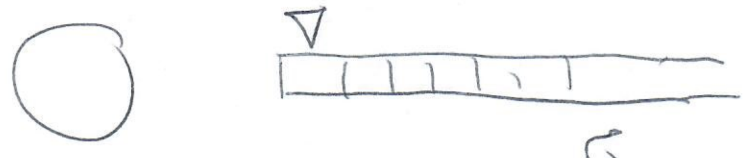
Oct 20 (1)

- One tape versus 2-tape versus k-tape
  PALINDROME, ADD
- Countable number of algorithms

=

1-tape

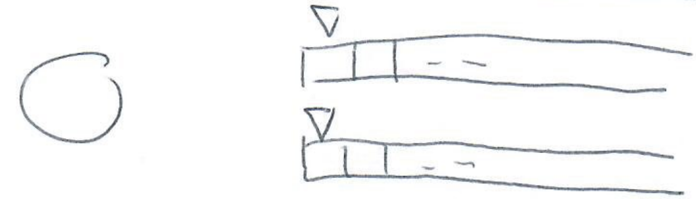


← input is written here

↖ work tape

=

2-tape



← input

← initially blank

$\delta : Q \times \Gamma^2 \to Q \times \Gamma^2 \times \{L, \cancel{R}, S\}^2$

k-tape
(k finite)

$\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, R, S\}^k$

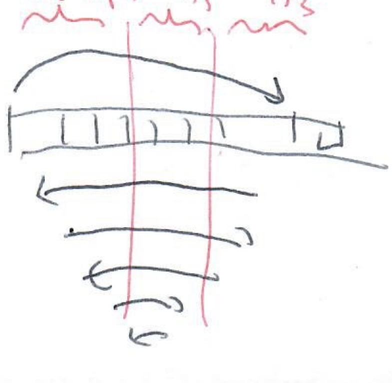
=

1-tape vs. 2-tape   not easy to prove c_M = depends on M

$PALINDROME_{\{0,1\}}$ takes time $\geq c_M n^2$ on any 1-tape

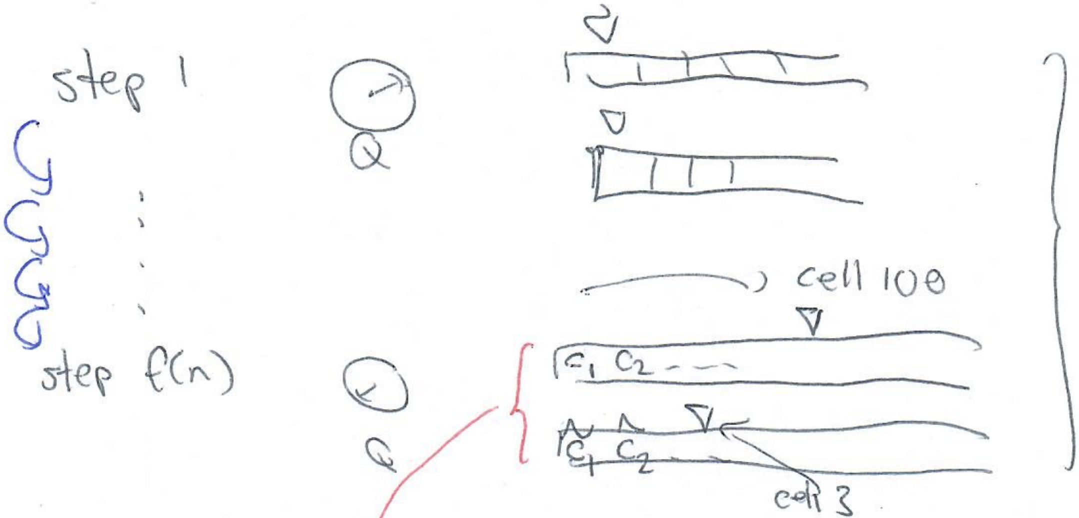
"   " $O(n)$ on a 2-tape machine

1/3 input   1/3   1/3

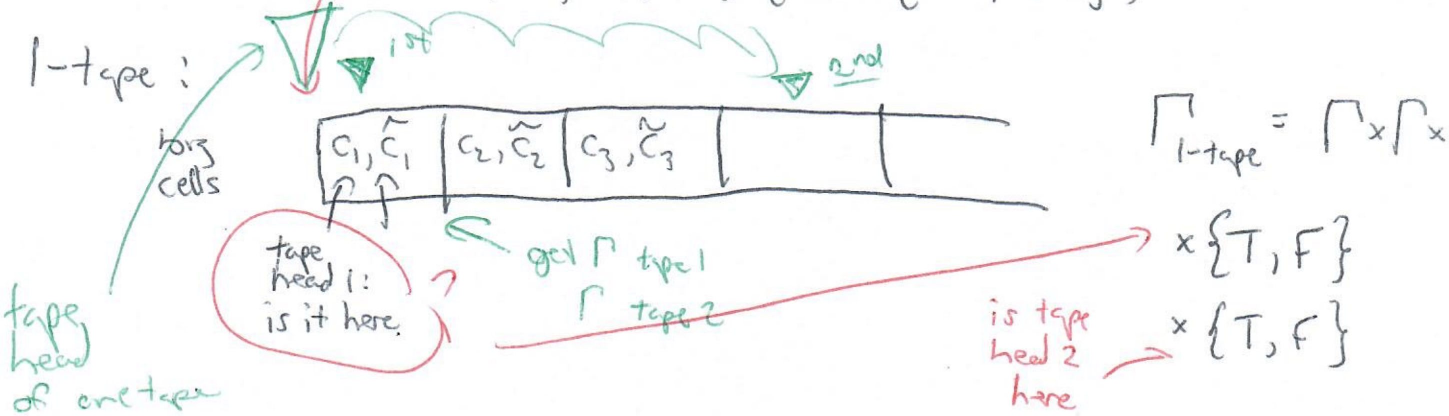


1-tape
Worst Case:
2-tape machine runs in time $f(n)$ on any string of size n,
there is a 1-tape machine runs in time $(f(n))^2$.

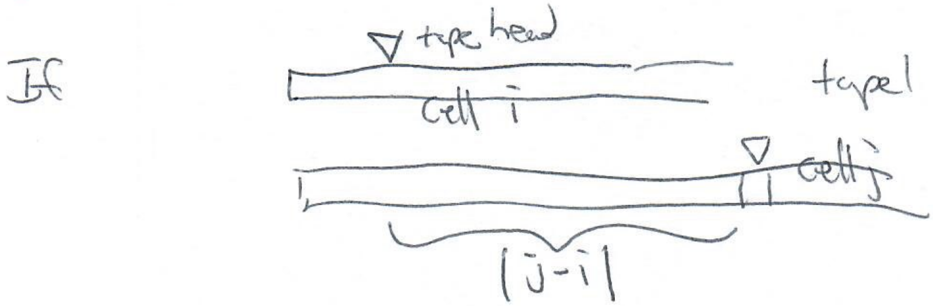
Idea: If L is recognized by a 2-tape machine



2-tape Machine: $(Q, \Sigma, \Gamma, \delta, q_{init}, q_{acc}, q_{rej})$



If



need at least $|j-i|$ steps on 1-tape

can't do $o(|j-i|)$!

For $f(n)$ steps 2-tape machine,

You have $f(n) \cdot (\underbrace{\max |j-i|}_{\leq f(n)})$ for the one tape

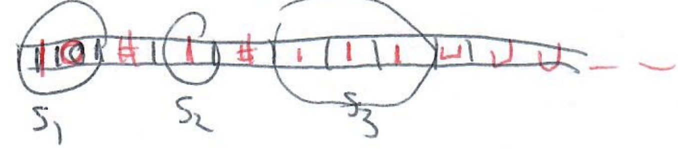
Time 1-tape simulation $\leq$ order $(f(n))^2$ ⟵ can't be improved (for all 2-tape machines)

$$\text{ADD} = \left\{ s \in \{0,1,\#\}^* \;\middle|\; \begin{array}{l} s = s_1 \# s_2 \# s_3 \text{ s.t.} \\ s_1, s_2, s_3 \in \{0,1\}^* \text{ with} \\ s_1 = s_2 + s_3 \text{ as binary integers} \end{array} \right\}$$

3

State addition problems

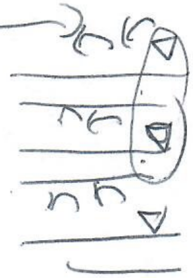
Show ADD is recognized by a 8-tape machine, give high-level description



Step 1

tape 2 $\quad s_2$
tape 3 $\quad s_3$
tape 4 $\quad$ carry bit
tape 8 $\quad s_1$

tape 5–7 ignore – –



} add as in (1st grade) ?
start at least sig bit
move ⟵

Claim (Definition): An <u>algorithm</u> to recognize a language $L \subset \Sigma^*$ is a $\left(\begin{array}{c}\text{one-tape} \\ \text{2-tape} \\ \vdots\end{array}\right)$ Turing machine, M, that recognizes the language.

Claim: There are languages in Σ^*, for any Σ, that ~~can't be~~ aren't recognized by any TM, i.e. can't be solved by an <u>algorithm</u>.

<u>Proof</u>: Σ alphabet $\Rightarrow$ Σ^* countably infinite $\Rightarrow$ Power(Σ^*) = {Languages} is uncountable.

Turing machine $(Q, \Sigma, \Gamma, --)$

Algorithm: Turing machines but don't care how states are named, symbols are named, --

Standardize:

$$\begin{cases} Q = \{1, \ldots, q\} \\ \Sigma = \{a, b, c\} \leftarrow \text{given} \\ \Gamma = \{a, b, c, \sqcup, 1, \ldots, S\} \\ \delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\} \leftarrow \text{has some description in terms of} \end{cases}$$

write this:

335 # a b c ⊔ 217 ## 1 # 1 # 3 # 20 # L
1 # 2 # 112 # a # R

states $\quad \Gamma$

Standardize TM's
C programs
Java programs
Python program

} strings over some alphabet $\Rightarrow$ countably many