ComS 331 **Spring 2025** **Name: Mobina Amrollahi**

HW 9 Due: April 4th 2025

1. Consider the Turing machine $M = (K, \Sigma, \delta, s_0)$ where $K = \{s_0, \ldots, s_9\}$, $\Sigma = \{0, 1, \#\}$, and δ is described by the following table:

old state	input	new state	action	old state	input	new state	action	old state	input	new state	action
s_0	0	s_0	R	s_0	1	s_0	R	s_0	#	s_1	L
s_1	0	s_2	#	s_1	1	s_6	#	s_1	#	h	#
s_2	0	s_2	0	s_2	1	s_2	1	s_2	#	s_3	L
s_3	0	s_3	L	s_3	1	s_3	L	s_3	#	s_4	R
s_4	0	s_5	#	s_4	1	s_4	1	s_4	#	s_0	#
s_5	0	s_5	0	s_5	1	s_5	1	s_5	#	s_0	R
s_6	0	s_6	0	s_6	1	s_6	1	s_6	#	s_7	L
s_7	0	s_7	L	s_7	1	s_7	L	s_7	#	s_8	R
s_8	0	s_8	0	s_8	1	s_9	#	s_8	#	s_0	#
s_9	0	s_9	0	s_9	1	s_9	1	s_9	#	s_0	R

(a) Explain the behavior of the Turing machine informally and give a compact expression for the language L it accepts.

Answer.

This Turing machine accepts the set of all even-length and odd-length palindromes over the binary alphabet $\{0, 1\}$. A compact way to define this language is:

$$L = \{wxw^R \mid w \in \{0,1\}^*,\ x \in \{0, 1, \epsilon\}\}$$

To prove that the given Turing machine accepts exactly the set of all palindromes, we proceed by induction on the length of the string $n = |w|$.

Base Cases. We consider the smallest possible strings:

(a) $w = \epsilon$: The following is the run of the machine on this string:

$$\#s_0\# \to s_1\#\# \to h\#\#$$

(b) $w = 0$. The following is the run of the machine on this string:

$$\#s_0 0\# \to \#0s_0\# \to \#s_1 0\# \to \#s_2\#\# \to s_3\#\#\# \to \#s_4\#\# \to \#s_0\#\# \to s_1\#\#\# \to h\#\#\#$$

(c) $w = 1$. The following is the run of the machine on this string:

$$\#s_0 1\# \to \#1s_0\# \to \#s_1 1\# \to \#s_6\#\# \to s_7\#\#\# \to \#s_8\#\# \to \#s_0\#\# \to s_1\#\#\# \to h\#\#\#$$

Thus, the machine halts and accepts all strings of length $n \leq 1$ that are palindromes.
Induction Hypothesis. Suppose, the machine halts and accepts all strings of length less than n that are palindromes and loops forever and does not halt otherwise.
Induction Step. We now consider a string $w = a_1a_2\cdots a_n$ of length $n \geq 2$. There will be two possibilities for a_1 and a_n. Whether $a_1 = a_n$ or $a_n \neq a_n$.

(a) $a_1 = a_n$. We will show the machine marks both the first and last characters, transitions into a new configuration, and proceeds recursively on the sub-string $a_2a_3\cdots a_{n-1}$, which is of length $n-2$. By the inductive hypothesis, the machine will accept this sub-string if it is a palindrome.

i. $a_1 = a_n = 0$.

$$\#s_00a_2\cdots 0\# \xrightarrow{n-1} \#0a_2\cdots a_{n-1}s_00\# \to \#0a_2\cdots a_{n-1}0s_0\# \to \#0a_2\cdots a_{n-1}s_10\#$$
$$\to \#0a_2\cdots a_{n-1}s_2\#\# \to \#0a_2\cdots s_3a_{n-1}\#\# \xrightarrow{n-2} \#s_30a_2\cdots a_{n-1}\#\# \to s_3\#0a_2\cdots a_{n-1}\#\#$$
$$\to \#s_40a_2\cdots a_{n-1}\#\# \to \#s_5\#a_2\cdots a_{n-1}\#\# \to \#\#s_0a_2\cdots a_{n-1}\#\#$$

Now the machine is back at the start state and operating on the substring of length $n-2$.

ii. $a_1 = a_n = 1$.

$$\#s_01a_2\cdots 1\# \xrightarrow{n-1} \#1a_2\cdots a_{n-1}s_01\# \to \#1a_2\cdots a_{n-1}1s_0\# \to \#1a_2\cdots a_{n-1}s_11\#$$
$$\to \#1a_2\cdots a_{n-1}s_6\#\# \to \#1a_2\cdots s_7a_{n-1}\#\# \xrightarrow{n-2} \#s_71a_2\cdots a_{n-1}\#\# \to s_7\#1a_2\cdots a_{n-1}\#\#$$
$$\to \#s_81a_2\cdots a_{n-1}\#\# \to \#s_9\#a_2\cdots a_{n-1}\#\# \to \#\#s_0a_2\cdots a_{n-1}\#\#$$

Again, the machine returns to the start state and processes the sub-string of length $n-2$.

(b) $a_n \neq a_n$. In this case, the machine attempts to match the first and last symbols but fails. It ends up looping in a non-halting configuration:

i. $a_1 = 1$ and $a_n = 0$.

$$\#s_01a_2\cdots a_{n-1}0\# \xrightarrow{n-1} \#0a_2\cdots a_{n-1}s_00\# \to \#0a_2\cdots a_{n-1}0s_0\# \to \#0a_2\cdots a_{n-1}s_10\#$$
$$\to \#0a_2\cdots a_{n-1}s_2\#\# \to \#0a_2\cdots s_3a_{n-1}\#\# \xrightarrow{n-2} \#s_30a_2\cdots\#\# \to s_3\#0a_2\cdots\#\#$$
$$\to \#s_41a_2\cdots\#\# \to \#s_41a_2\cdots\#\#$$

The machine loops in state s_4, writing 1's forever.

ii. $a_1 = 0$ and $a_n = 1$.

$$\#s_00a_2\cdots a_{n-1}1\# \xrightarrow{n-1} \#0a_2\cdots a_{n-1}s_01\# \to \#0a_2\cdots a_{n-1}1s_0\# \to \#0a_2\cdots a_{n-1}s_11\#$$
$$\to \#0a_2\cdots a_{n-1}s_6\#\# \to \#0a_2\cdots s_7a_{n-1}\#\# \xrightarrow{n-2} \#s_70a_2\cdots\#\# \to s_7\#0a_2\cdots\#\#$$
$$\to \#s_80a_2\cdots\#\# \to \#s_80a_2\cdots\#\#$$

The machine loops in state s_8, writing 0's forever.

(b) For a string $x \in L$, give the polynomial order for the number of steps required by M to accept x, as a function of $n = |x|$.
Answer.
Based on part (a), we know that the Turing machine checks whether an input string is a palindrome by comparing the first and last characters and then recursively working its way inward.

To compare the first and last characters, the machine:

- Scans from the beginning to the end of the string — this takes approximately n steps,
- Marks the last character (e.g., with a #), and based on whether it is a 0 or 1, takes the appropriate transition,
- Scans back to the beginning of the string — another n steps,
- Matches the first character with the one remembered from the end.

So, each outer pair comparison takes roughly $2n$ steps.

After removing the first and last characters, the machine repeats this process on a string of length $n-2$, then $n-4$, and so on.

The total number of steps is:

$$\sum_{i=0}^{k} 2(n-2i)$$

where $k = \lfloor \frac{n}{2} \rfloor$, since we stop when we reach the middle of the string.

We simplify this as:

$$2\sum_{i=0}^{k}(n-2i) = 2\left[(n-0)+(n-2)+(n-4)+\cdots+(n-2k)\right]$$

Using the formula for the sum of an arithmetic sequence:

$$\sum_{i=0}^{k}(n-2i) = \frac{(k+1)}{2}\left[n+(n-2k)\right] = \frac{(k+1)}{2}(2n-2k)$$

Therefore, the total number of steps is:

$$2 \cdot \frac{(k+1)}{2}(2n-2k) = (k+1)(2n-2k)$$

Now, the following inequalities hold for the total number of steps:

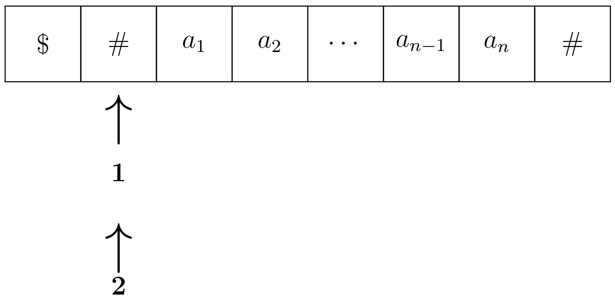
$$(\frac{n}{2}(n)) = \frac{n^2}{2} \leq (k+1)(2n-2k) \leq (n+1)2n = 2n^2+n$$

Both of the expressions $\frac{n^2}{2}$ and $2n^2+n$ belong to $\mathcal{O}(n^2)$. Hence, $(k+1)(2n-2k) \in \mathcal{O}(n^2)$.

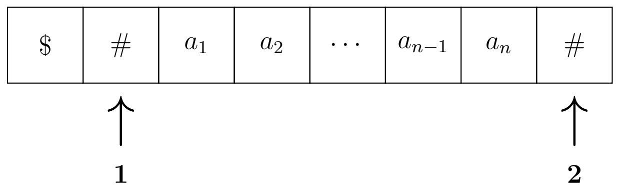
(c) Describe informally how L can be accepted in $O(n)$ steps by a two-head Turing machine.

Answer.

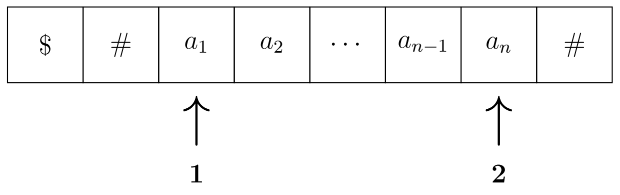
We describe an informal procedure for accepting L using a two-head Turing machine. Assume both tape heads start at the first symbol after the \$, and the input string w has length n:



(1) Move the second head one cell to the right one at a time until it reaches the # symbol.

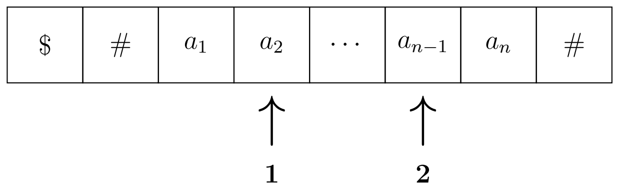


Then move the first head one step to the right and the second head one step to the left so they points to the last character of the string. This takes $n + 2$ steps.

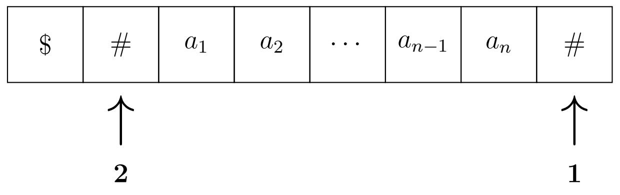


(2) Compare the symbols under both heads. *Reject* if they differ. If they are the same:

- Repeat until both heads are pointing at #:
  - Move the first head one step to the right and the second head one step to the left. This takes two steps.



  - Compare the symbols under both heads. *Reject* if they differ.



(3) *accept.*

Stage 1 takes $n + 2$ moves and stage two takes $2 \times n$ moves. Hence the total process takes $3n + 2$ moves which is $\in O(n)$.