# IT 427, Design and Analysis of Algorithms

## Programming Assignment 6: Closest Pair

**Due date:** Nov. 6, 2023, Monday, 11:55 PM 50 points (30 on programs, 20 on report)

In this assignment, you are asked to find the closest pair of points from a given set of points. Since the number of points, $n$, may be as many as a few millions, the brute-force approach with time complexity $O(n^2)$ is not acceptable. We will use a clever divided-and-conquer technique as introduced in the class to lower the time complexity to $O(n \log n)$.

We will consider points in a 2 dimensional Euclidian plane. The distance between two points $(x_1, y_1)$ and $(x_2, y_2)$ is simply the Euclidian distance, i.e, $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.

**Prepare your programs on Linux Server:** This is similar to the previous assignments.

- Make a directory `asg5` under your `IT427`, i.e., `~/IT427/asg6/`. All of your programs and needed files should be saved under your `~/IT427/asg6` before run `submit427.sh`.
- Check the contents of my `/home/ad.ilstu.edu/cli2/Public/IT427/asg6`. There are two files for points, `xypoints.txt` and `test.txt`. You don't really need the two files to test your program. You can follow the same format and create your own, but the two are good for testing.

**Program requirement:** I will compile and run your program named `closestPair.java` as follows:

```
javac closestPair.java
java closestPair xypoints.txt
```

**Input:** A text file named `xypoints.txt` that contains several sets of points will be given. Each set begins with `** Set No` $k$`:` and ends with a dash line as shown in the following:

```
** Set No 1:
(8678.55778, 2611.51237)
(8733.26269, 1280.41467)
(3128.95124, 5063.82165)
------------------------------
** Set No 2:
...
...
------------------------------
** Set No 5:
(8767.44180, 9607.62346)
(2857.17527, 5355.55109)
(7564.78567, 2502.53652)
(7808.30083, 1920.96191)
(4850.37368, 9613.42945)
------------------------------
...
...
```

The example above shows the first and fifth sets, each has 3 and 5 points, respectively.

**Output:** The output of your program should be in the following format: For each set of points, the first line shows the set number followed by the number of points in the set. The second line shows the closest pair of two points. The third line shows the distance of the two points follow by the time used to search the closest pairs in milliseconds. Two consecutive sets are separated by a space line. For double values, use `%10.5f` for points and `%11.6f` for the distance.

```
Set No 1:  3 points
     (8678.55778, 2611.51237)-(8733.26269, 1280.41467)
     distance = 1332.221346 (1 ms)

Set No 2:  3 points
     (6012.95287, 2037.61099)-(8469.59551, 5252.23522)
     distance = 4045.849973 (0 ms)

Set No 3:  3 points
     (3412.48519, 4400.57817)-(8251.40552, 7586.37621)
     distance = 5793.484194 (0 ms)
     ...
     ...
Set No 15:  100000 points
     (9509.90893, 1123.01714)-(9509.96575, 1123.04950)
     distance = 0.065389 (137 ms)

by Chung-Chih Li
```

Your program should not assume the number of sets in the input file and the number of points in each set. I will use a different file of points to test your program. The milliseconds are the time used for the search only, which should not include time for reading the text file and some necessary initialization. As the size of the set become bigger, the milliseconds become meaningful. For example, on a set of size 10,000, a correctly implemented program will spend less than 30 ms on our Linux machines. You can use `System.currentTimeMillis()` to get the system's current millisecond count.

**Important Notes:** The score of your program not only depends on the correctness of your program, but also on the efficiency of your program. If your program runs at $O(n^2)$, you will not get more than 50% of the score (on both program and report parts). Also, I will expect to see time complexity analysis with necessary mathematical proof and arguments to explain how to get $O(n \log n)$. That includes well-defined recurrence relation based on your divide-and-conquer algorithm with sufficient explanation. Without adequate arguments in your report, you will get at most 10 points on the report. (Study sections 5.1 and 5.2 for recurrence relation analysis.)

**Submission:** Programs (30 points) and Reports (20 points)

Submission details are same as the previous assignment. Run the submission script with the submission number changed to 6, but you can use the same secret name as follow:

```
bash /home/ad.ilstu.edu/cli2/Public/IT427/submit427.sh peekapoo 6
```

Note that, since I will keep updating `submit427.sh` for different assignment, you have to run the script from my `/home/ad.ilstu.edu/cli2/Public/IT427/` directly for the most recent updated version, i.e., don't copy it to your own directory. Also, once you run `submit427.sh`, don't make any change on the target directory, including compile and run your programs. That will mess up the permission. You will lose significant points if you fail to follow the instructions.

**★ Any plagiarism will receive 0 and be reported to the school ★**

1. Programs: 30 points. Submission on Linux server.

   The score is based on the correctness and the programming style, which includes efficiency, appropriateness of data structures, and documentation of your programs. At the beginning of every program file, put a section of comments including (1) your full name, (2) student ID, (3) a pledge of honesty that you do not copy/modify from other's codes and (4) a declaration of your copyright that no one else should copy/modify the codes. You will receive:

   (a) 30 points: No error with a good programming style.
   (b) 25 ∼ 29 points: Minor error and fair programming style.
   (c) 20 ∼ 24 points: Some error and not so good but acceptable programming style.
   (d) 10 ∼ 19 points: Too many error and bad programming style, but meaningful.
   (e) 5 ∼ 9 points: Compilable but not working and the program must show reasonable trying.
   (f) 0 points : Fail to meet any of aforementioned qualities or plagiarism involved.

2. Report: 20 points. Submission through Canvas.

   You have to write up a report and prepare it in pdf format. You don't have to put program output on the report as I will run and exam your program directly on some different input files.

   (a) 5 points on the cover page, and

   (b) 15 points on algorithm descriptions, summary of the methods, data structures, and efficiency analysis on time and space in details in terms of big-O notations. If there is any difficulties encountered in this assignment, you can report it.

   **If your analysis is not clearly related to your program and provides sufficient justification, your report score will not be higher than 50%.**