Guillermo Rivera
Project 2
CS 2223

For our project, we were asked to test out two programing methods and compare them. These methods were the Brute Force and Recursive methods. Using python 3.7.0a2 and the text editor IDLE, the methods were implemented and tested. After testing the methods with different values, their distance to calculate closest pairs and time to execute were displayed. The efficiencies are calculated in the following pages.

TE : Time Efficiency

Brute force

- Computes distances for all $n(n-1)/2$

$$\frac{n^2-n}{2}$$

Keep the Highest factor

- Brute force runs in tim $\boxed{\Theta(n^2)}$: $T(n)$

Divide and Conquer

- sort halves

$\frac{n}{2}$ $\frac{n}{2}$

- solve for each half
- merge together

n

$T(n) = T(n/2) + T(n/2) + O(n)$

$T(n) = 2T(n/2) + O(n)$

$\boxed{T(n) = \Theta(n \log n)}$

Space Efficiency

Brute force   def BF   A

```
i = 0        C     values B
Val = [ ]    D
J = i+1      nE
d = ..       n·n F
```

$$\underbrace{A+B+C+D}_{k_1} + \underbrace{nE}_{k_2} + \underbrace{n^2F}_{k_3}$$

$$k_1 + nk_2 + n^2k_3$$

↗ ↗ Increases

$$\boxed{O(n^2)}$$

Divide conquer

```
def Rec       A
    (in1, in2) B
     Pl  C
     Pr  D
     Ql  E
     Qr  F
      while
        Pl   nG
        Ql   nH
        Pr   nI
        Qr   nJ
      While
         While (k<=num)  n^2 K
```

$$\boxed{O(n^2)}$$

```
## Project 2
## Guillermo Rivera
import time
import sys
import math
import copy
import ast

print("Welcome to Guillermo's Closest Pair program")
print("Please follow the on screen Instructions")
print(" ")


def dist(x,y):#distance equation
    return math.sqrt((x[0]-y[0])**2+(x[1]-y[1])**2)

def BF(values):#Brute force checks all pairs
    i=0
    val=[float('inf'),0,0]
    while i <len(values):
        j=i+1
        while j <len(values):#loop rest of array
            d = dist(values[i],values[j])
            if d<val[0]:
                val=[d,i,j]
            j=j+1
        i=i+1
    r= [val[0],values[val[1]],values[val[2]]]
    return r[0]#return the distance

def Rec(in1,in2):#Recursive check
    if len(in1)<=3:
        result=BF(in1)
    else:
        x=0
        Pl=[]
        Pr=[]
        Ql=[]
        Qr=[]
        while x <len(in1)//2:#first half
            Pl.append(in1[x])
            Ql.append(in2[x])
            x=x+1
        while x<len(in1):#second half
            Pr.append(in1[x])
            Qr.append(in2[x])
            x=x+1
        Dl=Rec(Pl,Ql)#callback
        Dr=Rec(Pr,Qr)
        d=min(Dl,Dr)
        n=int(len(in1))
        m=in1[(n//2)-1][0]
        S=[]
        for element in in2:
            if math.fabs(element[0]-m)<d:
                S.append(element)
        dminsq=d**2
        num = len(S)
        i=0
        k=1
        while i < num-2:
            while (k<= num-1 )and(((S[k][1]-S[i][1])**2) < dminsq):
                var=dist(S[k],S[i])
                dminsq=min(var**2,dminsq)
                k=k+1
            i=i+1
        result=math.sqrt(dminsq)
    return result#return the distance

def effRec(inputval):#time efficiency recursive
    in1=copy.copy(inputval)
    in2=copy.copy(inputval)
    in1.sort(key=lambda t: t[0])# x
    in2.sort(key=lambda t: t[1])# y
    t0=time.time()
    result=Rec(in1,in2)
    t1=time.time()
    T=t1-t0
    print('The distance calculated by Recursion : '+str(result)+' took a time of: '+str(T))
    return

def effBF(inputval):#time efficincy brute force
    t0=time.time()
    result=BF(inputval)
    t1=time.time()
    T=t1-t0
    print('The distance calculated by Brute Force : '+str(result)+' took a time of: '+str(T))
    return

def effall(inputval): #run all
    effBF(inputval)
    effRec(inputval)
    return
```

```
#effBF(points)
#effRec(points)

while(True):
    userval= str(input("input the name of the file IE= 'input.txt' :"))
    print(" ")
    print("Your input was: "+userval+"\n Testing Brute Force and Recurssion for closest pair...")
    print(" ")
    # Get Input
    f=open(userval, 'r')
    userval = ast.literal_eval(f.read())
    effall(userval)
    continue
```