Lab Report No:	08
Lab Report Name:	Implementation of SJF Scheduling Algorithm
ID:	IT-17037

Objective:- To understand the Shortest Job First (SJF) scheduling algorithm technique in detail.

To simulate Shortest Job First (SJF) Scheduling Algorithm using C/C++.

To understand the advantage and disadvantage of Shortest Job First (SJF) scheduling algorithm technique.

Shortest Job First:- Shortest job first is a scheduling algorithm in which the process with the smallest execution time is selected for execution next. Shortest job first can be either preemptive or non-preemptive. Owing to its simple nature, shortest job first is considered optimal. It also reduces the average waiting time for other processes awaiting execution. Shortest job first is also known as shortest job next (SJN) and shortest process next (SPN).

Algorithm:

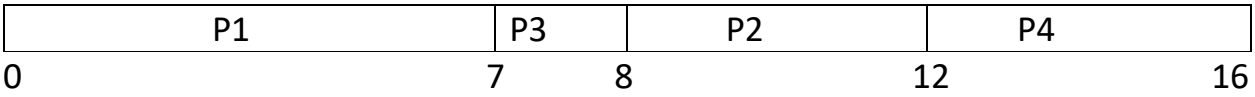
1. Sort all the process according to the arrival time.
2. Then select that process which has minimum arrival time and minimum Burst time.
3. After completion of process make a pool of process which after till the completion of previous process and select that process among the pool which is having minimum Burst time.

process	Arrival time	Burst time
P1	0	7
P2	2	4
P3	4	1
P4	5	4

1. Completion Time: Time at which process completes its execution.
2. Turn Around Time: Time Difference between completion time and arrival time. Turn Around Time = Completion Time – Arrival Time
3. Waiting Time (W.T): Time Difference between turn-around time and burst time.
   Waiting Time = Turn Around Time – Burst Time

In this post, we have assumed arrival times as 0, so turn around and completion times are same.

Gantt chart:-



process	Arrival time	Burst time	Completion time	Turn-around time	Waiting time
P1	0	7	7	7	0
P2	2	4	12	10	10
P3	4	1	8	4	3
P4	5	4	16	11	7

Average waiting time = (0+6+3+7)/4

=4ms

Average turn-around time = (7+10+8+11)/4

=8ms

Code:-

// C++ program to implement Shortest Job first with Arrival Time

#include<iostream>

```
using namespace std;
int mat[10][6];

void swap(int *a, int *b)
{
	int temp = *a;
	*a = *b;
	*b = temp;
}

void arrangeArrival(int num, int mat[][6])
{
	for(int i=0; i<num; i++)
	{
		for(int j=0; j<num-i-1; j++)
		{
			if(mat[j][1] > mat[j+1][1])
			{
				for(int k=0; k<5; k++)
				{
					swap(mat[j][k], mat[j+1][k]);
				}
			}
		}
```

```
        }
}

void completionTime(int num, int mat[][6])
{
        int temp, val;
        mat[0][3] = mat[0][1] + mat[0][2];
        mat[0][5] = mat[0][3] - mat[0][1];
        mat[0][4] = mat[0][5] - mat[0][2];

        for(int i=1; i<num; i++)
        {
                temp = mat[i-1][3];
                int low = mat[i][2];
                for(int j=i; j<num; j++)
                {
                        if(temp >= mat[j][1] && low >= mat[j][2])
                        {
                                low = mat[j][2];
                                val = j;
                        }
                }
                mat[val][3] = temp + mat[val][2];
                mat[val][5] = mat[val][3] - mat[val][1];
```

```
                mat[val][4] = mat[val][5] - mat[val][2];
                for(int k=0; k<6; k++)
                {
                        swap(mat[val][k], mat[i][k]);
                }
        }
}

int main()
{
        int num, temp;

        cout<<"Enter number of Process: ";
        cin>>num;

        cout<<"...Enter the process ID...\n";
        for(int i=0; i<num; i++)
        {
                cout<<"...Process "<<i+1<<"...\n";
                cout<<"Enter Process Id: ";
                cin>>mat[i][0];
                cout<<"Enter Arrival Time: ";
                cin>>mat[i][1];
                cout<<"Enter Burst Time: ";
```

```
            cin>>mat[i][2];
        }



        arrangeArrival(num, mat);
        completionTime(num, mat);


        cout<<"Process ID\tArrival Time\tBurst Time\tWaiting Time\tTurnaround Time\n";
        for(int i=0; i<num; i++)
        {

        cout<<mat[i][0]<<"\t\t"<<mat[i][1]<<"\t\t"<<mat[i][2]<<"\t\t"<<mat[i][4]<<"\t\t"<<mat[i][5]<<"\n";
        }
}
```

Output:-

```
Enter number of Process: 4
...Enter the process ID...
...Process 1...
Enter Process Id: 4
Enter Arrival Time: 5
Enter Burst Time: 4
...Process 2...
Enter Process Id: 3
Enter Arrival Time: 4
Enter Burst Time: 1
...Process 3...
Enter Process Id: 2
Enter Arrival Time: 2
Enter Burst Time: 4
...Process 4...
Enter Process Id: 1
Enter Arrival Time: 0
Enter Burst Time: 7
Process ID      Arrival Time    Burst Time      Waiting Time    Turnaround Time
1               0               7               0               7
3               4               1               3               4
4               5               4               3               7
2               2               4               10              14

Process returned 0 (0x0)   execution time : 47.147 s
Press any key to continue.
```

Conclusion:- SJF Is a good algorithm for the process that has a small execution time. Regarding the average and waiting time is pretty advantage then FCFS .This because the execution of small processes in start brings a little time waiting for long processes. One of the major disadvantage of this algorithm is: a process must wait in the queue if he has a large time of execution although he may be in the queue for a long time if on the queue come processes with short execution time [6]. Complexity is another of bugs of this algorithm which stands in the selection of the next process from the CPU.