

자료구조와 알고리즘

1강 - 알고리즘 기초

LECTURED BY SOONGU HONG

* 알고리즘 선택의 기준이 되는 시간 복잡도

시간 복잡도 유형

빅 오메가: 최선일 때 (best case)의 연산 횟수를 나타낸 표기법

빅 세타: 보통일 때 (average case)의 연산 횟수를 나타낸 표기법

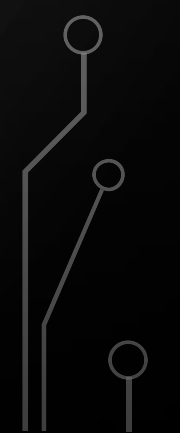

빅 오: 최악일 때 (worst case)의 연산 횟수를 나타낸 표기법

```
int findNumber = (int) (Math.random() * 100);  
for (int i = 0; i < 100; i++) {  
    if (i == findNumber) {  
        System.out.println(i);  
        break;  
    }  
}
```

- 위 예제코드는 빅-오메가 표기법의 시간복잡도는 1번,
빅-세타 표기법은 $2/N$ 번(50번)
빅-오 표기법은 N 번(100번)입니다.



알고리즘에서는 **빅-오 표기법**을 기준으로 코드를 작성하라!



프로그램에서는 다양한 테스트 상황이 있고 모든 케이스에서 문제 없이 실행되는 것을 목표로하기 때문에 최악의 상황을 염두에 두어야 한다.

* 빅-오 표기법의 특징

1. 상수항 무시: 빅-오표기법은 데이터 입력값(N)이 충분히 크다고 가정하고 있고, 알고리즘의 효율성 또한 N 의 크기에 따라 영향을 받기 때문에 상수항같은 사소한 부분은 무시한다.

Ex) $O(3N) \rightarrow O(N)$

2. 영향력 없는 항 무시 : 가장 영향력이 큰 항을 제외하고는 모두 무시한다.

Ex) $O(3N^2 + 2N + 1) \rightarrow O(N^2)$

```
int N = 1000000;  
int cnt = 0;  
for (int i = 0; i < N; i++) {  
    System.out.println("연산 횟수: " + cnt++);  
}
```

* 연산 횟수가 N번인 경우 - $O(N)$

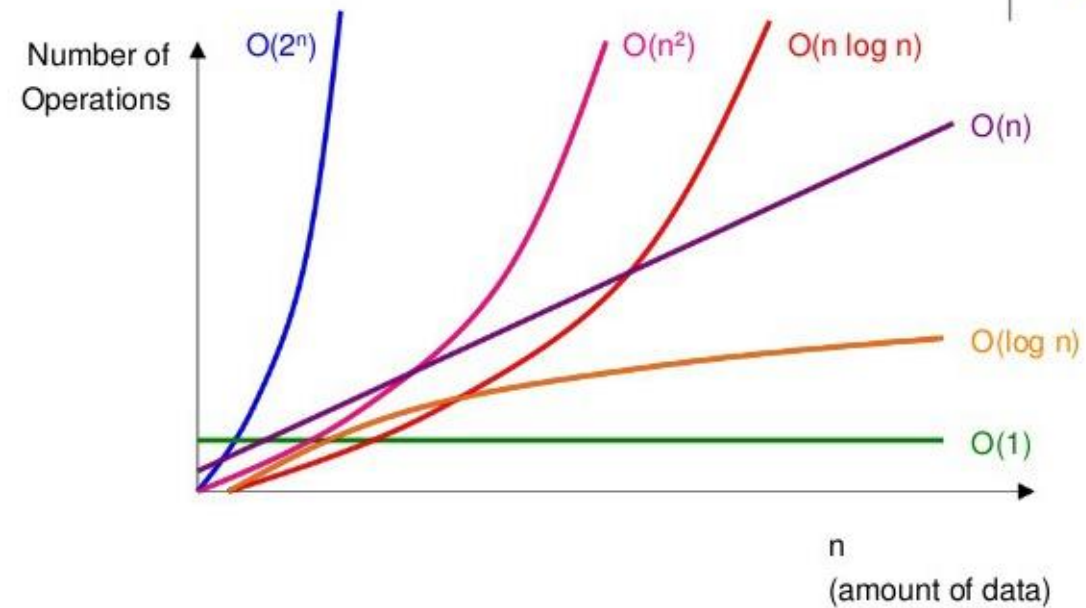
```
int N = 1000000;
int cnt = 0;
for (int i = 0; i < N; i++) {
    System.out.println("연산 횟수: " + cnt++);
}
for (int i = 0; i < N; i++) {
    System.out.println("연산 횟수: " + cnt++);
}
for (int i = 0; i < N; i++) {
    System.out.println("연산 횟수: " + cnt++);
}
```

* 연산 횟수가 3N번인 경우 - $O(N)$

```
int N = 100000;  
int cnt = 0;  
for (int i = 0; i < N; i++) {  
    for (int j = 0; j < N; j++) {  
        System.out.println("연산 횟수: " + cnt++);  
    }  
}
```

* 연산 횟수가 N^2 번인 경우 - $O(N^2)$

Comparing Big O Functions



수 정렬하기

1초 → 1억번



시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
1 초	128 MB	123978	70690	48912	58.409%

문제

N개의 수가 주어졌을 때, 이를 오름차순으로 정렬하는 프로그램을 작성하시오.

입력

첫째 줄에 수의 개수 $N(1 \leq N \leq 1,000)$ 이 주어진다. 둘째 줄부터 N개의 줄에는 수 주어진다. 이 수는 절댓값이 1,000보다 작거나 같은 정수이다. 수는 중복되지 않는다.

출력

첫째 줄부터 N개의 줄에 오름차순으로 정렬한 결과를 한 줄에 하나씩 출력한다.

- 시간 제한이 1초라면 일반적으로 1억번 연산 안에 문제를 해결해야 합니다. 정렬 알고리즘 중에 버블 정렬은 $O(N^2)$ 의 시간복잡도를 가지며 병합 정렬은 $O(n \log n)$ 의 시간복잡도를 가집니다.
- 따라서 위 문제는 버블 정렬 알고리즘으로 문제를 풀 때 1000의 제곱인 1,000,000번의 연산이 일어나며, 병합 정렬로 풀 시 $1000 \log 1000$ 의 값인 3,000번의 연산이 일어납니다.
- 즉, 둘 중 아무 정렬이나 써도 된다는 것입니다. 다만 N의 값이 커질 시 다시 생각해볼 문제입니다.