**LOVELY PROFESSIONAL UNIVERSITY**

School of Computer Science and Engineering Faculty of Technology And Sciences

Name of the faculty member : Priya Virdi Name of the student:Majji Naveen kumar

Course Code: CSE316 Course Admission No:11703801

Title: Operating System Roll No:04

Gmail:myself890@gmail.com

Github:

DESCRIPTION:

Shortest job first (SJF) or shortest job next, is a scheduling policy that selects the waiting process with the smallest execution time to execute next. SJN is a non-preemptive algorithm.

If the **arrival time** for processes are different, which means all the processes are not available in the ready queue at time 0, and some jobs arrive after some time, in such situation, sometimes process with short burst time have to wait for the current process's execution to finish, because in Non Pre-emptive SJF, on arrival of a process with short duration, the existing job/process's execution is not halted/stopped to execute the short job first.

This leads to the problem of **Starvation**, where a shorter process has to wait for a long time until the current longer process gets executed. This happens if shorter jobs keep coming, but this can be solved using the concept of **aging**.

It is practically infeasible as Operating System may not know burst time and therefore may not sort them. While it is not possible to predict execution time, several methods can be used to estimate the execution time for a job, such as a weighted average of previous execution times. SJF can be used in specialized environments where accurate estimates of running time are available.

PROCESS ARRAVIAL TIME BURST TIME

P1 0 20

P2 5 36

P3 13 19

P4 17 42

ALGORITHM:

1- Sort all the processes in increasing order

according to burst time.

2- Then simply, apply [FCFS](https://www.geeksforgeeks.org/program-fcfs-scheduling-set-1/).

CONSTRAINTS:

**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

CODE SNIPPET:

**import** java.util.Scanner;  
**public class** csk  
{  
 **public static void** main(String args[])  
 {  
 operating o1 = **new** operating();  
 o1.display();  
  
 }  
}  
  
  
**class** operating  
{  
  
 **void** display()  
 {  
 **int** i,j,k=1,n,min=0,btime=0,temp,sum=0;  
 **float** avgwt=0,avgtat=0;  
  
  
 Scanner s = **new** Scanner(System.***in***);  
 System.***out***.print(**"enter no of process = "**);  
 n = s.nextInt();  
 **int** pro[] = **new int**[n];  
 **int** burst[] = **new int**[n];  
 **int** arr[] = **new int**[n];  
 **int** wt[] = **new int**[n];  
 **int** tat[] = **new int**[n];  
 **int** st[] = **new int**[n];  
 **int** ft[] = **new int**[n];  
 **int** pri[] = **new int**[n];  
 **for**(i=0;i<n;i++)  
 {  
 pro[i] = i+1;  
 System.***out***.print(**"enter burst time for process\_id "**+pro[i]+**" = "** );  
 burst[i] = s.nextInt();  
 System.***out***.print(**"enter arrival time for process\_id "**+pro[i]+**" = "** );  
 arr[i] = s.nextInt();  
 }  
 **for**(i=0;i<n;i++)  
 {  
 **for**(j=0;j<n;j++)  
 {  
 **if**(arr[i]<arr[j])  
 {  
 temp=pro[j];  
 pro[j]=pro[i];  
 pro[i]=temp;  
 temp=arr[j];  
 arr[j]=arr[i];  
 arr[i]=temp;  
 temp=burst[j];  
 burst[j]=burst[i];  
 burst[i]=temp;  
 }  
 }  
 }  
  
  
 **for**(j=0;j<n;j++)  
 {  
  
 btime=btime+burst[j];  
 min=burst[k];  
 **for**(i=k;i<n;i++)  
 {  
  
 **if** (btime >= arr[i] && burst[i] <= min)  
 {  
 temp=pro[k];  
 pro[k]=pro[i];  
 pro[i]=temp;  
 temp=arr[k];  
 arr[k]=arr[i];  
 arr[i]=temp;  
 temp=burst[k];  
 burst[k]=burst[i];  
 burst[i]=temp;  
 }  
 }  
 **if**(k < n-1)  
 k++;  
 }  
 wt[0] = 0;  
  
  
 **for**(i=0; i<n; i++)  
 {  
  
 **if**(i==0)  
 st[i]=arr[i];  
 **else** st[i]=ft[i-1];  
  
 wt[i]=st[i]-arr[i];  
 ft[i]=st[i]+burst[i];  
 tat[i]=ft[i]-arr[i];  
 pri[i] = 1 + (wt[i]/burst[i]);  
 }  
  
 System.***out***.println(**"sorting according to arrival time"**);  
 System.***out***.println(**"process\_id\t\tBurst time\t\t Arrival time\t\t waiting time\t\tTurn-around\t\tpriority\n"**);  
 **for**( i=0;i<n;i++)  
 {  
 System.***out***.println(pro[i]+**"\t\t\t\t"**+burst[i]+**" \t\t\t\t\t"**+arr[i]+**"\t\t\t\t\t\t"**+wt[i]+**"\t\t\t\t"**+tat[i]+**"\t\t\t\t"**+pri[i]);  
 }  
 **for**(i=0;i<n;i++){  
 avgwt=avgwt+wt[i];  
 avgtat=avgtat+tat[i];  
 }  
 System.***out***.println(**"\naverage turn around time is "**+(**float**)(avgtat/n));  
 System.***out***.println(**"\naverage waiting time is "**+(**float**)(avgwt/n));  
  
 }  
}

TEST CASES:

enter no of process = 4

enter burst time for process\_id 1 = 20

enter arrival time for process\_id 1 = 0

enter burst time for process\_id 2 = 36

enter arrival time for process\_id 2 = 5

enter burst time for process\_id 3 = 19

enter arrival time for process\_id 3 = 13

enter burst time for process\_id 4 = 42

enter arrival time for process\_id 4 = 17

sorting according to arrival time

process\_id Burst time Arrival time waiting time Turn-around priority

1 20 0 0 20 1

3 19 13 7 26 1

2 36 5 34 70 1

4 42 17 58 100 2

average turn around time is 54.0

average waiting time is 24.75