Practical No 5

Aim : Develop, debug and Execute a C program to simulate the SJF CPU scheduling algorithms to find turnaround time and waiting time.

Apparatus: Computer system with windows installed in it.

Mingw compiler for C/C++, and a text editor for developing C code file (Dev C++).

Theory :

What is SJF scheduling?

- SJF is short for 'Shortest Job First' Scheduling algorithm.
- This algorithm associates with each process the length of the process's next CPU burst.
- In this scheduling scheme, when the CPU is available it is assigned to the process that has the smallest next CPU burst.
- This scheduling method reduces the average waiting time for other
- A more appropriate term for this scheduling method would be the shortest-next-CPU-burst algorithm, because scheduling depends on the length of the next CPU burst of a process rather than its total length.
- In preemptive approach, the new process arises when there is already executing process. If the burst of newly arriving process is less than that of the currently executing process, then the scheduler will prompt the execution of the process with lesser burst time.
- The downside of this algorithm is that it may cause starvation if shorter processes keep coming, this problem can be solved by using the concept of aging.

Example :

Process	Arrival Time	Burst time
P1	**0**	**8**
P2	**1**	**4**
P3	**2**	**9**
P4	**3**	**5**

Non Preemptive way: Shortest job first algorithm

P1 arrival time = 0

P2 arrival time = 1

P3 arrival time = 2

P4 arrival time = 3

P1 Burst time = 8

P2 Burst time = 4

P3 Burst time = 9

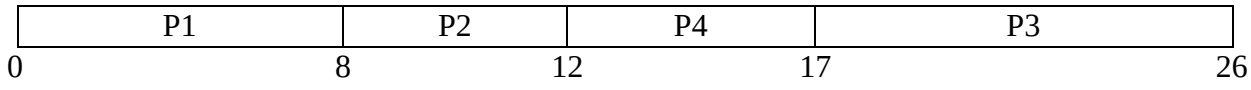
P4 Burst time = 5

According to the algorithm the OS schedules the shortest time remaining task first, so at first there is only one process present that is P1 with a burst time of 8ms. So it will get executed first.

After P1 is executed the process with the shortest burst time will be selected, i.e process P2 will get selected as it has the burst time of 4ms, the lowest of the available processes and process P2 will get executed.

After P2 is executed the process with the shortest burst time will be selected, i.e process P4 will get selected as it has the burst time has 5ms, the lowest of the available processes and process P4 will get executed.

After P4 is executed the last remaining process will be selected, i.e process P3 with the burst time of 9ms will get selected and it will get executed.



Average waiting time = (0 + (8-1) + (12-3) + (17 – 2))/4

= (0 + 7 + 9 + 15) / 4

= 31/4

= 7.75

Code :

```
#include<stdio.h>
int main()
{
    int i,n,process[10]={1,2,3,4,5,6,7,8,9,10},min,k=1,burstTime=0;
    int bt[10],temp,j,at[10],wt[10],tt[10],ta=0,sum=0;
    float wavg=0,tavg=0,tsum=0,wsum=0;
    printf(" -------Shortest Job First Scheduling ( NonPreemptive )-------\n");
    printf("\nEnter the No. of processes :");
    scanf("%d",&n);

    for(i=0;i<n;i++)
    {
        printf("\tEnter the burst time of process %d :",i+1);
        scanf(" %d",&bt[i]);
        printf("\tEnter the arrival time of process %d :",i+1);
        scanf(" %d",&at[i]);
    }

/*Sorting the tasks according to Arrival Time*/

    for(i=0;i<n;i++)
    {
        for(j=0;j<n;j++)
        {
            if(at[i]<at[j])
            {
                temp=process[j];
                process[j]=process[i];
                process[i]=temp;
                temp=at[j];
                at[j]=at[i];
                at[i]=temp;
                temp=bt[j];
                bt[j]=bt[i];
                bt[i]=temp;
            }
        }
    }
```

```
for(j=0;j<n;j++)
{
    burstTime=burstTime+bt[j];
    min=bt[k];
    for(i=k;i<n;i++)
    {
        if (burstTime>=at[i] && bt[i]<min)
        {
            temp=process[k];
            process[k]=process[i];
            process[i]=temp;
            temp=at[k];
            at[k]=at[i];
            at[i]=temp;
            temp=bt[k];
            bt[k]=bt[i];
            bt[i]=temp;
        }
    }
    k++;
}
wt[0]=0;
for(i=1;i<n;i++)
{
    sum=sum+bt[i-1];
    wt[i]=sum-at[i];
    wsum=wsum+wt[i];
}

wavg=(wsum/n);
for(i=0;i<n;i++)
{
    ta=ta+bt[i];
    tt[i]=ta-at[i];
    tsum=tsum+tt[i];
}

tavg=(tsum/n);

printf("\n");
printf("\n RESULT:-");
printf("\nProcess\t Burst\t Arrival\t Waiting\t Turn-around" );
```

```
 for(i=0;i<n;i++)
    {
        printf("\n process%d\t %d\t %d\t\t %d\t\t\t%d" ,process[i] ,bt[i] ,at[i]
,wt[i],tt[i]);
    }

    printf("\n\nAVERAGE WAITING TIME : %f",wavg);
    printf("\nAVERAGE TURN AROUND TIME : %f",tavg);
    return 0;
}
```

Output:

```
D:\coding\os4.exe
 -------Shortest Job First Scheduling ( NP )-------

Enter the No. of processes :4
Enter the burst time of 1 process :8
Enter the arrival time of 1 process :0
Enter the burst time of 2 process :4
Enter the arrival time of 2 process :1
Enter the burst time of 3 process :9
Enter the arrival time of 3 process :2
Enter the burst time of 4 process :5
Enter the arrival time of 4 process :3

 RESULT:-
Process         Burst   Arrival         Waiting         Turn-around
 process1       8       0               0                       8
 process2       4       1               7                       11
 process4       5       3               9                       14
 process3       9       2               15                      24

AVERAGE WAITING TIME : 7.75
AVERAGE TURN AROUND TIME : 14.25
--------------------------------
Process exited after 6.464 seconds with return value 0
Press any key to continue . . .
```

Conclusion:

Hence, by performing this practical I got to know about the concept of Shortest job first scheduler algorithm, its advantages, disadvantages, its use, and its implementation. I also developed, debugged and executed a c program to simulate the SJF (nonpreemprtive) CPU scheduling algorithms to find turnaround time and waiting time.