**자료구조 및 알고리즘 1**

**Term Project 보고서**

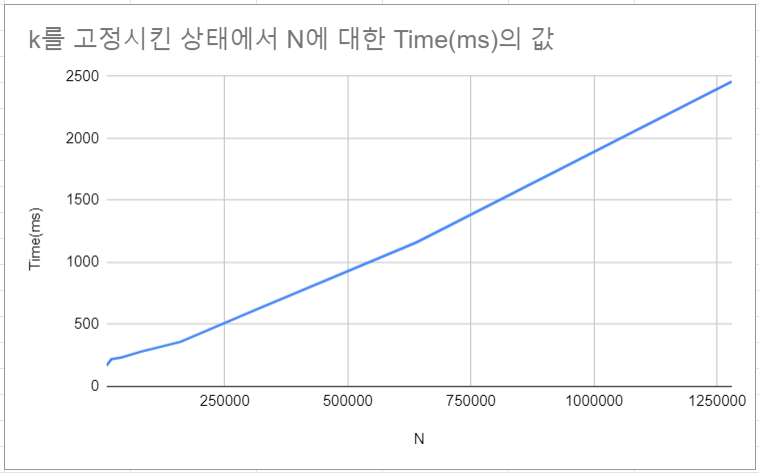
**생명과학과**

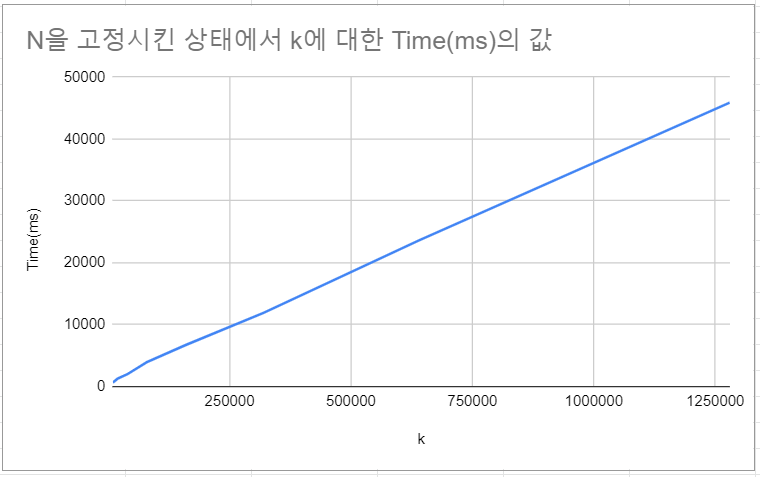
**2018111622**

**이유빈**

1. **K를 고정하고 N을 변화시킨 경우의 그래프**

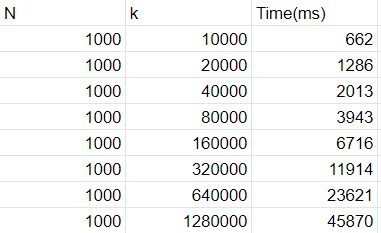
K를 1000으로 고정시킨 상태에서 N을 2배씩 증가시켰을 때, N에대한 Time(ms)의 값을 나타낸 그래프입니다.



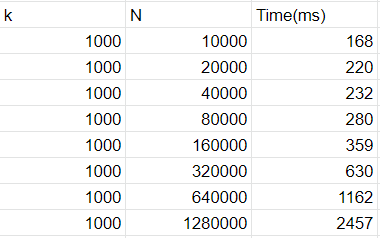
1. **N을 고정하고 K를 변화시킨 경우의 그래프**

N를 1000으로 고정시킨 상태에서 k를 2배씩 증가시켰을 때, k에대한 Time(ms)의 값을 나타낸 그래프입니다.

1. **알고리즘 성능 평가**



<Table 2 – N을 고정하고, k를 변화시킨 경우>



<Table 1 – k를 고정하고, N을 변화시킨 경우>

우선 k를 1000으로 고정시킨 상태에서 N을 2배씩 증가시킨 경우를 살펴보면, N에 대한 Time 값 역시 상대적으로 비슷하게 2배씩 증가하였습니다. 데이터의 수를 작게 기입하였을 때, 즉 N의 값이 작을 때에는 N에 대한 Time 값의 증가율이 일정하지 않았습니다. 그러나 해당 표를 그래표로 나타낸 것을 보면 알 수 있듯이, 전체를 봤을 때 N에 대해 Time 값의 증가율이 대체적으로 N에 비례하였습니다. 따라서 k를 고정시키고 N을 변화시키는 해당 알고리즘의 수행시간은 O(N)이 됩니다. 이는 소스를 통해서도 알 수 있었습니다. N개의 데이터를 랜덤으로 생성하는 부분은 for문을 N번 돌리기 때문에 해당 코드 수행시간은 N에 비례합니다. 또한 이 경우는 K를 고정시켰기 때문에, k번만큼 for문을 돌려 k개의 구간을 랜덤으로 생성하는 것은 상수시간 안에서 실행됩니다. 그리고 구간의 최솟값, 최댓값, 합계를 구하는 부분의 알고리즘에서, 첫번째 for문은 고정된 수인 k번 실행되고, 랜덤으로 만들어진 구간의 시작값과 종료값의 범위는 최소 0부터 최대 N-1이라서 중첩된 두번째 for문은 최대 N번 돌아갈 수 있기 때문에 이 부분의 수행시간은 N에 비례하기 됩니다. 따라서 k를 고정시킨 경우의 전체적인 알고리즘의 수행시간은 O(N)인 것입니다.

다음으로 N을 1000으로 고정시킨 상태에서 k를 2배씩 증가시킨 경우를 살펴보면, 마찬가지로 k에 대한 Time값은 약 2배씩 증가하였습니다. 해당 표를 그래표로 나타낸 것을 보면, Time 값이 k에 비례하여 증가하는 것을 확인할 수 있었습니다. 따라서 N을 고정시키고 k를 변화시키는 알고리즘의 수행시간은 O(k)가 됩니다. 이는 해당 부분의 소스를 통해서도 알 수 있었습니다. 우선 N을 고정시킨 상태이므로, N개의 데이터를 랜덤으로 생성하여 배열에 저장하는 부분은 상수시간 안에서 실행됩니다. 그리고 k개의 구간을 랜덤으로 생성하여 배열에 저장하는 부분은 k번만큼 for문을 돌리기 때문에 수행시간이 k에 비례합니다. 또한 구간의 최솟값, 최댓값, 합계를 구하는 부분의 알고리즘에서는, 첫번째 for문은 k번 실행되고, 랜덤으로 만들어진 구간의 시작값과 종료값의 범위는 0부터 고정된 수인 N-1로 항상 고정되어있어 중첩된 두번째 for문은 상수시간안에서 실행되기 때문에 해당 부분의 수행시간은 k에 비례합니다. 따라서 N을 고정시킨 경우의 전체적인 알고리즘의 수행시간은 O(k)인 것입니다.

위의 두 경우에서 구한 수행시간 두 가지를 곱해준 것이 이 전체 알고리즘의 성능을 평가하는 수행시간이 됩니다. 따라서 두 개를 곱한 값인 O(NK)가 해당 전체 알고리즘의 수행시간이 됩니다.