**Quick Sort**

**Divide And conquer** – Quick and merge sort

**Best case time complexity – (Olong(n))**

|  |  |  |
| --- | --- | --- |
|  | **Quick Sort** | **Merge Sort** |
| **Best Time Complexity** | O(nlog(n) | O(nlog(n) |
| **Worst case Cmplexity** | O(n^2) | O(nlog(n) |
| **Space Complexity** | O(1) | O(n) |
|  |  |  |

**Algorithm**-

1. Select an element as pivot
2. Find its position by putting all the smaller at left and greater at right
3. Recurse the same on left and right version of array

**Two ways to select the pivot**

1. Right most
2. Randomised

**Right most Pivot** – It is good when the right most element is an element that has n/2 or n-1/2 elements in right and left subtree

**Randomised Pivot** – This approach is best when the right most element has 0 or 1 element in its left or right subarray as then this kind of wuick sort will lead to average or wordt case

We make a random element as pivot such that the left and right subtree have more than 1 element each.

This improves the time complexity from O(n^2) to O(nlog(n)).

**So Quick sort is better than merge sort** bcz it has **space Complexity O(1)** and also it is **tail recursive**