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**

**Identify L to separate the points into two roughly equal parts. X\_x naturally breaks into two parts Q\_x and R\_x. |O(n)**

**Break X\_y into two parts as well Q\_y and R\_y. Q\_y will contain the same set of points as Q\_x but sorted based on y. Similarly for R\_y and R\_x. | O(n) // this is done by scanning X\_y and place each point either in Q\_y or R\_y based its x value.**

d1=recursively call enhanced with Q\_x, **Q\_y** |T(n/2)

d2=recursively call enhanced with R\_x, **R\_y**|T(n/2)

d=min(d1,d2) |c

**Scan X\_y in order to eliminate any point whose x is not within d of L - this will result in a M sorted based on y. |O(n)**

Apply cross-closest-part(M,d) as explained in the slide, which returns the smallest distance overall. |O(n)

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