CMSC 28100

Introduction to Complexity Theory

Autumn 2025
Instructor: William Hoza

Asymptotic notation

- Let $T, f: \mathbb{N} \rightarrow [0, \infty)$
- Roughly:
  - T is $O(f)$ if $T(n) \leq C \cdot f(n)$ for some large constant C
  - T is $\Omega(f)$ if $T(n) \geq c \cdot f(n)$ for some small constant c
  - T is $o(f)$ if $T(n) \leq c \cdot f(n)$ for every small constant c
  - T is $\omega(f)$ if $T(n) \geq C \cdot f(n)$ for every large constant C

Exponential vs. polynomial

- Proved last time: For every constant $k \in \mathbb{N}$, we have $n^k = o(2^n)$
- We say $T(n)$ is $\mathrm{poly}(n)$ if there is some constant k such that $T(n)$ is $O(n^k)$

Big-Θ

- Let $T, f : \mathbb{N} \to [0, \infty)$ be any two functions
- We say that T is $\Theta(f)$ if T is $O(f)$ and T is $\Omega(f)$
- Example: $0.1n^2 + 14$ is $\Theta(n^2)$ and $\Theta(n^2 + 2n^{1.4})$, but not $\Theta(n)$

Let $T(n) = 2^{3n+4}$. Which of the following statements is <u>false</u>?

A: $T(n)$ is $\Omega(2^n)$

B: $T(n)$ is $2^{\Theta(n)}$

C: $T(n)$ is $\Theta(2^{3n})$

D: $T(n)$ is $O(2^n)$

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

Summary of asymptotic notation

Notation	In words	Analogy
T is $o(f)$	$T(n)$ grows more slowly than $f(n)$	$<$
T is $O(f)$	$T(n)$ is at most $C \cdot f(n)$	$\leq$
T is $\Theta(f)$	$T(n)$ and $f(n)$ grow at the same rate	$=$
T is $\Omega(f)$	$T(n)$ is at least $c \cdot f(n)$	$\geq$
T is $\omega(f)$	$T(n)$ grows more quickly than $f(n)$	$>$

Note: Big-O is not just for time complexity!

- We can use asymptotic notation (big-O, etc.) any time we are trying to understand some kind of "scaling behavior"
- For example, let G be a simple undirected graph with N vertices
  - G has $O(N^2)$ edges
  - If G is connected, then G has $\Omega(N)$ edges
- Admittedly, we are especially interested in time complexity…

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 one-tape 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 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:**

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

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

- "Polynomial time"

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"
- Is this a good model? What about multi-tape Turing machines?

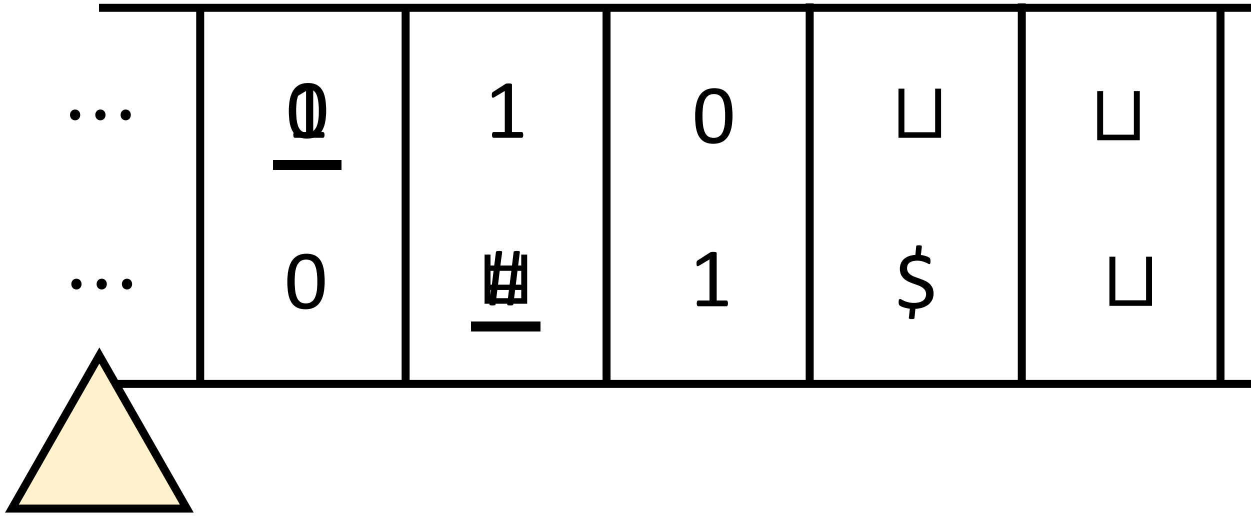
Multi-tape Turing machines, revisited

- 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 1 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 P

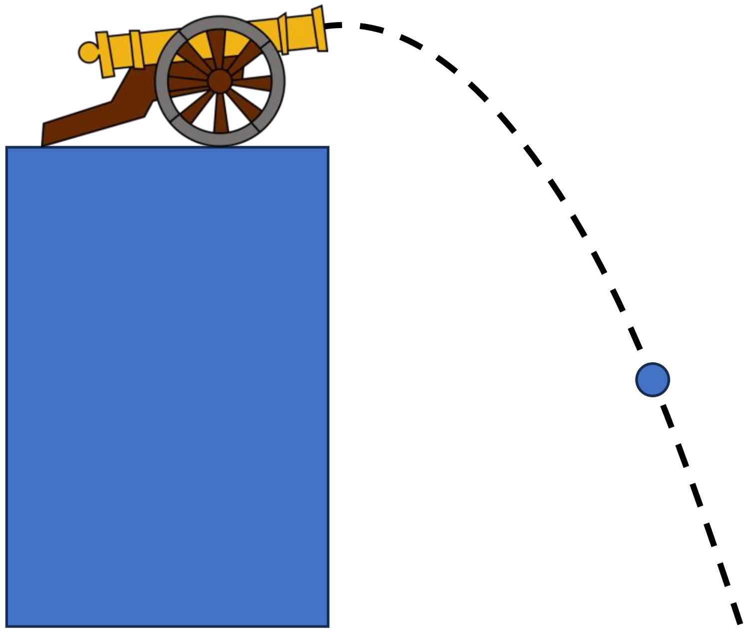
- Conclusion: We could define P using one-tape Turing machines or using multi-tape Turing machines
- Either way, we get the exact same set of languages

Theory vs. practice

- Disclaimer: P is not a perfect model of tractability
- Even if some problem is technically in P, it might not be "solvable in practice"
- Even if some problem is technically not in P, it might be "solvable in practice"

Analogy: Gravity

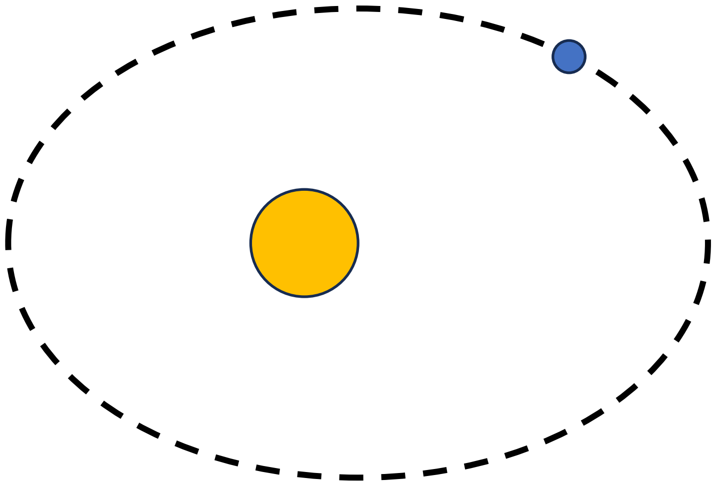
- Physics 101: "Gravity is a constant downward force of 9.8 N/kg"



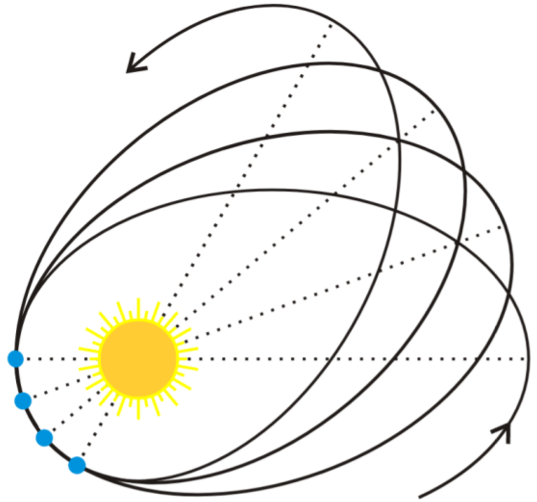
- Physics 102: Newton's Law of Gravitation:

$$F = G \cdot \frac{m_1 \cdot m_2}{r^2}$$

  - Better, but still not perfect!



Analogy: Gravity



- Newton's Law of Gravitation does not correctly predict Mercury's motion around the sun!
- "…all models are wrong, but some are useful." –George Box
- The complexity class P does not 100% align with the set of problems that are solvable in practice…
- But the alignment is pretty good, and studying P will absolutely give us real insights into the nature of computation

Which problems

can be solved

through computation?

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

Example: 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.
  - Time complexity is $\text{poly}(K)$, which is "pseudo-polynomial time"
  - "Polynomial time" means time complexity $\text{poly}(n)$, where $n = |\langle K \rangle| \approx \log K$!
- The theorem is true, but the proof is beyond the scope of this course

Pseudo-polynomial time

- Suppose $Y = \{\langle x, k \rangle : k \in \mathbb{N} \text{ and (something)}\}$
- "Polynomial time" means $\text{poly}(n)$ time where $n \approx |x| + \log k$
- "Pseudo-polynomial time" means $\text{poly}(n')$ time where $n' = |x| + k$
- $Y' = \left\{\left\langle x, 1^k \right\rangle : k \in \mathbb{N} \text{ and (something)}\right\}$
- If it's reasonable to assume that k is small, then pseudo-polynomial time might be good enough
- Interesting example: The knapsack problem

The knapsack problem

- Given: Positive integers $w_1, \ldots, w_k, v_1, \ldots, v_k, W, V$
- Question: Is there a set $S \subseteq \{1, 2, \ldots, k\}$ such that $\sum_{i \in S} w_i \leq W$ and $\sum_{i \in S} v_i \geq V$?
  - Interpretation: There are k items
  - Item i is worth v_i dollars, and it weighs w_i pounds
  - We want to collect items worth V dollars, but our knapsack can only hold W pounds

The knapsack problem

- There is no known polynomial-time algorithm that decides KNAPSACK
- However, there is a pseudo-polynomial-time algorithm!

The knapsack problem

- $\text{KNAPSACK} = \{\langle w_1, \dots, w_k, v_1, \dots, v_k, W, V\rangle : \text{there exists } S \subseteq \{1, 2, \dots, k\}$
  $\text{such that } \Sigma_{i \in S}\, w_i \leq W \text{ and } \Sigma_{i \in S}\, v_i \geq V\}$

Conjecture: $\text{KNAPSACK} \notin \text{P}$

The knapsack problem

- UNARY-VAL-KNAPSACK $= \{\langle w_1, \ldots, w_k, 1^{v_1}, \ldots, 1^{v_k}, W, 1^V \rangle$: there exists $S \subseteq \{1, 2, \ldots, k\}$ such that $\Sigma_{i \in S}\, w_i \leq W$ and $\Sigma_{i \in S}\, v_i \geq V\}$

Theorem: UNARY-VAL-KNAPSACK $\in$ P