Contents

[**USCSP301 - USCS303: Operating System (OS) Practical – 02** 2](#_Toc78051137)

[**Practical Date:** 24 – 07 – 2021 2](#_Toc78051138)

[**Alogrithm :-** 3](#_Toc78051139)

[**Step 1**: Input the number of processes required to be scheduled using SJF, burst time for each process. 3](#_Toc78051140)

[**Step 2**: Using enhanced bubble sort technique, sort the all given processes in ascending order according to burst time in a ready queue. 3](#_Toc78051141)

[**Step 3**: Calculate the Finish Time, Turn Around Time and Waiting Time for each process which in turn help to calculate Average Waiting Time and Average Turn Around Time required by CPU to schedule given set of process using SJF. 3](#_Toc78051142)

[**Implementation:-** 13](#_Toc78051143)

[**Input:-** 17](#_Toc78051144)

[**Output.** 18](#_Toc78051145)

[**Sample Output – 01** 19](#_Toc78051146)

[**Sample Output – 02** 20](#_Toc78051147)

[**Sample Output-03** 21](#_Toc78051148)

[**Sample Output-04** 22](#_Toc78051149)

## **USCSP301 - USCS303: Operating System (OS) Practical – 02**

### **Practical Date:** 24 – 07 – 2021

**Practical Aim:** Non-Preemtive CPU Schedualing algorithm where each process with the smallest burst time is executed time.

CPU scheduling algorithm are used for scheduling different process present in the ready queue with available resource in an optimal way so that each and every process get execute by CPU

Scheduling algorithm are broadly classified into two main type namely preemptive and non-preemptive .

FIRST COME FIRST OUT(FCFS) is also know as FIRST IN FIRST OUT (FIFO) SCHEDUAL algorithm is the and simplest CPU .

A process scheduling different process to be assigned to the CPU based on particular scheduling algorithm .there are six popular process scheduling algorithm which we are going to discuss in this chapter FIRST COME FIRST OUT(FCFS) scheduling

### **Alogrithm :-**

### **Step 1**: Input the number of processes required to be scheduled using SJF, burst time for each process.

### **Step 2**: Using enhanced bubble sort technique, sort the all given processes in ascending order according to burst time in a ready queue.

### **Step 3**: Calculate the Finish Time, Turn Around Time and Waiting Time for each process which in turn help to calculate Average Waiting Time and Average Turn Around Time required by CPU to schedule given set of process using SJF.

**Step 3.1:** for i = 0, Finish Time To = Arrival Time To + Burst Time T0

**Step 3.2:** for i >= 1, Finish Time T₁ = Burst Time T, + Finish Time Ti-1

**Step 3.3:** for i= 0, Turn Around Time T0 Finish Time T0 - Arrival Time T0

**Step 3.4:** for i >= 1, Turn Around Time T₁ = Finish Time T₁ - Arrival Time T

**Step 3.5:** for i = 0, Waiting Time To Turn Around Time T0 - Burst Time T0

**Step 3.6:** for i >= 1, Waiting Time T₁ = Turn Around Time T₁ - Burst Time Ti-1

**Step 4:** Process with less arrival time comes first and gets scheduled first by the CPU.

**Step 5:** Calculate the Average Waiting Time and Average Turn Around Time.

**Step 6:** Stop

**Solved Example :-**

**Example 1:** Consider the following example contain five processes .

|  |  |
| --- | --- |
| **Process Id** | **Burst Time** |
| P0 | 6 |
| P1 | 3 |
| P2 | 8 |
| P3 | 3 |
| P4 | 4 |

**Step 1:** Processes get execute according to their lowest burst time first .

|  |  |
| --- | --- |
| **Process Id** | **Burst Time** |
| P0 | 6 |
| P1 | 3 |
| P2 | 8 |
| P3 | 3 |
| P4 | 4 |

**Step 2:** Following shows the scheduling and execution of processes

**Step 2.1:**  At start P1 shortest execution time which is 0-3 second.

|  |  |
| --- | --- |
| **System time** | 0 |
| **Processes scheduling finish time** | P1 |
| **Finish time** | 0+3=3 |
| **Wating time** | 3-3=0 |
| **Turn Around time** | 3-0=3 |

**Step 2.2:**  next shortest execution time is for process P3 for duration 3-6 second.

|  |  |
| --- | --- |
| **System time** | 6 |
| **Processes scheduling finish time** | P1,p3 |
| **Finish time** | 3+3=6 |
| **Wating time** | 6-3=3 |
| **Turn Around time** | 6-0=6 |

**Step 2.3:** Next job with shortest execution time is P4 for a duration 6-10 second.

|  |  |
| --- | --- |
| **System time** | 10 |
| **Processes scheduling finish time** | P1,p3’p4 |
| **Finish time** | 6+4=10 |
| **Wating time** | 10-4=6 |
| **Turn Around time** | 10-0=10 |

**Step 2.4:**  Next job with shortest execution time is p0 for duration of 10-16 second.

|  |  |
| --- | --- |
| **System time** | 10 |
| **Processes scheduling finish time** | P1,p3,p3,p4,p0 |
| **Finish time** | 10+6=16 |
| **Wating time** | 16-6=10 |
| **Turn Around time** | 16-0=16 |

**Step 2.5 :** Similarly next job with shortest execution time is P2 for duration of 16-24 second.

|  |  |
| --- | --- |
| **System time** | 16 |
| **Processes scheduling finish time** | P1,p3,p3,p4,p0,p2 |

|  |  |
| --- | --- |
| **Finish time** | 16+8=24 |
| **Wating time** | 24-8=16 |
| **Turn Around time** | 24-0=24 |

**Step 3:** Calculate average wating time and average turn around time.

|  |
| --- |
| **Average wating time** = (0+3+6+10+16)/5  = 35/5  =7 |

|  |
| --- |
| **Average turn around time** =(3+6+10+16+24)/5  = 59/5  =11.8 |

**Step 4:** After scheduling of all provided processes.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Process id** | **Burst time** | **Arrival time** | **Finish Time**  **(Prev.Finish**  **Time + Burst**  **Time)** | **Turn Around**  **Time**  **(Finish Time- Arrival Time)** | **Waiting Time**  **(Turn Around**  **Time-Burst**  **Time)** |
| P1 | 3 | 0 | 0+3=3 | 3-0=3 | 3-3=0 |
| P3 | 3 | 0 | 3+3=6 | 6-0=6 | 6-3=3 |
| P4 | 4 | 0 | 6+4=10 | 10-0=10 | 10-4=6 |
| P0 | 6 | 0 | 10+6=16 | 16-0=16 | 16-6=10 |
| P2 | 8 | 0 | 16+8=24 | 24-0=24 | 24-8=16 |
| Average |  |  |  | 11.8000000 | 7.000000 |

**Gnatt chart : -**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **P0** | **P1** | **P2** | **P3** | **P4** |  |
| 0 | 3 | 6 | 10 | 16 | 24 |

**Example 2:** Consider the following example containing five processes arrive at same time.

|  |  |
| --- | --- |
| **Processes ID** | **Burst Time** |
| P0 | 2 |
| P1 | 1 |
| P2 | 6 |

**Solution :-**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Process id** | **Burst time** | **Arrival time** | **Finish Time**  **(Prev.Finish**  **Time + Burst**  **Time)** | **Turn Around**  **Time**  **(Finish Time- Arrival Time)** | **Waiting Time**  **(Turn Around**  **Time-Burst**  **Time)** |
| P1 | 1 | 0 | 1 | 1 | 1 |
| P0 | 2 | 0 | 3 | 3 | 3 |
| P2 | 6 | 0 | 9 | 9 | 9 |
| Average |  |  |  | 4.33333 | 1.33333 |

|  |  |  |  |
| --- | --- | --- | --- |
| **P0** | **P1** | **P2** |  |
| 0 | 1 | 3 | 9 |

**Example 3:** Consider the following example contain five processes arrive at same time .

|  |  |  |
| --- | --- | --- |
| **Process ID** | **Burst time** | |
| P0 | 25 | |
| P1 | 15 | |
| P2 | 10 | |
| P3 | 25 | |
| P4 | 10 | |
| P5 | | 25 |

**Solution :-**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Process id** | | **Burst time** | | **Arrival time** | | **Finish Time**  **(Prev.Finish**  **Time + Burst**  **Time)** | | **Turn Around**  **Time**  **(Finish Time- Arrival Time** | | **Waiting Time**  **(Turn Around**  **Time-Burst**  **Time)** | |
| P2 | | 10 | | 0 | | 10 | | 10 | | 0 | |
| P4 | | 10 | | 0 | | 20 | | 20 | | 10 | |
| P1 | | 15 | | 0 | | 35 | | 35 | | 20 | |
| P0 | | 25 | | 0 | | 60 | | 60 | | 35 | |
| P3 | | 25 | | 0 | | 85 | | 85 | | 60 | |
| P5 | | 25 | | 0 | | 110 | | 110 | | 85 | |
| Average | |  | |  | |  | | 53.3333 | | 35.000000 | |
| **P2** | **P4** | | **P1** | | **P0** | | **P3** | | **P5** | |  |
| 0 | 10 | | 20 | | 35 | | 60 | | 85 | | 110 |

**Example 4:** Consider the following example contain five processes arrive at same time .

|  |  |
| --- | --- |
| **Process Id** | **Burst Time** |
| P0 | 7 |
| P1 | 3 |
| P2 | 2 |
| P3 | 10 |
| P4 | 8 |

**Step 4:** After scheduling of all provided processes.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Process id** | **Burst time** | **Arrival time** | **Finish Time**  **(Prev.Finish**  **Time + Burst**  **Time)** | **Turn Around**  **Time**  **(Finish Time- Arrival Time** | **Waiting Time**  **(Turn Around**  **Time-Burst**  **Time)** |
| P2 | 2 | 0 | 2 | 2 | 0 |
| P1 | 3 | 0 | 5 | 5 | 2 |
| P0 | 7 | 0 | 12 | 12 | 5 |
| P4 | 8 | 0 | 20 | 20 | 12 |
| P3 | 10 | 0 | 30 | 30 | 20 |
| Average |  |  |  | 13.80000 | 7.800000 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **P2** | **P1** | **P0** | **P4** | **P3** |  |
| 0 | 2 | 5 | 12 | 20 | 30 |

### **Implementation:-**

//Name: Yash Patil

//Batch:B2

//PRN: 2020016400809191

//Date:24/7/2021

//Prac-02: SJF(with no preemption)Algorithm

import java.util.Scanner;

public class P2\_SJF\_YP

{

int burstTime[];

int arrivalTime[]={0};

String[] processId;

int numberOfProcess;

void getProcessData(Scanner input){

System.out.println("enter the number of process for Scheduling:");

int inputNumberOfProcess=input.nextInt();

numberOfProcess=inputNumberOfProcess;

burstTime=new int[numberOfProcess];

arrivalTime=new int[numberOfProcess];

processId=new String[numberOfProcess];

String st="p";

for(int i=0;i < numberOfProcess;i++){

processId[i]=st.concat(Integer.toString(i));

System.out.print("enter the burst time for process-"+(i)+":");

burstTime[i]=input.nextInt();

}

}

void sortAccordingBurstTime(int[] at,int[] bt,String[] pid){

boolean swapped;

int temp;

String stemp;

for (int i=0;i<numberOfProcess;i++){

swapped=false;

for (int j = 0;j<numberOfProcess-i-1;j++){

if(bt[j]>bt[j+1]){

temp=bt[j];

bt[j]=bt[j+1];

bt[j+1]=temp;

temp=at[j];

at[j]=at[j+1];

at[j+1]=temp;

stemp=pid[j];

pid[j]=pid[j+1];

pid[j+1]=stemp;

swapped=true;

}

}

if(swapped==false){

break;

}

}

}

void shortestJobFirstNPAlgorithm(){

int finishTime[]=new int[numberOfProcess];

int bt[]=burstTime.clone();

int at[]=arrivalTime.clone();

String pid[]=processId.clone();

int waitingTime[]=new int[numberOfProcess];

int turnAroundTime[]=new int[numberOfProcess];

sortAccordingBurstTime(at,bt,pid);

finishTime[0]=at[0]+bt[0];

turnAroundTime[0]=finishTime[0]-at[0];

waitingTime[0]=turnAroundTime[0]-bt[0];

for(int i=1;i<numberOfProcess;i++){

finishTime[i]=bt[i]+finishTime[i-1];

turnAroundTime[i]=finishTime[i]-at[i];

waitingTime[i]=turnAroundTime[i]-bt[i];

}

float sum=0;

for(int n:waitingTime){

sum+=n;

}

float averageWaitingTime=sum/numberOfProcess;

sum=0;

for(int n:turnAroundTime){

sum+=n;

}

float averageTurnAroundTime=sum/numberOfProcess;

System.out.println("SJF (with no preemption) Scheduling Algorithm :");

System.out.format("%20s%20s%20s%20s%20s%20s\n","ProcessId","BurstTime"

,"ArrivalTime","FinishTime","TurnAroundTime","WatingTime");

for(int i=0;i<numberOfProcess;i++){

System.out.format("%20s%20d%20d%20d%20d%20d\n",pid[i],bt[i],at[i]

,finishTime[i],turnAroundTime[i],waitingTime[i]);

}

System.out.format("%80s%20f%20f\n", "Average",averageTurnAroundTime,averageWaitingTime);

}

public static void main(String[] args){

Scanner input=new Scanner(System.in);

P2\_SJF\_YP obj=new P2\_SJF\_YP();

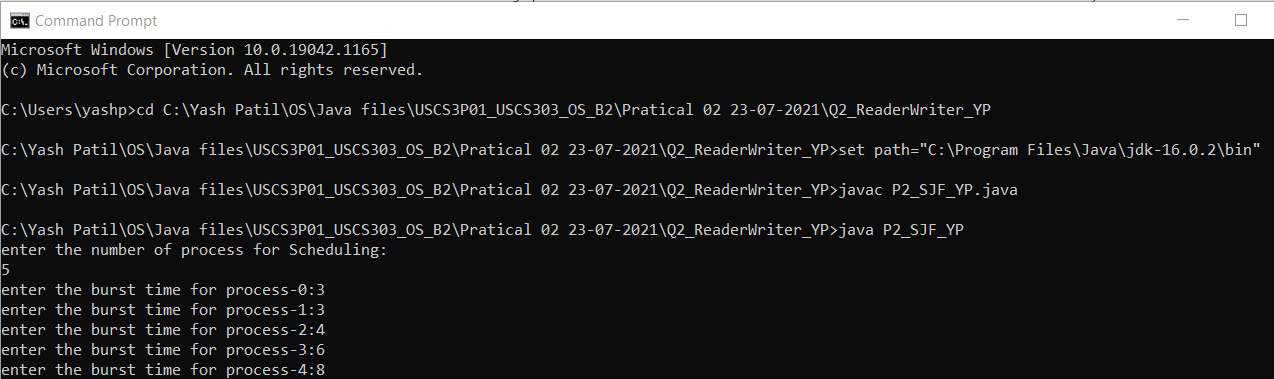
obj.getProcessData(input);

obj.shortestJobFirstNPAlgorithm();

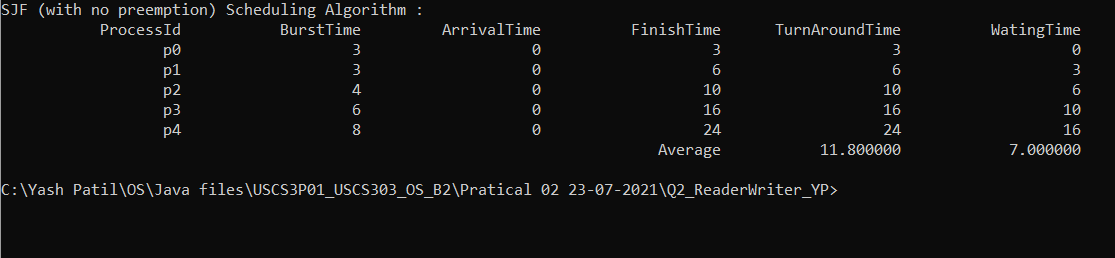
}

}

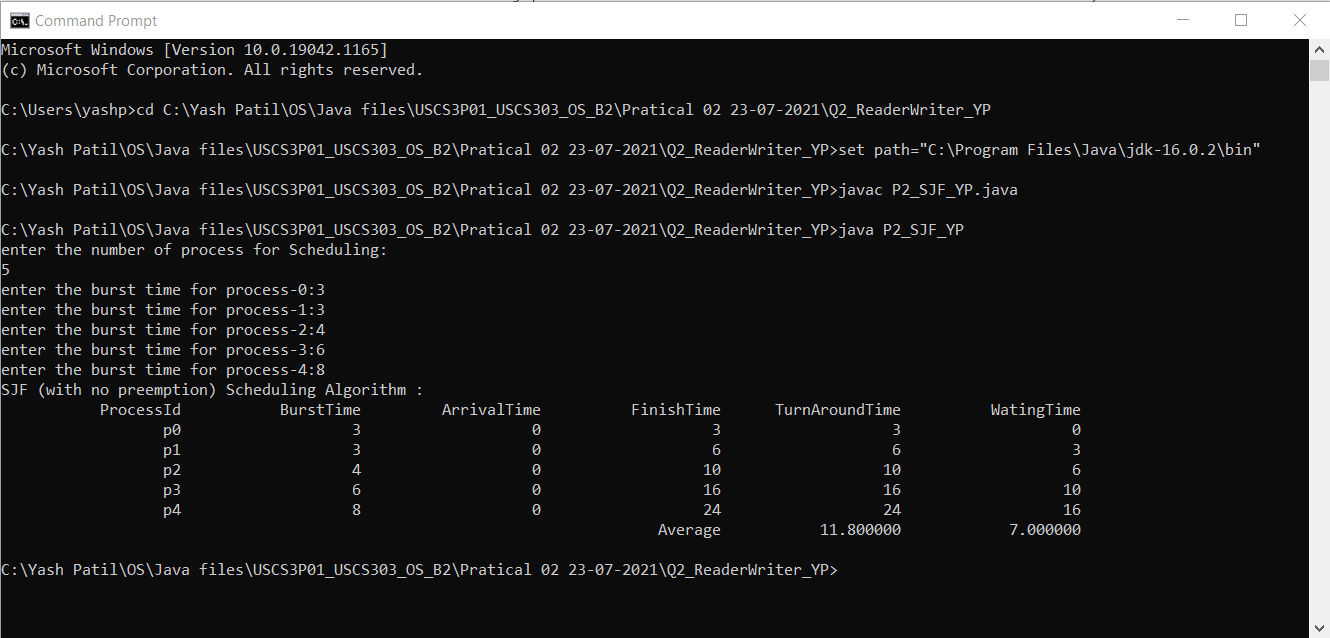
### **Input:-**



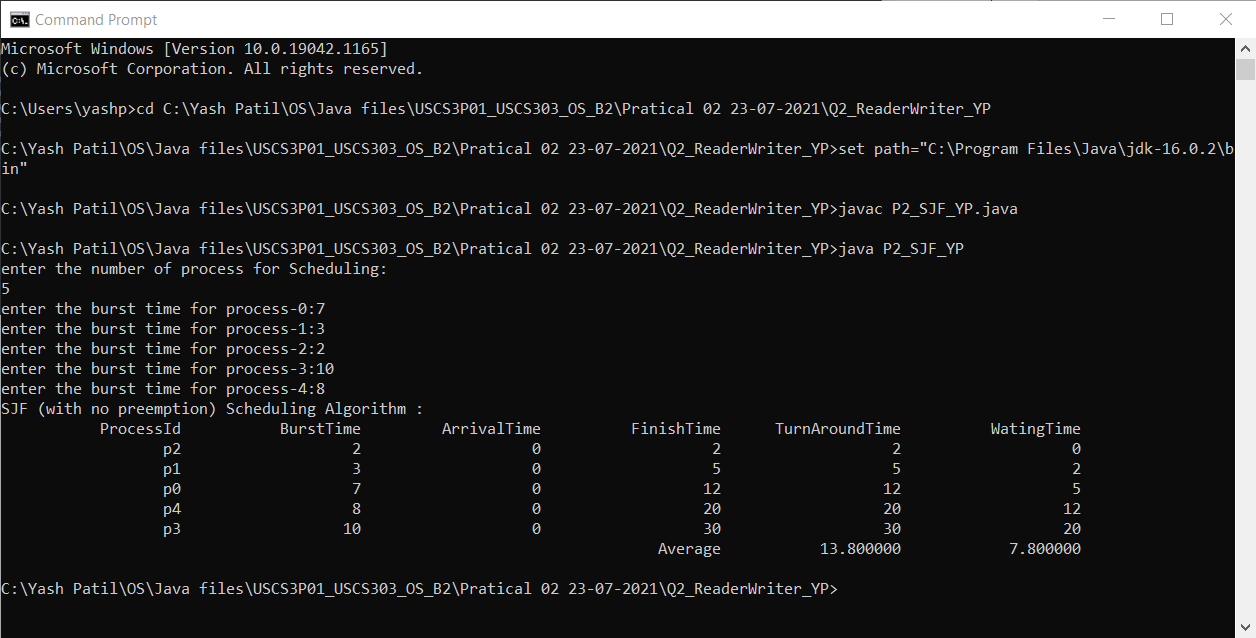
**Output.**



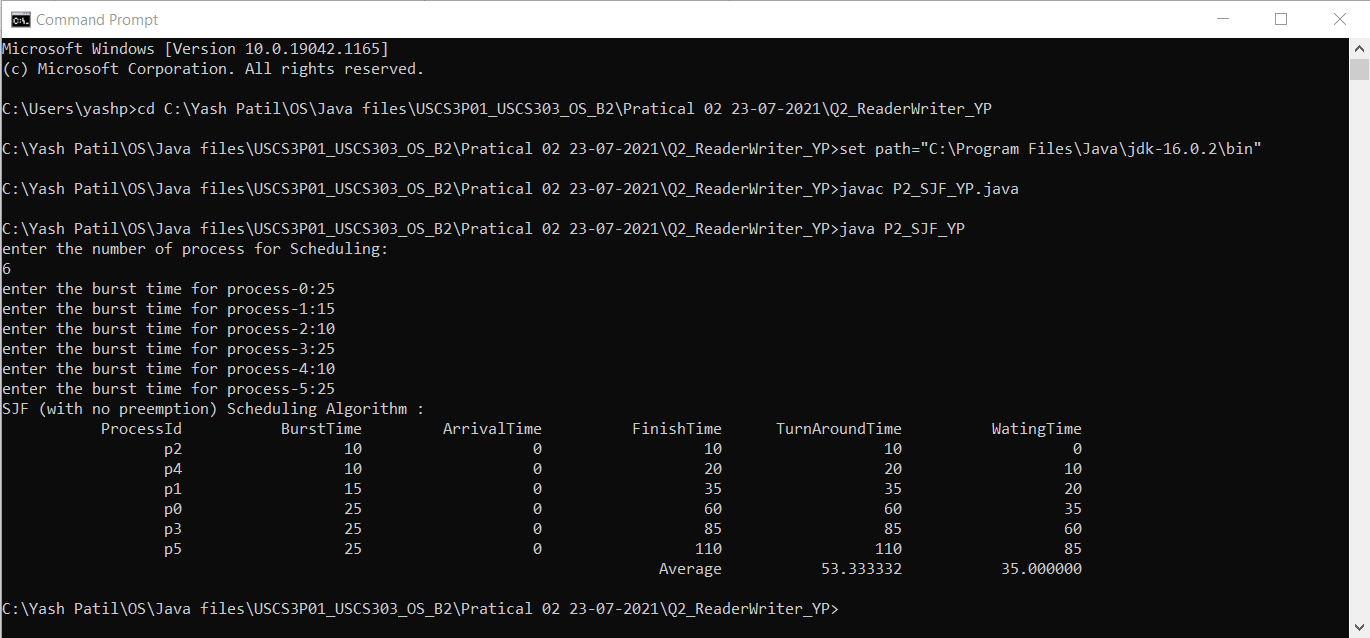
### **Sample Output – 01**



### **Sample Output – 02**



### **Sample Output-03**



### **Sample Output-04**

