<u>Meeting 3.2</u>: A smattering of complexity

I. Decision problems, counting problems, and computability

II. The usual suspects: P, FP, NP, PSPACE, EXP, # , ER, R, RE

III. Reductions and hardness

I. Decision problems, counting problems, and computability

A decision problem is a function

$$L: \{0,1\}^* = \bigcup_{k \geq 1} \{0,1\}^k \longrightarrow \{\underset{\text{No}}{\underset{\shortparallel}{0}}, \overset{\text{Yes}}{\overset{\shortparallel}{1}}\}$$

A counting problem is a function

$$F: \{0,1\}^* \longrightarrow \{0,1\}^* \text{ "}= \mathbb{N}, \text{ in binary"}$$

Remark: the domain of a problem, namely $\{0,1\}^*$, is typically an encoding of some interesting combinatorialized mathematical object.

Before doing complexity, we need to understand computability.

Turing machines are one way to make "algorithms" precise.

Turing Machines def'n from Arora + Barak's "Computational Complexity

Formal definition. Formally, a TM M is described by a tuple (Γ, Q, δ) containing:

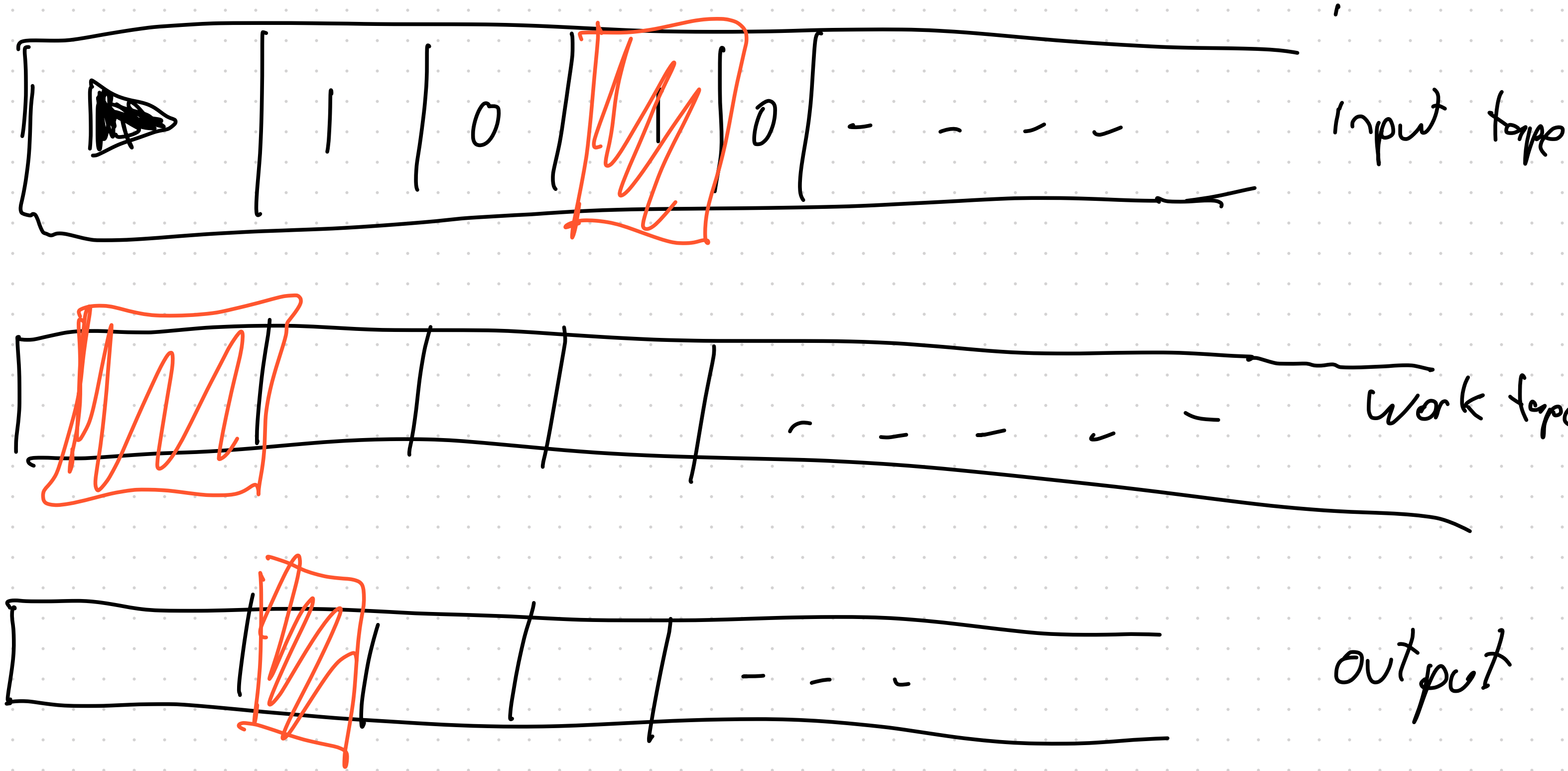
- A finite set Γ of the symbols that M's tapes can contain. We assume that Γ contains a designated "blank" symbol, denoted $\square$; a designated "start" symbol, denoted $\triangleright$; and the numbers 0 and 1. We call Γ the *alphabet* of M.
- A finite set Q of possible states M's register can be in. We assume that Q contains a designated start state, denoted $q_{\mathtt{start}}$, and a designated halting state, denoted $q_{\mathtt{halt}}$.
- A function $\delta : Q \times \Gamma^k \to Q \times \Gamma^{k-1} \times \{\mathsf{L}, \mathsf{S}, \mathsf{R}\}^k$, where $k \geq 2$, describing the rules M use in performing each step. This function is called the *transition function* of M (see Figure 1.2.)

k = # of memory tapes

Each tape has its own "Read-Write" head that moves Left + Right

IF			THEN			
Input symbol read	Work/ output tape symbol read	Current state	Move input head	New work/ output tape symbol	Move work/ output tape	New state
⋮	⋮	⋮	⋮	⋮	⋮	⋮
a	b	q	Right $\rightarrow$	b'	Left $\leftarrow$	q'
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Figure 1.2. The transition function of a two-tape TM (i.e., a TM with one input tape and one work/output tape).



Each tape has a "read-write head" that moves left or right

This is an important model, with some arbitrary choices.

<u>Church-Turing Thesis:</u> "Computable" does not depend on model of computation.

<u>Extended Church-Turing Thesis:</u> "Efficiently computable" does not depend on the model of computation, as long as its "realistic".

<u>Problem:</u> Quantum computers.

II. Usual suspects

A complexity class is any set of (decision or counting) problems. Typically interested in complexity classes defined by restraining the resources used by a Turing machine (space, time, etc...)

We'll start at the "top."

RE: recursively enumerable decision problems.

$L \in RE$ if there exists a Turing machine s.t. for all $x \in \{0,1\}^*$, if $L(x) = \text{Yes}$, then the Turing machine returns Yes when input x and enters the "HALT" state.

Example: Halting Problem.

Given a Turing machine T, determine if T halts when input an empty string.

In RE because we can build a Turing machine that runs other Turing machines inside of it (Universal Turing machine)

Example: (Homeomorphism problem for PL-manifolds)

$$L : \{0,1\}^* \times \{0,1\}^* \longrightarrow \{Yes, No\}$$

$$L(x,y) = \begin{cases} Yes & \text{if } x \text{ and } y \text{ represent triangulations of PL manifolds that are PL-homeomorphic} \\ No & \text{otherwise.} \end{cases}$$

coRE: Same as RE but swap role of YES and NO.

R: recursive (or computable) functions, defined by

$$R = RE \cap coRE$$

Intuition: a problem is in R if there is a way to solve it algorithmically, but w/ no bounds on resources required.

Non-examples: Neither Halting Problem nor Homeo. Problem for PL-manifolds is in R.

Example: 3-Manifold Homeomorphism Problem

$L : \{0,1\}^* \times \{0,1\}^* \longrightarrow \{Yes, No\}$

$$L(x,y) = \begin{cases} Yes & \text{if } x, y \text{ represent homeomorphic PL-3-manifolds} \\ No & \text{otherwise} \end{cases}$$

Why? Geometrization.