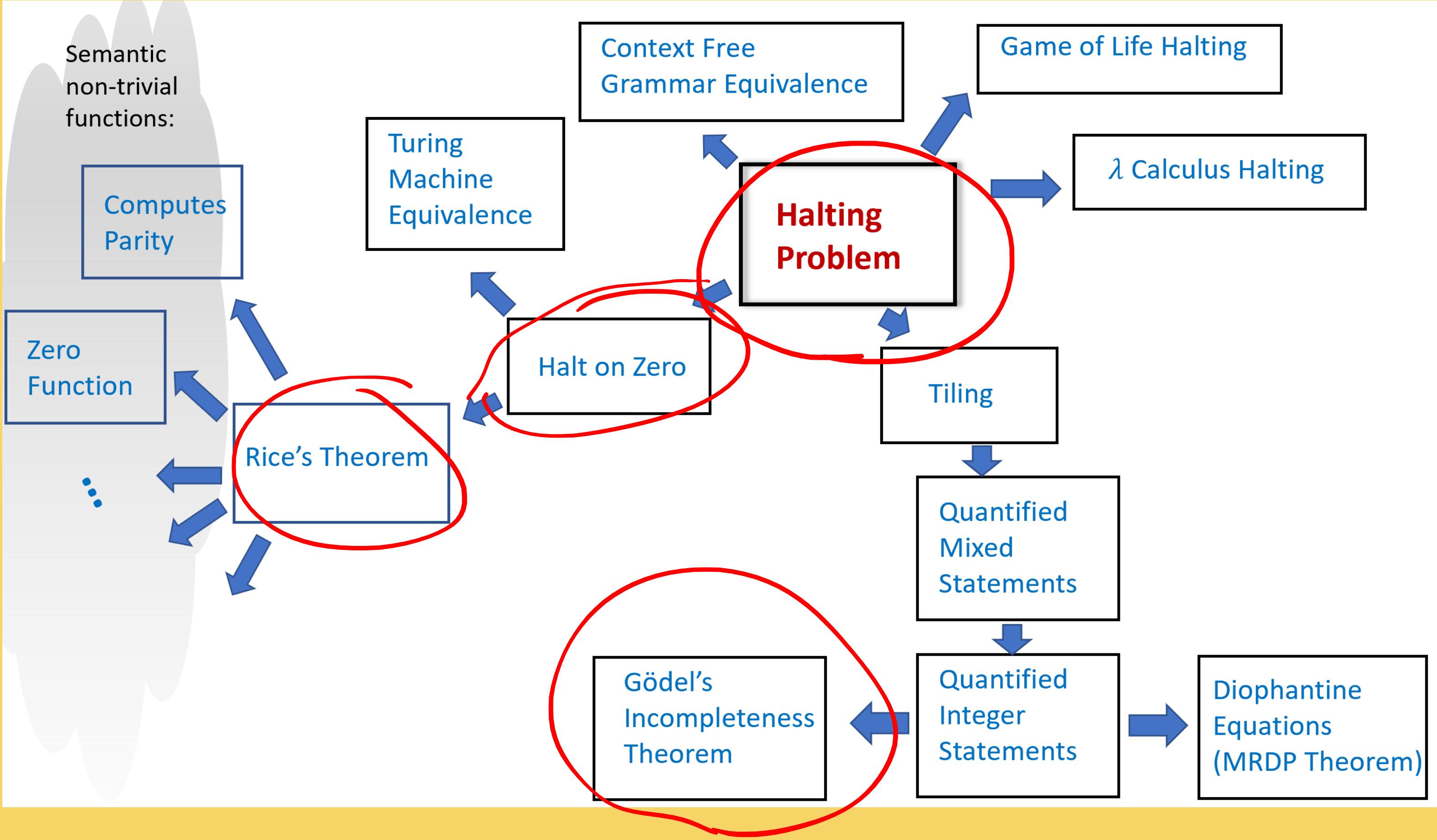
PS10 due next Friday, Apr 25.



Class 23: Complexity, Class P

University of Virginia
cs3120: DMT2
Wei-Kai Lin

PRR 10

Read the textbook example of halting on zero (Sec 9.1.4):
https://introtcs.org/public/lec_08_uncomputability.html#example-halting-on-the-zero-problem

Q1.3

2 Points

Which is correct?

- ○ The section shows that HALT is uncomputable
- ○ The section shows that HALTONZERO is uncomputable
- ◉ Neither is correct
- ○ Both are correct

Algorithm B for $HALT$ using A.

Hypothetical Algorithm A for $HALTONZERO$

$N_{M,x}$

$A(N_{M,x})$

Input: TM M, string x

Operation:

1. Write code of TM $N_{M,x}$:
`"Ignore input and run` $M(x)$"

2. Return $A(N_{M,x})$

Michael Sipser: *"If pigs could whistle then horses could fly"*.

Recap: Semantic Property

Two machines M_1, M_2 are **functionally equivalent** (denoted $M_1 \equiv M_2$) if $\forall x \in \{0,1\}^*, M_1(x) = M_2(x)$

A function $F: \{0,1\}^* \to \{0,1\}$, defined on Turin machines, is **semantic** if for every pair of functionally equivalent TMs (M_1, M_2), $F(M_1) = F(M_2)$.

Namely, this is a property of the **function**/language of the machine, not of the **behavior** of the machines

Recap: Rice's Theorem

If F is a sematic property, then either F is **not computable**, or F is trivial

Trivial: for all Turing machine M,

$$F(M) = 0$$
$$F(M) = 1$$

Complexity:
the “cost” of computation

How have we considered “reasonable cost” so far?

Cost-Sensitive Computing

Model of Computation	What to Count	Complexity Classes and Results
Boolean Circuits / NAND-CIRC programs	# gates	
DFA / NFA / Regexp	len of regexp or # states of DFA	
? (Algorithms course)	time complexity # basic ops (eg. # comparisons)	
Turing Machines		

Cost-Sensitive Computing

Model of Computation	What to Count	Complexity Classes and Results
Boolean Circuits / NAND-CIRC programs	Gates in Circuit Lines in NAND-CIRC program	$SIZE_{n,m}(s)$ Size Hierarchy Theorem
DFA / NFA / Regexp	Number of States Length of Regexp	NFA with n states $\sim$ DFA with 2^n
? (Algorithms course)	Constant time-ish operations (?)	$Sorting \in \Theta(n \log n)$ (number of comparisons to comparison-sort n items)
Turing Machines	# transit during $M(x)$ len of M	$= TIME$

Costs of Turing Machines

Length of description ("Kolomolgorov Complexity", Class 21) $|M|$

Amount of tape needed ("Space Complexity")

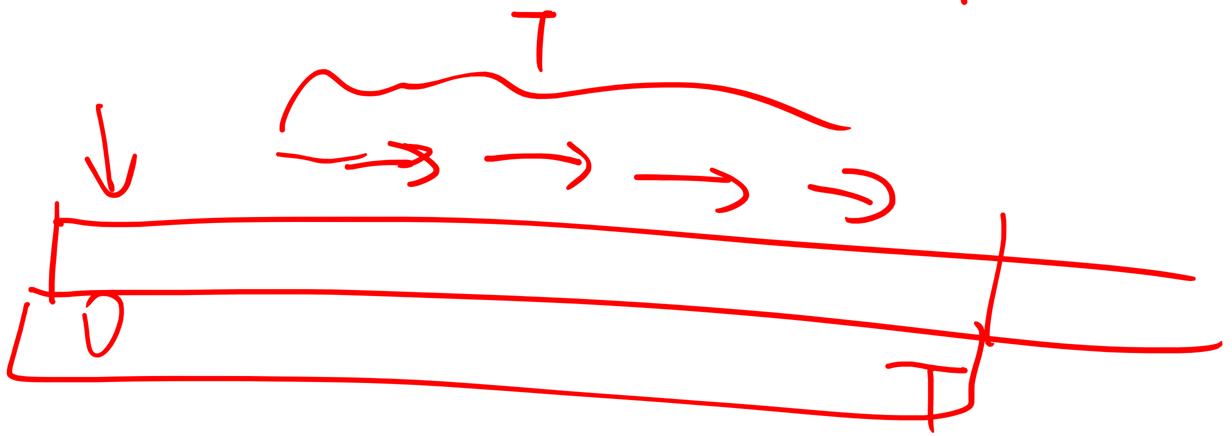
Number of steps before halting ("Running Time Complexity") TIME / transition

Relating Time and Space

If a TM running on an n-length input executes in $23(n^2)!$ steps, what can we say about its space complexity?

Claim: TM runs in time T

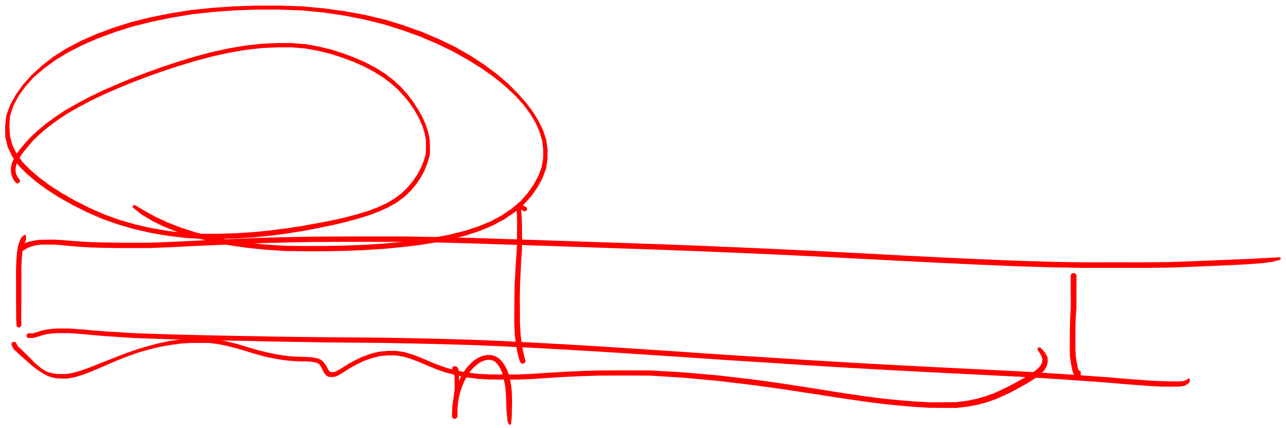
$\Rightarrow$ uses space T



Relating Space and Time

If a TM running on an n-length input executes in $\Theta(n)$ space, what can we say about its running time?

No. If additionally, TM halts



17

Space bounded computation

PLAN: Example of space bounded algorithms, importance of preserving space. The classes L and PSPACE, space hierarchy theorem, PSPACE=NPSPACE, constant space = regular languages.

We will not talk more about space complexity, but it is related to many topics in this course.

Defining TIME Complexity Classes

$TIME_{\mathsf{TM}}(T(n))$ is the set of Boolean functions for which a Turing Machine M exists such that M halts after at most $T(n)$ steps for all n-bit input and M computes the function.

Definition 13.1 (Running time (Turing Machines))

Let $T : \mathbb{N} \to \mathbb{N}$ be some function mapping natural numbers to natural numbers. We say that a function $F : \{0,1\}^* \to \{0,1\}^*$ is *computable in* $T(n)$ *Turing-Machine time (TM-time for short)* if there exists a Turing machine M such that for every sufficiently large n and every $x \in \{0,1\}^n$, when given input x, the machine M halts after executing at most $T(n)$ steps and outputs $F(x)$.

We define $TIME_{\mathsf{TM}}(T(n))$ to be the set of Boolean functions (functions mapping $\{0,1\}^*$ to $\{0,1\}$) that are computable in $T(n)$ TM time.

$TIME_{\mathsf{TM}}(T(n))$ is the set of Boolean functions for which a Turing Machine M exists such that M halts after at most $T(n)$ steps and M computes the function.

Boolean functions: $\{0,1\}^* \rightarrow \{0,1\}$

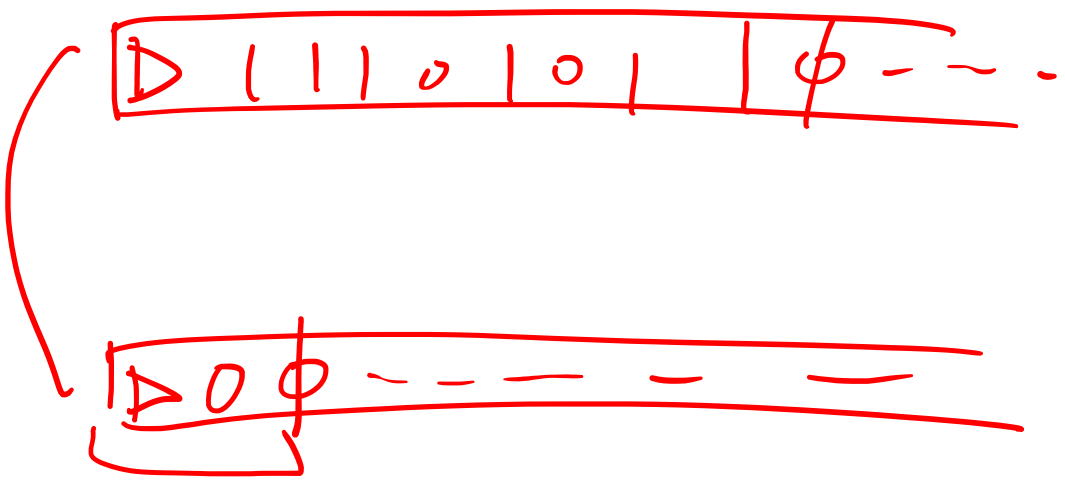
What functions are in $TIME_{\mathsf{TM}}(28)$?

$F(x) = 0$ $\qquad$ $F \in$

TM M computes F on every x . . .

$M(x)$:

~~output~~ ret 0



$TIME_{\mathrm{TM}}(T(n))$ is the set of Boolean functions for which a Turing Machine M exists such that M halts after at most $T(n)$ steps and M computes the function.

Boolean functions: $\{0,1\}^* \rightarrow \{0,1\}$

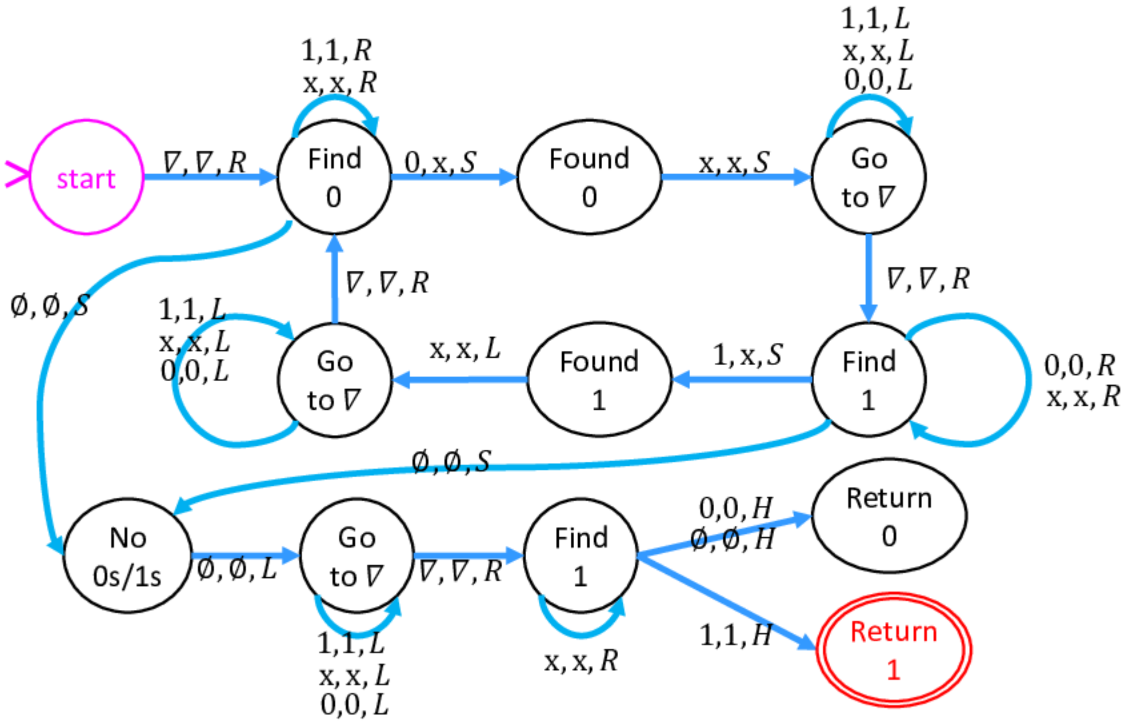
What functions are in $TIME_{\mathrm{TM}}(28)$?

Definition. The execution of a TM, $M = (\Sigma, k, \delta)$ on input $x \in \{0,1\}^*$ is this process:

1. Initialize T as $\triangleright, x_0, x_1, \ldots, x_{|x|-1}, \emptyset, \emptyset, \ldots.$
2. Initialize natural number variables, $i = 0, s = 0$.
3. **repeat**
    1. $(s', \sigma', D) = \boldsymbol{\delta}(s, T[i])$
    2. $s := s', T[i] := \sigma'$
    3. if D = $\mathbf{R}$: $i := i + 1$
       if D = $\mathbf{L}$: $i := \max\{i-1, 0\}$
       if D = $\mathbf{H}$: $\mathbf{break}$
4. If the process finishes, the output is $M(x) = T[1], \ldots, T[m]$ where $m > 0$ is the smallest integer, $T[m+1] \notin \{0,1\}$. Otherwise, $M(x) = \perp$.

$TIME_{\mathsf{TM}}(T(n))$ is the set of Boolean functions for which a Turing Machine M exists such that M halts after at most $T(n)$ steps and M computes the function.

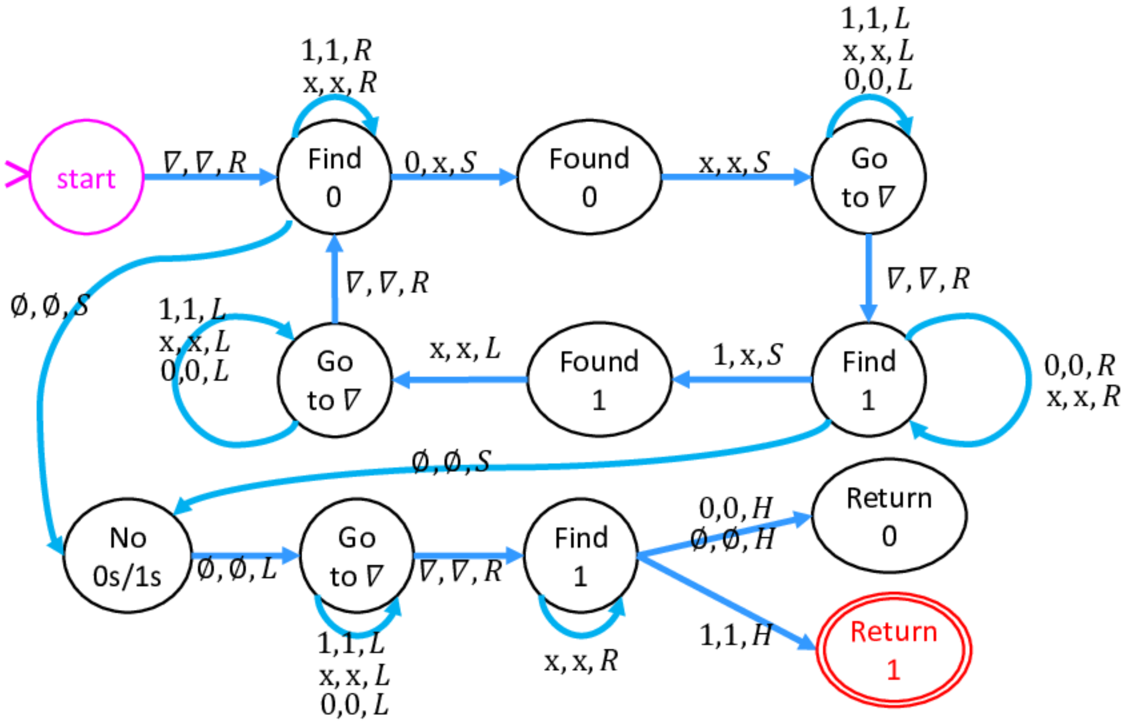
Is the MAJ machine in $TIME_{\mathsf{TM}}(17n^4)$?



MAJ machine from PS8

$TIME_{\mathsf{TM}}(T(n))$ is the set of Boolean functions for which a Turing Machine M exists such that M halts after at most $T(n)$ steps and M computes the function.

Is the MAJ machine in $TIME_{\mathsf{TM}}(17n^4)$?



MAJ machine from PS8

$\in EVEN$?

No, but this is a trick question like asking if there is an Orange in the set.

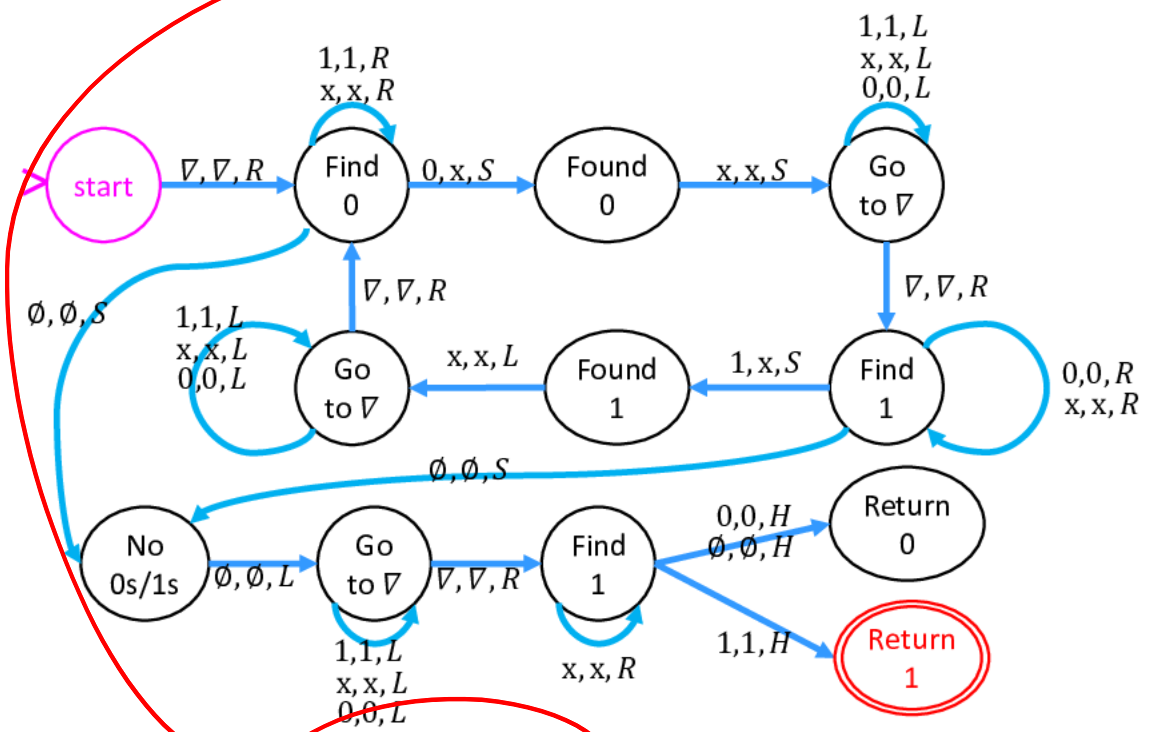
$TIME_{\mathsf{TM}}(17n^4)$ is a set of **Boolean functions**. It is about whether there exists a TM that can compute a function within $17n^4$ steps (where n is the size of the input).

$TIME_{\mathrm{TM}}(T(n))$ is the set of Boolean functions for which a Turing Machine M exists such that M halts after at most $T(n)$ steps and M computes the function.

Is the MAJ function in $TIME_{\mathrm{TM}}(17n^4)$?

$TIME_{\mathsf{TM}}(T(n))$ is the set of Boolean functions for which a Turing Machine M exists such that M halts after at most $T(n)$ steps and M computes the function.

Is the MAJ function in $TIME_{\mathsf{TM}}(17n^4)$?



MAJ machine from PS7

→ runs $17n^4$ $\Theta(n), \Theta(n^2)$

"Not yet" – if this machine can compute it for all length n inputs in $17n^4$ steps (where n is the size of the input) that would be a proof that MAJ $\in TIME_{\mathsf{TM}}(17n^4)$

$\leq 17n^2$

Class P

Definition: Complexity Class: P

A class for “Polynomial Time”

Class **P** = $\bigcup_{c \in \mathbb{N}} TIME_{TM}(n^c)$

1

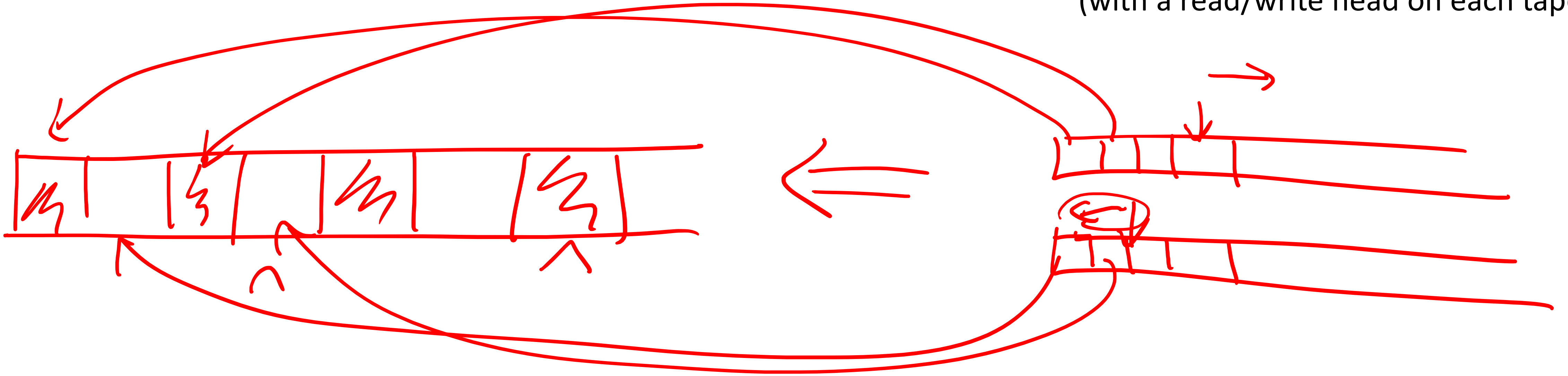
Note: TCS book definition is **P** = $\bigcup_{c \in \{1,2,3,\ldots\}} TIME_{TM}(n^c)$

P is Robust to Machine Definitions

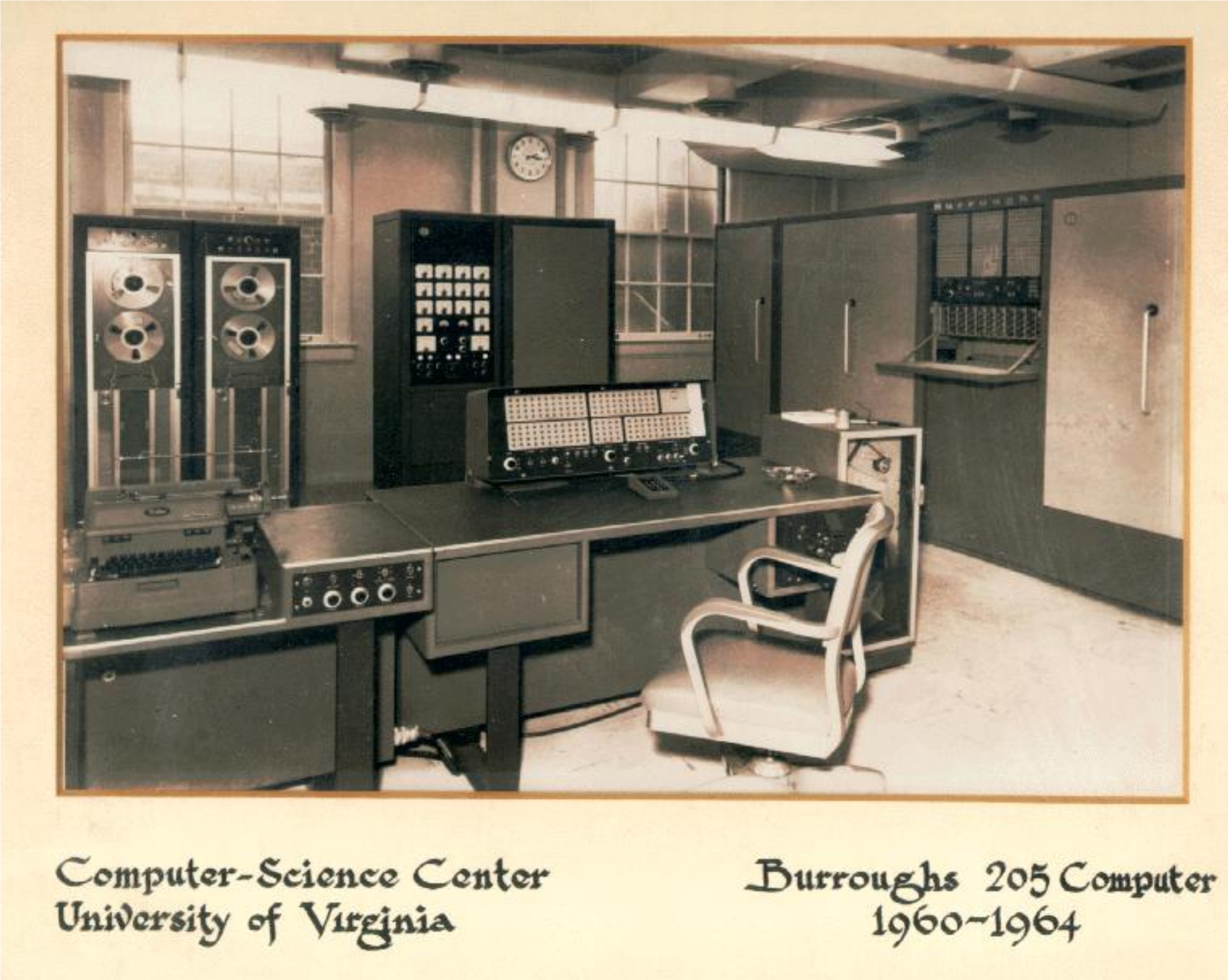
$\mathbf{P}_{TM} = \bigcup_{c \in \mathbb{N}} TIME_{TM}(n^c)$ One-way infinite tape TM

$\mathbf{P}_{TM2w} = \bigcup_{c \in \mathbb{N}} TIME_{TM2w}(n^c)$ Two-way infinite tape TM

$\mathbf{P}_{TM\times 2} = \bigcup_{c \in \mathbb{N}} TIME_{TM\times 2}(n^c)$ Two-tape TM
(with a read/write head on each tape)

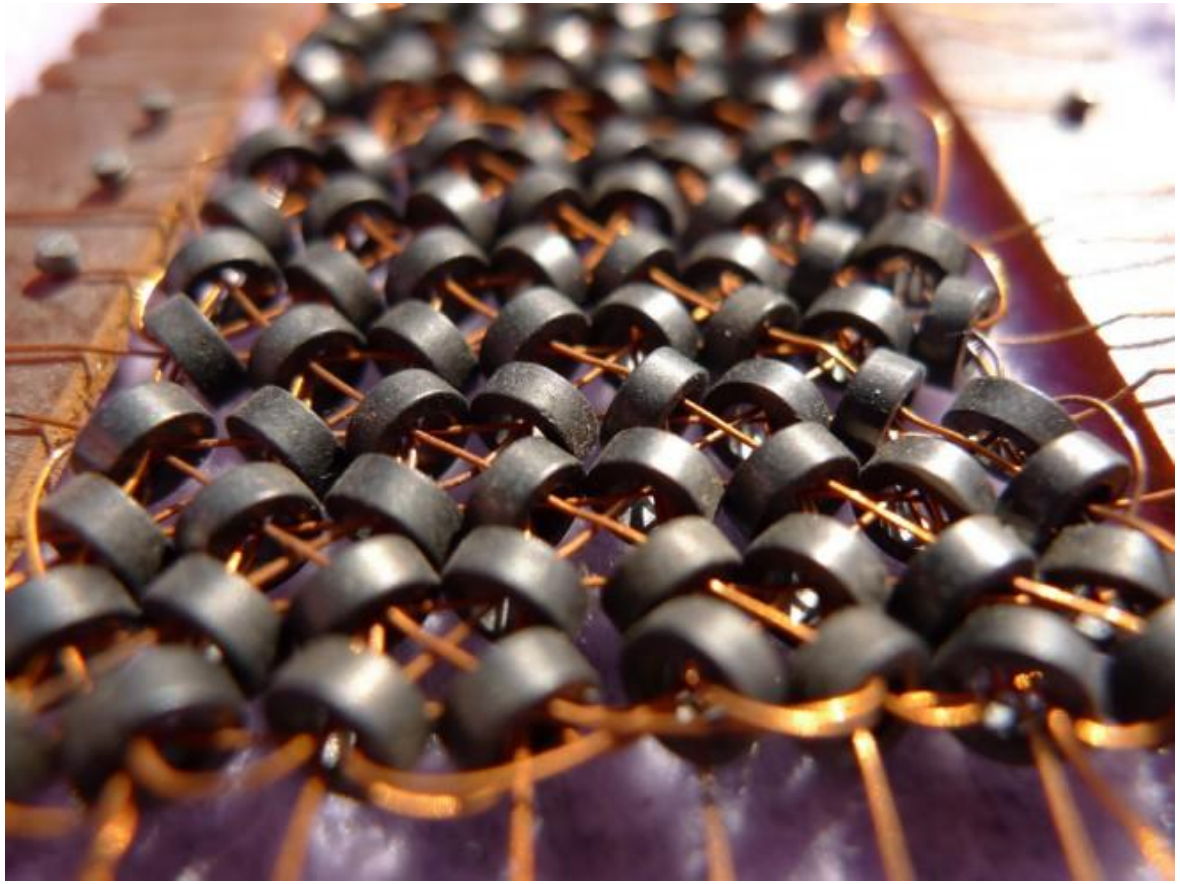


Tape-Based Memory

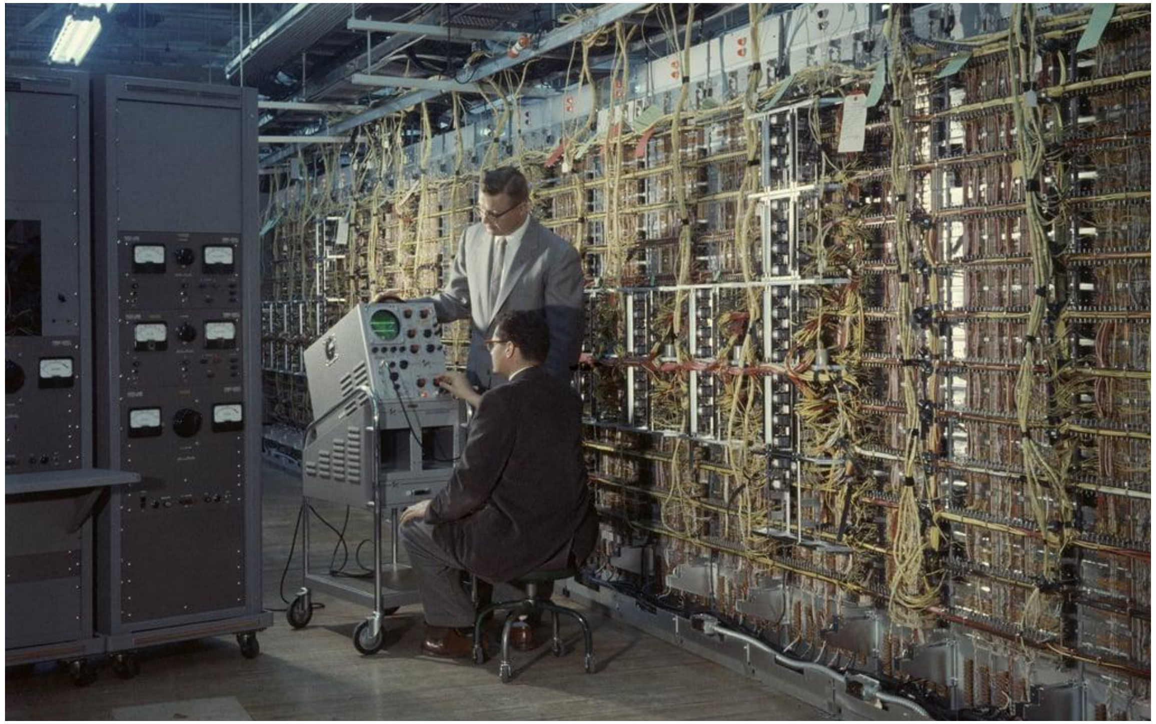


A tape was a reasonable memory model for practical computers until memories got big

Random-Access Memory



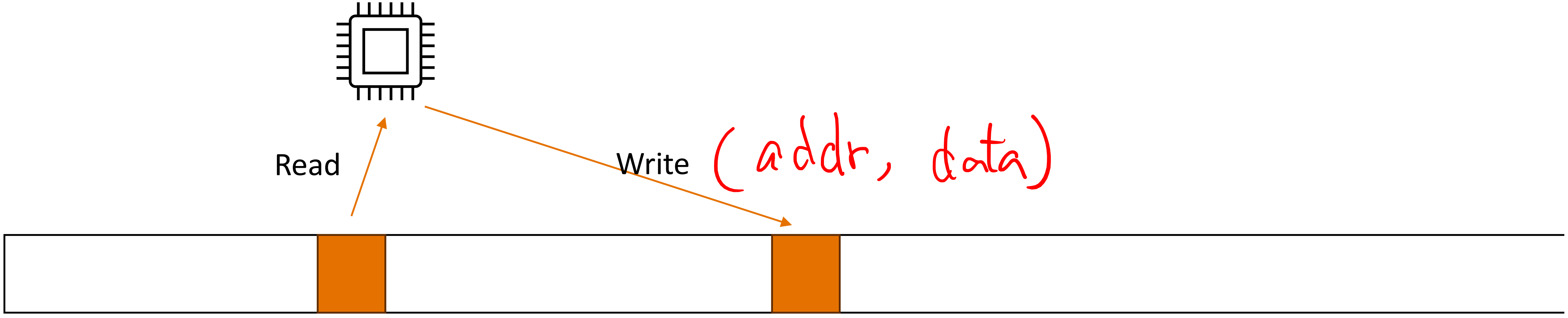
Magnetic Core Memory (Jay Forrester, 1951)
MIT tried to license patent for 2 cents/bit!



Semi-Automatic Ground Environment
FSQ-7 memory: 65,536 (2^{16}) 32-bit words

Random-Access Memory (RAM) Machines

- **Unbounded memory array**, can be read and written using natural number indices
- **CPU:**
  1. read or write an array "word" by index/address
  2. compute any function on constant number of "words" (registers)



Is a RAM machine equivalent (in computing power) to a TM?

RAM Machine Complexity

- **Unbounded memory array**, can be read and written using natural number indices
- **CPU:**
  1. read or write an array “word” by index/address
  2. compute any function on constant number of “words” (registers)

$$\mathbf{P}_{\mathrm{RAM}} = \bigcup_{c\in\mathbb{N}} TIME_{\mathrm{RAM}}(n^c)$$

Is $\mathbf{P}_{\mathrm{RAM}} = \mathbf{P}_{\mathrm{TM}}$*?*

$\mathbf{P}_{\mathrm{RAM}} = \bigcup_{c \in \mathbb{N}} TIME_{\mathrm{RAM}}(n^c)$

$$\mathbf{P}_{\mathrm{RAM}} = \mathbf{P}_{\mathrm{TM}\times 4} = \mathbf{P}_{\mathrm{TM}}$$

We can simulate RAM memory with a multi-tape TM:

- Have a second tape to keep track of current location (increment each time we move right, decrement for left)
- Have a third tape to record the target location
- Move until the two tapes match (maybe need another tape?)

Details not important (only impact c) – but if you don't like this sketchy argument see TCS 8.2 ("The gory details (optional)")

- Whatever a RAM machine can compute in $T(n)$ steps, a tape TM can compute $T(n)^4$ time (Theorem 13.5)

We expect you to understand the crux of this argument, but not the details which we won't cover

P is Robust to Machine Definitions

$\mathbf{P}_{TM} = \bigcup_{c\in\mathbb{N}} TIME_{TM}(n^c)$ — One-way infinite tape TM

$\mathbf{P}_{TM2w} = \bigcup_{c\in\mathbb{N}} TIME_{TM2w}(n^c)$ — Two-way infinite tape TM

$\mathbf{P}_{\mathrm{RAM}} = \bigcup_{c\in\mathbb{N}} TIME_{\mathrm{RAM}}(n^c)$ — **RAM** machine

$$\mathbf{P}_{TM} = \mathbf{P}_{TM2w} = \mathbf{P}_{\mathrm{RAM}}$$

P is Robust to Machine Definitions

$\mathbf{P}_{TM} = \bigcup_{c\in\mathbb{N}} TIME_{TM}(n^c)$ — One-way infinite tape TM

$\mathbf{P}_{TM2w} = \bigcup_{c\in\mathbb{N}} TIME_{TM2w}(n^c)$ — Two-way infinite tape TM

$\mathbf{P}_{\mathrm{RAM}} = \bigcup_{c\in\mathbb{N}} TIME_{\mathrm{RAM}}(n^c)$ — **RAM** machine

$\mathbf{P}_{\mathrm{Python}} = \bigcup_{c\in\mathbb{N}} TIME_{\mathrm{Python}}(n^c)$ — Idealized **Python** interpreter

ops + * bit ops if loop

$$\mathbf{P}_{TM} = \mathbf{P}_{TM2w} = \mathbf{P}_{\mathrm{RAM}} \stackrel{?}{=} \mathbf{P}_{\mathrm{Python}}$$

P is Robust to Machine Definitions

$\mathbf{P}_{TM} = \bigcup_{c \in \mathbb{N}} TIME_{TM}(n^c)$ One-way infinite tape TM

$\mathbf{P}_{TM2w} = \bigcup_{c \in \mathbb{N}} TIME_{TM2w}(n^c)$ Two-way infinite tape TM

$\mathbf{P}_{\mathrm{RAM}} = \bigcup_{c \in \mathbb{N}} TIME_{\mathrm{RAM}}(n^c)$ **RAM** machine

$\mathbf{P}_{\mathrm{Python}} = \bigcup_{c \in \mathbb{N}} TIME_{\mathrm{Python}}(n^c)$ Idealized **Python** interpreter

$$\mathbf{P}_{TM} = \mathbf{P}_{TM2w} = \mathbf{P}_{\mathrm{RAM}} = \mathbf{P}_{\mathrm{Python}}$$

P is Robust to (Most) Machine Definitions

$\mathbf{P}_{TM} = \cup_{c\in\mathbb{N}} TIME_{TM}(n^c)$ One-way infinite tape TM

$\mathbf{P}_{TM2w} = \cup_{c\in\mathbb{N}} TIME_{TM2w}(n^c)$ Two-way infinite tape TM

$\mathbf{P}_{\mathrm{RAM}} = \cup_{c\in\mathbb{N}} TIME_{\mathrm{RAM}}(n^c)$

RAM machine
- READ(i), WRITE(i, val), ADD(a,b), MULT(a,b)
- JUMP(k)

$\mathbf{P}_{\mathrm{Python}} = \cup_{c\in\mathbb{N}} TIME_{\mathrm{Python}}(n^c)$ Idealized **Python** interpreter

$\mathbf{P}_{\mathrm{QC}} = \cup_{c\in\mathbb{N}} TIME_{\mathrm{QC}}(n^c)$ Quantum Computer

$$\mathbf{P}_{TM} = \mathbf{P}_{TM2w} = \mathbf{P}_{\mathrm{RAM}} = \mathbf{P}_{\mathrm{Python}} \stackrel{?}{=} \mathbf{P}_{\mathrm{QC}}$$

Functions in P? Boolean Func

$$\mathbf{P}_{TM} = \cup_{c\in\mathbb{N}} TIME_{TM}(n^c)$$
$$= \mathbf{P}_{\text{Python}} = \cup_{c\in\mathbb{N}} TIME_{\text{Python}}(n^c)$$

SORTING

Input: A list of n natural numbers, $x_1, x_2, x_3, \ldots, x_n$. # bits

Output: An ordering of the input list, $x_{i_1}, x_{i_2}, \ldots, x_{i_n}$ where $\{i_1\} \cup \{i_2\} \cup \ldots \{i_n\} = \{1, 2, \ldots, n\}$ and for all $k \in \{1, 2, \ldots, n-1\}$, $x_{i_k} \leq x_{i_{k+1}}$.

YES $O(n \log n)$ # ops/comparison

Functions in P?

$$\mathbf{P}_{TM} = \bigcup_{c\in\mathbb{N}} TIME_{TM}(n^c)$$
$$= \mathbf{P}_{\text{Python}} = \bigcup_{c\in\mathbb{N}} TIME_{\text{Python}}(n^c)$$

$$HALTS(w,x) = \begin{cases} 1, & \text{if } TM_w \text{ terminates on } x \\ 0, & otherwise \end{cases}$$

Functions not in P

$$\mathbf{P}_{TM} = \bigcup_{c\in\mathbb{N}} TIME_{TM}(n^c)$$
$$= \mathbf{P}_{\text{Python}} = \bigcup_{c\in\mathbb{N}} TIME_{\text{Python}}(n^c)$$

$$HALTS(w,x) = \begin{cases} 1, & \text{if } TM_w \text{ terminates on } x \\ 0, & \textit{otherwise} \end{cases}$$

No TM M comp H in Time(n^c)

$$HALTS \notin \mathsf{P}$$

$TIME_{\mathrm{TM}}(T(n))$ is the set of Boolean functions for which a Turing Machine M exists such that M computes the function and M halts after at most $T(n)$ steps.

Functions in P?

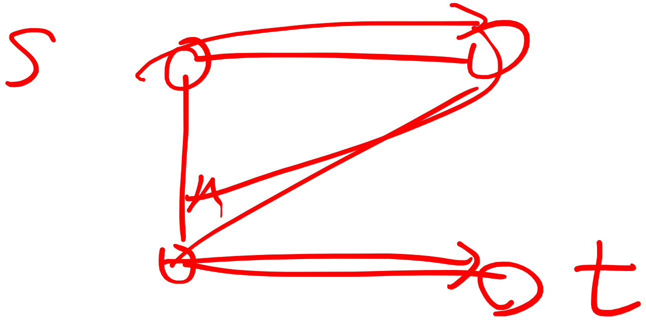
~~$F(G,s,t)$~~

LongestPath

Input: A finite graph $G = (V, E)$, two vertices, $s, t \in V$, and a path length, $\ell \in \mathbb{N}$.

Output: If there is a simple path from s to t in G of length at least ℓ, 1. Otherwise, 0.

$F(G, s, t, \ell) = 1$ iff



$, \ \ell = 3$

Definition: a *simple path* in a graph $G = (V, E)$ from $s, t \in V$ is a path from s to t where no node is repeated.

Functions in P?

LongestPath

Input: A finite graph $G = (V, E)$, two vertices, $s, t \in V$, and a path length, $\ell \in \mathbb{N}$.
Output: If there is a simple path from s to t in G of length at least ℓ, 1. Otherwise, 0.

LongestPath is **not known** to be in **P**: it might be, it might not be. (We'll see a more about this in future classes…)

Definition: a *simple path* in a graph $G = (V, E)$ from $s, t \in V$ is a path from s to t where no node is repeated.

“Exponential Time”: EXP

P = $\bigcup_{c\in\mathbb{N}} TIME_{TM}(n^c)$

EXP = $\bigcup_{c\in\mathbb{N}} TIME_{TM}(2^{n^c})$

input size

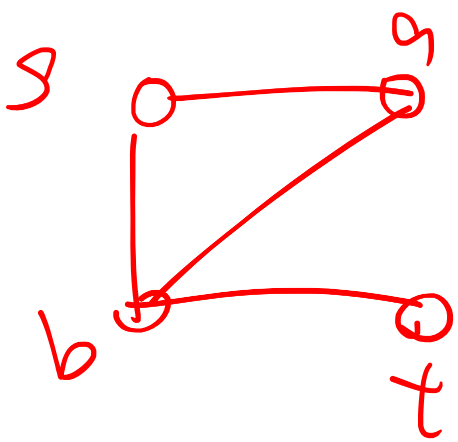
Is *LongestPath* in EXP?

P = $\bigcup_{c\in\mathbb{N}} TIME_{TM}(n^c)$ $\subsetneq$ **EXP** = $\bigcup_{c\in\mathbb{N}} TIME_{TM}(2^{n^c})$

LongestPath

Input: A finite graph $G = (V, E)$, two vertices, $s, t \in V$, and a path length, $\ell \in \mathbb{N}$.

Output: If there is a simple path from s to t in G of length at least ℓ, 1. Otherwise, 0.



$\ell = 3$,

$|V| \leq n$

$n^n \leq 2^{n \cdot \log n} \leq 2^{n^2}$

s, t, a, b ×

s, b, a, t ×

s, a, b, t ✓ ✓