DHA SUFFA UNIVERSITY
Department of Computer Science
CS-2004L
Operating Systems
Fall 2020

LAB 06 – SJF And SRTF Scheduling Algorithm

OBJECTIVE(S)

Understanding implementation of SJF and SRTF scheduling algorithms.

Shortest Job First Scheduling Algorithm

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.

- Shortest Job first has the advantage of having minimum average waiting time among all scheduling algorithms.
- It is a Greedy Algorithm.
- It may cause starvation if shorter processes keep coming. This problem can be solved using the concept of aging.

Aim

Write a C++ Program to Input processes from user and run them on SJF basis.

Algorithm

Step 1: Start the process

Step 2: Accept the number of processes in the ready Queue

Step 3: For each process in the ready Q, assign the process id and accept the CPU burst time

Step 4: Start the Ready Q according the shortest Burst time by sorting according to lowest to highest burst time.

Step 5: Set the waiting time of the first process as '0' and its turnaround time as its burst time.

Step 6: For each process in the ready queue, calculate

Turn-around time (TaT) of Process(n) = Completion Time – Arrival Time

Waiting Time for process(n) = Turn-around time – Burst Time

Step 7: Calculate

Average waiting time = Total waiting Time / Number of process

Average Turnaround time = Total Turnaround Time / Number of process

Step 8: Stop the process

Shortest Job First Scheduling Example

Example: Consider the following table:

Process no.	Arrival Time	Burst Time
P1	0	6
P2	2	1
P3	5	4
P4	6	3

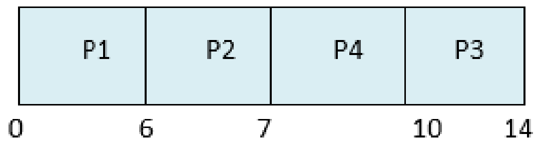
Find the average waiting time and average turnaround time using SJF (Shortest Job First) algorithm?

Solution

Using shortest job first algorithm, ready queue and Gantt chart are:

Ready Queue: P1, P2, P4, P3

Gantt chart



Therefore,

Process No.	Arrival Time	Burst Time	Completion Time	Turnaround Time	Waiting Time
P1	0	6	6	6 - 0 = 6	**6 - 6 = 0**
P2	2	1	7	7 - 2 = 5	**5 - 1 = 4**
P3	5	4	14	14 - 5 = 9	**9 - 4 = 5**
P4	6	3	10	10 - 6 = 4	**4 - 3 = 1**

So,

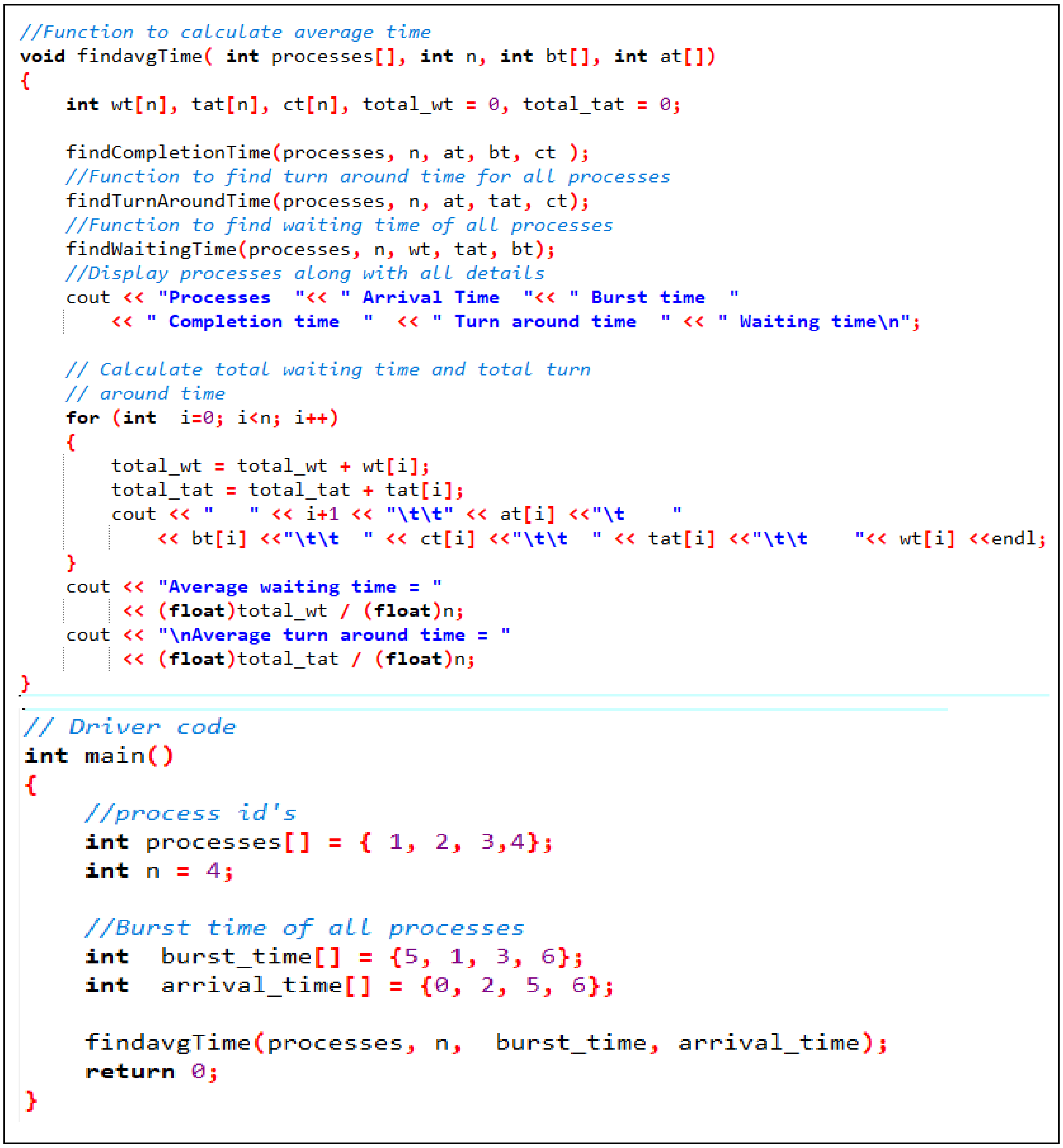
Average Turnaround time = (6 + 5 + 9 + 4) / (4) = 6

Average Waiting time = (0 + 4 + 5 + 1) / (4) = 2.5

CODE:

```
// C++ program to implement Shortest Job first with Arrival Time
#include<iostream>
using namespace std;

void swap(int *xp, int *yp)
{
    int temp = *xp;
    *xp = *yp;
    *yp = temp;
}
```

```
void findCompletionTime(int process[], int n, int at[], int bt[], int ct[])
{
    int i, j;
    for (i = 0; i < n-1; i++)
    {
    // Last i elements are already in place
        for (j = 0; j < n-i-1; j++)
        {
            if ((at[j] > at[j+1])&&(bt[j] > bt[j+1]))
            {
                swap(&at[j], &at[j+1]);
                swap(&bt[j], &bt[j+1]);
            }

        }
    }
    ct[0]=bt[0];
    for (int i = 1; i < n; ++i){
        ct[i] =  ct[i-1] + bt[i] ;
    }
}
void findTurnAroundTime( int processes[], int n, int at[], int tat[], int ct[])
{

    for (int  i = 0; i < n ; i++)
        tat[i] = ct[i] - at[i];
}
void findWaitingTime(int processes[], int n, int wt[],
                                     int tat[], int bt[])
{
    // waiting time for first process is 0
    wt[0] = 0;

    // calculating waiting time
    for (int  i = 1; i < n ; i++ )
        wt[i] =  tat[i] - bt[i] ;
}
```



```
//Function to calculate average time
void findavgTime( int processes[], int n, int bt[], int at[])
{
    int wt[n], tat[n], ct[n], total_wt = 0, total_tat = 0;

    findCompletionTime(processes, n, at, bt, ct );
    //Function to find turn around time for all processes
    findTurnAroundTime(processes, n, at, tat, ct);
    //Function to find waiting time of all processes
    findWaitingTime(processes, n, wt, tat, bt);
    //Display processes along with all details
    cout << "Processes  "<< " Arrival Time  "<< " Burst time  "
        << " Completion time  " << " Turn around time  " << " Waiting time\n";

    // Calculate total waiting time and total turn
    // around time
    for (int  i=0; i<n; i++)
    {
        total_wt = total_wt + wt[i];
        total_tat = total_tat + tat[i];
        cout << "   " << i+1 << "\t\t" << at[i] <<"\t    "
            << bt[i] <<"\t\t  " << ct[i] <<"\t\t  " << tat[i] <<"\t\t    "<< wt[i] <<endl;
    }
    cout << "Average waiting time = "
        << (float)total_wt / (float)n;
    cout << "\nAverage turn around time = "
        << (float)total_tat / (float)n;
}

// Driver code
int main()
{
    //process id's
    int processes[] = { 1, 2, 3,4};
    int n = 4;

    //Burst time of all processes
    int  burst_time[] = {5, 1, 3, 6};
    int  arrival_time[] = {0, 2, 5, 6};

    findavgTime(processes, n,  burst_time, arrival_time);
    return 0;
}
```

DHA SUFFA UNIVERSITY
Department of Computer Science
CS-2004L
Operating Systems
Fall 2020

Shortest Remaining Time First Scheduling Algorithm

SRTF, which stands for **Shortest Remaining Time First** is a scheduling algorithm used in Operating Systems, which can also be called as the preemptive version of the SJF scheduling algorithm. The process which has the least processing time remaining is executed first or the processor is allocated to the job closest to completion. SRTF algorithm makes the processing of the jobs faster than SJF scheduling algorithm.

Aim

Write a C++ Program to Input processes from user and run them on SRTF basis.

Algorithm

Step 1: Start the process

Step 2: Accept the number of processes in the ready Queue

Step 3: For each process in the ready Q, assign the process id and accept the CPU burst time and process arrival time.

Step 4: Compute the completion time of processes.

Step 5: For each process in the Ready Q calculate

Turn-around time (TaT) of Process(n) = Completion Time – Arrival Time

Waiting Time for Process(n) = Turn-around time – Burst Time

Step 6: Calculate

Average waiting time = Total (sum) waiting Time / Number of processes

Average Turnaround time = Total (sum) Turnaround Time / Number of processes

Step 7: Stop the process.

Shortest Remaining Time First Scheduling Example

Example: Consider the following table:

Process ID	Arrival Time (milliseconds)	Burst Time (milliseconds)
P1	0	8
P2	1	2
P3	4	3

Find the average waiting time and average turnaround time using SRTF (Shortest Remaining Time First) algorithm.

Solution

Using Shortest Remaining Time First algorithm, the Gantt chart is:

0 P1	1 P2	2 P2	3 P1	4 P3	5 P3	6 P3	7 P1	8 P1	9 P1	10 P1	11 P1	12 P1 13

Therefore,

P ID	Arrival Time	Burst Time	Completion time (milliseconds)	Turn Around Time (milliseconds)	Waiting Time (milliseconds)
P1	0	8	13	13	5
P2	1	2	3	2	0
P3	4	3	7	3	0

Average Turn Around Time = Total Turn Around Time / Total No. of Processes

= (13 + 2 + 3) / 3

= 6

Average Waiting Time = Total Waiting Time / Total No. of Processes

= (5 + 0 + 0) / 3

= 1.67

Explanation:

- At the 0th unit of the CPU, we have only process P1, so it gets executed for the 1-time unit.
- At the 1st unit of the CPU, the Process P2 also arrives. Now, the P1 needs 7 more units to be executed, and P2 needs only 2 units. So, P2 is executed by preempting P1.
- P2 gets completed at time unit 3, and now no new process has arrived. So, after the completion of P2, again P1 is sent for execution.
- Now, P1 has been executed for one unit only, and we have the arrival of a new process P3 at time unit 4. Now, the P1 needs 6-time units more and P3 needs only 3-time units. So, P3 is executed by preempting P1.
- P3 gets completed at time unit 7, and after that, we have the arrival of no other process. So again, P1 is sent for execution, and it gets completed at 13th unit.

DHA SUFFA UNIVERSITY
Department of Computer Science
CS-2004L
Operating Systems
Fall 2020

Assignment # 6

Q.1 Write a program to implement random PCB Generation with a record of Arrival Time of a Process, show queue processing through Heap.

Q.2 Write a C++ program to implement preemptive version of SJF scheduling algorithm.