Department of CSE(H)

THEORY OF COMPUTATION
23MT2014

Topic:

VARIANTS OF TM

Session - 22

33MT2014

THEORY OF COMPUTATION

Topic:

VARIANTS OF TURING MACHINE

CC - DR. ASESH KUMAR TRIPATHY

Session - 22

AIM OF THE SESSION

The aim of studying the variants of Turing machines is to extend the understanding of computational theory beyond the classical Turing machine model. This exploration provides insights into the versatility and limitations of different computational models, thereby broadening the scope of what can be computed and how efficiently it can be done.

INSTRUCTIONAL OBJECTIVES

This Session is designed to:

- Understand Turing Machine Variants.
- Analyse Computational Power.
- Apply Theoretical Concepts.
- Communicate Technical Ideas.

LEARNING OUTCOMES

At the end of this session, you should be able to:

- Explain Turing Machine Variants.
- Evaluate Computational Efficiency.
- Apply Theoretical Concepts.
- Understand Complexity Classes.
- Conduct Theoretical Analysis.

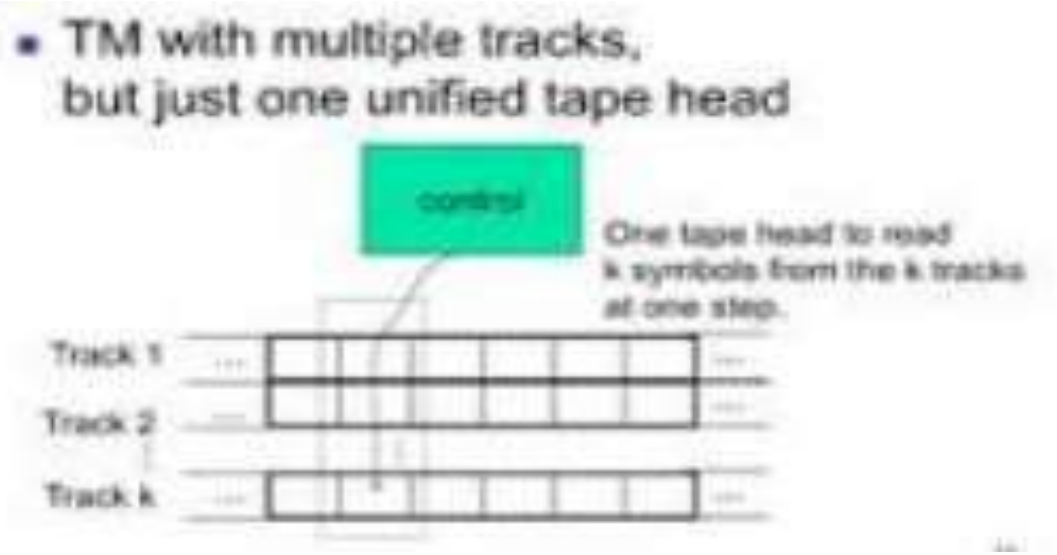
TURING MACHINE VARIANTS

There are Many variants but here we discuss this three variants of Turing Machine:

1. Multiple track Turing Machine
2. Multi-tape Turing Machine
3. Non-deterministic Turing Machine

MULTI TRACK TURING MACHINE

A Turing machine with k tracks (where k is greater than 0) features k tracks and a single read/write head that sequentially reads and writes on all tracks



DIFFEERENCE BETWEEN SINGLE TRACK AND MULTITRACK TM

1. Tape Structure
2. Head Movement
3. Computational Power

TAPE STRUCTURE

Single-track Turing Machine: Has a single tape where the input is written, and the machine's head can only read from and write to this single track.

Multi-track Turing Machine: This type of Turing Machine contain multiple track with single head and allowing the machine to read from and write to multiple tracks simultaneously.

HEAD MOVEMENT

Single-track Turing Machine: The head can move left or right along the single tape, one cell at a time.

Multi-track Turing Machine: In the case of n-track Turing machine, one head reads and writes on all tracks simultaneously.

COMPUTATIONAL POWER

Single-track Turing Machine: We Can simulate a multi-track Turing machine, albeit potentially less efficiently, by using its single track to emulate the behaviour of multiple tracks.

Multi-track Turing Machine: We Can potentially solve certain problems more efficiently than a single-track machine due to its ability to process multiple tracks simultaneously.

- Interestingly, a single-track Turing machine and a multi-track Turing machine have the same computational power.

EXAMPLE

Question : Addition of 2 numbers 3X2 using single tape Turing Machine

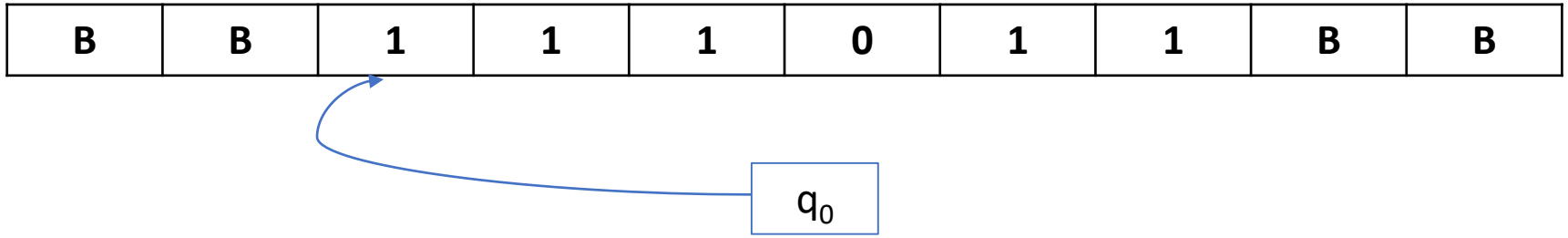
Answer :- Represent the numbers in unary form.

For example, the number 3 would be represented as 111, and the number 2 would be represented as 11. Use a delimiter (e.g., 0) to separate the two numbers on the tape.

B	B	1	1	1	0	1	1	B	B

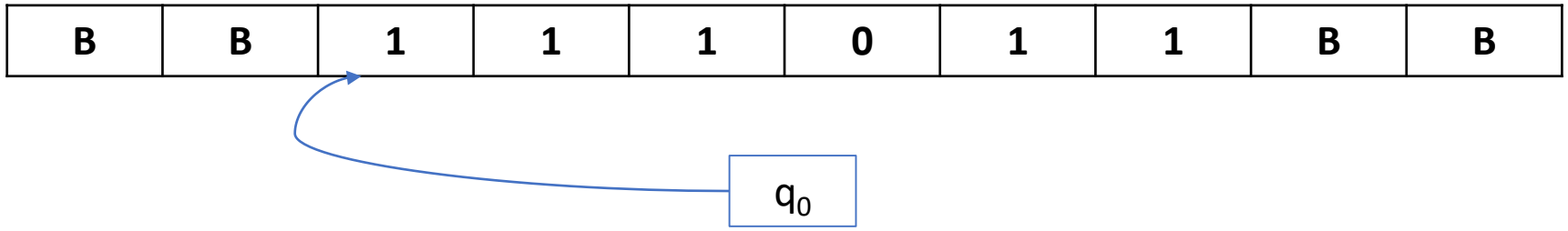
EXAMPLE CONTD....

Combine the unary representations into a single unary number that represents their sum.



Place the tape head at the beginning of the tape.
The machine starts in the initial state q0.

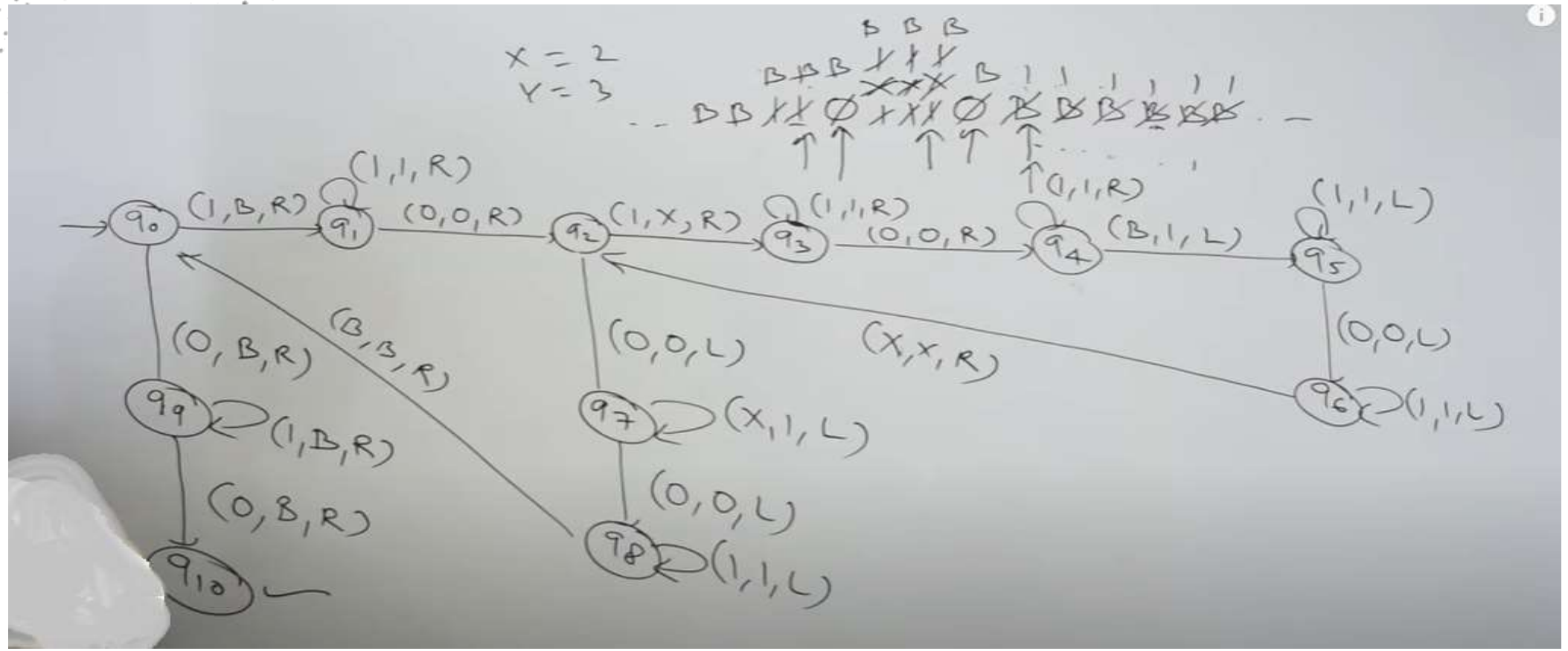
EXAMPLE CONTD….



Here N= 111 and M= 11 and 0 is delimiter or separator between M and N.

Basically, our aim when we got first 1 from N we make it Blank and the next one we make it 1 as it is and continue to right then when 0 found we replace with 0 and go right another two one will be copied to the blank this will be continue up to make all the 1's present in the N and finally we replace the 0 with blank

EXAMPLE CONTD….



EXAMPLE OF PROBLEM FOR MULTI TRACK TURING MACHINE

QUESTION :- ADDITION OF 2 INTEGER USING MULTI -TRACK TURING MACHINE (N XM)

ALGORITHM :

STEP 1 : PLACE N IN TRACK 1

STEP 2 : PLACE M IN TRACK 2

STEP 3 : IF T2=0 AND T3=0 RESULT IN T3

STEP 4 : IF T1=0 AND T3=0 RESULT IN T3

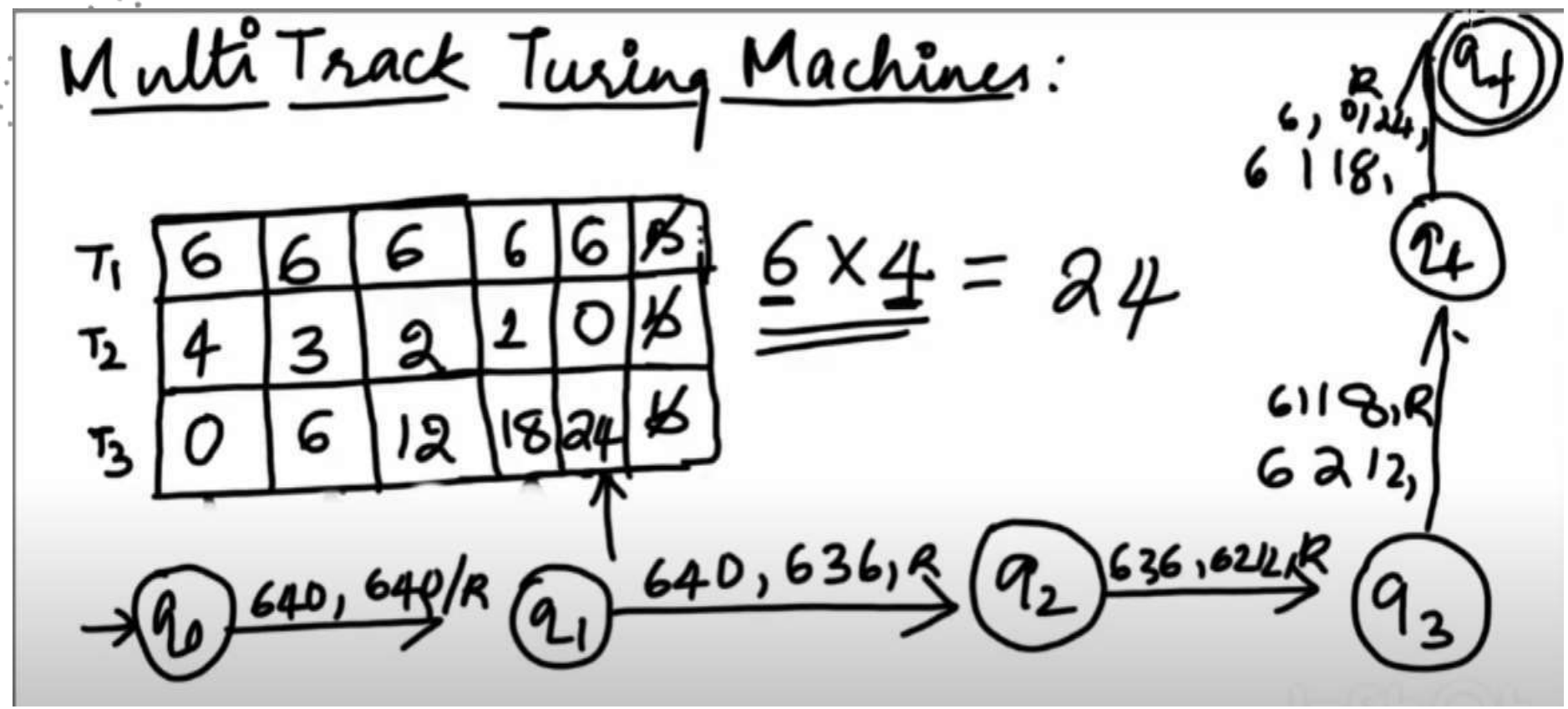
STEP 5 : T3=T3+T1

STEP 6 : T2=T2-1

STEP 7 : T2=0 THEN STOP

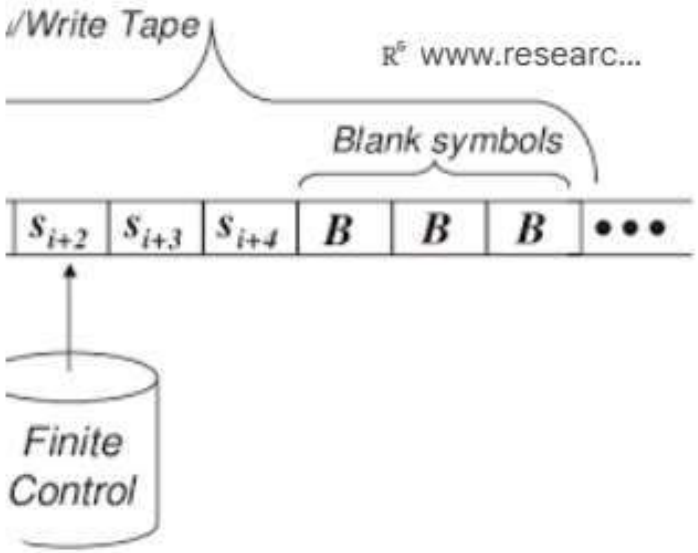
EXAMPLE OF PROBLEM FOR MULTI TRACK TURING MACHINE

QUESTION :- ADDITION OF 2 INTEGER USING MULTI -TRACK TURING MACHINE

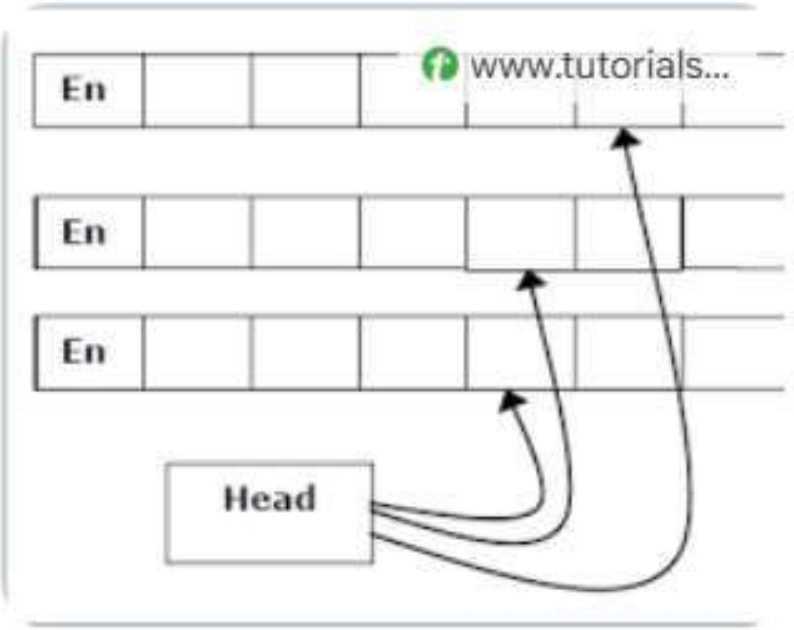


MULTI TAPE TURING MACHINE

Single-tape Turing Machine

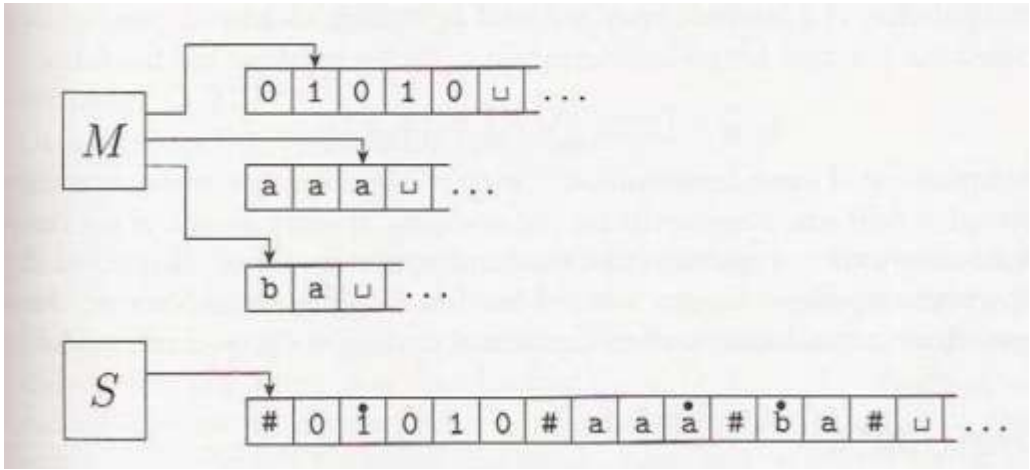


Multi-tape Turing Machine



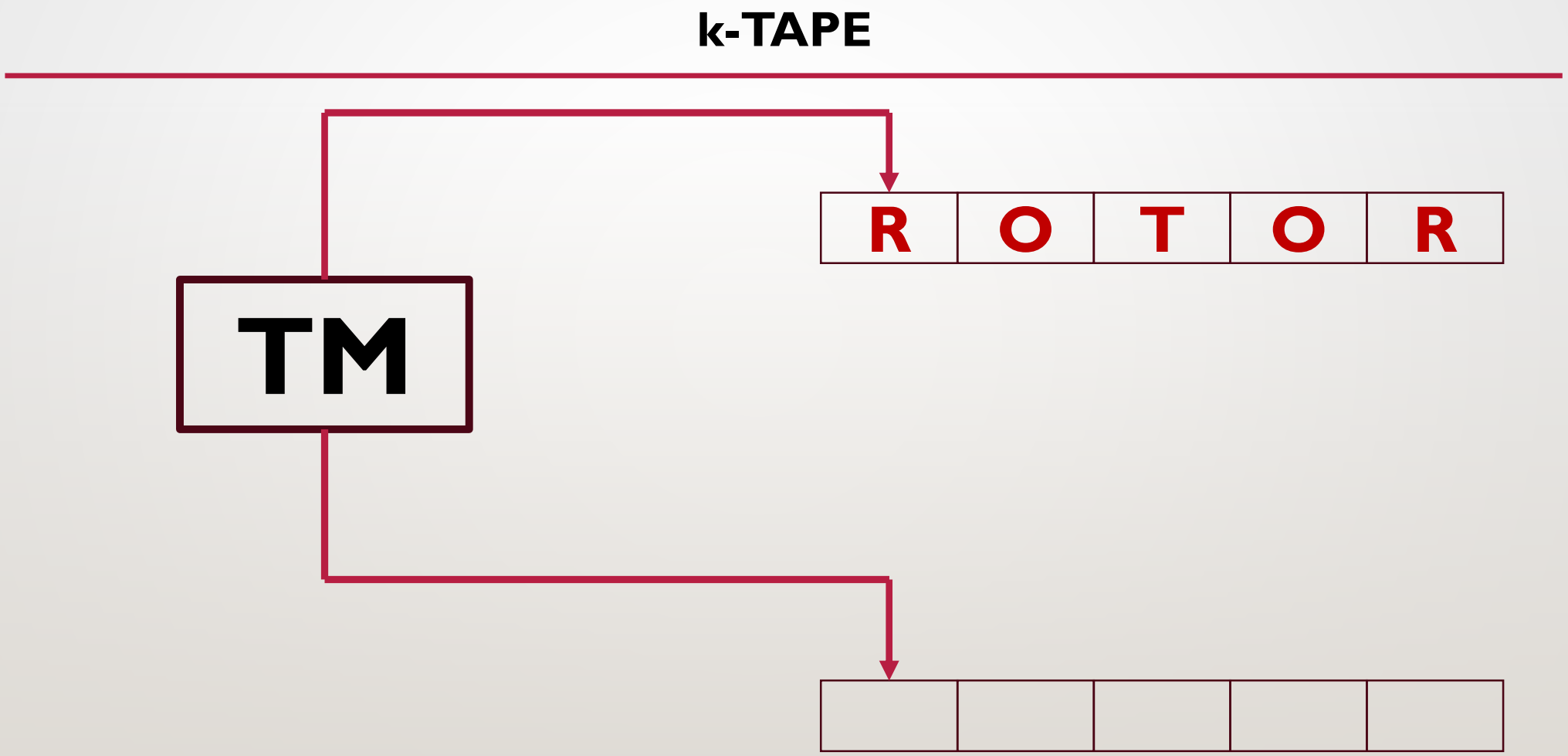
POWER COMPAIRISION

Surprisingly, single-tape and multi-tape Turing machines have the same computational power. This means they can recognize and generate the same set of languages, even though a multi-tape machine might seem more powerful at first glance.

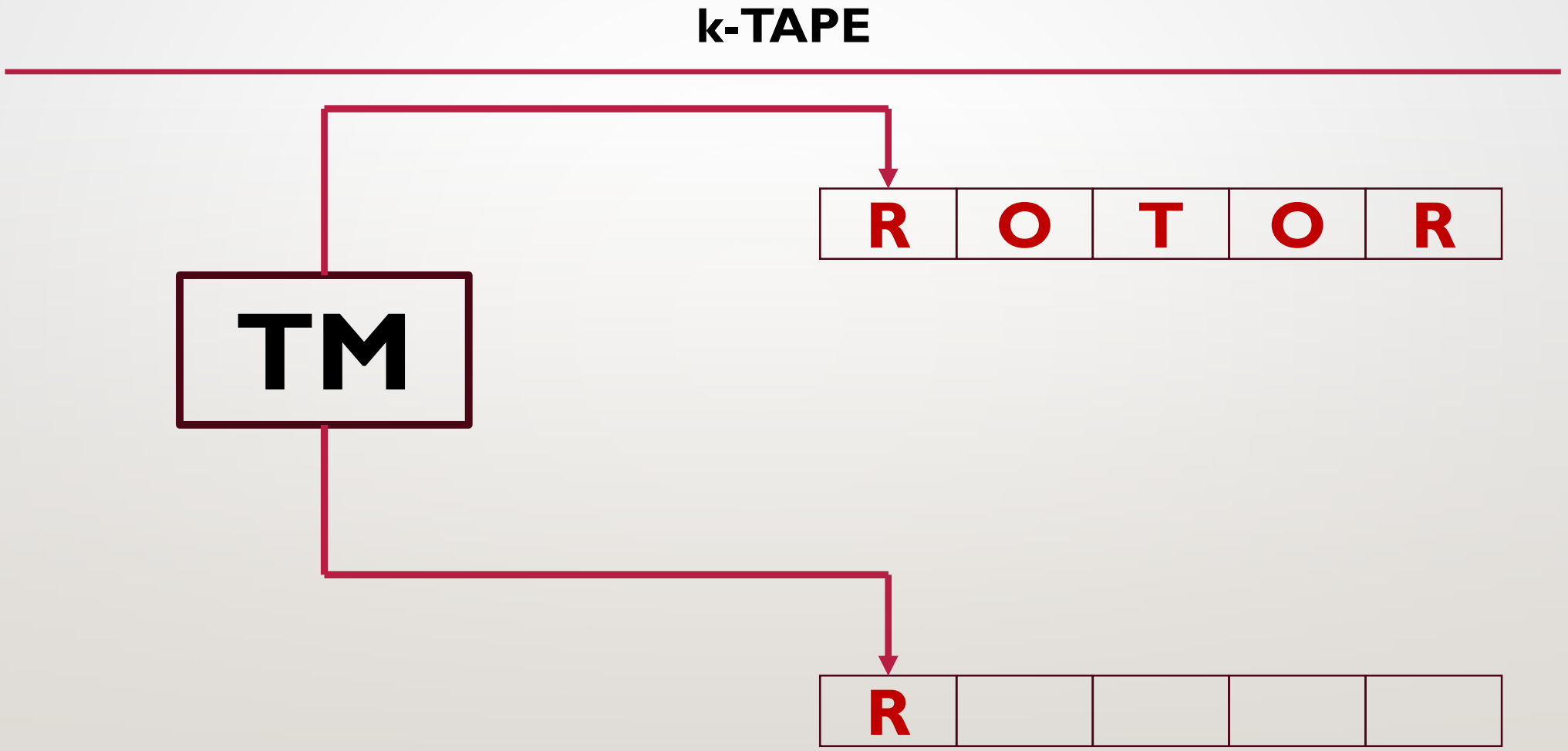


IF TIME COMPLEXITY OF A SINGLE TAPE TURING MACHINE TAKES N^2 TIME IN SAME CASE MULTI TAPE TURING MACHINE TAKE N TIME

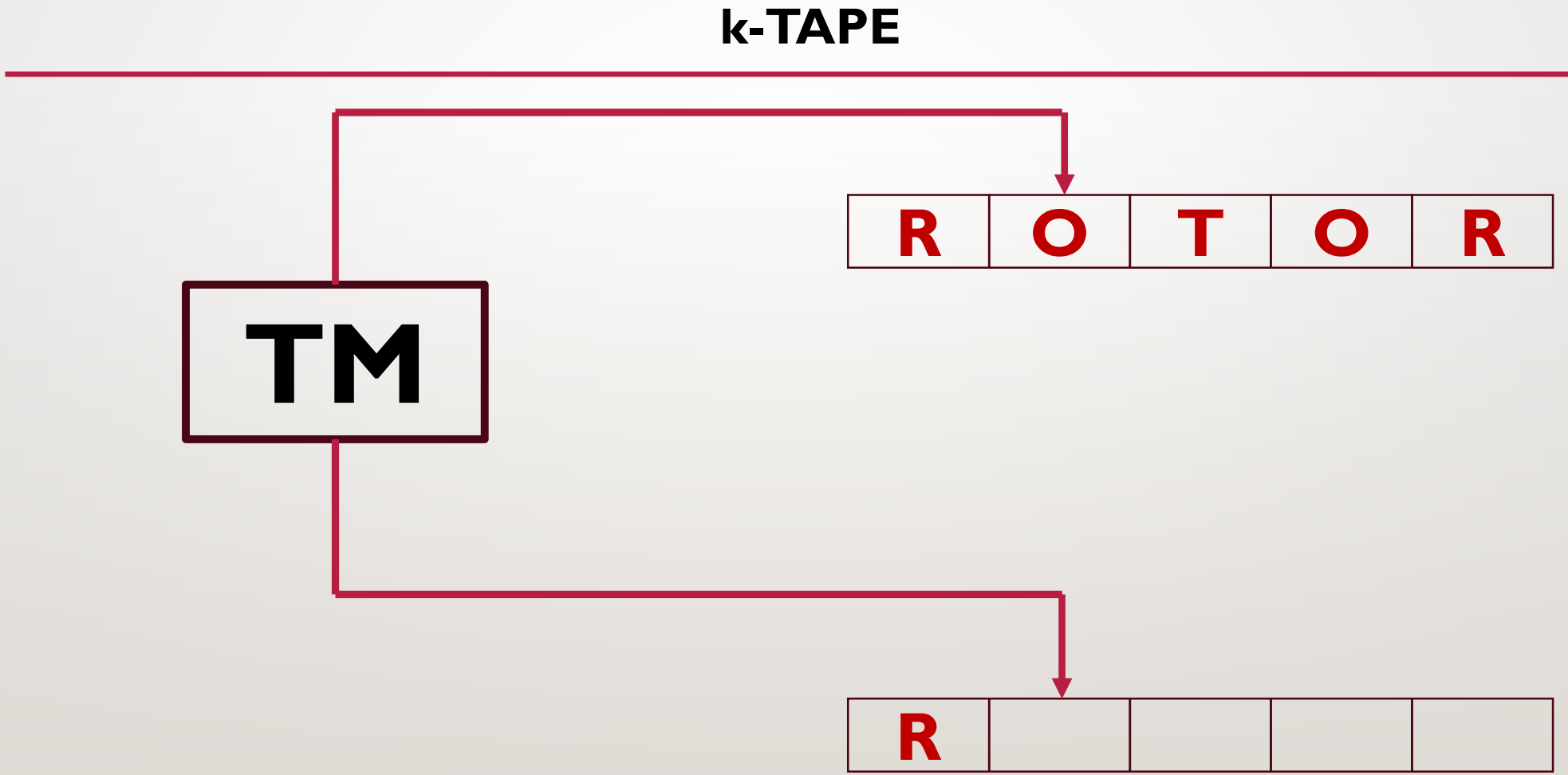
PALINDROME STRING VERIFICATION (MULTI TAPE)



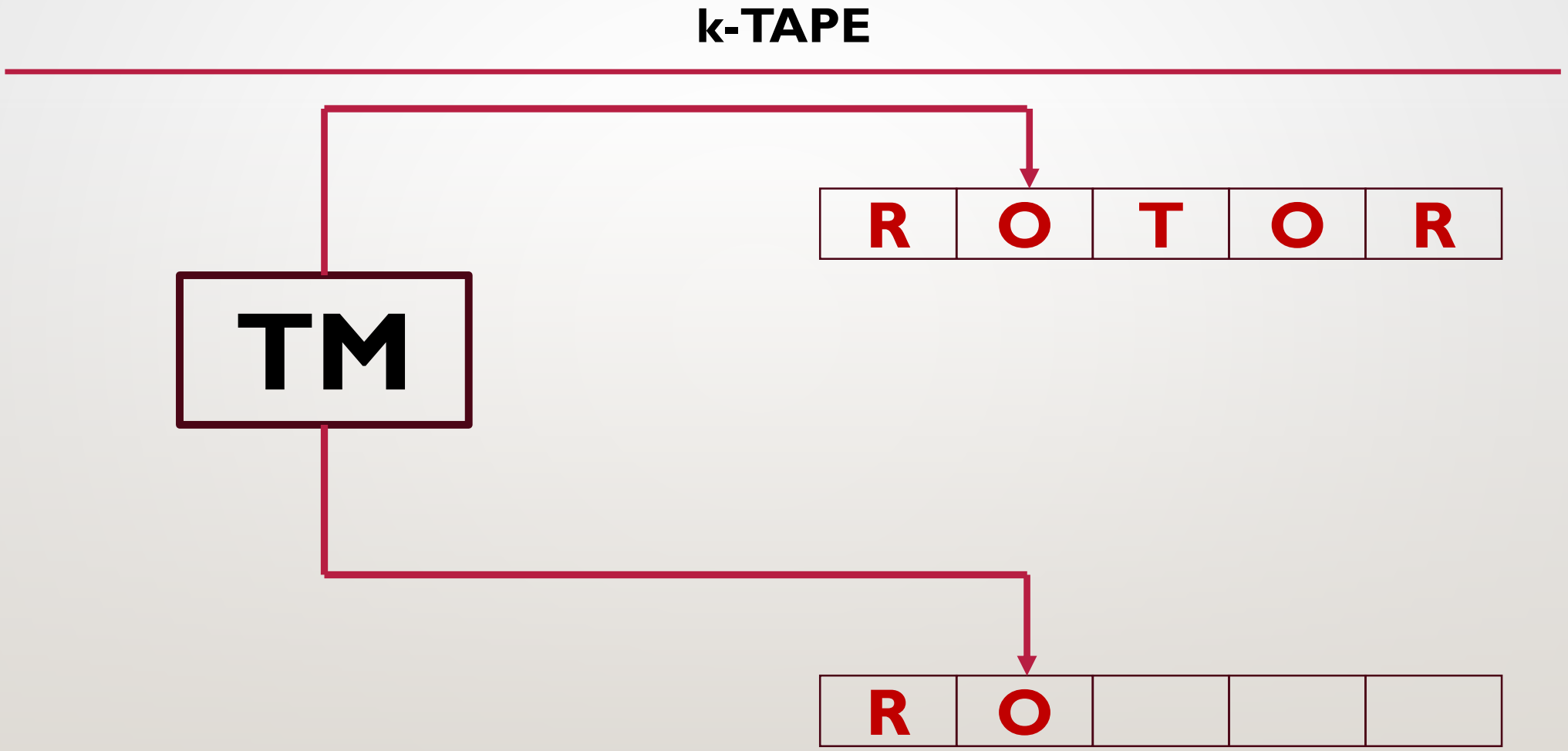
PALINDROME STRING VERIFICATION



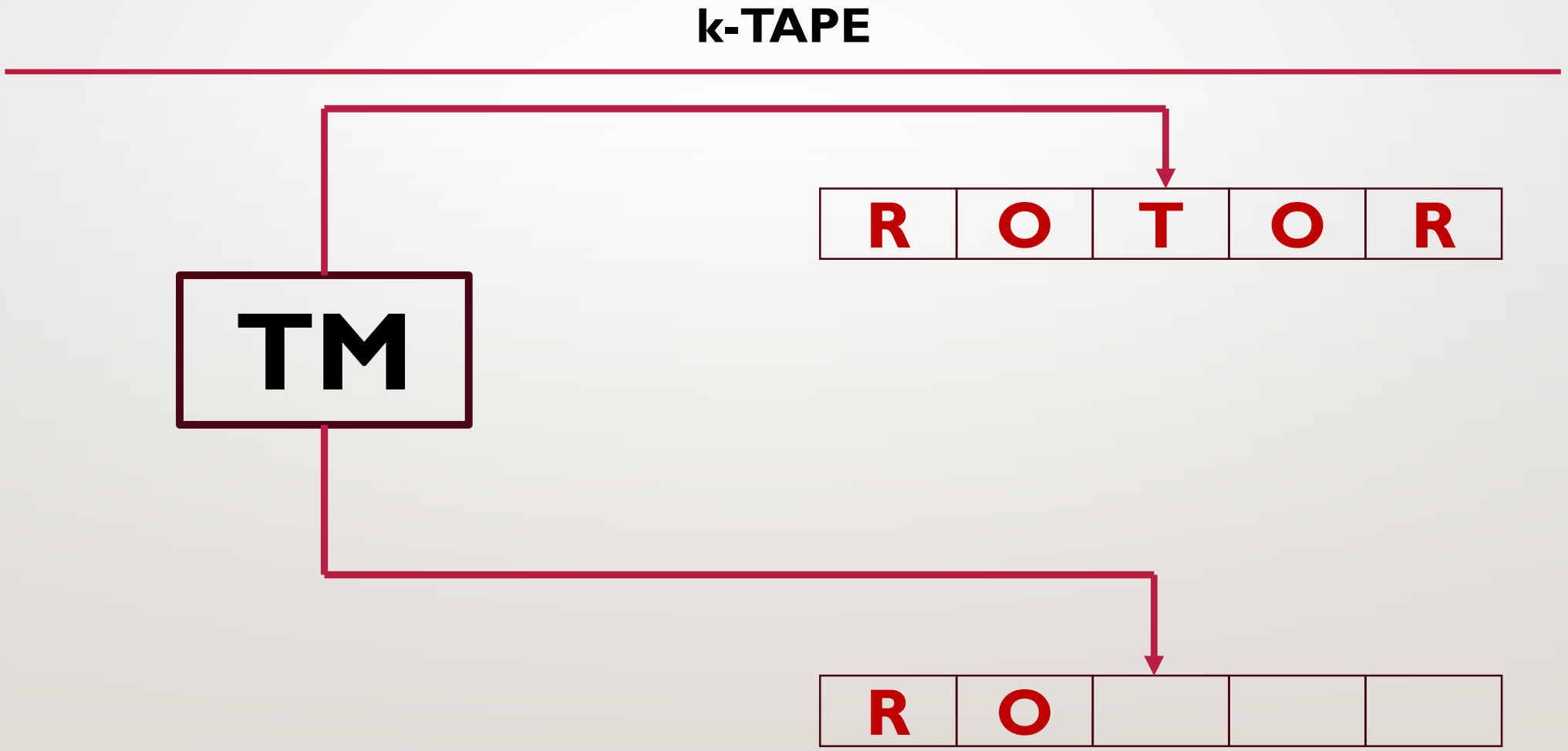
PALINDROME STRING VERIFICATION



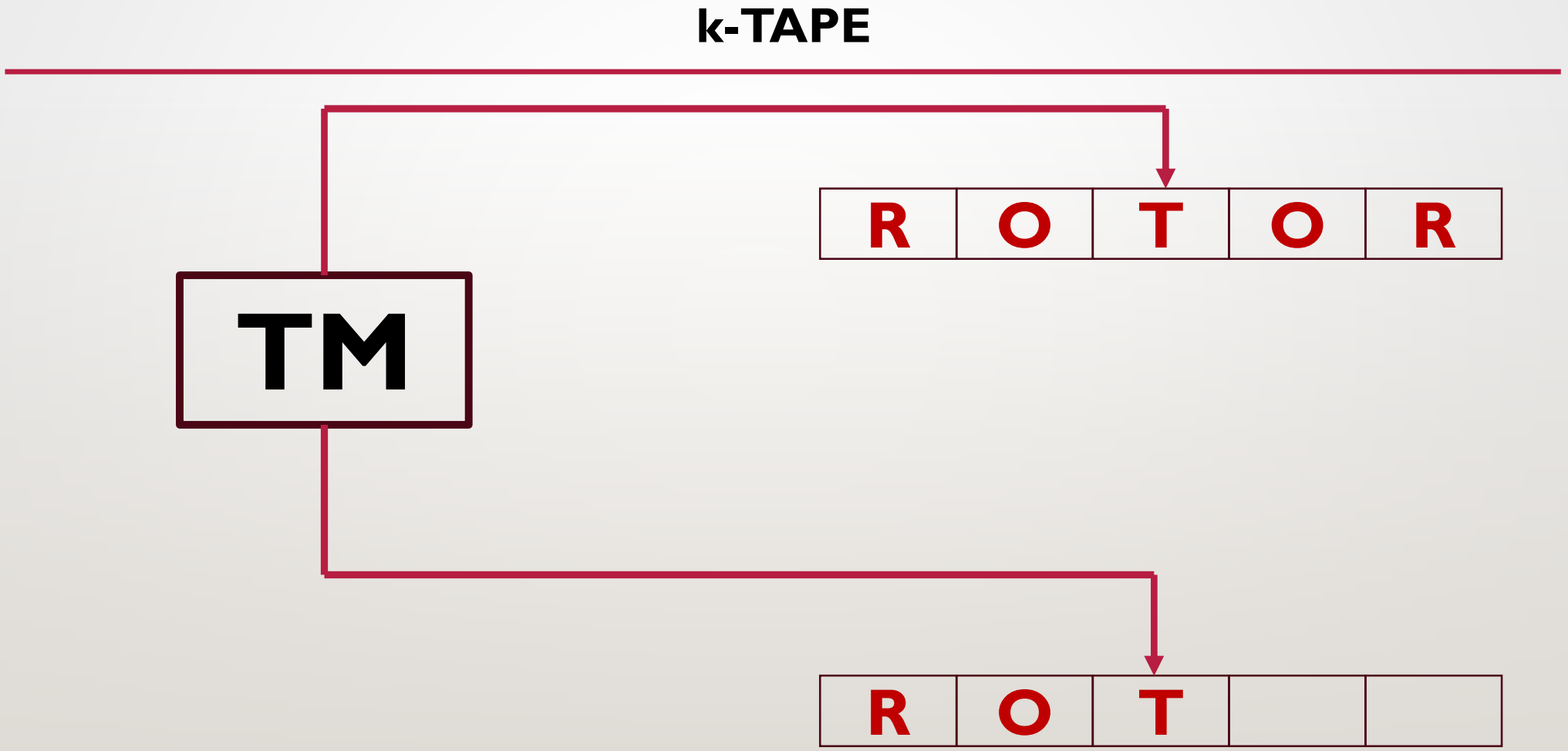
PALINDROME STRING VERIFICATION



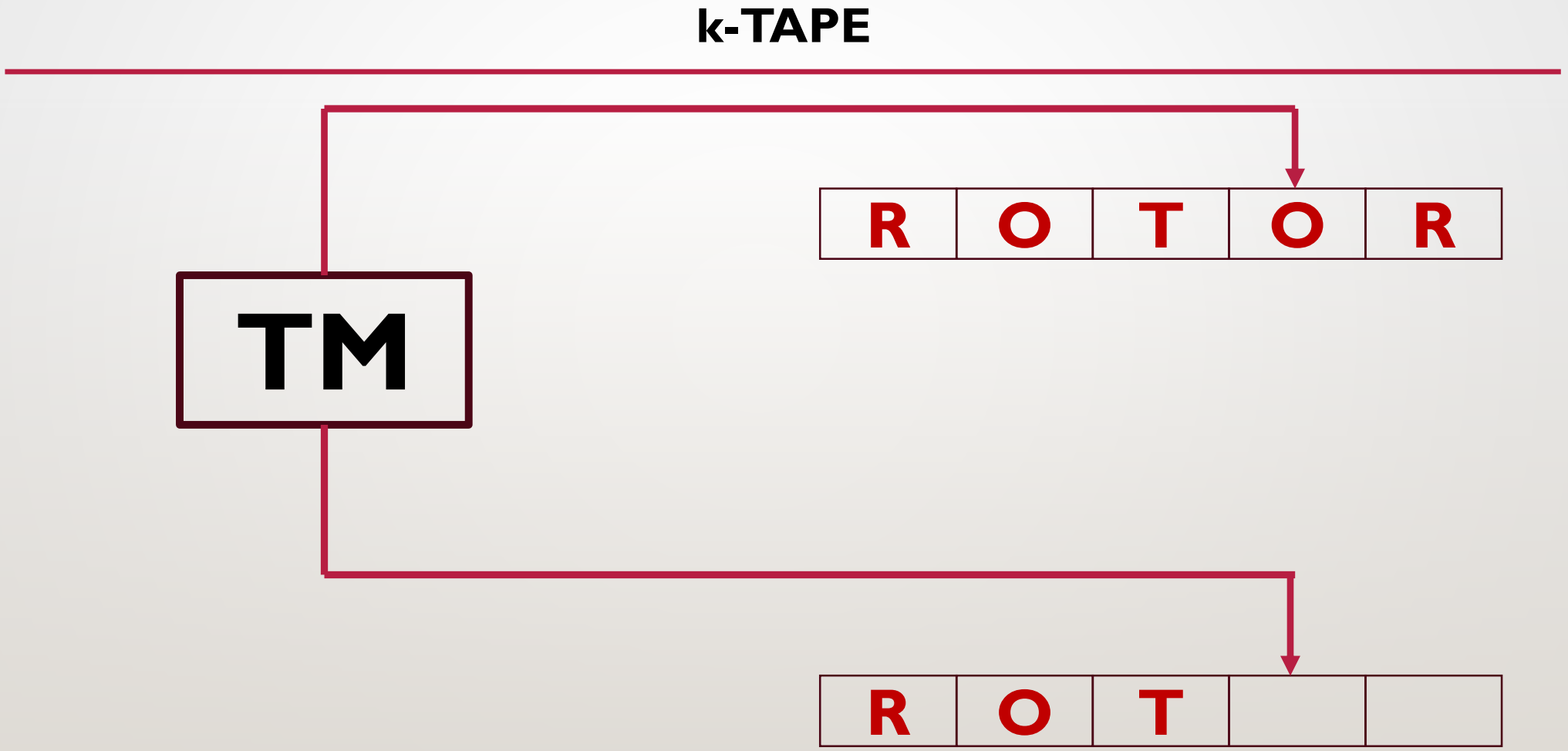
PALINDROME STRING VERIFICATION



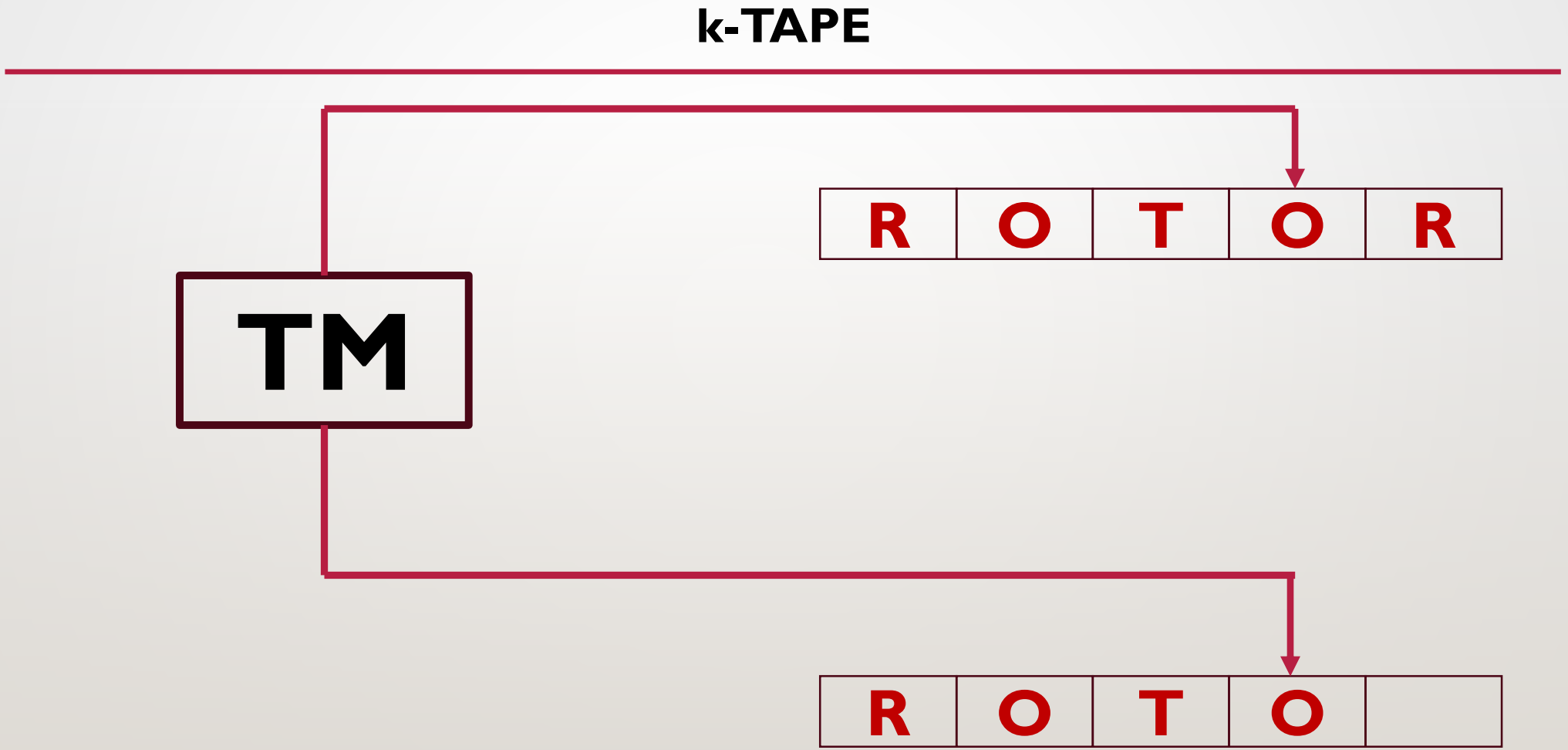
PALINDROME STRING VERIFICATION



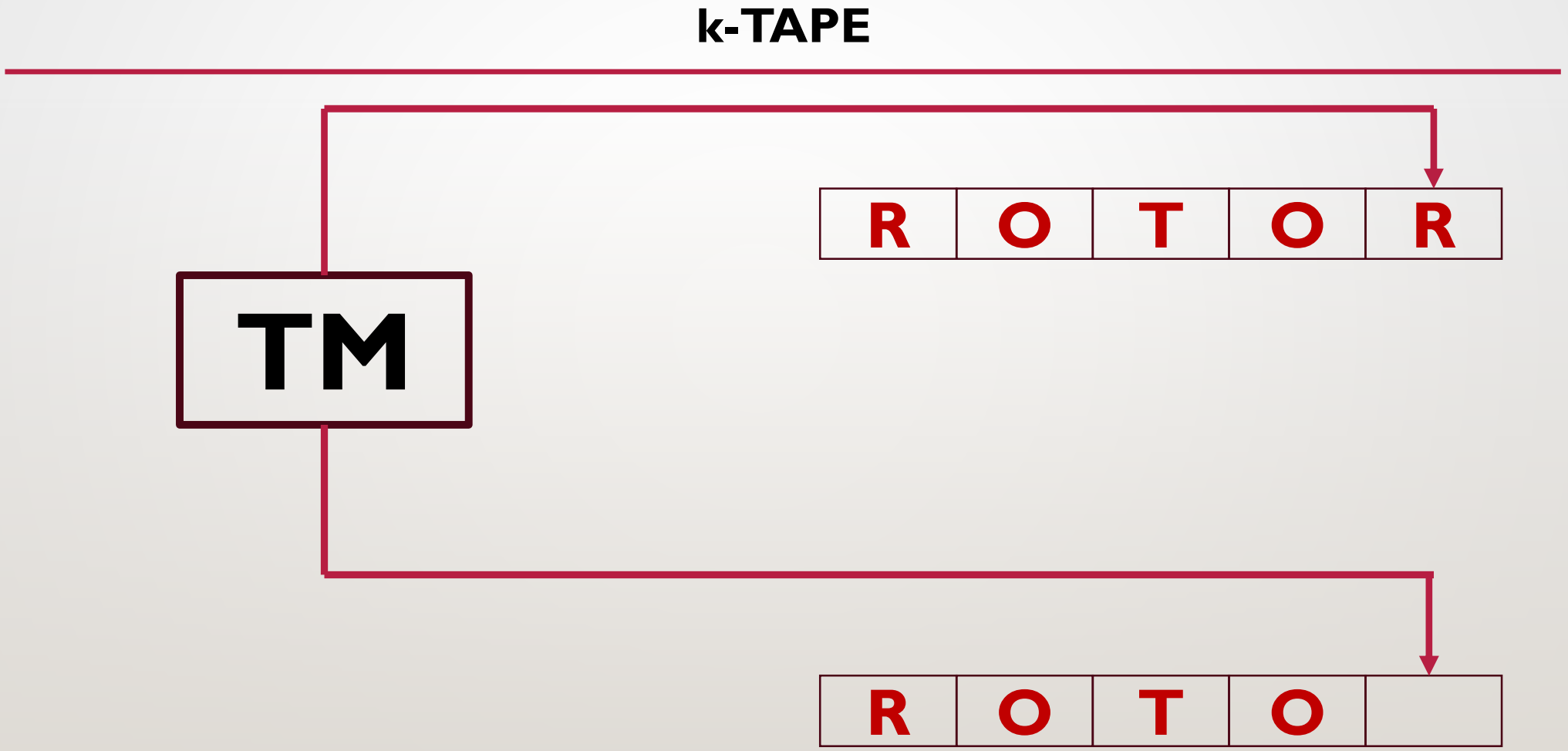
PALINDROME STRING VERIFICATION



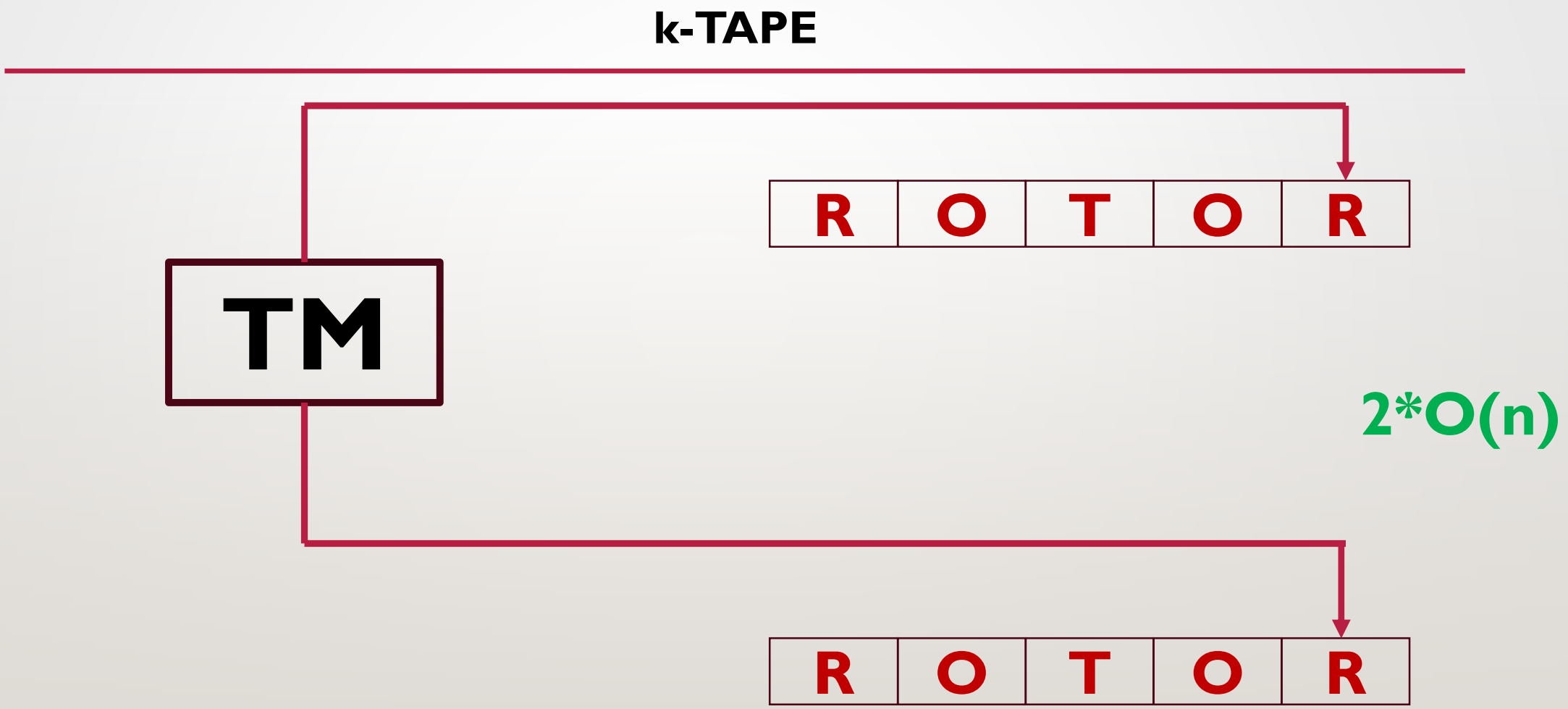
PALINDROME STRING VERIFICATION



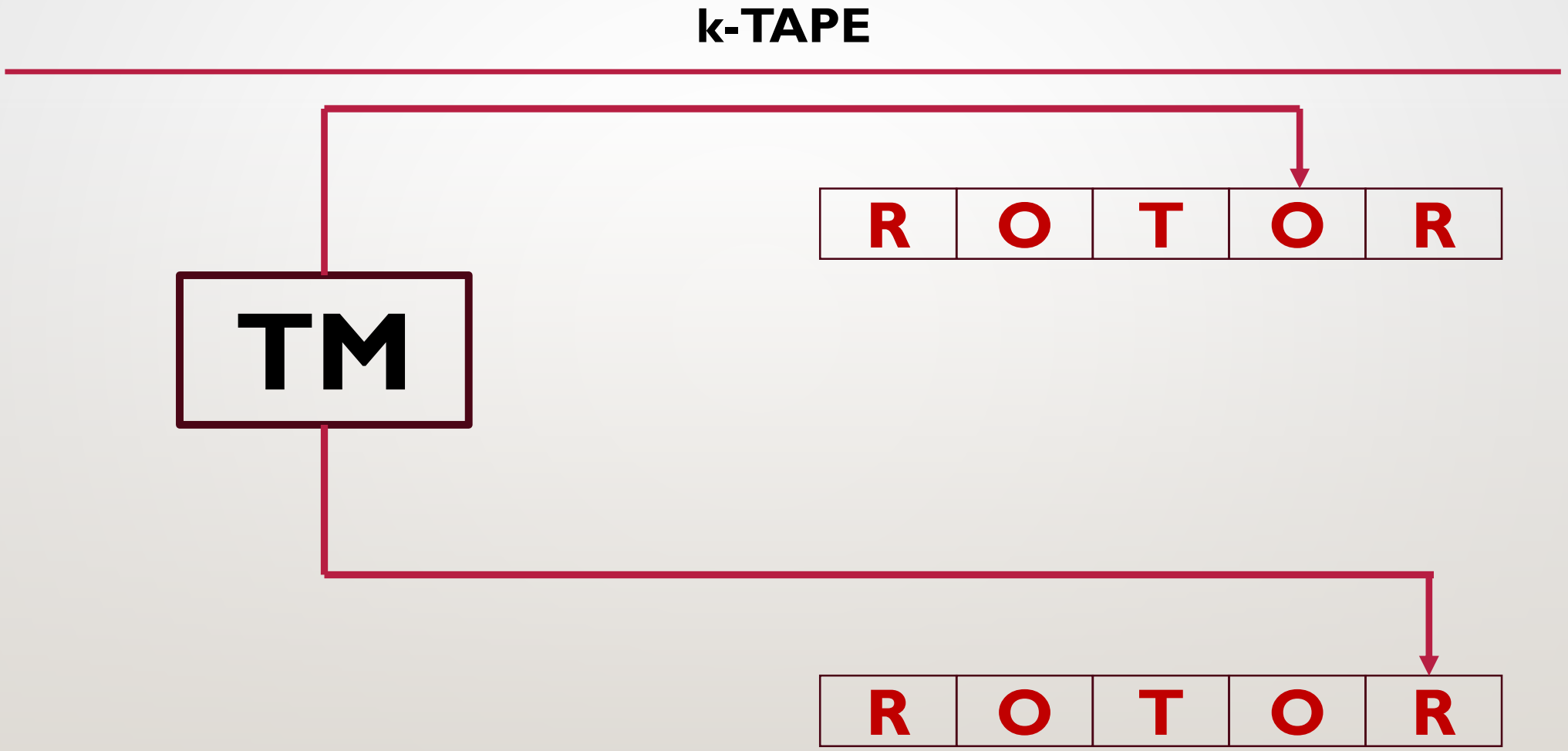
PALINDROME STRING VERIFICATION



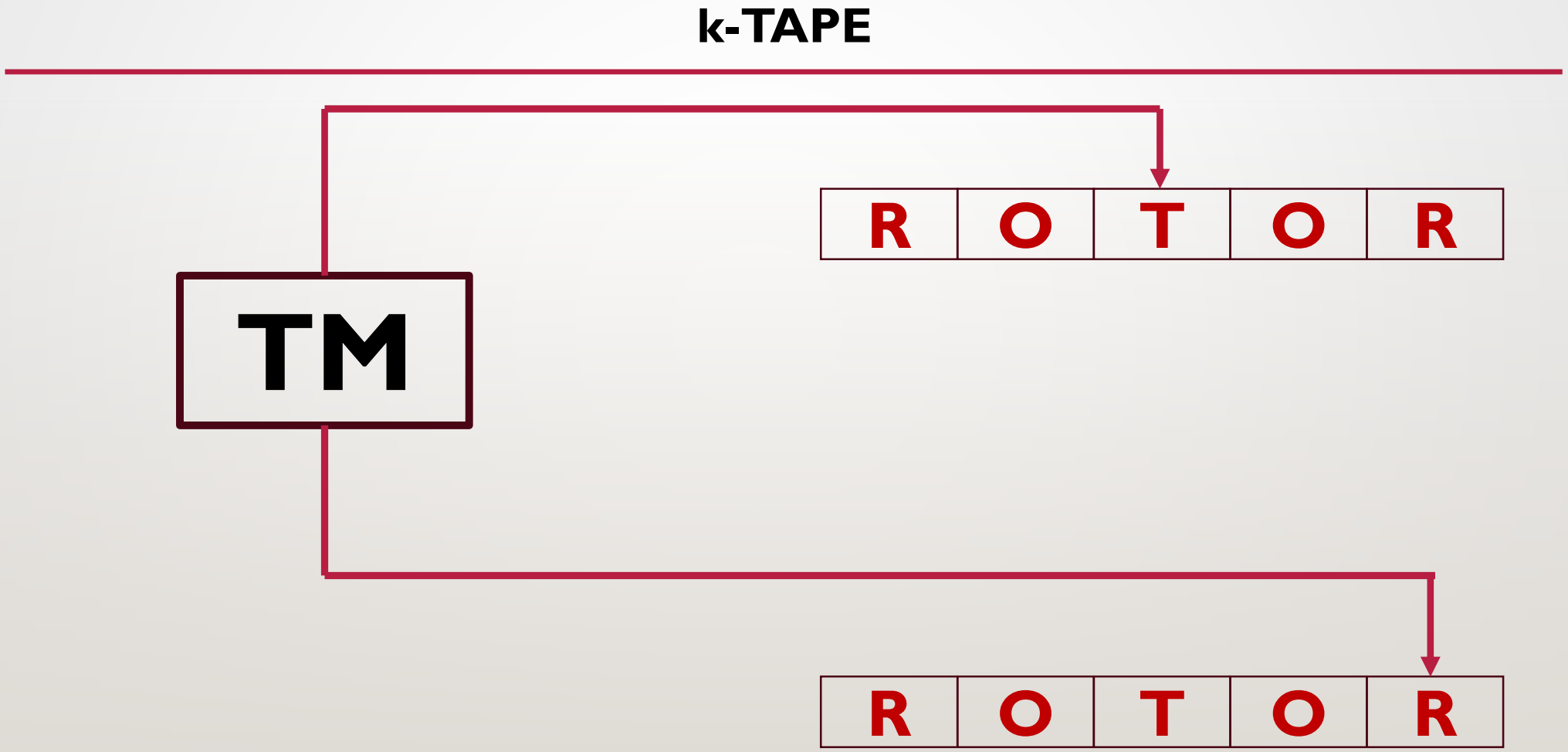
PALINDROME STRING VERIFICATION



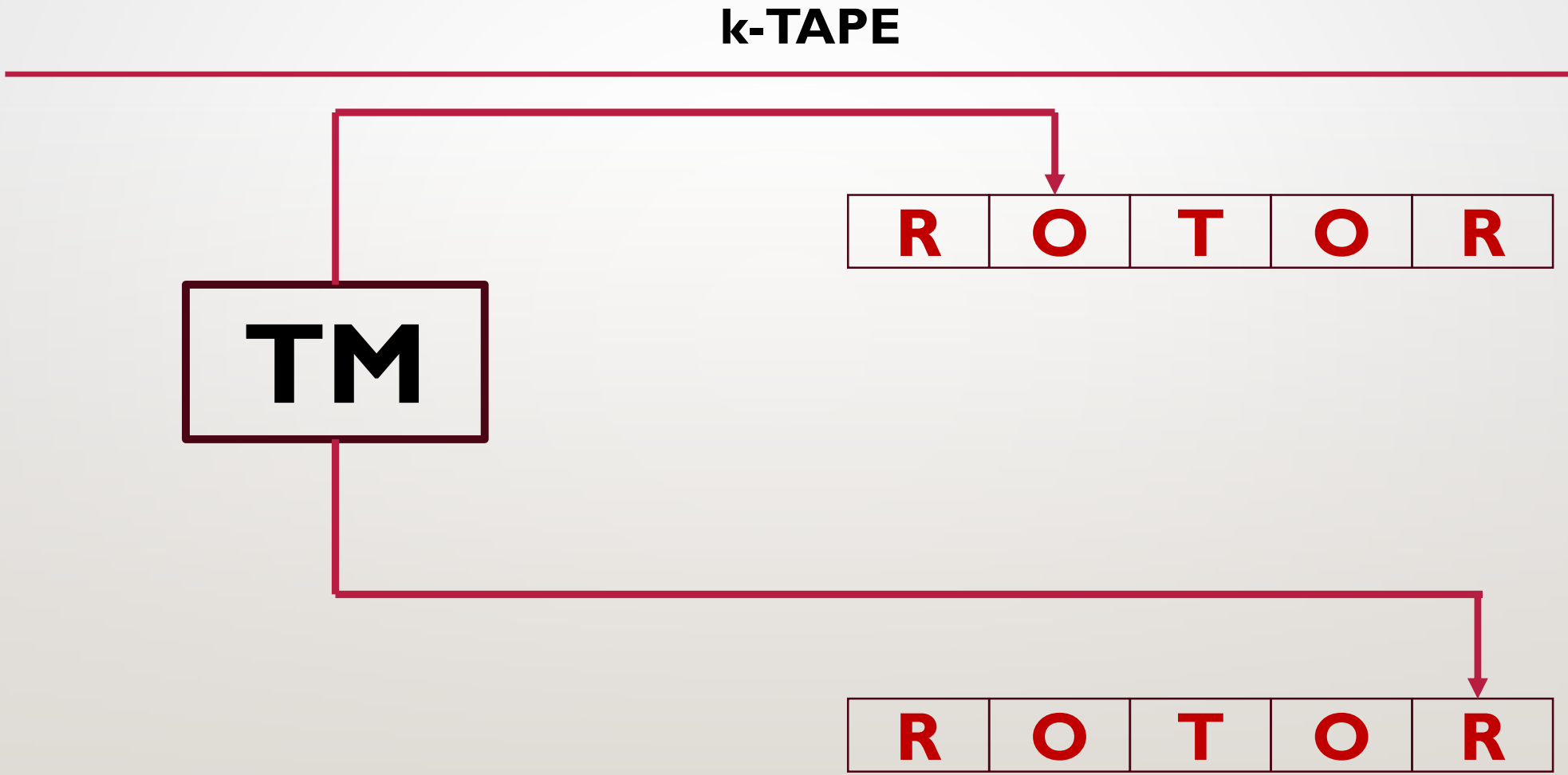
PALINDROME STRING VERIFICATION



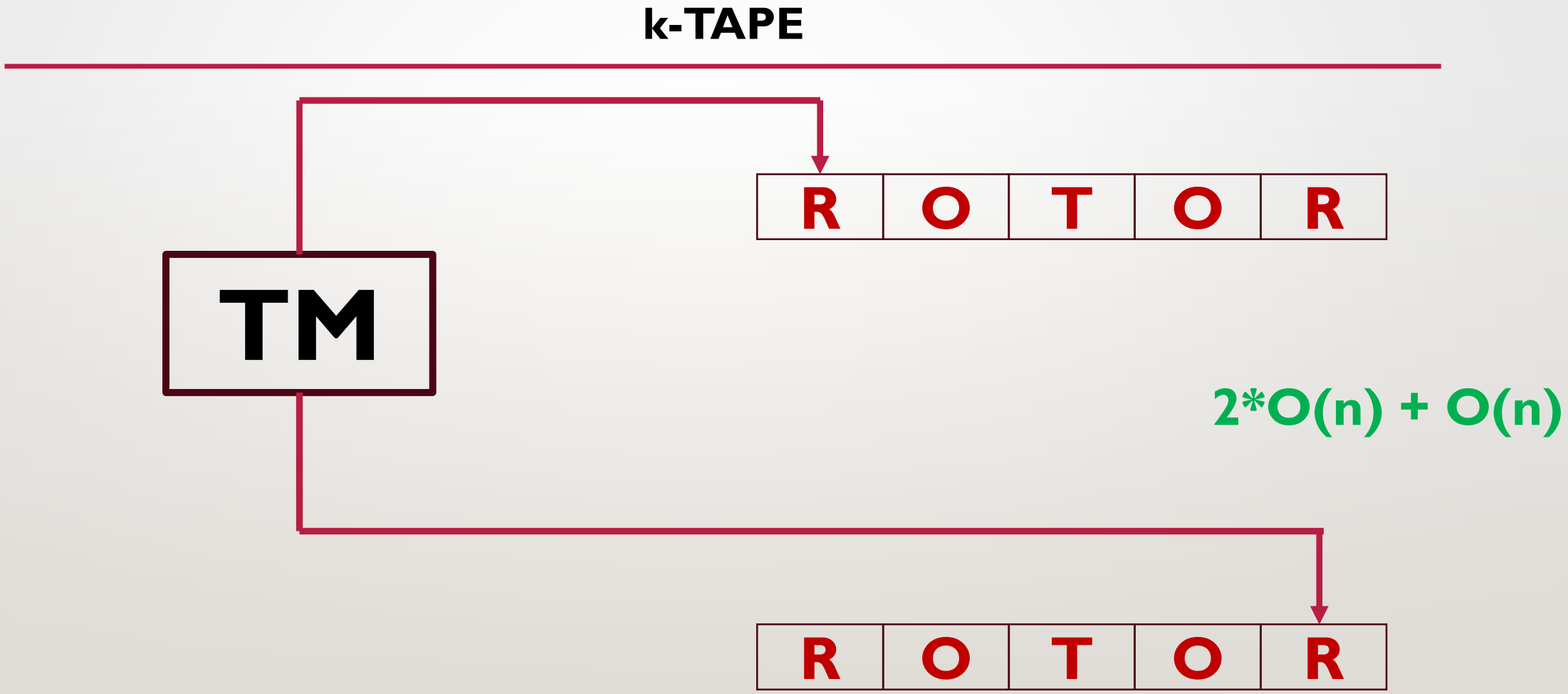
PALINDROME STRING VERIFICATION



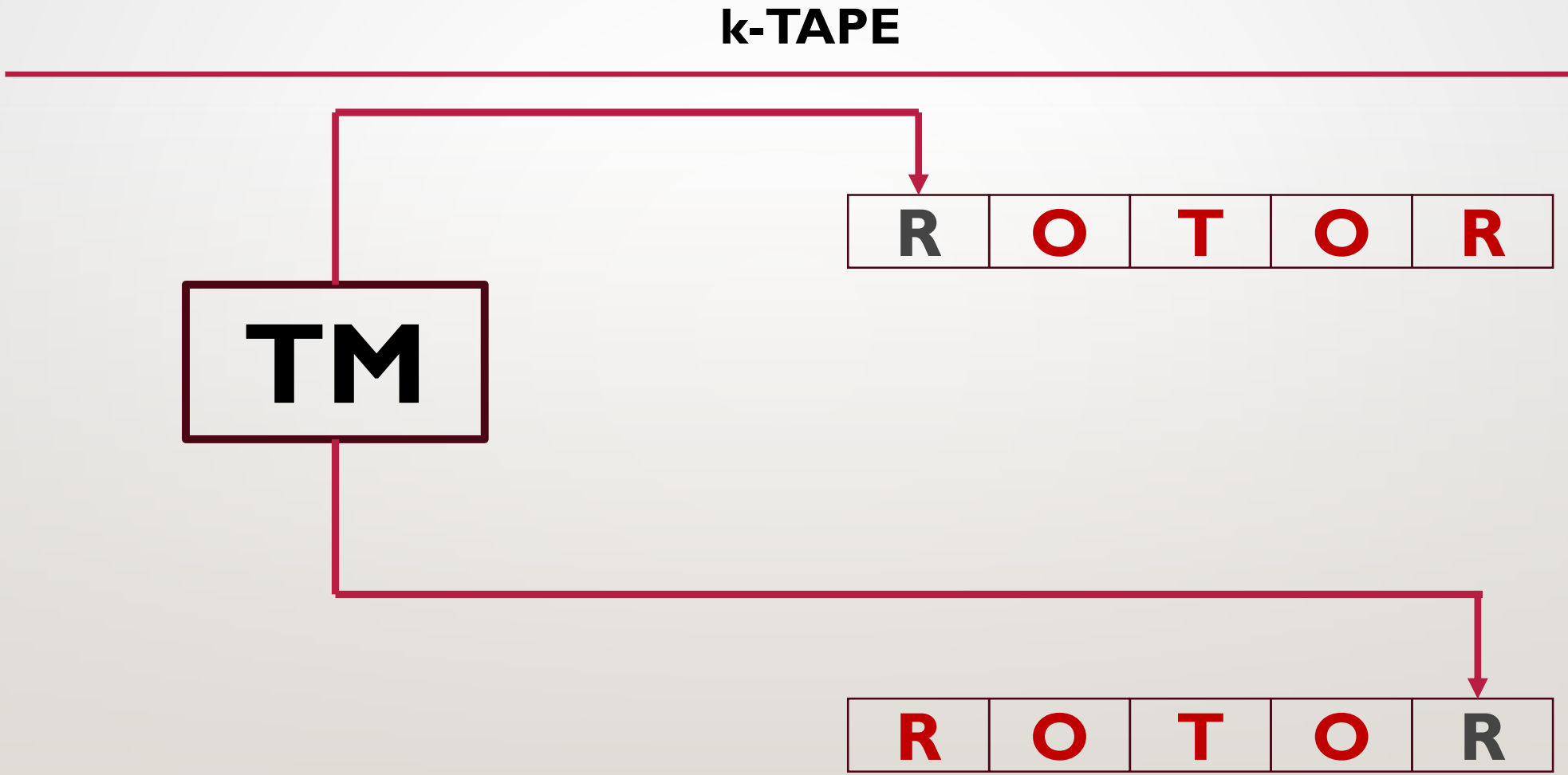
PALINDROME STRING VERIFICATION



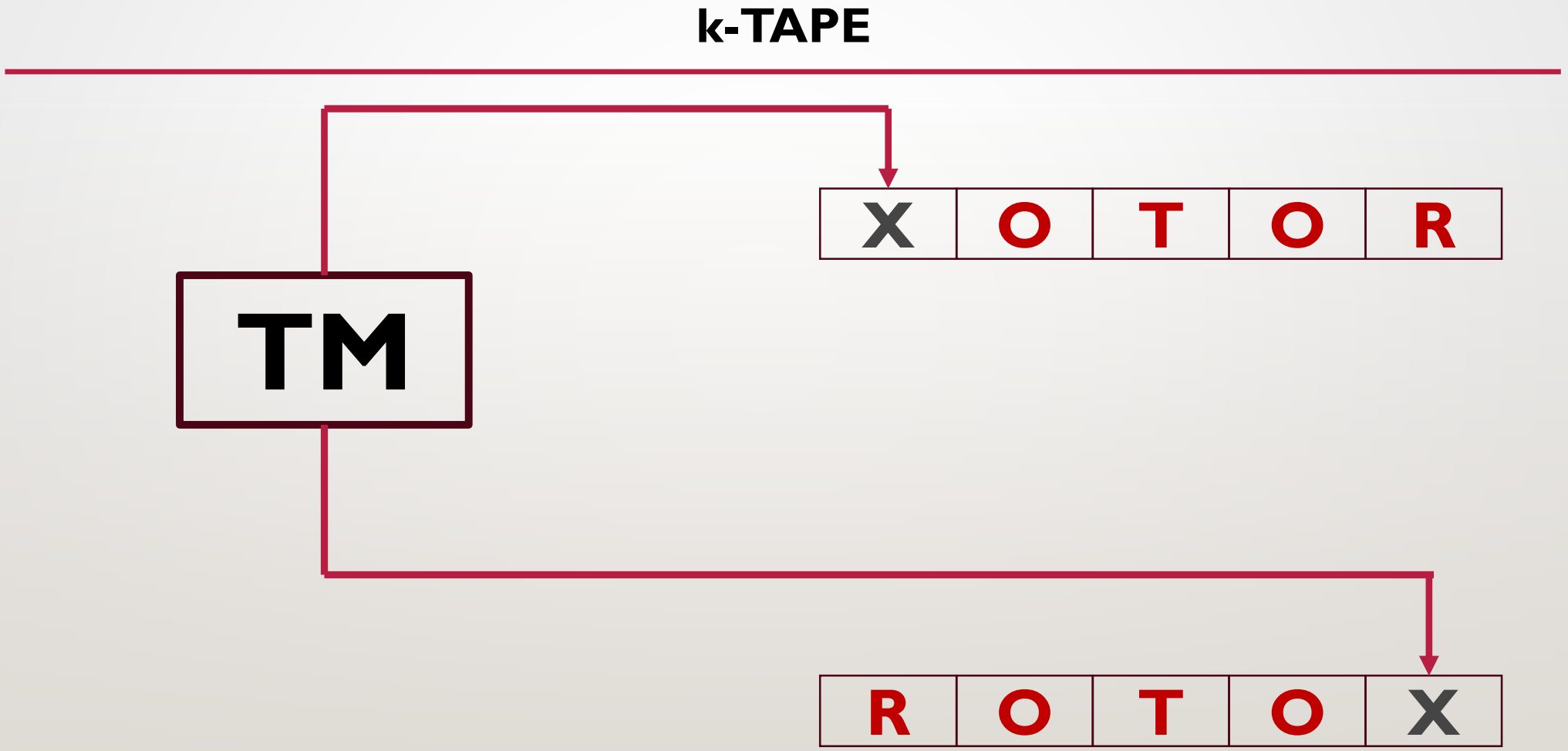
PALINDROME STRING VERIFICATION



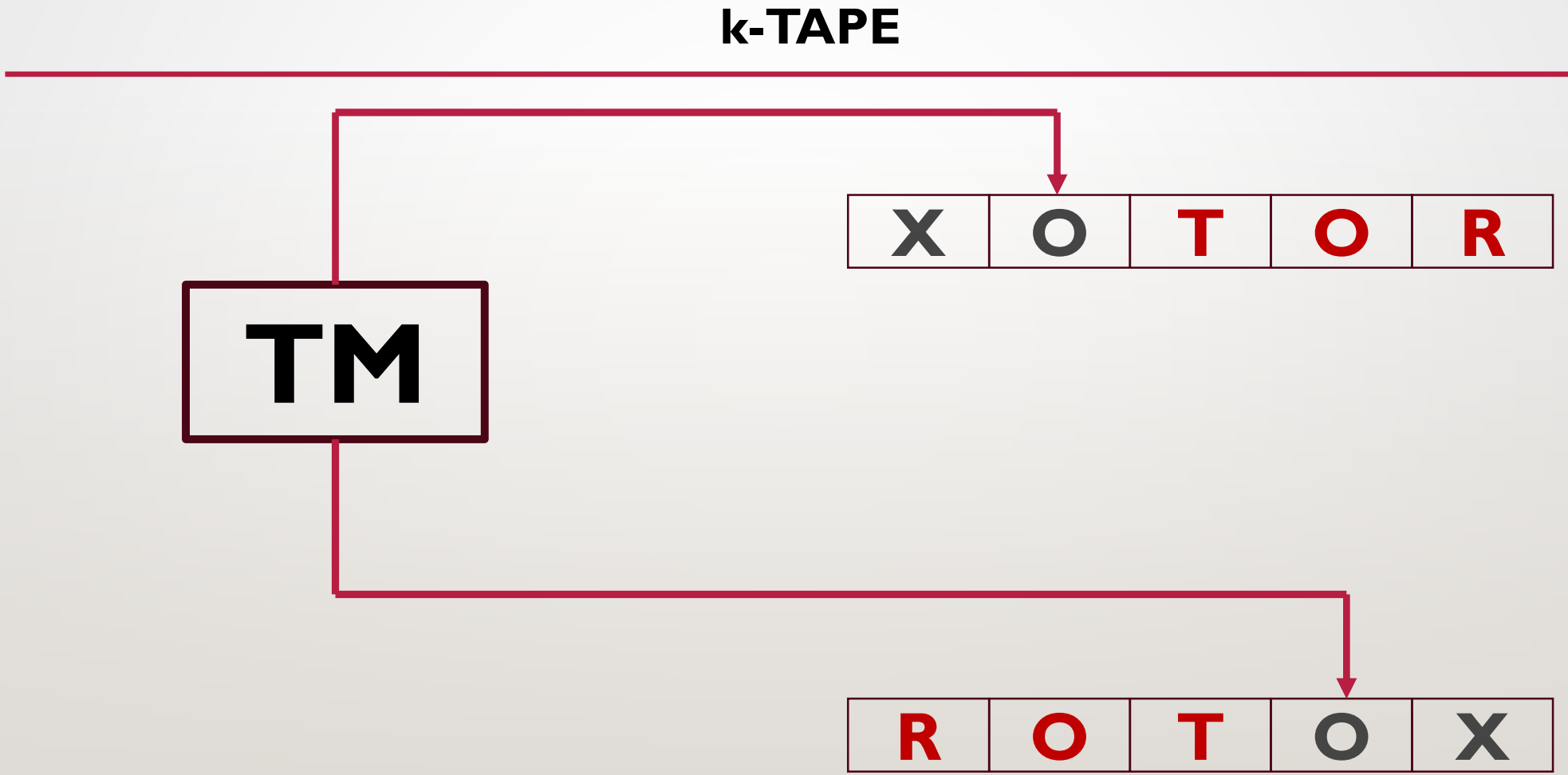
PALINDROME STRING VERIFICATION



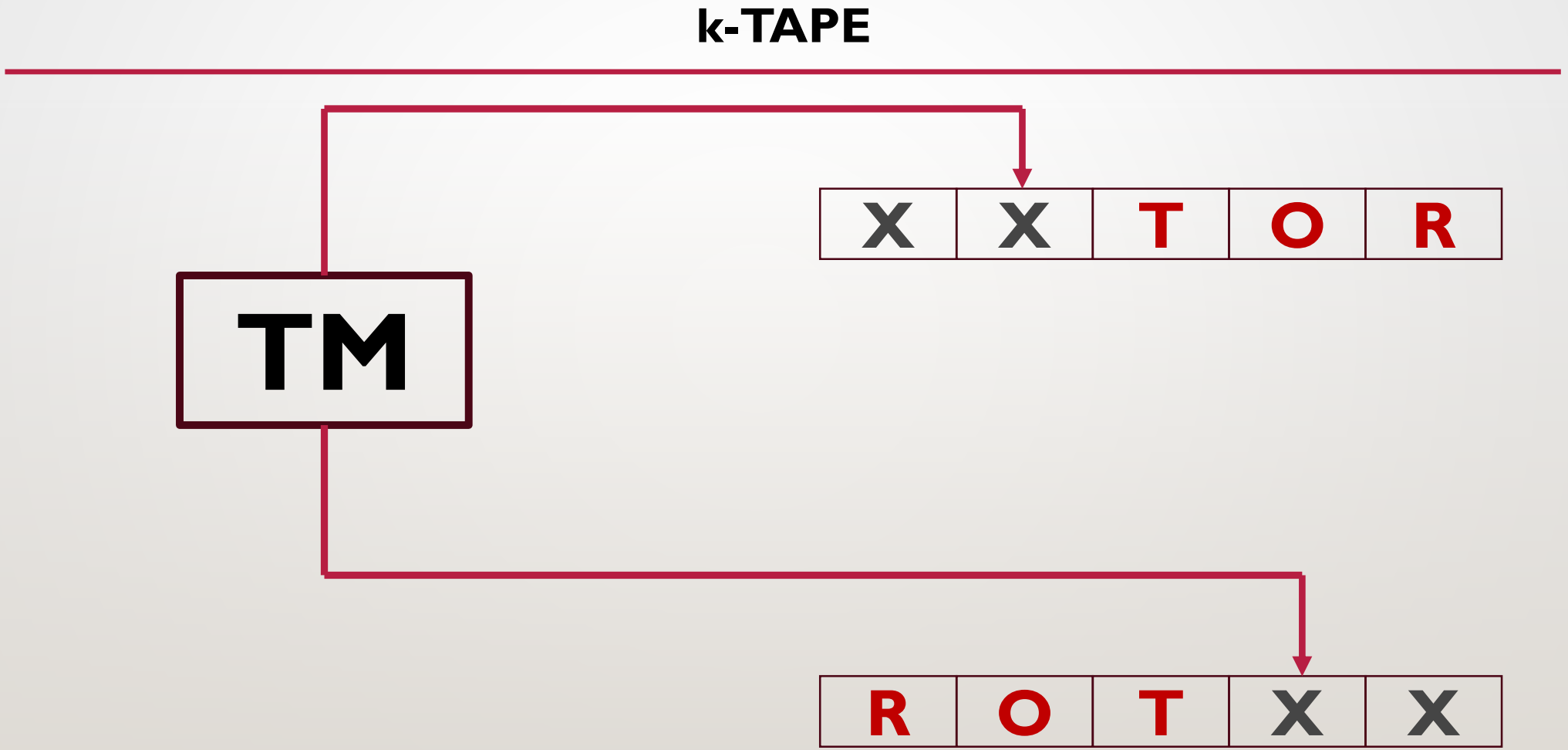
PALINDROME STRING VERIFICATION



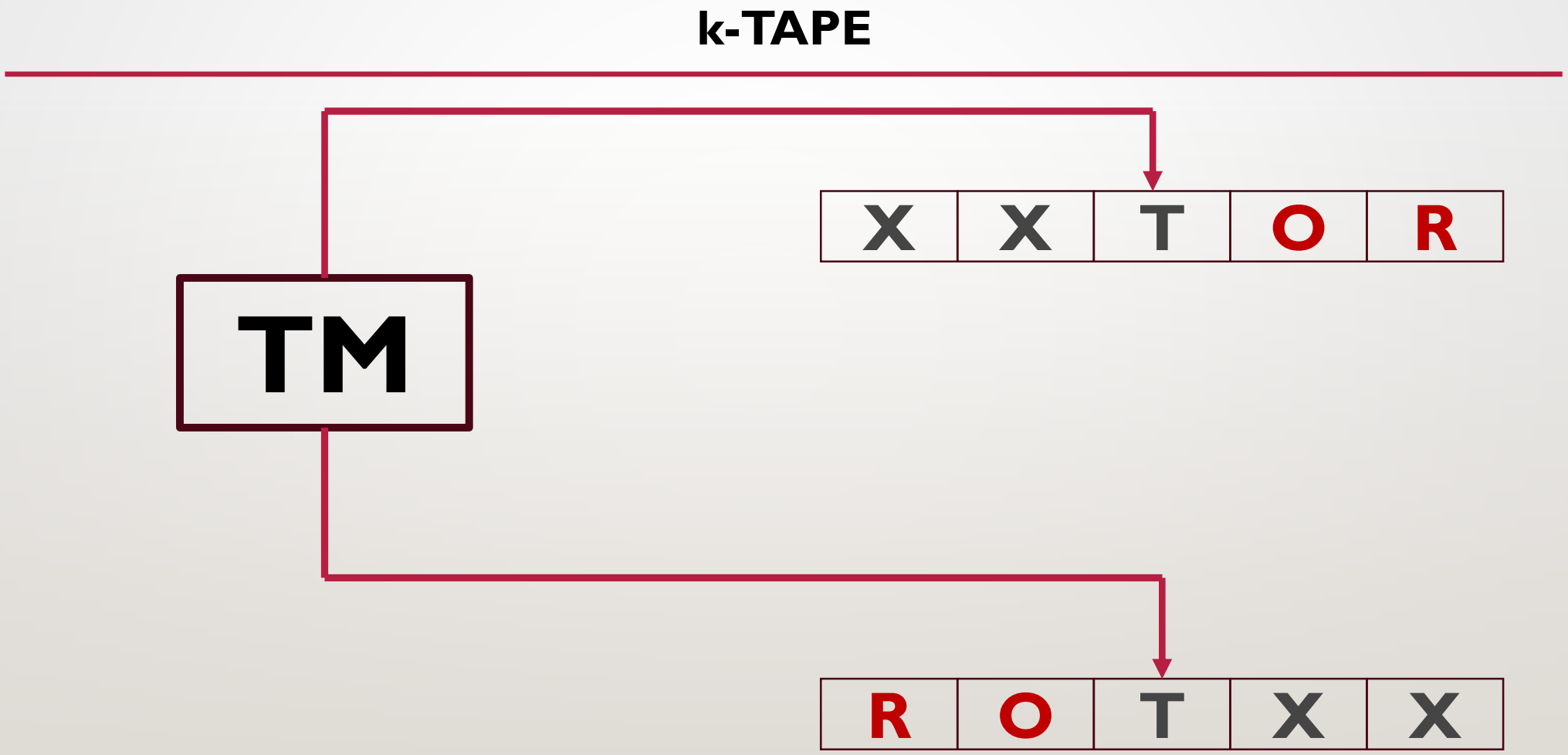
PALINDROME STRING VERIFICATION



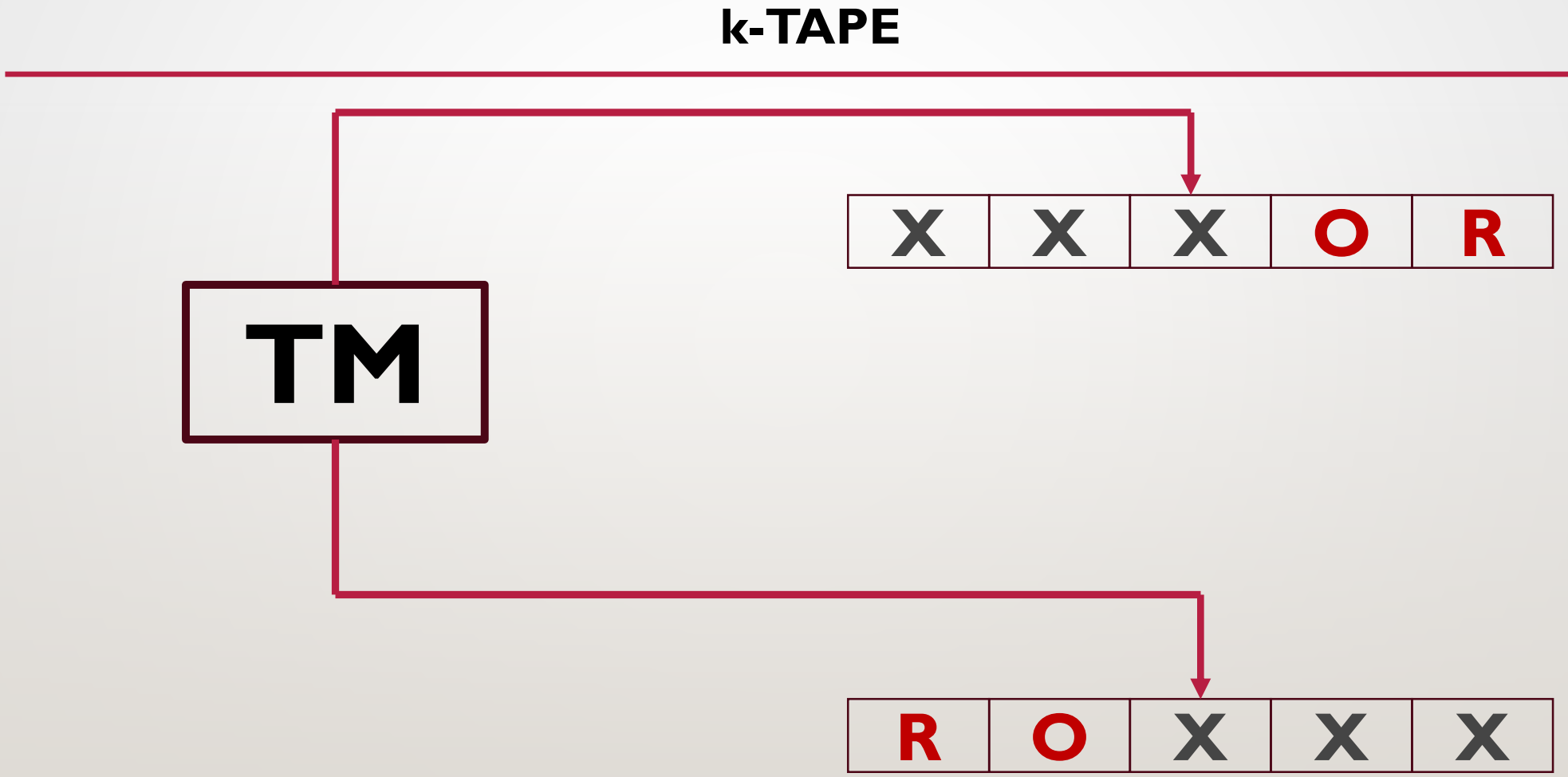
PALINDROME STRING VERIFICATION



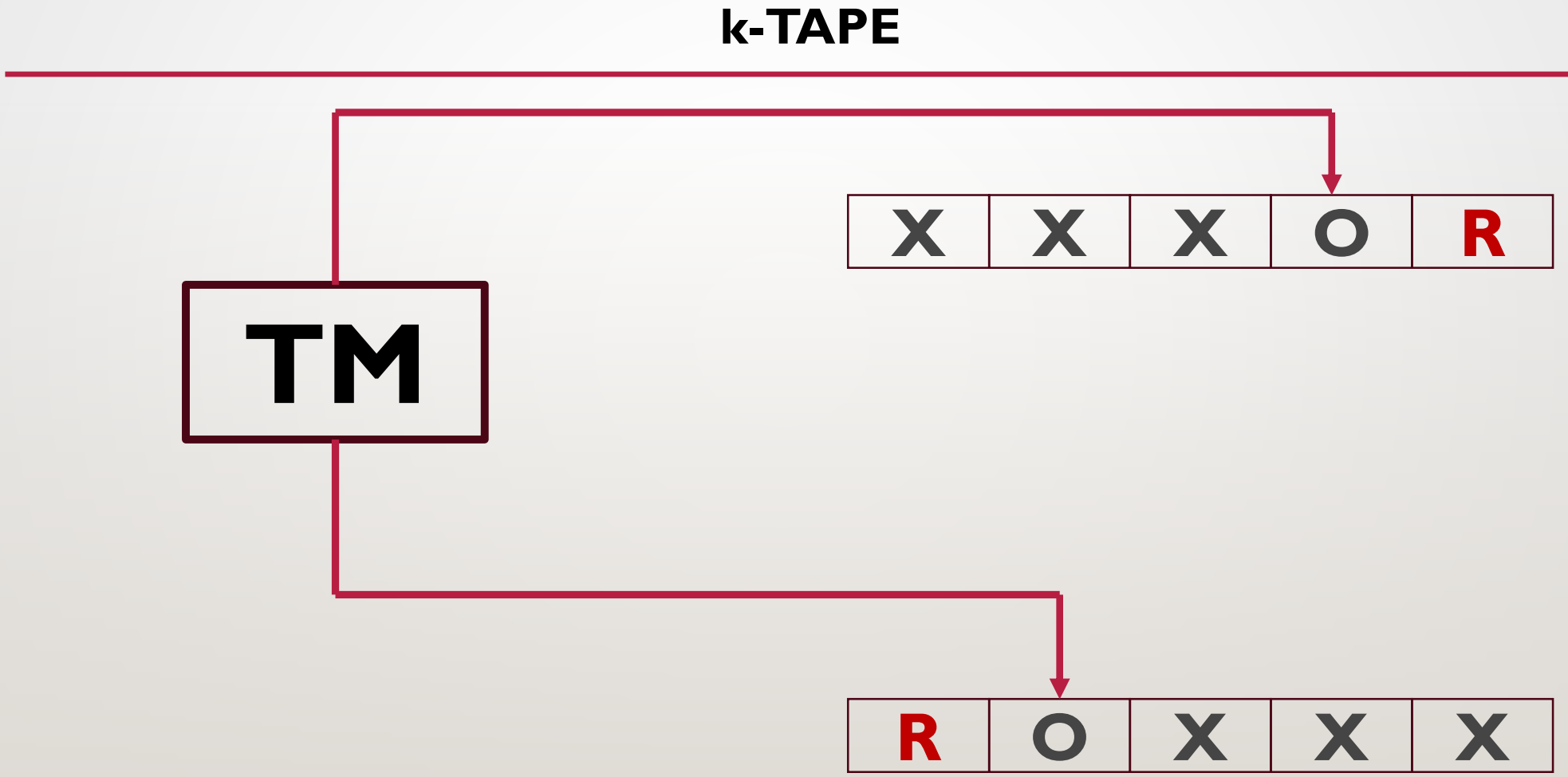
PALINDROME STRING VERIFICATION



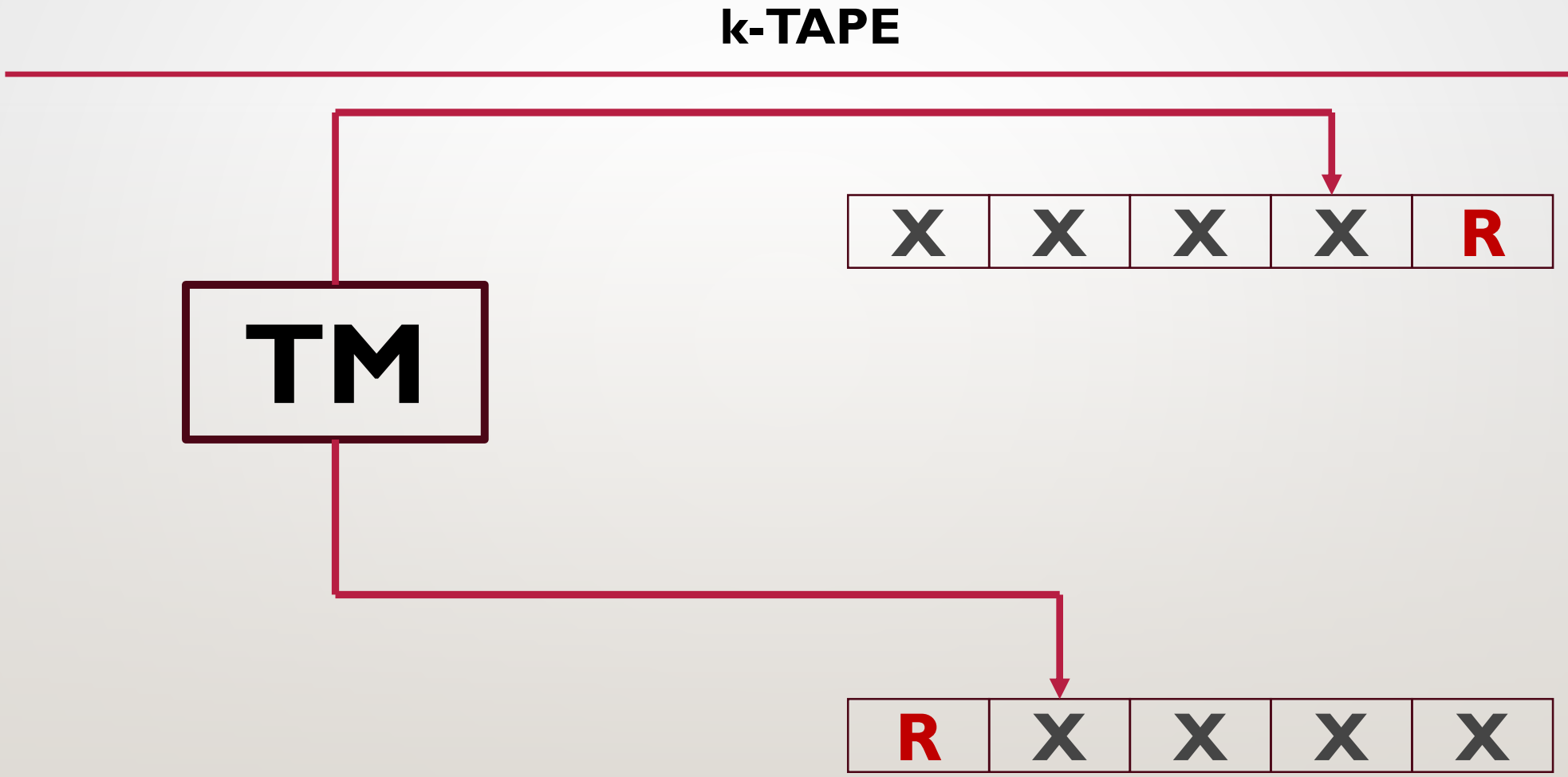
PALINDROME STRING VERIFICATION



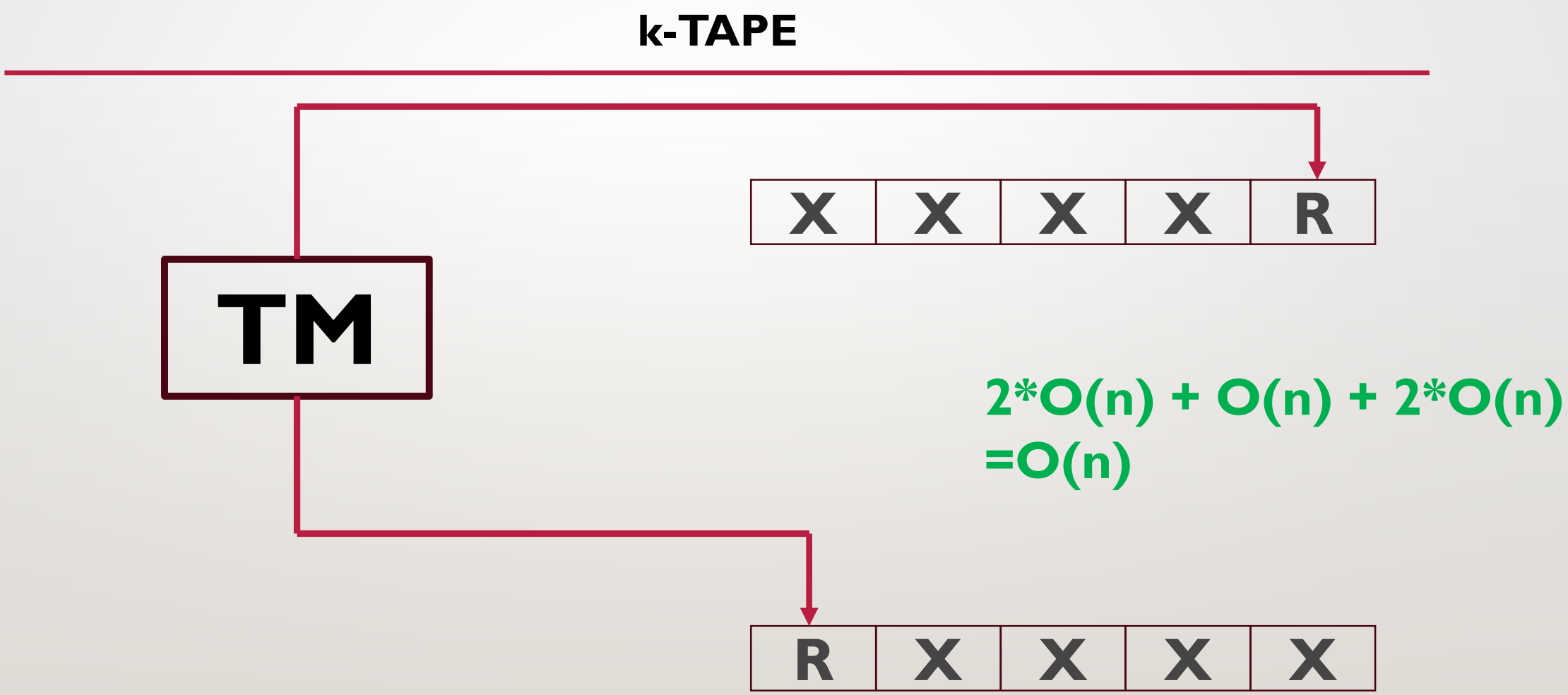
PALINDROME STRING VERIFICATION



PALINDROME STRING VERIFICATION

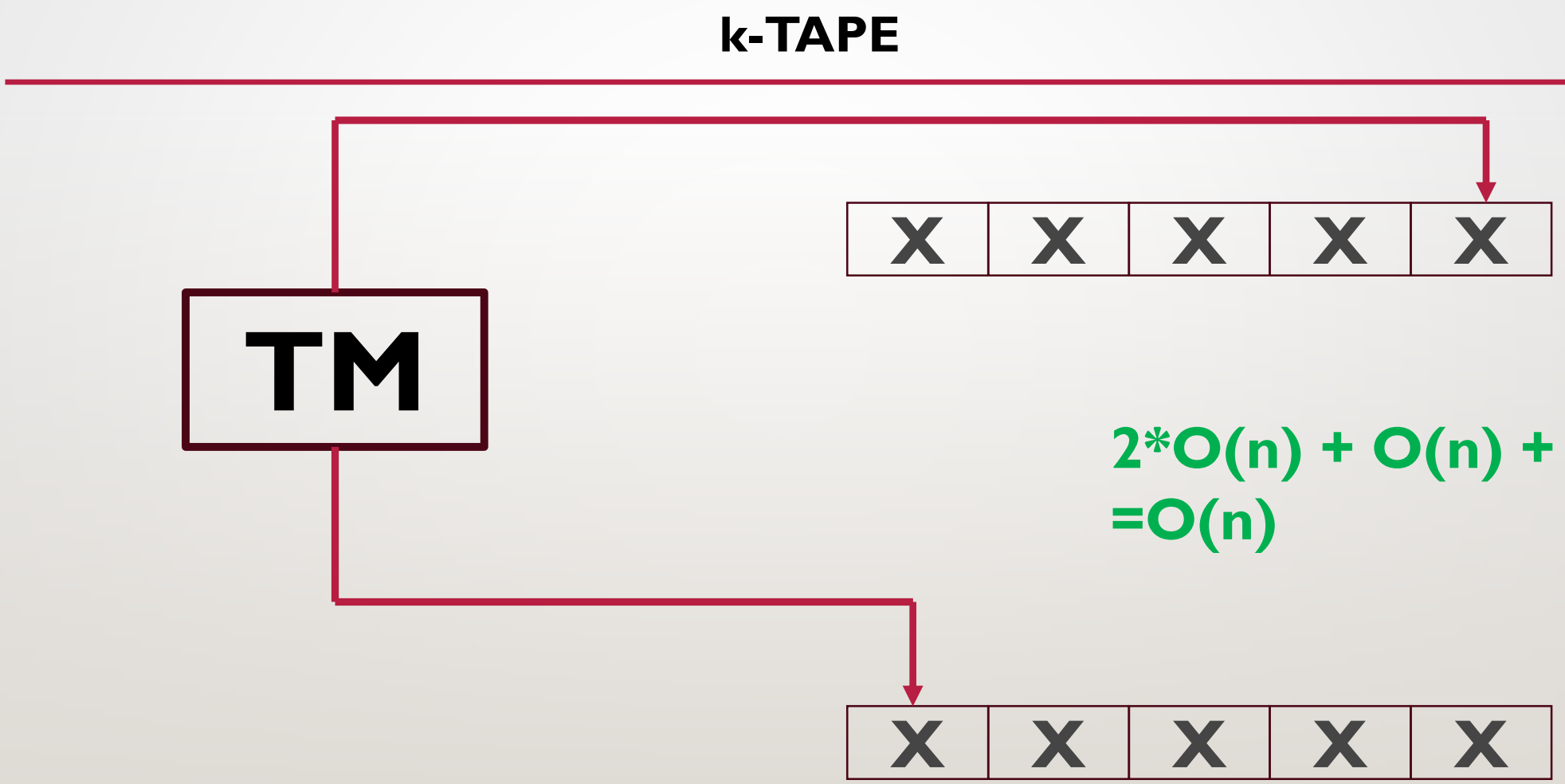


PALINDROME STRING VERIFICATION



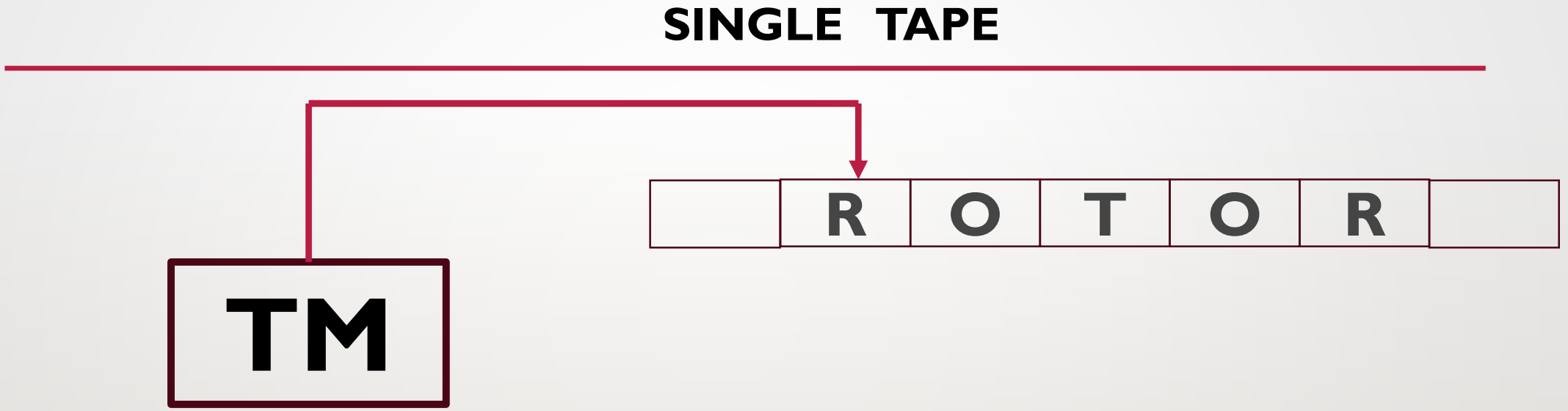
$2*O(n) + O(n) + 2*O(n)$
$=O(n)$

PALINDROME STRING VERIFICATION

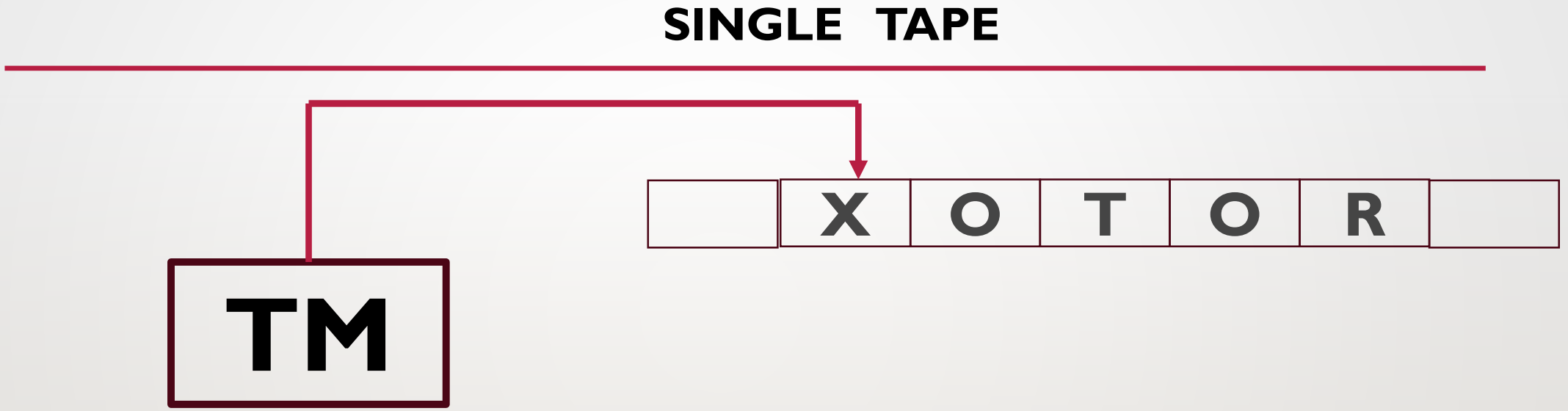


$2*O(n) + O(n) + 2*O(n)$
$=O(n)$

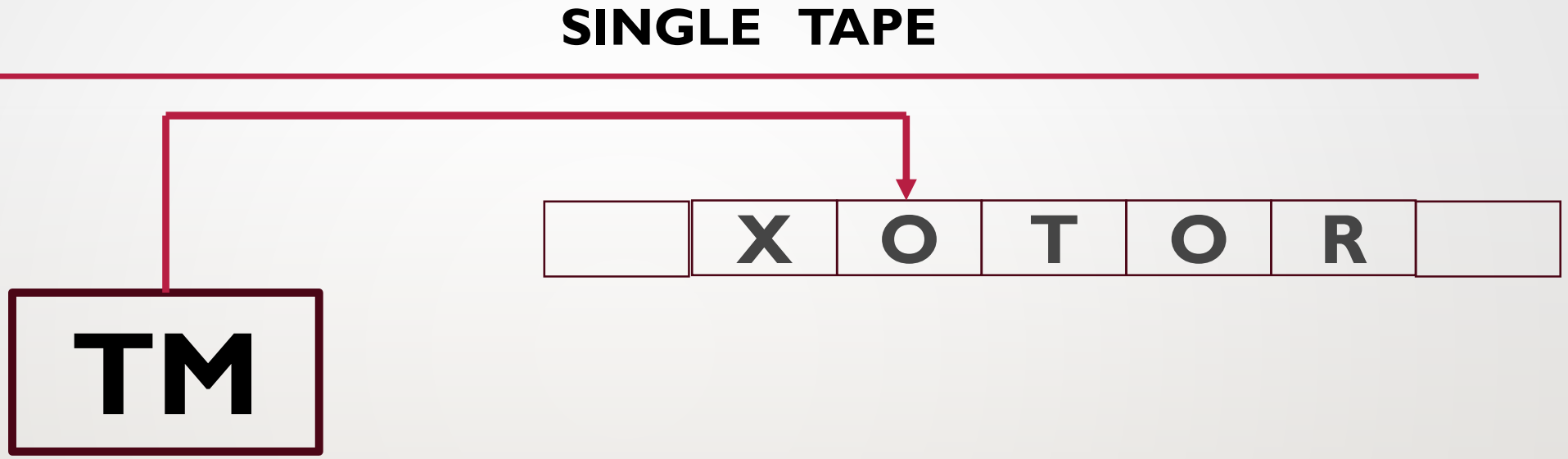
PALINDROME STRING VERIFICATION



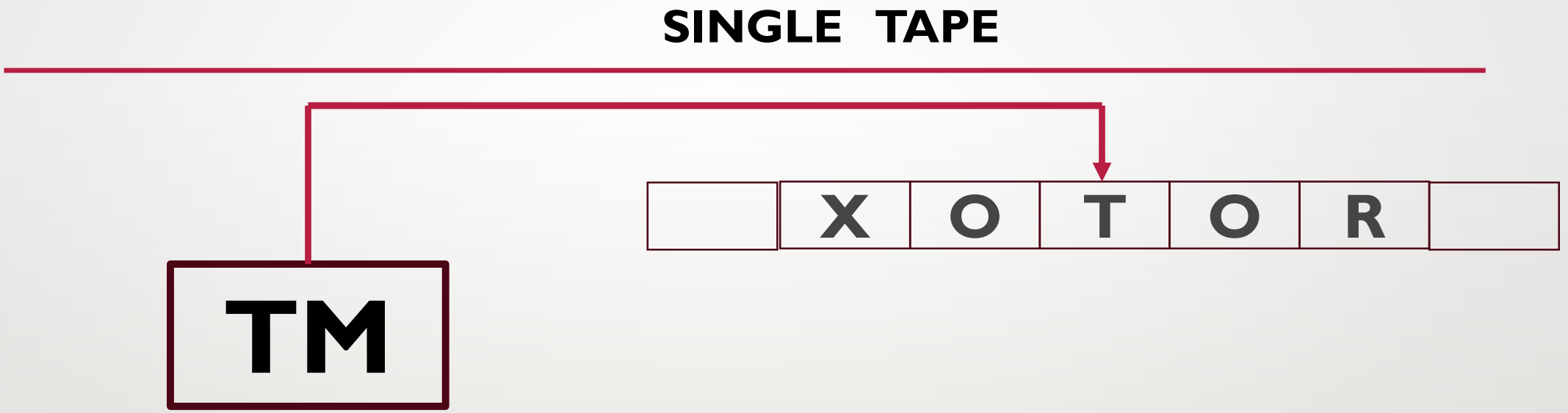
PALINDROME STRING VERIFICATION



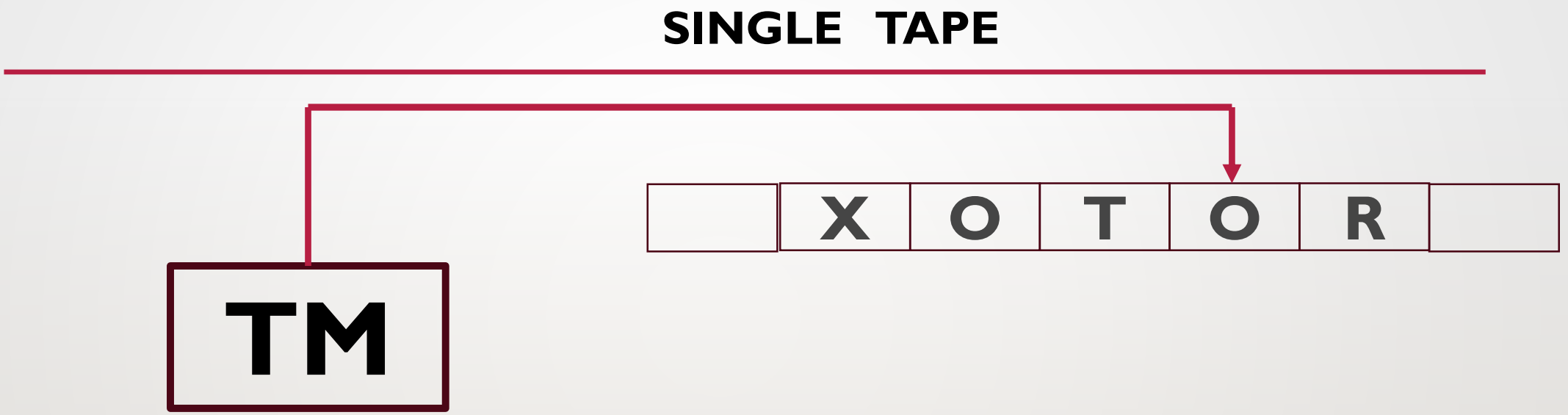
PALINDROME STRING VERIFICATION



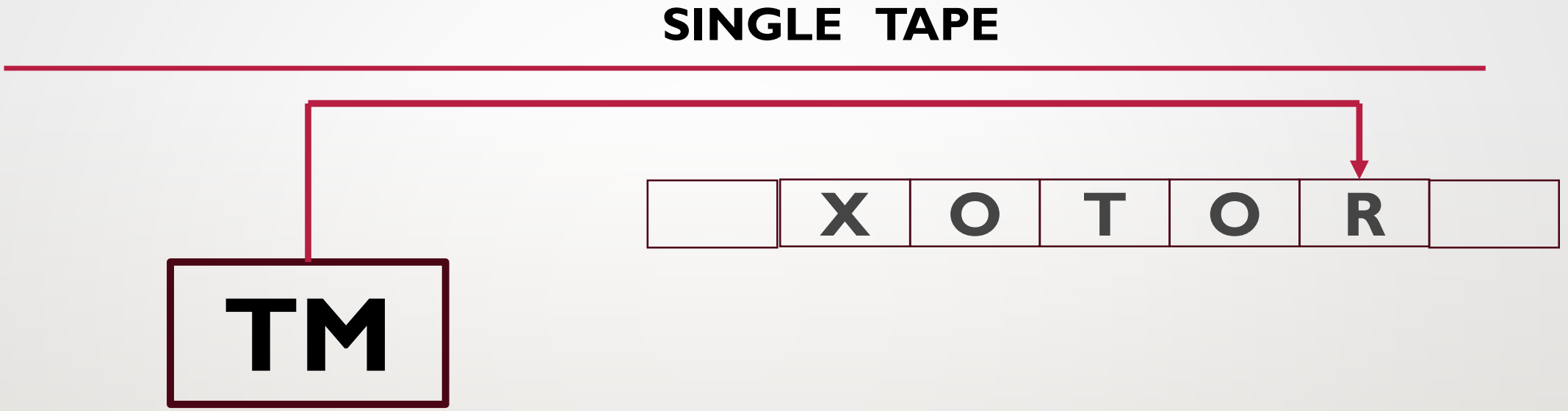
PALINDROME STRING VERIFICATION



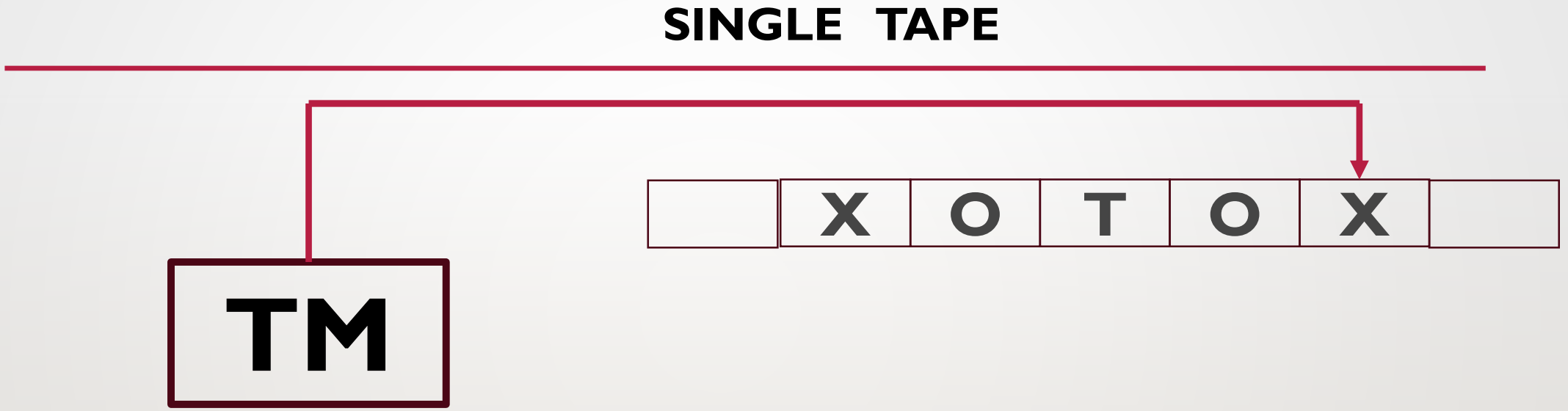
PALINDROME STRING VERIFICATION



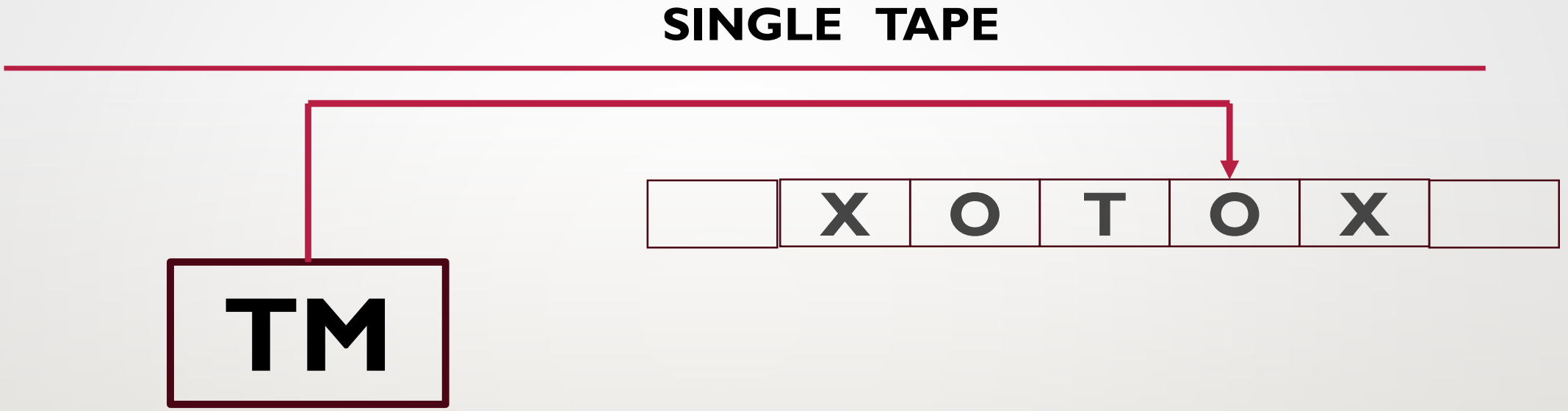
PALINDROME STRING VERIFICATION



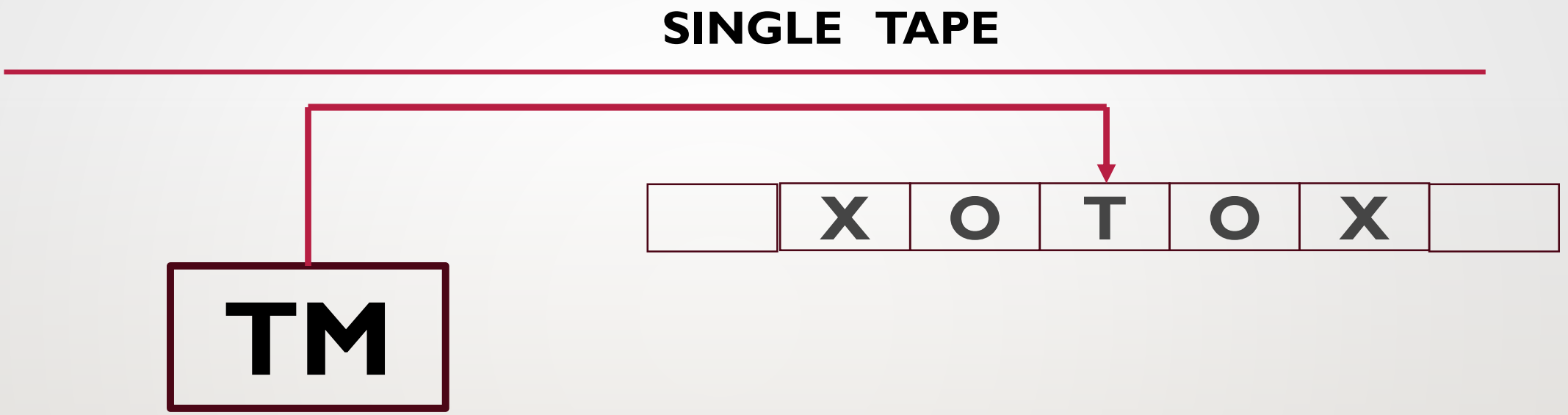
PALINDROME STRING VERIFICATION



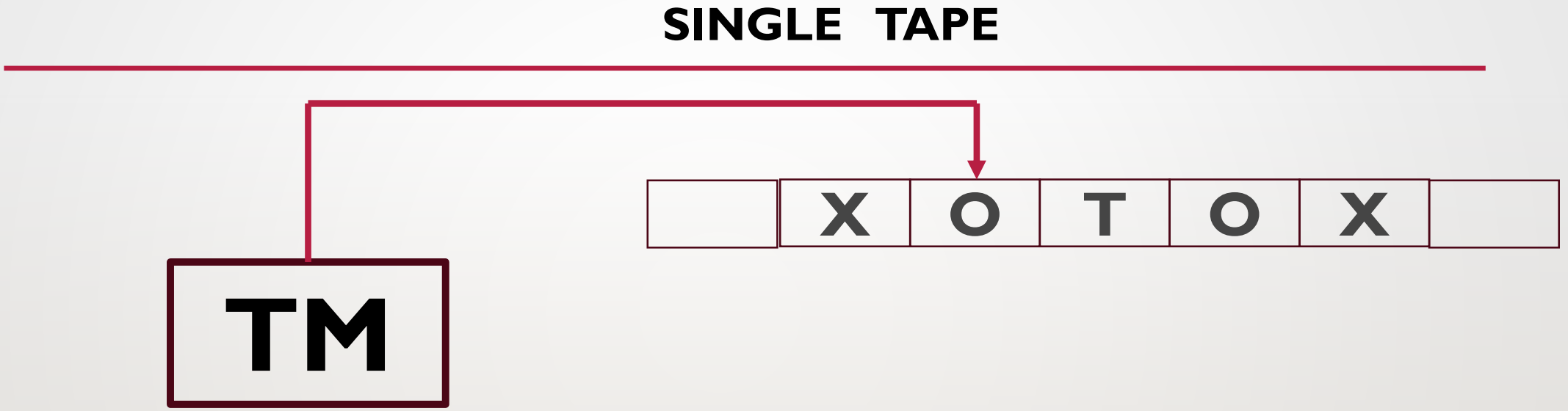
PALINDROME STRING VERIFICATION



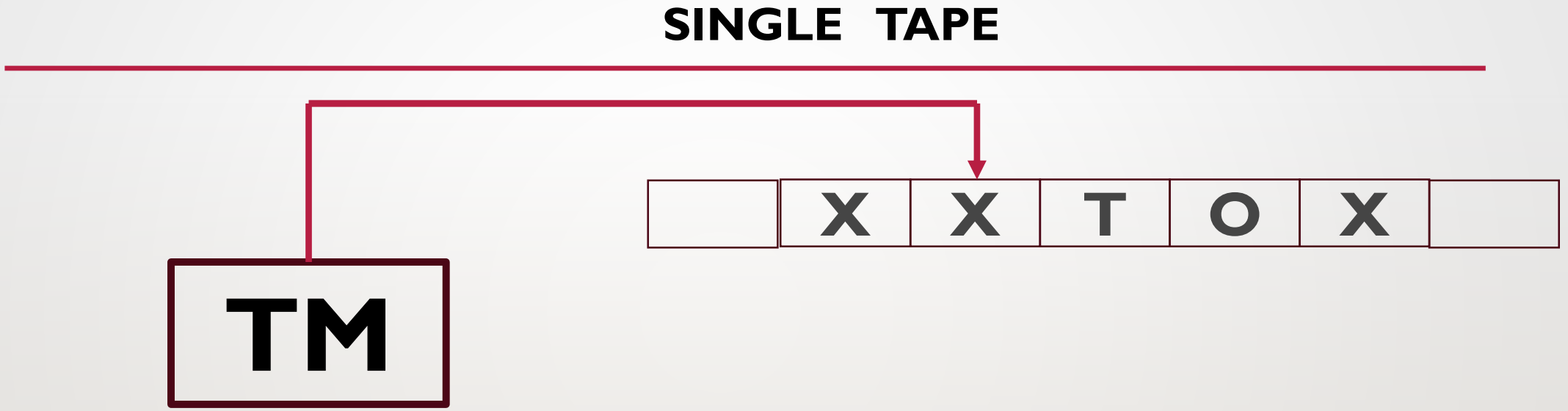
PALINDROME STRING VERIFICATION



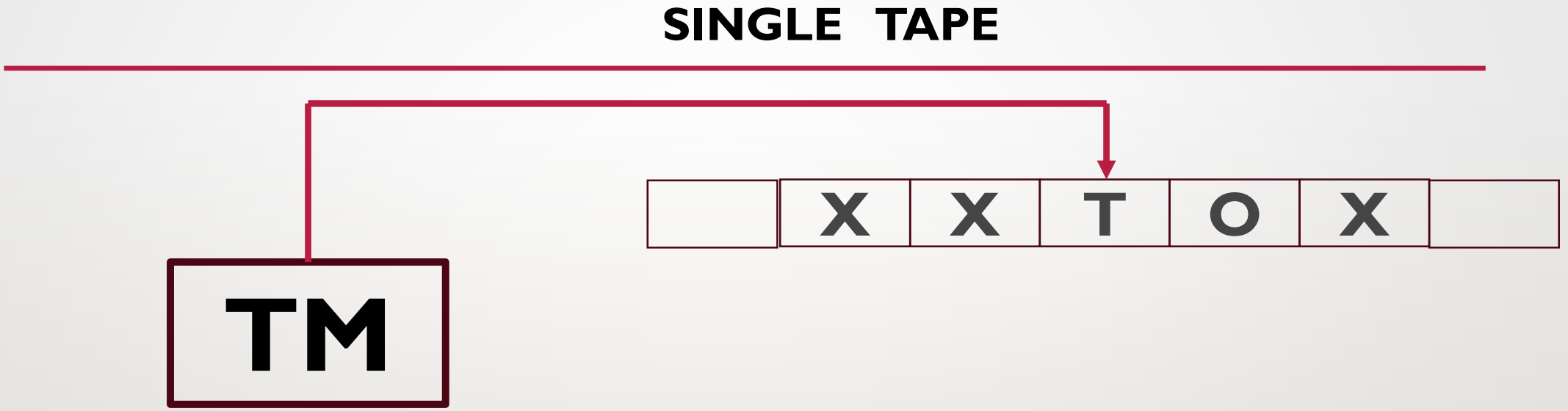
PALINDROME STRING VERIFICATION



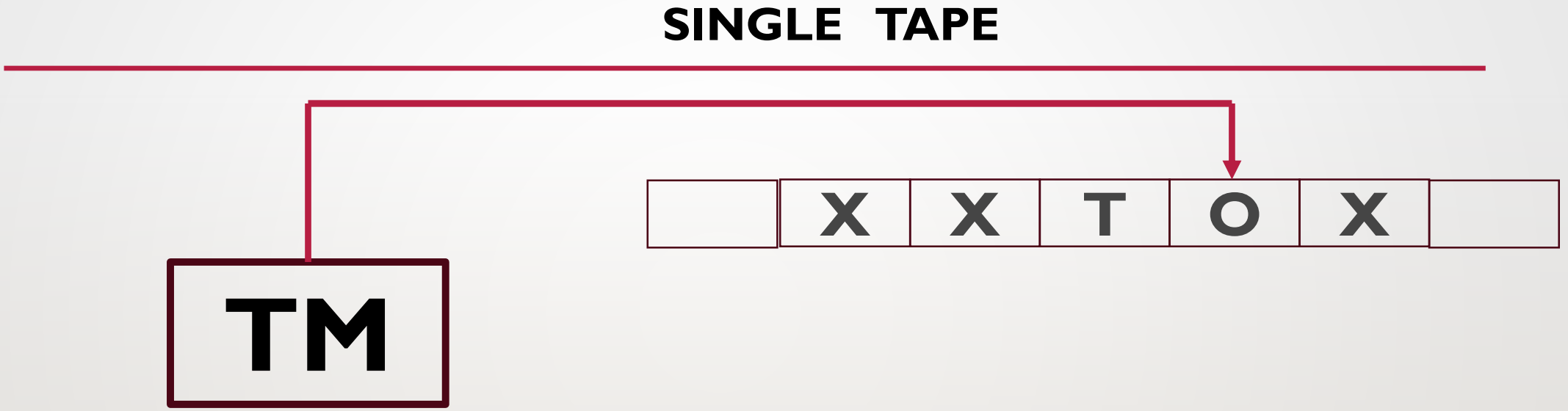
PALINDROME STRING VERIFICATION



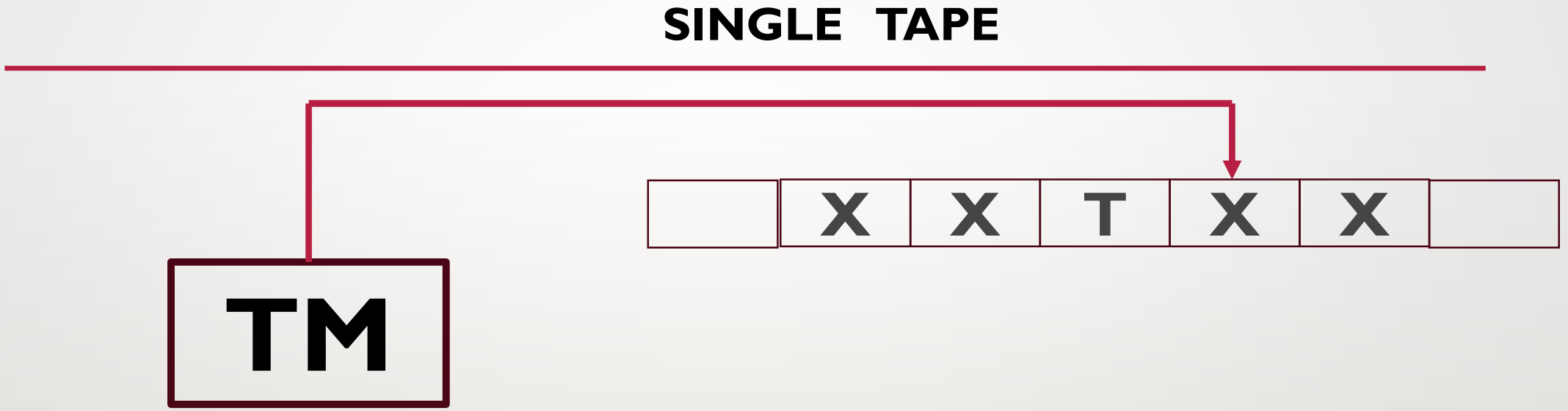
PALINDROME STRING VERIFICATION



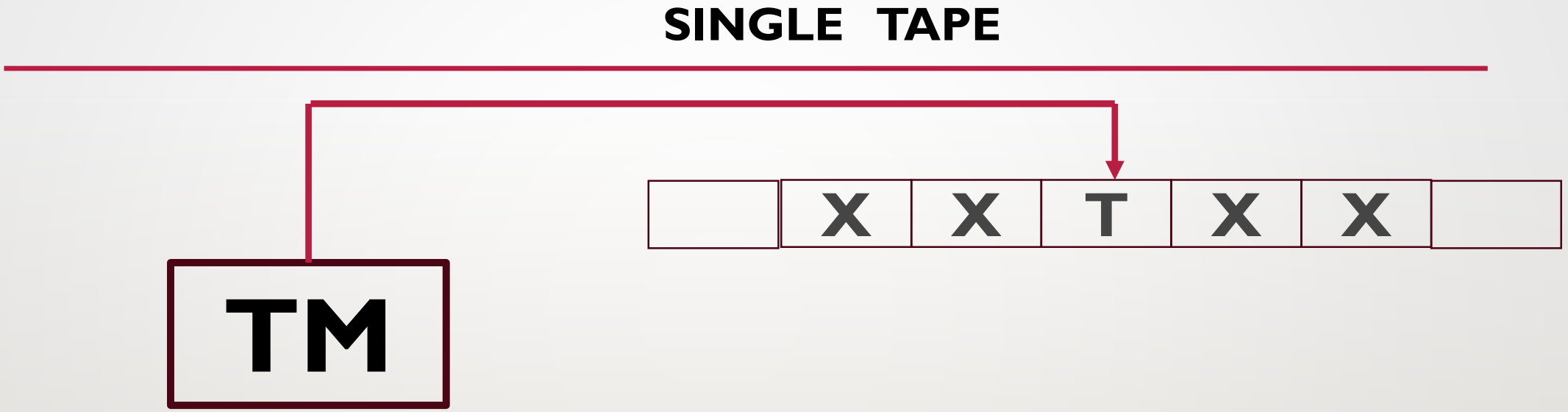
PALINDROME STRING VERIFICATION



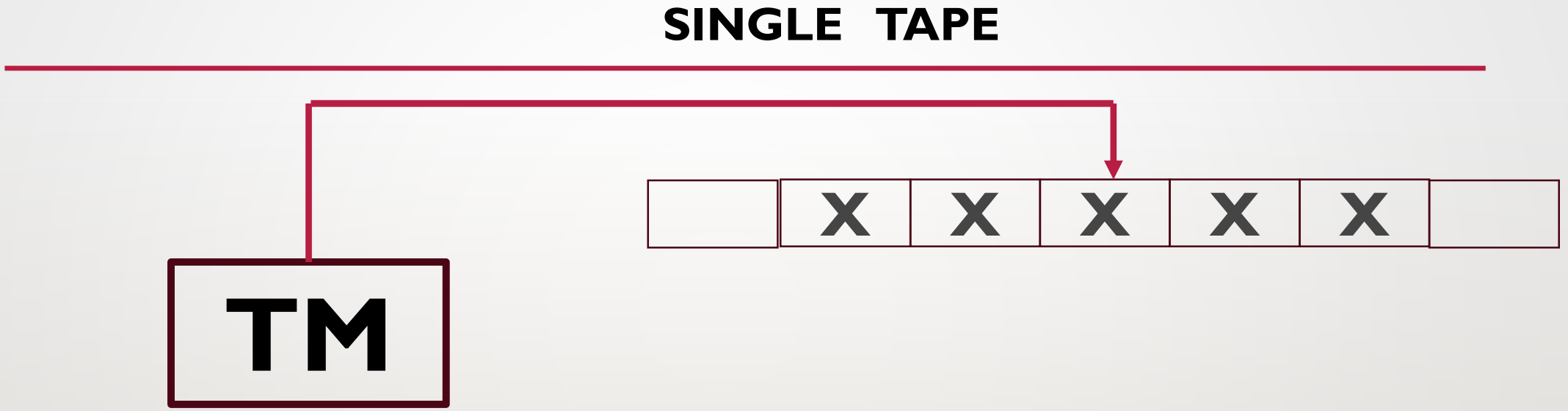
PALINDROME STRING VERIFICATION



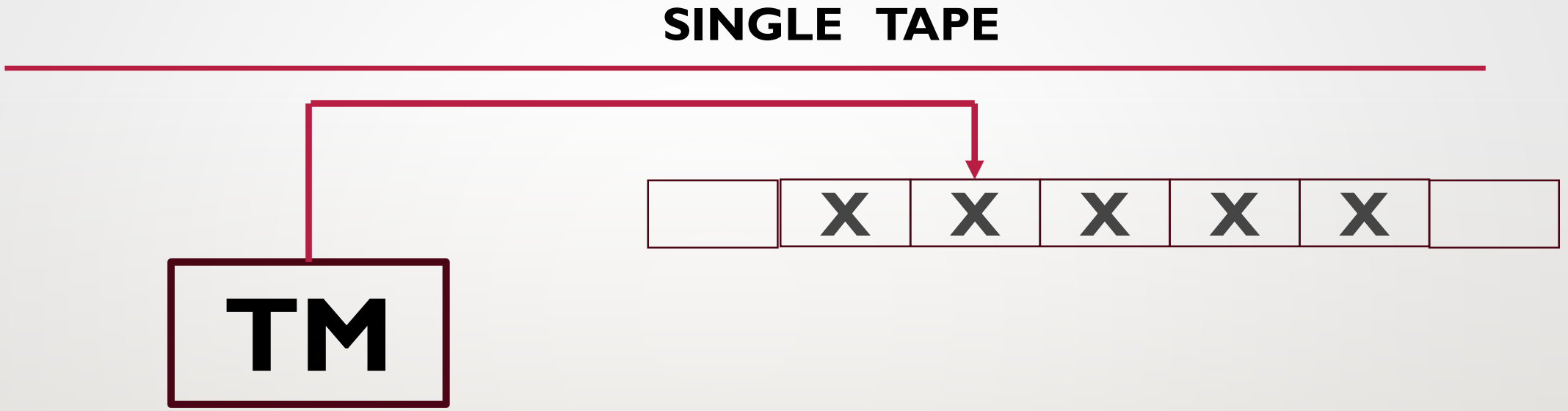
PALINDROME STRING VERIFICATION



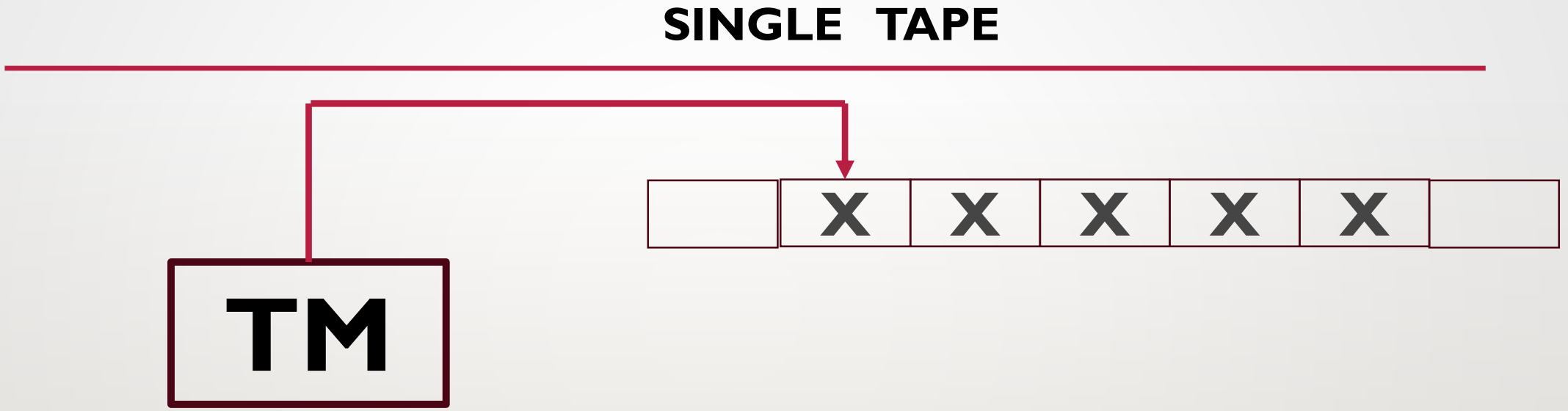
PALINDROME STRING VERIFICATION



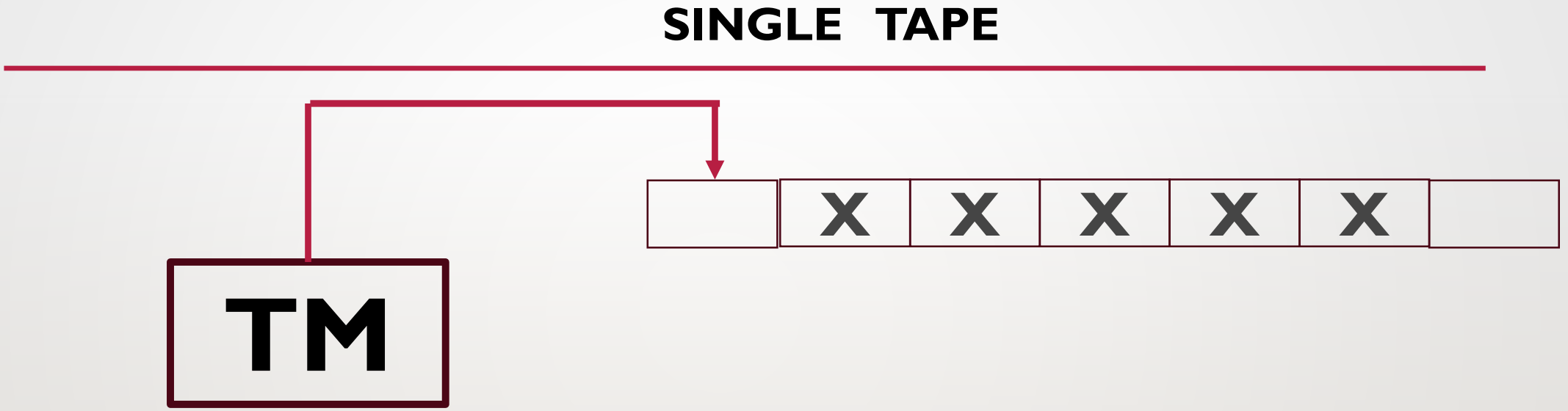
PALINDROME STRING VERIFICATION



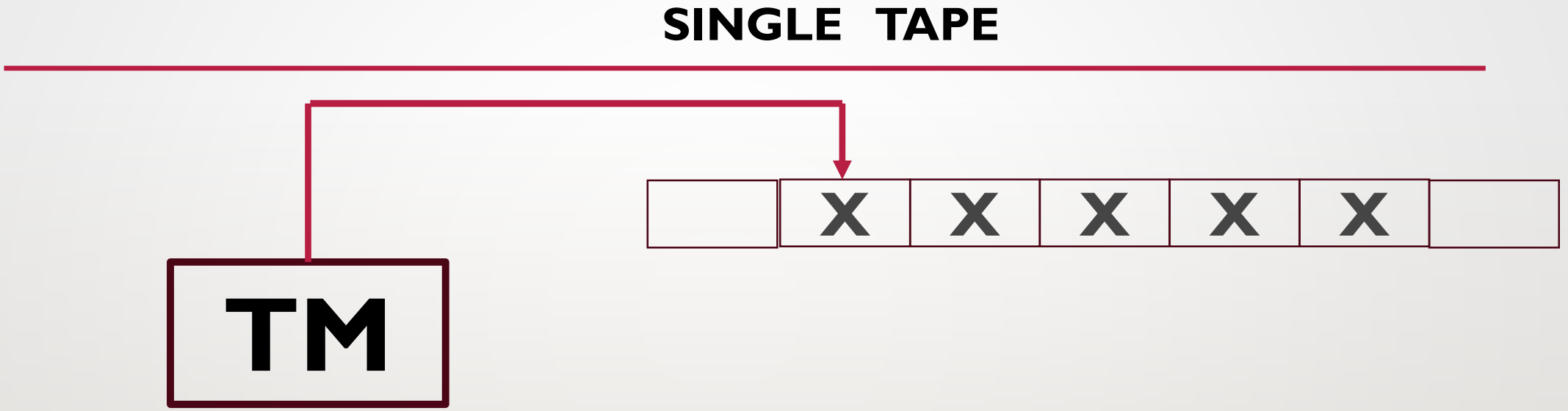
PALINDROME STRING VERIFICATION



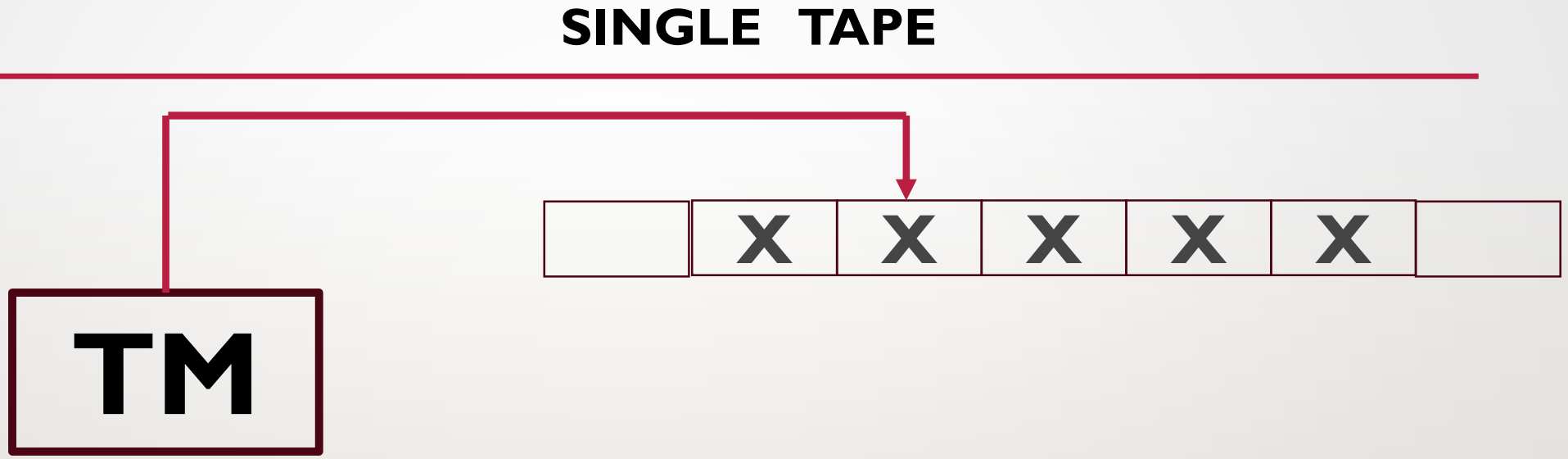
PALINDROME STRING VERIFICATION



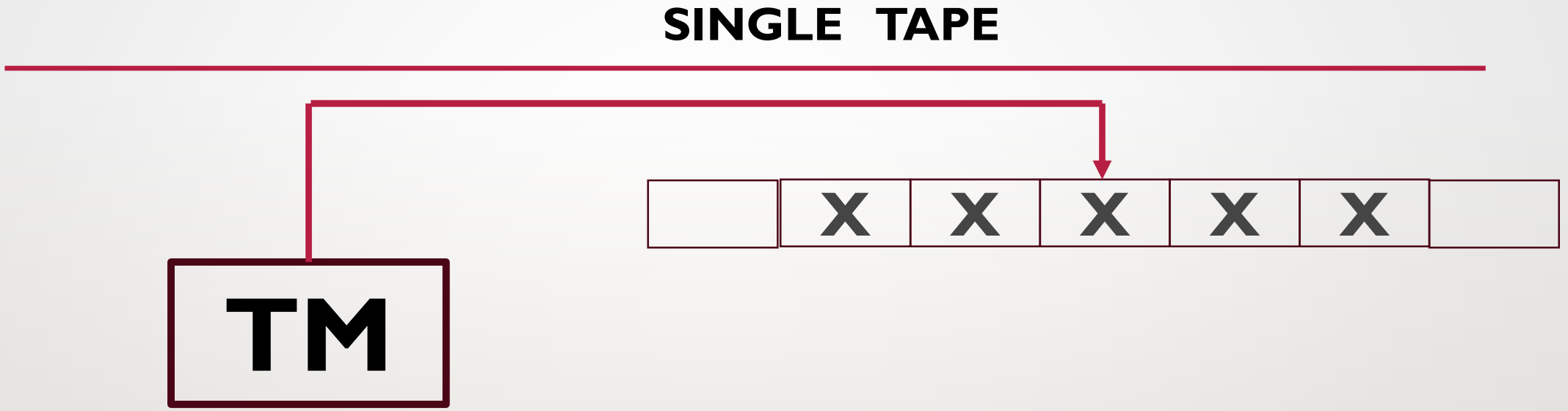
PALINDROME STRING VERIFICATION



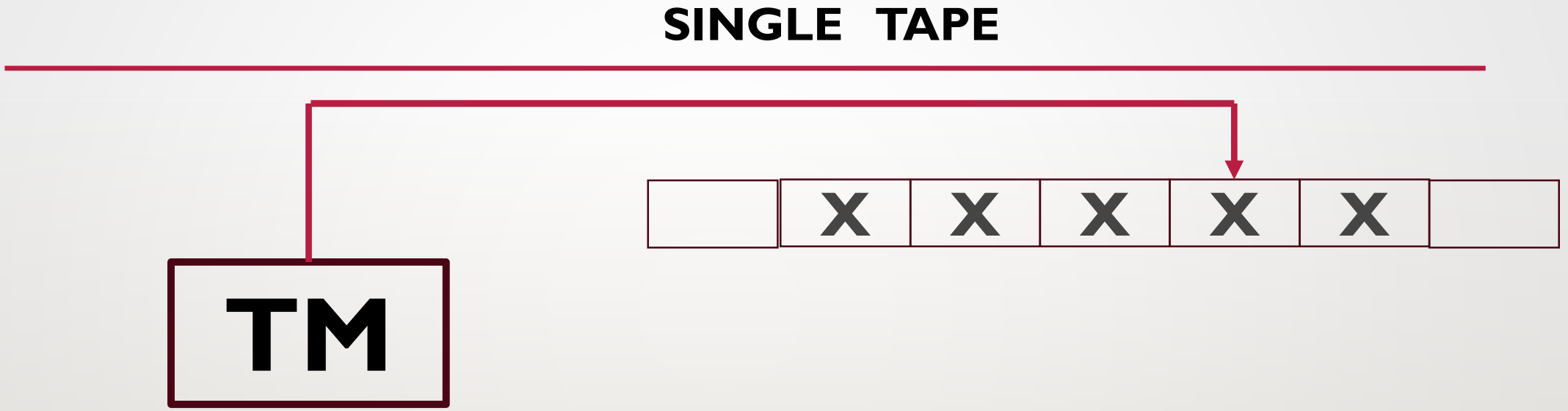
PALINDROME STRING VERIFICATION



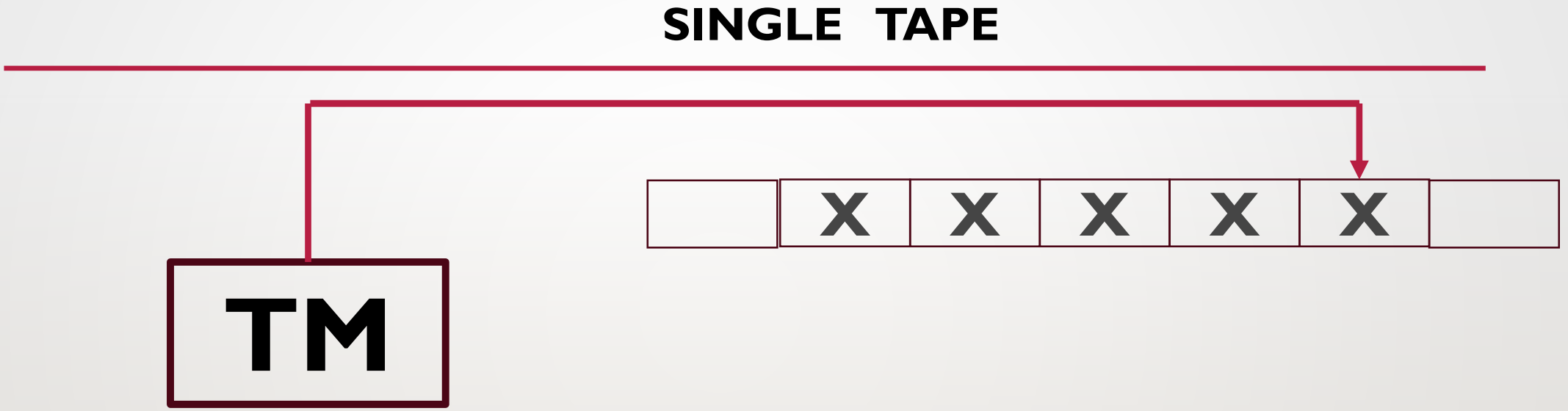
PALINDROME STRING VERIFICATION



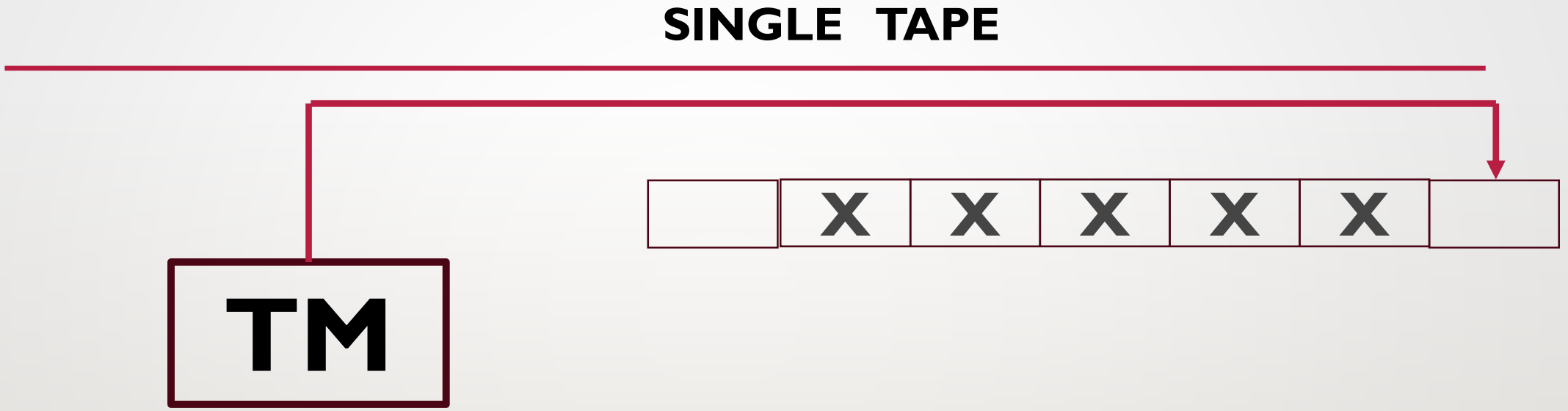
PALINDROME STRING VERIFICATION



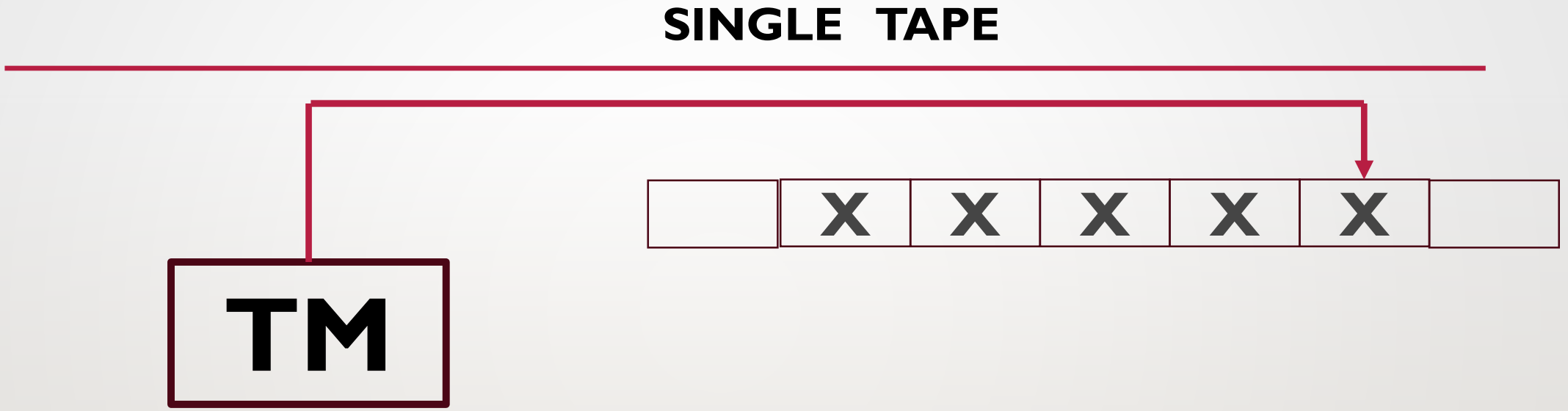
PALINDROME STRING VERIFICATION



PALINDROME STRING VERIFICATION



PALINDROME STRING VERIFICATION



$O(n^2)$

NON DETERMINASTIC TURING MACHINE

A non-deterministic Turing machine (NDTM) is a theoretical model of computation that extends the capabilities of a deterministic Turing machine (DTM) by allowing multiple possible transitions from a given state on a particular input symbol. **These are some key points of NDTM**

a) **Multiple Transitions**
b) **Branching**
c) **Simulation**
d) **Computational Power**
e) **Applications**

NON DETERMINASTIC TURING MACHINE CONTD....

Multiple Transitions - A DTM, which has a single transition function for each state and input symbol pair, an NDTM can have multiple possible transitions for the same state and input symbol.

Branching – Due to multiple transitions from a state gives rise to branching paths of computation. At each non-deterministic choice point, the machine can follow different paths, potentially exploring multiple possibilities simultaneously.

NON DETERMINASTIC TURING MACHINE CONTD....

Multiple Transitions - A DTM, which has a single transition function for each state and input symbol pair, an NDTM can have multiple possible transitions for the same state and input symbol.

Computational Power - Non-deterministic Turing machines are equivalent in computational power to deterministic Turing machines. This means that any problem that can be solved by an NDTM can also be solved by a DTM, and vice versa. If a single tape DTM to solve the problem N time then the NDTM solve it 2^n time due to tree structure.

Applications - Non-deterministic Turing machines are primarily used as a theoretical tool in the study of computability and complexity theory.

NDTM DESCRIPTION

An NDTM is a 7-tuple: $(Q, \Sigma, \Gamma, \delta, q0, q_{accept}, q_{reject})$

– Q is the set of states

– Σ is the input alphabet

– Γ is the tape alphabet

– δ: $Q \times \Gamma \rightarrow P(Q \times \Gamma \times \{L, R\})$ is the transition function

Here P is the power set.

– $q_0 \in Q$ is the start state

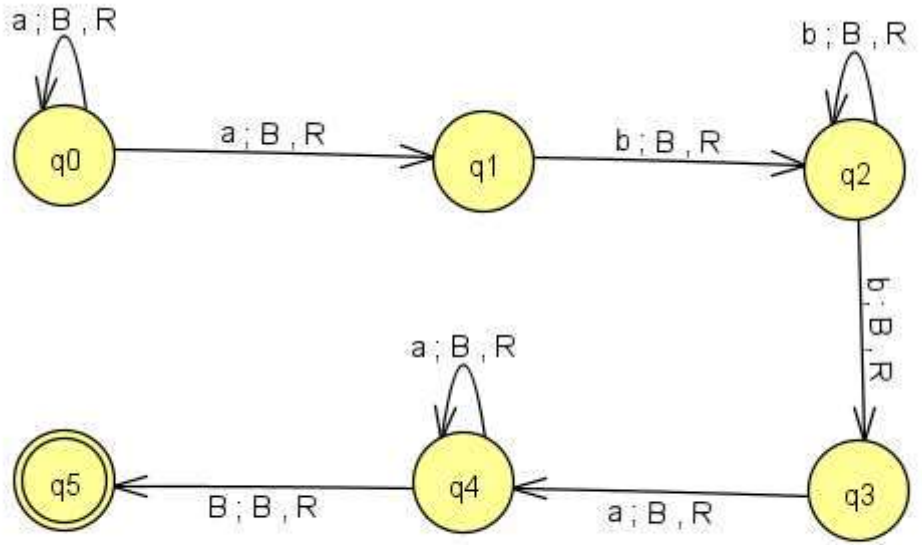
– $q_{accept} \in Q$ is the accept state

– $q_{reject} \in Q$ is the reject state

EXAMPLE OF NDTM

Example – check the string "aabbbaa" is accepted by the NDTM or not

δ(q0,a)={(q0,B,R),(q1,B,R)}
δ(q1,b)={q2,B,R)
δ(q2,b))={(q2,B,R),q3,B,R)}
δ(q3,a))={q4,B,R)
δ(q4,B) ={qf,B,H)



QUIZ TIME

In what wav is an NDTM different from a Deterministic Turing Machine (DTM)?

A. NDTM has more memory than DTM.

B. NDTM can make multiple transitions for a given state and symbol, whereas DTM has a single transition.

C. NDTM cannot solve the same problems as a DTM.

D. NDTM operates faster than a DTM.

Answer: B) NDTM can make multiple transitions for a given state and symbol, whereas DTM has a single transition.

QUIZ TIME

Which of the following is true about the languages recognized by NDTMs?

A. They are exactly the regular languages.

B. They are exactly the context-free languages.

C. They are exactly the context-sensitive languages.

D. They are exactly the recursively enumerable languages.

Answer: D) They are exactly the recursively enumerable languages.

QUIZ TIME

How does an NDTM accept an input string?

A. If at least one of the computation paths reaches an accepting state.

B. If all computation paths reach an accepting state.

C. If it reaches a rejecting state.

D. If it loops indefinitely on all paths.

Answer: A) If at least one of the computation paths reaches an accepting state.

QUIZ TIME

Can an NDTM be simulated by a DTM?

A. No, it is not possible.

B. Yes, with exponential time complexity in the worst case.

C. Yes, with the same time complexity.

D. Yes, but only for specific types of problems.

Answer: B) Yes, with exponential time complexity in the worst case.

QUIZ TIME

Which of the following is a key feature of Turing completeness?

A. The ability to compute any computable function.

B. The ability to compute any computable function within a finite amount of time.

C. The ability to compute any function.

D. The ability to compute any function within a finite amount of time.

Answer: A. The ability to compute any computable function.

QUESTIONS

Question 1: What is a Non-deterministic Turing Machine (NDTM)?

Answer: A Non-deterministic Turing Machine is a theoretical model of computation where multiple transitions are possible from a single state and input symbol, allowing it to explore many computational paths simultaneously.

Question 2: How does an NDTM differ from a Deterministic Turing Machine (DTM)?

Answer: Unlike a DTM, which has a single next state for each state and input symbol, an NDTM can have multiple possible next states, effectively simulating parallel computation paths.

Question 3: Can an NDTM solve problems faster than a DTM?

Answer: In theory, an NDTM can solve some problems faster than a DTM by exploring many computational paths at once, but this advantage is theoretical since real-world computers are deterministic.

Question 4: Are there any problems that an NDTM can solve but a DTM cannot?

Answer: No, any problem that can be solved by an NDTM can also be solved by a DTM; however, the DTM may require more time to explore all possible paths sequentially.

Question 5: What is the significance of NDTMs in computational theory?
Answer: NDTMs are significant in computational theory because they help define the class NP (nondeterministic polynomial time) and are used to explore the limits of efficient computation, particularly in the context of the P vs NP problem.

THANK YOU

Team – ATFL