점근표기법이란?
- 알고리즘의 성능을 수학적으로 표기하는 방법입니다. 알고리즘의 "효율성"을 평가하는 방법.
점근 표기법(Asymptotic notation)은 어떤 함수의 증가 양상을 다른 함수와의 비교로 표현하는 수론과 해석학의 방법이다.

Big-O
: 최악의 성능이 나올때 어느 정도의 연산량이 걸릴까
Big-Ω
: 최선의 성능이 나올때 어느 정도의 연산량이 걸릴까

Q. [3, 5, 6, 1, 2, 4] 같은 숫자로 이루어진 배열이 있을 때, 이 배열 내에 숫자가 존재한다면 True, 존재하지 않다면 False를 반환하시오.

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


def is_number_exist(number, array):
    # array 하나하나를 element 으로 지정
    for element in array:
        # element 가 number랑 같으면 true
        if number == element:
            return True
    # 아니면 false
    return False


result = is_number_exist(3, input)
print(result)
```

N
1

입력값에 최악일경우 N
출력값이 최상일경우 1
(3이제일앞에
있을경우 1번만에
True 반환)

알고리즘에서는 거의 모든 알고리즘을 빅오 표기법으로 분석합니다.
왜냐하면 대부분의 입력값이 최선의 경우일 가능성은 굉장히 적을 뿐더러,
우리는 최악의 경우를 대비해야 하기 때문입니다.

기억!
1. 입력값에 비례해서 얼마나 늘어날지 파악해보자. 1 ? N? N^2?
2. 공간복잡도 보다는 시간 복잡도를 더 줄이기 위해 고민하자.
3. 최악의 경우에 시간이 얼마나 소요될지(빅오 표기법)에 대해 고민하자.

알고리즘 문제

Q. 다음과 같이 0 혹은 양의 정수로만 이루어진 배열이 있을 때, 왼쪽부터 오른쪽으로 하나씩 모든 숫자를 확인하며 숫자 사이에 'X' 혹은 '+' 연산자를 넣어 결과적으로 가장 큰 수를 구하는 프로그램을 작성하시오.

단, '+'보다 'X' 를 먼저 계산하는 일반적인 방식과는 달리, 모든 연산은 왼쪽에서 순서대로 이루어진다.

[0, 3, 5, 6, 1, 2, 4]

```python
input = [0, 3, 5, 6, 1, 2, 4]


def find_max_plus_or_multiply(array):
    # 초기값 설정
    multiply_sum = 0

    # number 가 0 이나 1이거나 계산한 값이 0 이나 1이면
    # 곱화는 것보다 더하는 것이 더 큰 수가 나온다.
    for number in array:
        if number <= 1 or multiply_sum <= 1:
            multiply_sum += number
        else:
            multiply_sum *= number
    return multiply_sum


result = find_max_plus_or_multiply(input)
print(result)
```

Q. "abadabac" 와 같은 영어로 되어 있는 문자열이 있을 때, 이 문자열에서 반복되지 않는 첫번째 문자를 반환하시오. 만약 그런 문자가 없다면 _ 를 반환하시오.

```python
input = "abadabac"


def find_not_repeating_character(string):
    # 초기 array 설정
    alphabet_occurrence_array = [0] * 26

    # string 하나하나를 char 으로 지정
    for char in string:
        # char 이 알파벳이 아니라면 continue 맞다면 다음 코드라인 실행
        if not char.isalpha():
            continue
        # ASCII 코드 이용하기 (ord함수 사용으로 ASCII코드 10진법 숫자로 변환)
        arr_index = ord(char) - ord("a")  # 소문자만 쓴다는 전제하에 가능함
        alphabet_occurrence_array[arr_index] += 1

    # 초기값 설정 (array for not repeating alphabet)
    not_repeating_character_array = []
    # alphabet_occurrence_array 를 순서대로 스캔해서
    # 한번씩 밖에 사용되지 않은 문자들을 not_repeating_character_array 에
    # character 로 변환환서 넣어 준다.
    for index in range(len(alphabet_occurrence_array)):
        alphabet_occurrence = alphabet_occurrence_array[index]
        if alphabet_occurrence == 1:
            not_repeating_character_array.append(chr(index + ord("a")))

    # 다시 원래 input을 스캔해서 not_repeating_character_array에 있는
    # character 바로 반환.
    # 없을 시 "_" 반환.
    for char in string:
        if char in not_repeating_character_array:
            return char
        else:
            return "_"
```