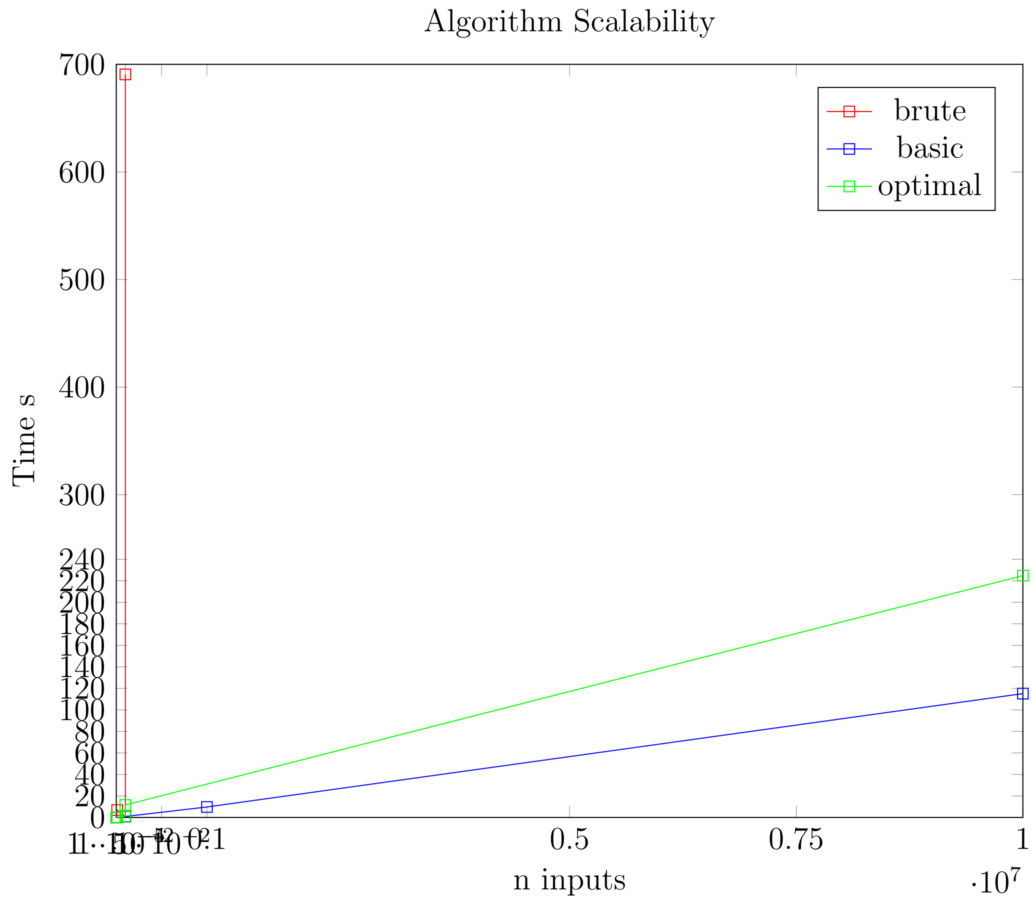
Programming Assignment 1 by Jordan Nguyen

1 Scalability



	10^2	10^3	10^4	10^5	10^6	10^7
Brute	.011 s	.086 s	6.867 s	690.670 s	NA	NA
Basic	.014 s	.020 s	.098 s	.908 s	9.743 s	115.112 s
Optimal	.014s	.020s	.124 s	1.145 s	11.624 s	224.787 s

Table 1: This table shows how long each algorithm takes to complete (in seconds) for a given number of inputs.

The basic and optimal divide and conquer algorithms clearly outperform the brute force algorithm when finding the closest pair. However, it is unexpected that the basic algorithm outperformed the optimal algorithm. I think this happens because my optimal algorithm creates twice as many vectors during the recursive step compared to the basic algorithm. In the basic algorithm, the recursive step creates two new vectors, the first half of X's and the second half of X's, and calls itself on each of these halves. On the other hand, the optimal solution creates four new vectors on each recursive call, the first half of X's, the second half of X's, the Y's that are before the X midpoint, and the Y's that come after the X midpoint. At large inputs, this extra memory slows down the optimal algorithm, thus making it perform slower than the basic algorithm.