Computational complexity : Lecture 5.

Agenda: Diagonalisation -
- ▷ Time hierarchy theorems
- ▷ Ladner's theorem.

Recap: - $P = \bigcup_{c \geq 0} \mathrm{DTIME}(n^c)$.
- $NP = \bigcup_{c \geq 0} \mathrm{NTIME}(n^c)$
- Also saw other classes like coNP, NEXP, EXP etc.

How do we show certain tasks are not in a class $\mathcal{C}$?

Qn: Can you give me a task that cannot be solved in P?

Halting ...

Qn: Can you give me a task that can be solved in $O(n^3)$ but cannot be solved in $O(n)$?

Task: On 1^n, print n^3 ones.

Qn: What about decision tasks? As languages, is $\mathrm{DTIME}(n) \subsetneq \mathrm{DTIME}(\underline{2^n})$
n^2

Diagonalisation:

- uncountability of reals
- ...

Time Hierarchy Theorem: Suppose $f: \mathbb{N} \to \mathbb{N}$ & $g: \mathbb{N} \to \mathbb{N}$ are time constructible and g is "sufficiently bigger"* than f. Then

$$\mathrm{DTIME}(f(n)) \subsetneq \mathrm{DTIME}(g(n)).$$

*: $f(n) \log f(n) = o(g(n))$

We will assume $f(n) = n$ $\quad g(n) = n^3$.

Candidate: $\{(\alpha, x) : M_\alpha \text{ accepts } x \text{ in } n^2 \text{ steps}\}$.

Pf: Idea:

	x_1	x_2	---
M_1	A	R	
M_2	A	A	
.	R	A	
.			
.			

$D(i) = \neg M_i(x_i)$

D: On input (α, x):

Use the UTM for n^2 steps to simulate M_α on (α, x).

Reject only if M_α accepts by then. Else, accept.

Claim 1: $L(D) \in \mathrm{DTIME}(n^3)$

Claim 2: $L(D) \notin \mathrm{DTIME}(n)$.

Recall: The UTM, on input $(\alpha, x, 1^t)$ can simulate M_α on x for t steps in time $c \cdot t \log t$. where c depends only on M_α's alphabet size, #tapes.

Pf of Claim 1: Duh! $\square$.

Pf of Claim 2: Suppose, for contradiction S solves it in time n.

Let α be a description of S that is really long.

$|\alpha| = \ell$

What does S do on $(\alpha, 1^\ell)$?

$$\begin{array}{ccc} S \text{ accepts } (\alpha, 1^\ell) & \Leftrightarrow & D \text{ accepts } (\alpha, 1^\ell) \\ \Updownarrow & & \Updownarrow \\ M_\alpha \text{ rejects } (\alpha, 1^\ell) & \Leftrightarrow & \text{UTM for } n^2 \text{ steps saw } M_\alpha \text{ reject } (\alpha, 1^\ell) \end{array}$$

t steps of M_α can be sim in $c \cdot t \log t$ steps of UTM.

$\Rightarrow$ If n is large enough, $n^2 > c \cdot n \log n$. $\square$.

What about with non-determinism?
If $f << g$, is $\mathrm{NTIME}(f) \subsetneq \mathrm{NTIME}(g)$?

Non-det time hierarchy thm: f, g time constructible with "$f << g$". Then $\mathrm{NTIME}(f) \subsetneq \mathrm{NTIME}(g)$.

Why doesn't the same proof work?

In non-determinism, flipping is expensive?

Recall NP vs co NP.

[AB] has a proof of this thm using "epochs".
We'll see a different proof by Fortnow-Santhanam.

Key: When we want to flip, we'll do det. simulation.

Pf: D: On input (α, x, y):

▷ If $|y| < |\alpha| + |x|$: Use the N. UTM for n^2 steps
Simululate M_α on $(\alpha, x, y0)$ & $(\alpha, x, y1)$
Accept if M_α accepts both.

▷ If $|y| \geq |\alpha| + |x|$:
Simulate deterministically M_α on (α, x, ε)
by using y as the guesses.
Flip the answer.

Claim: $L(D) \in \mathrm{NTIME}(n^3)$

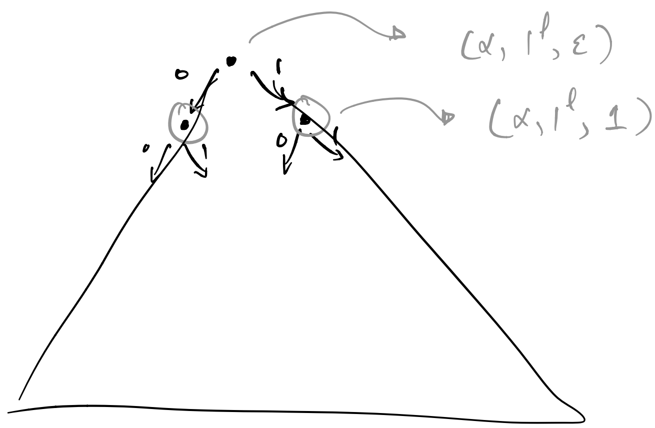
Pf: Duh!

Claim: $L(D) \notin \mathrm{NTIME}(n)$.

Pf: Assume S is a N.TM that solves $L(D)$ in time n

Assume α is a long enough encoding of S so that UTM for n^2 steps completely simulates n steps of S. Let $|\alpha| = \ell$

We will fix $\alpha, 1^\ell$ and vary y.



S acc $(\alpha, 1^\ell, \varepsilon)$

$\Updownarrow$

D accepts $(\alpha, 1^\ell, \varepsilon)$

$\Updownarrow$

D acc. $(\alpha, 1^\ell, 0)$ & $(\alpha, 1^\ell, 1)$

$\Updownarrow$

$\vdots$

D accept $(\alpha, 1^\ell, y)$ $\forall y: |y| < |\alpha| + \ell$

D accepts $(\alpha, 1^\ell, y)$ $\forall y: |y| \le |\alpha| + \ell$ $\iff$

$\Updownarrow$

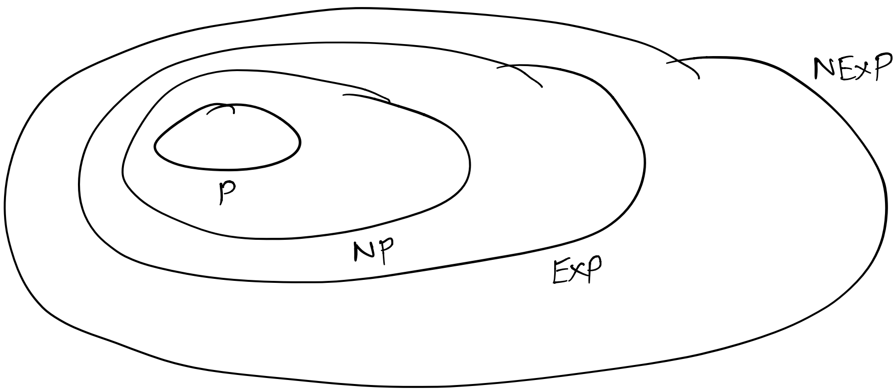
S, on guess y, rejects $(\alpha, 1^\ell, \varepsilon)$ $\forall y: |y| \le |\alpha| + \ell$. $\iff$ S rejects $(\alpha, 1^\ell, \varepsilon)$ $\square$.

Actual thm statement: f, g time-constructible with $f(n+1) = o(g(n))$, then $\mathrm{NTIME}(f(n)) \subsetneq \mathrm{NTIME}(g(n))$

Where is the log factor?!

Non. det UTM simulation only has a constant overhead.

(Ex 2.6 in Arora-Barak)



$P \subsetneq EXP$ $\qquad$ $NP \subsetneq NEXP$

Another application of diagonalisation:

Let us assume we are in the world where $P \neq NP$.

Can it be the case that <u>every</u> problem not in P is actually NP-complete?

Or are there "NP-intermediate" languages?

Ladner's Theorem: Suppose $P \neq NP$. Then there are languages that are neither in P nor NP-complete.

Pf: Idea: How can we make SAT easier?

Give more time...

For $H: \mathbb{N} \to \mathbb{N}$, non-dec. function.

$$SAT_H = \{ (\varphi, 1^{n^{H(n)}}) : |\varphi| = n, \ \varphi \in SAT \}.$$

SAT with the "right" padding.

Want to show: SAT_H is NP-hard $\Rightarrow$ $P = NP$. How?!

$$SAT \leq_p SAT_H$$

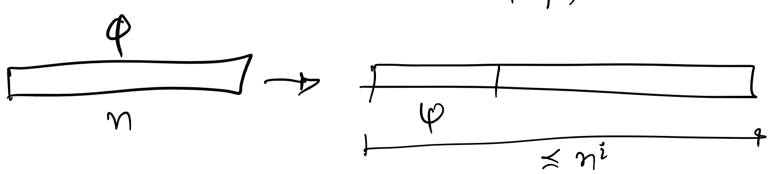
n^i

$$\varphi \longmapsto (\psi, 1^{m^{H(m)}})$$

$m = |\psi|$

Suppose $H(m) \to \infty$

$\Rightarrow$ For large m, $H(m) \geq 2i$



$\Rightarrow m^{H(m)} \lesssim n^i$

$m \lesssim n^{i/H(m)} \lesssim \sqrt{n}$

We also want $SAT_H \in NP$

$(\varphi, 1^{\ell}) \overset{?}{\in} SAT_H$ in NP?

If $H(n)$ can be computed in poly time, then $SAT_H \in NP$.

Want to show: $SAT_H \in P \Rightarrow P = NP$.

If it so happens that SAT_H is NP-complete, then $SAT_H \in P \Rightarrow P = NP$.

Suppose $H(n)$ is eventually constant,

ie $H(n) = i \quad \forall\, n \geq n_0$.

then $SAT \leq_p SAT_H$:

$$\varphi \mapsto (\varphi, 1^{n^i})$$

Cool Idea: We will cleverly choose H so that

H is eventually constant if $SAT_H \in P$

H is inc. otherwise.

Defn: $H(m)$:

The smallest $i \leq \log\log m$ s.t the machine

D.TM M_i solves SAT_H on inputs of length $\leq \log m$

in n^i time.

If no such i, then $H(m) = \log\log m$.

Ex: $H(m)$ is computable in poly(m) time.

Lem: If $SAT_H \in P$, then H is eventually constant.

Pf: If $SAT_H \in P \Rightarrow SAT_H \in DTIME(n^i)$ for some i.

$\therefore$ $SAT_H \in P \Rightarrow SAT_H$ is NP-hard $\Rightarrow$ $P = NP$

$SAT_H \notin P \Rightarrow H$ is increasing.

$\Rightarrow$ Any poly time redn from $SAT \to SAT_H$ gives a length-dec. redn from $SAT \to SAT$

$\Rightarrow$ SAT_H is NP-hard $\Rightarrow$ $P = NP$. $\square$

Summary:

- Diagonalisation is a powerful tool.

  $P \subsetneq EXP$, $NP \subsetneq NEXP$

Next time: - What are some limitations of this technique?

- Can "such arguments" show $P \neq NP$?