CS 121: Lecture 10
Turing Machines

Madhu Sudan

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

Book: https://introtcs.org

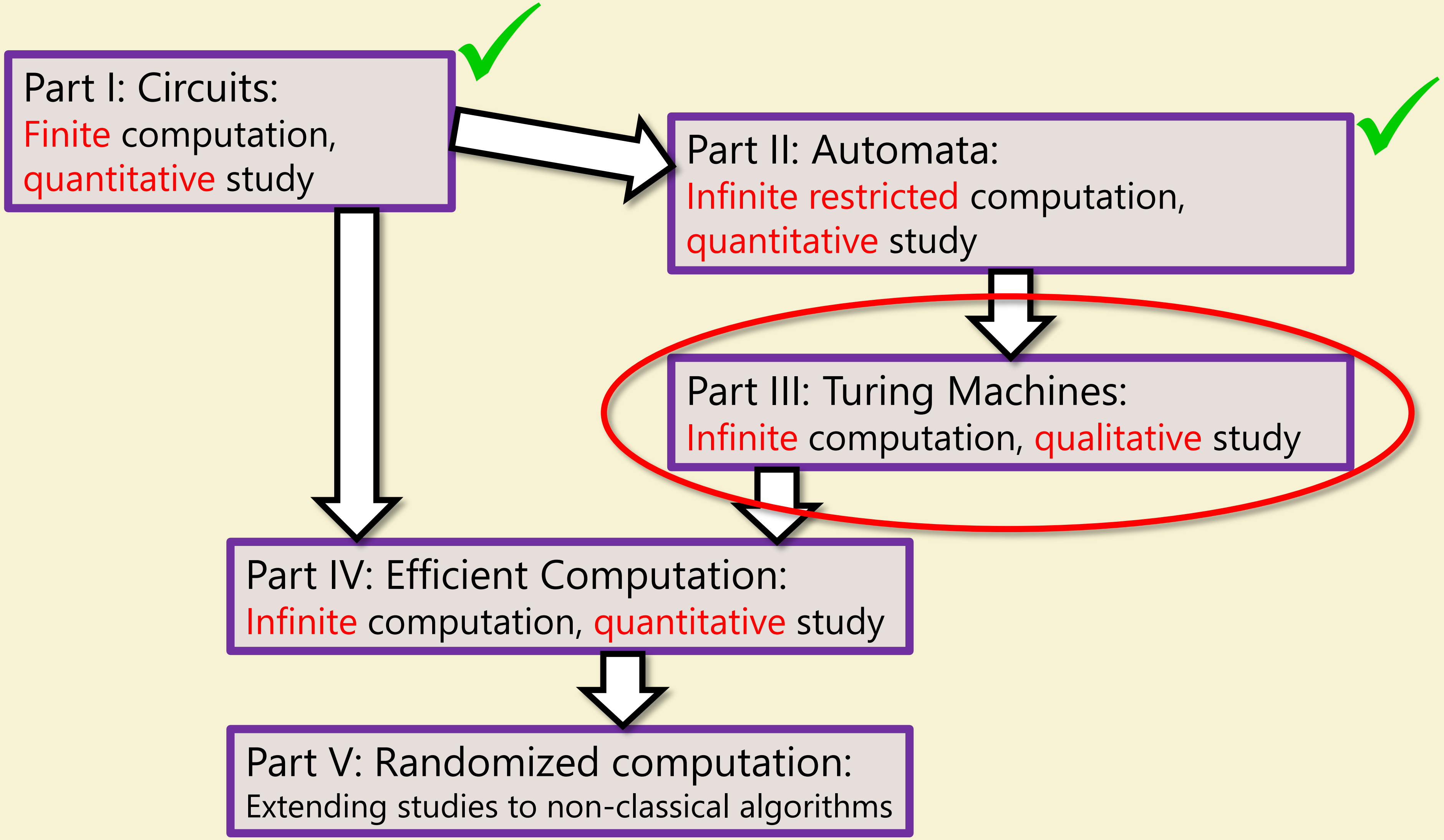
How to contact us
- The whole staff (faster response): CS 121 Piazza
- Only the course heads (slower): cs121.fall2020.course.heads@gmail.com

Announcements:

- Midterm 1 next week:
  - Logistics announcement by Thursday
  - Prep Material: Canvas → Files → Midterm Prep
  - Likely: 2 pages of typset cheatsheet allowed. No other external refs.
- Homework 3 due Thursday

- Advanced Sections: Christina Ilvento on Differential Privacy!

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:

- Definition of Turing Machines
- A function F not computable by DFA or Circuits
- Computing F with Turing Machine

Definition of Turing Machine (TM)

- Recall: DFA = Finite state control + input on tape + move right on each step.
- In a nutshell: TM = DFA + “Write” + “Move left+right on tape”
  - (Either “Write” / “Move left+right” on its own insufficient)

- TM: Main change:
  - More Involved Transition function: T (now δ):
    - δ: (current state, read symbol) $\mapsto$ (new state, write symbol, direction of move/halt)
  - Explicit halting (don’t just end after reading last input bit)
    - Computes functions: output = concatenation of $\{0,1\}$ symbols on tape.

Formal Definition

- (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 $X = H$

TM Example

- Example: $k = 1$; $\Sigma = \{0,1,\triangleright,\phi\}$; $\delta(0,\sigma) = \begin{cases} (0,0,R) \text{ if } \sigma \in \{0,1\} \\ (0,\phi,H) \text{ if } \sigma \notin \{0,1\} \end{cases}$

- What does TM output on $\triangleright 101\phi \ldots$ (in future, we won't write $\triangleright$ or ϕ)

TM Example

- Example: $k = 1$; $\Sigma = \{0,1,\triangleright,\phi\}$; $\delta(0,\sigma) = \begin{cases} (0,0,R) \text{ if } \sigma \in \{0,1\} \\ (0,\phi,H) \text{ if } \sigma \notin \{0,1\} \end{cases}$

- What does TM output on $\triangleright 101\phi$ … (in future, we won't write $\triangleright$ or ϕ)
- What function does TM compute.

A "hard" function

- $f:\{0,1\}^* \rightarrow \{0,1\},\ f(x) = 1 \Leftrightarrow x = 1^n$ for $n = 2^t$ for integer t

Exercise Break 1

- $f: \{0,1\}^* \to \{0,1\},\ f(x) = 1 \Leftrightarrow x = 1^n$ for $n = 2^t$ for integer t
- (30 sec) Prove that no circuit computes f
- (4 min 30 sec) Prove no DFA computes f
  - Part 1: Focus on big idea; defer calculations/parameter settings.
  - Part 2: Get your hands dirty; do calculations+parameter settings.

F is computable by a Turing Machine

Main Idea: Loop many times:

Scan string left to right

Replace every alternate 1 by 0;

reject if number of 1s is odd and greater than 2.

More details: Alphabet & States

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

0. Start/Not seen any ones
1. Move Right first one
2. Move Right even # of ones
3. Move Right odd # of ones
4. Move Left
5. Clean Right and Reject
6. Clean Left and Reject

State/Input	$\triangleright$	0	1	ϕ	$\#$
0					
1					
2					
3					
4					
5					
6					

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

0. Start/Not seen any ones
1. Move Right first one
2. Move Right even # of ones
3. Move Right odd # of ones
4. Move Left
5. Clean Right and Reject
6. Clean Left and Reject

$f: \{0,1\}^* \to \{0,1\},\ f(x) = 1 \Leftrightarrow x = 1^n$ for $n = 2^t$ for integer t

State/Input	$\triangleright$	0	1	ϕ	#
0					
1					
2					
3					
4					
5					
6					

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

0. Start/Not seen any ones
1. Move Right first one
2. Move Right even # of ones
3. Move Right odd # of ones
4. Move Left
5. Clean Right and Reject
6. Clean Left and Reject

Exercise Break 2:

Fill in rows for states 2 & 4

Keep answer ready (5 triples) to type into chat. Use D for $\triangleright$

Summary & Next

- Achieved today:
  - Defined TM
  - Shown it computes one function that DFA and circuits can’t
- Next Lecture:
  - More examples.
  - Towards equivalence with (all) programs