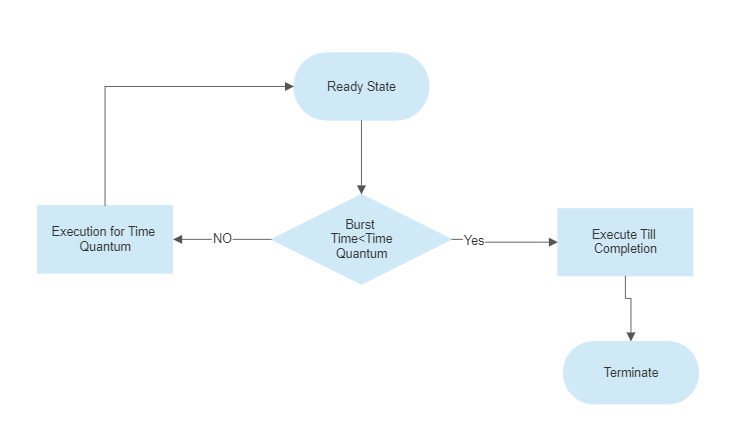
**Dynamic Implementation of SJF primitive Scheduling Algorithm**

**GROUP-4**

**FLOWCHART-**

**ALGORITHM-**

**Algorithm of SRTF (Shortest Remaining Time First)**

This algorithm is the preemptive version of SJF scheduling. The time quantum is 1 in this algorithm. In the below example there are 3 jobs P1,P2,P3. The arrival time and burst time are given below in the table.

| Process ID | Arrival Time | Burst Time |
| --- | --- | --- |
| P1 | 0 | 7 |
| P2 | 1 | 3 |
| P3 | 3 | 4 |

The Gantt chart is prepared according to the arrival and the burst time given in the table.

We know the Gantt chart is always starts from 0.

**Step 1 :** Since, at time 0, the only available process is P1 with Burst Time 7.

| P1 |
| --- |

0 1

**Step 2 :** As, the scheduling As the scheduling algorithm is SRTF the next process arrives at time unit 1 and the scheduler check for the least Burst Time P2 arrives into the queue.

| P1 P2 |
| --- |

0 1 2

**Step 3 :** As, P2 has again least Burst Time, again P2 arrives at time 2.

| P1 P2 P2 |
| --- |

0 1 2 3

**Step 4:** As, we seen P2 has least Burst Time among the other processes so we put it in the queue till its execution.

| P1 P2 P2 P2 |
| --- |

0 1 2 3 4

**Step 5 :** We are left with P2 and P3, P3 has least Burst Time so, P3 comes into the queue till its executed itself and Burst Time of P3 is 4.

| P1 P2 P2 P2 P3 |
| --- |

0 1 2 3 4 8

**Step 6 :** We have left with only one process ie P1 so we can easily run this process till its execution.

| P1 P2 P2 P2 P3 P1 |
| --- |

0 1 2 3 4 8 14

Once all the processes arrive, no preemption is done and the algorithm will work as SJF.

**PROGRAM-**

#include<stdio.h>

int main()

{

int at[10],bt[10],rt[10],endTime,i,smallest;

int remain=0,n,time,sum\_wait=0,sum\_turnaround=0;

printf("Enter no of Processes : ");

scanf("%d",&n);

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

{

printf("Enter arrival time for Process P%d : ",i+1);

scanf("%d",&at[i]);

printf("Enter burst time for Process P%d : ",i+1);

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

rt[i]=bt[i];

}

printf("\n\nProcess\t|Turnaround Time| Waiting Time\n\n");

rt[9]=9999;

for(time=0;remain!=n;time++)

{

smallest=9;

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

{

if(at[i]<=time && rt[i]<rt[smallest] && rt[i]>0)

{

smallest=i;

}

}

rt[smallest]--;

if(rt[smallest]==0)

{

remain++;

endTime=time+1;

printf("\nP[%d]\t|\t%d\t|\t%d",smallest+1,endTime-at[smallest],endTime-bt[smallest]-at[smallest]);

sum\_wait+=endTime-bt[smallest]-at[smallest];

sum\_turnaround+=endTime-at[smallest];

}

}

printf("\n\nAverage waiting time = %f\n",sum\_wait\*1.0/n);

printf("Average Turnaround time = %f",sum\_turnaround\*1.0/5);

return 0;

}

**RESULT-**

