Practical – 02: Shortest job first (SJF) Scheduling Algorithm …….....

Practical Date: 24th July, 2021 ……..

Practical Aim: Implement SJF (with no preemption) Scheduling Algorithm ..…….....

SJF is a Non-preemptive CPU Scheduling algorithm where each process with the **smallest burst time is executed first**. **Burst time** is the amount of time required by a process for its execution on the CPU..

More appropriate term for this scheduling method would be the **shortest next CPU burst** algorithm, because scheduling depends on the length of the next CPU burst of a process, rather than its total length.

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 T0 = Arrival Time T0 + Burst Time T0

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

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

Step 3.5: for i>=0, Waiting Time Ti = Turn Around Time Ti - 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.

Example of Shortest Job First algorithm

Consider the following example containing five processes arrive at same time.

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

Step 1: Processes get executed according to their arrival 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 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 is 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=6

Step 2.3: Next job with shortest execution time is P4 for a 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 the shortest duration 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	Turn Around Time	Waiting Time
			(Prev. finish time + Burst time	(Finish Time – Arrival 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= 26
			Average	11.800000	7.000000

Step 5: Stop.

Gnatt chart

P1	P3	P4	P0	P2	
0	3	6	10	16	24

Implement SJF scheduling algorithm in java

Implementation

Filename: P2_SJF_HS.java

Code:

```
// Name: Sumit Telawane
// Batch: B1
// PRN: 2020016400825777
// Date: 24th July, 2021
// Prac-02: SJF (with no preemption) Algorithm
import java.util.Scanner;
class P2_SJF_ST{
	// defining variables
	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]) {
        // swapping burst time
        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;
        // swapping process id
        stemp = pid[j];
        pid[j] = pid[j+1];
        pid[j+1] = pid[j+1];
        pid[j+1] = stemp;

        // enhanced 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);
  // calculating waiting & turn-around time for each process
  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;
    // print on console the order of processes scheduled using shorted job
    // first (with no preemption) Algorithm
    System.out.println("SJF(with no preemption) Scheduling 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", averageTurnAroundTime,averageWaitingTime);
  }
  public static void main(String[] args) {
    Scanner input = new Scanner(System.in);
    P2_SJF_ST obj = new P2_SJF_ST();
    obj.getProcessData(input);
    obj.shortestJobFirstNPAlgorithm(); } }
```

Input:

```
Command Prompt
'D:\os\p2' is not recognized as an internal or external command,
operable program or batch file.

D:\>CD D:\os\p2

D:\os\p2>javac P2_SJF_ST.java

D:\os\p2>java P2_SJF_ST
Enter the number of Process for Scheduling:
3
Enter the burst time for process - 0 : 2
Enter the burst time for process - 1 : 1
Enter the burst time for process - 2 : 6
```

Output:

```
SJF (with no preemption) Scheduling Algorithm :
          ProcessId          BurstTime        ArrivalTime         FinishTime      TurnAroundTime         WatingTime

                 p1                  1                  0                  1                  1                  0

                 p0                  2                  0                  3                  3                  1

                 p2                  6                  0                  9                  9                  3

                                                                     Average           4.333333           1.333333
```

Table:

Process Id	Burst Time	Arrival Time	Finish Time	Turn Around Time	Waiting Time
			(Prev. finish time + Burst time	(Finish Time – Arrival Time)	(Turn Around Time – Burst Time)
P1	1	0	0+1 = 1	1-0=1	1-0=0
P0	2	0	1+2 = 3	3-0=3	3-2=1
P2	6	0	3+6 = 9	9-0=9	6-3=3
			Average	4.333333	1.333333

Gnatt chart:

P1	P0	P2	
0	1	3	9

Sample output 1:

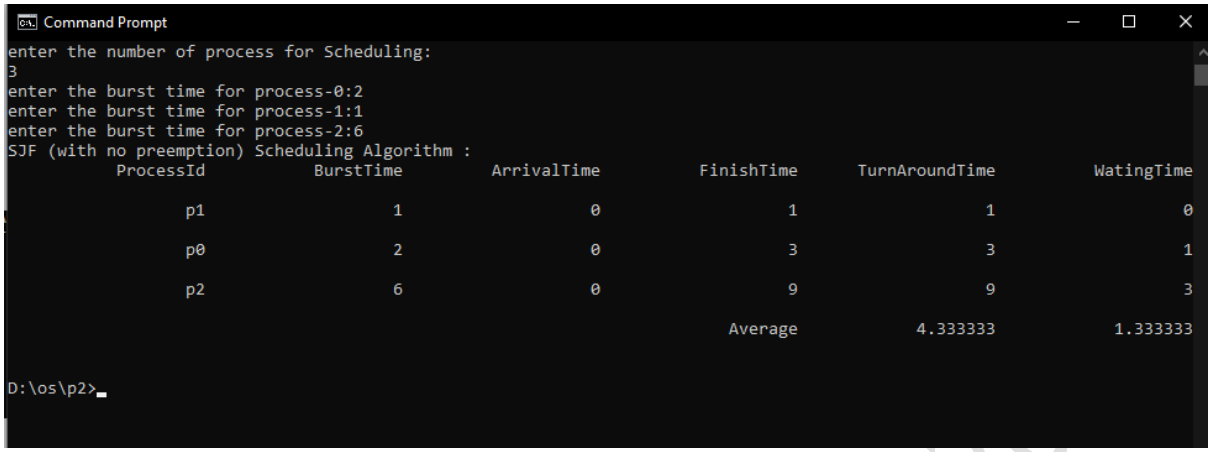

Table:

Process Id	Burst Time	Arrival Time	Finish Time	Turn Around Time	Waiting Time
			(Prev. finish time + Burst time	(Finish Time – Arrival Time)	(Turn Around Time – Burst Time)
P1	1	0	0+1 = 1	1-0=1	1-0=0
P0	2	0	1+2 = 3	3-0=3	3-2=1
P2	6	0	3+6 = 9	9-0=9	6-3=3
			Average	4.333333	1.333333

Gnatt chart:

P1	P0	P2	
0	1	3	9

Sample output 2:

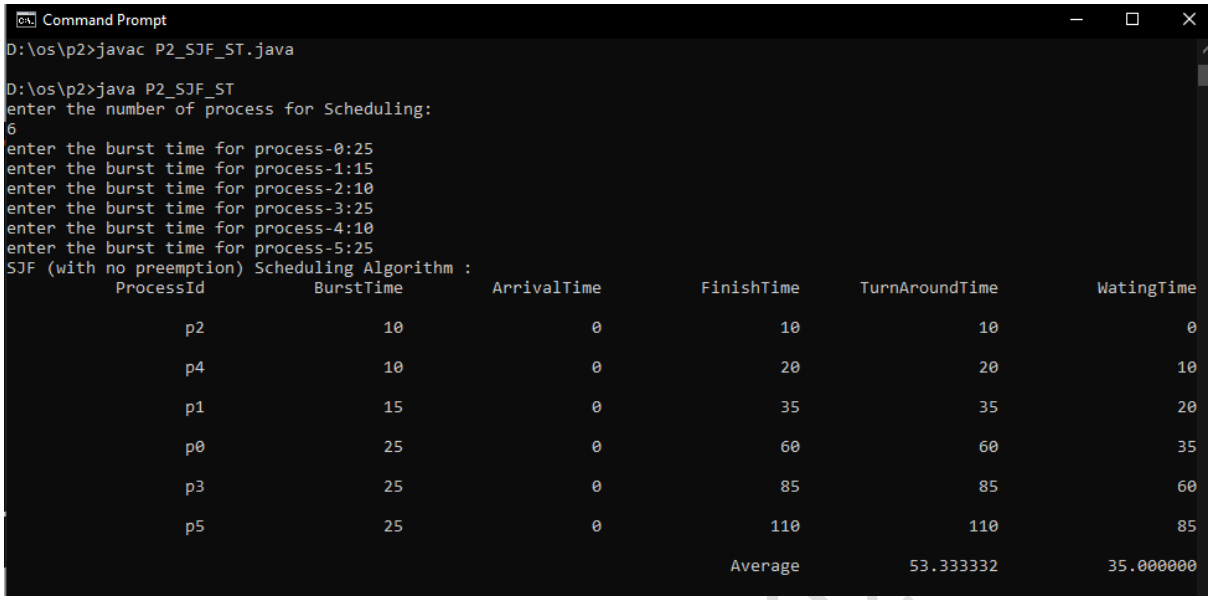

Table:

Process Id	Burst Time	Arrival Time	Finish Time	Turn Around Time	Waiting Time
			(Prev. finish time + Burst time	(Finish Time – Arrival 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	
0	10	20	35	60	85	110

Sample output 3:

```
D:\os\p2>javac P2_SJF_ST.java

D:\os\p2>java P2_SJF_ST
enter the number of process for Scheduling:
5
enter the burst time for process-0:6
enter the burst time for process-1:3
enter the burst time for process-2:8
enter the burst time for process-3:3
enter the burst time for process-4:4
SJF (with no preemption) Scheduling Algorithm :
          ProcessId          BurstTime        ArrivalTime         FinishTime      TurnAroundTime         WatingTime

                p1                  3                  0                  3                  3                  0

                p3                  3                  0                  6                  6                  3

                p4                  4                  0                 10                 10                  6

                p0                  6                  0                 16                 16                 10

                p2                  8                  0                 24                 24                 16

                                                                     Average        11.800000          7.000000
```

Table:

Process Id	Burst Time	Arrival Time	Finish Time	Turn Around Time	Waiting Time
			(Prev. finish time + Burst time	(Finish Time – Arrival Time)	(Turn Around Time – Burst Time)
P1	3	0	3	3	0
P3	3	0	6	6	3
P4	4	0	10	10	6
P0	6	0	16	16	10
P2	8	0	24	24	16
			Average	11.8000000	7.000000

Gnatt chart:

P1	P3	P4	P0	P2	
0	3	6	10	16	24

Sample output 4:

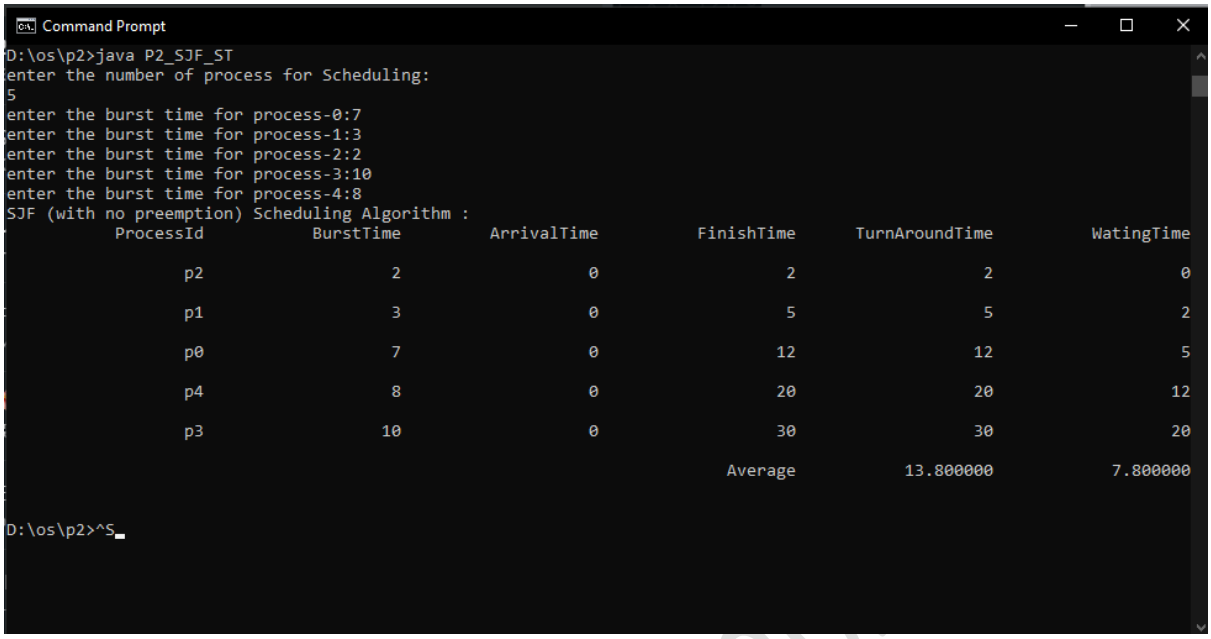

Table:

Process Id	Burst Time	Arrival Time	Finish Time	Turn Around Time	Waiting Time
			(Prev. finish time + Burst time	(Finish Time – Arrival 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	11.8000000	7.000000

Gnatt chart:

P2	P1	P0	P4	P3	
0	2	5	12	20	30