SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
COLLEGE OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF NETWORKING AND COMMUNICATIONS
21CSC202J-Operating Systems , Mini-Project Presentation

Emergency Room Simulation with Priority Scheduling

Avani Baderiya (RA2211003011780)
Neha Maurya (RA2211003011787)
Marushka Singhania (RA2211003011810)

Table of Contents

Abstract

The "Emergency Room Simulation with Priority Scheduling" project offers an insightful simulation of a real-world emergency room scenario using Python programming. This simulation mimics the dynamic environment of an emergency room, where patients with diverse medical needs arrive for urgent care. The system employs priority scheduling techniques, assigning patients varying priority levels based on the criticality of their conditions. Through this simulation, the program showcases the significance of effective priority scheduling in emergency healthcare settings. The project provides a practical understanding of how emergency rooms optimize patient treatment based on their medical urgency, offering valuable insights into the crucial role of efficient resource allocation during critical situations.

Introduction

In the realm of healthcare, efficient management of emergency room scenarios is paramount. The ability to prioritize patient care based on the severity of their conditions is crucial for saving lives and optimizing medical resources. The "Emergency Room Simulation with Priority Scheduling" project delves into this vital aspect of healthcare operations, offering a comprehensive exploration of priority scheduling techniques in emergency situations. Through a Python-based simulation, the project illustrates how hospitals and medical facilities can handle diverse patient arrivals, assess their urgency, and allocate resources effectively using priority scheduling algorithms. This simulation not only demonstrates the technical implementation of priority scheduling but also emphasizes its real-world implications, shedding light on the pivotal role it plays in the healthcare industry. By delving into this simulation, one can gain valuable insights into the intricate balance between patient urgency, medical resources, and efficient treatment protocols within the dynamic environment of an emergency room.

What is Priority Scheduling

Priority scheduling is a scheduling algorithm used in computer systems, operating systems, and other contexts, where tasks or processes are assigned priorities and scheduled for execution based on these priorities. The basic idea behind priority scheduling is to assign a priority to each task, with higher priorities given to tasks that are more urgent or important. The scheduler always selects the task with the highest priority for execution.

In priority scheduling, tasks are often represented as processes, jobs, or threads, and each task is associated with a priority value. The priority value can be an integer, where a lower number represents a higher priority (i.e., priority 1 is higher than priority 5). Sometimes, the priority value can also be a real number or other forms depending on the specific system or application.

Task Priority: Each task is assigned a priority value, indicating its urgency or importance.

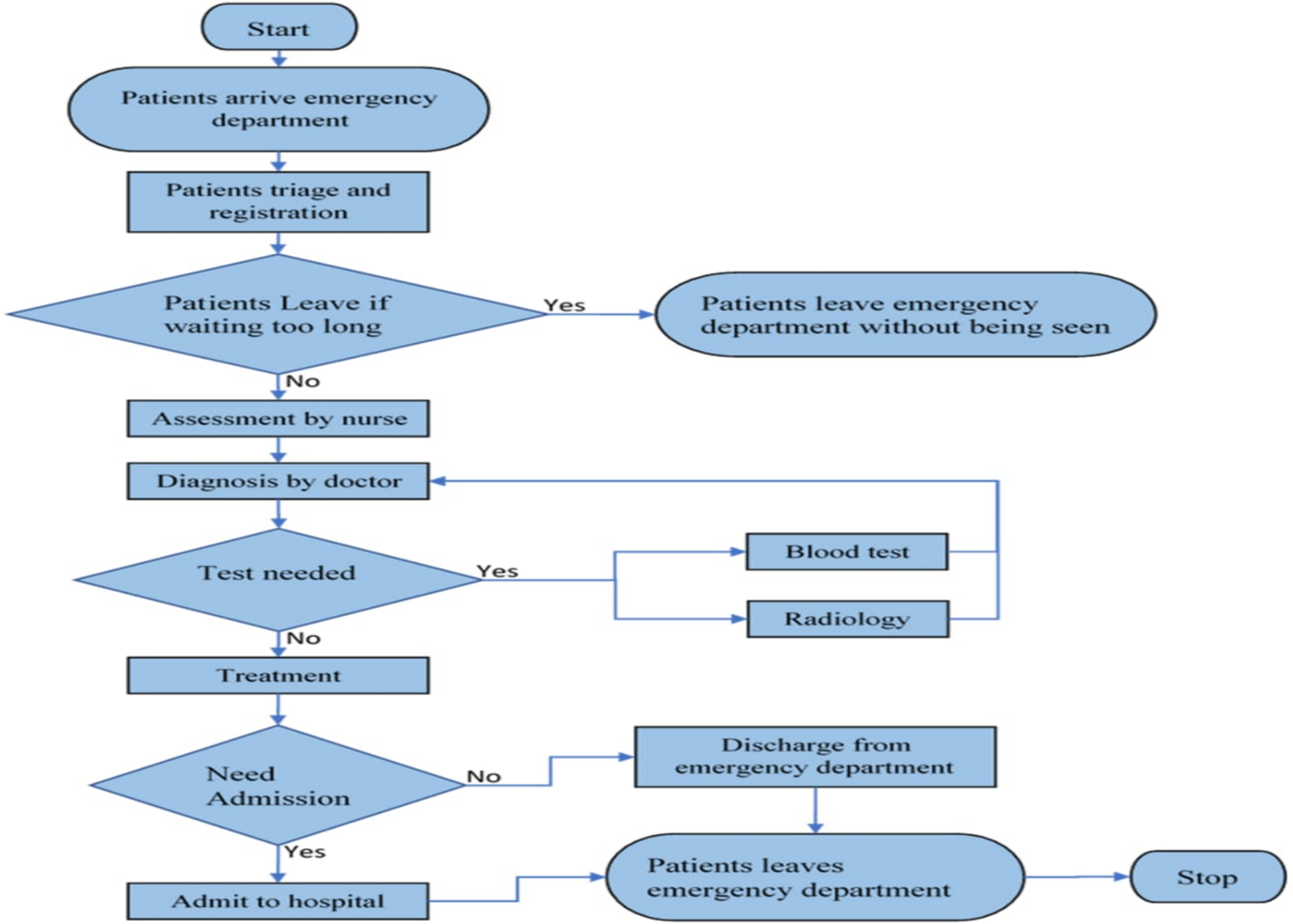
Scheduling Decision: The scheduler selects the task with the highest priority for execution. If there are multiple tasks with the same highest priority, various strategies can be used to decide which task to execute first, such as First Come First Serve (FCFS) within the same priority level.

Preemption: Priority scheduling can be either preemptive or non-preemptive. In preemptive priority scheduling, the currently executing task can be interrupted (preempted) if a higher-priority task becomes available. In non-preemptive priority scheduling, the current task continues to execute until it completes or enters a waiting state.

Starvation: One of the challenges with priority scheduling is the possibility of starvation, where lower priority tasks might never get a chance to execute if there are always higher-priority tasks in the system. To mitigate this, aging mechanisms can be implemented, where the priority of a task increases the longer it waits.

Dynamic Priority: In some systems, task priorities can change dynamically based on their behavior, execution history, or other factors. Dynamic priority adjustments can help in ensuring fairness and preventing starvation

FLOWCHART


Start
Patients arrive emergency department
Patients triage and registration
Patients Leave if waiting too long
Yes
Patients leave emergency department without being seen
No
Assessment by nurse
Diagnosis by doctor
Test needed
Yes
Blood test
Radiology
No
Treatment
Need Admission
No
Discharge from emergency department
Yes
Admit to hospital
Patients leaves emergency department
Stop

Implementation

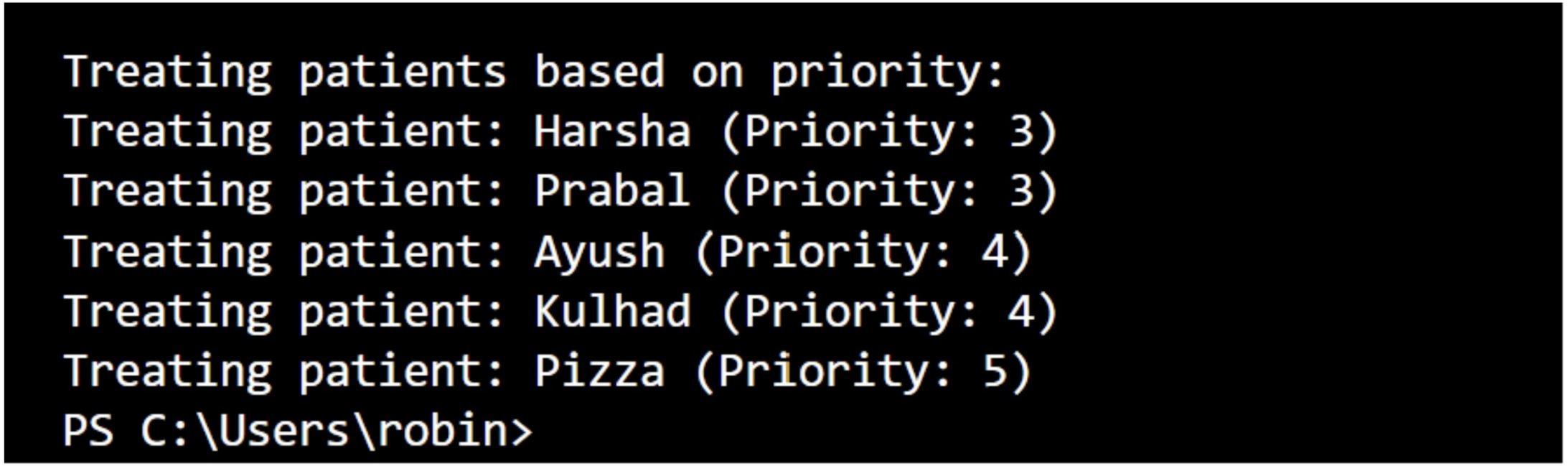
SRM INSTITUTE OF SCIENCE & TECHNOLOGY
Deemed to be University u/s 3 of UGC Act, 1956

Welcome | os.py

C: > Users > robin > os.py > ...

```python
import heapq
import random
import time

class EmergencyRoom:
    def __init__(self):
        self.patients_queue = []
        heapq.heapify(self.patients_queue)

    def add_patient(self, patient_name, priority):
        heapq.heappush(self.patients_queue, (priority, patient_name))

    def treat_patients(self):
        while self.patients_queue:
            priority, patient_name = heapq.heappop(self.patients_queue)
            print(f"Treating patient: {patient_name} (Priority: {priority})")
            time.sleep(1)  # Simulating treatment time

# Simulate patient arrivals and priority assignment
emergency_room = EmergencyRoom()

# Simulate patient arrivals and priority assignment
patients_data = [
    ("Harsha", random.randint(1, 5)),
    ("Ayush", random.randint(1, 5)),
    ("Prabal", random.randint(1, 5)),
    ("Kulhad", random.randint(1, 5)),
    ("Pizza", random.randint(1, 5)),
]

print("Patients arriving at the Emergency Room:")
for patient_name, priority in patients_data:
    print(f"Patient: {patient_name}, Priority: {priority}")
    emergency_room.add_patient(patient_name, priority)

print("\nTreating patients based on priority:")
emergency_room.treat_patients()
```

Results- Screen Shots of Output



Conclusion

In conclusion, priority scheduling is a fundamental concept in computer science and operating systems, allowing efficient allocation of resources based on task urgency and importance. This scheduling algorithm ensures that high-priority tasks are executed first, optimizing system performance and responsiveness.

In real-life scenarios, priority scheduling finds applications in various fields. From emergency rooms in healthcare, where critical patients are treated promptly, to computer systems, where tasks are prioritized for efficient execution, the concept of priority scheduling plays a vital role in enhancing operational efficiency and resource management.

Understanding and implementing priority scheduling algorithms are essential for creating responsive systems, especially in contexts where tasks have varying levels of urgency. By assigning and managing task priorities effectively, systems can deliver improved performance, meet deadlines, and provide timely responses to critical events, ultimately enhancing user satisfaction and system reliability.

SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
Deemed to be **University** *u/s 3 of UGC Act, 1956*

Thanks