Andrew Breyen
Programming Assignment 2 Report
CSCI338

The approach that you took to implementing this algorithm, i.e., were you able to implement the algorithm exactly as described in the book, did you need to make any modifications to it, etc?

I believe I implemented the algorithm as described in the book. I had to implement a few changes in order to make it work, such as creating helper methods and a Point class.

Methods/classes I created:
`class Point():` An object that holds x and y coordinates

`sortByX(p):` This is a helper method to sort a list of points by x coordinate. It's used as a key for the sorted(points, key=sortByX) call.

`stripClosest(strip, size, d)`: This method deals with determining the closest pairs middle strip of points. Parameters: Strip: A y coordinate sorted list of points in the middle strip. Size: the size of Strip. d: minimum distance to 'beat'

`closestPairs(Px, Py, n):` This is the bulk of the solution. It is the recursive part. Parameters: Px: Points sorted by x coordinate. Py: Points sorted by y coordinate. n: the size of Px

`distance(p1, p2):` This method determines the distance between two points. Parameters: p1: a point. p2: a point.

`bruteForce(p, n):` The brute force method. This method is used when the list of points is less than 3. Parameters: p: A list of points. n: the size of p.

A description of anything that I should know about your program. This could be limitations like, it only works for XXX files, but not YYY file, or challenges that prevented you from completing the assignment.

There were no issues getting my problem to work for a specific type of files. The only error I noticed was with rounding/truncating the results at the end. Because the points were so close together, rounding to 4 decimal places yielded a distance result of 0.0 for some of the larger .pts files. (I noticed this with 65536.pts) When you do not round (to change this to not round, modify line 196 from
`print("Closest Distance: "+str(closestDistanceRounded))` to
`print("Closest Distance: "+str(closestDistance))` the result is correct.

I believe this issue is a limitation of Python Floats, as when the rounded value is not used, the result is outputted in scientific notation. (for 65536.pts, it yields a result of 4.368065933567342e-05)

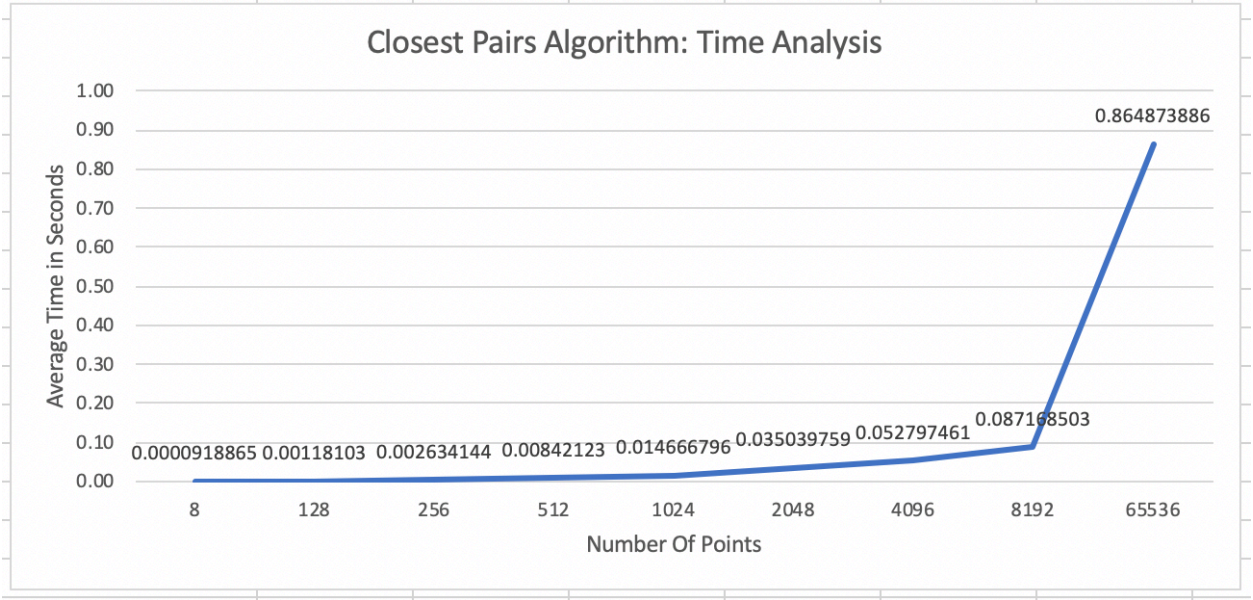
This is also an issue when reporting the points if they are very small floating point numbers: for example, in 65536.pts, Point1X is reported back as 2.8e-05 and Point2X is reported as 3.9e-05 due to being such a small floating point number.

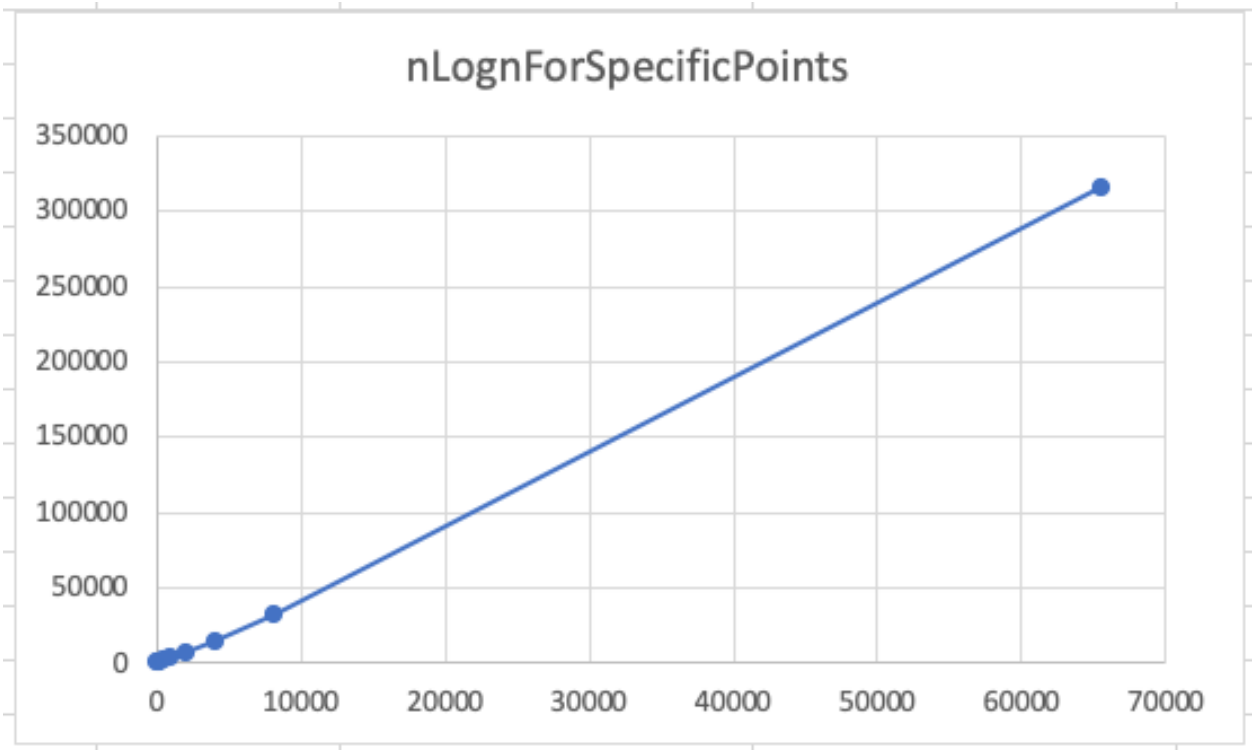
An analysis comparing and contrasting the theoretical analysis of EfficientClosestPair (from the book), with your experimental analysis. Does your experimental analysis support the theoretical analysis from the book? Why or why not? You should include plots to support your conclusions.

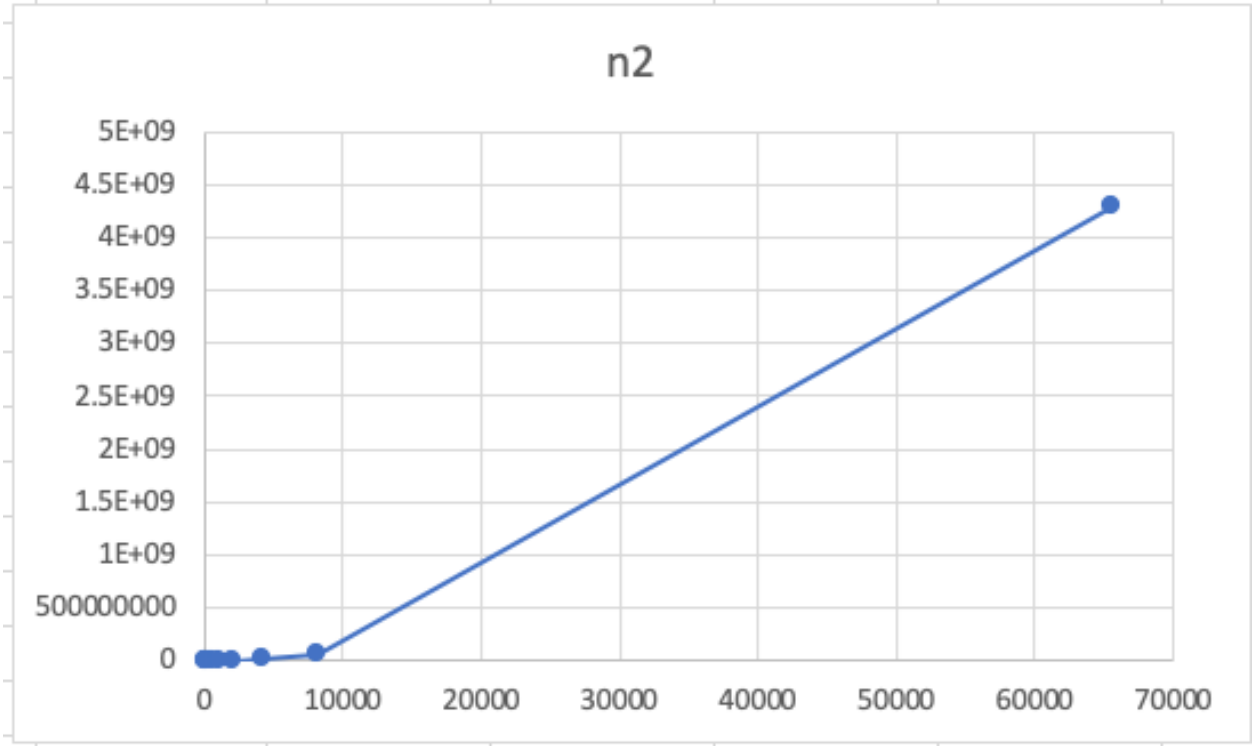
I believe that my implementation followed the nLogN theoretical analysis from the book.

We followed the books algorithm, and the time efficiency did not exponentially increase. D

numOfPoints	averageTimeInSeconds	nLognForSpecificPoints	logN	n2
8	0.0000918865	7.224719896	0.90308999	64
128	0.00118103	269.7228761	2.10720997	16384
256	0.002634144	616.5094311	2.40823997	65536
512	0.00842123	1387.14622	2.70926996	262144
1024	0.014666796	3082.547156	3.01029996	1048576
2048	0.035039759	6781.603742	3.31132995	4194304
4096	0.052797461	14796.22635	3.61235995	16777216
8192	0.087168503	32058.49042	3.91338994	67108864
65536	0.864873886	315652.8287	4.81647993	4294967296




nLognForSpecificPoints
350000
300000
250000
200000
150000
100000
50000
0
0
10000
20000
30000
40000
50000
60000
70000


n2
5E+09
4.5E+09
4E+09
3.5E+09
3E+09
2.5E+09
2E+09
1.5E+09
1E+09
500000000
0
0
10000
20000
30000
40000
50000
60000
70000