CSB 302: Operating System

Lab2: Job Scheduling Problem

Submitted By:

Name: **UMANG KUMAR**

Roll No: **201210051**

Branch: **CSE**

Semester: **5th Sem**

Submitted To: ***Dr. Rishav Singh***

Release Date: 29/08/2022 **Submitted Date:** 04/09/2022

NATIONAL INSTITUTE OF TECHNOLOGY DELHI

Department of Computer Science and Engineering

2022

1. Write a program to calculate average waiting time and average turnaround time using First Come First Serve Scheduling algorithm.

Code:

```
#include <bits/stdc++.h>
using namespace std;

int main() {
   int n;
   cout << "Number of entries: ";
   cin >> n;

   vector<vector<int> > data(n, vector<int>(3));
   cout << "Process\t Aval. Time \t Burst Time\n";
   for (int i = 0; i < n; i++) {
       cout << "P" << i + 1 << ": ";
       cin >> data[i][0];
       cin >> data[i][1];

       data[i][2] = i + 1;
   }
   cout << "\n\n";

   // sort by available time
   sort(data.begin(), data.end());

   int time = 0;
   double total_waiting_time = 0;
   double total_turnaround_time = 0;

   cout << "Process Description:\n";
   cout << "Process\t\tAval. T.\tBurst T.\tWaiting T.\tTurnaround T.\n";
   for (int i = 0; i < n; i++) {

       cout << "P" << data[i][2] << "\t\t";
       cout << data[i][0] << "\t\t";
       cout << data[i][1] << "\t\t";

       // update time
       time = max(time, data[i][0]);

       // print waiting and Turnaround time
```

```
        cout << time - data[i][0] << "\t\t";
        cout << time - data[i][0] + data[i][1] << "\t\t";

        // add to total waiting time and turnaround time
        total_waiting_time += time - data[i][0];
        total_turnaround_time += time - data[i][0] + data[i][1];

        // update time
        time += data[i][1];
        cout << "\n";
    }

    cout << "\n";
    cout << "Average Waiting Time: " << total_waiting_time / n << "\n";
    cout << "Average Turnaround Time: " << total_turnaround_time / n << "\n";
}
```

Output:

```
Number of entries: 5
Process  Aval. Time      Burst Time
P1: 0 8
P2: 4 3
P3: 6 2
P4: 2 4
P5: 5 3


Process Description:
Process         Aval. T.        Burst T.        Waiting T.      Turnaround T.
P1              0               8               0               8
P4              2               4               6               10
P2              4               3               8               11
P5              5               3               10              13
P3              6               2               12              14

Average Waiting Time: 7.2
Average Turnaround Time: 11.2
```

2. Write a program to calculate average waiting time and average turnaround time using Shortest Job First Scheduling algorithm.

Code:

```
#include <bits/stdc++.h>
using namespace std;

// fetch next job
int findNextJob(vector<int> done, vector<vector<int>> data, int n, int time) {
   // find the job which has already arrived and has minimum burst time
   for (int i = 0; i < n; i++) {
       if (find(done.begin(), done.end(), data[i][2]) == done.end()) {
           if (time >= data[i][1]) return data[i][2];
       }
   }

   // find job which will arrive first but have not arrived yet
   int indx = -1;
   int aval_time = INT_MAX;
   for (int i = 0; i < n; i++) {
       if (find(done.begin(), done.end(), data[i][2]) == done.end()) {
           if (aval_time > data[i][1]) {
               aval_time = data[i][1];
               indx = data[i][2];
           }
       }
   }

   // return id of the process/Job
   return indx;
}

int main() {
   cout << "Enter no. of Process: ";
   int n;
   cin >> n;

   vector<int> done;
   vector<vector<int>> data(n, vector<int>(3));

   cout << "Process\t Aval. Time \t Burst Time\n";
   for (int i = 0; i < n; i++) {
```

```
        cout << "P" << i + 1 << ": ";
        cin >> data[i][1] >> data[i][0];
        data[i][2] = i;
    }
    cout << "\n\n";

    // sort by burst time
    sort(data.begin(), data.end());

    int time = 0;
    double total_waiting_time = 0;
    double total_turnaround_time = 0;

    int cnt = 0;
    cout << "Process Description:\n";
    cout << "Process\t\tAval. T.\tBurst T.\tWaiting T.\tTurnaround T.\n";
    while (cnt < n) {
        // find next job
        int indx = findNextJob(done, data, n, time);

        // find the index of next job
        int j;
        for (int i = 0; i < n; i++) {
            if (data[i][2] == indx) {
                j = i;
            }
        }

        // print details
        time = max(time, data[j][1]);
        cout << "P" << data[j][2] + 1 << "\t\t";
        cout << data[j][1] << "\t\t";
        cout << data[j][0] << "\t\t";

        cout << time - data[j][1] << "\t\t";
        cout << time - data[j][1] + data[j][0] << "\t\t";

        // add current job waiting time to total waiting time and turnaround
        // time
        total_waiting_time += time - data[j][1];
        total_turnaround_time += time - data[j][1] + data[j][0];
        time += data[j][0];
```

```
        cout << "\n";

        cnt++;
        done.push_back(indx);
    }

    cout << "\n";
    cout << "Average Waiting Time: " << total_waiting_time / n << "\n";
    cout << "Average Turnaround Time: " << total_turnaround_time / n << "\n";
}
```

Output:

```
Enter no. of Process: 5
Process  Aval. Time      Burst Time
P1: 2 6
P2: 5 2
P3: 1 8
P4: 0 3
P5: 4 4


Process Description:
Process         Aval. T.        Burst T.        Waiting T.      Turnaround T.
P4              0               3               0               3
P1              2               6               1               7
P2              5               2               4               6
P5              4               4               7               11
P3              1               8               14              22

Average Waiting Time: 5.2
Average Turnaround Time: 9.8
```

3. Write a program to calculate average waiting time and average turnaround time using Priority Scheduling algorithm.

Code:

```
#include <bits/stdc++.h>
using namespace std;

// fetch next job
int findNextJob(vector<int> done, vector<vector<int>> data, int n, int time) {
    // find the job which has already arrived and has minimum burst time
    for (int i = 0; i < n; i++) {
        if (find(done.begin(), done.end(), data[i][3]) == done.end()) {
            if (time >= data[i][2]) return data[i][3];
        }
    }

    // find job which will arrive first but have not arrived yet
    int indx = -1;
    int aval_time = INT_MAX;
    for (int i = 0; i < n; i++) {
        if (find(done.begin(), done.end(), data[i][3]) == done.end()) {
            if (aval_time > data[i][2]) {
                aval_time = data[i][2];
                indx = data[i][3];
            }
        }
    }

    // return id of the process/Job
    return indx;
}

int main() {
    cout << "Enter no. of Process: ";
    int n;
    cin >> n;

    vector<int> done;
    vector<vector<int>> data(n, vector<int>(4));

    cout << "Process\t Aval. Time \t Burst Time \t Priority\n";
    for (int i = 0; i < n; i++) {
```

```
        cout << "P" << i + 1 << ": ";
        cin >> data[i][2] >> data[i][1] >> data[i][0];
        data[i][3] = i;
    }
    cout << "\n\n";

    // sort by burst time
    sort(data.begin(), data.end(), [&](auto a, auto b) {
        if (a[0] < b[0]) return true;
        if (a[0] == b[0] && a[2] <= b[2]) return true;
        return false;
    });

    int time = 0;
    double total_waiting_time = 0;
    double total_turnaround_time = 0;

    int cnt = 0;
    cout << "Process Description:\n";
    cout << "Process\t\tAval. T.\tBurst T.\tPriority\tWaiting T.\tTurnaround "
            "T.\n";
    while (cnt < n) {

        // find next job
        int indx = findNextJob(done, data, n, time);

        // find the index of next job
        int j;
        for (int i = 0; i < n; i++) {
            if (data[i][3] == indx) {
                j = i;
            }
        }

        // print details
        time = max(time, data[j][2]);
        cout << "P" << data[j][3] + 1 << "\t\t";
        cout << data[j][2] << "\t\t";
        cout << data[j][1] << "\t\t";
        cout << data[j][0] << "\t\t";

        cout << time - data[j][2] << "\t\t";
```

```
        cout << time - data[j][2] + data[j][1] << "\t\t";

        // add current job waiting time to total waiting time and turnaround time
        total_waiting_time += time - data[j][2];
        total_turnaround_time += time - data[j][2] + data[j][1];
        time += data[j][1];
        cout << "\n";

        cnt++;
        done.push_back(indx);
    }

    cout << "\n";
    cout << "Average Waiting Time: " << total_waiting_time / n << "\n";
    cout << "Average Turnaround Time: " << total_turnaround_time / n << "\n";
}
```

Output:

```
Enter no. of Process: 5
Process  Aval. Time      Burst Time      Priority
P1: 1 3 3
P2: 2 1 4
P3: 3 5 5
P4: 4 2 5
P5: 0 4 2


Process Description:
Process         Aval. T.        Burst T.        Priority        Waiting T.      Turnaround
T.
P5              0               4               2               0               4
P1              1               3               3               3               6
P2              2               1               4               5               6
P3              3               5               5               5               10
P4              4               2               5               9               11

Average Waiting Time: 4.4
Average Turnaround Time: 7.4
```