**컴퓨터공학과 2학년 20171697 최민영**

**CSE3081-1: Design and Analysis of Algorithms**

**Machine Problem 1: Maximum Subsequence Sum**

**Report**

* Experiment environment:

(노트북 사양)

CPU: intel(R) Core(TM) i5-6300U 2.40GHz 2.50 GHz

RAM: 8.00GB

System: 64비트 운영 체제

(Oracle VM virtualbox 사양)

OS: Ubuntu(64-bit)

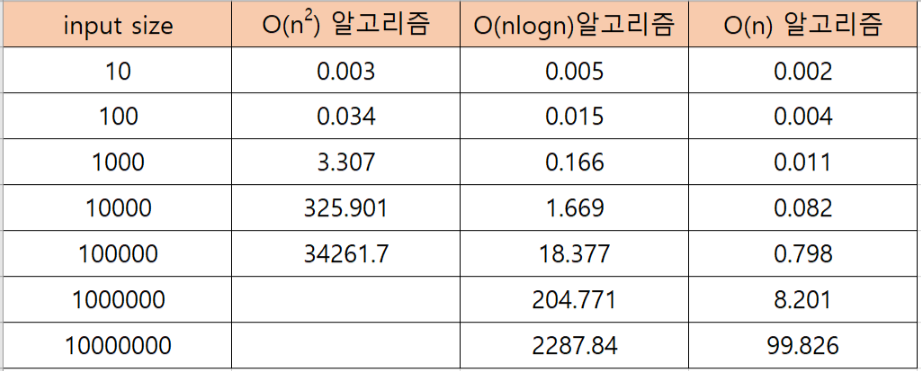
System: (base memory) 1024 MB / (Boot order) Floppy, Optical, Hard Disk

VM을 사용하여 모든 실험을 하였다.

* Experiment setup:

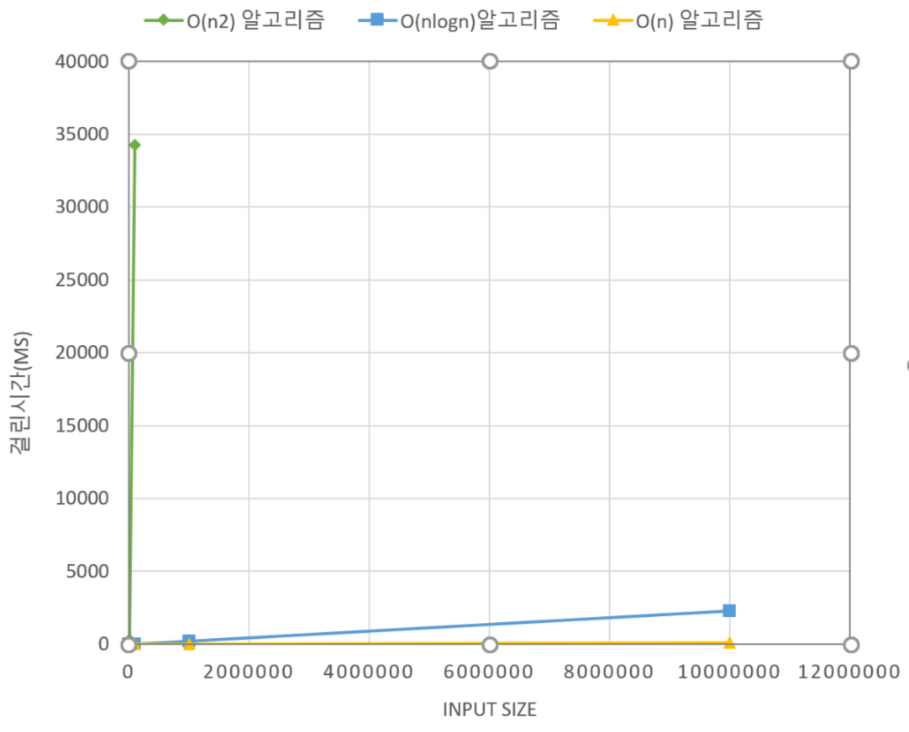
최대 연속된 부분합을 찾을 숫자 리스트의 크기를 달리하면서 O(n2), O(nlogn), O(n)의 시간복잡도를 갖는 알고리즘들의 수행시간을 측정하였다. 숫자 리스트의 크기를 키워가면서 각 알고리즘 수행시간의 증가폭을 살펴볼 수 있었다. 또한 같은 숫자 리스트의 크기에 대한 알고리즘 수행시간의 차이를 살펴볼 수 있었고 이 차이가 숫자 리스트의 크기가 커짐에 따라 어떻게 되는 지 살펴볼 수도 있었다. 숫자 리스트의 크기는 최소 8개에서 최대 10,000,000개일 때까지로 실험해보았다. 그리고 모든 숫자의 합이 int형이 벗어나지 않도록 적절히 입력파일은 만들어서 실험해보았다.

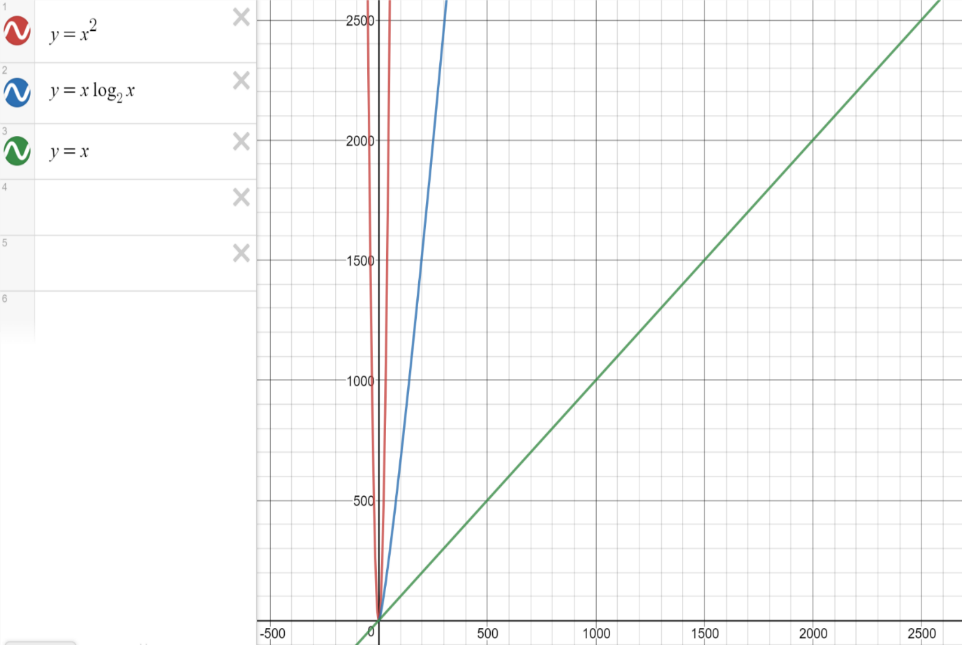
* Comments on the experiment:



10개일 때는 O(nlogn)이 O(n2)보다 더 오래걸렸지만 개수가 늘어날수록 O(n2)알고리즘의 수행시간이 기하급수적으로 늘어나면서 다른 알고리즘과의 차이가 크게 벌어졌다. 10개일 때만 제외하고는 O(n2), O(nlogn), O(n) 순으로 수행시간이 오래걸렸다. 즉, 이론적인 예상과 실험이 일치했다.

개수 증가에 따른 각각의 알고리즘의 증가폭 또한 예상과 일치했다. O(n2)에 경우에는 이론적으로 개수가 10배 늘어날수록 수행시간이 102배로 증가할 것이라고 예상할 수 있다. 표를 살펴보면, 10개일 때 0.003ms, 100개일 때 0.034ms ≒ 0.003 \* 100, 1000개일 때 3.307ms ≒ 0.034 \* 100 임을 알 수 있었다. O(nlog2n) 경우에는 이론적으로 개수가 10배 늘어날수록 수행시간이 10배보다 조금 더 크게( 10n(log2n + log210) ) 증가할 것이라고 예상할 수 있다. 또한 O(n) 경우에는 이론적으로 개수가 10배 늘어날수록 수행시간이 10배로 증가할 것이라고 예상할 수 있다. 표를 보면 이론과 거의 일치함을 알 수 있었다.

아래는 실험결과로 그린 그래프와 이론적에 따른 그래프를 비교해보았다.





위 표는 숫자 리스트의 크기가 작을 때를 살펴본 결과이다. 그 결과 크기가 35이기 전짜기는 O(n2)의 수행시간이 O(nlogn) 수행시간보다 적게 걸렸다. 이는 이론적인 예상과 벗어나는 결과였다. 나의 생각으로는 크기가 작을 때는 O(nlogn)이 O(n2)와 달리 재귀함수로 함수를 수행하는 시간보다 함수를 호출하고 다시 돌아오는 데 걸리는 시간이 수행시간에 더 크게 영향을 준 것 같다. 35개보다 적을 때도 O(n2)과 O(n)의 수행시간 차이와 O(nlogn)과 O(n)의 수행시간 차이는 확연히 보였고 35개보다 크기가 커지면, 수행시간이 큰 순서로 O(n2), O(nlogn), O(n)을 이루어 이론과 알맞는 결과를 보여준다.

