파이썬으로 배우는 데이터 구조

한동대학교
전산전자공학부
김영섭 교수

학습 목표

Big-O 표기법이 무엇인지 알고 직접 계산할 수 있다

Data Structures in Python
Chapter 2 - 2

- Performance Analysis
- **Big-O Notation**
- Big-O Properties
- Growth Rates
- Growth Rates Examples

그러므로 나의 사랑하는 자들아 너희가 나 있을 때 뿐 아니라 더욱 지금 나 없을 때에도 항상 복종하여 두렵고 떨림으로 너희 구원을 이루라 (Continue to work out your salvation with fear and trembling.) 빌2:12

나는 인애를 원하고 제사를 원하지 아니하며 번제보다 하나님을 아는 것을 원하노라 (호6:6)
하나님은 모든 사람이 구원을 받으며 진리를 아는데에 이르기를 원하시느니라 (딤전2:4)

그런즉 너희가 먹든지 마시든지 무엇을 하든지 다 하나님의 영광을 위하여 하라 (고전10:31)

Agenda & Reading

- **Big-O Notation**
  - **Asymptotic Analysis**
- Big-O Properties
  - Calculating Big-O

- References:
  - Textbook: Problem Solving with Algorithms and Data Structures
    - Chapter 3. Analysis
  - Textbook: www.github.idebtor/DSpy
    - Chapter 2.1 ~ 3

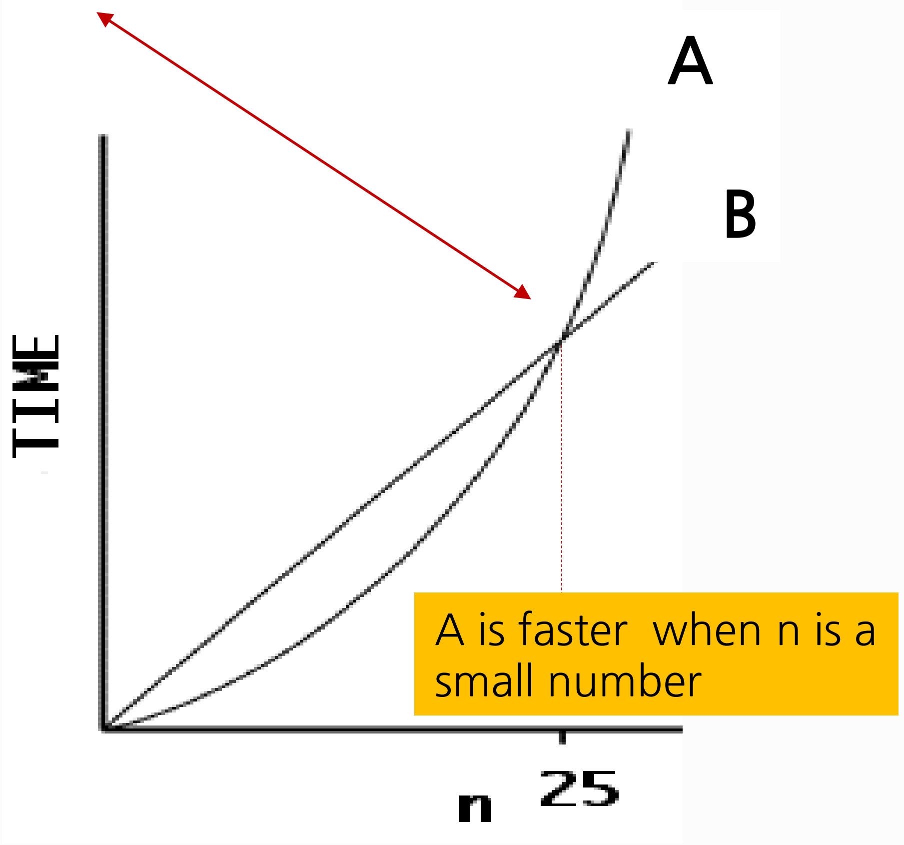
Review: Counting Operations - Growth Rate Function - A or B?

- Consider the following two algorithms:
  - Algorithm A: $\frac{n^2}{5}$
  - Algorithm B: $5 * n$

n	5	10	15	20	24	25	26	30
A	5	20	45	80	115	125	135	180
B	25	50	75	100	120	125	130	150

- If n is 10^6 ,
  - Algorithm A's time requirement is
    - $\frac{n^2}{5} = \frac{10^{12}}{5} = \boxed{2 * 10^{11}}$
  - Algorithm B's time requirement is
    - $5 * n = \boxed{5 * 10^6}$

- What does the **growth rate** tell us about the running time of the program?



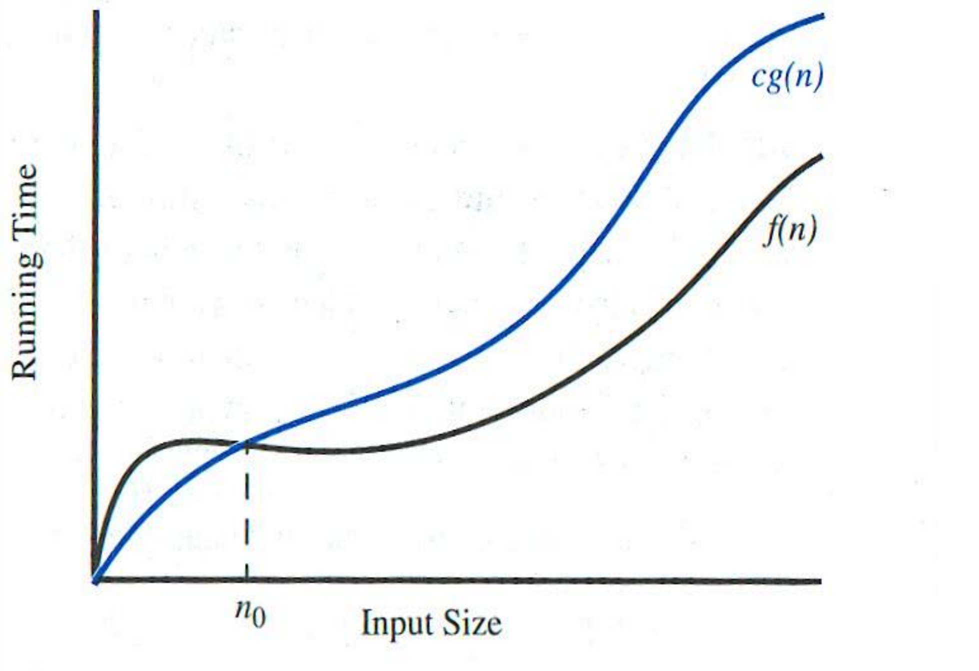
3 Big-O Definition

- Let $f(n)$ and $g(n)$ be functions that map non-negative integers to real numbers. We say that $\boldsymbol{f(n)}$ **is** $\boldsymbol{O(g(n))}$ if there is a real constant c, where $c > 0$ and an integer constant n, where $n \geq n_0$ such that $f(n) \leq c * g(n)$ for every integer $n \geq n_0$.

$$\boldsymbol{f(n) \leq c\, g(n)}, \qquad for\, n \geq \boldsymbol{n_0}$$

Then it is pronounced as " $f(n)$ $\boldsymbol{is}$ $big\ Oh\ of$ $g(n)$ or $\boldsymbol{f(n) = O(g(n))}$ "

- $f(n)$ describe the actual time of the program
- $g(n)$ is a much simpler function than $f(n)$
- With assumptions and approximations, we can use $g(n)$ to describe the complexity i.e., $\boldsymbol{O(g(n))}$



Big-O Notation is a mathematical formula that best describes an algorithm's performance.

3 Big-O Notation

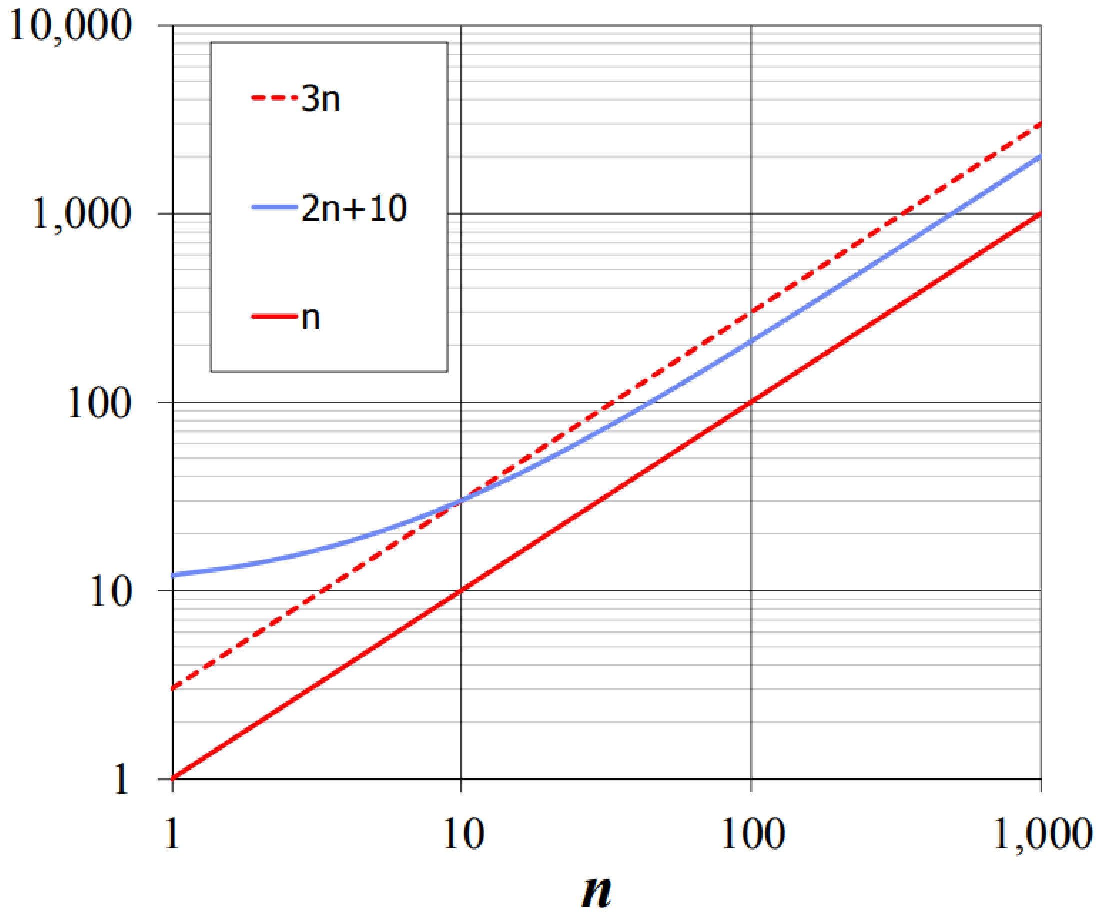
- We use Big-O notation (capital letter O) to specify the order of complexity of an algorithm.
  - $e.g.,\ O(n^2),\ O(n^3),\ O(n\ )$
  - If a problem of size n requires time that is directly proportional to n, the problem is $O(n)$ - that is, order n.
  - If the time requirement is directly proportional to n^2, the problem is $O(n^2)$, etc.

3 Big-O Examples

- Given functions $f(n)$ and $g(n)$, we say that $\boldsymbol{f(n)}$ **is** $\boldsymbol{O(g(n))}$ if there are positive constants, c, and n_0 such that $f(n) \leq c * g(n)$ for every integer $n \geq n_0$.

- Example:
  T(n) = 2n + 10
  T(n) is O(n)

- Question:

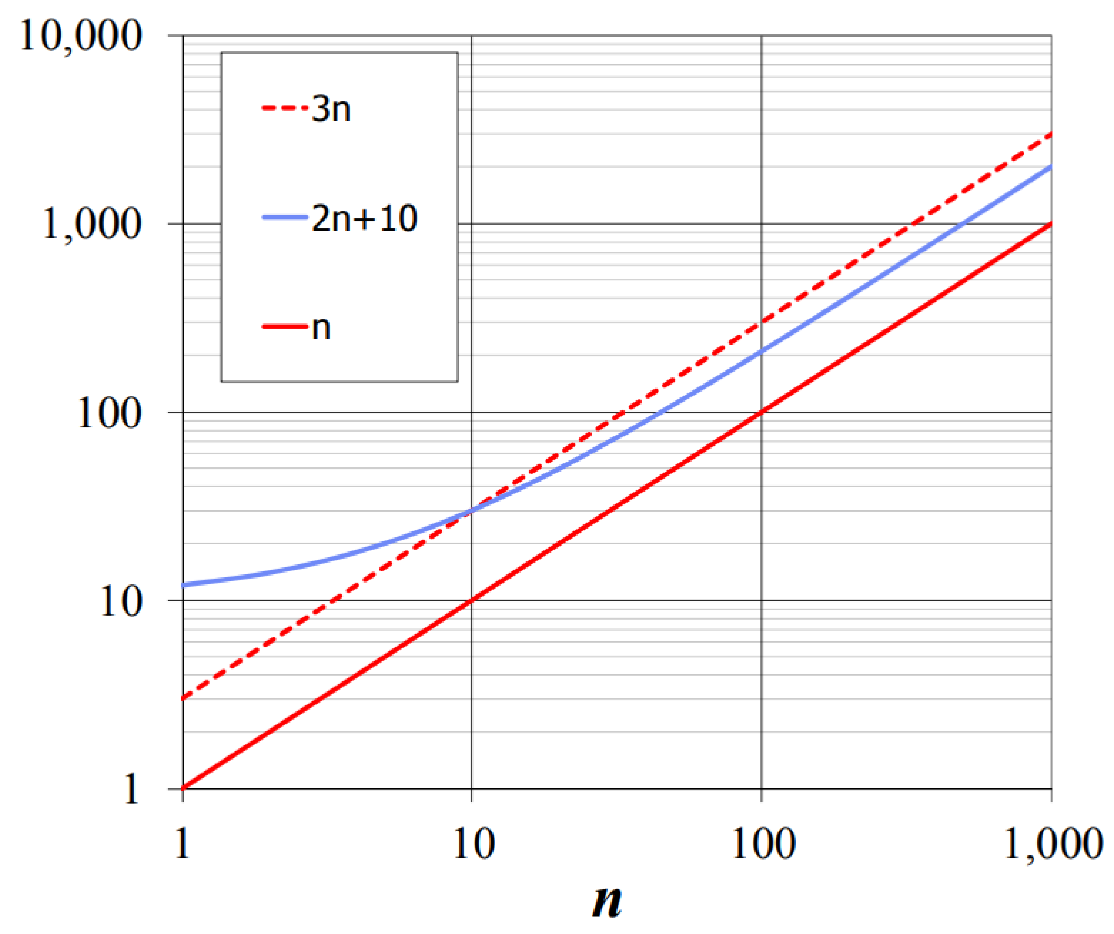


3 Big-O Examples

- Given functions $f(n)$ and $g(n)$, we say that $\boldsymbol{f(n)}$ **is** $\boldsymbol{O(g(n))}$ if there are positive constants, c, and n_0 such that $f(n) \le c * g(n)$ for every integer $n \ge n_0$.

- Example:
  T(n) = 2n + 10
  T(n) is O(n)

- Question:
  - n_0
  - c
  - g(n)
  - $f(n) \le c * g(n)$
  - $\boldsymbol{f(n)}$ **is** $\boldsymbol{O(g(n))}$

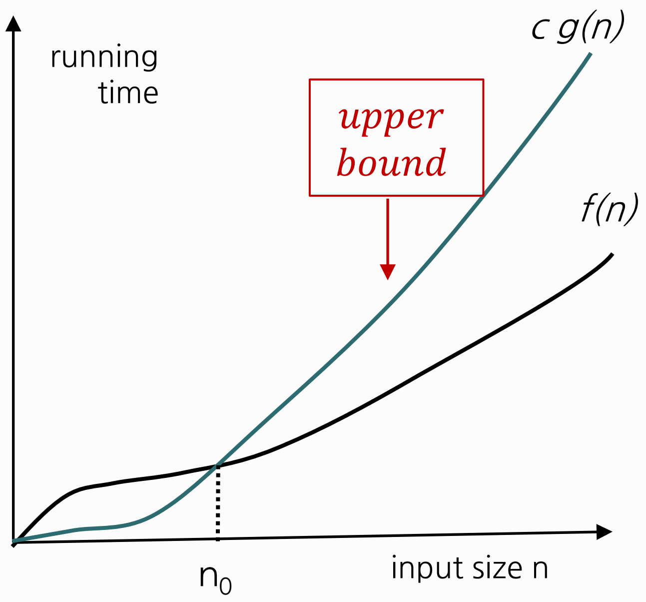


3 Big-O Examples

- Find c and n_0 to justify that the function $7n + 5$ is $O(n)$.

We must find c and n_0 such that

$$7n + 5 \leq c\,n \qquad for\ n \geq n_0$$



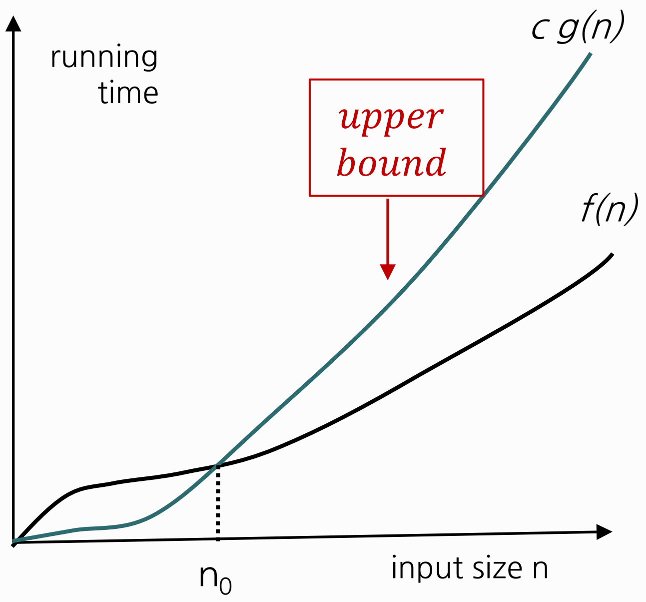
3 Big-O Examples

- Find $\boldsymbol{c}$ and $\boldsymbol{n_0}$ to justify that the function $7\boldsymbol{n} + 5$ **is** $\boldsymbol{O(n)}$.

We must find $\boldsymbol{c}$ and $\boldsymbol{n_0}$ such that

$$7n + 5 \le c\,n \qquad for\ n \ge \boldsymbol{n_0}$$
$$7n + 5 \le 7\,n + n$$
$$7n + 5 \le 8\,n \qquad for\ n \ge \boldsymbol{n_0} = 5$$

Therefore, $7n + 5 \le c\,n$ for $c = 8$ and $\boldsymbol{n_0} = 5$, $g(n) = n$ and $O(n)$



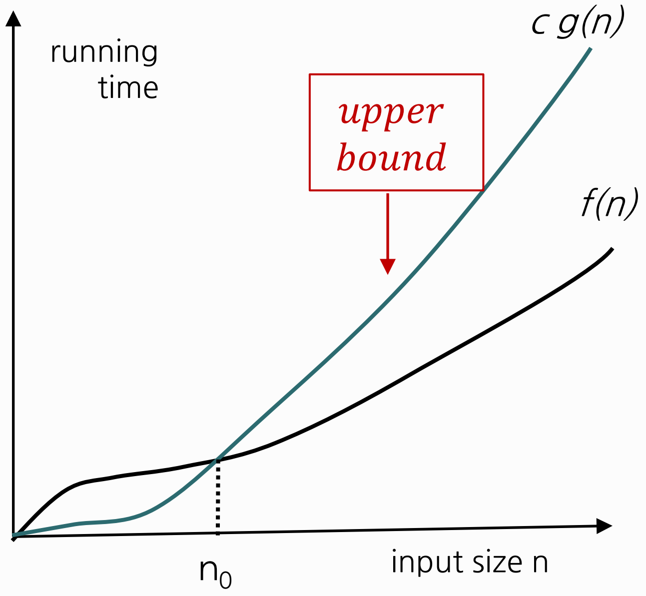
3 Big-O Examples

- Find c and n_0 to justify that the function $7n + 5$ is $O(n)$.

We must find c and n_0 such that

$$7n + 5 \leq c\,n \qquad for\ n \geq n_0$$
$$7n + 5 \leq 7\,n + n$$
$$7n + 5 \leq 8\,n \qquad for\ n \geq n_0 = 5$$

Therefore, $7n + 5 \leq c\,n$ for $c = 8$ and $n_0 = 5$, **f(n) is O(n)**



$7n + 5 \leq c\,n \quad for\ n \geq n_0$
$7n + 5 \leq 12\,n \quad for\ n \geq n_0 = 1$
Therefore, $7n + 5 \leq c\,n$ for $c = 12$ and $n_0 = 1$
$g(n) = n$, $f(n)$ is $O(n)$

3 Big-O Examples

- Find $\boldsymbol{c}$ and $\boldsymbol{n_0}$ to justify that the function $\boldsymbol{f(n) = 27n^2 + 16n}$ is $\boldsymbol{O(n^2)}$.

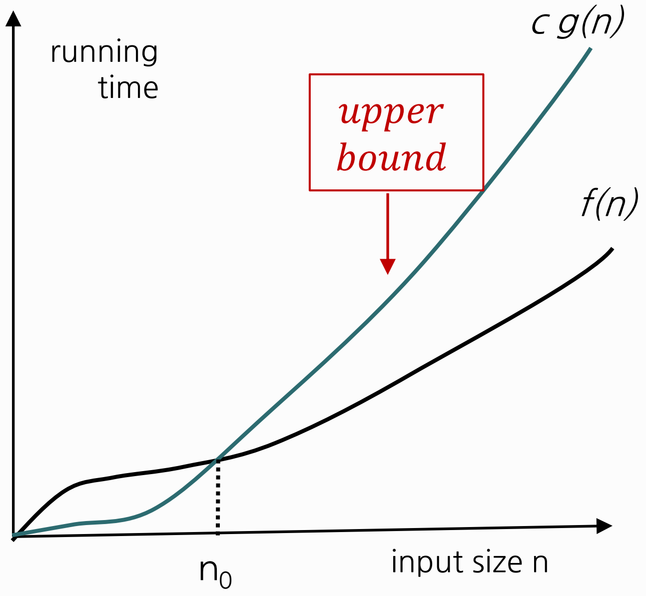
We must find $\boldsymbol{c}$ and $\boldsymbol{n_0}$ such that

For $\boldsymbol{16n \le n^2}$

$$\boldsymbol{27n^2 + 16n \le 27n^2 + n^2}$$

$$\boldsymbol{27n^2 + 16n \le 28n^2} \qquad for\ n \ge \boldsymbol{n_0} = 16$$

Hence, c = 28 and $\boldsymbol{n_0}$ = 16, Therefore, $\boldsymbol{g(n) = n^2, f(n)\ is\ O(n^2)}$.



$\boldsymbol{27n^2 + 16n}$ is $\boldsymbol{O(n^2)}$, we must find $\boldsymbol{c}$ and $\boldsymbol{n_0}$ such that

$$\boldsymbol{27n^2 + 16n \le 43n^2}$$

$$\boldsymbol{27n^2 + 16n \le 43n^2} \qquad for\ n \ge \boldsymbol{n_0} = 1$$

Hence, c = 43 and $\boldsymbol{n_0}$ = 1, Therefore, $\boldsymbol{g(n) = n^2, f(n)\ is\ O(n^2)}$.

3 Big-O Examples

$f(n) \le c * g(n)$ for every integer $n \ge n_0$

- Suppose an algorithm requires
  - T(n) = 7n-2 operations to solve a problem of size n

$$7n-2 \le 7 * n \text{ for all } n_0 \ge 1$$
$$\text{i.e., } c = 7,\ n_0 = 1$$

O(n)

  - T(n) = n^2 - 3 * n + 10 operations to solve a problem of size n

$$n^2 - 3 * n + 10 < 3 * n^2 \text{ for all } n_0 \ge 2$$
$$\text{i.e., } c = 3,\ n_0 = 2$$

O(n^2)

  - T(n) = $3n^3 + 20n^2 + 5$ operations to solve a problem of size n

$$3n^3 + 20n^2 + 5 < 4 * n^3 \text{ for all } n_0 \ge 21$$
$$\text{i.e., } c = 4,\ n_0 = 21$$

O(n^3)

3 Big-O Examples

1) $3n + 2 \;=$

2) $3n + 3 \;=$

3) $100n + 6 =$

4) $10n^2 + 4n + 2 \;=$

5) $6 * 2^n + n^2 \;=$

6) $3n + 3 \;=$

7) $10n^2 + 4n + 2 \;=$

8) $3n + 2 \neq O(1)$ as $3n + 2$ is $\boldsymbol{not}$ $\leq c$ for any c and all $n, n \geq n_0$.

9) $10n^2 + 4n + 2 \;\neq O(n)$

Summary

- Big-O Notation is a mathematical formula that best describes **an algorithm's performance**.
- **Big-O notation** is often called the asymptotic notation **(점근적 표기법)** since it uses so-called the **asymptotic analysis (점근적 분석)** approach.
- Normally **we assume worst-case analysis**, unless told otherwise.
- In some cases, it may need to consider the best, worst and/or average performance of an algorithm

학습 정리

1) Big-O(빅 오)은 알고리즘의 수행능력을 잘 나타내는 수학적인 표기법이다

2) Big-O를 계산할 때 주어진 함수들에서 가장 근접한 함수를 찾는 것이 좋다

파이썬으로 배우는 데이터 구조

수고했습니다
곧 다음 시간에
다시 뵙겠습니다