Simulation Based Assignment

Student Name: Prudhvi Rambha

Student ID : 11613210

Email Address: Prudhvirambha@gmail.com

GitHub Link: https://github.com/prudhvirambha/cse316

Faculty Name: Oshin

UID : 21980

Course Code : CSE 316

Lovely Professional University , Punjab

Question 1:

Ques. 9. Design a scheduler that uses a preemptive priority scheduling algorithm based on dynamically changing priority. Larger number for priority indicates higher priority.
Assume that the following processes with arrival time and service time wants to execute (for reference):

ProcessID	Arrival Time	Service Time
P1	0	4
P2	1	1
P3	2	2
P4	3	1

When the process starts execution (i.e. CPU assigned), priority for that process changes at the rate of m=1.When the process waits for CPU in the ready queue (but not yet started execution), its priority changes at a rate n=2. All the processes are initially assigned priority value of 0 when they enter ready queue for the first time . The time slice for each process is q = 1. When two processes want to join ready queue simultaneously, the process which has not executed recently is given priority. Calculate the average waiting time for each process. The program must be generic i.e. number of processes, their burst time and arrival time must be entered by user.

Description :

Process ID	ArrivalTime	Service time
P1	0	4
P2	1	1
P3	2	2
P4	3	1

The Processes in the above table execute on the basis of time slice which value is 1 and also on dynamically changing priorities .

Here's how there will get executed :

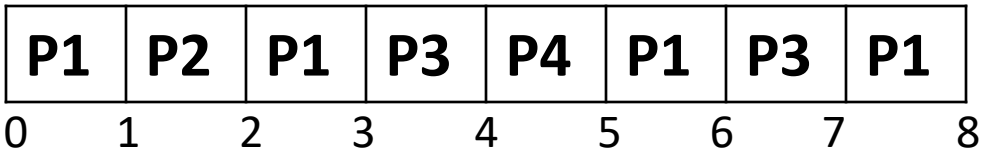
The Priorities changes in this manner

Time	Priority
0	p1=1,
1	p2=1,p1=2.
2	P2=2,p1=2,p3=0
3	P1=1,p3=2,p4=0
4	P3=1,p1=2,p4=2
5	P1=1,p3=2,p4=2

These are completion times of the processes:

1.The process P1 finished at 8.
2.The process P2 finished at 2.
3.The process P3 finished at 7.
4.The process P4 finished at 5.

Algorithm :



Gantt Chart :

```
Gantt Chart
{0}->P1<-{1}->P2<-{2}->P1<-{3}->P3<-{4}->P4<-{5}->P1<-{6}->P3<-{7}->P1<-{8}
```

Constraints :

Arrival Time	Service Time
0	4
1	1
2	2
3	1

Time Slice =1
Initially All priorities are =0
Priority for the running process =1
Priority for process waiting in Ready queue =2

Code Snippet :

```
#include<unistd.h>
#include<stdio.h>
int ct[40]={0},qt,qas[40]={0},c=0,st,btm[40]={0},tt,
wt,at[40],bt[40],zz;
void queue1(int fin,int tm);
void queue2(int fin, int tm);
void pop(int fin);
float att, awt;
int flg=0,tm=0,ee=0,fin=0;
void pop(int fin){
qas[ee]=fin+1;
ee++;
}
void queue1(int fin,int tm){
for(int x=fin+1;x<4;x++){
if(at[x]<=tm){
qas[ee]=x+1;
ee++;}
}
}
void queue2(int fin, int tm){
for(int x=fin+1;x<4;x++){
int fl=0;
for(int y=0;y<ee;y++){
if(qas[y]==x+1){
fl++;}}
if(at[x]<=tm && fl==0 && btm[x]!=0){
qas[ee]=x+1;
ee++;}
}
}
```

```
main()
{
printf("\nEnter time slice:");
scanf("%d",&qt);
printf("Enter no of process");
scanf("%d",&zz);
for(int x=0;x<zz;x++){
printf("\n Process %d",x+1);
printf("\n Arrival Time=");
scanf("%d",&at[x]);
printf(" Burst Time=");
scanf("%d",&bt[x]);
btm[x]=bt[x];}
printf("\n Gantt Chart \n {%d",at[0]);
do{
if(flg==0){
st=at[0];
if(btm[0]<=qt){
tm=st+btm[0];
btm[0]=0;
queue1(fin,tm);}
else{
btm[0]=btm[0]-qt;
tm=st+qt;
queue1(fin,tm);
pop(fin);}
}
else{
fin=qas[0]-1;
st=tm;
```

```
for(int x=0;x<ee && ee!=1;x++){
qas[x]=qas[x+1];}
ee--;
if(btm[fin]<=qt){
tm=st+btm[fin];
btm[fin]=0;
queue2(fin, tm);}
else{
btm[fin]=btm[fin]-qt;
tm=st+qt;
queue2(fin, tm);
pop(fin);}
}
if(btm[fin]==0){
ct[fin]=tm;
}
flg++;
printf("}->P%d<-{%d",fin+1,tm);
}while(ee!=0);
printf("}  ");
printf("\n\nProcess \t Arrival Time\t Burst Time \t
Completition Time\t Turn Around Time\t Waiting
Time\n");
for(int x=0;x<zz;x++){
tt=ct[x]-at[x];
wt=tt-bt[x];
printf("P%d\t\t  %d\t\t  %d\t\t  %d\t\t\t   %d\t\t\t
%d\n",x+1,at[x],bt[x],ct[x],tt,wt);
awt=awt+wt;
att=att+tt;
}
printf("\nAverage Turn Around Time: %f\nAverage
Waiting Time: %f",att/4,awt/4);
}
```

Boundary Conditions :

Time Slice Should be always 1 as it acts similarly as Round Robin alogorithm

Test Case :

This is the given default test case for reference.

Process ID	ArrivalTime	Service time
P1	0	4
P2	1	1
P3	2	2
P4	3	1

Have you made minimum 5 revisions of solution on GitHub? Yes

GitHub Link: https://github.com/prudhvirambha/cse316

Question 2:

Ques. 10. Design a scheduler with multilevel queue having two queues which will schedule the processes on the basis of pre-emptive shortest remaining processing time first algorithm (SROT) followed by a scheduling in which each process will get 2 units of time to execute. Also note that queue 1 has higher priority than queue 2. Consider the following set of processes (for reference)with their arrival times and the CPU burst times in milliseconds.

Process	Arrival-Time	Burst-Time
P1	0	5
P2	1	3
P3	2	3
P4	4	1

Description :

Process	Arrival-Time	Burst-Time
P1	0	5
P2	1	3
P3	2	3
P4	4	1

The Processes in the above table execute on the basis of time slice which value is 1 and also depends on priority of two queues. Using Shortest remaining time first.

The Priorities of first queue is more than the priority of 2nd queue

These are completion times of the processes:

1.The process P1 finished at 5.
2.The process P2 finished at 8.
3.The process P3 finished at 12.
4.The process P4 finished at 11.

If P1&P3 are in queue1 and p2&p4 are in queue2

Constraints :

Time Slice =2
Priority of first queue is more than priority of queue2

```
root@richard-parker:~/Desktop/git# ./a.out
Enter no. of processes: 4
Enter type of process: 1.Queue1 2.Queue2
>>>1
Enter arrival and burst time for p1: 0
5
Enter type of process: 1.Queue1 2.Queue2
>>>2
Enter arrival and burst time for p2: 1
3
Enter type of process: 1.Queue1 2.Queue2
>>>1
Enter arrival and burst time for p3: 2
3
Enter type of process: 1.Queue1 2.Queue2
>>>2
Enter arrival and burst time for p4: 4
1
Enter time quantum: 2
Process Id Arrival Time Brust Time Complition Time Turn Around Time Waiting Timr   Types
        1          0          5          5          5                   0                   1
        3          2          3          8          6                   3                   1
        2          1          3         12         11                   8                   2
        4          4          1         11          7                   6                   2

Average TAT = 7.25 ms
Average WT= 4.25 ms
```

Code Snippet :

```
#include<stdio.h>
#include<stdlib.h>
#include<strings.h>
#define size 40
#define info 42567
int time;
typedef struct process
{
        int pid,AT,BT,CT,TAT,WT,RT,type;
}process;
typedef struct queue{
        int q[size];
        int f,r;
}queue;

void output(process p[],int n)
{
        int i;
        printf("Process Id Arrival Time Brust Time Complition
Time Turn Around Time Waiting Timr   Types\n");
        for(i=0;i<n;i++)
        {
                printf("%10d %10d %10d %10d %10d
%20d
%20d\n",p[i].pid,p[i].AT,p[i].BT,p[i].CT,p[i].TAT,p[i].WT,p[i].type);
        }
        printf("\n");
}

void ins(queue *t,int ele)
{
        t->r++;
        t->q[t->r] = ele;
}
```

Code Snippet :

```
int qf(queue *t)
{
        return t->q[t->f];
}
int del1(queue *t){
        int i=t->f;
        int z=t->q[t->f];
        while(i < t->r){
                t->q[i] = t-
>q[i+1];
                i++;
        }
        t->r--;
        return z;
}

void sort(process p[],int n)
{
        int i,j;
        process val;
        for(i=0;i<n-1;i++)
        {
                val = p[i+1];
                for(j=i;j>=0;j--)

        if(val.type<p[j].type)

        p[j+1] = p[j];
                                else

        break;
                p[j+1] = val;
        }
}
```

Code Snippet :

```
int tnt(process p[],int n,int time)
{
        int i = pop(p,n,time);
        p[i].RT--;
        if(p[i].RT == 0)
        {
                p[i].CT  = time+1;
                p[i].TAT = p[i].CT - p[i].AT;
                p[i].WT  = p[i].TAT - p[i].BT;
        }
        return p[i].pid;
}


float avgTAT(process p[],int n)
{
        int i;
        float avg=0;
        for(i=0;i<n;i++)
                avg += p[i].TAT;
        avg = avg / n;
        return avg;
}
float avgWT(process p[],int n)
{
        int i;
        float avg=0;
        for(i=0;i<n;i++)
                avg += p[i].WT;
        avg = avg / n;
        return avg;
}
```

Code Snippet :

```
void mqu(process p[],int n,int tq,int chart[])
{
        int t,i,j,T=0,hp;
        int cur_pid;
        queue x;
        x.f = 0;
        x.r = -1;
        for(i=0;i<n;i++)
                T += p[i].BT;
        atsort(p,n);
        sort(p,n);
        for(t=0;t<T;t++)
        {
                for(i=0;i<n;i++)

        if(p[i].type == 2&&p[i].AT==t)

        ins(&x,p[i].pid);
        for(i=0;i<n;i++int tnt(process
p[],int n,int time)
{
        int i = pop(p,n,time);
        p[i].RT--;
        if(p[i].RT == 0)
        {
                p[i].CT  = time+1;
                p[i].TAT = p[i].CT -
p[i].AT;
                p[i].WT  = p[i].TAT -
p[i].BT;
        }
        return p[i].pid;
}


float avgTAT(process p[],int n)
{
        int i;
        float avg=0;
        for(i=0;i<n;i++)
                avg += p[i].TAT;
        avg = avg / n;
```

Code Snippet :

```
int rr(process p[],queue *t,int
tq,int n,int time)
{
        int i,runTime,cp;
        cp = qf(t);
        for(i=0;i<n;i++)
                if(p[i].pid ==
cp)

        break;
        p[i].RT--;
        runTime = p[i].BT-p[i].RT;
        if(runTime%tq == 0 &&
p[i].RT != 0)
        {
                del1(t);
                ins(t,cp);
        }
        if(p[i].RT == 0)
        {
                p[i].CT  =
time+1;
                p[i].TAT =
p[i].CT - p[i].AT;
                p[i].WT  =
p[i].TAT - p[i].BT;
                del1(t);
        }
        return p[i].pid;
}
```

Code Snippet :

```
int main()
{
        int i,n,q,chart[size];
        process p[size];
        float avg_tat,avg_wt;
        printf("Enter no. of
processes: ");
        scanf("%d",&n);
        for(i=0;i<n;i++)
        {
                printf("Enter
type of process: 1.Queue1 2.Queue2
\n>>>");

        scanf("%d",&p[i].type);
                printf("Enter
arrival and burst time for p%d: ",i+1);

        scanf("%d%d",&p[i].AT,&p[i]
.BT);
                p[i].pid = i+1;
                p[i].RT = p[i].BT;
        }

        printf("Enter time quantum:
");
        scanf("%d",&q);

        mqu(p,n,q,chart);
        output(p,n);
        avg_tat = avgTAT(p,n);
        printf("average TAT = %.2f
ms\n",avg_tat);
        avg_wt = avgWT(p,n);
        printf("average WT= %.2f
ms\n",avg_wt);

}
```

Boundary Conditions :

The Priority of first queue should be greater than the second queue and it should be SROT

Test Case :

This is the given default test case for reference.

Process	Arrival-Time	Burst-Time
P1	0	5
P2	1	3
P3	2	3
P4	4	1

Have you made minimum 5 revisions of solution on GitHub? Yes

GitHub Link: https://github.com/prudhvirambha/cse316