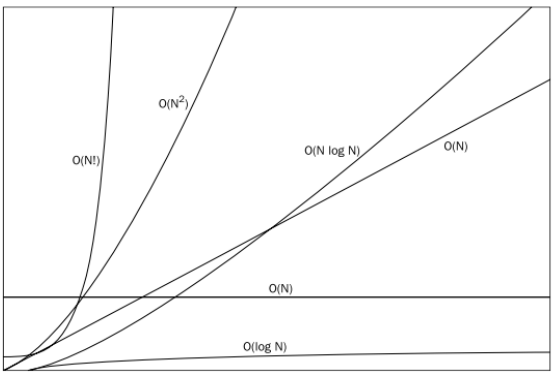

O(1)<O(logn)<O(n)<O(nlogn)<O(n2)<O(n3)<O(2n)<O(n!)

"Big O 표기법은 컴퓨터 공학에서 알고리즘의 복잡도 또는 성능을 표현하기 위해 사용된다. Big O는 특히 최악의 경우를 표현하며, 알고리즘의 실행시간이나, 사용 메모리 (메모리, 또는 디스크) 공간을 표현하기도 한다."

↳ Big O = time complexity

$O(1)$

O(1) 은 알고리즘의 실행시간 (또는 공간)이 입력되는 데이터의 크기에 상관없이 항상 같을 경우를 의미한다.

```
bool IsFirstElementNull(String[] strings)
                    {
                if(strings[0] == null)
                        {
                    return true;
                        }
                return false;
                    }
```

Colored by Color Scripter cs

$O(N)$

O(N) 는 입력되는 데이터의 양에 따라 비례하여 처리시간이 증가하는 알고리즘을 표현한다. 아래의 예는 어떻게 Big O가 최악의 경우에 대한 성능을 보여주고 있다. 주어진 문자열은 for 루프 중간에 찾아져 일찍 수행이 끝날 수도 있다. 그러나 Big O 표기는 항상 최대로 반복이 이뤄질 경우를 상정한 한계 값을 가정한다.

```
bool ContainsValue(String[] strings, String value)
                    {
            for(int i = 0; i < strings.Length; i++)
                        {
                    if(strings[i] == value)
                            {
                        return true;
                            }
                        }
                return false;
                    }
```

Colored by Color Scripter cs

$O(2N)$

O(2N) 은 입력 데이터가 하나 증가할 때마다 처리시간이 두배씩 증가하는 경우를 표현한다. O(2N)이 수행시간은 아주 빠르게 증가하여 엄청나게 커진다.

ex) Binary Search Tree : 이진 탐색(이분법) 트리

Log 함수

Log 함수는 설명하기가 좀 까다로와 일반적인 예를 들어보겠다.

이진 탐색(Binary search) 는 정렬된 데이터를 검색하는 기술이다. 이는 데이터의 중간 값을 취하여, 원하는 값이 맞는지 비교하는 방식으로 수행한다. 만약에 값을 찾았다면, 성공적으로 종료할 것이고, 찾는 값이 중간을 취한 값보다 클 경우, 큰 값들이 있는 상위 반값의 중간지점의 값을 비교를 시도할 것이다. 마찬가지로 찾는 값이 중간 지점의 값보다 작다면 작은 값들의 반쪽에서 값을 취할 것이다. 이런 방식으로 더이상 중간값을 취할 수 없을 때까지 수행을 반복한다.

이런 형태의 알고리즘의 경우 O(log N)으로 표기한다. 이진 탐색에서 반복 수행하는 입력자료는 초기 최대의 증가를 보이다가 점차로 평평해지는 곡선을 갖는다. 예를 들어 입력 데이터를 10개 처리하는데 1초가 걸린다면, 100개는 2초, 1000개는 3초가 걸리게 될 것이다. 입력 데이터량이 2배가 된다하더라도, 이로 인해 수행시간이 지연되는 것은 아주 적다. 왜냐하면 한번 반복 수행으로 입력 데이터의 절반이 수행이 되기 때문이다. 이진 탐색 같은 알고리즘은 대용량의 데이터를 처리하는데 아주 효율적이다.