알고리즘의 성능 분석과 시간 복잡도

2019.11.13 이정현

프로그램의 성능과 시간

1.현재 프로그램들의 규모가 이전에 비해서 엄청나게 커지고 있기 때문이다.

처리할 자료의 양이 커지면 커질수록 수행시간이 큰 프로그램과 작은 프로그램의 차이는 더욱 커진다 예를 들어 같은 일을 수행하는 프로그램이 있다.

프로그램 A의 수행시간은 n^2이고

프로그램 B의 수행시간은 2^n이라 가정하자

그럼 작은 수인 1,2,3,4등등 에서는 수행 시간의 차이가 크지 않다고 느껴 질 수 있다.

하지만 숫자가 100으로만 가도 A 프로그램은 10000초

B 프로그램은 4*10^22년이라는 엄청난 시간 차이가 나오게 된다.

따라서 프로그램에서의 수행 시간은 아주 중요한 부분을 차지하고 있다.

시간 복잡도와 빅O표기법(상한 식 표기=최악의 경우)

빅O표기법은 가장 영향을 많이 미치는 항만 시간 복잡도 함수의 수를 따진다.

FOR EXAMPLE

T(n)=n^2+n+1에서

n이 1000이라고 가정하면 이 식에서 n^2은1000000 n은1000으로 겨우 이 전체 식의 0.1%만을 차지하므로 가장 큰 항인 "n^2"만을 고려한다.

빅O표기법의 정의

모든 $n \geq n_0 > 0$ 에 대하여 $0 \leq f(n) \leq k \cdot g(n)$ 인 양의 상수 k와n_0가존재하면$f(n) = O(g(n))$ 이다.

예를 들어 설명하겠다.

f(n)=5n^2+3 일때

g(n)=n^2 이라고 가정하면

빅O표기법의 정의로 인하여 이 식에서 k와 n의 값이 존재하면 빅O표기법을 구할 수 있다.

$5n^2+3<=kn^2$

에서 k가 6이고 n0를 2라 가정하였을 때 n이 2 이상의 모든 수일 때 성립하므로 빅O표기법은 $O(n^2)$에서 성립한다.

다음으로는 빅오표기법을 이용하여 $6n^2+3$이 $O(n)$이 될 수 없음을 설명해 보겠다.

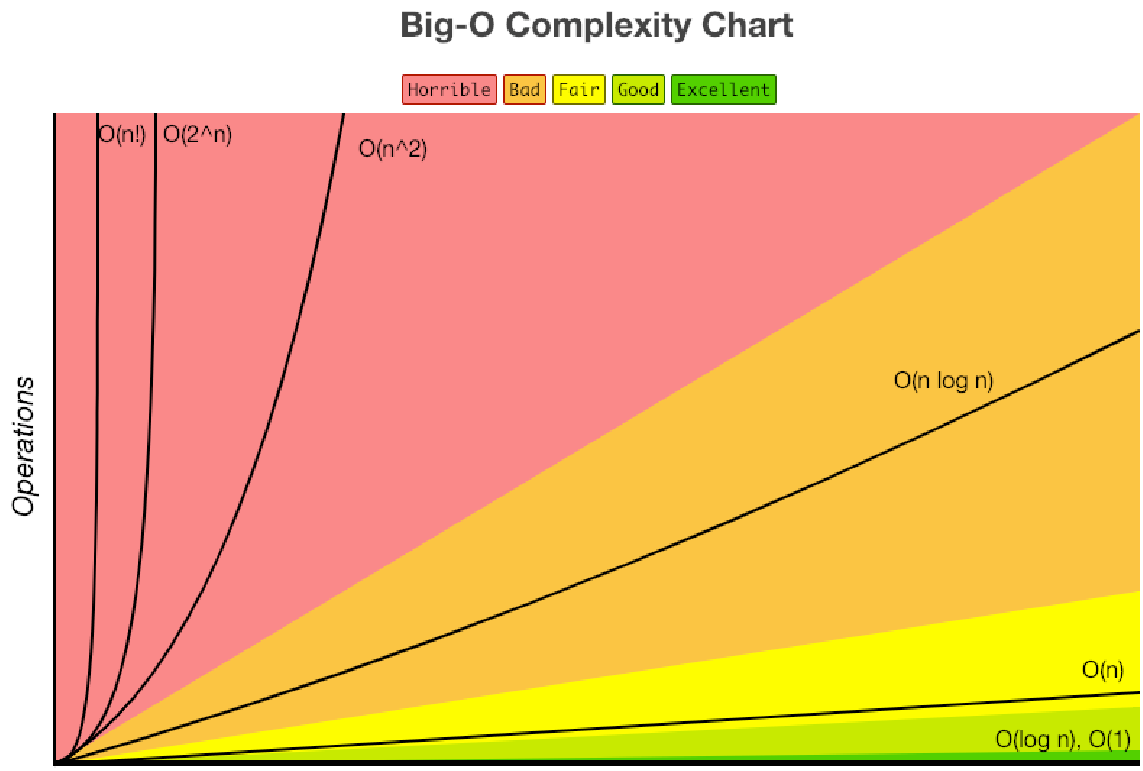
먼저 $f(n)= 6n^2+3$

$g(n)=n$으로 가정해보겠다.

$6n^2+3<=kn$

이 되고 이 식에서 n0를1 k를9 라고 가정해 보겠다.

이 경우n0값을 넣었을 때는 빅오표기법의 정의를 만족하나 n0보다 큰 n값인 2를 넣었을 때는 값이 맞아 떨어지지 않아 $O(n)$이라고 표기할 수 없게 된다.



빅O표기법의 시간복잡도 순서대로

O(1):상수형

O(log n):로그형

O(n):선형

O(n log n):선형로그형

O(n^2):2차형

O(n^3):3차형

O(2^n):지수형

O(n!):팩토리얼형

의 순으로 시간이 적게 소요된다..

빅오표기법의 상한식 표기방법은 상황에 따라서 여러가지 형태가 나올 수 있다.

예를 들어 f(n)=2n+1인 경우

f(n)= 2n+1

g(n)=n 이라 가정하면

2n+1<=kn 이라 놓고 k에 3 n0를 1로 설정할 경우 O(n)은 성립하고

또

f(n)= 2n+1

g(n)=n^2 이라 가정하면

2n+1<=kn^2 이라 놓고 k에2 n0를 1로 설정할 경우 n0보다 큰 모든 n에서 O(n^2)또한 성립하게 된다.

이러한 여러 형태가 나오는 문제를 보완하기 위하여 빅오메가표기법가 빅세타표기법이 나오게 되었다.

빅오메가표기법(하한 식 표기=최선의 경우)

빅오메가표기법의 정의는 다음과 같다.

두개의 함수 f(n)과 g(n)이 주어졌을 때 모든 n>n0에 대하여 |f(n)|>=k|g(n)|을 만족하는 2개의 상수 k와 n0가 존재하면 f(n)= Ω(g(n))이다.

예시로 알아보겠다.

$f(n) = 2n^2 + 3$, $g(n) = n^2$ 일 때, $f(n) = \Omega(n^2)$를 증명하라.

2n^2+3>=kn^2

k 를 1 n0 를 0 이라고 가정하면 n0 보다 큰 모든 n 에 대하여 성립하게 되므로 이 식은= $\Omega(n^2)$ 이라 할 수 있다.

빅세타표기법은 빅오와 빅오메가를 모두 만족하는 경우이다.

위에서 사용하였던

f(n)= 2n+1을 이용하여 설명하겠다. 이 함수는 빅오표기법 O(n)과 빅오메가표기법Ω(n)을 모두 만족하므로 f(n) = θ(n)이라고 표기할 수 있다.

마지막으로 위의 각 표기법에서 언급하였던 최선과 최악의 경우 또 평균의 경우를 알아보겠다.

최선의 경우= 수행시간이 가장 적은 경우

최악의 경우= 수행시간이 가장 오래 걸리는 경우

평균의 경우= 평균

탐색을 하는 알고리즘을 통해 설명 해 보겠다.

먼저 배열에 정수가 들어있다고 가정하겠다.

배열의 n번째 숫자를 화면에 출력하는 경우

이 경우 n번째 숫자를 화면에 출력한다➔ 즉 정확한 배열의 값을 한번에 출력하므로 시간 복잡도는

최선 최악의 경우가 모두 O(1)이 나오게 된다.

배열안의 숫자 중에서 최소값을 찾는 경우

이 경우 최소값이 맨 처음에 있든 맨 마지막에 있든 배열의 끝까지 가서 비교하여 찾아야 하므로 시간 복잡도는 최선 최악의 경우 모두 O(n)이 된다.

배열안의 숫자와 입력 값을 비교하여 찾는 경우

이 경우 만일 찾고자 하는 값이 맨 앞에 있는 경우는 1번만에 찾게 되므로 시간 복잡도가 O(1) 최선의 경우가 된다. 만일 찾고자 하는 값이 맨 마지막에 있는 경우 프로그램은 끝까지 수행 되어야 하므로 시간 복잡도는O(n) 최악이 되게 된다.