To answer the following questions, justify using the running times measured (the table). That is, you have to show whether your measurements are “consistent” with the theoretical running time or not. To do that, you have to look at your measurements in each row column-to-column, check if the measurements grow as the theoretical running time predicts. You have to include in your justification what cells of the table you are looking at. Provide your answers in the remarks section of the code header.

(a) (5 points) Are the running times measured for Brute Force consistent with the theoretical running time? (Y/N). Y

(b) (15 points) Justify.

- The brute force algorithm has theoretical running time O(n^2). This means that when n is multiplied by 10, the running time should increase by approximately 100 times, because (10n^2)/n^2 = 100. Looking at the table when n increases from 10^4 to 10^5, the running time increases from 40,145,100 nanoseconds to 1,954,768,500 nanoseconds. The ratio is 1,954,768,500/ 40,145,100 = 48.69. When n increases from 10^5 to 10^6, the running time increases from 1,954,768,500 nanoseconds to 242,662,110,900 nanoseconds. The ratio is 242,662,100,900/1,954,768,500 = 124.14. Although these ratios are not exactly 100, they are of the same order of magnitude and demonstrate strong quadratic growth, especially for larger values of n. The smaller input sizes do not scale perfectly because constant overhead, JVM warm-up effects, and timing fluctuations can distort measurements when n is small. Overall, as n increases, the running times grow in a manner consistent with the theoretical O(n^2) complexity

(c) (5 points) Are the running times measured for Divide & Conquer consistent with the theoretical running time? (Y/N)

- N

(d) (15 points) Justify.

- The divide and conquer algorithm has theoretical running time O(n log n). When n is multiplied by 10, the running time should increase by approximately 12 to 15 times for these input sizes, because: (10n log(10n)) / (n log(n)) = 10 · (log(10n) / log(n)). From the table, when n increases from 10^2 to 10^3, the time increases from 337,000 to 1,477,500 nanoseconds, which gives a ratio of approximately 4.38. When n increases from 10^3 to 10^4, the ratio is approximately 1.99. These values are significantly below the expected 12–15 times increase. When n increases from 10^4 to 10^5, the time increases from 2,943,200 to 39,091,700 nanoseconds, producing a ratio of approximately 13.28, which matches the expected theoretical growth. However, from 10^5 to 10^6, the ratio drops again to approximately 1.75, which is much smaller than expected. Since only one of the column-to-column comparisons closely matches the predicted scaling behavior and the others are significantly smaller, the measurements are not consistently aligned with the theoretical O(n log n) complexity. Variations may be caused by JVM warm-up effects, recursion overhead, or timing fluctuations. Based strictly on the table data, the divide and conquer results are not consistently following the expected theoretical growth pattern.