This is a discrete event simulation that uses an event trace driven as its input method.

Beware:

This program can’t run at load of more than 16,666 arrivals. I found that because I load every arrivals as like of a trace driven, it caps at 16, 666. I think it is because arraylist of objects takes a lot of memory space? And it doesn’t matter anyways as the stable state is below that.

My FCFS losses some data when running huge data cause of some weird bug that I can’t figure out.

Data structure used: ArrayList

Scheduling Algorithm used: FCFS and Linear Scan

Idling policy used: Stay and Bottom

I decided to run with 15,000 arrivals and departures, because it gave me the stable state of mean when running it multiple times with different parameters.

**First Come First Serve Analysis**

Assumptions

* Elevator is uninterruptible.
* When servicing an arrival, it dequeues all arrival request with the same floor and enqueues all departure request on that floor.
* After it enqueues request, it sorts the elevator request list and keeps the earliest request on top.
* Then it compares whether the earliest request on the elevator queue is earlier than the earliest request on the general request queue and the elevator base its decision from that.
* If general request queue is imminent than the request on elevator, then elevator goes to that general request floor and enqueues it.

**Test runs:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Scheduling | Lambda | Service | Idle Policy | Average Response time Up | Average Response Time Down |
| FCFS | 0.10 | 60 | Stay | 57.9647 | 67.4729 |
| FCFS | 0.10 | 60 | Bottom | 45.7985 | 81.1035 |

Under a low load, FCFS has almost the same response time compare to linear scan.

When the idle policy is changed from stay to bottom, the response time going up decreases as it saves travel time going to ground floor.

As expected, when idle policy is at bottom, the average response time goes up as it has more floors to travel compare to when the elevator just stays on the floor when idling.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Scheduling | Lambda | Service | Idle Policy | Average Response time Up | Average Response Time Down |
| FCFS | 0.50 | 60 | Stay | 109.3667 | 122.6851 |
| FCFS | 0.50 | 60 | Bottom | 138.0259 | 171.0670 |

When there is a heavy load, the average response time doubles up. This happens because more and more people that are in the elevator queue are going to get starved from its service as the elevator prioritize the earliest request (that maybe on the elevator request queue or in the general request queue).

If the idling policy is changed to bottom, both response time goes up. This might be the case when the elevator queue is empty, and it is assumed that elevator is uninterruptible, travels at the bottom but there might be departure request waiting during idle travel time.

Changing idle policy to bottom doesn’t help improve response time up as it always prioritizes the earliest request time.

For example:

When elevator goes at the bottom during idle, there might be an arrival waiting for an elevator but there is an earlier departure request. The elevator then ignores the arrival request and goes to that departure request’s floor and service it.

**Linear Scan Elevator Analysis**

Assumptions:

- It is assumed that the elevator is uninterruptible.

- That is when the direction of the elevator is going up it doesn’t stop for a down request.

- When it finishes, it goes to idle mode uninterrupted and then decides what next service is imminent.

**Test runs:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Scheduling | Lambda | Service | Idle Policy | Average Response time Up | Average Response Time Down |
| Linear | 0.10 | 60 | Stay | 56.9630 | 65.4363 |
| Linear | 0.10 | 60 | Bottom | 44.8473 | 77.7898 |

The response time for both goes down when idle policy is changed from stay to bottom if arrival rate is low.

It reduces the average response time going up because elevator is already at the bottom from when elevator is idle.

It increases the average response time going down because the position of the elevator is farther on the departure’s requests.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Scheduling | Lambda | Service | Idle Policy | Average Response time Up | Average Response Time Down |
| Linear | 0.50 | 60 | Stay | 78.5908 | 88.3251 |
| Linear | 0.50 | 60 | Bottom | 92.5308 | 121.1670 |

The response time for both goes up when idle policy is changed from stay to bottom if arrival rate is high. It is because the elevator is uninterrupted.

An example of when response time goes up:

- When elevator just finish its arrival service, it needs to go bottom floor uninterrupted. But then it decides which is the next most imminent event. It is possible that when the elevator reaches bottom, there are people that are waiting there but the elevator ignores it (assuming that the next imminent event is a departure) and goes up to service the most imminent event.

**Conclusion:**

Based on the results I got on the experiments, I would prefer to use a linear scan as the elevator’s scheduling algorithm because response time is reasonable under low or high load. Idle policy should be that the elevator stays on the floor of the last request before it goes idle because when using “bottom” idle policy only increases response times on heavy load.

There are graphs of results on the next pages and see the behavior of the elevator in each scenarios (i.e. changing scheduling policy, idle policy, and arrival rates).

**FCFS Graphs:**

**Idle Policy:** Bottom **Scheduling:** FCFS **Lambda rate:** 0.10

(Column A) Graph for FCFS Avg Response Time up of 45.7985

(Column C) Graph for FCFS Avg Response Time down of 81.1035

**Idle Policy:** Stay **Scheduling:** FCFS **Lambda rate:** 0.10

(Column A) Graph for FCFS Avg Response Time up of 57.9647

(Column C) Graph for FCFS Avg Response Time down of 67.4729

**Idle Policy:** Bottom **Scheduling:** FCFS **Lambda rate:** 0.50

(Column A) Graph for FCFS Avg Response Time up of 138.0259

(Column C) Graph for FCFS AVG Response Time down of 171.0670

**Idle Policy:** Stay **Scheduling:** FCFS **Lambda rate:** 0.50

(Column A) Graph for FCFS Avg Response Time up of 109.3667

(Column C) Graph for FCFS AVG Response Time down of 122.6851

**Linear Scan Graphs:**

**Idle Policy:** Stay  **Scheduling:** Linear  **Lambda rate:** 0.50

(Column A) Graph for Linear Avg Response Time up of 78.5908

(Column C) Graph for Linear Avg Response Time down of 88.3251

**Idle Policy:** Bottom **Scheduling:** Linear  **Lambda rate:** 0.50

(Column A) Graph for Linear Avg Response Time up of 92.5308

(Column C) Graph for Linear Avg Response Time down of 121.1670

**Idle Policy:** Stay  **Scheduling:** Linear  **Lambda rate:** 0.10

(Column A) Graph for Linear Avg Response Time up of 56.9630

(Column C) Graph for Linear Avg Response Time down of 65.4363

**Idle Policy:** Bottom **Scheduling:** Linear  **Lambda rate:** 0.10

(Column A) Graph for Linear Avg Response Time up of 44.8473

(Column C) Graph for Linear Avg Response Time down of 77.7898