자료구조

시간 복잡도와 공간 복잡도

자료구조와 알고리즘을 선택하는데 있어서
반드시 알아야하는 기본적인 척도입니다.

복잡도란?

- 알고리즘의 성능, 효율성을 나타내는 척도
- 크게 시간 복잡도(Time Complexity)와 공간 복잡도(Space Complexity)로 나눔
- 각 알고리즘이 주어진 특정 크기의 입력(N)을 기준으로
  수행시간(연산) 혹은 사용공간이 얼마나 되는지 객관적으로 비교할 수 있는 기준을 제시
- 복잡도를 나타내는 방법으로는 O(빅 오), Ω(오메가), Θ(세타) 등이 있고
  주로 빅 오와 세타 표기법이 많이 사용됨

복잡도란?

다시 말해, 어떤 알고리즘이 효율적인지를 판단하는 척도입니다.

알고리즘을 평가할 때 주로 수행시간과 메모리 사용량을 기준으로 두는데,
이 중 수행시간에 해당하는 것이 시간 복잡도, 메모리 사용량에 해당하는 것이 공간 복잡도입니다.

시간 복잡도

- 시간 복잡도란 특정 크기의 입력을 기준으로 할 때 필요한 연산의 횟수를 나타냅니다.
- 이름은 시간 복잡도이지만 실행시간이 아닌 연산 횟수를 세는 이유는 다음과 같습니다.
- 모든 OS, IDE 플랫폼에서 동일한 결과가 나오지 않습니다. 따라서 실행 시간 측정을 위한 또다른 방법이 필요합니다.

BEST CASE

최적의 입력으로
작업 완료까지
가장 연산 횟수가 적은 경우

WORST CASE

최악의 입력으로
작업 완료까지
가장 연산 횟수가 많은 경우

AVG CASE

여러 경우의 수를 고려,
총 연산횟수를 계산하고
시행 횟수로 나눈 경우

공간 복잡도

- 공간 복잡도란 프로그램 실행과 완료에 얼마나 많은 공간(메모리)가 필요한지를 나타냅니다.
- 알고리즘을 실행시키기 위해 필요한 공간은 두 가지로 나눌 수 있습니다.

고정 공간

알고리즘과 무관한 공간

코드가 저장되는 공간

알고리즘 실행을 위해 시스템이 필요로 하는 공간 등

가변 공간

문제를 해결하기 위해 알고리즘이 필요로 하는 공간

변수를 저장하는 공간

순환 프로그램일 경우 순환 스택(recursion stack) 등

시간 복잡도 vs 공간 복잡도

시간 복잡도는 얼마나 빠르게 실행되는지
공간 복잡도는 얼마나 많은 자원(메모리 공간)이 필요한지를 판단합니다.

시간 복잡도와 공간 복잡도는 반비례하는 경향이 있습니다.
그래서 보통 알고리즘의 성능을 판단할 때는 시간 복잡도 위주로 판단합니다.

빅 오(O) 표기법

- 빅 오(O) 표기법은 복잡도를 나타내는 점근 표기법 중 가장 많이 사용되는 표기법입니다.
- 가장 많이 사용되는 이유는 알고리즘 효율성을 상한선 기준으로 표기하기 때문입니다.
  최악의 경우를 고려하는 데 가장 좋은 표기법입니다. 알고리즘 효율성은 값이 클 수록,
  즉 그래프가 위로 향할수록 비효율적임을 의미합니다.
- 빅 오메가(Ω)는 하한선 기준, 빅 세타는 상한선과 하한선 사이를 기준으로 표기합니다.

빅 오(O) 표기법의 특징

상수항 무시

빅 오 표기법은 N이 충분히 크다고 가정합니다.
알고리즘의 효율성은 N 크기에 영향을 받으므로
상수항과 같은 사소한 부분은 무시합니다.

➡ O(2N)은 O(N)으로 간주

영향력 없는 항 무시

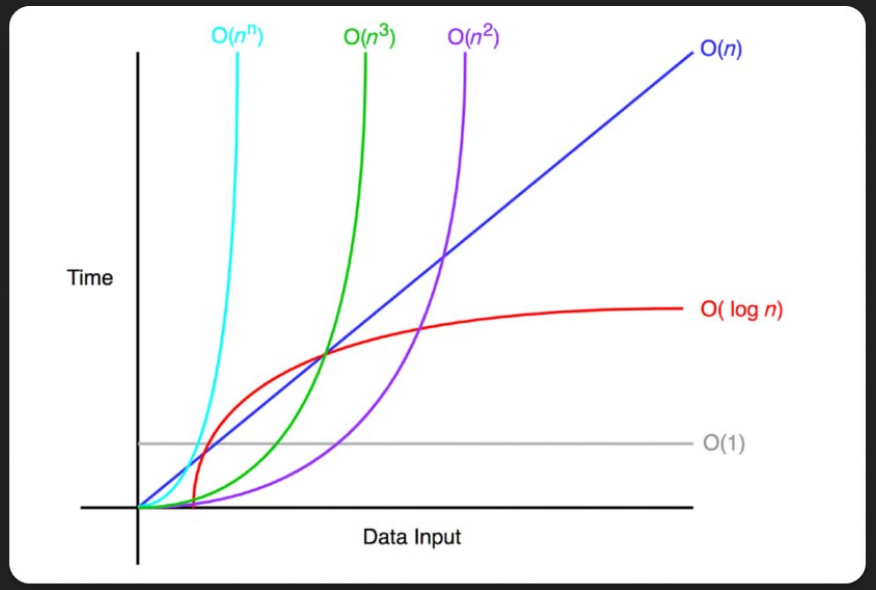
상수항을 무시하는 것과 같은 맥락으로
가장 영향력이 큰 항 이외에
영향력이 없는 항은 무시합니다.

➡ O(N ^ 2 + 2N + 1)은 O(N ^ 2) 으로 간주

시간 복잡도와 빅 오 표기법

- 시간 복잡도는 특정 크기의 입력(N)을 기준으로 실행하는 연산의 횟수입니다.
- 그렇다면 알고리즘이 실행될 때 모든 연산을 세어야 할까요?
- 빅 오 표기법의 특징대로 정답은 아니다 입니다. 핵심이 되는 연산의 횟수만 세면 됩니다.

시간 복잡도와 빅 오 표기법



복잡도	소요 시간	예제
O(1)	상수 시간	스택 push(), pop()
O(log N)	로그 시간	이진 트리
O(N)	직선적 시간	for 문
O(N log N)	선형 로그 시간	퀵 정렬(Quick sort), 병합 정렬(Merge sort), 힙 정렬(Heap sort)
O(N ^ 2)	2차 시간	이중 for 문, 삽입 정렬(Insertion sort), 거품 정렬(Bubble sort), 선택 정렬(Selection sort)
O(C ^ N)	지수 시간	피보나치 수열

$$O(1) < O(\log N) < O(N) < O(N \log N) < O(N^2) < O(2^N)$$

상수함수 < 로그함수 < 선형함수 < 다항함수 < 지수함수
왼쪽에서 오른쪽으로 갈 수록 성능이 떨어지며, 시간 복잡도가 좋지 않은 알고리즘

시간 복잡도를 구하는 요령

- 하나의 반복문을 사용하여 단일 요소 집합을 반복하는 경우 : O(N)
- 컬렉션의 절반 이상을 반복하는 경우 : O(N / 2) -> O(N)
- 두 개의 다른 반복문을 사용하여 두 개의 개별 컬렉션을 반복하는 경우 : O(N + M) -> O(N)
- 이중 반복문을 사용하여 단일 컬렉션을 반복하는 경우 : O(N^2)
- 이중 반복문을 사용하여 두 개의 다른 컬렉션을 반복하는 경우 : O(N * M) -> O(N^2)
- 컬렉션 정렬을 사용하는 경우 : O(N * log N)