PRACTICAL FILE

MODELING AND SIMULATION LAB

(CS 603)

BE CSE 6TH SEM

(GROUP-4)

University Institute of Engineering and Technology (UIET), Panjab University, Chandigarh, India- 160014

Under the guidance of

Priyanka Mam

Department of Computer Science and Engineering

Submitted By

Ojas Arora

Roll No: UE223073

Practical 8

Aim

To simulate code for Shortest Remaining Time First (SRTF) Scheduling Algorithm.

Introduction to Shortest Remaining Time First (SRTF) Algorithm

The **Shortest Remaining Time First (SRTF)** is a preemptive version of the **Shortest Job First (SJF)** algorithm. It selects the process with the shortest remaining CPU burst time for execution.

Key Characteristics of SRTF:

- It continuously checks the CPU burst length of arriving processes and preempts the current process if a shorter process arrives.
- It minimizes the average waiting time, making it an optimal scheduling algorithm in theory.
- The main challenge is predicting the exact CPU burst time for each process.

Working of SRTF:

- If no process is running, the CPU selects the process with the shortest burst time.
- If a new process arrives with a burst time shorter than the remaining time of the current process, the CPU switches to the new process.
- The cycle continues until all processes are executed.

Since SRTF is preemptive, it ensures that shorter tasks do not get delayed by longer ones, leading to better responsiveness in multitasking environments.

Advantages of SRTF:

- Ensures **shorter processes** are executed quickly.
- **Reduces average waiting time** compared to other scheduling algorithms.
- More efficient in **time-sharing** and **interactive systems**.

Disadvantages of SRTF:

- **Difficult to implement** as predicting exact burst times is complex.
- Can cause **starvation** for longer processes, as new short processes may continuously preempt them.
- Frequent **context switching** leads to increased overhead.

Code for Simulating Shortest Remaining Time First Algorithm

```
processes = [1 2 3 4 5];
arrival_time = [0 1 2 3 4];
burst_time = [6 1 2 3 4];

n = length(processes);
remaining_time = burst_time;
completion_time = zeros(1, n);
waiting_time = zeros(1, n);
turnaround_time = zeros(1, n);

time = 0;
completed = 0;
gantt = [];
current_process = -1;

while completed < n
    min_time = inf;
    selected_process = -1;

    for i = 1:n
        if arrival_time(i) <= time && remaining_time(i) > 0
            if remaining_time(i) < min_time
                min_time = remaining_time(i);
                selected_process = i;
            end
        end
    end

    if selected_process == -1
        gantt = [gantt; -1, time, time + 1];
        time = time + 1;
    else
        if current_process ~= selected_process
            gantt = [gantt; selected_process, time, time + 1];
            current_process = selected_process;
        else
            gantt(end, 3) = time + 1;
        end
        remaining_time(selected_process) = remaining_time(selected_process) - 1;
        time = time + 1;

        if remaining_time(selected_process) == 0
            completion_time(selected_process) = time;
            completed = completed + 1;
        end
    end
end

for i = 1:n
    turnaround_time(i) = completion_time(i) - arrival_time(i);
    waiting_time(i) = turnaround_time(i) - burst_time(i);
end

avg_tat = mean(turnaround_time);
avg_wt = mean(waiting_time);

fprintf('\n------------------------------------------------\n');
fprintf('| %-8s | %-4s | %-4s | %-4s | %-4s | %-4s |\n', 'Process', 'AT', 'BT', 'CT', 'TAT', 'WT');
fprintf('------------------------------------------------\n');
for i = 1:n
    fprintf('| %-8s | %-4d | %-4d | %-4d | %-4d | %-4d |\n', ['P' num2str(i)], arrival_time(i), burst_time(i), completion_time(i), turnaround_time(i), waiting_time(i));
end
fprintf('------------------------------------------------\n');
fprintf('Average Turnaround Time: %.2f\n', avg_tat);
fprintf('Average Waiting Time: %.2f\n', avg_wt);

figure;
for i = 1:size(gantt,1)
    if gantt(i,1) ~= -1
        rectangle('Position', [gantt(i,2), 0, gantt(i,3)-gantt(i,2), 1], 'FaceColor', [0.8 0.9 0.9]);
        text((gantt(i,2) + gantt(i,3)) / 2, 0.5, ['P' num2str(gantt(i,1))], 'HorizontalAlignment', 'center', 'FontSize', 10);
    end
end
for i = 1:size(gantt,1)
    line([gantt(i,2) gantt(i,2)], [0 1], 'Color', 'k', 'LineWidth', 1);
end
line([gantt(end,3) gantt(end,3)], [0 1], 'Color', 'k', 'LineWidth', 1);
line([gantt(1,2) gantt(end,3)], [0 0], 'Color', 'k', 'LineWidth', 1);
line([gantt(1,2) gantt(end,3)], [1 1], 'Color', 'k', 'LineWidth', 1);
for i = 1:size(gantt,1)
    text(gantt(i,2), -0.2, num2str(gantt(i,2)), 'HorizontalAlignment', 'center', 'FontSize', 8);
end
text(gantt(end,3), -0.2, num2str(gantt(end,3)), 'HorizontalAlignment', 'center', 'FontSize', 8);
title('Gantt Chart for Shortest Remaining Time First');
xlabel('Time');
ylim([-0.5 1.5]);
axis off;
```

Output

```
>> SRTF

-----------------------------------------------
| Process  | AT   | BT   | CT   | TAT  | WT   |
-----------------------------------------------
| P1       | 0    | 6    | 16   | 16   | 10   |
| P2       | 1    | 1    | 2    | 1    | 0    |
| P3       | 2    | 2    | 4    | 2    | 0    |
| P4       | 3    | 3    | 7    | 4    | 1    |
| P5       | 4    | 4    | 11   | 7    | 3    |
-----------------------------------------------
```

```
Average Turnaround Time: 6.00
Average Waiting Time: 2.80
```

Gantt Chart for Shortest Remaining Time First