알고리즘 복잡도 / Big-O / 점근적 분석

알고리즘 복잡도

> 알고리즘의 성능을 나타내는 척도

- 시간복잡도 - 알고리즘 수행 시간
- 공간복잡도 - 알고리즘 수행에 필요한 메모리

시간 복잡도(time complexity)

- 알고리즘의 수행시간
- 실제 수행시간은 실행환경의 영향을 받는다
- 실행 시간을 측정하는 대신 **연산의 실행 횟수**를 카운트
- 연산의 실행 횟수는 **입력 데이터의 크기에 관한 함수**로 표현한다
- 데이터의 크기가 같더라도 실제 데이터 내용에 따라서 달라진다
    - 최악의 경우 시간복잡도: worst -case analysis
    - 평균 시간 복잡도: average-case analysis

점근적 분석

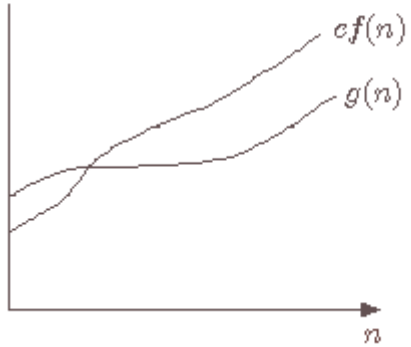
> 어떤 함수(알고리즘 시간 복잡도)의 증가 양상을 다른 함수와의 비교로 표현함

- 입력되는 데이터의 크기에 따라 수행 시간과 공간을 얼마나 차지하는지 측정
- 정확한 것은 아니고 대략 이런식으로 수행시간이 나온다는걸 측정하는 용도
- Big-O 표기법이 많이 사용됨

Big-O 표기법

> 최고차항만 가지고 수행시간을 표기

- 점근적 상한선 : '**최악의 경우**'를 전제로 최대치에서 알고리즘과 비교합니다.
"아무리 알고리즘이 나빠도 비교 함수와 같거나 좋다."



Big-O 표기 방법

1. 최고차항만 남긴다
2. 최고차항 계수(상수)는 생략
3. Big-O로 감싸줌

4. 시간 복잡도가 빠른 순
   O(1) > O(logn) > O(n) > O(nlogn) > O(n^2) > O(n^3) > O(2^n) > O(n!)

- O(1) : 입력 데이터와 상관 없이 일정한 실행 시간을 갖는 알고리즘을 뜻합니다.
- O(log n) : 입력 데이터가 증가하면 실행 시간이 조금씩 증가하는 알고리즘을 뜻합니다.

  -> 성능 좋은 탐색 알고리즘은 대부분 log n의 수행 시간을 갖습니다.

- O(n) : 입력 데이터 수와 비례하여 선형적으로 증가하는 알고리즘을 뜻합니다.

- O(n* log n) : 입력 데이터가 늘어난 양보다 조금 더 늘어난 수행 시간을 갖는 알고리즘을 뜻합니다.

  -> 커다란 문제를 나누어 해결하고 이를 다시 합치는 과정에서 나타냅니다.

- O(n^2) : 입력 데이터가 커질수록 배수로 늘어나는 수행시간을 갖는 알고리즘을 뜻합니다.

  -> n이 두배면 수행 시간은 네배로 늘어나고 데이터가 많으면 감당 할 수 없습니다.

- O(n^3) : 입력 데이터가 커질수록 배수로 늘어나는 수행시간을 갖는 알고리즘을 뜻합니다.

  -> n이 두배면 수행 시간은 여덟배로 늘어나고 데이터가 많으면 감당 할 수 없습니다.

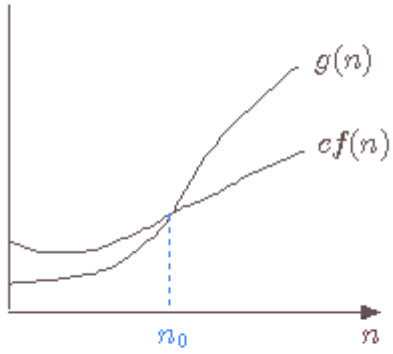
- O(2^n) : 입력 데이터가 증가하면 수행 시간이 급격하게 증가하는 알고리즘을 뜻합니다.

  -> 다른 방법을 찾아야 하며 이는 흔하지 않은 알고리즘입니다.
- O(n!) : 입력 데이터에 따라 입력 데이터 양의 팩토리얼만큼 증가

  -> 2, 4, 8, 16 ...

Ω(오메가 표기법)

- 점근적 하한선 : **'최선의 경우'**를 전제로 최소치에서 알고리즘을 비교합니다.
"아무리 알고리즘이 좋아도 비교 함수와 같거나 나쁘다."



$g \in \Omega(f)$

빅 오메가와 반대입니다.