공간 복잡도

- 알고리즘 계산 복잡도는 다음 두 가지 척도로 표현될 수 있음
  - 시간 복잡도 : 얼마나 빠르게 실행되는지
  - 공간 복잡도 : 얼마나 많은 저장공간이 필요한지

  좋은 알고리즘은 실행 시간도 짧고, 저장 공간도 적게 쓰는 알고리즘

  - 통상 둘 다를 만족시키기는 어려움
    - 시간과 공간은 반비례적 경향이 있음
    - 최근 대용량 시스템이 보편화되면서, 공간 복잡도보다는 시간복잡도가 우선
    - 그래서! 알고리즘은 시간 복잡도가 중심

공간 복잡도 대략적인 계산은 필요함

- 기존 알고리즘 문제는 예전에 공간 복잡도도 고려되어야할 때 만들어진 경우가 많음
- 그래서 기존 알고리즘 문제에 시간 복잡도뿐만 아니라, 공간 복잡도 제약 사항이 있는 경우가 있음
- 또한, 기존 알고리즘 문제에 영향을 받아서, 면접시에도 공간 복잡도를 묻는 경우도 있음

Complexity:

- expected worst-case time complexity: O(N)
- expected worst-case space complexity: O(N)

---

1. 공간 복잡도 (Space Complexity)

- 프로그램을 실행 및 완료하는데 필요한 저장공간의 양을 뜻함
- 총 필요 저장 공간
  - 고정 공간(알고리즘과 무관한 공간) : 코드 저장 공간, 단순 변수 및 상수
  - 가변 공간(알고리즘 실행과 관련있는 공간) : 실행 중 동적으로 필요한 공간
  - $S(P)=c+Sp(n)$

- c : 고정 공간
- $Sp(n)$: 가변 공간

> 빅 오 표기법을 생각해볼 때, 고정 공간은 상수이므로 공간 복잡도는 가변 공간예 좌우됨

---

2. 공간 복잡도 계산

- 공간 복잡도 계산은 알고리즘에서 실제 사용되는 저장 공간을 계산하면 됨
  - 이를 빅 오 표기법으오 표현할 수 있으면 됨

공간 복잡도 예제1

- n! 팩토리얼 구하기
  - n! = 1x2x3...xn
- n의 값에 상관없이 변수 n, 변수 fac, 변수 index만 필요함
- 공간 복잡도는 O(1)

```
def factorial(n):
  fac = 1
  for index in range(2,n+1):
    fac = fac * index
  return fac

factorial(3)
>>6
```

공간 복잡도 예제2

- n! 팩토리얼 구하기
  - n! = 1 x 2 x ... x n
- 재귀함수를 사용하였으므로, n에 따라, 변수 n이 n개가 만들어지게 됨
  - factorial 함수를 재귀 함수로 1까지 호출하였을 경우, n부터 1까지 스택에 쌓이게 됨
- 공간 복잡도는 O(n)

```
def factorial(n):
  if n>1:
    return n * factorial(n-1)
  else:
    return 1
```