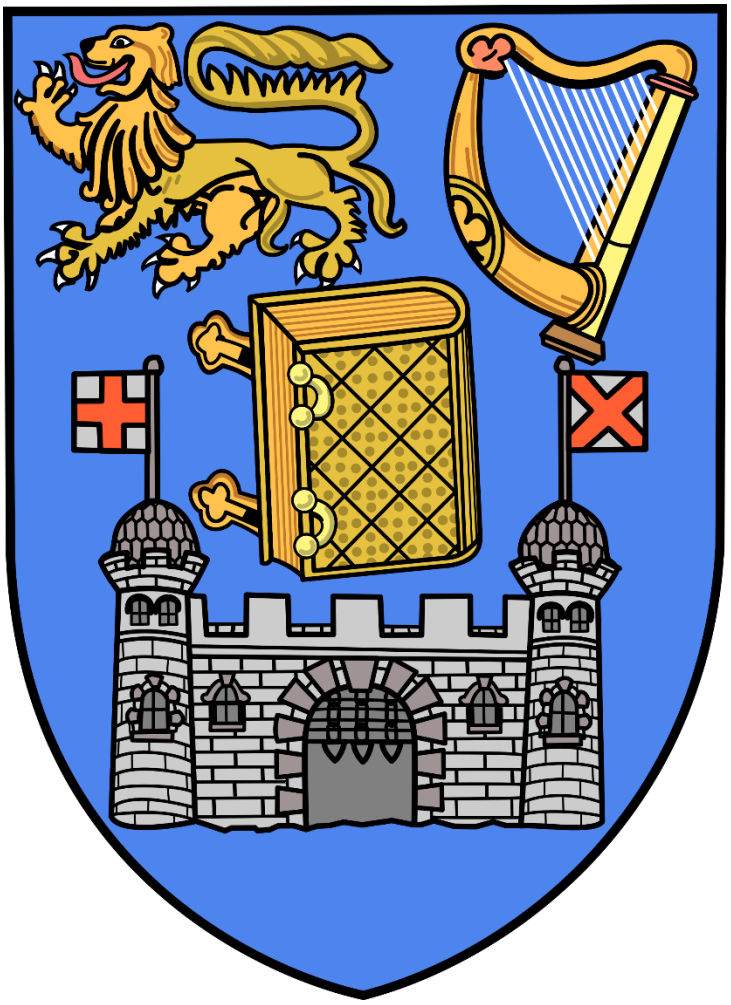


3D5A – Sorting Assignment

Name: Donal Tuohy

Student No: 14313774

Date: 15/11/16

Abstract

This report is an explanation and analysis of two types of sorting algorithms. They are:

- Quicksort
- Selection sort

The two have been implemented in a program for which the source code is attached and it is extensively commented on. Both algorithms have a probe count which allows the comparisons seen within this report. Best, average and worst case scenarios are also discussed for each.

It is shown that quicksort can be a much more efficient algorithm than selection sort.

Quicksort

Quicksort is a recursive divide and conquer sorting algorithm which works in two parts. Firstly, the input array is divided into values lower and higher than a certain number called “the pivot”. This is the divide part which does the majority of the work. The conquer section just passes the divided sections back to be divided once more. This continues until the whole array has been sorted and there is no more sections to divide.

The amount of probes for this algorithm depends on the arrangement of the values to be sorted. It could be sorted in the first call of the function it could take much longer.

The algorithm is coded into two functions for this lab. The first is the quicksort function which has two parameters. It takes in the first index of a section and the last. The second function, partition, is then called which takes in the same parameters. (Note: The array does not have to be passed into the functions as it is defined as a global array). The partition function then works through the array comparing each value to the pivot. Once the counter has reached the pivot, everything less than the pivot is on the left of the divide and everything greater on the right. The pivot is then slotted in between these and the divide index is passed back into two recursive quicksort functions. One for the left hand side and one for the right. It will stop when the same value is passed in twice to the quicksort function meaning there is nothing more to sort.

Selection Sort

Selection sort is a simplistic sorting algorithm which is called an “in-place algorithm” in computer science. It works by repeatedly finding the next value less than a set minimum. It then sets this value as minimum and then compares it with the rest of the array, witching the minimum as it goes. Once it has passed through the whole array. That minimum is set and in its final place. Then the next value up is set as minimum and the process is repeated until every index in set in place.

So for each value in the array you have to check the whole array. This leads to a best case scenario of (n^2). There is no way to terminate the sort if it has succeeded and it still has to

compare every value. This makes selection sort an unpopular sorting algorithm as it is so inefficient when large amounts of data have to be sorted.

One advantage however is that it is easily coded and recursion does not have to be used. You will see in the code that the selection sorting algorithm is coded in just eight lines. It just consists of two for loops iterating through the array.

Testing and Analysis

To compare the efficiency of these sorting algorithms, here is a table showing how the amount of probes can vary with different arrangements. Each time a test is run it is a completely different set of numbers.

Quicksort

Size of array	Test 1 probes	Test 2 probes	Test 3 probes
10	24	23	29
50	276	308	248
100	590	563	645
1000	9897	10561	12051

As you can see, for each test with the same size array, there can be a varying amount of probes. This is because the arrangement could be close to been sorted. The nearer the arrangement is to been sorted, the less probes quicksort will use.

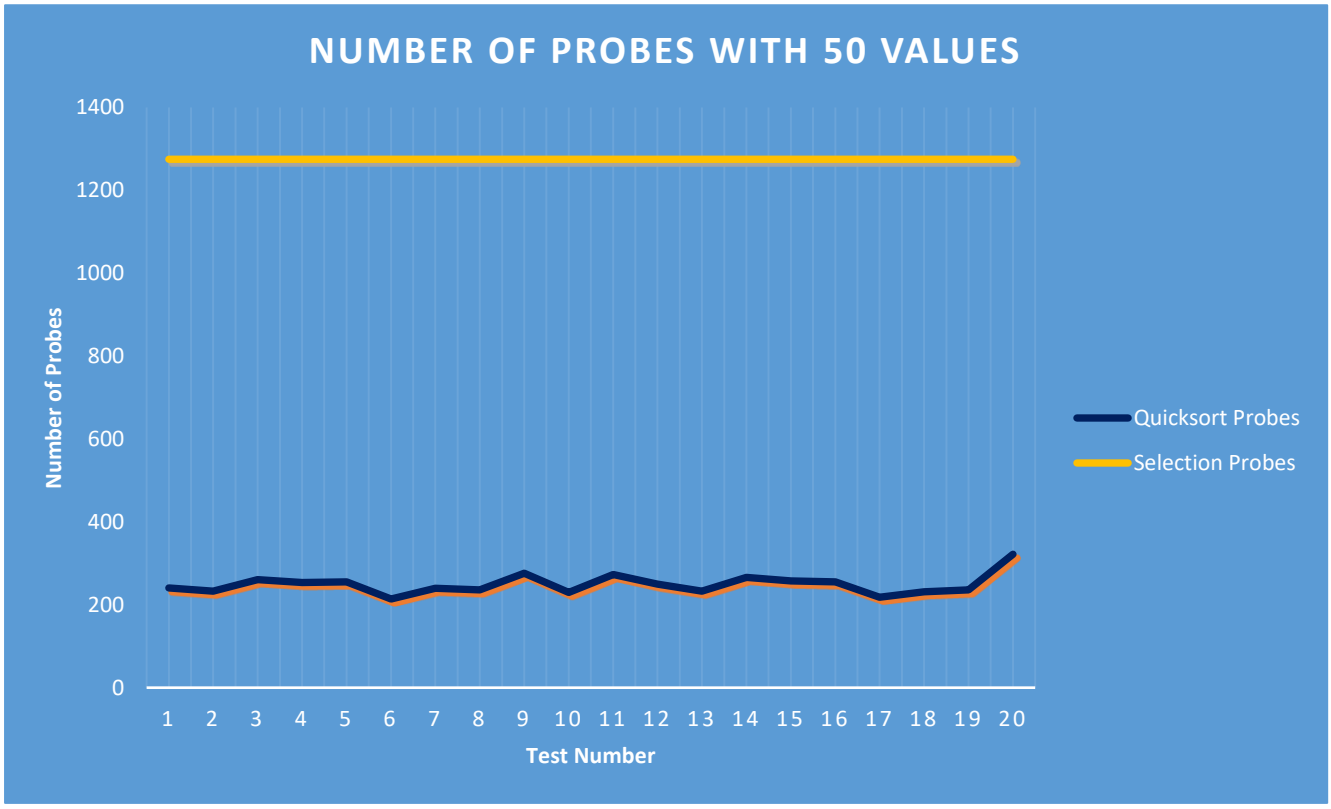
Selection Sort

Size of array	Test 1 probes	Test 2 probes	Test 3 probes
10	55	55	55
50	1275	1275	1275
100	5050	5050	5050
1000	500500	500500	500500

This shows however how inefficient selection sort can be. No matter what arrangement the array is in, it will always take the max number of probes as you can see above. The array could actually be fully sorted before it is passed into the algorithm and it would use the same amount of probes if the array was absolutely random.

Below is a table which shows the Big O Complexities. A graph is also attached on the next page showing the amount of probes per test for the two sorts. This will allow you to visualise the difference between the two.

Algorithm Name	Big O Complexity		
----------	**Best**	**Average**	**Worst**
Quicksort	O(n log(n))	O(n log(n))	$O(n^2)$
Selection Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$



Conclusion

After completing this assignment, one would realise the importance of efficient algorithms and how a well-designed one can greatly reduce the cost. At a small scale the difference can be negligible but once you increase that scale the difference between and efficient algorithm and an inefficient one is massive.

Quicksort is the most popular sorting algorithm due to its speed and speed is also the reason selection sort is very unpopular. There is other sorting algorithms such as Merge Sort, Insertion Sort and Bubble Sort. These have not been mentioned or studied in the report but they consist of many different techniques and all result in unique complexities. When deciding on which algorithm to use , one should look at several different types to find the best suited one to the task at hand.