THEORY OF FORMAL LANGUAGES

Turing Machine as Language Recognizer

Lecture 2

Prof. K. R. Chowdhary • Professor of Computer Science

EXAMPLE 2.1

Recognizing $L = \{ a^n b^n \mid n \geq 0 \}$

Idea: pair off each a with a matching b by marking them off, one round-trip at a time.

1 Replace the left-most a by X; move right until the left-most b is found.

2 Replace this b by Y; move left to the right-most X, then one step right onto the left-most remaining a.

3 Repeat steps 1 and 2, in order: 1, 2, 1, 2, …

4 While searching for b, if a blank B is hit first ($|a^n| = |b^n|$), halt without accepting.

5 If a is not found but b is ($a^n \neq b^n$), halt without accepting.

6 After changing a b to Y, if no more a's remain, check that no b's remain either — if so, accept.

FORMAL SPECIFICATION

The Machine M, in Full

Q $\{q_0, q_1, q_2, q_3, q_4\}$

Σ $\{a, b\}$

Γ $\{\triangleright, a, b, X, Y, B\}$

s q_0

H $\{q_4\}$

TABLE 2.1 — TRANSITION FUNCTION

state	a	b	X	Y	B
q_0	(q_1, X, R)			(q_3, Y, R)	(q_4, B, L)
q_1	(q_1, a, R)	(q_2, Y, L)		(q_1, Y, R)	
q_2	(q_2, a, L)		(q_0, X, R)	(q_2, Y, L)	
q_3				(q_3, Y, R)	(q_4, B, L)

Blank cells mean "no transition defined" — those inputs never occur in that state.

Transition Diagram for $a^n b^n$

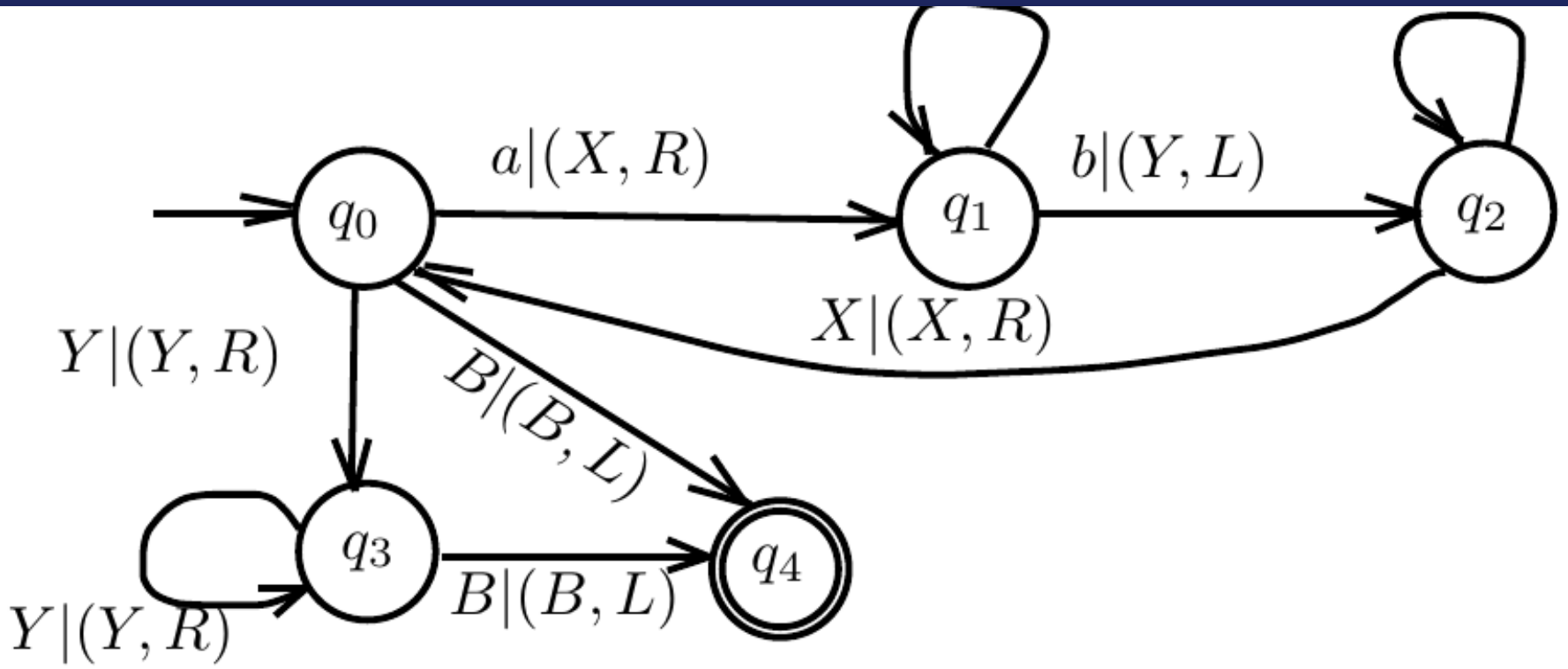

Figure 2.1 — Transition diagram for the TM accepting L = { a"b" | n ≥ 0 }

WORKED TRACE

Accepting w = aabb

Each line is one move (⊢). B marks the end of the input on the tape.

q_0aabbB	$\vdash q_2XaYbB$	$\vdash Xq_2XYYB$
$\vdash Xq_1abbB$	$\vdash Xq_0aYbB$	$\vdash XXq_0YYB$
$\vdash Xaq_1bbB$	$\vdash XXq_1YbB$	$\vdash XXYq_3YB$
$\vdash Xaq_2YbB$	$\vdash XXYq_1bB$	$\vdash XXYYq_3B$
$\vdash Xq_2aYbB$	$\vdash XXq_2YYB$	$\vdash XXYq_4YB$

Final configuration $XXYq_4YB$ has $q_4 \in H \rightarrow$ aabb is accepted.

EXAMPLE 2.2

How Many Steps Does It Take?

One round trip (find a b, come back to the next a) takes about 2n steps for an input of length n.

This round trip repeats n times — contributing $n \times n = n^2$ steps.

Add n steps for the $q_0 \to q_3$ sweep, plus the q_3 self-loop, plus one final $q_3 \to q_4$ step.

TOTAL TRANSITIONS

$$n^2 + n + 1$$

TIME COMPLEXITY

$$O(n^2)$$

For $|w| = n$, the exact count $(n^2+n+1)/2$ still grows quadratically.

EXAMPLE 2.3

Extending to $L = \{ a^n b^n c^n \mid n \geq 0 \}$

The same marking technique extends naturally by adding one more state after q_2 (call it q_2') to also pair off c's.

START	aaabbbccc
AFTER ROUND 1	XaaYbbZcc
AFTER ROUND 2	XXaYYbZZc
FINAL	XXXYYYZZZ

EXAMPLE 2.4

Equal Number of 0's and 1's

On each pass right, one 0 and one 1 are changed to X, then the head returns to the left end. When the whole string is X's, it is accepted.

state	0	1	X	B
q_0	(q_1, X, R)	(q_2, X, R)	(q_0, X, R)	(q_4, B, L)
q_1	$(q_1, 0, R)$	(q_3, X, L)	(q_1, X, R)	
q_2	(q_3, X, L)	$(q_2, 1, R)$	(q_2, X, R)	
q_3	$(q_3, 0, L)$	$(q_3, 1, L)$	(q_3, X, L)	(q_0, B, R)

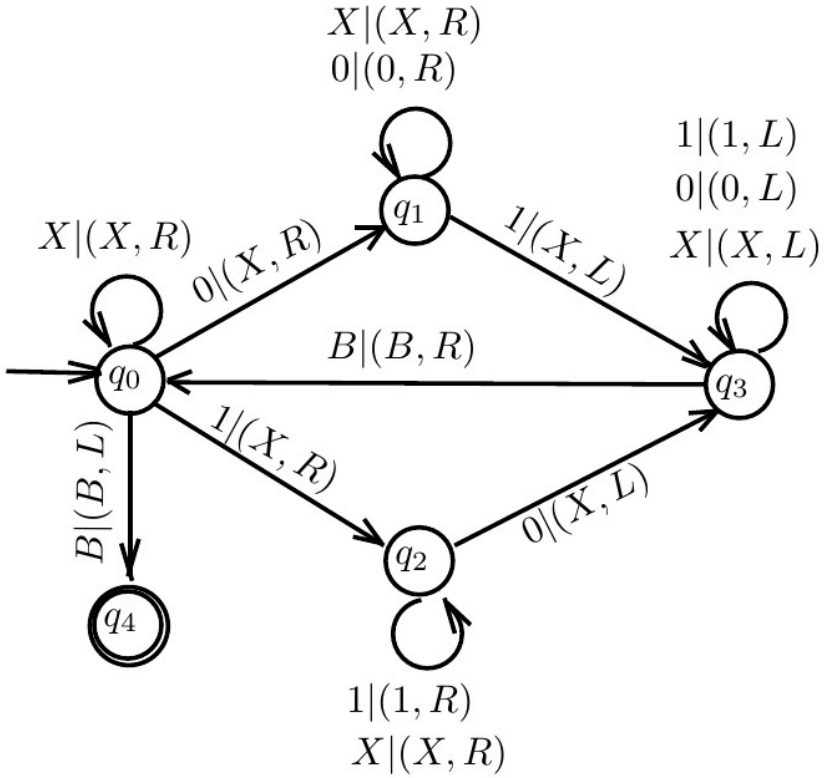

Figure 2.2 — TM for equal 0's and 1's

EXAMPLE 2.5

Palindromes: L = { ww^R | w ∈ {0,1}* }

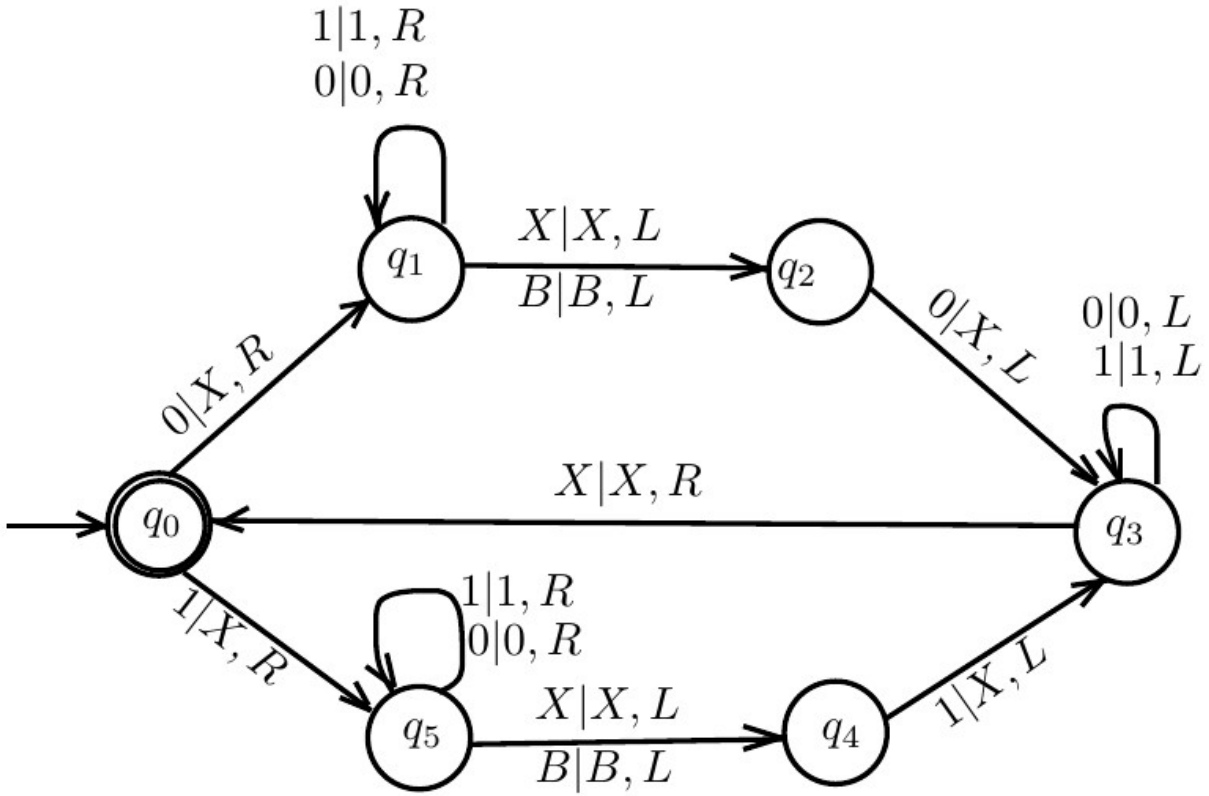

Figure 2.3 — TM for even-length palindromes

HOW IT WORKS

- Match the outermost pair of symbols, replace both with X, and work inward.
- This diagram covers even-length strings only — the language of even palindromes.
- A near-identical machine (Figure 2.4) handles odd-length palindromes, with one extra middle symbol untouched.

Odd-Length Palindromes

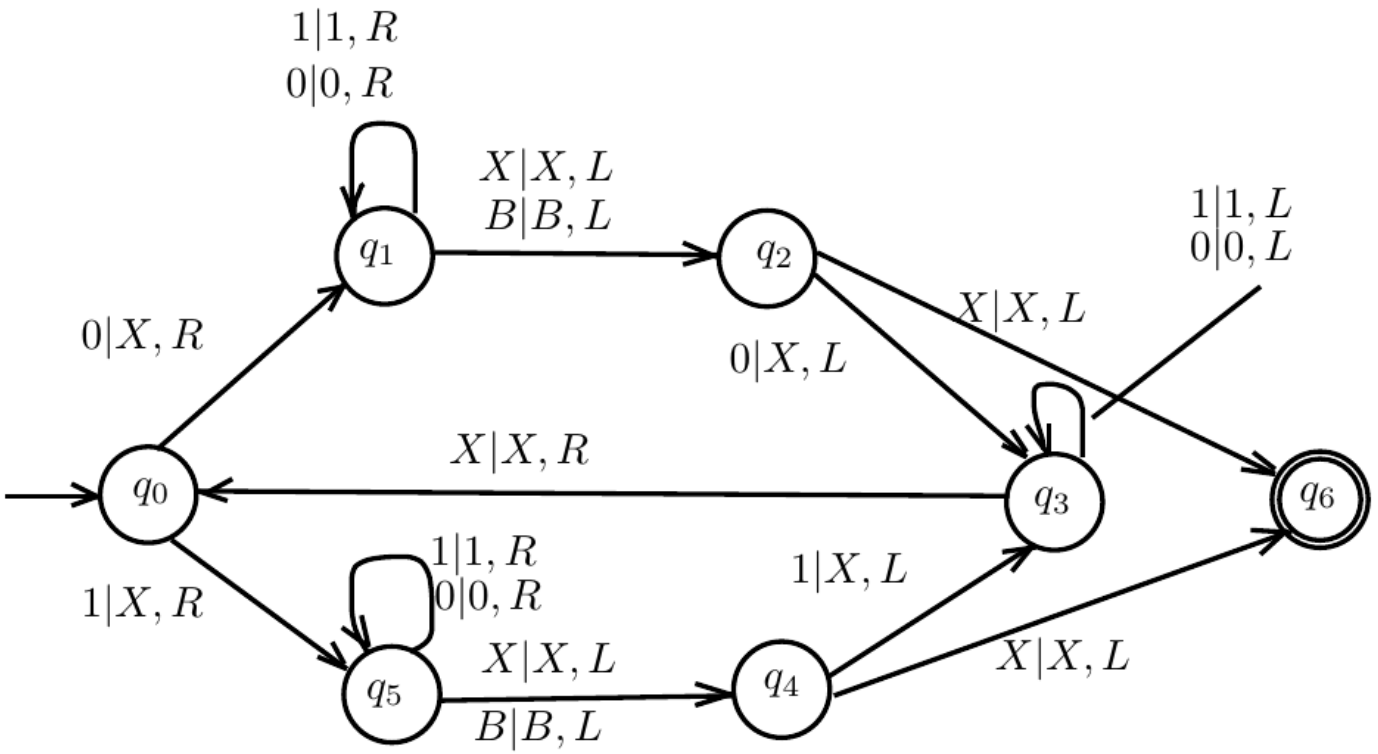

Figure 2.4 — TM for L = { w | w = w^R ∈ {0,1} }, w an odd palindrome*

EXAMPLE 2.6

Matching Halves: L = { w#w | w ∈ {0,1}* }

1. Mark the symbol under the head (remembering whether it was 0 or 1).
2. Move right past # to find the first unmarked matching symbol; reject if a blank or no match is found.
3. Mark that symbol as X too, then return left to the next unmarked symbol after #.
4. Repeat until every symbol left of # is marked, then check the right side has no unmarked symbols left.
5. If none remain, accept; otherwise reject.

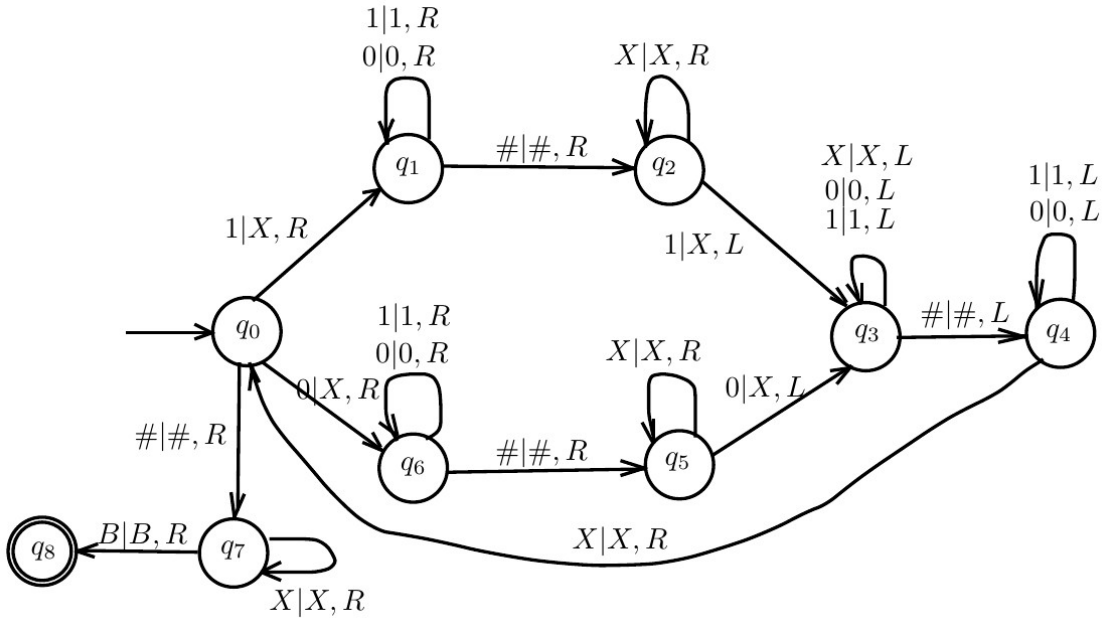

Figure 2.5 — TM for L = { w#w | w ∈ {0,1} }*

KEY DEFINITIONS

Deciding vs. Accepting a Language

M decides L

Recursive Language

For every string w: if w ∈ L, M(w) = “Yes”; if w ∉ L, M(w) = “No”. M always halts with a definite answer.

M accepts L

Recursively Enumerable

For every string w: if w ∈ L, M(w) = “Yes”; if w ∉ L, M(w) = “Undecided” — M may loop forever instead of rejecting.

Every example in this lecture built a TM that decides its language — halting with Yes or No on every input. That makes each of these languages Recursive.

EXAMPLE 2.7

Well-Matched Parentheses are Recursive

RECURSIVE DEFINITION OF L

Base: () ∈ L

Induction: if w ∈ L, then w(), ()w, and (w) are all in L

Since matched-parenthesis strings can be built and checked by this rule, a TM can decide membership — so L is a Recursive Language.

RECURSIVE FUNCTIONS

$$f : \Sigma^* \to \Gamma^*$$

M computes f if, for every w ∈ Σ*, M(w) = f(w). If such an M exists, f is called a recursive function.

A Turing Machine can be viewed as an algorithm for string problems — any computable object (graphs, numbers, networks) must first be encoded as a string before a TM can decide the corresponding language.

BUILDING BLOCK

Copying a String: w → wBw

Each symbol is copied to a new location after B: a is temporarily replaced by X, b by Y — marking what has already been copied — then both copies are restored to their original symbols.

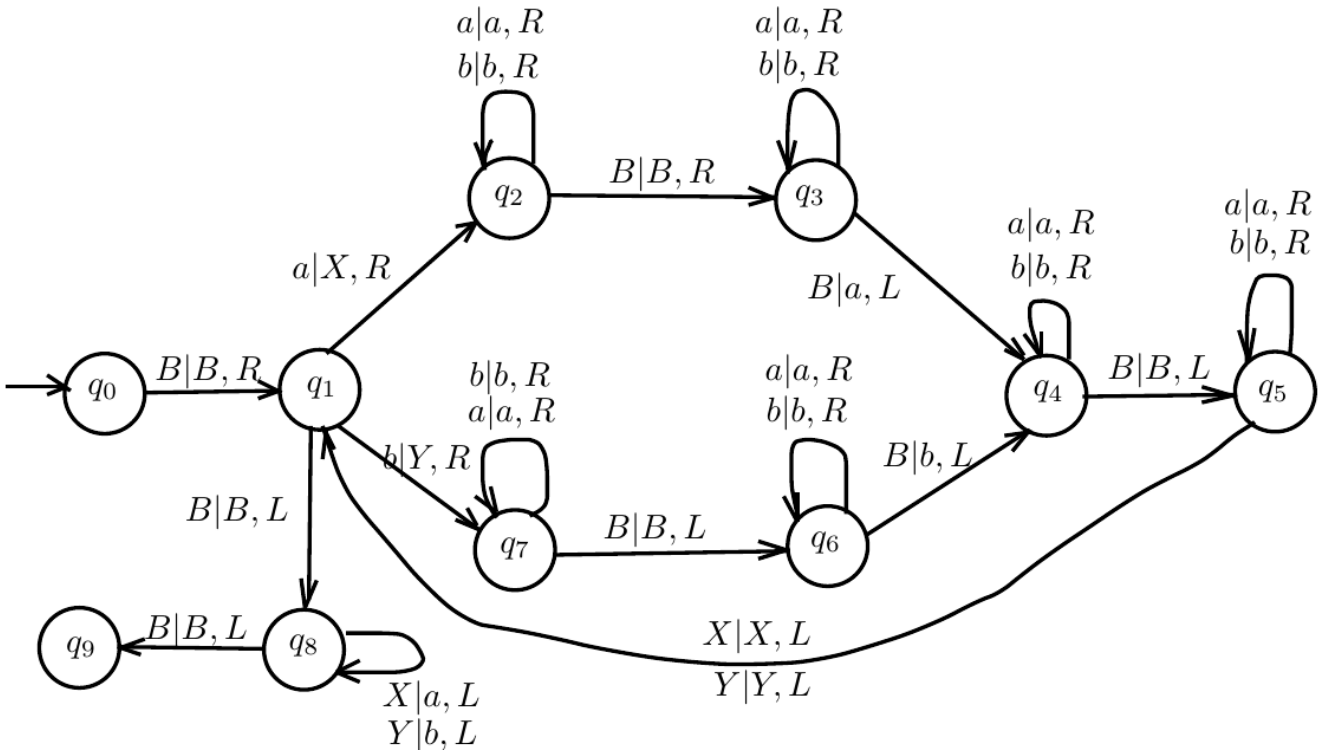

Figure 2.6 — A TM that produces output wBw for input wB

Key Takeaways

1. A TM recognizes $a^n b^n$ (and, with one more state, $a^n b^n c^n$) by marking matched symbols in passes — costing $O(n^2)$ time.

2. The same marking idea extends to equal-count strings, even/odd palindromes, and w#w matching.

3. A language is Recursive if some TM always halts with Yes/No; it is Recursively Enumerable if a TM may loop instead of rejecting.

4. Any computable problem must first be encoded as a string — copying ($w \rightarrow wBw$) is a key building block for more complex TMs.