Quicksort

Sorting Problem

`Input:` An array of numbers a[1 ... n]

`Output` Sorted array

Algorithm

We know in divide and conquer we have to split the problem into subproblems. Then solve these smaller subproblems(easier).

1. Choose a any index value(randomly/always first/always last) as pivot
2. Partition the array in following manner

   Left partition :- numbers smaller than pivot

   Right partition :- array of numbers greater than pivot
3. This can be done by easily comparing each element with pivot
4. Solve the left partition recursively
5. Solve the right partition recursively

Time analysis

Master theorem Analysis for Quicksort(Random Pivot)

This will be probabilistic. In the expected worst case each partition will divide always at least n/4 elements in one side .

Partition we compare with each element of array so complexity = O(n)

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

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

Using Master theorem Complexity = O(n logn) (Expected worst case not absolute)

In the absolute worst -case (very unlucky/ negligible probability of happening). Always divide into 1 and n-1.

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

Resulting in time-complexity = O(n^2).

Master theorem equation for Quicksort(Always last)

Partition we compare with each element of array so complexity = O(n)

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

where k is between 1 and n-1

Considering the worst case k =1

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

using master theorem in worst case complexity = O(n^2)

Considering the worst case k =n/2

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

using master theorem in best case complexity = O(nlogn)

Considering the avg case k =n/9

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

using master theorem in Average case complexity = O(nlogn)

Space complexity Analysis

Since we are only partitioning in the existing array by using swap elements. We only need 1-D array of length n to store the input array.

Space complexity = O(n)