Name: Vivek Vikram Pundkar

Roll no: 77

GR no: 11910860

Division: C

Batch: 3

Case 1. Normal partition method: Quick select

Case 2. Randomized partition method

Case 3. Using median

Case 1. Normal partition method

Quicksort algorithm is a sorting algorithm which separates the array of elements into two partitions and then sort each partition recursively. It uses divide and conquer strategy for sorting. Performance of this algorithm based on the pivot we chose. In simple quicksort, pivot is the last or first element.

Analysis:

Worst Case:

$T(n) = T(n-1)+O(n)$

$T(n) = T(n-1)+n$

$T(n-1)= T(n-2)+n-1$

By substitution,

$T(n) = T(n-2)+n-1+n$

$T(n) = T(n-3)+n-2+n-1+n$

Similarly Substituting K,

$T(n) = T(n-k) + n-(k-1) + n-(k-3)....\ n-1$

Considering $n-k = 1$

$T(n)= T(1) +2+3+4+... +K$

$T(n)= k(k+1)/2$

$T(n) = K^2$

$T(n)= o(n^2)$ ….(Worst case complexity)

Average Case:

$T(n)= 2*T(n/2)+O(n)$

By using masters theorem we get,

$T(n)= O(nlogn)$

Best Case: O(n* log n) When pivot is chosen as the middle element every iteration.

Input: 7, 10, 4, 3, 20, 15

Output:10

Case 2. Using randomized Quick sort

Step 1 – First we select a random element as the pivot.

Step 2 – Then we swap the first element of the array with the random pivot.

Step 3 – Then we implement the partition function as the same as general Quicksort but now using a random pivot.

Step 4 – After partitioning we call Quicksort recursively on the sub array.

Input: 7, 10, 4, 3, 20, 15

Output: 10

Time taken by QuickSort, in general, can be written as follows.

T(n) = T(i) + T(n-i-1) + O(n)

The first two terms are for two recursive calls, the last term is for the random partition process. k is the number of elements which are smaller than pivot.

Best case: The best case is the same as general Quicksort when the pivot element is in the middle of the array and the partitioning divides the array in two equal halves.

2T(n/2) + O(n)

The solution of above recurrence is O(nlogn)

Worst case: In worst case, each partition divides an array such that one side has n/4 elements and other side has 3n/4 elements. The worst case height of a recursion tree is Log ¾ n which is O(Log n).

T(n) < T(n/4) + T(3n/4) + O(n)

T(n) < 2T(3n/4) + O(n)

Solution of the above recurrence is O(n Log n) .

Case3: Using Median

T(n) =c*n/5 + c(7*n/10 +6) +O(n)

= c*n/5 + 7*n*c/10 +6*c +O(n)

=9*c*n/10 +7*C+O(n)

=c*n

Time complexity: O(N)