**Contents**

[USCSP301 – USCS303 : Operating system (OS) Practical – 02 2](#_Toc81558183)

[Practical – 02 Shortest Job first Scheduling Algorithm 2](#_Toc81558184)

[Practical Date : 24 July 2021 2](#_Toc81558185)

[Practical Aim : Implement SJF (with no preemption) scheduling algo in Java 2](#_Toc81558186)

[Algorithm : 2](#_Toc81558187)

[Solved Example 2](#_Toc81558188)

[Gnatt Chart 5](#_Toc81558190)

[Implementation 8](#_Toc81558191)

[Input 14](#_Toc81558197)

[Output :- 14](#_Toc81558198)

[Sample output 01 15](#_Toc81558199)

[Sample output 02 15](#_Toc81558200)

[Sample output 03 16](#_Toc81558201)

[Sample output 04 16](#_Toc81558202)

**USCSP301 – USCS303 : Operating system (OS) Practical – 02**

**Practical – 02 Shortest Job first Scheduling Algorithm**

# Practical Date : 24 July 2021

# Practical Aim : Implement SJF (with no preemption) scheduling algo in Java

**Algorithm :**

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 To

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

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

Step 3.4: for i >= 1, Turn Around Time Ti = Finish Time To - Arrival Time To

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

Step 3.6: for i >= 1, Waiting Time Ti- 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 01**

Consider the following example containing five processes arrive at same time

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

Step 1: Processes get executed according to their lowest burst time.

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

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

Step 2.1: At start P1 has the shortest execution time which is 0-3 seconds

System Time : 0

Process Scheduled : P1

Finish Time : 0 + 3 = 3

Waiting Time : 3 – 3 = 0

Turn Around Time : 3 - 0 = 3

Step 2.2: Next shortest execution time for process P3 for duration 3-6 seconds.

System Time : 3

Process Scheduled : P1,P3

Finish Time : 3 + 3 = 6

Waiting Time : 6 – 3 = 3

Turn Around Time : 6 - 0 = 0

Step 2.3: Next shortest execution time for process P4 for duration 6-10 seconds.

System Time : 6

Process Scheduled : P1,P3,P4

Finish Time : 6 + 4 = 10

Waiting 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 seconds.

System Time : 10

Process Scheduled : P1,P3,P4,P0

Finish Time : 10 + 6 = 16

Waiting 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 seconds.

System Time : 16

Process Scheduled : P1,P3,P4,P0,P2

Finish Time : 16 + 8 = 24

Waiting Time : 24 – 8 = 16

Turn Around Time : 24 – 0 = 24

Step 3: Calculate Average Waiting Time and Average Turn Around Time.

Average Waiting 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 | 7.0000000 |

Step 5 : Stop

**Gnatt Chart**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **P1** | **P3** | **P4** | **P0** | **P2** |

Example 02

|  |  |  |  |
| --- | --- | --- | --- |
| Process ID | | Burst Time | |
| P0 | | 2 | |
| P1 | | 1 | |
| P2 | | 6 | |
| 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 | | (0+1=)1 | | (2-0=)1 | | (2-2=)0 |
| P0 | 2 | 0 | | (1+2=)3 | | (3-0=)3 | | (3-1=)1 |
| P2 | 6 | 0 | | (3+6=)9 | | (9-0=)9 | | (9-6=)3 |
| Average |  |  | |  | | 4.333333 | | 1.333333 |

**Gnatt Chart**

|  |  |  |
| --- | --- | --- |
| **P1** | **P0** | **P2** |

**Example 03**

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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+0=)2 | (2-0=)2 | (2-2=)0 |
| P1 | 3 | 0 | (2+3=)5 | (5-0=)5 | (5-3=)2 |
| P0 | 7 | 0 | (5+7=)12 | (12-0=)12 | (12-7=)5 |
| P4 | 8 | 0 | (20+8)20 | (20-0=)20 | (20-8=)12 |
| P3 | 10 | 0 | (21+10=)30 | (20-0=)30 | (30-10=)20 |
| Average |  |  |  | 13.800000 | 7.800000 |

**Gnatt Chart**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **P2** | **P1** | **P0** | **P4** | **P3** |

**Example 4**

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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 |  |  |  | 53.333332 | 35.000000 |

**Gnatt Chart**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **P2** | **P4** | **P1** | **P0** | **P3** | **P5** |

**Implementation**

**Java Program:-**

**//**Name: Yash Parab

//Batch: B1

//PRN: 2020016400922513

//Date: 24 July 2021

// Prac-02 : SJF 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.print("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("Enteer the BurstTime for process - " + (i) + " : ");

burstTime[i]=input.nextInt(); } //for loop ends

}

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]){

//swapping bursttime

temp=bt[j];

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

bt[j+1]=temp;

//swapping arrival time

temp=at[j];

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

at[j+1] = temp;

//swappingprocessid

stemp=pid[j];

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

pid[j+1]=stemp;

//enhanched bubble sort

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) Schedulling Algorithm :");

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

"ProcessId", "BurstTime","ArrivalTime","FinishTime","TurnAroundTime", "WaitingTime");

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();

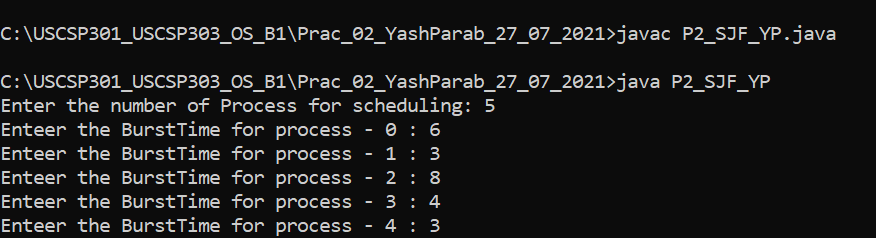
obj.getProcessData(input);

obj.shortestJobFirstNPAlgorithm();

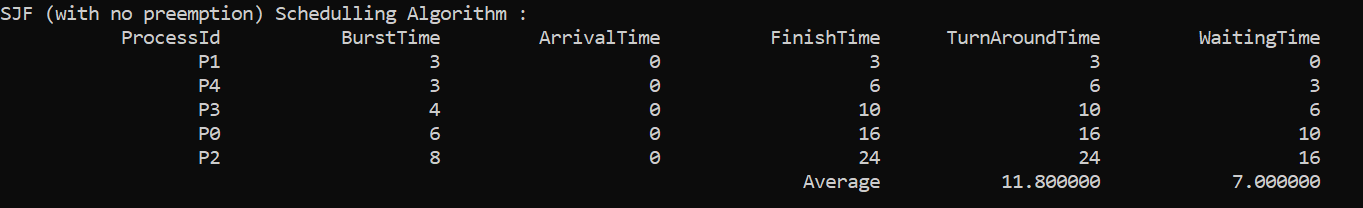
}

}

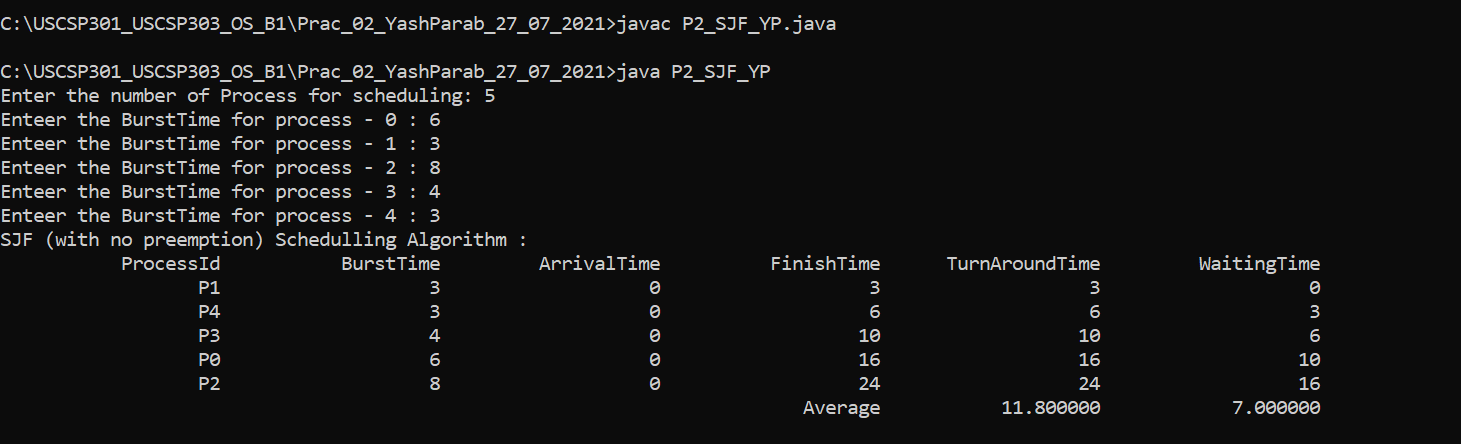
# Input



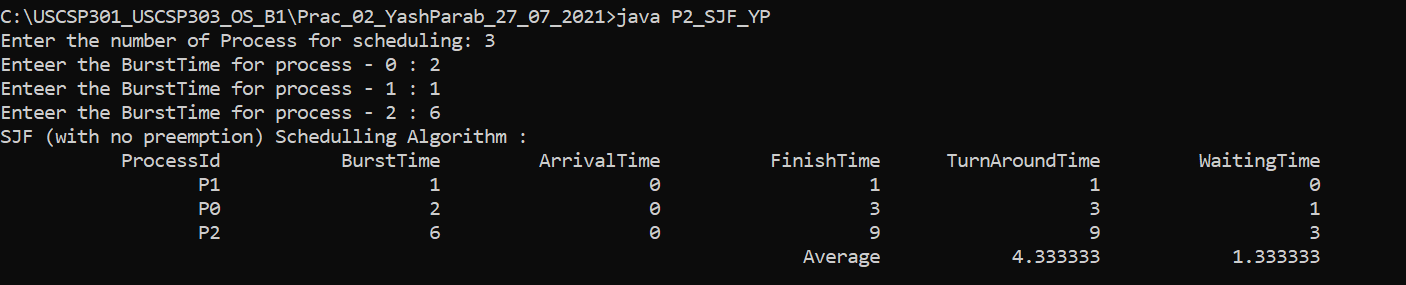
# Output :-



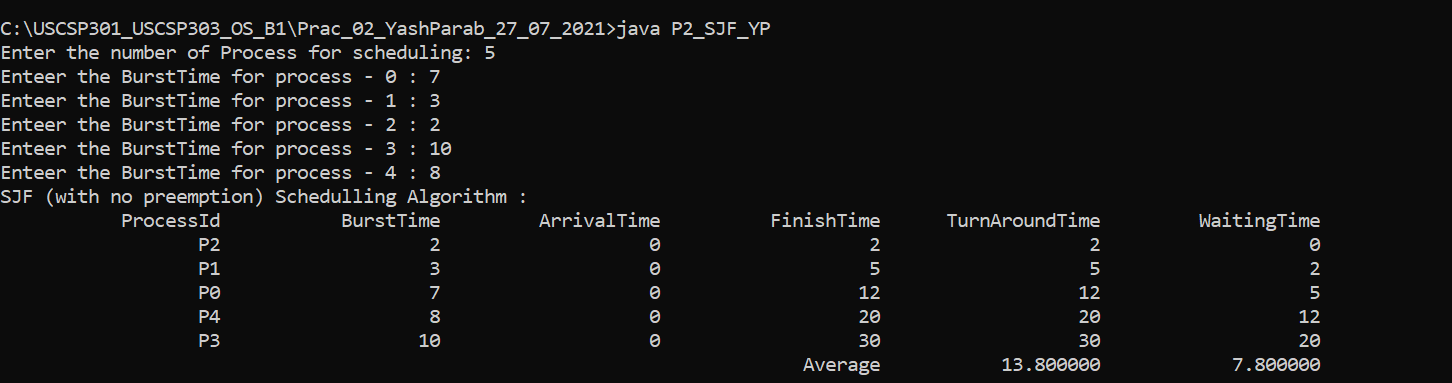
# Sample output 01



# Sample output 02



**Sample output 03**



**Sample output 04**

