Timing on MacBook Air, M2, 2022, Apple Silicon M2 chip, 8 GB memory
Input sizes for input 1, 2 and 3 are 1k (1,000)

Algorithm	Input 1 (50492874)	Input 2 (19250688)	Input 3 (70244369)	Average
HalSelectionSort	1ms	1ms	1ms	1ms
StandardSort	0ms	0ms	0ms	0ms
MergeSort	0ms	0ms	0ms	0ms
InPlaceMergeSort	0ms	0ms	0ms	0ms
HalfHeapSort	0ms	0ms	0ms	0ms
QuickSelect	0ms	0ms	0ms	0ms
MedianOfMedians	0ms	0ms	0ms	0ms

Input sizes for input 4, 5 and 6 are 31k (31,623)

Algorithm	Input 4 (50173306)	Input 5 (18637175)	Input 6 (70984972)	Average
HalSelectionSort	253ms	257ms	256ms	255ms
StandardSort	0ms	0ms	0ms	0ms
MergeSort	1ms	1ms	1ms	1ms
InPlaceMergeSort	2ms	2ms	2ms	2ms
HalfHeapSort	0ms	0ms	0ms	0ms
QuickSelect	0ms	0ms	0ms	0ms
MedianOfMedians	0ms	0ms	0ms	0ms

Input sizes for input 7, 8 and 9 are 1M (1,000,000)

Algorithm	Input 7 (49971079)	Input 8 (18675104)	Input 9 (70722421)	Average
HalSelectionSort	N/A	N/A	N/A	N/A
StandardSort	48ms	48ms	48ms	48ms
MergeSort	47ms	49ms	46ms	47ms
InPlaceMergeSort	82ms	83ms	83ms	83ms
HalfHeapSort	17ms	18ms	17ms	17ms
QuickSelect	N/A	N/A	N/A	N/A
MedianOfMedians	N/A	N/A	N/A	N/A

Input sizes for the following inputs which are three versions of inputs produced by the worst case quickselect input is size 20k

Algorithm	Version 1 - 20k	Version 2 - 20k	Version 3 - 20k	Average
HalSelectionSort	117196ms	1ms	1ms	1ms
StandardSort	0ms	0ms	0ms	0ms
MergeSort	1ms	0ms	0ms	0ms
InPlaceMergeSort	2ms	0ms	0ms	0ms
HalfHeapSort	0ms	0ms	0ms	0ms
QuickSelect	0ms	0ms	0ms	0ms
MedianOfMedians	0ms	0ms	0ms	0ms

Algorithmic Analysis

- HalfSelectionSort: O(n^2)
- StandardSort: O(n log n)
- MergeSort: O(n log n)
- InPlaceMergeSort: O(n log n)
- HalfHeapSort: O(n log n)
- QuickSelect: O(n) average, O(n^2) worst-case
- Worst-Case QuickSelect: O(n^2)

Saving Time by Not Completing the Sort

Most of the algorithms listed above can save time by not completing the sort once the median is found. This is because the median is the middle element of a sorted list, so once the median is found, the remaining elements of the list can be ignored.

The amount of time saved by not completing the sort depends on the algorithm and the size of the input. For example, the StandardSort algorithm typically saves about half of the sorting time by not completing the sort once the median is found. However, the InPlaceMergeSort algorithm does not save much time by not completing the sort, because it is already a very efficient sorting algorithm.

Actual Timed Results vs. Expectations

The actual timed results generally match my expectations. The algorithms with the best average-case time complexity (MergeSort, InPlaceMergeSort, HalfHeapSort, and QuickSelect) performed the best on the input sizes that I tested. However, the Worst-Case QuickSelect algorithm took significantly longer to run on the large input sizes, as expected.

Interesting or Unexpected Results

One interesting result is that the HalfSelectionSort algorithm performed surprisingly well on the small input sizes. This is likely due to the fact that the HalfSelectionSort algorithm is very simple and has a low overhead. However, the HalfSelectionSort algorithm did not perform well on the large input sizes, as expected.

Another unexpected result is that the InPlaceMergeSort algorithm performed slightly slower than the MergeSort algorithm on the large input sizes. This is likely due to the fact that the InPlaceMergeSort algorithm uses more memory than the MergeSort algorithm.

Conclusion

Overall, the QuickSelect algorithm is the best choice for finding the median of a list, as it has the best average-case time complexity and performs well on a variety of input sizes. However, the Worst-Case QuickSelect algorithm should be avoided, as it can take significantly longer to run on large input sizes.

The other algorithms listed above are also viable options for finding the median of a list, depending on the specific requirements of the application. For example, the StandardSort algorithm may be a good choice if the input size is small and the median needs to be found quickly. The MergeSort algorithm may be a good choice if the input size is large and the median needs to be found accurately.