Basic Topic so far!

Present a prob (eg Max-Flow-Prob) give an algorithm for the prob.

For each prob we gave a bd on time, space, on work (eg $O(n^3)$)

Alice & Bob Prob:

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^{3 \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 \( \text{deg}(f) = n \) & \( \text{deg}(g) = m \)

then \( \text{deg}(f \circ g) = n \cdot m \)

---

What is a Prob?

Input - Output form

\[ G = (V, E), a, b \in V \text{ encoded as a string in binary} \]

Output: Short path from \( a \) to \( b \)

encoded as a string in binary
Language Form: e.g.

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

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

\[ \text{Thm } \text{Prime } \in P \hspace{1cm} (\text{no proof}) \]

\[ \text{Def } \text{A cycle } C \text{ of } G \text{ is Hamiltonian if} \]

1) \( C \) is simple
2) \( C \) contains every vertex of \( G \).

\[ \text{Ham-Cycle} = \{ G \mid G \text{ has a Ham-cycle} \} \]

\[ \{ G \text{ has a Ham-cycle of } G \} \]
Witness and Certificates

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

**Note:** \( H \in \text{P} \)

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

\[ \text{Witness} \]

\[ \text{NP} = \{ L \mid \exists R \text{ (poly-time relation) } \wedge \]
\[ X \in L \iff (\exists y R(x, y) \wedge |y| \leq |x|^k \text{ some } k) \} \]

Thus, \( \text{Ham-Cycle} \in \text{NP} \)

\[ L = \{ x \in \{0,1\}^* \mid x \notin \text{L3} \} \]

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

Many-One: \( L_1 \leq_p L_2 \) if \( \exists f \text{ s.t.} 
\begin{align*}
1) & f \text{ is poly-time computable} \\
2) & x \in L_1 \iff f(x) \in L_2
\end{align*}
Turing Reduction:

$L_1 \leq_T^P L_2$ iff $\exists$ poly-time alg $A$ st

1) $L_1 = L_2$ is accepted by $A$

2) $A$ may require calls to oracle for $L_0$.

Claims:

1) $L_1 \leq_P L_2 \leq_P L_3 \implies L_1 \leq_P L_3$

2) $L_1 \leq_T^P L_2 \leq_T^P L_3 \implies L_1 \leq_T^P L_3$

3) $L_1 \leq_P L_2 \land L_2 \in P \implies L_1 \in P$
Def. $L$ is NP-Hard if $\forall L' \in \text{NP}$, $L' \leq_p L$.

Def. $L$ is 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 ($\text{poly} \cdot \#\text{of processors}$)
4) Arrays of finite controls
5) $\lambda$-calculus

Not known

Quantum Computers
The First NP-Complete Prob

**Boolean Formula**

- Variables: $x_1, \ldots, x_n$, $\overline{x}_i$ denote not $x_i$
- Literals: $x_1, \overline{x}_2, \ldots, x_n, \overline{x}_n$, $a_i$ denote a literal

$\land \equiv \text{and} \quad \lor \equiv \text{or}$

**Boolean Formula** = Formula made from $\land, \lor, \neg, \text{ literals}$

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

**Example**: $(x_1 \lor x_2 \lor \overline{x}_3)(x_1 \lor \overline{x}_2 \lor x_4)(\ldots)$

**Clause**

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

ie $\forall \text{ variables} \rightarrow \{T, F\}$ s.t. $T(\varphi)$ is true
(Cook's Thm) CNF is NP-Complete

No proof! See Kogen