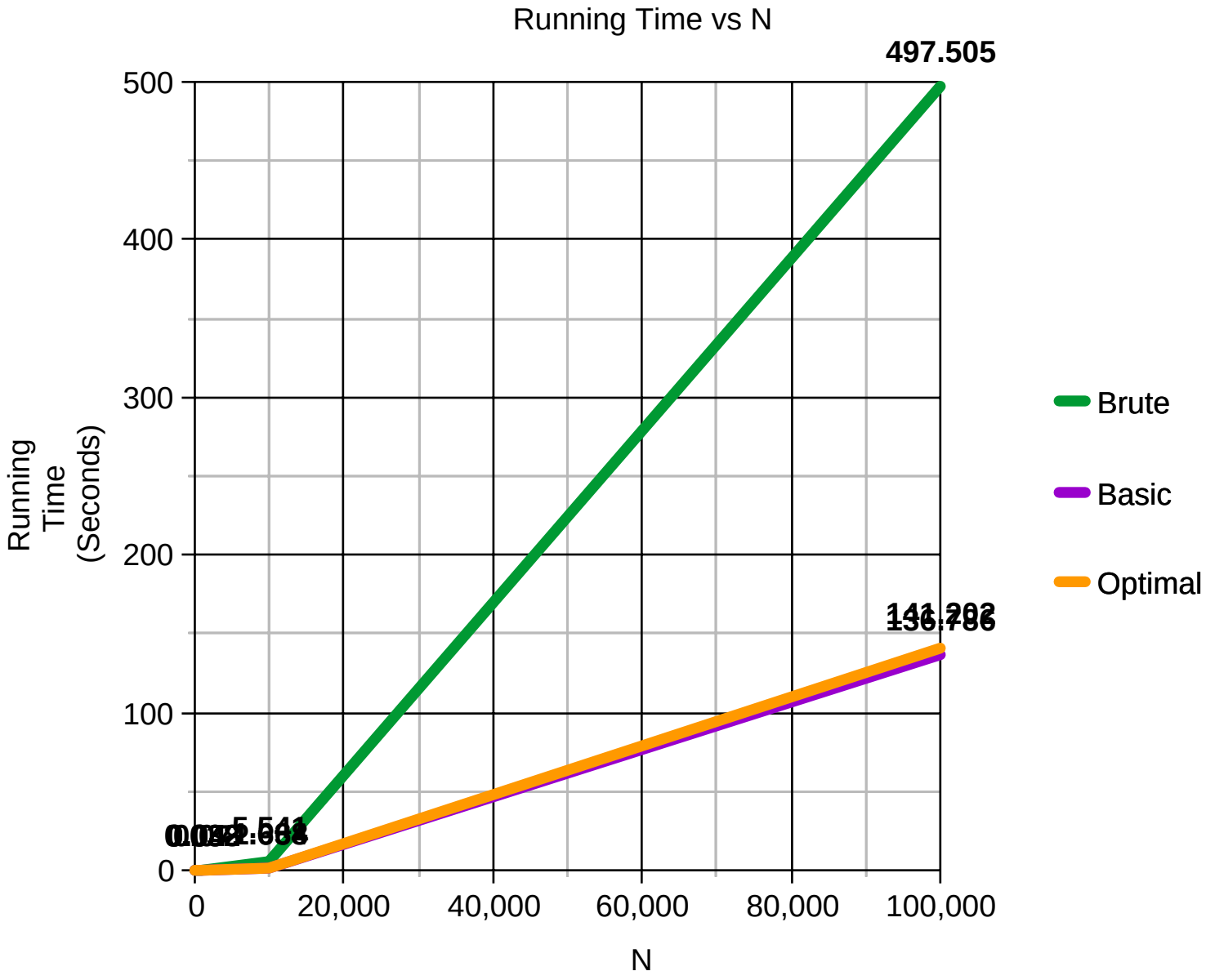
Running Time vs N
497.505
500
400
300
200
100
0
Running Time (Seconds)
0
20,000
40,000
60,000
80,000
100,000
N
Brute
Basic
Optimal

	Brute	Basic	Optimal
100	0.04	0.04	0.04
1000	0.07	0.038	0.032
10000	5.541	1.638	1.664
100000	497.505	136.786	141.202

Taking N = 100,000, then O(N^2) = 497.505, O(n(logn)^2) = 136.785, and O(nlogn) = 141.202. Since the code for the brute algorithm is the most simple to analyze, I will take O(100,000^2) = 497.505 (the code provided for the brute algorithm simply runs through the list of points once per each element of the list, which equates to n^2 comparisons). Therefore, O(1) should equal 4.975e-8. Using the basic algorithm O(100,000 * (log(2)(100,000))^2) = 136.786, so O(1) = 4.95e-6, meaning the basic algorithm does not conform to the expected limit of running time. Similarly, O(1) for the optimal algorithm = 8.501e-5, so the optimal algorithm also does not conform to the expected running time.

Taking N = 10,000, then O(1) for the brute algorithm = 5.541e-8. O(1) for basic = 9.277e-7. O(1) for optimal = 1.25e-5.

The reason for the basic and optimal algorithms being slower than expected are due to the recursive costs of splitting the list (list(s) for the optimal algorithm), creating the strip, and comparing through the strip. These operations would increase the running time because of how they are called recursively at every divide/merge.

The reason for the optimal algorithm being as slow as the basic algorithm is that the cost of building the y sorted list, sorting the entire list by the y coordinate once, dividing it by an x coordinate, and passing that list to every recursive call slows the optimal algorithm down.