**EXERCISE-1**

Implement CPU Scheduling Algorithms: **Shortest Job First**.

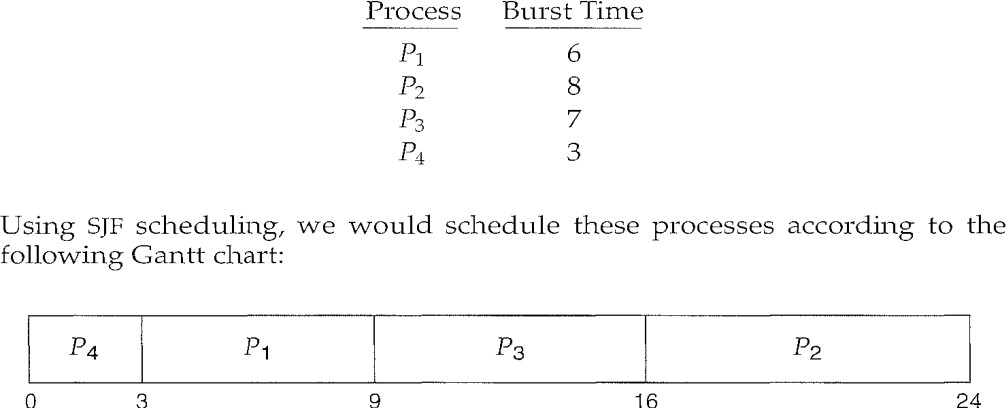
**AIM:**

To Implement CPU Scheduling Algorithms: **Shortest Job First**.

**DESCRIPTION:**

## SJF CPU scheduling algorithm:

* A different approach to CPU scheduling is the shortest-job-first (SJF) scheduling algorithm.
* When the CPU is available, it is assigned to the process that has the smallest next CPU burst.
* If the next CPU bursts of two processes are the same, FCFS scheduling is used to break the tie.
* As an example of SJF scheduling, consider the following set of processes, with the length of the CPU burst given in milliseconds



* The waiting time is 3 milliseconds for process P1, 16 milliseconds for process P2, 9 milliseconds for process P3, and 0 milliseconds for process P4.
* Thus, the average waiting time is (3 + 16 + 9 + 0) / 4 = 7 milliseconds.
* The real difficulty with the SJF algorithm knowing the length of the next CPU request.
* The next CPU burst is generally predicted as an exponential average of the measured lengths of previous CPU bursts.

Burst Time: The process needs time to complete its execution is called Burst Time.

Arrival Time: The time at which the process arrived is called Arrival time. Generally, starts from 0.

Waiting Time: The amount of time the process waited for its execution.

Finish Time: The time at which the process completes its execution.

**Program:**

#include<stdio.h>

struct process

{

char name[10];

int bt,wt,ft,tat;

}p[10],temp;

int main()

{

int i,j,n,ttat=0,twt=0;

float atat,awt;

printf("Enter the number of processes:");

scanf("%d",&n);

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

{

printf("\nEnter the name of the process:");

scanf("%s",p[i].name);

printf("Enter the burst time of the %s process:",p[i].name);

scanf("%d",&p[i].bt);

printf("\n");

}

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

{

for(j=i+1;j<n;j++)

{

if(p[i].bt>p[j].bt)

{

temp=p[i];

p[i]=p[j];

p[j]=temp;

}

}

}

printf("Sorted order is:");

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

printf(" %s",p[i].name);

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

{

if(i==0)

{

p[i].wt=0;

p[i].ft=p[i].bt;

}

else

{

p[i].wt=p[i-1].bt+p[i-1].wt;

p[i].ft=p[i].wt+p[i].bt;

}

}

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

{

p[i].tat=p[i].ft;

ttat=ttat+p[i].ft;

twt=twt+p[i].wt;

}

atat=(float)ttat/n;

awt=(float)twt/n;

printf("\n\n");

printf("p.name\t bt\t wt\t ft\t tat\n");

printf("----------------------------------------------");

printf("\n");

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

{

printf(" %s\t%d\t %d\t %d\t %d\t",p[i].name,p[i].bt,p[i].wt,p[i].ft,p[i].tat);

printf("\n");

}

printf("\n\n\n");

printf("The total turn around time is %d\n",ttat);

printf("\nThe total waiting time is %d\n",twt);

printf("\nThe average turn around time is %f\n",atat);

printf("\nThe average waiting time is %f\n",awt);

}

**Input:**

Initially, we will be entering no.of processes. And the for each process we need to enter process name and process burst time.

**Output:**

We output the values of finish time, waiting time and turn-around time for each process using SJF CPU Scheduling Algorithm. And then we output the values of average waiting time and average turn-around time.

