Implementation of Six CPU Scheduling Algorithms

Balite, Marie Edcel D.1, de Jesus, Nikka Marie C.2

***Abstract*— In this paper, the different CPU scheduling algorithms will be discussed one by one, followed by sample test cases used accordingly. Each test case will be acquired randomly.**

***Keywords***—  **First-Come-First-Serve scheduling; Shortest Job First scheduling; Shortest Remaining Time First; Non-preemptive Priority scheduling; Preemptive Priority scheduling; Round Robin scheduling**

1. Introduction

Scheduling is the basis of multiprogramming. The objective of scheduling is to make the computer productive by switching the CPU among processes and to have some process executing at all times to maximize CPU utilization. Whenever the CPU becomes idle, the CPU scheduler selects one of the processes ready to be executed in the ready queue and allocates the CPU to that process. All the records in the ready queue, which are generally process control blocks (PCBs) of the processes are conceptually lined up waiting for a chance to execute in the CPU. However, note that ready queues are not necessarily a first-in-first-out (FIFO) queue. Given this, the severity of starvation of the processes residing in the ready queue varies depending on which CPU scheduling algorithm is used. In this paper, six CPU scheduling algorithms namely: first-come, first-served (FCFS), shortest-job-first (SJF), shortest-remaining-time-first (SRTF), nonpreemptive and preemptive priority scheduling, and round robin scheduling. and their programming implementations are presented and briefly defined. Three test cases with increasing sizes of 10, 15, and 20 which consists of randomly generated processes are used to simulate how each of the scheduling algorithms work. The results and observations on each simulation are discussed and used to test the correctness of the programming implementations for each of the six CPU scheduling algorithms.

1. Programming Implementation

The task is to correctly implement each of the six scheduling algorithms. The specifications require a Process Control Block (PCB) which holds the values of a process, such as process id, arrival time, CPU burst time, priority and an execution history information (i.e. querying for an execution history information of a process such as the number of times it has been pre-empted must be possible). It is also important that the machine problem properly implements all six classical scheduling algorithms.

The programming implementation of this machine problem is written in Java where one class for the process control block, one for the scheduler, and six classes for each classical scheduling algorithms were created. An instance of the process control block class holds values of a process and all functions necessary to satisfy the specifications mentioned. The scheduler class is where the queues are generated and where the incoming processes are inserted into the ready queue. Inside this class is also a thread that increments variable every 100 milliseconds. This thread serves as a clock that is independent from the system. On each of the six classes for the scheduling algorithms is another thread for the execution of the processes.

1. Algorithms and Test Cases

Scheduling algorithms can be preemptive and nonpreemptive.

*Note: The two scheduling categories do not necessarily mean they are the same with Preemptive Priority Scheduling Algorithm and Non-preemptive Priority Scheduling Algorithm.*

In non-preemptive scheduling, once the CPU has been allocated to a process, the CPU is kept by the process until it is released either by terminating or by switching to a waiting state.

In preemptive scheduling on the other hand, the processes are usually assigned with priorities. When a process with a higher priority arrives during the execution of another process, the execution will be interrupted for some time and will be resumed later when the priority task has finished its execution.

Here are the test cases for each of the six classical scheduling algorithms.

1. *First-Come- First- Serve (FCFS) Scheduling*

As the name suggests, in this scheduling algorithm, the process which arrives first, gets the CPU allocated first.

1. *Shortest Job First (SJF) Scheduling*

Basically, the idea behind this algorithm is that the process which has the smallest amount of burst time will be executed first. SJF is of two types: Non-preemptive and Preemptive, although SJF is usually referred to as the non-preemptive type, while the preemptive one is commonly known as the SRTF or Shortest Remaining Time First.

1. *Shortest Remaining Time First (SRTF) Scheduling*

SRTF scheduling is basically the preemptive type of SJF. This means that when a process with a shorter or smaller amount of burst time arrives during the execution of a longer process, the execution will be interrupted for some time and will be resumed later when the priority task has finished its execution.

1. *Non-preemptive Priority Scheduling*

In non-preemptive priority algorithm, processes are executed based on their assigned priorities, but with respect of course to their arrival time.

1. *Preemptive Priority Scheduling*

As stated above, processes with higher priority are executed first. If it happens that a process is being executed then another process with a higher priority arrives, its execution will be interrupted, and will then be resumed after the task has finished its execution.

1. *Round Robin (RR) Scheduling*

Round robin scheduling is the preemptive type of First-Come First-Served scheduling. The processes are dispatched in a first-in first-out manner, but each of them is only allowed to execute for only a limited of time. The limits assigned to each process is called the time quantum.

If the process finishes before the time quantum expires, it is swapped out of the CPU just like the normal FCFS algorithm.

If the time quantum expires first, the process is swapped out and moved to the back end of the queue.

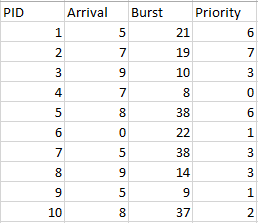
The queue is maintained in a circular manner so that when all processes have had a turn, the first process is given another turn, and so on.

1. Results and Discussion
2. Conclusion

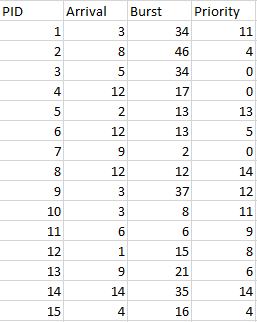
// Amo na ini at test cases?

1. Scheduling Algorithms

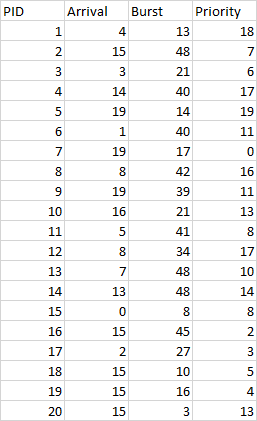
Sample test case #1:



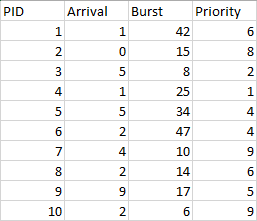
Sample test case#2:



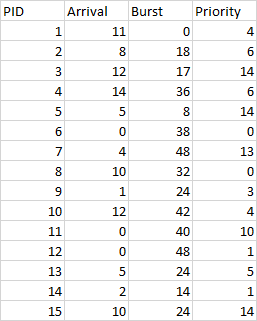
Sample test case #3:



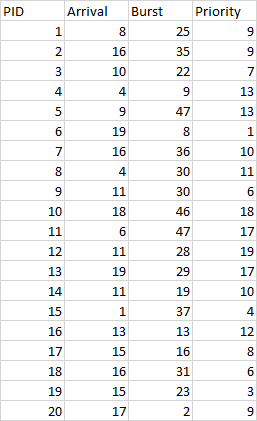
Sample test case #1:



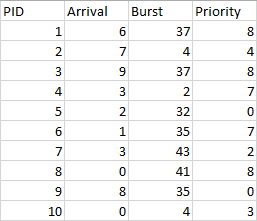
Sample test case #2:



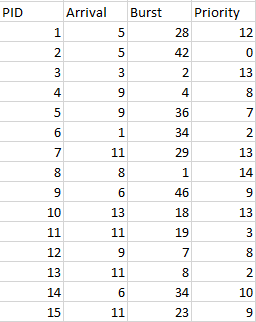
Sample test case #3:



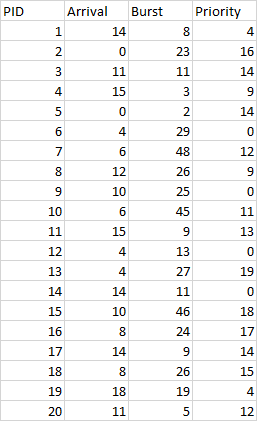
Sample test case #1:



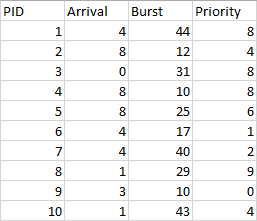
Sample test case #2:



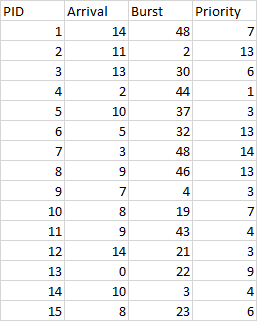
Sample test case #3:



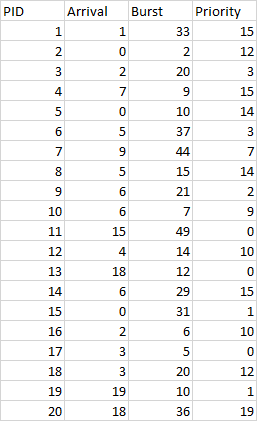
Sample test case #1:



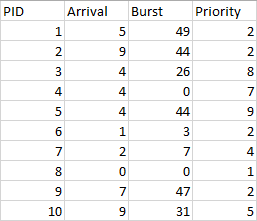
Sample test case #2:



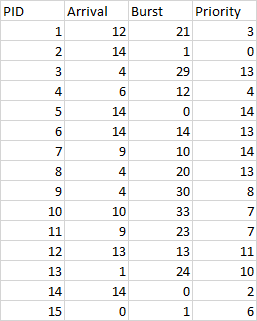
Sample test case #3:



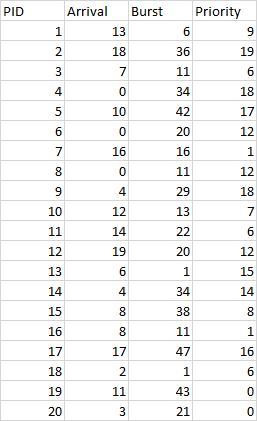
Sample test case #1:



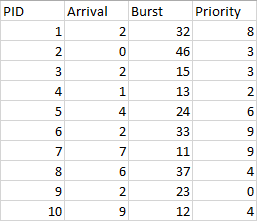
Sample test case #2:



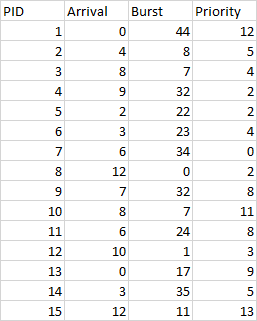
Sample test case #3:



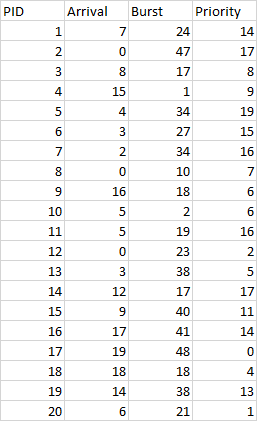
Sample test case #1:



Sample test case #2:



Sample test case #3:



References

1. Studytonight, 2018. What is CPU Scheduling? Retrieved April 20, 2018, from https://www.studytonight.com/operating-system/cpu-scheduling
2. CPU Scheduling. Retrieved April 20, 2018, from https://www.cs.uic.edu/~jbell/CourseNotes/OperatingSystems/5\_CPU\_Scheduling.html
3. Romero, V.M., 2018, CPU Scheduling.