Patient Management System

Nathaniel Pyo & Jimmy Cheung

Introduction

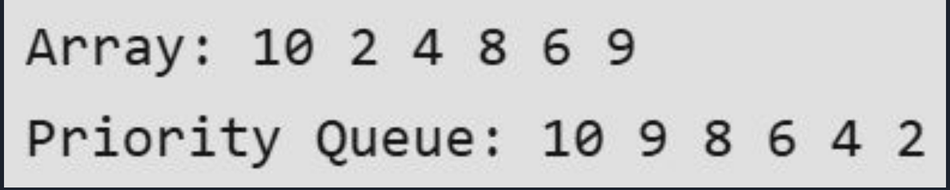
Problem Statement: Hospital patient management queues can be difficult to arrange when considering both severity and the time a patient has been waiting.

Project Scope
The aim is to construct an algorithm the receives patient information and organizes a priority queue structured around two principles: severity and wait time.

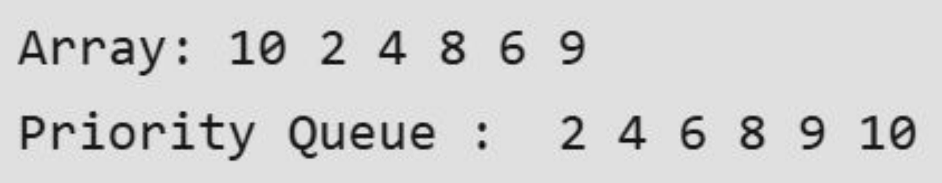
Objectives | Goals
Prioritize patient severity over arrival order
Prevent excessive delays and address time-sensitivity
Incorporate a timer to override past a max wait time threshold

Literature Review

Priority Queue



```
Array: 10 2 4 8 6 9
Priority Queue: 10 9 8 6 4 2
```

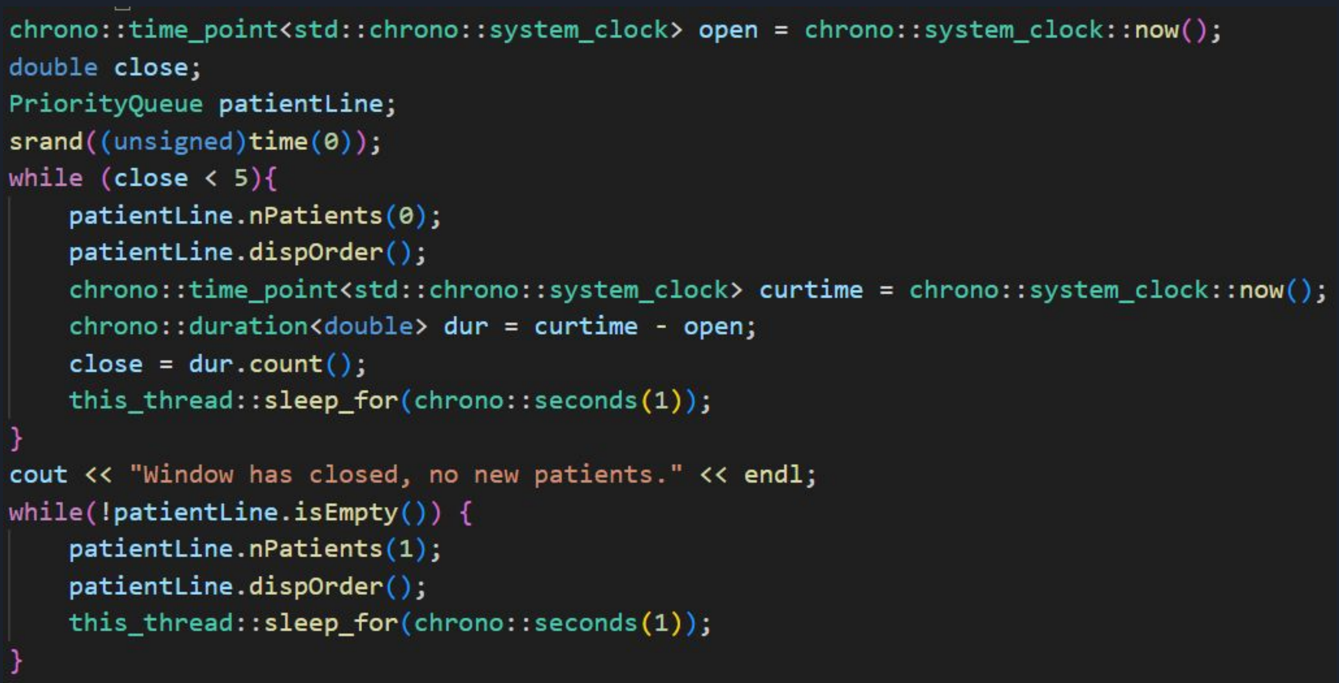


```
Array: 10 2 4 8 6 9
Priority Queue :  2 4 6 8 9 10
```

Methodology

Clerk Window

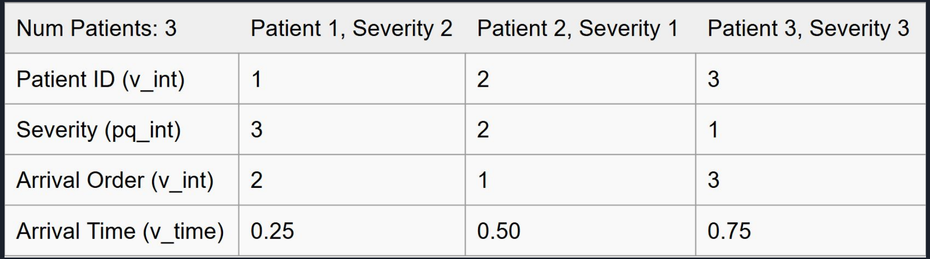
1. The window will be open for 5 seconds, check duration to verify
2. Add and serve patients while the window is open
3. Print current patient served and line status
4. Implement a slight delay between new patients added
5. Close window and serve remaining patients



```
chrono::time_point<std::chrono::system_clock> open = chrono::system_clock::now();
double close;
PriorityQueue patientLine;
srand((unsigned)time(0));
while (close < 5){
    patientLine.nPatients(0);
    patientLine.dispOrder();
    chrono::time_point<std::chrono::system_clock> curtime = chrono::system_clock::now();
    chrono::duration<double> dur = curtime - open;
    close = dur.count();
    this_thread::sleep_for(chrono::seconds(1));
}
cout << "Window has closed, no new patients." << endl;
while(!patientLine.isEmpty()) {
    patientLine.nPatients(1);
    patientLine.dispOrder();
    this_thread::sleep_for(chrono::seconds(1));
}
```

Add Patients

1. Generate random number 0-3 representing number of new patients
2. Generate random number 1-3 for each patient representing severity
3. Record patient ID, severity arrival order, and arrival time in vector
4. Record severity in priority queue



Num Patients: 3	Patient 1, Severity 2	Patient 2, Severity 1	Patient 3, Severity 3
Patient ID (v_int)	1	2	3
Severity (pq_int)	3	2	1
Arrival Order (v_int)	2	1	3
Arrival Time (v_time)	0.25	0.50	0.75

Serve Patients

1. Check if wait time threshold of 3 seconds has been reached
    a. Call function to calculate duration after being sent arrival time
2. If max wait time not reached, serve highest severity patient
3. If max wait time reached, serve patient for which max time reached
4. Match and remove all data attributes (ID, severity, order, time)
    a. Call auxiliary functions to match up data attribute indices

RECAP: welcome back everyone

Priority queue implementation to treat on severity basis unless wait time > 3

Wait time function checks if wait threshold (3 seconds) passed

Random number of patients generated (0-3) with random severity (1-3)

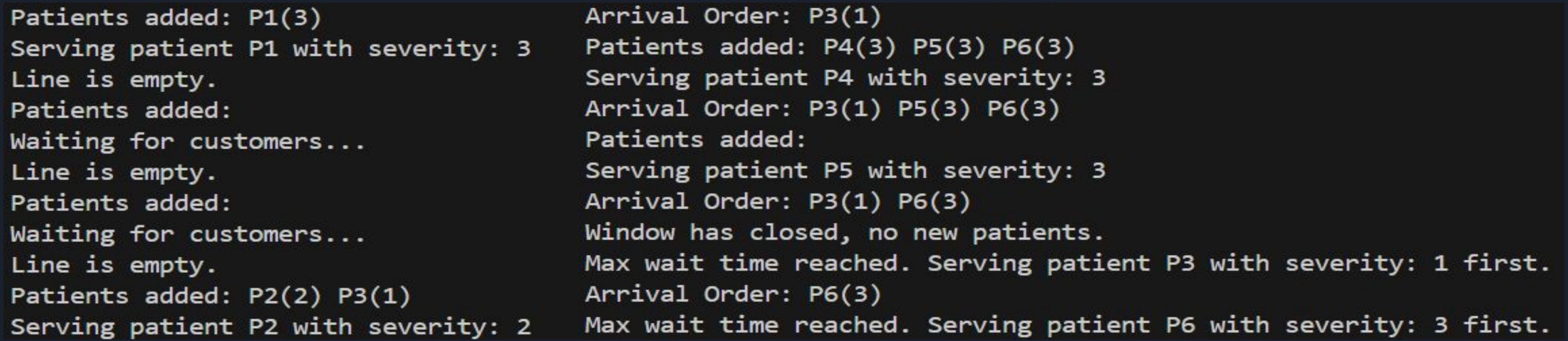
Window open for 5 seconds, no new patients after

Time Complexity

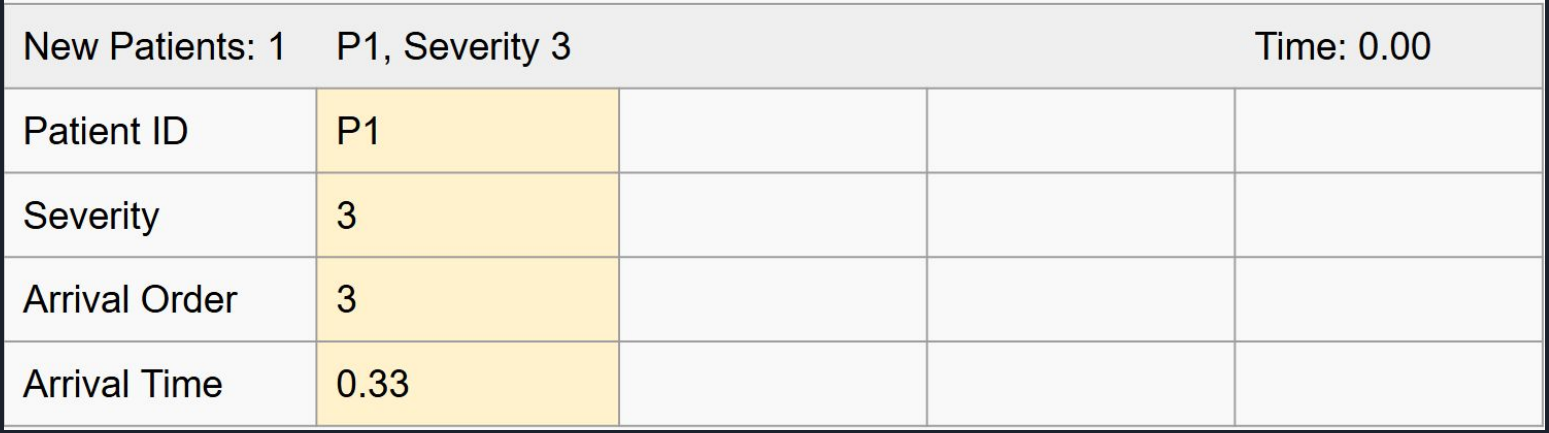
1. Window operations utilize a while loop: O(n)
2. Adding patients uses a combination of if statements and for loops
    a. priority_queue.push() is O(logn) complexity
    b. Push function contained within for loop: O(n*logn)
3. Serving patients uses a combination of if and for
    a. priority_queue.pop() is O(logn) complexity
    b. Pop function contained within for/while loop: O(n*logn)
4. For loops in aux functions for matching indices and checking wait time: O(n)

Total for queueing algorithm: **O(n*logn)**

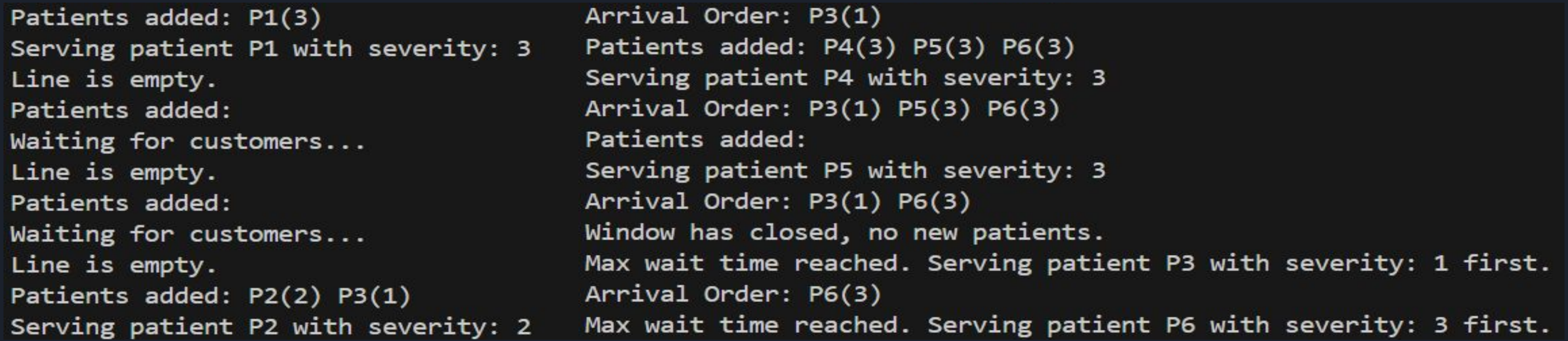
Analysis and Results



```
Patients added: P1(3)
Serving patient P1 with severity: 3
Line is empty.
Patients added:
Waiting for customers...
Line is empty.
Patients added:
Waiting for customers...
Line is empty.
Patients added: P2(2) P3(1)
Serving patient P2 with severity: 2
Arrival Order: P3(1)
Patients added: P4(3) P5(3) P6(3)
Serving patient P4 with severity: 3
Arrival Order: P3(1) P5(3) P6(3)
Patients added:
Serving patient P5 with severity: 3
Arrival Order: P3(1) P6(3)
Window has closed, no new patients.
Max wait time reached. Serving patient P3 with severity: 1 first.
Arrival Order: P6(3)
Max wait time reached. Serving patient P6 with severity: 3 first.
```

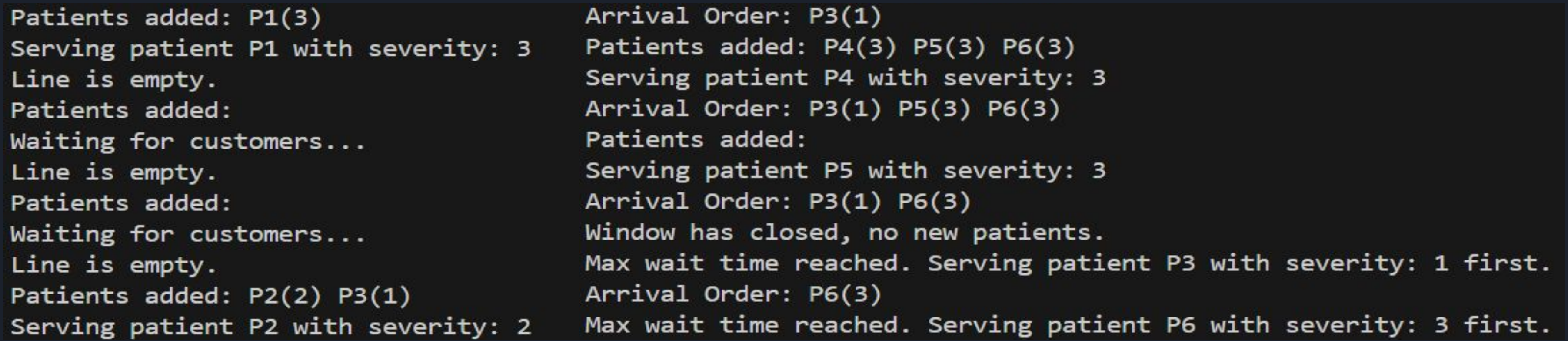


New Patients: 1	P1, Severity 3			Time: 0.00
Patient ID	P1			
Severity	3			
Arrival Order	3			
Arrival Time	0.33			



```
Patients added: P1(3)
Serving patient P1 with severity: 3
Line is empty.
Patients added:
Waiting for customers...
Line is empty.
Patients added:
Waiting for customers...
Line is empty.
Patients added: P2(2) P3(1)
Serving patient P2 with severity: 2
Arrival Order: P3(1)
Patients added: P4(3) P5(3) P6(3)
Serving patient P4 with severity: 3
Arrival Order: P3(1) P5(3) P6(3)
Patients added:
Serving patient P5 with severity: 3
Arrival Order: P3(1) P6(3)
Window has closed, no new patients.
Max wait time reached. Serving patient P3 with severity: 1 first.
Arrival Order: P6(3)
Max wait time reached. Serving patient P6 with severity: 3 first.
```

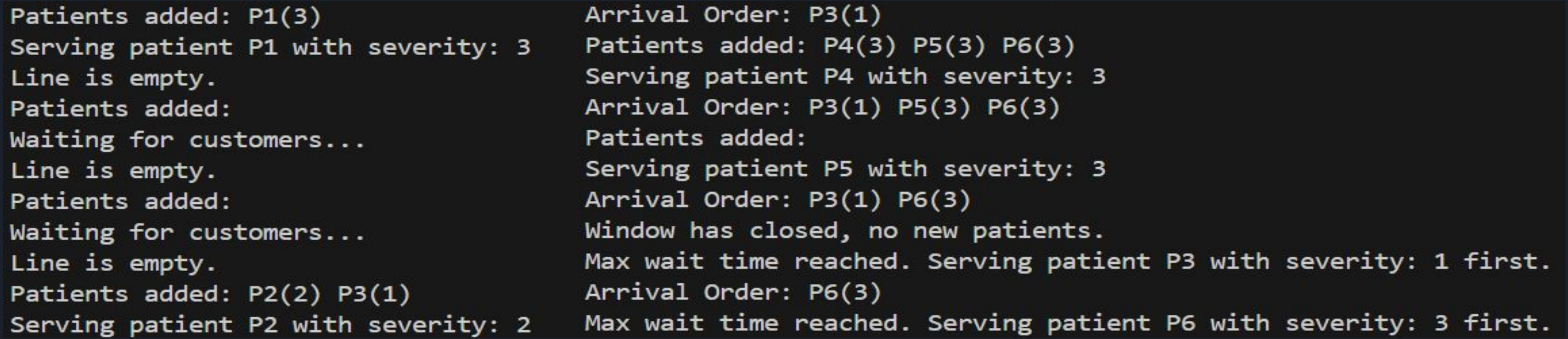
New Patients: 0				Time: 1.00
Patient ID				
Severity				
Arrival Order				
Arrival Time				



```
Patients added: P1(3)
Serving patient P1 with severity: 3
Line is empty.
Patients added:
Waiting for customers...
Line is empty.
Patients added:
Waiting for customers...
Line is empty.
Patients added: P2(2) P3(1)
Serving patient P2 with severity: 2
```

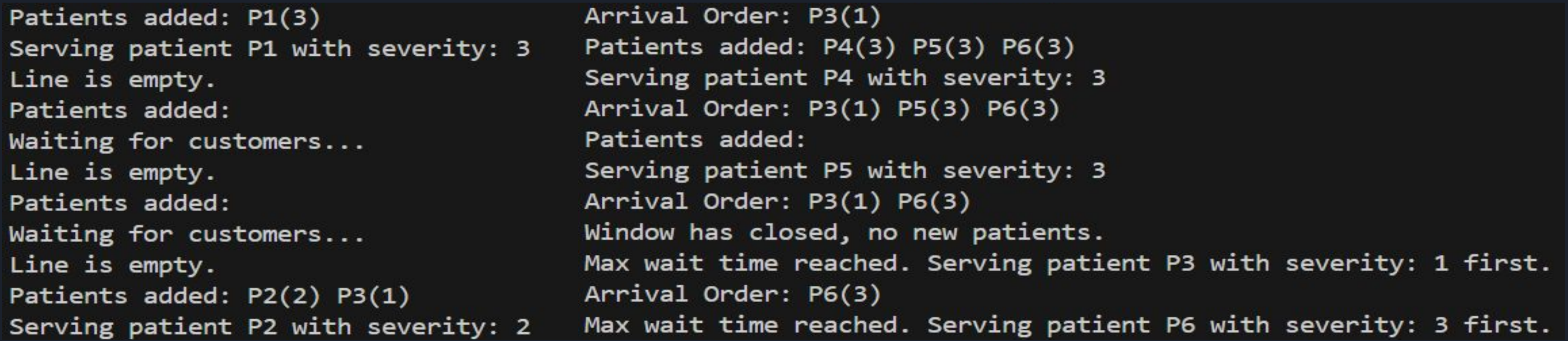
```
Arrival Order: P3(1)
Patients added: P4(3) P5(3) P6(3)
Serving patient P4 with severity: 3
Arrival Order: P3(1) P5(3) P6(3)
Patients added:
Serving patient P5 with severity: 3
Arrival Order: P3(1) P6(3)
Window has closed, no new patients.
Max wait time reached. Serving patient P3 with severity: 1 first.
Arrival Order: P6(3)
Max wait time reached. Serving patient P6 with severity: 3 first.
```

New Patients: 0				Time: 1.33
Patient ID				
Severity				
Arrival Order				
Arrival Time				

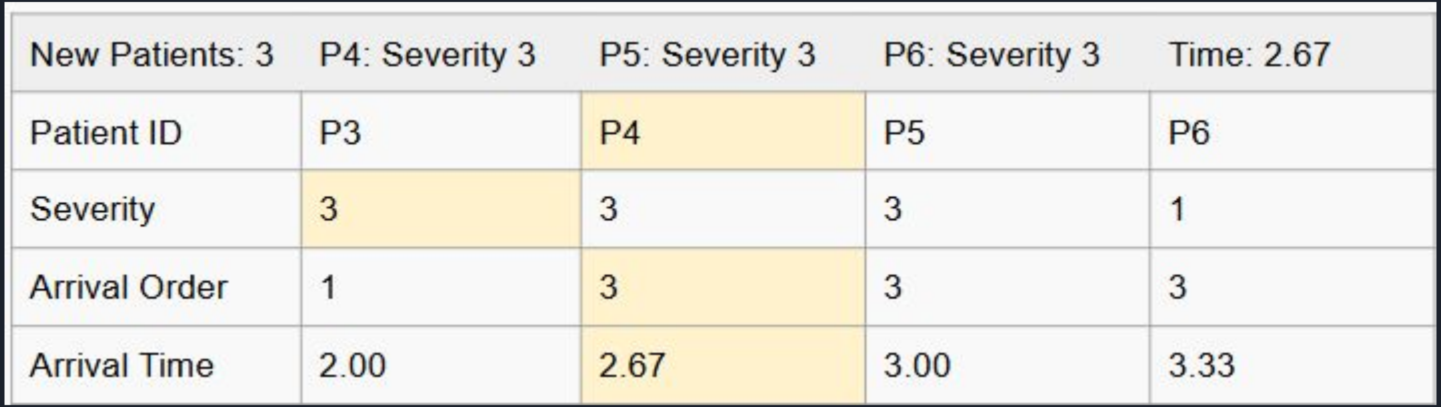


```
Patients added: P1(3)
Serving patient P1 with severity: 3
Line is empty.
Patients added:
Waiting for customers...
Line is empty.
Patients added:
Waiting for customers...
Line is empty.
Patients added: P2(2) P3(1)
Serving patient P2 with severity: 2
Arrival Order: P3(1)
Patients added: P4(3) P5(3) P6(3)
Serving patient P4 with severity: 3
Arrival Order: P3(1) P5(3) P6(3)
Patients added:
Serving patient P5 with severity: 3
Arrival Order: P3(1) P6(3)
Window has closed, no new patients.
Max wait time reached. Serving patient P3 with severity: 1 first.
Arrival Order: P6(3)
Max wait time reached. Serving patient P6 with severity: 3 first.
```

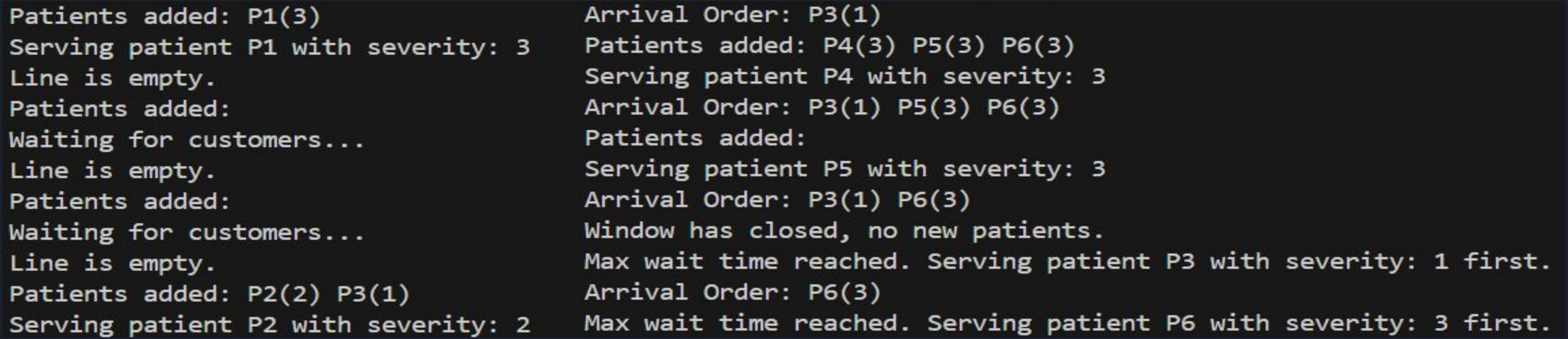
New Patients: 2	P2, Severity 2	P3, Severity 1		Time: 1.67
Patient ID	P2	P3		
Severity	2	1		
Arrival Order	2	1		
Arrival Time	1.67	2.00		



```
Patients added: P1(3)
Serving patient P1 with severity: 3
Line is empty.
Patients added:
Waiting for customers...
Line is empty.
Patients added:
Waiting for customers...
Line is empty.
Patients added: P2(2) P3(1)
Serving patient P2 with severity: 2
Arrival Order: P3(1)
Patients added: P4(3) P5(3) P6(3)
Serving patient P4 with severity: 3
Arrival Order: P3(1) P5(3) P6(3)
Patients added:
Serving patient P5 with severity: 3
Arrival Order: P3(1) P6(3)
Window has closed, no new patients.
Max wait time reached. Serving patient P3 with severity: 1 first.
Arrival Order: P6(3)
Max wait time reached. Serving patient P6 with severity: 3 first.
```

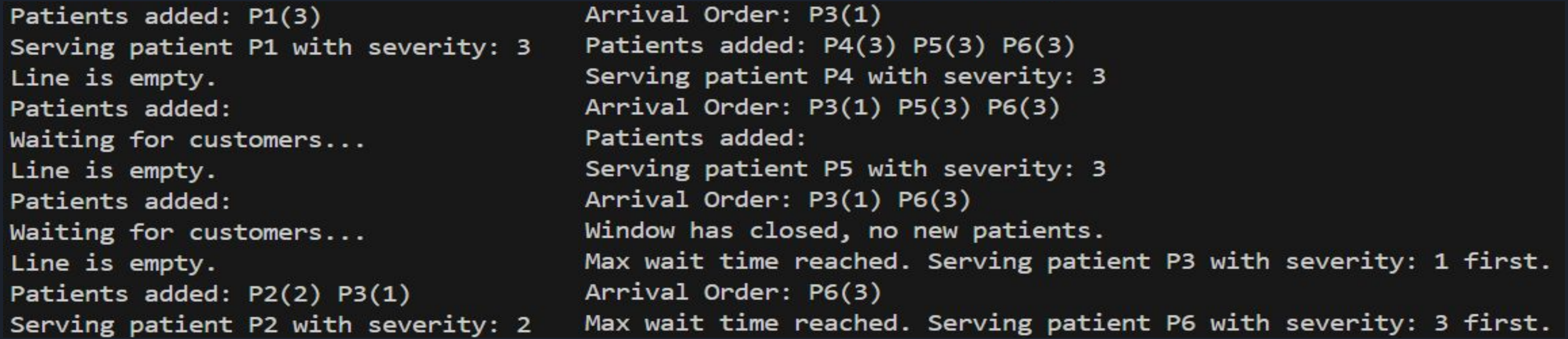


New Patients: 3	P4: Severity 3	P5: Severity 3	P6: Severity 3	Time: 2.67
Patient ID	P3	P4	P5	P6
Severity	3	3	3	1
Arrival Order	1	3	3	3
Arrival Time	2.00	2.67	3.00	3.33

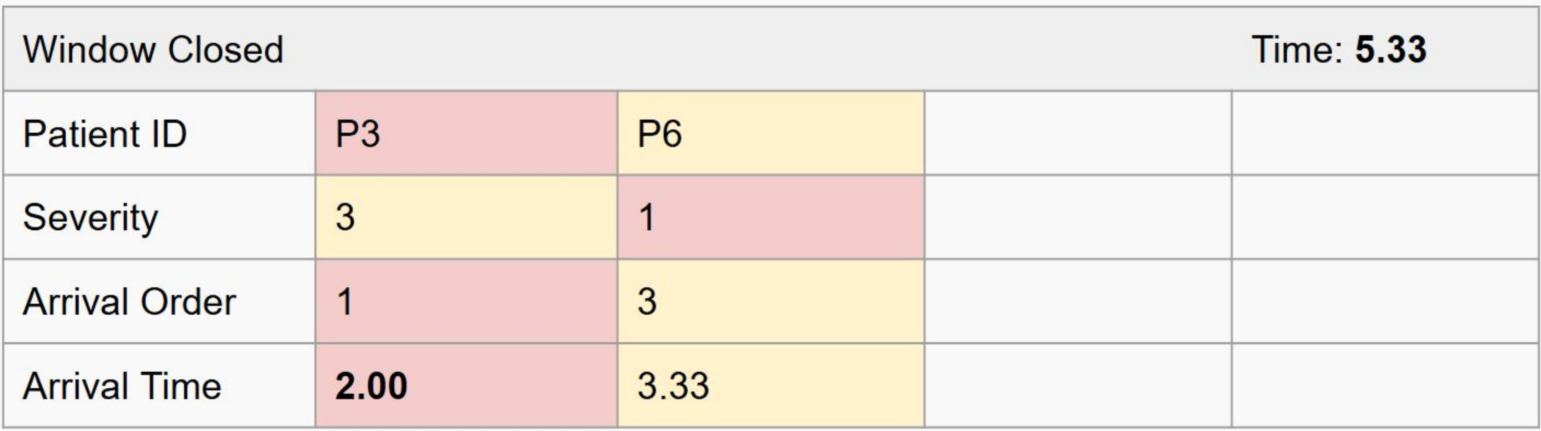


```
Patients added: P1(3)
Serving patient P1 with severity: 3
Line is empty.
Patients added:
Waiting for customers...
Line is empty.
Patients added:
Waiting for customers...
Line is empty.
Patients added: P2(2) P3(1)
Serving patient P2 with severity: 2
Arrival Order: P3(1)
Patients added: P4(3) P5(3) P6(3)
Serving patient P4 with severity: 3
Arrival Order: P3(1) P5(3) P6(3)
Patients added:
Serving patient P5 with severity: 3
Arrival Order: P3(1) P6(3)
Window has closed, no new patients.
Max wait time reached. Serving patient P3 with severity: 1 first.
Arrival Order: P6(3)
Max wait time reached. Serving patient P6 with severity: 3 first.
```

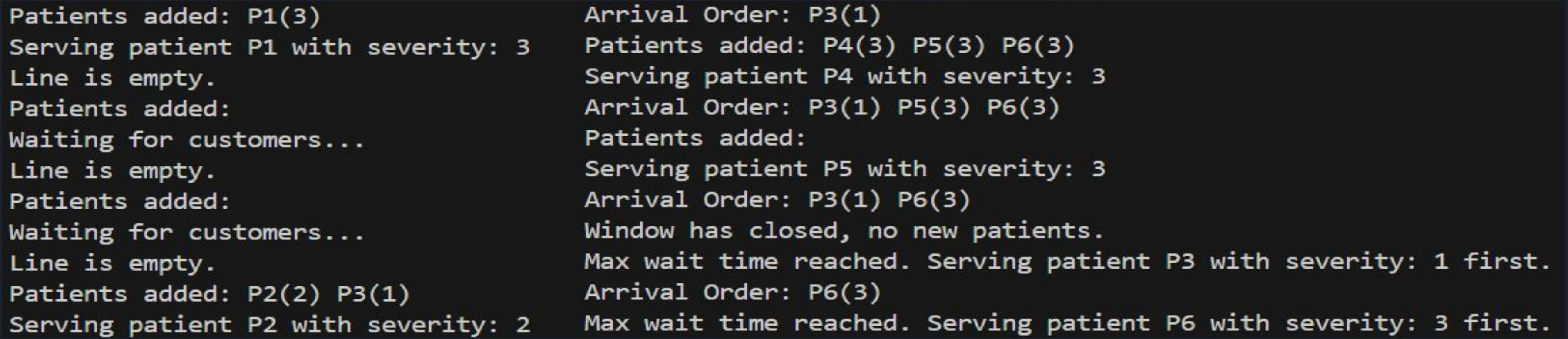
New Patients: 0				Time: 4.00
Patient ID	P3	P5	P6	
Severity	3	3	1	
Arrival Order	1	3	3	
Arrival Time	2.00	3.00	3.33	



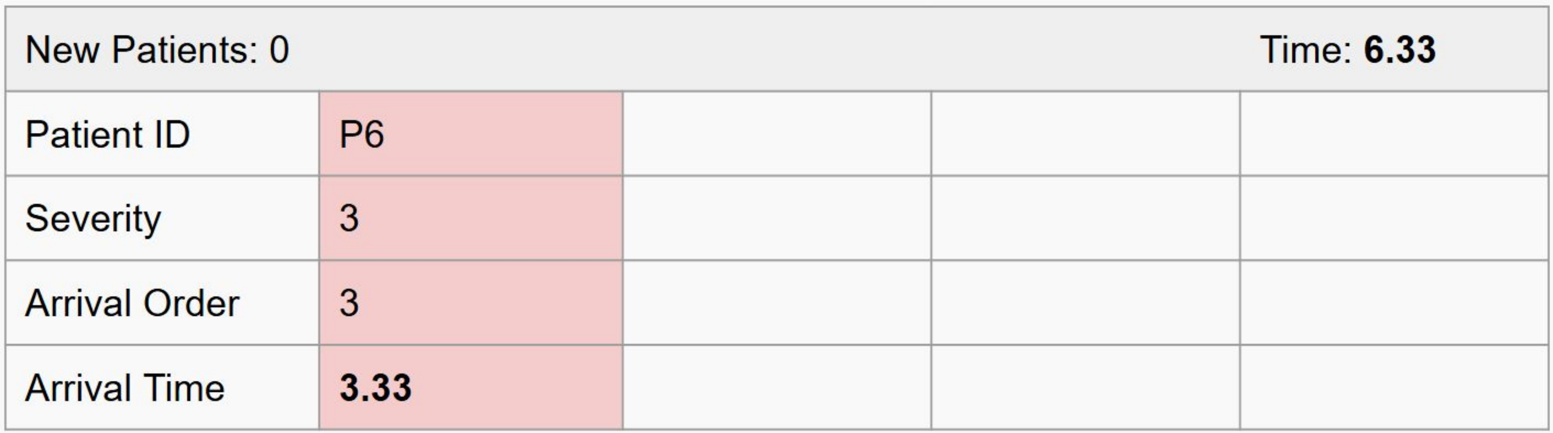
```
Patients added: P1(3)
Serving patient P1 with severity: 3
Line is empty.
Patients added:
Waiting for customers...
Line is empty.
Patients added:
Waiting for customers...
Line is empty.
Patients added: P2(2) P3(1)
Serving patient P2 with severity: 2
Arrival Order: P3(1)
Patients added: P4(3) P5(3) P6(3)
Serving patient P4 with severity: 3
Arrival Order: P3(1) P5(3) P6(3)
Patients added:
Serving patient P5 with severity: 3
Arrival Order: P3(1) P6(3)
Window has closed, no new patients.
Max wait time reached. Serving patient P3 with severity: 1 first.
Arrival Order: P6(3)
Max wait time reached. Serving patient P6 with severity: 3 first.
```



Window Closed				Time: **5.33**
Patient ID	P3	P6		
Severity	3	1		
Arrival Order	1	3		
Arrival Time	**2.00**	3.33		



```
Patients added: P1(3)
Serving patient P1 with severity: 3
Line is empty.
Patients added:
Waiting for customers...
Line is empty.
Patients added:
Waiting for customers...
Line is empty.
Patients added: P2(2) P3(1)
Serving patient P2 with severity: 2
Arrival Order: P3(1)
Patients added: P4(3) P5(3) P6(3)
Serving patient P4 with severity: 3
Arrival Order: P3(1) P5(3) P6(3)
Patients added:
Serving patient P5 with severity: 3
Arrival Order: P3(1) P6(3)
Window has closed, no new patients.
Max wait time reached. Serving patient P3 with severity: 1 first.
Arrival Order: P6(3)
Max wait time reached. Serving patient P6 with severity: 3 first.
```



New Patients: 0				Time: **6.33**
Patient ID	P6			
Severity	3			
Arrival Order	3			
Arrival Time	**3.33**			

Key Findings and Implications

- Order Preservation: Input order is only preserved in priority queues for identical elements
- Allows for the algorithm to sync data attributes by finding the first match
- Wait Time Check: Logically, the patient that is first in the arrival order vector will have been waiting the longest, time complexity is reduced by only checking wait time for that patient
- Hours Spent: 6 per week, 24 per month

Project Limitations and Conclusions

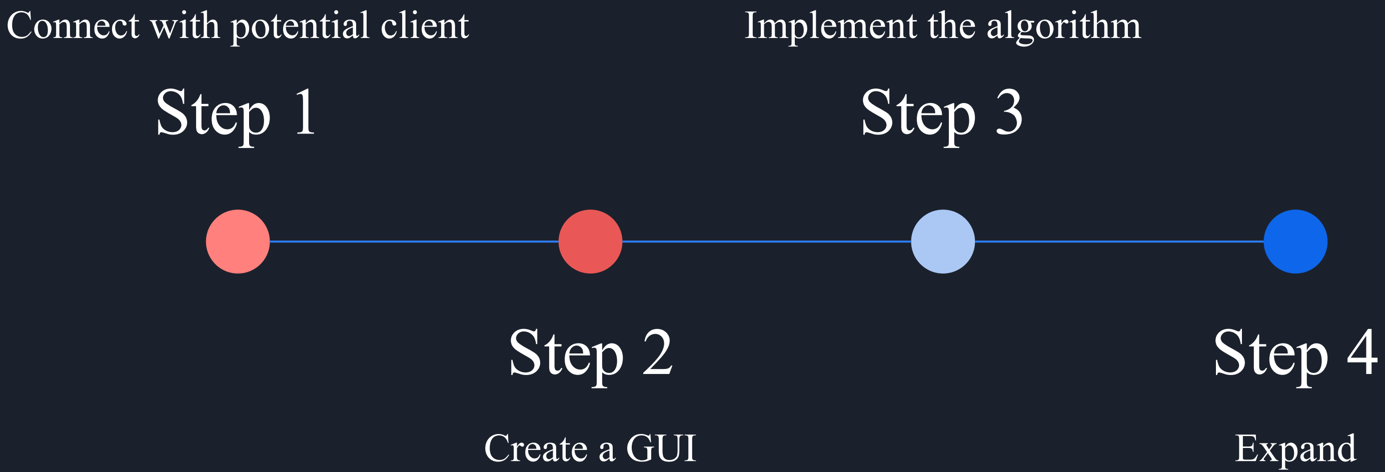
Space Limitations: The algorithm solution is optimized for time complexity but utilizes 4 different data attributes across all functions. These data attributes are dynamically modified.

Priority Queues: The automatic sort functionality is considerably useful but does not preserve order. For algorithms where order preservation is unnecessary, priority queues are incredibly efficient and compact for their auto-sort capabilities.

Project Limitations and Conclusions

Time Complexity: The time complexity of the algorithm was simplified due to the efficiency of priority queues. Some of the most efficient sorting algorithms are O(n*logn), and for the push and pop functions being O(logn), operating across all array elements is consistent at O(n*logn)

Future Work



Step 1

Connect with potential client

Step 2

Create a GUI

Step 3

Implement the algorithm

Step 4

Expand

References

GeeksforGeeks. (2024, October 11). Priority queue in C++ Standard Template Library (STL). GeeksforGeeks. https://www.geeksforgeeks.org/priority-queue-in-cpp-stl/