ĐẠI HỌC QUỐC GIA TPHCM

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA KỸ THUẬT MÁY TÍNH

HỆ ĐIỀU HÀNH

Báo cáo thực hành LAB 4

Môn học: Hệ điều hành

Sinh viên thực hiện:

- Đặng Quốc Cường
- MSSV : 23520192
- Lớp : IT007.P11.CTTN

Giảng viên hướng dẫn:
Thân Thế Tùng

Ngày 10 tháng 11 năm 2024

Mục lục

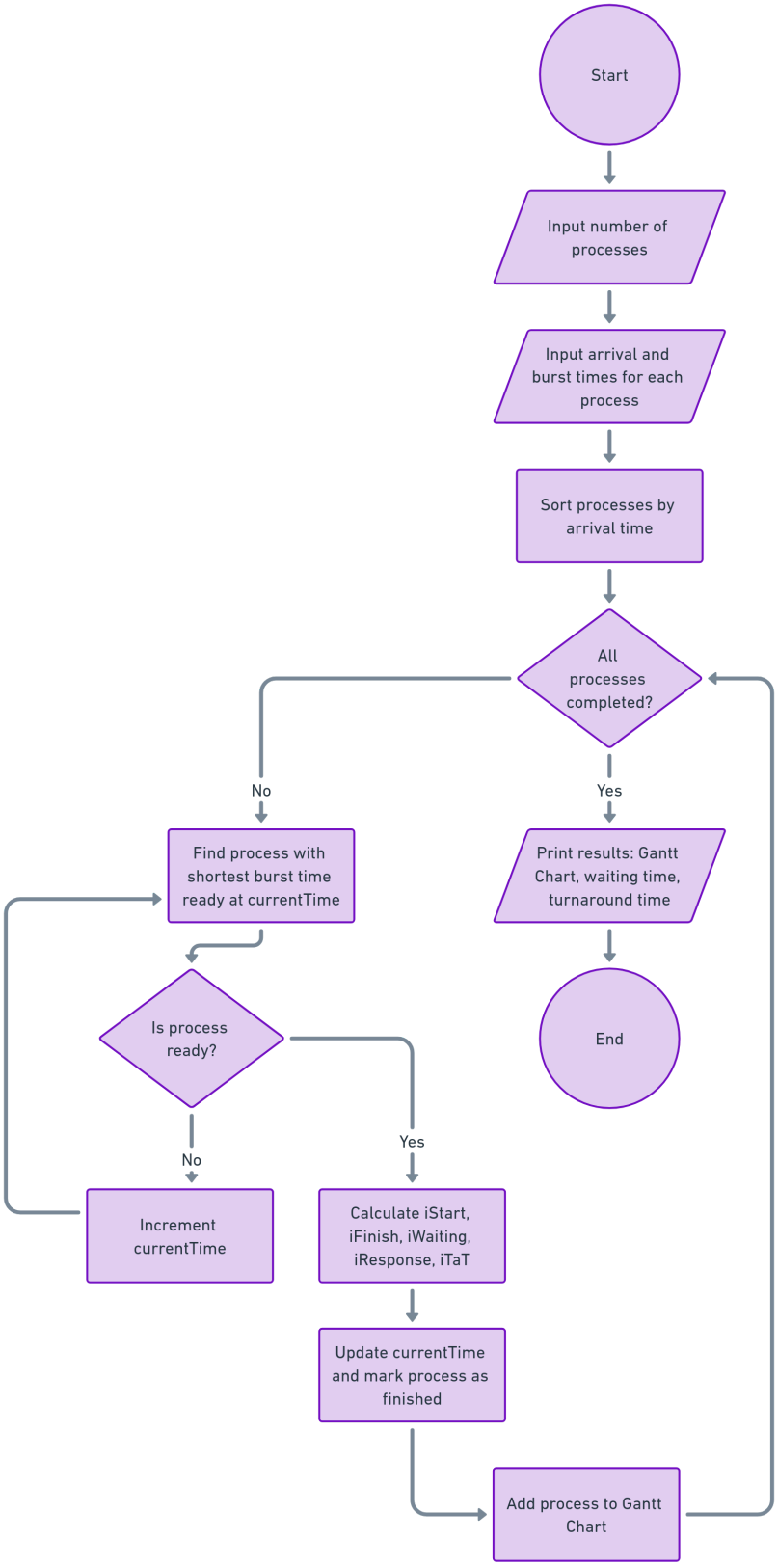
1 TỔNG QUAN

Tiêu chí	1	2	3
Trình bày cách làm	✓	✓	✓
Chụp hình minh chứng	✓	✓	✓
Giải thích kết quả	✓	✓	✓

TỰ CHẤM ĐIỂM : 10

2 SJF

2.1 Lưu đồ giải thuật


Start
Input number of processes
Input arrival and burst times for each process
Sort processes by arrival time
All processes completed?
No
Yes
Find process with shortest burst time ready at currentTime
Print results: Gantt Chart, waiting time, turnaround time
End
Is process ready?
No
Yes
Increment currentTime
Calculate iStart, iFinish, iWaiting, iResponse, iTaT
Update currentTime and mark process as finished
Add process to Gantt Chart

Kiểm thử

Giả sử ta có 5 tiến trình với thời điểm đến và thời gian burst như sau:

Tiến trình	Thời điểm đến	Thời gian burst
P1	0	7
P2	2	4
P3	4	1
P4	5	4
P5	6	2

Chạy tay thuật toán SJF

Khởi tạo

- **currentTime** = 0
- **completed** = 0 (chưa có tiến trình nào hoàn thành)
- **Gantt Chart**: Khởi tạo trống.

Thực hiện từng bước

Thời điểm `currentTime = 0`

- **Kiểm tra tiến trình sẵn sàng**: Tại thời điểm `currentTime = 0`, chỉ có **P1** đến nên được đưa vào lựa chọn.
- **Chọn tiến trình có thời gian burst ngắn nhất**: **P1** là tiến trình duy nhất, vì vậy nó được chọn để thực hiện.
- **Thực hiện P1**:
  - **Thời gian thực hiện** = 7 (hoàn thành toàn bộ thời gian burst).
  - **Cập nhật**:
    - `currentTime` = 0 + 7 = 7
    - **iStart** của **P1** = 0, **iFinish** = 7
    - **iWaiting** của **P1** = 0 (bắt đầu ngay khi đến)
    - **iResponse** của **P1** = 0
    - **iTaT** của **P1** = 7 - 0 = 7
  - **Gantt Chart** cập nhật: `| P1 (0-7) |`
  - **completed** = 1.

Thời điểm `currentTime = 7`

- **Kiểm tra tiến trình sẵn sàng**: Tại thời điểm `currentTime = 7`, các tiến trình **P2**, **P3**, **P4**, và **P5** đã đến.
- **Chọn tiến trình có thời gian burst ngắn nhất**: **P3** có thời gian burst ngắn nhất (1 đơn vị), nên nó được chọn.
- **Thực hiện P3**:
  - **Thời gian thực hiện** = 1.
  - **Cập nhật**:
    - `currentTime` = 7 + 1 = 8
    - **iStart** của **P3** = 7, **iFinish** = 8
    - **iWaiting** của **P3** = 7 - 4 = 3
    - **iResponse** của **P3** = 3
    - **iTaT** của **P3** = 8 - 4 = 4
  - **Gantt Chart** cập nhật: `| P1 (0-7) | P3 (7-8) |`
  - **completed** = 2.

Thời điểm `currentTime = 8`

- **Kiểm tra tiến trình sẵn sàng**: Tại thời điểm `currentTime = 8`, các tiến trình **P2**, **P4**, và **P5** đã đến.
- **Chọn tiến trình có thời gian burst ngắn nhất**: **P5** có thời gian burst ngắn nhất (2 đơn vị), nên nó được chọn.
- **Thực hiện P5**:
  - **Thời gian thực hiện** = 2.
  - **Cập nhật**:
    - `currentTime` = 8 + 2 = 10
    - **iStart** của **P5** = 8, **iFinish** = 10
    - **iWaiting** của **P5** = 8 - 6 = 2
    - **iResponse** của **P5** = 2
    - **iTaT** của **P5** = 10 - 6 = 4
  - **Gantt Chart** cập nhật: `| P1 (0-7) | P3 (7-8) | P5 (8-10) |`
  - **completed** = 3.

Thời điểm `currentTime = 10`

- **Kiểm tra tiến trình sẵn sàng**: Tại thời điểm `currentTime = 10`, các tiến trình **P2** và **P4** đã đến.
- **Chọn tiến trình có thời gian burst ngắn nhất**: **P2** có thời gian burst ngắn nhất (4 đơn vị), nên nó được chọn.
- **Thực hiện P2**:
  - **Thời gian thực hiện** = 4.
  - **Cập nhật**:
    - `currentTime` = 10 + 4 = 14
    - **iStart** của **P2** = 10, **iFinish** = 14
    - **iWaiting** của **P2** = 10 - 2 = 8
    - **iResponse** của **P2** = 8
    - **iTaT** của **P2** = 14 - 2 = 12
  - **Gantt Chart** cập nhật: `| P1 (0-7) | P3 (7-8) | P5 (8-10) | P2 (10-14) |`
  - **completed** = 4.

Thời điểm `currentTime = 14`

- **Kiểm tra tiến trình sẵn sàng**: Tại thời điểm `currentTime = 14`, chỉ còn **P4** chưa hoàn thành.
- **Thực hiện P4**:
  - **Thời gian thực hiện** = 4.
  - **Cập nhật**:
    - `currentTime` = 14 + 4 = 18
    - **iStart** của **P4** = 14, **iFinish** = 18
    - **iWaiting** của **P4** = 14 - 5 = 9
    - **iResponse** của **P4** = 9
    - **iTaT** của **P4** = 18 - 5 = 13
  - **Gantt Chart** cập nhật: `| P1 (0-7) | P3 (7-8) | P5 (8-10) | P2 (10-14) | P4 (14-18) |`
  - **completed** = 5.

Kết quả cuối cùng

Gantt Chart hoàn chỉnh là: `P1 (0-7) → P3 (7-8) → P5 (8-10) → P2 (10-14) → P4 (14-18)`.

2.2 Code cho giải thuật

```
#include<bits/stdc++.h>

using namespace std;

typedef struct {
    int iPID;               // Process ID
    int iArrival;           // Arrival Time
    int iBurst;             // Burst Time
    int iStart;             // Start Time
    int iFinish;            // Finish Time
    int iWaiting;           // Waiting Time
    int iResponse;          // Response Time
    int iTaT;               // Turnaround Time
} PCB;

struct GANT {
    int start;
    int finish;
    int iPID;

    GANT(int _start = 0, int _finish = 0, int _iPID = 0)
        : start(_start), finish(_finish), iPID(_iPID) {}
} ;

bool compareArrival(PCB a, PCB b) {
    return a.iArrival < b.iArrival;
}

bool compareBurst(PCB a, PCB b) {
    return a.iBurst < b.iBurst;
}
bool compareID(PCB a, PCB b) {
    return a.iPID < b.iPID;
}

void calculateTimes(vector<PCB>& processes, vector<GANT>& Gantt) {
    int currentTime = 0;
    for (int i = 0; i < processes.size(); i++) {
        int index = -1;
        int minBurst = INT_MAX;

        for (int j = 0; j < processes.size(); j++) {
            if (processes[j].iArrival <= currentTime && processes[j].iBurst <
   minBurst && processes[j].iFinish == -1) {
                minBurst = processes[j].iBurst;
                index = j;
            }
        }

        if (index == -1) {
            currentTime++;
            i--;
            continue;
```

```
        }

        processes[index].iStart = currentTime;
        processes[index].iFinish = processes[index].iStart + processes[index].iBurst;
        processes[index].iWaiting = processes[index].iStart - processes[index].iArrival;
        processes[index].iResponse = processes[index].iStart - processes[index].iArrival;
        processes[index].iTaT = processes[index].iFinish - processes[index].iArrival;

        currentTime += processes[index].iBurst;
        processes[index].iFinish = currentTime;

        Gantt.push_back(GANT(processes[index].iStart, processes[index].iFinish, processes[index].iPID));
    }
}

void printResults(vector<PCB>& processes, vector<GANT>& gantt) {
    float totalWaitingTime = 0, totalTurnaroundTime = 0;
    cout << "Gantt Chart:\n";
    for (const auto& process : gantt) {
        cout << "| P" << process.iPID << " ";
    }
    cout << "|\n";

    cout << "\n"
     << left << setw(10) << "PID"
     << setw(10) << "Arrival"
     << setw(10) << "Burst"
     << setw(10) << "Start"
     << setw(10) << "Finish"
     << setw(10) << "Waiting"
     << setw(10) << "Response"
     << setw(10) << "Turnaround" << "\n";

    sort(processes.begin(), processes.end(), compareID);
    for (const auto& process : processes) {
        cout << left << setw(10) << process.iPID
            << setw(10) << process.iArrival
            << setw(10) << process.iBurst
            << setw(10) << process.iStart
            << setw(10) << process.iFinish
            << setw(10) << process.iWaiting
            << setw(10) << process.iResponse
            << setw(10) << process.iTaT << "\n";

        totalWaitingTime += process.iWaiting;
        totalTurnaroundTime += process.iTaT;
    }

    cout << "\nAverage Waiting Time: " << totalWaitingTime / processes.size() << "\n";
```

```
    cout << "Average Turnaround Time: " << totalTurnaroundTime / processes.size() << "\n";
}

int main() {
    int n;
    cout << "Enter number of processes: ";
    cin >> n;

    vector<PCB> processes(n);
    vector<GANT> gantt;
    for (int i = 0; i < n; i++) {
        processes[i].iPID = i + 1;
        cout << "Enter arrival time and burst time for process " << i + 1 << ": ";
        cin >> processes[i].iArrival >> processes[i].iBurst;
        processes[i].iStart = processes[i].iFinish = -1; // Initialize start and finish times
    }

    sort(processes.begin(), processes.end(), compareArrival); // Sort processes by arrival time

    calculateTimes(processes, gantt);
    printResults(processes, gantt);

    return 0;
}
```

Code : SJF

Test Case 1

PID	Arrival Time	Burst Time
1	0	8
2	1	4
3	2	9
4	3	5
5	4	2

Tính toán chi tiết:

- **Tại thời điểm** 0**:**
  - P1 chạy từ 0 đến 8.
  - Finish Time của P1 = 8, Waiting Time = 0, Turnaround Time = 8.
- **Tại thời điểm** 8**:**
  - P5 có Burst Time ngắn nhất (2) nên được chọn.
  - P5 chạy từ 8 đến 10.
  - Finish Time của P5 = 10, Waiting Time = 4, Turnaround Time = 6.

- **Tại thời điểm** 10**:**
  - P2 có Burst Time ngắn nhất (4) nên được chọn.
  - P2 chạy từ 10 đến 14.
  - Finish Time của P2 = 14, Waiting Time = 9, Turnaround Time = 13.
- **Tại thời điểm** 14**:**
  - P4 có Burst Time ngắn nhất (5) nên được chọn.
  - P4 chạy từ 14 đến 19.
  - Finish Time của P4 = 19, Waiting Time = 11, Turnaround Time = 16.
- **Tại thời điểm** 19**:**
  - P3 là tiến trình còn lại.
  - P3 chạy từ 19 đến 28.
  - Finish Time của P3 = 28, Waiting Time = 17, Turnaround Time = 26.

Kết quả:

PID	Arrival Time	Burst Time	Start Time	Finish Time	Waiting Time	Turnaround Time
1	0	8	0	8	0	8
2	1	4	10	14	9	13
3	2	9	19	28	17	26
4	3	5	14	19	11	16
5	4	2	8	10	4	6

Trung bình Waiting Time: 8.2
Trung bình Turnaround Time: 13.8

Chạy code

```
[~/UIT/OS/LAB/LAB04]
 dplayergod  main  cd "/home/dplayergod/UIT/OS/LAB/LAB04/" && g++ SJF.cpp -o SJF && "/home/dplayergod/UIT/OS/LAB/LAB04/"SJF
Enter number of processes: 5
Enter arrival time and burst time for process 1: 0 8
Enter arrival time and burst time for process 2: 1 4
Enter arrival time and burst time for process 3: 2 9
Enter arrival time and burst time for process 4: 3 5
Enter arrival time and burst time for process 5: 4 2
Gantt Chart:
| P1 | P5 | P2 | P4 | P3 |

PID       Arrival   Burst     Start     Finish    Waiting   Response  Turnaround
1         0         8         0         8         0         0         8
2         1         4         10        14        9         9         13
3         2         9         19        28        17        17        26
4         3         5         14        19        11        11        16
5         4         2         8         10        4         4         6

Average Waiting Time: 8.2
Average Turnaround Time: 13.8
```

Hình 1: Test 01

Test Case 2

PID	Arrival Time	Burst Time
1	0	3
2	1	7
3	3	2
4	5	5
5	6	1

Tính toán chi tiết:

- **Tại thời điểm** 0**:**
  - P1 chạy từ 0 đến 3.
  - Finish Time của P1 = 3, Waiting Time = 0, Turnaround Time = 3.
- **Tại thời điểm** 3**:**
  - P3 có Burst Time ngắn nhất (2) nên được chọn.
  - P3 chạy từ 3 đến 5.
  - Finish Time của P3 = 5, Waiting Time = 0, Turnaround Time = 2.
- **Tại thời điểm** 5**:**
  - P4 có Burst Time ngắn hơn (5) nên được chọn.
  - P4 chạy từ 5 đến 10.
  - Finish Time của P4 = 10, Waiting Time = 0, Turnaround Time = 5.
- **Tại thời điểm** 10**:**
  - P5 có Burst Time ngắn nhất (1) nên được chọn.
  - P5 chạy từ 10 đến 11.
  - Finish Time của P5 = 11, Waiting Time = 4, Turnaround Time = 5.
- **Tại thời điểm** 11**:**
  - P2 là tiến trình còn lại.
  - P2 chạy từ 11 đến 18.
  - Finish Time của P2 = 18, Waiting Time = 10, Turnaround Time = 17.

Kết quả:

PID	Arrival Time	Burst Time	Start Time	Finish Time	Waiting Time	Turnaround Time
1	0	3	0	3	0	3
2	1	7	11	18	10	17
3	3	2	3	5	0	2
4	5	5	5	10	0	5
5	6	1	10	11	4	5

Trung bình Waiting Time: 2.8
Trung bình Turnaround Time: 6.4

```
  [~/UIT/OS/LAB/LAB04]
  dplayergod  main  cd "/home/dplayergod/UIT/OS/LAB/LAB04/" && g++ SJF.cpp -o SJF && "/home/dplayergod/UIT/OS/LAB/LAB04/"SJF
Enter number of processes: 5
Enter arrival time and burst time for process 1: 0 3
Enter arrival time and burst time for process 2: 1 7
Enter arrival time and burst time for process 3: 3 2
Enter arrival time and burst time for process 4: 5 5
Enter arrival time and burst time for process 5: 6 1
Gantt Chart:
| P1 | P3 | P4 | P5 | P2 |

PID       Arrival   Burst     Start     Finish    Waiting   Response  Turnaround
1         0         3         0         3         0         0         3
2         1         7         11        18        10        10        17
3         3         2         3         5         0         0         2
4         5         5         5         10        0         0         5
5         6         1         10        11        4         4         5

Average Waiting Time: 2.8
Average Turnaround Time: 6.4
```

Hình 2: Test 02

Test Case 3

PID	Arrival Time	Burst Time
1	0	5
2	2	3
3	3	8
4	5	6
5	6	4

Tính toán chi tiết:

- **Tại thời điểm** 0**:**
  - P1 chạy từ 0 đến 5.
  - Finish Time của P1 = 5, Waiting Time = 0, Turnaround Time = 5.
- **Tại thời điểm** 5**:**
  - P2 có Burst Time ngắn nhất (3) nên được chọn.
  - P2 chạy từ 5 đến 8.
  - Finish Time của P2 = 8, Waiting Time = 3, Turnaround Time = 6.
- **Tại thời điểm** 8**:**
  - P5 có Burst Time ngắn nhất (4) nên được chọn.
  - P5 chạy từ 8 đến 12.
  - Finish Time của P5 = 12, Waiting Time = 2, Turnaround Time = 6.
- **Tại thời điểm** 12**:**
  - P4 có Burst Time ngắn nhất (6) nên được chọn.
  - P4 chạy từ 12 đến 18.
  - Finish Time của P4 = 18, Waiting Time = 7, Turnaround Time = 13.
- **Tại thời điểm** 18**:**
  - P3 là tiến trình còn lại.
  - P3 chạy từ 18 đến 26.
  - Finish Time của P3 = 26, Waiting Time = 15, Turnaround Time = 23.

Kết quả:

PID	Arrival Time	Burst Time	Start Time	Finish Time	Waiting Time	Turnaround Time
1	0	5	0	5	0	5
2	2	3	5	8	3	6
3	3	8	18	26	15	23
4	5	6	12	18	7	13
5	6	4	8	12	2	6

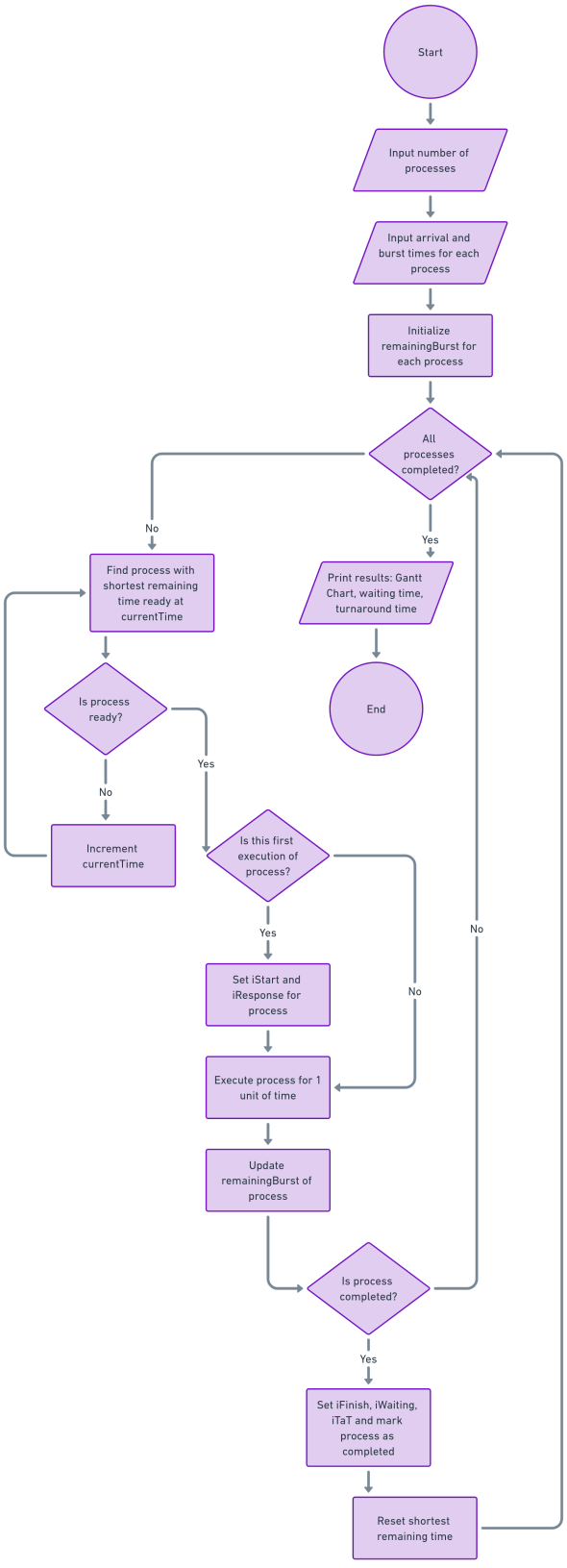
Trung bình Waiting Time: 5.4
Trung bình Turnaround Time: 10.6

```
  [~/UIT/OS/LAB/LAB04]
  dplayergod  main  cd "/home/dplayergod/UIT/OS/LAB/LAB04/" && g++ SJF.cpp -o SJF && "/home/dplayergod/UIT/OS/LAB/LAB04/"SJF
Enter number of processes: 5
Enter arrival time and burst time for process 1: 0 5
Enter arrival time and burst time for process 2: 2 3
Enter arrival time and burst time for process 3: 3 8
Enter arrival time and burst time for process 4: 5 6
Enter arrival time and burst time for process 5: 6 4
Gantt Chart:
| P1 | P2 | P5 | P4 | P3 |

PID       Arrival   Burst     Start     Finish    Waiting   Response  Turnaround
1         0         5         0         5         0         0         5
2         2         3         5         8         3         3         6
3         3         8         18        26        15        15        23
4         5         6         12        18        7         7         13
5         6         4         8         12        2         2         6

Average Waiting Time: 5.4
Average Turnaround Time: 10.6
```

Hình 3: Test 03

3 SRTF

3.1 Lưu đồ


Start
Input number of processes
Input arrival and burst times for each process
Initialize remainingBurst for each process
All processes completed?
No
Yes
Find process with shortest remaining time ready at currentTime
Print results: Gantt Chart, waiting time, turnaround time
End
Is process ready?
Yes
No
Increment currentTime
Is this first execution of process?
Yes
No
Set iStart and iResponse for process
Execute process for 1 unit of time
Update remainingBurst of process
Is process completed?
No
Yes
Set iFinish, iWaiting, iTaT and mark process as completed
Reset shortest remaining time

Kiểm thử

Giả sử ta có 5 tiến trình với thời điểm đến và thời gian burst như sau:

Tiến trình	Thời điểm đến	Thời gian burst
P1	0	2
P2	1	3
P3	2	1
P4	3	2
P5	4	1

Các bước tính toán chi tiết

Khởi tạo

- **currentTime** = 0
- **completed** = 0 (số tiến trình đã hoàn thành)
- **Gantt Chart**: Bắt đầu trống

Thực hiện từng bước

Thời điểm `currentTime = 0`

- Tiến trình sẵn sàng: **P1** (arrival = 0, burst = 2).
- Chọn tiến trình có thời gian burst ngắn nhất: **P1**.
- Thực hiện **P1** trong 1 đơn vị thời gian.
  - **remainingBurst** của **P1** giảm từ 2 xuống 1.
  - **currentTime** tăng lên 1.
  - **iStart** của **P1** = 0.
  - **Gantt Chart** cập nhật: `| P1 (0-1) |`

Thời điểm `currentTime = 1`

- Tiến trình sẵn sàng: **P1** (remainingBurst = 1), **P2** (arrival = 1, burst = 3).
- Chọn tiến trình có thời gian burst ngắn nhất: **P1**.
- Thực hiện **P1** trong 1 đơn vị thời gian.
  - **remainingBurst** của **P1** giảm từ 1 xuống 0.
  - **currentTime** tăng lên 2.
  - **iFinish** của **P1** = 2.
  - **iWaiting** của **P1** = 0.
  - **iTaT** của **P1** = 2.
  - **Gantt Chart** cập nhật: `| P1 (0-2) |`

Thời điểm `currentTime = 2`

- Tiến trình sẵn sàng: **P2** (remainingBurst = 3), **P3** (arrival = 2, burst = 1).
- Chọn tiến trình có thời gian burst ngắn nhất: **P3**.
- Thực hiện **P3** trong 1 đơn vị thời gian.
  - **remainingBurst** của **P3** giảm từ 1 xuống 0.
  - **currentTime** tăng lên 3.
  - **iFinish** của **P3** = 3.
  - **iWaiting** của **P3** = 0.
  - **iTaT** của **P3** = 1.
  - **Gantt Chart** cập nhật: `| P1 (0-2) | P3 (2-3) |`

Thời điểm `currentTime = 3`

- Tiến trình sẵn sàng: **P2** (remainingBurst = 3), **P4** (arrival = 3, burst = 2).
- Chọn tiến trình có thời gian burst ngắn nhất: **P4**.
- Thực hiện **P4** trong 1 đơn vị thời gian.
  - **remainingBurst** của **P4** giảm từ 2 xuống 1.
  - **currentTime** tăng lên 4.
  - **iStart** của **P4** = 3.
  - **Gantt Chart** cập nhật: `| P1 (0-2) | P3 (2-3) | P4 (3-4) |`

Thời điểm `currentTime = 4`

- Tiến trình sẵn sàng: **P2** (remainingBurst = 3), **P4** (remainingBurst = 1), **P5** (arrival = 4, burst = 1).
- Chọn tiến trình có thời gian burst ngắn nhất: **P4**.
- Thực hiện **P4** trong 1 đơn vị thời gian.
  - **remainingBurst** của **P4** giảm từ 1 xuống 0.
  - **currentTime** tăng lên 5.
  - **iFinish** của **P4** = 5.
  - **iWaiting** của **P4** = 0.
  - **iTaT** của **P4** = 2.
  - **Gantt Chart** cập nhật: `| P1 (0-2) | P3 (2-3) | P4 (3-5) |`

Thời điểm `currentTime = 5`

- Tiến trình sẵn sàng: **P2** (remainingBurst = 3), **P5** (remainingBurst = 1).
- Chọn tiến trình có thời gian burst ngắn nhất: **P5**.
- Thực hiện **P5** trong 1 đơn vị thời gian.
  - **remainingBurst** của **P5** giảm từ 1 xuống 0.
  - **currentTime** tăng lên 6.
  - **iFinish** của **P5** = 6.
  - **iWaiting** của **P5** = 1.
  - **iTaT** của **P5** = 2.
  - **Gantt Chart** cập nhật: `| P1 (0-2) | P3 (2-3) | P4 (3-5) | P5 (5-6) |`

Thời điểm `currentTime = 6`

- Tiến trình sẵn sàng: **P2** (remainingBurst = 3).
- Chọn tiến trình có thời gian burst ngắn nhất: **P2**.
- Thực hiện **P2** trong 3 đơn vị thời gian.
  - **remainingBurst** của **P2** giảm từ 3 xuống 0.
  - **currentTime** tăng lên 9.
  - **iFinish** của **P2** = 9.
  - **iWaiting** của **P2** = 5.
  - **iTaT** của **P2** = 8.
  - **Gantt Chart** cập nhật: `| P1 (0-2) | P3 (2-3) | P4 (3-5) | P5 (5-6) | P2 (6-9) |`

Kết quả cuối cùng

Gantt Chart hoàn chỉnh là: `P1 (0-2) → P2 (2-3) → P4 (3-5) → P5 (5-6) → P2(6-9)`.

3.2 Code cho giải thuật

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

typedef struct {
    int iPID;               // Process ID
    int iArrival;           // Arrival Time
    int iBurst;             // Burst Time
    int iStart;             // Start Time
    int iFinish;            // Finish Time
    int iWaiting;           // Waiting Time
```

```
    int iResponse;          // Response Time
    int iTaT;               // Turnaround Time
    int remainingBurst;     // Remaining Burst Time
    bool isCompleted = false; // Completion status
} PCB;

struct GANT {
    int start;
    int finish;
    int iPID;

    GANT(int _start = 0, int _finish = 0, int _iPID = 0)
        : start(_start), finish(_finish), iPID(_iPID) {}
};

void calculateSRTF(vector<PCB>& processes, vector<GANT>& gantt) {
    int currentTime = 0;
    int completed = 0;
    int n = processes.size();
    int shortestRemainingTime = INT_MAX;
    int shortestIndex = -1;
    bool foundProcess = false;

    // Initialize remainingBurst for each process
    for (int i = 0; i < n; i++) {
        processes[i].remainingBurst = processes[i].iBurst;
    }

    while (completed != n) {        // Execute the selected process for 1 unit of time

        foundProcess = false;

        // Find the process with the shortest remaining time at the current time
        for (int i = 0; i < n; i++) {
            if (processes[i].iArrival <= currentTime && !processes[i].isCompleted &&
                processes[i].remainingBurst < shortestRemainingTime && processes[i].remainingBurst > 0) {

                shortestRemainingTime = processes[i].remainingBurst;
                shortestIndex = i;
                foundProcess = true;
            }
        }

        if (!foundProcess) {
            currentTime++;
            continue;
        }

        // Record the start time of the process if this is its first execution
        if (processes[shortestIndex].remainingBurst == processes[shortestIndex].iBurst) {
            processes[shortestIndex].iStart = currentTime;
```

```
62              processes[shortestIndex].iResponse = processes[shortestIndex].iStart
    - processes[shortestIndex].iArrival;
63          }
64
65          // Execute the selected process for 1 unit of time
66          processes[shortestIndex].remainingBurst--;
67
68          if (gantt.size() && processes[shortestIndex].iPID == gantt.back().iPID)
    gantt.back().finish = currentTime + 1;
69          else gantt.push_back(GANT(currentTime, currentTime + 1, processes[
    shortestIndex].iPID)); // Record Gantt chart entry
70          currentTime++;
71
72          // If the process is completed
73          if (processes[shortestIndex].remainingBurst == 0) {
74              processes[shortestIndex].iFinish = currentTime;
75              processes[shortestIndex].iTaT = processes[shortestIndex].iFinish -
    processes[shortestIndex].iArrival;
76              processes[shortestIndex].iWaiting = processes[shortestIndex].iTaT -
    processes[shortestIndex].iBurst;
77              processes[shortestIndex].isCompleted = true;
78              completed++;
79              shortestRemainingTime = INT_MAX; // Reset the shortest remaining
    time value
80          }
81      }
82  }
83
84  void printResults(vector<PCB>& processes, vector<GANT>& gantt) {
85      float totalWaitingTime = 0, totalTurnaroundTime = 0;
86
87      // Print Gantt Chart
88      cout << "Gantt Chart:\n";
89      for (const auto& entry : gantt) {
90          cout << "| P" << entry.iPID << " (" << entry.start << "," << entry.
    finish << ") ";
91      }
92      cout << "|\n";
93
94      // Print Process Information
95      cout << "\n"
96          << left << setw(10) << "PID"
97          << setw(10) << "Arrival"
98          << setw(10) << "Burst"
99          << setw(10) << "Start"
100         << setw(10) << "Finish"
101         << setw(10) << "Waiting"
102         << setw(10) << "Response"
103         << setw(10) << "Turnaround" << "\n";
104
105     for (const auto& process : processes) {
106         cout << left << setw(10) << process.iPID
107              << setw(10) << process.iArrival
108              << setw(10) << process.iBurst
109              << setw(10) << process.iStart
```

```
            << setw(10) << process.iFinish
            << setw(10) << process.iWaiting
            << setw(10) << process.iResponse
            << setw(10) << process.iTaT << "\n";

        totalWaitingTime += process.iWaiting;
        totalTurnaroundTime += process.iTaT;
    }

    cout << "\nAverage Waiting Time: " << totalWaitingTime / processes.size() << "\n";
    cout << "Average Turnaround Time: " << totalTurnaroundTime / processes.size() << "\n";
}

int main() {
    int n;
    cout << "Enter number of processes: ";
    cin >> n;

    vector<PCB> processes(n);
    vector<GANT> gantt;
    for (int i = 0; i < n; i++) {
        processes[i].iPID = i + 1;
        cout << "Enter arrival time and burst time for process " << i + 1 << ": ";
        cin >> processes[i].iArrival >> processes[i].iBurst;
        processes[i].iStart = processes[i].iFinish = -1; // Initialize start and finish times
    }

    calculateSRTF(processes, gantt);
    printResults(processes, gantt);

    return 0;
}
```

Code : SRTF

Test Case 1

PID	Arrival Time	Burst Time
1	0	8
2	1	4
3	2	9
4	3	5
5	4	2

Tính toán chi tiết (SRTF):

- **Tại thời điểm** 0**:**
  - P1 chạy từ 0 đến 1.

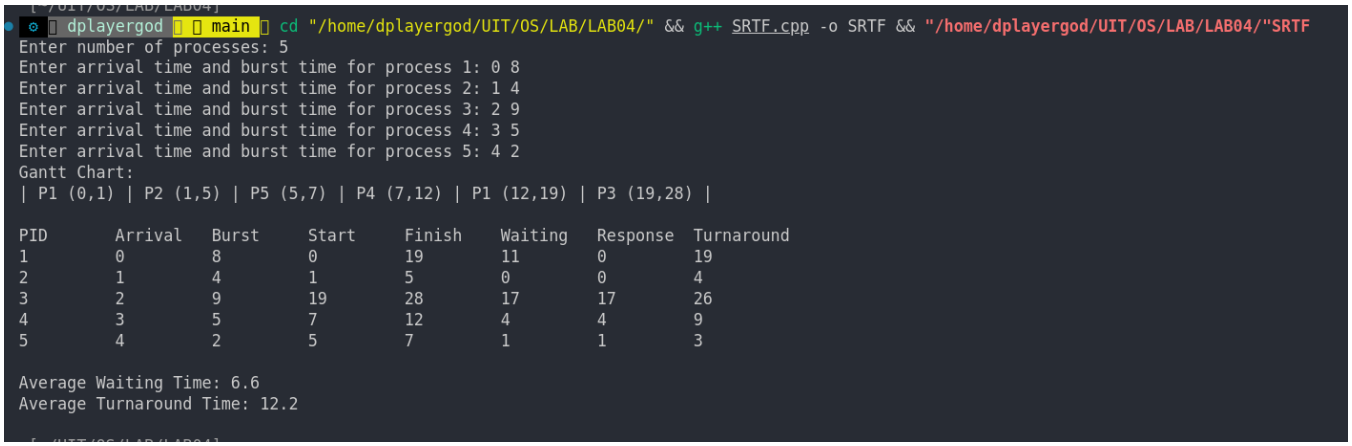
  - Thời gian còn lại của P1 là 7.

- **Tại thời điểm** 1**:**
  - P2 đến với Burst Time là 4, thay thế P1, và chạy từ 1 đến 5.
  - P2 kết thúc với Finish Time = 5, Waiting Time = 0, Turnaround Time = 4.
- **Tại thời điểm** 5**:**
  - P5 có Burst Time là 2 nên chạy từ 5 đến 7.
  - P5 kết thúc với Finish Time = 7, Waiting Time = 1, Turnaround Time = 3.
- **Tại thời điểm** 7**:**
  - P4 có Burst Time là 5 và chạy từ 7 đến 12.
  - P4 kết thúc với Finish Time = 12, Waiting Time = 4, Turnaround Time = 9.
- **Tại thời điểm** 12**:**
  - P1 tiếp tục và chạy từ 12 đến 19.
  - P1 kết thúc với Finish Time = 19, Waiting Time = 11, Turnaround Time = 19.
- **Tại thời điểm** 19**:**
  - P3 chạy từ 19 đến 28.
  - P3 kết thúc với Finish Time = 28, Waiting Time = 17, Turnaround Time = 26.

Kết quả:

PID	Arrival Time	Burst Time	Start Time	Finish Time	Waiting Time	Turnaround Time
1	0	8	0	19	11	19
2	1	4	1	5	0	4
3	2	9	19	28	17	26
4	3	5	7	12	4	9
5	4	2	5	7	1	3

Trung bình Waiting Time: 6.6
Trung bình Turnaround Time: 12.2



Hình 4: Test 01

Test Case 2

PID	Arrival Time	Burst Time
1	0	3
2	1	7
3	3	2
4	5	5
5	6	1

Tính toán chi tiết (SRTF):

- **Tại thời điểm** 0**:**
  - P1 chạy từ 0 đến 3.
  - P1 kết thúc với Finish Time = 3, Waiting Time = 0, Turnaround Time = 3.
- **Tại thời điểm** 3**:**
  - P3 đến với Burst Time là 2 và chạy từ 3 đến 5.
  - P3 kết thúc với Finish Time = 5, Waiting Time = 0, Turnaround Time = 2.
- **Tại thời điểm** 5**:**
  - P4 đến với Burst Time là 5 nên được chọn để chạy.
  - P4 chạy từ 5 đến 6, sau đó bị gián đoạn khi P5 đến.
  - Thời gian còn lại của P4 là 4.
- **Tại thời điểm** 6**:**
  - P5 đến với Burst Time là 1, nhỏ hơn thời gian còn lại của P4, nên chạy từ 6 đến 7.
  - P5 kết thúc với Finish Time = 7, Waiting Time = 0, Turnaround Time = 1.
- **Tại thời điểm** 7**:**
  - P4 tiếp tục với Burst Time còn lại là 4 và chạy từ 7 đến 11.
  - P4 kết thúc với Finish Time = 11, Waiting Time = 2, Turnaround Time = 6.
- **Tại thời điểm** 11**:**
  - P2 là tiến trình còn lại và chạy từ 11 đến 18.
  - P2 kết thúc với Finish Time = 18, Waiting Time = 10, Turnaround Time = 17.

Kết quả:

PID	Arrival Time	Burst Time	Start Time	Finish Time	Waiting Time	Turnaround Time
1	0	3	0	3	0	3
2	1	7	11	18	10	17
3	3	2	3	5	0	2
4	5	5	5	11	0	6
5	6	1	6	7	0	1

Trung bình Waiting Time: 2.4
Trung bình Turnaround Time: 5.8

```
[~/UIT/OS/LAB/LAB04]
 dplayergod  main  cd "/home/dplayergod/UIT/OS/LAB/LAB04/" && g++ SRTF.cpp -o SRTF && "/home/dplayergod/UIT/OS/LAB/LAB04/"SRTF
Enter number of processes: 5
Enter arrival time and burst time for process 1: 0 3
Enter arrival time and burst time for process 2: 1 7
Enter arrival time and burst time for process 3: 3 2
Enter arrival time and burst time for process 4: 5 5
Enter arrival time and burst time for process 5: 6 1
Gantt Chart:
| P1 (0,3) | P3 (3,5) | P4 (5,6) | P5 (6,7) | P4 (7,11) | P2 (11,18) |

PID       Arrival   Burst     Start     Finish    Waiting   Response  Turnaround
1         0         3         0         3         0         0         3
2         1         7         11        18        10        10        17
3         3         2         3         5         0         0         2
4         5         5         5         11        1         0         6
5         6         1         6         7         0         0         1

Average Waiting Time: 2.2
Average Turnaround Time: 5.8
```

Hình 5: Test 02

Test Case 3

PID	Arrival Time	Burst Time
1	0	5
2	2	3
3	4	7
4	5	2
5	6	4

Tính toán chi tiết (SRTF):

- **Tại thời điểm** 0**:**
  - P1 chạy từ 0 đến 2.
  - Thời gian còn lại của P1 là 3.
- **Tại thời điểm** 2**:**
  - P2 đến với Burst Time là 3, nhưng P1 vẫn tiếp tục chạy vì thời gian còn lại của P1 là 3, bằng với thời gian Burst của P2.
  - P1 tiếp tục chạy từ 2 đến 5 và kết thúc.
  - P1 kết thúc với Finish Time = 5, Waiting Time = 0, Turnaround Time = 5.
- **Tại thời điểm** 5**:**
  - P4 đến với Burst Time là 2, nhỏ hơn thời gian Burst của P2 (3), nên P4 được ưu tiên.
  - P4 chạy từ 5 đến 7.
  - P4 kết thúc với Finish Time = 7, Waiting Time = 0, Turnaround Time = 2.
- **Tại thời điểm** 7**:**
  - P2 tiếp tục với Burst Time còn lại là 3 và chạy từ 7 đến 10.
  - P2 kết thúc với Finish Time = 10, Waiting Time = 5, Turnaround Time = 8.
- **Tại thời điểm** 10**:**
  - P5 đến với Burst Time là 4, nhỏ hơn thời gian Burst của P3 (7), nên P5 được ưu tiên.
  - P5 chạy từ 10 đến 14.
  - P5 kết thúc với Finish Time = 14, Waiting Time = 4, Turnaround Time = 8.
- **Tại thời điểm** 14**:**
  - P3 là tiến trình còn lại và chạy từ 14 đến 21.
  - P3 kết thúc với Finish Time = 21, Waiting Time = 10, Turnaround Time = 17.

Kết quả:

PID	Arrival Time	Burst Time	Start Time	Finish Time	Waiting Time	Turnaround Time
1	0	5	0	5	0	5
2	2	3	7	10	5	8
3	4	7	14	21	10	17
4	5	2	5	7	0	2
5	6	4	10	14	4	8

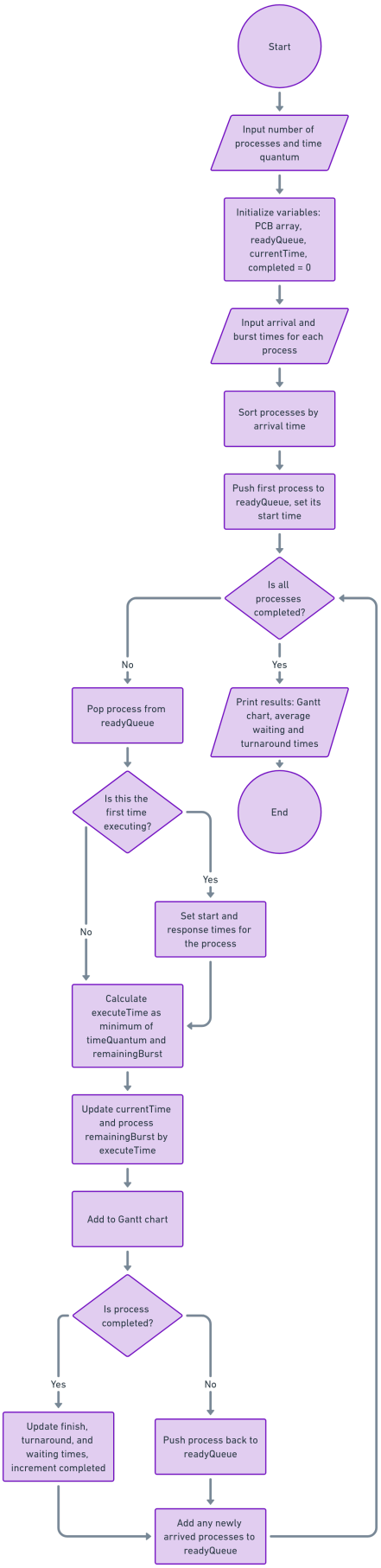
Trung bình Waiting Time: 3.8
Trung bình Turnaround Time: 8.0

```
[~/UIT/OS/LAB/LAB04]
 dplayergod  main  cd "/home/dplayergod/UIT/OS/LAB/LAB04/" && g++ SRTF.cpp -o SRTF && "/home/dplayergod/UIT/OS/LAB/LAB04/"SRTF
Enter number of processes: 5
Enter arrival time and burst time for process 1: 0 5
Enter arrival time and burst time for process 2: 2 3
Enter arrival time and burst time for process 3: 4 7
Enter arrival time and burst time for process 4: 5 2
Enter arrival time and burst time for process 5: 6 4
Gantt Chart:
| P1 (0,5) | P4 (5,7) | P2 (7,10) | P5 (10,14) | P3 (14,21) |

PID       Arrival   Burst     Start     Finish    Waiting   Response  Turnaround
1         0         5         0         5         0         0         5
2         2         3         7         10        5         5         8
3         4         7         14        21        10        10        17
4         5         2         5         7         0         0         2
5         6         4         10        14        4         4         8

Average Waiting Time: 3.8
Average Turnaround Time: 8
```

Hình 6: Test 03

4 RR

4.1 Lưu đồ


Start
Input number of processes and time quantum
Initialize variables: PCB array, readyQueue, currentTime, completed = 0
Input arrival and burst times for each process
Sort processes by arrival time
Push first process to readyQueue, set its start time
Is all processes completed?
No
Yes
Pop process from readyQueue
Print results: Gantt chart, average waiting and turnaround times
End
Is this the first time executing?
Yes
No
Set start and response times for the process
Calculate executeTime as minimum of timeQuantum and remainingBurst
Update currentTime and process remainingBurst by executeTime
Add to Gantt chart
Is process completed?
Yes
No
Update finish, turnaround, and waiting times, increment completed
Push process back to readyQueue
Add any newly arrived processes to readyQueue

Kiểm thử

Giả sử ta có 5 tiến trình với thời điểm đến và thời gian burst như sau. Time quantum là 3 đơn vị thời gian.

Tiến trình	Thời điểm đến	Thời gian burst
P1	0	4
P2	1	5
P3	2	2
P4	3	1
P5	4	6

Time Quantum: 3

Bước chạy tay chi tiết theo lưu đồ

Khởi tạo

- **currentTime** = 0
- **completed** = 0 (số tiến trình đã hoàn thành)
- **readyQueue** ban đầu rỗng.

Tại `currentTime = 0`

- **P1** có thời điểm đến là 0, được đưa vào `readyQueue`.
- **Thực hiện P1**:
  - `remainingBurst` của **P1** là 4, lớn hơn `timeQuantum` (3). Do đó, `executeTime` cho **P1** là 3.
  - **Cập nhật**:
    - `currentTime` = 0 + 3 = 3
    - `remainingBurst` của **P1** = 4 - 3 = 1
  - **Gantt Chart** cập nhật: `P1 (0-3)`
  - **Kiểm tra hoàn thành**: **P1** chưa hoàn thành (vẫn còn 1 đơn vị thời gian). Thêm **P1** trở lại `readyQueue`.

Tại `currentTime = 3`

- **Kiểm tra tiến trình mới đến**:
  - **P2**, **P3**, và **P4** đã đến trước thời điểm hiện tại (`currentTime = 3`). Thêm **P2**, **P3**, và **P4** vào `readyQueue`.
- **Thực hiện P2**:
  - `remainingBurst` của **P2** là 5, lớn hơn `timeQuantum`. Do đó, `executeTime` cho **P2** là 3.

  - **Cập nhật**:
    * `currentTime` = 3 + 3 = 6
    * `remainingBurst` của **P2** = 5 - 3 = 2
  - **Gantt Chart** cập nhật: `P1 (0-3) → P2 (3-6)`
  - **Kiểm tra hoàn thành**: **P2** chưa hoàn thành (vẫn còn 2 đơn vị thời gian). Thêm **P2** trở lại `readyQueue`.

Tại `currentTime = 6`

- **Kiểm tra tiến trình mới đến**:
  - **P5** đã đến trước thời điểm hiện tại (`currentTime = 6`). Thêm **P5** vào `readyQueue`.
- **Thực hiện P3**:
  - `remainingBurst` của **P3** là 2, nhỏ hơn `timeQuantum`. Do đó, `executeTime` cho **P3** là 2.
  - **Cập nhật**:
    * `currentTime` = 6 + 2 = 8
    * `remainingBurst` của **P3** = 2 - 2 = 0
  - **Gantt Chart** cập nhật: `P1 (0-3) → P2 (3-6) → P3 (6-8)`
  - **Kiểm tra hoàn thành**: **P3** hoàn thành (`remainingBurst` = 0). Tăng `completed` lên 1.

Tại `currentTime = 8`

- **Thực hiện P4**:
  - `remainingBurst` của **P4** là 1, nhỏ hơn `timeQuantum`. Do đó, `executeTime` cho **P4** là 1.
  - **Cập nhật**:
    * `currentTime` = 8 + 1 = 9
    * `remainingBurst` của **P4** = 1 - 1 = 0
  - **Gantt Chart** cập nhật: `P1 (0-3) → P2 (3-6) → P3 (6-8) → P4 (8-9)`
  - **Kiểm tra hoàn thành**: **P4** hoàn thành (`remainingBurst` = 0). Tăng `completed` lên 2.

Tại `currentTime = 9`

- **Thực hiện P1** (lượt tiếp theo):
  - `remainingBurst` của **P1** là 1, nhỏ hơn `timeQuantum`. Do đó, `executeTime` cho **P1** là 1.
  - **Cập nhật**:
    * `currentTime` = 9 + 1 = 10
    * `remainingBurst` của **P1** = 1 - 1 = 0

- **Gantt Chart** cập nhật: `P1 (0-3) → P2 (3-6) → P3 (6-8) → P4 (8-9) → P1 (9-10)`
- **Kiểm tra hoàn thành**: **P1** hoàn thành (`remainingBurst` = 0). Tăng `completed` lên 3.

Tại `currentTime = 10`

- **Thực hiện P5**:
  - `remainingBurst` của **P5** là 6, lớn hơn `timeQuantum`. Do đó, `executeTime` cho **P5** là 3.
  - **Cập nhật**:
    - `currentTime` = 10 + 3 = 13
    - `remainingBurst` của **P5** = 6 - 3 = 3
  - **Gantt Chart** cập nhật: `P1 (0-3) → P2 (3-6) → P3 (6-8) → P4 (8-9) → P1 (9-10) → P5 (10-13)`
  - **Kiểm tra hoàn thành**: **P5** chưa hoàn thành (`remainingBurst` > 0). Thêm **P5** trở lại `readyQueue`.

Tại `currentTime = 13`

- **Thực hiện P2** (lượt tiếp theo):
  - `remainingBurst` của **P2** là 2, nhỏ hơn `timeQuantum`. Do đó, `executeTime` cho **P2** là 2.
  - **Cập nhật**:
    - `currentTime` = 13 + 2 = 15
    - `remainingBurst` của **P2** = 2 - 2 = 0
  - **Gantt Chart** cập nhật: `P1 (0-3) → P2 (3-6) → P3 (6-8) → P4 (8-9) → P1 (9-10) → P5 (10-13) → P2 (13-15)`
  - **Kiểm tra hoàn thành**: **P2** hoàn thành (`remainingBurst` = 0). Tăng `completed` lên 4.

Tại `currentTime = 15`

- **Thực hiện P5** (lượt tiếp theo):
  - `remainingBurst` của **P5** là 3, bằng `timeQuantum`. Do đó, `executeTime` cho **P5** là 3.
  - **Cập nhật**:
    - `currentTime` = 15 + 3 = 18
    - `remainingBurst` của **P5** = 3 - 3 = 0
  - **Gantt Chart** cập nhật: `P1 (0-3) → P2 (3-6) → P3 (6-8) → P4 (8-9) → P1 (9-10) → P5 (10-13) → P2 (13-15) → P5 (15-18)`
  - **Kiểm tra hoàn thành**: **P5** hoàn thành (`remainingBurst` = 0). Tăng `completed` lên 5.

Kết quả cuối cùng

Gantt Chart hoàn chỉnh là: P1 (0-3) → P2 (3-6) → P3 (6-8) → P4 (8-9) → P1 (9-10) → P5 (10-13) → P2 (13-15) → P5 (15-18).

4.2 Code cho giải thuật

```cpp
#include<bits/stdc++.h>

using namespace std;

typedef struct {
    int iPID;               // Process ID
    int iArrival;           // Arrival Time
    int iBurst;             // Burst Time
    int iStart;             // Start Time
    int iFinish;            // Finish Time
    int iWaiting;           // Waiting Time
    int iResponse;          // Response Time
    int iTaT;               // Turnaround Time
    int remainingBurst;     // Remaining Burst Time
} PCB;

struct GANT {
    int start;
    int finish;
    int iPID;

    GANT(int _start = 0, int _finish = 0, int _iPID = 0)
        : start(_start), finish(_finish), iPID(_iPID) {}
} ;

void calculateRR(vector<PCB>& processes, vector<GANT>& gantt, int timeQuantum) {
    int currentTime = 0;
    queue<int> readyQueue;
    vector<bool> inQueue(processes.size(), false);
    int completed = 0;

    // Sort by Arrival Time
    sort(processes.begin(), processes.end(), [](PCB a, PCB b) {
        return a.iArrival < b.iArrival;
    });
    cout << '\n';

    readyQueue.push(0); // Add the first process to the queue
    inQueue[0] = true;
    processes[0].iStart = max(currentTime, processes[0].iArrival);

    while (completed < processes.size()) {
        int index = readyQueue.front();
        readyQueue.pop();
        inQueue[index] = false;

        if (processes[index].remainingBurst == processes[index].iBurst) {
```

```
48              processes[index].iStart = max(currentTime, processes[index].iArrival
    );
49              processes[index].iResponse = processes[index].iStart - processes[
    index].iArrival;
50          }
51
52          int executeTime = min(timeQuantum, processes[index].remainingBurst);
53          int lastTime = currentTime;
54          currentTime = max(currentTime, processes[index].iArrival) + executeTime;
55          processes[index].remainingBurst -= executeTime;
56
57          if (gantt.size() && processes[index].iPID == gantt.back().iPID) gantt.
    back().finish = currentTime;
58          else gantt.push_back(GANT(lastTime, currentTime, processes[index].iPID))
    ;
59
60          for (int i = 0; i < processes.size(); i++) {
61              if (processes[i].iArrival <= currentTime && processes[i].
    remainingBurst > 0 && !inQueue[i] && i != index) {
62                  readyQueue.push(i);
63                  inQueue[i] = true;
64              }
65          }
66
67          if (processes[index].remainingBurst == 0) {
68              processes[index].iFinish = currentTime;
69              processes[index].iTaT = processes[index].iFinish - processes[index].
    iArrival;
70              processes[index].iWaiting = processes[index].iTaT - processes[index
    ].iBurst;
71              completed++;
72          } else {
73              readyQueue.push(index);
74              inQueue[index] = true;
75          }
76      }
77  }
78
79  void printResults(vector<PCB>& processes, vector<GANT>& gantt) {
80      float totalWaitingTime = 0, totalTurnaroundTime = 0;
81
82      cout << "Gantt Chart:\n";
83      for (const auto& process : gantt) {
84          cout << "| P" << process.iPID << " " << "(" << process.start << ',' <<
    process.finish << ") ";
85      }
86      cout << "|\n";
87
88      cout << "\n"
89          << left << setw(10) << "PID"
90          << setw(10) << "Arrival"
91          << setw(10) << "Burst"
92          << setw(10) << "Start"
93          << setw(10) << "Finish"
94          << setw(10) << "Waiting"
```

```
        << setw(10) << "Response"
        << setw(10) << "Turnaround" << "\n";

    sort(processes.begin(), processes.end(), [](PCB a, PCB b) {
        return a.iPID < b.iPID;
    });

    for (const auto& process : processes) {
        cout << left << setw(10) << process.iPID
             << setw(10) << process.iArrival
             << setw(10) << process.iBurst
             << setw(10) << process.iStart
             << setw(10) << process.iFinish
             << setw(10) << process.iWaiting
             << setw(10) << process.iResponse
             << setw(10) << process.iTaT << "\n";

        totalWaitingTime += process.iWaiting;
        totalTurnaroundTime += process.iTaT;
    }

    cout << "\nAverage Waiting Time: " << totalWaitingTime / processes.size() << "\n";
    cout << "Average Turnaround Time: " << totalTurnaroundTime / processes.size() << "\n";
}

int main() {
    int n, timeQuantum;
    cout << "Enter number of processes: ";
    cin >> n;
    cout << "Enter time quantum: ";
    cin >> timeQuantum;

    vector<PCB> processes(n);
    vector<GANT> gantt;
    for (int i = 0; i < n; i++) {
        processes[i].iPID = i + 1;
        cout << "Enter arrival time and burst time for process " << i + 1 << ": ";
        cin >> processes[i].iArrival >> processes[i].iBurst;
        processes[i].remainingBurst = processes[i].iBurst; // Initialize remaining burst time
        processes[i].iStart = processes[i].iFinish = -1;    // Initialize start and finish times
    }

    calculateRR(processes, gantt, timeQuantum);
    printResults(processes, gantt);

    return 0;
}
```

Code : RR

Test Case 1

PID	Arrival Time	Burst Time
1	0	8
2	1	4
3	2	9
4	3	5
5	4	2

Tính toán chi tiết (RR với Time Quantum = 4):

- **Tại thời điểm** 0**:**
  - P1 chạy từ 0 đến 4, còn lại 4.
- **Tại thời điểm** 4**:**
  - P2 chạy từ 4 đến 8, hoàn thành.
  - P2 có Finish Time = 8, Waiting Time = 3, Turnaround Time = 7.
- **Tại thời điểm** 8**:**
  - P3 chạy từ 8 đến 12, còn lại 5.
- **Tại thời điểm** 12**:**
  - P4 chạy từ 12 đến 16, còn lại 1.
- **Tại thời điểm** 16**:**
  - P5 chạy từ 16 đến 18, hoàn thành.
  - P5 có Finish Time = 18, Waiting Time = 12, Turnaround Time = 14.
- **Tại thời điểm** 18**:**
  - P1 chạy tiếp từ 18 đến 22, hoàn thành.
  - P1 có Finish Time = 22, Waiting Time = 14, Turnaround Time = 22.
- **Tại thời điểm** 22**:**
  - P3 chạy tiếp từ 22 đến 26, còn lại 1.
- **Tại thời điểm** 26**:**
  - P4 chạy từ 26 đến 27, hoàn thành.
  - P4 có Finish Time = 27, Waiting Time = 19, Turnaround Time = 24.
- **Tại thời điểm** 27**:**
  - P3 chạy tiếp từ 27 đến 28, hoàn thành.
  - P3 có Finish Time = 28, Waiting Time = 17, Turnaround Time = 26.

Kết quả:

PID	Arrival Time	Burst Time	Finish Time	Waiting Time	Turnaround Time
1	0	8	22	14	22
2	1	4	8	3	7
3	2	9	28	17	26
4	3	5	27	19	24
5	4	2	18	12	14

Trung bình Waiting Time: 13.0
Trung bình Turnaround Time: 18.6

```
dplayergod  main  cd "/home/dplayergod/UIT/OS/LAB/LAB04/" && g++ RR.cpp -o RR && "/home/dplayergod/UIT/OS/LAB/LAB04/"RR
Enter number of processes: 5
Enter time quantum: 4
Enter arrival time and burst time for process 1: 0 8
Enter arrival time and burst time for process 2: 1 4
Enter arrival time and burst time for process 3: 2 9
Enter arrival time and burst time for process 4: 3 5
Enter arrival time and burst time for process 5: 4 2

Gantt Chart:
| P1 (0,4) | P2 (4,8) | P3 (8,12) | P4 (12,16) | P5 (16,18) | P1 (18,22) | P3 (22,26) | P4 (26,27) | P3 (27,28) |

PID       Arrival   Burst     Start     Finish    Waiting   Response  Turnaround
1         0         8         0         22        14        0         22
2         1         4         4         8         3         3         7
3         2         9         8         28        17        6         26
4         3         5         12        27        19        9         24
5         4         2         16        18        12        12        14

Average Waiting Time: 13
Average Turnaround Time: 18.6
```

Hình 7: Test 01

Test Case 2

PID	Arrival Time	Burst Time
1	0	6
2	1	8
3	2	7
4	3	3
5	4	4

Tính toán chi tiết (RR với Time Quantum = 4):

- **Tại thời điểm** 0**:**
  - P1 chạy từ 0 đến 4, còn lại 2.
- **Tại thời điểm** 4**:**
  - P2 chạy từ 4 đến 8, còn lại 4.
- **Tại thời điểm** 8**:**
  - P3 chạy từ 8 đến 12, còn lại 3.
- **Tại thời điểm** 12**:**
  - P4 chạy từ 12 đến 15, hoàn thành.
  - P4 có Finish Time = 15, Waiting Time = 9, Turnaround Time = 12.
- **Tại thời điểm** 15**:**
  - P5 chạy từ 15 đến 19, hoàn thành.
  - P5 có Finish Time = 19, Waiting Time = 11, Turnaround Time = 15.
- **Tại thời điểm** 19**:**
  - P1 chạy tiếp từ 19 đến 21, hoàn thành.
  - P1 có Finish Time = 21, Waiting Time = 15, Turnaround Time = 21.
- **Tại thời điểm** 21**:**
  - P2 chạy tiếp từ 21 đến 25, hoàn thành.
  - P2 có Finish Time = 25, Waiting Time = 16, Turnaround Time = 24.
- **Tại thời điểm** 25**:**
  - P3 chạy tiếp từ 25 đến 28, hoàn thành.
  - P3 có Finish Time = 28, Waiting Time = 19, Turnaround Time = 26.

Kết quả:

PID	Arrival Time	Burst Time	Finish Time	Waiting Time	Turnaround Time
1	0	6	21	15	21
2	1	8	25	16	24
3	2	7	28	19	26
4	3	3	15	9	12
5	4	4	19	11	15

Trung bình Waiting Time: 14.0
Trung bình Turnaround Time: 19.6

```
dplayergod  main  cd "/home/dplayergod/UIT/OS/LAB/LAB04/" && g++ RR.cpp -o RR && "/home/dplayergod/UIT/OS/LAB/LAB04/"RR
Enter number of processes: 5
Enter time quantum: 4
Enter arrival time and burst time for process 1: 0 6
Enter arrival time and burst time for process 2: 1 8
Enter arrival time and burst time for process 3: 2 7
Enter arrival time and burst time for process 4: 3 3
Enter arrival time and burst time for process 5: 4 4

Gantt Chart:
| P1 (0,4) | P2 (4,8) | P3 (8,12) | P4 (12,15) | P5 (15,19) | P1 (19,21) | P2 (21,25) | P3 (25,28) |

PID       Arrival   Burst     Start     Finish    Waiting   Response  Turnaround
1         0         6         0         21        15        0         21
2         1         8         4         25        16        3         24
3         2         7         8         28        19        6         26
4         3         3         12        15        9         9         12
5         4         4         15        19        11        11        15

Average Waiting Time: 14
Average Turnaround Time: 19.6
```

Hình 8: Test 02

Test Case 3

PID	Arrival Time	Burst Time
1	0	5
2	1	3
3	2	8
4	3	6
5	4	4

Tính toán chi tiết (RR với Time Quantum = 4):

- **Tại thời điểm** 0**:**
  - P1 chạy từ 0 đến 4, còn lại 1.
- **Tại thời điểm** 4**:**
  - P2 chạy từ 4 đến 7, hoàn thành.
  - P2 có Finish Time = 7, Waiting Time = 3, Turnaround Time = 6.
- **Tại thời điểm** 7**:**
  - P3 chạy từ 7 đến 11, còn lại 4.
- **Tại thời điểm** 11**:**
  - P4 chạy từ 11 đến 15, còn lại 2.
- **Tại thời điểm** 15**:**
  - P5 chạy từ 15 đến 19, hoàn thành.
  - P5 có Finish Time = 19, Waiting Time = 11, Turnaround Time = 15.
- **Tại thời điểm** 19**:**
  - P1 chạy tiếp từ 19 đến 20, hoàn thành.
  - P1 có Finish Time = 20, Waiting Time = 15, Turnaround Time = 20.
- **Tại thời điểm** 20**:**
  - P3 chạy tiếp từ 20 đến 24, hoàn thành.
  - P3 có Finish Time = 24, Waiting Time = 14, Turnaround Time = 22.
- **Tại thời điểm** 24**:**
  - P4 chạy tiếp từ 24 đến 26, hoàn thành.
  - P4 có Finish Time = 26, Waiting Time = 17, Turnaround Time = 23.

Kết quả:

PID	Arrival Time	Burst Time	Finish Time	Waiting Time	Turnaround Time
1	0	5	20	15	20
2	1	3	7	3	6
3	2	8	24	14	22
4	3	6	26	17	23
5	4	4	19	11	15

Trung bình Waiting Time: 12.0
Trung bình Turnaround Time: 17.2

```
dplayergod  main  cd "/home/dplayergod/UIT/OS/LAB/LAB04/" && g++ RR.cpp -o RR && "/home/dplayergod/UIT/OS/LAB/LAB04/"RR
Enter number of processes: 5
Enter time quantum: 4
Enter arrival time and burst time for process 1: 0 5
Enter arrival time and burst time for process 2: 1 3
Enter arrival time and burst time for process 3: 2 8
Enter arrival time and burst time for process 4: 3 6
Enter arrival time and burst time for process 5: 4 4

Gantt Chart:
| P1 (0,4) | P2 (4,7) | P3 (7,11) | P4 (11,15) | P5 (15,19) | P1 (19,20) | P3 (20,24) | P4 (24,26) |

PID       Arrival   Burst     Start     Finish    Waiting   Response  Turnaround
1         0         5         0         20        15        0         20
2         1         3         4         7         3         3         6
3         2         8         7         24        14        5         22
4         3         6         11        26        17        8         23
5         4         4         15        19        11        11        15

Average Waiting Time: 12
Average Turnaround Time: 17.2
```

Hình 9: Test 03

5 Phụ lục

- Liên kết `github` : link.