Mawlana Bhashani Science and Technology University

Lab Report

Report No: 08

Course Code: ICT – 3110

Course title: Operating Systems Lab

Date of Performance:

Date of Submission: 12/09/2020

Submitted by

Name: Pritom Saha

ID: IT – 17010

3rd year 1st semester

Dept. of ICT

MBSTU

Submitted to

Nazrul Islam

Assistant Professor

Dept. of ICT

MBSTU

Experiment no: 08

Experiment name: Implementation of SJF Scheduling Algorithm.

Theory:

SJF stands for Shortest Job First. It is an algorithm which schedules policy that selects the waiting process with the smallest execution time to execute next.

Working process:

1. First of all, we have to sort all the process according to the arrival time.
2. Then, we have to select the process that 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.

We will use some calculations. Those are as follows –

1. Completion time: The required time at which process completes its execution.
2. Turnaround time: Turnaround time = Completion time – Arrival time.
3. Waiting time = Turnaround time – Burst time.

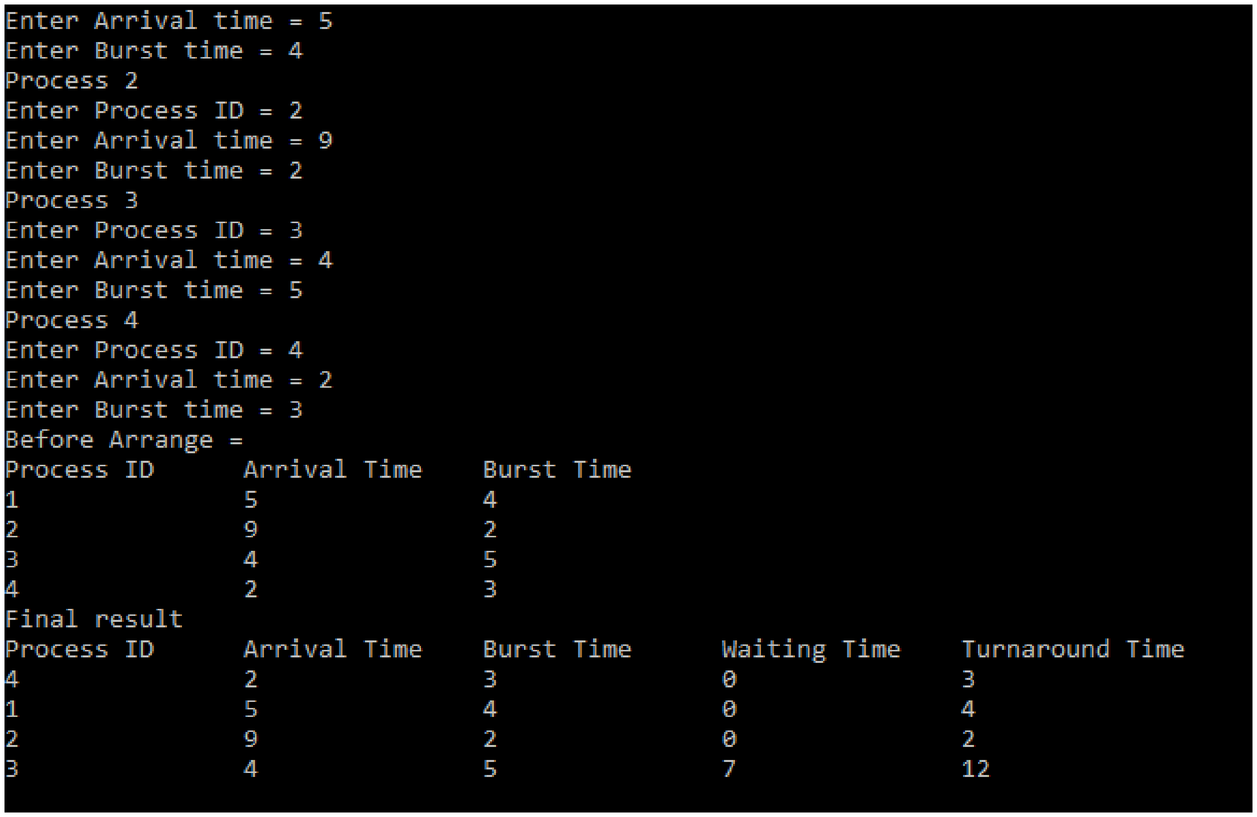
Code:

```
#include<bits/stdc++.h>

using namespace std;

int g[20][20];

signed main() {
```

```
ios :: sync_with_stdio(false);
cin.tie(0); cout.tie(0);
int n;
cout << "Enter the number of process = ";
cin >> n;
for (int i = 0; i < n; i++) {
    cout << "Process " << i + 1 << endl;
    cout << "Enter Process ID = ";
    cin >> g[i][0];
    cout << "Enter Arrival time = ";
    cin >> g[i][1];
    cout << "Enter Burst time = ";
    cin >> g[i][2];
}
cout << "Before Arrange = " << endl;
cout<<"Process ID\tArrival Time\tBurst Time\n";
for (int i = 0; i < n; i++) {
    cout << g[i][0] << "\t\t" << g[i][1] << "\t\t" << g[i][2] << endl;
}
for (int i = 0; i < n; i++) {
    for (int j = 0; j < n - i - 1; j++) {
        if (g[j][1] > g[j + 1][1]) {
            for (int k = 0; k < 5; k++) {
                swap(g[j][k], g[j + 1][k]);
            }
        }
    }
}
int temp, value;
g[0][3] = g[0][1] + g[0][2];
g[0][5] = g[0][3] - g[0][1];
g[0][4] = g[0][5] - g[0][2];
for (int i = 1; i < n; i++) {
    temp = g[i - 1][3];
    int low = g[i][2];
    for (int j = i; j < n; j++) {
        if (temp >= g[j][1] && low >= g[j][2]) {
            low = g[j][2];
            value = j;
        }
    }
    g[value][3] = temp + g[value][2];
    g[value][5] = g[value][3] - g[value][1];
    g[value][4] = g[value][5] - g[value][2];
    for (int k = 0; k < 6; k++) {
        swap(g[value][k], g[i][k]);
```

```
                }
        }
        cout << "Final result " << endl;
        cout << "Process ID\tArrival Time\tBurst Time\tWaiting Time\tTurnaround Time" << endl;
        for (int i = 0; i < n; i++) {
                cout << g[i][0] << "\t\t" << g[i][1] << "\t\t" << g[i][2] << "\t\t" << g[i][4] << "\t\t" << g[i][5] << endl;
        }
        return 0;
}
```

Output:



```
Enter Arrival time = 5
Enter Burst time = 4
Process 2
Enter Process ID = 2
Enter Arrival time = 9
Enter Burst time = 2
Process 3
Enter Process ID = 3
Enter Arrival time = 4
Enter Burst time = 5
Process 4
Enter Process ID = 4
Enter Arrival time = 2
Enter Burst time = 3
Before Arrange =
Process ID      Arrival Time    Burst Time
1               5               4
2               9               2
3               4               5
4               2               3
Final result
Process ID      Arrival Time    Burst Time      Waiting Time    Turnaround Time
4               2               3               0               3
1               5               4               0               4
2               9               2               0               2
3               4               5               7               12
```

<u>Discussion</u>:

SJF is a greedy algorithm. It has the advantage of having a minimum average waiting time along all scheduling algorithm. It may cause starvation if shorter processes keep coming.