Report on simulation based project

Submitted to: Keshav Dhir

Course Code: CSE 316 **Course Title: Operating System**

Submitted by Ankit Biswal

Registration Number: 11802036

Roll Number: A17

Date of Submission: 29/03/2020

Name: Ankit Biswal **Registration Number**: 11802036

Email Address: ankitbiswal123@gmail.com

GitHub Link:

Description:

To solve the problem LJF (Longest Job First) scheduling algorithm is applied. Longest Job Scheduling Algorithm keeps track of the Burst time of all the available processes at the arrival time itself and then assigns the processor to that process which has the longest burst time. It a type of non-pre-emptive scheduling algorithm where once a process starts its execution, it cannot be interrupted in between its processing and any other process can be executed only after the assigned process has completed its processing and has been terminated.

It is like SJF scheduling algorithm. But, in this scheduling algorithm, we give priority to the process having the longest burst time. This is non-pre-emptive in nature i.e., when any process starts executing, can't be interrupted before complete execution.

The LJF scheduling algorithm is just as the SJF. The only difference is that The SJF scheduling algorithm executes the processes with the shortest burst time first, whereas, in LJF, the processes with the longest burst time are executed first.

Code:

```c
#include<stdio.h>

void main()
{
    /*  at = Arrival Time       bt = Burst Time     p = Process
        wt = Waiting Time       tat = Turnaround Time
        avg_wt = Average Waiting Time     avg_tat = Avrage Turnaround Time      */

    int at[20],bt[20],p[20],wt[20],tat[20],i,j,n,total=0,pos,temp;
    float avg_wt,avg_tat;

    printf("Enter number of process: ");
    scanf("%d",&n);

    printf("\nEnter Arrival Time: \n");
    for(i=0;i<n;i++)
    {
        printf("p%d: ",i+1);
        scanf("%d",&at[i]);

        p[i]=i+1;           //contains process number
        bt[i] = 2 * at[i];      // Calculate burst time from arrival time.
    }


    //sorting burst time in decending order using selection sort
    for(i=0;i<n;i++)
    {
        pos=i;
        for(j=i+1;j<n;j++)
        {
            if(bt[i]<bt[j] && p[i]<p[j] && at[i]<at[j])
            {
                temp=at[i];
                at[i]=at[j];
                at[j]=temp;

                temp=bt[i];
                bt[i]=bt[j];
                bt[j]=temp;

                temp=p[i];
                p[i]=p[j];
                p[j]=temp;
            }
        }

    }
```

```
49
50      wt[0]=0;    //waiting time for first process will be zero
51
52      //calculate waiting time
53      for(i=1;i<n;i++)
54      {
55          wt[i]=0;
56          for(j=0;j<i;j++)
57              wt[i]+=bt[j];
58
59          total+=wt[i];
60      }
61
62      avg_wt=(float)total/n;      //average waiting time
63      total=0;
64
65      printf("\nProcess \t Arrival time  \t Burst Time \tWaiting Time \t Turnaround Time");
66
67      for(i=0;i<n;i++)
68      {
69          tat[i]=bt[i]+wt[i];     //calculate turnaround time
70          total+=tat[i];
71          printf("\n p%d \t\t %d \t\t %d \t\t %d \t\t %d",p[i],at[i],bt[i],wt[i],tat[i]);
72      }
73
74      avg_tat=(float)total/n;     //average turnaround time
75
76      printf("\n\nAverage Waiting Time = %f",avg_wt);
77      printf("\nAverage Turnaround Time = %f\n",avg_tat);
78  }
```

Algorithm:

1. Sort all the process according to the arrival time.
2. Then select that process which has maximum arrival time and maximum 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 maximum Burst time.

Boundary Conditions:

➔ Burst Time = 2 * Arrival Time.

```
21      for(i=0;i<n;i++)
22      {
23          printf("p%d: ",i+1);
24          scanf("%d",&at[i]);
25          p[i]=i+1;           //contains process number
26          bt[i] = 2 * at[i];      // Calculate burst time from arrival time.
27      }
```

➔ Scheduler selects the process with largest burst time from the queue for the execution.

Sort the burst and arrival time in descending order for given arrival time.

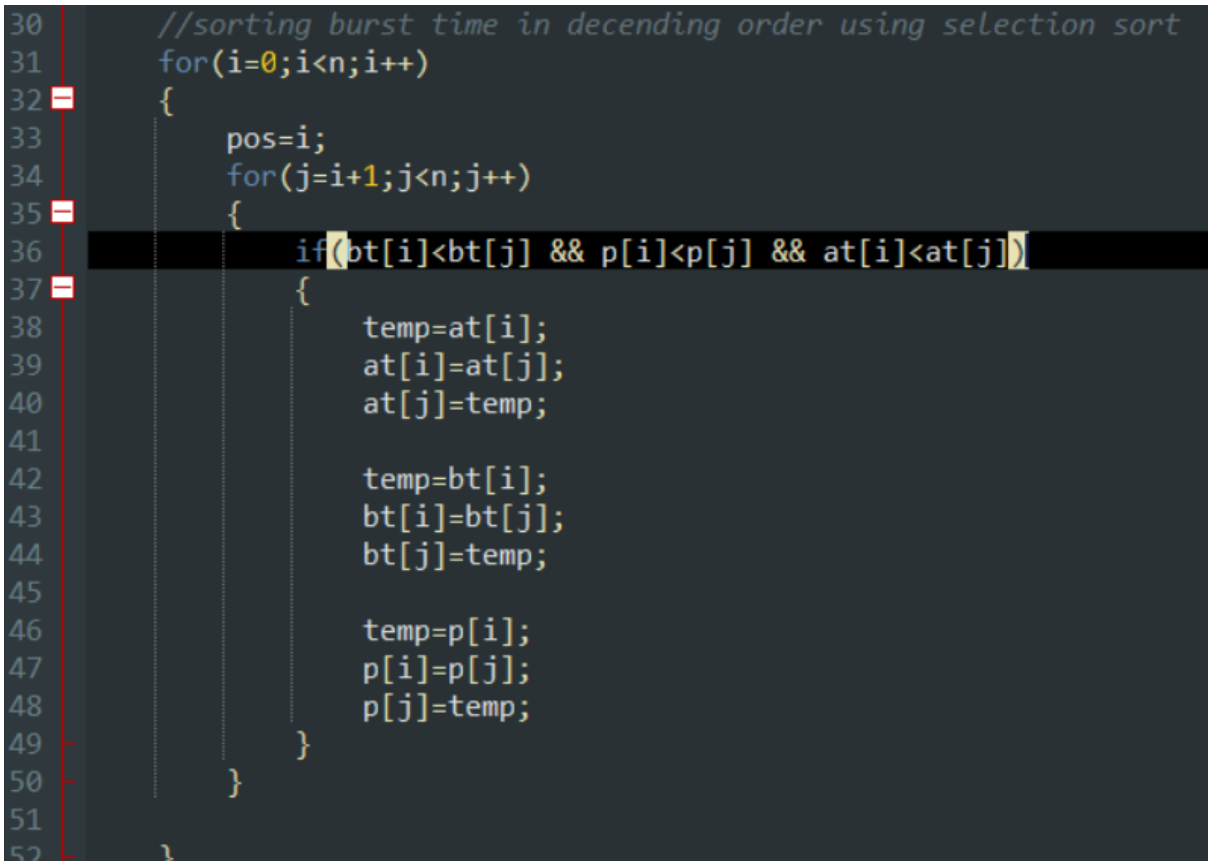
```
30      //sorting burst time in decending order using selection sort
31      for(i=0;i<n;i++)
32      {
33          pos=i;
34          for(j=i+1;j<n;j++)
35          {
36              if(bt[i]<bt[j] && p[i]<p[j] && at[i]<at[j])
37              {
38                  temp=at[i];
39                  at[i]=at[j];
40                  at[j]=temp;
41
42                  temp=bt[i];
43                  bt[i]=bt[j];
44                  bt[j]=temp;
45
46                  temp=p[i];
47                  p[i]=p[j];
48                  p[j]=temp;
49              }
50          }
51
52      }
```

➔ Compute the average waiting time and average turnaround time.

```
54      wt[0]=0;    //waiting time for first process will be zero
55
56      //calculate waiting time
57      for(i=1;i<n;i++)
58      {
59          wt[i]=0;
60          for(j=0;j<i;j++)
61              wt[i]+=bt[j];
62
63          total+=wt[i];
64      }
65
66      avg_wt=(float)total/n;      //average waiting time
67      total=0;
68
69      printf("\nProcess \t Arrival time  \t Burst Time \tWaiting Time \t Turnaround Time");
70
71      for(i=0;i<n;i++)
72      {
73          tat[i]=bt[i]+wt[i];     //calculate turnaround time
74          total+=tat[i];
75          printf("\n p%d \t\t %d \t\t %d \t\t %d \t\t %d",p[i],at[i],bt[i],wt[i],tat[i]);
76      }
77
78      avg_tat=(float)total/n;     //average turnaround time
```

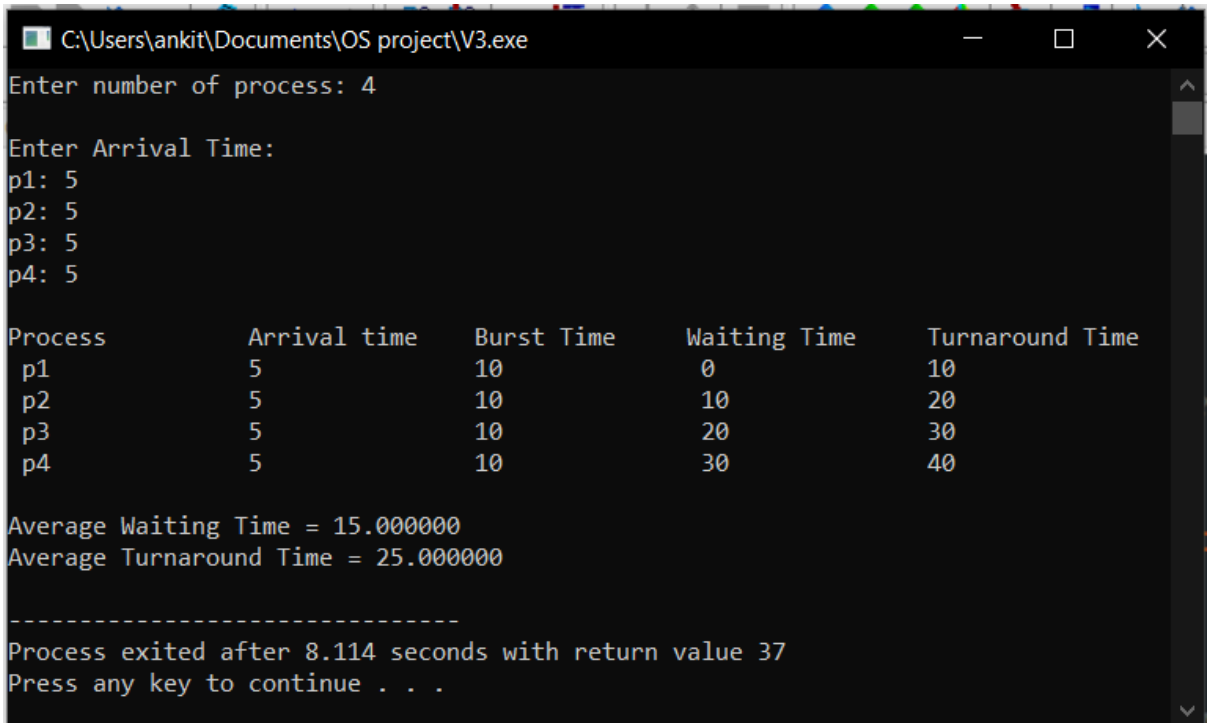
Complexity of Algorithm:

$O(n^2)$, as the code having highest complexity was the selection sort used to sort the arrival time, burst time and process in descending order.



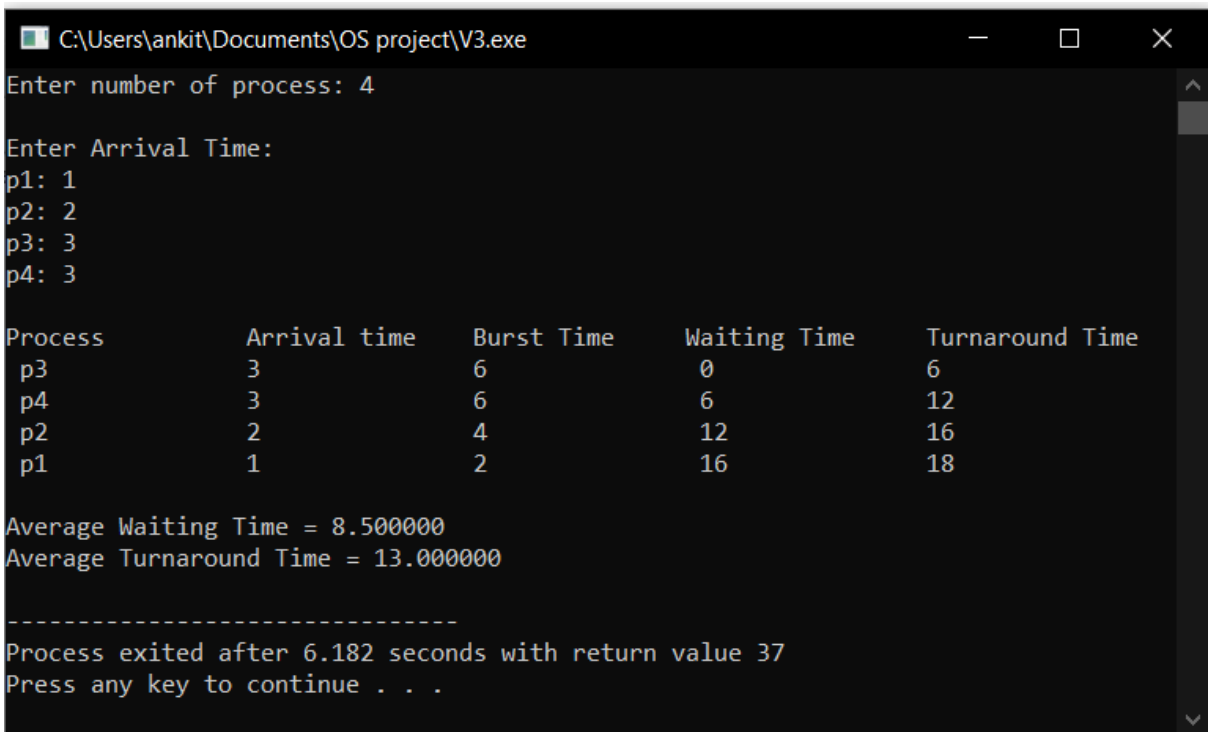
```
30      //sorting burst time in decending order using selection sort
31      for(i=0;i<n;i++)
32      {
33          pos=i;
34          for(j=i+1;j<n;j++)
35          {
36              if(bt[i]<bt[j] && p[i]<p[j] && at[i]<at[j])
37              {
38                  temp=at[i];
39                  at[i]=at[j];
40                  at[j]=temp;
41
42                  temp=bt[i];
43                  bt[i]=bt[j];
44                  bt[j]=temp;
45
46                  temp=p[i];
47                  p[i]=p[j];
48                  p[j]=temp;
49              }
50          }
51
52      }
```

Test Cases:



In this case all the processes are having same arrival times, hence their burst times are equal, as burst time is double of arrival time as per the problem given.

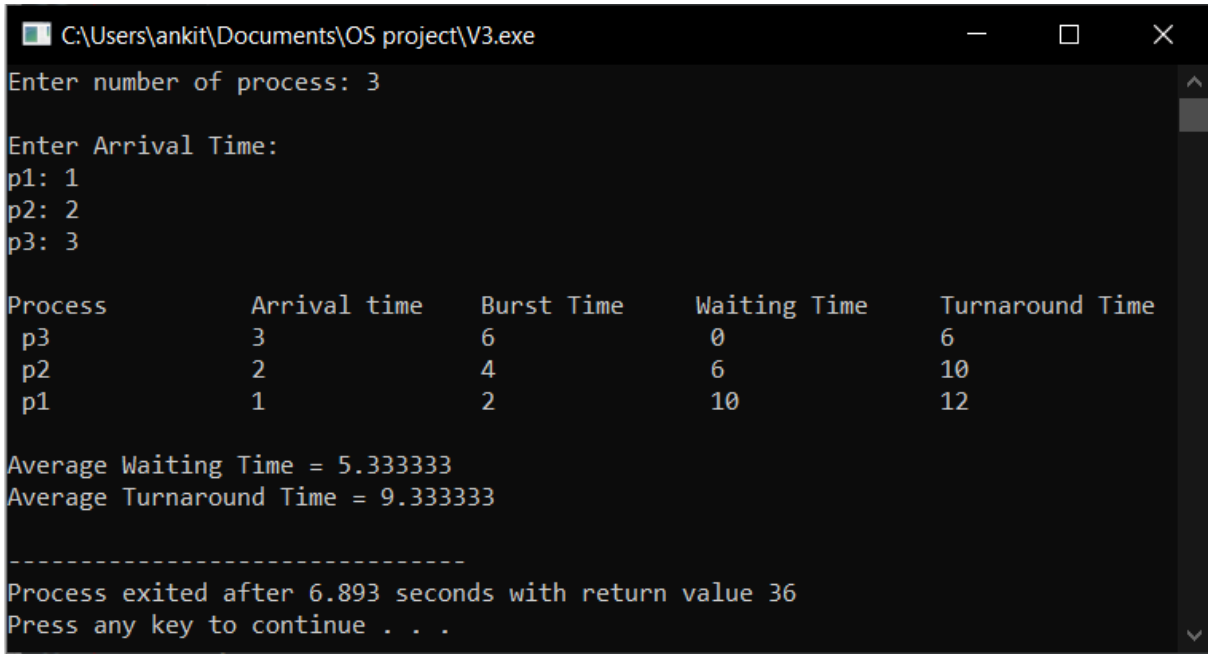
So, the execution of processes occurs according to the number assigned to the process i.e. from lowest to highest.

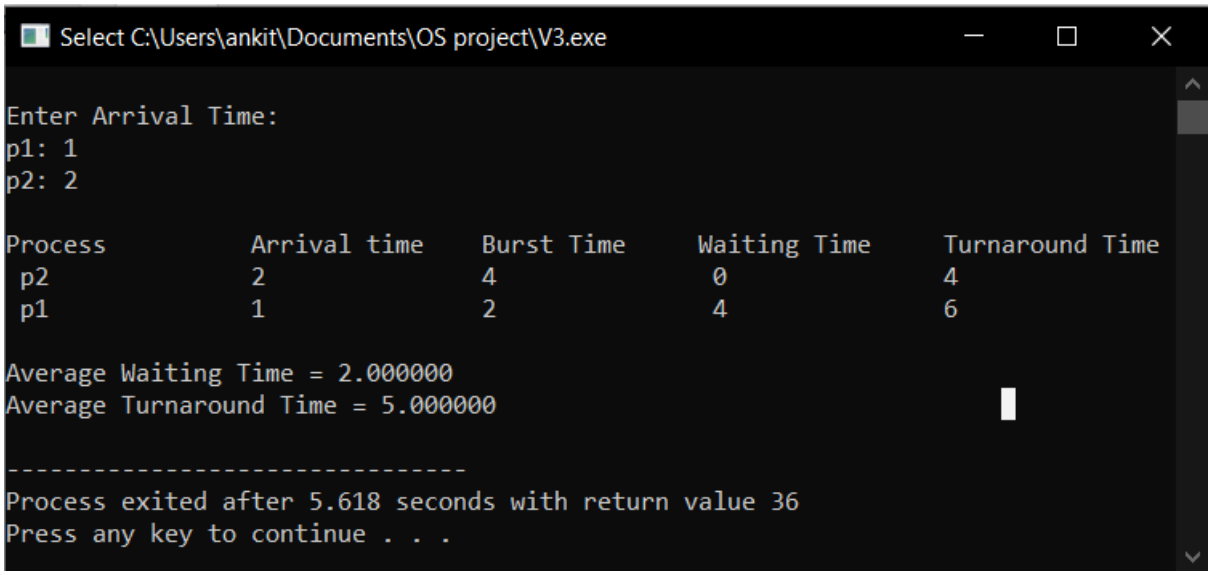


In this case all the processes are not equal except p3 and p4.

The process having largest burst time will occur first highest i.e. from highest to lowest.

But the execution of processes having same burst time will occurs according to the number assigned to the process i.e. from lowest to highest.





In above two cases all the processes are having different burst times, so the execution will occur first highest i.e. from highest to lowest.

The average waiting time if processes are executed according to Shortest Job First scheduling approach with the same attribute values.

```
Enter number of process: 2

Enter Arrival Time:
p1: 1
p2: 2

Process          Arrival time    Burst Time     Waiting Time     Turnaround Time
 p1              1               2               0               2
 p2              2               4               2               6

Average Waiting Time = 1.000000
Average Turnaround Time = 4.000000
```

```
Enter number of process: 3

Enter Arrival Time:
p1: 1
p2: 2
p3: 3

Process          Arrival time    Burst Time     Waiting Time     Turnaround Time
 p1              1               2               0               2
 p2              2               4               2               6
 p3              3               6               6               12

Average Waiting Time = 2.666667
Average Turnaround Time = 6.666667
```

```
Enter number of process: 4

Enter Arrival Time:
p1: 1
p2: 2
p3: 3
p4: 3

Process          Arrival time    Burst Time     Waiting Time     Turnaround Time
 p1              1               2               0               2
 p2              2               4               2               6
 p3              3               6               6               12
 p4              3               6               12              18

Average Waiting Time = 5.000000
Average Turnaround Time = 9.500000
```

```
Enter number of process: 4

Enter Arrival Time:
p1: 5
p2: 5
p3: 5
p4: 5

Process          Arrival time    Burst Time     Waiting Time     Turnaround Time
 p1              5               10              0               10
 p2              5               10              10              20
 p3              5               10              20              30
 p4              5               10              30              40

Average Waiting Time = 15.000000
Average Turnaround Time = 25.000000
```