# ILE3-014 빅오 표기법

## 알고리즘의 복잡도

- **시간 복잡도**
  수행하는데 시간이 오래 걸린다
- **공간 복잡도**
  수행하는데 메모리를 많이 사용한다

### 시간 복잡도를 측정하는 방법

#### 함수의 수행 시간을 조사하는 방법

함수 시작 전 / 후의 시간을 측정하여 소요 시간을 확인

```
import time

start = time.time()

func()

end = time.time()
print(end - start)
```

> 입력 데이터가 있는 경우 매번 테스트를 해야 소요 시간을 알 수 있음

#### 수행되는 회수를 수학적으로 표현하는 방법

데이터의 개수에 따라 연산되는 횟수를 수학식으로 표현하는 방법

```
2n^2 + 5n + 2
```

> 입력 데이터가 들어오는 개수에 따라 실행을 해보지 않아도 예측이 가능 함

#### 빅오 표기법

**big-O notation**
알고리즘의 시간 복잡도를 표현하는 방법

점근 표기법의 일종 (증감 추세를 비교하는 표기법)

**표현 방법**

1. **수행되는 회수를 수학적으로 표현한다**

```
2n^2 + 5n + 2
```

2. **영향력이 없는 항을 제거한다**
   차수가 낮은 항은 차수가 높은 값에 비하여 소요 시간에 미치는 영향력이 미미하기 때문에 차수가 낮은 항은 제거

```
2n^2
```

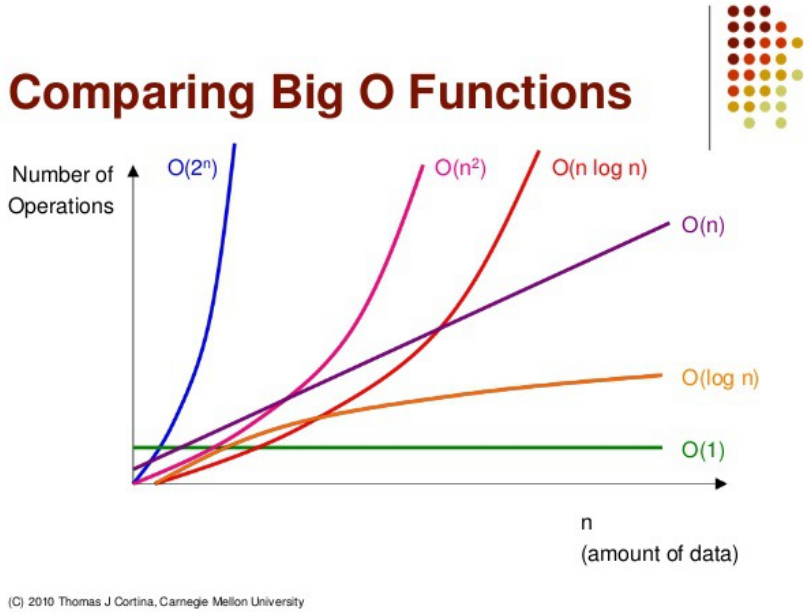
3. 상수항을 제거
   상수항은 입력값이 커질 수록 영향력이 감소하므로 제거

```
n^2
```

3. 빅오 표기법으로 표현

```
O(n^2)
```

**빅오 표기법에 따른 소요 시간 그래프**




**코드에서 빅오 표기법을 확인하는 방법**

차수를 높이는 주범인 `for` / `foreach` 문이 몇중으로 중첩되어 있는가를 확인

> 알고리즘 내에서 호출하는 함수 내의 for문 포함

**Exmaple**

버블 정렬 알고리즘

**버블 정렬**

버블 정렬 정적 시각화[1]

| | |
|---|---|
| **분류** | 정렬 알고리즘 |
| **자료 구조** | 배열 |
| 최악 시간복잡도 | $O(n^2)$ 비교, $O(n^2)$ 교환 |
| 최선 시간복잡도 | $O(n)$ 비교, $O(1)$ 교환 |
| 평균 시간복잡도 | $O(n^2)$ 비교, $O(n^2)$ 교환 |
| 공간복잡도 | $O(1)$ 보조 |

```
def bubbleSort(x):
    length = len(x)-1
    for i in range(length):
        for j in range(length-i):
            if x[j] > x[j+1]:
            x[j], x[j+1] = x[j+1], x[j]
    return x
```

`for i` 내에 `for j`가 중첩되어 있으므로

```
O(n^2)
```

## 과제

1. 이전 과제인 부분집합을 구하고 집합 연산을 수행하는 알고리즘에 대해서 빅오 표기법으로 알고리즘의 복잡도를 구하세요
2. 여러 정렬 알고리즘에 대해 조사하고, 각 알고리즘의 복잡도를 확인하세요