Hybrid Algorithm

My Problems:

- The best sorting complexity i can get from given algorithms is n*log(n) from Quick_sort and Merge_sort. but both algorithms have some problems:
    1. Quick_sort worst case is n^2 when i get several bad pivot points and it increases the recursion stack space too.
    2. Merge_sort is doing much comparisons. so its n*log(n) not efficient mush.

The Goal of Hybrid Algorithm:

- Solve the recursion stack space problem and the worst case of time complexity for Quick_sort.
- Make the complexity n*log(n) for all cases and better than Merge_sort and Quick_sort.

Algorithms used:

1. **Quick sort**: Used as Basic Algorithm.
2. **Insertion**: Used When partition size is less than 16.
    a. 16 is the best size based on researches of researchers.
3. **Merge sort**: Used when the depth of tree becomes 2*log(n)
    a. we use this depth because after more 2*log(n) depth the Quick sort becomes not efficient for the stack space and time complexity.

Why those Algorithms:

1. **Quick sort**: because it separates the problem to two independent problems so the processor can run both problems in parallel fastly and use the cache memory efficiently.
2. **Insertion**: because it's the best sorting Algorithm for small sizes.
3. **Merge sort**: to avoid the problem of the recursion stack space. instead of that using much stack space we use less stack space and more heap space when we take copy of data to Merge thim (it's better to use heap sort instead of merge sort because heap sort doesn't need much heap space as merge sort).

Time Complexity:

- Worst Case, Best case and Average case: n*log(n).