**LOVELY PROFESSIONAL UNIVERSITY**

**NAME**: G.PRANEETH KUMAR REDDY

**REGISTRATION NO**: 11709172

**ROLL NO**: A23

**E-mail**:gaddampraneeth143@gmail.com

**GITHUB LINK:**

**DESCRIPTION:** Shortest job first (SJF) is a scheduling algorithm that selects the waiting process with the smallest execution time to execute next It is non-preemptive algorithm for the given problem. This scheduling algorithm is optimal if all the jobs/processes are available at the same time. If the arrival time for processes are different, which means all the processes are not available in the ready queue at time zero, 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.

**ALGORITHM**:

SJF is based on prediction of burst time. SJF schedules a process with smallest burst time. In simple words, This Algorithms executes process with their increasing order of CPU burst time. If all processes are having same CPU burst time then this works as FCFS algorithm. SJF minimizes average waiting time, however it’s hard to predict CPU burst time in advance.

**CONSTRAINTS:**

**Arrival time:** The time at which the process enters into ready queue.

**Turnaround time:** The interval between the time of submission of a process to the time of completion.

**Waiting time:** The total amount of the time a process spends in ready queue.

**Completion time:** The time at which process completes its execution.

**Burst time:** The time needed by CPU to completes its execution.

**CODE SNIPPET:**

#include<stdio.h>

#include<iostream>

#include<conio.h>

Using namespace std;

void project();

class A

{

int i,j,k,n,po,minimum,btime,temp,sum,ta;

float wavg,tavg,tsum,wsum;

public:

void project(int n)

{

int process[n],burst[n],arrival[n],waiting[n],turn[n],start[n],finish[n],pri[n];

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

{

process[i]=i+1;

printf("enter burst time = \t");

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

printf("enter arrival time = \t");

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

}

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

{

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

{

if(arrival[i]<arrival[j])

{

temp=process[j];

process[j]=process[i];

process[i]=temp;

temp=arrival[j];

arrival[j]=arrival[i];

arrival[i]=temp;

temp=burst[j];

burst[j]=burst[i];

burst[i]=temp;

}

}

}

k=1;

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

{

btime=btime+burst[j];

minimum=burst[k];

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

{

if(btime>=arrival[i]&&burst[i]<=minimum)

{

temp=process[k];

process[k]=process[i];

process[i]=temp;

temp=arrival[k];

arrival[k]=arrival[i];

arrival[i]=temp;

temp=burst[k];

burst[k]=burst[i];

burst[i]=temp;

}

}

k++; }

waiting[0] = arrival[0];

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

{

if(i==0)

start[i]=arrival[i];

else

start[i]=finish[i-1];

if(start[i] < arrival[i]){

start[i]=arrival[i];

}

waiting[i]=start[i]-arrival[i];

finish[i]=start[i]+burst[i];

turn[i]=finish[i]-arrival[i];

pri[i] = 1 + (waiting[i]/burst[i]);

}

printf("\nProcess\t Burst\t Arrival\t Waiting Time\t Turn-around\tpriority" );

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

{

printf("\n p%d\t %d\t %d\t\t %d\t\t\t%d\t\t%d",process[i],burst[i],arrival[i],waiting[i],turn[i],pri[i]);

}

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

wsum=wsum+waiting[i];

tsum=tsum+turn[i];

}

tavg=tsum/n;

wavg=wsum/n;

printf("\t Average turn around tim is %f",tavg);

printf("\t Average waiting time is %f",wavg);

}

}ob;

main()

{

int n;

printf("enter number of process = ");

scanf("%d",&n);

ob.project(n);

}

**BOUNDARY CONDITIONS:**

The size of array intialised in the code are fixed and have fixed memory allocated.

The values like burst time and arrival time given by the user is taken by default as integer as it is declared.

**TEST CASES:**

enter number of process = 4

enter burst time = 20

enter arrival time = 0

enter burst time = 36

enter arrival time = 5

enter burst time = 19

enter arrival time = 13

enter burst time = 42

enter arrival time = 17

**OUTPUT:**

Process Burst Arrival Waiting Time Turn-around priority

P1 20 0 0 20 1

P2 19 13 7 26 1

P3 36 5 34 70 1

p4 42 17 58 100 2

Average turnaround time is 54.000000

Average waiting time is 24.750000