🔷

시간 복잡도(Big O)

시간 복잡도(Time Compexity)

- 컴퓨터 프로그램의 **입력값과 연산 수행 시간의 상관관계**
- 효율적 알고리즘이란?
  - 입력값이 커짐에 따라 증가하는 시간의 비율을 최소화한 알고리즘
- Big O 표기법

Big O 표기법

시간 복잡도를 표기하는 방법

- **Big O : 상한 점근 (최악)**
- Big Ω: 하한 점근 (최선)
- Big θ: 그 둘의 평균

Big O 표기법의 종류

1. O(1)

- **상수 복잡도(constant complexity)**
  - 입력값이 증가하더라도 시간이 늘어나지 않음

```
def print_array(arr):
    print(arr[0])

def double_print_array(arr):
    print(arr[0])
    print(arr[0])
```

2. O(n)

- **선형 복잡도(linear complexity)**
  - 입력값과 시간이 같은 비율로 증가

```
def print_all_array(arr):
    for a in arr:
        print(a)
```

3. O(logn)

- **로그 복잡도 (logarithmic complexity)**
  - 입력값이 기하급수적으로 증가하는 동안 실행시간은 선형적으로 증가
  - ex) Binary Search

4. O(n^2)

- **이차 복잡도(quadratic complexity)**
  - 입력값이 증가함에 따라 시간이 n의 제곱수의 비율로 증가

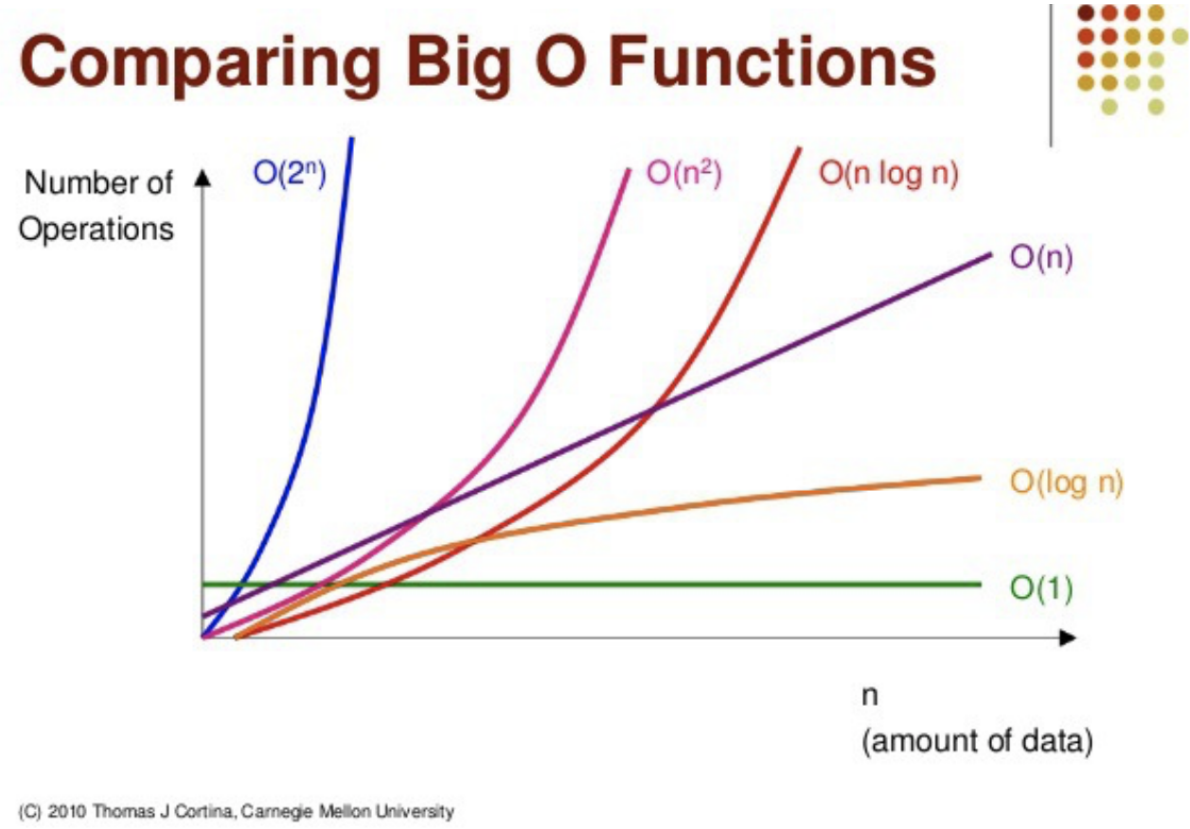
```
def print_twice(arr):
    for n in arr:
        for x in arr:
            print(x, n)
```

5. O(2^n)

- **기하급수적 복잡도(exponential complexity)**
  - 입력의 크기에 따라 시간이 기하급수적으로 증가

```
def fib(n):
    if n == 0:
        return 0
    elif n == 1 or n == 2:
        return 1
    else:
        return fib(n - 1) + fib(n - 2)
```

그래프를 통한 시간복잡도 간 비교



Reference

https://youtu.be/BEVnxbxBqi8

https://hanamon.kr/알고리즘-time-complexity-시간-복잡도/