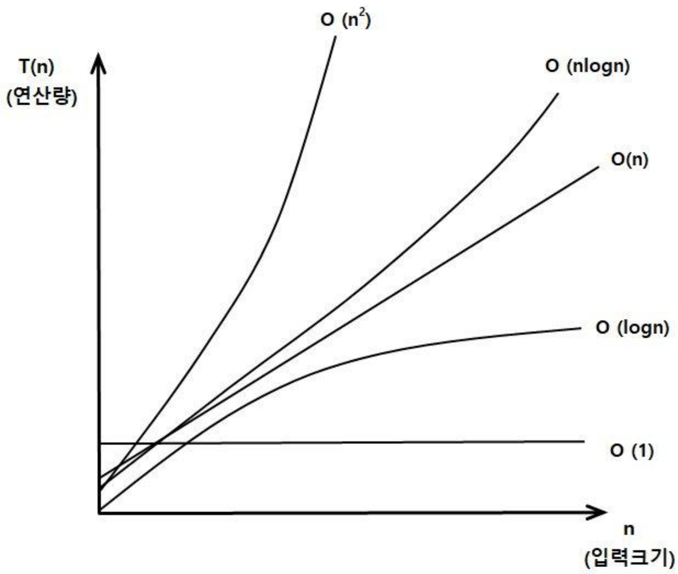
시간복잡도

시간복잡도란 입력크기와 문제를 해결하는데 걸리는 시간(프로그램의 연산횟수 : 연산량)과의 함수관계를 의미합니다.

예) $T(n) = 2n^2 + 3n$



시간복잡도를 표현하는 방법

1. Big - O 표기법
   최악의 상황으로 연산량을 계산하는 표기법
   예) 배열에서 5를 찾는 경우(순차탐색)

3	2	11	6	7	9	8	4	1	5

2. Big - Ω 표기법
   최선의 상황으로 연산량을 계산하는 표기법
   예) 배열에서 5를 찾는 경우(순차탐색)

5	7	8	6	2	9	11	4	1	3

3. Big - Θ 표기법
   최악과 최선의 평균으로 연산량을 계산하는 표기법

Big-O 표기법의 종류

1. O(1)
처리해야할 데이터양(입력크기)와 상관없이 항상 일정한 연산량을 갖고 있는 알고리즘

```
for i in range(10):
    sum += i
```

2. O(n)
처리해야할 데이터양과 비례해 연산량도 증가하는 알고리즘

```
for i in range(n):
    sum = sum + i
```

→

$T(n) = 2n$
Big-O : O(n)

3. O(n^2)
처리해야할 데이터양이 증가할수록 데이터양의 제곱만큼 연산량이 증가하는 알고리즘

```
for i in range(n):
    sum = 0;
    for j in range(n):
        sum = sum + i
```

→

$T(n) = 2n^2 + n$
Big-O : O(n^2)

```
for i in range(n):
    sum = 0;
    for j in range(i+1):
        sum = sum + i
```

→

$T(n) = \frac{1}{2}n^2 + \frac{1}{2}n$
Big-O : O(n^2)

※ 로그값 계산하는 법

1) $\log_a a = 1$, $\log_a a^b = b\log_a a = b$

2) $\log_2 1024 = \log_2 2^{10} = 10\log_2 2 = 10$

3) $\log_2 131072 = \log_2 2^{17} = 17$

4) $\log_2 1048576 = \log_2 2^{20} = 20$

5) $\log_2 16777216 = \log_2 2^{24} = 24$

4. O(logn)

처리해야할 데이터양이 증가해도 연산량이 별로 증가하지 않는 알고리즘

```
n = 1024
cnt = 0
i = 1
while i < n:
    i = i * 2
    cnt += 1
print(cnt)
```

5. O(nlogn)

```
n = 1024
cnt = 0
for i in range(n):
    j = 1
    while j < n:
        j = j * 2
        cnt += 1
print(cnt)
```