WESTERN NEW ENGLAND UNIVERSITY
COLLEGE OF ARTS AND SCIENCES

CS 366 Design and Analysis of Algorithms October 18, 2021

Programming Assignment #2
Due Date: November 1, 2021

Objective: Implement two different algorithms (brute force and divide-and-conquer) for the Closest Pair of Points Problem. Analyze and compare running times (both theoretical and actual clock time) for the two methods.

Directions:

1. Download the source code `ClosestPair.java`, and study it carefully. Notice that an array `p` of points is constructed and filled with random numbers of type `double` in the range [0, 1000) x [0, 1000). Fill in the missing pieces of the program as follows.
2. **Brute force method**
   a. Write a non-recursive (brute force) method that finds the Euclidean distance between **<u>each</u>** pair of points and returns the minimum distance. You may want to write a helper function that calculates

$$\text{Euclidean distance} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

   b. Your algorithm must output **<u>both</u>** the minimum distance and the pair of points that are at that minimum distance.
   c. Use the system time function `System.nanoTime()` to determine the actual elapsed time in nanoseconds for this method of solving the problem. Output the elapsed time for this method.

3. **Divide-and-conquer method**
   a. Write a recursive (divide-and-conquer method) to find the distance between the closest pair of points.
      - Follow the algorithm provided in class as closely as possible. Use the same procedure names, parameter and variable names whenever possible.
      - You will have to make some adjustments since the pseudocode indexes an array from 1 to n, whereas Java indexes arrays from 0 to n-1.
      - You may have to adjust for other possible boundary conditions.
      - Recall what we discussed in class:
        In the algorithm `rec_cl_pair(p, i, j)`, the statement `sort(p,i,j)` is not a full-blown sort of the entire array. Rather, this is handling a base case of at most three points. Sort those (at most) three points by y-coordinate (brute force). Also, the statement `merge(p, i, m, j)` is a merge by y-coordinate. As part of mergeSort (given to you in the source code), you have a merge by x-coordinate. You need to write a similar method to merge by y.
   b. Output the minimum distance from this divide-and-conquer method. (The distances from the two different methods of solving should be the same!)
      BONUS: Keep track of the actual points that are at the minimum distance and output the pair of points.

(*continued on back*)

c. Use the system time function `System.nanoTime()` to determine the actual elapsed time in nanoseconds for this method of solving the problem. Output the elapsed time.

Sample input:
```
100
```

Sample output:
```
Brute Force
Minimum distance: 85.82421172389543
(x1, y1) = (608.4239938219546, 975.0133499574413)
(x2, y2) = (688.447327852766, 943.9962364776022)
Time = 8636792 nanoseconds

Divide-and-conquer
Minimum distance: 85.82421172389543
Time = 2566133 nanoseconds
```

4. Analyze the theoretical running times.
   a. Find a theta notation for the brute force method.
   b. Write a recurrence for the divide-and-conquer method. See notes from class. Use the Master Theorem to solve the recurrence.
   c. In theory, which method should be faster?

5. Analyze the actual running times (clock times).
   a. Run your program and experiment with various input sizes. Make a table that shows the results of the input sizes and clock times for each of the two methods. (Try a wide range of values for n: since n is of type int, in Java, it could be any positive integer up to $2^{31} - 1$ or 2147483647.)
   b. For approximately what values of n does it appear that the brute force method is faster? For approximately what values of n does it appear that the divide-and-conquer method is faster?
   c. Explain why you think this might be the case.

6. In your source code, include documentation! Include a header that contains your name, date, title and description of assignment. Include comments for each procedure that describe the purpose, parameters and return values. Comment the source code that I gave you to demonstrate that you understand how each part works. Also comment any parts of source code that are not entirely obvious (e.g., there are at most 8 points that need to be compared).

7. To submit your assignment, create a folder named with your first and last name (e.g., if your full name is Iam Groot, create a folder named **IamGroot**). Note: If you know that your last name is unique in the class, you may use only your last name. Inside the folder, include your source code (.java or .cpp file). Include file(s) that contain your output and answers (parts 4 and 5 of the assignment). Zip the file and upload it to the Kodiak Assignments folder for Program 2.