Runtime analysis Report

Maria Bartnik, Hugo Madrid, Redvers Vecchio

The task was to analyze the runtime of the implemented BucketSort, that was taking an array of integers as an parameter, under two different scenarios;  
The test scenarios being:  
1. Input numbers are distributed evenly  
2. Input numbers are distributed not evenly.

The test cases were divided based on the size of the input array and the distribution of the array. We tested for small arrays (of sizes between 100 – 1000) and bigger arrays of 5000, 10000 and 50000; Each of the sizes had two variations- evenly distributed and not evenly distributed. The numbers in the arrays were generated randomly.

A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated  
  
The testing for both cases in all of the sizes occurred by starting counter of time in nano seconds, sort the given array using bucket sort implementation and then stop the timer. The final result was divided by 100 to make the numbers nicer to read.

The results of the tests were then plotted into a graph illustrating the runtime of BucketSort as a function of the size of the input array (n) for both scenarios.

A screenshot of a cell phone

Description automatically generated

From the results it can be noticed that the bigger the array size is the longer it took to sort the array using the BucketSorting. Although many factors affect the run time of the array, in this example it can be seen that the evenly distributed array took consistently longer to sort than the not evenly. As shown in the graph, this difference increases as the size of the array gets longer. Having an array of evenly distributed numbers produces buckets with similar number of elements. Having the buckets almost full means that the algorithm is closer to its optimal runtime, as less empty spaces in the buckets means less useless repetitions of the loop, thus sorting the array in a smaller amount of time.