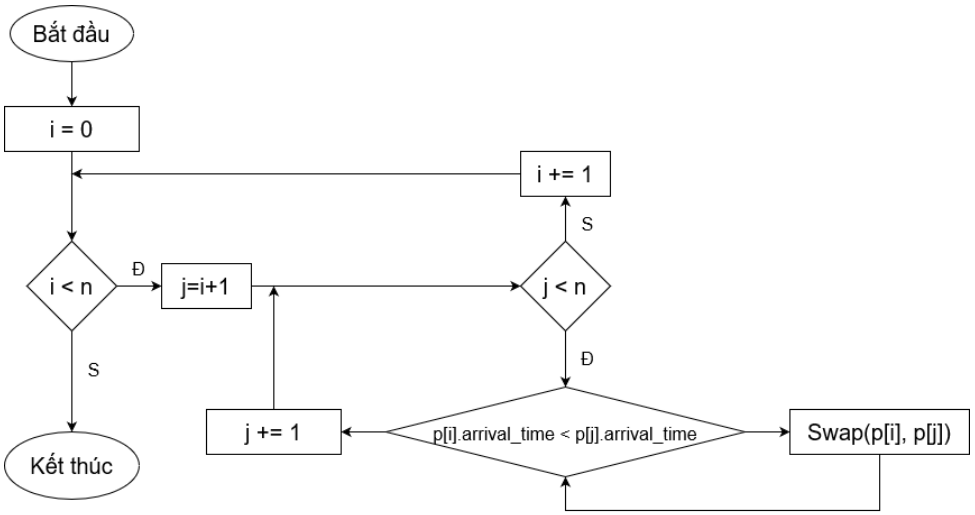
Họ và tên: Bùi Lê Nhật Tri
MSSV: 23521634
Lớp: IT00007.P11.1

BÁO CÁO LAB 4

Section 4.5

Câu 1: Viết chương trình mô phỏng giải thuật SJF.

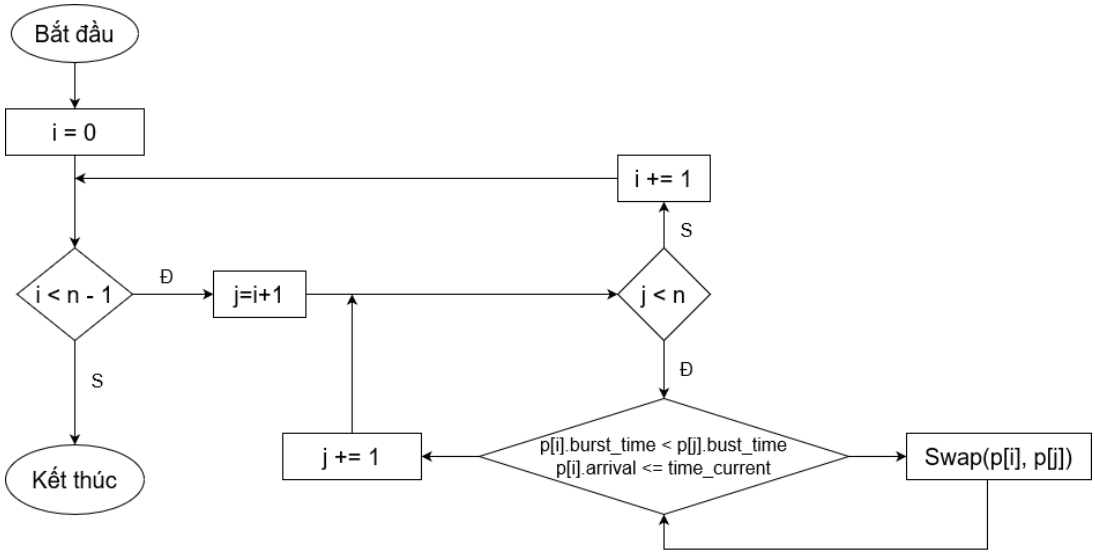
1.1. Hàm sort các tiến trình theo arrival time



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.

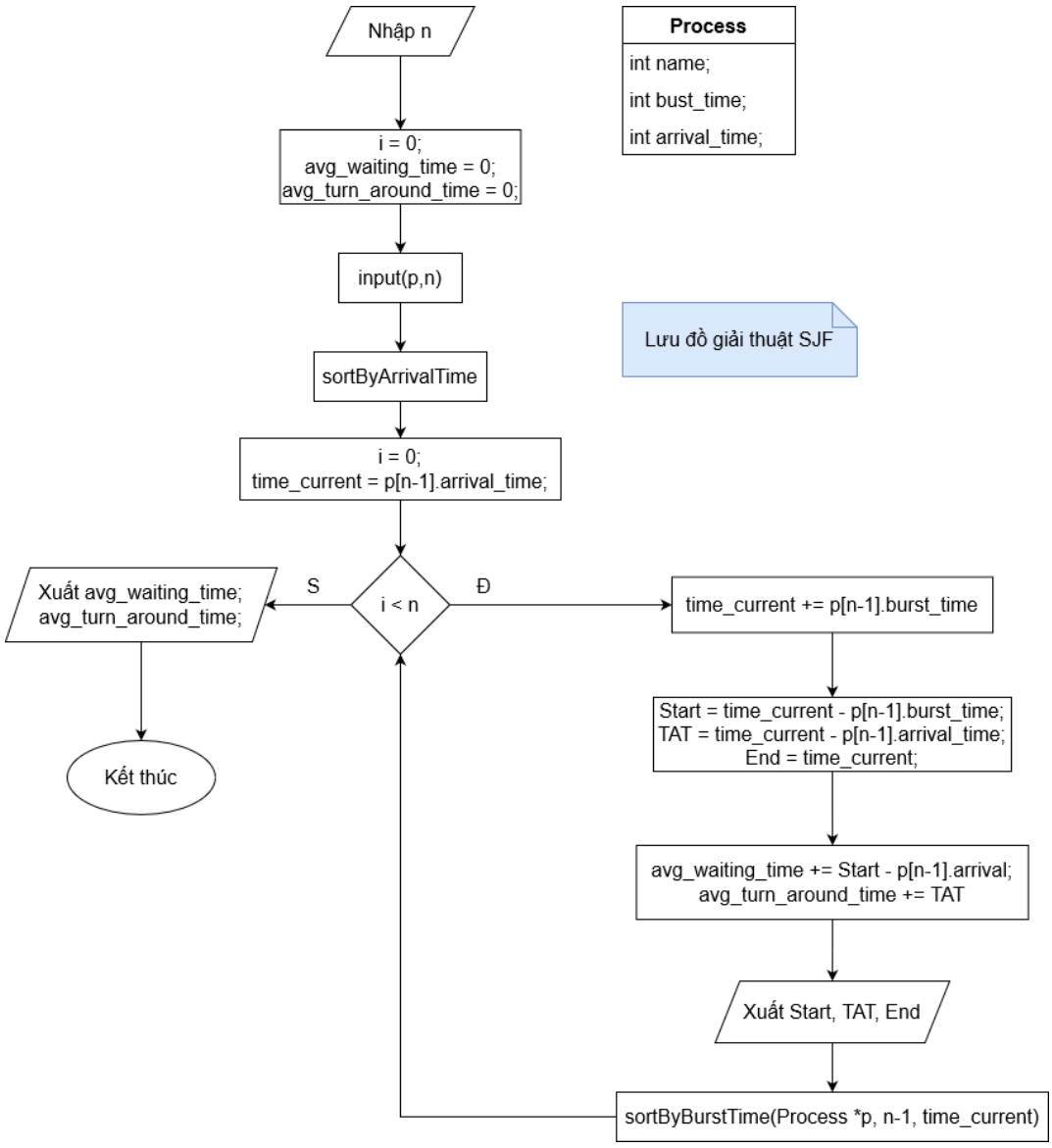
1.2. Hàm sort các tiến trình theo burst time



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



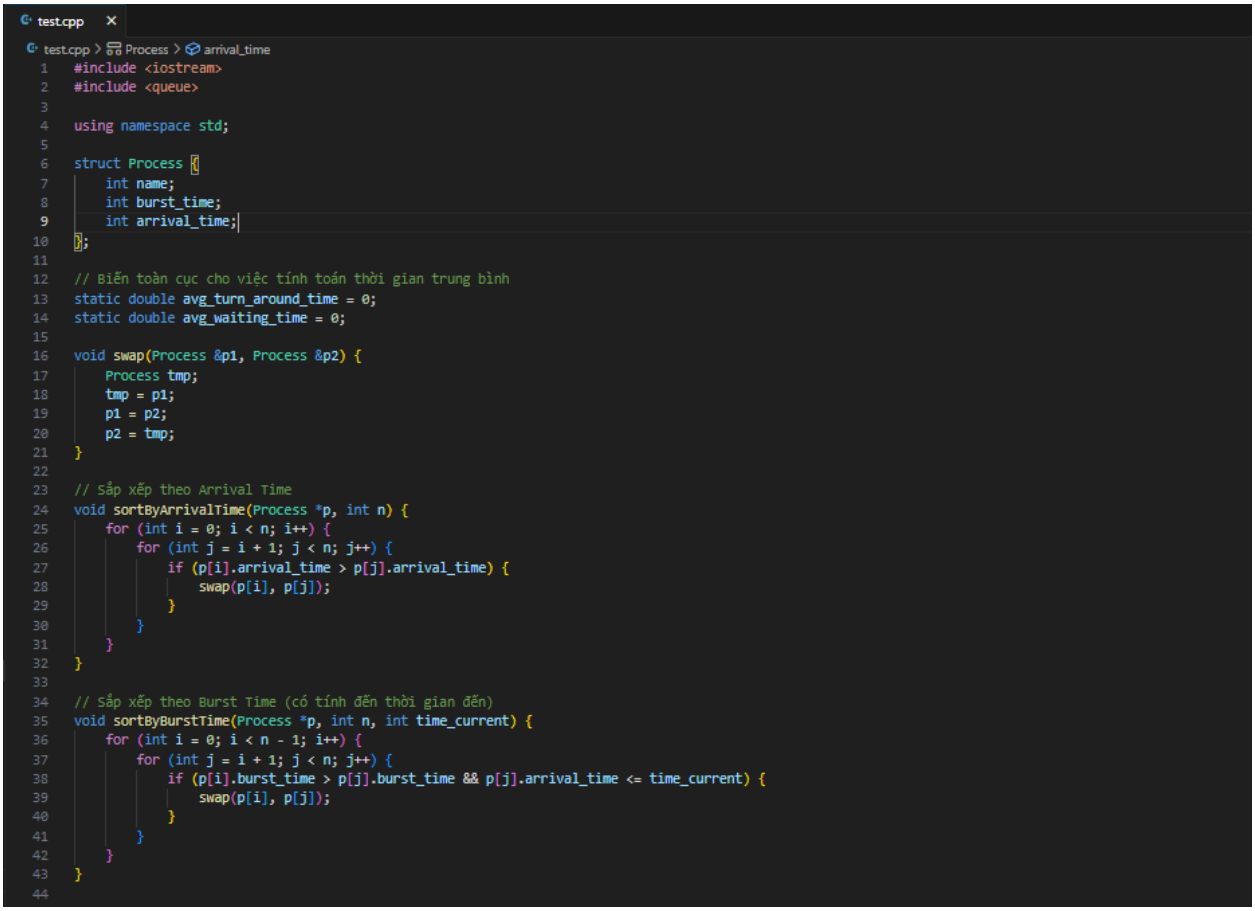
Hình 3: Lưu đồ giải thuật SJF

– **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.

1.4. Code của giải thuật



Hình 4: Code từ dòng 1 - 44

```cpp
void sortByBurstTime(Process *p, int n, int time_current) {
}

void Input(Process *p, int n) {
    for (int i = 0; i < n; i++) {
        cout << "------------------\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;
    }
}

// Hàm xử lý SJF (Non-preemptive)
void SelectionFunction(Process *p, int n) {
    int time_current = 0;
    sortByArrivalTime(p, n);  // Sắp xếp theo thời gian đến

    // Bắt đầu xử lý từng tiến trình
    for (int i = 0; i < n; i++) {
        if (i == 0) {
            time_current = p[i].arrival_time; // Nếu tiến trình đầu tiên, khởi tạo thời gian hiện tại bằng thời gian đến của tiến trình đầu
        }

        // Thời gian hiện tại = thời gian hiện tại + thời gian burst của tiến trình đang xử lý
        time_current += p[i].burst_time;

        // Tính toán thời gian đợi và thời gian turnaround
        avg_waiting_time += time_current - p[i].arrival_time - p[i].burst_time;
        avg_turn_around_time += (time_current - p[i].arrival_time);

        // In thông tin tiến trình
        cout << "P" << p[i].name << "\t" << p[i].arrival_time << "\t" << p[i].burst_time << "\t" << time_current - p[i].burst_time << "\t"
             << (time_current - p[i].arrival_time) << "\t" << time_current << endl;

        // Sắp xếp lại các tiến trình chưa được thực hiện theo Burst Time
        sortByBurstTime(p + i + 1, n - i - 1, time_current);
    }
}

int main() {
    int n;
    Process p[100];

    cout << "Nhap so luong tien trinh: ";
```

Hình 5: Code từ dòng 45 - 85

```cpp
    cout << "Nhap so luong tien trinh: ";
    cin >> n;

    Input(p, n);

    cout << "PName\tArrTime\tBurTime\tStart\tTAT\tFinish\n";
    SelectionFunction(p, n);

    // In thời gian trung bình
    cout << "Thoi gian cho trung binh: " << avg_waiting_time / n << endl;
    cout << "Thoi gian hoan thanh trung binh: " << avg_turn_around_time / n << endl;

    return 0;
}
```

Hình 6: Code từ dòng 85 - 99

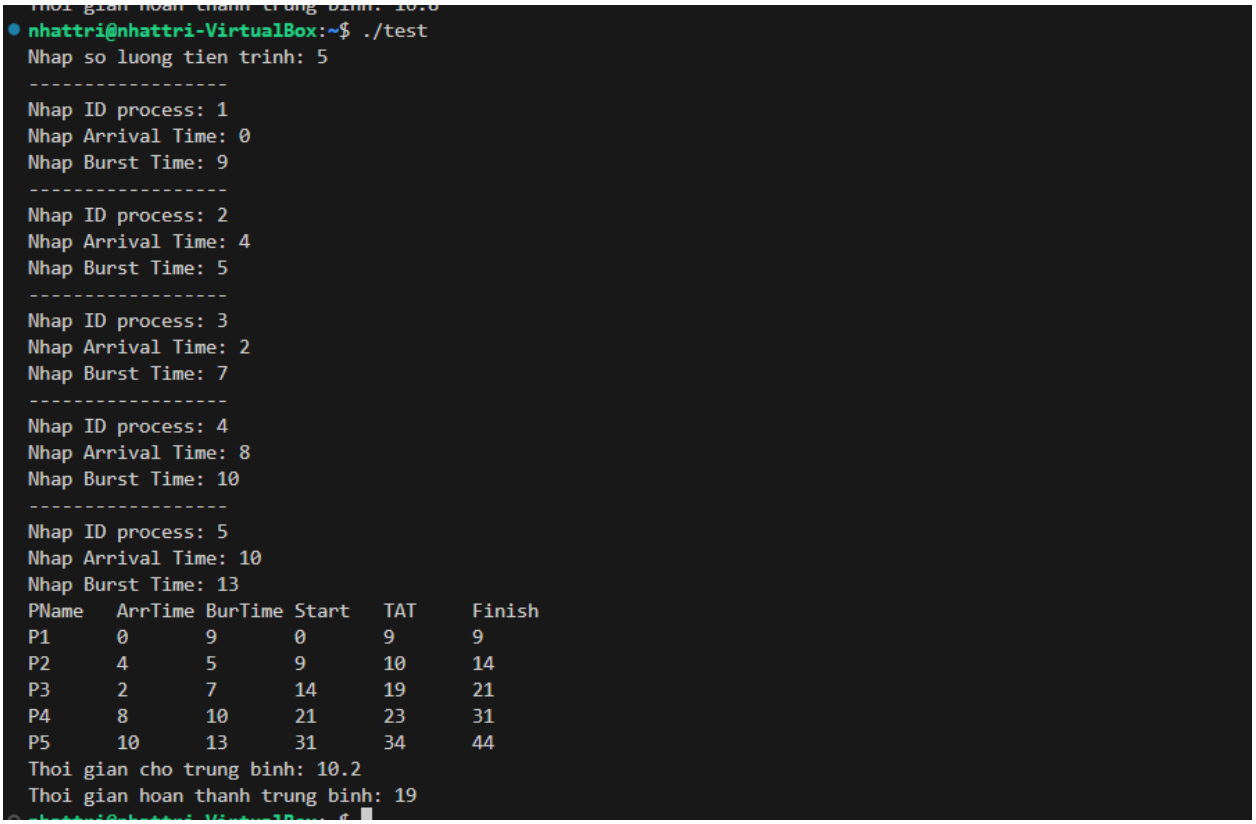
1.5. Test case

– Ví dụ 1:

Process 1	Arrival Time	Burst Time
P1	0	9
P2	4	5

P3	2	7
P4	8	10
P5	10	13

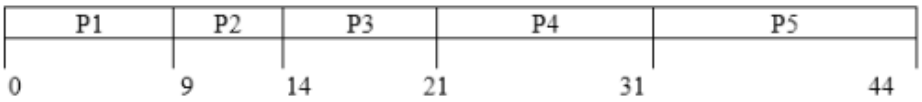
- Kết quả khi chạy code



Hình 7: 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 8:Kết quả khi giải tay ví dụ 1 bằng giải thuật SJF

– Ví dụ 2:

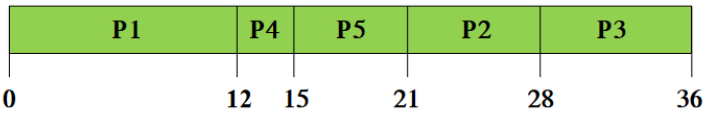
Process	Arriva Time	Burst Time
P1	0	12
P2	2	7
P3	5	8
P4	9	3
P5	12	6

- Kết quả khi chạy code

```
nhattri@nhattri-VirtualBox:~$ ./test
Nhap so luong tien trinh: 5
------------------
Nhap ID process: 1
Nhap Arrival Time: 0
Nhap Burst Time: 12
------------------
Nhap ID process: 2
Nhap Arrival Time: 2
Nhap Burst Time: 7
------------------
Nhap ID process: 3
Nhap Arrival Time: 5
Nhap Burst Time: 8
------------------
Nhap ID process: 4
Nhap Arrival Time: 9
Nhap Burst Time: 3
------------------
Nhap ID process: 5
Nhap Arrival Time: 12
Nhap Burst Time: 6
PName   ArrTime BurTime Start   TAT     Finish
P1      0       12      0       12      12
P4      9       3       12      6       15
P5      12      6       15      9       21
P2      2       7       21      26      28
P3      5       8       28      31      36
Thoi gian cho trung binh: 9.6
Thoi gian hoan thanh trung binh: 16.8
```

Hình 9: 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

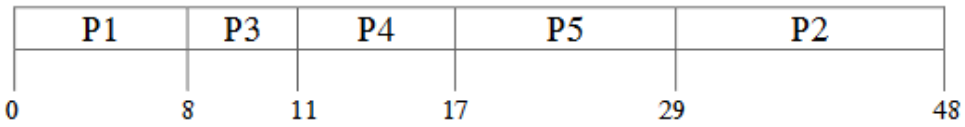
Hình 10:Kết quả khi giải tay ví dụ 2 bằng giải thuật SJF

– 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:

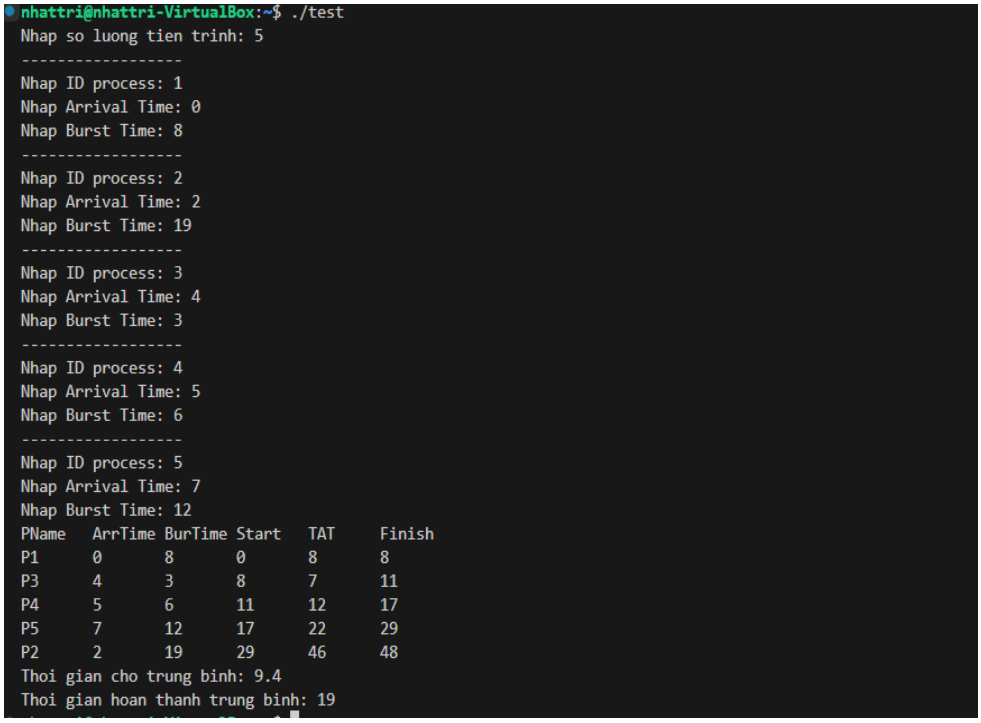


+ Thời gian đáp ứng trung bình là: 9.4

+ Thời gian hoàn thành trung bình:19.

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

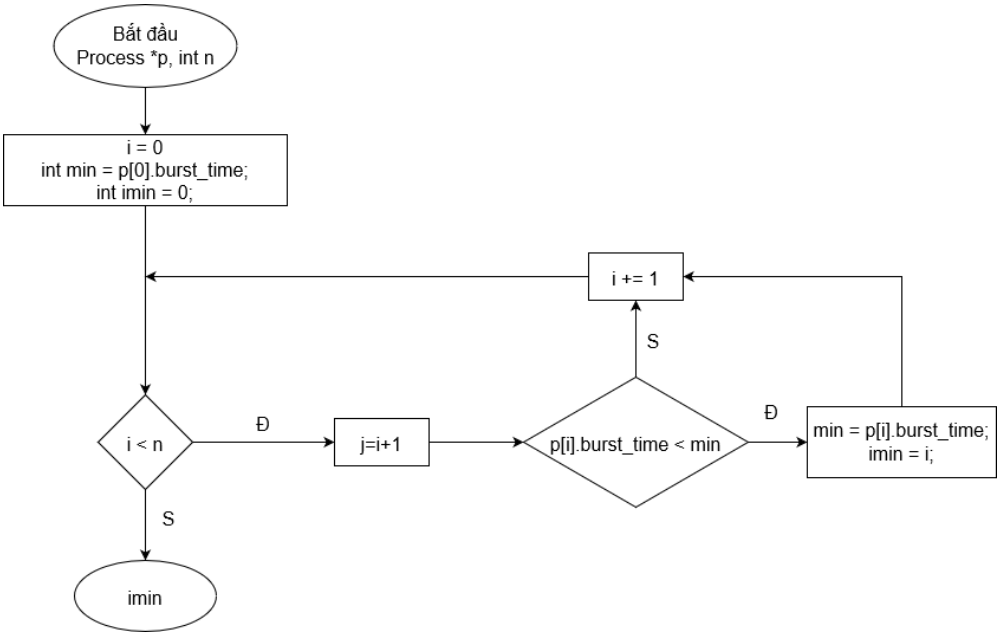
- Kết quả khi chạy code



Hình 12: Kết quả khi giải ví dụ 3 bằng code giải thuật SJF

Câu 2: Viết chương trình mô phỏng giải thuật SRTF.

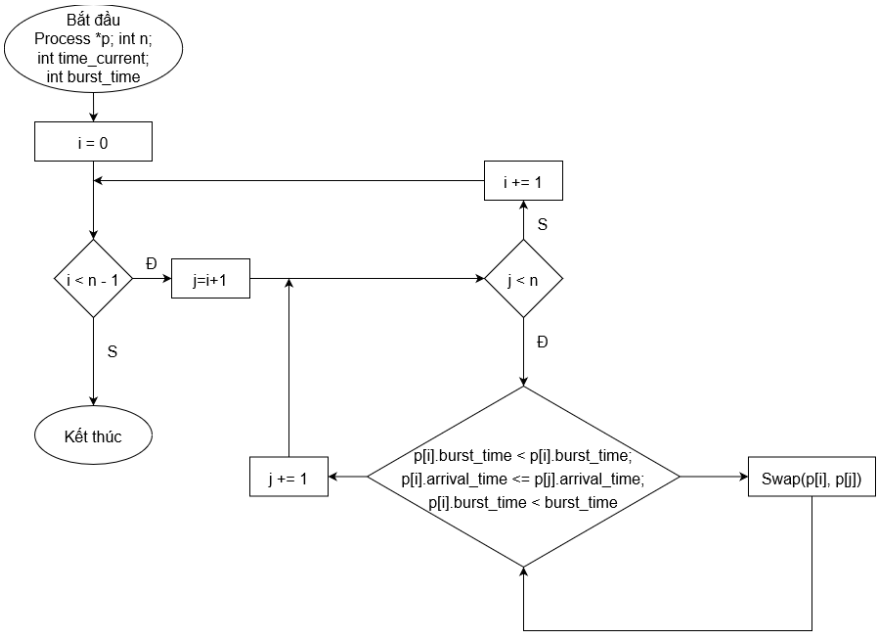
2.1. Hàm tìm ra tiến trình có burst time nhỏ nhất.



Hình 13: 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.

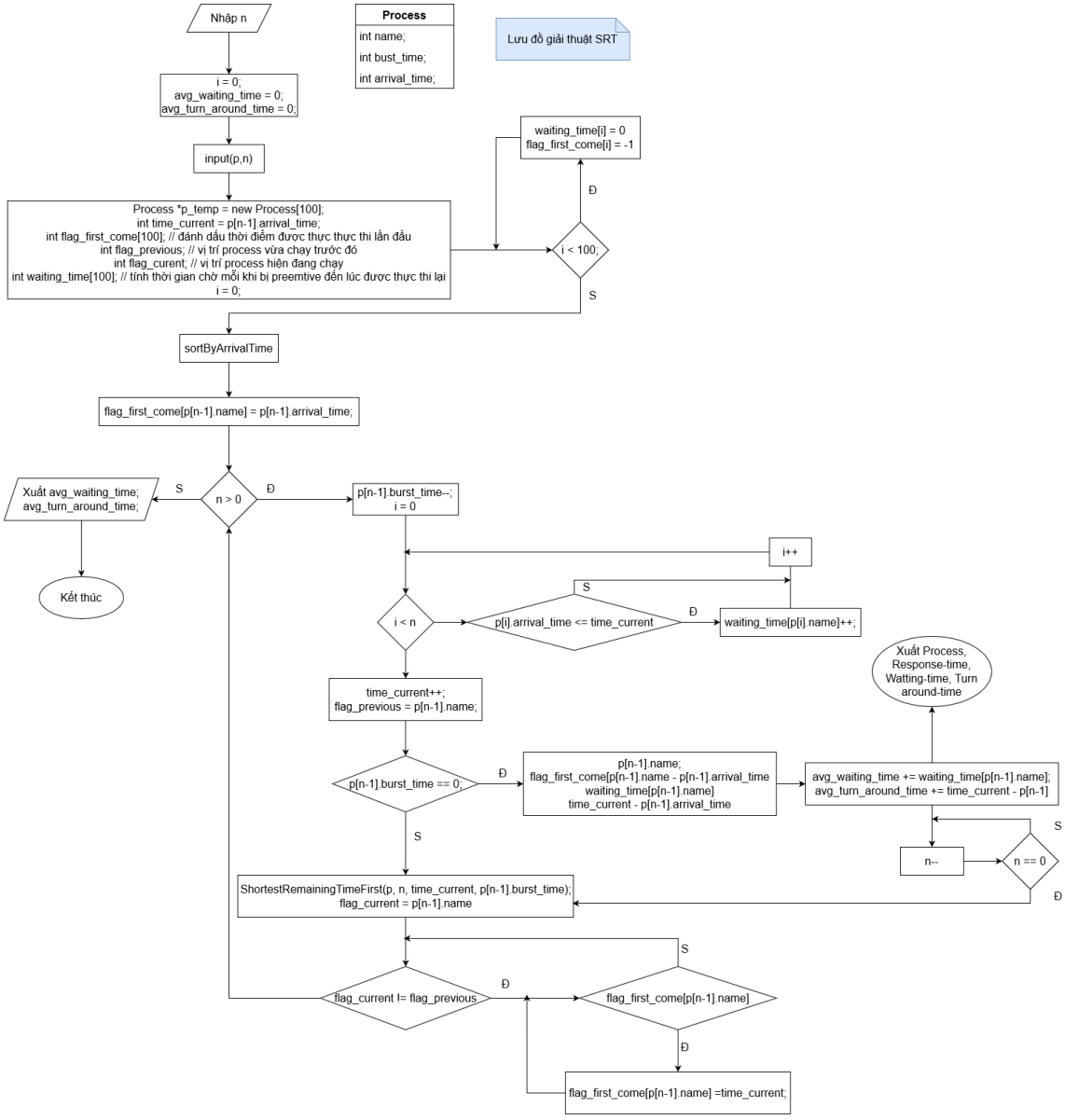


Hình 14: Hàm ShortestRemainingTimeFirst.

- **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 15: Lưu đồ giải thuật SRTF

2.4. Code giải thuật SRTF

```cpp
#include <iostream>
#include <algorithm>
#include <iomanip>
#include <string.h>

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;
};

int main() {
    int n;
    struct process p[100];

    float avg_turnaround_time;
    float avg_waiting_time;
    float avg_response_time;
    float cpu_utilisation;
    int total_turnaround_time = 0;
    int total_waiting_time = 0;
    int total_response_time = 0;
    int total_idle_time = 0;
    float throughput;
    int burst_remaining[100];
    int is_completed[100];

    memset(is_completed, 0, sizeof(is_completed));
    cout << setprecision(2) << fixed;

    cout << "Nhap so luong process: ";
    cin >> n;
    for (int i = 0; i < n; i++) {
        cout << "Nhap ID process: ";
        cin >> p[i].pid;
        cout << endl;
```

Hình 16: Code giải thuật SRT từ dòng 1 - 44

```cpp
        cout << "Nhap arrival time: ";
        cin >> p[i].arrival_time;
        cout << "Nhap burst time: ";
        cin >> p[i].burst_time;

        burst_remaining[i] = p[i].burst_time;
        cout << endl;
        }

    int completed = 0;
    int current_time = 0;
    int prev = 0;

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

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

        if (idx != -1) {
            if (burst_remaining[idx] == p[idx].burst_time) {
                p[idx].start_time = current_time;
                total_idle_time += p[idx].start_time - prev;
            }

            burst_remaining[idx]--;
            current_time++;
            prev = current_time;
```

Hình 17: Code giải thuật SRT từ dòng 45 - 85

```
85            prev = current_time;
86
87            if (burst_remaining[idx] == 0) {
88                p[idx].completion_time = current_time;
89                p[idx].turnaround_time = p[idx].completion_time - p[idx].arrival_time;
90                p[idx].waiting_time = p[idx].turnaround_time - p[idx].burst_time;
91                p[idx].response_time = p[idx].start_time - p[idx].arrival_time;
92
93                total_turnaround_time += p[idx].turnaround_time;
94                total_waiting_time += p[idx].waiting_time;
95                total_response_time += p[idx].response_time;
96
97                is_completed[idx] = 1;
98                completed++;
99                }
100           }
101           else {
102               current_time++;
103           }
104       }
105
106   int min_arrival_time = 10000000;
107   int max_completion_time = -1;
108
109   for (int i = 0; i < n; i++) {
110       min_arrival_time = min(min_arrival_time, p[i].arrival_time);
111       max_completion_time = max(max_completion_time, p[i].completion_time);
112   }
113
114   avg_turnaround_time = (float)total_turnaround_time / n;
115   avg_waiting_time = (float)total_waiting_time / n;
116   avg_response_time = (float)total_response_time / n;
117   cpu_utilisation = ((float)(max_completion_time - total_idle_time) / (float)max_completion_time) * 100;
118   throughput = float(n) / (max_completion_time - min_arrival_time);
119
120   cout << endl << endl;
121
122   cout << "#pName\t" << "AT\t" << "BT\t" << "ST\t" << "CT\t" << "TAT\t" << "WT\t" << "RT\t" << "\n" << endl;
```

Hình 18: Code giải thuật SRT từ dòng 85 - 122

```
122   cout << "#pName\t" << "AT\t" << "BT\t" << "ST\t" << "CT\t" << "TAT\t" << "WT\t" << "RT\t" << "\n" << endl;
123
124   for (int i = 0; i < n; i++) {
125       cout << p[i].pid << "\t" << p[i].arrival_time << "\t" << p[i].burst_time << "\t" <<
126           p[i].start_time << "\t" << p[i].completion_time << "\t" << p[i].turnaround_time << "\t" <<
127           p[i].waiting_time << "\t" << p[i].response_time << "\t" << "\n" << endl;
128   }
129
130   cout << "Thoi gian hoan thanh trung binh: = " << avg_turnaround_time << endl;
131   cout << "Thoi gian dap ung trung binh: = " << avg_waiting_time << endl;
132   }
```

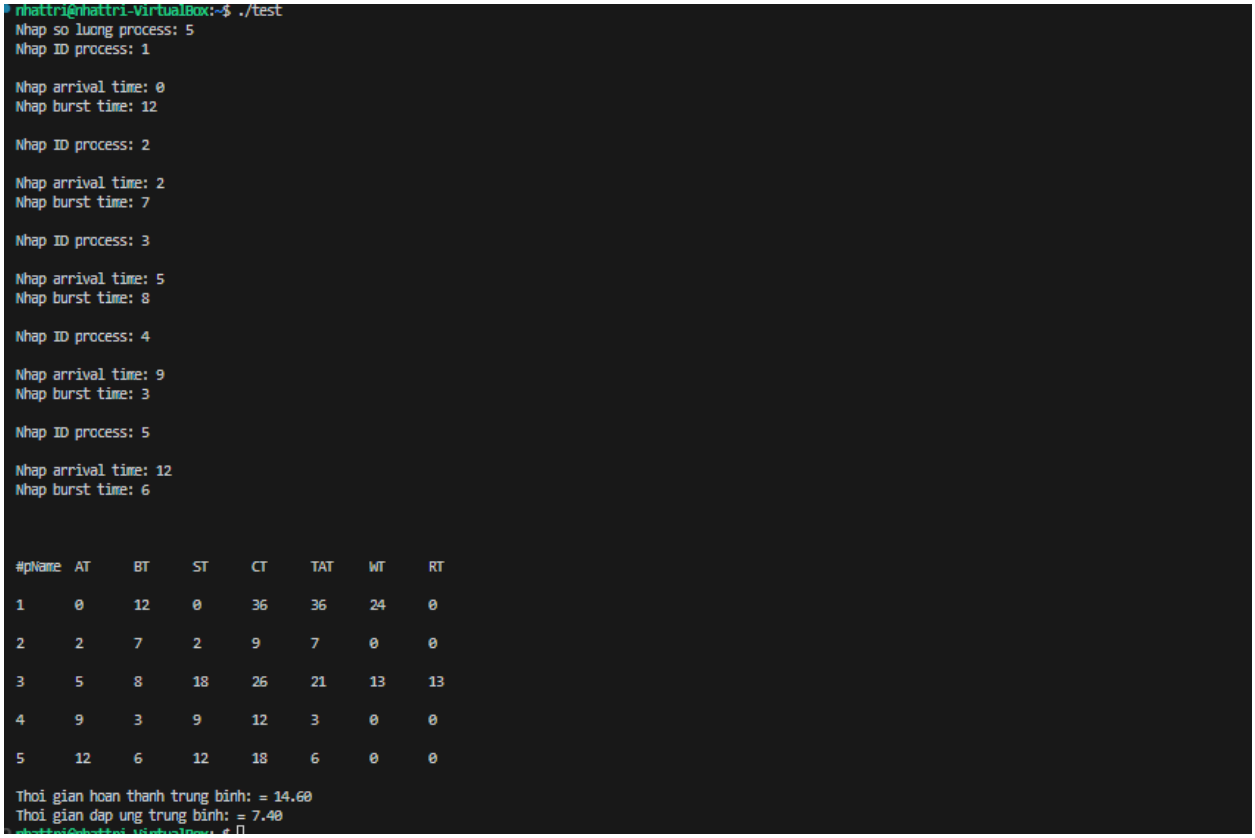
Hình 19: Code giải thuật SRT từ dòng 122 - 132

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

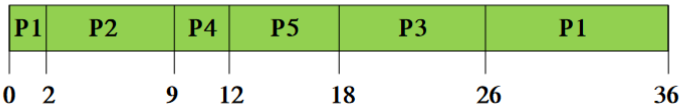
- Kết quả khi chạy code:



Hình 20: Kết quả khi giải ví dụ 1 bằng code giải thuật SRTF

- Kết quả khi giải tay:

- Giản đồ Gantt



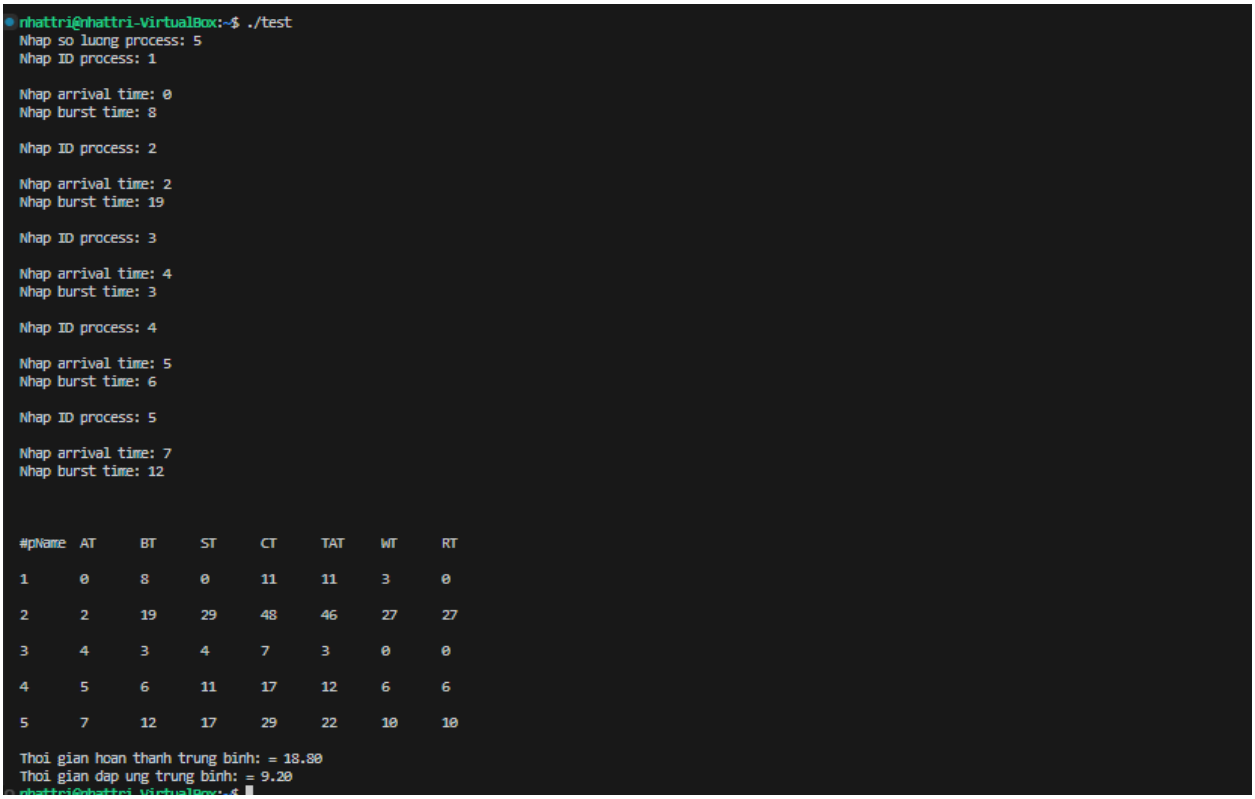
- Thời gian chờ:
  - P1 = 24, P2 = 0, P3 = 13, P4 = 0, P5 = 0
  - Thời gian chờ trung bình: (24 + 0 + 13 + 0 + 0)/5 = 7.4
- Thời gian đáp ứng:
  - P1 = 0, P2 = 0, P3 = 13, P4 = 0, P5 = 0
  - Thời gian đáp ứng trung bình: (0 + 0 + 13 + 0 + 0)/5 = 2.6
- Thời gian hoàn thành:
  - P1 = 36, P2 = 7, P3 = 21,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

Hình 21: Kết quả khi giải tay ví dụ 1 bằng giải thuật SRTF

– Ví dụ 2:

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

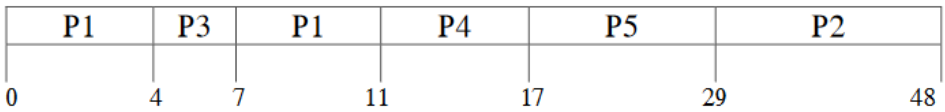
- Kết quả khi chạy code:



Hình 22: Kết quả khi giải ví dụ 2 bằng code giải thuật SRTF

- 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 23: Kết quả khi giải tay ví dụ 2 bằng giải thuật SRTF

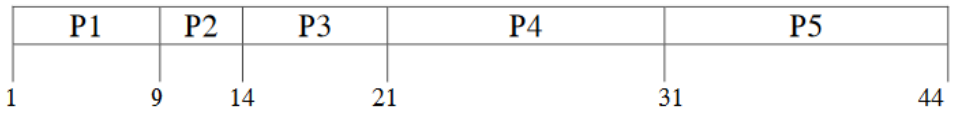
– Ví dụ 3:

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:

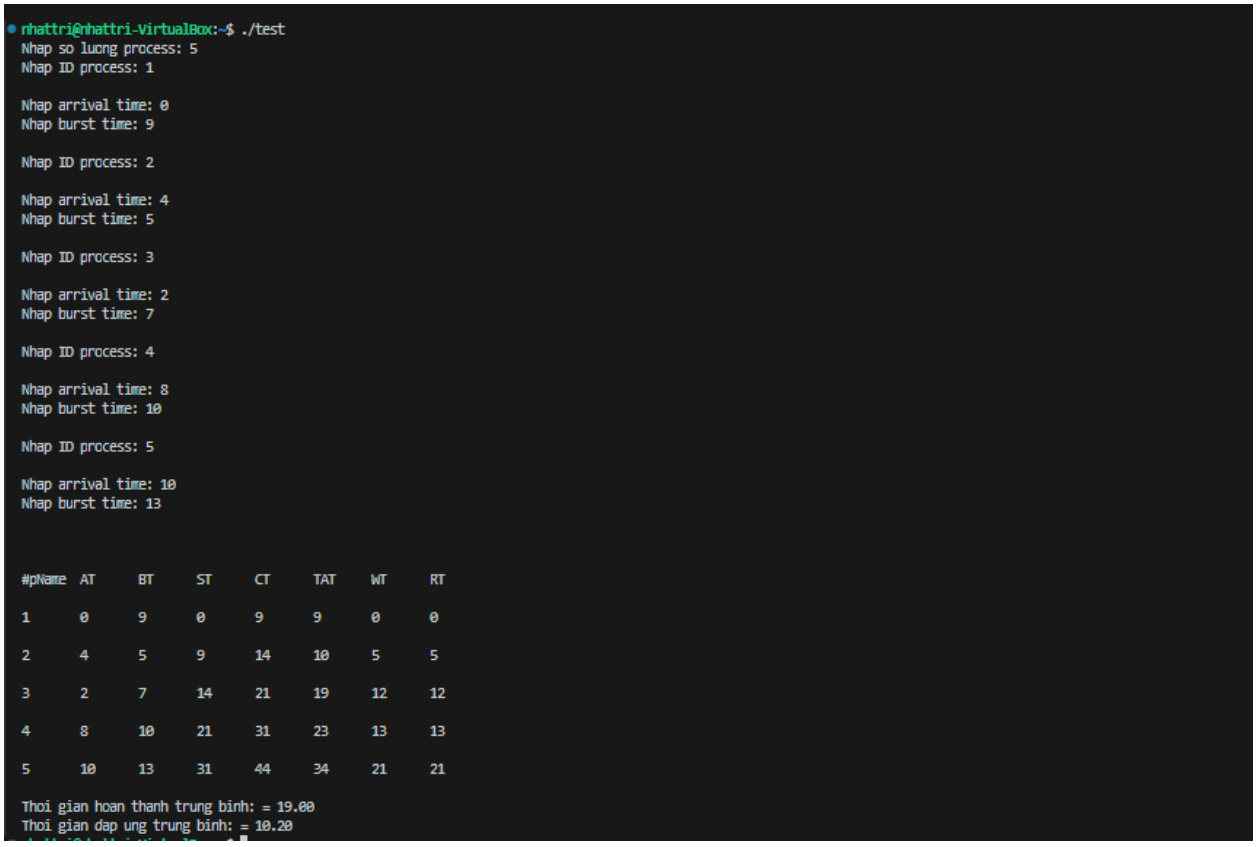


+ *Thời gian đáp ứng trung bình là: 10.2*

+ *Thời gian hoàn thành trung bình: 19*

Hình 24: Kết quả khi giải tay ví dụ 3 bằng giải thuật SRTF

- Kết quả khi chạy code



Hình 25: Kết quả khi giải ví dụ 3 bằng code giải thuật SRTF