Họ và tên: Đặng Thiên Ân
MSSV: 23520003
Lớp: IT007.P110.2

HỆ ĐIỀU HÀNH
BÁO CÁO LAB 4

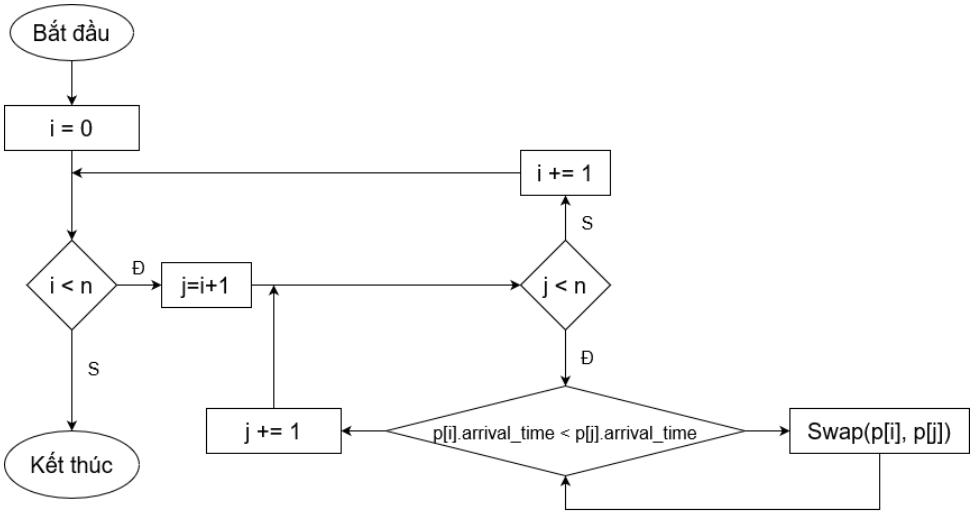
Tiêu Đề: 23520003-LAB4
Nội dung:

1> Xong
2> Xong
3> Xong
4> Xong

Section 4.5

Task name 1: Viết chương trình mô phỏng giải thuật SJF.

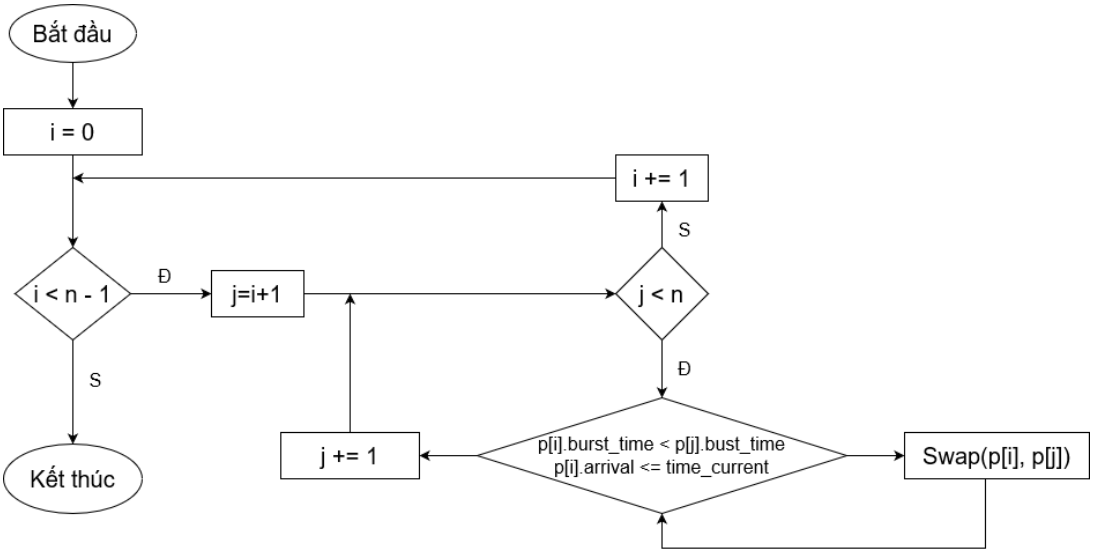
Hàm sort các tiến trình theo arrival time



Hình 1: Lưu đồ hàm sort các tiến trình dựa vào 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.

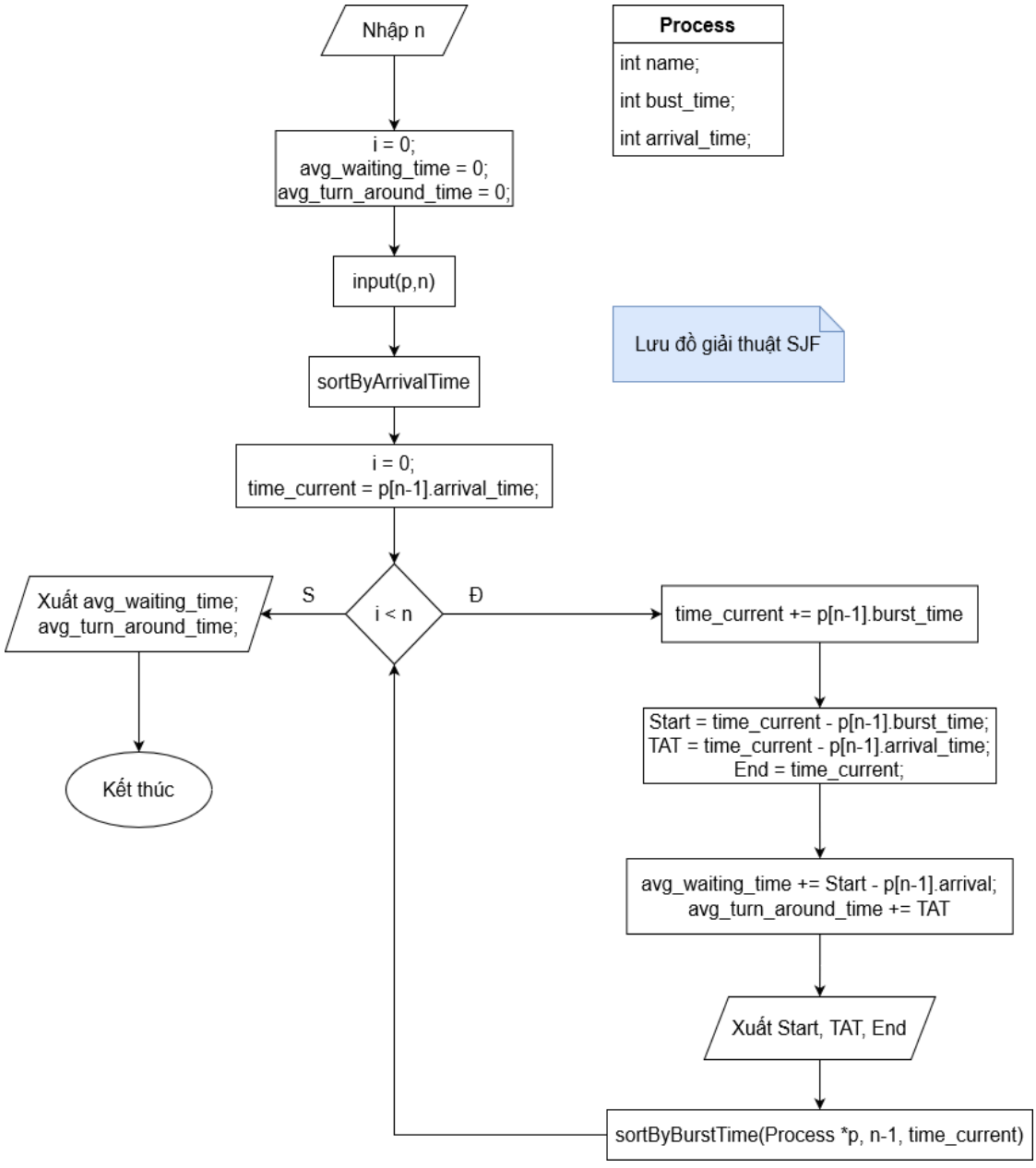
Hàm sort các tiến trình theo burst time



Hình 2: Lưu đồ hàm sort các tiến trình dựa vào 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.

1.3. Lưu đồ giải thuật SJF



Hình 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 Star, TAT, End.
  - ắp xếp lại các tiến trình còn lại dựa vào hàm sortByBurstTime và lặp lại đối với các tiến trình còn lại.

```cpp
#include <stdio.h>
#include <iostream>
#include <queue>
#include <vector>
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 = 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 << "-----Process " << i << " -----\n";
        cout << "Nhap ID process: "; cin >> p[i].name;
        cout << "Nhap Arrival Time: "; cin >> p[i].arrival_time;
        cout << "Nhap Burst Time: "; cin >> p[i].burst_time;
    }
}

void sortByBurstTime(Process *p, int n, int time_current){
    for(int i = 0; i < n; 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]);
            }
        }
    }
}
```

Hình 4: Code từ dòng 1 – 49

```
50
51 void SelectionFunction(Process *p, int n){
52     int time_current;
53     sortByArrivalTime(p, n);
54     time_current = p[0].arrival_time;
55     for(int i = 0; i < n; i++){
56         time_current += p[i].burst_time;
57         avg_waiting_time += time_current - p[i].arrival_time - p[i].burst_time;
58         avg_turn_around_time += (time_current - p[i].arrival_time);
59         cout << "Process " << p[i].name << " \tArrival Time " << p[i].arrival_time << " \tBurst Time " << p[i].burst_time << " \tStart " << time_current -
   p[i].burst_time << " \tTAT " << time_current - p[i].arrival_time << " \tFinish " << time_current << endl;
60     }
61     sortByBurstTime(p, n, time_current);
62 }
63
64 int main(){
65     Process *p = new Process[100];
66     int n;
67     cout << "nhap so luong process: "; cin >> n;
68     Input(p, n);
69     cout << "Name\tArrtime\tBurttime\tStart\tTAT\tFinish\n";
70     SelectionFunction(p, n);
71     cout << "Thoi gian dap ung trung binh: " << avg_waiting_time / n << " end";
72     cout << "Thoi gian hoan thanh trung binh: " << avg_turn_around_time / n << " end";
73     delete[] p; // Giải phóng bộ nhớ
74     return 0;
75 }
```

Hình 5: Code từ dòng 50 – 75

1.5. Test case 1

– Ví dụ 1:

Process 1	Arrival Time	Burst Time
P1	0	9
P2	4	5
P3	2	7
P4	8	10
P5	10	13

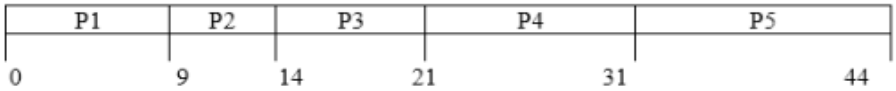
- Kết quả khi chạy code:

```
pentakll4002@123123:~$ ./sjf
nhap so luong process: 5
-----Process 0 -----
Nhap ID process: 1
Nhap Arrival Time: 0
Nhap Burst Time: 9
-----Process 1 -----
Nhap ID process: 2
Nhap Arrival Time: 4
Nhap Burst Time: 5
-----Process 2 -----
Nhap ID process: 3
Nhap Arrival Time: 2
Nhap Burst Time: 7
-----Process 3 -----
Nhap ID process: 4
Nhap Arrival Time: 8
Nhap Burst Time: 10
-----Process 4 -----
Nhap ID process: 5
Nhap Arrival Time: 10
Nhap Burst Time: 13
Name    Arrtime Burttime        Start   TAT     Finish
Process 1       Arrival Time 0  Burst Time 9    Start 0         TAT 9   Finish 9
Process 3       Arrival Time 2  Burst Time 7    Start 9         TAT 14  Finish 16
Process 2       Arrival Time 4  Burst Time 5    Start 16        TAT 17  Finish 21
Process 4       Arrival Time 8  Burst Time 10   Start 21        TAT 23  Finish 31
Process 5       Arrival Time 10         Burst Time 13   Start 31        TAT 34  Finish 44
Thoi gian dap ung trung binh: 10.6 endThoi gian hoan thanh trung binh: 19.4 endpentakll4002@123123:~
$
```

Hình 5: Kết quả khi giải ví dụ 1 bằng code giải thuật SJF

- Kết quả khi giải tay:

+ Giản đồ Gantt:



+ Thời gian đáp ứng:

P1 = 0, P2 = 5, P3 = 12, P4 =13, P5 = 21

⇨ Thời gian đáp ứng trung bình: (0 + 5 + 12 + 13 + 21) / 5 = 10.2

+ Thời gian đợi:

P1 = 0, P2 = 5, P3 = 12, P4 =13, P5 = 21

⇨ Thời gian đợi trung bình: (0 + 5 + 12 + 13 + 21) / 5 = 10.2

+ Thời gian hoàn thành:

P1 = 9, P2 = 10, P3 = 19, P4 = 23, P5 = 34

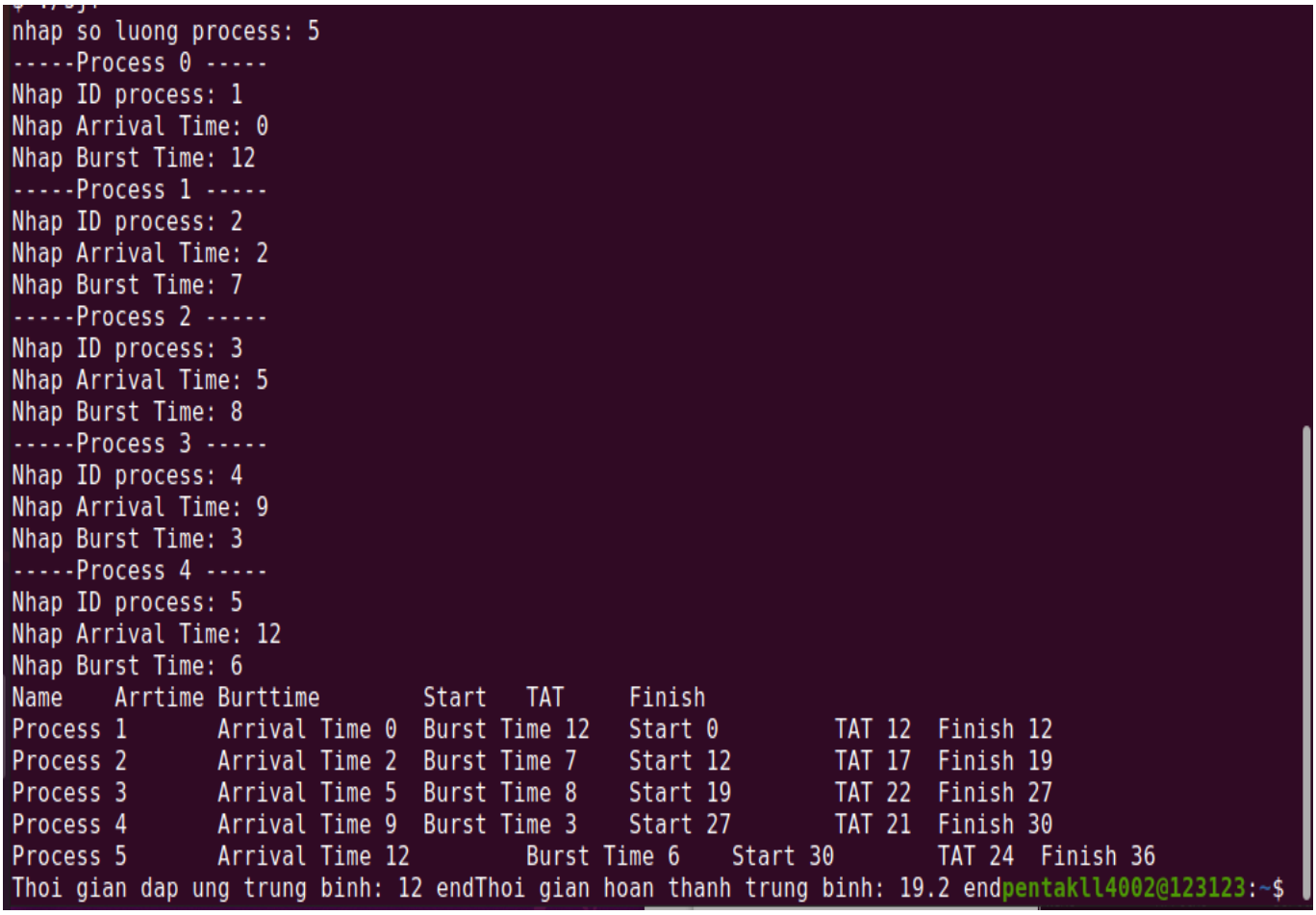
⇨ Thời gian hoàn thành trung bình: (9 + 10 + 19 + 23 + 34) / 5 = 19

Hình 6:Kết quả khi giải tay ví dụ 1 bằng giải thuật SJF

– Ví dụ 2:

Process	Arriva Time	Burst Time
P1	0	12
P2	2	7
P3	5	8
P4	9	3
P5	12	6

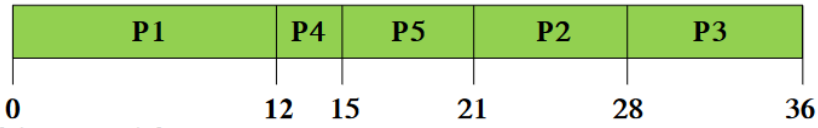
- Kết quả khi chạy code



Hình 7: Kết quả khi giải ví dụ 2 bằng code giải thuật SJF

- Kết quả khi giải tay

- Giản đồ Gantt



- Thời gian chờ:
  - P1 = 0, P2 = 19, P3 = 23, P4 = 3, P5 = 3
  - Thời gian chờ trung bình: (0 + 19 + 23 + 3 + 3)/5 = 9.6
- Thời gian đáp ứng:
  - P1 = 0, P2 = 19, P3 = 23, P4 = 3, P5 = 3
  - Thời gian đáp ứng trung bình: (0 + 19 + 23 + 3 + 3)/5 = 9.6
- Thời gian hoàn thành:
  - P1 = 12, P2 = 26, P3 = 31, P4 = 6, P5 = 9
  - Thời gian hoàn thành trung bình: (12 + 26 + 31 + 6 + 9)/5 = 16.8

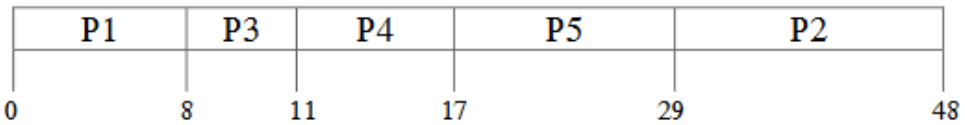
Hình 8:Kết quả khi giải tay ví dụ 2 bằng giải thuật SJF

– Ví dụ 3:

Process	Arriva Time	Burst Time
P1	0	8
P2	2	19
P3	4	3
P4	5	6
P5	7	12

- Kết quả khi giải tay

+ Giản đồ Gantt:



+ Thời gian đáp ứng trung bình là: 9.4

+ Thời gian hoàn thành trung bình: 19.

Hình 9: Kết quả khi giải tay ví dụ 3 bằng giải thuật SJF

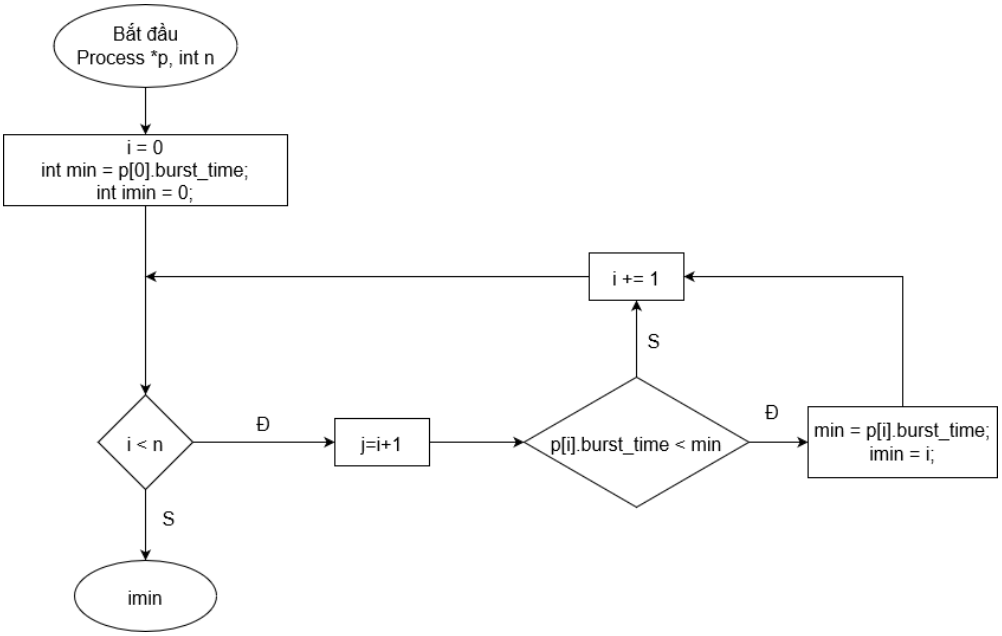
- Kết quả khi chạy code

```
pentakll4002@123123:~$ ./sjf
nhap so luong process: 5
-----Process 0 -----
Nhap ID process: 1
Nhap Arrival Time: 0
Nhap Burst Time: 8
-----Process 1 -----
Nhap ID process: 2
Nhap Arrival Time: 2
Nhap Burst Time: 19
-----Process 2 -----
Nhap ID process: 3
Nhap Arrival Time: 4
Nhap Burst Time: 3
-----Process 3 -----
Nhap ID process: 4
Nhap Arrival Time: 5
Nhap Burst Time: 6
-----Process 4 -----
Nhap ID process: 5
Nhap Arrival Time: 7
Nhap Burst Time: 12
Name    Arrtime Burttime        Start   TAT     Finish
Process 1       Arrival Time 0  Burst Time 8    Start 0         TAT 8   Finish 8
Process 2       Arrival Time 2  Burst Time 19   Start 8         TAT 25  Finish 27
Process 3       Arrival Time 4  Burst Time 3    Start 27        TAT 26  Finish 30
Process 4       Arrival Time 5  Burst Time 6    Start 30        TAT 31  Finish 36
Process 5       Arrival Time 7  Burst Time 12   Start 36        TAT 41  Finish 48
Thoi gian dap ung trung binh: 16.6 endThoi gian hoan thanh trung binh: 26.2 endpentakll4002@123123:~
$
```

Hình 10: Kết quả khi giải ví dụ 3 bằng code giải thuật SJF

Task name 2: Viết chương trình mô phỏng giải thuật SRT.

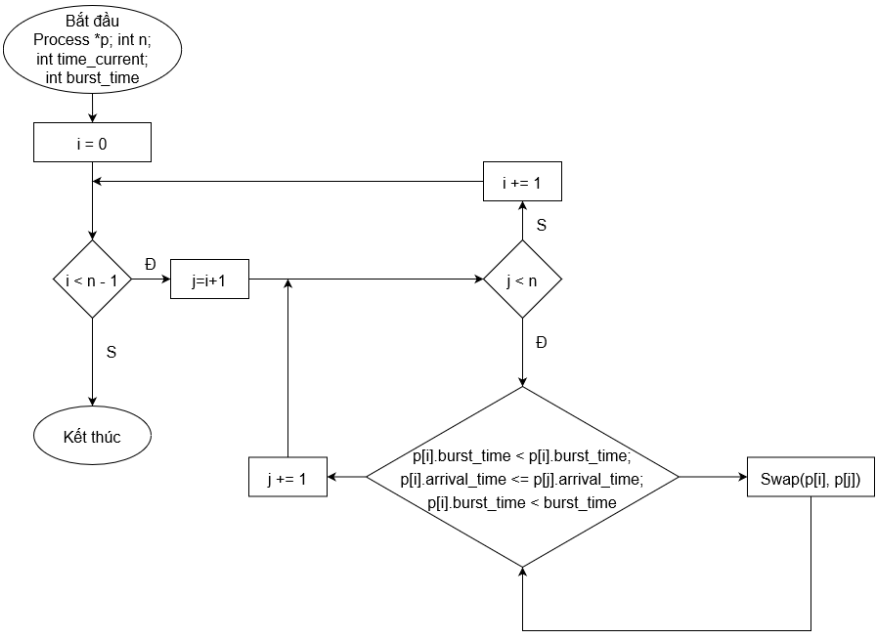
2.1. Hàm tìm ra tiến trình có burst time nhỏ nhất.



Hình 11: Lưu đồ hàm minBurstTime

- **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.

2.2. Hàm sort các tiến tiền dựa theo tiến trình có burst_time nhỏ hơn burst của tiến trình đang thực thi.

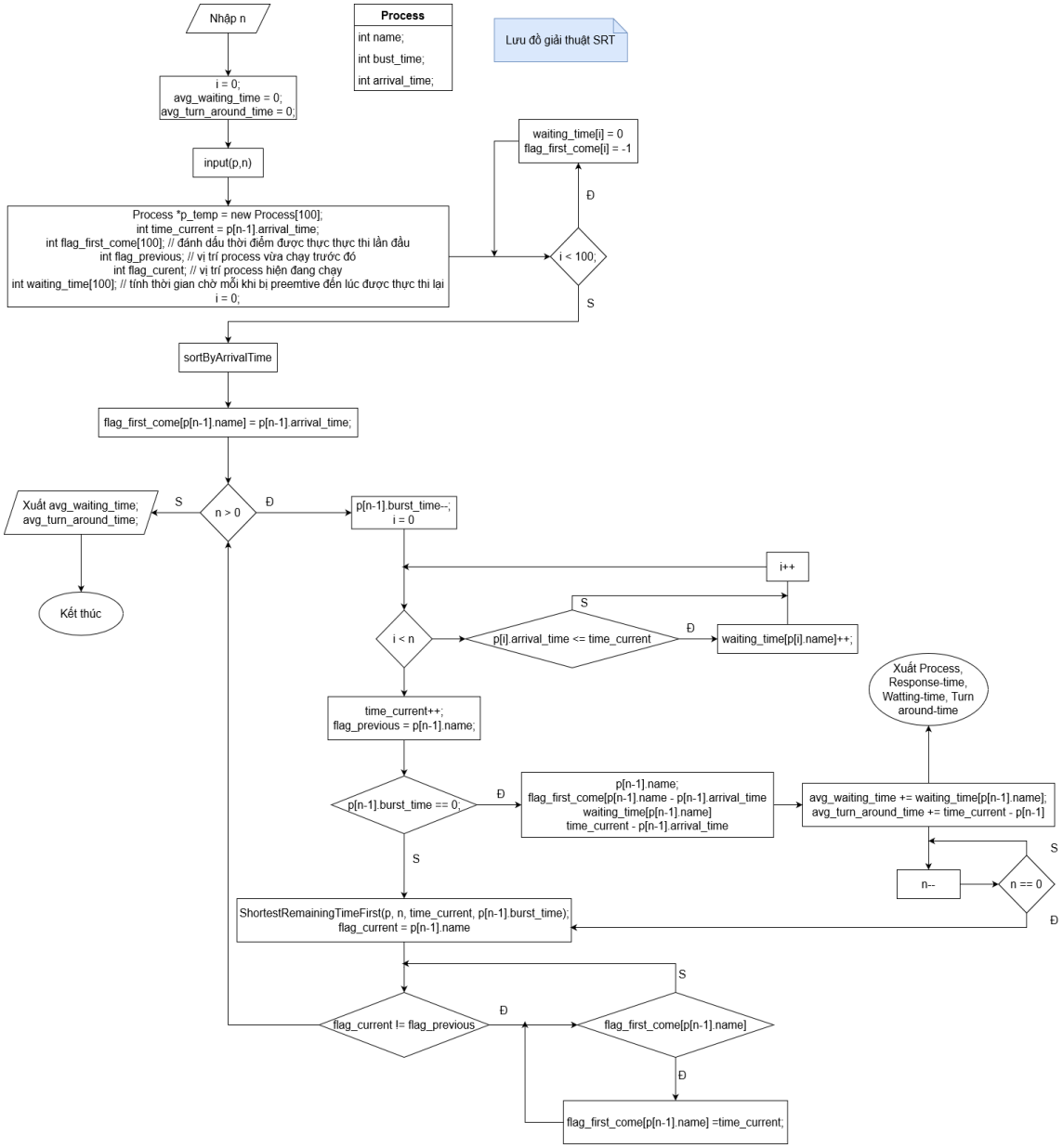


Hình 12: Hàm ShortestRemainingTimeFirst.

- **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.

2.3. Lưu đồ giải thuật SRTF

- **Giải thích:**
  - 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.



Hình 13: Lưu đồ giải thuật SRTF

2.4. Code giải thuật SRTF

```cpp
/*####################
University of Information Technology
IT007.P110.2
Dang Thien An, 23520003
File: srtf.cpp
####################*/

#include <iostream>
#include <iomanip>
#include <algorithm>
#include <vector>
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;
    bool is_completed; // Track completion status
};

float avg_turnaround_time = 0;
float avg_waiting_time = 0;
float avg_response_time = 0;

void findCompletionTime(vector<Process> &p, int n) {
    int current_time = 0;
    int completed = 0;

    while(completed != n) {
        int idx = -1;
        int mn = 10000000;

        for(int i = 0; i < n; i++) {
            if(p[i].arrival_time <= current_time && !p[i].is_completed) {
                if(p[i].burst_time < mn) {
                    mn = p[i].burst_time;
                    idx = i;
                }
            }
        }

        if(idx != -1) {
            if(p[idx].arrival_time > current_time) {
                current_time = p[idx].arrival_time;
            }
            p[idx].start_time = current_time;
```

Hình 14: Code giải thuật SRT từ dòng 1 - 51

```
51              p[idx].start_time = current_time;
52              current_time += p[idx].burst_time;
53              p[idx].completion_time = current_time;
54              p[idx].turnaround_time = p[idx].completion_time - p[idx].arrival_time;
55              p[idx].waiting_time = p[idx].turnaround_time - p[idx].burst_time;
56              p[idx].response_time = p[idx].start_time - p[idx].arrival_time;
57
58              avg_turnaround_time += p[idx].turnaround_time;
59              avg_waiting_time += p[idx].waiting_time;
60              avg_response_time += p[idx].response_time;
61
62              p[idx].is_completed = true; // Mark process as completed
63              completed++;
64          } else {
65              current_time++; // No process is ready, increment time
66          }
67      }
68
69      avg_turnaround_time /= n; // Calculate average
70      avg_waiting_time /= n; // Calculate average
71      avg_response_time /= n; // Calculate average
72  }
73
74  int main() {
75      int n;
76      cout << "Enter the number of processes: ";
77      cin >> n;
78
79      vector<Process> p(n);
80
81      cout << "Enter the arrival time and burst time of each process:\n";
82      for(int i = 0; i < n; i++) {
83          cout << "P" << i + 1 << ": ";
84          cin >> p[i].arrival_time >> p[i].burst_time;
85          p[i].pid = i + 1;
86          p[i].is_completed = false; // Initialize completion status
87      }
88
89      findCompletionTime(p, n);
90
91      cout << "#PName\tAT\tBT\tST\tCT\tTAT\tWT\tRT\n";
92      for(int i = 0; i < n; i++) {
93          cout << "P" << p[i].pid << "\t" << p[i].arrival_time << "\t"
94               << p[i].burst_time << "\t" << p[i].start_time << "\t"
95               << p[i].completion_time << "\t" << p[i].turnaround_time << "\t"
96               << p[i].waiting_time << "\t" << p[i].response_time << "\n";
97      }
98
99      cout << "Average Turnaround Time: " << avg_turnaround_time << endl;
100     cout << "Average Waiting Time: " << avg_waiting_time << endl;
101     cout << "Average Response Time: " << avg_response_time << endl;
102 }
```

Hình 16: Code giải thuật SRT từ dòng 51 – 102

2.5. Test case

– Ví dụ 1:

Process 1	Arrival Time	Burst Time
P1	0	12
P2	2	7
P3	5	8
P4	9	3
P5	12	6

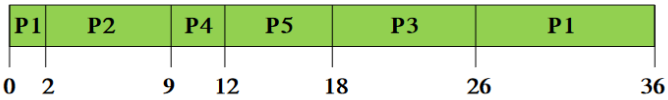
- Kết quả khi chạy code:

```
pentakll4002@123123:~$ gedit srtf.cpp
pentakll4002@123123:~$ g++ -o srtf srtf.cpp
pentakll4002@123123:~$ ./srtf
Enter the number of processes: 5
Enter the arrival time and burst time of each process:
P1: 0
12
P2: 2 7
P3: 5 8
P4: 9 3
P5: 12 6
#PName  AT      BT      ST      CT      TAT     WT      RT
P1      0       12      0       12      12      0       0
P2      2       7       21      28      26      19      19
P3      5       8       28      36      31      23      23
P4      9       3       12      15      6       3       3
P5      12      6       15      21      9       3       3
Average Turnaround Time: 16.8
Average Waiting Time: 9.6
Average Response Time: 9.6
pentakll4002@123123:~$
```

Hình 15: Kết quả khi giải ví dụ 1 bằng code giải thuật SRT

- Kết quả khi giải tay:

  - Giản đồ Gantt



  - Thời gian chờ:
    - P1 = 24, P2 = 0, P3 = 13, P4 = 0, P5 = 0
    - Thời gian chờ trung bình: (24 + 0 + 13 + 0 + 0)/5 = 7.4
  - Thời gian đáp ứng:
    - P1 = 0, P2 = 0, P3 = 13, P4 = 0, P5 = 0
    - Thời gian đáp ứng trung bình: (0 + 0 + 13 + 0 + 0)/5 = 2.6
  - Thời gian hoàn thành:
    - P1 = 36, P2 = 7, P3 = 21, P4 = 3, P5 = 6
    - Thời gian hoàn thành trung bình: (36 + 7 + 21 + 3 + 6)/5 =14.6

Hình 16: Kết quả khi giải tay ví dụ 1 bằng giải thuật SRT

– Ví dụ 2:

Process 1	Arrival Time	Burst Time
P1	0	8
P2	2	19
P3	4	3
P4	5	6
P5	7	12

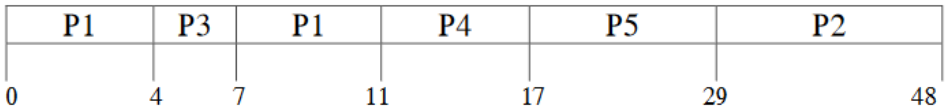
- Kết quả khi chạy code:

```
pentakll4002@123123:~$ ./srtf
Enter the number of processes: 5
Enter the arrival time and burst time of each process:
P1: 0 8
P2: 2 19
P3: 4 3
P4: 5 6
P5: 7 12
#PName  AT      BT      ST      CT      TAT     WT      RT
P1      0       8       0       8       8       0       0
P2      2       19      29      48      46      27      27
P3      4       3       8       11      7       4       4
P4      5       6       11      17      12      6       6
P5      7       12      17      29      22      10      10
Average Turnaround Time: 19
Average Waiting Time: 9.4
Average Response Time: 9.4
pentakll4002@123123:~$
```

Hình 17: Kết quả khi giải ví dụ 2 bằng code giải thuật SRT

- Kết quả khi giải tay:

+ Giản đồ Gantt:



+ Thời gian đáp ứng trung bình là: 9.2.

+ Thời gian hoàn thành trung bình: 18.8.

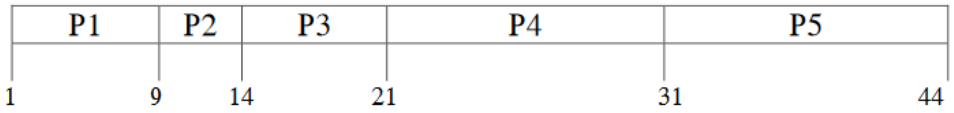
Hình 18: Kết quả khi giải tay ví dụ 2 bằng giải thuật SRT

– Ví dụ 3:

Process 1	Arrival Time	Burst Time
P1	0	9
P2	4	5
P3	2	7
P4	8	10
P5	10	13

- Kết quả khi giải tay:

+ Giản đồ Gantt:



+ Thời gian đáp ứng trung bình là: 10.2

+ Thời gian hoàn thành trung bình: 19

Hình 19: Kết quả khi giải tay ví dụ 3 bằng giải thuật SRT

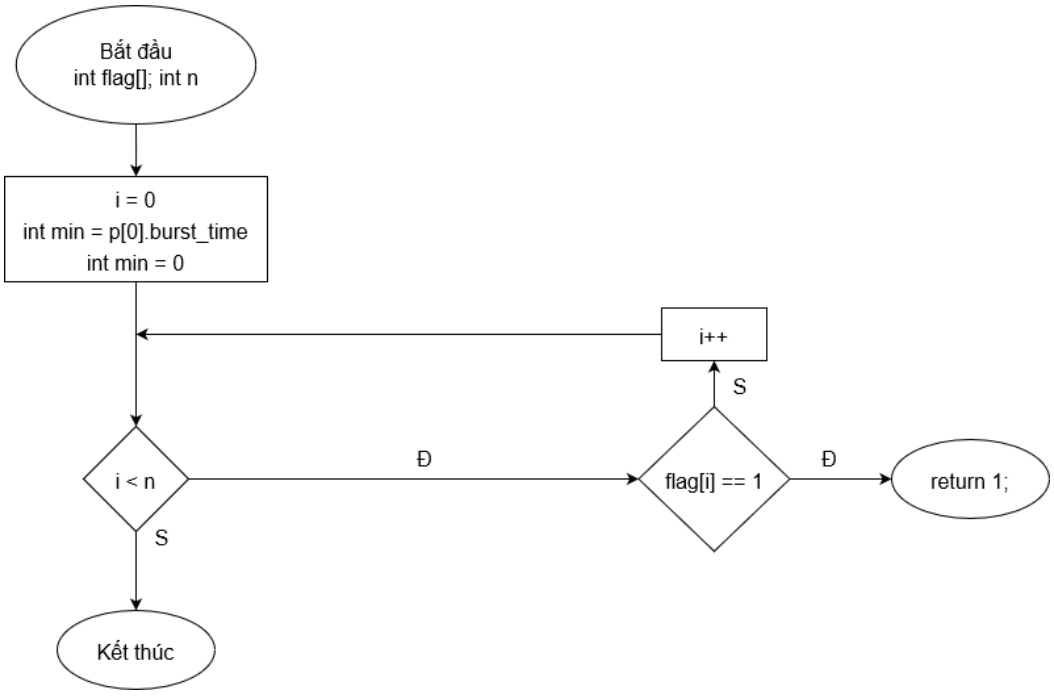
- Kết quả khi chạy code:

```
pentakll4002@123123:~$ ./srtf
Enter the number of processes: 5
Enter the arrival time and burst time of each process:
P1: 0 9
P2: 4 5
P3: 2 7
P4: 8 10
P5: 10 13
#PName  AT      BT      ST      CT      TAT     WT      RT
P1      0       9       0       9       9       0       0
P2      4       5       9       14      10      5       5
P3      2       7       14      21      19      12      12
P4      8       10      21      31      23      13      13
P5      10      13      31      44      34      21      21
Average Turnaround Time: 19
Average Waiting Time: 10.2
Average Response Time: 10.2
pentakll4002@123123:~$
```

Hình 20: Kết quả khi giải ví dụ 3 bằng code giải thuật SRT

3. Task name 3

3.1.Hàm kiểm tra



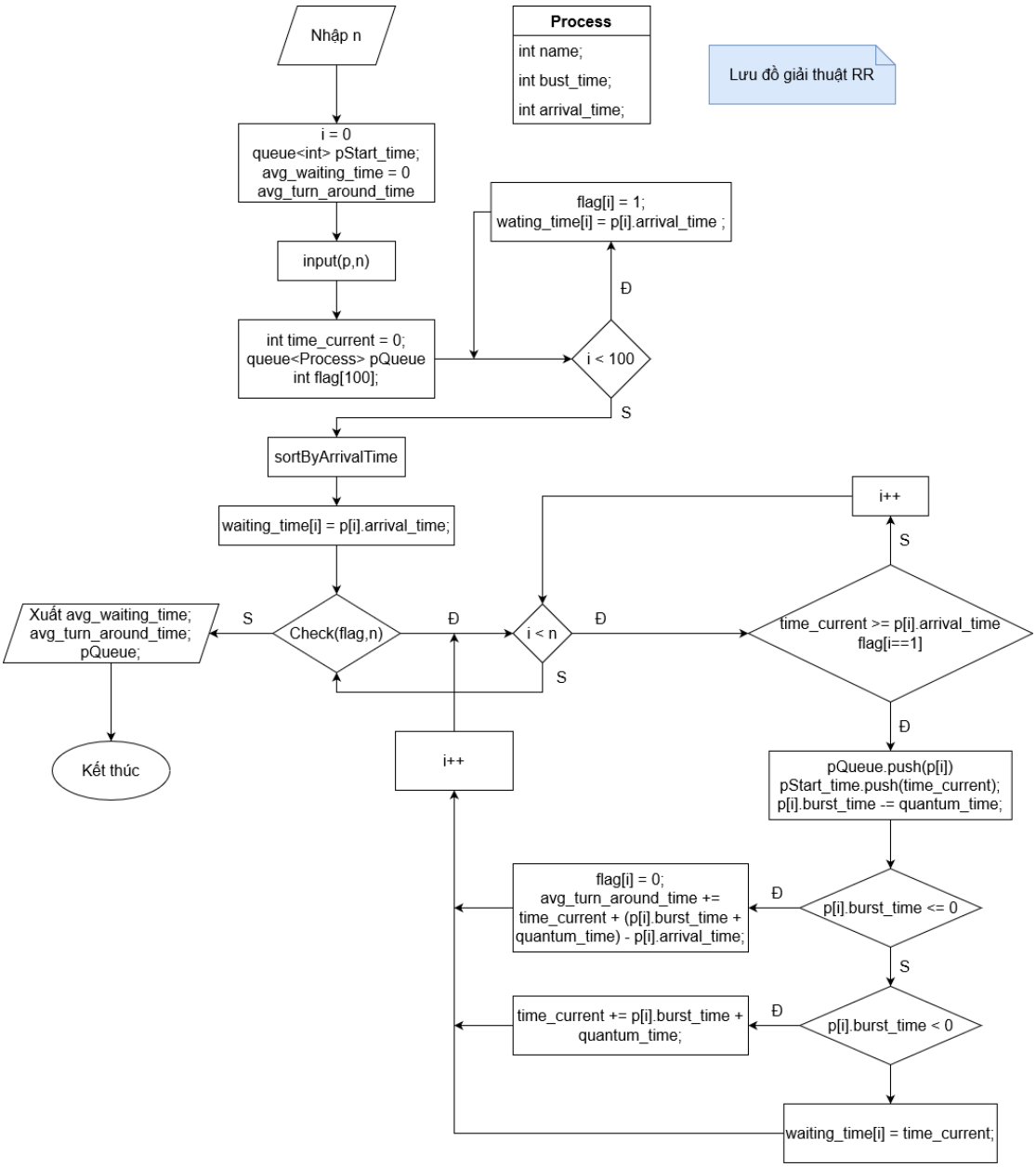
Hình 21: Hàm kiểm tra

– **Giải thích:** Hàm check có chức năng kiểm tra lại có phần tử nào trong đó bằng 1 hay ko. Nếu có thì sẽ return ra 1, còn tát cả đều bằng 0 thì return ra 0.

Giải thuật RR

- **Giải thích:**
  - Tương tự như các giải thuật trên ta sẽ tiến hành tạo struct và tiến hành nhập các thông tin. Và nhập quantumtime.
  - Tạo list flag có tác dụng kiểm tra xem busrt_time của tiến trình còn hay không. Ban đầu ta sẽ gắn hết bằng 1. Và waiting_time sẽ bằng thời gian đến.
  - Tiến hành sort theo arrival time. Và tạo ra 1 list các tiến trình rỗng khác
  - Dùng hàm check để kiểm tra xem còn tiến trình nào vẫn còn burst_time hay không.
  - Duyệt qua lần lượt tất cả các process. Tìn trình nào đã đến và còn burst time thì được xét. Ta sẽ tạo bản sao và được gắn vào list qQueue và các thông tin khác sẽ được lưu. Sau đó dảm burstTime đi với số lượng = quantum_time

  - Còn nếu burst_time == 0 thì cho flag[i] =0 và tính thời gian đợi và thười gian hoàng thành. Nếu bursttime < 0 thì thời timeline lúc này sẽ được cộng thêm burst_time và quantum_time. Các trường hợp khác thì time_current là timeline lúc này được cộng thêm quantum_time.
  - Tiến hành với các tiến trình khác và kiểm tra day flag còn phần tử nào bằng 1 hay không. Nếu hết rỏi thì xuất ra qQueue. Từ đó sẽ lấy được thông tin cụ thể.



Hình 22: Lưu đồ giải thuật RR

3.3. Code giải thuật RR

```
queue<Process> SelectionFunction(Process *p, int n, int quantum_time) {
    int time_current = 0;
    int flag_c = 1;
    queue<Process> pQueue;
    int flag[100];
    sortByArrivalTime(p, n);
    for (int i = 0; i < n; i++) {
        flag[i] = 1;
        waiting_time[i] = p[i].arrival_time;
    }
    while (check(flag, n)) {
        // Duyệt qua hết 1 lượt các process
        for (int i = 0; i < n; i++) {
            // Process nào đã đến và còn burst time mới được xét
            if (time_current >= p[i].arrival_time && flag[i] == 1) {
                if (flag_c == 1) {
                    time_current = p[i].arrival_time;
                    flag_c = 0;
                }
                pstart_time.push(time_current);
                p[i].burst_time -= quantum_time;
                if (p[i].burst_time <= 0) {
                    flag[i] = 0;
                    ave_turnaround_time += time_current + p[i].burst_time + quantum_time - p[i].arrival_time;
                }
                waiting_time[i] += (time_current - waiting_time[i]);
                if (p[i].burst_time < 0) {
                    time_current += p[i].burst_time + quantum_time;
                } else {
                    time_current += quantum_time;
                }
            } else {
                time_current += quantum_time;
            }
            waiting_time[i] = time_current;
        }
    }
    return pQueue;
}
```

Hình 23: Code giải thuật RR

3.4. Test case