CSE 417 HW 4, YANG ZHANG, 1030416, zhy9036@uw.edu

Problem 1

The solution for finding closest distance of two points in 1-dimensional is not proper for 2-dimensional case. The reason is that in 2-dimensional the distance between 2 points are affected by both the difference between x-coordinates and the difference between y-coordinates. Therefore, sorting by one coordinates may not returns the correct answer. For example, if we have many points that distributes centered in one axis, like the picture showing below:

Sorted points by x-coordinates will not yield the closest pair (the pair with smallest distance in x-axis has largest distance in y-axis)

Problem 2

a) Suppose there are three points: p1 = (x, y1), p2 = (x, y2), p3 = (x, y3),
And y3 – y1 > = 2(y2 – y1)

If p3 and p1 belong to right area plus this area only has those two points, and p2 belongs to left area. The δ/2 of left half is (y3 – y1)/2, which is equal or bigger than the distance between p1 and p2. Therefore, if we draw the box regards to p1, p2 will also be included in this box.

Therefore, the statement is false in this case.

b) **Prove:**
Suppose there are two points A and B that have same x-coordinate and are both on line L, where A belongs to right half, while b belongs to the left half. Since the delta is constant for both side. There are at most two points with the delta/2 box of A and the delta/2 box of B. In other words, the boxes share the same two points. Moreover, other points are not on L, therefore, the old rule can applied for those points. i.e. no more than 6 points can be within delta/2 area for those points. Therefore, in total, there are still at most 8 point within this region.

Problem 3

Result for provided test case:

Algorithm / **Test Case**	**Burte-Force N^2**	**CP nlogn**	**CP nlog2n**
File 1	Infinity	Infinity	Infinity
File 2	1.0	1.0	1.0
File 3	0.99	0.99	0.99

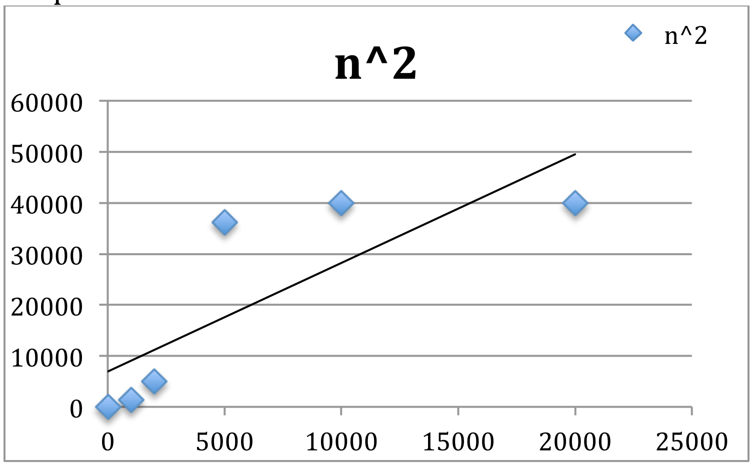
Time Analyses:

Case 1: points are distributed evenly

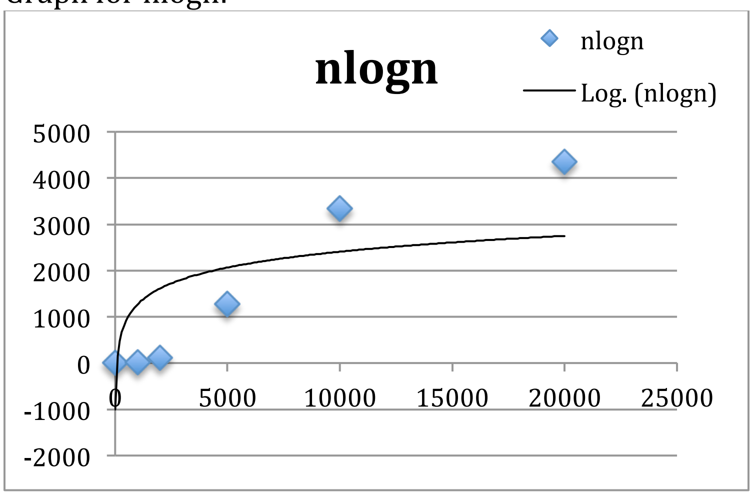
Data:

n	n^2	nlogn	nlog2n
10	0	0	0
1000	1459.1	17.7861	28.3582
2000	5713	121.23	178.45
5000	35084	1276.907	1318.0001
10000	X	3335.182	3389.46
20000	X	4350.2048	4445.1799

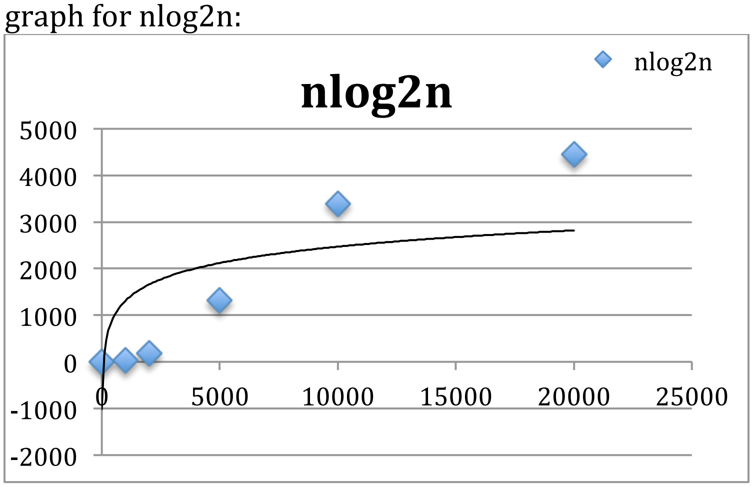
Graph for n^2:



Graph for nlogn:



graph for nlog2n:

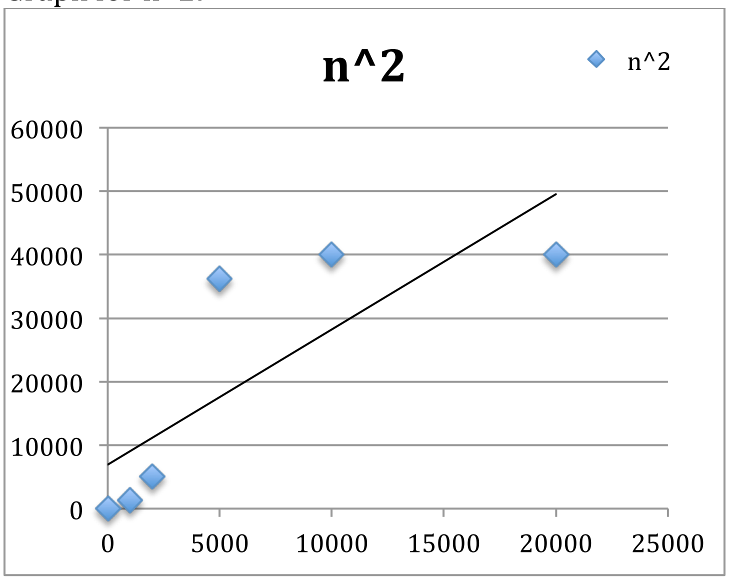


Case 2: points are distributed within range(0,1)

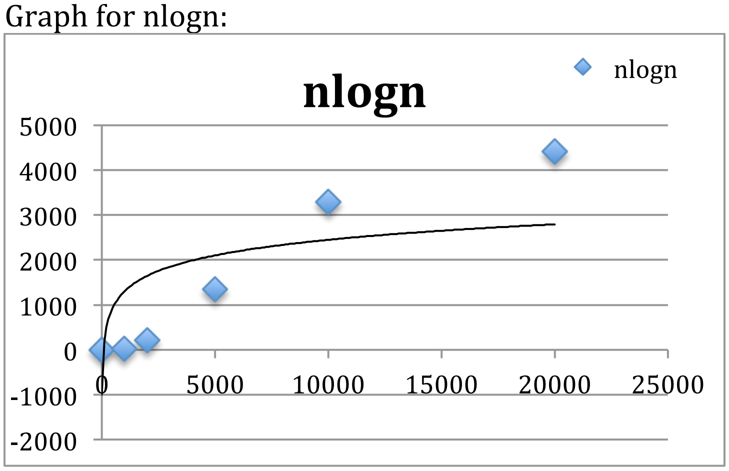
Data:

n	n^2	nlogn	nlog2n
10	0	0	0
1000	1303.5	20.345	23.13
2000	5014	204.12	230.03
5000	36214	1356.09	1405.68
10000	X	3290.65	4000.24
20000	X	4409.87	4789.21

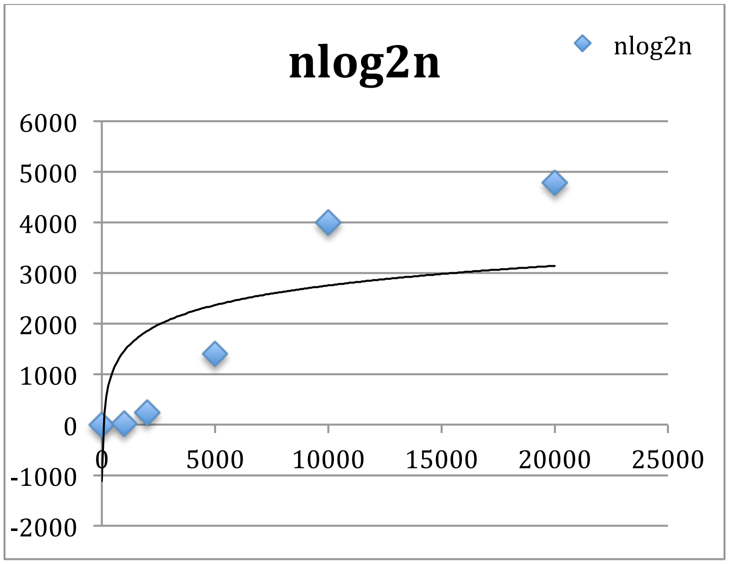
Graph for n^2:



Graph for nlogn:



graph for nlog2n:



After testing the three algorithms, I concluded that in both cases, the run time increasing speed of O(n^2) is always the fastest, then is O(nlog2n). Finally, the run time increasing of O(nlogn) is the slowest. However, there is not a significant run time difference between different case for the same algorithm.

In my test case, when n is bigger than 100, a notable difference of run time can be observed between O(n^2) and O(log) family. In other word, when n is bigger than 100, the divide and conquer algorithms are faster than the naïve method.