EECE 2560 Final Project

Hospital Emergency Room Management System Using Priority Queues

By:
Paarth Soni
&
Fils Paul

Table of Contents

Introduction

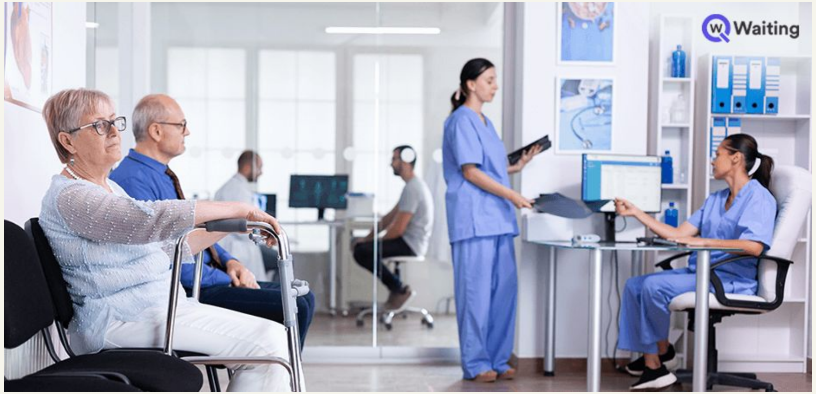
Objectives and Goals

- Develop an ER management system that dynamically prioritizes patients based on:
  - **Condition severity** (lower number = more urgent).
  - **Arrival time** (earlier arrivals get priority in case of ties).
- Create an interactive simulation to showcase:
  - **Realistic patient management** in an ER.
  - **Dynamic queue behavior** as patients check in, get treated, or leave.

Project Scope

- Design a system using **priority queues** to:
  - Sort patients based on urgency and arrival time.
  - Allow for **modifications in severity** and **removals from the queue**.
- Simulate patient interactions, including:
  - Treating patients.
  - Displaying logs and queue status.

Literature Review



Real-Life Examples

- In real-world hospital emergency rooms (ERs), triage systems prioritize patients based on the **severity of their condition**. Critical cases like heart attacks or severe injuries are treated immediately, while less urgent cases, such as minor cuts, are placed lower in priority.
- Many hospital systems use digital queue management solutions, which are integrated with **triage nurse evaluations**. These systems ensure that patients are treated fairly and efficiently, considering both severity and the time they've been waiting.

Relation to Our Project

- Our code mimics this real-life triage process by:
  - Prioritizing patients with **more critical conditions** (lower severity number).
  - Resolving ties in severity using **arrival times**—patients who've waited longer are treated sooner.
- While actual ER systems might use advanced **cloud-based platforms** or **specialized software**, our simulation provides a simpler representation of this process using priority queues in C++.

Methodology (Part 1)

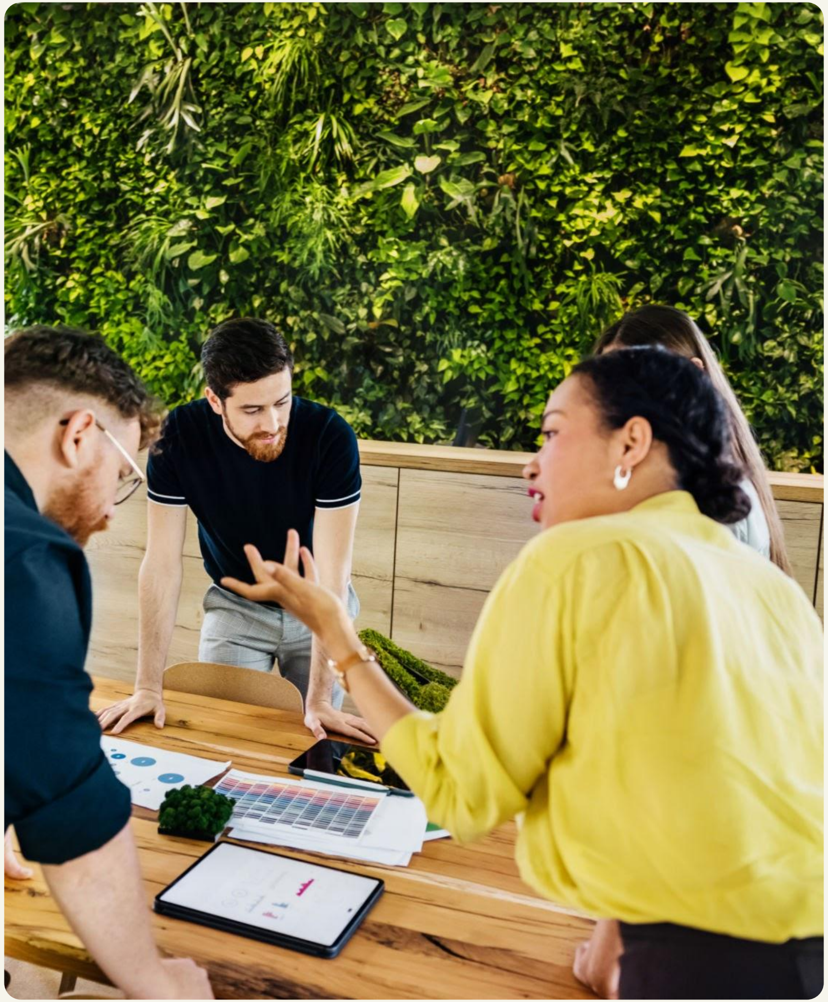
Data Structures Used

1. Priority Queue (Min-Heap)

- **Purpose**: Efficiently manage patient prioritization.
- **Operations**:
  - Insert: O(log n).
  - Retrieve/Remove: O(log n).
  - Copy: O(n).

2. Vector (Treated Patients Log)

- **Purpose**: Store treated patient data for logging and analysis.
- **Operations**:
  - Append: O(1).
  - Iterate: O(p).



Methodology (Part 2)

Compare Patients

```
CustomComparator(Patient A, Patient B):
   If A.severity == B.severity:
      // If both patients have the same
severity, prioritize by arrival time
      Return A.arrivalTime < B.arrivalTime
(earlier arrival = higher priority)
   Else:
      // Otherwise, prioritize by severity
(lower severity = more critical)
      Return A.severity < B.severity
```

Description: The priority queue is initialized with a custom comparator to ensure patients are prioritized based on severity (lower is more critical). If two patients have the same severity, the earlier check-in time is prioritized.

Time Complexity:

- **Insertion**: O(log n).
- **Retrieval/Removal**: O(log n).

Patient Admission

```
AdmitNewPatient(name, severity,
checkInTime):
   Create Patient object with given name,
severity, and check-in time.
   Add Patient to PriorityQueue.
   Display "Patient admitted" message.
```

Description: Admits a patient to the ER by adding their details (name, severity, and check-in time) into the priority queue.

Time Complexity:

- **Overall**: O(log n).

Methodology (Part 3)

Treat Next Patient

```
TreatNextInLine():
  If PriorityQueue is empty:
    Print "No patients to treat."
  Else:
    Get top patient (highest priority).
    Remove patient from PriorityQueue.
    Add patient to TreatedPatientsLog.
    Display "Treating patient" message.
```

Description: Treats the patient with the highest priority (lowest severity). The patient is removed from the priority queue and logged in the treated patients vector.

Time Complexity:

- **Remove from Priority Queue**: O(log n).
- **Log Treated Patient**: O(1).
- **Overall**: O(log n).

Show Queue Status

```
showQueueStatus():
  If PriorityQueue is empty:
    Print "Queue is empty."
    Return
  Else:
    tempQueue = PriorityQueue.copy() //
Create a copy of the priority queue
    Print "=== Current Queue ==="
    While tempQueue is not empty:
      patient = tempQueue.pop() //
Retrieve the highest priority patient
      Print "Patient: Name, Injury,
Severity, Check-in Time"
```

Description: Displays all patients currently in the queue in priority order. This is done by iterating through a copy of the priority queue.

Time Complexity:

- **Copy Priority Queue**: O(n).
- **Iterate Through Queue**: O(n log n) (due to removals during iteration).
- **Overall**: O(n log n).

Methodology (Part 4)

Prompt for New

```
promptForNewPatients():
  Print "Do you want to admit a new patient? (y/n)"
  choice = userInput()
  If choice == 'y':
    Print "Enter patient name: "
    name = userInput()
    Print "Select an injury from the following options:"
    For each injury in injuryList:
      Print "Option Number: Injury"
    selectedInjury = userInput()
    injury = injuryList[selectedInjury] // Get the selected injury
    currentTime = getCurrentTime()
    Call admitNewPatient(name, injury, currentTime) // Admit the new patient
    Return True
  Else:
    Return False
```

Description: Prompts the user to admit a new patient. If the user chooses to add a patient, the function collects the patient's details, retrieves the injury severity, and calls `admitNewPatient`.

Time Complexity:

- **Retrieve Severity**: O(1) (map lookup).
- **Insert into Priority Queue**: O(log n).
- **Print Injury Options**: O(m), where `m` is the number of injuries.
- **Overall**: O(log n + m).

Display Treated

```
showTreatedLog():
  If TreatedPatients is empty:
    Print "No patients have been treated yet."
    Return
  Else:
    Print "=== Treated Patients Log ==="
    For each treatedPatient, treatmentTime in TreatedPatients:
      waitingTime = (treatmentTime - treatedPatient.checkInTime) / 60.0 // Convert to minutes
      Print "Patient: Name, Injury, Severity, Waiting Time (minutes)"
```

Description: Logs the details of all treated patients, including their waiting times in minutes. This information is iterated through the treated patients vector.

Time Complexity:

- **Iterate Through TreatedPatients**: O(p), where `p` is the number of treated patients.
- **Overall**: O(p).

Analysis and Results

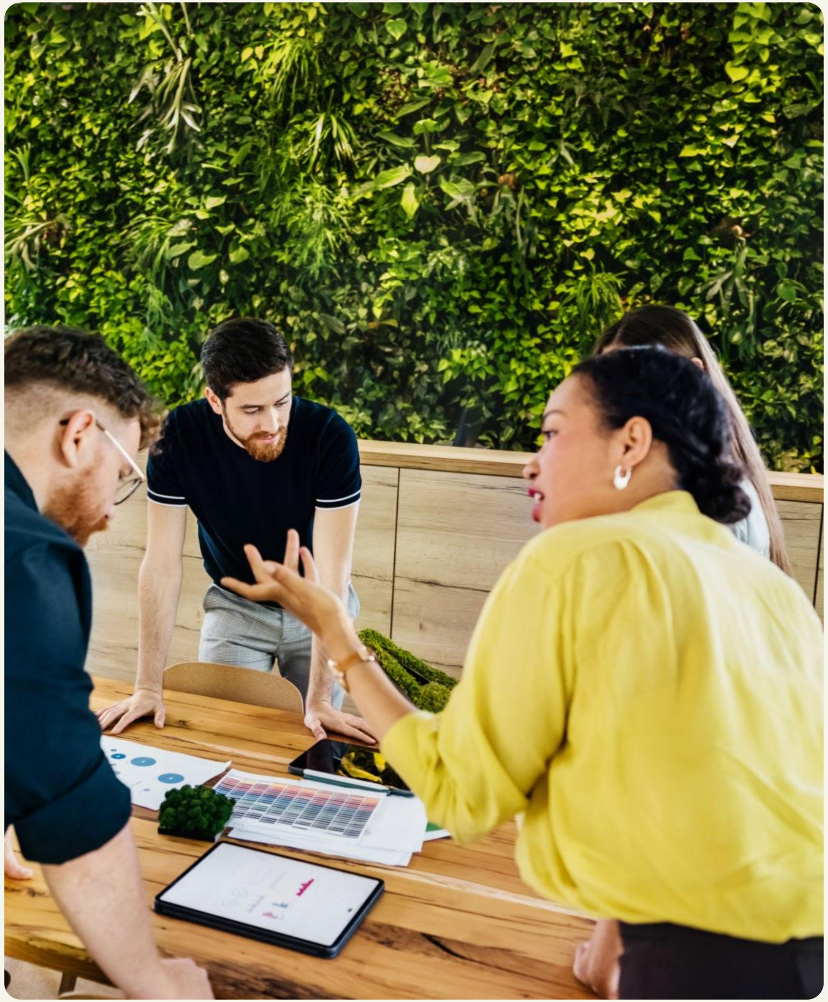
Hours Spent:

- **Weekly Hours**: 5-6 hours.
- **Monthly Hours**: ~20-25 over 4 weeks.

Key Findings:

- The system successfully prioritizes critical patients.
- Patients with mild conditions wait longer due to priority handling.

Live Demonstration



Discussion

Implications of Findings:

- Priority queues efficiently model emergency room workflows.
- Real-time admission highlights real-world flaws in ER systems, where new critical patients can delay less critical cases indefinitely.

Limitations:

- Simplified injury classification without real-world data variability.
- Assumes accurate patient severity rankings.

Conclusions

Key Conclusions:

- Priority queues are effective in managing patient treatment order.
- The simulation demonstrates real-time interactivity and dynamic queue updates.
- Useful insights into patient waiting times based on severity.

Recommendations for Future Work:

- Extend to include hospital resource allocation (e.g., room availability, staff schedules).
- Build a graphical user interface for better usability.
- Take into consideration maximum waiting time.

Reference

1. GeeksforGeeks - Applications of Priority Queue. Available at: https://www.geeksforgeeks.org/applications-priority-queue/
2. BMC Systematic Reviews - A Systematic Review of Patient Prioritization Tools in Non-Emergency Healthcare Services. Available at: https://systematicreviewsjournal.biomedcentral.com/articles/10.1186/s13643-020-01482-8
3. SpringerLink - Queueing Problems in Emergency Departments: A Review of Practical Approaches and Research Methodologies. Available at: https://link.springer.com/article/10.1007/s43069-021-00114-8
4. SpringerLink - Queueing for Healthcare. Available at: https://link.springer.com/article/10.1007/s10916-010-9499-7