Name: Phạm Nguyễn Hải Anh

ID: 21520586

Class: IT007.ANTN

OPERATING SYSTEM
LAB X'S REPORT

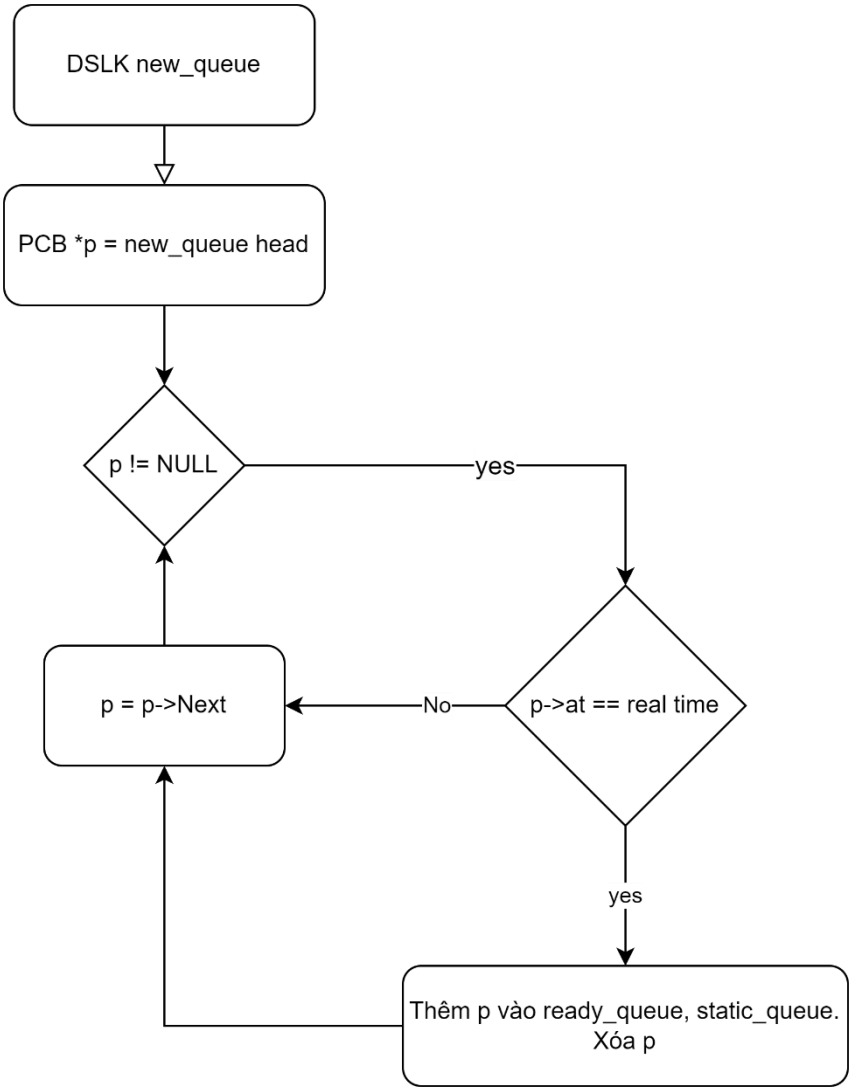
SUMMARY

Self-scrores:

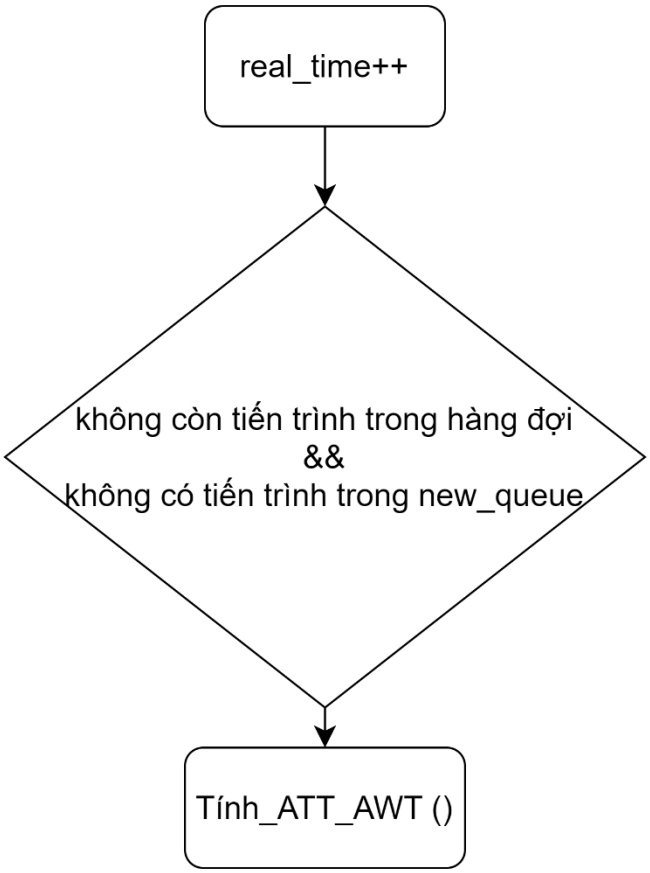
Note: Export file to **PDF and name the file by following format:*
Student ID_LABx.pdf

Trước khi bắt đầu, các giải thuật bên dưới đều có chung 3 luồng, chỉ khác nhau ở tham số của đoạn code hàm thuc_thi. Soucre code lưu ở [Lab4 - Google Drive]

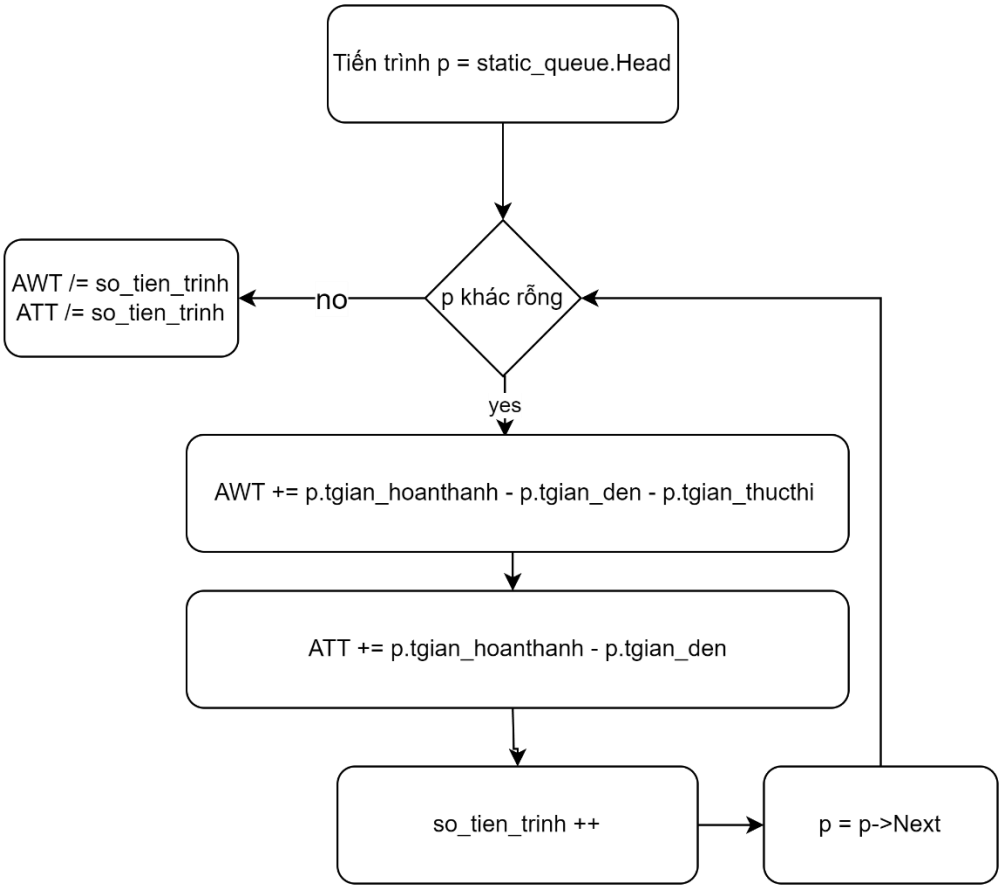
Luồng 1:



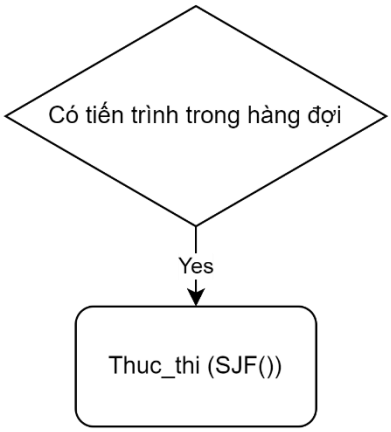
Luồng 2:


real_time++
không còn tiến trình trong hàng đợi
&&
không có tiến trình trong new_queue
Tính_ATT_AWT ()

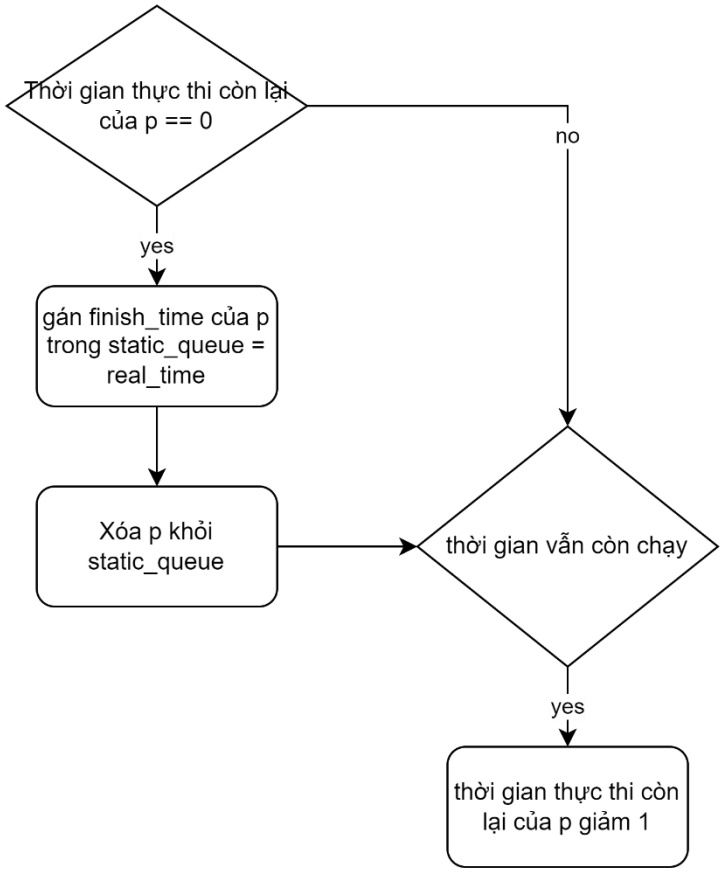
Tinh_AWT_ATT():



Luồng 3:

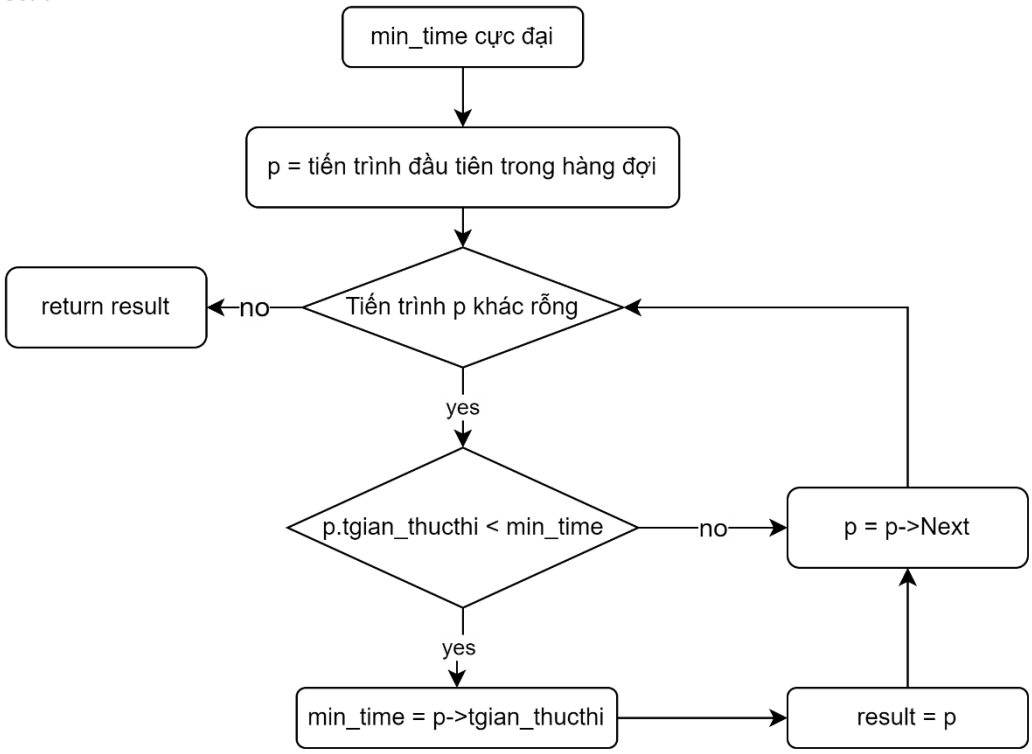


Thuc_thi(Tiến trình p):

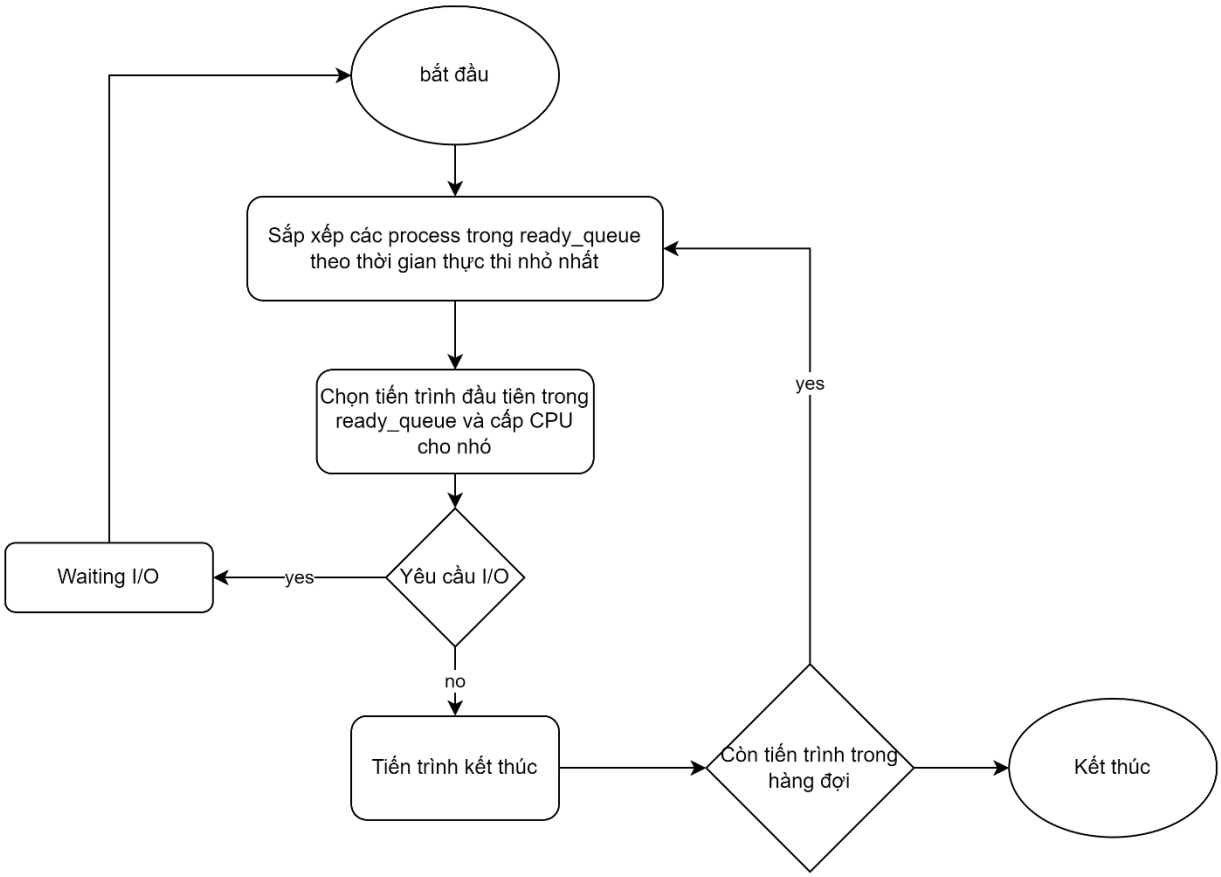


1. SJF:

SJF:



Thuật toán này nhìn có vẻ đúng, nhưng thật chất nó có trưng dụng. Vì thế em đã thay thế nó bằng 1 kiểu khác:



Tính đúng đắn:

Input theo format PID – AT – BT:

1 0 13

2 4 9

3 6 4

4 7 18

5 12 10

Thời gian t = 0:

Ready queue chỉ có process pid 1 nên pid 1 là tiến trình có thời gian thực thi nhỏ nhất. Chạy pid 1.

Process pid 1 chạy tới t=13, kiểm tra trong hàng đợi thấy còn tiến trình:

Ready queue có process pid 2, process pid 3, process pid 4, process pid 5. Pid 3 có thời gian thực thi nhỏ nhất, chạy pid 3.

Chạy tới t=13+4=17:

Ready queue có process pid 2, process pid 4, process pid 5. Pid 2 có thời gian thực thi nhỏ nhất, chạy pid 2.

Chạy tới t=17+9=26:

Ready queue có process pid 4, process pid 5. Pid 5 có thời gian thực thi nhỏ nhất, chạy pid 5.

Chạy tới t=26+10=36:

Ready queue có process pid 4 nên Pid 4 có thời gian thực thi nhỏ nhất, chạy pid 4.

Chạy tới t=36+18=54:

Ready queue không còn tiến trình nên kết thúc chương trình.

Code chính:

```
void non_preemptive_sjf(int n)
{
    // To store the number of processes that have been completed
    int counter = n;

    // To keep an account of the number of processes that have been arrived
    int upper_range = 0;

    // Current running time
    int tm = min(INT_MAX, ar[upper_range + 1].at);

    // To find the list of processes whose arrival time is less than or equal to the current time
    while (counter) {
        for (; upper_range <= n;) {
            upper_range++;
            if (ar[upper_range].at > tm || upper_range > n) {
                upper_range--;
                break;
            }

            update(1, 1, n, upper_range,
                ar[upper_range].id, ar[upper_range].bt);
        }

        // To find the minimum of all the running times from the set of processes whose arrival time is less than or equal to the current time
        util1 res = query(1, 1, n, 1, upper_range);

        // Checking if the process has already been executed
        if (res.bt1 != INT_MAX) {
            counter--;
            int index = mp[res.p_id];
            tm += (res.bt1);

            // Calculating and updating the array with the current time, turn around time and waiting time
            ar[index].ct = tm;
            ar[index].tat = ar[index].ct - ar[index].at;
            ar[index].wt = ar[index].tat - ar[index].bt;

            // Update the process burst time with infinity when the process is executed
            update(1, 1, n, index, INT_MAX, INT_MAX);
        }
        else {
            tm = ar[upper_range + 1].at;
        }
```

Thực thi:

Test case 1:

Input theo format PID – AT – BT:

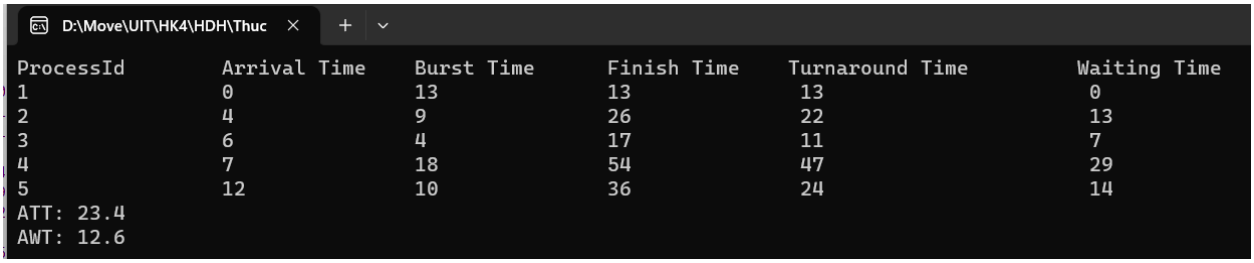
1 0 13

2 4 9

3 6 4

4 7 18

5 12 10



Test case 2:

Input theo format PID – AT – BT:

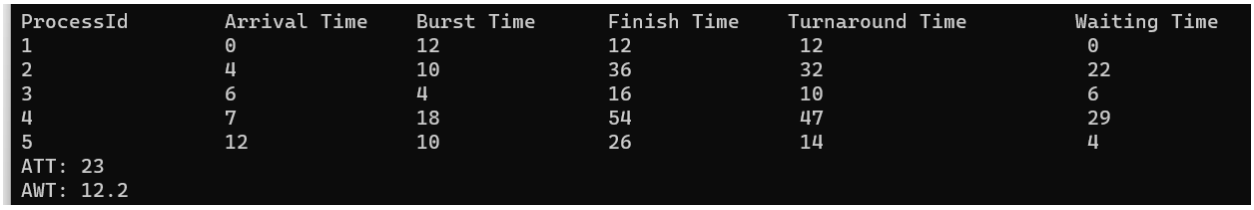
1 0 12

2 4 10

3 6 4

4 7 18

5 12 10



Test case 3:

Input theo format PID – AT – BT:

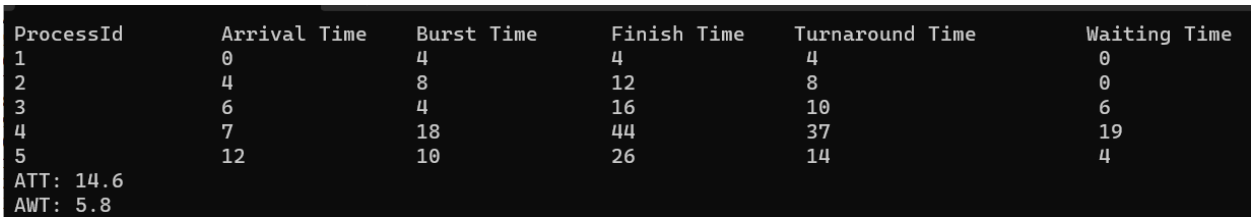
1 0 4

2 4 8

3 6 4

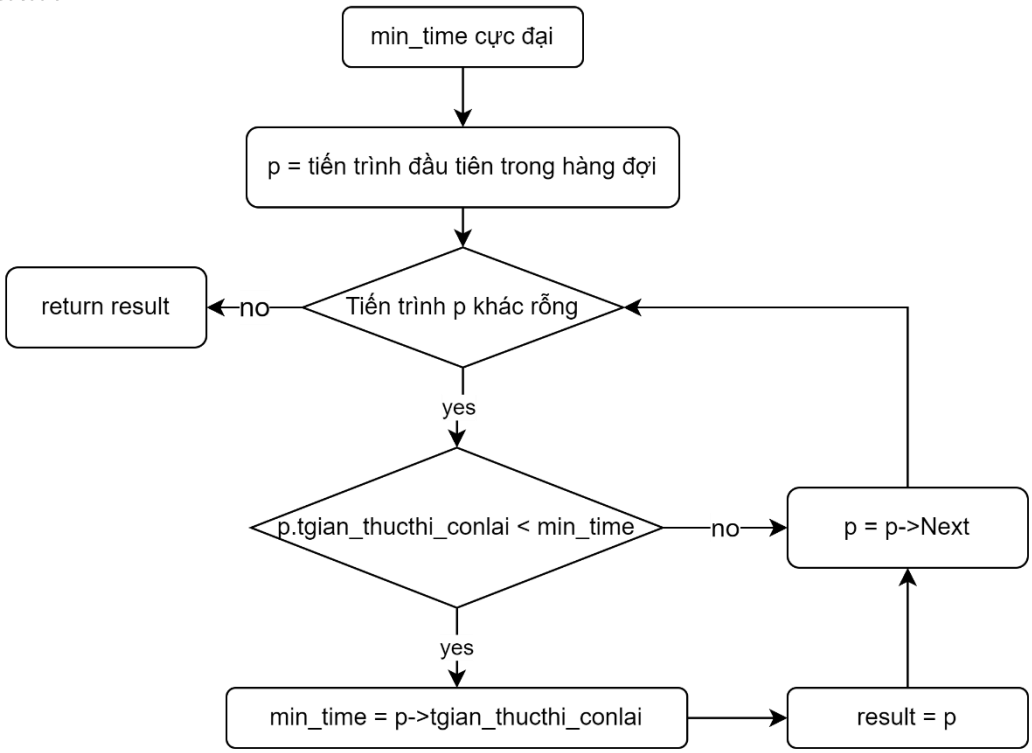
4 7 18

5 12 10



2. SRTF:

SRTF:



Tính đúng đắn:

Input theo format PID – AT – BT:

```
1 1 3
2 0 2
3 4 1
4 6 2
5 5 3
```

Tại t=0:

Ready queue chỉ có mỗi process pid=2, nên process này có thời gian thực thi còn lại nhỏ nhất. Chạy pid 2.

Tại t=1:

Ready_queue có process pid=2, process pid=1. Thời gian thực thi còn lại của pid=2 là 1, của pid=1 là 3 nên process này có thời gian thực thi còn lại nhỏ nhất. Chạy pid 2.

Tại t=2:

Ready_queue chỉ có process pid=1 nên process này có thời gian thực thi còn lại nhỏ nhất. Chạy pid 1.

Tại t=3:

Ready_queue chỉ có process pid=1 nên process này có thời gian thực thi còn lại nhỏ nhất. Chạy pid 1.

Tại t=4:

Ready_queue có process pid=1, process pid=3. Thời gian thực thi còn lại của pid=3 và pid=1 là 1 nên process pid=1 vẫn chạy.

Tại t=5:

Ready_queue có process pid=3, process pid=5. Thời gian thực thi còn lại của pid=3 là 1, của pid=5 là 3 nên process pid=3 có thời gian thực thi còn lại nhỏ nhất. Chạy pid 3.

Tại t=6:

Ready_queue có process pid=4, process pid=5. Thời gian thực thi còn lại của pid=4 là 2, của pid=5 là 3 nên process pid=4 có thời gian thực thi còn lại nhỏ nhất. Chạy pid 4.

Tại t=7:

Ready_queue có process pid=4, process pid=5. Thời gian thực thi còn lại của pid=4 là 1, của pid=5 là 3 nên process pid=4 có thời gian thực thi còn lại nhỏ nhất. Chạy pid 4.

Còn lại là chương trình sẽ chạy process pid=5.

Code chính:

```
159
160 PCB* find_shortest_remaining_time() {
161     int min_time = 9999999;
162     PCB* result = new PCB;
163     for (PCB* p = ready_queue.pHead; p != NULL; p = p->pNext) {
164         if (p->remaining_time < min_time) {
165             min_time = p->remaining_time;
166             result = p;
167         }
168     }
169     return result;
170 }
171
172 void* short_scheduler(void *message) {
173     while (1) {
174         if (ready_queue.pHead != NULL) {
175             execute(find_shortest_remaining_time());
176         }
177     }
178 }
179
180 void* long_scheduler(void *message) {
181     while (1) {
182         for (PCB *p = new_queue.pHead; p != NULL; p = p->pNext) {
183             if (p->arrival_time == system_time) {
184                 add_task(ready_queue, create_task(p->pID, p->arrival_time, p->burst_time));
185                 add_task(statistics_list, create_task(p->pID, p->arrival_time, p->burst_time));
186                 remove_task(new_queue, p);
187             }
188         }
189     }
190 }
191
```

Thực thi: Input theo format PID – AT – BT:

Test case1:

```
1 1 3
2 0 2
3 4 1
4 6 4
5 5 5
```

```
Grant diagram:
2
2
1
1
1
3
4
4
4
4
5
5
5
5
5
pID     arrival_time    burst_time      finnish_time
2       0               2               2
1       1               3               5
3       4               1               6
5       5               5               15
4       6               4               10
avg_wait_time: 1.4
avg_turnaround_time: 4.4
```

Test case 2:

```
1 1 3
2 0 2
3 4 1
4 6 2
5 5 3
```

```
Grant diagram:
2
2
1
1
1
3
4
4
5
5
5
pID     arrival_time    burst_time      finnish_time
2       0               2               2
1       1               3               5
3       4               1               6
5       5               3               11
4       6               2               8
avg_wait_time: 1
avg_turnaround_time: 3.2
```

Test case 3:

```
1 1 3
2 0 2
3 4 4
4 6 2
5 5 3
```

```
Grant diagram:
2
2
1
1
1
5
5
5
4
4
3
3
3
3
pID     arrival_time    burst_time      finnish_time
2       0               2               2
1       1               3               5
3       4               4               14
5       5               3               8
4       6               2               10
avg_wait_time: 1.8
avg_turnaround_time: 4.6
```