Definition

Quicksort algorithm is an efficient sorting algorithm that is used to place the elements in the array in a systematic order based on comparison.

Order of complexity

Quicksort algorithm in it’s best case shows order O(nlogn) and worst case O(n^2) . For this assignment the most efficient algorithm was the one that used Recursion O(nlogn) as oppose to the Iterative algorithm which is of order O(n^2).

Algorithm

1. Choose an element, called the pivot. In my C++ as well as Mips implementation I chose the last value in the array to be the pivot.
2. Then reorder the array so that all the elements with values less than that of the pivot came before the pivot, while all the elements with values greater than or equal to came to the right of the pivot. This is known as Partitioning the array. This involves using a partitioning index
3. Recursively apply the above steps to the sub array of elements with smaller

values and separately to the sub-array of elements with greater values.

Simple Quicksort Algorithm

**function** *quicksort*(array)

less, equal, greater **:=** three empty arrays

**if** length(array) > 1

pivot **:=** *select any element of* array

**for each** x **in** array

**if** x < pivot **then add** x **to** less

**if** x = pivot **then add** x **to** equal

**if** x > pivot **then add** x **to** greater

quicksort(less)

quicksort(greater)

array **:=** concatenate(less, equal, greater)

Better Quicksort Algorithm

**function** *quicksort*(array)

**if** length(array) > 1

pivot **:=** *select any element of* array

left **:= first index of** array

right **:=** **last index of** array

**while** left ≤ right

**while** array[left] < pivot

left := left + 1

**while** array[right] > pivot

right := right - 1

**if** left ≤ right

**swap** array[left] **with** array[right]

left := left + 1

right := right - 1

quicksort(array **from first index to** right)

quicksort(array **from** left **to last index**)