Moshe Shtaygrud
CSCI 340

Project 1
Anne Thompson

Documentation:

When running the code, directly run the file "*jobScheduler.java*". The code will output the initial state of the jobs followed by all the algorithms and their respective Gantt Diagrams and results (ATT, Throughput). Please be sure **not** to wrap the lines in the output final so that the Gantt diagrams can be displayed correctly in the output panel. Please make sure the Gantt Diagrams can be displayed in one line for the output to look normal. I will include a sample "output.txt" file where I documented the results from one of the simulations. All the scheduling algorithms are in the jobScheduler class.

FIFO Algorithm:

Data Structures:
This implementation utilizes an ArrayList to manage the completed jobs. The decision to use an ArrayList stems from the sorting of the jobs list by arrival times. Consequently, a Queue data structure isn't necessary because accessing the next element in the sorted list inherently corresponds to the next job in line for execution.
Runtime:
The runtime complexity of the FIFO algorithm in this implementation is O(nlogn), where 'n' represents the number of jobs in the list. This complexity arises from looping over all the jobs and completing them sequentially, one after the other, but with the overhead of sorting the jobs based on arrival times. As a result, the algorithm's performance scales logarithmically with the number of jobs.

SJF Algorithm:

Data Structures:

- PriorityQueue: Utilized to maintain a queue of jobs prioritized by their CPU burst times. The priority queue is implemented with a comparator that sorts jobs first by their CPU burst times and then by their arrival times.
- ArrayList: Used to store completed jobs. As jobs are processed and completed, they are added to this list.

Runtime:
The runtime complexity of the SJF algorithm in this implementation varies based on the number of jobs and their characteristics. However, in the worst case, where all jobs have different burst times, the runtime complexity is O(n^2), where 'n' represents the number of jobs. This complexity arises from the nested loop structure, where for each job, all jobs are checked to find the one with the shortest burst time.

SRT Algorithm:

Data Structures:

- PriorityQueue: Utilized to maintain a queue of jobs prioritized by their remaining burst times. The priority queue is implemented with a comparator that sorts jobs first by their remaining burst times and then by their arrival times.
- ArrayList: Used to store completed jobs. As jobs are processed and completed, they are added to this list.

Runtime:
In the worst-case scenario, where all jobs have different remaining burst times, the runtime complexity is O(n^2), where 'n' represents the number of jobs. This complexity arises from the nested loop structure, where for each job, all jobs are checked to find the one with the shortest remaining burst time

Highest Priority Algorithm:

Data Structures:

- PriorityQueue: Utilized to maintain a queue of jobs prioritized by their priority levels. The priority queue is implemented with a comparator that sorts jobs first by their ASSIGNED priority in descending order and then by their arrival times.
- ArrayList: Used to store completed jobs. As jobs are processed and completed, they are added to this list.

The runtime complexity is O(n^2), where 'n' represents the number of jobs. This complexity arises from the nested loop structure, where for each job, all jobs are checked to find the one with the highest priority. Despite this, the Highest Priority algorithm can provide efficient execution for jobs with varying priorities.

Round Robin Algorithm:

Data Structures:

- Queue: Utilized to maintain a circular queue of jobs waiting for execution. Implemented using a LinkedList, this queue manages the order in which jobs are processed.
- ArrayList: Used to store completed jobs. As jobs are processed and completed, they are added to this list.

The runtime complexity of the Round Robin algorithm in this implementation depends on the number of jobs, their burst times, and the specified quantum. In the worst-case scenario, where jobs have varying burst times, the runtime complexity can be O(n^2), where 'n' represents the number of jobs. This complexity arises from the nested loop structure and the time quantum, which can result in multiple iterations over the list of jobs. However, Round Robin is known for its simplicity and fair allocation of CPU time.

Moshe Shtaygrud
CSCI 340

Project 1
Anne Thompson

Summary of results:

FIFO:

Average Turn Around Time: 22.6
The throughput at time 100 is: 8
Observation:
The FIFO algorithm executes jobs based on their arrival sequence, processing the earliest arriving jobs first. However, this approach may result in delayed turnaround times for jobs arriving later, particularly if longer CPU burst jobs are prioritized at the beginning, leading to what is known as the "convoy effect."

SJF:

Average Turn Around Time: 20.04
The throughput at time 100 is: 9
Observation:
The SJF algorithm favors jobs with shorter CPU bursts, typically resulting in reduced average turnaround times. Consequently, it tends to achieve a greater throughput within the specified time frame compared to FIFO, where jobs are processed in the order of their arrival without considering their burst durations.

SRT:

Average Turn Around Time: 19.28
The throughput at time 100 is: 9
Observation:
As a preemptive scheduling method, SRT enables job interruption based on their remaining time, contributing to the minimization of turnaround times. This feature facilitates the swift completion of shorter jobs, ultimately enhancing overall throughput.

Highest Priority:

Average Turn Around Time: 25.8
The throughput at time 100 is: 8
Observation:
Priority scheduling arranges jobs for execution based on their priority levels, potentially causing interruptions and job neglect if the priorities are unevenly distributed. Consequently, this imbalance can lead to prolonged turnaround times compared to alternative scheduling algorithms.

Round Robin (Q=2) :

Without context switch:

Average Turn Around Time: 29.64
The throughput at time 100 is: 8

With context switch = 1

Average Turn Around Time: 124.48
The throughput at time 100 is: 1

Moshe Shtaygrud
CSCI 340

Project 1
Anne Thompson

Observation:
The round-robin algorithm employs a set time quantum to manage job execution. While this method ensures fair processing of jobs, it can contribute to extended turnaround times because of frequent context switches. Additionally, it may reduce throughput as jobs requiring more time to execute within the allocated quantum may face delays.

Final Words:
SJF and SRT algorithms excel in minimizing average turnaround times by prioritizing shorter jobs or those with shorter remaining times.
FIFO and Priority scheduling algorithms exhibit moderate performance, with Priority scheduling possibly leading to job starvation due to its rigid priority rules.
Round Robin algorithm tends to have longer average turnaround times and lower throughput as a consequence of frequent context switching. Nonetheless, it maintains fairness among jobs owing to its fixed time quantum.

GITHUB LINK:
https://github.com/roomfulmoshe/CS340_Project1.git