알고리즘의 수학적 배경

알고리즘이 개발시, 성능평가 기준으로 사용됨.

밀레니엄 난제인 P = NP 문제의 의미가 여기에서 나옴. (모든 Compute Problem들이, 선형시간내에 해결가능하나의 문제)

Big-O Notation

> 알고리즘 성능 평가 방법 중 가장 많이 사용되는 방법, 최고의 성능과 최악의 성능을 측정하는 방법

$\begin{aligned} T(N) \leq CF(N),\ N \geq N_1\ 이라는\ 두\ 조건을\ 만족하는 \ 상수\ C와\ N_1이\ 존재한다면\ T(N) = O(F(N))\ 이라고\ 한다. \ \end{aligned}$

Omega Notation

> 알고리즘의 성능이 최고인 경우를 측정한다.

$\begin{aligned} T(N) \geq CF(N),\ N \geq N_1\ 이라는\ 두 \ 조건을\ 만족하는 \ 상수\ C와\ N_1이\ 존재한다면,\ T(N) = \Omega(F(N))\ 이라고\ 한다. \end{aligned}$

Theta Notation

> 최고/최악이 아닌 정확한 알고리즘을 측정하는 방식

$\begin{aligned} T(N) = O(F(N))이라는\ 조건을\ 만족하는\ 상수 C_1,N_1\ 이\ 존재하고 \ T(N) = \Omega(F(N)) 이라는\ 조건을\ 만족하는\ 상수\ C_2, N_1이 존재할때,\ T(N) = \Theta(F(N))이라고 한다. \end{aligned}$

분석과정

$\sum_{i=1}^{100} i$ 이라는 수학식이 있을때, 코드는 다음과 같다.

```
#include <stdio.h>

void main()
{
    int Sum = 0;
    int i=0;

    for(i=1; i<=100; i++)
        Sum += i;
}
```

위의 알고리즘을 분석하기 전에는 다음과 같은 몇 가지 가정이 필요하다. ***(C++ 기준)***

1. 헤더 파일은 알고리즘의 성능에 영향을 주지않는다.
2. 함수 진입과 함수 반환은 알고리즘의 성능에 영향을 주지 않는다.
3. 프로그램은 첫 번째 행부터 마지막 행까지 차례로 실행된다.

위 가정대로라면, 1행의 헤더 파일과 3,4행은 알고리즘에 성능에 영향을 주지 않으며, 5행 부터 12행까지의 코드를 보면 된다. 5, 6, 12 행은 1회만 실행되지만, 8행부터 10행은 반복문이다. 8행은 101회, 9행은 100회 실행된다.

알고리즘 성능에 영향을 주는 코드	실행횟수
int Sum = 0	1
for(i=1; i<=100; i++)	101회
iSum += i	100회

위 표의 실행 횟수를 Big-O 표기법으로 나타내면 다음과 같다.

$\sum_{i=1}^{100} i = O(202) = O(1)$

> 204라는 상수의 존재는 알고리즘의 성능에 아무런 영향을 끼치지 못한다. 따라 빅오표기법으로 나타내면 위 알고리즘은 O(1)이 된다. 알고리즘 성능이 고정적이라는 것. 즉 1부터 100까지 합의 구하는 경우라면 위 알고리즘 최적화 하더라도 알고리즘 성능에는 거의 영향을 주지않는다.

$\sum_{i=1}^{N} i$ 의 경우엔, 코드는 일부가 달라진다.

```
for(i=0; i<N; i++)
    Sum++;
```

이 프로그램은 N에 따라, for 문의 반복횟수가 결정되며, 반복 횟수는 성능에 큰 영향을 준다.

$\sum_{i=1}^{N} C = C + \cdots + C\ (N\ times) = O(C \times N) = O(N)$

표기법의 종류

- $O(1)$
- $O(logN)$
- $O(N)$
- $O(N log N)$
- $O(N^2)$
- $O(N^3)$
- $O(2^n)$

> 기타 시그마 연산은, 수1에 나오는 공식 참고 ***(특히 중첩 시그마)***