İlke Kaş

21803184

Section 3

CS201 HOMEWORK ASSIGNMENT 2

1. **Computer Specifications:**

- **Processor:** 2 GHz Quad-Core Intel Core i5
- **RAM:** 16 GB 3733 MHz
- **Operating System:** macOS Catalina version 10.15.7

2. **Obtained Values from Algorithms**

- **Algorithm 1:**

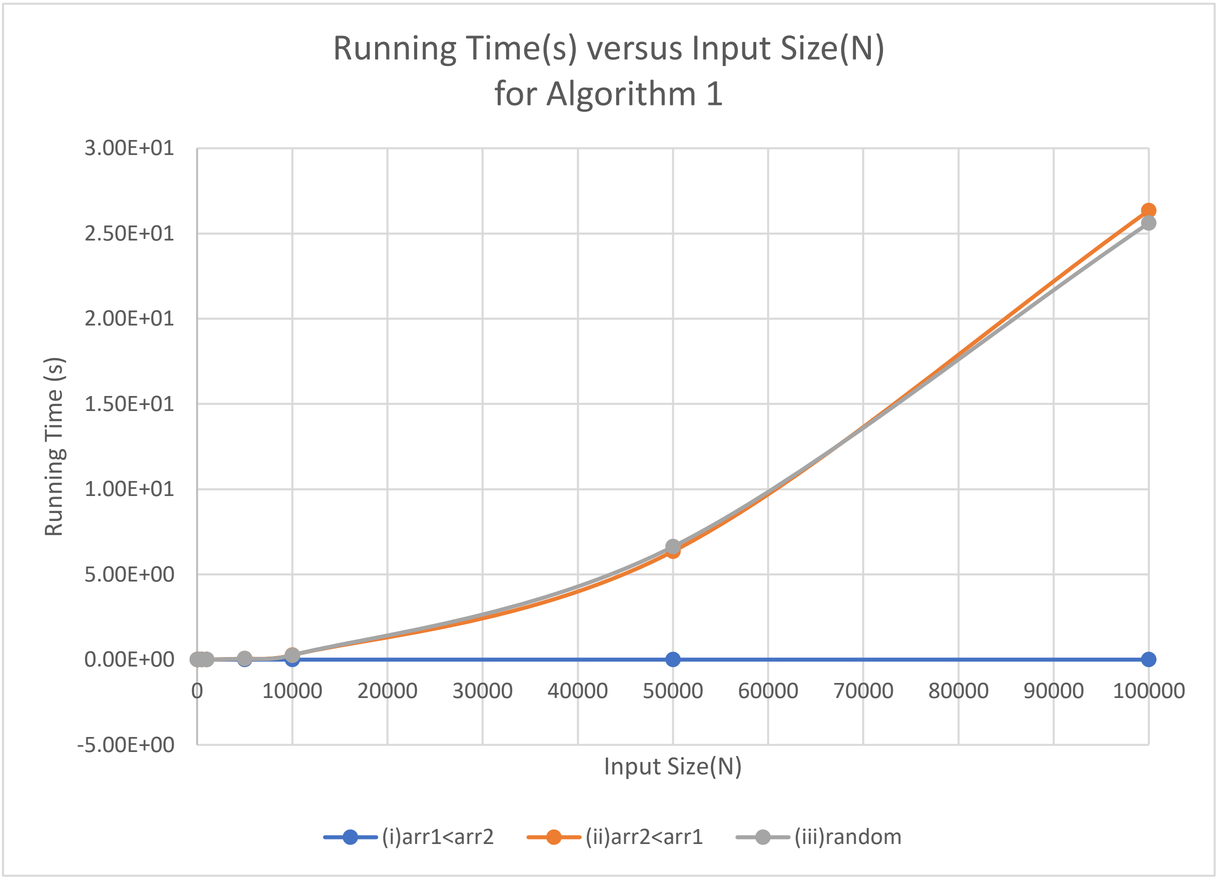

Figure 1

For Algorithm 1, one can conclude from **Figure 1** that the situation which all elements of the arr1 is smaller than all elements of arr2 is the best case. The reason of that can be understood by analyzing Algorithm 1. When (i) arr1 < arr2 for all elements in algorithm 1, to find the right place of one element in arr2, there is two consecutive for loops that traverse each items in arrays once. Thus its complexity will be linear as blue line. On the other hand, when (ii) all elements of arr1 is greater than arr2, to find the right place and shift other elements according to that place, algorithm 1 should also traverse the arr3 for each element of arr2. This lead us to two nested loops whose complexity will be quadratic. This part of algorithm is also valid for random values of arr1 and arr2. However, the worst case is the case which for all elements, arr2 < arr1. The reason for that, amount of the iteration of shifting loop is greater or equal in (ii) than (iii). Therefore, the case (iii) can be thought as the average case since we know that $T_{avg}(N) \leq T_{worst}(N)$.

Since the worst case is (ii), worst time complexity can be determined by two nested loops that traverse each items in arrays once and it is $T_{worst}(N) = O(N^2)$.

- **Algorithm 2:**

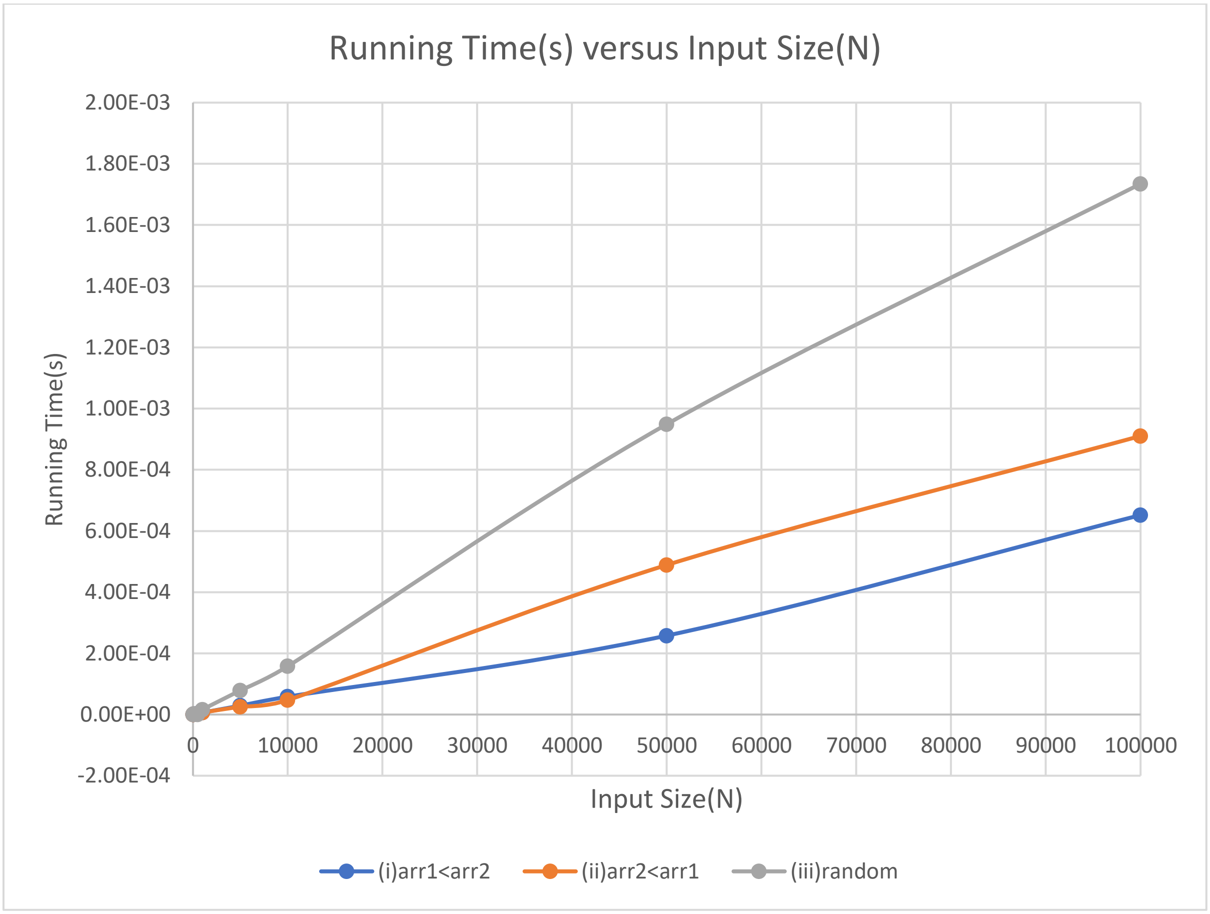

Figure 2

For Algorithm 2, one can conclude from **Figure 2** that the situation which all elements of the arr1 is smaller than all elements of arr2 is the best case. The reason of that can be understood by analyzing Algorithm 2. There are theree consecutive while loop that traverse each element once through arrays. Therefore, worst time complexity of this algorithm is linear: $T_{worst} = O(N)$. In there, all of the three cases (i),(ii) and (iii) has the same time complexity. The best case will have to do minimum number of comparisons. Therefore both of the situations (i) and (ii) should be the best case. However, in **Figure 2,** case(i) seems as the best case. This is not caused by algorithm. On the other side, worst case caused by the maximum number of comparision which is caused by the case (iii). Within these cases, one can choose the case (iii) as average case.

3. Expected Worst Case Growth Rates

- **Algorithm 1:**

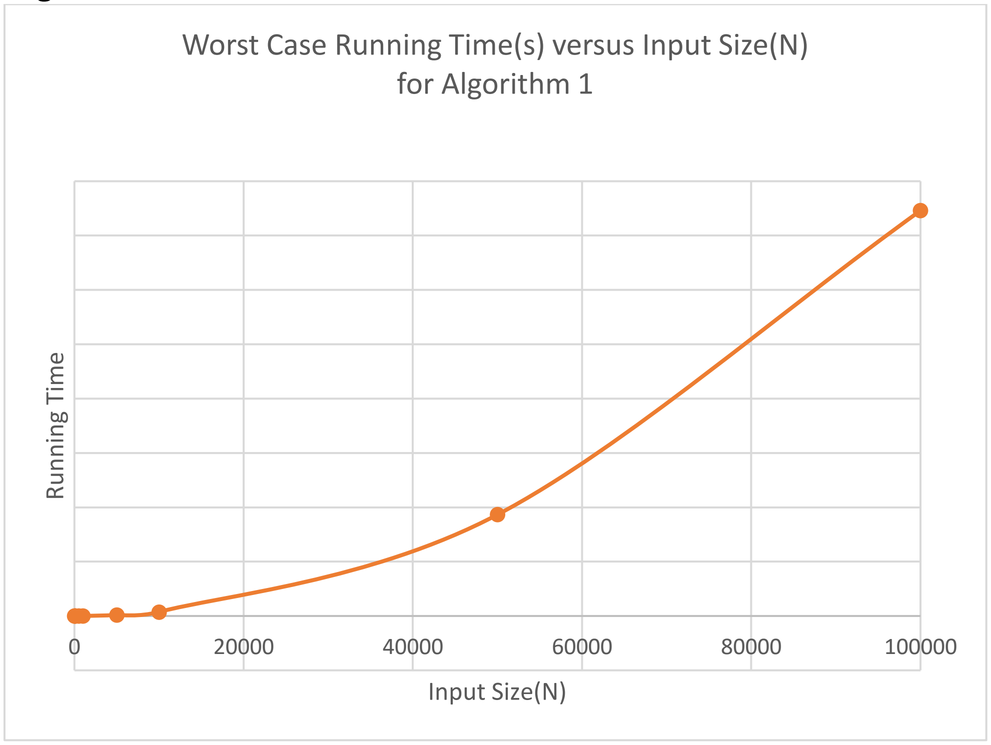

Figure 3

- **Algorithm 2:**

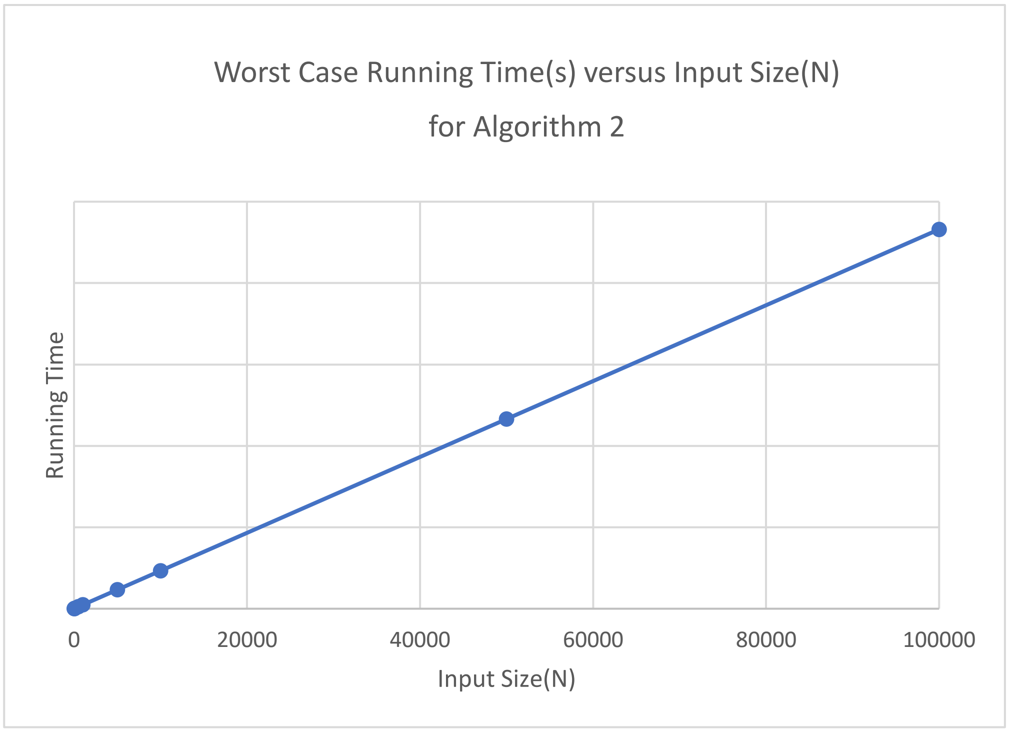

Figure 4

4. **Comparison**

For algorithm 1, as discussed in the second part, the worst case is the case (ii) where all elements of arr2 is greater than all elements of arr1. To merge these two arrays in this case, program executes two nested for loops each traverse the arrays' items once. As seen in **Figure 1,** the growth rate of orange line seems quadratic as in **Figure 3**.

For algorithm 2, as discussed in the second part, the worst case is the case(iii) where there is no relation of order between elements of arr1 and arr2 and since that, number of comparisons have possibly the maximum value. To merge these arrays in this algorithm, program executes three consecutive loop each traverse the array's items once. Therefore, its growth rate seems linear as in **Figure 4.**

As seen from **Figure 1** and **Figure 2,** worst case's lines are not exactly the same with the expected growth rates because expected growth rates are theoretical. In obtained values, there are other factors such as the computer's specifications that affect the execution time of the algorithms. However, these factors do not affect the time complexity of the algorithms.