알고리즘 설계와 분석(CSE3081/2 반) HW1

2020.10.6 (화) / 20181202 김수미

1. 컴퓨터 실험 환경

OS : Windows 10 Home (맥북 Parallels)

CPU : Intel(R) Core(TM) i5-7Y54 CPU @ 1.20GHz

RAM : 4.00GB

Compiler : Microsoft Visual Studio Community 2017 15.9.9

2. 알고리즘 구현 방법에 대한 간단한 설명

1) Algorithm3 : Summed Area Table/ $O(n^4)$

Summed Area Table 을 이용해 Maximum Sum Subrectangle 을 구하기 위해 두가지 함수를 구현했다.

하나는 위-왼쪽 모서리의 인덱스부터 아래-오른쪽 모서리의 인덱스까지로 이루어지는 Subrectangle 의 Sum 을 구하는 함수이고, 다른 하나는 모든 Subrectangle 의 Sum 값을 비교하며 최대가 되는 값을 찾는 함수이다. 두 함수는 각각 $O(1)$, $O(n^4)$ 의 시간복잡도를 가지기 때문에 해당 알고리즘의 전체 시간복잡도는 $O(n^4)$ 이 된다.

2) Algorithm4 : Divide and Conquer / $O(n^3 logn)$

강의자료에 기술된 방법을 활용하여 해당 알고리즘을 구현했다. MSS(2D)의 해당 열은 어디이건 i 에서 j 임을 이용해서, 가능한 모든 (i, j) 조합에 대해 MSS(1D)를 Divide and Conquer 알고리즘을 사용하여 찾는 것이다. 가능한 모든 (i, j) 조합을 찾는 시간복잡도가 $O(n^2)$ 이므로 전체 시간복잡도는 Divide and Conquer 알고리즘의 시간복잡도인 $O(\mathrm{n}logn)$ 과 $O(n^2)$ 를 곱한 $O(n^3 logn)$ 이 된다.

3) Algorithm5 : Kadane's Algorithm / $O(n^3)$

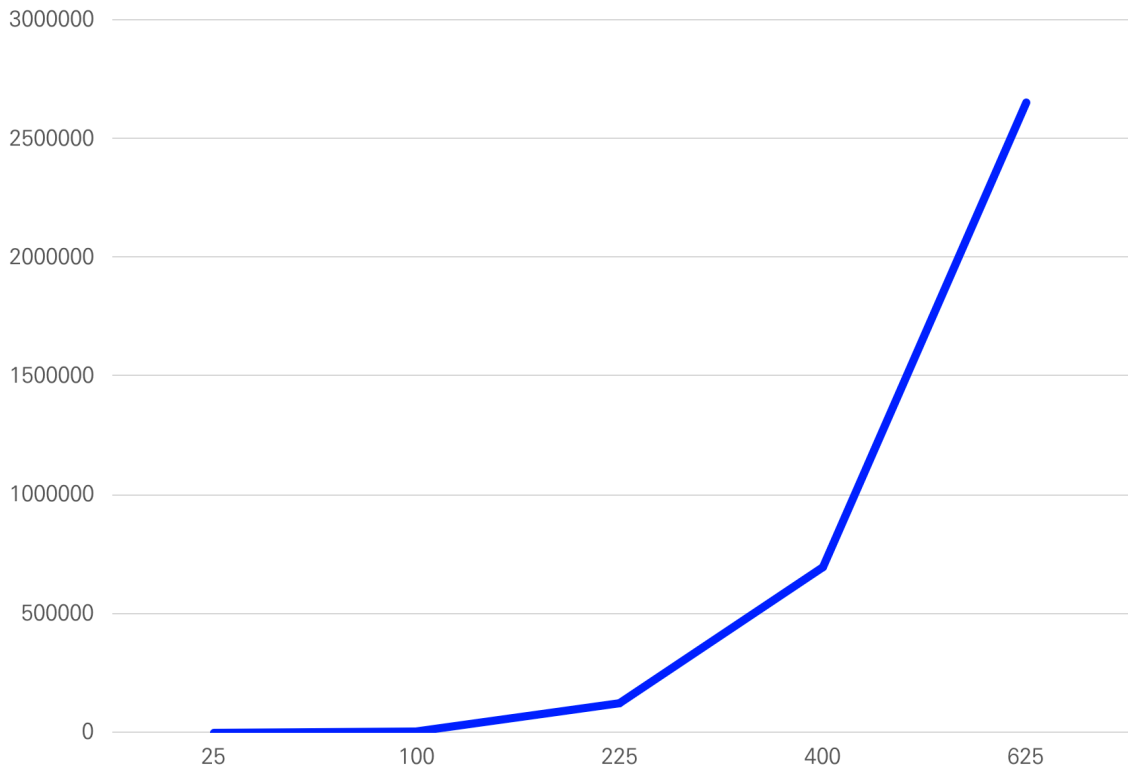
4 번 알고리즘과 전체적인 구성은 동일하며, MSS(1D)를 Kadane 알고리즘을 사용하여 찾는다는 점에서 차이가 있다. 따라서 전체 시간복잡도는 Kadane 알고리즘의 시간복잡도인 $O(n^2)$과 능한 모든 (i, j) 조합을 찾는 시간복잡도 $O(n^2)$ 를 곱한 $O(n^4)$ 이 된다.

3. 알고리즘 별 수행 시간 측정 결과 및 분석

N 의 크기는 결과를 확인하는데 까지 걸리는 시간이 너무 길어지지 않도록 하는 범위 내에서 선택했고, 2 차원 배열이 N*N 배열이기 때문에 모든 N 은 제곱수로 설정되었다. 측정한 시간의 오차를 줄여 실험 결과의 신빙성을 높이기 위해 동일한 n 에 대해 각 5 개씩의 서로 다른 데이터에 대한 시간을 측정했고, 하나의 동일한 데이터에 대해서도 각각 5 번씩 결과를 측정해 나온 값들의 평균을 최종 시간 측정값으로

사용했다. 측정 결과를 한눈에 확인하기 위해 표와 그래프로 나타냈으며 시간 표기 단위는 ms(밀리초)이다. (컴퓨터 실험 환경이 좋은 편은 아니기에, 소요시간이 N 의 크기 변화에 정비례 하지는 않는 듯하며 충분히 큰 N 값에 대해서도, 타 보고서에 비해 N 값이 상대적으로 작을 수 있다.)

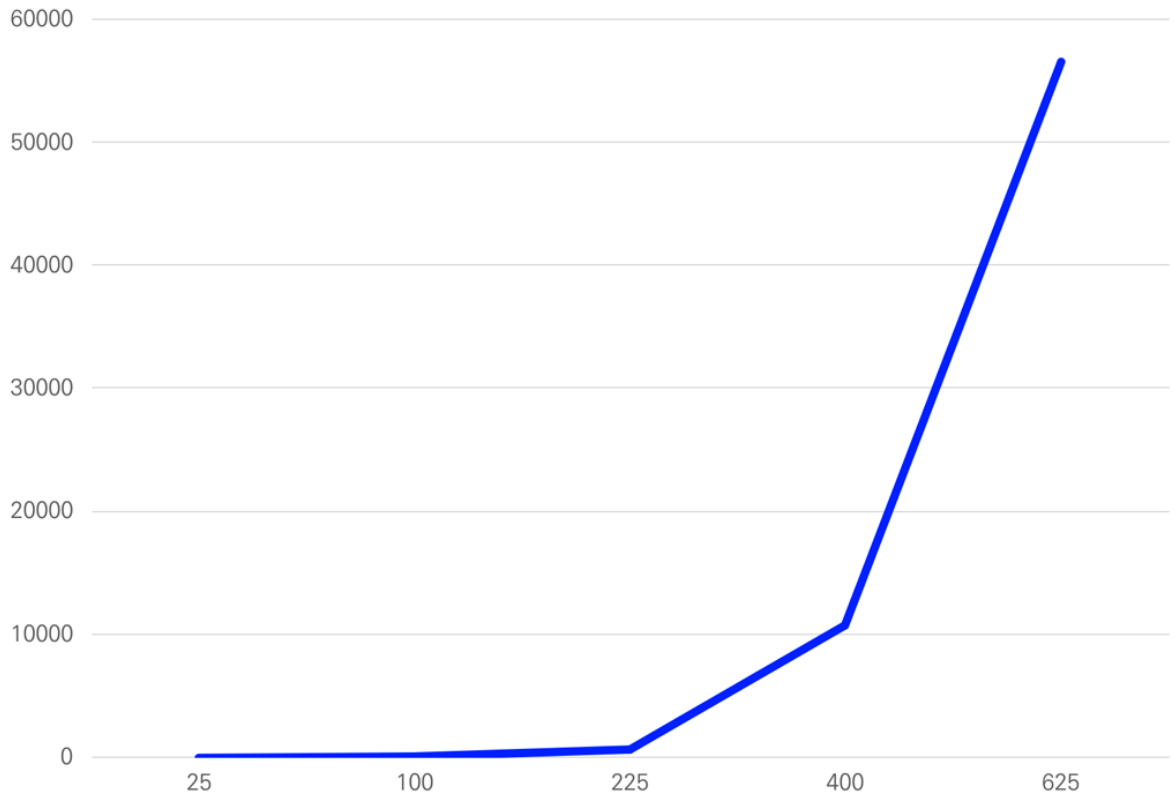
1) Algorithm3 : Summed Area Table / $O(n^4)$



Algorithm3 시간 측정 결과 (가로축 : N, 세로축 : time)

(time=ms)	N=25	N=100	N=225	N=400	N=625
Case1	11	909	103531	339722	2693870
Case2	9	877	127654	653588	2948632
Case3	8	835	114987	713206	2349187
Case4	11	877	113081	985154	2677141
Case5	22	897	149786	767527	2598912
평균	12.2	879	121807.8	6972.6	2653548.4

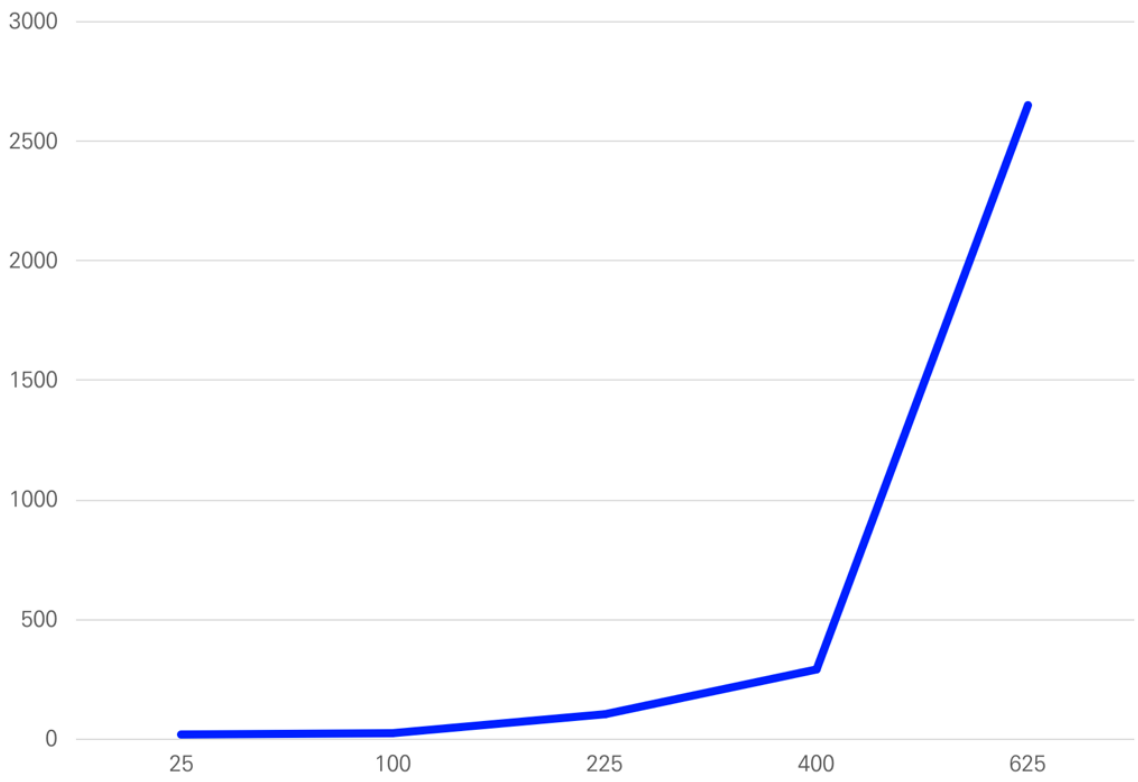
2) Algorithm4 : Divide and Conquer / $O(n^3 logn)$



Algorithm4 시간 측정 결과 (가로축 : N, 세로축 : time)

(time=ms)	N=25	N=100	N=225	N=400	N=625
Case1	15	6	550	3408	14862
Case2	5	65	556	10826	89668
Case3	42	61	553	12265	67585
Case4	12	119	621	14345	65390
Case5	20	74	596	12651	45148
평균	18.8	65	575.2	10699	56530.6

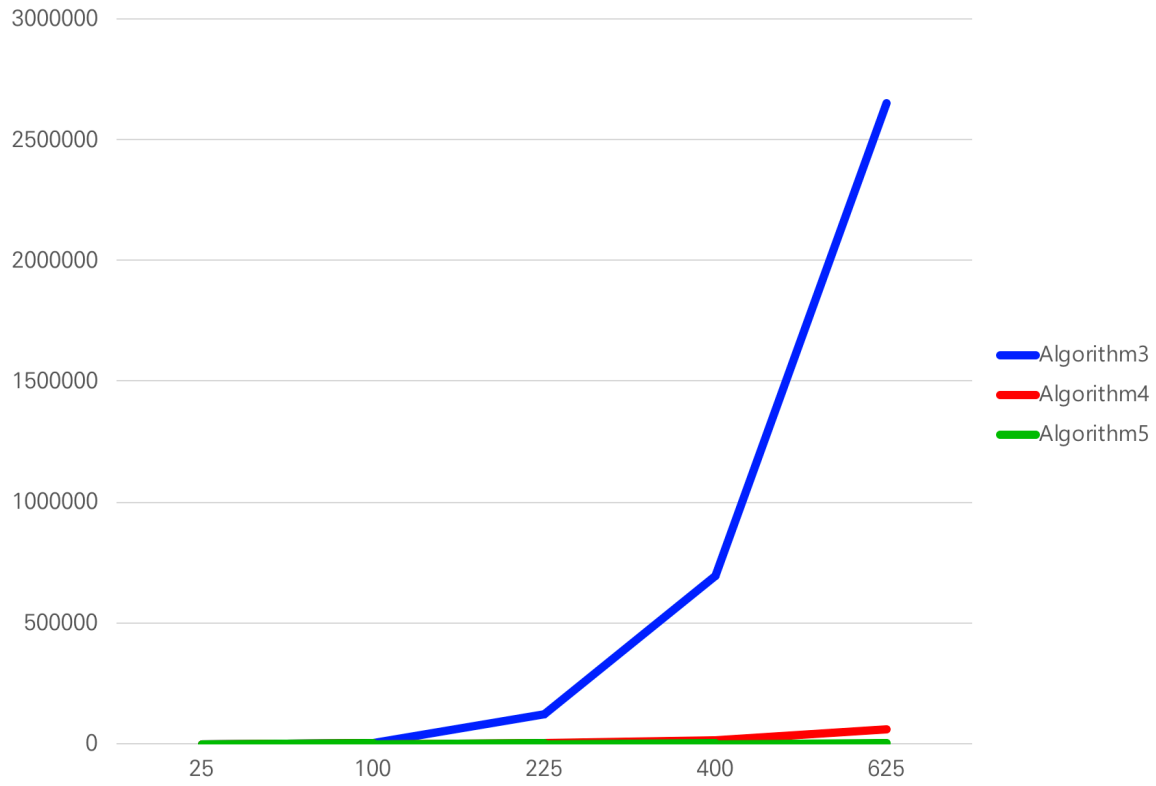
3) Algorithm5 : Kadane's Algorithm / $O(n^3)$



Algorithm5 시간 측정 결과 (가로축 : N, 세로축 : time)

(time=ms)	N=25	N=100	N=225	N=400	N=625
Case1	10	18	82	323	1498
Case2	17	22	194	309	1589
Case3	12	25	96	270	2245
Case4	13	26	72	252	5668
Case5	42	35	70	293	2255
평균	18.8	25.2	102.8	289.4	2651

4) 세 알고리즘 실행 시간 비교 및 분석



입력 데이터 크기에 따른 시간 측정 결과 (가로축 : N, 세로축 : time)

(time=ms)	N=25	N=100	N=225	N=400	N=625
Algorithm3 ($O(n^4)$)	12.2	879	121807.8	6972.6	2653548.4
Algorithm4 ($O(n^3 logn)$)	18.8	65	575.2	10699	56530.6
Algorithm5 ($O(n^3)$)	18.8	25.2	102.8	289.4	2651

각각 $O(n^4)$, $O(n^3 logn), O(n^3)$ 의 시간복잡도를 가지는 세 알고리즘의 수행 시간을 측정 후 비교해 보았다. 그래프를 통해서 알 수 있듯이 데이터의 크기가 커질수록, 세 알고리즘의 수행 시간 사이의 간극이 커짐을 그래프를 통해 쉽게 확인할 수 있다. 즉, 충분히 큰 입력데이터에 대해서 세 알고리즘은 $O(n^3)$, $O(n^3 logn)$, $O(n^4)$ 의 시간복잡도를 가지는 순으로 효율적이라고 이야기할 수 있다($O(n^3)$ 이 가장 효율적, $O(n^4)$ 이 가장 비효율적).

4. 실험을 통해 발견한 사실 및 느낀 점

먼저 이론적으로만 살펴보았던 시간복잡도에 관한 개념을 직접 실험을 통해 체감하고 도식화 해 볼 수 있어 좋았다. 특히 데이터 크기가 작을 때는 Algorithm4 나 5 보다도 Algorithm3 이 빠르게 계산되는 경우가 있었지만, 데이터 크기가 커질수록 Algorithm4 나 5 가 훨씬 빠른 시간 내에 계산을 수행하는 것을 확인함으로써 코드를 작성할 때 효율적인 시간복잡도를 가지는 알고리즘을 선택하는 것이 중요하다는 것을 느낄 수 있었다.