

Nguyễn Nguyên Ngọc Anh, 22520058
Mai Thanh Bách, 22520090
Huỳnh Lê Minh Đại, 22520102
Hồ Tiến Vũ Bình, 22520129
Lớp: IT007.O11.1

HỆ ĐIỀU HÀNH

BÁO CÁO LAB 4

CHECKLIST

3.5. BÀI TẬP THỰC HÀNH

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

3.6. BÀI TẬP ÔN TẬP

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

Tự chấm điểm: 10

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:*

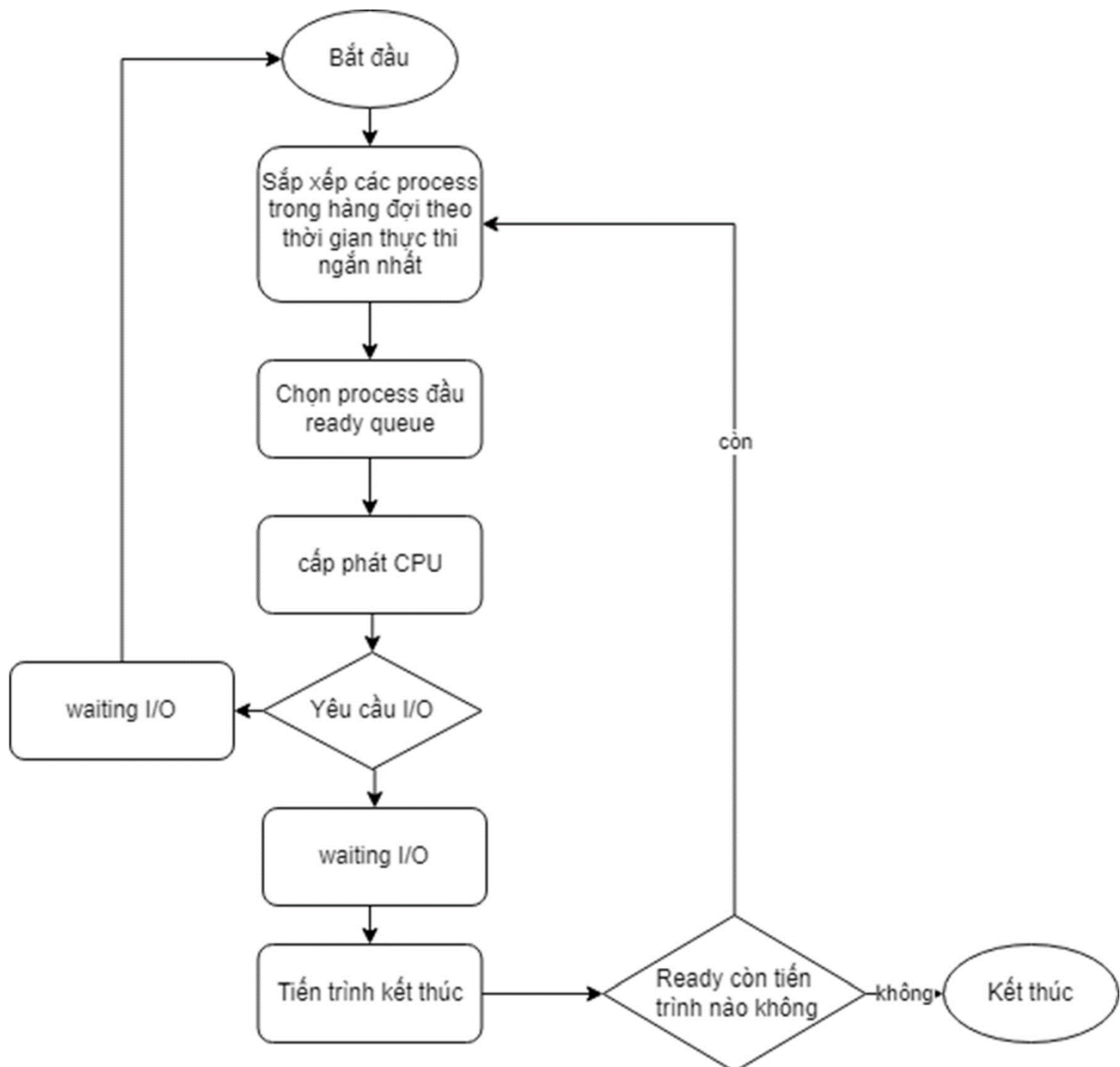
<Tên nhóm>_LAB3.pdf

2.5. BÀI TẬP THỰC HÀNH

1. Giải thuật Shortest-Job-First

Trả lời...

+ Lưu đồ giải thuật



+ **Tính đúng đắn**

Test case:

Tiến trình	Arrival Time	Burst Time
P1	0	6
P2	2	2
P3	3	8
P4	5	4
P5	8	6

+ **Chạy từng bước:**

1. Bắt đầu

2. Kiểm tra hàng đợi

3. Ready queue

[

P1(burst time = 5)

] <> rỗng => tiếp tục

4. Trong ready queue có: P1 có burst time ít nhất (5) => chọn P1

5. Thực thi P1

6. Đến thời điểm time=6, kiểm tra lại hàng đợi; ready queue lúc này:

[

P2(burst time = 2)

P3(burst time = 8)

P4(burst time = 4)

]

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

7. Trong ready queue có: P2 có burst time ít nhất (2) => chọn P2

8. Thực thi P2

9. Đến thời điểm time=8, kiểm tra lại hàng đợi; ready queue lúc này:

[

P4(burst time = 4)

P5(burst time = 6)

P3(burst time = 8)

]

10. Trong ready queue có: P4 có burst time ít nhất (4) => chọn P4

11. Thực thi P4

12. Đến thời điểm time=12, kiểm tra lại hàng đợi; ready queue lúc này:

[

P5(burst time = 6)

P3(burst time = 8)

]

13. Trong ready queue có: P5 có burst time ít nhất (6) => chọn P5

14. Thực thi P5

15. Đến thời điểm time=18, kiểm tra lại hàng đợi; ready queue lúc này:

[

P3(remaining time = 8)

]

16. Trong ready queue có: P3 có burst time ít nhất (8) => chọn P3

17. Thực thi P3

18. Đến thời điểm time=26, kiểm tra lại hàng đợi; ready queue lúc này:

[

P3(remaining time = 0)

]

19. P3 có remaining time = 0 -> Đã hoàn thành -> Xóa khỏi ready queue. Ready queue lúc này:

[]

20. Kiểm tra hàng đợi

21. Ready queue

[] == rỗng

Kết thúc

+ Code

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <limits.h>

struct Process {
    int pid;
    int arrival_time;
    int burst_time;
    int waiting_time;
    int turnaround_time;
};

void findWaitingTime(struct Process proc[], int n) {
    int remaining_time[n];
    int complete = 0, t = 0;
```

```
for (int i = 0; i < n; i++)
    remaining_time[i] = proc[i].burst_time;

while (complete != n) {
    int min = INT_MAX, shortest = -1;
    int check = 0;

    for (int j = 0; j < n; j++) {
        if (remaining_time[j] > 0 && proc[j].arrival_time <= t &&
remaining_time[j] < min) {
            min = remaining_time[j];
            shortest = j;
            check = 1;
        }
    }

    if (check == 0) {
        t++;
        continue;
    }

    remaining_time[shortest]--;

    if (remaining_time[shortest] == 0) {
        complete++;

        proc[shortest].waiting_time = t + 1 - proc[shortest].burst_time -
proc[shortest].arrival_time;
```

```
        if (proc[shortest].waiting_time < 0)
            proc[shortest].waiting_time = 0;
    }
    t++;
}
}

void findTurnAroundTime(struct Process proc[], int n) {
    for (int i = 0; i < n; i++)
        proc[i].turnaround_time = proc[i].burst_time + proc[i].waiting_time;
}

void findAverageTime(struct Process proc[], int n) {
    int total_wt = 0, total_tat = 0;

    findWaitingTime(proc, n);

    findTurnAroundTime(proc, n);

    printf("\nProcess ID  Burst Time  Waiting Time  Turnaround Time\n");

    for (int i = 0; i < n; i++) {
        total_wt = total_wt + proc[i].waiting_time;
        total_tat = total_tat + proc[i].turnaround_time;
        printf("%d\t\t%d\t\t%d\t\t%d\n", proc[i].pid, proc[i].burst_time,
proc[i].waiting_time, proc[i].turnaround_time);
    }
}
```



```
printf("\nAverage waiting time = %f", (float)total_wt / (float)n);
printf("\nAverage turnaround time = %f", (float)total_tat / (float)n);
}

void generateRandomProcesses(struct Process proc[], int n) {
    srand(time(0));
    for (int i = 0; i < n; i++) {
        proc[i].pid = i + 1;
        proc[i].arrival_time = rand() % 21; // Random Arrival Time in [0, 20]
        proc[i].burst_time = rand() % 11 + 2; // Random Burst Time in [2, 12]
        proc[i].waiting_time = 0;
        proc[i].turnaround_time = 0;
    }
}

int main() {
    int n;
    printf("Enter the number of processes: ");
    scanf("%d", &n);

    struct Process proc[n];

    generateRandomProcesses(proc, n);

    findAverageTime(proc, n);

    return 0;
}
```

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

}

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

+ **Chạy thử:**

-

Process ID	Arrival Time	Burst Time
1	20	5
2	12	9
3	7	7
4	0	9
5	11	9

Thời gian chờ (waiting time): $P1 = 0$, $P2 = 9$, $P3 = 2$, $P4 = 0$, $P5 = 19$

Thời gian kết thúc(finish time): $P1 = 42$, $P2 = 25$, $P3 = 16$, $P4 = 9$, $P5 = 37$

thời gian hoàn thành(turnaround time): $P1 = 5$, $P2 = 18$, $P3 = 9$, $P4 = 9$, $P5 = 28$

Thời gian chờ trung bình : 6

Thời gian hoàn thành trung bình: 13.8

Biểu đồ gantt:

| P4 | P3 | P2 | P5 | P1 |

0 9 16 25 37 42

+Test case 2:

Process ID	Arrival Time	Burst Time
1	1	4
2	15	6
3	13	10
4	10	2
5	13	5
6	20	2

Thời gian chờ (waiting time): $P1 = 0$, $P2 = 5$, $P3 = 13$, $P4 = 0$, $P5 = 0$, $P6 = 0$

Thời gian kết thúc(finish time): $P1 = 5$, $P2 = 24$, $P3 = 34$, $P4 = 5$, $P5 = 18$, $P6 = 22$

Thời gian hoàn thành(turnaround time): $P1 = 4$, $P2 = 11$, $P3 = 23$, $P4 = 2$, $P5 = 5$, $P6 = 2$

Thời gian chờ trung bình : 3.83

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

Thời gian hoàn thành trung bình: 7.83

Biểu đồ gantt:

| P1 | P4 | P5 | P6 | P2 | P3 |
1 5 10 12 18 24 34

+Test case 3:

Process ID	Arrival Time	Burst Time
1	13	2
2	18	9
3	0	2
4	4	9

Thời gian chờ (waiting time): $P1 = 0$, $P2 = 0$, $P3 = 0$, $P4 = 0$,

Thời gian kết thúc (finish time): $P1 = 15$, $P2 = 27$, $P3 = 2$, $P4 = 13$

Thời gian hoàn thành (turnaround time): $P1 = 2$, $P2 = 9$, $P3 = 2$, $P4 = 9$

Thời gian chờ trung bình : 0

Thời gian hoàn thành trung bình: 5.5

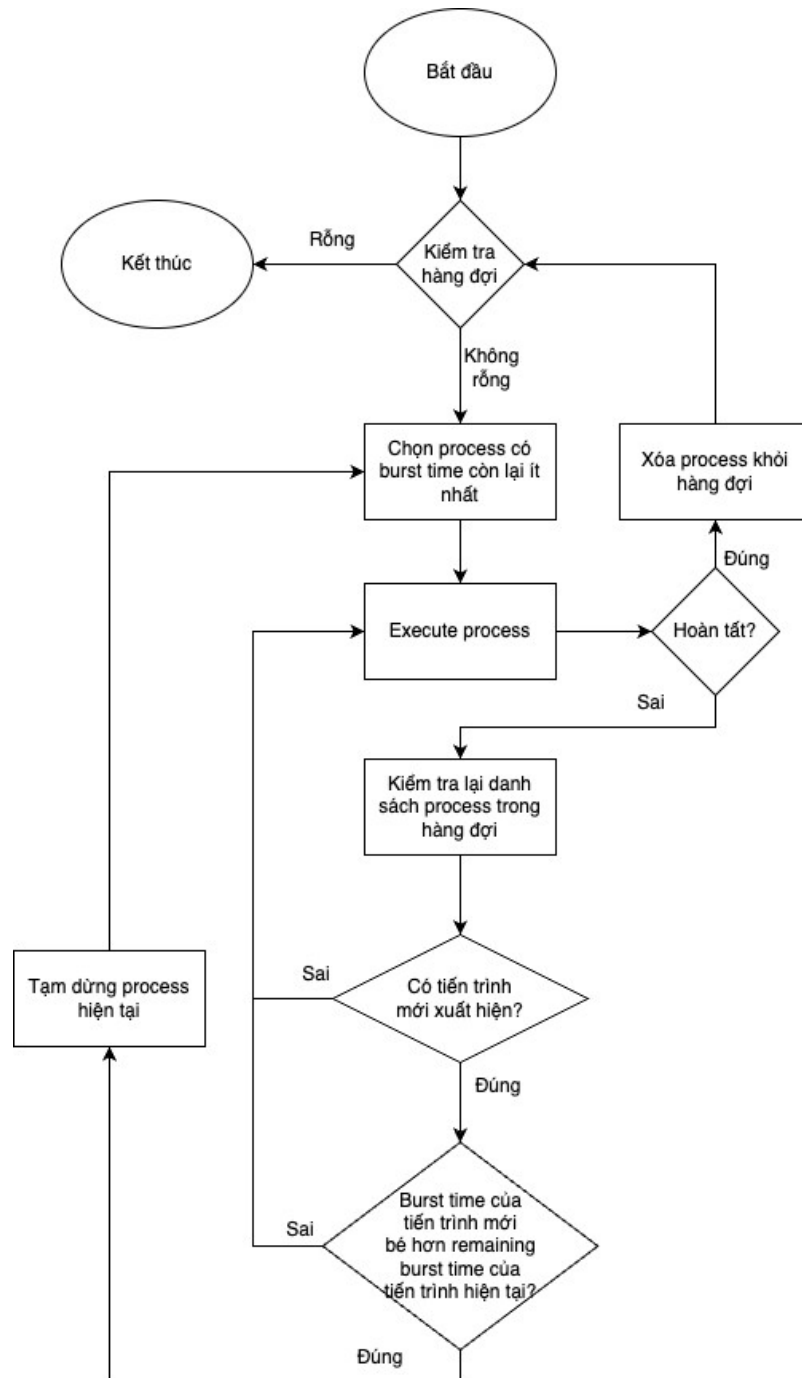
Biểu đồ gantt

| P3 | P4 | P1 | P2 |
0 2 4 13 22

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

Trả lời... **SRTF**

+ Lưu đồ giải thuật



- Chạy từng bước với test case mẫu để chứng minh lưu đồ

Tiến trình	Arrival Time	Burst Time
P1	0	5
P2	1	2
P3	3	8
P4	3	4
P5	4	6

Chạy từng bước:

1. Bắt đầu
2. Kiểm tra hàng đợi
3. Ready queue
[
 P1(remaining time = 5)
] <> rỗng => tiếp tục
4. Trong ready queue có: P1 có burst time ít nhất (5) => chọn P1
5. Thực thi P1
6. Đến thời điểm time=1, kiểm tra lại hàng đợi; ready queue lúc này:
[
 P1(remaining time = 4)
 P2(remaining time = 2)
]
7. Hàng đợi có tiến trình mới P2 xuất hiện. Kiểm tra thấy P2 có remaining time = 2 < remaining time của P1 = 4 => Tạm dừng P1
8. Trong ready queue có: P2 có burst time ít nhất (2) => chọn P2
9. Thực thi P2
10. Đến thời điểm time=2, kiểm tra lại hàng đợi; ready queue lúc này:
[
 P1(remaining time = 4)
 P2(remaining time = 1)
 P3(remaining time = 8)
]
11. Hàng đợi có tiến trình mới P3 xuất hiện. Kiểm tra thấy P3 có remaining time = 8 > remaining time của P2 = 1 => Tiếp tục P2
12. Đến thời điểm time=3, kiểm tra lại hàng đợi; ready queue lúc này:
[
 P1(remaining time = 4)

P2(remaining time = 0)
P3(remaining time = 8)
P4(remaining time = 4)

]

13. P2 có remaining time = 0 -> Đã hoàn thành -> Xóa khỏi ready queue. Ready queue lúc này:

[

P1(remaining time = 4)
P3(remaining time = 8)
P4(remaining time = 4)

]

14. Kiểm tra hàng đợi

15. Ready queue

[

P1(remaining time = 4)
P3(remaining time = 8)
P4(remaining time = 4)

] <> rỗng => tiếp tục

16. Trong ready queue có: P1 có burst time ít nhất (4) => chọn P1

17. Thực thi P1

18. Đến thời điểm time=4, kiểm tra lại hàng đợi; ready queue lúc này:

[

P1(remaining time = 3)
P3(remaining time = 8)
P4(remaining time = 4)
P5(remaining time = 6)

]

19. Hàng đợi có tiến trình mới P5 xuất hiện. Kiểm tra thấy P5 có remaining time = 6 > remaining time của P1 = 3 => Tiếp tục P1

20. Đến thời điểm time=7, kiểm tra lại hàng đợi; ready queue lúc này:

[

P1(remaining time = 0)
P3(remaining time = 8)
P4(remaining time = 4)
P5(remaining time = 6)

]

21. P1 có remaining time = 0 -> Đã hoàn thành -> Xóa khỏi ready queue. Ready queue lúc này:

[

P3(remaining time = 8)
P4(remaining time = 4)


```
                P5(remaining time = 6)
            ]
22. Kiểm tra hàng đợi
23. Ready queue
    [
        P3(remaining time = 8)
        P4(remaining time = 4)
        P5(remaining time = 6)
    ] <> rỗng => tiếp tục
24. Trong ready queue có: P4 có burst time ít nhất (4) => chọn P4
25. Thực thi P4
26. Đến thời điểm time=11, kiểm tra lại hàng đợi; ready queue lúc này:
    [
        P3(remaining time = 8)
        P4(remaining time = 0)
        P5(remaining time = 6)
    ]
27. P4 có remaining time = 0 -> Đã hoàn thành -> Xóa khỏi ready queue. Ready
    queue lúc này:
    [
        P3(remaining time = 8)
        P5(remaining time = 6)
    ]
28. Kiểm tra hàng đợi
29. Ready queue
    [
        P3(remaining time = 8)
        P5(remaining time = 6)
    ] <> rỗng => tiếp tục
30. Trong ready queue có: P5 có burst time ít nhất (6) => chọn P5
31. Thực thi P5
32. Đến thời điểm time=17, kiểm tra lại hàng đợi; ready queue lúc này:
    [
        P3(remaining time = 8)
        P5(remaining time = 0)
    ]
33. P5 có remaining time = 0 -> Đã hoàn thành -> Xóa khỏi ready queue. Ready
    queue lúc này:
    [
        P3(remaining time = 8)
    ]
34. Kiểm tra hàng đợi
```

35. Ready queue

```
[  
    P3(remaining time = 8)  
] <=> rỗng => tiếp tục
```

36. Trong ready queue có: P3 có burst time ít nhất (8) => chọn P3

37. Thực thi P3

38. Đến thời điểm time=25, kiểm tra lại hàng đợi; ready queue lúc này:

```
[  
    P3(remaining time = 0)  
]
```

39. P3 có remaining time = 0 -> Đã hoàn thành -> Xóa khỏi ready queue. Ready queue lúc này:

```
[ ]
```

40. Kiểm tra hàng đợi

41. Ready queue

```
[ ] == rỗng
```

42. Kết thúc

Code:

```
C srtf.c > Input_Process(PROCESS **, int *)  
1  #include <stdio.h>  
2  #include <stdlib.h>  
3  #include <time.h>  
4  #include <string.h>  
5  #define MAX_PROCESSES 100  
6  struct PROCESS  
7  {  
8      char NAME;  
9      int BURST_TIME;  
10     int ARRIVAL_TIME;  
11     int REPOSE_TIME;  
12     int TURN_AROUND_TIME;  
13     int FINISH_TIME;  
14     int WAITING_TIME;  
15     int done;  
16     int BURST_TIME_REMAINING;  
17 };  
18  
19 double avg_waiting_time = 0;  
20 double avg_turn_around_time = 0;  
21  
22 void Add_Process(int* size, struct PROCESS** P, struct PROCESS process)  
23 {  
24     *size += 1;  
25     *P = (struct PROCESS*)realloc(*P, sizeof(struct PROCESS) * (*size));  
26     (*P)[*size - 1] = process;  
27 }  
28
```

```
28
29 void swap(struct PROCESS* p1, struct PROCESS* p2)
30 {
31     struct PROCESS tmp = *p1;
32     *p1 = *p2;
33     *p2 = tmp;
34 }
35
36 void sortByArrivalTime(struct PROCESS* P, int size)
37 {
38     for (int i = 0; i < size; i++)
39     {
40         for (int j = i + 1; j < size; j++)
41         {
42             if (P[i].ARRIVAL_TIME > P[j].ARRIVAL_TIME)
43             {
44                 swap(&P[i], &P[j]);
45             }
46         }
47     }
48 }
49
50 void sortByBurstTimeRemaining(struct PROCESS* P, int n, int time_current)
51 {
52     for (int i = 0; i < n; i++)
53     {
54         for (int j = i + 1; j < n; j++)
55         {
56             if (P[i].BURST_TIME_REMAINING > P[j].BURST_TIME_REMAINING)
57             {
58                 swap(&P[i], &P[j]);
59             }
60         }
61     }
62     for (int i = 0; i < n; i++)
63     {
64         if (P[i].done == 0 && P[i].ARRIVAL_TIME <= time_current)
65         {
66             swap(&P[i], &P[0]);
67             break;
68         }
69     }
70 }
71
72 void Input_Process(struct PROCESS** P, int* size)
73 {
74     printf("Nhập số lượng process: ");
75     scanf("%d", size);
76     *P = (struct PROCESS*)malloc(sizeof(struct PROCESS) * (*size));
77     srand(time(NULL));
78     for (int i = 0; i < *size; i++)
79     {
80         (*P)[i].ARRIVAL_TIME = rand() % 21; // Random arrival time between 0 to 20
81         (*P)[i].BURST_TIME = rand() % 11 + 2; // Random burst time between 2 to 12
82
83         (*P)[i].NAME = '1' + i;
84         (*P)[i].BURST_TIME_REMAINING = (*P)[i].BURST_TIME;
85     }
86 }
87
```

```
87
88 char* space(int n)
89 {
90     char* res = (char*)malloc(sizeof(char) * (n + 1));
91     for (int i = 0; i < n; i++)
92     {
93         res[i] = ' ';
94     }
95     res[n] = '\0';
96     return res;
97 }
98
99 void Output_Process(int numberofprocesses, struct PROCESS* P)
100 {
101     sortByArrivalTime(P, numberofprocesses);
102     char* Attribute[6] = { "Process", "Arrival Time", "Burst Time", "Finish Time", "Turnaround Time", "Waiting Time" };
103     int num_columns = sizeof(Attribute) / sizeof(Attribute[0]);
104     char** board = (char**)malloc(sizeof(char*) * (numberofprocesses + 4));
105     for (int i = 0; i < numberofprocesses + 4; i++)
106     {
107         board[i] = (char*)malloc(sizeof(char) * 100);
108     }
109
110     sprintf(board[0], "\tShortest Remaining Time First, SRTF\n");
111
112     for (int i = 0; i < numberofprocesses + 1; i++)
113     {
114         if (i == 0)
115         {
116             sprintf(board[i + 1], "%10s%13s%11s%11s%16s%13s\n", Attribute[0],
117                 Attribute[1], Attribute[2], Attribute[3], Attribute[4], Attribute[5]);
118         }
119         else
120         {
121             int ele = i - 1;
122             char Value[6][10];
123             sprintf(Value[0], "%c", P[ele].NAME);
124             sprintf(Value[1], "%d", P[ele].ARRIVAL_TIME);
125             sprintf(Value[2], "%d", P[ele].BURST_TIME);
126             sprintf(Value[3], "%d", P[ele].FINISH_TIME);
127             sprintf(Value[4], "%d", P[ele].TURN_AROUND_TIME);
128             sprintf(Value[5], "%d", P[ele].WAITING_TIME);
129
130             sprintf(board[i + 1], "%s%s%s%s%s%s%s%s\n", space(10 - strlen(Value[0])), Value[0],
131                 space(13 - strlen(Value[1])), Value[1], space(11 - strlen(Value[2])), Value[2],
132                 space(11 - strlen(Value[3])), Value[3], space(16 - strlen(Value[4])), Value[4],
133                 space(13 - strlen(Value[5])), Value[5]);
134         }
135     }
136
137     char avg_values[100];
138     sprintf(avg_values, "Average Turnaround Time: %.2f\nAverage Waiting Time: %.2f\n", avg_turn_around_time, avg_waiting_time);
139     sprintf(board[numberofprocesses + 3], "%s", avg_values);
140 }
```

```
141     for (int i = 0; i < numberofprocesses + 4; i++)
142     {
143         printf("%s", board[i]);
144         free(board[i]);
145     }
146     free(board);
147 }
148
149 int isCompleted(struct PROCESS* P, int Size)
150 {
151     for (int i = 0; i < Size; i++)
152     {
153         if (P[i].done == 0)
154         {
155             return 0;
156         }
157     }
158     return 1;
159 }
160
161 void AVG(struct PROCESS* P, int Size)
162 {
163     for (int i = 0; i < Size; i++)
164     {
165         avg_turn_around_time += (double)P[i].TURN_AROUND_TIME / Size;
166         avg_waiting_time += (double)P[i].WAITING_TIME / Size;
167     }
168 }
169
170 int main()
171 {
172     int numberofprocesses = 1;
173     struct PROCESS* PROCESSES = NULL;
174     Input_Process(&PROCESSES, &numberofprocesses);
175     int i = 0;
176     for (int time_current = 0; !isCompleted(PROCESSES, numberofprocesses); time_current++)
177     {
178         sortByBurstTimeRemaining(PROCESSES, numberofprocesses, time_current);
179         PROCESSES[i].BURST_TIME_REMAINING--;
180         if (PROCESSES[i].BURST_TIME_REMAINING == 0)
181         {
182             PROCESSES[i].done = 1;
183             PROCESSES[i].FINISH_TIME = time_current + 1;
184             PROCESSES[i].TURN_AROUND_TIME = PROCESSES[i].FINISH_TIME - PROCESSES[i].ARRIVAL_TIME;
185             PROCESSES[i].WAITING_TIME = PROCESSES[i].TURN_AROUND_TIME - PROCESSES[i].BURST_TIME;
186         }
187     }
188     AVG(PROCESSES, numberofprocesses);
189     Output_Process(numberofprocesses, PROCESSES);
190     free(PROCESSES);
191     return 0;
192 }
193
```

+ Tính đúng đắn của code:

Testcase1:

```
hotienvubinh-22520129@LAPTOP-J7OECDF:~/Lab4$ ./srtf
Nhập số lượng process: 5
Shortest Remaining Time First, SRTF
| Process | Arrival Time | Burst Time | Finish Time | Turnaround Time | Waiting Time |
|-----|-----|-----|-----|-----|-----|
| 3 | 0 | 4 | 4 | 4 | 0 |
| 2 | 14 | 8 | 22 | 8 | 0 |
| 4 | 16 | 10 | 32 | 16 | 6 |
| 1 | 18 | 12 | 54 | 36 | 24 |
| 5 | 19 | 10 | 42 | 23 | 13 |
Average Around Time: 17.40
Average Waiting Time: 8.60
```

Chạy tay:

Tiến trình	Arrival time	Burst time
P1	18	12
P2	14	8
P3	0	4
P4	16	10
P5	19	10

Biểu đồ Grant

P3	Đợi	P2	P4	P5	P1	
0	4	14	22	32	42	54

Thời gian chờ (wating time) : $P1 = 42 - 18 = 24$ | $P2 = 0$ | $P3 = 0$ | $P4 = 22 - 16 = 6$ |

$P5 = 32 - 19 = 13$

Thời gian kết thúc (finish time): $P1 = 54$ | $P2 = 22$ | $P3 = 4$ | $P4 = 32$ | $P5 = 42$

Thời gian hoàn thành (turnaround time): $P1 = 36$ | $P2 = 8$ | $P3 = 4$ | $P4 = 16$ | $P5 = 23$

Average Turnaround Time: 17.40

Average Waiting Time: 8.60

Testcase 2:

```
● hotienvubinh-22520129@LAPTOP-J70ECDJF:~/Lab4$ ./srtf
Nhập số lượng process: 5
Shortest Remaining Time First, SRTF
| Process | Arrival Time | Burst Time | Finish Time | Turnaround Time | Waiting Time |
|-----|-----|-----|-----|-----|-----|
| 3 | 0 | 8 | 8 | 8 | 0 |
| 2 | 1 | 10 | 36 | 35 | 25 |
| 4 | 8 | 7 | 15 | 7 | 0 |
| 1 | 16 | 4 | 20 | 4 | 0 |
| 5 | 17 | 7 | 27 | 10 | 3 |
Average Around Time: 12.80
Average Waiting Time: 5.60
```

Chạy tay:

Tiến trình	Arrival time	Burst time
P1	16	4
P2	1	10
P3	0	8
P4	8	7
P5	17	7

Biểu đồ Grant:

P3	P4	P2	P1	P5	P2	
0	8	15	16	20	27	36

Thời gian chờ (wating time) : P1 = 0 | P2 = 25 | P3 = 0 | P4 = 0 | P5 = 3

Thời gian kết thúc (finish time): P1 = 20 | P2 = 36 | P3 = 8 | P4 = 15 | P5 = 27

Thời gian hoàn thành (turnaround time): P1 = 4 | P2 = 35| P3 = 8 | P4 = 7| P5 = 10

Average Turnaround Time: 12.8

Average Waiting Time: 5.6

Testcase 3:

```
● hotienvubinh-22520129@LAPTOP-J70ECDJF:~/Lab4$ ./srtf
Nhập số lượng process: 5
Shortest Remaining Time First, SRTF
| Process | Arrival Time | Burst Time | Finish Time | Turnaround Time | Waiting Time |
|-----|-----|-----|-----|-----|-----|
| 5 | 0 | 5 | 5 | 5 | 0 |
| 1 | 7 | 11 | 24 | 17 | 6 |
| 4 | 8 | 3 | 11 | 3 | 0 |
| 2 | 10 | 3 | 14 | 4 | 1 |
| 3 | 20 | 12 | 36 | 16 | 4 |
Average Around Time: 9.00
Average Waiting Time: 2.20
```

Chạy tay:

Tiến trình	Arrival time	Burst time
P1	7	11
P2	10	3
P3	20	12
P4	8	3
P5	0	5

Biểu đồ Grant:

P5	Đợi	P1	P4	P2	P1	P3	
0	5	7	8	11	14	24	36

Thời gian chờ (wating time) : P1 = 6 | P2 = 1 | P3 = 4 | P4 = 0 | P5 = 0

Thời gian kết thúc (finish time): P1 = 24 | P2 = 14 | P3 = 36 | P4 = 11 | P5 = 5

Thời gian hoàn thành (turnaround time): P1 = 17 | P2 = 4 | P3 = 16 | P4 = 3 | P5 = 5

Average Turnaround Time: 9

Average Waiting Time: 2.2

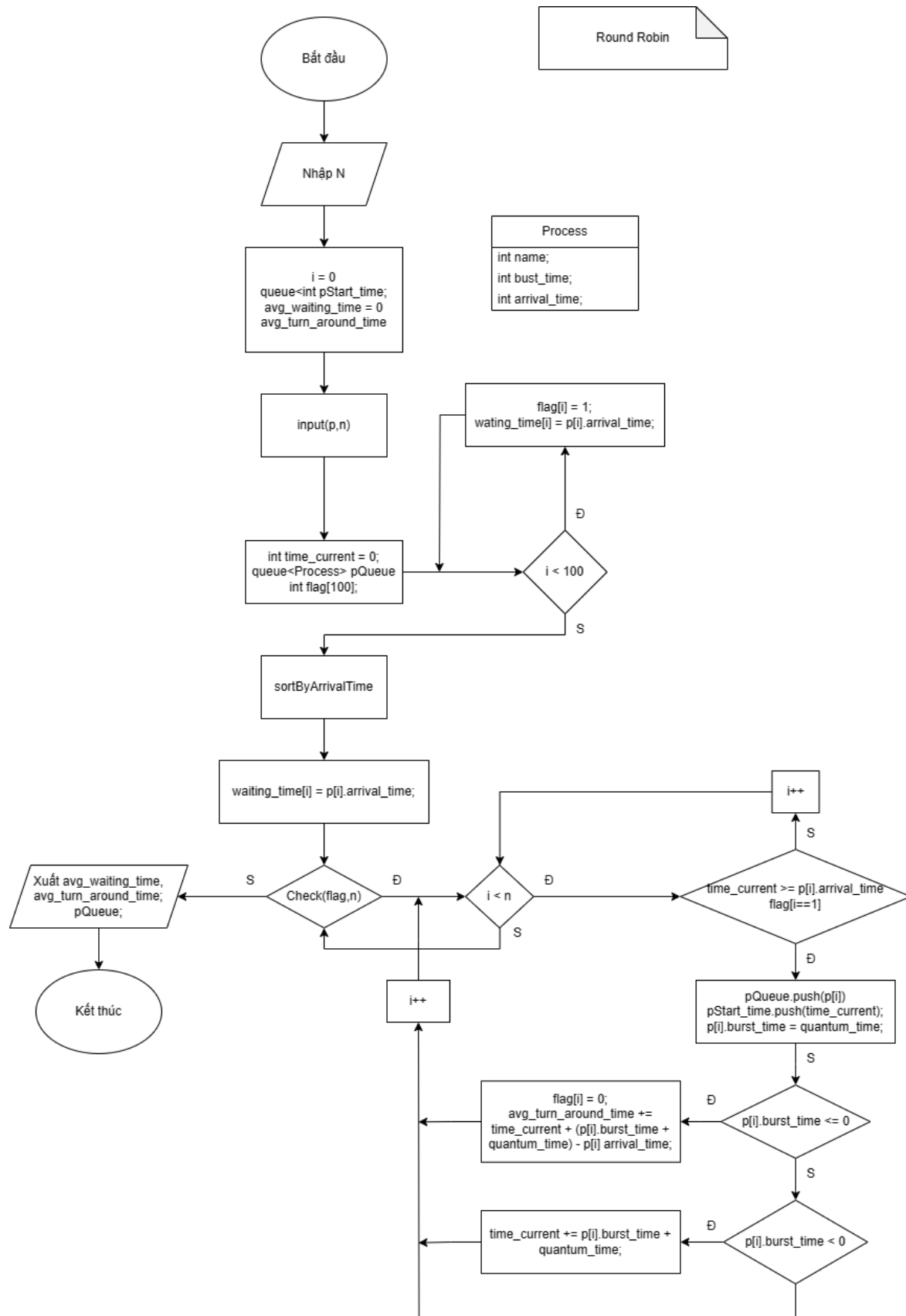
2.6. BÀI TẬP ÔN TẬP

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

Trả lời...

Round Robin

Lưu đồ giải thuật:



Chạy tay lưu đồ giải thuật:

P1 0 5

P2 2 2

P3 3 8

P4 5 4

P5 8 6

1. Bắt đầu

2. Kiểm tra hàng đợi

3. time = 0, ready queue:

[

P1 (BT = 5)

]

4. Thực thi P1

5. time = 2, ready queue:

[

P1 (BT = 3)

P2 (BT = 2)

]

6. thực thi P1

7. time = 3, ready queue:

[

P1 (BT = 2)

P2 (BT = 2)

P3 (BT = 8)

]

8. Thực thi P1

8. time = 5, P1 hoàn thành, xóa P1 khỏi queue, ready queue:

[

P2 (BT = 2)

P3 (BT = 8)

P4 (BT = 4)

]

9. Thực thi P2

10. time = 7, P2 hoàn thành, xóa P2 khỏi queue, ready queue:

[

P3 (BT = 8)

P4 (BT = 4)

]

11. Thực thi P3

12. time = 8; ready queue:

[

P3 (BT = 7)

P4 (BT = 4)

P5 (BT = 6)

12. Thực thi P3

13. time = 12, hết quantum time, ready queue:

[

P4 (BT = 4)

P5 (BT = 6)

P3 (BT = 2)

]

14. Thực thi P4

15. time = 16, P4 hoàn thành, xóa P4 khỏi queue, ready queue:

[

P5 (BT = 6)

P3 (BT = 2)

]

16. Thực thi P5

17. time = 21, hết quantum time, ready queue:

[

P3 (BT = 2)

P5 (BT = 1)

]

18. Thực thi P3

19. time = 23, P3 hoàn thành, xóa P3 khỏi queue, ready queue:

[

P5 (BT = 1)

]

20. Thực thi P5

21. time = 23, P1 hoàn thành, xóa P1 khỏi queue, ready queue:

[]

22. Kiểm tra hàng đợi

23. Ready queue [] == rỗng

24. Kết thúc

Hiện thực code:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

void SearchStack01(int pnt, int tmx, int n);
void SearchStack02(int pnt, int tmx, int n);
void AddQue(int pnt);

int at[50], bt[50], ct[50] = { 0 }, qt, rqi[50] = { 0 }, st, flg = 0, tmx = 0, noe = 0, pnt =
```

```
0, btm[50] = { 0 }, tt, wt;
float att, awt;

int main()
{
    int n;
    printf("Nhap vao so tien trinh: ");
    scanf("%d", &n);

    srand(time(NULL));

    for (int x = 0; x < n; x++)
    {
        printf("Process %d\n", x + 1);
        printf("AT=");
        at[x] = rand() % 21;
        printf("%d\n", at[x]);
        printf("BT=");
        bt[x] = rand() % 11 + 2;
        printf("%d\n", bt[x]);
        btm[x] = bt[x];
    }

    for (int i = 0; i < n - 1; i++)
    {
        for (int j = 0; j < n - i - 1; j++)
        {
            if (at[j] > at[j + 1])
```

```
{
    int temp = at[j];
    at[j] = at[j + 1];
    at[j + 1] = temp;

    temp = bt[j];
    bt[j] = bt[j + 1];
    bt[j + 1] = temp;

    temp = btm[j];
    btm[j] = btm[j + 1];
    btm[j + 1] = temp;
}
}
}

printf("Nhap quantum time: ");
scanf("%d", &qt);

system("clear");

printf("\n\nGANTT CHART\n%d", at[0]);
do
{
    if (flg == 0)
    {
        st = at[0];
```

```
if (btm[0] <= qt)
{
    tmx = st + btm[0];
    btm[0] = 0;
    SearchStack01(pnt, tmx, n);
}
else
{
    btm[0] = btm[0] - qt;
    tmx = st + qt;
    SearchStack01(pnt, tmx, n);
    AddQue(pnt);
}
}
else
{
    pnt = rqi[0] - 1;
    st = tmx;
    for (int x = 0; x < noe && noe != 1; x++)
    {
        rqi[x] = rqi[x + 1];
    }
    noe--;

    if (btm[pnt] <= qt)
    {
        tmx = st + btm[pnt];
        btm[pnt] = 0;
```



```
        SearchStack02(pnt, tmx, n);
    }
    else
    {
        btm[pnt] = btm[pnt] - qt;
        tmx = st + qt;
        SearchStack02(pnt, tmx, n);
        AddQue(pnt);
    }
}

if (btm[pnt] == 0)
{
    ct[pnt] = tmx;
}

flg++;
printf("-p%d-[%d]", pnt + 1, tmx);
} while (noe != 0);

printf("\n\nPROCESS\t AT\t BT\t CT\t TT\t WT\n");
for (int x = 0; x < n; x++)
{
    tt = ct[x] - at[x];
    wt = tt - bt[x];
    printf("P%d\t%d\t%d\t%d\t%d\t%d\n", x + 1, at[x], bt[x], ct[x], tt, wt);
    awt = awt + wt;
    att = att + tt;
}
```

```
}

printf("\nAVERAGE TT: %f\nAVERAGE WT: %f\n", att / n, awt / n);
return 0;
}

void SearchStack01(int pnt, int tm, int n)
{
    for (int x = pnt + 1; x < n; x++)
    {
        if (at[x] <= tm)
        {
            rqi[noe] = x + 1;
            noe++;
        }
    }
}

void SearchStack02(int pnt, int tm, int n)
{
    for (int x = pnt + 1; x < n; x++)
    {
        int fl = 0;
        for (int y = 0; y < noe; y++)
        {
            if (rqi[y] == x + 1)
            {
                fl++;
            }
        }
    }
}
```

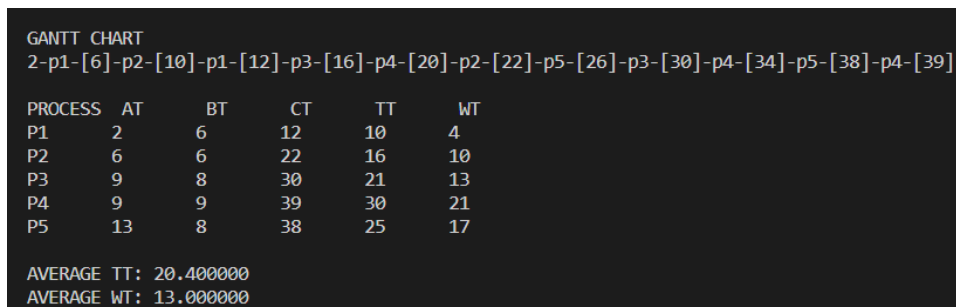
```
    }
}

if (at[x] <= tmx && fl == 0 && btm[x] != 0)
{
    rqi[noe] = x + 1;
    noe++;
}
}
}

void AddQue(int pnt)
{
    rqi[noe] = pnt + 1;
    noe++;
}
```

Chạy Test case:

Test case 1: Quantum time = 4



GANTT CHART
2-p1-[6]-p2-[10]-p1-[12]-p3-[16]-p4-[20]-p2-[22]-p5-[26]-p3-[30]-p4-[34]-p5-[38]-p4-[39]

PROCESS	AT	BT	CT	TT	WT
P1	2	6	12	10	4
P2	6	6	22	16	10
P3	9	8	30	21	13
P4	9	9	39	30	21
P5	13	8	38	25	17

AVERAGE TT: 20.400000
AVERAGE WT: 13.000000

Kiểm tra:

Gantt Chart

| Time | 2 | 6 | 10 | 12 | 16 | 20 | 22 | 26 | 30 | 34 | 38 | 39

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

| P1 | P2 | P1 | P3 | P4 | P2 | P5 | P3 | P4 | P5 | P4 |

CT: P1 = 12, P2 = 10, P3 = 30, P4 = 39, P5 = 38

TT: P1 = 10, P2 = 16, P3 = 21, P4 = 30, P5 = 25

WT: P1 = 4, P2 = 10, P3 = 13, P4 = 21, P5 = 17

ATT = 20.4

AWT = 13

Test case 2: Quantum time = 5

GANTT CHART					
8-p1-[13]-p2-[18]-p1-[19]-p3-[23]-p2-[28]-p4-[33]-p5-[38]-p2-[40]-p4-[45]-p5-[48]-p4-[49]					
PROCESS	AT	BT	CT	TT	WT
P1	8	6	19	11	5
P2	9	12	40	31	19
P3	17	4	23	6	2
P4	19	11	49	30	19
P5	20	8	48	28	20
AVERAGE TT: 21.200001					
AVERAGE WT: 13.000000					

Kiểm tra:

Gantt Chart

| Time | 8 | 13 | 18 | 19 | 23 | 28 | 33 | 38 | 40 | 45 | 48 | 49 |

| P1 | P2 | P1 | P3 | P2 | P4 | P5 | P2 | P4 | P5 | P4 |

CT: P1 = 19, P2 = 40, P3 = 23, P4 = 49, P5 = 48

TT: P1 = 11, P2 = 31, P3 = 6, P4 = 30, P5 = 28

WT: P1 = 5, P2 = 19, P3 = 2, P4 = 19, P5 = 20

ATT = 21.2

AWT = 13

Test case 3: Quantum time = 6

GANTT CHART					
6-p1-[12]-p2-[18]-p1-[20]-p3-[26]-p4-[32]-p5-[38]-p2-[39]-p3-[43]-p4-[47]-p5-[50]					
PROCESS	AT	BT	CT	TT	WT
P1	6	8	20	14	6
P2	11	7	39	28	21
P3	13	10	43	30	20
P4	14	10	47	33	23
P5	15	9	50	35	26
AVERAGE TT: 28.000000					
AVERAGE WT: 19.200001					

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

Kiểm tra:

Gantt Chart

| Time | 6 | 12 | 18 | 20 | 26 | 32 | 38 | 39 | 43 | 47 | 50

| P1 | P2 | P1 | P3 | P4 | P5 | P2 | P3 | P4 | P5 |

CT: P1 = 20, P2 = 39, P3 = 43, P4 = 47, P5 = 50

TT: P1 = 14, P2 = 28, P3 = 30, P4 = 33, P5 = 35

WT: P1 = 6, P2 = 21, P3 = 20, P4 = 23, P5 = 26

ATT = 28

AWT = 19.2

