Assignment: asg4

Name: satish Dodda

ULID: 809961786

Secret Directory: behappy

Methods discerptions:

**find\_distance:**  this will accept two parameters point x and point y, finds the Euclidean distance, and return it

**find\_closest\_pair\_using\_brute\_force:**  this method will use the brute force method to find the closest pair and its distance, and it will accept two parameters x, and strip, x is a list of points in strip and strip is Boolean values whether it is true or false if it is true then it is finding the closet pair in strip for that, It has to compare every point in the strip with next 7 nearest values in the strip and find the closest and pair and return it, otherwise it will find the closest pair for x values

**recursively\_find\_closest\_pair:**  this method will accept two parameters of type list, x is a list sorted by using the x coordinate, and y is a list sorted by using the y coordinate. This method recursively finds the closest pair in both half and strip and returns the minimum among then

**find\_closest\_pair:**  this will accept a list of points as a parameter, will sort the points using x and y coordinates, and call the method recursively\_find\_closest\_pair by passing sorted x and y coordinate values this method will return a closet pair of points, and its distance back to find\_closest\_pair method this method return closest pair of points and its distance

**Algorithm**:

First, sort the list of points using the x coordinate and y coordinate store then in two variables of the type list, and pass these lists to the recursively\_find\_closest\_pair method this method will recursively find the midpoint and divide the x into two halves named then x\_left and x\_right , for y values it will divide then into two half based on midpoint value if values are less than mid-point those values are in y\_left half and values greater than mid-point is in y\_right half, y is sorted at the begin because to eliminate sorting while finding closest pair value in strip and dividing the y values in two halves because to find the strip values easily with less number of comparison, by recursively calling the method recursively\_find\_closest\_pair by passing x\_left half y\_left half and x\_right half and y\_right half if length of x is less than 4 then using find\_closest\_pair\_using\_brute\_force method to find the closest pair and it distance while backtracking will form a strip with points less the min distance till then and will call the find\_closest\_pair\_using\_brute\_force method by passing the strip value list and for every point it will find the distance with nearest 7 points and it will return the closest pair in strip compare the closest pair values among left half ,right half and strip and return min among then

**Data structure used :**

List l to store the points in the form of tuples in the list

Sort\_by\_x is a list of the points sorted using the x coordinate

Sort\_by\_y is a list of the points sorted using the y coordinate

X\_left is a list to store the left half-point of x

X\_right is a list to store the right half-points of x

Y\_left is a list to store the left half-point of y

Y\_right is a list to store the right half-point of y

**Time complexity:**

Sorting the points using x and y coordinates will take O(nlogn) time complexity

Creating the strip values takes O(n)

Find the closest pair using brute force for the strip values is O(7N) that will O(n)

Diving the y right and left half will take O(n)

Each time list is divided into two halves, each having n/2 points

the recurrence relation will T(n)= 2T(n/2)+o(n)

for sorting it will O(nlogn) and recursively divided and conquer to find the closest pair it will take O(nlonn)

**Total time complexity will be O(nlogn)**

**Space complexity:**

Total space is used to recursively stack is O(logn) but space is used to store the points is O(N) is greater than O(logn)

**Overall space complexity is O(n)**

**Problems faced :**

**It is algorithm is very straight forward and easy to understand and implement so I haven’t faced any difficulties while implementing it**