0.1 Merge Sort

Time Complexity: Sorting arrays on different machines. Merge Sort is a recursive algorithm and time complexity can be expressed as following recurrence relation.
$T(n) = 2T(\frac{n}{2}) + \theta(n)$
The above recurrence can be solved either using Recurrence Tree method or Master method. It falls in case II of Master Method and solution of the recurrence is $\theta(n \log n)$. Time complexity of Merge Sort is $\theta(n \log n)$ in all 3 cases (worst, average and best) as merge sort always divides the array in two halves and take linear time to merge two halves.

Auxiliary Space: $O(n)$

Algorithmic Paradigm: Divide and Conquer

Sorting In Place: No in a typical implementation

Stable: Yes

Applications of Merge Sort

Merge Sort is useful for sorting linked lists in O(nLogn) time.In case of linked lists 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, we can insert items in the middle in O(1) extra space and O(1) time. 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.

Inversion Count Problem Used in External Sorting

0.2 Heap Sort

Notes: Heap sort is an in-place algorithm. Its typical implementation is not stable, but can be made stable (See this)

Time Complexity: Time complexity of heapify is $O(\log n)$. Time complexity of createAndBuildHeap() is $O(n)$ and overall time complexity of Heap Sort is $O(n \log n)$.

Applications of HeapSort 1. Sort a nearly sorted (or K sorted) array 2. k largest(or smallest) elements in an array

Heap sort algorithm has limited uses because Quicksort and Mergesort are better in practice. Nevertheless, the Heap data structure itself is enormously used. See Applications of Heap Data Structure