Complexity Theory I

Chanathip Namprempre

Department of Computer Science
Reed College

Outline

Outline

Complexity Theory

Complexity theory focuses on decidable problems.

We talk about time complexity first.

Some problems take little time to solve. Some require much more time.

We are interested in categorizing decidable problems into different groups according to the amount of time required to solve them.

A simple example

Consider the following language:

$$L = L(0^*1^*)$$

Certainly, L is decidable. (In fact, it can be recognized by a DFA.)

Questions

- Can you write a deterministic TM recognizing L?
- How much time does your TM take?

Try it on these inputs: 01, 10000, 0000011

Clearly, the running time of your TM varies depending on the length of the input.

Question

Let n be the input length, how much time does your TM take?

Running time of a TM

Definition

Let M be a deterministic TM that halts on all inputs. The running time or time complexity of M is the function $f : \mathrm{N} \to \mathrm{N}$, where $f(n)$ is the maximum number of steps that M uses on any input of length n.

Let $f : \mathrm{N} \to \mathrm{N}$, and let M be a TM.
The following statements are equivalent:

- $f(n)$ is the running time of M
- M runs in time $f(n)$
- M is an $f(n)$ time TM

Asymptotic Analysis

In this course, we only care about the running time of a TM on large inputs. This means that we ignore constant terms.

So we use asymptotic analysis when analyzing the running time of our TMs.

In this type of analysis, we use asymptotic notation.

Big-O Notation

Definition

Let f and g be two functions from N to R^+. We say that $f(n) = O(g(n))$ if there exist positive integers c and n_0 so that, for every integer $n \geq n_0$,

$$f(n) \leq cg(n) .$$

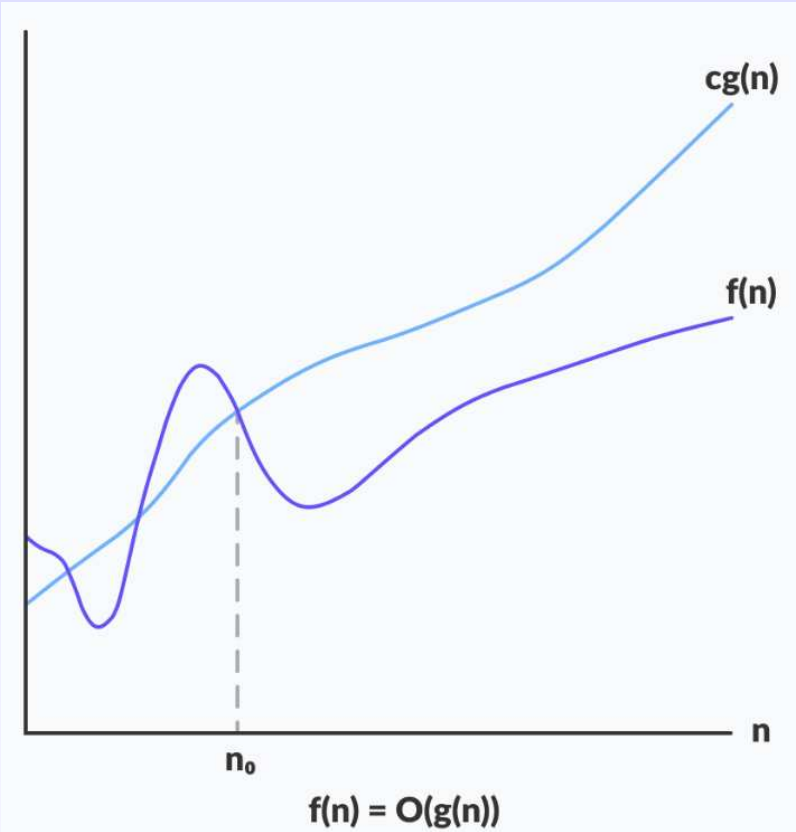
Let $f, g : N \to R^+$. The following statements are equivalent:

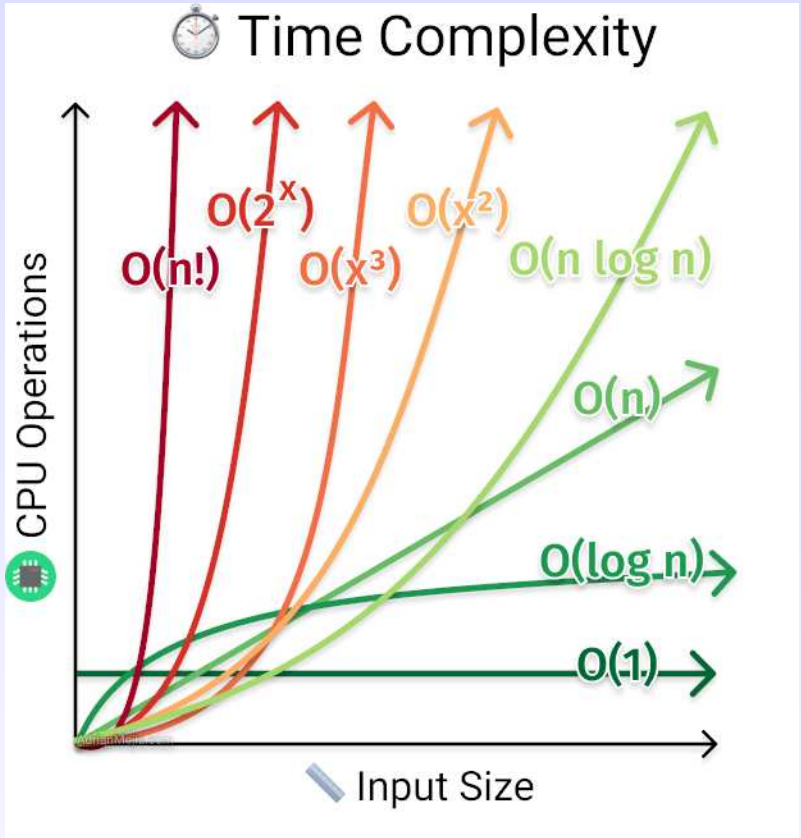
- $f(n) = O(g(n))$
- $g(n)$ is an (asymptotic) upper bound for $f(n)$

Intuitively, $f(n) = O(g(n))$ means that f is no more than g if we disregard differences up to a constant factor.

Examples

$$5n^3 + 2n^2 + 6 = O(n^3) \; ; \; 5n^3 + 2n^2 + 6 = O(n^4) \; ; \; 5n^3 + 2n^2 + 6 \neq O(n^2)$$





Sources:

https://www.programiz.com/dsa/asymptotic-notations

https://thecodingbay.com/learn-everything-about-big-o-notation/

Small-O Notation

Definition

Let f and g be two functions from N to R^+. We say that $f(n) = o(g(n))$ if there, for any real number $c > 0$, there exists a positive integer n_0 so that, for every integer $n \geq n_0$,

$$f(n) < cg(n) .$$

Intuitively, $f(n) = O(g(n))$ means that f is strictly less than g if we disregard differences up to a constant factor.

Examples

$$n \log n = o(n^2) \; ; \; \sqrt{n} = o(n) \; ; \; n \neq o(n)$$

Time Complexity Class: Definition

We can group problems by the time it takes a deterministic TM to solve them.

Definition

Let $t : \mathrm{N} \to \mathrm{N}$ be a function. We define the time complexity class $\mathrm{TIME}(t(n))$ as follows:

$$\mathrm{TIME}(t(n)) = \{L \mid L \text{ is a language decided by an } O(t(n)) \text{ time TM } \}.$$

Example

$$L(0^*1^*) \in \mathrm{TIME}(n)$$

Can it be the case that $L \in \mathrm{TIME}(\log n)$?

[This is like asking whether it was possible to construct a TM deciding L in time $O(\log n)$.]

Time complexity class: More examples

Example

$$\{0^n 1^n \mid n \geq 0\} \in \mathsf{TIME}(n)$$

Can you write a deterministic TM M deciding L?

- Does your machine M use 1 tape? What is its running time?
- Does your machine M use 2 tapes? What is its running time?

Bottom line

If you change the computation model, you can often change the running time.

Running time of 1-tape TM compared to multitape TM

Usually, we can make things faster if we use more tapes.

BUT the improvement will always be within a polynomial factor.

Here is the reason:

Theorem

Let $t(n) : \mathrm{N} \to \mathrm{N}$ *where* $t(n) \geq n$. *Then,*

every k-tape TM running in time $t(n)$ *can be simulated by a single-tape, deterministic TM running in time* $O(k \cdot t^2(n))$.

Deterministic models are polynomially equivalent

In complexity theory, polynomial differences in running time are considered to be small whereas exponential differences are considered to be large.

Examples: Compare n^3 and 2^n. Try with $n = 1000$.

Fact

All reasonable deterministic computational models are polynomially equivalent.

Translation: Any one of them can simulate another with only a polynomial increase in running time.

When we focus on problems that are solvable in polynomial time, our results do not depend on the model of computation being used.

Bottom line

We can focus on fundamental properties of computation rather than the exact model in which the computation is performed.

Running time of nondeterministic TM

Definition

Let N be a nondeterministic TM that is a decider. The running time of N is the function $f : \mathrm{N} \to \mathrm{N}$ where $f(n)$ is the maximum number of steps that N uses on any branch of its computation on any input of length n.

Example

$$\{ww \mid w \in \{0,1\}^*\}$$

Can you write an NTM to solve this problem? What is its running time?

Nondeterministic TM can give us upto exponential speedup

Theorem

Let $t(n) : \mathrm{N} \to \mathrm{N}$ *where* $t(n) \geq n$. *Then,*

every NTM running in time $t(n)$ *can be simulated by a single-tape, deterministic TM running in time* $2^{O(t(n))}$.

Example

$$\{ww \mid w \in \{0,1\}^*\}$$

How fast could your NTM solve this problem? What about a single-tape TM simulating it?

Outline

The class P

Definition

P is the class of languages that are decidable in polynomial time on a deterministic single-tape TM. In other words,

$$\mathrm{P} = \bigcup_k \mathrm{TIME}((n^k)) \,.$$

We are interested in P because it is robust under composition.

[If a subroutine runs in polynomial-time and we call it polynomial number of times, we still run in polynomial time.]

Some languages in P

1. Let PATH $= \{\langle G, s, t\rangle \mid G$ is a directed graph that has a directed path from s to $t\}$.

$$\text{PATH} \in \text{P}$$

2. Let RELPRIME $= \{\langle x, y\rangle \mid x$ and y are relatively prime $\}$.

$$\text{RELPRIME} \in \text{P}$$

3. Let L be a context-free language.

$$\text{L} \in \text{P}$$

Intuition

In complexity theory, problems in P are considered "easy" to solve.

Closure properties of P

Theorem

Let L, L_1, L_2 be languages.

1. *If $L \in \mathrm{P}$, then $\overline{L} \in \mathrm{P}$.*
2. *If $L_1, L_2 \in \mathrm{P}$, then $L_1 \cup L_2 \in \mathrm{P}$.*
3. *If $L_1, L_2 \in \mathrm{P}$, then $L_1 \cap L_2 \in \mathrm{P}$.*