

Table of contents

Introduction.....	1
Shortest Job First (SJF) NON-PREEMTIVE.....	2
Flow chart.....	3
Block diagram.....	4
Source code/ Simulation (Input).....	5
Simulations (Output).....	5
Scope and limitations.....	6
Recommendations.....	6
Hurdles.....	7
Documentations.....	8
Curriculum Vitae.....	9

Introduction

In the world of operating systems, effective CPU scheduling is absolutely vital for maximizing system efficiency. Without proper CPU scheduling, system performance can be significantly compromised. The deliberate distribution of the CPU's processing time among multiple processes is necessary for enabling the execution of one while others may be temporarily deferred due to factors such as I/O operations.

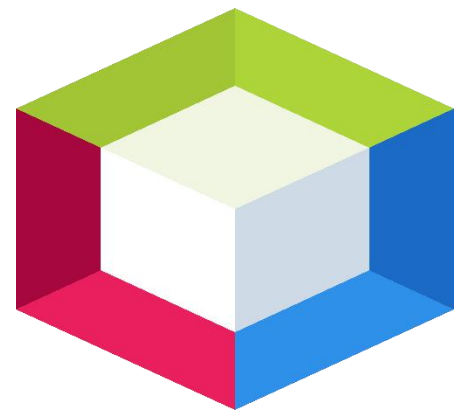
CPU scheduling is not just useful for improving system speed, fairness, and efficiency; it is absolutely essential. Therefore, our project aims to create a scheduling program that will clearly demonstrate the workings of this algorithm. Through this program, we will be able to see how different processes are prioritized and executed based on various factors. This will help us gain a deeper understanding of how CPU scheduling can significantly enhance the overall performance of an operating system.

For this project, we are tasked to create the preemptive Multi-Level shortest Job First (MLSJF) algorithm. Within this project, the following tools used is going to be used for this exercise.

JAVA – a programming language that is widely used due to being a flexible, independent, and provides lot of possibilities such as object oriented, networkcentric, and many more in which can be used as a platform.



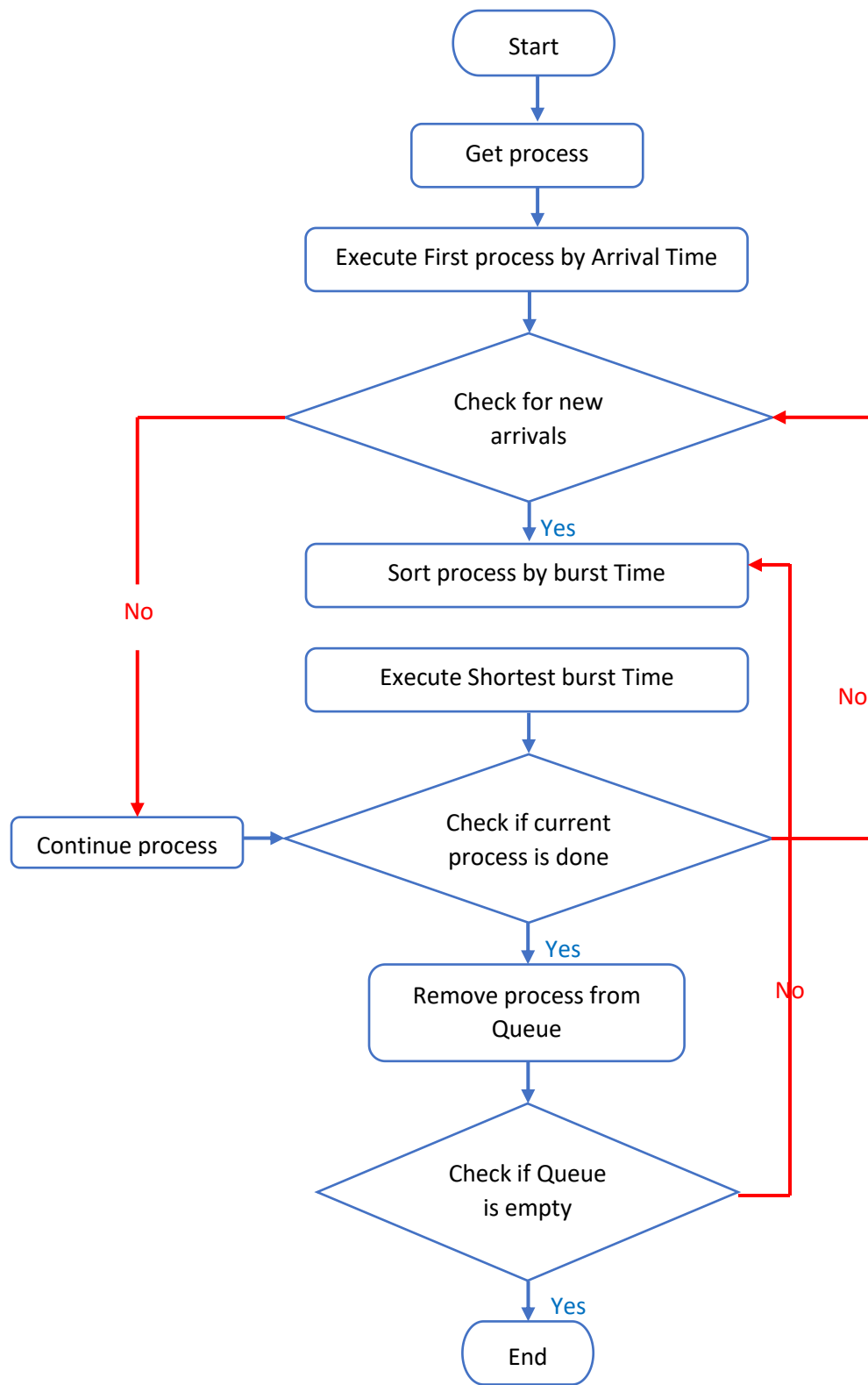
NetBeans IDE – is an integrated development environment (IDE) for the programming language Java.



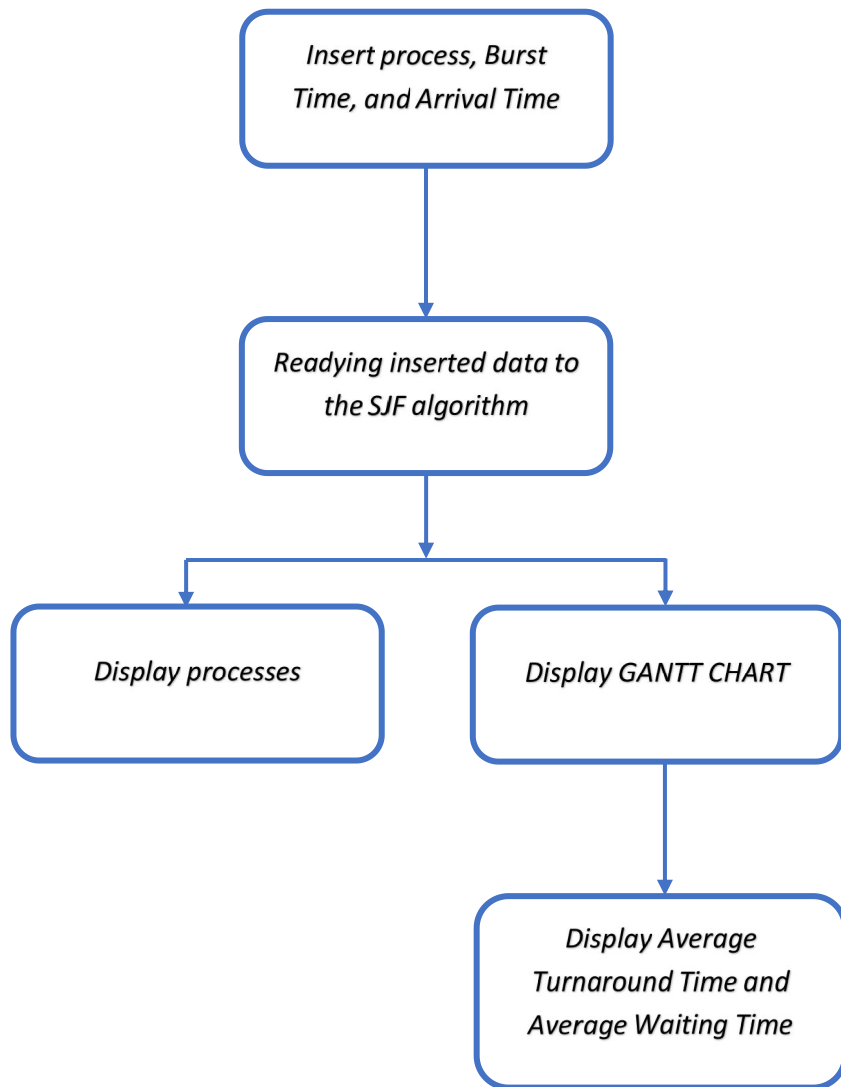
Shortest Job First (SJF) NON-PREEMTIVE

In the Shortest Job First (SJF) algorithm, processes are arranged in the order of their burst times, which represents the time needed for a process to finish its execution. The process with the smallest burst time is given priority. In the non-preemptive version of the Shortest Job First (SJF) algorithm, processes are scheduled based on their burst times, with the process having the shortest burst time given priority. Once a process begins its execution, it continues until completion without interruption. The scheduler selects the next process with the shortest burst time only after the current process finishes its execution.

FLOW CHART



BLOCK DIAGRAM

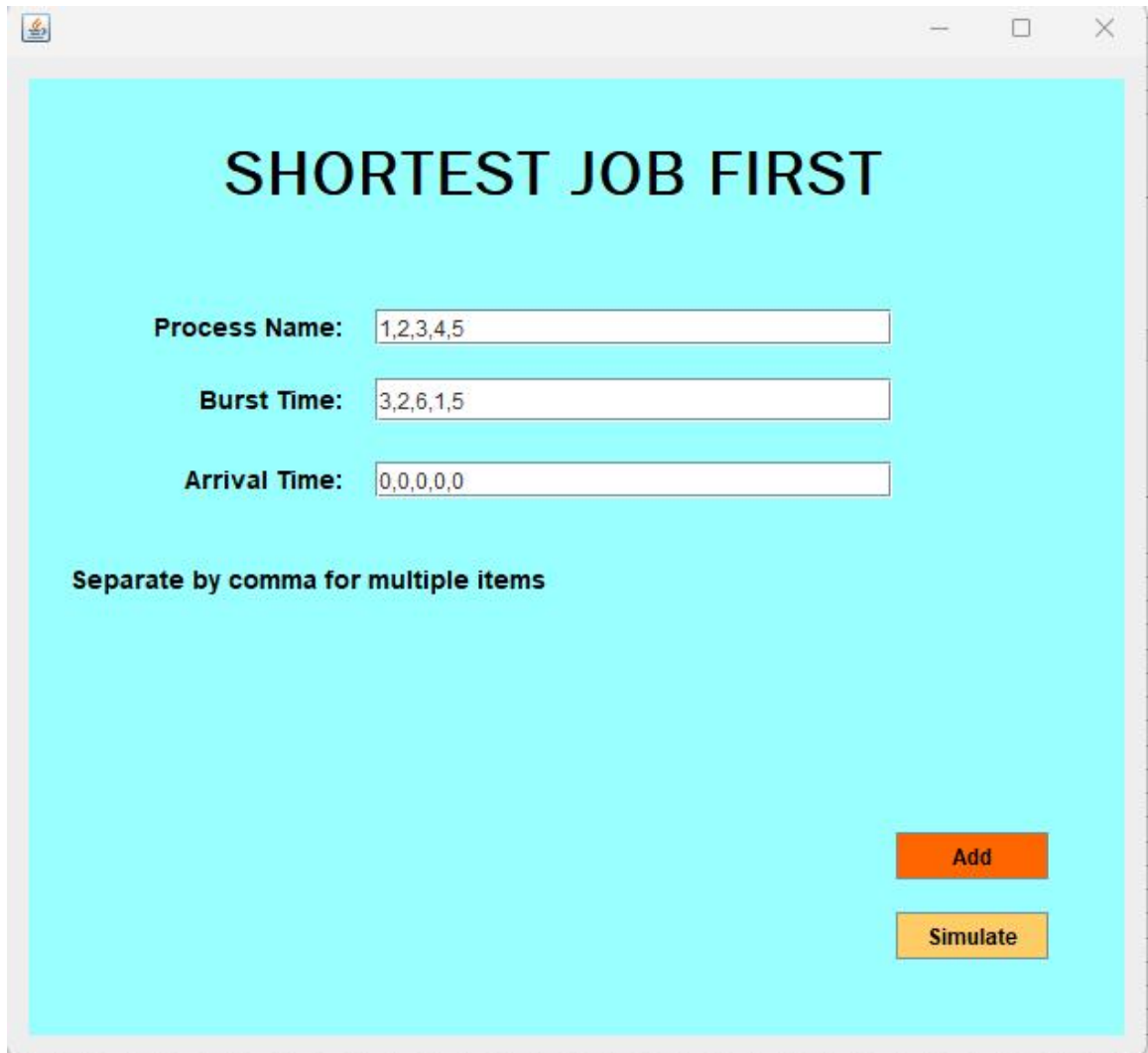


Source code

<https://github.com/itsmemike52/SJF-NON-PREEMTIVE->

Simulation

Input



The screenshot shows a Java Swing window with a light blue background. The title bar at the top contains a small icon on the left and standard minimize, maximize, and close buttons on the right. The main content area has the text "SHORTEST JOB FIRST" in large, bold, black capital letters. Below this, there are three input fields, each preceded by a label: "Process Name:" with the value "1,2,3,4,5", "Burst Time:" with the value "3,2,6,1,5", and "Arrival Time:" with the value "0,0,0,0,0". Below these fields is the instruction "Separate by comma for multiple items". In the bottom right corner, there are two buttons: an orange "Add" button and a yellow "Simulate" button.

SHORTEST JOB FIRST

Process Name: 1,2,3,4,5

Burst Time: 3,2,6,1,5

Arrival Time: 0,0,0,0,0

Separate by comma for multiple items

Add

Simulate

Output

PROCESSSES

Process Name	Arrival Time	Burst Time
4	0	1
2	0	2
1	0	3
5	0	5
3	0	6

GANTT CHART

Process Name	Arrival Time	Process Time	Burst Time	Turn Around Time	Waiting Time
4	0	0	1	1	0
2	0	1	2	3	1
1	0	3	3	6	3
5	0	6	5	11	6
3	0	11	6	17	11

Average Turnaround Time: **7 Units**

Average Waiting Time: **2 Units**

BACK

Scope and Limitations

SJF non-preemptive scheduling aims to minimize the total waiting time and turnaround time, making it an optimal algorithm in terms of minimizing the average completion time. It is efficient when the burst times of processes are accurately known in advance. It schedules the process with the shortest burst time first, leading to better resource utilization. SJF can achieve high throughput when it consistently selects processes with shorter burst times, allowing for more processes to be completed in a given time frame. SJF non-preemptive scheduling has some problems like longer tasks might wait too much, we need to know task times beforehand, it can lead to underusing resources, it's a bit hard to set up, it's not so predictable with changing situations, and it doesn't consider task priority.

Recommendations

In order to have a successful execution of my project, you need to input numbers with a comma after it because the comma determines that it will go to the next execution. As you can see it has a normal GUI and not so stylish because I'm out of time but still, it works. So I recommend to input the numbers according the rules I given so that it will have a successful executions. There is a possibility of bugs and error of this so if there is, you can tell me so that I'll make a resolve.

Hurdles

In my project I don't really know how to start about the codes because I'm not really good at it but I started to think and brain storming about it since I know how the short job first works . It's just about the code but anyways I keep myself productive and do some research, though it's a shame that I'm relying on the internet but most importantly I'm not copying any but just use the internet as a guide. I manage to do my task even though it drain a lot of time. I did it step by step, do some trial and error, and resolve it eventually. There is actually a time that the problem was so hard to fix like how I make the process number successive and I am started to have an anxiety but I keep myself strong and do it again until I fixed it.

Though the challenge was really hard and since I'm a sensitive and emotional person, there was a time that I was nearly giving up but I was thinking if they can make it, then why can't I? So I open my laptop again and try and try until I make it.

So you who will going to read my documentations. Please do not give up because trying and failing is not bad but a good lesson to be better.(I'm not better though)

Documentations

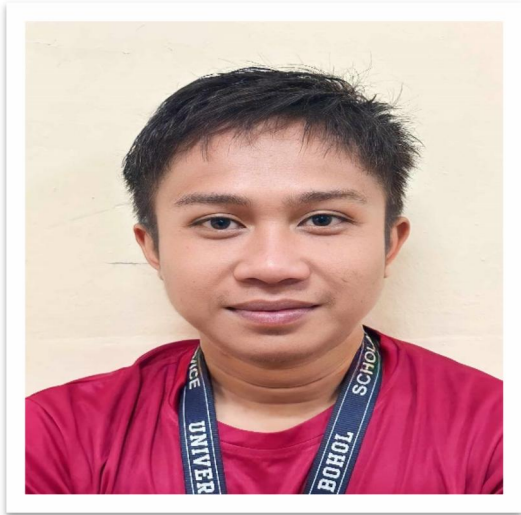
This is the time when I'm defending my project in the faulty, It work well though but there is some missing so I debugged it.



After a few hours, I finally did it successfully, because of my classmate who help and guide me. Thank to our teacher for accepting my project.



Curriculum Vitae



Name: John Michael D. Conarco

Age: 24 Years Old

Date of Birth: May 24, 1999

Status: Single

Address: Tupas, Antequera, Bohol

Mobile: 09102965485

Email: jmdconarco@universityofbohol.edu.ph

Motto: *You are never too old to set another goal or to dream a new dream.*

Portfolio: <https://michael-2su.pages.dev/>