Q) Why quick sort is preferred over MergeSort for sorting Arrays?

Sol: (i) Quick sort in its general form is an in-place sort (where as merge sort requires O(N) extra space, where N denoting the array size which may be quite expensive!

Allocating and de-allocating the extra space used for merge sort increases the running time of the algorithm.

(ii) Comparing average complexity we find that both type of sort have O(nlgn) average complexity but the constants differ. For arrays, merge sort loses due to use of extra O(N) storage space.

randomized Quick sort works well in practice.

(iii) Quick sort is also a cache friendly sorting algorithm as it has good locality of reference when used for arrays.

(iv) Quick Sort is also tail recursive, tail call optimization is done.

In linked list, we can insert item in the middle in O(1) extra space & O(1) time. Therefore merge operation of merge sort can be implemented without extra space for linked list.

In arrays, we can do random access as elements are continuous in memory but can't do random access in linked list.

Quick Sort requires a lot of this kind of acces.

In linked list to acces i^{th} index, we have to travel each and every node from the head to i^{th} node as we dont have continuous block of memory. Therefore, the overhead increases for quicksort. Merge sort accesses data

Sequentially and the need of random accen is low: