Assignment 1 Design and Analysis of Algorithms
Kuanysh Beibarys SE-2413

1. MergeSort (improved Merge Sort)

The bottom line: divide the array into two parts, recursively sort each one and merge it.

Optimization: for small subarrays (< 10 elements), we use insertion sort instead of recursion — this speeds up the work. The buffer is created once and reused.

Difficulty: O(n log n) in time, O(n) in memory.

Recursion depth: ≈ log n.

2. QuickSort (improved Quick Sort)

The bottom line: we select a random pivot, divide the array into elements < pivot and > pivot, and sort recursively.

Optimization: we always recursively enter a smaller part of the array, and process the larger part with a loop — so the stack depth is ≈ O(log n).

Difficulty: average O(n log n), worst O(n2) (but almost never happens due to randomization).

Memory: O(log n) for recursion.

3. Deterministic Select (Median-of-Medians)
The bottom line: we find the kth element (for example, the median) in O(n) without a full sort.

How it works:
We divide the array into groups of 5, sort each one and take their medians.
Recursively we search for the median of these medians (it will be the pivot).
We divide the array by pivot and recurse only into the necessary part.

Difficulty: O(n) even in the worst case.

Recursion depth: limited, we always go to the smaller part

4. Closest Pair of Points

The bottom line: we are looking for the minimum distance between two points on the plane.

How it works:
Sorting the points by x.
We divide it in half and recursively calculate the minimum on the left and on the right.
We check the "strip" around the middle, looking only at the nearest neighbors on y.

Difficulty: O(n log n).

Memory: O(n) per auxiliary array.
.