Alikhan’s Algorithm (Selection Sort)

1. Algorithm Overview

Selection Sort is a simple sorting algorithm. On each step it looks for the smallest element in the unsorted part of the array and swaps it with the current element. After each pass, one more element is in its correct place.

In my partner’s code this algorithm is implemented together with a PerformanceTracker, which measures comparisons, swaps, array accesses and execution time. This makes it possible to check both theory and practice.

2. Complexity Analysis

- Best Case: $\Omega(n^2)$. Even if the array is sorted, the algorithm still makes all comparisons.
- Average Case: $\Theta(n^2)$. Always about $n^2/2$comparisons.
- Worst Case: $O(n^2)$. Reverse order also gives quadratic time.
- Space Complexity: $O(1)$because it sorts in place.

Comparison with Insertion Sort (my algorithm):

- Insertion Sort can be faster on almost sorted arrays ($\Omega(n)$), while Selection Sort is always quadratic.
- Selection Sort makes fewer swaps (max $n - 1$).
- In practice Insertion Sort is more efficient for real data.

3. Code Review

Good points:

- Code is clear and short.
- Metrics are included.
- Swap is done only when needed.

Problems:

- No checks for null or empty array.
- Test class does not use assertArrayEquals, only prints output.
- Only one test case is used, no edge cases.
- Array access counting could be more precise.

Suggestions:

- Add input validation.
- Add more tests (empty, duplicates, sorted, reversed).
- Save results to file instead of printing long arrays.
- Option to choose different algorithms in CLI.

4. Empirical Results

The benchmark runner tests sizes: 10, 100, 1000, 5000. Example results:

n=10 comparisons=45 swaps=5
n=100 comparisons=4950 swaps=97
n=1000 comparisons=499500 swaps=980

Findings:

- Comparisons grow as n^2, exactly as theory says.
- Swaps grow almost linearly.
- Time grows very fast for large arrays.
- Insertion Sort shows better results on nearly sorted inputs.

5. Conclusion

Selection Sort is easy to understand but slow. It always works in quadratic time and does not take advantage of partially sorted data. My partner's implementation is correct and clean, but tests should be improved.

Compared with my algorithm (Insertion Sort), Selection Sort does fewer swaps but is generally less efficient. This algorithm is good for study, but not for real large datasets.

Alikhan did well