Lecture 17 CSE 431 Intro to Theory of Computation

Previously: $NP = \bigcup_k NTIME(n^k)$

$P = \bigcup_k TIME(n^k)$

Is $P \stackrel{?}{=} NP$?

- polynomial-time mapping reductions
  $A \leq_m^P B$ (or $A \leq_p B$)
  just like def^n of mapping reduction except f must be poly time computable

$A \leq_m^P B$ "A is at most as hard as B up to 'polynomial slop'"

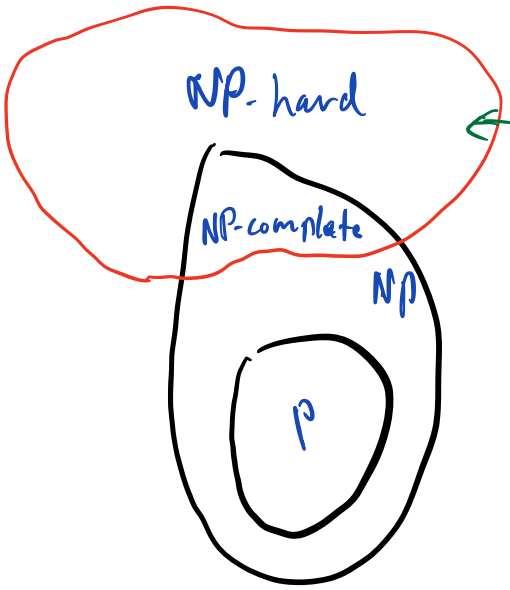
Thm
- If $A \leq_m^P B$ and $B \in P$ then $A \in P$
- If $A \leq_m^P B$ and $B \in NP$ then $A \in NP$

We now add def^{n}s:

Def^n. B is NP-hard iff $\forall A \in NP.\ A \leq_m^P B$

Def^n. B is NP-complete iff
- $B \in NP$ and
- B is NP-hard

We now look at a picture of these classes



could contain extremely hard problems

Thm [Cook, Levin] 3SAT is NP-complete

↑ special case of SAT for 3CNF formulas

We prove this via a different SAT problem:

CIRCUIT-SAT = {<C>: C is a Boolean circuit s.t. ∃ input y s.t. $C(y)=1$}

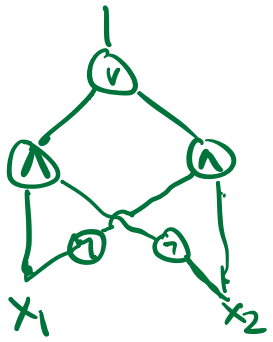
Boolean circuit: inputs $x_1, \ldots, x_n$

gates: (bottom up)  OR  AND  NOT

Given by graph: output gate special marked gate.

Computes a Boolean function $f: \{0,1\}^n \to \{0,1\}$

e.g. Parity, aka. XOR
$x_1 \oplus x_2$



size = # gates

depth = length of longest path from input to output.

eg. Can combine these units into a larger circuit for functions like $x_1 \oplus x_2 \oplus \cdots \oplus x_n$

Using just $\wedge, \vee, \neg$ takes $O(n^2)$ gates to compute $x_1 \oplus x_2 \oplus \cdots \oplus x_n$

Note: TMs work on inputs of all lengths
Circuits only work for a fixed input length

Thm (Shannon) Almost all functions on n bits require circuit size $\Omega(2^n/n)$

Proof (Counting)

- How many circuits of size S are there on n inputs?

S gates: • Each gate described by
gate type — 3 choices
2 wires each input wire — $S+n$ options (other gates ↑ input vars)
$\leq 3 \cdot (S+n)^2$ options
: • Output gate — S options

Total: $S \cdot (3(S+n)^2)^S$

without loss of generality $S \geq n$

So $S^{O(S)}$ circuits of size $\leq S$.

- How many functions on n bits are there?

  2^n bit vector inputs 2 choices for each

  total: 2^{2^n} functions

- To get at least $1-o(1)$ (almost all) fraction of all functions in size S we need

$$2^{2^n}(1-o(1)) \leq S^{O(S)}$$

taking logarithms we need

$$2^n - o(1) \leq O(S \log S)$$

$\therefore S$ must be $\Omega(2^n/n)$

(Actually $2^n/n - o(2^n/n)$) ∎

Note: This doesn't tell us that any specific function is hard.

- We will see that any problem in P actually has circuits of size $n^{O(1)}$ for n-bit inputs

So how do we show that any $A \in NP$ mapping reduces to CIRCUIT-SAT?

Basic idea.

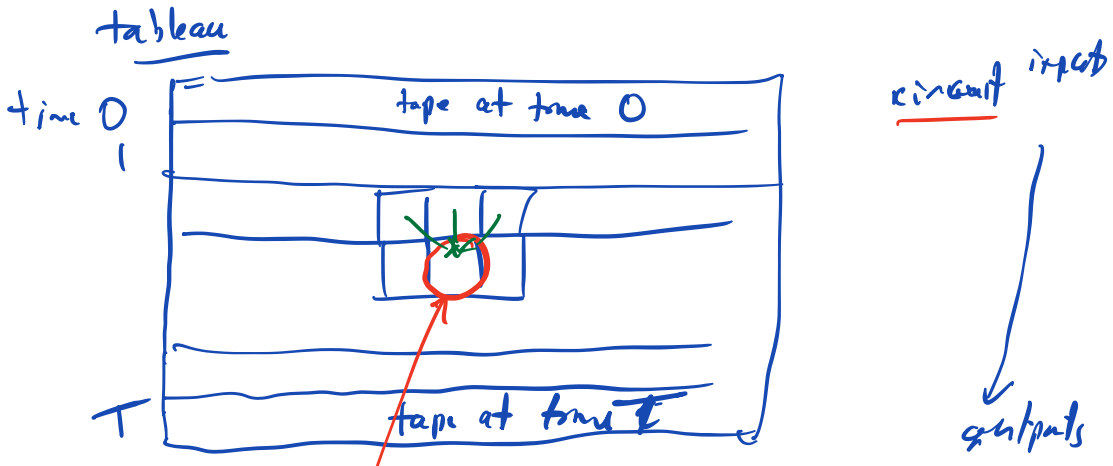
$A \in NP$ with verifier V_A

$A \xrightarrow{f} \text{CIRCUIT-SAT}$

$x \longmapsto \langle C_{V_A, x} \rangle$

↑ circuit depending on x, and using V_A that takes possible certificate y as input

s.t. $x \in A \iff C_{V_A, x}$ is satisfiable.



represent contents $\in \Gamma$ using bits

a b c d $\in \Gamma$

0 1 0 0 ← exactly one gate has value 1

"1-hot encoding"

Details next class

represent state $\in Q$ using bits

q_0 q_1 ... q_k Q

0 1 0 0 ← state if head points there

or 0 0 0 0 ← head not there