21BCE7371
Radha Krishna Garg

OS-LAB 4 SCHEDULING ALGORITHMS

1. FCFS Scheduling Algorithms

CODE:

```
import java.util.*;
class FCFS {
public static void main(String args[])
{
Scanner sc = new Scanner(System.in);
System.out.println("enter no of process: ");
int n = sc.nextInt();
int pid[] = new int[n];     // process ids
int ar[] = new int[n];      // arrival times
int bt[] = new int[n];      // burst or execution times
int ct[] = new int[n];      // completion times
int ta[] = new int[n];      // turn around times
int wt[] = new int[n];      // waiting times
int temp;
float avgwt=0,avgta=0;

for(int i = 0; i < n; i++)
{
System.out.println("enter process " + (i+1) + " arrival time: ");
ar[i] = sc.nextInt();
System.out.println("enter process " + (i+1) + " brust time: ");
bt[i] = sc.nextInt();
pid[i] = i+1;
}

//sorting according to arrival times
for(int i = 0 ; i <n; i++)
{
for(int  j=0;  j < n-(i+1) ; j++)
{
if( ar[j] > ar[j+1] )
```

```
{
temp = ar[j];
ar[j] = ar[j+1];
ar[j+1] = temp;
temp = bt[j];
bt[j] = bt[j+1];
bt[j+1] = temp;
temp = pid[j];
pid[j] = pid[j+1];
pid[j+1] = temp;
}
}
}
// finding completion times
for(int  i = 0 ; i <= n; i++)
{
if( i == 0)
{
ct[i] = ar[i] + bt[i];
}
else
{
if( ar[i] > ct[i-1])
{
ct[i] = ar[i] + bt[i];
}
else
ct[i] = ct[i-1] + bt[i];
}
ta[i] = ct[i] - ar[i] ;           // turnaround time= completion time-
arrival time
wt[i] = ta[i] - bt[i] ;           // waiting time= turnaround time-
burst time
avgwt += wt[i] ;                  // total waiting time
avgta += ta[i] ;                  // total turnaround time
}
System.out.println("\npid  arrival  brust  complete turn waiting");
for(int  i = 0 ; i< n;  i++)
{
System.out.println(pid[i] + "   \t " + ar[i] + "\t" + bt[i] + "\t" +
ct[i] + "\t" + ta[i] + "\t"  + wt[i] ) ;
}
sc.close();
```

```
System.out.println("\naverage waiting time: "+ (avgwt/n));      //
printing average waiting time.
System.out.println("average turnaround time:"+(avgta/n));     //
printing average turnaround time.
}
}
```

OUTPUT

```
enter no of process:
3
enter process 1 arrival time:
0
enter process 1 brust time:
1
enter process 2 arrival time:
1
enter process 2 brust time:
2
enter process 3 arrival time:
2
enter process 3 brust time:
3

pid  arrival  brust  complete turn waiting
1       0      1       1       1       0
2       1      2       3       2       0
3       2      3       6       4       1

average waiting time: 0.33333334
average turnaround time:2.3333333
PS C:\Users\krish\Documents\PROJECTS_RK\Schedulling Algorithms (Operating Systems)>
```

2. SJF

CODE:

```
import java.util.*;

public class SJF {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        System.out.println("enter no of process:");
        int n = sc.nextInt();
        int pid[] = new int[n];
        int at[] = new int[n]; // at means arrival time
        int bt[] = new int[n]; // bt means burst time
        int ct[] = new int[n]; // ct means complete time
```

```
        int ta[] = new int[n]; // ta means turn around time
        int wt[] = new int[n]; // wt means waiting time
        int f[] = new int[n]; // f means it is flag it checks
process is completed or not
        int st = 0, tot = 0;
        float avgwt = 0, avgta = 0;

        for (int i = 0; i < n; i++) {
            System.out.println("enter process " + (i + 1) + "
arrival time:");
            at[i] = sc.nextInt();
            System.out.println("enter process " + (i + 1) + "
brust time:");
            bt[i] = sc.nextInt();
            pid[i] = i + 1;
            f[i] = 0;
        }
        boolean a = true;
        while (true) {
            int c = n, min = 999;
            if (tot == n) // total no of process = completed
process loop will be terminated
                break;
            for (int i = 0; i < n; i++) {
                /*
                 * If i'th process arrival time <= system time
and its flag=0 and burst<min
                 * That process will be executed first
                 */
                if ((at[i] <= st) && (f[i] == 0) && (bt[i] <
min)) {
                    min = bt[i];
                    c = i;
                }
            }
            /*
             * If c==n means c value can not updated because no
process arrival time< system
             * time so we increase the system time
             */
            if (c == n)
                st++;
            else {
```

```
                ct[c] = st + bt[c];
                st += bt[c];
                ta[c] = ct[c] - at[c];
                wt[c] = ta[c] - bt[c];
                f[c] = 1;
                tot++;
            }
        }
        System.out.println("\npid  arrival brust  complete turn
waiting");
        for (int i = 0; i < n; i++) {
            avgwt += wt[i];
            avgta += ta[i];
            System.out.println(pid[i] + "\t" + at[i] + "\t" +
bt[i] + "\t" + ct[i] + "\t" + ta[i] + "\t" + wt[i]);
        }
        System.out.println("\naverage tat is " + (float) (avgta /
n));
        System.out.println("average wt is " + (float) (avgwt /
n));
        sc.close();
    }
}
```

OUTPUT

```
enter no of process:
3
enter process 1 arrival time:
0
enter process 1 brust time:
1
enter process 2 arrival time:
0
enter process 2 brust time:
2
enter process 3 arrival time:
2
enter process 3 brust time:
2

pid  arrival brust  complete turn waiting
1       0       1       1       1       0
2       0       2       3       3       1
3       2       2       5       3       1

average tat is 2.3333333
average wt is 0.6666667
PS C:\Users\krish\Documents\PROJECTS_RK\Schedulling Algorithms (Operating Systems)>
```

3. PRIORITY SCHEDULING

CODE:

```
import java.util.Scanner;

public class PriorityScheduling {

    int burstTime[];
    int priority[];
    int arrivalTime[];
    String[] processId;
    int numberOfProcess;

    void getProcessData(Scanner input) {
        System.out.print("Enter the number of Process for
Scheduling          : ");
        int inputNumberOfProcess = input.nextInt();
        numberOfProcess = inputNumberOfProcess;
        burstTime = new int[numberOfProcess];
        priority = new int[numberOfProcess];
        arrivalTime = new int[numberOfProcess];
        processId = new String[numberOfProcess];
        String st = "P";
        for (int i = 0; i < numberOfProcess; i++) {
            processId[i] = st.concat(Integer.toString(i));
            System.out.print("Enter the burst time   for Process
- " + (i) + " : ");
            burstTime[i] = input.nextInt();
            System.out.print("Enter the arrival time for Process
- " + (i) + " : ");
            arrivalTime[i] = input.nextInt();
            System.out.print("Enter the priority     for Process
- " + (i) + " : ");
            priority[i] = input.nextInt();
        }
    }

    void sortAccordingArrivalTimeAndPriority(int[] at, int[] bt,
int[] prt, String[] pid) {

        int temp;
```

```
        String stemp;
        for (int i = 0; i < numberOfProcess; i++) {

            for (int j = 0; j < numberOfProcess - i - 1; j++) {
                if (at[j] > at[j + 1]) {
                    // swapping arrival time
                    temp = at[j];
                    at[j] = at[j + 1];
                    at[j + 1] = temp;

                    // swapping burst time
                    temp = bt[j];
                    bt[j] = bt[j + 1];
                    bt[j + 1] = temp;

                    // swapping priority
                    temp = prt[j];
                    prt[j] = prt[j + 1];
                    prt[j + 1] = temp;

                    // swapping process identity
                    stemp = pid[j];
                    pid[j] = pid[j + 1];
                    pid[j + 1] = stemp;

                }
                // sorting according to priority when arrival
timings are same
                if (at[j] == at[j + 1]) {
                    if (prt[j] > prt[j + 1]) {
                        // swapping arrival time
                        temp = at[j];
                        at[j] = at[j + 1];
                        at[j + 1] = temp;

                        // swapping burst time
                        temp = bt[j];
                        bt[j] = bt[j + 1];
                        bt[j + 1] = temp;

                        // swapping priority
                        temp = prt[j];
                        prt[j] = prt[j + 1];
```

```
                    prt[j + 1] = temp;

                    // swapping process identity
                    stemp = pid[j];
                    pid[j] = pid[j + 1];
                    pid[j + 1] = stemp;

                }
            }
        }

    }
}

    void priorityNonPreemptiveAlgorithm() {
        int finishTime[] = new int[numberOfProcess];
        int bt[] = burstTime.clone();
        int at[] = arrivalTime.clone();
        int prt[] = priority.clone();
        String pid[] = processId.clone();
        int waitingTime[] = new int[numberOfProcess];
        int turnAroundTime[] = new int[numberOfProcess];

        sortAccordingArrivalTimeAndPriority(at, bt, prt, pid);

        // calculating waiting & turn-around time for each
process
        finishTime[0] = at[0] + bt[0];
        turnAroundTime[0] = finishTime[0] - at[0];
        waitingTime[0] = turnAroundTime[0] - bt[0];

        for (int i = 1; i < numberOfProcess; i++) {
            finishTime[i] = bt[i] + finishTime[i - 1];
            turnAroundTime[i] = finishTime[i] - at[i];
            waitingTime[i] = turnAroundTime[i] - bt[i];
        }
        float sum = 0;
        for (int n : waitingTime) {
            sum += n;
        }
        float averageWaitingTime = sum / numberOfProcess;

        sum = 0;
```

```
        for (int n : turnAroundTime) {
            sum += n;
        }
        float averageTurnAroundTime = sum / numberOfProcess;

        // print on console the order of processes along with
their finish time & turn
        // around time
        System.out.println("Priority Scheduling Algorithm : ");
        System.out.format("%20s%20s%20s%20s%20s%20s%20s\n",
"ProcessId", "BurstTime", "ArrivalTime", "Priority",
                "FinishTime", "WaitingTime", "TurnAroundTime");
        for (int i = 0; i < numberOfProcess; i++) {
            System.out.format("%20s%20d%20d%20d%20d%20d%20d\n",
pid[i], bt[i], at[i], prt[i], finishTime[i],
                    waitingTime[i], turnAroundTime[i]);
        }

        System.out.format("%100s%20f%20f\n", "Average",
averageWaitingTime, averageTurnAroundTime);
    }

    public static void main(String[] args)
    {
        Scanner input = new Scanner(System.in);
        PriorityScheduling obj = new PriorityScheduling();
        obj.getProcessData(input);
        obj.priorityNonPreemptiveAlgorithm();
    }

}
```

OUTPUT

```
Enter the number of Process for Scheduling          : 3
Enter the burst time   for Process - 0 : 1
Enter the arrival time for Process - 0 : 0
Enter the priority     for Process - 0 : 3
Enter the burst time   for Process - 1 : 2
Enter the arrival time for Process - 1 : 1
Enter the priority     for Process - 1 : 2
Enter the burst time   for Process - 2 : 3
Enter the arrival time for Process - 2 : 0
Enter the priority     for Process - 2 : 1
Priority Scheduling Algorithm :
           ProcessId          BurstTime        ArrivalTime           Priority         FinishTime        WaitingTime      TurnAroundTime
                  P2                  3                  0                  1                  3                  0                  3
                  P0                  1                  0                  3                  4                  3                  4
                  P1                  2                  1                  2                  6                  3                  5
                                                                                       Average          2.000000          4.000000
PS C:\Users\krish\Documents\PROJECTS_RK\Schedulling Algorithms (Operating Systems)>
```

4. ROUND ROBIN

CODE:

```
import java.util.Scanner;

public class RoundRobin {
public static void main(String args[])
{
int n,i,qt,count=0,temp,sq=0,bt[],wt[],tat[],rem_bt[];
float awt=0,atat=0;
bt = new int[10];
wt = new int[10];
tat = new int[10];
rem_bt = new int[10];
Scanner s=new Scanner(System.in);
System.out.print("Enter the number of process (maximum 10) = ");
n = s.nextInt();
System.out.print("Enter the burst time of the process\n");
for (i=0;i<n;i++)
{
System.out.print("P"+i+" = ");
bt[i] = s.nextInt();
rem_bt[i] = bt[i];
}
System.out.print("Enter the quantum time: ");
qt = s.nextInt();
while(true)
{
for (i=0,count=0;i<n;i++)
{
temp = qt;
if(rem_bt[i] == 0)
{
count++;
continue;
}
if(rem_bt[i]>qt)
rem_bt[i]= rem_bt[i] - qt;
else
if(rem_bt[i]>=0)
{
```

```
temp = rem_bt[i];
rem_bt[i] = 0;
}
sq = sq + temp;
tat[i] = sq;
}
if(n == count)
break;
}
System.out.print("---------------------------------------------
-----------------------------------");
System.out.print("\nProcess\t      Burst Time\t       Turnaround
Time\t          Waiting Time\n");
System.out.print("---------------------------------------------
-----------------------------------");
for(i=0;i<n;i++)
{
wt[i]=tat[i]-bt[i];
awt=awt+wt[i];
atat=atat+tat[i];
System.out.print("\n "+(i+1)+"\t "+bt[i]+"\t\t "+tat[i]+"\t\t
"+wt[i]+"\n");
}
awt=awt/n;
atat=atat/n;
System.out.println("\nAverage waiting Time = "+awt+"\n");
System.out.println("Average turnaround time = "+atat);
}
}
```

OUTPUT

```
P0 = 3
P1 = 2
P2 = 6
Enter the quantum time: 2
-------------------------------------------------------------------------------
Process       Burst Time              Turnaround Time           Waiting Time
-------------------------------------------------------------------------------
 1       3              7              4

 2       2              4              2

 3       6              11             5

Average waiting Time = 3.6666667

Average turnaround time = 7.3333335
PS C:\Users\krish\Documents\PROJECTS_RK\Schedulling Algorithms (Operating Systems)>
```