# CMSC 28100

# Introduction to Complexity Theory

Spring 2025
Instructor: William Hoza

Which problems

can be solved

through computation?

# Deciding a language in time $T$

- Let $Y \subseteq \{0,1\}^*$ and let $T: \mathbb{N} \to [0, \infty)$ be a function
- **Definition:** We say that $Y$ can be decided in time $T$ if there exists a Turing machine $M$ such that
  - $M$ decides $Y$, and
  - For every $n \in \mathbb{N}$ and every $w \in \{0,1\}^n$, the running time of $M$ on $w$ is at most $T(n)$

# The complexity class $\mathrm{P}$

- **Definition:** For any function $T: \mathbb{N} \to [0, \infty)$, we define

$$\mathrm{TIME}(T) = \{Y \subseteq \{0,1\}^* : Y \text{ can be decided in time } O(T)\}$$

- **Definition:**

$$\mathrm{P} = \{Y \subseteq \{0,1\}^* : Y \text{ can be decided in time } \mathrm{poly}(n)\}$$

$$= \bigcup_{k=1}^{\infty} \mathrm{TIME}(n^k)$$

- “Polynomial time”

# $\mathrm{P}$: Our model of tractability

- Let $Y \subseteq \{0,1\}^*$
- If $Y \in \mathrm{P}$, then we will consider $Y$ "tractable"
- If $Y \notin \mathrm{P}$, then we will consider $Y$ "intractable"

Which problems

can be solved

through computation?

# Which languages are in $\mathbf{P}$?

# Example 1: Primality testing

- $\text{PRIMES} = \{\langle K \rangle : K \text{ is a prime number}\}$

**Theorem:** $\text{PRIMES} \in \text{P}$

- **Proof attempt:** For $M = 2, 3, \ldots, K - 1$, check if $K/M$ is an integer.
- That proof is not correct. The algorithm runs in $\text{poly}(K)$ time, but our time budget is only $\text{poly}(n)$ where $n = |\langle K \rangle| \approx \log K$!
- The theorem is true, but the proof is beyond the scope of this course

# Example 2: The $\mathrm{EVENPAL}^*$ problem

- Let $\mathrm{EVENPAL} = \{x \in \mathrm{PALINDROMES} : |x| \text{ is even}\}$
- Let $\mathrm{EVENPAL}^* = \{x_1 x_2 \dots x_k : k \geq 0 \text{ and } x_1, \dots, x_k \in \mathrm{EVENPAL}\}$
- Example: $100111 \in \mathrm{EVENPAL}^*$
  - $x_1 = 1001$ and $x_2 = 11$
- Example: $1010 \notin \mathrm{EVENPAL}^*$

# Deciding $\mathbf{EVENPAL}^*$ in polynomial time

- $\mathrm{EVENPAL}^* = \{x_1 x_2 \ldots x_k : k \geq 0 \text{ and } x_1, \ldots, x_k \in \mathrm{EVENPAL}\}$

**Theorem:** $\mathrm{EVENPAL}^* \in \mathrm{P}$

- **Proof attempt 1**: Given $w \in \{0,1\}^*$, try all possible decompositions

$$w = x_1 x_2 \ldots x_k$$

  - Time complexity $\Omega(2^n)$… 😟

# Deciding $\mathbf{EVENPAL}^*$ in polynomial time

- $\text{EVENPAL}^* = \{x_1 x_2 \dots x_k : k \geq 0 \text{ and } x_1, \dots, x_k \in \text{EVENPAL}\}$

**Theorem:** $\text{EVENPAL}^* \in \text{P}$

- **Proof:** We'll use an algorithm technique called "dynamic programming"
- Key observation: If $w \in \{0,1\}^* \setminus \{\epsilon\}$, then $w \in \text{EVENPAL}^*$ if and only if there exist $u \in \text{EVENPAL}^*$ and $y \in \text{EVENPAL}$ such that $w = uy$ and $|u| < |w|$

# Deciding $\mathrm{EVENPAL}^*$ in polynomial time

- Let $w$ be the input, $w = w_1 w_2 \ldots w_n$, where $w_i \in \{0, 1\}$
- Plan: For each $i \in \{0, 1, \ldots, n\}$, we will compute a Boolean value $b_i$ that indicates whether $w_1 w_2 \ldots w_i \in \mathrm{EVENPAL}^*$

1) Let $b_0 =$ Tr
2) For $i = 1$ to
   a) If there ... $i =$ True
   b) Otherw
3) Accept if $b_n$

**What should $b_0$ be?**

**A:** True

**B:** False

**C:** It depends on $w$

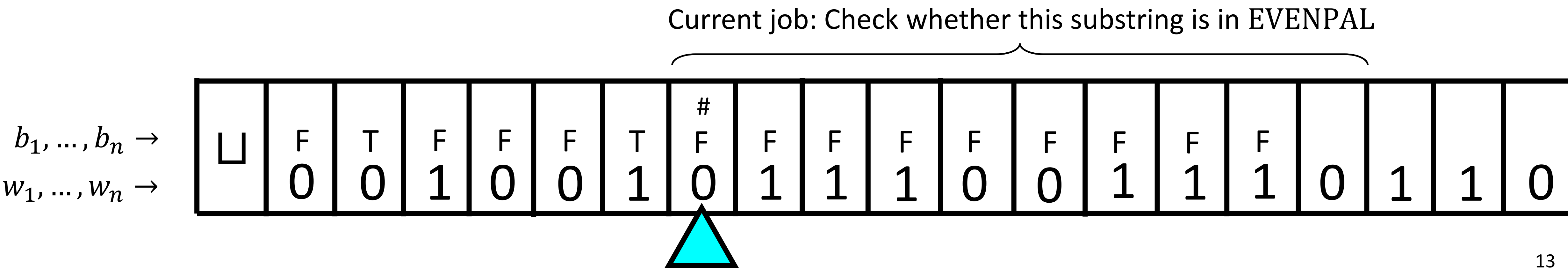
**D:** It's not well-defined

Respond at PollEv.com/whoza or text "whoza" to 22333

# Deciding $\mathrm{EVENPAL}^*$ in polynomial time

1) Let $b_0 = \mathrm{True}$
2) For $i = 1$ to $n$:
    a) If there exists $j < i$ such that $b_{j-1}$ is True and $w_j \dots w_i \in \mathrm{EVENPAL}$, then set $b_i = \mathrm{True}$
    b) Otherwise, set $b_i = \mathrm{False}$
3) Accept if $b_n$ is True; reject if $b_n$ is False

- TM implementation: Store $b_i$ in $w_i$'s cell, and write # in $w_j$'s cell



# Deciding $\mathrm{EVENPAL}^*$ in polynomial time

1) Let $b_0 = \mathrm{True}$
2) For $i = 1$ to $n$:
   a) If there exists $j < i$ such that $b_{j-1}$ is True and $w_j \dots w_i \in \mathrm{EVENPAL}$, then set $b_i = \mathrm{True}$
   b) Otherwise, set $b_i = \mathrm{False}$
3) Accept if $b_n$ is True; reject if $b_n$ is False

- Outer loop ($i$) does $O(n)$ iterations; inner loop ($j$) does $O(n)$ iterations
- We can check whether $w_j \dots w_i \in \mathrm{EVENPAL}$ in time $O(n^2)$
- Total time complexity: $O(n^4) = \mathrm{poly}(n)$ ✔

# Time complexity: Theory vs. practice

- ⚠️ **Caution:** It takes time to move the head to a desired location!
- E.g., consider an algorithm for deciding $\mathrm{PALINDROMES}$:

Given an array of bits $x$:

1) For $i = 1$ to $n$:
    a) If $x[i] \neq x[n-i]$, reject
2) Accept

$\leftarrow n$ iterations

$\leftarrow O(1)$ time per iteration “in practice,”

but not on a Turing machine!

# Is the Turing machine model a good model?

- We defined $\mathrm{P}$ to be the set of languages that can be decided in polynomial time on a Turing machine
- **OBJECTION:** “Time complexity on a Turing machine doesn’t match time complexity in practice, so we should use a more powerful model of computation.”

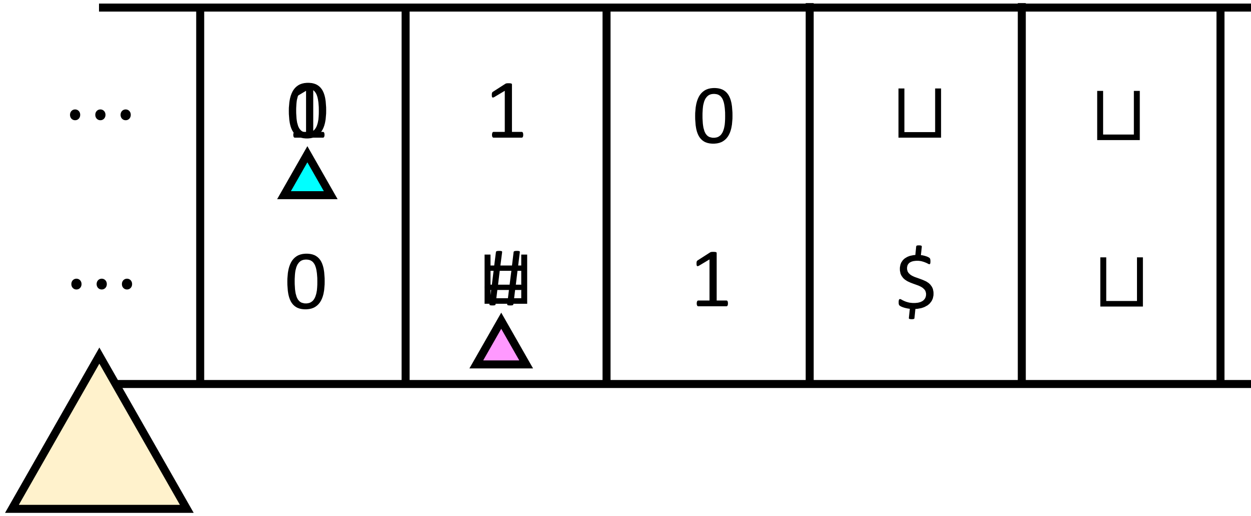
# Multi-tape Turing machines, revisited

- Let $Y \subseteq \{0, 1\}^*$, let $k$ be a positive integer, and let $T: \mathbb{N} \to \mathbb{N}$

**Theorem:** If there is a $k$-tape Turing machine that decides $Y$ with time complexity $T(n)$, then there is a $1$-tape Turing machine that decides $Y$ with time complexity $O(T(n)^2)$.

# Efficiently simulating $k$ tapes using one tape

- **Proof sketch (1 slide):** For simplicity, assume $T(n) \geq n$
- Recall: To simulate step $i$, we scan back and forth over $n + 2i$ cells of the tape




- Simulating one step of the $k$-tape machine takes $O\big(n + T(n)\big)$ steps
- Overall time complexity: $T(n) \cdot O\big(n + T(n)\big) = O(T(n)^2)$

# Robustness of $\mathrm{P}$

- Conclusion: We could define $\mathrm{P}$ using one-tape Turing machines or using multi-tape Turing machines
- Either way, we get the exact same set of languages
- Another example: The “word RAM” model

# Word RAM model (RAM = Random Access Machine)

- (This model will not be on homework exercises or exams)
- A word RAM program consists of a list of instructions
- First few instruction types:
  - $R_i \leftarrow R_j$ or $R_i \leftarrow c$ where $i, j, c \in \mathbb{N}$
  - $R_i \leftarrow R_j$ op $R_k$ where op $\in \{$ `+, -, *, /, %, ==, <, >, &&, ||, &, |, ^, <<, >>` $\}$
  - IF $R_i$ GOTO $k$
  - ACCEPT or REJECT

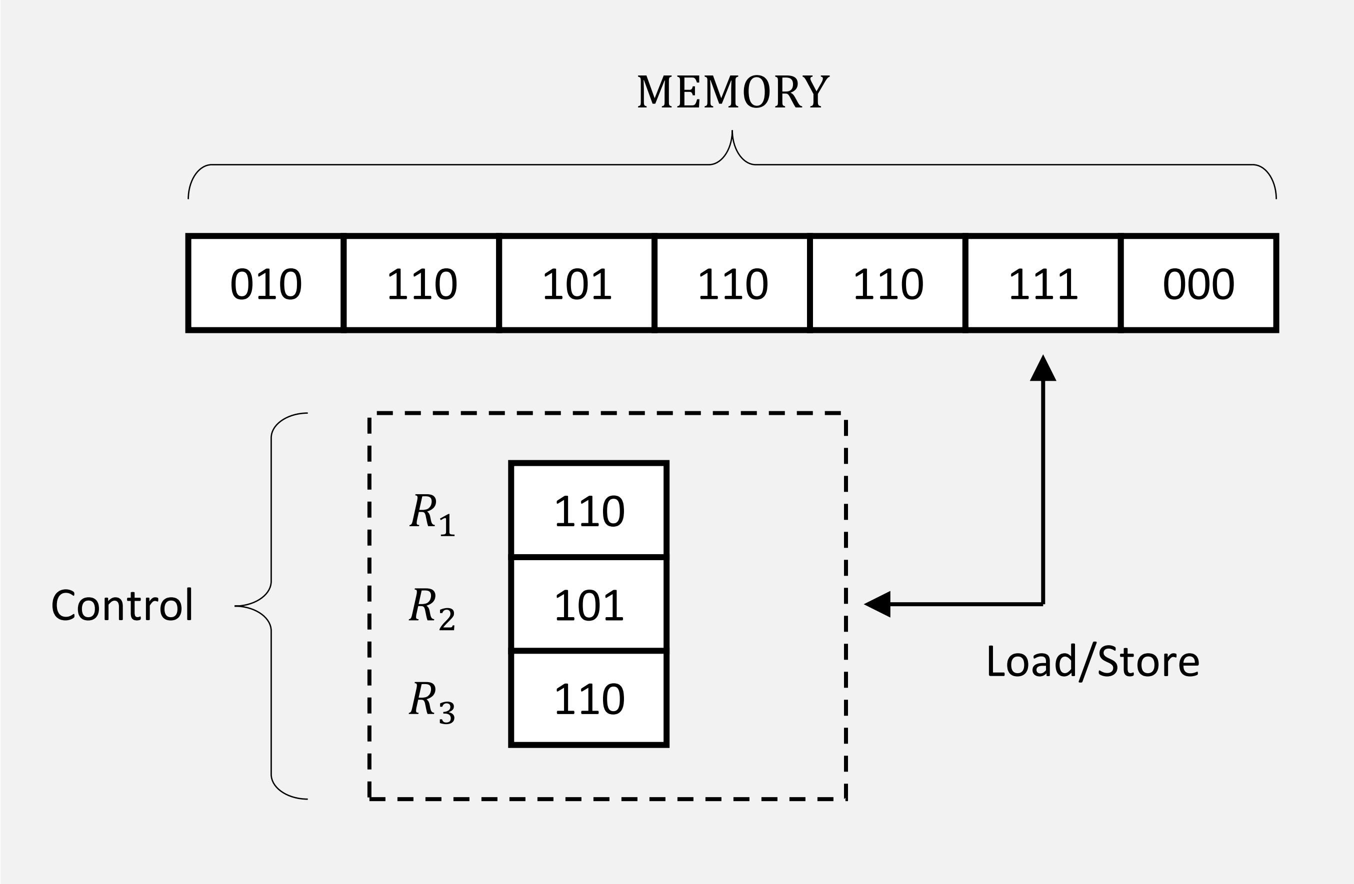
$R_i$ is a "global variable" of type `unsigned int`

(The details are not completely standardized. This is just one reasonable version of the model)

# Word RAM model

- Each $R_i$ holds a $k$-bit "word" representing a number in $\{0, 1, \dots, 2^k - 1\}$
- $k$ is called the "word size"
- In practice, maybe $k = 64$
- In theory, we think of $k$ as "large enough" and growing with $n$
- Operations on words take $O(1)$ time, unlike TM model!

# Word RAM model




- There is also a large memory (an array of words)
- Instructions:
  - $R_i \leftarrow \mathrm{MEMORY}[R_j]$
  - $\mathrm{MEMORY}[R_i] \leftarrow R_j$
- Instantly access any desired location in memory, unlike the TM model!

The version of the word RAM model described here is based on the lecture notes for CS 1200 at Harvard: https://harvard-cs-1200.github.io/cs1200/

# Word RAM model

- Let the input be $w \in \{0,1\}^n$
- Initially, $R_0 = n$ and $\mathrm{MEMORY}$ has $n$ cells, with $\mathrm{MEMORY}[i] = w_i$
- A special instruction "$\mathrm{MALLOC}$" extends $\mathrm{MEMORY}$, creating one new cell
- If $n \geq 2^k$, or if $c \geq 2^k$ for some constant $c$ in the program, or if $\mathrm{MEMORY}$ ever has more than $2^k$ cells, then the program crashes
- Reading to/writing from a nonexistent $\mathrm{MEMORY}$ cell does nothing

# Word RAM model

The version of the word RAM model described here is based on the lecture notes for CS 1200 at Harvard: https://harvard-cs-1200.github.io/cs1200/

- Let $Y \subseteq \{0,1\}^*$, let $P$ be a word RAM program, and let $T: \mathbb{N} \to \mathbb{N}$
- We say that $P$ decides $Y$ within time $T$ if:
  - For every $w \in Y$, for every $k \in \mathbb{N}$, if we run $P$ on input $w$ with word size $k$, then $P$ crashes or accepts within $T(|w|)$ steps
  - For every $w \notin Y$, for every $k \in \mathbb{N}$, if we run $P$ on input $w$ with word size $k$, then $P$ crashes or rejects within $T(|w|)$ steps
  - For every $w \in \{0,1\}^*$, there exists $k \in \mathbb{N}$ such that if we run $P$ on input $w$ with word size $k$, then $P$ halts without crashing.

# Word RAM model

- Word RAM time complexity closely matches time complexity "in practice" on ordinary computers
- Some version of the word RAM model is typically assumed (implicitly or explicitly) in algorithms courses and the computing industry

# Robustness of $\mathrm{P}$

- Let $Y \subseteq \{0, 1\}^*$

**Theorem:** If there is a word RAM program that decides $Y$ in time $\mathrm{poly}(n)$, then there is a Turing machine that decides $Y$ in time $\mathrm{poly}(n)$.

- Proof omitted