

Tên nhóm: Embe

Thành viên:

Lê Thị Loan-22520781

Trần Danh Vinh-22521681

Từ Thị Hồng Phúc-22521146

Lê Dương Hoàng Kim Phụng-22521163

Lớp:IT007.015.2

HỆ ĐIỀU HÀNH
BÁO CÁO LAB 4

CHECKLIST

3.5. BÀI TẬP THỰC HÀNH

	BT 1	BT 2
Vẽ lưu đồ giải thuật	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Chạy tay lưu đồ giải thuật	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Hiện thực code	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Chạy code và kiểm chứng	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

3.6. BÀI TẬP ÔN TẬP

	BT 1
Vẽ lưu đồ giải thuật	<input checked="" type="checkbox"/>
Chạy tay lưu đồ giải thuật	<input checked="" type="checkbox"/>
Hiện thực code	<input checked="" type="checkbox"/>
Chạy code và kiểm chứng	<input checked="" type="checkbox"/>

Tư chấm điểm: 12

Báo cáo thực hành môn Hệ điều hành - Giảng viên: Trần Hoàng Lộc.

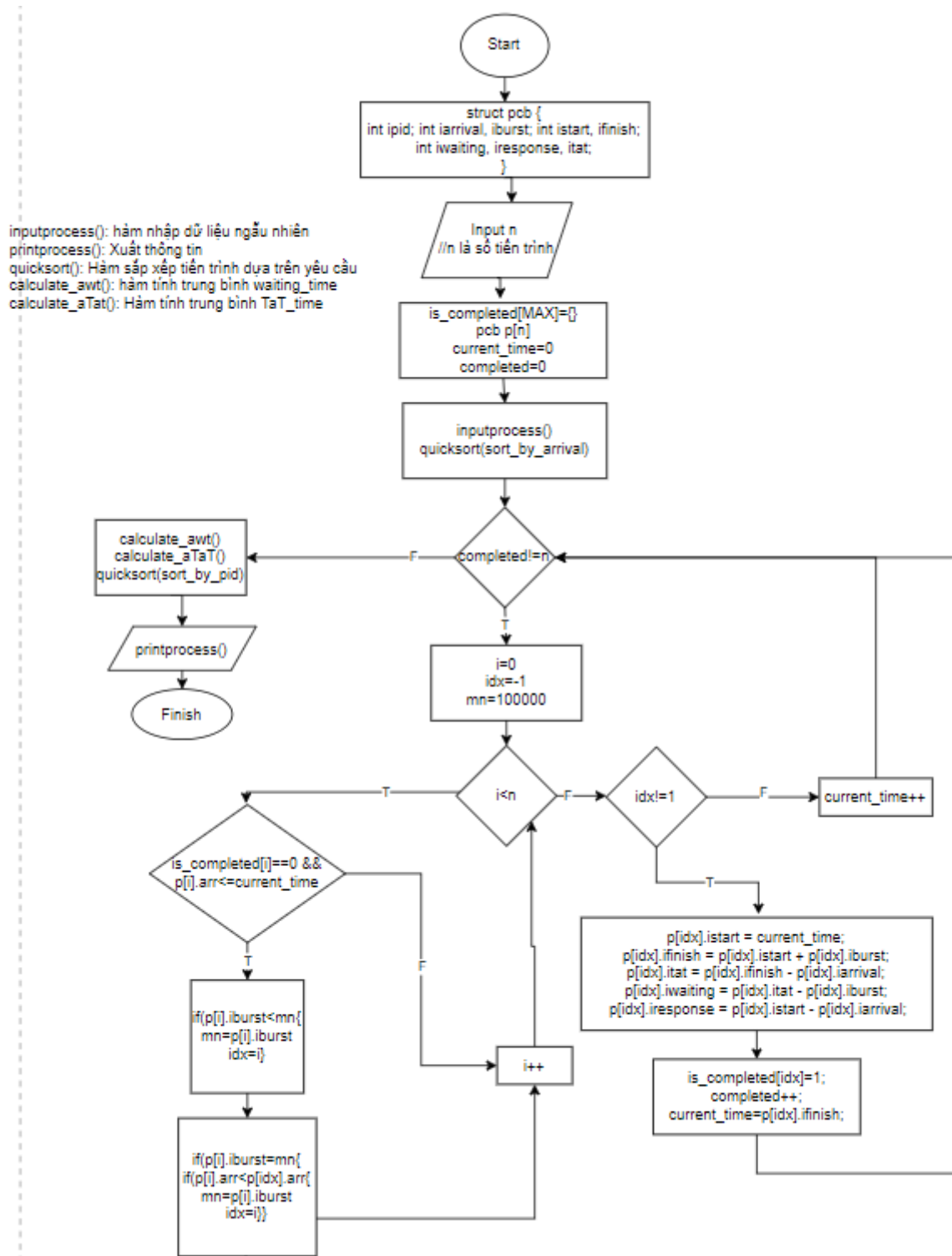
**Lưu ý: Xuất báo cáo theo định dạng PDF, đặt tên theo cú pháp:*

<Tên nhóm>_LAB3.pdf

2.5. BÀI TẬP THỰC HÀNH

1. Giải thuật Shortest-Job-First

Lưu đồ giải thuật



Chạy tay lưu đồ giải thuật

Test case:

Process	Arrival_time	Burst_time
1	0	10
2	1	3
3	2	2
4	3	1
5	4	5

Giản đồ Gantt:

P1	P4	P3	P2	P5	
0	10	11	13	16	21

- Nhập $n=5$
- Vì các tiến trình đã đúng thứ tự theo thời gian arrival_time nên mảng các tiến trình sau khi sắp xếp vẫn được giữ nguyên
- $\text{Current_time}=0$; $\text{completed}=0$;
- Mảng $\text{is_completed}[4] = 0$ là mảng thể hiện trạng thái của tiến trình. Nếu tiến trình chưa hoàn thành thì vị trí của tiến trình đó trong mảng sẽ là 0, ngược lại thì là 1. Ban đầu, chưa tiến trình nào hoàn thành nên mảng này sẽ có giá trị là 0 ở tất cả index
- **(LẦN LẶP 1)** Số tiến trình đã hoàn thành $\text{completed}=0$ khác với $n = 5$ nên bắt đầu thực hiện vòng lặp tìm tiến trình có burst_time nhỏ nhất trong danh sách hàng chờ
 - $\text{idx}=-1$: vị trí của tiến trình được chạy
 - $\text{mn}=100000$: giá trị burst_time của tiến trình $p[\text{idx}]$
 - Tìm được $p[0]$ (**P1**) thỏa vì
$$P[0].\text{arrival} = 0 = \text{current_time}$$
$$\text{is_completed}[0]=0 \rightarrow \text{tiến trình chưa hoàn thành}$$
$$p[0].\text{burst}=10 < \text{mn}$$
 - $\text{idx}=0$; $\text{mn}=10$;
 - Vì tìm được tiến trình thỏa nên thông tin của tiến trình được tính toán:
$$\text{Start}=0; \text{Finish}=10; \text{Tat}=10; \text{Waiting}=0; \text{Response}=0$$
$$\text{completed} = 1, \text{is_finish}[0]=1; \text{current_time}=10$$

- **(LẦN LẶP 2)** Số tiến trình đã hoàn thành $completed = 1$ khác với $n = 5$ nên bắt đầu thực hiện vòng lặp tìm tiến trình có $burst_time$ nhỏ nhất trong danh sách hàng chờ
 - $idx = -1$: vị trí của tiến trình được chạy
 - $mn = 100000$: giá trị $burst_time$ của tiến trình $p[idx]$
 - Tìm được $p[3]$ (**P4**) thỏa vì
$$p[3].iarrival = 3 < current_time = 10$$
$$is_completed[3] = 0 \rightarrow \text{tiến trình chưa hoàn thành}$$
$$p[3].iburst = 1 < mn = 2$$
 - $idx = 3; mn = 1$;
 - Vì tìm được tiến trình thỏa nên thông tin của tiến trình được tính toán:
$$Start = 10; Finish = 11; Tat = 8; Waiting = 7; Response = 7$$
$$completed = 2, is_finish[3] = 1; current_time = 11$$
- **(LẦN LẶP 3)** Số tiến trình đã hoàn thành $completed = 2$ khác với $n = 5$ nên bắt đầu thực hiện vòng lặp tìm tiến trình có $burst_time$ nhỏ nhất trong danh sách hàng chờ
 - $idx = -1$: vị trí của tiến trình được chạy
 - $mn = 100000$: giá trị $burst_time$ của tiến trình $p[idx]$
 - Tìm được $p[2]$ (**P3**) thỏa vì
$$p[2].iarrival = 3 < current_time = 11$$
$$is_completed[2] = 0 \rightarrow \text{tiến trình chưa hoàn thành}$$
$$p[2].iburst = 2 < mn = 3$$
 - $idx = 2; mn = 2$;
 - Vì tìm được tiến trình thỏa nên thông tin của tiến trình được tính toán:
$$Start = 11; Finish = 13; Tat = 11; Waiting = 9; Response = 9$$
$$completed = 3, is_finish[2] = 1; current_time = 13$$
- **(LẦN LẶP 4)** Số tiến trình đã hoàn thành $completed = 3$ khác với $n = 5$ nên bắt đầu thực hiện vòng lặp tìm tiến trình có $burst_time$ nhỏ nhất trong danh sách hàng chờ
 - $idx = -1$: vị trí của tiến trình được chạy
 - $mn = 100000$: giá trị $burst_time$ của tiến trình $p[idx]$
 - Tìm được $p[1]$ (**P2**) thỏa vì
$$p[1].iarrival = 1 < current_time = 13$$
$$is_completed[1] = 0 \rightarrow \text{tiến trình chưa hoàn thành}$$
$$p[1].iburst = 3 < mn = 100000$$
 - $idx = 1; mn = 3$;
 - Vì tìm được tiến trình thỏa nên thông tin của tiến trình được tính toán:
$$Start = 13; Finish = 16; Tat = 15; Waiting = 12; Response = 12$$

completed = 4, is_finish[1]=1;current_time=16

- **(LẦN LẶP 5)** Số tiến trình đã hoàn thành completed =4 khác với n = 5 nên bắt đầu thực hiện vòng lặp tìm tiến trình có burst_time nhỏ nhất trong danh sách hàng chờ

- idx=-1: vị trí của tiến trình được chạy
- mn=100000: giá trị burst_time của tiến trình p[idx]
- Tìm được p[4] **(P5)** thỏa vì
p[4].iarrival = 4 < current_time = 16
is_completed[4]=0 → tiến trình chưa hoàn thành
p[4].iburst=5 < mn =100000

- idx=4; mn=5;
- Vì tìm được tiến trình thỏa nên thông tin của tiến trình được tính toán:
Start=16; Finish=21; Tat=17; Waiting=12; Response=12
completed = 5, is_finish[4]=1;current_time=21

- **(LẦN LẶP 6)** Số tiến trình đã hoàn thành completed =5 bằng với n = 5 nên kết thúc vòng lặp while vì tất cả tiến trình đều đã hoàn thành

- Tính toán các giá trị trung bình:

Avg_TaT=(10+8+11+15+17)/5=12.2

Abg_WT=(0+7+9+12+12)/5=8

- Sắp xếp lại mảng các tiến trình theo process ID và xuất ra màn hình

→ **Kết quả đúng so với chạy tay giải thuật SJF**

Hiện thực code

Khai báo thư viện và một kiểu dữ liệu mới chứa thông tin của tiến trình gọi là pcb

```
#include <stdio.h>
#include <stdlib.h>
#define sort_by_arrival 0
#define sort_by_pid 1
#define sort_by_burst 2
#define sort_by_start 3

typedef struct {
    int ipid;
    int iarrival, iburst;
    int istart, ifinish;
    int iwaiting, ireponse, itat;
} pcb;
```

Báo cáo thực hành môn Hệ điều hành - Giảng viên: Trần Hoàng Lộc.

Hàm nhập arrival_time (0-20) và burst_time (2-12) ngẫu nhiên

```
void inputprocess(int n, pcb p[]) {
    srand(time(0));
    for (int i = 0; i < n; i++) {
        p[i].ipid=i+1;
        p[i].iarrival=rand() % 21;
        p[i].iburst=(rand() % 11) + 2;
    }
}
```

Hàm xuất các thông tin của tiến trình

```
void printprocess(int n, pcb p[]) {
    for (int i = 0; i < n; i++) {
        printf("P%d \t ", p[i].ipid);
        printf("Arrival: %d \t", p[i].iarrival);
        printf("Burst:  %d \t", p[i].iburst);
        printf("Start:  %d \t", p[i].istart);
        printf("Finish: %d \t", p[i].ifinish);
        printf("Waiting: %d \t", p[i].iwaiting);
        printf("Response: %d \t", p[i].iresponse);
        printf("Tunaround: %d \t", p[i].itat);
        printf("\n");
    }
}
```

Hàm xuất giản đồ Gantt

```
void exportganttchart(int n, pcb p[]) {
    printf("Gantt Chart:\n");
    if(p[0].istart!=0){
        printf("|0| \t ");
    }
    for(int i=0;i<n;i++){
        if(i==0)
            printf("|%d| \t PROCESS %d \t |%d|",p[i].istart,p[i].ipid,p[i].ifinish);
        else if (p[i].istart==p[i-1].ifinish)
            printf(" \t PROCESS %d \t |%d|",p[i].ipid,p[i].ifinish);
        else
            printf(" \t |%d| \t PROCESS %d \t |%d|",p[i].istart,p[i].ipid,p[i].ifinish);
    }
    printf("\n");
}
```

Các hàm để sắp xếp tiến trình theo yêu cầu

```
int swapprocess(pcb *p, pcb *q) {
    pcb temp = *p;
    *p = *q;
    *q = temp;
}

int partition(pcb p[], int low, int high, int icriteria) {
    pcb pivot = p[high];
    int i = (low - 1);
    for (int j = low; j <= high - 1; j++) {
        if (icriteria == sort_by_arrival && p[j].iarrival < pivot.iarrival) {
            i++;
            swapprocess(&p[i], &p[j]);
        } else if (icriteria == sort_by_pid && p[j].ipid < pivot.ipid) {
            i++;
            swapprocess(&p[i], &p[j]);
        } else if (icriteria == sort_by_burst && p[j].iburst < pivot.iburst) {
            i++;
            swapprocess(&p[i], &p[j]);
        } else if (icriteria == sort_by_start && p[j].istart < pivot.istart) {
            i++;
            swapprocess(&p[i], &p[j]);
        }
    }
    swapprocess(&p[i + 1], &p[high]);
    return (i + 1);
}

void quicksort(pcb p[], int low, int high, int icriteria) {
    if (low < high) {
        int pi = partition(p, low, high, icriteria);
        quicksort(p, low, pi - 1, icriteria);
        quicksort(p, pi + 1, high, icriteria);
    }
}
```


Báo cáo thực hành môn Hệ điều hành - Giảng viên: Trần Hoàng Lộc.

Hàm tính toán Turnaround_Time trung bình và Waiting_time trung bình và Response_Time trung bình

```
void calculate_awt(int n, pcb p[]) {
    int total_waiting_time = 0;
    for (int i = 0; i < n; i++) {
        total_waiting_time += p[i].iwaiting;
    }
    float avg_waiting_time = (float) total_waiting_time / n;
    printf("Average Waiting Time: %.2f\n", avg_waiting_time);
}

void calculate_atat(int n, pcb p[]) {
    int total_turnaround_time = 0;
    for (int i = 0; i < n; i++) {
        total_turnaround_time += p[i].itat;
    }
    float avg_turnaround_time = (float) total_turnaround_time / n;
    printf("Average Turnaround Time: %.2f\n", avg_turnaround_time);
}

void calculate_art(int n, pcb p[]) {
    int total_response_time = 0;
    for (int i = 0; i < n; i++) {
        total_response_time += p[i].iresponse;
    }
    float avg_response_time = (float) total_response_time / n;
    printf("Average Response Time: %.2f\n", avg_response_time);
}
```

Hàm main

```
int main(){

    int is_completed[100]; //Mảng lưu trạng thái của tiến trình, có giá trị 0 nếu tiến trình chưa hoàn thành
    for(int i=0;i<100;i++){
        is_completed[i]=0;
    }

    int n;
    printf("Nhap so tien trinh: ");
    scanf("%d", &n);
    pcb p[n];

    inputprocess(n,p);
```

```
quicksort(p, 0, n - 1, sort_by_arrival);

int current_time=0; //biến ghi nhận thời điểm hiện tại
int completed=0; //biến ghi nhận số lượng tiến trình đã hoàn thành

while(completed!=n){
    int idx=-1; //Lưu vị trí của tiến trình thỏa yêu cầu của giải thuật SJF
    int mn=100000; //Biến lưu burst_time của tiến trình p[idx] trên

    //Vòng lặp dưới đây tìm ra tiến trình có burst_time nhỏ nhất tại thời điểm hiện tại
    current_time
    for(int i=0;i<n;i++){
        if(p[i].iarrival <= current_time && is_completed[i]==0){
            if(p[i].iburst<mn){
                mn=p[i].iburst;
                idx=i;
            }
            if(p[i].iburst==mn){
                if(p[i].iarrival<p[idx].iarrival){
                    mn=p[i].iburst;
                    idx=i;
                }
            }
        }
    }

    //Nếu tìm được tiến trình thỏa vòng lặp trên thì thực hiện tính toán
    //Nếu không tìm được tiến trình thỏa yêu cầu thì tăng current_time
    //quay trở lại vòng lặp while để kt xem còn tiến trình nào chưa hoàn thành không
    if(idx!=-1){
        p[idx].istart = current_time;
        p[idx].ifinish = p[idx].istart + p[idx].iburst;
        p[idx].itat = p[idx].ifinish - p[idx].iarrival;
        p[idx].iwaiting = p[idx].itat - p[idx].iburst;
        p[idx].iresponse = p[idx].istart - p[idx].iarrival;

        is_completed[idx]=1; //Cập nhật trạng thái của tiến trình thành đã hoàn thành
        completed++; //tăng số lượng tiến trình đã hoàn thành
        current_time=p[idx].ifinish; //Cập nhật lại giá trị của current_time
    }
    else {
        current_time++;
    }
}
```

```
//Tính giá trị trung bình
calculate_atat(n,p);
calculate_awt(n,p);
calculate_art(n,p);

//xuất ra thông tin của tiến trình
quicksort(p, 0, n - 1, sort_by_pid);
printprocess(n,p);

//Xuất ra màn hình giản đồ Gantt
quicksort(p,0,n-1,sort_by_start);
exportganttchart(n,p);
return 0;
}
```

Chạy code và kiểm chứng

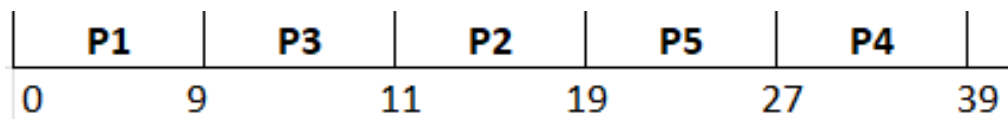
Test case 1:

- Chạy code:

```
lethilooan-22520781@LTL:~/Lab4$ ./SJF
Nhap so tien trinh: 5
Average Turnaround Time: 17.00
Average Waiting Time: 9.20
Average Response Time: 9.20
P1   Arrival: 0   Burst: 9   Start: 0   Finish: 9   Waiting: 0   Response: 0   Turnaround: 9
P2   Arrival: 3   Burst: 8   Start: 11  Finish: 19  Waiting: 8   Response: 8   Turnaround: 16
P3   Arrival: 3   Burst: 2   Start: 9   Finish: 11  Waiting: 6   Response: 6   Turnaround: 8
P4   Arrival: 3   Burst: 12  Start: 27  Finish: 39  Waiting: 24  Response: 24  Turnaround: 36
P5   Arrival: 11  Burst: 8   Start: 19  Finish: 27  Waiting: 8   Response: 8   Turnaround: 16
Gantt Chart:
|0| PROCESS 1 |9| PROCESS 3 |11| PROCESS 2 |19| PROCESS 5 |27| PROCESS 4 |39|
```

- Chạy tay:

Giản đồ Gantt:



Kết quả chạy tay:

	P1	P2	P3	P4	P5	TB
Tunaround_Time	9	16	8	36	16	Avg: 17
Waiting_Time	0	8	6	24	8	Avg: 9.2
Response_Time	0	8	6	24	8	Avg: 9.2

➔ Kết quả chạy tay giống với kết quả chạy code

Test case 2:

- Chạy code:

```

lethiloan-22520781@LTL:~/Lab4$ ./SJF
Nhap so tien trinh: 5
Average Turnaround Time: 11.40
Average Waiting Time: 5.60
Average Response Time: 5.60
P1  Arrival: 20  Burst: 3  Start: 29  Finish: 32  Waiting: 9  Response: 9  Turnaround: 12
P2  Arrival: 18  Burst: 5  Start: 32  Finish: 37  Waiting: 14  Response: 14  Turnaround: 19
P3  Arrival: 11  Burst: 6  Start: 11  Finish: 17  Waiting: 0  Response: 0  Turnaround: 6
P4  Arrival: 1  Burst: 3  Start: 1  Finish: 4  Waiting: 0  Response: 0  Turnaround: 3
P5  Arrival: 12  Burst: 12  Start: 17  Finish: 29  Waiting: 5  Response: 5  Turnaround: 17
Gantt Chart:
|0|  |1|  PROCESS 4  |4|  |11| PROCESS 3  |17| PROCESS 5  |29| PROCESS 1  |32| PROCESS 2  |37|

```

- Chạy tay:

Giản đồ Gantt:

		P4			P3			P5			P1		P2	
0	1	4	11	17	29	32	37							

Kết quả chạy tay:

	P1	P2	P3	P4	P5	TB
Tunaround_Time	12	19	6	3	17	Avg: 11.4
Waiting_Time	9	14	0	0	5	Avg: 5.6
Response_Time	9	14	0	0	5	Avg: 5.6

➔ Kết quả chạy tay giống với kết quả chạy code

Test case 3:

- Chạy code:

```

lethiloan-22520781@LTL:~/Lab4$ ./SJF
Nhap so tien trinh: 5
Average Turnaround Time: 12.80
Average Waiting Time: 4.60
Average Response Time: 4.60
P1  Arrival: 18  Burst: 11  Start: 32  Finish: 43  Waiting: 14  Response: 14  Turnaround: 25
P2  Arrival: 8  Burst: 10  Start: 13  Finish: 23  Waiting: 5  Response: 5  Turnaround: 15
P3  Arrival: 5  Burst: 8  Start: 5  Finish: 13  Waiting: 0  Response: 0  Turnaround: 8
P4  Arrival: 19  Burst: 9  Start: 23  Finish: 32  Waiting: 4  Response: 4  Turnaround: 13
P5  Arrival: 1  Burst: 3  Start: 1  Finish: 4  Waiting: 0  Response: 0  Turnaround: 3
Gantt Chart:
|0|  |1|  PROCESS 5  |4|  |5|  PROCESS 3  |13| PROCESS 2  |23| PROCESS 4  |32| PROCESS 1  |43|

```

- Chạy tay:

Giản đồ Gantt:

		P5			P3		P2		P4		P1	
0	1	4	5	13	23	32	43					

Kết quả chạy tay:

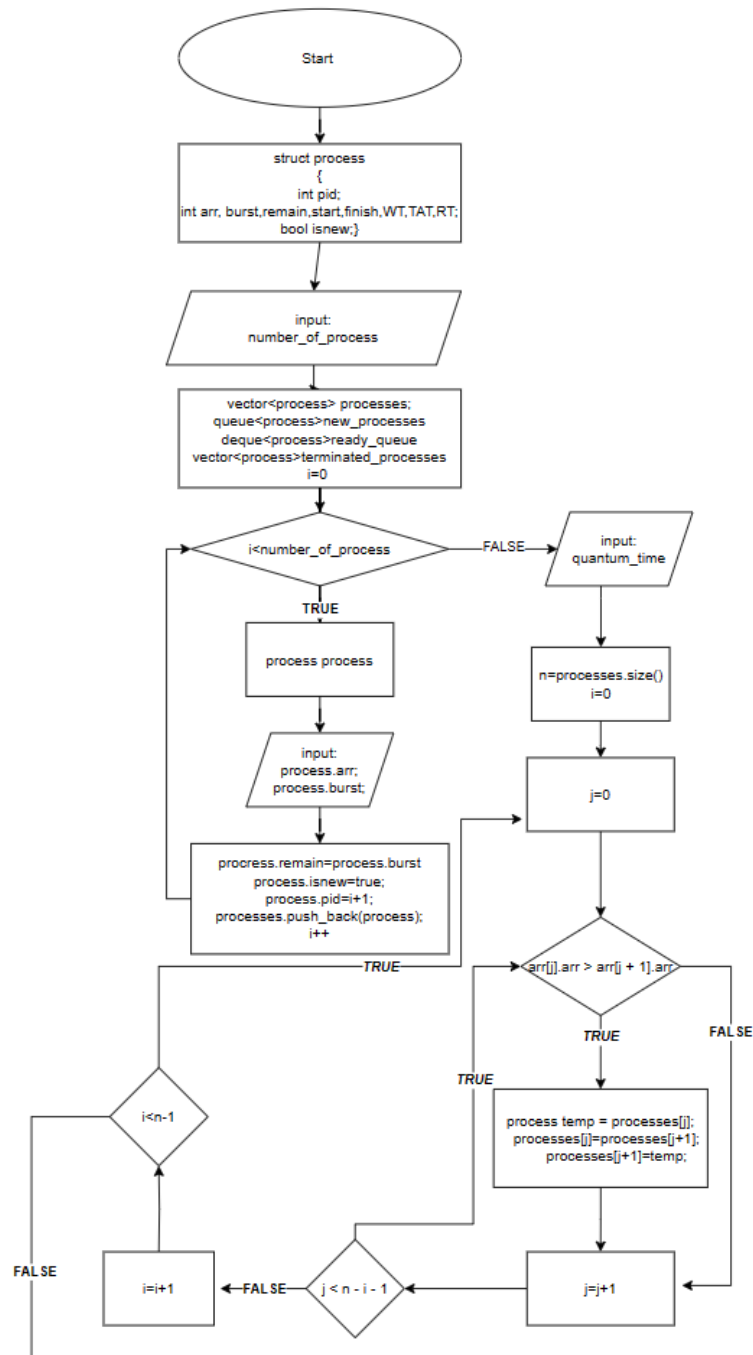
	P1	P2	P3	P4	P5	TB
Tunaround_Time	25	15	8	13	3	Avg: 12.8
Waiting_Time	14	5	0	4	0	Avg: 4.6
Response_Time	14	5	0	4	0	Avg: 4.6

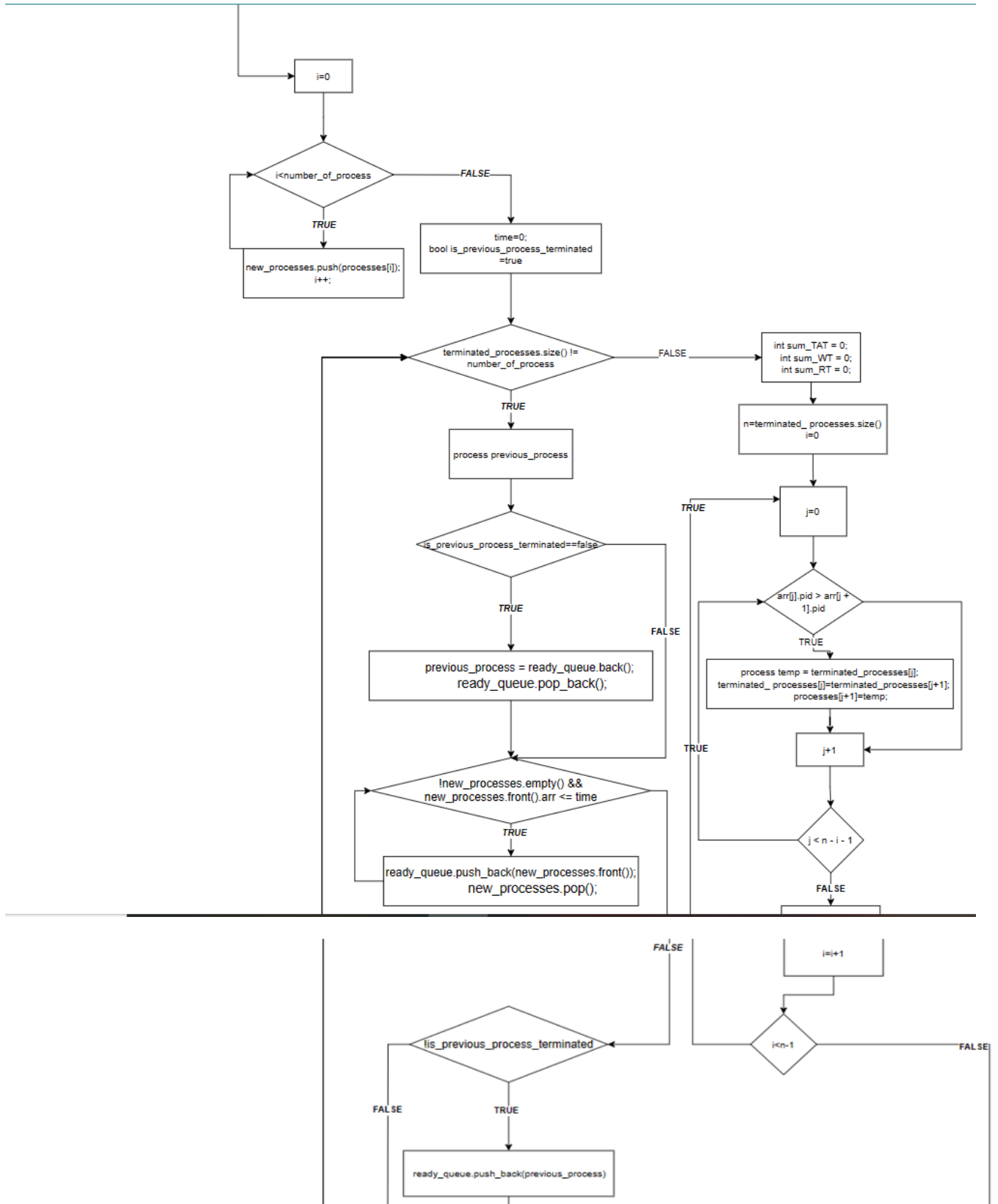
➔ Kết quả chạy tay giống với kết quả chạy code

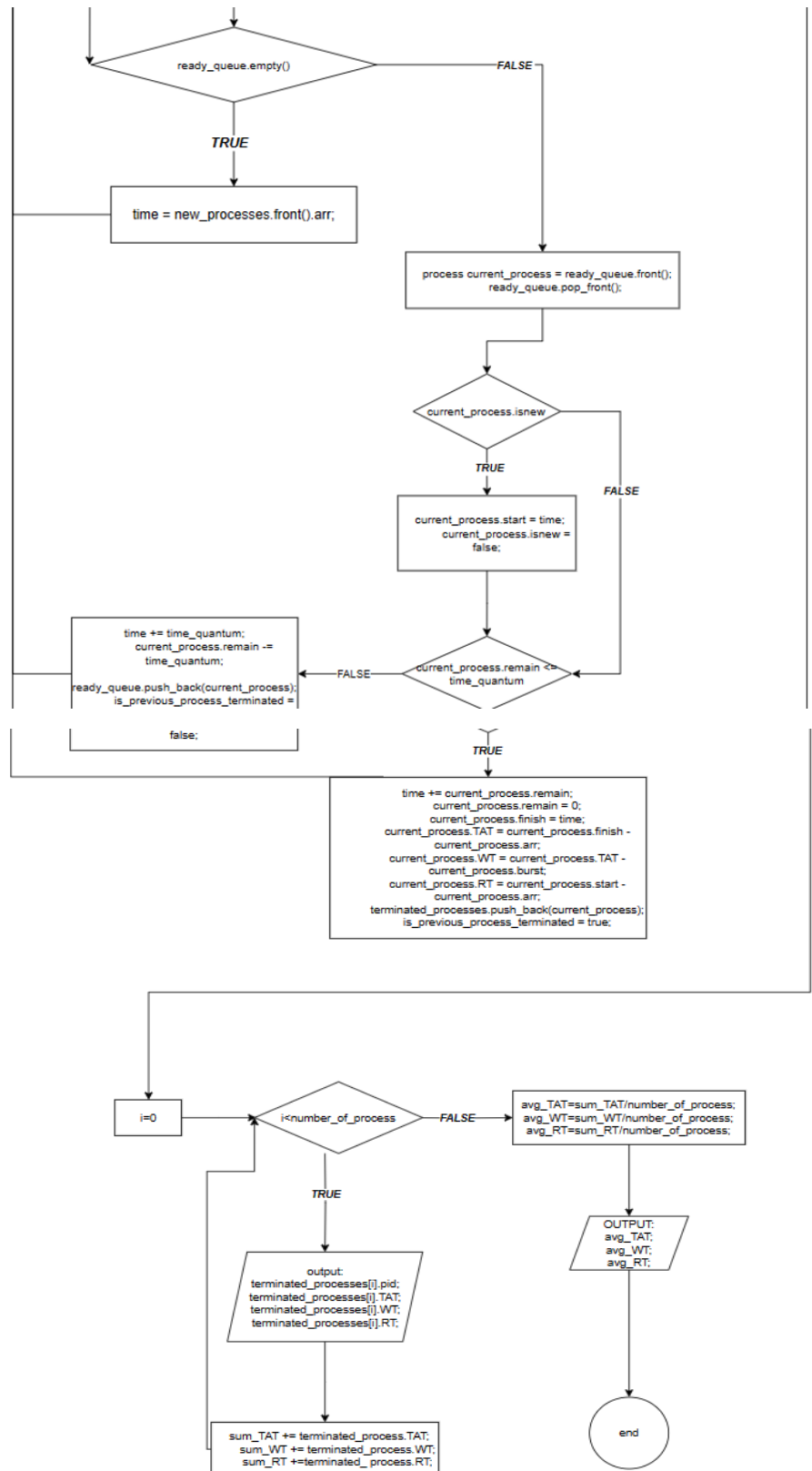
2. Giải thuật Shortest-Remaining-Time-First hoặc Round Robin

Sơ đồ roundrobin:

Link sơ đồ khối roundrobin: <https://s.net.vn/sodoroundrobinnhomembe>







2. Kiểm tra tính đúng đắn của sơ đồ :

pid	Arrival time	Burst time
1	0	12
2	2	7
3	5	8
4	9	3
5	12	6

Quantum time = 5

Sơ đồ gantt chart:

P1	P2	P3	P1	P4	P2	P5	P3	P1	P5	
0	5	10	15	20	23	25	30	33	35	36

Chạy tay sơ đồ giải thuật:

Mảng processes = [p1, p2, p3, p4, p5] sau khi được sắp xếp lại sẽ trở thành

=> mảng processes = [p1, p2, p3, p4, p5]

Đẩy các phần tử ở mảng processes vào mảng new_processes

=> new_processes = [p1, p2, p3, p4, p5]

ready_queue = [];

terminated_processes = [];

+ các mốc thời gian (khi có quantum time= 5)

Tại time = 0

new_processes = [p2, p3, p4, p5]

ready_queue = [p1]

p1.start = 0

p1.remain = 7

time = 5

ready_queue = [p1]

Tại time = 5

```
new_proceses = [p4, p5]
ready_queue = [p2, p3, p1]
p2.start = 5
p2.remain = 2
time = 10
ready_queue = [p3, p1, p2]
```

Tại time=10

```
new_proceses = [p5]
ready_queue = [p3, p1, p4, p2]
p3.start = 10
p3.remain = 3
time = 15
ready_queue = [p1, p4, p2, p3]
```

Tại time = 15

```
new_processes = []
ready_queue = [p1, p4, p2, p5, p3]
p1.remain = 2
time = 20
ready_queue = [p4, p2, p5, p3, p1]
```

Tại time = 20

```
ready_queue = [p4, p2, p5, p3, p1]
p4.remain = 0
time = 23
p4.finish = 23
p4.TAT = 14
p4.WT = 11
p4.RT = 11
ready_queue = [p2, p5, p3, p1]
terminated_processes = [p4]
```

Tại time = 23

ready_queue = [p2, p5, p3, p1]

p2.remain = 0

time = 25

p2.finish = 25

p2.TAT = 23

p2.WT = 16

p2.RT = 3

ready_queue = [p5, p3, p1]

terminated_processes = [p4, p2]

Tại time = 25

ready_queue = [p5, p3, p1]

p5.start = 25

p5.remain = 1

time = 30

ready_queue = [p3, p1, p5]

Tại time = 30

ready_queue = [p3, p1, p5]

p3.remain = 0 time = 33

p3.finish = 33

p3.TAT = 28

p3.WT = 20

p3.RT = 5

ready_queue = [p1, p5]

terminated_processes = [p4, p2, p3]

Tại time = 33

```
ready_queue = [p1, p5]
p1.remain = 0
time = 35
p1.finish = 35
p1.TAT = 35
p1.WT = 23
p1.RT = 0
ready_queue = [p5]
terminated_processes = [p4, p2, p3, p1]
```

Tại time = 35

```
ready_queue = [p5]
p5.remain = 0
time = 36
p5.finish = 36 ;      p5.TAT = 24 ;
p5.WT = 18 ;      p5.RT = 13;
ready_queue = [] ;      terminated_processes = [p4, p2, p3, p1, p5]
```

Khi terminated_processes.size() == number_process => hoàn thành

Source code:

```
1  #include <bits/stdc++.h>
2  #include <iostream>
3  #include <iomanip>
4  #include <climits>
5  using namespace std;
6
7  // tạo struct process lưu các giá trị của 1 process
8  /*
9   pid: pid of process
10  arr: arrival time
11  burst: burst time
12  remain :remain time
13  start: start time
14  finish :finish time
15  WT: waiting time
16  TAT : turn around time
17  RT: reponse time
18  */
19  struct process
20  {
21      int pid;
22      int arr, burst, remain, start, finish, WT, TAT, RT;
23      bool isNew;
24  };
25  // hàm đổi vị trí 2 process cho nhau
26  void swapProcesses(process &a, process &b)
27  {
28      process temp = a;
29      a = b;
30      b = temp;
31  };
```

```
32 // hàm sắp xếp các process theo thời gian đến tăng dần
33 void sortProcessesByArrival(vector<process> &arr)
34 {
35     int n = arr.size();
36
37     for (int i = 0; i < n - 1; i++)
38     {
39         for (int j = 0; j < n - i - 1; j++)
40         {
41             if (arr[j].arr > arr[j + 1].arr)
42             {
43                 // Nếu arr[j] lớn hơn arr[j + 1], đổi chỗ
44                 swapProcesses(arr[j], arr[j + 1]);
45             }
46         }
47     }
48 };
49 void sortProcessesByPid(vector<process> &arr)
50 {
51     int n = arr.size();
52
53     for (int i = 0; i < n - 1; i++)
54     {
55         for (int j = 0; j < n - i - 1; j++)
56         {
57             if (arr[j].pid > arr[j + 1].pid)
58             {
59                 // Nếu arr[j] lớn hơn arr[j + 1], đổi chỗ
60                 swapProcesses(arr[j], arr[j + 1]);
61             }
62         }
63     }
64 };
```

```
65 int main()
66 {
67     int number_of_process;
68     vector<process> processes;
69     cout << "nhap so luong tien trinh: ";
70     cin >> number_of_process;
71
72     srand(time(0));
73     for (int i = 0; i < number_of_process; i++)
74     {
75         process process;
76         process.arr = rand() % 21;
77         process.burst = (rand() % 11) + 2;
78         process.pid = i + 1;
79         process.isnew = true;
80         process.remain = process.burst;
81         processes.push_back(process);
82     };
83
84     sortProcessesByArrival(processes);
85     queue<process> new_processes;
86     for (auto process : processes)
87         new_processes.push(process);
88     cout << "nhap quantum time :";
89     int time_quantum;
90     cin >> time_quantum;
91     int time = 0;
92     deque<process> ready_queue;
93     vector<process> terminated_processes;
94     bool is_previous_process_terminated = true;
95     while (terminated_processes.size() != number_of_process)
96     {
97         process previous_process;
98         if (!is_previous_process_terminated)
99         {
100             previous_process = ready_queue.back();
101             ready_queue.pop_back();
102         }
103         while (!new_processes.empty() && new_processes.front().arr <= time)
104         {
105             ready_queue.push_back(new_processes.front());
106             new_processes.pop();
107         }
108         if (!is_previous_process_terminated)
109             ready_queue.push_back(previous_process);
110
111         if (ready_queue.empty())
112         {
113             time = new_processes.front().arr;
114             continue;
115         }
116         process current_process = ready_queue.front();
117         ready_queue.pop_front();
```

```

116 ready_queue.pop_front();
117 if (current_process.isnew)
118 {
119     current_process.start = time;
120     current_process.isnew = false;
121 }
122 if (current_process.remain <= time_quantum)
123 {
124     time += current_process.remain;
125     current_process.remain = 0;
126     current_process.finish = time;
127     current_process.TAT = current_process.finish - current_process.arr;
128     current_process.WT = current_process.TAT - current_process.burst;
129     current_process.RT = current_process.start - current_process.arr;
130     terminated_processes.push_back(current_process);
131     is_previous_process_terminated = true;
132 }
133 else
134 {
135     time += time_quantum;
136     current_process.remain -= time_quantum;
137     ready_queue.push_back(current_process);
138     is_previous_process_terminated = false;
139 }
140 }
141 int sum_TAT = 0;
142 int sum_WT = 0;
143 int sum_RT = 0;
144 sortProcessesByPid(terminated_processes);
145 cout << "PID\t Arrival time\t Burst time\t Response time\t Turnaround time\t Waiting time " << endl;
146 for (auto process : terminated_processes)
147 {
148     cout << " " << process.pid << "\t" << process.arr << "\t\t\t" << process.burst << "\t\t"
149     << process.RT << "\t\t" << process.TAT << "\t\t" << process.WT << endl;
150     sum_TAT += process.TAT;
151     sum_WT += process.WT;
152     sum_RT += process.RT;
153 }
154 float avg_TAT = (float)sum_TAT / number_of_process, avg_WT = (float)sum_WT / number_of_process, avg_RT = (float)sum_RT / number_of_process;
155 cout << " thời gian hoàn thành trung bình : " << avg_TAT << endl;
156 cout << " thời gian đáp ứng trung bình : " << avg_RT << endl;
157 cout << " thời gian chờ trung bình : " << avg_WT << endl;
158 }

```

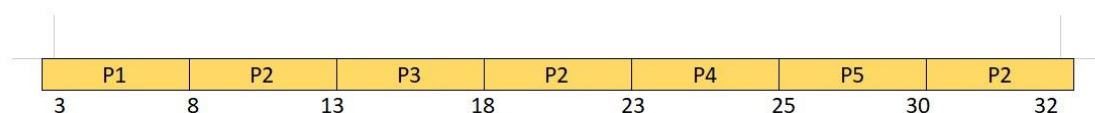
2. Kiểm tra tính đúng đắn của code :

Test case 1:

pid	Arrival time	Burst time
P1	3	5
P2	6	12
P3	12	5
P4	19	2
P5	19	5

Quantum time =5

Sơ đồ gantt chart



PID	Turnaround time	Waiting time	Response time
P1	5	0	0
P2	26	14	2
P3	6	1	1
P4	6	4	4
P5	11	6	6

Kết quả chạy code

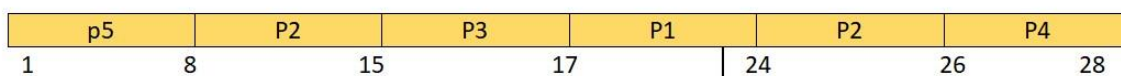
```
tutthongphuc-22521146@DESKTOP-IFQ3611:~/lab4$ ./roundrobin.c
nhap so luong tien trinh: 5
nhap quantum time :5
PID    Arrival time    Burst time    Response time    Turnaround time    Waiting time
1       3                5             0               5                 0
2       6                12            2               26                14
3       12               5             1               6                 1
4       19               2             4               6                 4
5       19               5             6               11                6
thoi gian hoan thanh trung binh : 10.8
thoi gian dap ung trung binh : 2.6
thoi gian cho trung binh : 5
```

Test case 2:

pid	Arrival time	Burst time
P1	13	7
P2	3	9
P3	8	2
P4	17	2
P5	1	7

Quantum time =7

Sơ đồ gantt chart



PID	Turnaround time	Waiting time	Response time
P1	11	4	4
P2	23	14	5
P3	9	7	7
P4	11	9	9
P5	7	0	0

Kết quả chạy code:

```
tutthongphuc-22521146@DESKTOP-IFQ361J:~/lab4$ ./roundrobin.c
nhap so luong tien trinh: 5
nhap quantum time :7
PID      Arrival time    Burst time    Response time    Turnaround time    Waiting time
1         13                7              4                11                 4
2         3                 9              5                23                 14
3         8                 2              7                 9                  7
4        17                2              9                11                 9
5         1                 7              0                 7                  0
thoi gian hoan thanh trung binh : 12.2
thoi gian dap ung trung binh : 5
thoi gian cho trung binh : 6.8
```

Test case 3:

pid	Arrival time	Burst time
P1	16	2
P2	1	11
P3	9	10
P4	13	10
P5	20	4

Quantum time =8

Sơ đồ gantt chart:

P2	P3	P2	P4	P1	P3	P5	P4	
1	9	17	20	28	30	32	36	38

PID	Turnaround time	Waiting time	Response time
P1	14	12	12
P2	19	8	0
P3	23	13	0
P4	25	15	7
P5	16	12	12

Kết quả chạy tay:

```
tutthongphuc-22521146@DESKTOP-IFQ361J:~/lab4$ ./roundrobin.c
nhap so luong tien trinh: 5
nhap quantum time :8
PID    Arrival time    Burst time    Response time    Turnaround time    Waiting time
1       16               2             12              14                12
2       1               11            0               19                8
3       9               10            0               23                13
4       13              10            7               25                15
5       20              4             12              16                12
thoi gian hoan thanh trung binh : 19.4
thoi gian dap ung trung binh : 6.2
thoi gian cho trung binh : 12
```

2.6. BÀI TẬP ÔN TẬP

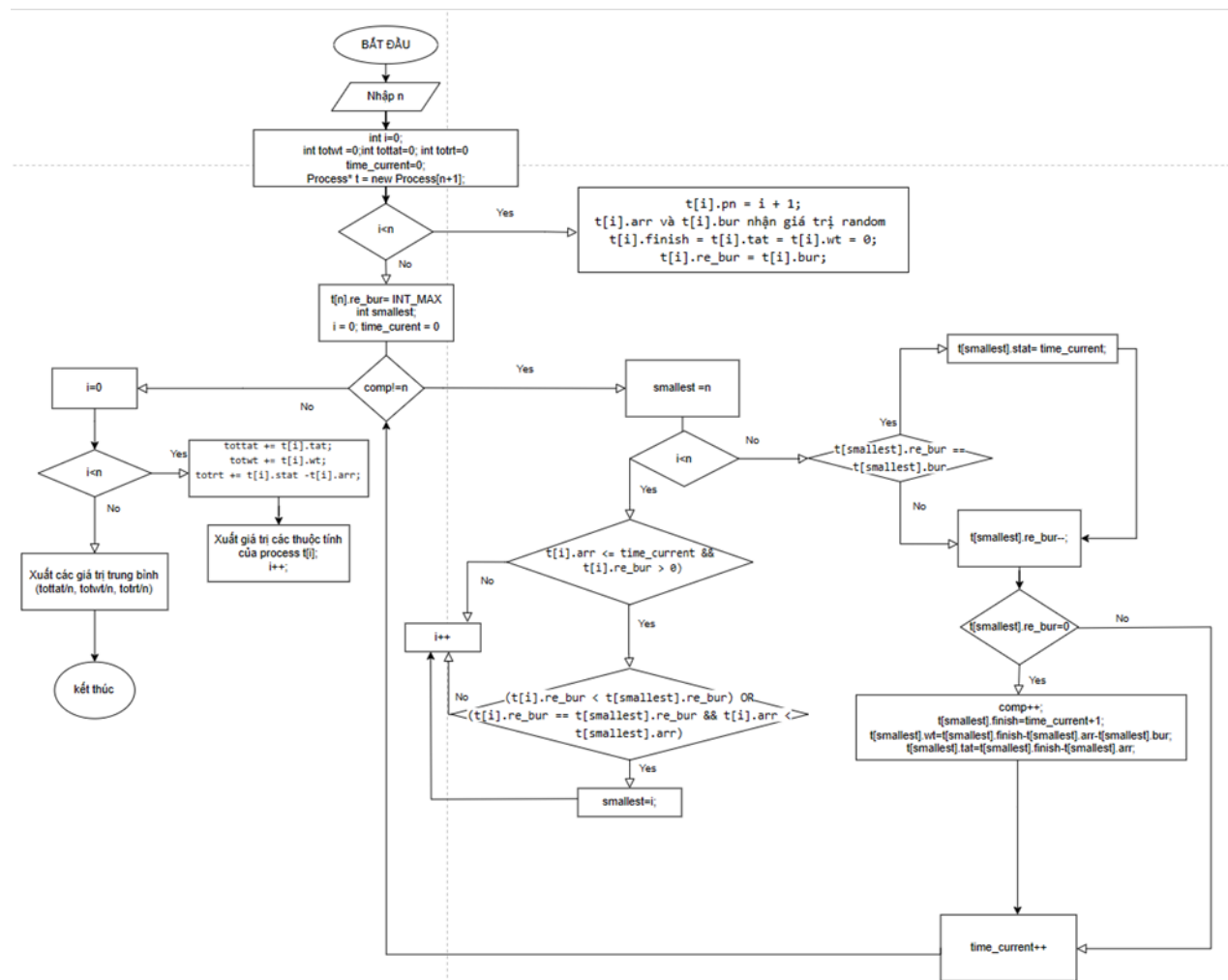
1. Giải thuật Shortest-Remaining-Time-First hoặc Round Robin

SRTF

1. Lưu đồ thuật toán

Ý tưởng:


- Dùng vòng lặp for và mảng lưu các đối tượng process.
- Cho biến `time_current` tăng dần từng đơn vị bằng vòng lặp for, tượng trưng cho thời gian, `time_current` tăng đến khi tất cả các tiến trình đã hoàn tất.
- Trong vòng lặp mỗi lần tăng `time_current` sẽ duyệt qua mảng các tiến trình tìm tiến trình đã đến và có `remaining_burst_time` nhỏ nhất gán là `t[smallest]`:
 - Giảm `remaining_burst_time` của `t[smallest]` 1 đơn vị
 - Tăng `time_current` 1 đơn vị và tiếp tục vòng lặp đến khi tất cả tiến trình hoàn thành (`remaining_burst_time = 0`)



2. Kiểm tra tính đúng đắn của lưu đồ

Process Name	Arrival Time	Burst Time
1	3	4
2	10	7
3	1	9
4	8	4
5	2	6

Theo lý thuyết:

	P3		P5		P1		P5		P4		P2		P3	
0	1	2	3	7	12	16	23	31						

Process	Arrival Time	Burst Time	Turnaround Time	Response Time	Waiting Time
1	3	4	4	0	0
2	10	7	13	6	6
3	1	9	30	0	21
4	8	4	8	4	4
5	2	6	10	0	4

* Theo lưu đồ thuật toán:

$t = [t_0, t_1, t_2, t_3, t_4, t_5]$

time_curent = 0

totwt = 0, tottat = 0, tottrt = 0

$t[5].re_bur = INT_MAX$

tại current_time = 0

smallest = 5

$t[5].stat = 0$

comp = 0

$t[5].re_bur = INT_MAX - 1$

tại current_time = 1

smallest = 2

$t[2].stat = 1$

comp = 0

$t[2].re_bur = 8$

tại current_time = 2

smallest = 4
t[4].stat = 2
comp = 0
t[4].re_bur = 5

tại current_time = 3

smallest = 0
t[0].stat = 3
comp = 0
t[0].re_bur = 3

tại current_time = 6

smallest = 0
comp = 1
t[0].re_bur = 0
t[0].finish = 7
t[0].wt = 0
t[0].tat = 4

tại current_time = 7

smallest = 4
comp = 1
t[4].re_bur = 4

tại current_time = 11

smallest = 4

comp = 2
t[4].re_bur = 0
t[4].finish = 12
t[4].wt = 4
t[4].tat = 10

tại current_time = 12

smallest = 3
t[3].stat = 12
comp = 2
t[3].re_bur = 3

tại current_time = 15

smallest = 3

comp = 3
t[3].re_bur = 0
t[3].finish = 16
t[3].wt = 4
t[3].tat = 8

tại current_time = 16

smallest = 1
t[1].stat = 16
comp = 3
t[1].re_bur = 6

tại current_time = 22

smallest = 1
comp = 4
t[1].re_bur = 0
t[1].finish = 23
t[1].wt = 6
t[1].tat = 13

tại current_time = 23

smallest = 2
comp = 4
t[2].re_bur = 7

tại current_time = 30

smallest = 2
comp = 5
t[2].re_bur = 0
t[3].finish = 31
t[3].wt = 21
t[3].tat = 30

comp = 5 dừng vòng lặp

$$\text{tottat} = 4 + 13 + 30 + 8 + 10 = 65$$

$$\text{totwt} = 0 + 6 + 0 + 4 + 0 = 10$$

$$\text{totrt} = 0 + 6 + 21 + 4 + 4 = 35$$

Average Turnaround Time: 13

Average Waiting: 2

Average Response Time: 7

=> giải thuật kết thúc

3. Code

```
SRTF.cpp X
SRTF.cpp > main()
1  #include <iostream>
2  #include <string.h>
3  #include <iomanip>
4  #include <climits>
5
6  using namespace std;
7  // Tạo struct Process lưu các giá trị của Process
8  /*
9      arr : arrival time
10     bur : burst time
11     stat: start time
12     finish: finish time
13     tat: turnaround time
14     wt: waiting time
15     re_burst: remaining burst time
16  */
17  struct Process
18  {
19      int pn;
20      int arr, bur, stat, finish, tat, wt, re_bur;
21  };
22
23  int main()
24  {
25      // n: số tiến trình
26      // i: biến điều khiển vòng lặp
27      int n, i;
28      // comp: số tiến trình đã hoàn thành
29      int comp = 0;
30      // thời gian hiện tại
31      int time_current;
32
33      struct Process *t;
34      int totwt = 0, tottat = 0, totrt = 0;
35      cout << "Enter the number of processes: ";
36      cin >> n;
37
38      // tạo mảng n+1 phần tử
39      t = new Process[n + 1];
40
41      srand(time(0));
42      for (int i = 0; i < n; i++)
43      {
44          t[i].pn = i + 1;
45          t[i].arr = rand() % 21;
46          t[i].bur = (rand() % 11) + 2;
47          t[i].finish = t[i].tat = t[i].wt = 0;
48          t[i].re_bur = t[i].bur;
49      } // gán các giá trị cho thuộc tính Process, arrival time, burst time là giá trị random
50
```

Báo cáo thực hành môn Hệ điều hành - Giảng viên: Trần Hoàng Lộc.

```
51 t[n].re_bur = INT_MAX;
52 // smallest: index của process được đưa vào thực thi, có giá trị re_bur nhỏ nhất
53 int smallest;
54 for (time_current = 0; comp != n; time_current++) // lặp đến khi tất cả các process hoàn thành: comp=n
55 {
56     smallest = n; // vì t[n].re_bur có giá trị INT_MAX nên process i nào đến đầu tiên thì smallest sẽ được gán bằng i
57
58     for (i = 0; i < n; i++) // duyệt qua tất cả process
59     {
60         if (t[i].arr <= time_current && t[i].re_bur > 0) // nếu t[i] đã đến và t[i] chưa chạy xong thì
61         {
62             // chọn tiến trình có thời gian burst còn lại nhỏ nhất
63             if (t[i].re_bur < t[smallest].re_bur)
64                 smallest = i;
65             /* trường hợp tiến trình đến có burst time đúng bằng remaining burst time còn lại
66             của tiến trình đang thực thi thì tiến trình đến trước thực hiện trước*/
67             if (t[i].re_bur == t[smallest].re_bur)
68                 if (t[i].arr < t[smallest].arr)
69                     smallest = i;
70         }
71     }
72     // tính start time
73     if (t[smallest].re_bur == t[smallest].bur)
74         t[smallest].stat = time_current;
75     // tính lại remaining time của tiến trình
76     t[smallest].re_bur--;
77
78     if (t[smallest].re_bur == 0) // Khi tiến trình hoàn thành
79     {
80         comp++; // tăng số tiến trình đã hoàn thành
81         t[smallest].finish = time_current + 1; // finish time
82         t[smallest].wt = t[smallest].finish - t[smallest].arr - t[smallest].bur; // waiting time
83         t[smallest].tat = t[smallest].finish - t[smallest].arr; // turnaround time
84     }
85 }
```


```
86 // In kết quả
87 cout << "Process" << setw(10) << "ArrTime" << setw(12) << "BurstTime" << setw(6) << "TAT" << setw(15)
88 << "ResponseTime" << setw(10) << "Finish" << setw(15) << "Waiting Time\n";
89 for (i = 0; i < n; i++)
90 {
91     tottat += t[i].tat;
92     totwt += t[i].wt;
93     tottrt += t[i].stat - t[i].arr;
94     cout << t[i].pn << setw(12) << t[i].arr << setw(11) << t[i].bur << setw(10) << t[i].tat << setw(10)
95 << t[i].stat - t[i].arr << setw(13) << t[i].finish << setw(10) << t[i].wt << "\n";
96 }
97
98 cout << "Average Turnaround Time: " << (float)tottat / n << "\n";
99 cout << "Average Waiting : " << (float)totwt / n << "\n";
100 cout << "Average Response Time : " << (float)tottrt / n << "\n";
101 return 0;
102 }
```

4. Kiểm tra tính đúng đắn của code

Test case 1:

Process Name	Arrival Time	Burst Time
1	3	4
2	10	7
3	1	9
4	8	4
5	2	6

Theo lý thuyết:

	P3	P5	P1	P5	P4	P2	P3	
0	1	2	3	7	12	16	23	31

Process	Arrival Time	Burst Time	Turnaround Time	Response Time	Waiting Time
1	3	4	4	0	0
2	10	7	13	6	6
3	1	9	30	0	21
4	8	4	8	4	4
5	2	6	10	0	4

Average Turnaround Time: 13

Average Waiting: 7

Average Response Time: 2

Kết quả code:

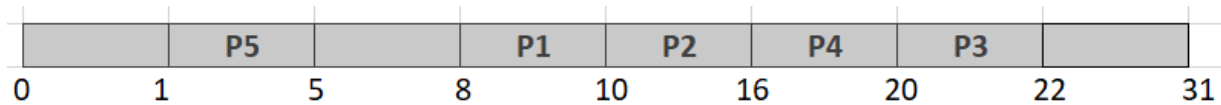
Process	ArrTime	BurstTime	TAT	ResponseTime	Finish	Waiting Time
1	3	4	4	0	7	0
2	10	7	13	6	23	6
3	1	9	30	0	31	21
4	8	4	8	4	16	4
5	2	6	10	0	12	4
Average Turnaround Time: 13						
Average Waiting : 7						
Average Response Time : 2						

o leduonghoangkimphuong-22521163@PhuongLDHK:~\$

Test case 2:

Process Name	Arrival Time	Burst Time
1	8	2
2	9	6
3	20	2
4	12	4
5	1	4

Theo lý thuyết:



Process	Arrival Time	Burst Time	Turnaround Time	Response Time	Waiting Time
1	8	2	2	0	0
2	9	6	7	1	1
3	20	2	2	0	0
4	12	4	8	4	4
5	1	4	4	0	0

Average Turnaround Time: 4.6

Average Waiting: 1

Average Response Time: 1

Kết quả code:

```
● leduonghoangkimphuong-22521163@PhuongLDHK:~$ ./SRTF
Enter the number of processes: 5
Process  ArrTime  BurstTime  TAT  ResponseTime  Finish  Waiting Time
1         8         2         2         0         10         0
2         9         6         7         1         16         1
3        20         2         2         0         22         0
4        12         4         8         4         20         4
5         1         4         4         0          5         0
Average Turnaround Time: 4.6
Average Waiting : 1
Average Response Time : 1
```

Test case 3:

Process Name	Arrival Time	Burst Time
1	10	11
2	8	2
3	18	9
4	10	12
5	14	5

Theo lý thuyết:

		P2	P1	P5	P1	P3	P4	
0	8	10	14	19	26	35	47	

Process	Arrival Time	Burst Time	Turnaround Time	Response Time	Waiting Time
1	10	11	16	0	5
2	8	2	2	0	0
3	18	9	17	8	8
4	10	12	37	25	25
5	14	5	5	0	0

Average Turnaround Time: 15.4

Average Waiting: 7.6

Average Response Time: 6.6

Kết quả code:

```
● leduonghoangkimphuong-22521163@PhuongLDHK:~$ ./SRTF
Enter the number of processes: 5
Process  ArrTime  BurstTime  TAT  ResponseTime  Finish  Waiting Time
1         10         11        16         0          26         5
2          8          2         2         0          10         0
3         18          9        17         8          35         8
4         10         12       37        25          47        25
5         14          5         5         0          19         0
Average Turnaround Time: 15.4
Average Waiting : 7.6
Average Response Time : 6.6
```