GoodLuck | Page No.
Date

Assignment - C1

Title :- Scheduling problem

Problem statements :-

Write a Java program (using OOP features) to implement following scheduling algorithm FCFS, SJF (premptive, Non premptive), Priority (Non premptive) & Round Robin (premptive)

Objective :-

i) Process scheduling in multitasking & multiuser OS.

ii) Implementation of scheduling algorithms.

S/W & H/W :-

C++ editors & compilers for linux OS, keyboard, mouse, etc.

Outcomes :-

The student will be able to

i) Compare the scheduling algorithms

ii) Implement FCFS, SJF, RR scheduling algorithms.

Theory :-

1] FCFS scheduling :-

The process requests are schedules in the order of their arrival time. The pending request are in a queue. The first request in the queue is scheduled first. The coming requests are added to the end of queue.

Algorithm :-

1] Input the process along burst time
2] Input arrival time for all process
3] Sort according to their arrival time along with indices
4] Perform process in sorted order
5] stop.

2] Shortest Job First :-

SJF is a scheduling policy that selects the waiting process with the smallest execution time to execute next.

Shortest Job first has the advantage of having a

a) Minimum average waiting time among all algorithms
b) It is greedy algorithm.

Algorithm :-

i) Sort all the process according to their arrival time.
ii) Then select that process which has minimum arrival time & minimum burst time
iii) After completion of process make a pool of process which after till the completion of process & select that process among the pool which is having minimum burst time

3] Round Robin scheduling :-
schedules using the time slicing. The amount of CPU time a process may use when allocated is limited. The process is time-empted. If the process requires more time or if process requires I/O operation before the time slice. It makes weighted turnaround time approximately equal & all time but throughput may not be well as all processes are treated equally

Algorithm :-

1] Get the input for process with arrival time & burst time. take quantum.
2] Sort processes according to arrival time.
3] Process till all processes are done
4] End.

4] Priority based scheduling :-
It is nonpremptive algorithm & one of the common scheduling algorithm in batch system Each process is assigned a priority & process with highest priority is executed first & so on. Processes with same priority are executed on FCFS basis.

Algorithm :-

1] Get Input for process including arrival time, burst time & priority.
2] Sort process according to arrival time
3] If process have same arrival time, sort them by priority.
4] Print process according to index

Conclusion :-

We have learnt & successfully implemented the scheduling algorithms.

```
package SheduleAlgo;


import java.util.*;


class ScheduleAlgo {


 static void fcfs()
 {
  System.out.println("\n\n\t####### FCFS #######");

  int pr,temp,ftime=0;
  float avgw=0,avgt=0;
  ArrayList<Integer> ti = new ArrayList<>();
  Scanner sc = new Scanner(System.in);
  System.out.print("\n\tEnter number of processes : ");
  pr = sc.nextInt();
  int at[],bt[],tt[],wt[],id[];
  at = new int[pr];
  bt = new int[pr];
  tt = new int[pr];
  wt = new int[pr];
  id= new int[pr];

  for(int i=0;i<pr;i++)
  {
   System.out.print("\n\tArrival time of P"+(i+1)+"  : ");
   at[i] = sc.nextInt();
   System.out.print("\n\tBurst time of P"+(i+1)+"  : ");
   bt[i] = sc.nextInt();
   id[i]=i+1;
  }
  for(int i = 0 ; i <pr; i++)
  {
   for(int  j=0;  j < pr-(i+1) ; j++)
   {
    if( at[j] > at[j+1] )
    {
     temp = at[j];
     at[j] = at[j+1];
     at[j+1] = temp;
     temp = bt[j];
     bt[j] = bt[j+1];
     bt[j+1] = temp;
     temp = id[j];
     id[j] = id[j+1];
     id[j+1] = temp;
    }
   }
  }
```

```
 ti.add(at[0]);
 for(int i=0;i< pr;i++)
 {
  if(i==0)
   ftime = at[i]+bt[i];
  else
  {
   if(ftime > at[i])
    ftime +=bt[i];
   else
    ftime = at[i] + bt[i];
  }

  ti.add(ftime);

  tt[i] = ftime - at[i];
  wt[i] = tt[i] - bt[i];
  avgw += wt[i];
  avgt += tt[i];
 }

 System.out.println("\n\n\t*****************************************");
 System.out.println("\tID AT BT TAT WT");
 System.out.println("\t*****************************************");
 for(int  i = 0 ; i< pr;  i++)
 {
  System.out.println("\tP"+ id[i] + "  \t " + at[i] + "\t" + bt[i]  + "\t" + tt[i] + "\t  "  + wt[i] ) ;
 }

 System.out.println("\n\tAverage turnaround time : "+(avgt/pr));
 System.out.println("\n\tAverage waiting time : "+ (avgw/pr));

 System.out.println("\n\tGantt chart :\n\t");

  for(int i=0;i<pr;i++)
   System.out.println("\t\t" +ti.get(i)+ "--->"+ ti.get(i+1) + " : P"+ id[i]);

  System.out.println("\t=======================");


}

 static void sjf()
 {
  System.out.println("\n\n\t####### SJF #######");
  int pr,temp,ftime=0;
  float avgw=0,avgt=0;
  Scanner sc = new Scanner(System.in);
  System.out.print("\n\tEnter number of processes : ");
  pr = sc.nextInt();
  int at[],bt[],tt[],wt[],id[],rt[];
  at = new int[pr]; bt = new int[pr]; tt = new int[pr];wt = new int[pr];  id= new int[pr]; rt = new int[pr]; // memory all
ocation
```

```
ArrayList<Integer> seq = new ArrayList<>();
ArrayList<Integer> ti = new ArrayList<>();
ArrayList<Integer> s = new ArrayList<>();

for(int i=0;i<pr;i++)
{
 System.out.print("\n\tArrival time of P"+(i+1)+" : ");
 at[i] = sc.nextInt();
 System.out.print("\n\tBurst time of P"+(i+1)+" : ");
 bt[i] = sc.nextInt();
 id[i]=i+1;
}
for (int i = 0; i < pr; i++)
      rt[i] = bt[i];

int complete=0,t=0,min = Integer.MAX_VALUE;
int f_t,min_index=0;
boolean c = false;


while(complete!=pr)
{
 for(int i=0;i<pr;i++)
 {
  if( (at[i]<= t) && (rt[i]>0) && (rt[i]< min) )
  {
   min = rt[i];
   min_index = i;
   c = true;
  }
 }

 if(c == false)
 {
  t++;
  continue;
 }

 seq.add(min_index);
 rt[min_index]--;
 min = rt[min_index];
 if(min == 0)
  min = Integer.MAX_VALUE;

 if(rt[min_index] == 0)
 {
  complete++;
  c = false;

  f_t = t+1;
  tt[min_index] = f_t - at[min_index];
  wt[min_index] = tt[min_index] - bt[min_index];
  avgt+=tt[min_index];
  avgw+=wt[min_index];
```

```
 }
 t++;
}

System.out.println("\n\n\t*****************************************");
System.out.println("\tID AT BT TAT WT");
System.out.println("\t*****************************************");
for(int  i = 0 ; i< pr;  i++)
{
 System.out.println("\tP"+ id[i] + "  \t " + at[i] + "\t" + bt[i]  + "\t" + tt[i] + "\t"  + wt[i] ) ;
}

System.out.println("\n\tAverage turnaround time : "+(avgt/pr));
System.out.println("\n\tAverage waiting time : "+ (avgw/pr));

ti.add(0);
int no=1,i;
for(i=0;i<seq.size()-1;i++)
{
 if(seq.get(i) == seq.get(i+1))
 {
  no++;
  continue;
 }
 s.add(seq.get(i));
 ti.add(no);
 no++;

}
s.add(seq.get(i-1));
ti.add(no);


System.out.println("\n\tGantt chart :\n\t");

 for(i=0;i<s.size();i++)
  System.out.println("\t\t"+ti.get(i)+ "--->"+ ti.get(i+1) + " : P"+(s.get(i)+1));

 System.out.println("\t======================");
}

static void rrobin()
{
System.out.println("\n\n\t####### Round Robin (Preemptive) #######");
int pr,temp,ftime=0,quantum;
float avgw=0,avgt=0;
Scanner sc = new Scanner(System.in);
System.out.print("\n\tEnter number of processes : ");
pr = sc.nextInt();
System.out.print("\n\tEnter the quantum : ");
quantum = sc.nextInt();
int at[],bt[],tt[],wt[],id[],rt[],inq[];
ArrayList<Integer> seq = new ArrayList<>();
ArrayList<Integer> ti = new ArrayList<>();
```

```
 at = new int[pr]; bt = new int[pr]; tt = new int[pr];wt = new int[pr];  id= new int[pr]; rt = new int[pr]; inq= new int[
pr]; // memory allocation

 for(int i=0;i<pr;i++)
 {
  System.out.print("\n\tArrival time of P"+(i+1)+" : ");
  at[i] = sc.nextInt();
  System.out.print("\n\tBurst time of P"+(i+1)+" : ");
  bt[i] = sc.nextInt();
  id[i]=i+1;
  inq[i]=0;
 }
   for(int i = 0 ; i <pr; i++)
   {
    for(int  j=0;  j < pr-(i+1) ; j++)
    {
     if( at[j] > at[j+1] )
     {
      temp = at[j];
      at[j] = at[j+1];
      at[j+1] = temp;
      temp = bt[j];
      bt[j] = bt[j+1];
      bt[j+1] = temp;
      temp = id[j];
      id[j] = id[j+1];
      id[j+1] = temp;
     }
    }
   }


 for (int i = 0; i < pr; i++)
       rt[i] = bt[i];

 Queue<Integer> q = new LinkedList<>();
 int complete=0,index=0,t=at[index]; ti.add(at[index]);
 q.add(0);

 while(complete!=pr)
 {
  index = q.remove();
  inq[index]=0;
  if(rt[index]>0)
  {
   seq.add(id[index]);
   if(rt[index]<=quantum)
   {
    t+=rt[index];
    rt[index]=0;
    complete++;
    tt[index] = t - at[index];
    wt[index] = tt[index] - bt[index];
    avgw+=wt[index];
```

```
   avgt+=tt[index];
  }
  else
  {
   rt[index]-=quantum;
   t+=quantum;
  }
  ti.add(t);
 }

 int i=index+1;
 while(i<pr)
 {
  if((at[i]<=t) && (rt[i]>0))
  {
   if(inq[i] == 0)
   {   q.add(i);
    inq[i]=1;
   }
  }
  if(at[i]>t)
   break;
  i++;
 }
 if(rt[index]!=0)
 {
  q.add(index);
 }
}

 System.out.println("\n\n\t*****************************************");
 System.out.println("\tID AT BT TAT WT");
 System.out.println("\t*****************************************");
 for(int  i = 0 ; i< pr;  i++)
 {
  System.out.println("\tP"+ id[i] + "  \t " + at[i] + "\t" + bt[i]  + "\t" + tt[i] + "\t"  + wt[i] ) ;
 }

 System.out.println("\n\tAverage turnaround time : "+(avgt/pr));
 System.out.println("\n\tAverage waiting time : "+ (avgw/pr));

 System.out.println("\n\tGantt chart : \n\t");

  for(int i=0;i<seq.size();i++)
     System.out.println("\t\t" + ti.get(i)+ "--->" + ti.get(i+1)+ ":P"+seq.get(i));


  System.out.println("\t========================");
}

static void priority()
{
 System.out.println("\n\n\t####### Priority (Non-Preemptive) #######");
```

```
  int pr,temp,ftime=0;
  float avgw=0,avgt=0;
  Scanner sc = new Scanner(System.in);
  System.out.print("\n\tEnter number of processes : ");
  pr = sc.nextInt();
  int at[],bt[],tt[],wt[],id[],p[],pt[];
  ArrayList<Integer> seq = new ArrayList<>();
  ArrayList<Integer> ti = new ArrayList<>();
  at = new int[pr]; bt = new int[pr]; tt = new int[pr];wt = new int[pr];  id= new int[pr]; p = new int[pr]; pt=new int[pr]
; // memory allocation

  for(int i=0;i<pr;i++)
  {
   System.out.print("\n\tArrival time of P"+(i+1)+" : ");
   at[i] = sc.nextInt();
   System.out.print("\n\tBurst time of P"+(i+1)+" : ");
   bt[i] = sc.nextInt();
   System.out.print("\n\tPriority of P"+(i+1)+"  : ");
   pt[i] = sc.nextInt();
   id[i]=i+1;
  }


  for(int i = 0 ; i <pr; i++)
  {
   for(int  j=0;  j < pr-(i+1) ; j++)
   {
    if( at[j] > at[j+1] )
    {
     temp = at[j];
     at[j] = at[j+1];
     at[j+1] = temp;
     temp = bt[j];
     bt[j] = bt[j+1];
     bt[j+1] = temp;
     temp = id[j];
     id[j] = id[j+1];
     id[j+1] = temp;
     temp = pt[j];
     pt[j] = pt[j+1];
     pt[j+1]=temp;
    }
   }
  }

  for (int i = 0; i < pr; i++)
        p[i] = pt[i];

  int index=0,min=Integer.MAX_VALUE;
  int complete=0,t=at[0]; ti.add(at[0]);

  while(complete!=pr)
  {
   for(int i=0;(i<pr) && (at[i]<=t);i++)
```

```
  {
   if( (pt[i]< min) && (p[i]!=-1) )
   {
    index = i;
    min = pt[i];
   }
  }

  seq.add(id[index]);
  complete++;
  t+=bt[index];
  ti.add(t);
  p[index]=-1;
  tt[index]= t - at[index];
  wt[index] = tt[index] - bt[index];

  avgw +=wt[index];
  avgt +=tt[index];

  min=Integer.MAX_VALUE;
 }

 System.out.println("\n\n\t*****************************************");
 System.out.println("\tID AT BT prior TAT WT");
 System.out.println("\t*****************************************");
 for(int  i = 0 ; i< pr;  i++)
 {
  System.out.println("\tP"+ id[i] + "  \t " + at[i] + "\t" + bt[i]  + "\t"+ pt[i]  + "\t" + tt[i] + "\t"  + wt[i] ) ;
 }

 System.out.println("\n\tAverage turnaround time : "+(avgt/pr));
 System.out.println("\n\tAverage waiting time : "+ (avgw/pr));

 System.out.println("\n\tGantt chart : \n\t");

  for(int i=0;i<seq.size();i++)
     System.out.println("\t\t" + ti.get(i)+ "--->"+ ti.get(i+1) + ":P"+seq.get(i));


  System.out.println("\t=========================");

}

public static void main(String[] args) {


 Scanner sc = new Scanner(System.in);
 int ch;
 String ans;
 System.out.println("\n\n\t####### Scheduling Algorithms #######");


 do
 {
```

```
  System.out.println("\n\n\t 1. FCFS" + "\n\t 2. SJF" + "\n\t 3. Round Robin" + "\n\t 4. Priority" + "\n\t 0. Exit");
  System.out.print("\n\tEnter Your Choice : ");
  ch = sc.nextInt();

  switch(ch)
  {
    case 1:
   fcfs();
   break;
    case 2:
   sjf();
   break;
    case 3:
   rrobin();
   break;
    case 4:
   priority();
   break;
    default:
   if(ch != 0)
   {
    System.out.println("\n\tInvalid Input");
   }
  }



 }while(ch!=0);


}

}
```

####### Scheduling Algorithms #######

1. FCFS
2. SJF
3. Round Robin
4. Priority
0. Exit

Enter Your Choice : 1

####### FCFS #######

Enter number of processes : 3

Arrival time of P1 : 0

Burst time of P1 : 6

Arrival time of P2 : 4

Burst time of P2 : 4

Arrival time of P3 : 3

Burst time of P3 : 2

****
ID AT BT TAT WT
****
P1 0 6 6 0
P3 3 2 5 3
P2 4 4 8 4

Average turnaround time : 6.3333335

Average waiting time : 2.3333333

Gantt chart :

0--->6 : P1
6--->8 : P3
8--->12 : P2
==========

1. FCFS
2. SJF
3. Round Robin
4. Priority

```
 0. Exit

Enter Your Choice : 2


####### SJF #######

Enter number of processes : 4

Arrival time of P1 : 0

Burst time of P1 : 4

Arrival time of P2 : 2

Burst time of P2 : 7

Arrival time of P3 : 3

Burst time of P3 : 2

Arrival time of P4 : 3

Burst time of P4 : 2


****************************************
ID AT BT TAT WT
****************************************
P1    0 4 4 0
P2    2 7 13 6
P3    3 2 3 1
P4    3 2 5 3

Average turnaround time : 6.25

Average waiting time : 2.5

Gantt chart :

 0--->4 : P1
 4--->6 : P3
 6--->8 : P4
 8--->15 : P2
========================


 1. FCFS
 2. SJF
 3. Round Robin
 4. Priority
 0. Exit

Enter Your Choice : 3
```

####### Round Robin (Preemptive) #######

Enter number of processes : 3

Enter the quantum : 2

Arrival time of P1 : 1

Burst time of P1 : 5

Arrival time of P2 : 0

Burst time of P2 : 4

Arrival time of P3 : 2

Burst time of P3 : 7

****
ID AT BT TAT WT
****
P2 0 4 8 4
P1 1 5 12 7
P3 2 7 14 7

Average turnaround time : 11.333333

Average waiting time : 6.0

Gantt chart :

0--->2:P2
2--->4:P1
4--->6:P3
6--->8:P2
8--->10:P1
10--->12:P3
12--->13:P1
13--->15:P3
15--->16:P3
==========

1. FCFS
2. SJF
3. Round Robin
4. Priority
0. Exit

Enter Your Choice : 4

####### Priority (Non-Preemptive) #######

Enter number of processes : 4

Arrival time of P1 : 0

Burst time of P1 : 4

Priority of P1 : 3

Arrival time of P2 : 1

Burst time of P2 : 2

Priority of P2 : 2

Arrival time of P3 : 2

Burst time of P3 : 3

Priority of P3 : 4

Arrival time of P4 : 4

Burst time of P4 : 2

Priority of P4 : 1

****
ID AT BT prior TAT WT
****
P1 0 4 3 4 0
P2 1 2 2 7 5
P3 2 3 4 9 6
P4 4 2 1 2 0

Average turnaround time : 5.5

Average waiting time : 2.75

Gantt chart :

0--->4:P1
4--->6:P4
6--->8:P2
8--->11:P3
==========

1. FCFS
2. SJF
3. Round Robin

4. Priority
0. Exit

Enter Your Choice : 0