

**Mawlana Bhashani Science and Technology University**

**Lab-Report**

Report No:08

Course code: ICT-3110

Course title: Operating System Lab

Date of Performance:01/09/2020

Date of Submission:10/09/2020

**Submitted by Submitted To**

Name:Md Milon Ahmed

ID:IT-18034

3rd  year 1st semester

Session: 2017-2018

Dept. of ICT

MBSTU.

Nazrul Islam

Assistant Professor

Dept. of ICT

MBSTU.

# Experiment no : 08

# Experiment Name : Implementation of SJF Scheduling Algorithm

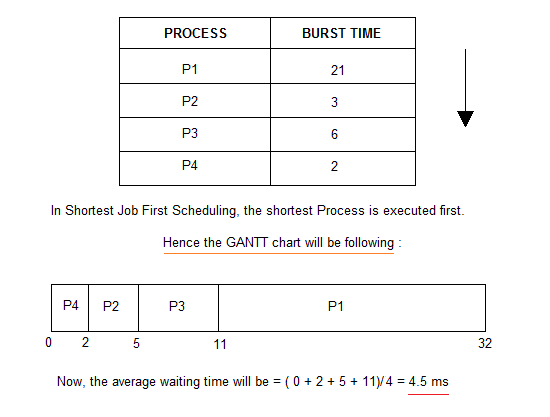
# Theory:

SJF is a scheduling algorithm that assigns to each process the length of its next CPU burst/execution time. CPU is then given to the process with the minimal CPU burst from the waiting queue. SJF is provably optimal, in that for a given set of processes and their CPU bursts/execution times it gives the least average waiting time for each process. The average waiting time for a process is defined by:

S=(W1+W2+...+Wn)/n [1], where Wk=Wk-1+tk-1 [2] is the waiting time for a kth process and ti is the execution time/length of next CPU burst of the ith process; 1<= k, i <=n (actually, the execution time of the last process in the queue, tn, does not affect any waiting times), and W0=0.

If we replace [2] into [1], we get: WS=((n-1)t1+(n-2)t2+...+(n-k)tk+... +tn-1)/n [3]

Now let us suppose that we have an arbitrary set of *n* CPU bursts, { t1, t2, ... , tn }.



# Working Procedure:

Coding implementation with c…

#include<stdio.h>

void main()

{

    int 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 Burst Time:\n");

    for(i=0;i<n;i++)

    {

        printf("p%d:",i+1);

        scanf("%d",&bt[i]);

        p[i]=i+1;           //contains process number

    }

    //sorting burst time in ascending order using selection sort

    for(i=0;i<n;i++)

    {

        pos=i;

        for(j=i+1;j<n;j++)

        {

            if(bt[j]<bt[pos])

                pos=j;

        }

        temp=bt[i];

        bt[i]=bt[pos];

        bt[pos]=temp;

        temp=p[i];

        p[i]=p[pos];

        p[pos]=temp;

    }

    wt[0]=0;            //waiting time for first process will be zero

    //calculate waiting time

    for(i=1;i<n;i++)

    {

        wt[i]=0;

        for(j=0;j<i;j++)

            wt[i]+=bt[j];

        total+=wt[i];

    }

    avg\_wt=(float)total/n;      //average waiting time

    total=0;

    printf("\nProcess\t    Burst Time    \tWaiting Time\tTurnaround Time");

    for(i=0;i<n;i++)

    {

        tat[i]=bt[i]+wt[i];     //calculate turnaround time

        total+=tat[i];

        printf("\np%d\t\t  %d\t\t    %d\t\t\t%d",p[i],bt[i],wt[i],tat[i]);

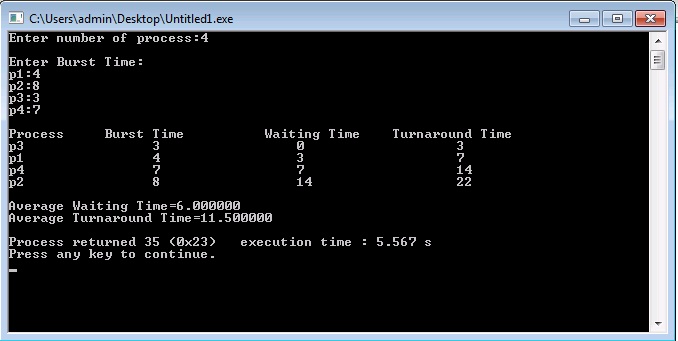
    }

    avg\_tat=(float)total/n;     //average turnaround time

    printf("\n\nAverage Waiting Time=%f",avg\_wt);

   printf("\nAverage Turnaround Time=%f\n",avg\_tat);

}

Output: 

# Discussion:

1.SJF method gives the lowest average waiting time for a specific set of processes.

2.It is appropriate for the jobs running in batch, where run times are known in advance.

3.For the batch system of long-term scheduling, a burst time estimate can be obtained from the job description.