**알고리즘 과제 1 보고서**

20140132 조민호

- Experiment environment

아래와 같은

CPU : 2.3 GHz Intel Core i5

Memory : 8GB 2133 MHz LPDDR3

OS type version : macOS mojave 10.14.6

같은 환경에서 setup 해서 실험했다.

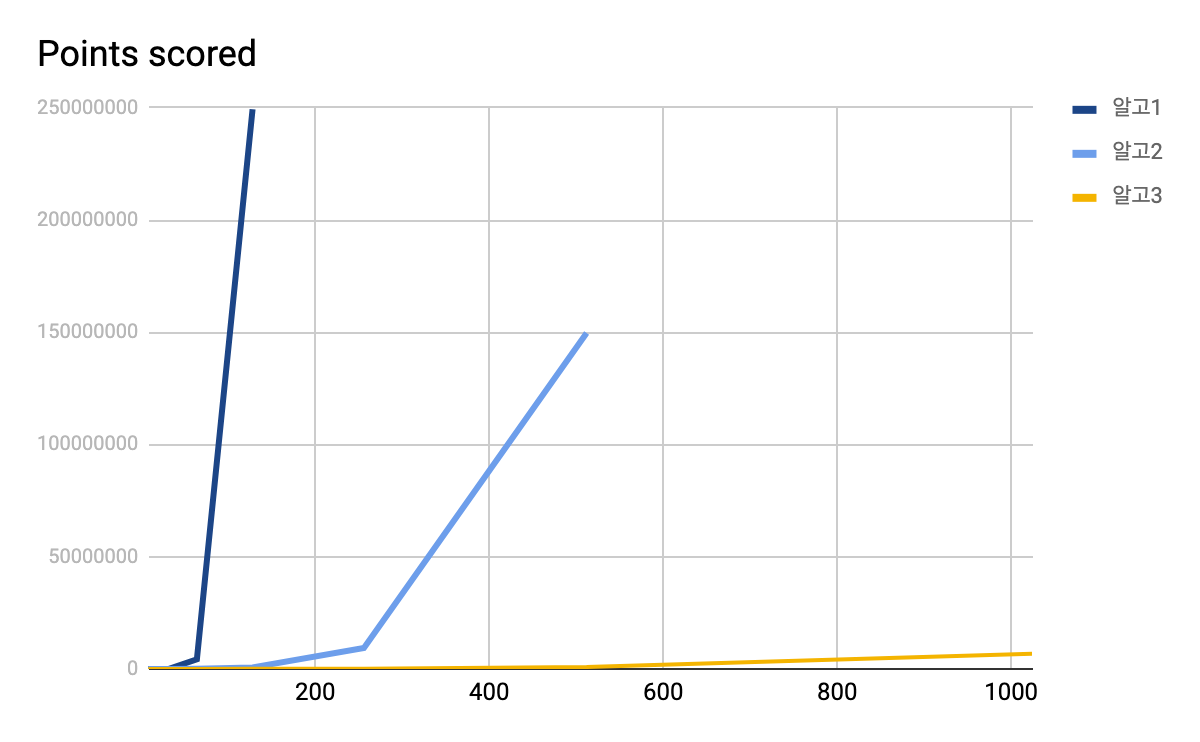
- Experiment Setup

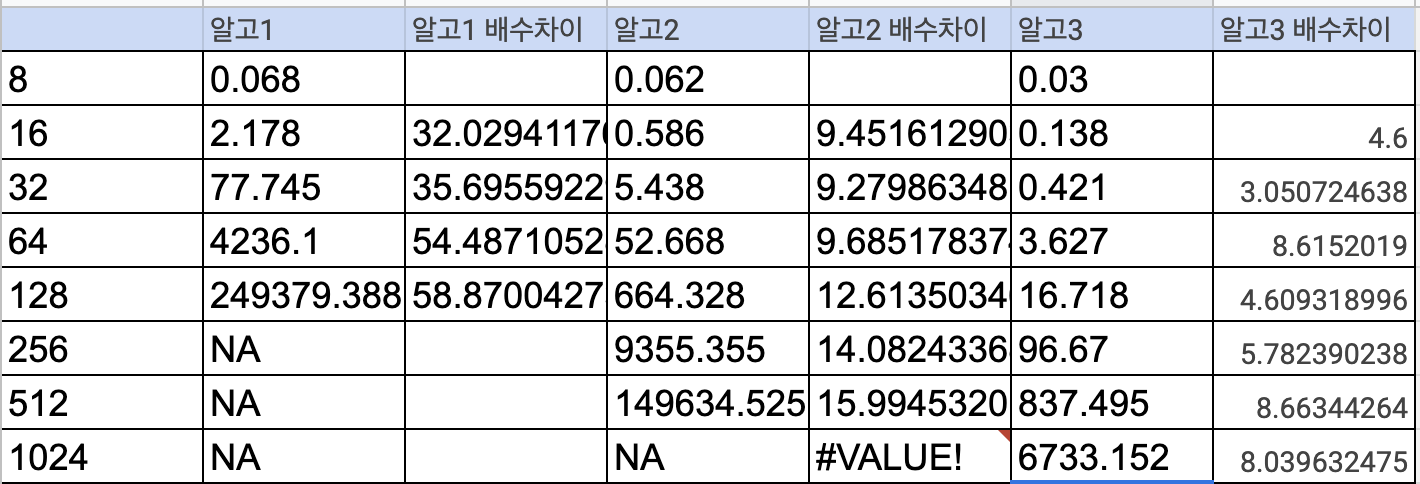
배수로 쉽게 계산하기 위하여 가로와 세로가 같은 square를 사용했다. input은 -20과 20사이의 수이다. size는 가로, 세로가 10, 100, 1000, 10000을 테스트 해보았다. input별로 시간을 나타내면(단위 ms) 아래와 같았다. 데이터가 커지면 커질수록 복잡도가 폭발적으로 커지기 때문에 1000개를 넘어가니 컴퓨터가 실행하지 못했다.

|  |  |  |  |
| --- | --- | --- | --- |
|  | 알고리즘(1) n^6 | 알고리즘(2) n^4 | 알고리즘(3) n^3 |
| 10개 | 0.183 | 0.118 | 0.045 |
| 100개 | 58466.184 | 265.081 | 8.486 |
| 1000개 | NA | NA | 5733.052 |
| 10000개 | NA | NA | NA |

때문에 조금 더 배수를 줄여서 변화를 살펴보기 위해 2의 배수로 테스트 해보았다

|  |  |  |  |
| --- | --- | --- | --- |
|  | 알고리즘(1) n^6 | 알고리즘(2) n^4 | 알고리즘(3) n^3 |
| 8 | 0.068 | 0.062 | 0.030 |
| 16 | 2.178 | 0.586 | 0.138 |
| 32 | 77.745 | 5.438 | 0.421 |
| 64 | 4236.100 | 52.668 | 3.627 |
| 128 | 249379.388 | 664.328 | 16.718 |
| 256 | NA | 9355.355 | 96.670 |
| 512 | NA | 149634.525 | 837.495 |
| 1024 | NA | NA | 6733.152 |



이를 두배씩 커지는 배수차이를 확인해보면, 

알고리즘 1은 O(n^6)을 가진다.(for loop 6 depth) 은 두배씩 값이 커질 때 2^6=64에 가까워 질 것이다. 실제 결과값을 보면 58로 점점 64에 가까워 진다. 다만 100 이상만 되어도 loop가 1e+12 만큼이나 커지므로 더 이상은 돌아가지 않았다.

알고리즘 2는 O(n^4)을 가진다.(for loop 4 depth) 은 두배씩 값이 커질 때 2^4=16에 가까워 질 것이다. 실제 결과값을 보면 15.99로 점점 16에 가까워 진다. 다만 1000 이상이 되면 또한 1e+12가 되므로 더이상은 돌아가지 않았다.

알고리즘 3은 O(n^3)을 가진다.(for loop 3 depth) 은 두배씩 값이 커질 때 2^3=8에 가까워 질 것이다. 실제 결과값을 보면 8.03로 점점 8에 가까워 진다.

- Your comments on the experience

우리의 input 데이터 이외에 O(n^6)을 구현할 때에는 array가 아닌 단순한 this\_sum, max\_sum만이 필요했다. 하지만 O(n^4)에 가면서 new\_col\_sum, col\_sum의 1차원 array가 필요했다. 또 O(n^3)에 가면서는 col\_sum을 일차원 array가 아닌 2차원 array로 증량할 필요가 있었다. 이렇게 time complexity와 space complexity 두개가 한꺼번에 해결되기는 힘들고 하나가 해결되면 하나가 커지는 방식으로 자주 일어나는 것 같았다. 이는 space complexity를 높임으로써 그 이전의 상태를 기억할 수 있게 되기 때문일 것이다.

또 알고리즘을 돌려보면서 1e+12 의 값이 넘어가면 일반적인 pc 컴퓨터가 돌릴 수 없는 정도의 값이 되는구나를 알게 되었다.