CS 325 Analysis of Algorithms

Implementation Assignment 1: Divide and Conquer

Daniel Green, Grayland Lunn, and Alex Tovar

Pseudo Code:

The code uses an object distObj that has a list starting with smallest distance and the following pairs sharing that smallest distance.

Brute Force:

```
brute_force(list of points)
    point1 = first point in list
    point2 = second point in list
    min = the distance of these first pair of points
    ln = length of list
    if ln is equal to 2
        if min is less than current objects distance
            set min as the objects current smallest distance
            add pair of points to list of smallest distance points
            return object
    else
        for i in range of ln – 1
            for j in range of i+1 to ln
                dist = get distance between point i and point j
                if dist is less than distObj
                    distObj = set dist as new min distance
                    clear list of points and add new points to list
                elif dist is equal to current min distance
                    add points to list
    return distObj
```

Enhanced Divide and Conquer:

```
closest_pair(list sorted by x, object list of smallest distance and points)
        length = length of xlist
        index = 0
        if length <= 3                          #base case
                for point in xlist
                        if index + 1 is greater then length of xlist
                                then add the two points
                        increase index
                return distObj
        mid = middle of list
        leftArrayX = split the left half
        rightArrayX = split the right half
        distObj1 = recursive call closest_pair(leftArrayX, distObj)
        distObj2 = recursive call closest_pair(rightArrayX, distObj)
        merge distObj1 and distObj2
        get delta by getting distance from combined lists
        min_cross  = closest_cross_pair(xlist, delta)          #find min distance on boundary
        return list of merging distObj1 and min_cross

closest_cross_pair( list, delta)
        distObj = get current smallest distance
        length = length of list
        mid = length of list divided by 2
        if length is less than 7
                strip = list
        else
                strip = list from range of [mid – delta, mid +delta]
```

```
    sort strip by y coordinates
    index = 0
    for point in strip
        for i in range of 1 through 8
            if index + i < length of strip
                then add those points to distObj
            increase index by 1
    return distObj
```

Asymptotic Analysis:

Brute Force Algorithm:

In the brute force algorithm, we have a nested for loop. Making this algorthim have a runtime of $O(n^2)$. There are constant operations as well but the nested for loop overtakes them.

Enhanced Divide and Conquer:

In the enhance Divide and Conquer, the elements are pre-sorted when entering the function. In each recursive call we split the array in two halves. Then when we call the closest_cross_pair function, it runs a for loop adding points that have the minimal distance. The for loop has an O(n) runtime, and the recursive call has a O(logn) runtime because each call halves the problem. So the total runtime will be O(nlogn).

Plotting and Analysis:

plots and explanation go here