I fixed up some small spelling mistakes as well as added to a few of the sections. Just copy and paste these sections into the report. Feel free to edit and add to them if you pick up any mistakes or want to add/change what I’ve said.

Just a note on Choice of Problem Size section, this could be separate to brute and selection sections as the problem size is the same for both algorithms.

# Summary

The purpose of this report is to analyse and compare the average complexity of the selection median algorithm against a brute force solution. This report uses algorithm analysis techniques to experimentally determine the average case efficiency of both the selection median and brute force median algorithm. The expected theoretical efficiencies of the algorithms are evaluated using mathematical analysis and are contrasted against the computed experimental results. The results for each algorithm are also compared to each other and the most efficient algorithm is determined.

# Brute Force

The brute force median algorithm works by manually checking each value of the array, and determining if the elements’ position in a sorted array is in the median position of jk=2j. It does this by checking each element of the array against every other element of the array, and counting the number of elements that are less than the value, and the number of elements that are greater than the value.

Recalling that the median value of a list of numbers is the element that occurs in center position (rounded up for the purpose of this algorithm), the algorithm tests if a value is the median value by testing if half the elements in the array are less than the value. This can be seen in the pseudocode included in the next section

# Median Selection Algorithm

The selection algorithm for finding the median value of the array derives much of it's logic from the Quicksort algorithm. The selection algorithm uses a recursive method to determine the median of the array.

The recursive function first performs a partition on the array where all elements less than a pivot element are swapped until the pivot element swapped into its correct position. All elements with an index lower than that of the pivot, now also have a value which is less than that of the pivot. If the new position of the pivot is the middle of the array then the value of the pivot is returned as it is the median of the array. This check is the base case of the recursive function.

Otherwise, if the position of the pivot is less than or greater than that of the middle index then the recursive function is called again and then performs the partition on either the lower or upper partition of the array respectively. This is performed until the base case is met and the element in the middle of the array is returned.

It should be noted that there is no guarantee that the whole array will be completely sorted by this process. However, all elements at a lower index than that of the median will have a value that is lower than that of the median element and all elements with an index greater than that of the median element will have values that are also greater than the value of the median.

Furthermore, in contract to the brute force algorithm, when this algorithm is performed on an even array size, returns the value which is towards the lower section of the sorted array.

# Choice of problem size (identical for both algorithms)

The algorithms are tested using randomly generated arrays of sizes 1 through to 999 in steps of 3. Additionally, the test for each array size is repeated 2000 times in order to normalize the results.

The upper limit was chosen as array sizes greater than 1000 have brute force time complexities which are significantly large. This limits the number of averages that can be taken for each array size as the total time to run all tests increases to a value that is impractical to perform repeatedly.

The odd step size of 3 is chosen so that tests are performed for both odd and even array sizes.

# Choice of Basic Operations – Selection

The operation that best defines the complexity and running time of the selection median algorithm is the comparison A[j] < pivot\_val which is performed by the partition sort logic borrowed from the Quicksort algorithm. This comparison operation is performed more than any other operation in the algorithm - a minimum of n 􀀀 1 times, and a maximum of (n 􀀀 1)2 times.

# Images

Also see the svg images in the images folder







