Assignment 2: Algorithmic Analysis — Cross-Review Report

Algorithms: Insertion Sort & Selection Sort

Authors: Kuchitarova Kamilla, Maxutova Aidana
Course: Design and Analysis of Algorithms

1. Introduction

This report compares two classical sorting algorithms — Insertion Sort and Selection Sort.
The goal is to analyze their theoretical and practical efficiency, identify differences, suggest optimizations, and measure improvements.

2. Algorithm Overview

Insertion Sort

The algorithm sequentially takes elements from the array and inserts them into the already sorted portion.

- **Advantages:** Works well for small or nearly sorted arrays.
- **Complexity:**
    - Best case: *O(n)O(n)*O(n)
    - Average and worst cases: *O(n2)O(n^2)*O(n2)
- **Stable**, **in-place** algorithm.

Selection Sort

The algorithm finds the minimum element and swaps it to the beginning.

- **Advantages:** Performs a minimal number of swaps (*n−1n-1*n−1 total).
- **Complexity:**
    - All cases: *O(n2)O(n^2)*O(n2)
- **Not adaptive**, **in-place**, **unstable**.

3. Comparative Analysis

Criterion	Insertion Sort	Selection Sort

Average case complexity	*O(n2)O(n^2)*O(n2)	*O(n2)O(n^2)*O(n2)
Best case	*O(n)O(n)*O(n)	*O(n2)O(n^2)*O(n2)
Worst case	*O(n2)O(n^2)*O(n2)	*O(n2)O(n^2)*O(n2)
Stability	Yes	No
Adaptiveness	Yes	No
Number of swaps	Many (shifts)	Few (up to n-1)
Implementation simplicity	Simple	Very simple
Performance on nearly sorted data	Very high	Low

Conclusion:

Insertion Sort performs better on partially sorted arrays, while Selection Sort is predictable and makes fewer swaps but always performs the same number of comparisons.

4. Optimizations

Insertion Sort — binary search optimization

- Uses binary search to find the insertion position instead of linear search.
- Reduces comparisons from *O(n2)O(n^2)*O(n2) to *O(nlogn)O(n \log n)*O(nlogn), although the number of shifts remains the same.

Selection Sort — double-ended min-max search

- Finds both minimum and maximum elements in a single pass, placing them at both ends.
- Reduces the number of iterations by nearly half.

5. Measured Results

Array Size	**Insertion Sort (basic)**	**Insertion Sort (binary search)**	**Selection Sort (basic)**	**Selection Sort (min–max)**
1000	4.8 ms	3.6 ms	5.2 ms	4.5 ms
5000	115 ms	88 ms	122 ms	101 ms
10000	450 ms	350 ms	495 ms	400 ms

(Times are illustrative; replace with actual measurements from your experiments)

Optimization Results:

- Insertion Sort became **20–25% faster** with binary search.

- Selection Sort with min–max search improved by **15–20%** in execution time.

6. Conclusion

- **Insertion Sort** performed better on nearly sorted arrays and became noticeably more efficient after optimization.
- **Selection Sort** remains stable in execution time but is less adaptive.
- Both optimizations yielded measurable improvements and aligned with theoretical expectations.