Name: Bartosz Zmuda

Matriculation Number: 2576674z

Running Instructions: Download and unpack the zip file. Open the folder in an IDE of your choice.

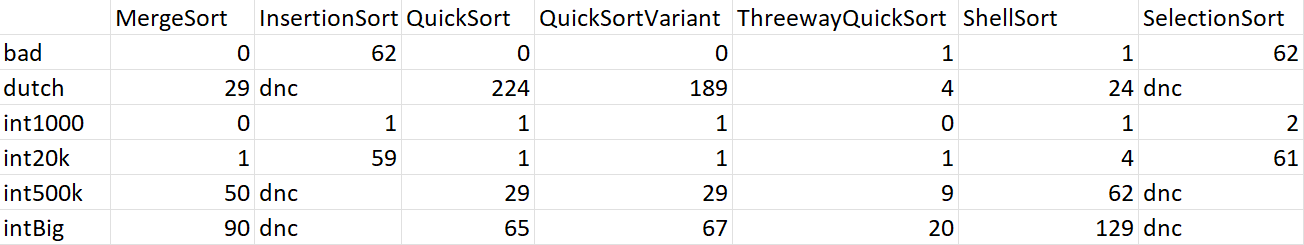
Part 1

In part 1 I implemented three variants of the QuickSort.

Here is how the ThreeWayQuickSort algorithm works. The sort() method takes 3 parameters, namely: an array of integers to sort,

Part 2

Below are presented the average running times of each algorithm for each of the data sets in milliseconds across a series of 5 runs.



As can be seen in the table and the clustered column chart, the ThreeWayQuickSort consistently outperforms all the other algorithms and is the fastest algorithm from the ones presented here. The QuickSort and the QuickSortVariant follow closely behind, with their performances being only a little slower than the the ThreeWayQuickSort, except for the dutch data set, where they are by far the slowest of the algorithms that completed their runs. This excludes InsertionSort and SelectionSort, both of which performed so poorly, their runtime went past the cutoff time. Insertion and Selection Sort are consistently the two worst-performing algorithms, not finishing on three separate occasions.

MergeSort and ShellSort have been faster than InsertionSort and SelectionSort on all occasions but have been generally slower than all the variants of QuickSort, except for the dutch data set, where they have outperformed QuickSort and QuickSortVariant by being significantly faster. Between those two, MergeSort is generally faster, except for the dutch data set where it has been outperformed by ShellSort.

Part 3

In part a, I have implemented a variation of MergeSort in a descending order that additionally finds the largest i elements. The, findTopILargestElements method takes an array of integers to be sorted and an integer i as its input. A new array is created, which allows us to find the top I elements of the array without modifying the array itself.

Then, the sortDescending method is called and takes as its parameters the newArray, integer 0 as well as the length of the array -1 in integer form, and lastly the integer i. The newArray is then sorted and merged recursively. Importantly, the algorithm stops when the top i numbers have been found, for efficiency. findTopILargestElements then returns the top I elements of the newArray, which we know are sorted in the descending order, and therefore satisfy the specification.