Technical Report: Final Project EECE 2560: Fundamentals of Engineering Algorithms

Xiangcheng Samuel Ying
Department of Electrical and Computer Engineering
Northeastern University
`Ying.xian@northeastern.edu`

November 18, 2024

Contents

1 Project Scope **2**

2 Project Plan **2**
2.1 Timeline 2
2.2 Milestones 2

3 Team Roles **3**

4 Methodology **3**
4.1 Pseudocode and Complexity Analysis 3
4.2 Data Collection and Preprocessing 3

5 Results **4**

6 Discussion **4**

7 Conclusion **4**

8 References **5**

A Appendix A: Code **5**

B Appendix B: Additional Figures **5**

1 Project Scope

This project aims to design a hospital emergency room (ER) patient management system that prioritizes patients based on their medical condition severity and time spent waiting. The system uses priority queues to ensure that patients in critical condition are treated first while also accounting for patients with extended wait times. Key objectives include:

- Developing a priority queue-based patient triage system.
- Balancing patient severity and wait time to ensure equitable care.
- Automating patient registration and prioritization for improved ER efficiency.

2 Project Plan

2.1 Timeline

The overall timeline for the project is divided into phases:

- **Week 1 (October 7 - October 13)**: Define project scope, establish team roles, and outline skills/tools.
- **Week 2 (October 14 - October 20)**: Begin development, set up the project repository, and start coding basic system functionalities such as patient registration and priority assignment.
- **Week 3 (October 21 - October 27)**: Continue development, implement the priority queue logic, and integrate the backend for patient data storage.
- **Week 4 (October 28 - November 3)**: Complete the backend, integrate the user interface, and begin testing the system with various patient scenarios.
- **Week 5 (November 4 - November 10)**: Finalize the system, conduct extensive testing, and refine the technical report.
- **Week 6 (November 11 - November 17)**: Revise the technical report and prepare the final presentation.
- **Week 7 (November 18 - November 28)**: Present the system, submit the final report, and close the project.

2.2 Milestones

Key milestones include:

- Project Scope and Plan Completion (October 7).

- GitHub Repository Setup and Initial Development (October 9).
- Backend Development and Priority Queue Implementation (October 28).
- System Testing and Report Draft (November 10).
- Final Presentation and Report Submission (November 28).

3 Team Roles

Team Member 1 (me): Responsible for frontend design and the user interface that allows hospital staff to register patients and view their priority status. Backend developer focusing on building the database and priority queue logic to manage patient data and wait times. Responsible for documentation and report writing, including maintaining GitHub and coordinating project updates. Quality assurance, focused on testing the system for edge cases, such as patients waiting for too long or severe cases arriving.

4 Methodology

4.1 Pseudocode and Complexity Analysis

This algorithm manages patients in an ER using a max-heap, prioritizing patients by severity and escalating priority for long wait times.
Pseudocode:

```
function InsertPatient(heap, patient):
    patient.priority = patient.severity + calculateWaitPriority(patient.checkInTime)
    add patient to heap
    heapifyUp(heap, lastIndex)
function TreatNextPatient(heap):
    patient = heap[0] // Get the patient with highest priority
    remove patient from heap
    heapifyDown(heap, 0)
    return patient
function UpdatePriorities(heap):
    for each patient in heap:
        patient.priority = patient.severity + calculateWaitPriority(patient.checkInTime)
    rebuildHeap(heap)
function calculateWaitPriority(checkInTime):
    elapsedTime = currentTime - checkInTime
    return scalingFactor * elapsedTime
```

4.2 Data Collection and Preprocessing

Data Collection

- Patient Data: Simulated input includes patient ID, name, severity (scale of 1 to 10), and check-in time (timestamp).
- Sources: Test cases were generated using synthetic datasets to simulate real-world ER scenarios.

Preprocessing Steps

1. Standardization: Ensure all severity scores and timestamps are in a consistent format.
2. Initial Priority Assignment: For each patient, calculate priority as severity + wait time weight
3. Heap Construction: Insert all preprocessed patient data into a max-heap for initial prioritization.

5 Results

Key Performance Metrics

- Average Insertion Time: O(log n), confirmed across datasets with varying n values.
- Average Dequeue Time: O(logn), consistent with theoretical analysis.
- Accuracy of Prioritization: Tested by ensuring patients with higher severity and/or longer wait times were treated first.

6 Discussion

The priority queue effectively prioritizes patients with higher severity scores and those with long wait times. The system maintained consistent performance under simulated peak conditions, ensuring high-severity cases were addressed promptly.

7 Conclusion

Key Findings
The system efficiently handled dynamic prioritization of patients using a max-heap, with time complexities suitable for real-time ER environments.
Severity-based prioritization and wait-time adjustment ensured fairness and responsiveness.
Limitations
The heap structure can become a bottleneck at extreme scales.
Dynamic priority adjustment requires fine-tuning to avoid edge-case bias.
Future Improvements

Implementing a distributed priority queue for large-scale environments.
Adding real-time analytics for staff decision support.
Integrating machine learning to refine severity scoring and predict patient needs.

8 References

Include all sources cited in the report.

A Appendix A: Code

Include relevant code sections here.

B Appendix B: Additional Figures

Provide any additional figures or tables that support the analysis.