알고리즘 개요

- 알고리즘은 어떤 문제를 풀기 위한 절차나 방법
- 알고리즘은 주어진 '입력'을 '출력'으로 만드는 과정
- 알고리즘의 각 단계는 구체적이고 명료해야 한다
- 알고리즘은 유한한 결과를 도출해야 한다
- 알고리즘은 보다 효율적인 처리를 할 수 있다
- 어떤 문제를 푸는 방법이 꼭 한가지는 아니다
    - 절대값은 음수에 – 부호를 취하거나
    - 주어진 수에 제곱 후 제곱근을 취하는 방법이 있다
- 이 중 효율적인 알고리즘을 작성하는 것이 목표
- 문제를 푸는 과정 중 적당한 알고리즘을 사용하려면 각 알고리즘들이 어떠한 특징을 지니고 있는지 알아야 함
- 알고리즘 분석은 알고리즘의 성능이나 특징을 분석하는 것이다
- 알고리즘 분석 후 코드로 옮기기 위해 각 사용할 언어의 기본적인 문법을 숙지하고 있을 필요가 있고, 자료구조를 알고 있다면 좀 더 효과적인 알고리즘을 작성할 수 있다

알고리즘 시간복잡도 및 빅오 표기법

빅오(Big-O) 표기법은 불필요한 연산을 제거하여 알고리즘 분석을 쉽게 할 목적으로 사용된다

시간 복잡도 : 알고리즘의 성능을 설명하는 것

좀 더 풀어서, 알고리즘을 수행하기 위해 프로세스가 수행해야하는 연산을 수치화 한 것

시간 복잡도에서 가장 중요하게 보는 것은 연산에 결과에 영향을 미치는 n 의 단위 이다

Ex)

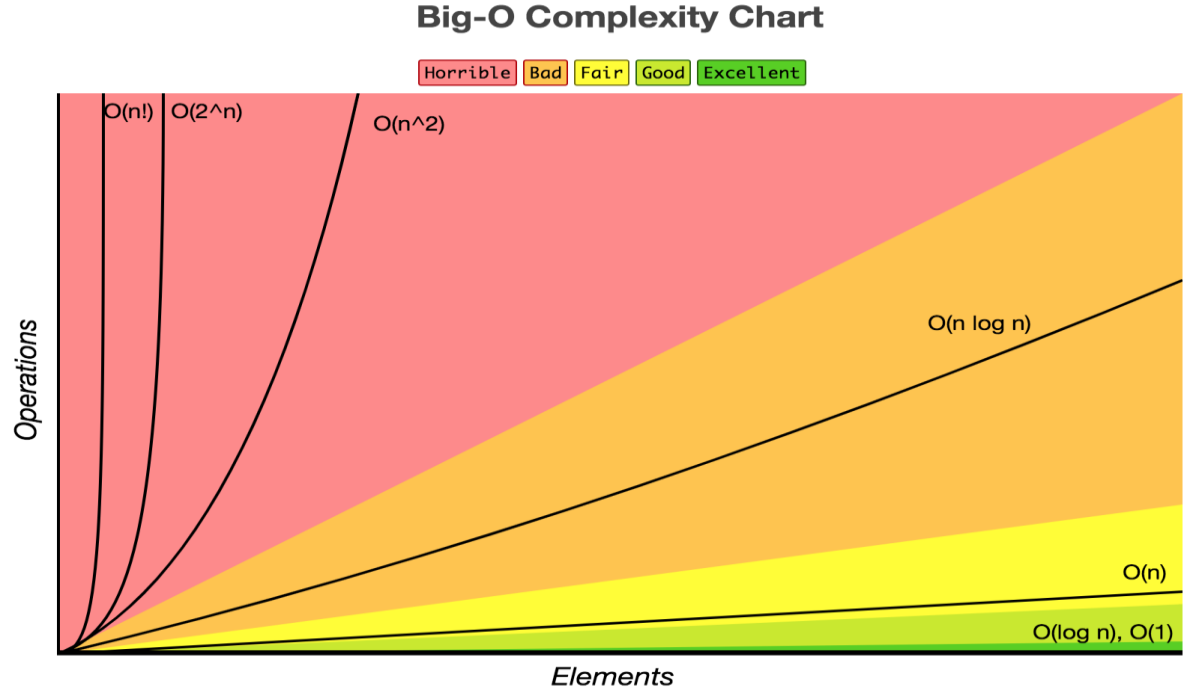
1	O(1)	→ 상수
2n + 20	O(n)	→ n 이 연산 결과에 가장 큰 영향
3n^2	O(n^2)	→ n^2 이 가장 큰 영향

문제 해결 단계를 나열하면

1. O(1) 상수 시간 : 문제를 해결하는 오직 한 단계만 처리
2. O(log n) 로그 시간 : 문제를 해결하는데 필요한 단계들이 연산마다 특정 요인에 의해 줄어든다 (밑 수는 상관없음. 보통 2)
3. O(n) 직선적 시간 : 문제를 해결하기 위한 단계의 수와 입력 값 n 이 1:1 의 관계를 가짐

4. O(n log n) 선형 로그 시간: 문제를 해결하기 위한 단계의 수가 N * (log N) 번 만큼의 수행 시간을 가진다
5. O(n^2) 2 차 시간 : 문제를 해결하기 위한 단계의 수는 입력값 n 의 제곱
6. O(C^n) 지수 시간 : 문제를 해결하기 위한 단계의 수는 주어진 상수값 C 의 n 제곱

복잡도	1	10	100
O(1)	1	1	1
O(log N)	0	2	5
O(N)	1	10	100
O(N log N)	0	20	461
O(N^2)	1	100	10000
O(2^N)	1	1024	엄청 큼
O(N!)	1	3628800	엄청 큼



O(1) < O(log n) < O(n) < O(n log n) < O(n^2) < O(2^n) < O(n!)