알고리즘 실습 보고서

-D&C2-

전공 :컴퓨터공학과

분반 : 05반

학번 :201701988

이름 :김수빈

1. 실행 환경

본 실습은 Windows10 64bit, jdk 1.8.0_221, Eclipse EE가 설치된 환경에서 실행되었다

2. 과제 설명

- 출제된 과제에 대한 설명

첫 번째 과제는 data03_inversion.txt 파일에서 숫자를 읽어 Divide & Conquer 방식을 사용하여 순서가 바뀐 inversion의 수를 세어 콘솔창에 출력하는 프로그램 구현이고, 두 번째는 data03_closest.txt 파일에서 좌표 값을 읽어 Divide & Conquer 방식을 사용하여 가장 거리가 가까운 쌍을 구해 그 거리를 콘솔창에 출력하는 프로그램 구현이다.

3. 문제 해결 방법

- 문제를 해결하기 위해 자신이 사용한 방법, 아이디어에 대한 설명

hw04_DC2 클래스에서 main이 실행되며, 실행되면 각 text 파일을 finding_closest_pair 클래스의 finding 메소드, inversion_counting 클래스의 counting 메소드에 매개변수로 넘겨 호출한다.

먼저, 첫 번째 과제를 구현하기 위해, inversionsList 클래스를 생성하였다. inversionsList는 배열과 inversion 개수를 저장하는 count 값을 갖는다.

counting 메소드는 inversion counting을 수행하는 메소드다. 받은 파일을 읽어 숫자를 배열로 저장하고, sort_count 메소드를 호출하여 결과 값을 받아와 Input Data와 Output Data를 콘솔창에 출력한다.

sort_count 메소드는 inversionsList list를 매개변수로 받아 배열의 길이가 1이면 list의 count를 0으로 하여 반환한다. 아닐 시, list의 배열을 반으로 나누어 새로운 inversionsList를 2개 생성한다. 각 inversionsList에 대해 sort_count를 재귀적으로 호출하여 각각의 inversion count를 갖고 오도록 한다. 반환받은 두 개의 inversionsList를 merge_sort를 통해 합치고, count 값을 모두 더해 list의 count에 저장한다.

merge_sort 메소드는 두 개의 inversionsList를 받아 합치는 메소드이다. 반환할 inversionsList의 배열은 두 개의 inversionsList의 배열의 길이를 합친 길이를 갖는다.inversion count값을 셀 변수와 두 배열의 비교할 값의 인덱스를 가리킬 변수들이 필요하다. while문을 통해 두 개의 배열의 값을 비교하여 반환할 inversionsList의 배열에 삽입한다. 이때, 매개변수 Alist와 Blist에 대하여, Blist의 배열원소가 Alist 보다 작거나 같을 경우, inversion이 발생한 것이며, inversion count 값을 Alist의 남은 원소들만큼 증가시킨다. 어느 한쪽의 삽입이 모두 끝났을 때, arraycopy를 통해 남은 배열은 한꺼번에 삽입한다.

두 번째 과제를 구현하기 위해 point 클래스를 생성하였다. double형의 x와 y값을 가진다.

finding 메소드는 closest pair를 찾는 메소드다. 받은 파일을 읽어 좌표값을 읽어 point 객체로 저장하며, point 객체들은 다시 배열에 저장된다. closest_pair 메소드를 호출하여 결과 값을 받아와 Input Data와 Output Data를 콘솔창에 출력한다.

closest 메소드는 먼저 매개변수로 받은 point 객체 배열의 길이를 검사하여 3이하일 경우는 2중 for문을 통해 각 좌표의 거리를 계산한다. 가장 작은 거리의 값을 리턴한다.

3보다 큰 길이의 배열일 경우, 배열을 x좌표에 따라 오름차순으로 정렬한 후 반으로 나눈다.

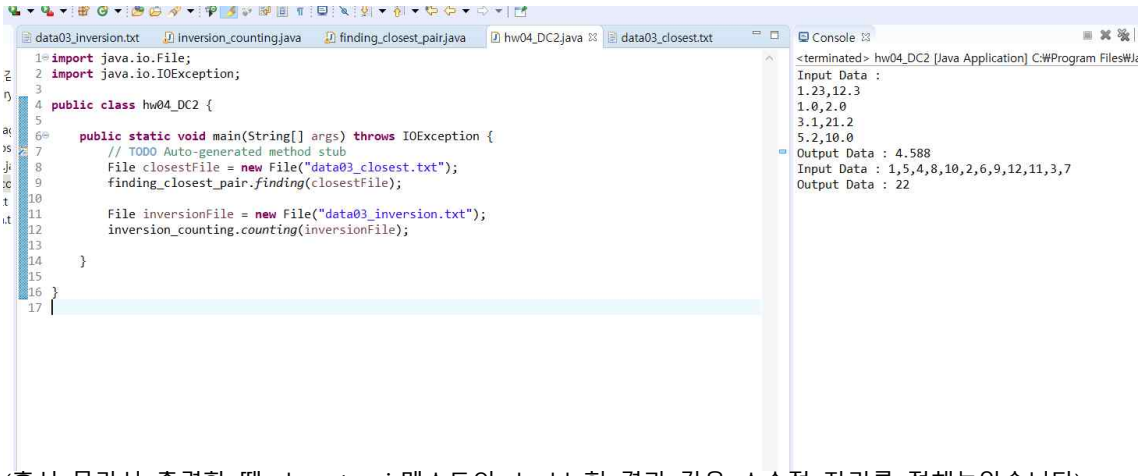
이때, 나누어진 경계에 해당하는 x좌표 2개의 평균을 L이라 한다. 반으로 나눠진 각 배열에 대해 closest_pair 메소드를 재귀적으로 호출하여 각각의 최소 거리를 가져온다. 더 작은 값을 theta로 한다. L과의 각 점들의 x좌표간 거리가 ±theta 보다 작거나 같은 경우 temp_list에 저장한다. temp_list는 y값에 따라 sort한 후 각 점들에 대해 다음을 반복한다.

y값 ±theta인 점들과의 거리를 구해 theta보다 작을 경우 theta를 갱신한다. 이때, y축으로 정렬되어 있기 때문에, 탐색하고자 하는 점이 temp_list[i]일 경우, I-1부터 탐색하여 y좌표간 거리가 -theta범위를 벗어나면 break를 통해 빠져나온다. 마찬가지로 I+1부터 탐색하여 +theta 범위를 벗어나면 break를 통해 빠져나온다. 탐색이 끝나면 theta를 리턴한다..

sorting 메소드는 divide & conquer를 사용하여 오름차순 정렬하는 메소드이다. point 객체를 정렬하기 위해 flag 값을 추가로 입력받으며, 0이면 x좌표에 따라, 나머지는 y좌표에 따라 정렬한다. inversion_counting과 동일한 방식으로 진행한다. 길이가 1이하이면 (길이가 0인 경우는 L±theta 범위 내에 값이 없어 temp_list 내 값이 없는 경우에 해당한다), 리턴하고 반으로 나눠 각각 sorting을 호출한 후 merge를 호출한다. merge 메소드에서는 inversion_counting 클래스에서와 마찬가지로 진행하나, flag에 따라 point 객체의 x값 또는 y값을 비교하도록 구현했다.

- 성능 비교 (과제에서 요구 했을 시)

4. 결과 화면



(혹시 몰라서 출력할 때 closest_pair메소드의 double형 결과 값은 소수점 자리를 정해놓았습니다)

5. 느낀점 및 고찰

closest pair finding을 구현할 때, 헷갈리는 말들이 많아 어려웠습니다. 수업 중에는 window를 두어 그 내의 점들만 비교한다고 이해했는데, 여기서는 δ를 통해 비교한다고 해서 이론적으로 어려운 부분이 있었습니다. 또, 자바로 구현할 경우 파이썬처럼 배열 슬라이싱이 안돼서 코드가 많이 길어질 수밖에 없어서 아쉬웠습니다.