CS 121: Lecture 11
More on Turing Machines

Madhu Sudan

https://madhu.seas.Harvard.edu/courses/Fall2020

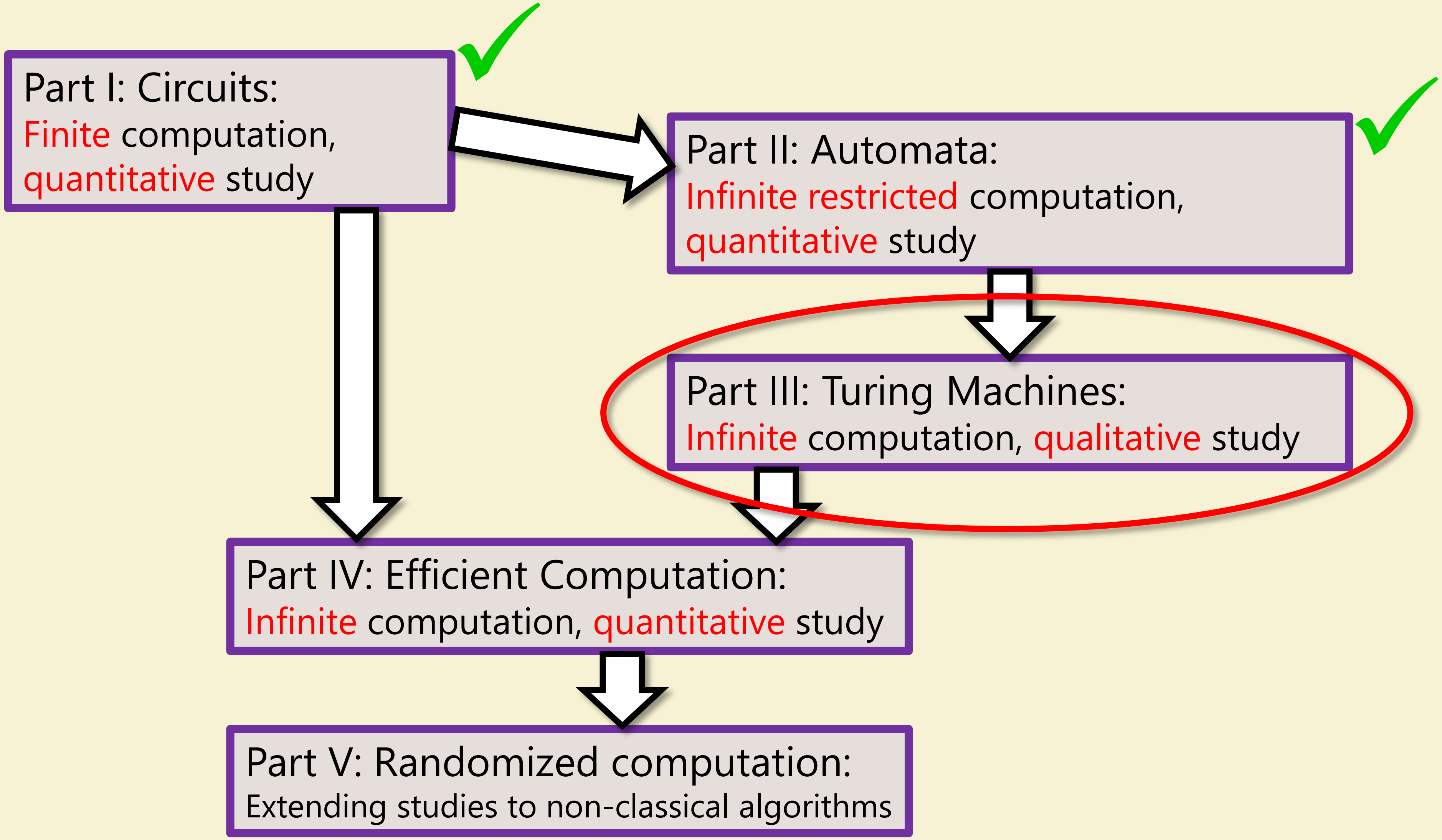
Book: https://introtcs.org

How to contact us:
- The whole staff (faster response): CS 121 Piazza
- Only the course heads (slower): cs121.fall2020.course.heads@gmail.com

Announcements:

- Advanced Sections: Christina Ilvento on Differential Privacy!
- Homework 3 due today.
- Sample midterm available for tech/TeX/rules.
- Actual Midterm:
  - Pick up on Canvas;
  - TeX your answers ;
  - Submit on Gradescope-submit your answers like a problem set.
- Section: no video this week; review for midterm.
  - Section on Turing Machines: next week.
- Midterm review materials:
  - Diego/Joanna’s handout
  - Past midterms: two on finite automata without solutions; several from Boaz with solutions.

Where we are:


Part I: Circuits:
Finite computation, quantitative study
Part II: Automata:
Infinite restricted computation, quantitative study
Part III: Turing Machines:
Infinite computation, qualitative study
Part IV: Efficient Computation:
Infinite computation, quantitative study
Part V: Randomized computation:
Extending studies to non-classical algorithms

Today:

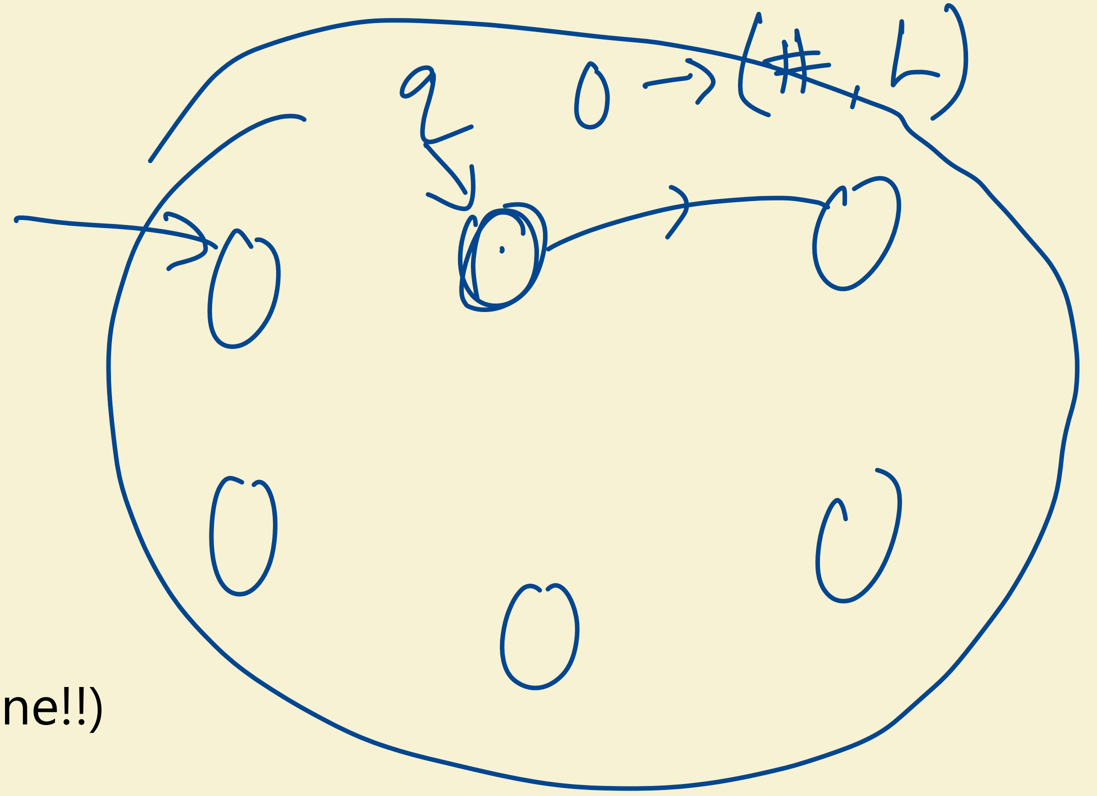
- Part 1: More examples of Turing Machines
  - TM to compute $PAL: \{0,1\}^* \to \{0,1\}$ where $PAL(x) = 1 \Leftrightarrow x = x^R$
  - TM to compute $h: \{0,1\}^* \to \{0,1\}^*$, where $h(x) = y$ where $x = yz$ and $|y| \in \{|z|, |z| + 1\}$
- Part 2: (Discussion) Looking to the future:
  - Computable functions.
    - Def (7.2 in Barak): Function computable ⇔ computable by TM
  - Equivalence with other computing & non-computing models: Multiple tapes, RAM, λ-calculus, polynomials …

Recall Turing Machines

- (Barak, Definition 7.1):
- TM on k states and alphabet $\Sigma \supseteq \{0,1,\triangleright,\phi\}$

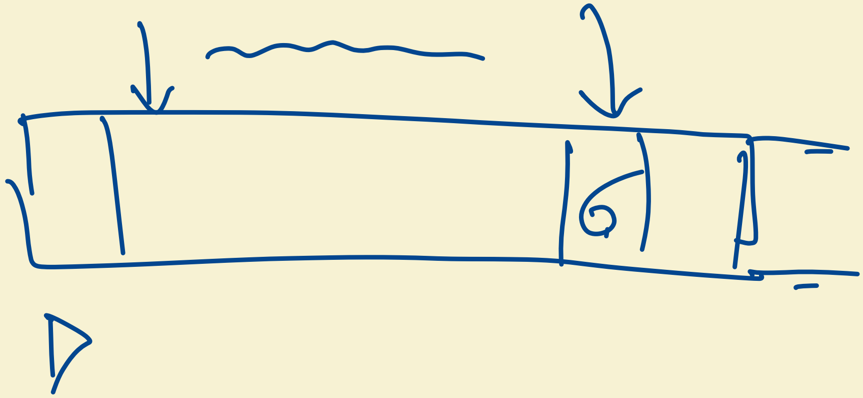
  is given by $\delta: [k] \times \Sigma \to [k] \times \Sigma \times \text{Action}$,

  where $\text{Action} = \{L, R, S, H\}$
  - L=Left, R=Right, S=Stay (don't move), H=Halt (done!!)



- Operation:
  - Start in state 0, Tape $T = \triangleright x_0 \ldots x_{n-1}\phi\phi\phi \ldots$, Head (i) at x_0
  - General step: current state q ; input symbol σ:

    Let $\delta(q,\sigma) = (r,\tau,X) \Rightarrow$ Write τ on tape (overwriting σ) ; Move to state r;
    Move Head left $(i \leftarrow i - 1)$ if $X = L$; right if $X = R$; don't move if $X = S$.
  - Repeat General step until $X = H$



Recognizing Palindromes

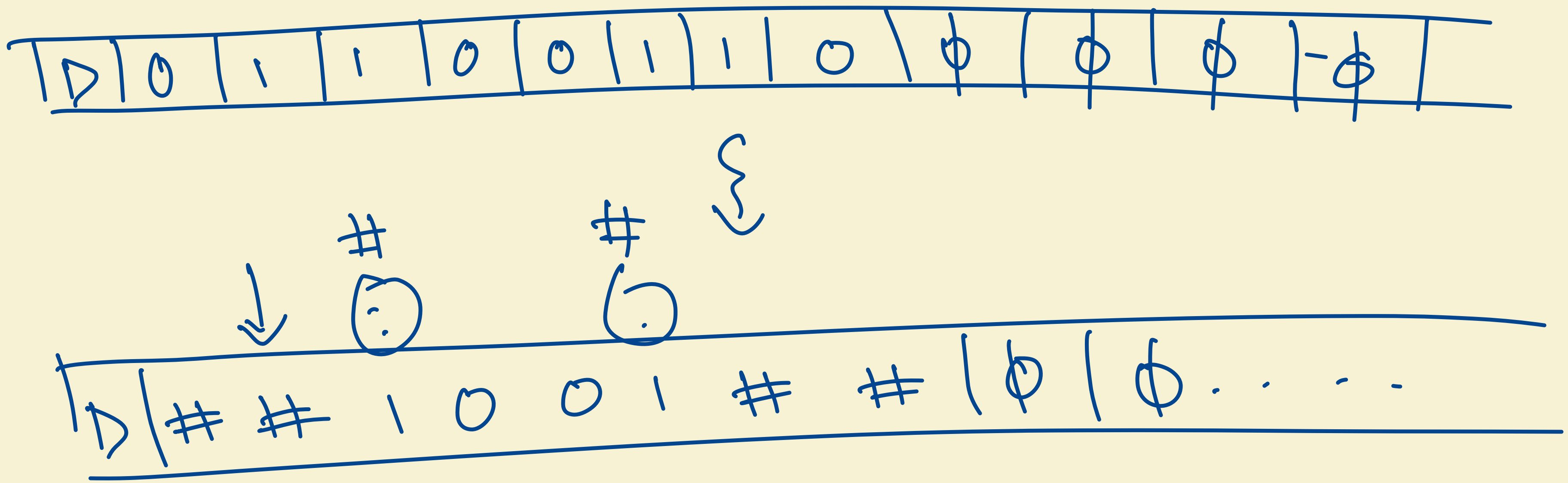
→ LEVEL

→ 0110110

→ 0110

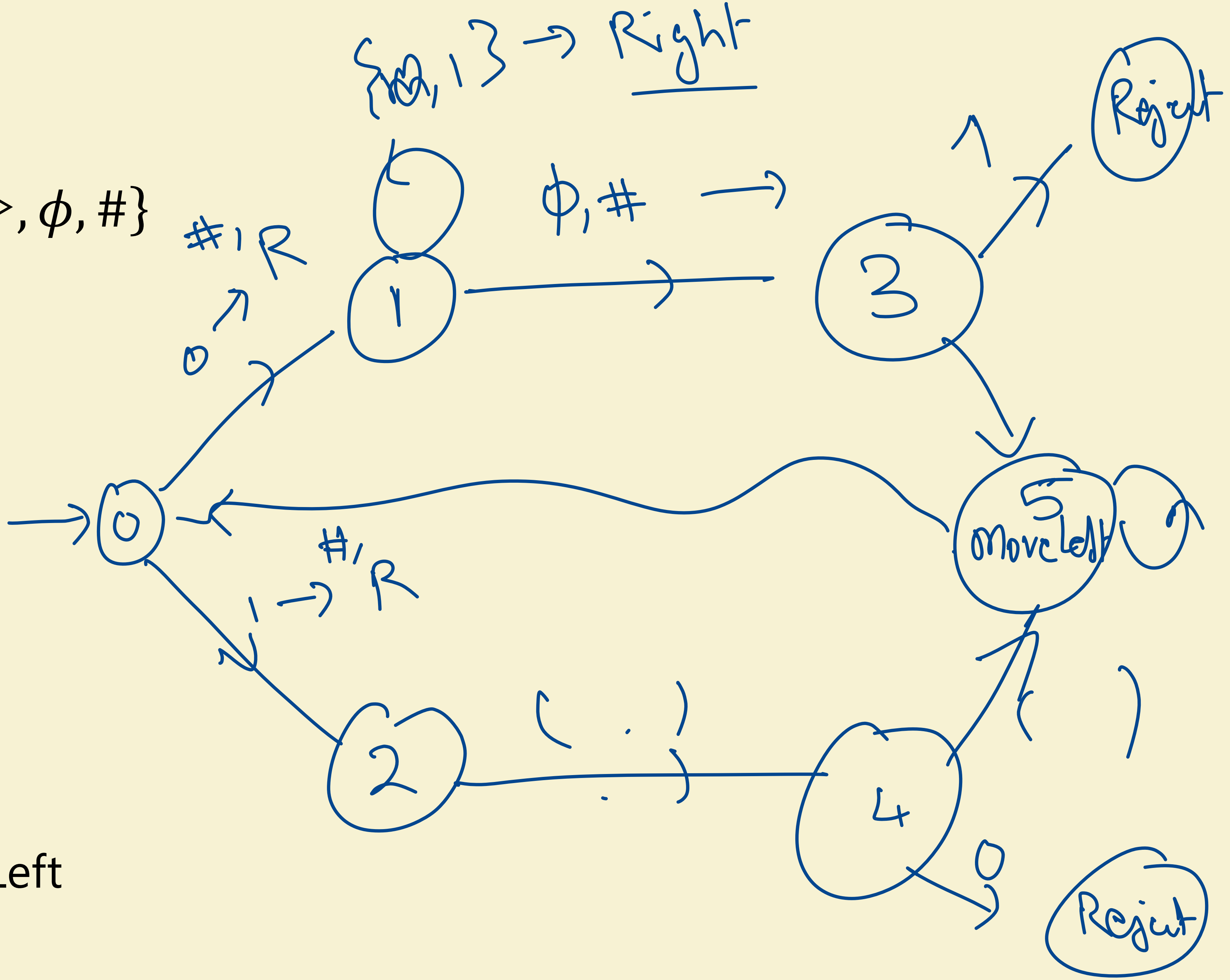
→ 001 ✗

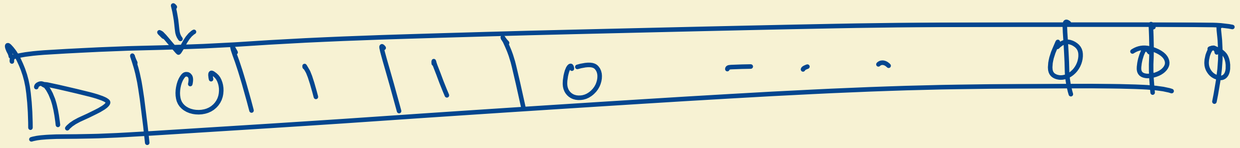
- $PAL: \{0,1\}^* \to \{0,1\}$ where $PAL(x) = 1 \Leftrightarrow x = x^R$
- Overview/Idea:
  - Scan left to right between #s.
  - Replace extreme symbols by # if they match, Reject if they don't
  - Till middle region is empty.



More details:

- Alphabet: $\Sigma = \{0, 1, \triangleright, \phi, \#\}$
- States:
  - 0: Start
  - 1: Scan Right 0
  - 2: Scan Right 1
  - 3: Check 0
  - 4: Check 1
  - 5: Move Left
  - 6: Accept and Halt
  - 7: Reject and Clean Left





State/Input	▷	0	1	ϕ	#
0	invalid	(1,#,R)	(2,#,R)	(-,1,H)	(-,1,H)
1	invalid	(1,0,R)	(1,1,R)	(3,ϕ,L)	(3,#,L)
2	invalid				
3	invalid	(5,#,L)	(7,0,L)		
4	invalid				
5				invalid	
6	invalid				
7				invalid	

Alphabet: $\Sigma = \{0,1,\triangleright,\phi,\#\}$

States:

0: Start

1: Scan Right 0

2: Scan Right 1

3: Check 0

4: Check 1

5: Move Left

6: Accept and Halt

7: Reject and Clean Left

Alphabet: $\Sigma = \{0, 1, \triangleright, \phi, \#\}$

States:

0: Start

1: Scan Right 0

2: Scan Right 1

3: Check 0

4: Check 1

5: Move Left

6: Accept and Halt

7: Reject and Clean Left

State/Input	$\triangleright$	0	1	ϕ	#
0	invalid	(1,#,R)	(2,#,R)	(6,1,H)	(6,1,H)
1	invalid	(1,0,R)	(1,1,R)	$(3,\phi,L)$	$(3,\cancel{\phi}\,\#,L)$
2	invalid	(2,0,R)	(2,1,R)	$(4,\phi,L)$	$(4,\cancel{\phi}\,\#,L)$
3	invalid	(5,#,L)	(7,0,L)	invalid	(6,1,H)
4	invalid	(7,0,L)	(5,#,L)	invalid	(6,1,H)
5	invalid	(5,0,L)	(5,1,L)	invalid	(0,#,R)
6	invalid	-	-	-	-
7	$(-,\triangleright,H)$	(7,#,L)	(7,#,L)	invalid	(-,-,H)

Alphabet: $\Sigma = \{0, 1, \triangleright, \phi, \#\}$

States:

0: Start

1: Scan Right 0

2: Scan Right 1

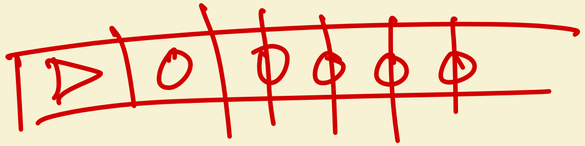
3: Check 0

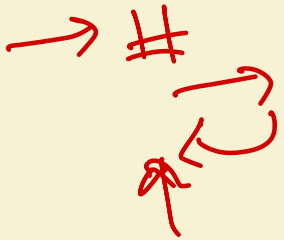
4: Check 1

5: Move Left

6: Accept and Halt

7: Reject and Clean Left





Exercise Break 1

- Design TM to compute $h: \{0,1\}^* \to \{0,1\}^*$, where $h(x) = y$ where $x = yz$ and $|y| \in \{|z|, |z| + 1\}$
  1. Formulate your plan
  2. Break from Break (Return from Break + Discuss Plan)
  3. Choose your alphabet $\{\triangleright, 0, 1, \phi, \#, \#_0, \#_1\}$
  4. Set up the states
  5. Start thinking about key transitions

$x = 0110101$ $y = 0110$ $z = 101$

$x = 000111$ $y = 000$ $z = 111$

One solution:

- Alphabet: $\Sigma = \{0,1,\triangleright,\phi,\#,\#_0,\#_1\}$
- States:
  - 0: Start: Replace b by $\#_b$
  - 1: Zig Right
  - 2. Erase last symbol
  - 3. Zag Left

0 1 1 0 0

$\#_0$ $\#_1$ $\#_1$ # #

0 1 1 # #

0 1 1 0 0

$\#_0$ 1 1 0 #

0 $\#_1$ 1 # #

0 1 $\#_1$ # #

0 1 1 0 0

1 1 0 # # 0

1 # # # 0 1

Alphabet: $\Sigma = \{0,1,\triangleright,\phi,\#,\#_0,\#_1\}$
States:
0: Start: Replace b by $\#_b$
1: Zig Right
2. Erase last symbol
3. Zag Left, Replace $\#_b$ by b, go to start

State/Input	$\triangleright$	0	1	$\boldsymbol{\phi}$	$\#$	$\#_{\mathbf{0}}$	$\#_{\mathbf{1}}$
0	invalid	(1,$\#_0$,R)	(1,$\#_0$,R)	(-,ϕ,H)	(-,#,H)	invalid	invalid
1	invalid	(1,0,R)	(1,1,R)	(2,ϕ,L)	(2,#,L)	invalid	invalid
2	invalid	(3,#,L)	(3,#,L)	invalid	invalid	(-,0,H)	(-,1,H)
3	invalid	(3,0,L)	(3,1,L)	invalid	invalid	(0,0,R)	(0,1,H)

Computable Functions

- **Definition (7.1 in Barak):** A function $f: \{0,1\}^* \to \{0,1\}^*$ is computable if and only if it is computable by a Turing Machine.
- **Warning:** Definition, not a Theorem!
- **Definition:** $R = \{\, f: \{0,1\}^* \to \{0,1\} \quad | \quad f$ is computable $\}$
  - Why R? ("Recursive")

- **Turing-Church Thesis:** f is computable by a physical process if and only if it is computable (by a Turing Machine).

In following lectures

- Turing Equivalence
  - Turing machines can simulate other Turing Machines
    - With multiple tapes
    - With accept/reject states
    - With 1 tape and multiple heads
  - RAM programs: (Main diff: Can read Tape[i] and then Tape[$3i$+25] in O(1) steps.
  - High-level programs – C++, Python …
  - Rewrite systems; Λ-Calculus ; Hilbert Problem
- **Universal** TMs: TM that takes other TMs as input and runs them!
- **Uncomputability** … the bane of computing.