



Dash



All



Articles



Videos



Problems



Quiz

<< Prev

Next >>

Merge Sort Analysis

Analysis of Merge Sort:

A merge sort consists of several passes over the input. The first pass merges segments of size 1, the second merges segments of size 2, and the

i_{th}

pass merges segments of size 2^{i-1} . Thus, the total number of passes is $\lceil \log_2 n \rceil$. As merge showed, we can merge two sorted segments in linear time, which means that each pass takes $O(n)$ time. Since there are $\lceil \log_2 n \rceil$ passes, the total computing time is $O(n \log n)$.

Applications of Merge Sort:

- Merge Sort is useful for sorting linked lists in $O(N \log N)$ time. In the case of linked lists, the case is different mainly due to the difference in memory allocation of arrays and linked lists. Unlike arrays, linked list nodes may not be adjacent in memory. Unlike an array, in the linked list, we can insert items in the middle in $O(1)$ extra space and $O(1)$ time. Therefore, the merge operation of merge sort can be implemented without extra space for linked lists.

In arrays, we can do random access as elements are contiguous 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 the linked list. Quick Sort requires a lot of this kind of access. In a 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 a contiguous block of memory. Therefore, the overhead increases for quicksort. Merge sort accesses data sequentially and the need of random access is low.

- [Inversion Count Problem](#)
- Used in [External Sorting](#)

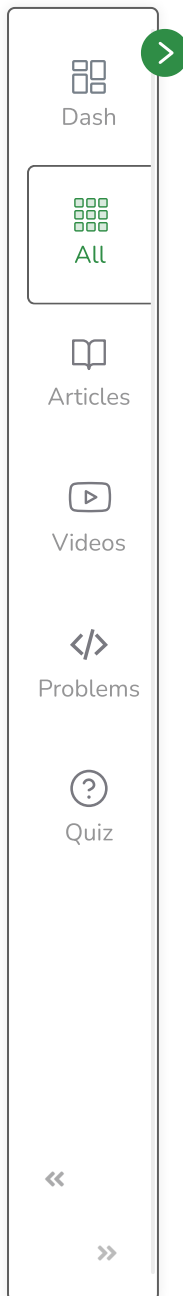
Advantages of Merge Sort:

- Merge sort has a time complexity of $O(n \log n)$, which means it is relatively efficient for sorting large datasets.
- Merge sort is a stable sort, which means that the order of elements with equal values is preserved during the sort.
- It is easy to implement thus making it a good choice for many applications.
- It is useful for external sorting. This is because merge sort can handle large datasets, it is often used for external sorting, where the data being sorted does not fit in memory.
- The merge sort algorithm can be easily parallelized, which means it can take advantage of multiple processors or cores to sort the data more quickly.
- Merge sort requires relatively few additional resources (such as memory) to perform the sort. This makes it a good choice for systems with limited resources.

Drawbacks of Merge Sort:

- Slower compared to the other sort algorithms for smaller tasks. Although efficient for large datasets it's not the best choice for small datasets.
- The merge sort algorithm requires an additional memory space of $O(n)$ for the temporary array. This is to store the subarrays that are used during the sorting process.
- It goes through the whole process even if the array is sorted.
- It requires more code to implement since we are dividing the array into smaller subarrays and then merging the sorted subarrays back together.
- [Recent Articles on Merge Sort](#)
- [Coding practice for sorting.](#)
- [Quiz on Merge Sort](#)

Solution of the drawback for additional storage:




Use linked list.


Mark as Read


 Report An Issue


If you are facing any issue on this page. Please let us know.






Dash


All


Articles


Videos


Problems


Quiz

