**CSE3081-2: Design and Analysis of Algorithms (Fall 2019)**

**Machine Problem 1: Maximum Sum Subrectangle in a 2D array**

학과 : 컴퓨터공학 학번 : 20181668 이름 : 이예진

**1. Experiment environment**

\*The hardware specification of the machine

-CPU speed : 1.80GHz (8CPUs), ~2.0GHz

-Memory size : 8192MB RAM (8GB)

-OS type and version : Windows 10 Home 64비트 (10.0, 빌드 18362)

-개인 노트북(LG전자 올뉴그램 15Z980-GA70K, 코어i7-8550U)으로 모든 실험을 진행했다.

-Linux 서버에서 개인 계정으로 모든 실험을 진행했다.

**2. Experiment setup**

(1) 변수

-max\_subrect라는 사용자 정의 구조체를 선언하여 이용하였다. 이 구조체는 최대값의 정보를 가지는 변수 max와 각각 행, 열의 정보를 가지는 변수 row, col이 있고 2차원 배열에 해당하는 정보를 가지는 2차원 배열 변수 rect가 존재한다. 구조체에 존재하는 모든 변수는 정수형의 자료형을 가진다. 2차원 배열 변수 rect는 동적 할당을 이용하여 공간 낭비를 줄였다.

-algorithm\_2라는 함수에서 Save\_sum이라는 1차원 배열 변수를 생성하여 사용하였다. 이 변수는 입력 받은 2차원 배열의 column 개수만큼의 원소를 갖는다. 처음에는 0으로 초기화되어 있는데 각 column index에 해당되는 값들이 column이 증가하는 방향을 따라 더해진다.

-algorithm\_3라는 함수에서 Row\_sum이라는 1차원 배열 변수를 생성하여 사용하였다. 이 변수는 입력 받은 2차원 배열의 row 개수만큼의 원소를 갖는다. 처음에는 0으로 초기화되어 있는데 각 row index에 해당되는 값들이 index에 해당되는 위치에 더해진다.

(2) 함수

-open\_input이라는 함수를 이용해서 input 파일을 열어 안에 존재하는 정보를 읽고 구조체 안의 적절한 변수에 저장하도록 했다. 우선, 2차원 배열의 첫 번째 원소를 최대 값으로 저장했다. 2차원 배열의 원소가 가지는 값의 범위가 모두 음수 또는 양수인지, 0을 포함하는지 모르기 때문이다. 입력 파일을 읽는 함수를 따로 만들어서 알고리즘 수행 시간에 입력 파일을 읽는 시간을 포함시키지 않을 수 있었고, 따라서 더 정확하게 시간을 측정할 수 있었다.

-algorithm\_1, algorithm\_2, algorithm\_3이라는 함수들을 만들어 각각 번호에 해당하는 알고리즘을 구현했다. algorithm\_1 함수에서는 2차원 배열의 모든 시작점과 끝점의 경우를 고려하여 가능한 직사각형 범위의 합을 모두 구하는 알고리즘을 구현하였다. algorithm\_2 함수에서는 column 개수만큼의 1차원 배열을 선언하여 같은 column에 존재하는 원소들을 더했다. 더한 다음 1차원 배열 내에서 연속되는 합의 모든 경우를 고려하여 최대값을 찾도록 했다. algorithm\_3 함수에서는 row 개수만큼의 1차원 배열을 선언하여 사용했다. 입력 받은 2차원 배열의 row 위치를 index로 하여 원소들을 1차원 배열에 더했다. 1차원 배열의 index를 증가시키며 값을 더해서 연속되는 합을 계산하였고, 최대값을 찾도록 했다.

(4) 단위

-알고리즘 시간 측정

알고리즘을 끝낸 시간을 저장한 변수 finish\_t에서 알고리즘을 시작한 시간을 나타내는 변수 start\_t를 뺀 다음 1000을 곱하고 CLOCKS\_PER\_SEC로 나누어 milliseconds 단위로 나타냈다. double형으로 형변환을 한 다음 출력하도록 했다.

(3) The range of input size

-input size의 범위는 기본적으로 정수 범위가 되도록 int형 변수를 이용했다. input 값을 저장할 행, 열에 대한 변수와 2차원 배열은 모두 정수형 변수를 이용하도록 만들었다.

-처음에는 과제 자료에 주어진 4X5크기의 input을 이용했다.

-그래프를 그리기 위해 input size를 1X1, 2X2, 5X5, 10X10, 25X25, 50X50, 100X100, 150X150로 정하여 실험을 수행하였다. 각 input의 범위는 -row ~ +row으로 하였다. 그래프의 편의를 위해 정사각형 모양의 2차원 배열을 input으로 정했고 테스트는 다양한 크기로 진행하였다.

(input00001.txt~input00008.txt)

-모든 원소가 0인 10X10 input을 이용했다. (input00009.txt)

-모든 원소가 음수인 10X10, 50X25 input을 이용했다. (input00010txt~input00011.txt)

-input 파일을 생성할 때는 교수님이 제공하신 프로그램을 이용했다.

![텍스트이(가) 표시된 사진

자동 생성된 설명]()![스크린샷, 텍스트이(가) 표시된 사진

자동 생성된 설명]()![텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명]()![텍스트이(가) 표시된 사진

자동 생성된 설명]()

**3. 실험**

(1) 실험 목적

동일한 문제에 대하여 동일한 결과를 생성하는 3가지의 서로 다른 알고리즘들이 어떻게 다르게 작동하는 지를 보인다. 3가지의 서로 다른 알고리즘들은 각각 O(n6), O(n4), O(n3)에 해당하는 시간 복잡도를 가진다. Maximum Sum Subrectangle in a 2D array 알고리즘의 수행 시간은 입력의 크기가 커질수록 늘어나는데, 3가지의 알고리즘들은 각각 다른 수행 시간 변화량을 가진다.

(2) 실험 결과

-수행 시간의 단위는 milliseconds이다.





-모든 알고리즘은 입력된 배열의 크기가 증가할수록 알고리즘 수행 시간이 증가하였다. 입력된 배열의 크기가 매우 작은 경우에는 시간 복잡도가 역전되는 상황도 존재했다. 입력 크기가 작은 경우를 제외하면 같은 입력에 대하여 O(n6) > O(n4) > O(n3) 순서로 수행 시간이 큰 것으로 결과가 나왔다. 입력 크기가 커질수록 알고리즘 간의 수행 시간 차이도 증가하는 것으로 나타났다.

-최대값은 입력 파일 순서대로 0, 3, 12, 118, 528, 1771, 11448, 12795로, 3개의 알고리즘이 모두 같은 결과값을 가졌다.



-input00009.txt는 모든 원소가 0인 경우, input00010.txt와 input00011.txt는 모든 원소가 음수인 경우의 입력 파일을 만들어 실험을 진행하였다. 위의 그래프와 같이 같은 입력에 대하여 O(n6) >= O(n4) >= O(n3) 순서로 수행 시간이 큰 것으로 결과가 나왔다. 3개의 알고리즘 모두 같은 최대값이 나왔다. (입력 파일 순서대로 0, -1, -2가 최대값으로 나왔다.)

**4. Your comments on the experience**

입력 받을 2차원 배열의 크기가 클수록 시간 복잡도가 작은 알고리즘이 효율적이라는 점을 실험을 통해 알 수 있었다. 실험을 참고하면, 입력의 크기가 클수록 알고리즘 간의 수행 시간 차이가 극명하게 드러나는 것을 볼 수 있다. 실험 결과를 표로 정리하고, 표를 이용하여 그래프를 그려서 실험 결과를 시각화 했는데, 이는 실험 결과를 이해하는 데 도움을 주었다. 2차원 배열의 크기가 매우 작으면 알고리즘을 수행하는 데 걸리는 시간이, 시간 복잡도가 큰 알고리즘이 작은 알고리즘보다 적게 걸리는 경우도 존재했다. 처음에는 실험 결과가 잘못된 것으로 판단했는데, 계산을 토대로 그래프를 그려보는 과정을 통해 값이 역전되는 상황이 존재함을 알게 되었다.