Today:

- Finish PALINDROME

\[
\begin{align*}
\text{Chap} & \ 3,4 \\
\text{Ch} & \ 4.2 \\
\end{align*}
\]

- Multi-tape TM
- Non-det TM
- Standardized TM
- Universal TM

Uncomputability in CPSC 421
PALINDROME $\{a, b\}$

$$= \{ w \in \{a, b\}^* \mid w \text{ reverse } w \}$$

Idea: (1)

$\Gamma = \{ a, b, w, \# \}$

first saw "a"

if we read "b" keep going

then reject
if

1st & last

symbols match

G move left

```
<table>
<thead>
<tr>
<th>#</th>
<th>#</th>
<th>a</th>
<th>b</th>
<th>b</th>
<th>#</th>
</tr>
</thead>
</table>
```

```
<table>
<thead>
<tr>
<th>#</th>
<th>#</th>
<th>a</th>
<th>b</th>
<th>b</th>
<th>#</th>
</tr>
</thead>
</table>
```

```
<table>
<thead>
<tr>
<th>#</th>
<th>#</th>
<th>a</th>
<th>b</th>
<th>b</th>
<th>#</th>
<th>t</th>
</tr>
</thead>
</table>
```

```
<table>
<thead>
<tr>
<th>#</th>
<th>#</th>
<th>a</th>
<th>b</th>
<th>b</th>
<th>#</th>
</tr>
</thead>
</table>
```

Lab
Does this work?

In all cases
(1) We are convinced that the TM already built:

- correctly rejects even length non-palindromes

What about

- accept even length palindromes?

```
| a | b | b | a | # | # | ...
```

\[ \square \]

\[ \Rightarrow \quad \text{# # # #} \]

\[ \Rightarrow \quad \text{# # b #} \]

\[ \Rightarrow \quad \text{# # # #} \]
What about accepting palindromes of odd length?

\[ \text{\begin{array}{c}
\text{a} & \text{a} & \text{a} & \text{a} & \text{a} & \text{a} & \text{a} \\
\end{array}} \]

\[ \text{↑} \quad \text{→ moved} \]

\[ \# \quad \# \quad \# \]

\[ \swarrow \]

\[ \# \quad \# \quad \# \]

\[ \searrow \]
Multi-tape TM:

2-tape TM:

initially: input aaa

Formally: \((Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej})\)

\[\delta: Q \times \Gamma^2 \rightarrow Q \times \Gamma^2 \times \{L, R, S\}^2\]
$\delta(q, y_1, y_2) = (q', y_1', y_2', \frac{R}{5}, \frac{R}{5})$

new state

write over on tape 1
write over on tape 2

movement

tape head
tape 1
tape 2

(1) How long does palindrome take to solve on 1-tape

[alacacagag]
steps $\leq$ time

for

$n \sim \text{order } (n^2)$

where $n = \text{size of input} = \text{length of input as string}$