OPERATING SYSTEMS PROJECT REPORT

Submitted By :

Name : Patel Dev Prakashbhai

Reg. No : 11801970

Section : K18EN

Roll No : B26

Group : 2

Email : dppatel4565@gmail.com

GitHub Link : https://github.com/dppatel522/OS-Project

Submitted To :

Priya Virdi

INDEX

QUESTION STATEMENT

Que 4: Consider a scheduling approach which is non pre-emptive similar to shortest job next in nature. The priority of each job is dependent on its estimated run time, and also the amount of time it has spent waiting. Jobs gain higher priority the longer they wait, which prevents indefinite postponement. The jobs that have spent a long time waiting compete against those estimated to have short run times.

The priority can be computed as:
Priority = 1+ Waiting time / Estimated run time

Using the data given below compute the waiting time and turnaround time for each process and average waiting time and average turnaround time.

Process	Arrival time	Burst time
P1	0	20
P2	5	36
P3	13	19
P4	17	42

DESCRIPTION

- Non-preemptive Scheduling is used when a process terminates, or a process switches from running to waiting state.
- In this scheduling, once the resources is allocated to a process, the process holds the CPU till it gets terminated or it reaches a waiting state.
- In case of non-preemptive scheduling does not interrupt a process running CPU in middle of the execution. Instead, it waits till the process complete its CPU burst time and then it can allocate the CPU to another process.
- Algorithms based on non-preemptive scheduling are: Shortest Job First (SJF basically non preemptive) and Priority (non-preemptive version)

CODE

```
#include<stdio.h>

#include<conio.h>

int main()

{

  printf("\t\t\t---------------------- Scheduling ----------------------\n\n\n\n");

  long int n,i=0,j=0;

  printf("Enter Number of Processes : ");

  scanf("%ld",&n );

  double
priority[n],avg_waiting,avg_turnaround,burstTime[n],arrivalTime[n],waitingTime[n],turnaround
Time[n], process[n], temp, completionTime[n],min,sum=0,sum2=0,wait_final, turnaround_final,
wait_avg, turnaround_avg;

  for(i=0;i<n;i++)

  {

    printf("\nEnter Burst Time for Process [%d] : ", i+1 );

    scanf("%lf", &burstTime[i]);

    printf("Enter Arrival Time for Process [%d] : ", i+1 );

    scanf("%lf", &arrivalTime[i] );

    process[i]=i+1;

  }

  printf("\n\n\t\t\t -------------- Entered Values are --------------\n\n");

  printf("\t\t\t---------------------------------------\n");

  printf("\t\t\t| Process | Arrival Time | Burst Time |\n");

  printf("\t\t\t---------------------------------------\n");

  for(i=0;i<n;i++)
```

```
 {
  printf("\t\t\t|  P[%0.0lf]   |      %0.0lf     |     %0.0lf
|\n",process[i],arrivalTime[i],burstTime[i]);
 }
  printf("\t\t\t--------------------------------------\n");
 printf("\n\n\t\t\t-------- Sorting Processes according to Arrivaltime --------\n");
 for(i=0;i<n;i++)
 {
  for(j=0;j<n;j++)
  {
    if(arrivalTime[i]<arrivalTime[j])
    {
      temp = burstTime[j];
      burstTime[j] = burstTime[i];
      burstTime [i] = temp;
            temp = process[j];
      process[j] = process[i];
      process[i] = temp;
            temp = arrivalTime[j];
      arrivalTime[j] = arrivalTime[i];
      arrivalTime[i] = temp;
    }} }
 printf("\n\n\t\t\t -------------- Now Values are --------------\n\n");
 printf("\t\t\t--------------------------------------\n");
 printf("\t\t\t| Process | Arrival Time | Burst Time |\n");
 printf("\t\t\t--------------------------------------\n");
 for(i=0;i<n;i++)
 {
   printf("\t\t\t|  P[%0.0lf]   |      %0.0lf     |     %0.0lf
|\n",process[i],arrivalTime[i],burstTime[i]);
 }
   printf("\t\t\t---------------------------------------\n");
 long int k = 1;
 double b_time = 0;
 for(j=0;j<n;j++)
 {
   b_time = b_time + burstTime[j];
   min = burstTime[k];
   for(i=k;i<n;i++)
   {
     if((b_time >= arrivalTime[i])&&(burstTime[i]<min))
     {
       temp = burstTime[k];
       burstTime[k] = burstTime[i];
       burstTime[i] = temp;
       temp = arrivalTime[k];
       arrivalTime[k] = arrivalTime[i];
       arrivalTime[i] = temp;
       temp = process[k];
       process[k] = process[i];
       process[i] = temp;
     }}
   k++;     }
 waitingTime[0] = 0;
 for(i=1;i<n;i++)
 {
   sum += burstTime[i-1];
   waitingTime[i] = sum - arrivalTime[i];
   wait_final += waitingTime[i];
 }
 wait_avg = wait_final/n;
 for(i=0;i<n;i++)
 {
   sum2 += burstTime[i];
   turnaroundTime[i] = sum2 - arrivalTime[i];
   turnaround_final += turnaroundTime[i];
 }
 turnaround_avg=turnaround_final/n;
printf("\n\n\t\t\t -------------- Now Values are --------------\n\n");
 printf("\t\t\t---------------------------------------------------------------------------\n");
 printf("\t\t\t| Process | Arrival Time | Burst Time |  Waiting Time  |  Turn Around Time  |\n");
 printf("\t\t\t---------------------------------------------------------------------------\n");
 for(i=0;i<n;i++)
 {
   printf("\t\t\t|  P[%0.0lf]   |      %0.0lf      |    %0.0lf     |       %0.0lf       |        %0.0lf
|\n",process[i],arrivalTime[i],burstTime[i],waitingTime[i],turnaroundTime[i]);
 }
   printf("\t\t\t---------------------------------------------------------------------------\n");
 completionTime[0] = burstTime[0];
 for(i=1;i<n;i++)
 {
   completionTime[i] = completionTime[i-1] + burstTime[i];
  }
  for(i=0;i<n;i++)
  {
   priority[i] = 1+waitingTime[i]/completionTime[i];
   printf("%lf\n",priority[i]);
  }
  printf("\n\n\t\t\t -------------- Final Values are --------------\n\n");
  printf("\t\t\t----------------------------------------------------------------------------\n");
  printf("\t\t\t| Process | Arrival Time | Burst Time |  Waiting Time  |  Turn Around Time  |\n");
  printf("\t\t\t----------------------------------------------------------------------------\n");
  printf("\t\t\t|  P[%0.0lf]   |      %0.0lf      |     %0.0lf      |       %0.0lf       |        %0.0lf
|\n",process[0],arrivalTime[0],burstTime[0],waitingTime[0],turnaroundTime[0]);
  for(i=n-1;i>0;i--)
  {
   printf("\t\t\t|  P[%0.0lf]   |      %0.0lf      |     %0.0lf      |       %0.0lf       |        %0.0lf
|\n",process[i],arrivalTime[i],burstTime[i],waitingTime[i],turnaroundTime[i]);
  }
   printf("\t\t\t----------------------------------------------------------------------------\n");
  printf("\n\n\n\t\t\tAverage Turn Around Time : %lf",turnaround_avg);
  printf("\n\t\t\tAverage Waiting Time     : %lf\n\n",wait_avg)
  getch();
  return 0;
}
```

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.

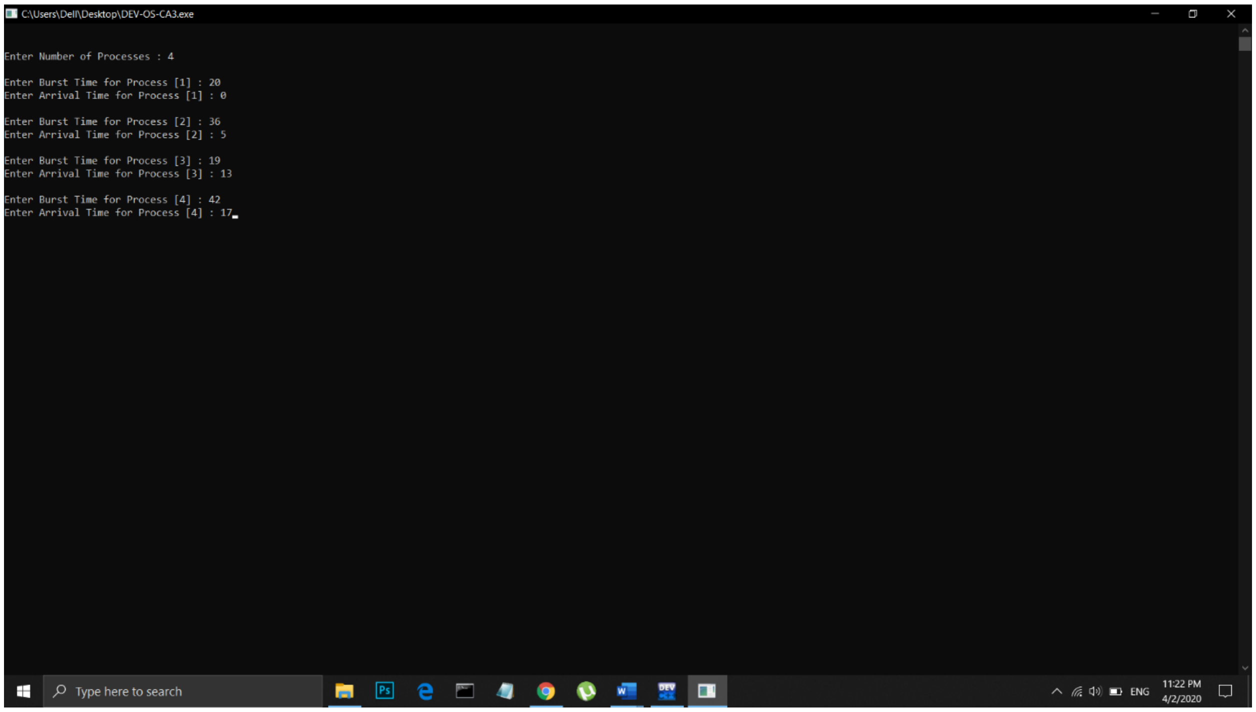
COMPLIEXITY

- Complexity of this algorithm is **O(nlogn).**

Activity On GitHub

- I have uploaded the project on the GitHub. And I have also done the revisions of the project on the GitHub as my project is going on.
- I have made more than 6 revisions on that project and also uploaded the documented description file with it.

SNAPSHOTS



C:\Users\Dell\Desktop\DEV-OS-CA3.exe

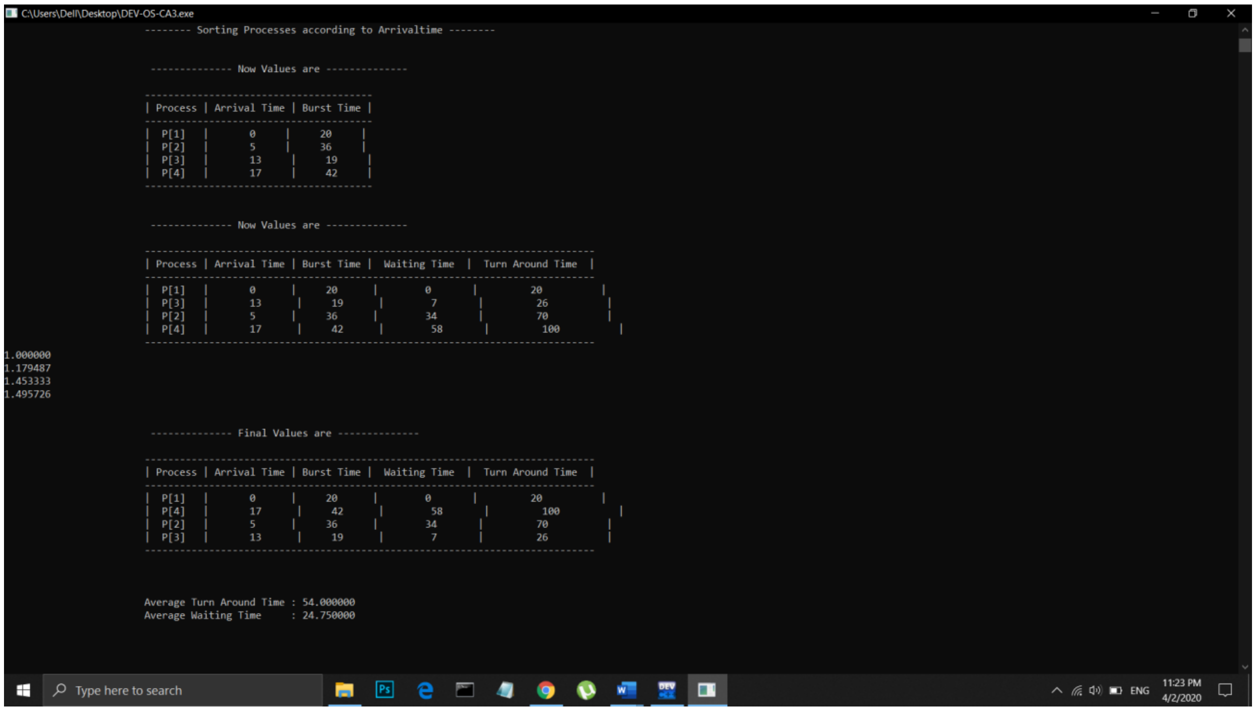
```
Enter Number of Processes : 4

Enter Burst Time for Process [1] : 20
Enter Arrival Time for Process [1] : 0

Enter Burst Time for Process [2] : 36
Enter Arrival Time for Process [2] : 5

Enter Burst Time for Process [3] : 19
Enter Arrival Time for Process [3] : 13

Enter Burst Time for Process [4] : 42
Enter Arrival Time for Process [4] : 17_
```



C:\Users\Dell\Desktop\DEV-OS-CA3.exe

```
                        -------- Sorting Processes according to Arrivaltime --------


                         ------------- Now Values are -------------

                        --------------------------------------
                        | Process | Arrival Time | Burst Time |
                        --------------------------------------
                        |  P[1]   |       0     |     20     |
                        |  P[2]   |       5     |     36     |
                        |  P[3]   |       13     |     19     |
                        |  P[4]   |       17     |     42     |
                        --------------------------------------


                         ------------- Now Values are -------------

                        -----------------------------------------------------------------------------
                        | Process | Arrival Time | Burst Time |  Waiting Time  |  Turn Around Time  |
                        -----------------------------------------------------------------------------
                        |  P[1]   |       0     |     20     |       0       |        20          |
                        |  P[3]   |       13     |     19     |       7       |        26          |
                        |  P[2]   |       5     |     36     |       34      |        70          |
                        |  P[4]   |       17     |     42     |       58       |        100          |
                        -----------------------------------------------------------------------------
1.000000
1.179487
1.453333
1.495726



                         ------------- Final Values are -------------

                        -----------------------------------------------------------------------------
                        | Process | Arrival Time | Burst Time |  Waiting Time  |  Turn Around Time  |
                        -----------------------------------------------------------------------------
                        |  P[1]   |       0     |     20     |       0       |        20          |
                        |  P[4]   |       17     |     42     |       58       |        100          |
                        |  P[2]   |       5     |     36     |       34      |        70          |
                        |  P[3]   |       13     |     19     |       7       |        26          |
                        -----------------------------------------------------------------------------



                        Average Turn Around Time : 54.000000
                        Average Waiting Time     : 24.750000
```