Algorithm research – quicksort

An algorithm, according to the Random House Unabridged Dictionary, is a set of rules for solving a problem in a finite number of steps. One of the fundamental problems of computer science is sorting a set of items. The solutions to these problems are known as sorting algorithms and rather ironically, “ the process of applying an algorithm to an input to obtain an output is called a computation” [http://mathworld. wolfram.

com/Algorithm. html]. The quest to develop the most memory efficient and the fastest sorting algorithm has become one of the great mathematical challenges of the last half century, resulting in many tried and tested algorithms available to the individual who needs to sort a list of data. In fact new sorting algorithms are still being developed today, take for example the Library sort, which was published in 2004.

Of all the popular sorting algorithms, I have chosen to research and explain in detail an algorithm known as the ‘ Quicksort’. Quicksort is a popular and speedy sorting algorithm that is the multi-purpose, sorting algorithm of choice for many mathematicians and computer scientists. Though of course the choosing of an algorithm comes down to which algorithm is best suited to the clients needs, and is dependent on the specific set of data to be sorted, Quicksort has proven to fulfill the required criteria on many occasions. C. A. R.

Hoare developed the Quicksort algorithm in the year 1960, while he was working for a small, English scientific computer manufacturer named Elliott Brothers (London) Ltd. Sorting algorithms are designed to be fast, and efficient. To be able to sort a list of data as quickly as possible, using as little

memory as possible. To measure or classify an algorithm according to these two criteria, we measure the algorithm’s computational complexity. The computational complexity of a sorting algorithm is it’s worst, average and best behavior.

Sorting algorithms are generally classified by their computational complexity of element comparisons, against the size of the list. This is represented with what is known as ‘ Big O notation’, for example where the ideal behavior is O(n), most algorithm’s behavior is O(nlogn), and bad behavior is O(n? . O(n) behavior means that a sorting algorithm would take ten times longer to sort a list of one hundred elements, than it would to sort a list of one thousand elements. O(n?) behavior is quadratic behavior, and this would take one hundred times longer to sort a list of one hundred elements, than it would to sort a list of one thousand elements.

In Big O notation, the ‘ O’ symbol is an asymptotic upper bound, or the greatest element in a subset of another, partly ordered set. The ‘ n’ is the size of the list being sorted. The popular sorting algorithms are dived into two classes, O(n? , which are generally slower but simpler, and O(nlogn) which are faster but more complicated. The Quicksort algorithm is of the O(nlogn) class, and makes O(nlogn) comparisons on average.

In worst case scenarios, Quicksort makes O(n?) comparisons. In a very simplistic explanation, The Quicksort algorithm sorts a list of data in three steps. These three steps are listed below: 1: Select a ‘ Pivot’. The pivot is an element from the list.

2: Reorder the list by putting all elements with values that are less than the pivot, before the pivot; and all elements with values greater than the pivot after the pivot. Elements with values equal to the pivot can go either way. After the list has been reordered in such a way, the pivot will be in its final position, this operation is known as the ‘ partition operation’. 3: Recursively sort the elements of lesser value than the pivot, and the elements of greater value than the pivot.

The Quicksort algorithm is a comparison sort. Comparison sorts compare the values of elements in a set of data by a single abstract comparison operation, which is most often a ‘ less than or equal to’ operation. The Quicksort algorithm is not a stable sorting algorithm. In a stable sorting algorithm, if two elements have equal values, they will maintain their relative order when the list is sorted.

The Quicksort is also massively recursive. “ Recursion, in mathematics and computer science, is a method of defining functions in which the function being defined is applied within its own definition. The term is also used more generally to describe a process of repeating objects in a self-similar way” [http://en. wikipedia. org/wiki/Recursion].

Sorting algorithms can also be classified by the amount of memory space a computer will need to sort the algorithm. The simplest form of the Quicksort algorithm, while being fast, requires ? (n) extra storage space, which is comparatively not very good. There is however, a more complicated version of Quicksort that uses an in-place partition algorithm. An in-place algorithm is one that doesn’t require extra space to sort the array with sub arrays and

such. Instead it swaps the elements around inside the original array and thereby only needing the space of the original array, plus a small amount of working space for local variables etc as all sorts do.

The in-place version can sort a list using O(nlogn) space on average. The Quicksort algorithm uses what is known as a ‘ Divide and conquer’ strategy to sort lists of data. Using the ‘ divide and conquer’ strategy, “ we must divide the problem into two smaller sub-problems, solve each of them recursively, and then meld the two partial solutions into one solution to the full problem” [http://www2. toki. or.

id/book/AlgDesignManual/BOOK/BOOK2/NODE53. HTML]. Another point of interest for the Quicksort sorting algorithm, is that is can easily be parallelized. In computer science, “ a parallel algorithm, as opposed to a traditional sequential algorithm, is one which can be executed a piece at a time on many different processing devices, and then put back together again at the end to get the correct result” [http://en. wikipedia. org/wiki/Parallel_algorithm].

This means when running on a system with multiple processors, Quicksort can split its sub-set’s to be sorted by individual processors and greatly speed up the time in which it sorts the set of elements. In summary, Quicksort is a comparison sort that is massively recursive, not stable and easily parallelized. Quicksort makes O(nlogn) comparisons on average and at worst makes O(n?) comparisons. Quicksort is one of the fastest sorting algorithms and a very popular choice for sorting sets of data, generally only competing

with other sorts for their better worst case behaviors e. g. Heapsort and Mergesort, or when a stable sort is needed eg.

Mergesort.