Semi-Optional Project 1: Closest Points Implementation

Due Whenever

This optional assignment will have you implementing in python the shortest distance problem we went over in class, and then running with it a little further. For credit, you must implement the following functions:

(a) A specialized mergesort(A, dim) (or any other $n \log(n)$ sort), which sorts a ordered pair (implemented as a python tuple) according either to the x or y axis depending on the second argument.

(b) A function generate_pairs(n, x_range, y_range) which generates random pairs for the purpose of testing. **This function must make sure that the set generated has no duplicates!**

(c) A function dist(p_1, p_2) which calculates the Euclidean distance between two points.

(d) The actual function closest_points(A) which returns *both* the shortest distance between two points from the set A as well as a/the pair of points which has this distance. We never went over in class how to return the actual points, but it's not that hard to figure out, and I leave it to you to think out the details!

(e) A brute force function brute_force_closest_points(A) which does the same thing but in quadratic time using the brute force approach.

(e) A function which compares the runtimes of those algorithms and displays the plot comparing the two like we usually do. (Call it whatever)

(f) A function which uses matplotlib to create a scatterplot of your data and which colors the two points returned differently from the other points, as well as draws the line segment between them.

(g) And finally... a generalization of all of this to 3D! It works in almost the exact same way, with, perhaps surprisingly, the exact same runtime.

Your submission should be in the form of a jupyter notebook presentation. Explanation can be minimal but you should guide a grader through confirming that all of your code works as intended, and displaying test plots which demonstrate the intended efficiency.