Bryan Kenote – Ryan Rapp – David Baugh

CS325 – Assignment1

**Find the Closest Pair Assignment**

**Pseudo-Code:**

**Brute Force**

for i=0 to coordinates size

coord1 = coordiantes[i]

for k=i to coordinates size

coord2 = coordiantes[k]

dist = distanceFormaula(coord1, coord2)

if dist < smallestDistance

smallestDistance = dist

found.clear

if dist == smallestDist

found.add(coord1, coord2)

return found

**Divide and Conquer**

if n <= 3

return bruteforce(coordList)

else

Compute separation line L

leftPairs = ClosestPair(left)

rightPairs = ClosestPair(right)

leftDist = DistanceForm(leftPairs)

rightDist = DistanceForm(rightPairs)

min = min(leftDist, rightDist)

middleDist = DistanceForm(getL, getR)

smallestDist = min(middleDist, min)

closestPairs.add(smallestDist)

return closestPairs

**Enhanced Pseudo-Code**

if(coordX <= 3)

Return bruteforce(coordX)

else

left = coordinate(coordX/2)

right = coordinate(coordX/2)

leftPair = ClosetsPair(left, coordY)

rightPair = ClosestPair(right, coordY)

leftDist = distanceForm(leftPair.next())

rightDist = distanceForm(rightPair.next())

middle = Intersection(left, right, min)

if(middle.size > 0)

middleDist = distanceForm(middle.next.getL(), middle.next.getR())

smallestDist = min(min, middleDist)

add smallestPair to closestPair

return closestPair

**Asymptotic Analysis of Runtime:**

*Please analyze the runtime for the three algorithms. In particular, please provide the recursive relation of the runtime for algorithm 2 and 3 and solve them.*

**Brute Force**

Within the brute force there are two for loops and then many simple operations. The for loops are equal to O(n) each and each other operation is O(c). The complete run time of the Brute Force program is O().

**Divide and Conquer**

To start off, the closestPair left and right both cost T(n/2). To compute the separation line and to sort all of the points that are within delta from the separation line is O(n logn). It is O(n) to get all of those points within delta and to find the closest-pair.

T(n) = 2T(n/2)+ cnlogn

T(n) = O()

**Enhanced**

This is basically the same as the divide and conquer but with the main difference that because there is a master pre-sorted list, we cut down on the need to sort every time. This allowed a O(n logn) to become a O(n).

T(n) = 2T(n/2) + cn

T(n) = O(nlogn)

**Plot the Runtime:**

*Plot the empirically measured runtime of the three algorithms as a function of the input size. Your plot should have clearly labeled axes and legends.*

**Interpretation and Discussion:**

*Discuss the runtime plot. Do the growth curves match your expectation based on their theoretical bounds? Discuss and provide possible explanations for any discrepancy between the experimental runtime and the asymptotic runtime.*

For our particular test case, the Enhanced algorithm is by far the slowest algorithm. It is obvious that it was implemented incorrectly. As is should theoretically be much faster than the brute force and divide and conquer algorithms. Brute Force and Divide and Conquer both came out with the desired outcome.