알고리즘 실습 보고서

- D&C 2 -

전공 : 컴퓨터공학과

분반 : 05반

학번 : 201502085

이름 : 이규봉

1. 실행 환경

과제의 코드를 테스트한 환경은 아래와 같습니다. 따로 사용한 라이브러리는 없습니다.

- Windows 10
- Pycharm 2019.02
- Python 3.7.4

2. 과제 설명

- Counting Inversions
- Closest Pair of Points

3. 문제 해결 방법

- 문제를 해결하기 위해 자신이 사용한 방법, 아이디어에 대한 설명
  (구현한 코드에 대한 설명 등)

- Counting Inversions

: 기본적으로 Divide and Conquer 전략을 사용해 brute force 보다 빠른 방식으로 문제를 해결했습니다. main에서 sort_and_count 함수를 호출하면 인자로 넘겨받은 리스트를 반으로 쪼개 왼쪽 리스트에서의 inversion 개수를 세고 오른쪽 리스트에서의 inversion 개수를 세고, merge_and_count 함수를 호출해 양 쪽 리스트를 병합하며 각각의 경우의 inversion 개수를 세 세 값을 더해 r에 저장합니다. 이 과정이 재귀적으로 일어나기 때문에 (base case는 리스트 인자의 크기가 1인 경우입니다.) 결과적으로 구한 r 값은 전체 리스트에서의 inversion의 개수가 됩니다.

- Closest Pair of Points

: 위 알고리즘과 마찬가지로 기본적으로 Divide and Conquer 전략을 사용합니다. base case는 리스트의 크기가 3 이하인 경우이며, 이 경우 문제의 크기가 충분히 작기 때문에 brute force 방식을 사용해 가장 작은 거리를 구하게 됩니다. 문제의 크기가 최대 3이기 때문에, base case에선 최대 3+2+1 = 6번의 비교를 하게 됩니다. base case가 아닌 일반적인 경우엔 양 쪽의 점들의 개수를 동일하게 맞출 수 있게 선을 그어 divide 합니다. 이 때 선의 x 좌표는 middle_x_index-1 번 째 리스트의 x좌표와 middle_x_index 번 째 리스트의 x 좌표의 1/2로 합니다. divide된 subproblem들은 다시 재귀적으로 스스로를 호출해, 이 영역에서의 가장 작은 거리 값을 sigma1로, 타 영역에서의 가장 작은 거리 값을 sigma2로 두고, 두 값 중 최소값을 sigma로 합니다. 그리고 중간에 그어둔 선에서 sigma를 뺀 값보다 크거나, sigma를 더한 값 보다 작은 리스트의 원소들만을 추출하고, 이 값들을 Y 좌표로 오름차순 정렬합니다.
이 후에 이 리스트의 점들을 순회하며, sigma보다 작은 거리 값을 갖는 경우가 있다면 sigma를 업데이트 하되, y좌표 값이 sigma보다 멀리 떨어진 케이스엔 거리 값을 구하지 않고 건너뜁니다.

4. 결과 화면

5. 느낀점 및 고찰

두 점 사이의 최단 거리 문제는 백준 온라인 저지에서 본 적이 있지만 너무 어려워 보여서 푸는 것을 포기했었습니다. 그런데, 알고리즘 시간에 직접 문제 풀이를 다루고 의사 코드를 바탕으로 실제로 코드 작성도 해 보니 어느 정도 자신감도 붙고 공부에도 도움이 된 것 같습니다.