1-2. 알고리즘 성능 평가

※ 복잡도 : 알고리즘의 성능을 나타내는 척도

- 시간 복잡도 : 특정한 크기의 입력에 대해서 알고리즘 수행 기간 분석
- 공간 복잡도 : 〃 알고리즘 메모리 사용량 분석

⇒ 동일한 기능을 수행하는 알고리즘이 있다면 일반적으로 복잡도가 낮을수록 좋은 알고리즘.

복잡도 ⇒ 오스코드가 많다Ⓧ, 이해하기 복잡Ⓧ

⇓

특정 함수의 성능.

시간 ⇒ 실행 소요 시간

공간 ⇒ 많은 메모리

※ Big-O 표기법

- 가장 빠르게 증가하는 항만을 고려하는 표기법
  - 함수의 성향만을 나타냄
- 예를 들어 횟수가 $3N^3 + 5N^2 + 1,000,000$인 알고리즘.
  ⇒ 빅오표기법에서는 차수가 큰 항만을 남기므로 $O(N^3)$으로 표현
  (계수는 무시.)

↓

극한의 개념 N이 엄청 큰 수.

$3N^3$은 상당히 작은 수 ⇒ 함수의 성향만 고려시 함수의 성능 가능가능

좋음 ↑

$O(1)$: 상수시간 ⇒ 몇번의 연산만 거치면 ..

$O(\log N)$: 로그시간 ⇒ $\log(N)$에 비례

$O(N)$: 선형시간

$O(N \log N)$: 로그선형시간

$O(N^2)$: 이차시간

$O(N^3)$: 삼차시간

$O(2^N)$: 지수시간

↓ 나쁨

⭐ 시간 복잡도 계산해보기

#1) N개의 데이터의 합을 계산

⇒ 수행시간은 데이터 개수 N에 비례

⇒ $O(N)$

#2) 이중 반복문 (N=5 일 2번)

⇒ $O(N^2)$

⇒ 모든 이중 반복문이 $O(N^2)$인 것은 아님

⭐ 알고리즘 설계 Tip

python은 5~15초 가량 소요.

PyPy > C > Python

↳ 메모리 사용될 수도.

빠름

느림.

코딩테스트의 시간 제한은 1~5초 가량 ...

↳ 명세되지 않음.

⭐ 요구사항에 따라 적절한 알고리즘 설계

시간제한 1초.

⇒ N의 범위가 500인 경우, $O(N^3)$

⇒ N의 범위가 2,000인 경우, $O(N^2)$

⇒ N의 범위가 100,000인 경우, $O(N \log N)$

⇒ N의 범위가 10,000,000인 경우, $O(N)$

⭐ 알고리즘 문제 해결 과정

① 지문 읽기 및 컴퓨터적 사고

② 요구사항(복잡도) 분석

③ 문제해결을 위한 아이디어 찾기

④ 소스코드 설계 및 코딩.