**Why is**[**Merge Sort**](http://geeksquiz.com/merge-sort/)**preferred for Linked Lists?**

* In case of [linked lists](http://geeksquiz.com/linked-list-set-1-introduction/) the case is different mainly due to difference in memory allocation of arrays and linked lists. Unlike arrays, linked list nodes may not be adjacent in memory.
* Unlike array, in [linked list](http://geeksquiz.com/linked-list-set-1-introduction/), we can insert items in the middle in O(1) extra space and O(1) time if we are given reference/pointer to the previous node. Therefore merge operation of merge sort can be implemented without extra space for linked lists.
* In arrays, we can do random access as elements are continuous in memory. Let us say we have an integer (4-byte) array A and let the address of A[0] be x then to access A[i], we can directly access the memory at (x + i\*4). Unlike arrays, we can not do random access in linked list.
* Quick Sort requires a lot of this kind of access. In linked list to access i’th index, we have to travel each and every node from the head to i’th node as we don’t have continuous block of memory. Therefore, the overhead increases for quick sort. Merge sort accesses data sequentially and the need of random access is low.

**Why is**[**Quick Sort**](http://geeksquiz.com/quick-sort/)**preferred for arrays?**

* Quick Sort in its general form is an in-place sort (i.e. it doesn’t require any extra storage) whereas merge sort requires O(N) extra storage, 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.
* Comparing average complexity we find that both type of sorts have O(NlogN) average complexity but the constants differ. For arrays, merge sort loses due to the use of extra O(N) storage space.
* Most practical implementations of Quick Sort use randomized version. The randomized version has expected time complexity of O(nLogn). The worst case is possible in randomized version also, but worst case doesn’t occur for a particular pattern (like sorted array) and randomized Quick Sort works well in practice.
* Quick Sort is also a cache friendly sorting algorithm as it has good [locality of reference](http://en.wikipedia.org/wiki/Locality_of_reference) when used for arrays.
* Quick Sort is also [tail recursive](https://www.geeksforgeeks.org/tail-recursion/), therefore tail call optimizations is done.

**Why quicksort is better than mergesort ?**

Despite of better worst case performance of merge sort, [quicksort](https://www.geeksforgeeks.org/quick-sort/) is considered better than [mergesort](https://www.geeksforgeeks.org/merge-sort/). There are certain reasons due to which

quicksort is better especially in case of arrays:

1. **Auxiliary Space :** Mergesort uses extra space, quicksort requires little space and exhibits good cache locality. Quick sort is an in-place sorting algorithm. In-place sorting means no additional storage space is needed to perform sorting. Merge sort requires a temporary array to merge the sorted arrays and hence it is not in-place giving Quick sort the advantage of space.
2. **Worst Cases :**The worst case of quicksort **O(n2)** can be avoided by using randomized quicksort. It can be easily avoided with high probability by choosing the right pivot. Obtaining an average case behavior by choosing right pivot element makes it improvise the performance and becoming as efficient as Merge sort.
3. **Locality of reference :** Quicksort in particular exhibits good cache locality and this makes it faster than merge sort in many cases like in virtual memory environment.
4. **Merge sort is better for large data structures:**Mergesort is a stable sort, unlike quicksort and heapsort, and can be easily adapted to operate on linked lists and very large lists stored on slow-to-access media such as disk storage or network attached storage. Refer [this](https://www.geeksforgeeks.org/why-quick-sort-preferred-for-arrays-and-merge-sort-for-linked-lists/) for details

**/\* The std::sort() function which is present in C++ STL is a hybrid sorting algorithm provides average and worst case time complexity of O(nlogn). The sorting algorithm which it uses is called Introsort.  
Introsort is combination of both quicksort and heapsort, It begins with quicksort and switch to heapsort if recursion depth exceeds a level based on the number of elements being sorted.**

**\*/**