알고리즘 1차 과제

알고리즘 시간 복잡도 분석

2020. 09. 24. 목

컴퓨터공학과

2019305059

이현수

시간복잡도 분석은 단위연산이 수행되는 횟수를 입력 크기에 대한 함수로 구하는 분석으로 알고리즘을 분석하는 표준이다. (단위연산: 비교 연산과 같이 기본이 되는 연산의 수행 횟수를 사용.)

- **모든 경우 시간복잡도 분석(Every-case time complexity analysis)**

  입력의 크기가 같을 경우, 입력의 값과 상관없이 항상 알고리즘 성능이 같은 경우

  예) 배열의 수 더하기, 교환정렬, 행렬곱셈

- **그렇지 않은 경우 분석**

  - 최악의 경우 시간복잡도

  일반적으로 최악 경우 분석으로 알고리즘 복잡도를 나타낸다. 최악 경우 분석은 '어떤 입력이 주어지더라도 얼마 이상을 넘지 않는다' 라는 표현이다.

  - 평균의 경우 시간복잡도

  입력의 확률 분포를 가정하여 분석하는데, 대부분의 경우 균등 분포를 가정한다. 즉, 입력이 무작위로 주어진다고 가정하는 것이다. 실제값이 평균에 크게 벗어나지 않을 경우에만 평균을 '전형적'이라고 할 수 있다. 특히 핵발전소 감시 시스템과 같이 반응시간을 취급하는 알고리즘의 경우 특히 주의 필요.

  - 최선의 경우 시간복잡도

  거의 사용되지 않으나, 최적 알고리즘을 고안하는데 참고 자료로서 활용되기도 한다. 일반적으로 별다른 의미가 없다.

- 차수 : 알고리즘의 복잡도를 표시하기 위하여 사용하는 표기법

  일차시간 알고리즘은 시간복잡도가 n, 10n 등 인 알고리즘이고, 이차시간 알고리즘은 시간복잡도가 n^2, 0.01n^2 등 인 알고리즘이다. 이때 어떤 일차시간 알고리즘은 궁극적으로 어떤 이차시간 알고리즘보다 효율적이다.

  만약 시간복잡도 함수가 $2n^2$+5n+100 이라면 다 무시하고 O(n^2)이라고 표기한다.

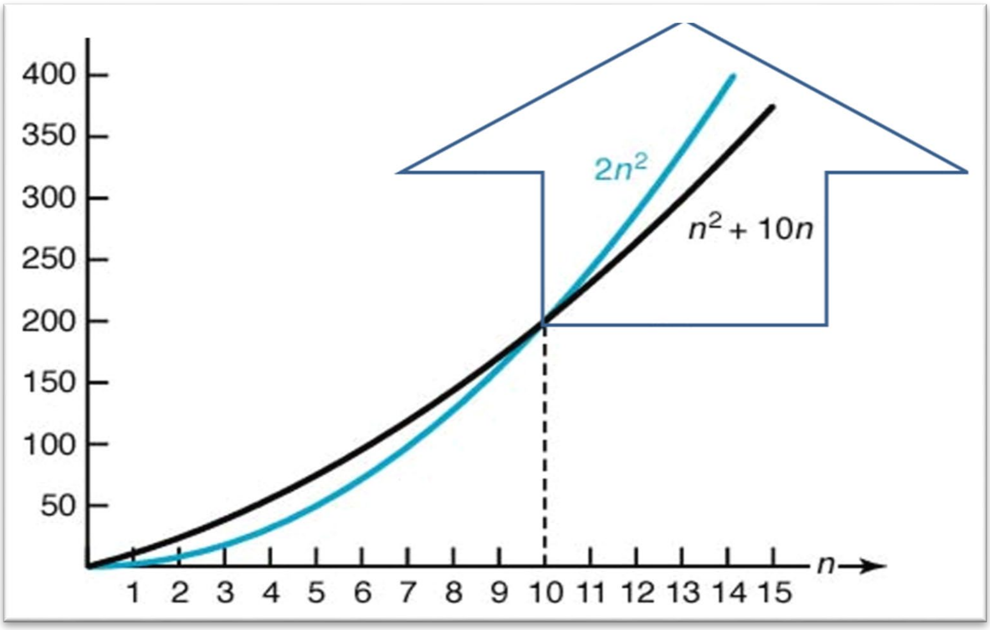
시간복잡도는 입력크기에 대한 함수로 표기하는데, 이 함수는 주로 여러 개의 항을 가지는 다항식이다. 그래서 이를 단순한 함수로 표현하기 위해 점근적 표기를 사용한다. 점근적 표기법은 입력크기 n이 무한대로 커질 때의 복잡도를 간단히 표현하기 위해 사용하는 표기법으로 실행환경과 무관하게 개략적으로 분석한다. 다음과 같은 점근적 표기를 사용한다.

- O(Big-Oh)-표기

- Ω (Big-Omega)-표기

- Θ(Theta)-표기

1. O(Big-Oh)-표기 (점근적 상한) - worst



시간적 복잡도를 데이터 크기 N의 함수로 표시하면서 이때 계수는 무시한다.
빅오 표기는 "아무리 길어도 최대 이 시간까지이다" 라는 것을 의미한다.

사진의 그래프를 보면 f(n)을 단순화 한 n^2에 임의의 상수 c(c=2, c>0)를 곱한 cn^2, 즉 $2n^2$이 n이 증가함에 따라 f(n) = n^2 + 10n의 상한이 된다.

즉 f(n)= n^2 +10n과 2 n^2 이 교차하는 n = 10 이후 모든 n에 대해, 즉 무한대로 증가할 때 f(n)=n^2+10n은 $2n^2$보다 절대로 커질 수 없다. 따라서 O(n^2)이 f(n)=n^2+10n의 점근적 상한이 된다.

*** 빅오 표시 예시**

$2n^2$ + 10n + 50 --→ O(n^2)

$100n^3$ + $50n^2$ --→ O(n^3)

50n + 100 --→ O(n)

*** 바싹 다가선 정의**

알고리즘의 특성을 표현할 때 바싹 다가선 상한을 사용한다.
예로 들어 $5N^2$ + 10N = O(N^2) = O(N^3) = O(N^4)표현도 가능하지만
바싹 다가선 정의는 $5N^2$ + 10N = O(N^2) 이다.

*** 컴퓨터 분야에서 시간복잡도를 위해 자주 사용하는 O-표기**

O(1) 상수 시간

O($\log n$) 로그(대수) 시간

O(n) 선형 시간

O(n$\log n$) 로그 선형 시간

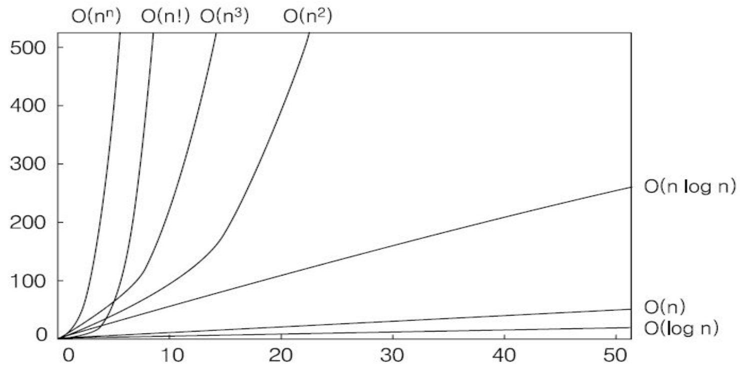
O(n^2) 제곱 시간

O(n^3) 세제곱 시간

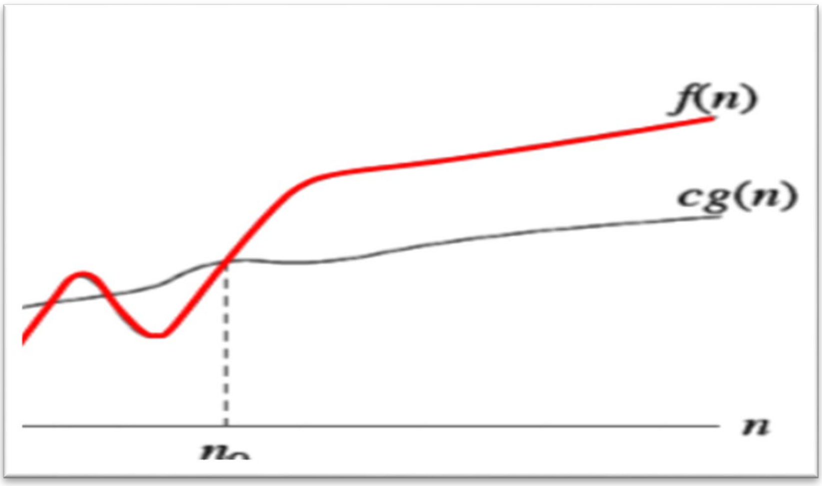
O(n^k) 다항식 시간, k는 상수

O(2^n) 지수 시간

O(n!) factorial



2. Ω (Big-Omega)-표기 (점근적 하한) - best



Ω-표기는 빅오 기호의 반대개념으로 알고리즘 수행시간의 하한을 뜻하며 "최소한 이정도 시간은 걸린다"라는 뜻이다.

사진의 그래프를 보면 n이 n_0이후부터는 즉, 그 이후의 모든 n에 대해서 cg(n)이 시간복잡도 함수 f(n)보다 항상 작다는 것을 알 수 있다. 그러므로 n이 증가함에 따라 Ω(g(n))이 점근적 하한이라고 할 수 있다.

*** 빅 오메가 예시**

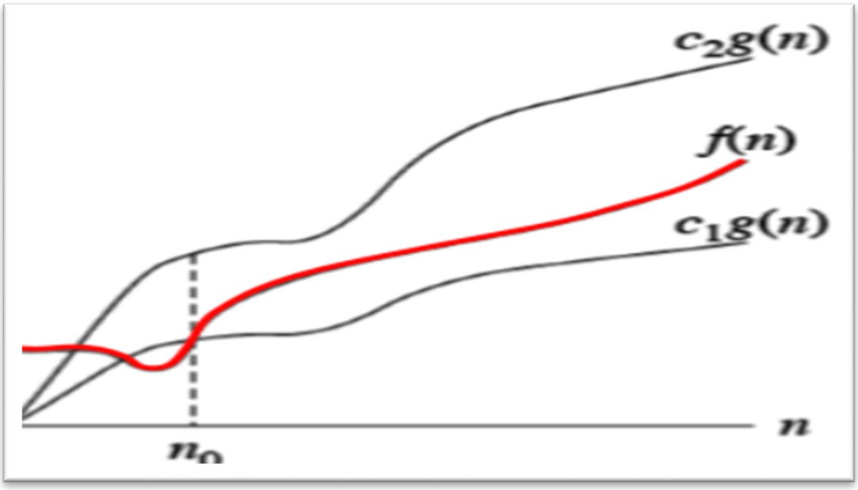
n개의 데이터가 있을 때 모든 정렬알고리즘은 Ω(n).

-> n개의 데이터를 정렬하기 위해서는 n개의 데이터 모두를 읽지 않고 정렬을 완료할 수 없다.

n개의 데이터가 배열에 있을 때 배열에 있는 n개의 데이터 값의 합 구하기 알고리즘은 Ω(n).

-> 배열안의 n개의 데이터를 모두 읽어야 더한값을 알 수 있다.

3. Θ(Theta)-표기 (동일한 증가율) - average



Θ-표기는 복잡도 O-표기와 Ω-표기가 같은 경우를 뜻한다.

사진의 그래프를 보면 n이 n_0이후부터는 즉, 그 이후의 모든 n에 대해서 복잡도 함수 f(n)이 상한 O(g(n))과 하한 Ω(g(n))을 동시에 만족한다는 것을 볼 수 있다.

* **Θ-표기 예시**

F(n) = n^2 + 5n + 3 = O(n^2)이고 F(n) = n^2 + 5n + 3 = Ω (n^2)이므로 f(n) = **Θ(**n^2)이다.

Ex1) n개의 데이터가 있는 배열에서 특정 원소를 순차탐색을 할 때

O(n) : 최악의 경우 마지막까지 탐색을 해야함.

Ω(1) : 운이 좋게 바로 특정 원소를 찾을 때.

빅 오와 빅 오메가가 다르기 때문에 Θ는 존재하지 않는다.

Ex2) n개의 데이터가 있는 배열에서 모든 원소의 값을 더할 때

O(n)

Ω(n)

빅 오와 빅 오메가가 같기 때문에 Θ는 Θ(n)로 존재한다.