

# Advanced Algorithms – Detailed Project Report

This report presents a comprehensive explanation of the Advanced Algorithms project completed as part of the Master's program. The report documents the problem definition, system design, algorithmic approach, implementation details, complexity analysis, results, and learning outcomes. It is intended to serve as a formal academic project report.

## 1. Introduction

Advanced Algorithms is a core subject that focuses on designing efficient solutions to computational problems using structured algorithmic techniques. This project applies theoretical algorithm concepts to a real-world inspired scenario: emergency response management.

## 2. Problem Statement

In real-world emergency response systems, incidents cannot be addressed strictly on a first-come-first-served basis. Critical factors such as severity, status, and response time must be considered to ensure effective prioritization. The challenge addressed in this project is to design a system that prioritizes emergency incidents using algorithmic decision-making.

## 3. Objectives

- Design a priority-based incident management system
- Apply sorting and searching algorithms
- Use appropriate data structures
- Analyze time and space complexity
- Demonstrate real-world application of algorithm theory

## 4. System Overview

The Emergency Response Priority Manager is a console-based Python application. It allows users to add, manage, search, and prioritize emergency incidents using core algorithmic techniques.

## 5. Data Model

Each emergency incident is represented as a structured record with the following fields:

1. Incident ID
2. Severity Level
3. Severity Score
4. Location Code
5. Response Time
6. Status
7. Description

## **6. Data Structures Used**

Python lists are used as dynamic arrays to store incident records. A queue is implemented using list operations to manage incoming emergency reports. These structures were chosen for simplicity and educational clarity.

## **7. Algorithms Used**

Insertion Sort:

Insertion sort is used to order incidents based on defined priority rules. It supports multi-key comparison and is suitable for small to medium datasets.

Linear Search:

Linear search is implemented to locate incidents by Incident ID.

Filtering Algorithm:

Linear filtering is used to retrieve incidents based on severity and status.

## **8. Priority Logic**

The priority of incidents is determined using the following hierarchy:

1. Status Priority: OPEN > DISPATCHED > RESOLVED
2. Severity Score: Higher scores indicate higher urgency
3. Response Time: Earlier response times receive higher priority

This multi-level comparison ensures that the most critical incidents are processed first.

## **9. Implementation Details**

The system supports multiple operations including adding incidents, processing queue entries, updating records, deleting incidents, sorting by priority, and searching records. The design emphasizes modular functions and clarity of logic.

## **10. Time and Space Complexity Analysis**

Insertion Sort:

- Worst-case time complexity:  $O(n^2)$
- Best-case time complexity:  $O(n)$
- Space complexity:  $O(1)$

Linear Search and Filtering:

- Time complexity:  $O(n)$

Queue Dequeue Operation:

- `pop(0)` has  $O(n)$  time complexity due to element shifting.

## **11. Results and Observations**

The system successfully prioritizes incidents based on defined rules. The output demonstrates correct ordering and reliable searching. The results validate the correctness of the implemented algorithms.

## **12. Limitations**

The use of a list-based queue introduces inefficiencies during dequeue operations. Insertion sort may not scale efficiently for very large datasets.

## **13. Future Enhancements**

- Use deque for optimized queue operations
- Implement more efficient sorting algorithms
- Add persistent storage or database integration
- Introduce automated testing

## **14. Learning Outcomes**

This project strengthened understanding of:

- Algorithm design and analysis
- Data structure selection
- Complexity evaluation
- Applying theory to practical scenarios

## **15. Conclusion**

The Advanced Algorithms project demonstrates the effective application of classical algorithmic techniques in a real-world inspired context. The emphasis on priority-based logic, complexity analysis, and structured documentation reflects a strong foundation in algorithmic problem-solving.