



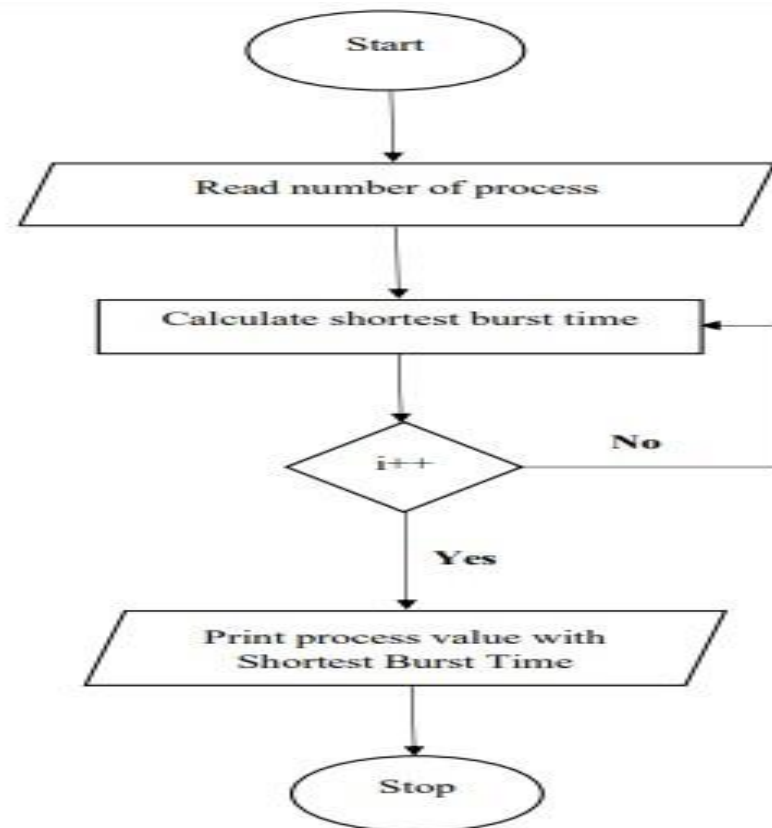
# OPERATING SYSTEM TUTORIAL ASSIGNMENT

SUBMITTED BY NARAYANI H AND THANUSHREE G

# **SJF (SHORTEST JOB FIRST) SCHEDULING ALGORITHM**

Shortest Job First (SJF) is an algorithm in which the process having the smallest execution time is chosen for the next execution. This scheduling method can be preemptive or non-preemptive. This is an approach which considers the CPU burst. When CPU is available, the process having the smallest CPU burst is allocated with CPU. Now it may happen that two or more processes have the same CPU burst then, which process to allocate will be decided as per FCFS scheduling.

# FLOWCHART



## FORMULA

- ▶ Arrival Time: Time at which the process arrives in the ready queue.
- ▶ Completion Time: Time at which process completes its execution.
- ▶ Burst Time: Time required by a process for CPU execution.
- ▶ Turn Around Time: Time Difference between completion time and arrival time.
- ▶ Turn Around Time = Completion Time – Arrival Time
- ▶ Waiting Time(W.T): Time Difference between turn around time and burst time.
- ▶ Waiting Time = Turn Around Time – Burst Time

## ADVANTAGES

- ▶ Reduces average waiting time.
- ▶ It provides good CPU utilization than FCFS (First Come First Search).

## DISADVANTAGES

- ▶ SJF can't be implemented for CPU scheduling for the short term. It is because there is no specific method to predict the length of the upcoming CPU burst time.
- ▶ This algorithm may cause very long turn-around time or starvation.

# CODE

```
1  #include<stdio.h>
2  int main()
3  {
4      int bt[20],p[20],wt[20],tat[20],i,j,n,total=0,pos,temp;
5      float avg_wt,avg_tat;
6      printf("Enter number of process:");
7      scanf("%d",&n);
8      printf("\nEnter Burst Time:\n");
9      for(i=0;i<n;i++){
10         printf("p%d:",i+1);
11         scanf("%d",&bt[i]);
12         p[i]=i+1;
13     }
14     //sorting of burst times
15     for(i=0;i<n;i++){
16         pos=i;
17         for(j=i+1;j<n;j++)
18         {
19             if(bt[j]<bt[pos])
20                 pos=j;
21         }
22         temp=bt[i];
23         bt[i]=bt[pos];
24         bt[pos]=temp;
25         temp=p[i];
26         p[i]=p[pos];
```

```

25         temp=p[i];
26         p[i]=p[pos];
27         p[pos]=temp;
28     }
29     wt[0]=0;
30     for(i=1;i<n;i++)
31     {
32         wt[i]=0;
33         for(j=0;j<i;j++)
34             wt[i]+=bt[j];
35
36         total+=wt[i];
37     }
38     avg_wt=(float)total/n;
39     total=0;
40     printf("\nProcess\t    Burst Time\t    \tWaiting Time\tTurnaround Time");
41     for(i=0;i<n;i++)
42     {
43         tat[i]=bt[i]+wt[i];
44         total+=tat[i];
45         printf("\np%d\t\t    %d\t\t\t    %d\t\t\t\t%d",p[i],bt[i],wt[i],tat[i]);
46     }
47     avg_tat=(float)total/n;
48     printf("\n\nAverage Waiting Time=%f",avg_wt);
49     printf("\nAverage Turnaround Time=%f\n",avg_tat);
50 }

```



# OUTPUT

```
PS E:\Mixture\BE\Fifth sem\OS> cd "e:\Mixture\BE\Fifth sem\OS\" ; if ($?) { gcc sjf.c -o sjf } ; if ($?) { .\sjf }  
Enter number of process:5
```

```
Enter Burst Time:
```

```
p1:4
```

```
p2:3
```

```
p3:7
```

```
p4:1
```

```
p5:2
```

Process	Burst Time	Waiting Time	Turnaround Time
p4	1	0	1
p5	2	1	3
p2	3	3	6
p4	1	0	1
p5	2	1	3
p2	3	3	6
p1	4	6	10
p3	7	10	17

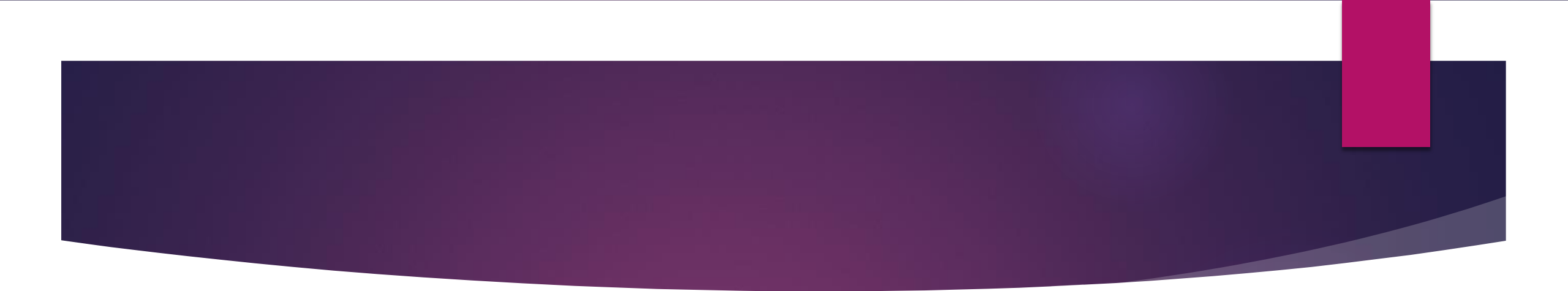
```
Average Waiting Time=4.000000
```

```
Average Turnaround Time=7.400000
```

```
PS E:\Mixture\BE\Fifth sem\OS> █
```

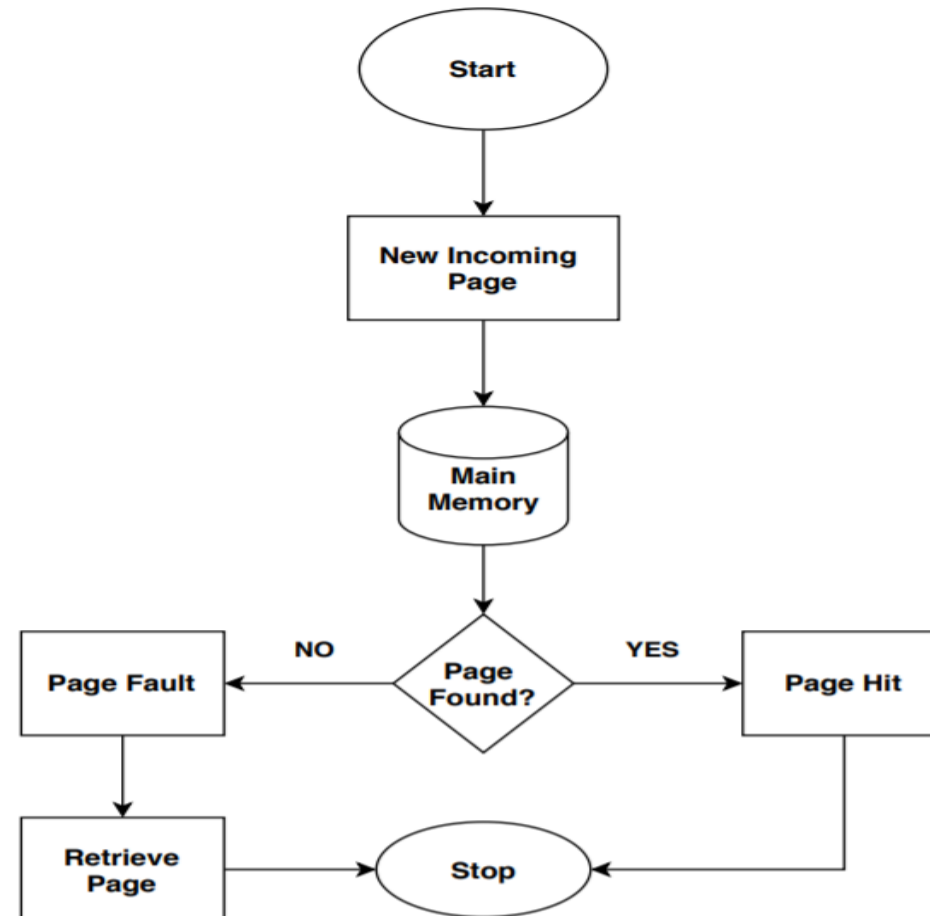
## FIFO (FIRST IN FIRST OUT)

In operating system that uses paging for memory management, page replacement algorithm are needed to decide which page needed to be replaced when new page comes in, which is not present in memory. Page fault occurs and operating system replaces one of the existing pages with new page. Different page replacement algorithms suggest different ways to decide which page to replace.



The target of all algorithms is to reduce number of page faults. This is the simplest page replacement algorithm. In this, the operating system keeps track of all pages in the memory in a queue, the oldest page is in the front of the queue. When a page needs to be replaced page in the front the queue is selected for removal

# FLOWCHART



## FORMULA

- ▶ Page fault is an interruption that occurs when a software program attempts to access a memory block not currently stored in the system's RAM. This exception tells the operating system to find the block in virtual memory so it can be sent from a device's storage (SSD or HD) to RAM.
- ▶ Page Hit – If CPU tries to retrieve the needed page from main memory, and that page is existed in the main memory (RAM), then it is known as “PAGE HIT”.

# ADVANTAGES

- ▶ It is easy to understand and implement.

# DISADVANTAGES

- ▶ The efficiency is low.
- ▶ When we increase the number of frames while using FIFO, we are giving more memory to processes. So, page faults should decrease, but here the page faults are increasing. This problem is called as “**Belady’s Anomaly**”.

# CODE

```
1  #include<stdio.h>
2
3  int main()
4  {
5      int i,j,n,a[50],frame[10],no,k,avail,count=0;
6      printf("\n ENTER THE NUMBER OF PAGES:\n");
7      scanf("%d",&n);
8
9      printf("\n ENTER THE PAGE NUMBER :\n");
10     for(i=1;i<=n;i++)
11         scanf("%d",&a[i]);
12
13     printf("\n ENTER THE NUMBER OF FRAMES :");
14     scanf("%d",&no);
15     for(i=0;i<no;i++)
16         frame[i]= -1;
17     j=0;
18
19     printf("ref string\t page frames\n");
20     for(i=1;i<=n;i++){
21         printf("%d\t\t",a[i]);
22         avail=0;
23         for(k=0;k<no;k++)
24             if(frame[k]==a[i])
25                 avail=1;
```



```
26
27     if (avail==0)
28     {
29         frame[j]=a[i];
30         j=(j+1)%no;
31         count++;
32     }
33     for(k=0;k<no;k++)
34         printf("%d\t",frame[k]);
35     printf("\n");
36 }
37
38 printf("Page fault is %d",count);
39 return 0;
40 }
```

# OUTPUT

```
PS E:\Mixture\BE\Fifth sem\OS> cd "e:\Mixture\BE\Fifth sem\OS\" ; if ($?) { gcc sjf.c -o sjf } ; if ($?) { .\sjf }  
Enter number of process:5
```

```
Enter Burst Time:
```

```
p1:4
```

```
p2:3
```

```
p3:7
```

```
p4:1
```

```
p5:2
```

Process	Burst Time	Waiting Time	Turnaround Time
p4	1	0	1
p5	2	1	3
p2	3	3	6
p4	1	0	1
p5	2	1	3
p2	3	3	6
p1	4	6	10
p3	7	10	17

```
Average Waiting Time=4.000000
```

```
Average Turnaround Time=7.400000
```

```
PS E:\Mixture\BE\Fifth sem\OS> █
```