시간 복잡도란?
- 입력값과 문제를 해결하는 데 걸리는 시간과의 상관관계를 말합니다.
입력값이 2배로 늘어났을 때 문제를 해결하는 데 걸리는 시간은 몇배로 늘어나는지를 보는 것이죠.
-입력값이 늘어나도 걸리는 시간이 덜 늘어나는 알고리즘이 좋은 알고리즘입니다.

각 줄이 실행되는 걸 1번의 연산이 된다.

```
for num in array:                # array 의 길이만큼 아래 연산이 실행
    for compare_num in array:    # array 의 길이만큼 아래 연산이 실행
        if num < compare_num:    # 비교 연산 1번 실행
            break
    else:
        return max_num
```

array의 길이 X array의 길이 X 비교 연산 1번 만큼의 시간 필요.
(array(입력값)의 길이는 보통 "N"이라고 표현

N X (N X 1) = N X N

```
input = [3, 5, 6, 1, 2, 4]


def find_max_num(array):
    max_num = array[0]
    for num in array:
        if num > max_num:
            max_num = num
    return max_num

result = find_max_num(input)
print(result)
```

대입
→ 연산 1번
→ array의 길이만큼
→ 비교연산
→ 대입연산

1 + (N X (1 + 1))
= 1 + N X 2 = 2N + 1

N^2 > 2N+1
숫자는 신경쓰지말고 입력값(N)에 비례해서 어느 정도로 증가하는 지만 파악하면 됩니다.

> 즉, $2N + 1$의 연산량이 나온 첫번째 풀이 방법은 N 만큼의 연산량이 필요하다
> N^2 의 연산량이 나온 두번째 풀이 방법은 N^2 만큼의 연산량이 필요하다.
>
> 참고로, 만약 상수의 연산량이 필요하다면, 1 만큼의 연산량이 필요하다고 말하면 됩니다.