Programming Assignment 1

Joni Vrapi

09/18/2022

Statement of Integrity: I, Joni Vrapi, attempted to answer each question honestly and to the best of my abilities. I cited any, and all, help that I received in completing this assignment.

Problem 2a.

M-CLOSEST-POINTS(P, m)

```
1  pairs = [ ]
2  pointA = null
3  pointB = null
4  for i = 1; i < P.length; i++
5        pointA = P[i]
6        for j = i + 1; j < P.length; j++
7              pointB = P[j]
8              distance = sqrt((pointA.x − pointB.x)^2 + (pointA.y − pointB.y)^2)
9              pairs[pairs.length + 1] = (pointA, pointB, distance)
10             return pairs.timSort(key = distance)[:m]
```

Problem 2b. The worst case running time for this algorithm is $O(n^2)$. There is a nested loop, which does not execute n times for each iteration of the outer loop. It executes only the number of times required to check only not-previously-checked points, relative to the outer loop. This roughly translates to $\frac{n^2}{2}$ iterations, however asymptotically this is still n^2.

Problem 3. A problem similar to this (find the closest pair in a set P) is able to run in $O(nlog(n))$ worst case complexity because it utilizes a divide and conquer algorithm which is able to reduce the problem size by half upon each iteration. In this problem, however, the requirement to return the m closest pairs made it difficult for me to come up with an $O(nlog(n))$ solution to this problem. I have a sneaking suspicion that it is still possible, however. As it pertains to improving the worst case running time of this algorithm, I am not sure how to significantly improve it. As you can see in line 10, after populating the pairs array with all the pairs (which include their distances), I use TimSort to sort the array by distance, then return the first part of the array (up to

m). I believe a performance improvement could be made here by inserting new elements into the array in such a way as to preserve their order, negating the need to sort them at the end. This would remove a factor of $O(nlog(n))$, which would increase the performance of this function, however the worst case scenario would still be $O(n^2)$.