TODAY

- TIME + SPACE HIERARCHY THEOREMS

- "More of some resource ⇒ More power"

- Reading: [Arora-Barak] 3.1, 3.2

- Basics:

  - Algorithm: → Turing Machine
    → NAND++ program
    → High-level language program (C++)

    Common features: ① Every algorithm of constant size
    ② Can enumerate all algorithms.
    ③ Some notion of time, space.
    ④ Universal Algorithm.

  - Languages $L \subseteq \{0,1\}^* \triangleq \bigcup_{n \geq 0} \{0,1\}^n$
    ⇓
    Associated problem: given $x \in \{0,1\}^*$, decide if $x \in L$? (same as decision problem)

Computability & Time (Space) Complexity

Language $L \subseteq \{0,1\}^*$:

Defn: L is computable if $\exists$ algorithm A s.t.

$\forall x \quad x \in L \Leftrightarrow A(x)=1$ ["A solves L"]

& A halts in finite time for every x.

[E.g. O.K. to run in time $2^{2^{2^x}}$].

Defn: L has time complexity $T(n)$ if $\exists$ A solving L with running time $T(|x|)$ for every x.

(similarly Space complexity $S(n)$)

Defn: TIME $(t(n))$ denotes all ~~computat~~ languages that have time complexity $O(t(n))$.

(Similarly SPACE $(s(n))$)

[Upto $t(n)$ vs $t(n)\log t(n)$ – model is not important. Even tighter for space.]

$$P \triangleq \bigcup_c \text{Time}(n^c).$$

Main Theorem for Today

$$\text{Time}(n^c) \subsetneq \text{Time}(n^{c+1})$$

[more precise careful versions exist. Exercise.]

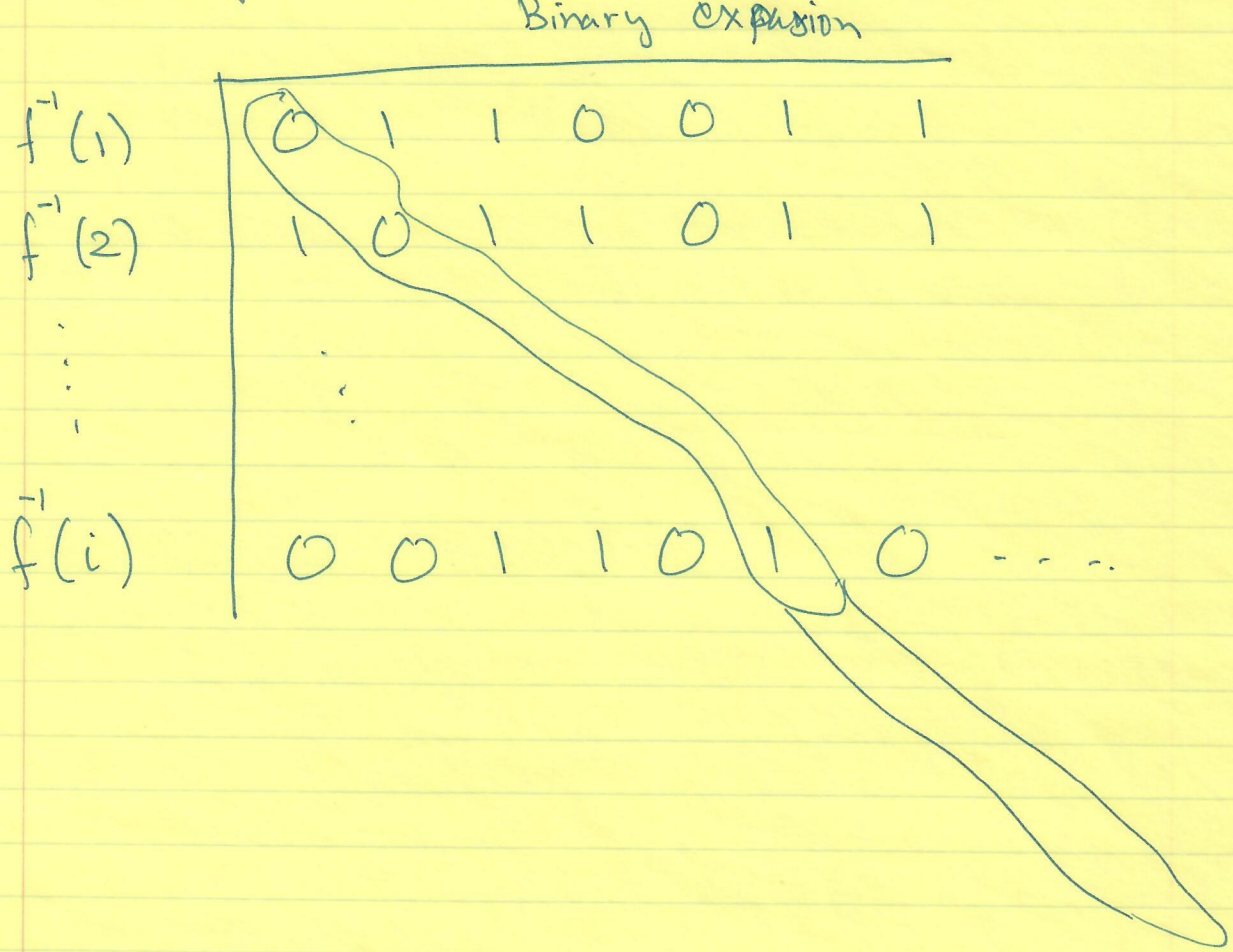
Key Idea: Diagonalization

- Invented by Cantor ⇐ deemed crazy
- Used by Turing / Gödel ... [Halting Problem is not computable]
- Repurposed for Hierarchy Theorems [Hartmanis + Stearns '65].

Cantor's Thm : # reals > # integers

[there is no injective function $f: [0,1] \to \mathbb{Z}^{>0}$]

Proof: Suppose there is ... enumerate



① Take Diagonal ;

② Complement; get real number in [0,1]

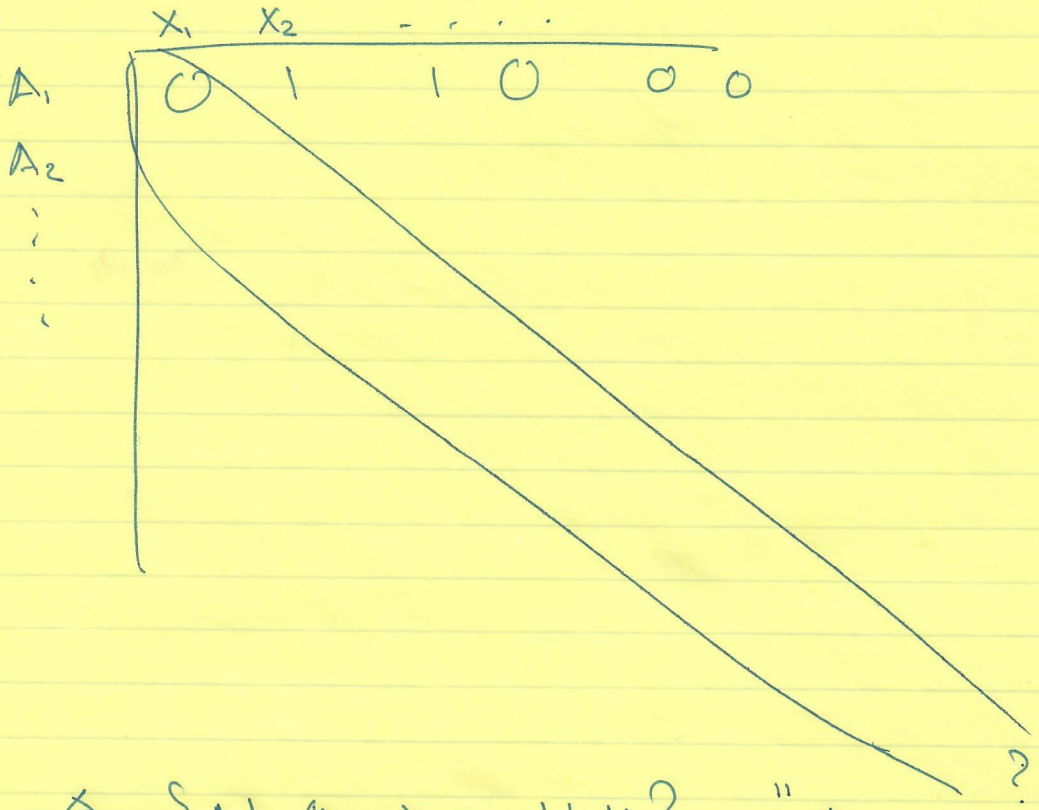
③ which in not equat to any row.

Turing's Theorem (cheap version): $\exists$ uncomputable function.

Proof: same matrix; rows = algorithms; cols = inputs.

Turing's Theorem (strong version): Halting Problem is undecidable.

- $Halt \triangleq \{ (A,x) \mid A \text{ halts on } x \}$



$D\text{-}halt \triangleq \{A \mid (A,A) \in Halt\}$ "Take the diagonal"

Halt - Computable $\Rightarrow$ D-Halt Computable

$\Rightarrow$ $\overline{D\text{-}Halt}$ Computable $\Leftarrow$ complement

Say u/g B solves it. Then $B(B) = ?$

Key Ingredients

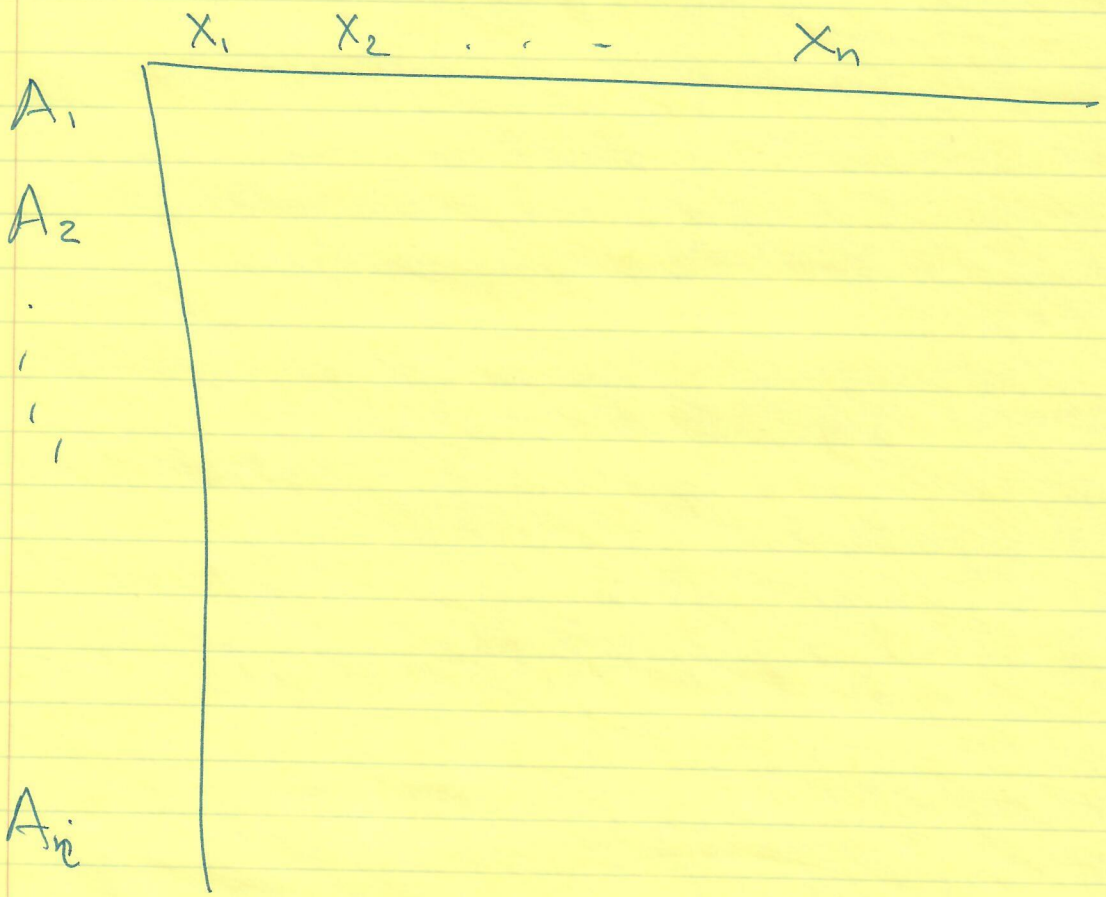
(1) Algorithms = Input ; (Universal TM ; Simulation time)

$\Downarrow$

$\exists$ recursive language that is not computable "

(2) Enumerate all Algorithms .

Modifying for $\text{TIME}(n^2) \neq \text{TIME}(n^3)$



Q1. Can we enumerate all algorithms, running in time $O(n^2)$

NOT OBVIOUS !!

Natural method doesn't work.

"Enumerating $\text{TIME}(n^2)$":

$$A \longrightarrow (A, c, n_0)$$

$$\Uparrow$$

run $A(x)$ for $c \cdot n^2 + n_0$ steps
if $A(x)$ halts output $A(x)$
else output 0

Claims: (1) $\forall\ A, c, n_0$ (A, c, n_0) runs in $\text{TIME}(n^2)$

(2) $\forall\ A \in \text{TIME}(n^2)$ $\exists\ c, n_0$ s.t. $\forall x$.

$$A(x) = (A, c, n_0)(x).$$

Now enumerate (A, c, n_0) ....

Done? $\text{Time}(n^2) \subsetneq \text{Time}(n^3)$?

Not YET.

Diagonal computation takes more than n^3 steps.

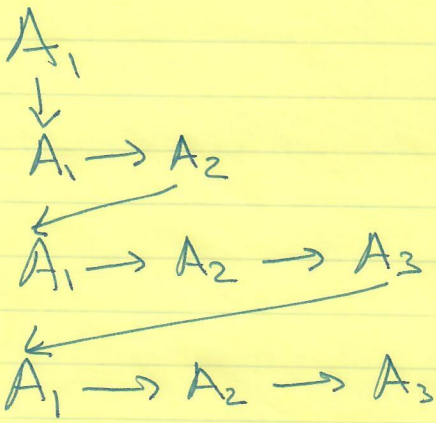
for example, maybe A_i runs in time $2^i \cdot n^2$.

can't simulate $\underline{A_i(x_i)}$ & complement !

Language L : ~~Sum~~

$L(x_i)$: - Simulate $A_i(x_i)$ for $|x_i|^3$ steps
- if $A_i(x_i)$ halts output $\overline{A_i(x_i)}$
- else 0.

- Clearly $L \in$ Time (n^3).
- Why is $L \notin$ Time (n^2) ?
- Not for natural enumeration of Time (n^2) but lets use the following



[Every A_i appears infinitely often.]

For every A_i
Eventually $\overline{|x_j|}$ is large enough so that $L(x_j) = \overline{A_i(x_j)}$. At this stage we get $L \neq L(A_i)$.

Putting things together .... $\text{TIME}(n^2) \subsetneq \text{TIME}(n^3)$.

Strengthenings

- $\text{Time}(n^2) \subsetneq \text{Time}(n^{2.1})$
- $\text{Time}(n^2) \subsetneq \text{Time}(n^2 \cdot (\log n)^{1.00001})$

"log n" loss inherited from "simulation" delay

Abstractly

← Technical, uninteresting

$\forall$ $f(n)$, $g(n)$, $h(n)$ "nice" s.t.,

$f(n) \log f(n) = o(g(n))$ &

$f(n) = o(h(n))$

$\text{TIME}(f(n)) \neq \text{TIME}(g(n))$

$\text{SPACE}(f(n)) \neq \text{SPACE}(h(n))$

Looking ahead

- $NTIME(t(n)) \stackrel{?}{\neq}$?
  - Proof only gives $\neq$ $co\text{-}NTIME(t(n)^2)$
  - Better proof gives $\neq$ ~~co~~ $NTIME(\omega(t(n+1)))$

    [Won't cover this in course]

- ~~No~~ Lots of other interesting things:

  e.g. if $C_1 \stackrel{c}{\neq} C_2$ ← classes

  then $\exists\, C_{1.5}$ s.t.

  $$C_1 \subsetneq C_{1.5} \subsetneq C_2$$

  "as dense as rationals ...". [Ladner's thm]

- Both proofs above "Lazy Diagonalization".