NP-Completeness

Basic Topic so far:

Present a problem (e.g., Max-Flow-Prob) and give an algorithm for the problem.

For each problem, we gave a bound on time, space, or work (e.g., $O(n^3)$).

Alice & Bob Problem:

Bob's job: Write efficient code for 2-coloring a graph.

Alice's job: "3-coloring a graph!"
Run-time Guarantees

All have been polynomial time:

$O(n^k)$ for some $k$

Times like $O(2^n)$ or $O(n \log n)$ are not polynomial time guarantees.

Problem Size and Actual Size

<table>
<thead>
<tr>
<th>Prob</th>
<th>Prob Size</th>
<th>Actual Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G = (V, E)$</td>
<td>$V + E$</td>
<td>$V^2, (V+E)\log V$</td>
</tr>
<tr>
<td>Data Structure Keys</td>
<td>$n$</td>
<td>$n \log n$</td>
</tr>
<tr>
<td>Matrix $n \times n$</td>
<td>$n$</td>
<td>$n^2 \log n \log n$</td>
</tr>
</tbody>
</table>
Claim! A polynomial of a polynomial is a polynomial.

If $f, g \in \mathbb{Q}[x]$ then $f(g(x)) \in \mathbb{Q}[x].$

In particular $\deg(f) = n$ & $\deg(g) = m$ then $\deg(f \circ g) = nm.$

---

What is a Prob?

Input-Output Form

eds

Input: $G = (V, E)$ $a, b \in V$ encoded as a string in binary

Output: Shortest path from $a$ to $b$ encoded as a string in binary.
Language form: e.g.,

\[ \text{Prime} = \{ x \in \{0,1\}^* \mid x \text{ is a number } n \text{ in binary } \land \]  
\[ n \text{ is a prime number} \} \]

\[ \text{Def: } P = \{ L \subseteq \{0,1\}^* \mid \exists \text{ poly-time alg for deciding } L \} \]

\[ \text{Thm: } \text{Prime} \in P \text{ (no proof)} \]

\[ \text{Def: A cycle } C \text{ of } G \text{ is Hamiltonian if} \]
\[ 1) C \text{ is simple} \]
\[ 2) C \text{ contains every vertex of } G. \]

\[ \text{Ham-Cycle} = \{ G \mid G \text{ has a Ham-cycle}\} \]
\[ \{ \exists \text{ Ham-cycle of } G \} \]
**Witness and Certificates**

$$H = \{ (G,C) \mid C \text{ is a Ham-cycle of } G \}$$

**Note:** $H \in P$

$$\text{Ham-Cycle} = \{ G \mid \exists C \text{ s.t. } (G,C) \in H \}$$

**Witness**

$$\text{NP} = \{ L \mid \exists R \text{ (poly-time relation) } \land \forall x \in L \exists y \in \{0,1\}^* \land |y| \leq |x|^k \text{ for some } k \}$$

**Thus** $\text{Ham-Cycle} \in \text{NP}$

$$L = \{ x \in \{0,1\}^* \mid x \notin L3 \}$$

**Def** $\text{co-NP} = \{ L \subseteq \{0,1\}^* \mid \overline{L} \in \text{NP} \}$
Reducibility

Many-One: \( L_1 \leq_p L_2 \) if \( \exists \) f such that

1. f is poly-time computable
2. \( x \in L_1 \) iff \( f(x) \in L_2 \)
Turing Reduction!

\[ l_1 \leq_T l_2 \text{ iff } \exists \text{ poly-time alg } A \text{ s.t.} \]

1) \( l_1 = \text{lang accepted by } A \)
2) A may require calls to
   oracle for \( l_2 \).

Claims:

1) \( l_1 \leq_p l_2 \leq_p l_3 \implies l_1 \leq_p l_3 \)
2) \( l_1 \leq_T l_2 \leq_T l_3 \implies l_1 \leq_T l_3 \)
3) \( l_1 \leq_p l_2 \land l_2 \in \text{NP} \implies l_1 \in \text{NP} \)
Def \( L \) is NP-Hard if \( \forall L' \in \text{NP} \) \( L' \leq_p L \)

Def \( L \) is \( \text{NP-Complete} \) if

1) \( L \in \text{NP} \)
2) \( L \) is NP-Hard
Models of computation for which Poly-time are equivalent!

1) RAM
2) Turing Machines
3) Parallel-RAM \( \text{poly} \) (# of processors)
4) Arrays of finite controls
5) \( \lambda \)-calculus

Not known

Quantum Computers
The First NP-Complete Problem

**Boolean Formula**

- **Variables**: $x_1, \ldots, x_n$  \quad $\bar{x}_i$ denotes $\neg x_i$
- **Literals**: $x_1, \bar{x}_2, \ldots, x_n, \bar{x}_n$  \quad $q_i$ denotes a literal

$\land \equiv$ and  $\lor \equiv \lor$ (or)

**Boolean Formula** = Formula made from $\land$, $\lor$, Literals

**Conjunctive Normal Form** = Formula $\land \lor ($literals$)$

**Example**:

$(x_1 \lor x_2 \lor \bar{x}_3)(x_1 \lor x_2 \lor x_4)(\ldots)$

Clause

CNF $\equiv \{ \varphi \in \text{cnf.} \mid \varphi \text{ is satisfiable} \}$

i.e. $\uparrow$ Variables $\rightarrow \{ T, F \}$ s.t. $T(\varphi)$ is true
(Cook's Thm) CNF is NP-Complete

No proof! See Kogem