Time and Space Complexity of an Algorithm

f(n) → algorithm

- Time complexity of an algorithm depends on the number of instructions executed. This number is primarily dependent on the size of the program's input and the algorithm used.

~~machine~~ ~~programming~~ ~~Language~~

- The space needed by a program depends on:

Flop ← cpu, HPC

✓ Fixed part includes space needed for storing instructions, constants, variables, and structured variables.

GPU → TFlop

int[32]

✓ Variable part includes space needed for recursion stack, and for structured variables that are allocated space dynamically during the run-time of the program.

linked list

Asymptotic Analysis

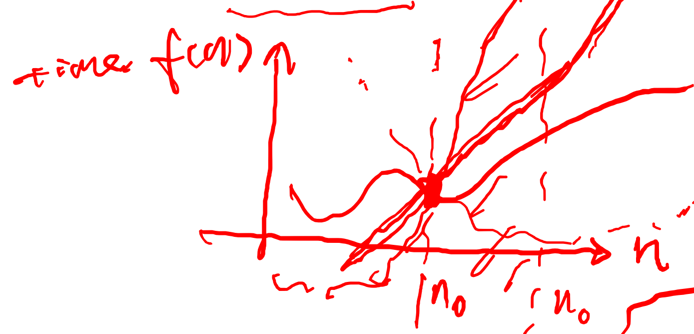
$\lim_{n \to \infty} f(n)$

- A method of describing the limiting behavior of a function or algorithm when the input size or value approaches infinity or a certain point. It is often used to measure the efficiency of algorithms in terms of time complexity, by ignoring the less significant factors and focusing on the dominant terms.

Big O Notation

→ upper bound $n \to \infty$

- If f(n) and g(n) are functions defined on positive integer number n, then



set

$$f(n) = O(g(n))$$

- That is, f of n is big O of g of n if and only if there exists positive constants c and n_0, such that

set

$$f(n) \le cg(n) \quad \text{for all } n \ge n_0$$

$f(n) \le 4n$ $O(4n)$ $O(n)$

- This means that for large amounts of data, f(n) will grow no more than a constant factor than g(n). Hence, g provides an upper bound.

Big O Notation

- O(g(n)) is the SET of ALL functions f(n) that satisfies the above conditions
- Big-O notation, where the "O" stands for "order of", is concerned with what happens for very large values of n.
- When expressing complexity using Big O notation, constant multipliers are ignored. So a O(4n) algorithm is equivalent to O(n), which is how it should be written.

Big O → worst case,

EXAMPLE: Inventory

- Suppose an algorithm for processing a retail store's inventory takes:
  - 10,000 milliseconds to read the initial inventory from disk, and then
  - 10 milliseconds to process each transaction (items acquired or sold).

n_0

$g(n)$, c, n_0 ↓

→ all $n \geq n_0$

$f(n) = 10000 + 10 \cdot n$

Processing n transactions takes (10,000 + 10 n) ms.

$f(n) \leq c \cdot g(n)$

assume $g(n) = n$

$$f(n) = 10{,}000 + 10n$$

- If we assume $g(n) = n$, we can choose $c = 20$, $n_0 = 1000$
- So $f(n) = O(g(n))$

$f(n) = 10000 + 10 \cdot n$

assume $g(n) = n$

find one c and one n_0

$$10000 + 10 \cdot n \le c \cdot n \quad (1)$$

~~as~~ chose $c = 20$

plug into (1)

$$10000 + 10 \cdot n \le 20 \cdot n,$$

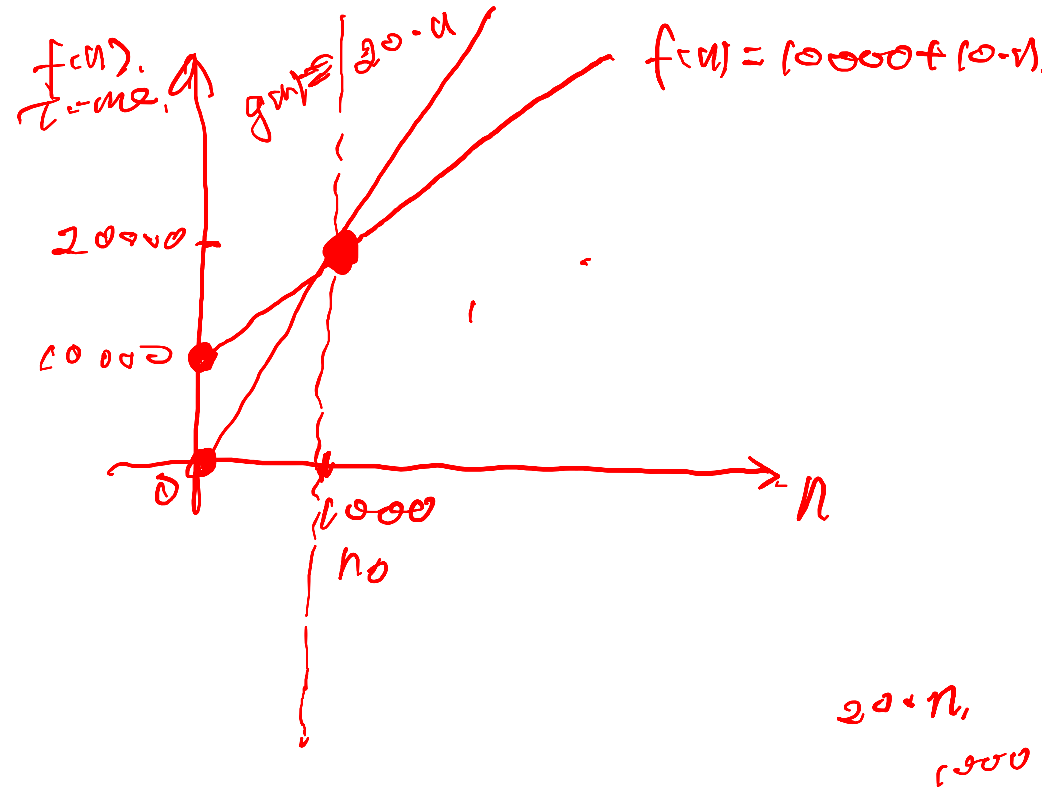
$$10000 \le 10n$$

$$n \ge \underbrace{1000}_{= n_0}$$

$c = 20$

$n_0 = 1000$

$g(n) = n.$

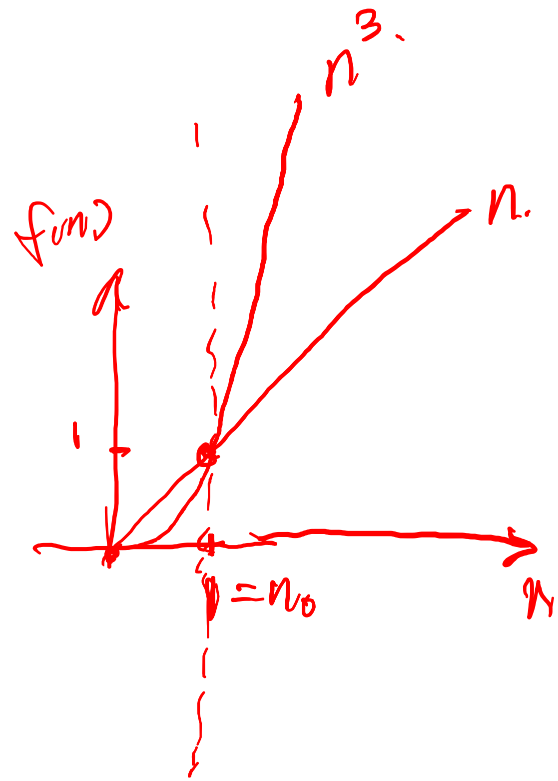


EXAMPLES

- Is 1,000,000 n in O(n)? $\leq c$, $c = 2,000,000$, $n_0 = 0$, $\leq g(n)$, Yes.
- Is n in O(n³)? $= g(n)$ Yes. c, n_0, $c = 1$

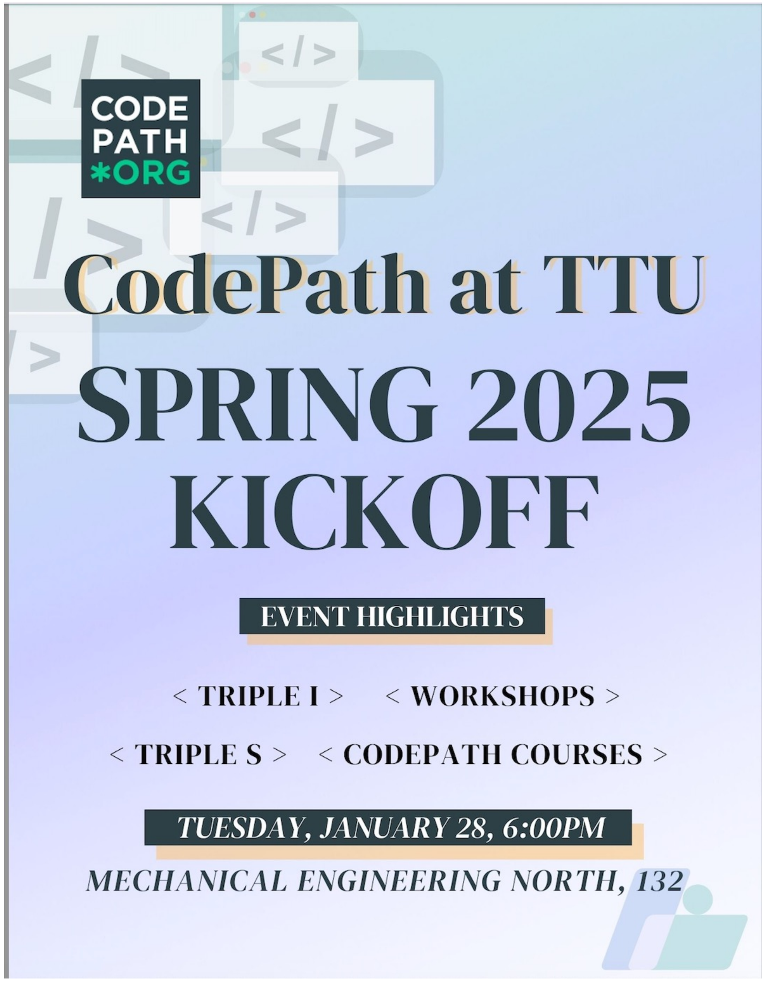
$$n \leq 1 \cdot n^3$$
$$1 \leq n^2$$
$$n \geq 1 = n_0$$

- Is $n^3 + n^2 + n$ in $O(n^3)$?
- Is n^2 in $O(n)$?
- Is e^{3n} in $O(e^n)$?
- Is 10^n in $O(2^n)$?



CodePath Event

- **Date & Time:** January 28th at 6 PM
- **Location:** MEN 132
- **About CodePath:**
  CodePath focuses on equipping students with the essential tools, resources, and mentorship required to thrive in the technology industry.
- **Upcoming Events:**
  - **Triple S:** Students Software Spotlight
  - **Triple I:** Innovation Ignition Initiative
- **Workshops:**
  - Data Structures and Algorithms
  - Web Development
- **Contact:** tylbowen@ttu.edu



G-SWEP Event

- Google + Basta's coding mentorship program – G-SWEP
  - G-SWEP is on experimental learning program where historically underrepresented groups master the technical interview and prepare for great careers in tech through 1:1 support with google software engineers.
  - The application closes Jan. 27.