_{X, B}$$

$s.t.$ some input $Z$ satisfies

$$C(Z) = 1 \Rightarrow \exists Y \leq B \ s.t. \ Y | X.$$

Idea: will produce

$$C(Z) = 1 \Rightarrow Z = (Y, W)$$

$s.t. \ Y \leq B \ s.t. \ Y | X.$
Build circuit

\[ \hat{C}(X, Y, W, B) = 1 \text{ if } Y \cdot W = X \]
\[ \quad \& \quad Y \leq B \]

How to build \( \hat{C} \) ?
\[ \hat{C}(Y, W, X, B) = \hat{C}_1(Y, W, X) \land \hat{C}_2(Y, B) \]

\[ \hat{C}_1(Y, W, X) : Y \ast W = X ? \]

Exercise: Build multiplier circuit!

must have size \( \text{poly}(n) \)!
Exercise: Build ≤ ? circuit:

\[ C \Rightarrow C \]

Set the hardwire values of \( X, B \):

\[ C \text{ has satisfying assignment} \]

\( \Rightarrow \) \( X \text{ has factor } \leq B. \)
Sanity Check

Which problem is easy?
Which problem is hard?

Can show similarly:
- Clique \leq \text{Ckt-SAT}
- Integer Programming \leq \text{Ckt-SAT}
- Divisor \leq \text{Ckt-SAT}

How: build circuit that checks if a solution is really a solution

Next lecture "NP", "NP" completeness.