CENTRAL UNIVERSITY OF TAMIL NADU

DEPARTMENT OF COMPUTER SCIENCE.

MASTER OF SCIENCE(C.Sc)

FIRST COMES FIRST SERVE SCHEDULING

A PROJECT REPORT

Submitted by

DIVYASRI S R - (P201306)

RAMELLI ANVESH - (P2013)

SUSMINU S KUMAR - (P2013)

ANDUGULA SHIVALEELA - (P201302)

Under the guidance of

Dr. P. THIYAGARAJAN

TABLE OF CONTENTS

ABSTRACT:

The project named “First Come First Serve Scheduling” which is undertaken by us, is basically a python program that simulates the First Come First Serve Scheduling Algorithm.

INTRODUCTION:

An Operating System can be defined as an interface between user and hardware. OS is responsible for the execution of all the processes, Resource Allocation, CPU management, File Management and many other tasks. The purpose of an operating system is to provide an environment in which a user can execute programs in convenient and efficient manner.

Scheduling is one of the basic functions of any Operating System. Scheduling algorithms plays a most important role in the OS design because scheduling of all the computer resources (like CPU) needs to be done before using it. The goal of scheduling is to possibly make the best use of the CPU i.e., maximizing throughput, maximizing CPU utilization, minimising Response time, reducing Turnaround time and minimizing Waiting time.

FIRST COMES FIRST SERVE SCHEDULING:

In this project, we have mainly concentrated on First Come First Serve CPU Scheduling algorithm in which the processes get served by the CPU in ascending order of their arrival to the ready queue. This algorithm is non-preemptive in nature and the processes are treated equally. First Come First Serve, is just like FIFO (First in First out) Queue data structure, where the data element which is added to the queue first, is the one who leaves the queue first. FCFS algorithm is used in batch systems. In the real world, many applications such as Train and Bus Ticket Reservations, Movie ticket reservations, Bank transaction apps uses FCFS model.

AIM:

The aim of this project is to develop a python program on FCFS scheduling algorithm which accepts the user input such as the process id, execution time and arrival time. With the inputs the following needs to be implemented:

1)Drawing the Gantt chart for the given input with FCFS algorithm.

2)Calculating the waiting time of each process.

3)Calculating the average waiting time.

4)Calculating the turnaround time of each process.

5)Calculating average turnaround time.

CONVENTIONS / MODEL:

The first step we took before starting the project was to study about scheduling and FCFS concepts. Then we came up with different scenarios where we need to concentrate to do some operations to rule out the test scenarios. Once we understood the concept each member in the group started coding separately, to get various different ideas and for exploration purpose. When we started the coding, we kept an open mind for versatility of using different packages, functions and concepts. We explored our best on both the concepts of Operating System Scheduling and python programming. Once everyone tried their maximum to produce the codes, each and everyone of us revealed our trails to others in the group. The test cases were tested on all the programs. Considering both pros and cons of all programs we picked and combined the best features in every program and created a model. We then together worked on the model to get the finalized code which can give a reliable output for all the test cases.

IMPLEMENTATION:

The task was to implement FCFS code that accepts the user inputs.

For the python program of FCFS, we have used the Data structure List of Lists to store the values and manipulated it throughout the code to get the perfect outcomes.

The reason why we chose list of lists is that First of all, we're reducing multiple lines of code into few lines. Secondly, the code is faster, as Python will allocate the list's memory first before adding the elements to it, instead of having to resize on runtime.

Few predefined functions under certain packages were incorporated in the program to make it more reliable. Some functions were defined by us in the code for interpretation and calculation purpose like sorting of arrival time, getting each element from the list.

The average waiting time and average turnaround time was calculated using predefined function of Library working with arrays

The table creation and drawing of the Gantt Chart were produced using python GUI libraries.

SOURCE CODE AND FUNCTIONALITY:

```
#import packages
import numpy as np
import plotly.figure_factory as ff
import matplotlib.pyplot as plt

#Take input from the user and append it to list of list
p = int(input("Enter total number of processes: "))
la=[]
for i in range (p):
    at = int(input("Enter process arrival time for process %d: " %i))
    bt = int(input("Enter process burst time for process %d : " %i))
    data=[i,at,bt]
    la.append(data)

#sorting of arrival time
def arrival(elem):
    return elem[1]
la.sort(key=arrival)

#Calculate the Completion time and insert it to the list
completiontime = la[0][1] + la[0][2]
for i in range (len(la)):
    if i == 0:
        completiontime = la[0][1] + la[0][2]
        la[i].insert(3,completiontime)
    else:
        completiontime = completiontime + la[i][2]
        la[i].insert(3,completiontime)

#Calculate the Turn Around time and insert it to the list
for tat in range (len(la)):
    la[tat].insert(4,(la[tat][3] - la[tat][1]))
```

```
#Calculate the Waiting Time and insert it to the list
for wt in range (len(la)):
    la[wt].insert(5,(la[wt][4] - la[wt][2]))

#Gantt Chart data
gs=[]
for s in range (len(la)):
    if s == 0:
        st = la[0][1]
        gs.append(st)
    else:
        st = la[s][3] - la[s][2]
        gs.append(st)
gp=[row[0] for row in la]
gb=[row[2] for row in la]

#Plot the gantt chart
def gantt(Pro, b, st):
    for j in range(len(Pro)):
        i = Pro[j] - 1
        plt.barh(i, b[j],left=st[j])
gantt(gp, gb, gs)
plt.yticks(np.arange(max(gp)), np.arange(1, max(gp) + 1))
plt.title("Gantt Chart")
plt.xlabel("<-- Time -->")
plt.ylabel("<-- Process --> ")
plt.show()

#Table creation and calculation of averages
avg=np.mean(la, axis = 0)
la.insert(0,["Process","Arrival time", "Burst time","Completion time","Turn Around time","Waiting time"])
fig = ff.create_table(la)
fig.show()
print("The Average Turn Around Time of the processes is %f" %avg[4])
print("The Average Waiting Time of the processes is %f" %avg[5])
```

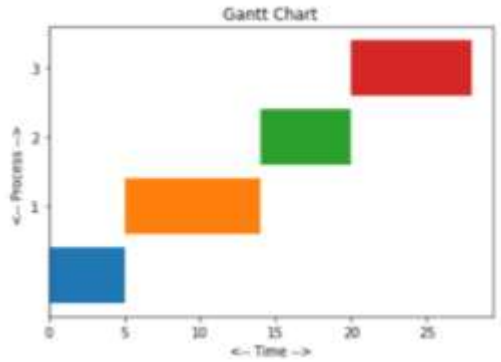
TEST CASES:

CASE1: Input of sorted Arrival time.

Input:

```
Enter total number of processes: 4
Enter process arrival time for process 0: 0
Enter process burst time for process 0 : 5
Enter process arrival time for process 1: 1
Enter process burst time for process 1 : 9
Enter process arrival time for process 2: 2
Enter process burst time for process 2 : 6
Enter process arrival time for process 3: 3
Enter process burst time for process 3 : 8
```

Output:



Process	Arrival time	Burst time	Completion time	Turn Around time	Waiting time
0	0	5	5	5	0
1	1	9	14	13	4
2	2	6	20	18	12
3	3	8	28	25	17

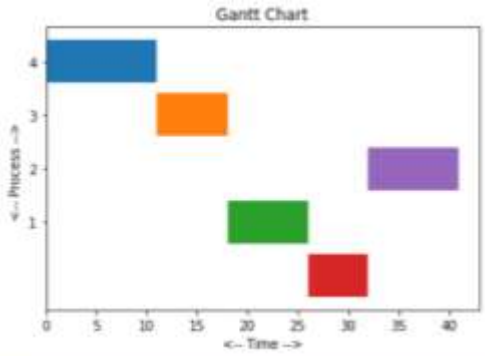
```
The Average Turn Around Time of the processes is 15.250000
The Average Waiting Time of the processes is 8.250000
```

CASE2: Input of Unsorted arrival time.

Input:

```
Enter total number of processes: 5
Enter process arrival time for process 0: 3
Enter process burst time for process 0 : 6
Enter process arrival time for process 1: 2
Enter process burst time for process 1 : 8
Enter process arrival time for process 2: 4
Enter process burst time for process 2 : 9
Enter process arrival time for process 3: 1
Enter process burst time for process 3 : 7
Enter process arrival time for process 4: 0
Enter process burst time for process 4 : 11
```

Output:



Process	Arrival time	Burst time	Completion time	Turn Around time	Waiting time
4	0	11	11	11	0
3	1	7	18	17	10
1	2	8	26	24	16
0	3	6	32	29	23
2	4	9	41	37	28

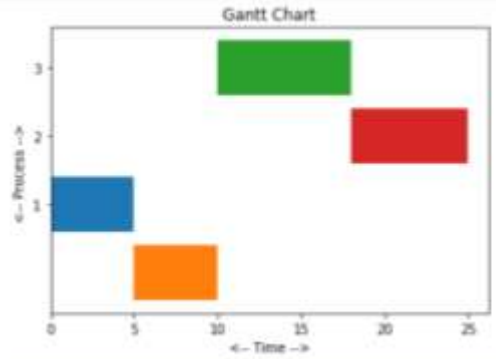
```
The Average Turn Around Time of the processes is 23.600000
The Average Waiting Time of the processes is 15.400000
```

CASE3: Two or more process having same arrival time.

Input:

```
Enter total number of processes: 4
Enter process arrival time for process 0: 1
Enter process burst time for process 0 : 5
Enter process arrival time for process 1: 0
Enter process burst time for process 1 : 5
Enter process arrival time for process 2: 2
Enter process burst time for process 2 : 7
Enter process arrival time for process 3: 1
Enter process burst time for process 3 : 8
```

Output:



Process	Arrival time	Burst time	Completion time	Turn Around time	Waiting time
1	0	5	5	5	0
0	1	5	10	9	4
3	1	8	18	17	9
2	2	7	25	23	16

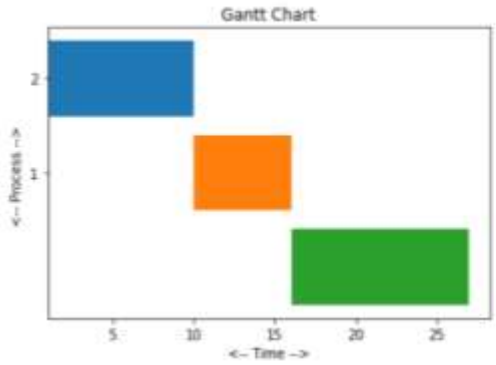
```
The Average Turn Around Time of the processes is 13.500000
The Average Waiting Time of the processes is 7.250000
```

CASE4: When the process arrives at non zero arrival time

Input:

```
Enter total number of processes: 3
Enter process arrival time for process 0: 3
Enter process burst time for process 0 : 11
Enter process arrival time for process 1: 2
Enter process burst time for process 1 : 6
Enter process arrival time for process 2: 1
Enter process burst time for process 2 : 9
```

Output:



Process	Arrival time	Burst time	Completion time	Turn Around time	Waiting time
2	1	9	10	9	0
1	2	6	16	14	8
0	3	11	27	24	13

```
The Average Turn Around Time of the processes is 15.666667
The Average Waiting Time of the processes is 7.000000
```

RESULT:

The algorithm which we developed to code was tested on various inputs. The outputs produced by the program exactly gives the perfect answers that we worked out theoretically in the OS class. Apart from just implementing the algorithm we have tried using this in the real time Digi Cart application, but due to time constraints and limited resources we were unable to complete the project

But in future the following can be implemented:

1) Using Pandas library big data input file can be accepted and FCFS can be implemented on the data.
2) Adding OOPS concepts to the program will make the program more versatile.

CONCLUSION:

The First Come First Serve Scheduling project helped us to get a better understanding on scheduling concepts, how it works and to implement the algorithm both in program and real world technical problems. The project have also given us the chance to explore more on python and its libraries, inculcate the exploration into our ideas, execute it and analyze the quality of the results we have got in every stages

REFERENCE:

- *http://www.cs.put.poznan.pl/akobusinska/downloads/Operating_Systems_Concepts.pdf*
- *https://library.kre.dp.ua/Books/2-4%20kurs/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F%20%2B%20%D0%BC%D0%BE%D0%B2%D0%B8%20%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F/Python/Python_3_Object_Oriented_Programming.pdf*
- *https://indico.in2p3.fr/event/16864/contributions/63125/attachments/48552/61399/Python_Libraries.pdf*
- *https://matplotlib.org/stable/gallery/lines_bars_and_markers/barh.html*