Báo cáo thực hành môn Hệ điều hành - Giảng viên: Lê Hoài Nghĩa.

Họ và tên	MSSV	Lớp
Lại Quan Thiên	22521385	IT007.O211.1
Đặng Đức Tài	22521270	
Mai Nguyễn Nam Phương	22521164	
Phùng Trần Thế Nam	21522366	

HỆ ĐIỀU HÀNH
BÁO CÁO LAB 4

CHECKLIST

3.5. BÀI TẬP THỰC HÀNH

	BT 1	BT 2
Vẽ lưu đồ giải thuật	☒	☒
Chạy tay lưu đồ giải thuật	☒	☒
Hiện thực code	☒	☒
Chạy code và kiểm chứng	☒	☒

3.6. BÀI TẬP ÔN TẬP

	BT 1
Vẽ lưu đồ giải thuật	☒
Chạy tay lưu đồ giải thuật	☒
Hiện thực code	☒
Chạy code và kiểm chứng	☒

Tự chấm điểm: 9.5/10

Báo cáo thực hành môn Hệ điều hành - Giảng viên: Lê Hoài Nghĩa.

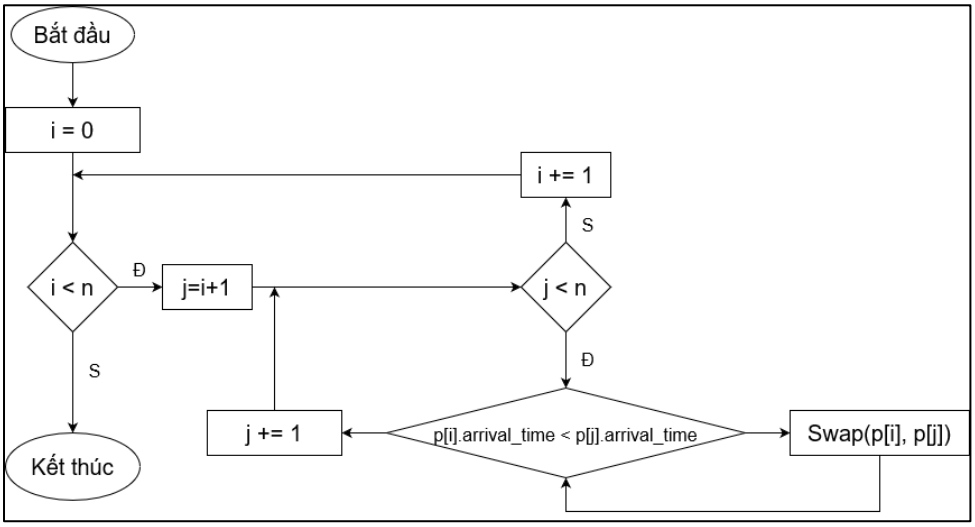
**Lưu ý: Xuất báo cáo theo định dạng PDF, đặt tên theo cú pháp:*
<Tên nhóm>_LAB4.pdf

3.5. BÀI TẬP THỰC HÀNH

1. Giải thuật Shortest-Job-First

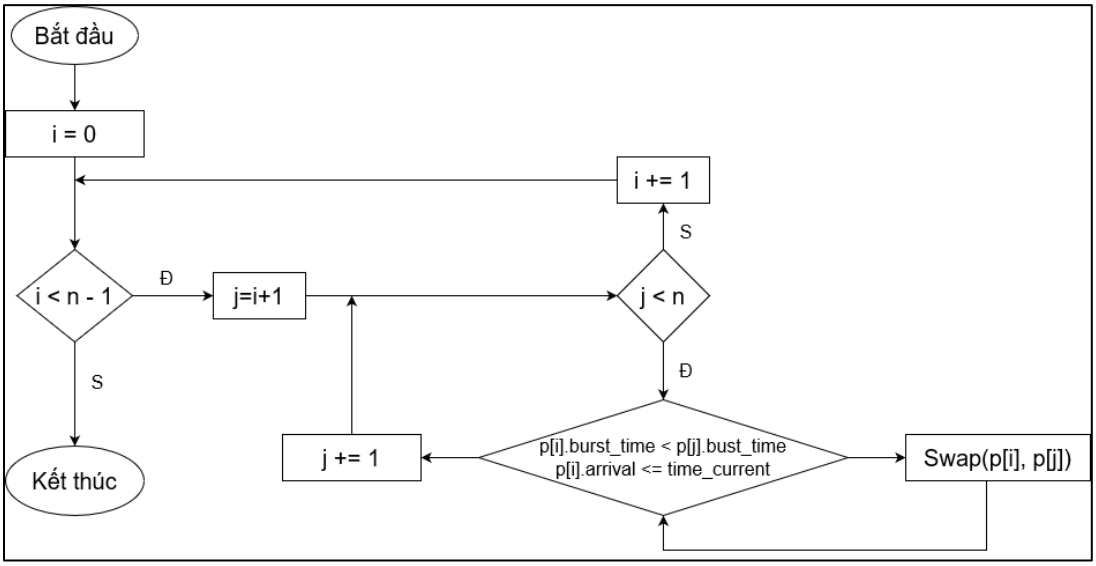
1.1. Vẽ lưu đồ thuật toán

1.1.1. Hàm sort các tiến trình theo arrival_time



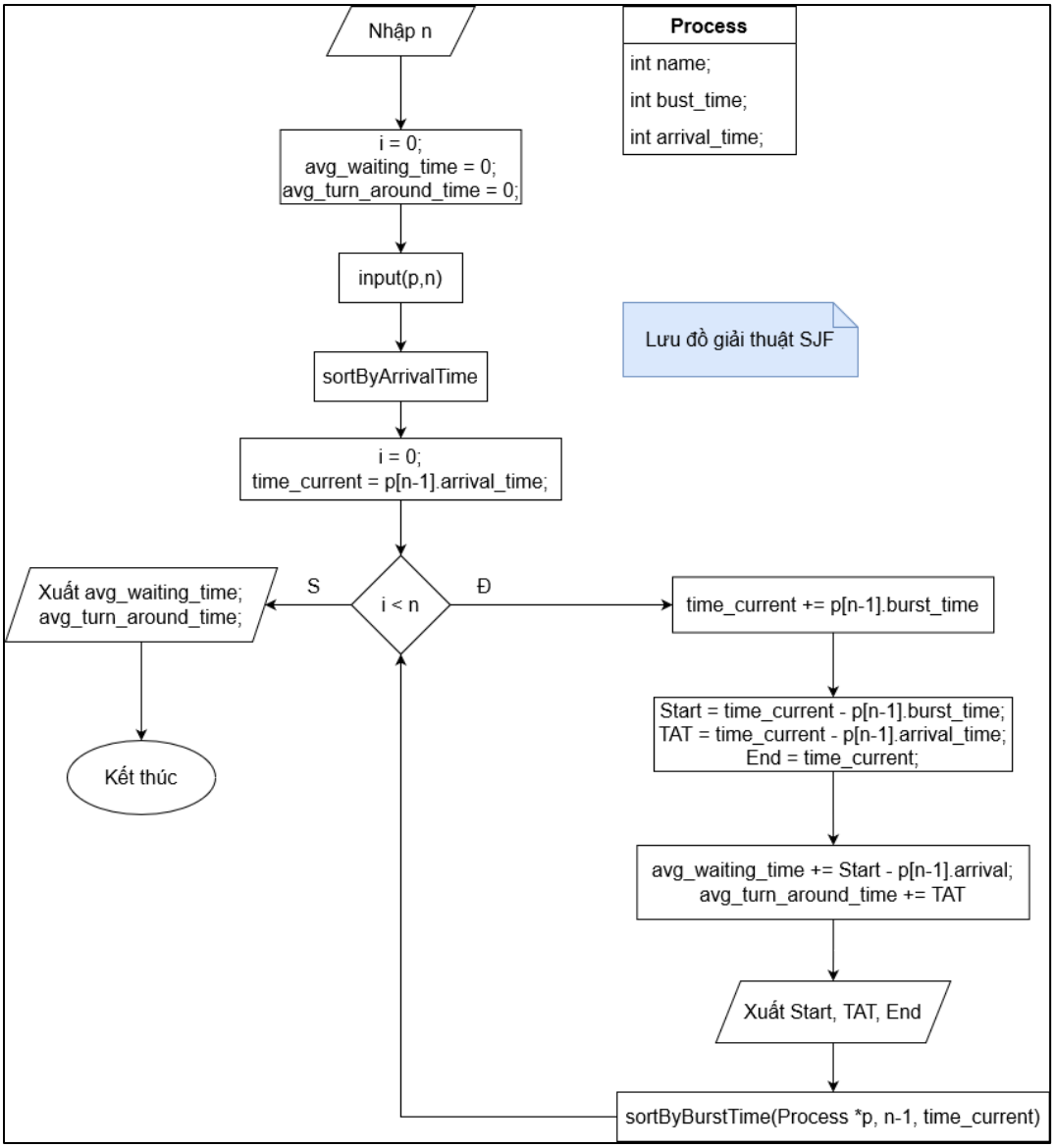
Giải thích: Chúng ta sẽ sử dụng thuật toán nổi bọt để lọc quá hết các cặp phần tử và sắp xếp lại theo thứ tự có arrival_time giảm dần.

1.1.2. Hàm sort các tiến trình theo burst_time



Giải thích: ***Tương tự chúng ta sẽ sử dụng thuật toán nổi bọt để lọc quá hết các cặp phần ử và sắp xếp lại các tiến trình chưa xử lý theo thứ tự có burst_time tăng dần. Và tá xét điều kiện là arrival_time phải bé hơn hoặc bằng thời gian hiện tại đang thực thi.***

1.1.3. Lưu đồ giải thuật SJF



Giải thích:

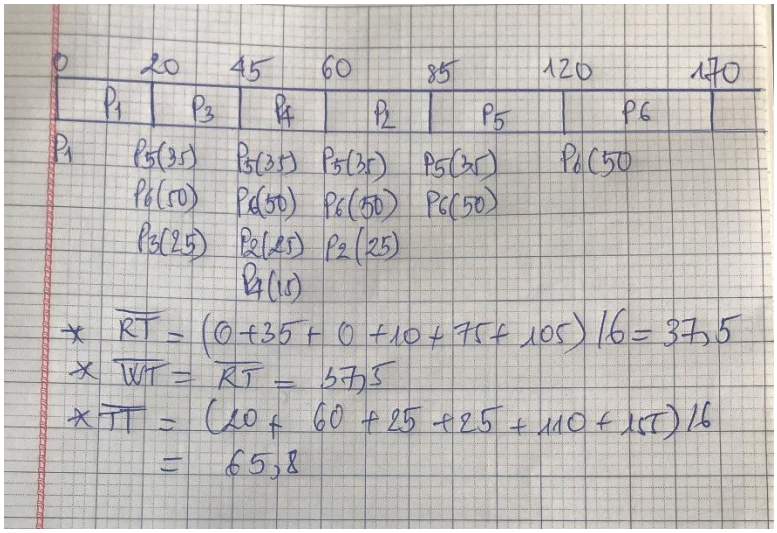
- Đầu tiên ta sẽ tạo ra một struct tên process với 3 thông tin cơ bản như trên. Sau đó chúng ta khai báo thêm 2 biến toàn cục là biến tổng thời gian đợi và thời gian thực hiện trong hệ thống.
- Tiến hành nhập n là số process, Sau đó dùng hàm Input để nhập các thông tin của các process.
- Sắp xếp lại các tiến trình bằng hàm SortByArrivalTime. Sau đó khai báo thêm biến time_current = thời gian vào của tiến trình có arrival_time bé nhất.
- Cho các tiến trình vào vòng lặp lấy ra phần tử ngoài cùng lúc này tiến trình đầu tiên được thực thi, time_current lúc này đã được cộng thêm busrt_time của tiến trình đó lúc này time_current là thời gian kết thúc của tiến trình trong vòng lặp.
- Tiến hành tính toán các thời gian Start, TAT, End.
- Sắp xếp lại các tiến trình còn lại dựa vào hàm sortByBurstTime và lặp lại đối với các tiến trình còn lại.

1.2. Chạy tay lưu đồ giải thuật SJF

- Cho dữ liệu như bảng dưới:

Process	**Arrival Time**	**Burst Time**
P1	0	20
P2	25	25
P3	20	25
P4	35	15
P5	10	35
P6	15	50

- Kết quả khi chạy tay giải thuật:


0 20 45 60 85 120 170
P1 P3 P4 P2 P5 P6
P1 P5(35) P3(25) P4(15) P2(25) P5(35) P6(50)
P6(50) P6(50) P6(50) P6(50)
P3(25) P2(25) P2(25)
P4(15)
* RT = (0+35+0+10+75+105)/6 = 37,5
* WT = RT = 37,5
* TT = (20+60+25+25+110+157)/6
= 65,8

1.3. Hiện thực code

sjf.cpp

```cpp
#include <stdio.h>
#include <iostream>
#include <queue>
#include <iomanip>

using namespace std;

struct Process
{
    int name;
    int burst_time;
    int arrival_time;
};

static double avg_turn_around_time = 0;
static double avg_waiting_time = 0;

void swap(Process &p1, Process &p2)
{
    Process tmp;
    tmp = p1;
    p1 = p2;
    p2 = tmp;
}

void sortByArrivalTime(Process *p, int n)
{
    for (int i = 0; i < n; i++)
    {
        for (int j = i + 1; j < n; j++)
        {
            if (p[i].arrival_time < p[j].arrival_time)
                swap(p[i], p[j]);
        }
    }
}

void Input(Process *p, int n)
{
    for (int i = 0; i < n; i++)
    {
        cout << "-----------------------\n";
        cout << "Nhap ID process: ";
        cin >> p[i].name;
        srand(time(NULL));
        cout << "Arrival Time: ";
        p[i].arrival_time = rand() % 21;
        cout << p[i].arrival_time << endl;
        cout << "Burst Time: ";
       p[i].burst_time = rand() % 11 + 2;
        cout << p[i].burst_time << endl;

    }
}
```

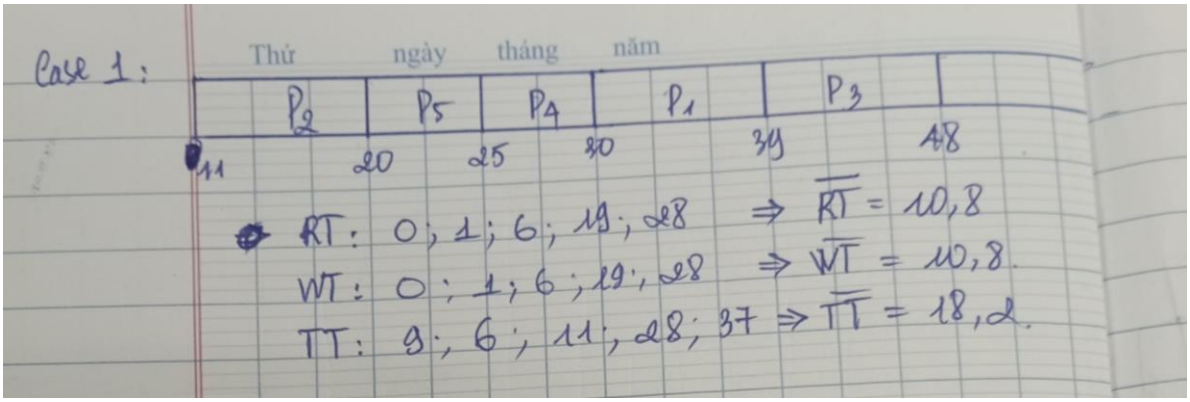
sjf.cpp

```cpp
void sortByBurstTime(Process *p, int n, int time_current)
{
    for (int i = 0; i < n - 1; i++)
    {
        for (int j = i + 1; j < n; j++)
        {
            if (p[i].burst_time < p[j].burst_time && p[i].arrival_time <= time_current)
            {
                swap(p[i], p[j]);
            }
        }
    }
}

void SelectionFunction(Process *p, int n)
{

    int time_current;

    // Sort theo arrival time
    sortByArrivalTime(p, n);

    // Ham lua chon quyet dinh xem process nao vao queue truoc;
    time_current = p[n - 1].arrival_time;

    for (int i = 0; i < n; n--)
    {
        time_current += p[n - 1].burst_time;

        avg_waiting_time += time_current - p[n - 1].arrival_time - p[n - 1].burst_time;

        avg_turn_around_time += (time_current - p[n - 1].arrival_time);

        cout << setw(10) << left << p[n - 1].name
             << setw(10) << left << p[n - 1].arrival_time
             << setw(10) << left << p[n - 1].burst_time
             << setw(10) << left << time_current - p[n - 1].burst_time
             << setw(10) << left << time_current - p[n - 1].arrival_time
             << setw(10) << left << time_current
             << endl;
        sortByBurstTime(p, n - 1, time_current);
    }
}

int main()
{
    Process *p = new Process[100];
    queue<Process> pQueue;
    int n;

    cout << "Nhap so luong process: ";
    cin >> n;

    Input(p, n);

    cout << "\n----------------------\n";

    cout << left << setw(10) << "PName"
         << left << setw(10) << "Arrtime"
         << left << setw(10) << "Bursttime"
         << left << setw(10) << "Start"
         << left << setw(10) << "TAT"
         << left << setw(10) << "Finish" << endl;
    SelectionFunction(p, n);

    cout << "----------------------\n";

    cout << "Thoi gian dap ung trung binh: " << avg_waiting_time / n << endl;

    cout << "Thoi gian doi trung binh: " << avg_waiting_time / n << endl;

    cout << "Thoi gian hoan thanh trung binh: " << avg_turn_around_time / n << endl;

    return 0;
}
```

1.4. Trình bày tính đúng đắn của code bằng cách chạy ít nhất 03 test case, mỗi test case 5 tiến trình, so sánh kết quả chạy tay và chạy code

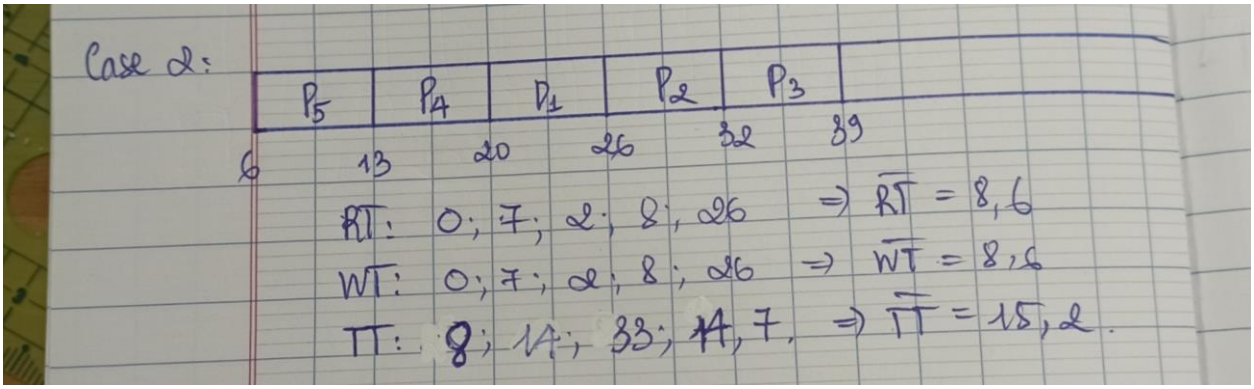
Test Case 1:

```
wanthinnn@laiquanthien-22521385:~/Downloads/dieuphoitientrinh$ ./sjf
Nhap so luong process: 5
----------------------
Nhap ID process: 1
Arrival Time: 11
Burst Time: 9
----------------------
Nhap ID process: 2
Arrival Time: 11
Burst Time: 9
----------------------
Nhap ID process: 3
Arrival Time: 11
Burst Time: 9
----------------------
Nhap ID process: 4
Arrival Time: 19
Burst Time: 5
----------------------
Nhap ID process: 5
Arrival Time: 19
Burst Time: 5

----------------------
PName     Arrtime   Bursttime Start     TAT       Finish
2         11        9         11        9         20
5         19        5         20        6         25
4         19        5         25        11        30
1         11        9         30        28        39
3         11        9         39        37        48
----------------------
Thoi gian dap ung trung binh: 10.8
Thoi gian doi trung binh: 10.8
Thoi gian hoan thanh trung binh: 18.2
wanthinnn@laiquanthien-22521385:~/Downloads/dieuphoitientrinh$
```



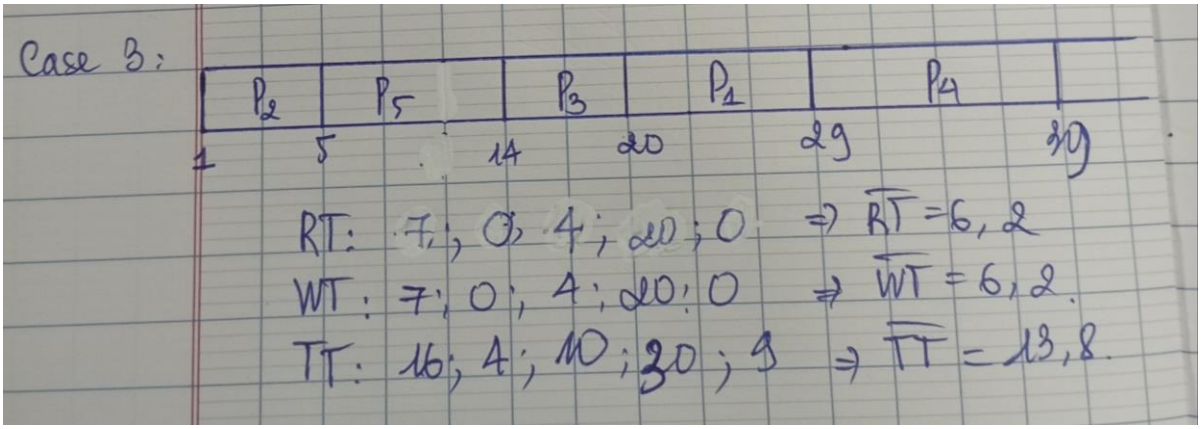
Test Case 2:

```
wanthinnn@laiquanthien-22521385:~/Downloads/dieuphoitientrinh$ ./sjf
Nhap so luong process: 5
----------------------
Nhap ID process: 1
Arrival Time: 18
Burst Time: 6
----------------------
Nhap ID process: 2
Arrival Time: 18
Burst Time: 6
----------------------
Nhap ID process: 3
Arrival Time: 6
Burst Time: 7
----------------------
Nhap ID process: 4
Arrival Time: 6
Burst Time: 7
----------------------
Nhap ID process: 5
Arrival Time: 6
Burst Time: 7

----------------------
PName     Arrtime   Bursttime Start     TAT       Finish
5         6         7         6         7         13
4         6         7         13        14        20
1         18        6         20        8         26
2         18        6         26        14        32
3         6         7         32        33        39
----------------------
Thoi gian dap ung trung bình: 8.6
Thoi gian doi trung bình: 8.6
Thoi gian hoan thanh trung binh: 15.2
wanthinnn@laiquanthien-22521385:~/Downloads/dieuphoitientrinh$
```



Test Case 3:

```
wanthinnn@laiquanthien-22521385:~/Downloads/dieuphoitientrinh$ ./sjf
Nhap so luong process: 5
----------------------
Nhap ID process: 1
Arrival Time: 13
Burst Time: 9
----------------------
Nhap ID process: 2
Arrival Time: 1
Burst Time: 4
----------------------
Nhap ID process: 3
Arrival Time: 10
Burst Time: 6
----------------------
Nhap ID process: 4
Arrival Time: 9
Burst Time: 10
----------------------
Nhap ID process: 5
Arrival Time: 5
Burst Time: 9

----------------------
PName     Arrtime   Bursttime Start     TAT       Finish
2         1         4         1         4         5
5         5         9         5         9         14
3         10        6         14        10        20
1         13        9         20        16        29
4         9         10        29        30        39
----------------------
Thoi gian dap ung trung binh: 6.2
Thoi gian doi trung binh: 6.2
Thoi gian hoan thanh trung binh: 13.8
wanthinnn@laiquanthien-22521385:~/Downloads/dieuphoitientrinh$
```

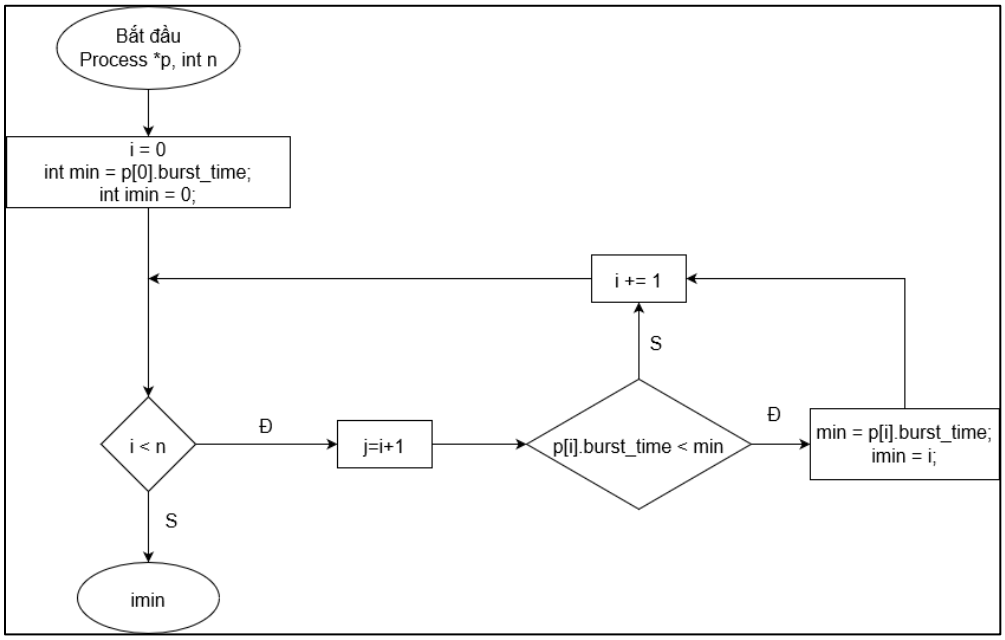


=> Nhận xét: số liệu từ giải tay và code là giống nhau, không có sai số *(số liệu khi chạy trên code được làm tròn)*

2. Giải thuật Shortest-Remaining-Time-First

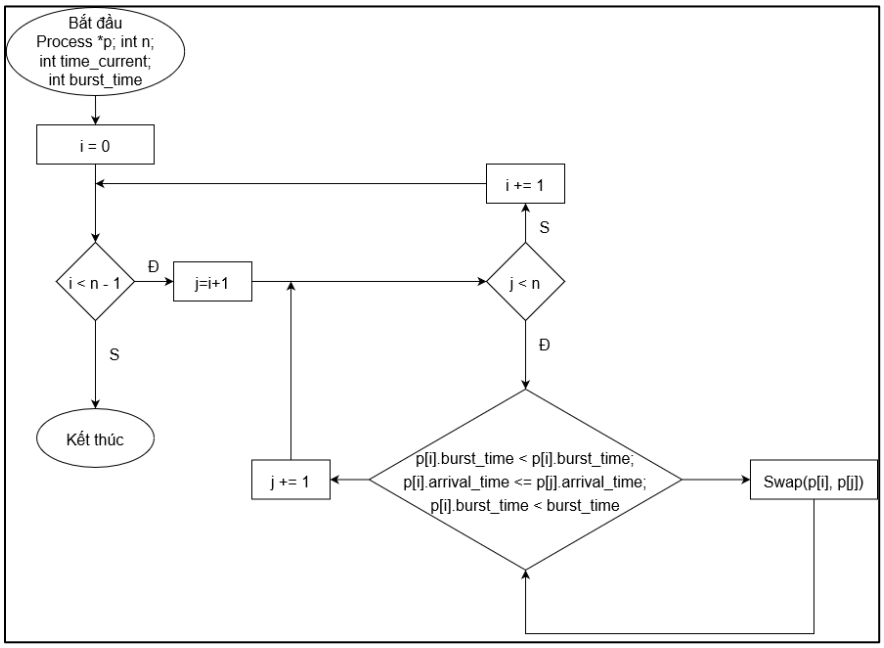
2.1. Lưu đồ giải thuật SRTF

2.1.1. Hàm tìm ra tiến trình có burst time nhỏ nhất.



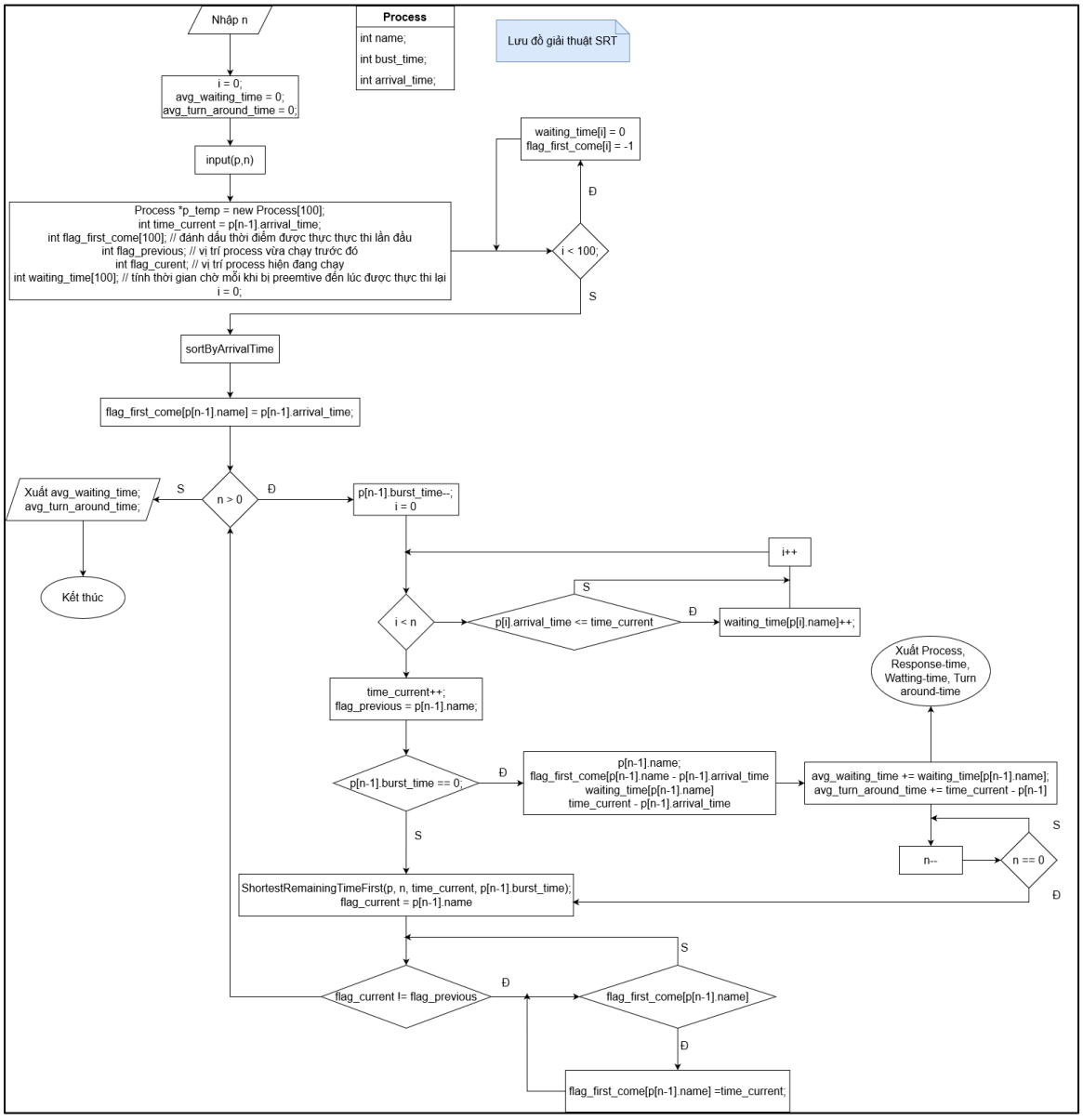
- Giải thích: Hàm có chức năng tìm ra tiến trình có burstime nhỏ nhất bằng cách lọc qua tất cả các tiến trình trong hàng đợi.

2.1.2. Hàm sắp xếp các tiến trình dựa theo tiến trình có burst_time nhỏ hơn burst_time của tiến trình đang thực thi



- Giải thích: Hàm dùng phương pháp nổi bọt để lọc qua các cặp tiến trình và sort các giá trị có burst_time nhỏ hơn burst time của tiến trình đang được thực thi.

2.1.3. Lưu đồ giải thuật SRTF

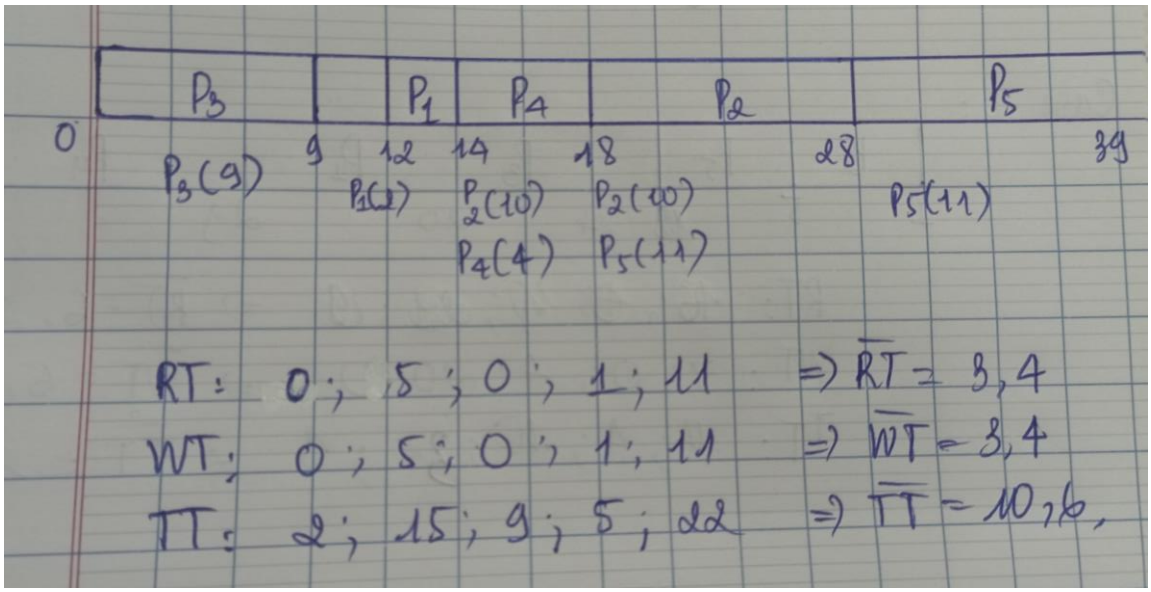


- Giải thích Lưu đồ giải thuật SRTF

- ***Các bước đầu sẽ là tạo struct và tiến hành nhập các process tương tự như giải thuật SJF.***
- ***Sau đó ta sẽ có các biến như là time_current là timeline của chương trình, flag_first_com là list đánh dấu các thời điểm thực thi lần đầu.***
- ***flag_previous: Vị trí của process vừa chạy trước đó, lag_current: vị trí của tiến trình dang chạy; waitting_time: là thời gian chờ mỗi khi bị preemtive đến lúc được thực thi lại.***
- ***Ta chạy hàm for cho các mảng waiting_time và flag_first_come để đánh dấu. -1 là chỉ truy cập 1 lần.***
- ***Sau đó sử dụng hàm SortByArrivalTime để sort tiến trinh.***
- ***Duyệt từ cuối lên. Ta xép từ từ chậm rãi. Hàm for đầu tiên có tác dụng là tăng waiting time khi processs đã đến hàng đượi mà chưa được thực thi.***
- ***Tăng timeline lên dần, và lưu tên process sắp rời đi.***
- ***Với hàm if tiếp theo là nếu đã thực thi hết, không còn burst thì xuất trạng thái. Và ta tính các thông tin Start, TAT, End và cộng dồn thời gian chờ và thời gian hoàn thành. Sau đó giảm n-- để thu hẹp các tiến trình. Khi nào n = 0 thì thoát vòng lặp.***
- ***Dùng Hàm ShortestRemainingTimeFirst(p, n, time_current, p[n-1] .burst_time) để chọn ra các tiến trình có burst < burst còn lại của p[flag_current].***
- ***Hàm if ở cuối có nghĩa là nếu xảy ra trường hợp chuyển ngữ cảnh thì thì thời điểm đánh dấu sẽ bằng timeline chương trình.***

2.2. Chạy tay lưu đồ giải thuật SRTF

```
#pName  AT      BT      ST      CT      TAT     WT      RT
1       12      2       12      14      2       0       0
2       13      10      18      28      15      5       5
3       0       9       0       9       9       0       0
4       13      4       14      18      5       1       1
5       17      11      28      39      22      11      11

Thoi gian hoan thanh trung binh: 10.60
Thoi gian dap ung trung binh: 3.40
Thoi gian doi trung binh: 3.40
```



2.3. Thực hiện code cho giải thuật SRTF

srtf.cpp

```cpp
#include <iostream>
#include <algorithm>
#include <iomanip>
#include <string.h>
#include <cstdlib>
#include <ctime>
using namespace std;

struct process
{
    int pid;
    int arrival_time;
    int burst_time;
    int start_time;
    int completion_time;
    int turnaround_time;
    int waiting_time;
    int response_time;
};

int main()
{
    srand(time(0));
    int n;
    struct process p[100];
    float avg_turnaround_time;
    float avg_waiting_time;
    float avg_response_time;
    float cpu_utilisation;
    int total_turnaround_time = 0;
    int total_waiting_time = 0;
    int total_response_time = 0;
    int total_idle_time = 0;
    float throughput;
    int burst_remaining[100];
    int is_completed[100];
    memset(is_completed, 0, sizeof(is_completed));
    cout << setprecision(2) << fixed;
    cout << "Nhap so luong process: ";
    cin >> n;

    for (int i = 0; i < n; i++)
    {
        cout << "----------------------------" << endl;
        cout << "Nhap ID process: ";
        cin >> p[i].pid;
        p[i].arrival_time = rand() % 21;
        cout << "Arrival time: " << p[i].arrival_time << endl;
        p[i].burst_time = rand() % 11 + 2;
        cout << "Burst time: " << p[i].burst_time << endl;
        burst_remaining[i] = p[i].burst_time;
        cout << endl;
    }
    int current_time = 0;
    int completed = 0;
    int prev = 0;

    while (completed != n)
    {
        int idx = -1;
        int mn = 10000000;
        for (int i = 0; i < n; i++)
        {
            if (p[i].arrival_time <= current_time && is_completed[i] == 0)
            {
                if (burst_remaining[i] < mn)
                {
                    mn = burst_remaining[i];
                    idx = i;
                }
                if (burst_remaining[i] == mn)
                {
                    if (p[i].arrival_time < p[idx].arrival_time)
                    {
                        mn = burst_remaining[i];
                        idx = i;
                    }
                }
            }
        }
        //end part 1
```

srtf.cpp

```
        //part 2
        if (idx != -1)
        {
            if (burst_remaining[idx] == p[idx].burst_time)
            {
                p[idx].start_time = current_time;
                total_idle_time += p[idx].start_time - prev;
            }
            burst_remaining[idx] -= 1;
            current_time++;
            prev = current_time;

            if (burst_remaining[idx] == 0)
            {
                p[idx].completion_time = current_time;
                p[idx].turnaround_time = p[idx].completion_time - p[idx].arrival_time;
                p[idx].waiting_time = p[idx].turnaround_time - p[idx].burst_time;
                p[idx].response_time = p[idx].start_time - p[idx].arrival_time;

                total_turnaround_time += p[idx].turnaround_time;
                total_waiting_time += p[idx].waiting_time;
                total_response_time += p[idx].response_time;

                is_completed[idx] = 1;
                completed++;
            }
        }
        else
        {
            current_time++;
        }
    }
    int min_arrival_time = 10000000;
    int max_completion_time = -1;
    for (int i = 0; i < n; i++)
    {
        min_arrival_time = min(min_arrival_time, p[i].arrival_time);
        max_completion_time = max(max_completion_time, p[i].completion_time);
    }

    avg_turnaround_time = (float)total_turnaround_time / n;
    avg_waiting_time = (float)total_waiting_time / n;
    avg_response_time = (float)total_response_time / n;
    cpu_utilisation = ((max_completion_time - total_idle_time) / (float)max_completion_time) * 100;
    throughput = float(n) / (max_completion_time - min_arrival_time);

    cout << endl
         << endl;

    cout << "#pName\t"
         << "AT\t"
         << "BT\t"
         << "ST\t"
         << "CT\t"
         << "TAT\t"
         << "WT\t"
         << "RT\t"
         << "\n"
         << endl;

    for (int i = 0; i < n; i++)
    {
        cout
            << p[i].pid << "\t"
            << p[i].arrival_time << "\t"
            << p[i].burst_time << "\t"
            << p[i].start_time << "\t"
            << p[i].completion_time << "\t"
            << p[i].turnaround_time << "\t"
            << p[i].waiting_time << "\t"
            << p[i].response_time << "\t"
            << "\n"
            << endl;
    }
    cout << "Thoi gian hoan thanh trung binh: " << avg_turnaround_time << endl;
    cout << "Thoi gian dap ung trung binh: " <<  avg_response_time << endl;
    cout << "Thoi gian doi trung binh: " << avg_waiting_time << endl;
}
```

Giải thích:

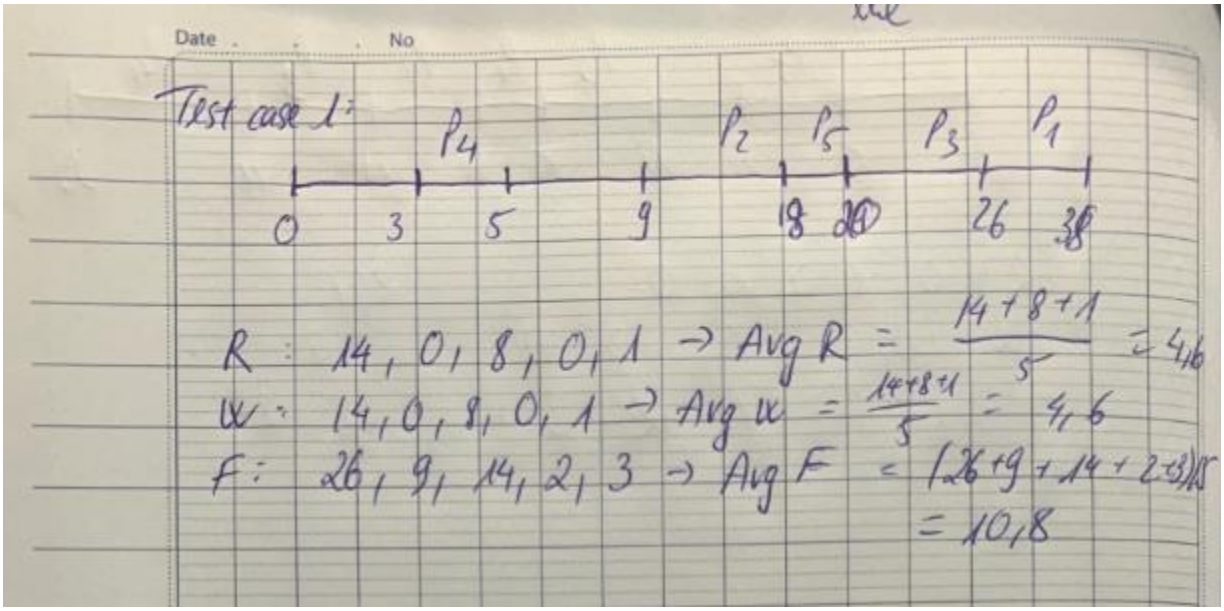
1. **Khai báo thư viện và khai báo cấu trúc process:**
   - Sử dụng các thư viện iostream, algorithm, iomanip, string.h, cstdlib, và ctime.
   - Định nghĩa một cấu trúc process để lưu trữ thông tin của mỗi tiến trình.
2. **Hàm main():**
   - Khởi tạo các biến và mảng cần thiết.
   - số lượng tiến trình từ người dùng.
   - Sử dụng hàm rand() để tạo ngẫu nhiên thời gian đến (arrival_time) và thời gian chạy (burst_time) cho mỗi tiến trình.
   - Khởi tạo mảng burst_remaining[] để theo dõi thời gian chờ đợi còn lại cho mỗi tiến trình.
   - Khởi tạo mảng is_completed[] để theo dõi trạng thái hoàn thành của mỗi tiến trình.
3. **Vòng lặp chính:**
   - Tiến hành lập lịch thực hiện các tiến trình bằng thuật toán SJF.
   - Mỗi lần lặp, chương trình tìm tiến trình có thời gian chờ đợi ngắn nhất và chưa hoàn thành để thực thi.
   - Cập nhật thời gian và thông tin của các tiến trình khi chúng được thực thi.
4. **Tính toán và xuất kết quả:**
   - Tính toán thời gian hoàn thành trung bình, thời gian chờ đợi trung bình, thời gian đáp ứng trung bình và tỷ lệ sử dụng CPU.
   - Tính toán tỷ lệ hoàn thành (throughput).
   - Xuất thông tin của từng tiến trình và các kết quả đã tính toán ra màn hình.

2.4. Trình bày tính đúng đắn của code bằng cách chạy ít nhất 03 test case, mỗi test case 5 tiến trình, so sánh kết quả chạy tay và chạy code

Test Case 1:

```
mainguyennamphuong@namphuong:~/Desktop/Code$ ./test
Nhap so luong process: 5
----------------------------
Nhap ID process: 1
Arrival time: 12
Burst time: 12

----------------------------
Nhap ID process: 2
Arrival time: 9
Burst time: 9

----------------------------
Nhap ID process: 3
Arrival time: 12
Burst time: 6

----------------------------
Nhap ID process: 4
Arrival time: 3
Burst time: 2

----------------------------
Nhap ID process: 5
Arrival time: 17
Burst time: 2


#pName  AT      BT      ST      CT      TAT     WT      RT
1       12      12      26      38      26      14      14
2       9       9       9       18      9       0       0
3       12      6       20      26      14      8       8
4       3       2       3       5       2       0       0
5       17      2       18      20      3       1       1

Thoi gian hoan thanh trung binh: 10.80
Thoi gian dap ung trung binh: 4.60
Thoi gian doi trung binh: 4.60
mainguyennamphuong@namphuong:~/Desktop/Code$
```

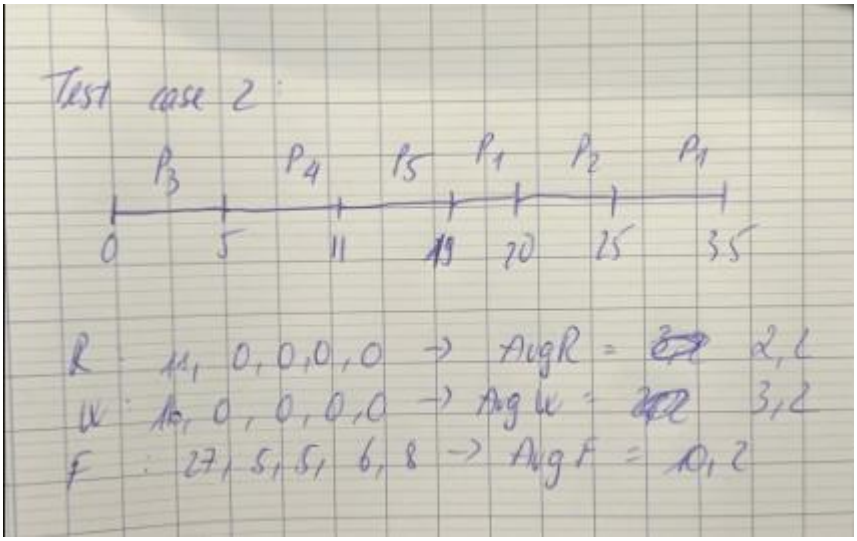
Giải thuật tay:



Test Case 2:

```
nalnguyennamphuong@namphuong:~/Desktop/Code$ ./test
Nhap so luong process: 5
----------------------------
Nhap ID process: 1
Arrival time: 8
Burst time: 11

----------------------------
Nhap ID process: 2
Arrival time: 20
Burst time: 5

----------------------------
Nhap ID process: 3
Arrival time: 0
Burst time: 5

----------------------------
Nhap ID process: 4
Arrival time: 5
Burst time: 6

----------------------------
Nhap ID process: 5
Arrival time: 11
Burst time: 8


#pName  AT      BT      ST      CT      TAT     WT      RT
1       8       11      19      35      27      16      11
2       20      5       20      25      5       0       0
3       0       5       0       5       5       0       0
4       5       6       5       11      6       0       0
5       11      8       11      19      8       0       0

Thoi gian hoan thanh trung binh: 10.20
Thoi gian dap ung trung binh: 2.20
Thoi gian doi trung binh: 3.20
nalnguyennamphuong@namphuong:~/Desktop/Code$
```

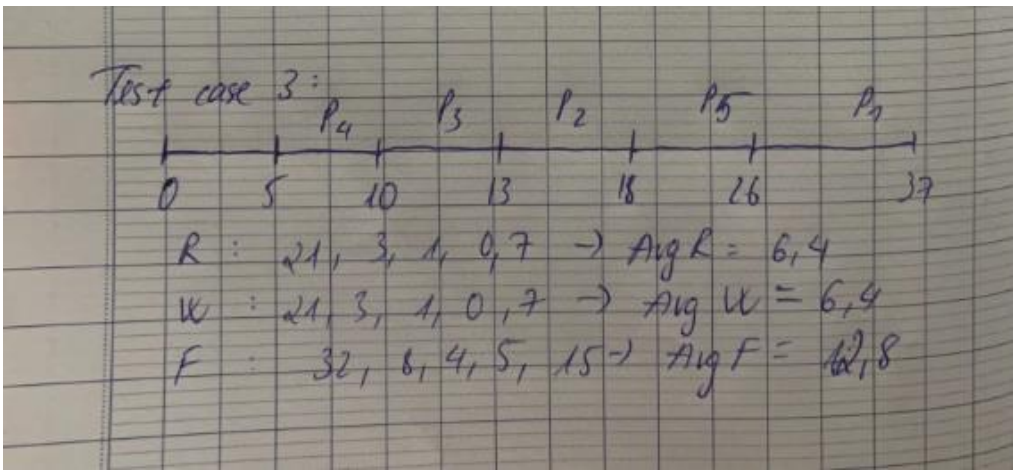
Giải thuật tay:



Test Case 3:

```
mainguyennamphuong@namphuong:~/Desktop/Code$ ./test
Nhap so luong process: 5
-----------------------------
Nhap ID process: 1
Arrival time: 5
Burst time: 11

-----------------------------
Nhap ID process: 2
Arrival time: 10
Burst time: 5

-----------------------------
Nhap ID process: 3
Arrival time: 9
Burst time: 3

-----------------------------
Nhap ID process: 4
Arrival time: 5
Burst time: 5

-----------------------------
Nhap ID process: 5
Arrival time: 11
Burst time: 8


#pName  AT      BT      ST      CT      TAT     WT      RT

1       5       11      26      37      32      21      21

2       10      5       13      18      8       3       3

3       9       3       10      13      4       1       1

4       5       5       5       10      5       0       0

5       11      8       18      26      15      7       7

Thoi gian hoan thanh trung binh: 12.80
Thoi gian dap ung trung binh: 6.40
Thoi gian doi trung binh: 6.40
mainguyennamphuong@namphuong:~/Desktop/Code$
```

Giải thuật tay:

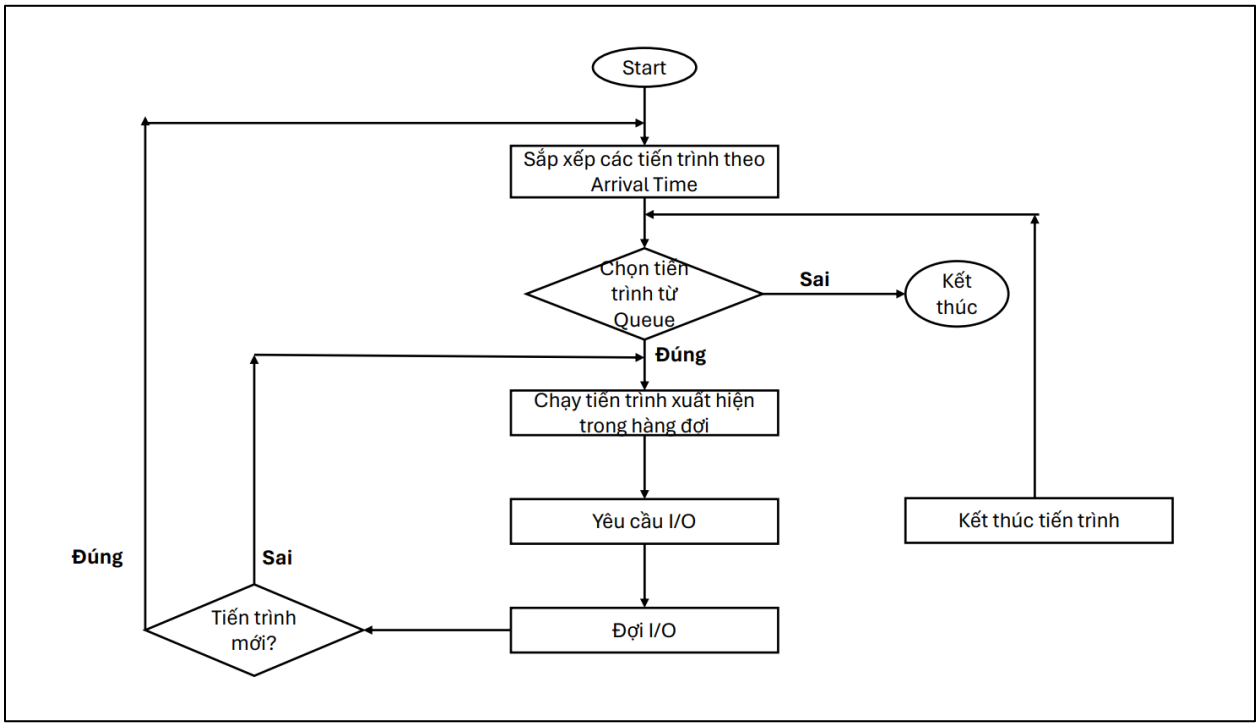


=> Nhận xét: số liệu từ giải tay và code là giống nhau, không có sai số *(số liệu khi chạy trên code được làm tròn)*

3.6. BÀI TẬP ÔN TẬP

Giải thuật Round Robin

1. Lưu đồ giải thuật Round-Robin



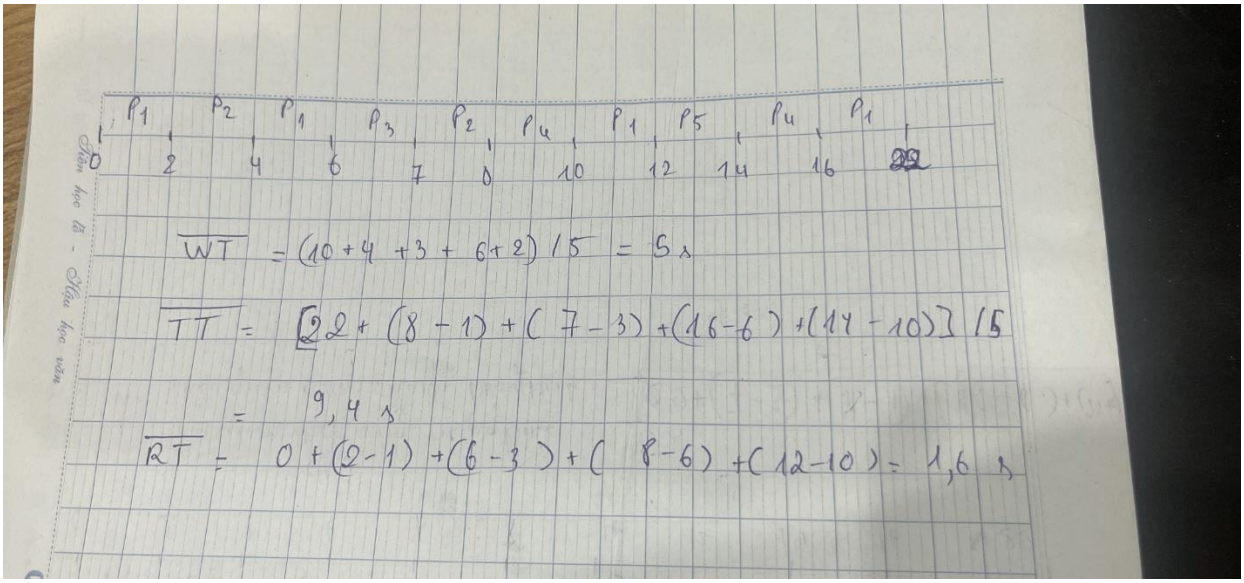
Giải thích:

- Đầu tiên, cần có một hàng đợi. Trong đó, các quy trình sẽ được sắp xếp theo thứ tự đến trước xử lý trước.
- Một giá trị lượng tử được cấp phát để thực hiện một quá trình.
- Tiến trình đầu tiên được thực hiện đến khi kết thúc giá trị lượng tử. Tiếp đến, mỗi ngắt sẽ được tạo ra và trạng thái được lưu.
- Từ CPU chuyển sang quy trình tiếp theo, phương pháp tương tự và được thực hiện tuần hoàn.
- Các bước tương tự được lặp đi lặp lại đến khi quá trình kết thúc.

2. Chạy tay lưu đồ giải thuật RR.

TT	AT	BT
P1	0	12
P2	1	3
P3	3	1
P4	6	4
P5	10	2

Quantumtime = 2



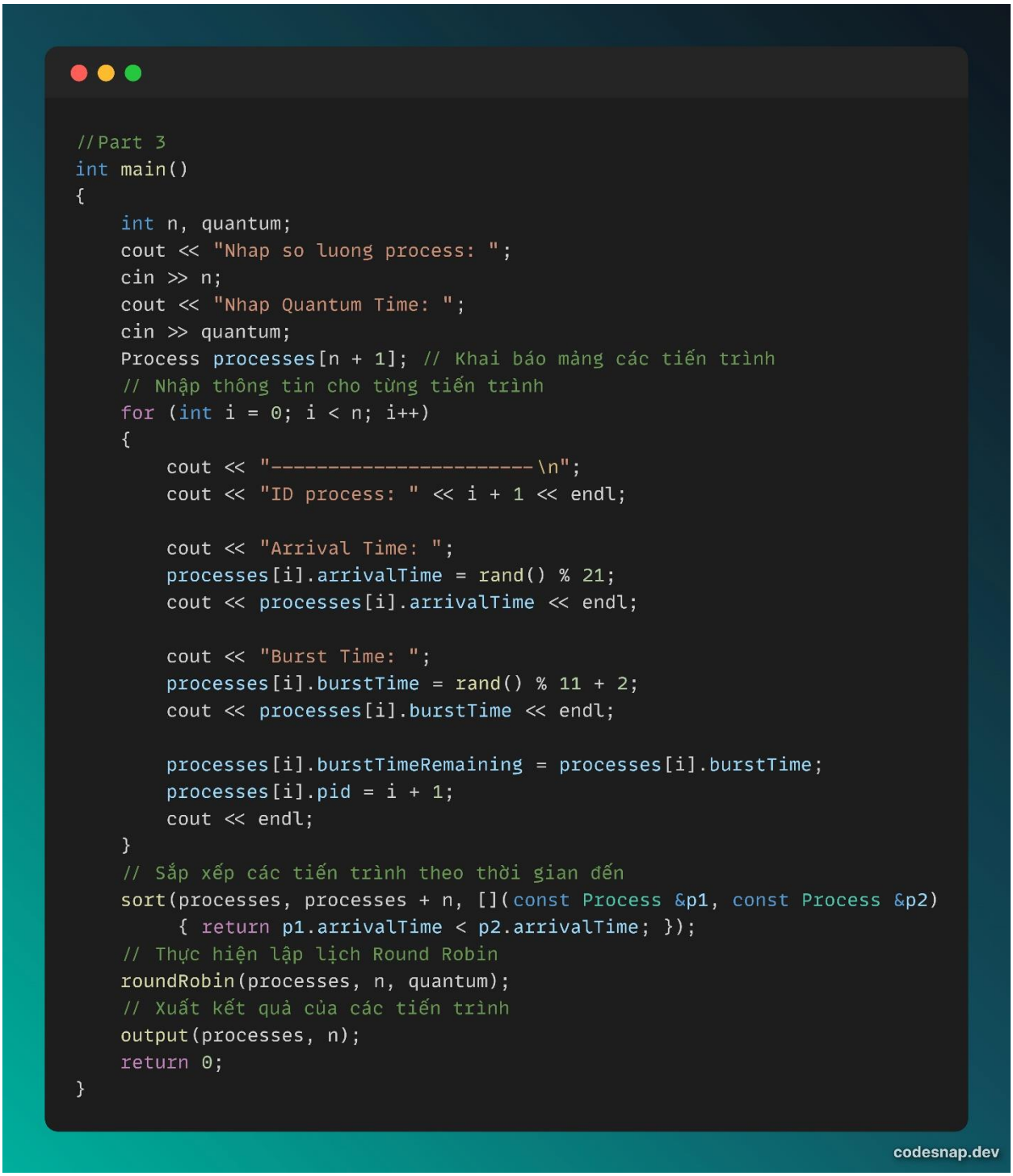
3. Thực hiện code cho giải thuật RR.

rr.cpp

```cpp
#include <bits/stdc++.h> // Thư viện thường được sử dụng trong Competitive Programming

using namespace std;

// Khai báo cấu trúc Process để mô tả thông tin của mỗi tiến trình
struct Process
{
    int pid;                // ID của tiến trình
    int arrivalTime;        // Thời gian đến
    int burstTime;          // Thời gian thực hiện
    int burstTimeRemaining; // Thời gian thực hiện còn lại sau mỗi lần thực hiện
    int completionTime;     // Thời gian hoàn thành
    int turnaroundTime;     // Thời gian quay vòng
    int waitingTime;        // Thời gian chờ
    bool isComplete;        // Đánh dấu xem tiến trình đã hoàn thành hay chưa
    bool inQueue;           // Đánh dấu xem tiến trình đã được thêm vào hàng đợi hay chưa
};

// Hàm kiểm tra và thêm các tiến trình mới vào hàng đợi
void checkForNewArrivals(Process processes[], const int n, const int currentTime, queue<int> &readyQueue)
{
    for (int i = 0; i < n; i++)
    {
        Process p = processes[i];
        // Kiểm tra nếu có tiến trình mới đến
        // Nếu có, thêm vào hàng đợi
        if (p.arrivalTime <= currentTime && !p.inQueue && !p.isComplete)
        {
            processes[i].inQueue = true;
            readyQueue.push(i);
        }
    }
}

// Hàm cập nhật hàng đợi sau mỗi lần thực hiện tiến trình
void updateQueue(Process processes[], const int n, const int quantum, queue<int> &readyQueue, int &currentTime, int &programsExecuted)
{
    int i = readyQueue.front();
    readyQueue.pop();
    // Nếu tiến trình sắp hoàn thành thực hiện
    // Cập nhật thông tin và kiểm tra tiến trình mới đến
    if (processes[i].burstTimeRemaining <= quantum)
    {
        processes[i].isComplete = true;
        currentTime += processes[i].burstTimeRemaining;
        processes[i].completionTime = currentTime;
        processes[i].waitingTime = processes[i].completionTime - processes[i].arrivalTime - processes[i].burstTime;
        processes[i].turnaroundTime = processes[i].waitingTime + processes[i].burstTime;
        if (processes[i].waitingTime < 0)
            processes[i].waitingTime = 0;
        processes[i].burstTimeRemaining = 0;
        // Nếu vẫn còn tiến trình chưa được thêm vào hàng đợi
        // Kiểm tra và thêm tiến trình mới đến vào hàng đợi
        if (programsExecuted != n)
        {
            checkForNewArrivals(processes, n, currentTime, readyQueue);
        }
    }
    // Nếu tiến trình chưa hoàn thành nhưng đã sử dụng hết thời gian quantum
    // Cập nhật thông tin và kiểm tra tiến trình mới đến
    else
    {
        processes[i].burstTimeRemaining -= quantum;
        currentTime += quantum;
        if (programsExecuted != n)
        {
            checkForNewArrivals(processes, n, currentTime, readyQueue);
        }
        // Thêm tiến trình chưa hoàn thành vào hàng đợi để thực hiện tiếp
        readyQueue.push(i);
    }
}
// End part 1
```

codesnap.dev

```cpp
//Part 2
// Hàm xuất kết quả của các tiến trình đã được thực hiện
void output(Process processes[], const int n)
{
    double avgWaitingTime = 0;
    double avgTurntaroundTime = 0;
    // Sắp xếp các tiến trình theo ID
    sort(processes, processes + n, [](const Process &p1, const Process &p2)
         { return p1.pid < p2.pid; });
    // Xuất kết quả của từng tiến trình
    for (int i = 0; i < n; i++)
    {
        // cout << "Process " << processes[i].pid << ": Waiting Time: " << processes[i].waitingTime << " Turnaround Time: "
        //  << processes[i].turnaroundTime << endl;

        cout << left << setw(10) << "PName"
             << left << setw(10) << "Arrtime"
             << left << setw(10) << "Bursttime"
             << left << setw(10) << "Waitting Time"
             << left << setw(10) << "TAT" << endl;

        cout << setw(10) << left << processes[i].pid
             << setw(10) << left << processes[i].arrivalTime
             << setw(10) << left << processes[i].burstTime
             << setw(10) << left << processes[i].waitingTime
             << setw(10) << left << processes[i].turnaroundTime
             << endl;

        avgWaitingTime += processes[i].waitingTime;
        avgTurntaroundTime += processes[i].turnaroundTime;
    }
    cout << "-----------------------\n";

    // Xuất thời gian chờ trung bình và thời gian quay vòng trung bình
    cout << fixed << setprecision(2) << "Thoi gian doi trung binh: " << avgWaitingTime / n << endl;
    cout << fixed << setprecision(2) << "Thoi gian hoan thanh trung binh: " << avgTurntaroundTime / n << endl;
}

// Hàm lập lịch Round Robin cho các tiến trình
void roundRobin(Process processes[], int n, int quantum)
{
    queue<int> readyQueue;
    readyQueue.push(0); // Thêm tiến trình đầu tiên vào hàng đợi
    processes[0].inQueue = true;
    int currentTime = 0;      // Thời gian hiện tại sau mỗi lần thực hiện
    int programsExecuted = 0; // Số lượng tiến trình đã thực hiện
    // Lặp cho đến khi hàng đợi trống
    while (!readyQueue.empty())
    {
        // Cập nhật hàng đợi và thời gian hiện tại sau mỗi lần thực hiện
        updateQueue(processes, n, quantum, readyQueue, currentTime, programsExecuted);
    }
}
// End Part 2
```



```cpp
//Part 3
int main()
{
    int n, quantum;
    cout << "Nhap so luong process: ";
    cin >> n;
    cout << "Nhap Quantum Time: ";
    cin >> quantum;
    Process processes[n + 1]; // Khai báo mảng các tiến trình
    // Nhập thông tin cho từng tiến trình
    for (int i = 0; i < n; i++)
    {
        cout << "-----------------------\n";
        cout << "ID process: " << i + 1 << endl;

        cout << "Arrival Time: ";
        processes[i].arrivalTime = rand() % 21;
        cout << processes[i].arrivalTime << endl;

        cout << "Burst Time: ";
        processes[i].burstTime = rand() % 11 + 2;
        cout << processes[i].burstTime << endl;

        processes[i].burstTimeRemaining = processes[i].burstTime;
        processes[i].pid = i + 1;
        cout << endl;
    }
    // Sắp xếp các tiến trình theo thời gian đến
    sort(processes, processes + n, [](const Process &p1, const Process &p2)
         { return p1.arrivalTime < p2.arrivalTime; });
    // Thực hiện lập lịch Round Robin
    roundRobin(processes, n, quantum);
    // Xuất kết quả của các tiến trình
    output(processes, n);
    return 0;
}
```

codesnap.dev

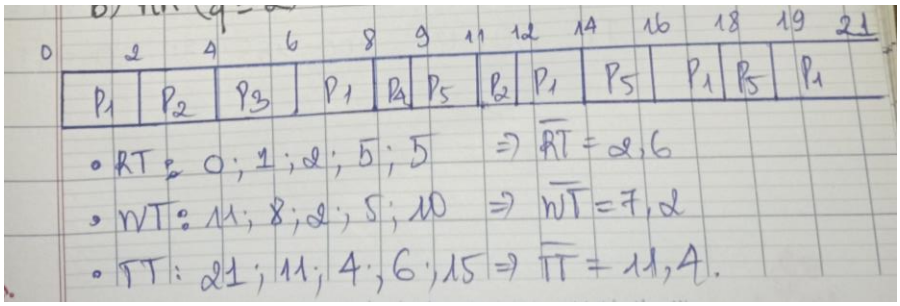
Giải thích code: *(xem ảnh)*

4. Trình bày tính đúng đắn của code bằng cách chạy ít nhất 03 test case, mỗi test case 5 tiến trình, so sánh kết quả chạy tay và chạy code

Test Case 1:

```
Nhap so luong process: 5
Nhap Quantum Time: 2
----------------------
ID process: 1
Arrival Time: 0
Burst Time: 10

----------------------
ID process: 2
Arrival Time: 1
Burst Time: 3

----------------------
ID process: 3
Arrival Time: 2
Burst Time: 2

----------------------
ID process: 4
Arrival Time: 3
Burst Time: 1

----------------------
ID process: 5
Arrival Time: 4
Burst Time: 5

PName     Arrtime   Bursttime Waitting TimeTAT
1         0         10        11        21
PName     Arrtime   Bursttime Waitting TimeTAT
2         1         3         8         11
PName     Arrtime   Bursttime Waitting TimeTAT
3         2         2         2         4
PName     Arrtime   Bursttime Waitting TimeTAT
4         3         1         5         6
PName     Arrtime   Bursttime Waitting TimeTAT
5         4         5         10        15
----------------------
Thoi gian doi trung binh: 7.20
Thoi gian hoan thanh trung binh: 11.40
```

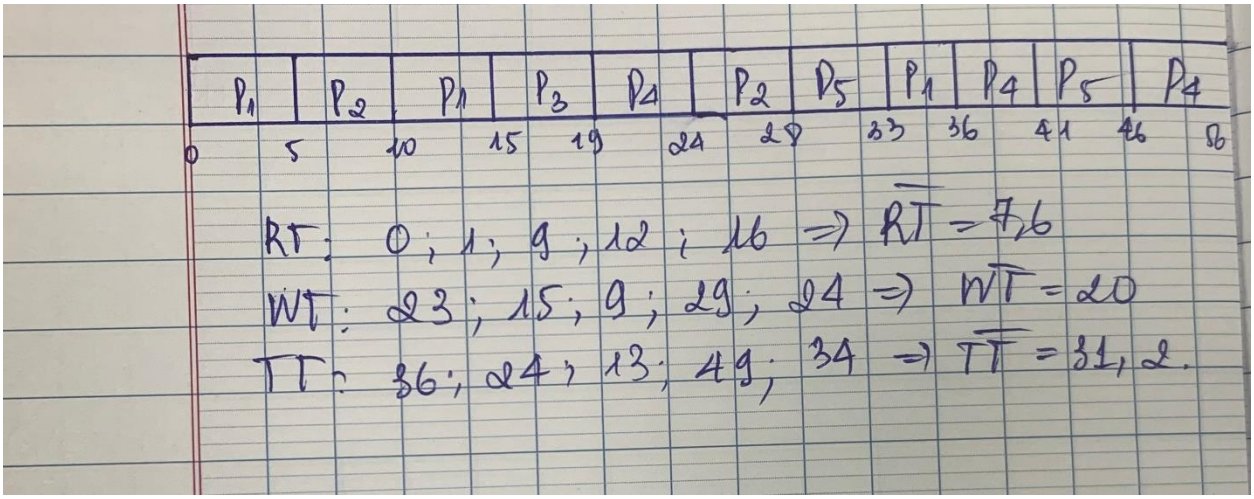
Giải tay:



Test Case 2:

```
Nhap so luong process: 5
Nhap Quantum Time: 5
----------------------
ID process: 1
Arrival Time: 0
Burst Time: 13

----------------------
ID process: 2
Arrival Time: 4
Burst Time: 9

----------------------
ID process: 3
Arrival Time: 6
Burst Time: 4

----------------------
ID process: 4
Arrival Time: 7
Burst Time: 20

----------------------
ID process: 5
Arrival Time: 12
Burst Time: 10

PName     Arrtime   Bursttime Waitting TimeTAT
1         0         13        23        36
PName     Arrtime   Bursttime Waitting TimeTAT
2         4         9         15        24
PName     Arrtime   Bursttime Waitting TimeTAT
3         6         4         9         13
PName     Arrtime   Bursttime Waitting TimeTAT
4         7         20        29        49
PName     Arrtime   Bursttime Waitting TimeTAT
5         12        10        24        34
----------------------
Thoi gian doi trung binh: 20.00
Thoi gian hoan thanh trung binh: 31.20
```

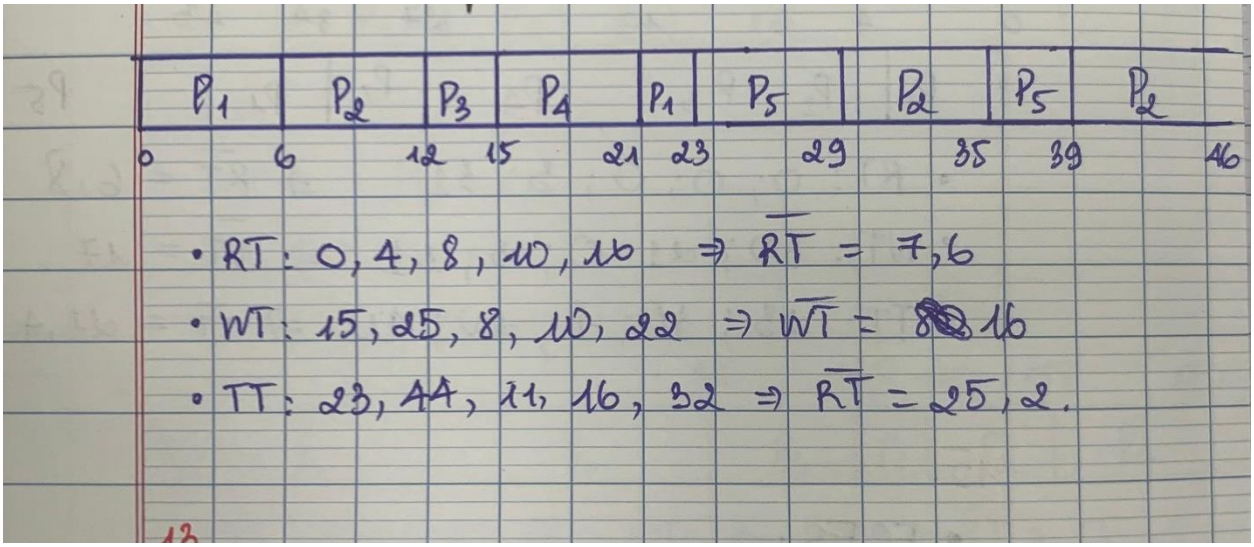
Giải tay:


P1 P2 P1 P3 P4 P2 P5 P1 P4 P5 P4
0 5 10 15 19 24 29 33 36 41 46 56
RT: 0; 1; 9; 12; 16 ⇒ $\overline{RT}$ = 7,6
WT: 23; 15; 9; 29; 24 ⇒ $\overline{WT}$ = 20
TT: 36; 24; 13; 49; 34 ⇒ $\overline{TT}$ = 31,2.

Test Case 3:

```
Nhap so luong process: 5
Nhap Quantum Time: 6
----------------------
ID process: 1
Arrival Time: 0
Burst Time: 8

----------------------
ID process: 2
Arrival Time: 2
Burst Time: 19

----------------------
ID process: 3
Arrival Time: 4
Burst Time: 3

----------------------
ID process: 4
Arrival Time: 5
Burst Time: 6

----------------------
ID process: 5
Arrival Time: 7
Burst Time: 10

PName     Arrtime   Bursttime Waitting TimeTAT
1         0         8         15        23
PName     Arrtime   Bursttime Waitting TimeTAT
2         2         19        25        44
PName     Arrtime   Bursttime Waitting TimeTAT
3         4         3         8         11
PName     Arrtime   Bursttime Waitting TimeTAT
4         5         6         10        16
PName     Arrtime   Bursttime Waitting TimeTAT
5         7         10        22        32
----------------------
Thoi gian doi trung binh: 16.00
Thoi gian hoan thanh trung binh: 25.20
```

Giải tay:



=> Nhận xét: số liệu từ giải tay và code là giống nhau, không có sai số *(số liệu khi chạy trên code được làm tròn)*