Trade Off Analysis

Sorting Algorithm	Time Complexity	Comparisons / Swaps	Memory Usage
Quick	O(n log n)	High for worst case	O(1)*
Merge	O(n log n)	Moderate	O(n log n)**
Heap	O(n log n)	Moderate	O(1)***
Custom In-Place Merge	O(n log^2 n)	Moderately high	O(1)****
Insertion	O(n^2)	Low for near-sorted	O(1)
Selection	O(n^2) always	High	O(1)

*Because quick sort is a recursive function, then it will require O(log n) stack space, with O(n) worst case.
**The merge sort we implemented in class has a space complexity of O(n log n), however merge sort can be improved to a space complexity of (n), while maintaining the time complexity of O(n log n).
***The heap sort implementation I made did not use the input array for the heap, resulting in O(n) space complexity.
****In reality, a true in-place merge sort would be much more complicated. In part of brevity, I found a solution that gave up some time complexity O(n log^2 n) and stack space O(log^2 n). Applicable? No.

Sorting Algorithm	Randomized Data	Sorted Data	Reverse Sorted Data	Near Sorted Data	Small Datasets	Large Datasets
Quick	Moderately Fast	Moderately Fast	Slow*	Moderately Fast*	Fast	Fast
Merge	Moderately Fast	Moderately Fast	Moderately Fast	Moderately Fast	Moderate	Moderate
Heap	Moderately Fast	Moderately Fast	Moderately Fast	Moderately Fast	Moderate	Moderate
Custom In-Place Merge	Moderate	Moderate	Moderate	Moderate	Moderate	Moderate
Insertion	Slow	Fast	Slow	Fast	Fast	Slow
Selection	Slow	Slow	Slow	Slow	Moderate	Slow

*Quick sort’s factors depend on optimizations regarding pivot selection.

Sorting Decision Framework

1. Is the dataset small?
    a. Yes: Use Insertion Sort
    b. No: Proceed to Step 2
2. Is stability required?
    a. Yes: Use Merge Sort
    b. No: Proceed to Step 3
3. Is memory usage a concern?
    a. Yes: Use Quicksort
    b. No: Proceed to Step 4
4. Is in-place sorting needed?
    a. Yes: Use Heapsort*
    b. No: Proceed to Step 5
5. Is the dataset nearly sorted?
    a. Yes: Use Insertion Sort
    b. No: Use Merge Sort

*Use an in-place heap sort algorithm.