UNIVERSITY OF ASIA PACIFIC

Course Title: Operating System Lab
Course Code: CSE 406

Date of Submission: 21 October,2025

Submitted by:	**Submitted to:**
Name: Md.Sarjil Hasan	Atia Rahman Orthi
Reg: 22101168	Lecturer
Sec : D	University of Asia Pacific

Shortest Job First (SJF) CPU Scheduling Algorithm:

Problem Statement:

Given a set of processes, each with a specific arrival time and burst time, design a scheduling algorithm using the Shortest Job First (SJF) Non-Preemptive Scheduling Algorithm. The goal is to compute for each process:

Process ID	Arrival Time	Burst Time
P1	0	3
P2	1	2
P3	2	1
P4	3	4
P5	0	2

Completion Time (CT)

Turnaround Time (TAT) = CT – Arrival Time(AT)

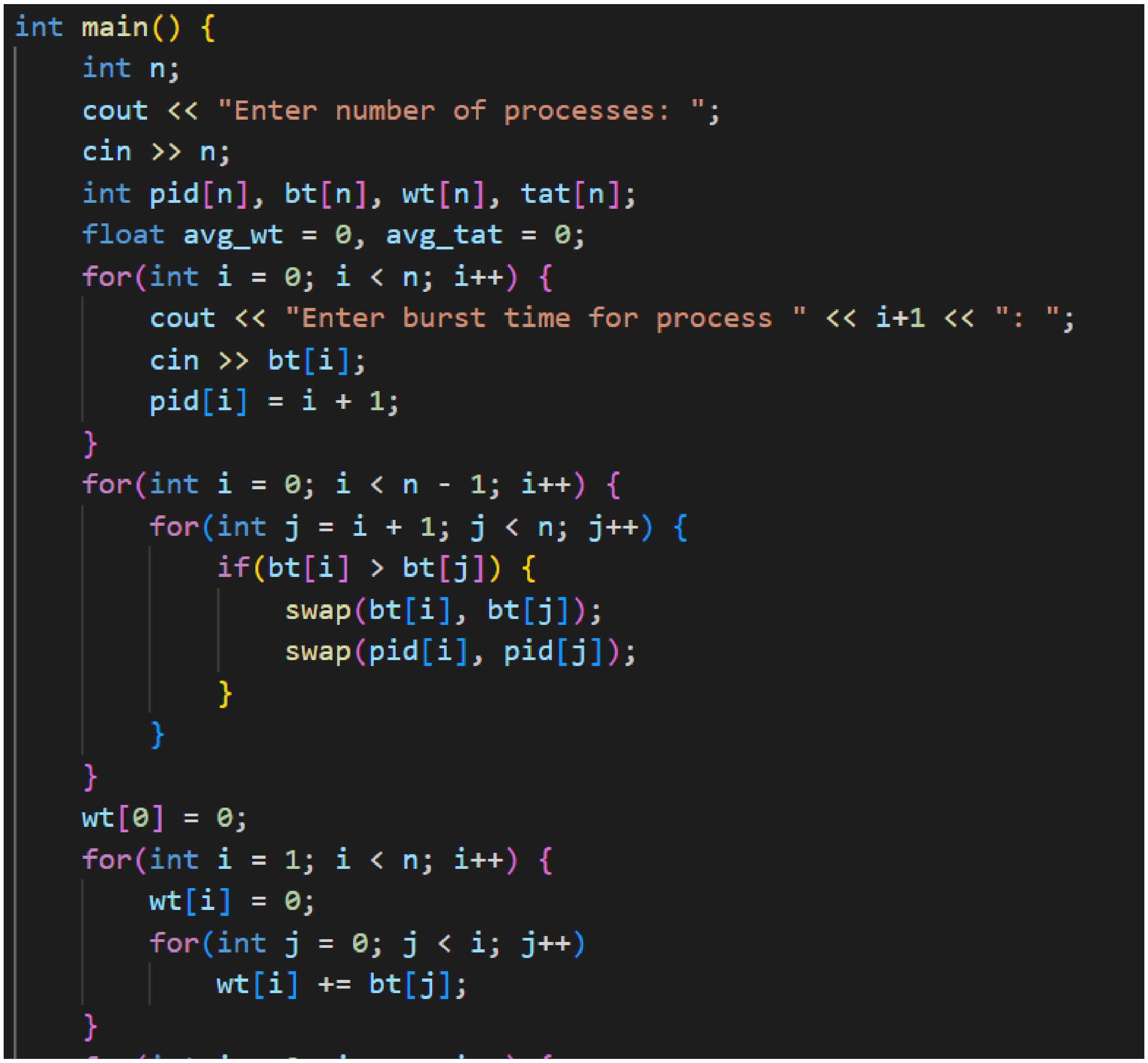
Waiting Time (WT) = TAT – Burst Time(BT)

The algorithm should also compute and display the average waiting time and average turnaround time. Processes should be executed in the order of their arrival time

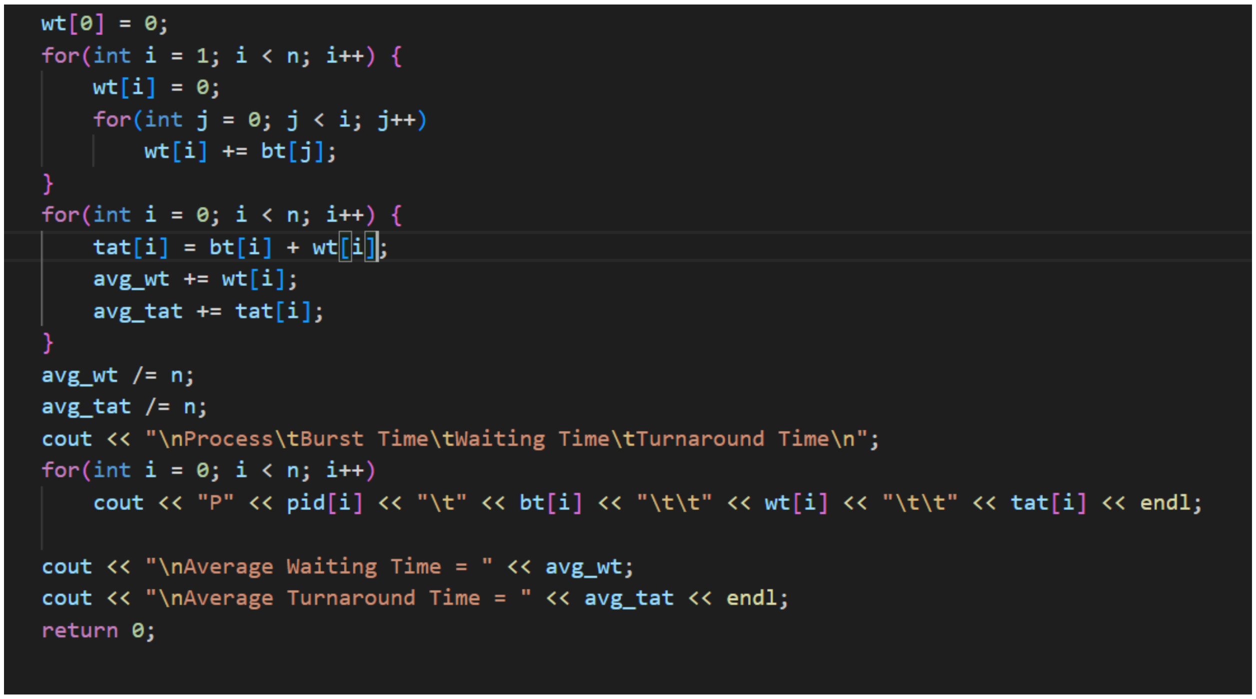
Objective:

The main objective of this experiment is to implement and understand the Shortest Job First (SJF) CPU scheduling algorithm.
SJF scheduling aims to reduce the average waiting time and turnaround time by selecting the process with the smallest burst time to execute next.

Code:

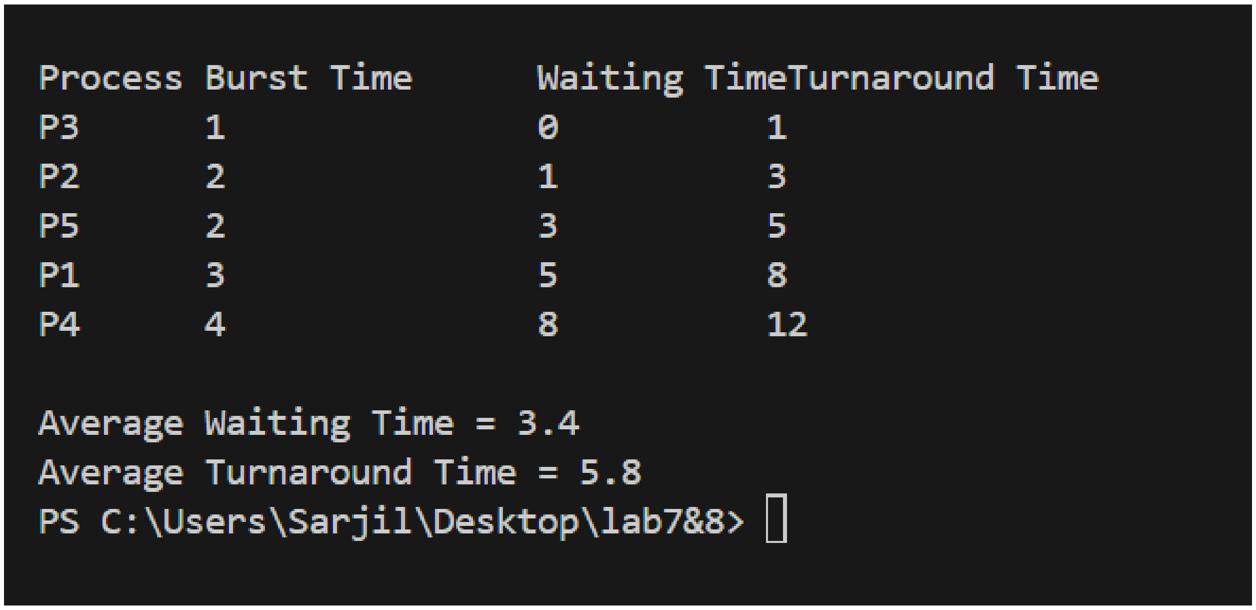


```cpp
int main() {
    int n;
    cout << "Enter number of processes: ";
    cin >> n;
    int pid[n], bt[n], wt[n], tat[n];
    float avg_wt = 0, avg_tat = 0;
    for(int i = 0; i < n; i++) {
        cout << "Enter burst time for process " << i+1 << ": ";
        cin >> bt[i];
        pid[i] = i + 1;
    }
    for(int i = 0; i < n - 1; i++) {
        for(int j = i + 1; j < n; j++) {
            if(bt[i] > bt[j]) {
                swap(bt[i], bt[j]);
                swap(pid[i], pid[j]);
            }
        }
    }
    wt[0] = 0;
    for(int i = 1; i < n; i++) {
        wt[i] = 0;
        for(int j = 0; j < i; j++)
            wt[i] += bt[j];
    }
```



```cpp
wt[0] = 0;
for(int i = 1; i < n; i++) {
    wt[i] = 0;
    for(int j = 0; j < i; j++)
        wt[i] += bt[j];
}
for(int i = 0; i < n; i++) {
    tat[i] = bt[i] + wt[i];
    avg_wt += wt[i];
    avg_tat += tat[i];
}
avg_wt /= n;
avg_tat /= n;
cout << "\nProcess\tBurst Time\tWaiting Time\tTurnaround Time\n";
for(int i = 0; i < n; i++)
    cout << "P" << pid[i] << "\t" << bt[i] << "\t\t" << wt[i] << "\t\t" << tat[i] << endl;

cout << "\nAverage Waiting Time = " << avg_wt;
cout << "\nAverage Turnaround Time = " << avg_tat << endl;
return 0;
```



```
Process Burst Time      Waiting TimeTurnaround Time
P3      1               0          1
P2      2               1          3
P5      2               3          5
P1      3               5          8
P4      4               8          12

Average Waiting Time = 3.4
Average Turnaround Time = 5.8
PS C:\Users\Sarjil\Desktop\lab7&8> 
```

Discussion:

Here,the Shortest Job First (SJF) algorithm was implemented, which is a CPU Scheduling method which selects the process with shortest burst time from a given set of processes. This algorithm aims to minimize average waiting and turnaround time by prioritizing shorter tasks. This algorithm follows a non-preemptive approach,which means,once a process starts execution, it will run to completion without interruption. If no processes are available currently, the CPU remains idle. Some advantages and disadvantages of SJF are:

Advantages:

- Simple logic: prioritizes processes with shortest burst time.
- Minimizes average waiting time by selecting shortest tasks first.
- Improves system throughput:many short processes get executed quickly.

Disadvantages:

- Difficult to predict burst time: exact execution time is often not known in actual systems.
- If short processes keep arriving, long processes keep getting delayed indefinitely.
- Non-preemptive, if a process starts,it runs to completion even if a shorter process arrives.

Conclusion:

SJF is a CPU scheduling algorithm that reduces average waiting and turnaround times by prioritizing shorter processes;therefore it is more efficient than most non preemptive methods. It requires knowing the burst time in advance and its practical use gets limited as there is a risk of starving the longer processes.

GitHub link:

https://github.com/Sarjil-SarZzz/CSE406-LAB/blob/main/sjf.cpp