**Project 3 Part 4 (randomized approach average O(n))**

Name: Michael Fatemi Period: 7

Date: 12/13/2020

Is your lab name l034?(lowercase L followed by digits 034) \_\_\_\_\_Yes\_\_\_\_\_\_

Did you created a class to store a point? \_\_\_\_\_Yes\_\_\_\_\_

Did you use a vector to store the points you read? \_\_\_\_\_Yes\_\_\_\_\_\_

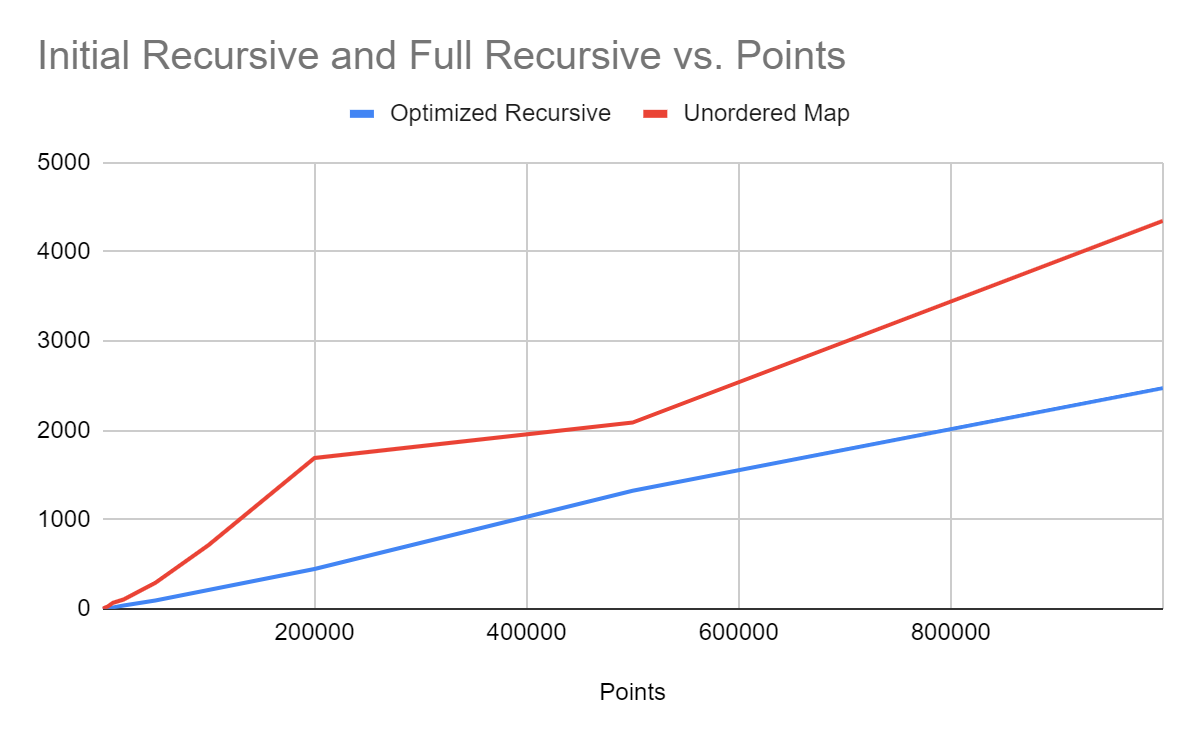
Does your main contain only 2 calls of: part3() and part4() (NO part1/2!!)? \_\_\_\_\_Yes\_\_\_\_\_\_

(in main you may also have the part to display results for the 2 methods and them also in the txt file)

Did you use an unordered\_map for your dictionary? \_\_\_\_\_Yes\_\_\_\_\_\_

Did you implement the Knuth algorithm to randomize the points? \_\_\_\_\_Yes\_\_\_\_\_\_

1. **Paste here a clear picture of the graph that compares the running times of the “full recursive” algorithm and “randomized” algorithm versus number of points. (use 2 different colors for the 2 graphs, colors that can be visible even if you print in black and white). Each point on this graph should be an average of several runs for that size:**



1. **Paste here the content of the results.txt when you run your lab on the content of the file points10k.txt and points100k.dat**

**For 10k:**

Recursive Optimized x [2] 30ms (15/cycle)

(0.49999999999999883,0.50000000000000033) (0.49999999999999889,0.50000000000000033) 5.5511151231257827e-017

Hashing Method x [2] 84ms (42/cycle)

(0.50000000000000089,0.49999999999999939) (0.50000000000000089,0.49999999999999944) 5.5511151231257827e-017

**For 100k:**

Recursive Optimized x [2] 507ms (253/cycle)

(0.49999999999973127,0.49999999999889994) (0.49999999999975941,0.49999999999890893) 2.9545963426100509e-014

Hashing Method x [2] 968ms (484/cycle)

(0.49999999999975941,0.49999999999890893) (0.49999999999973127,0.49999999999889994) 2.9545963426100509e-014