1. Situations for Best and Worst Case for QuickSort Algorithm

* Best Case
* The pivot to rely on for partitioning always stays in the middle during swapping.
* Few Swaps as possible.

Eg: First swap the pivot into the last place then arrange the others so that the elements smaller than the povot come before the larger elements and finally swap the pivot from its position with the first of the larger elements. Doing this recursively

A different method is to put the pivot into the first slot before arranging and swap it with the last not exceeding the pivot.

* Worst Case
* This is when the pivot is destined to be at one end of the array. That is, the pivot is either the smallest or largest value among the elements of arrays.
* When the list is either arranged in ascending or descending order.

1. Analysis (in terms of time complexity) of best and worst case scenarios of quicksort algorithm.

* Worst Case: O(n2)

Original call in terms of its partition:

cn where c = a constant   
 n = number of recursive call

Thus for (n-1) we have c(n-1)  
Hence for (n-1), (n-2), (n-3), …...1  
cn + c(n-1) + c(n-2) + ….. + 2c = c(n + (n-1) + (n-2) +.......+2)  
 = c((n+1)(n/2) -1 )

Using big-oh notation, we have O(n2)

* Best Case: O(nlog2n)

Having either an odd or even number of elements, with pivot right in the middle, produces at most n/2 elements.

Hence, total partitioning time for all subproblems of size:

cn, <2.n/2; <4cn/4; <8cn/8,...... = cn

Thus for every n>N, we have nc = cn

Using big-oh notation we have O(nlog2n)

ID: 10682923