Cs501
Mansi Shah
(19526)
Week4: Homework 4

Q-13) What is the Big-O time complexity of QuickSort?

Ans: For the given program of Quick-Sort first two terms are for recursive function and the last term for the partition process. Where K is the number of elements which values are smaller than pivot.

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

The time taken by QuickSort depends upon the input array and partition strategy. There are three cases to find a Big-O of Quick sort.

Worst Case*:* This Case apply when the partition process always consider greatest /smallest element as pivot. In given program partition strategy where last element is always consider as pivot, the worst case would occur when the array is already sorted in increasing or decreasing order. Following is recurrence for worst case.

$T(n) = T(0) + T(n-1) + O(n)$

which is equivalent to

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

Big-O notation for given program is: $O(n^2)$

Best Case: The best case occurs when the partition process always picks the middle element as pivot. Following is recurrence for best case.

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

For Best case it is $O(n*\log n)$.

Average Case:
We can get an idea of average case by considering the case when partition puts O(n/9) elements in one set and O(9n/10) elements in other set. Following is recurrence for this case.

$T(n) = T(n/9) + T(9n/10) + O(n)$

For Average case it is O(nLogn)

Quick sort for n items, generally it performs faster than $O(n^2)$ in O(n*Log n).

Q-17) Dominant term(s) and Big-O

Expression	Dominant term(s)	O(...)
$5 + 0.001n^3 + 0.025n$	$0.001n^3$	$O(n^3)$
$500n + 100n^{1.5} + 50n \log_{10}n$	$100n^{1.5}$	$O(n^{1.5})$
$0.3n + 5n^{1.5} + 2.5 \cdot n^{1.75}$	$2.5 \cdot n^{1.75}$	$O(n^{1.75})$
$n^2 \log_2 n + n(\log_2 n)^2$	$n^2 \log_2 n$	$O(n^2 \log n)$
$n \log_3 n + n \log_2 n$	$n \log_3 n + n \log_2 n$	$O(n \log n)$
$3 \log_8 n + \log_2 \log_2 \log_2 n$	$3 \log_8 n$	$O(\log n)$
$100n + 0.01n\ 2$	$0.01n2$	$O(n^2)$
$0.01n + 100n^2$	$100n^2$	$O(n^2)$
$2n + n^{0.5} + 0.5n^{1.25}$	$0.5n^{1.25}$	$O(n^{1.25})$
$0.01n \log_2 n + n(\log_2 n)^2$	$n(\log_2 n)^2$	$O(n(\log n)^2)$
$100n \log_3 n + n^3 + 100n$	n^3	$O(n^3)$
$0.003 \log_4 n + \log_2 \log_2 n$	$0.003 \log_4 n$	$O(\log n)$

Q-28) What is the Big-O Time Complexity Analysis of BubbleSort?

Ans: The algorithm works by comparing each item in the list with the item next to it, and swapping them if required. In other words, the largest element has bubbled to the top of the array. The algorithm repeats this process until it makes a pass all the way through the list without swapping any items.

For the given example Bubble sort
Worst & Average Case Time Complexity: O(n*n).
Worst case occurs when array is reverse sorted.
Best Case Time Complexity: O(n). Best case occurs when array is already sorted.

Q-29) What is the Big-O Time Complexity Analysis of Linear Search?

Ans: For Linear Search, the worst case happens when the element to be searched (x in the above code) is not present in the array. When x is not present, the search() functions compares it with all the elements of arr[] one by one.

Therefore, the worst case time complexity of linear search would be Θ(n).