CSCI 3100

Hardness & Undecidability
Today:

- HW due
- Office hours today: 12-2:30
Fundamental question: Are there "harder" problems? How do we rank?
- Polynomial
- Unsolvvable?

Undecidability:
Some problems are impossible to solve!
The Halting Problem:
Given a program P and input I, does P halt or run forever if given I?

Output: True/False
(Utility should be obvious!)

Note: Can't just simulate P on I. Why?
We'd never output False!
The halting problem is undecidable.

Proof: by contradiction - suppose we have such a program \( h \):

\[
h(P,I) = \begin{cases} 
3 & \text{if } P \text{ halts on input } I \\
0 & \text{otherwise} 
\end{cases}
\]
Now define a program $g$ that uses $h$: 

$$g(X) := \begin{cases} 0 & \text{if } h(X,X) = 0 \\ \text{loop forever} & \text{else} \end{cases}$$

The contraction: What does $g(g)$ do?

- Calls $h(g,g)$:
  - If $h(g,g) = 1$, that means $g$ halts on input $g$.
  - But then $g(g)$ should run forever.
  - If $h(g,g) = 0$, then $g$ on input $g$ runs forever.
  - But by definition of $g$, it should return 0 and halt.

Either way, impossible!
So... what next?

Clearly, many things are solvable in polynomial time.
Some things are impossible.
But—what is in between?

Idea:

- Some things require exponential time.
- Subexponential (but super polynomial)
  eg. $2^{\sqrt{n}}$
  (eg. factoring)
The first problem found: Boolean circuits

An AND gate, an OR gate, and a NOT gate.

A boolean circuit. Inputs enter from the left, and the output leaves to the right.

Given a set of inputs, can clearly calculate output in linear time \((m \# \text{ inputs} + n \# \text{ gates})\).

How? "reverse" BFS \(O(m+n)\)
Q: Given such a boolean circuit, is there a set of inputs which result in \text{TRUE} output?

Known as \textsc{CircuitSatisfiability} (or \textsc{Circuit SAT})
Best known algorithm:
Try all $2^n$ possible inputs.

Running time:

$$2^n \times O(m+n) \leq O(2^n)$$

Note:
Consider only decision problems:
so Yes/No output

P: Set of decision problems that can be solved in polynomial time.

Ex:  - Is list sorted?
     - Is x is list?
     - Is list of length k?
     - Is there a flow of value k in G?

NP: Set of problems such that, if the answer is yes + you hand me proof, I can verify/check in polynomial time.

Ex:  - Circuit SAT

Co-NP: can check "No" answers
**Def:** NP-Hard

$\times \text{ is NP-Hard} 

\iff 

If any NP-Hard problem could be solved in polynomial time, then $P=NP$.

So if any NP-Hard problem could be solved in polynomial time, then all of NP could be.

*(Paths story in reading...)*
Cook-Levin Thm:
Circuit SAT is NP-Hard.

Polynomial hierarchy
NP-Complete:
- In NP
- And NP-Hard
To prove NP-Hardness of A:

Reduction:
Reduce a known NP-hard problem to A.
We've seen reductions!

\[ \text{O(docto scheduling)} \leq \text{O(network flow)} \]
This will feel odd, though:

To prove a new problem is hard, we'll show how we could solve a known hard problem using a new problem as a subroutine.

Why?

Well, if a poly time algorithm existed, than you'd also be able to solve the hard problem! (Therefore, can't be any such solution.)
Other NP-Hard Problems:

SAT: Given a boolean formula, is there a way to assign inputs so result is 1?

Ex: \((a \lor b \lor c \lor \bar{d}) \iff ((b \land \bar{c}) \lor (\bar{a} \Rightarrow \bar{d}) \lor (c \neq a \land b))\),

\(n\) variables,
\(m\) clauses

In NP:

Given assignment can check in poly time.
**Thm:** SAT is NP-Hard.

**PF:** Reduce CIRCUIT SAT to SAT:

\[
(y_1 = x_1 \land x_4) \land (y_2 = \overline{x_4}) \land (y_3 = x_3 \land y_2) \land (y_4 = y_1 \lor x_2) \land \\
(\overline{y_5} = \overline{x_2}) \land (y_6 = \overline{x_5}) \land (y_7 = y_3 \lor y_5) \land (z = y_4 \land y_7 \land y_6) \land z
\]

A boolean circuit with gate variables added, and an equivalent boolean formula.
So our reduction:

\[
T_{CSAT}(n) \leq O(n) + T_{SAT}(O(n)) \implies T_{SAT}(n) \geq T_{CSAT}(\Omega(n)) - O(n)
\]

Tomorrow: More reductions!