Welcome!

Fine Grained Algorithms & Complexity

CS6100

Topics in Design & Analysis of Algorithms

Akanksha Agrawal

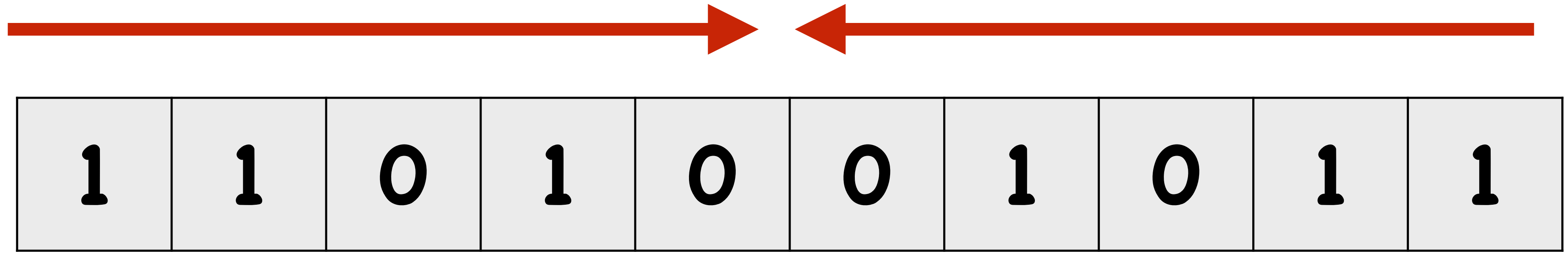
Different computers, different running times?

Computation Model

All the animated images used are the freely available ones from the results of Google's search engine.

Computation Model

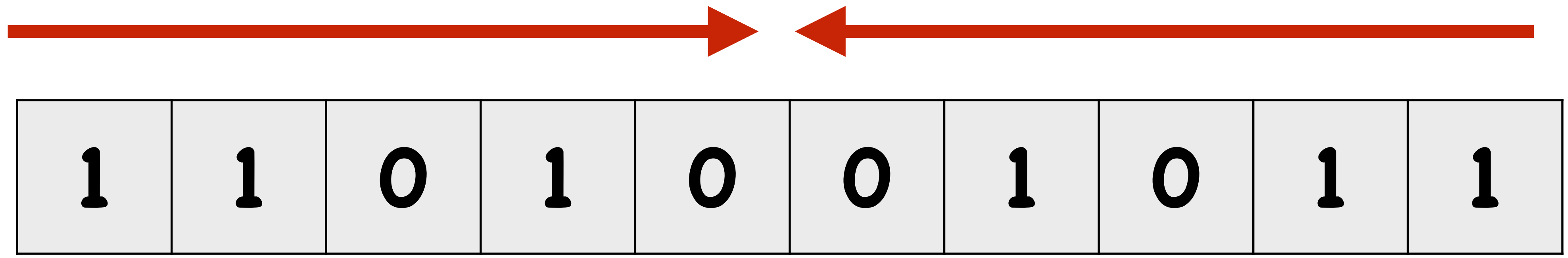
Palindrome Problem



1	1	0	1	0	0	1	0	1	1

Computation Model

Palindrome Problem



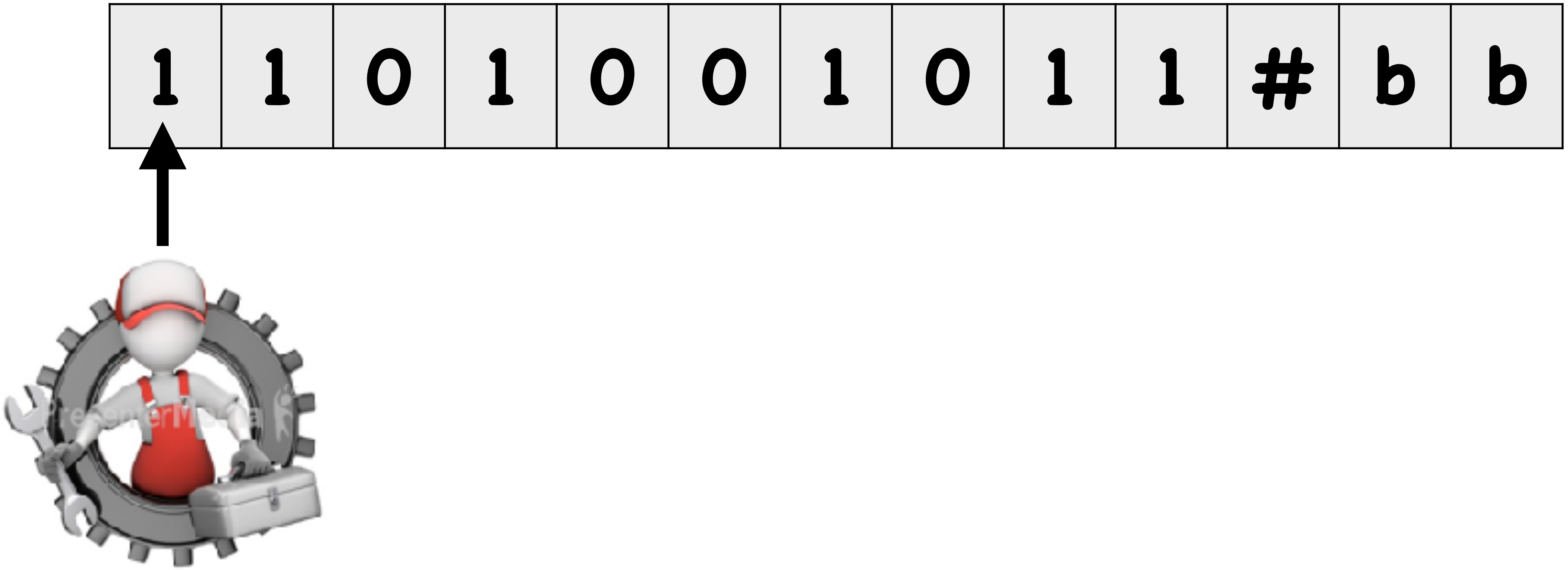
1	1	0	1	0	0	1	0	1	1

How do we test if the given string is a palindrome?

Computation Model

Model 1: Single Tape Turing Machine

Palindrome Problem



1	1	0	1	0	0	1	0	1	1	#	b	b

Computation Model

Model 1: Single Tape Turing Machine

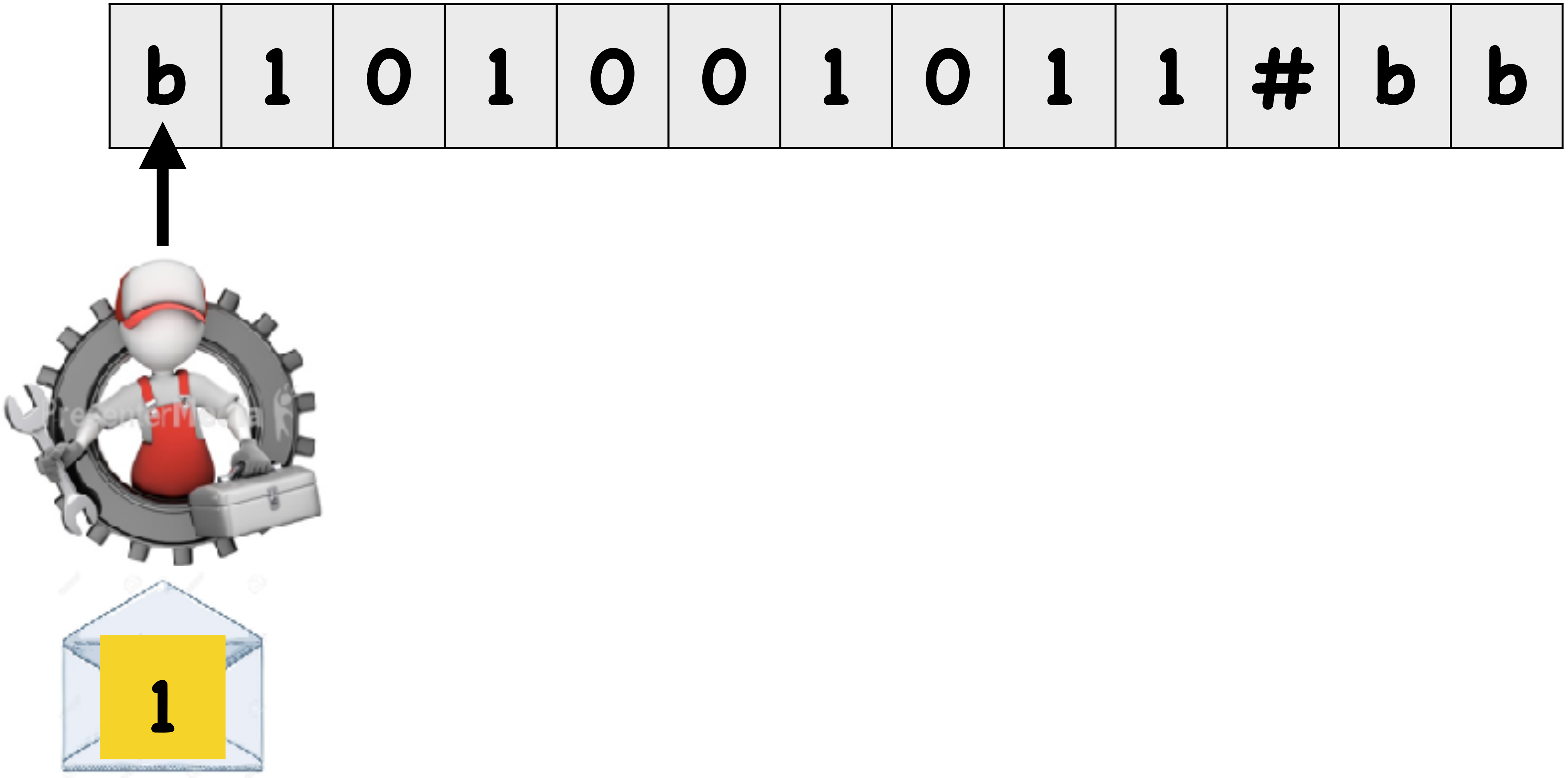
Palindrome Problem

1	1	0	1	0	0	1	0	1	1	#	b	b

Computation Model

Model 1: Single Tape Turing Machine

Palindrome Problem



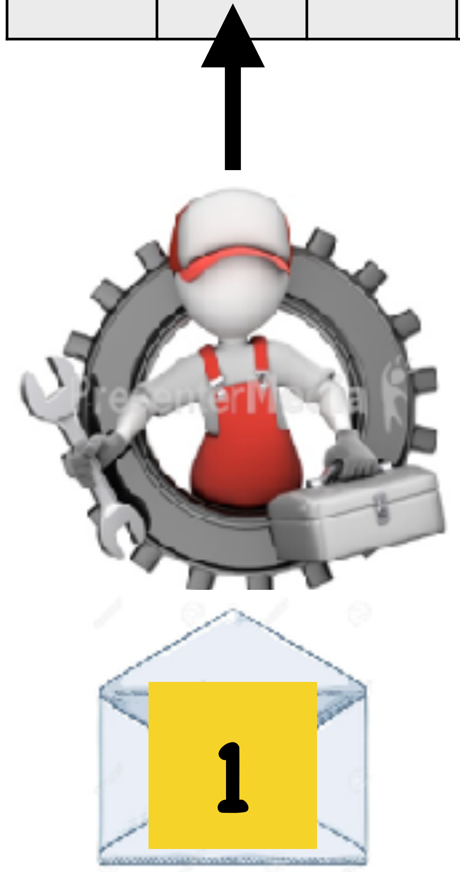
b	1	0	1	0	0	1	0	1	1	#	b	b

Computation Model

Model 1: Single Tape Turing Machine

Palindrome Problem

b	1	0	1	0	0	1	0	1	1	#	b	b



Computation Model

Model 1: Single Tape Turing Machine

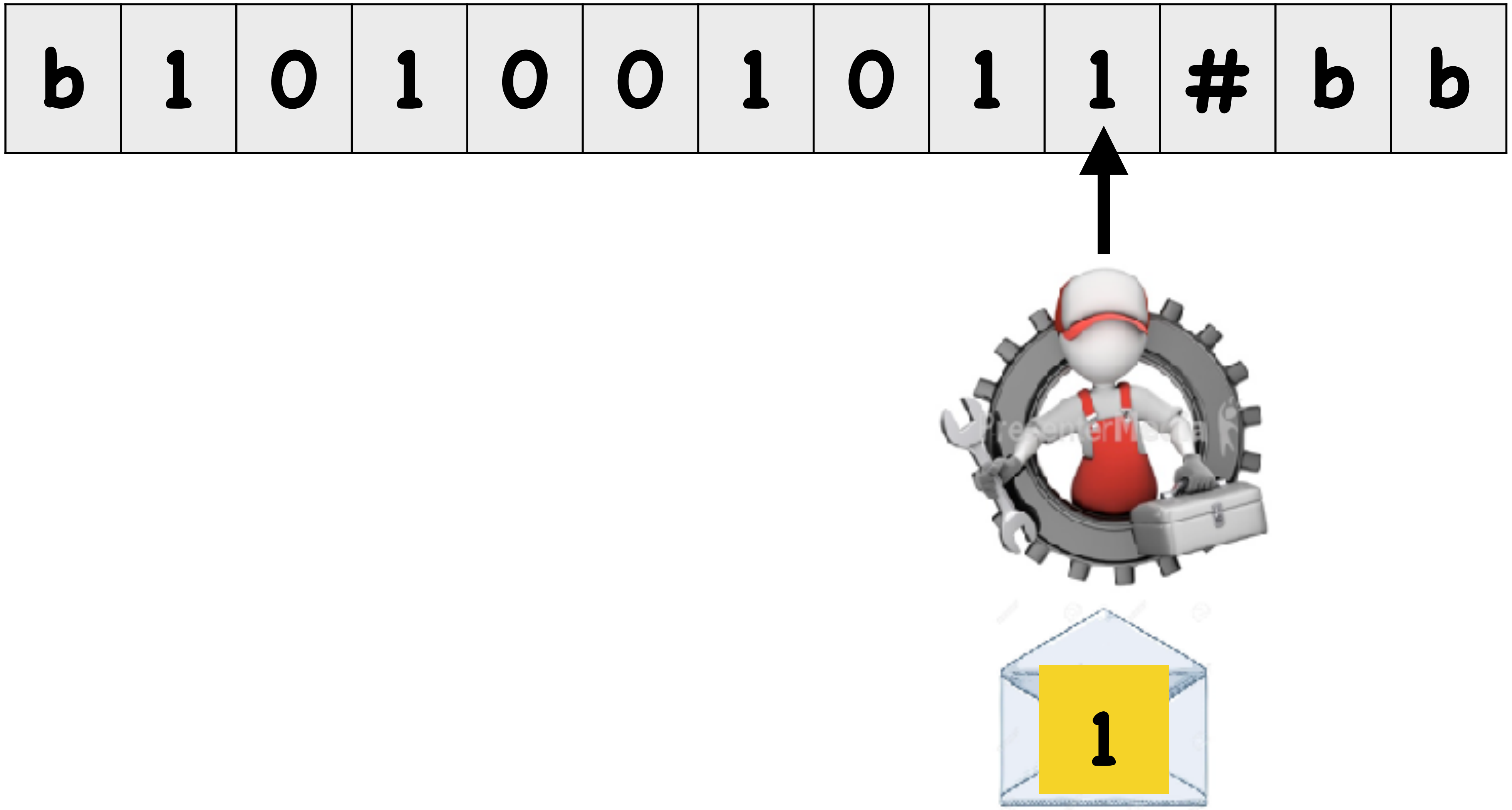
Palindrome Problem

b	1	0	1	0	0	1	0	1	1	#	b	b

Computation Model

Model 1: Single Tape Turing Machine

Palindrome Problem



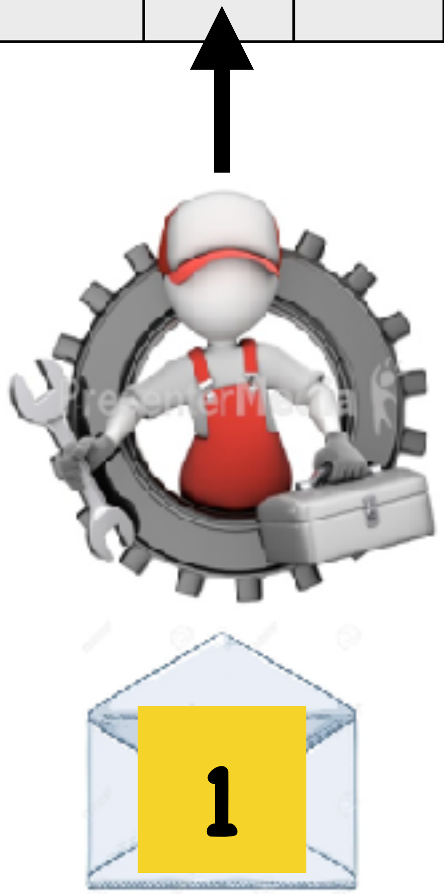
b	1	0	1	0	0	1	0	1	1	#	b	b

Computation Model

Model 1: Single Tape Turing Machine

Palindrome Problem

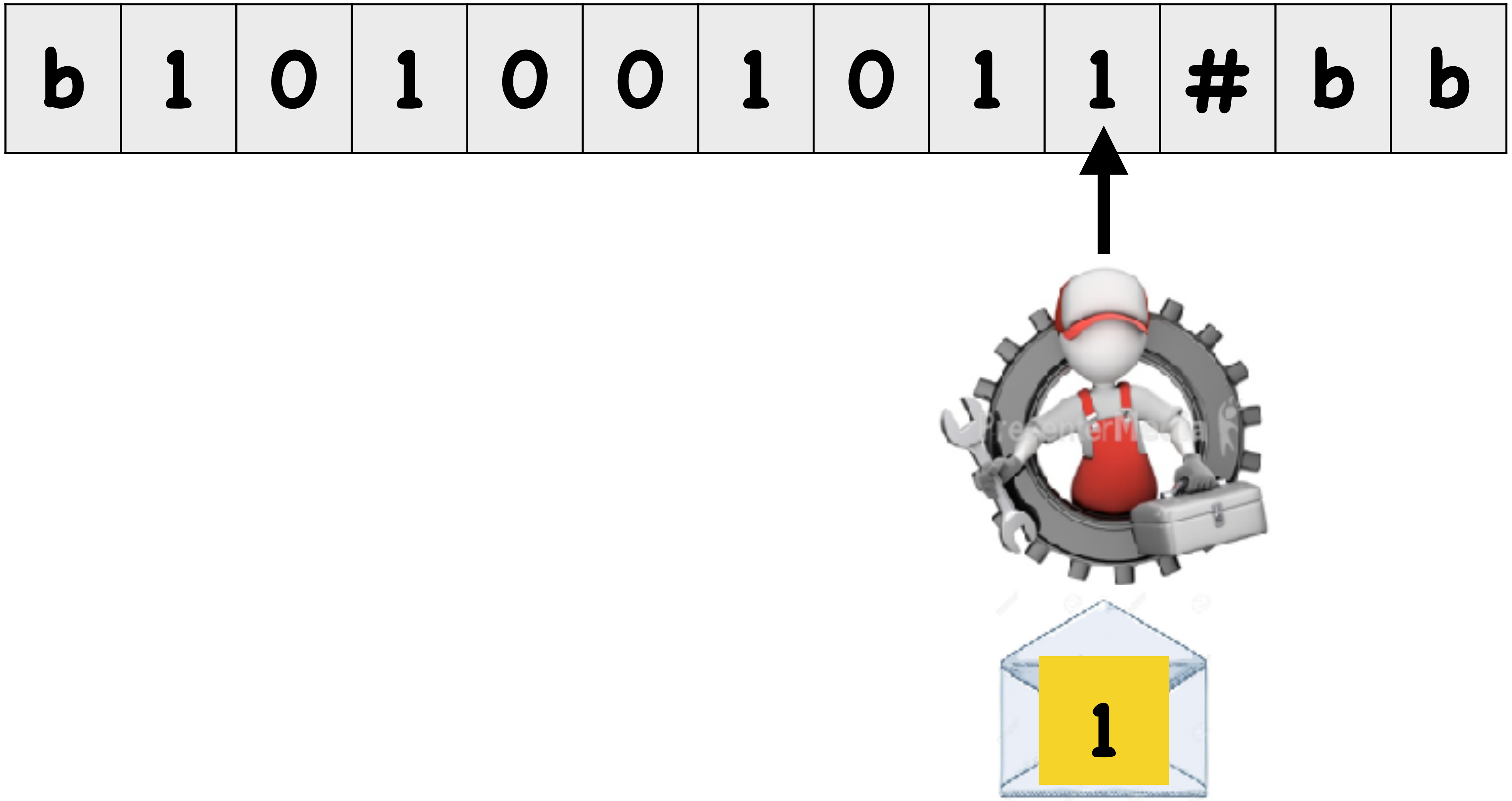
b	1	0	1	0	0	1	0	1	1	#	b	b



Computation Model

Model 1: Single Tape Turing Machine

Palindrome Problem

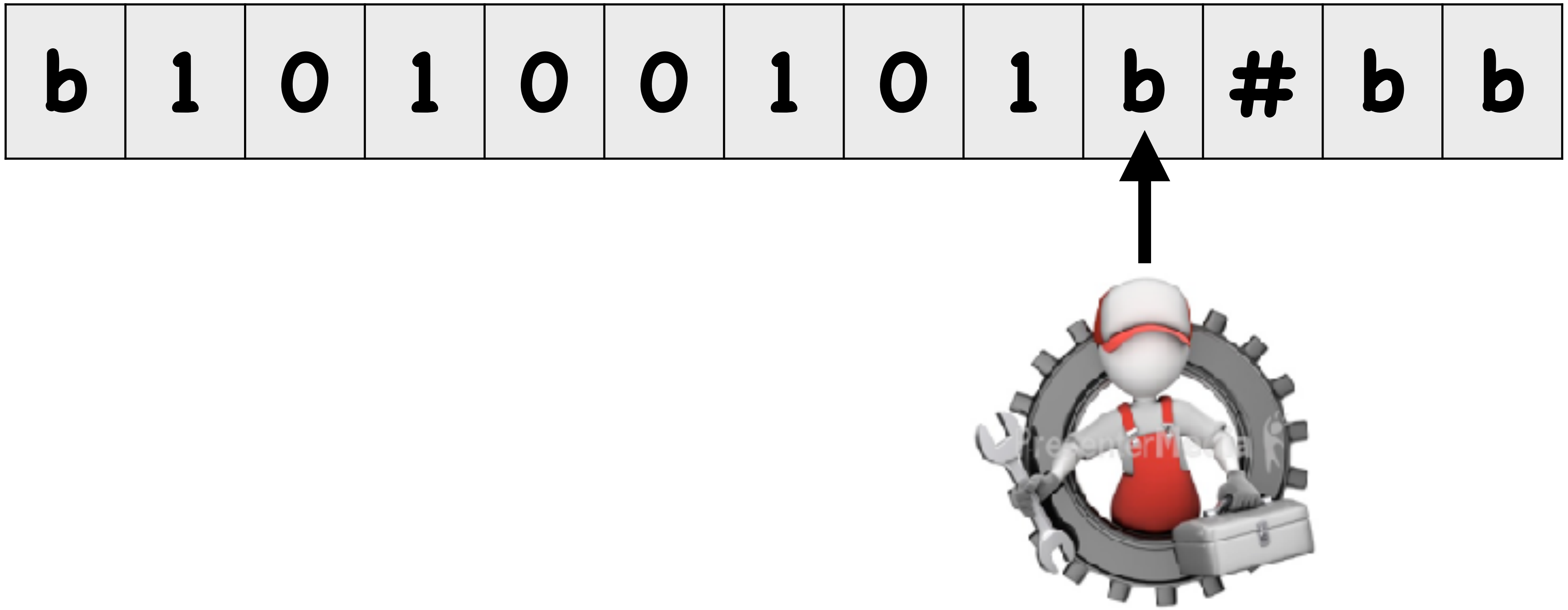


b	1	0	1	0	0	1	0	1	1	#	b	b

Computation Model

Model 1: Single Tape Turing Machine

Palindrome Problem

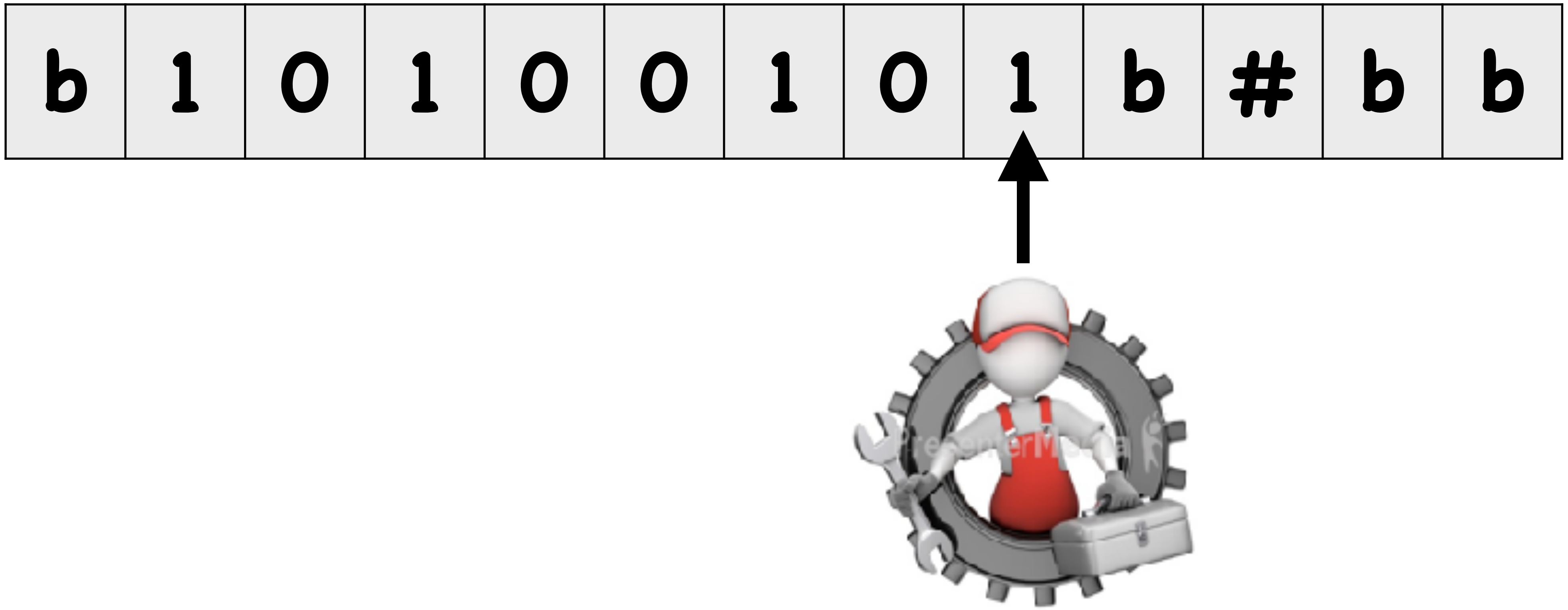


b	1	0	1	0	0	1	0	1	b	#	b	b

Computation Model

Model 1: Single Tape Turing Machine

Palindrome Problem

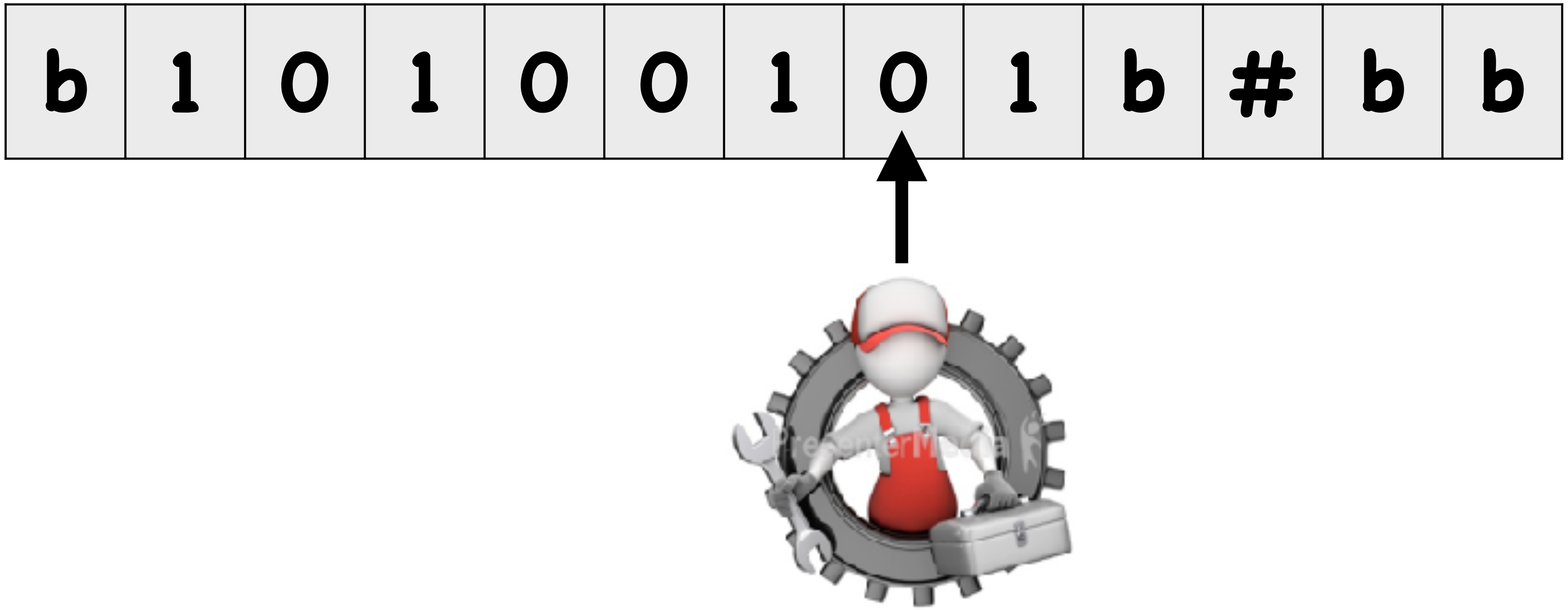


b	1	0	1	0	0	1	0	1	b	#	b	b

Computation Model

Model 1: Single Tape Turing Machine

Palindrome Problem

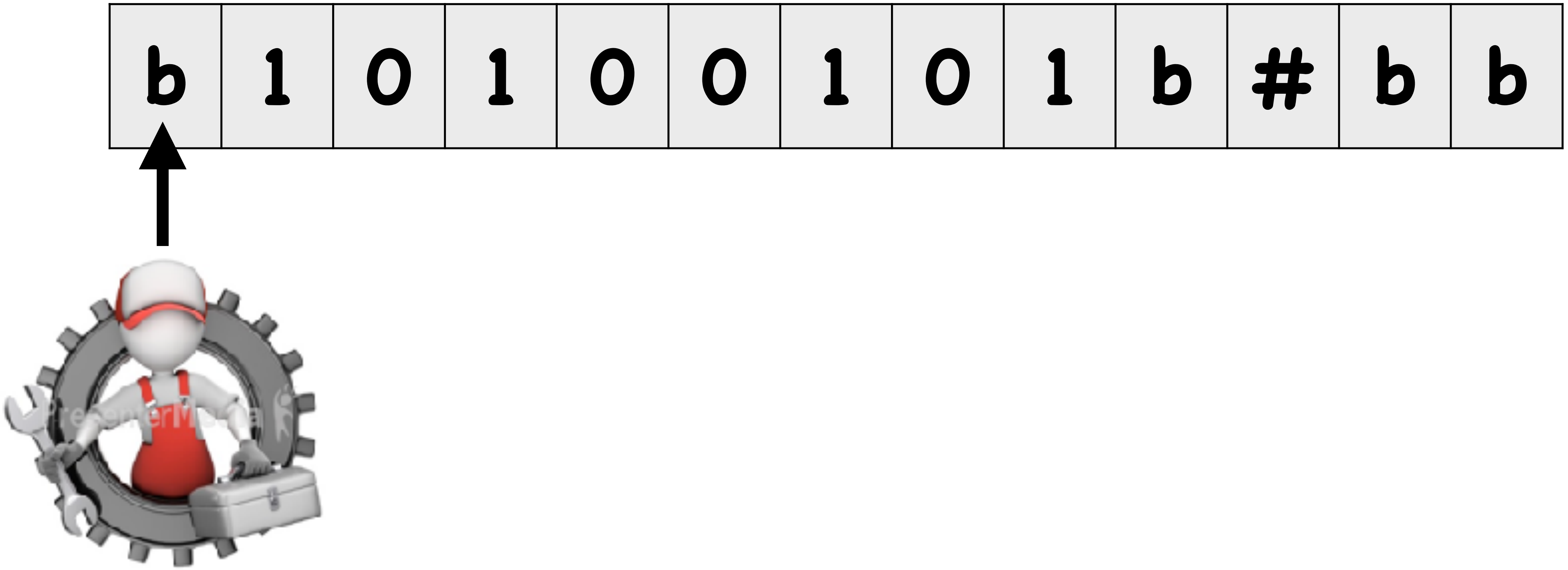


b	1	0	1	0	0	1	0	1	b	#	b	b

Computation Model

Model 1: Single Tape Turing Machine

Palindrome Problem

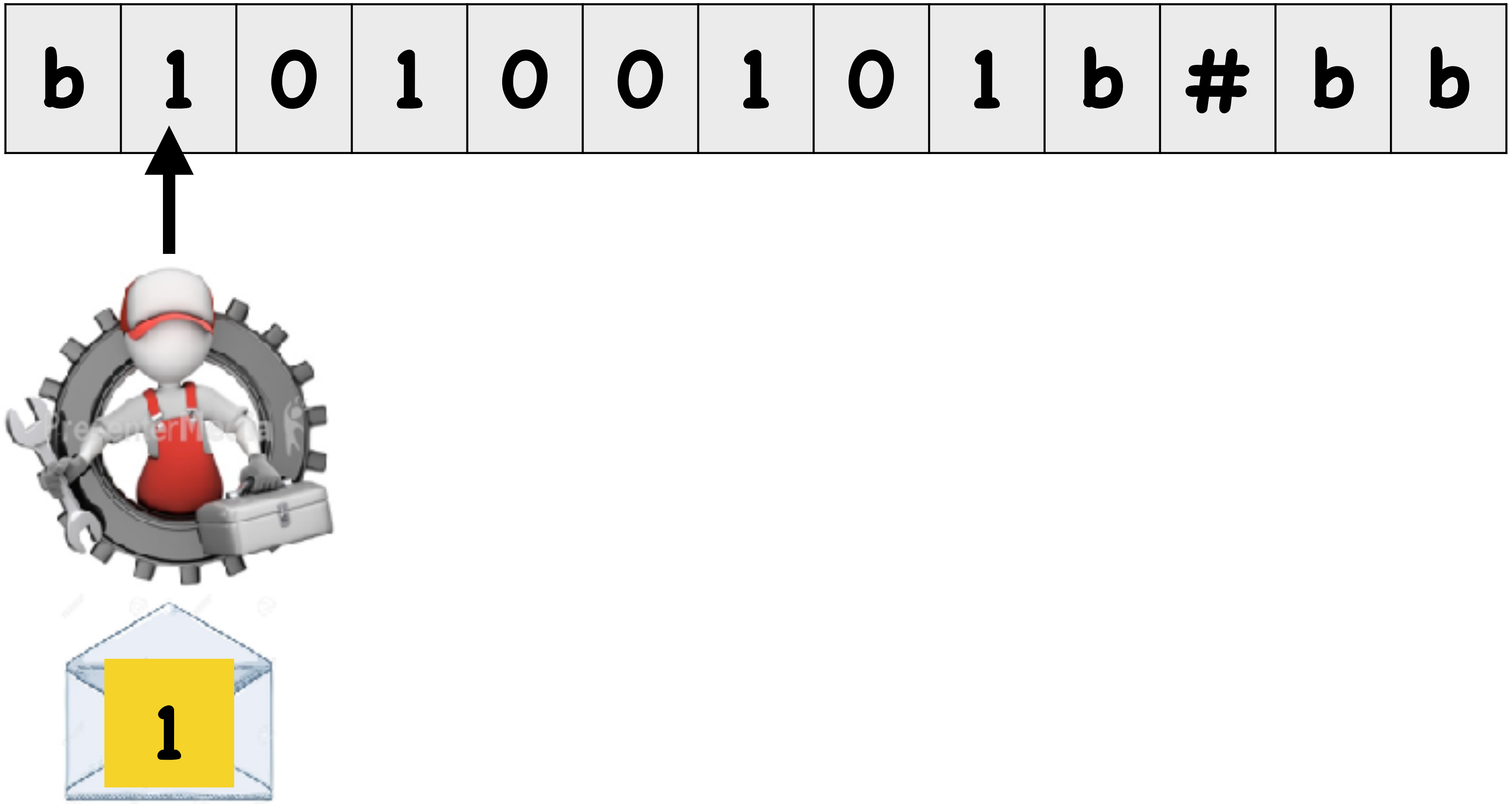


b	1	0	1	0	0	1	0	1	b	#	b	b

Computation Model

Model 1: Single Tape Turing Machine

Palindrome Problem



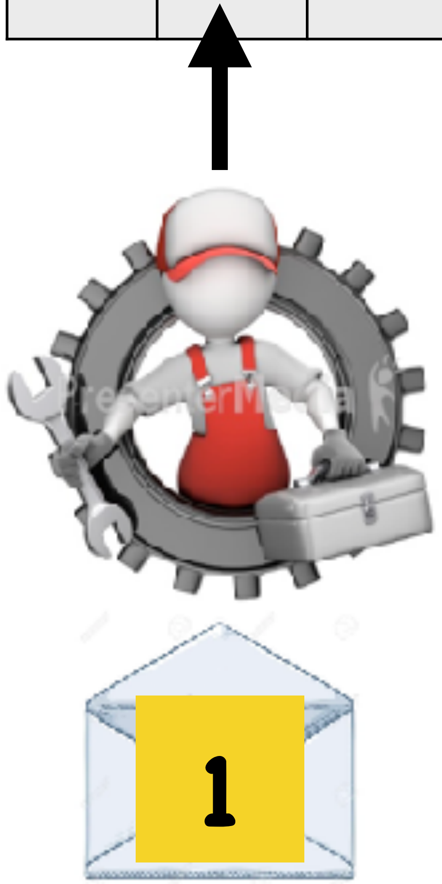
b	1	0	1	0	0	1	0	1	b	#	b	b

Computation Model

Model 1: Single Tape Turing Machine

Palindrome Problem

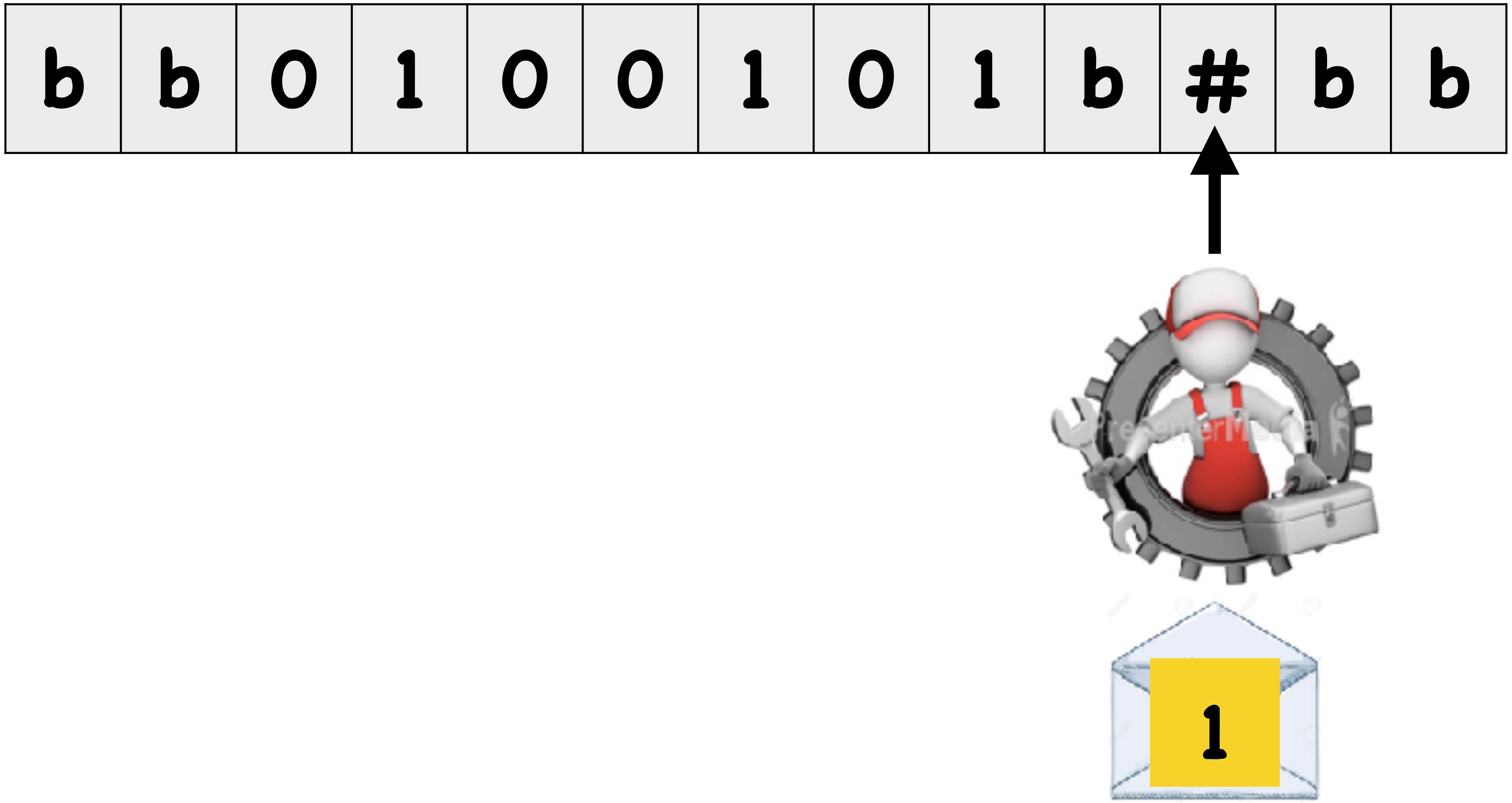
b	b	0	1	0	0	1	0	1	b	#	b	b



Computation Model

Model 1: Single Tape Turing Machine

Palindrome Problem

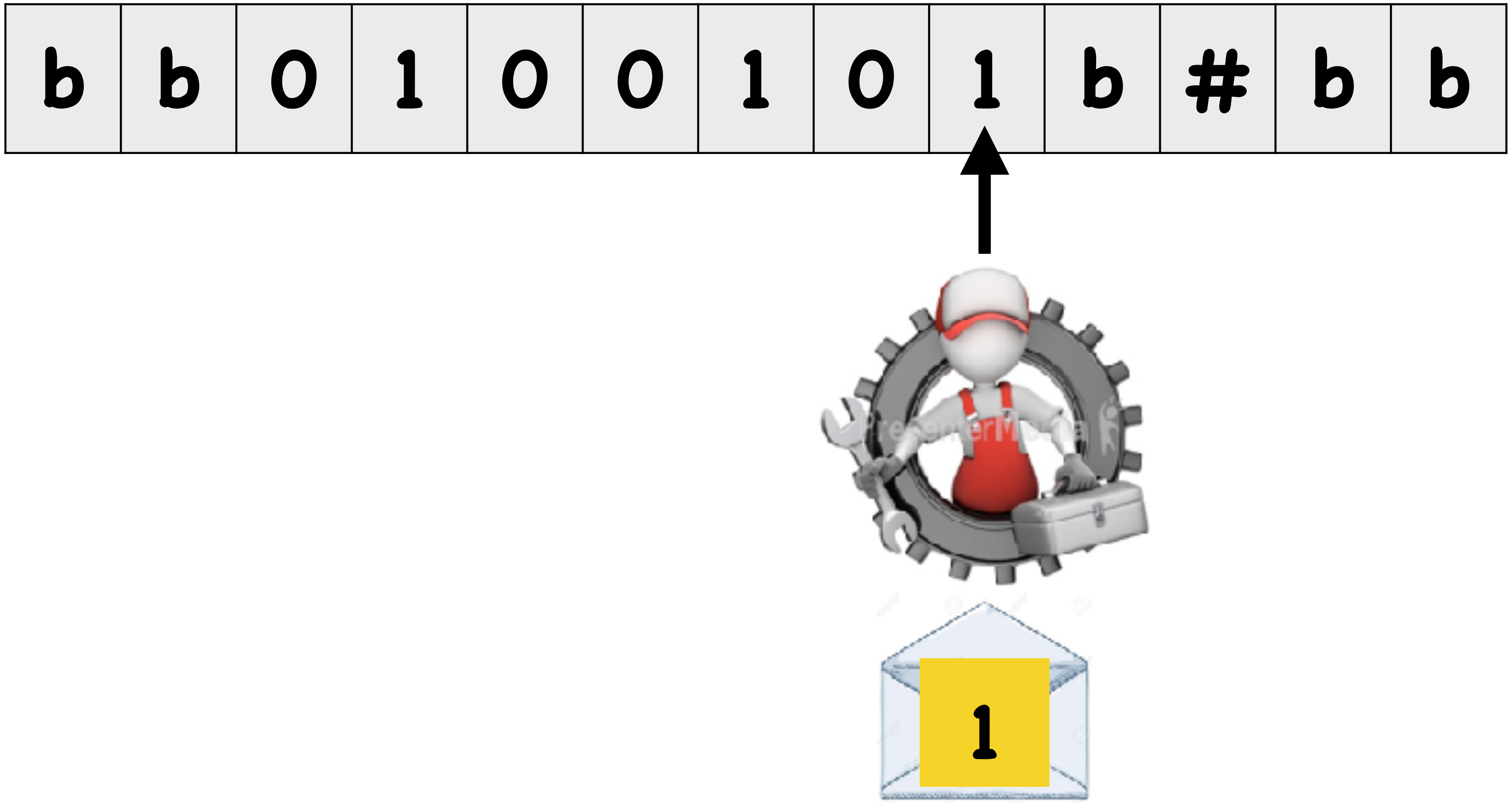


b	b	0	1	0	0	1	0	1	b	#	b	b

Computation Model

Model 1: Single Tape Turing Machine

Palindrome Problem

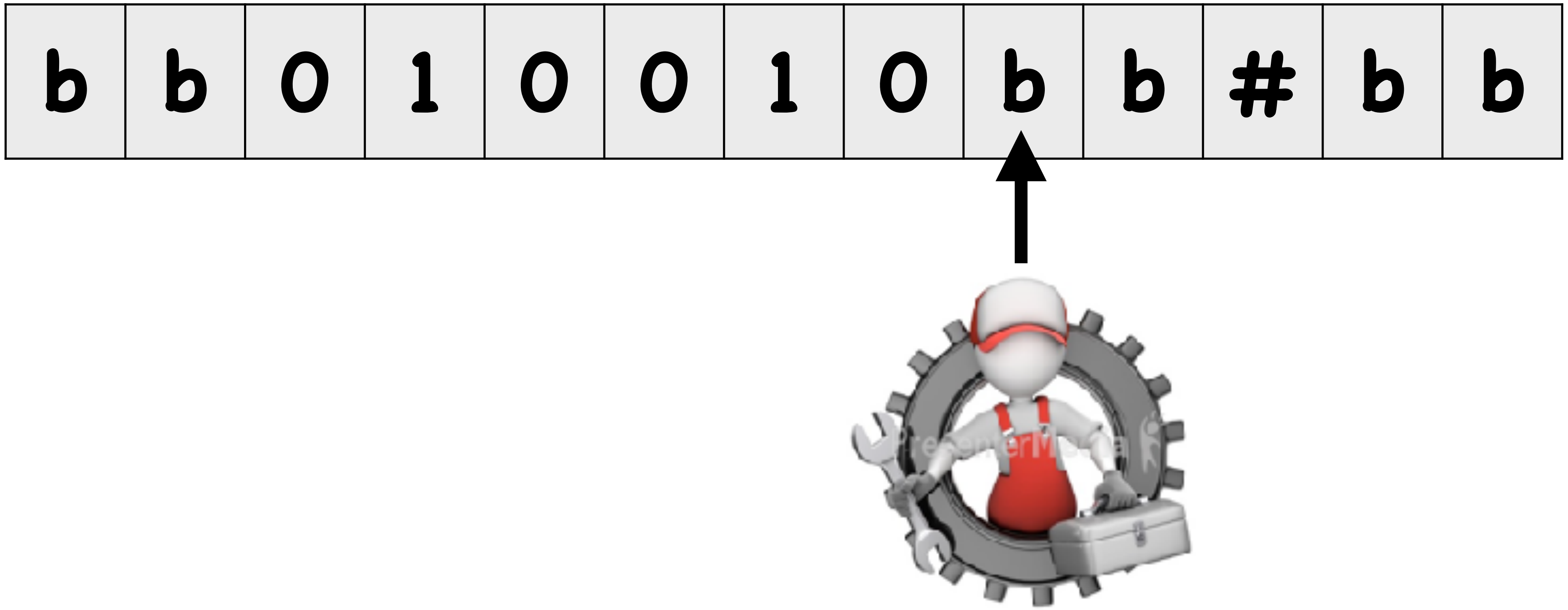


b	b	0	1	0	0	1	0	1	b	#	b	b

Computation Model

Model 1: Single Tape Turing Machine

Palindrome Problem

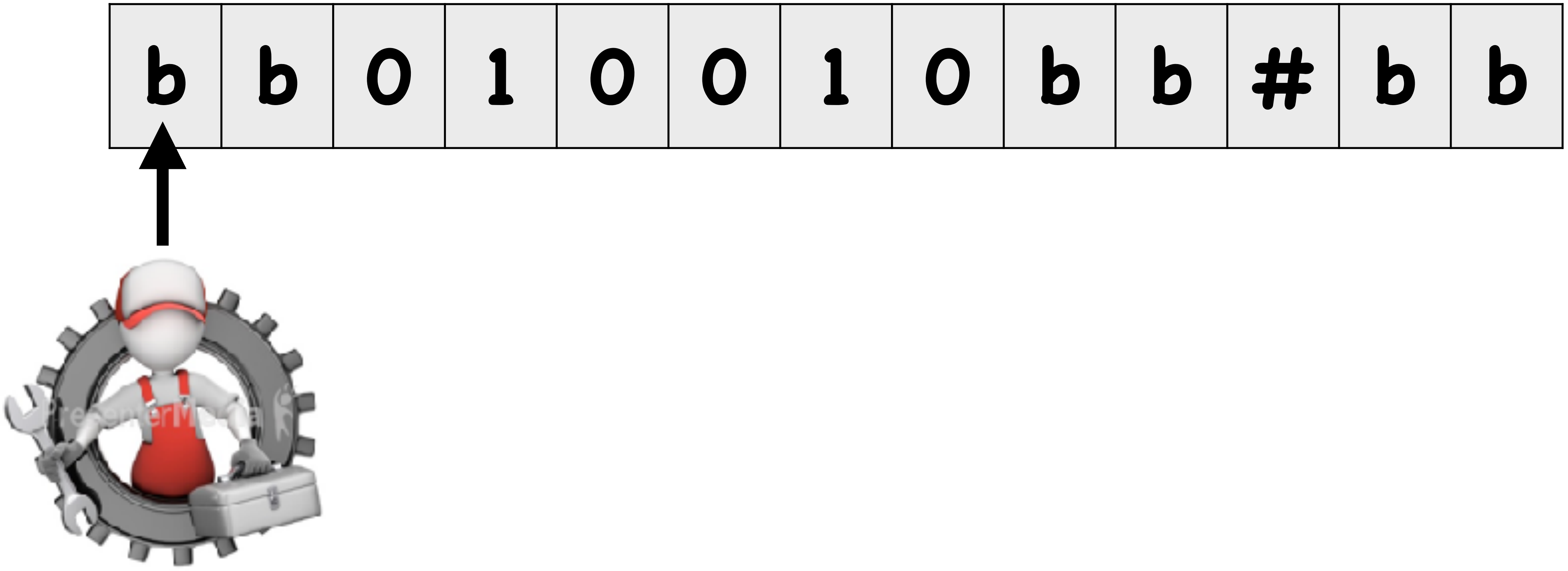


b	b	0	1	0	0	1	0	b	b	#	b	b

Computation Model

Model 1: Single Tape Turing Machine

Palindrome Problem

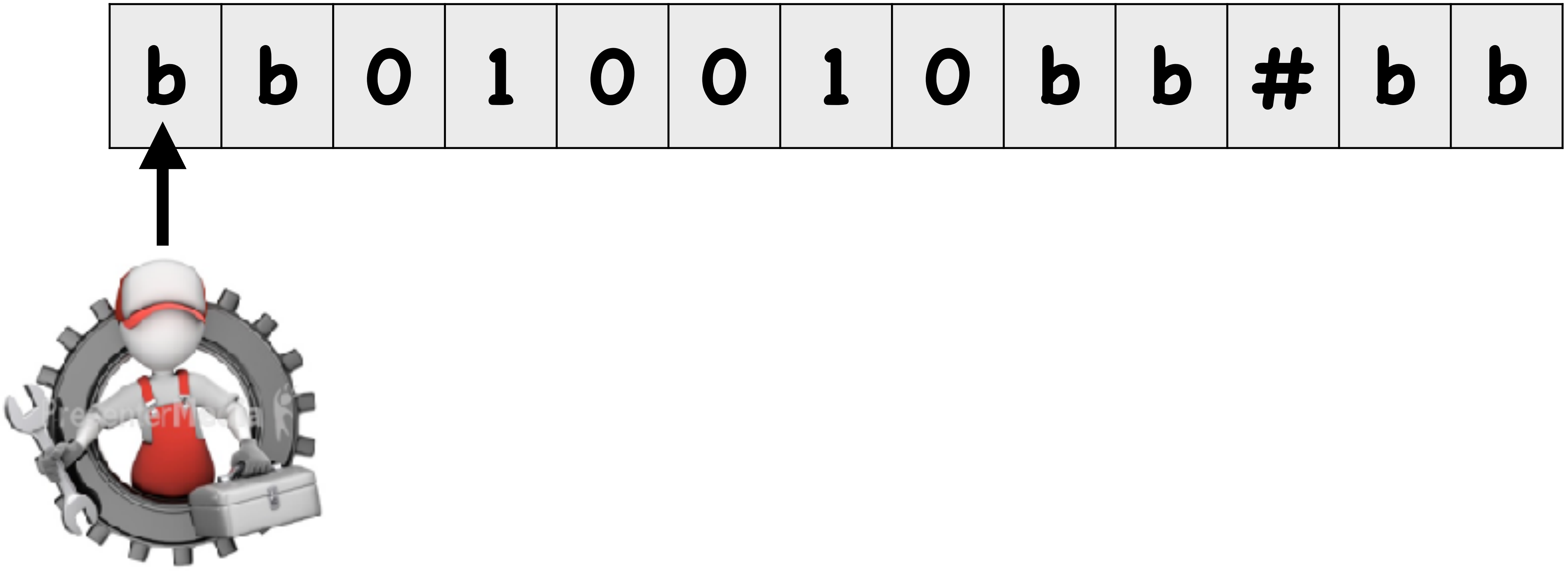


b	b	0	1	0	0	1	0	b	b	#	b	b

Computation Model

Model 1: Single Tape Turing Machine

Palindrome Problem



b	b	0	1	0	0	1	0	b	b	#	b	b

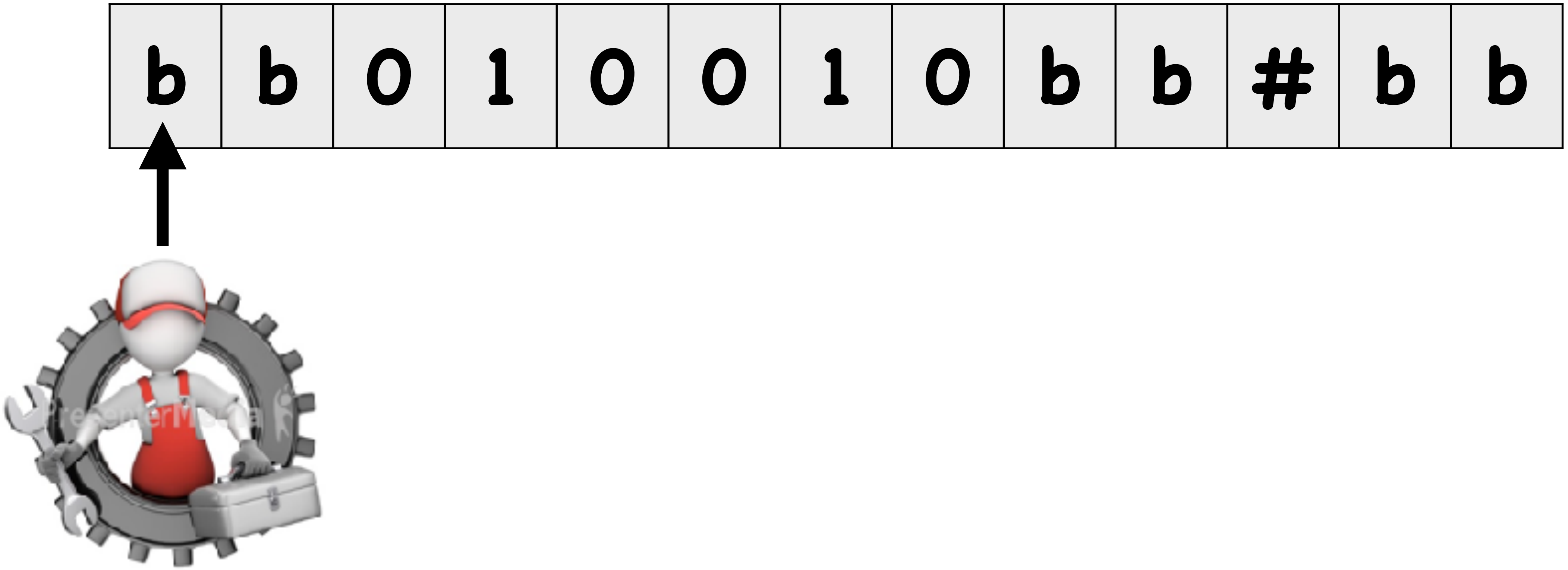
Repeat!

Computation Model

Model 1: Single Tape Turing Machine

O(n²) time

Palindrome Problem



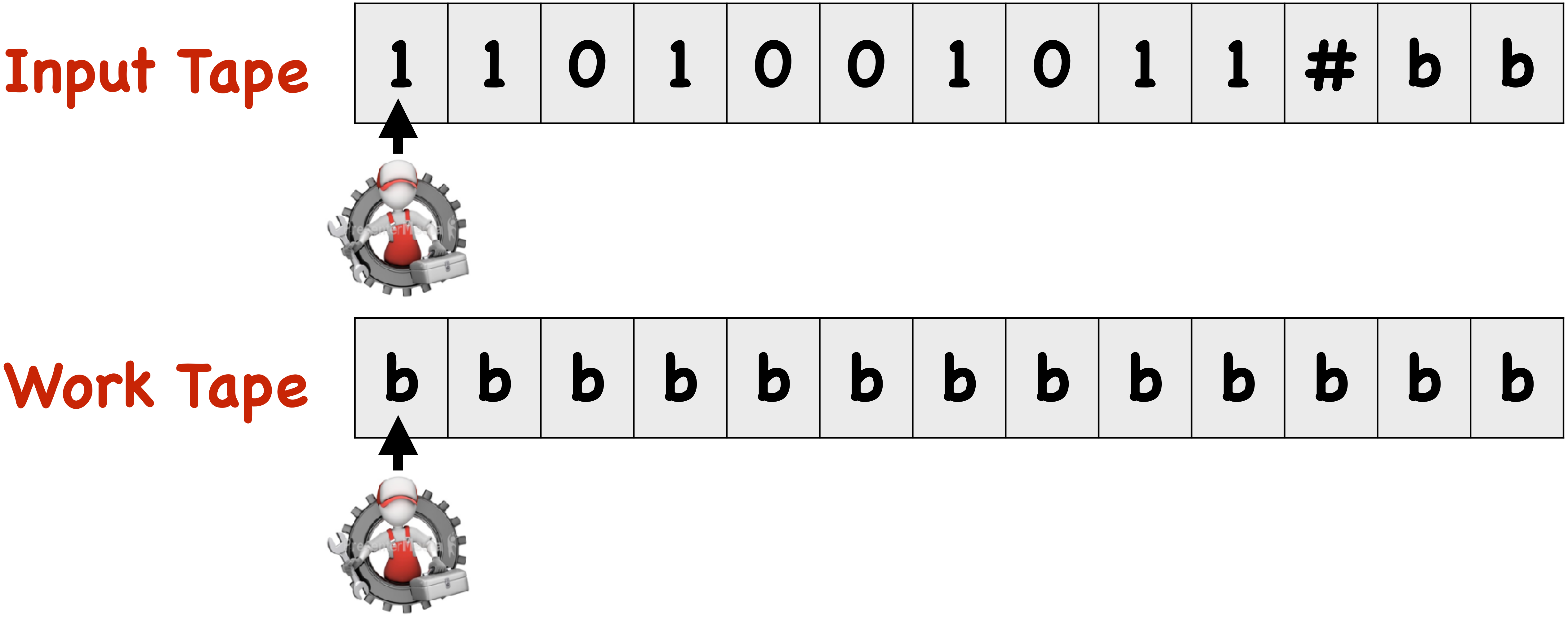
b	b	0	1	0	0	1	0	b	b	#	b	b

Repeat!

Computation Model

Model 2: Two Tape Turing Machine:

Palindrome Problem



Input Tape

1	1	0	1	0	0	1	0	1	1	#	b	b

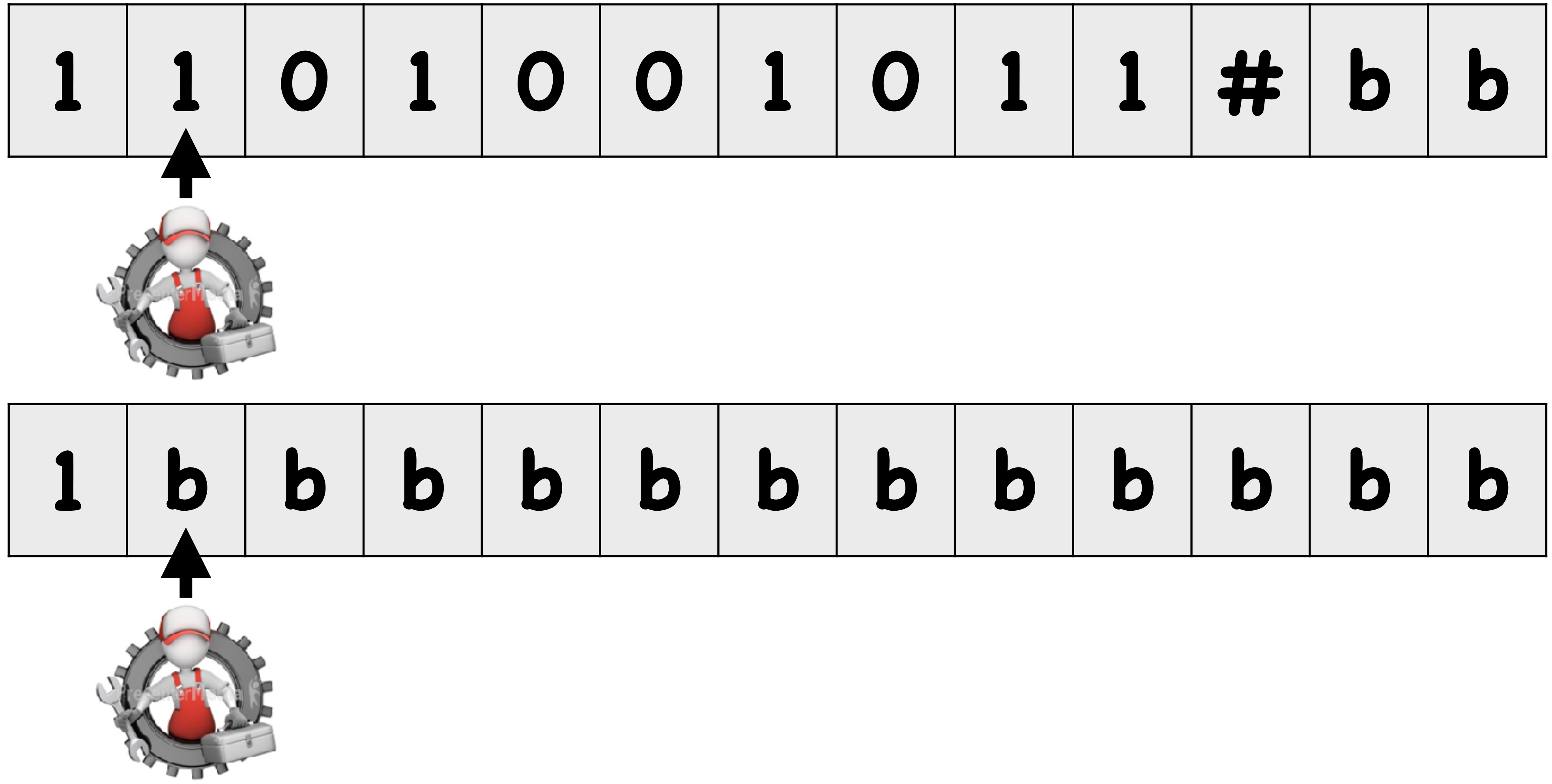
Work Tape

b	b	b	b	b	b	b	b	b	b	b	b	b

Computation Model

Model 2: Two Tape Turing Machine:

Palindrome Problem

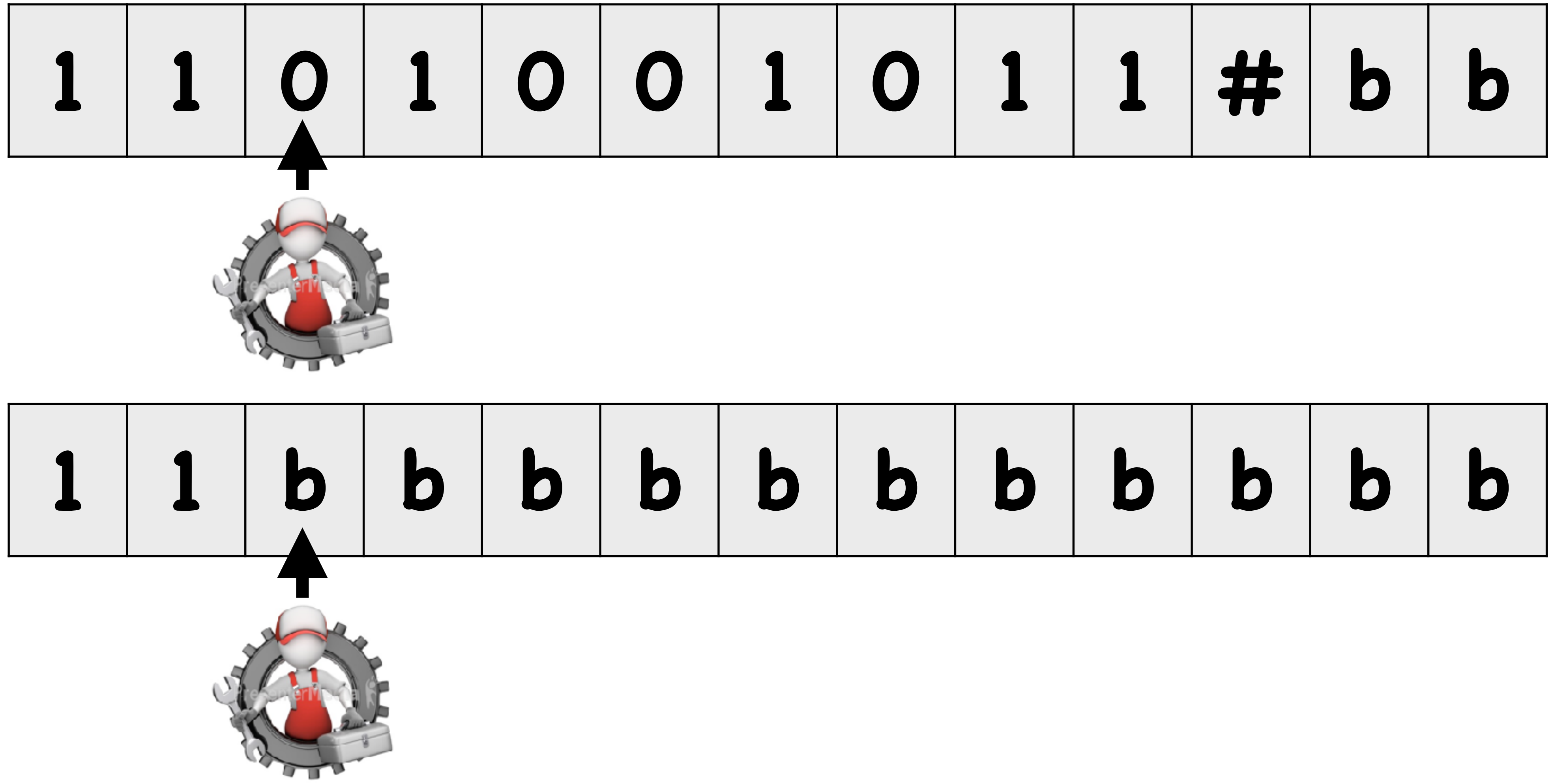


Input Tape	1	1	0	1	0	0	1	0	1	1	#	b	b
Work Tape	1	b	b	b	b	b	b	b	b	b	b	b	b

Computation Model

Model 2: Two Tape Turing Machine:

Palindrome Problem

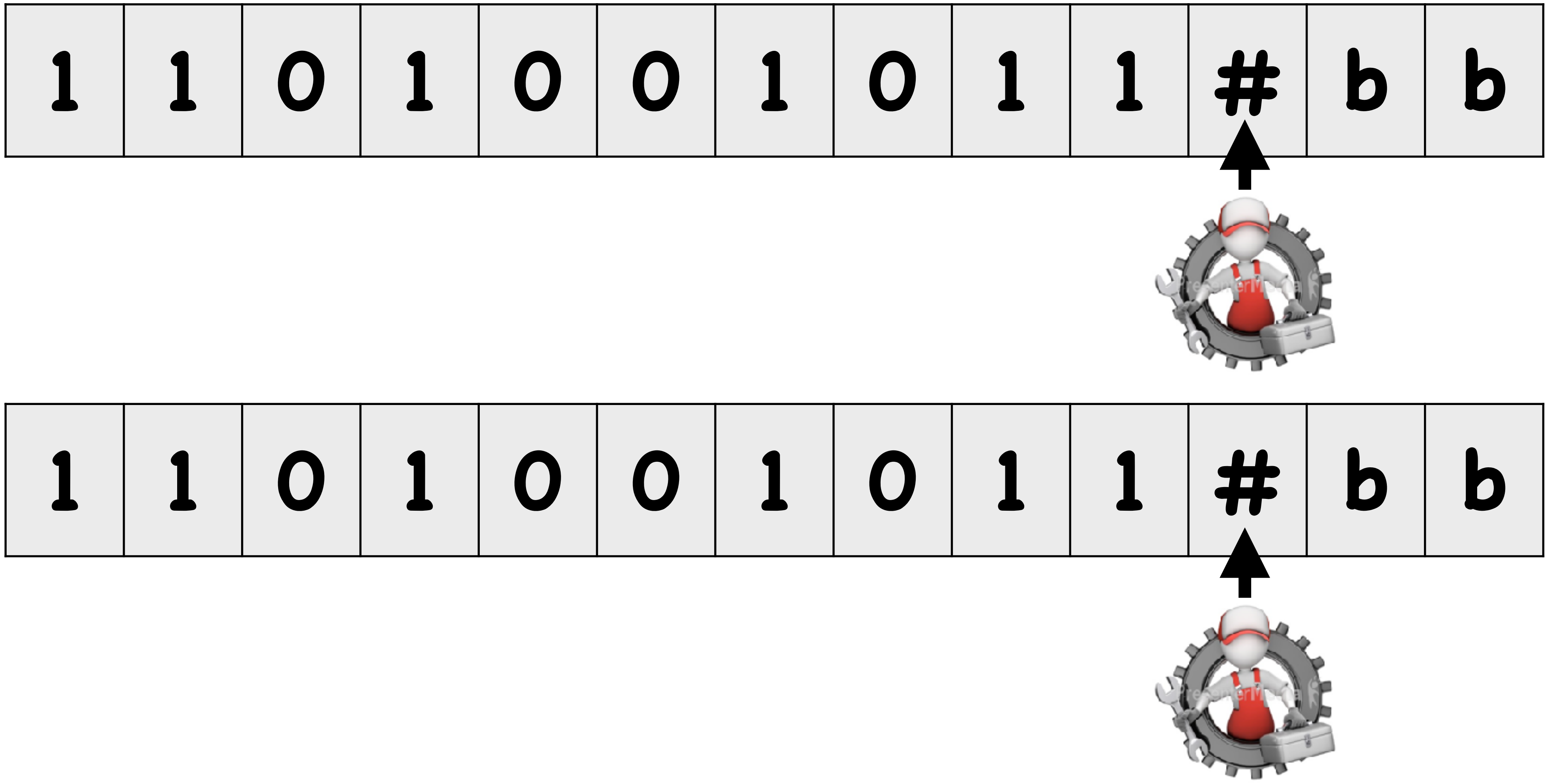


Input Tape	1	1	0	1	0	0	1	0	1	1	#	b	b
Work Tape	1	1	b	b	b	b	b	b	b	b	b	b	b

Computation Model

Model 2: Two Tape Turing Machine:

Palindrome Problem

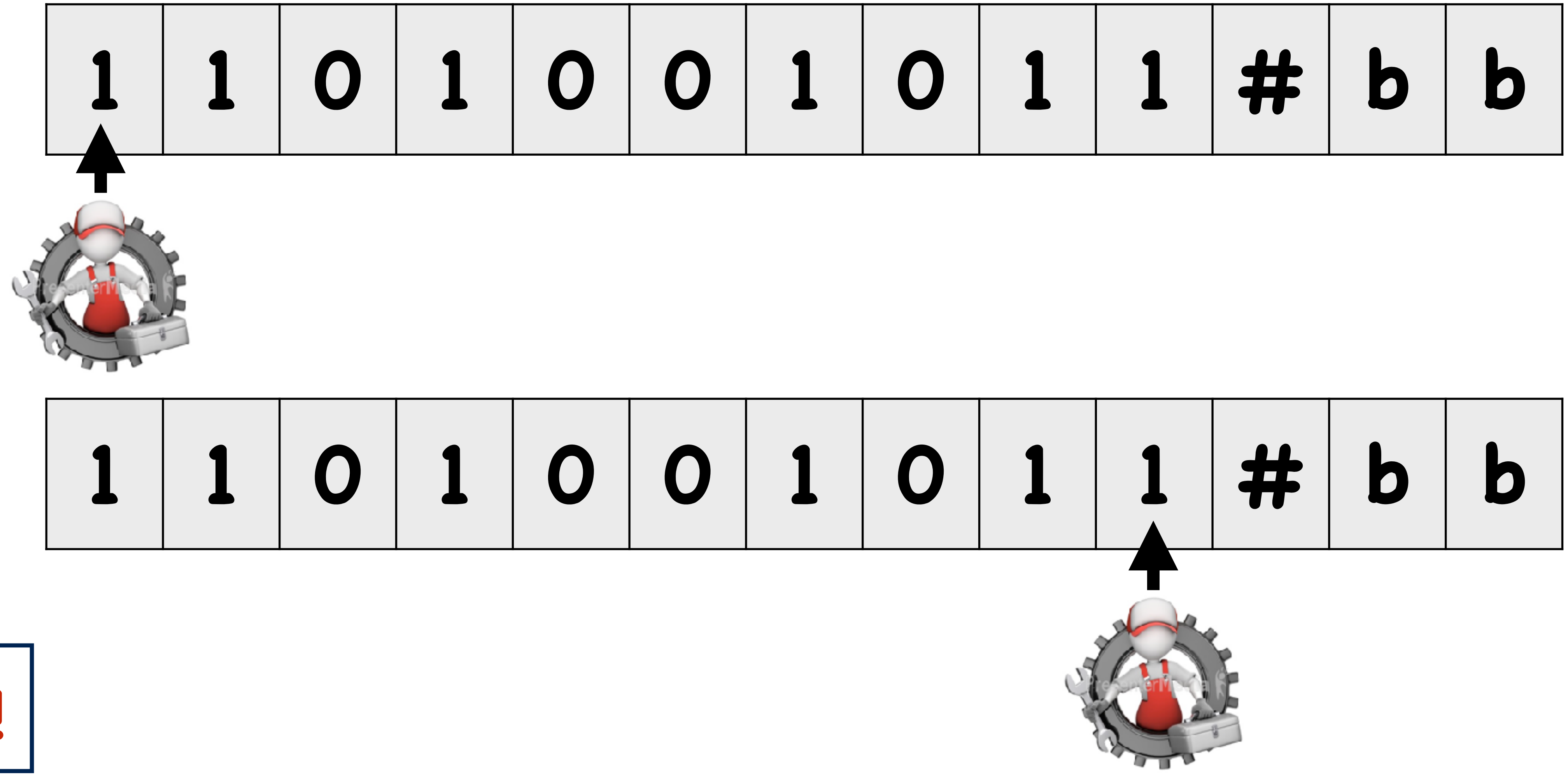


Input Tape	1	1	0	1	0	0	1	0	1	1	#	b	b
Work Tape	1	1	0	1	0	0	1	0	1	1	#	b	b

Computation Model

Model 2: Two Tape Turing Machine:

Palindrome Problem



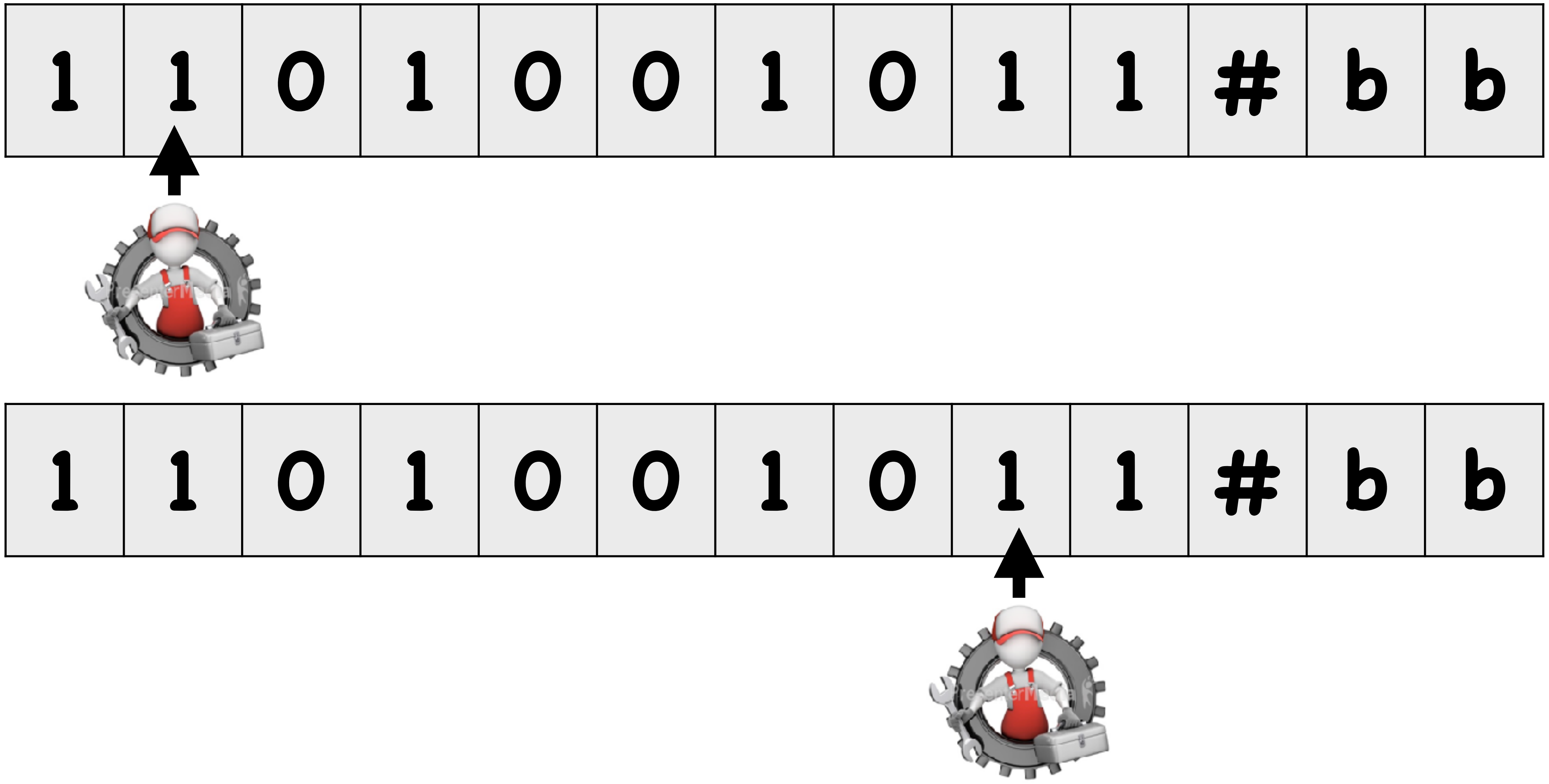
Input Tape	1	1	0	1	0	0	1	0	1	1	#	b	b
Work Tape	1	1	0	1	0	0	1	0	1	1	#	b	b

Compare!

Computation Model

Model 2: Two Tape Turing Machine:

Palindrome Problem



Input Tape	1	1	0	1	0	0	1	0	1	1	#	b	b
Work Tape	1	1	0	1	0	0	1	0	1	1	#	b	b

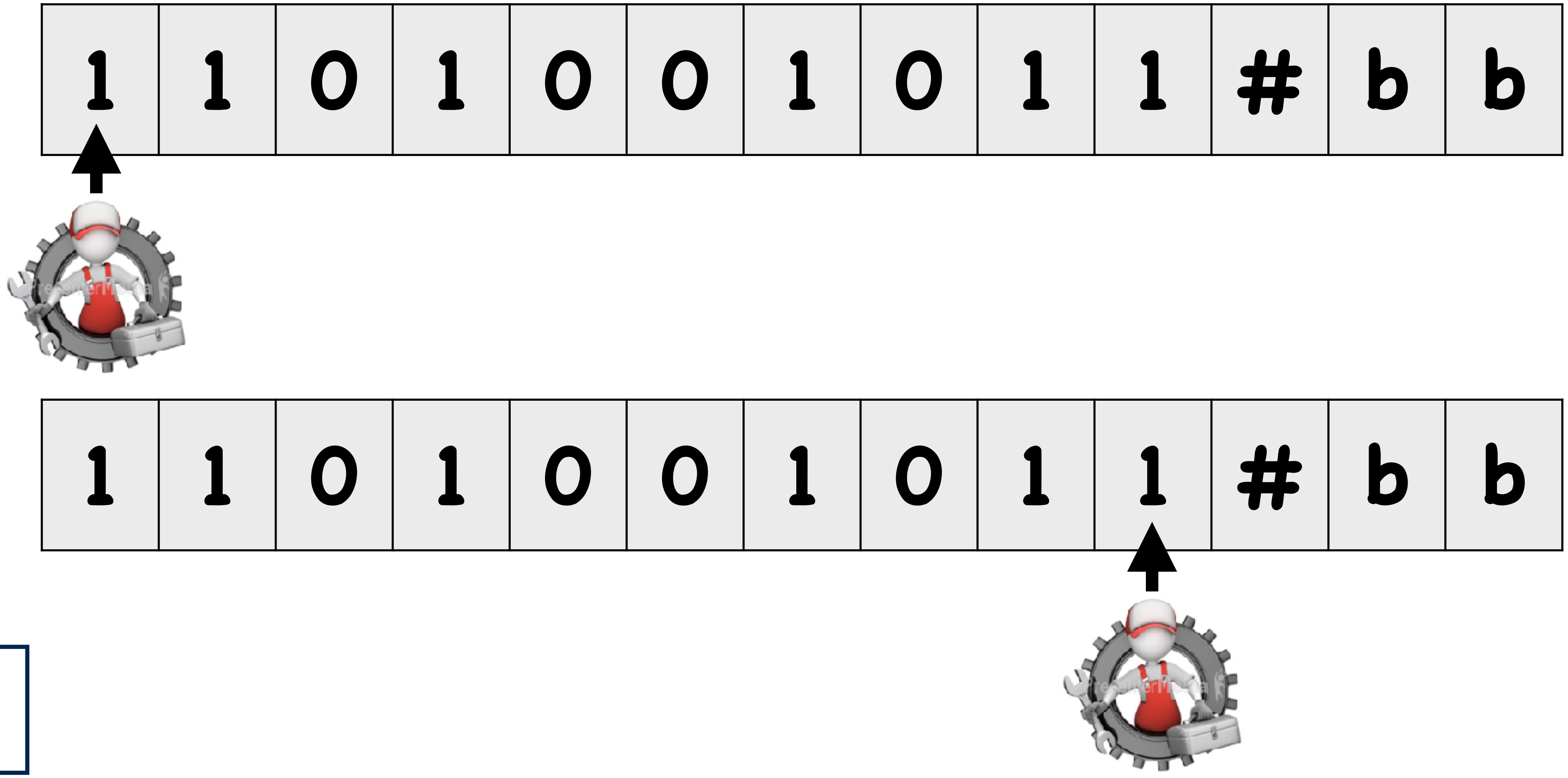
Compare!

Computation Model

Model 2: Two Tape Turing Machine:

O(n) time

Palindrome Problem



Input Tape

1	1	0	1	0	0	1	0	1	1	#	b	b

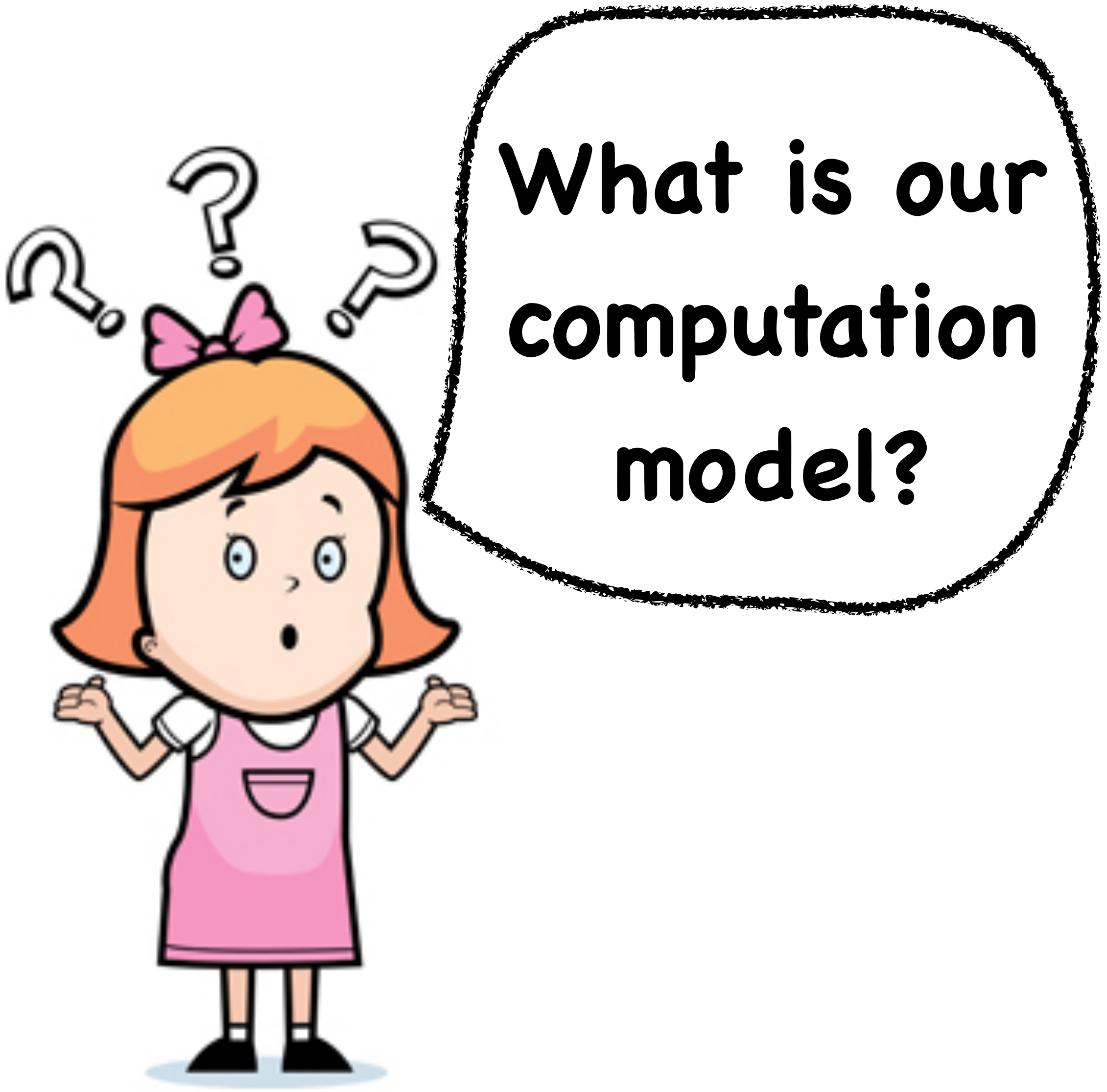
Work Tape

1	1	0	1	0	0	1	0	1	1	#	b	b

Compare!

Computation Model

Our Computation Model



Computation Model

Our Computation Model

Earth is round

But it looks flat, so it must be flat!

Computation Model

Our Computation Model

Earth is round, but for the purpose of building a house you can assume it to be flat.

Computation Model

Our Computation Model: Word RAM Model

✦ Basic operations on words take constant time.

Computation Model

Our Computation Model: Word RAM Model

✦ Basic operations on words take constant time.

✦ All the basic elements in the input can be represented in a word.

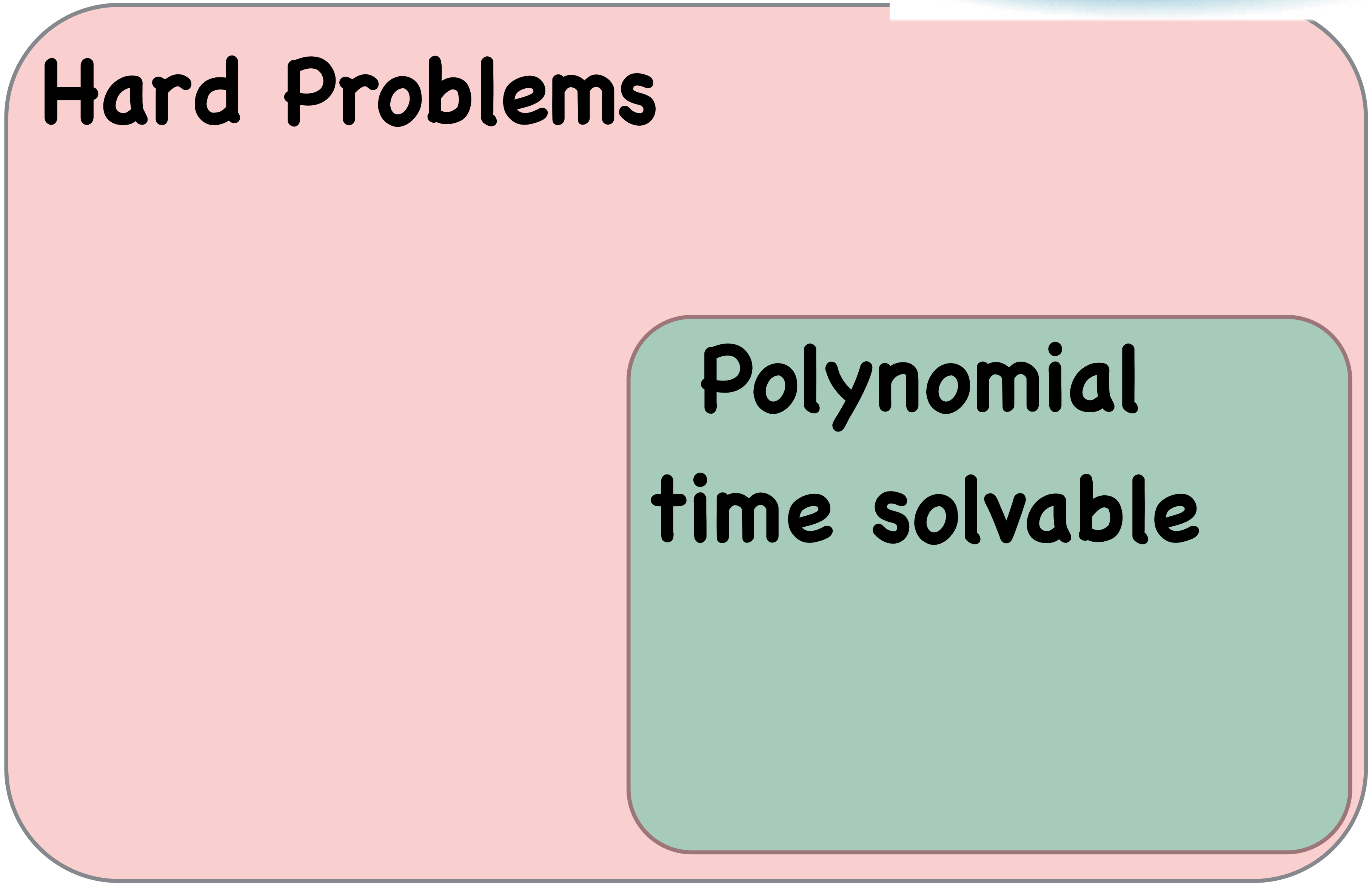
Computation Model

Our Computation Model: Word RAM Model

- ✦ Basic operations on words take constant time.
- ✦ All the basic elements in the input can be represented in a word.
- ✦ Read/write operation of a word takes constant time.

ALGORITHMIC RESEARCH?

FAST(ER) ALGORITHMS!

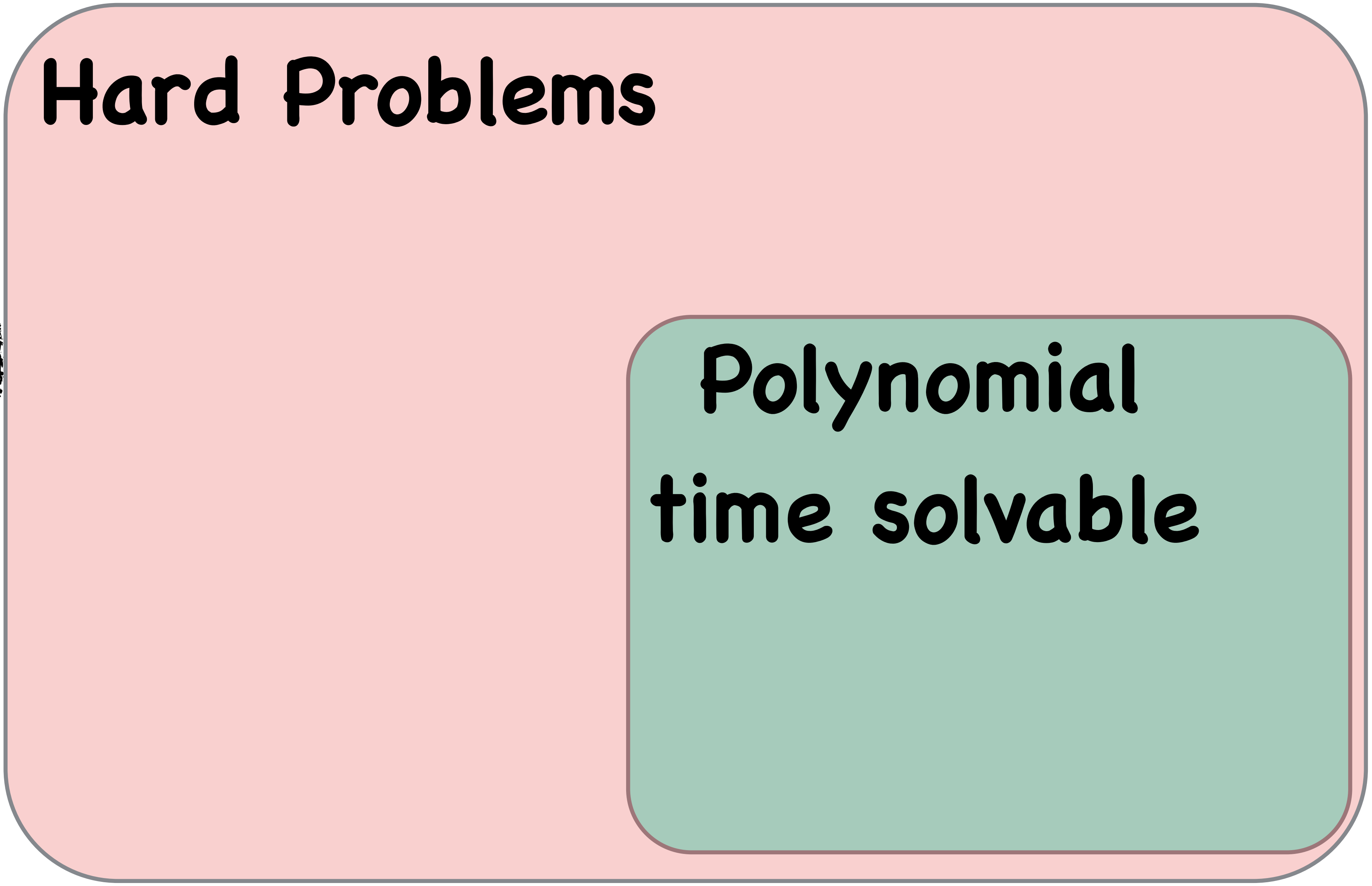


Hard Problems

Polynomial time solvable

ALGORITHMIC RESEARCH?

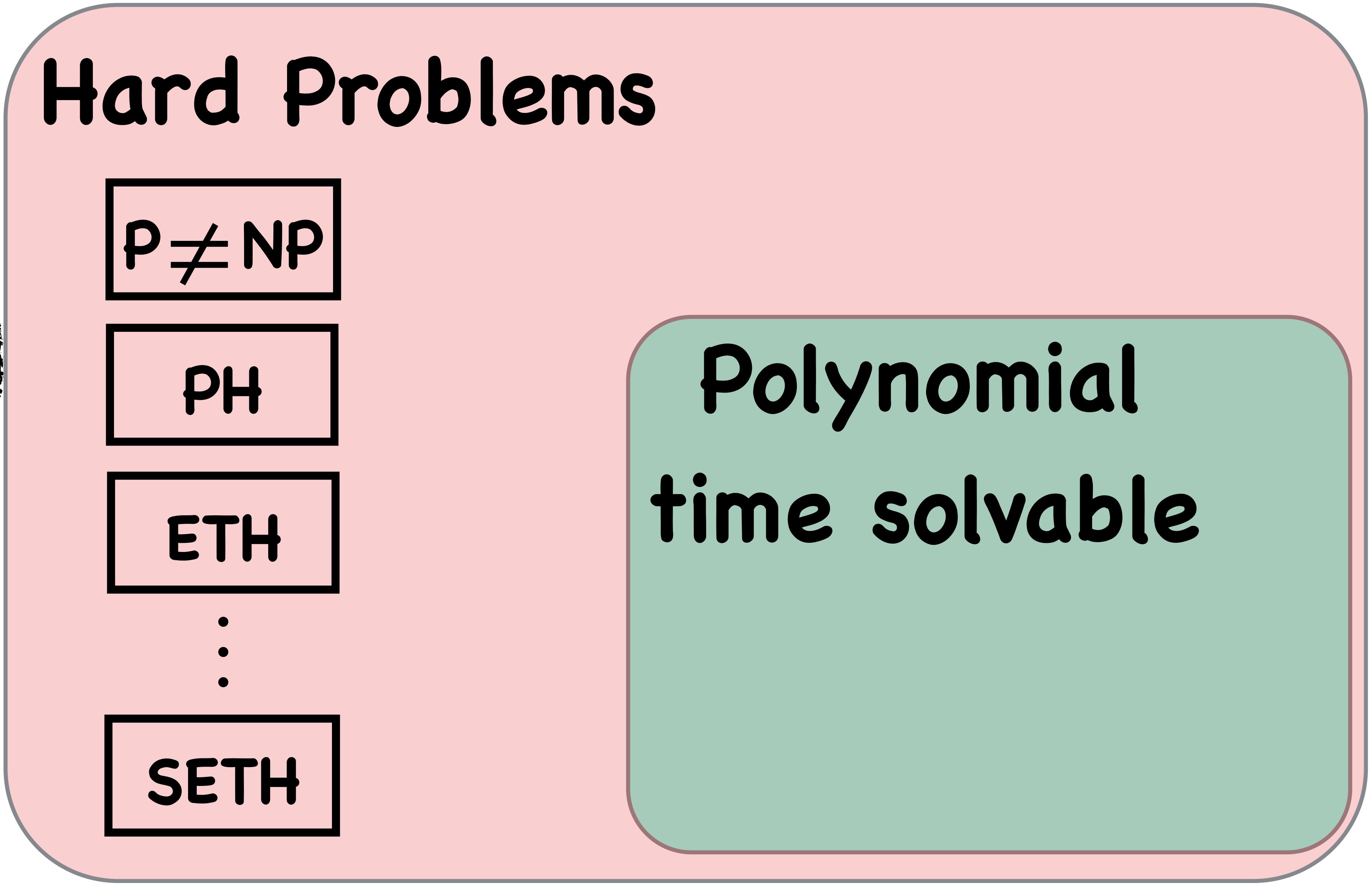
No FAST(ER) ALGORITHM!


Hard Problems

Polynomial time solvable

ALGORITHMIC RESEARCH?

No FAST(ER) ALGORITHM!



Hard Problems

$P \neq NP$

PH

ETH

⋮

SETH

Polynomial time solvable

ALGORITHMIC RESEARCH?

No FAST(ER) ALGORITHM!

Hard Problems

P≠NP

PH

ETH

⋮

SETH

Polynomial time solvable

???

ALGORITHMIC RESEARCH?

Unconditional Lower Bounds?

Hard Problems

- $P \neq NP$
- PH
- ETH
- ⋮
- SETH

Polynomial time solvable

???

ALGORITHMIC RESEARCH?

Unconditional Lower Bounds

We can concretely prove statements like: there is no algorithm for our problem running in time “better” than O(n log n)

ALGORITHMIC RESEARCH?

Unconditional Lower Bounds

- ✦ Finding the maximum of n numbers require n-1 comparisons
- ✦ The best comparison based sorting algorithm must use $\Omega(n \log n)$ time.

And a few more...

ALGORITHMIC RESEARCH?

Unconditional Lower Bounds?

Seems difficult

Hard Problems

$P \neq NP$

PH

ETH

⋮

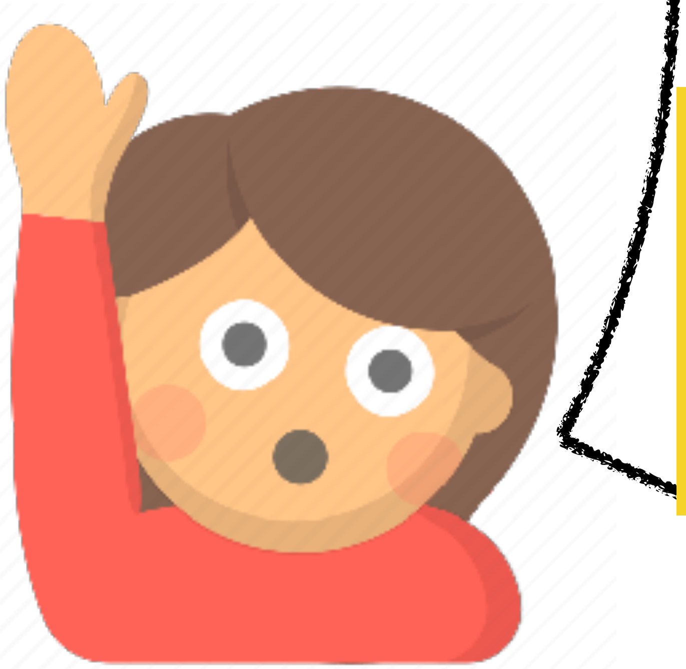
SETH

Polynomial time solvable

???

ALGORITHMIC RESEARCH?

Conditional Lower Bounds?



We strongly believe that for some complicated problem, obtaining “better” algorithms are not possible

↓

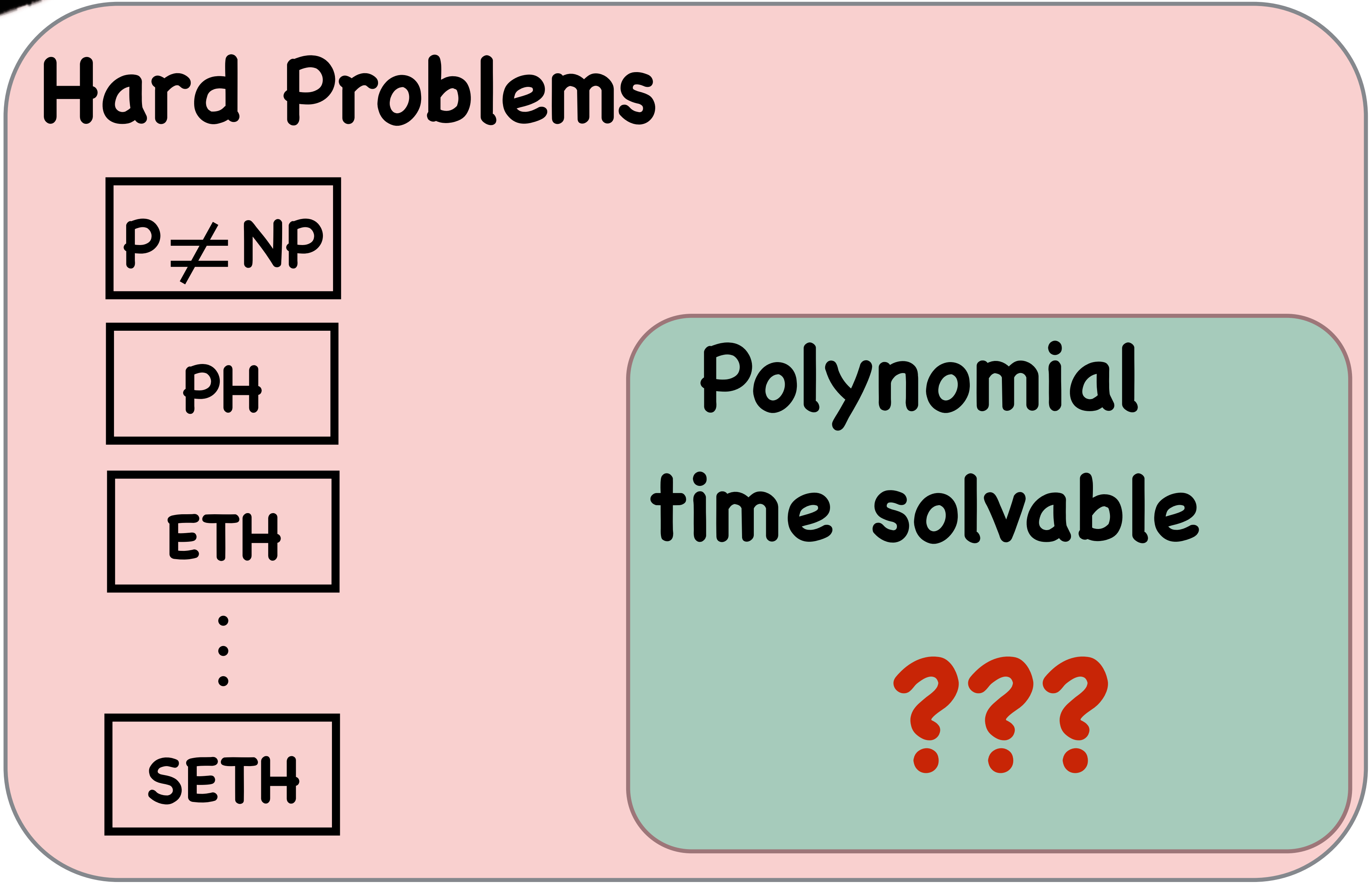
For the problem at hand, if we obtain a “better” algorithm, then the complicated problem has a better algorithm.

ALGORITHMIC RESEARCH?

Conditional Lower Bounds?

ALGORITHMIC
RESEARCH?

Conditional Lower Bounds?



Hard Problems

$P \neq NP$

PH

ETH

⋮

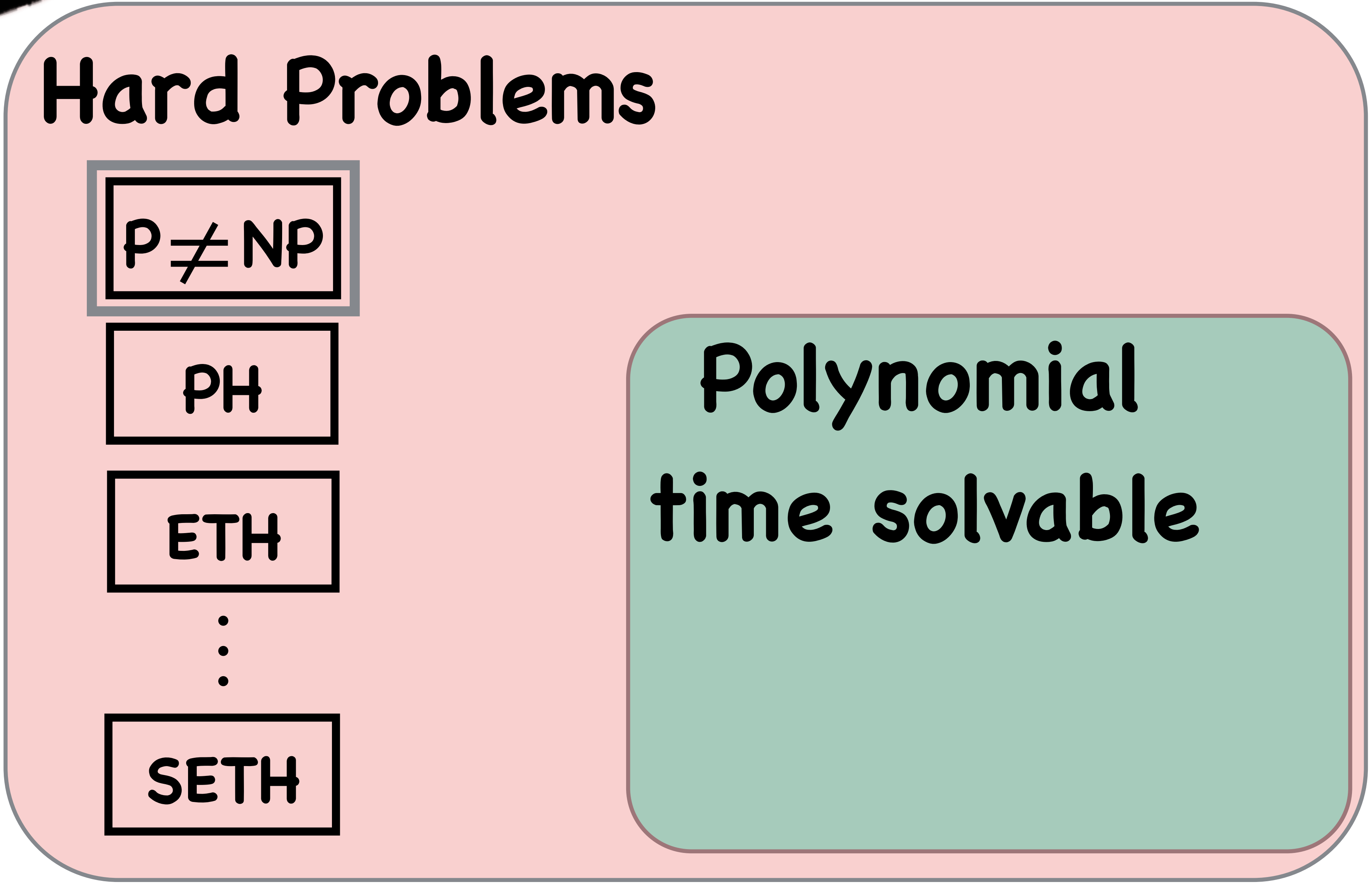
SETH

Polynomial
time solvable

???

ALGORITHMIC RESEARCH?

Conditional Lower Bounds?



Hard Problems

P≠NP

PH

ETH

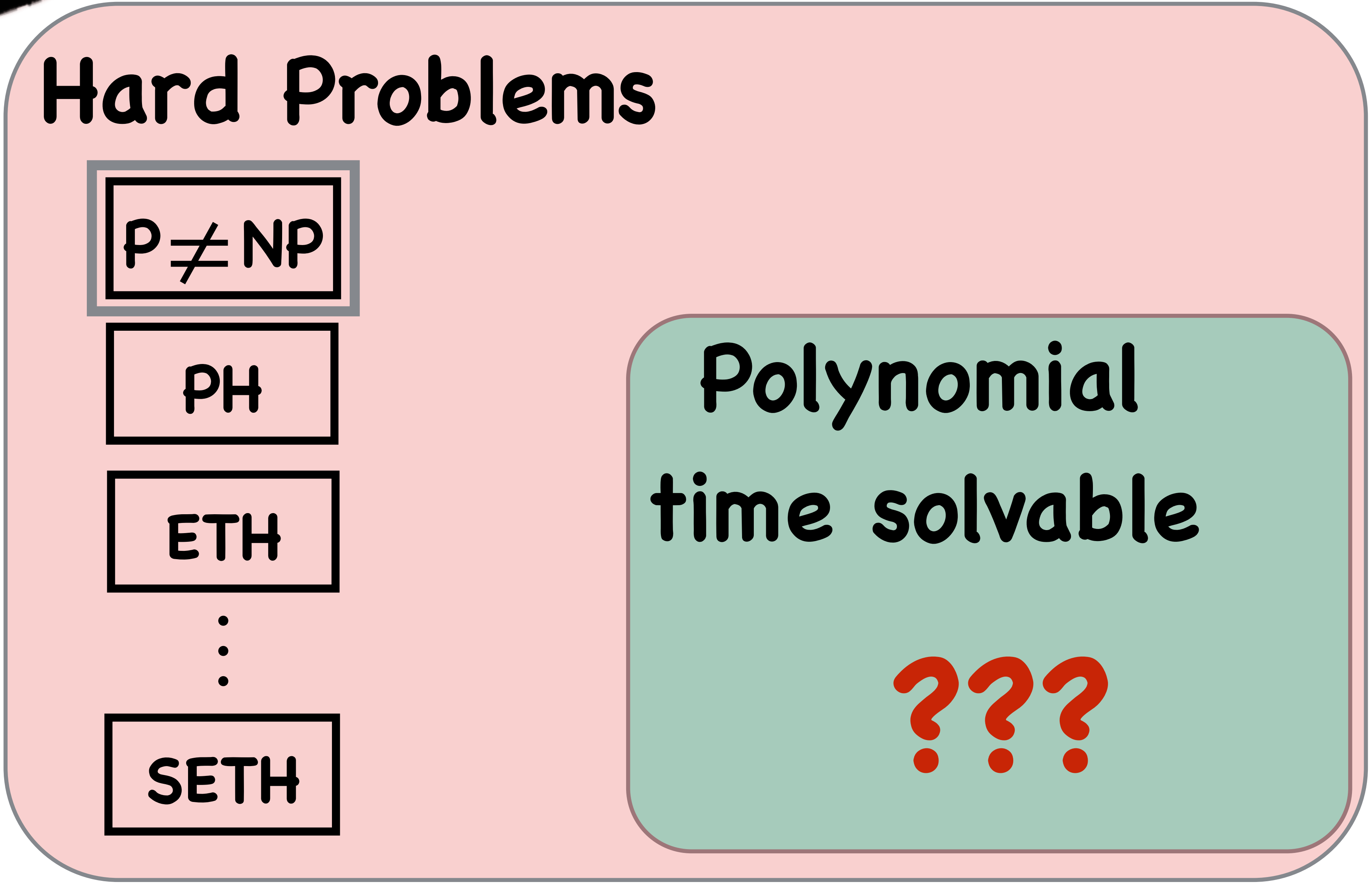
⋮

SETH

Polynomial time solvable

ALGORITHMIC
RESEARCH?

Conditional Lower Bounds?


Hard Problems

P≠NP

PH

ETH

⋮

SETH

Polynomial time solvable

???

ALGORITHMIC RESEARCH?

Conditional Lower Bounds?

Any favourite problem in P?

Hard Problems

P≠NP

PH

ETH

⋮

SETH

Polynomial time solvable

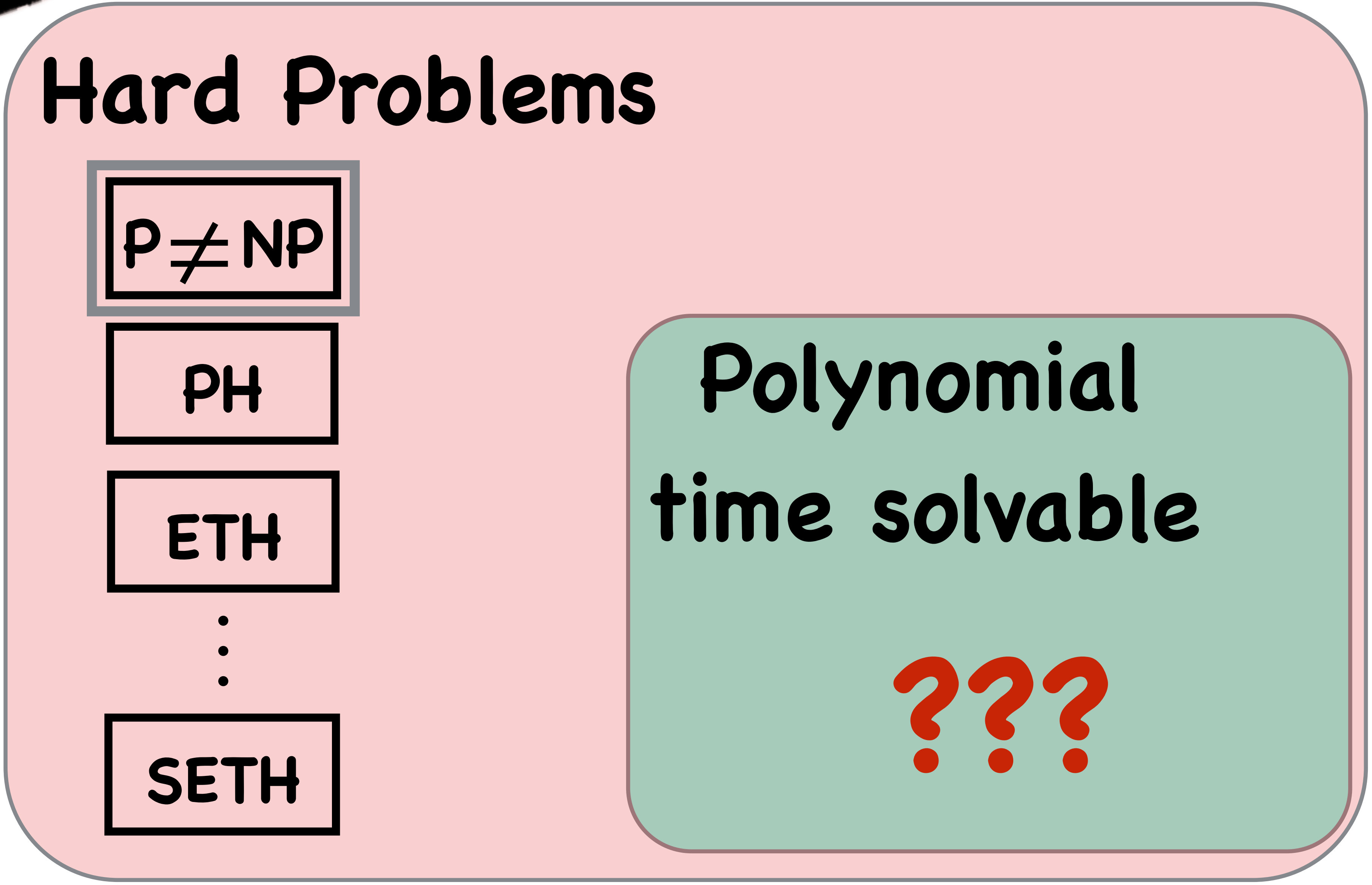
???

ALGORITHMIC RESEARCH?

Conditional Lower Bounds?

Seems difficult

Can we find all pair shortest path in $O(n^{2.99})$-time?



Hard Problems

$P \neq NP$

PH

ETH

⋮

SETH

Polynomial time solvable

???

Following the league…

Mimicking Approach for NP-Hardness Results

Consider a problem P

- ✦ P admits an algorithm running in time n^k, where k is some constant.
- ✦ Despite of lot of work no significantly better algorithm for P has been obtained.

Here, by significantly better algorithm we mean an algorithm running in time n^{k-e}, where e>0.

Mimicking Approach for NP-Hardness Results

Consider a problem P

✦ P admits an algorithm running in time n^k, where k is some constant.

✦ Despite of lot of work no significantly better algorithm for P has been obtained.

Here, by significantly better algorithm we mean an algorithm running in time n^{k-e}, where e>0.

New problem Q

Mimicking Approach for NP-Hardness Results

Consider a problem P

- P admits an algorithm running in time n^k, where k is some constant.
- Despite of lot of work no significantly better algorithm for P has been obtained.

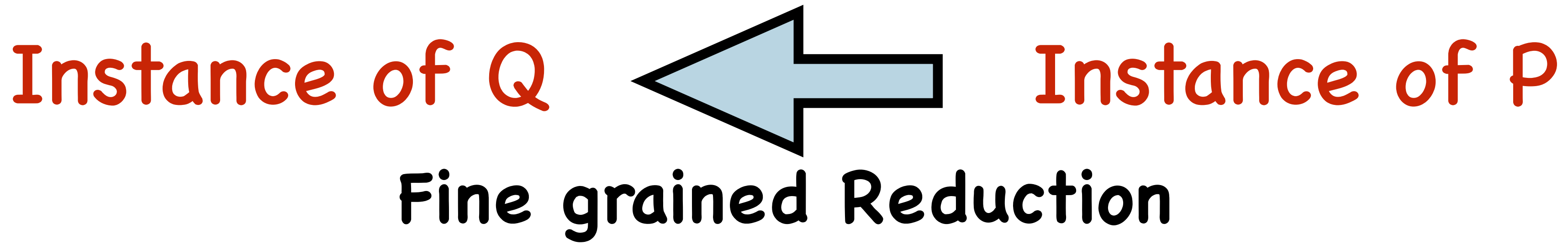
Better algorithm for Q → Better algorithm for P

New problem Q

Mimicking Approach for NP-Hardness Results

Consider a problem P

- P admits an algorithm running in time n^k, where k is some constant.
- Despite of lot of work no significantly better algorithm for P has been obtained.



Instance of Q

Instance of P

Fine grained Reduction

Evidence that problem Q does not admit very fast algorithm as well.

New problem Q

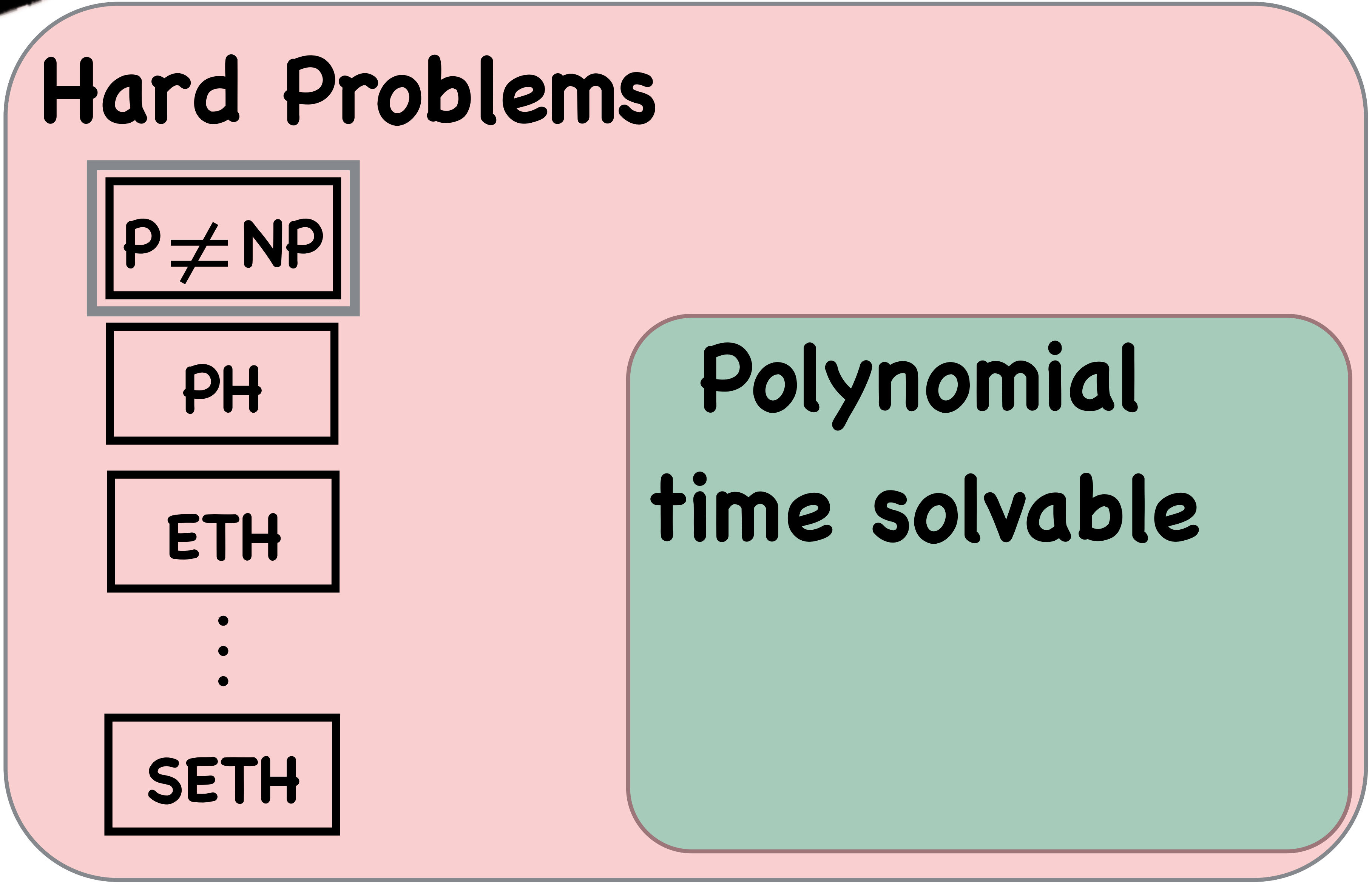
Focus of Recent Works

- ✦ Mimicking the approach towards showing hardness results: Identifying hard problems.

- ✦ Basing the hardness results on some reasonable Complexity Theoretic Conjectures.

ALGORITHMIC RESEARCH?

Conditional Lower Bounds?



Hard Problems

P≠NP

PH

ETH

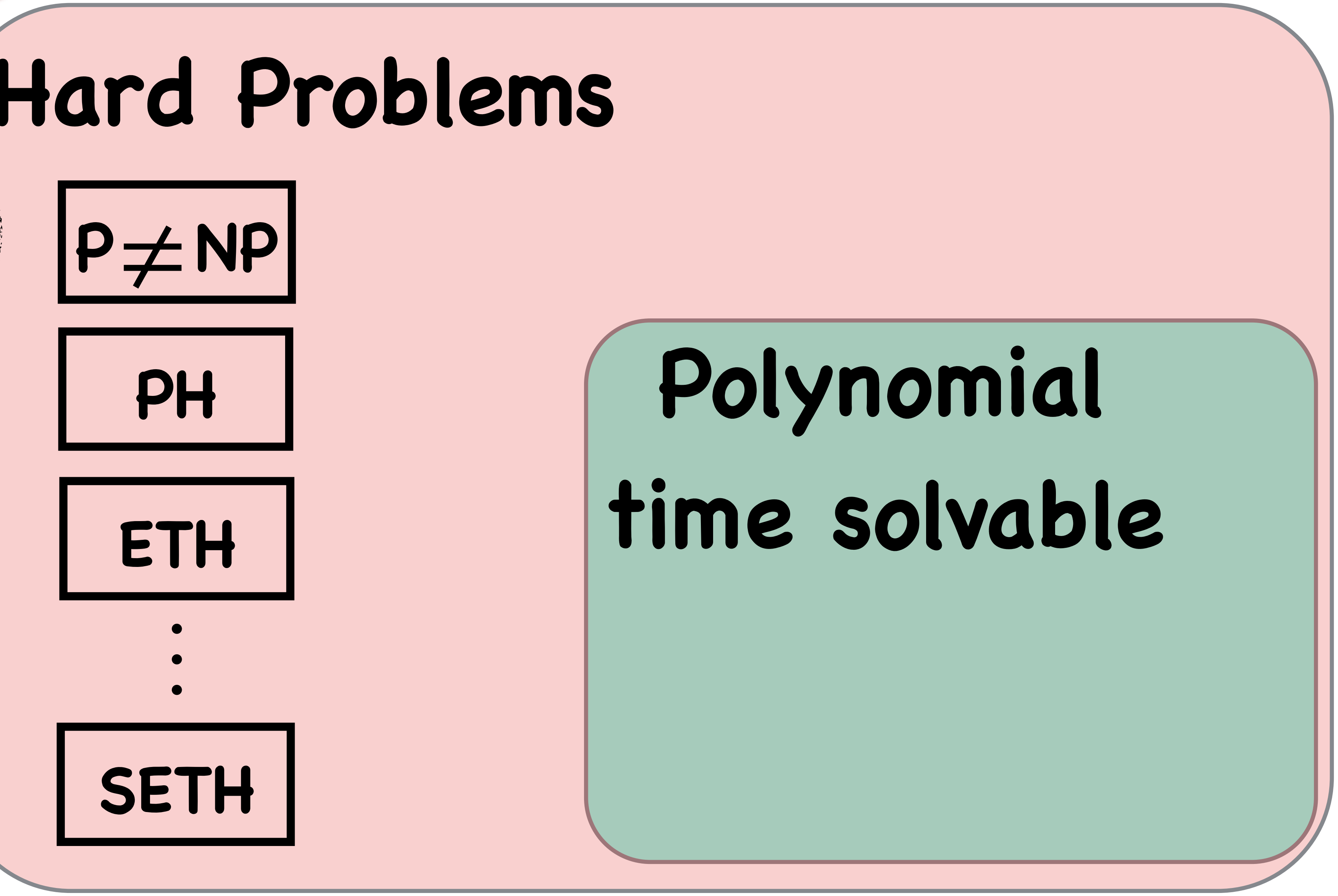
⋮

SETH

Polynomial time solvable

ALGORITHMIC RESEARCH?

Conditional Lower Bounds?


Hard Problems

P≠NP

PH

ETH

⋮

SETH

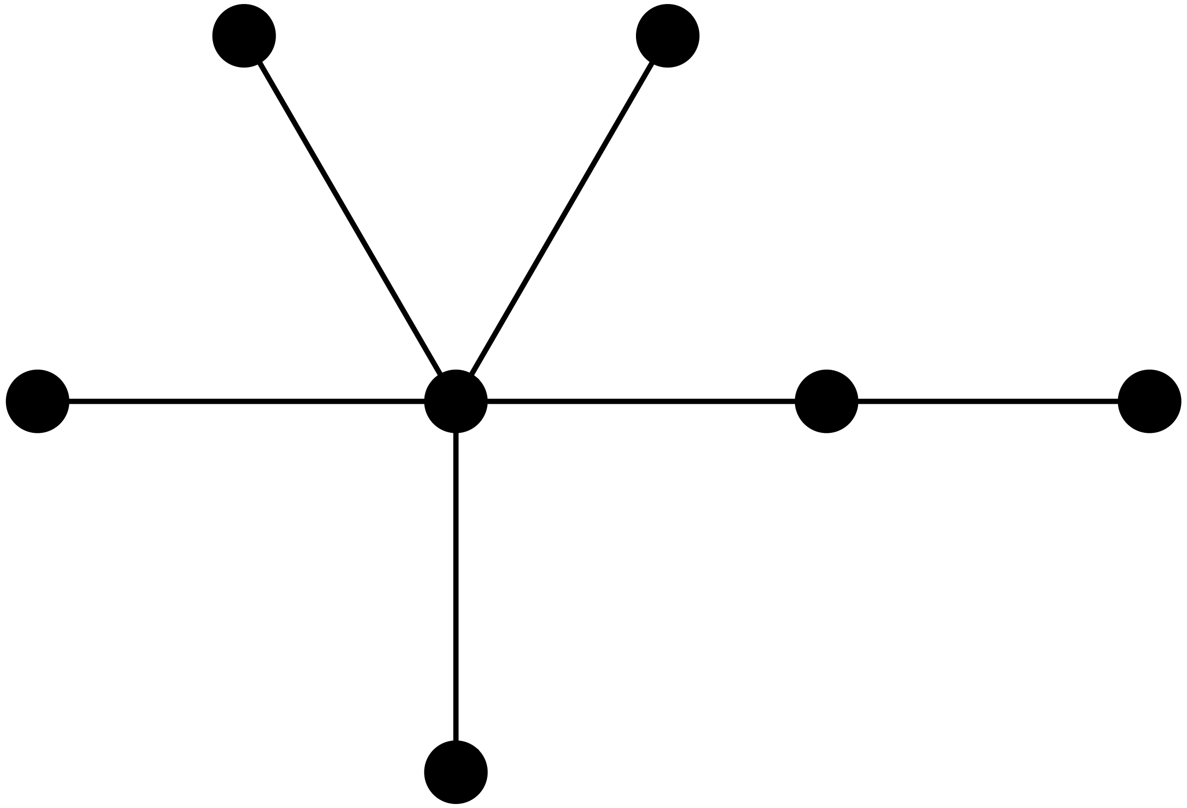
Polynomial time solvable

Some Problems in P with no improvements

Graph Algorithms:

✦ Finding a centre of a graph.

Centre: $$\arg \min_{v \in V(G)} \max_{u \in V(G)} \operatorname{dist}(u, v)$$



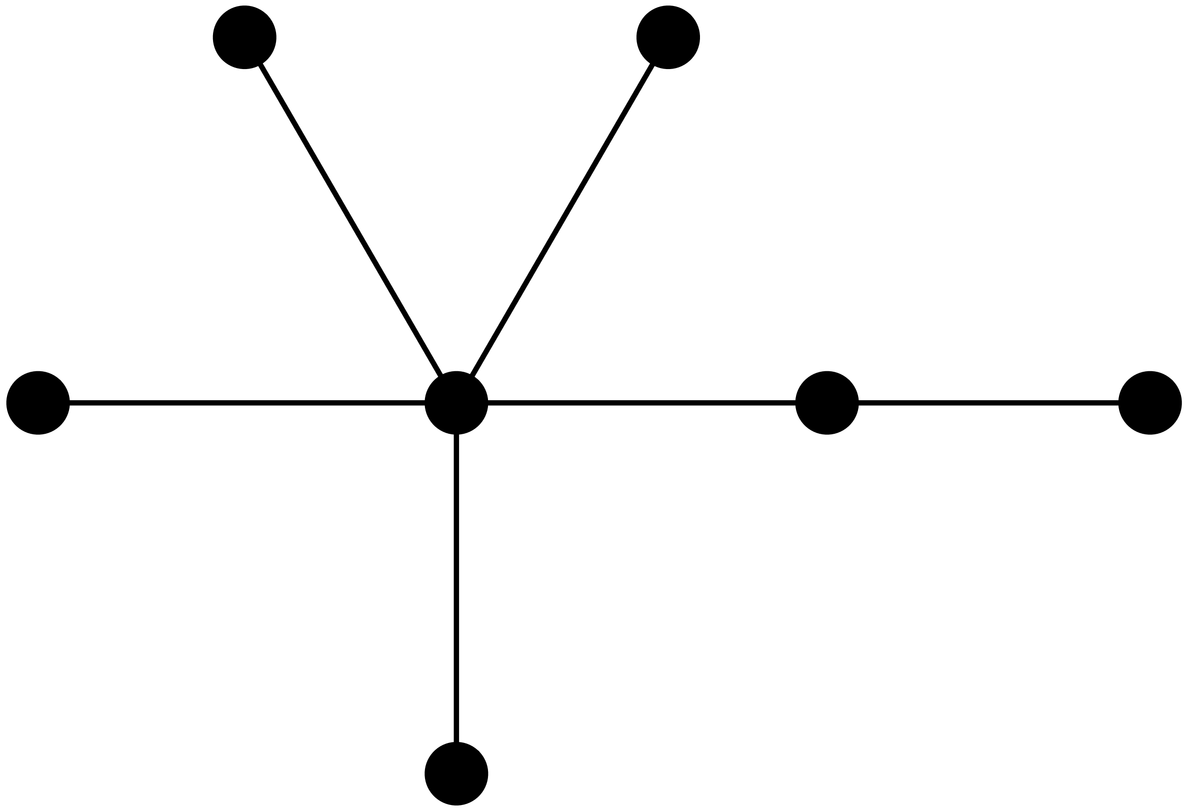
Input: Graph G

Some Problems in P with no improvements

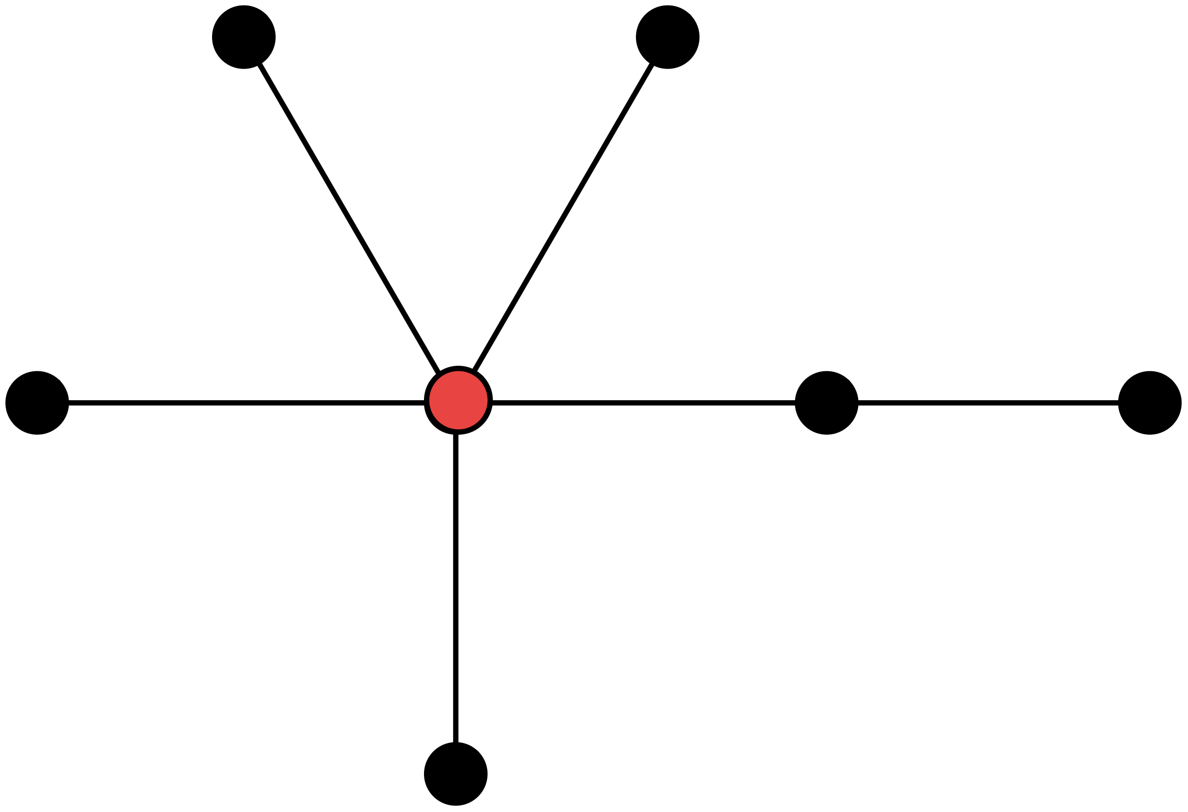
Graph Algorithms:

✦ Finding a centre of a graph.

Centre: $\arg \min\limits_{v \in V(G)} \max\limits_{u \in V(G)} \operatorname{dist}(u, v)$



Input: Graph G



Output: Centre of G

Some Problems in P with no improvements

Graph Algorithms:

- ✦ Finding a centre of a graph.

  - ☑ Can be computed using Floyd-Warshall's algorithm for computing all pair shortest path. -O(n^3) time

No better algorithm known.

Some Problems in P with no improvements

Computational Biology:

✦ Longest Common Subsequence.

a a b c c d e w f g h

x y a c b p c a e h b

Some Problems in P with no improvements

Computational Biology:

- Longest Common Subsequence.

a a b c c d e w f g h

x y a c b p c a e h b

a b c e h

Some Problems in P with no improvements

Computational Biology:

✦ Longest Common Subsequence.

☑ Can be computed using a classical dynamic programming based algorithm. -O(n^2) time

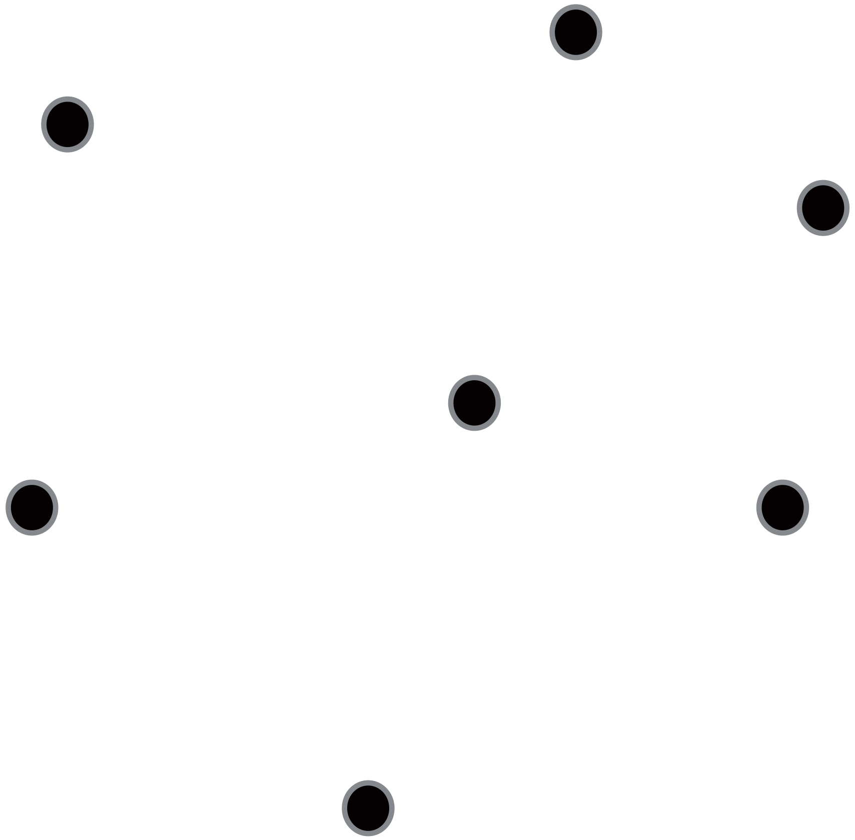
No (significantly) better algorithm known.

Some Problems in P with no improvements

Computational Geometry:

- Points in general position.

(no three point collinear)



Input: Points in the plane

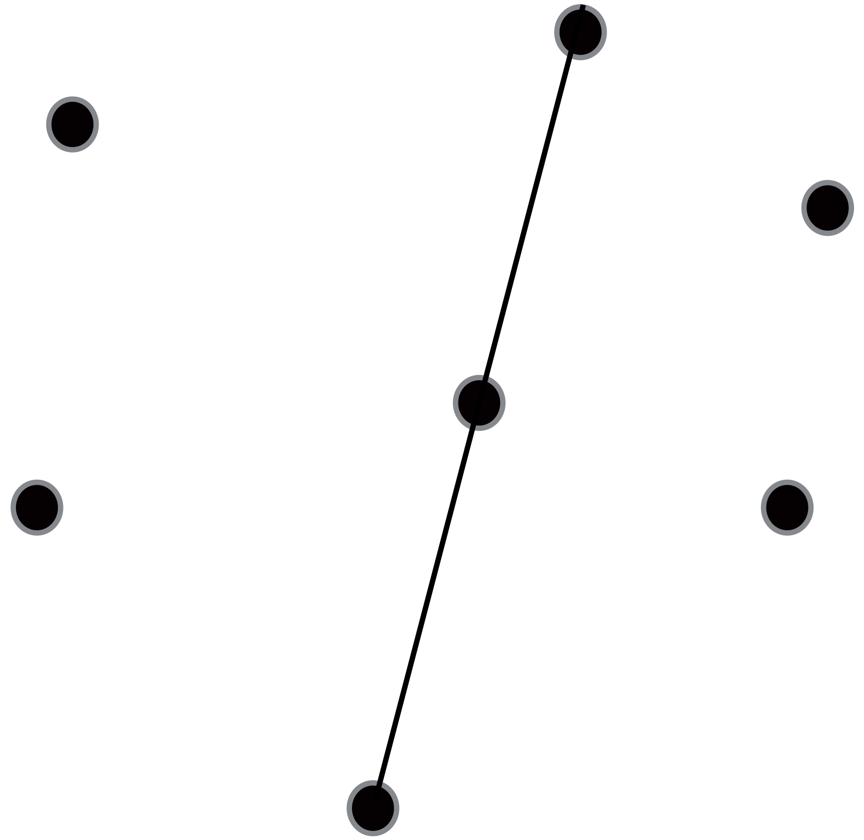
Output: Yes/ No

Some Problems in P with no improvements

Computational Geometry:

✦ Points in general position.

(no three point collinear)



No

Input: Points in the plane

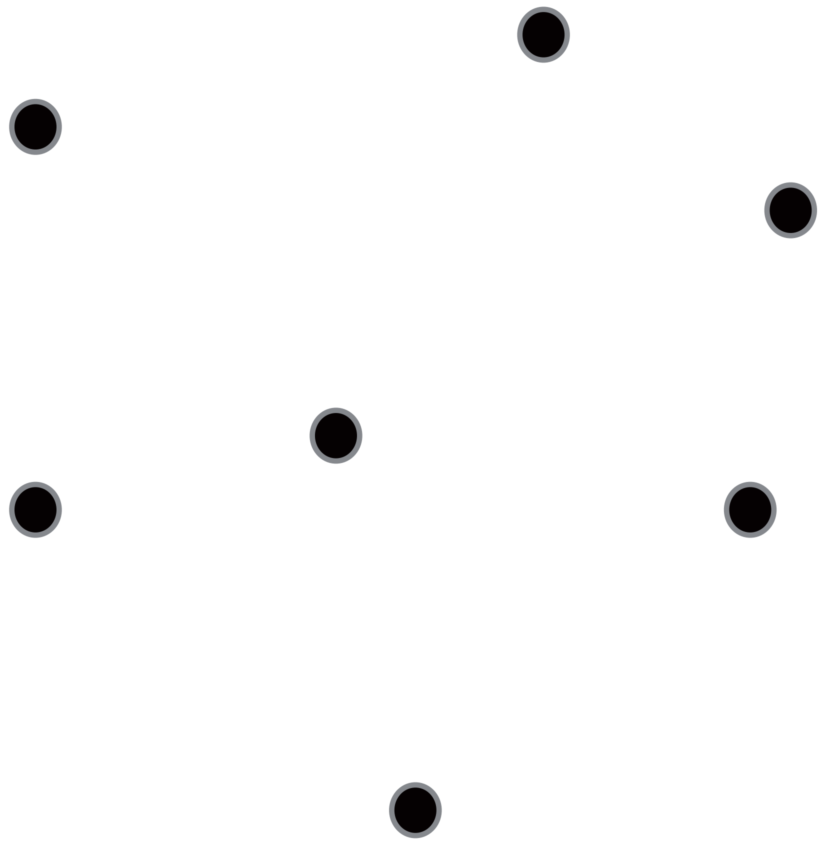
Output: Yes/ No

Some Problems in P with no improvements

Computational Geometry:

✦ Points in general position.

(no three point collinear)



Yes

Input: Points in the plane **Output: Yes/ No**

Some Problems in P with no improvements

Computational Geometry:

- ✦ Points in general position.

  ☑ Can be computed using a classical algorithm.
  -$\hat{O}(n^2)$ time

No (significantly) better algorithm known.

This course

- Revise basics of algorithmic analysis
- Revise theory of NP-completeness
- Unconditional lower bounds
- Improvements using lookups
- Polynomial Method
- Decision Trees
- Better algorithms for 3-SUM
- FFT based improvements
- Famous conjectures from P
- Relating problems in P and beyond
- BMM and its applications
- Understanding recent research paper(s)

Revise basics of algorithmic analysis

Revise theory of NP-completeness

Unconditional lower bounds

Improvements using lookups

Polynomial Method

Decision Trees

Better algorithms for 3-SUM

FFT based improvements

Famous conjectures from P

Relating problems in P and beyond

BMM and its applications

Understanding recent research paper(s)

Thanks