|  |
| --- |
| Name: Phạm Đức Thể  ID: 19522253  Class: IT007.M14.2 |

OPERATING SYSTEM  
LAB 4 REPORT

**SUMMARY**

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | | **Status** | **Page** |
| Section 4.5 | Ex 1 | Hoàn thành | 2 |
| Ex 2 | Hoàn thành | 9 |
| Ex 3 | Hoàn thành | 19 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |  |
|  |  |  |

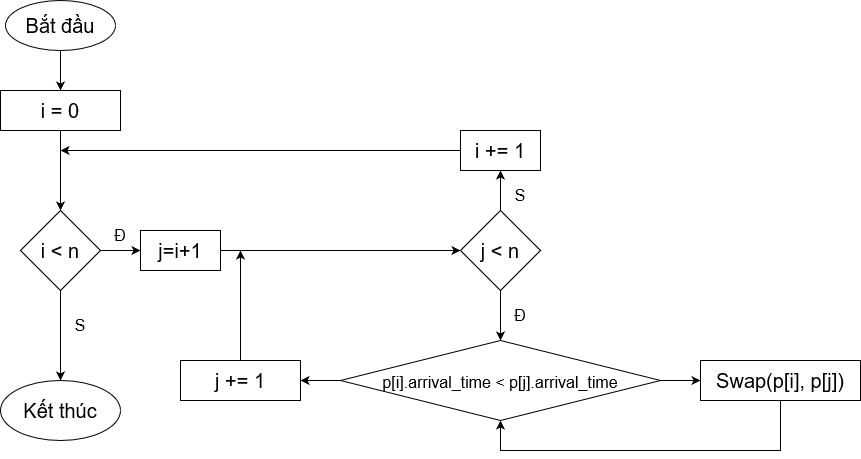
**Self-scrores: 10/10**

*\*Note: Export file to* ***PDF*** *and name the file by following format:* ***LAB X – <Student ID>.pdf***

# Section 4.5

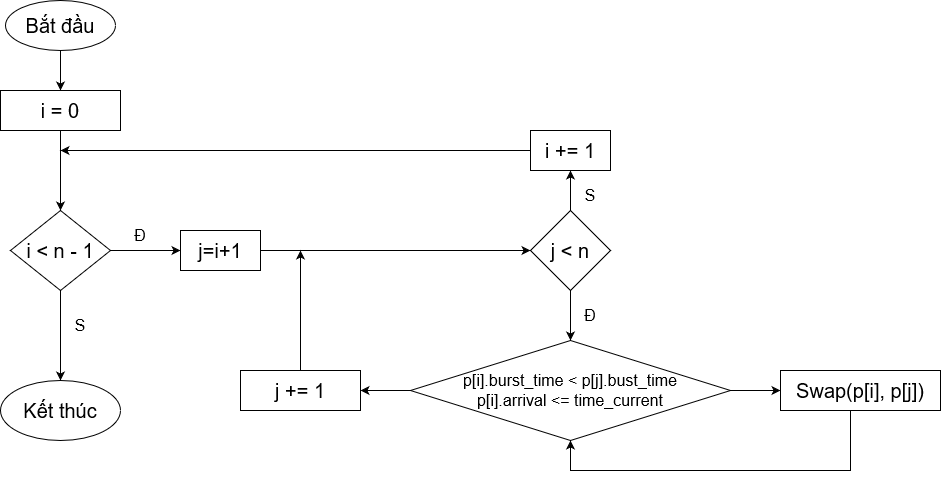
## Task name 1: Viết chương trình mô phỏng giải thuật SJF.

* 1. **Hàm sort các tiến trình theo arrival time**



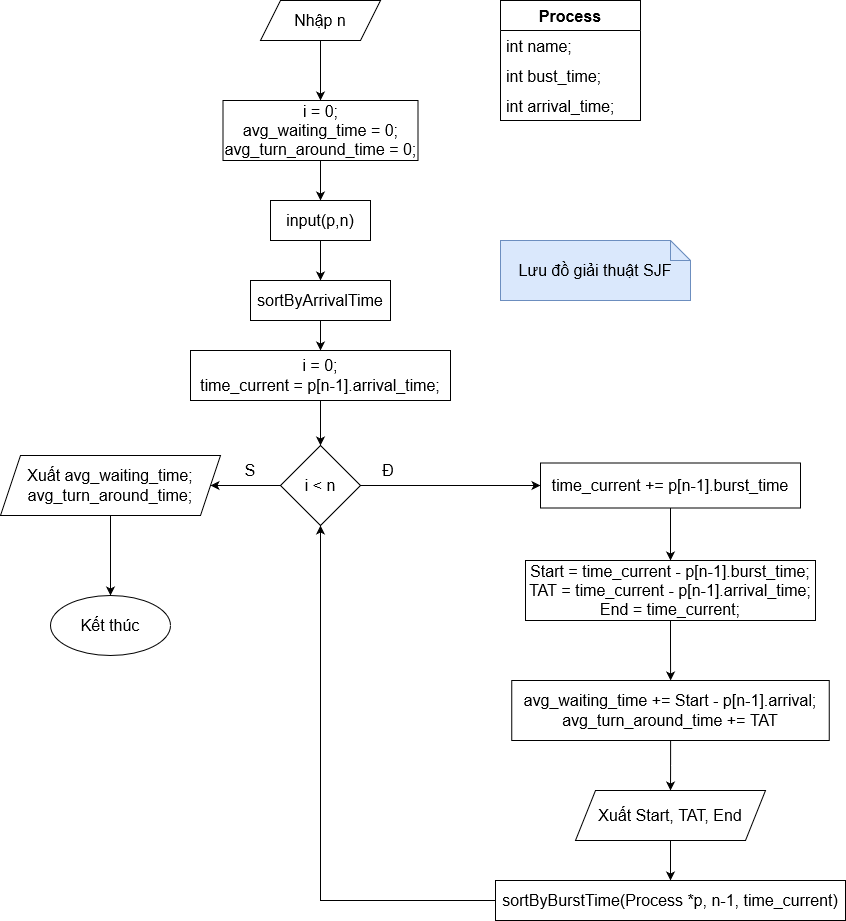
Hình 1: Lưu đồ hàm sort các tiến trình dựa vào arrival\_time

* + **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.
  1. **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. **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.
  1. **Code của giải thuật**

Text

Description automatically generated

Hình 4: Code từ dòng 1 - 49

Text

Description automatically generated

Hình 5: Code từ dòng 49 - 88

* 1. **Test case**
  + Ví dụ 1:

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

* + - Kết quả khi chạy code

Text

Description automatically generated

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

* + - Kết quả khi giải tay

Graphical user interface, application

Description automatically generated

Hình 7: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

Text

Description automatically generated

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

* + - Kết quả khi giải tay

A picture containing text

Description automatically generated

Hình 9: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

Diagram

Description automatically generated

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

* + - Kết quả khi chạy code

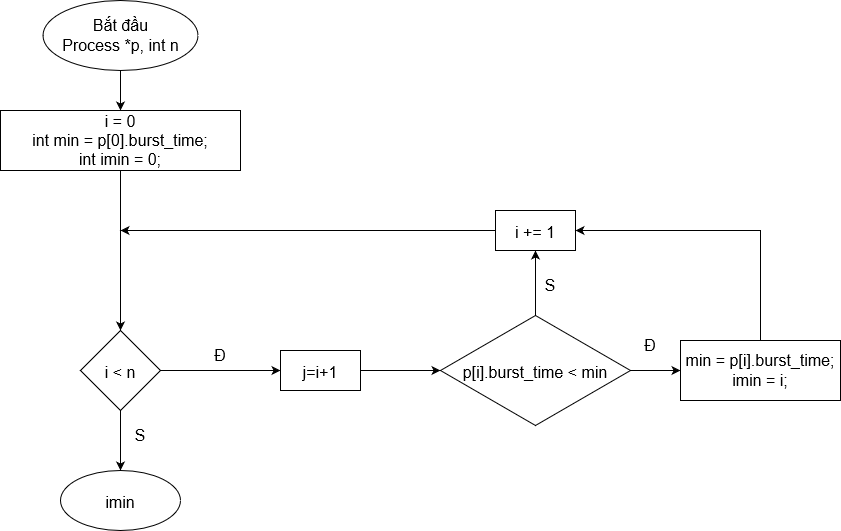
Text

Description automatically generated

Hình 11: Kết quả khi giải ví dụ 3 bằng code giải thuật SJF

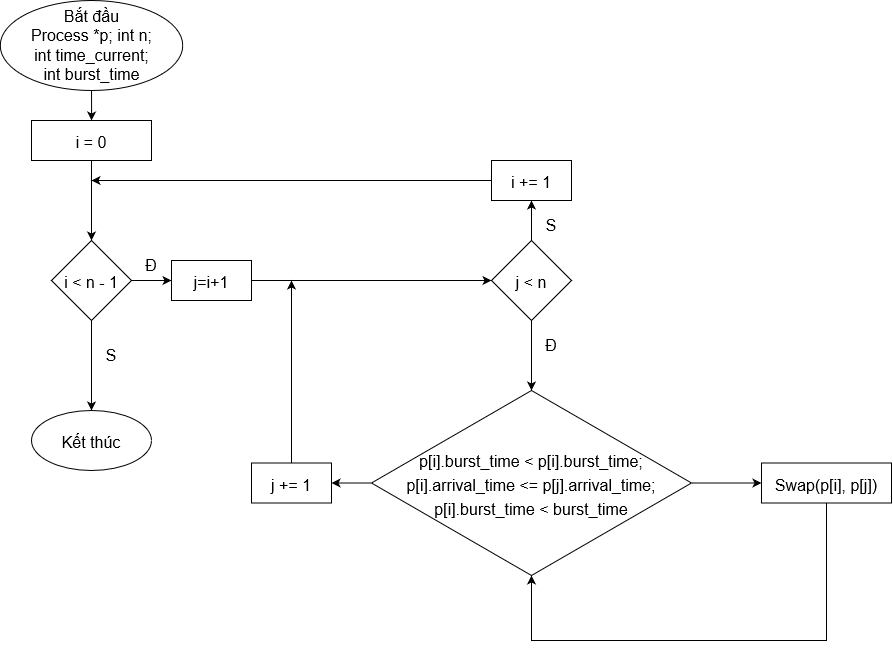
## Task name 2: Viết chương trình mô phỏng giải thuật SRT.

* 1. **Hàm tìm ra tiến trình có burst time nhỏ nhất.**



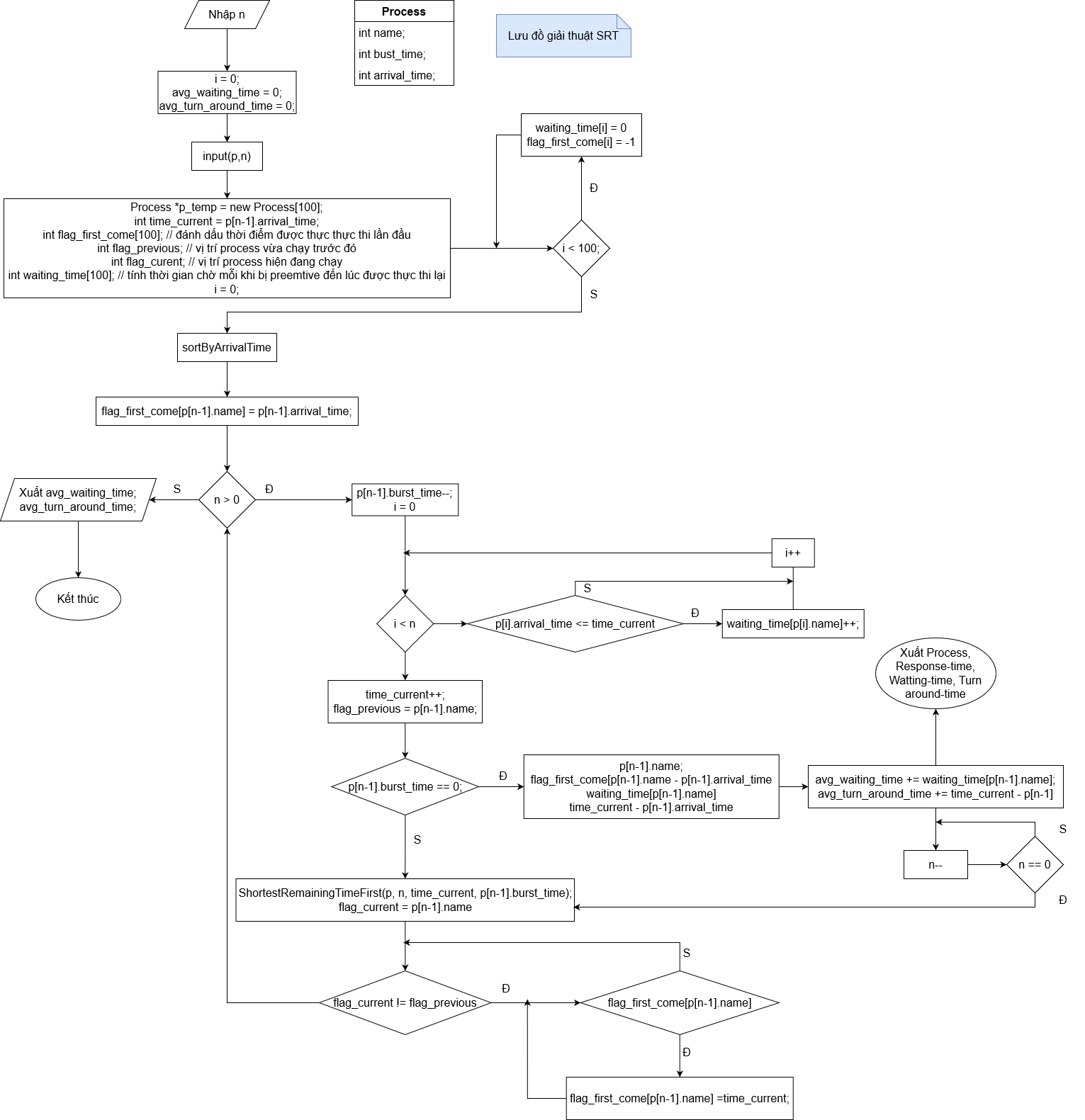
Hình 12: Lưu đồ hàm minBurstTime

* + **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.
  1. **Hàm sort các tiến tiền dựa theo tiến trình có burst\_time nhỏ hơn burst của tiến trình đang thực thi.**



Hình 13: 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.
  1. **Lưu đồ giải thuật SRT**
  + **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 14: Lưu đồ giải thuật SRT

* 1. **Code giải thuật SRT**

Text

Description automatically generated

Hình 15: Code giải thuật SRT từ dòng 1 - 50

Text

Description automatically generated

Hình 16: Code giải thuật SRT từ dòng 47 - 95

Text

Description automatically generated

Hình 17: Code giải thuật SRT từ dòng 92 - 128

* 1. **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:

Text

Description automatically generated

Hình 18: Kết quả khi giải ví dụ 1 bằng code giải thuật SRT

* + - Kết quả khi giải tay:

Table

Description automatically generated with low confidence

Hình 19: 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:

Text

Description automatically generated

Hình 20: Kết quả khi giải ví dụ 2 bằng code giải thuật SRT

* + - Kết quả khi giải tay:

Chart, box and whisker chart

Description automatically generated

Hình 21: 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:

Diagram

Description automatically generated with medium confidence

Hình 22: Kết quả khi giải tay ví dụ 3 bằng giải thuật SRT

* + - Kết quả khi chạy code

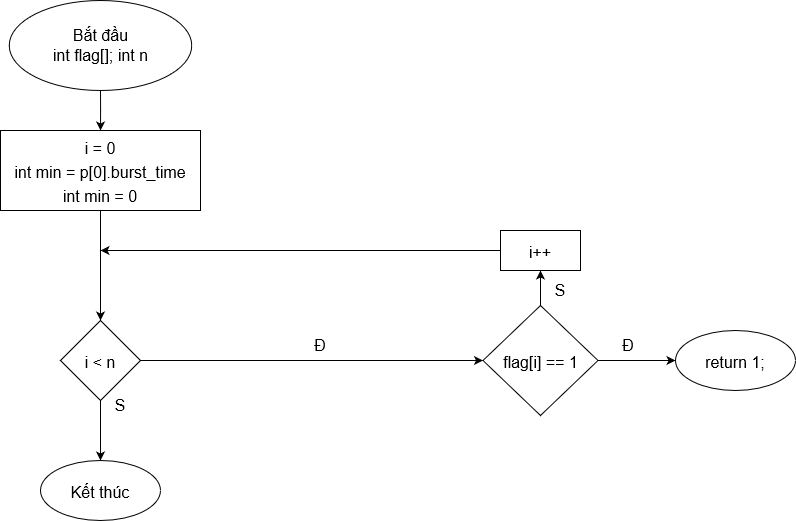
Calendar

Description automatically generated

Hình 23: Kết quả khi giải ví dụ 3 bằng code giải thuật SRT

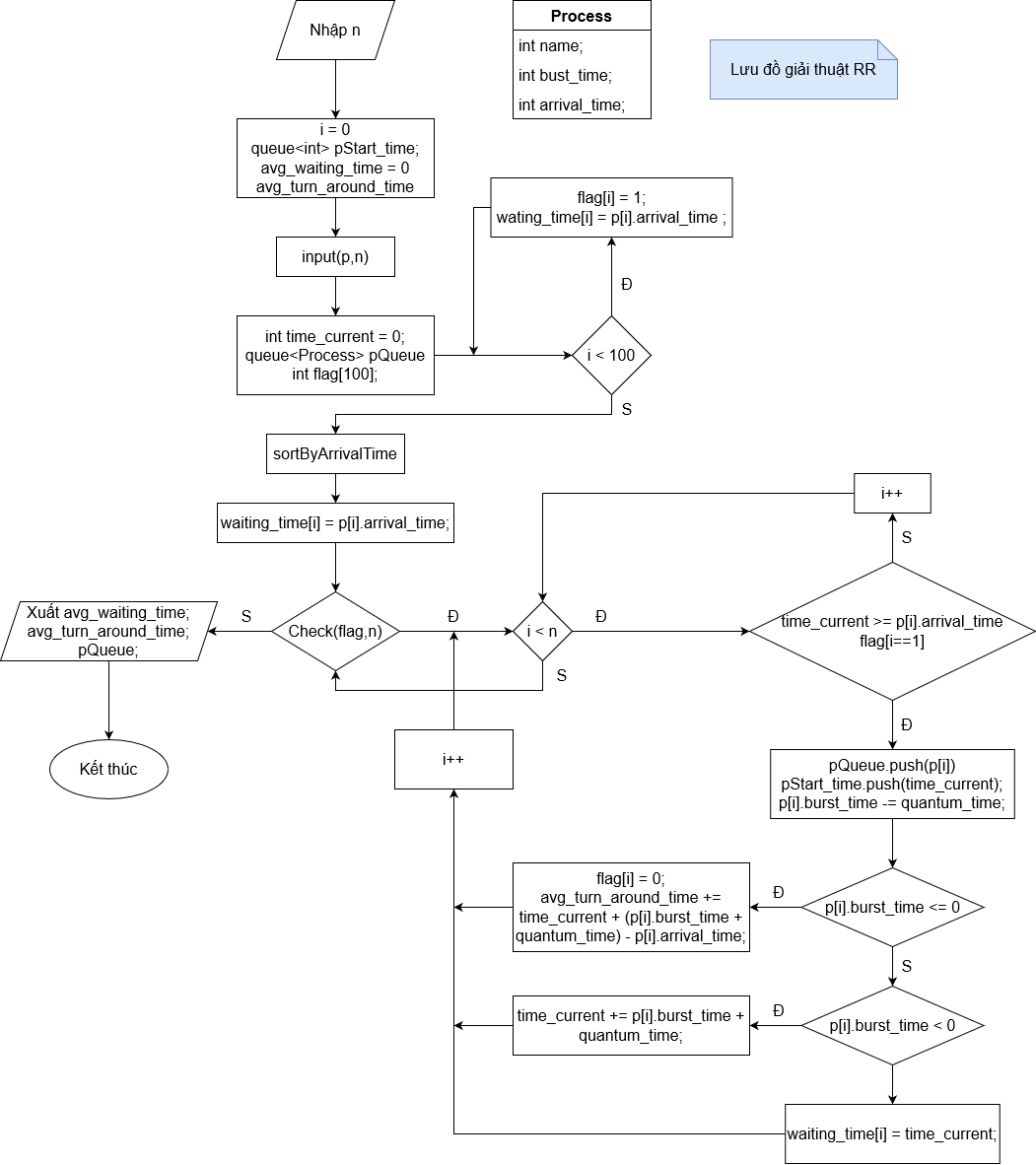
## Task name 3

* 1. **Hàm kiểm tra**

****

Hình 24: 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.
  1. **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 25: Lưu đồ giải thuật RR

* 1. **Code giải thuật RR**

Text

Description automatically generated with medium confidence

Hình 26: Code giải thuật RR

* 1. **Test case**