ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA CÔNG NGHỆ THÔNG TIN

PBL4-DỰ ÁN HỆ ĐIỀU HÀNH & MẠNG MÁY TÍNH

Đề tài: Mô phỏng các thuật toán lập lịch CPU

SINH VIÊN THỰC HIỆN:

Lý Gia Khánh **LỚP: 21TCLC_DT4 NHÓM: 10**

Trần Thị Hiền Lương **LỚP: 21TCLC_DT4 NHÓM: 10**

GIẢNG VIÊN HƯỚNG DẪN: Nguyễn Thị Lệ Quyên

Đà Nẵng, 1/2024

MỤC LỤC

MỤC LỤC 1
DANH SÁCH HÌNH VẼ 3
MỞ ĐẦU (GIỚI THIỆU ĐỀ TÀI) 5
CHƯƠNG 1. CƠ SỞ LÝ THUYẾT 6
1.1. Các khái niệm 6
1.1.1. Tiến trình là gì? 6
1.1.2. Vì sao phải lập lịch cho tiến trình? 7
1.1.3. Mục tiêu của lập lịch 7
1.2. Các thuật toán lập lịch 8
1.2.1. First-Come First-Served (FCFS) 8
1.2.2. Shortest Job First (SJF) 9
1.2.3. Shortest Remaining Time First (SRTF) 10
1.2.4. Round Robin 11
1.2.5. Priority Non-Preemptive 12
1.2.6. Priority Preemptive 13
CHƯƠNG 2. THIẾT KẾ VÀ XÂY DỰNG HỆ THỐNG 15
2.1. Mô hình thuật toán lập lịch 15
2.1.1. First-Come First-Served (FCFS) 15
2.1.2. Shortest Job First (SJF) 17
2.1.3. Shortest Remaining Time First (SRTF) 19
2.1.4. Round Robin 21
2.1.5. Priority Preemptive 23
2.1.6. Priority Non-Preemptive 25
CHƯƠNG 3. DEMO ỨNG DỤNG VÀ ĐÁNH GIÁ KẾT QUẢ 27
3.1. Demo ứng dụng 27
3.1.1. Giao diện người dùng 27
3.1.2. Thêm mới các tiến trình trước khi thực hiện mô phỏng 27

3.1.3. Sửa các thông tin của tiến trình 35

3.1.4. Xoá tiến trình 37

3.1.5. Mô phỏng các thuật toán lập lịch 39

3.2. Đánh giá kết quả 45

3.2.1. First-Come First-Served (FCFS) 45

3.2.2. Shortest Job First (SJF) 45

3.2.3. Shortest Remaining Time First (SRTF) 46

3.2.4. Round Robin 46

3.2.5. Priority Preemptive 47

3.2.6. Priority Non-Preemptive 48

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 49

TÀI LIỆU THAM KHẢO 51

DANH SÁCH HÌNH VẼ

Hình 1: Trạng thái của tiến trình 6
Hình 2: Ví dụ minh hoạ thuật toán FCFS 8
Hình 3: Ví dụ minh hoạ thuật toán SJF 9
Hình 4: Ví dụ minh hoạ thuật toán SRTF 10
Hình 5: Ví dụ minh hoạ thuật toán Round Robin 12
Hình 6: Ví dụ minh hoạ thuật toán Priority Non-Preemptive 13
Hình 7: Ví dụ minh hoạ thuật toán Priority Preemptive 14
Hình 8: Sơ đồ khối thuật toán FCFS 16
Hình 9: Sơ đồ khối thuật toán Short Job First 18
Hình 10: Sơ đồ khối thuật toán SRTF 20
Hình 11: Sơ đồ khối thuật toán Round Robin 22
Hình 12: Sơ đồ khối thuật toán Priority Preemptive 24
Hình 13: Sơ đồ khối thuật toán Priority Non-Preemptive 26
Hình 14: Giao diện người dùng 27
Hình 15: Giao diện nhập thông tin của tiến trình 28
Hình 16: Giao diện sau khi thêm tiến trình 29
Hình 17: Giao diện nhập số lượng tiến trình cần random 30
Hình 18: Giao diện sau khi random thành công 31
Hình 19: Thông tin file cần đọc 32
Hình 20: Giao diện chọn file 33
Hình 21: Giao diện đọc file thành công 34
Hình 22: Giao diện chọn một tiến trình cần cập nhật 35
Hình 23: Giao diện cập nhật thông tin thành công 36
Hình 24: Giao diện chọn tiến trình cần xoá 37
Hình 25: Giao diện sau khi xoá thành công 38
Hình 26: Giao diện mô phỏng thuật toán FCFS 39
Hình 27: Giao diện mô phỏng thuật toán SJF 40
Hình 28: Giao diện mô phỏng thuật toán SRTF 41
Hình 29: Giao diện mô phỏng thuật toán Round Robin 42

Hình 30: Giao diện mô phỏng thuật toán Priority Preemptive 43

Hình 31: Giao diện mô phỏng thuật toán Priority Non-Preemptive 44

MỞ ĐẦU (GIỚI THIỆU ĐỀ TÀI)

- Trong thế giới ngày nay, hệ điều hành đóng vai trò quan trọng như một "não bộ" của máy tính, điều chỉnh và quản lý tài nguyên để đảm bảo hoạt động mượt mà của hệ thống. Một phần quan trọng của hệ điều hành là khả năng quản lý công việc và lên lịch cho các tiến trình, và điều này thường được thực hiện thông qua các thuật toán lập lịch.
- Thuật toán lập lịch là những chiến lược mà hệ điều hành sử dụng để quyết định thứ tự thực hiện các công việc, từ đó tối ưu hóa hiệu suất và sử dụng tài nguyên một cách hiệu quả. Trong môi trường máy tính hiện đại, nơi mà hàng trăm, thậm chí hàng nghìn nhiệm vụ có thể chạy cùng một lúc, việc áp dụng các thuật toán lập lịch hiệu quả sẽ trở thành chìa khóa để đảm bảo hệ thống hoạt động mượt mà và đáp ứng nhanh chóng đối với nhu cầu người sử dụng.
- Các thuật toán lập lịch còn giúp ưu tiên các nhiệm vụ quan trọng, đảm bảo rằng công việc quan trọng sẽ được hoàn thành trước. Điều này mang lại lợi ích lớn trong việc đảm bảo sự cân bằng giữa tất cả các tài nguyên, tránh tình trạng quá tải hoặc thất thoát tài nguyên. Ngoài ra, khả năng điều chỉnh độ ưu tiên của các thuật toán này cũng là một điểm lợi thế lớn. Điều này giúp hệ thống linh hoạt thích ứng với tình hình hoạt động hiện tại. Đồng thời, chúng giúp tiết kiệm năng lượng bằng cách quản lý việc bật/tắt các thành phần không cần thiết của hệ thống.
- Việc hiểu rõ và sử dụng hiệu quả các thuật toán lập lịch không chỉ nâng cao khả năng quản lý của hệ thống mà còn tạo ra những trải nghiệm người dùng tốt hơn trong thế giới kỹ thuật số ngày nay. Đây chính là lý do chúng em chọn đề tài**: "Mô phỏng các thuật toán lập lịch CPU".**

CHƯƠNG 1. CƠ SỞ LÝ THUYẾT

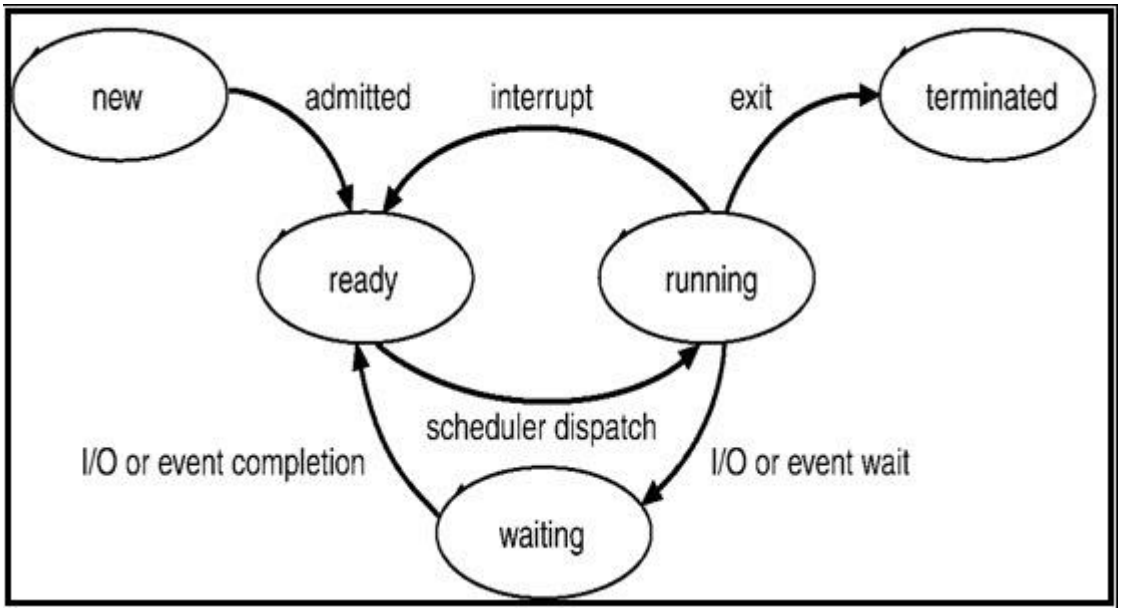
1.1. Các khái niệm

1.1.1. Tiến trình là gì?

a) Định nghĩa

- Tiến trình là một chương trình đang chạy. Mỗi tiến trình được cấp một không gian bộ nhớ độc lập và có thể chứa một hoặc nhiều luồng thực hiện. Hệ điều hành quản lý các tiến trình, kiểm soát quyền truy cập vào bộ nhớ và tài nguyên, và lập lịch để đảm bảo rằng máy tính có thể đồng thời thực hiện nhiều công việc một cách hiệu quả. Các tiến trình giúp tổ chức và phối hợp các hoạt động trên hệ thống, đóng vai trò quan trọng trong quản lý tài nguyên và cung cấp trải nghiệm đa nhiệm cho người sử dụng.
- Tiến trình thực hiện một cách có tuần tự, nghĩa là điểm khởi tạo của một tiến trình là điểm kết thúc của một tiến trình trước đó.

b) Các trạng thái của tiến trình



Hình 1: Trạng thái của tiến trình

- New: Tiến trình vừa được tạo. Trong giai đoạn này, hệ điều hành đang chuẩn bị các tài nguyên cần thiết cho tiến trình.
- Ready (Sẵn Sàng): Tiến trình đã có đủ tài nguyên và đang chờ được cấp CPU để bắt đầu thực hiện. Ở trạng thái này, tiến trình có thể được lập lịch để chạy.
- Running (Đang Chạy): Các lệnh của tiến trình đang được thực thi trên CPU. Trong thời gian này, tiến trình sử dụng các tài nguyên và thực hiện các công việc được giao.
- Waiting (Blocked - Đang Đợi): Tiến trình đang ở trong trạng thái đợi khi thực hiện các thao tác I/O (Input/Output) như đọc từ hoặc ghi vào bộ nhớ, tệp tin, hoặc các hoạt động mạng. Trong thời gian này, CPU có thể được cấp cho một tiến trình khác.

- Terminated (Kết Thúc): Tiến trình đã hoàn thành công việc của mình và đã kết thúc. Trong giai đoạn này, hệ điều hành giải phóng tất cả tài nguyên mà tiến trình đã sử dụng.

1.1.2. Vì sao phải lập lịch cho tiến trình?

- Trong môi trường hệ thống đa nhiệm (multitasking), bộ nhớ thường chứa đồng thời nhiều tiến trình (process). Tuy nhiên, chỉ một tiến trình được thực thi tại mỗi thời điểm. Điều này đặt ra yêu cầu cần phải quản lý hiệu quả việc phân chia và sắp xếp thời gian sử dụng CPU cho các tiến trình. Để giải quyết vấn đề này, quá trình lập lịch CPU trở nên quan trọng để đảm bảo sự công bằng, hiệu suất và phản hồi nhanh chóng đối với cả người sử dụng và hệ thống.
- Lập lịch CPU là quá trình quyết định xem tiến trình nào sẽ được thực thi tiếp theo và được CPU cấp phát thời gian xử lý. Việc này giúp tối ưu hóa sự sử dụng tài nguyên CPU và đảm bảo rằng mỗi tiến trình đều nhận được cơ hội sử dụng CPU một cách công bằng, ngăn chặn tình trạng ưu tiên quá mức của một số tiến trình so với các tiến trình khác.
- Lập lịch CPU cũng đóng vai trò quan trọng trong việc đáp ứng yêu cầu của người sử dụng, như đảm bảo các ứng dụng mở trước mắt người dùng được xử lý một cách linh hoạt và nhanh chóng. Đồng thời, nó cũng giúp hệ thống duy trì hiệu suất ổn định và tránh tình trạng chết đói tài nguyên.
- Tóm lại, lập lịch CPU là một phần quan trọng của quản lý tiến trình trong hệ thống đa nhiệm, nhằm đảm bảo sự công bằng và hiệu suất trong việc phân phối thời gian sử dụng CPU cho các tiến trình của cả người sử dụng và hệ thống.

1.1.3. Mục tiêu của lập lịch

Mục tiêu chính của lập lịch CPU là đảm bảo việc quản lý và phân phối thời gian sử dụng CPU một cách hiệu quả, nhằm đáp ứng các yêu cầu đa dạng của các tiến trình trong hệ thống đa nhiệm. Đồng thời, lập lịch CPU cũng nhằm tối ưu hóa hiệu suất hệ thống và cung cấp trải nghiệm người dùng mượt mà và nhanh chóng.

Mục tiêu cụ thể của lập lịch CPU bao gồm:

- Sự công bằng: Đảm bảo mọi tiến trình đều có cơ hội sử dụng CPU một cách công bằng, tránh tình trạng ưu tiên quá mức của một số tiến trình so với các tiến trình khác và không có tiến trình nào phải chờ đợi vô hạn để được cấp phát CPU.
- Tính hiệu quả: Tối ưu hóa sự sử dụng tài nguyên CPU để đạt được hiệu suất tốt nhất từ hệ thống, giảm độ trễ và đảm bảo thời gian đáp ứng nhanh chóng đối với các yêu cầu người sử dụng, cần giữ cho CPU càng bận càng tốt.
- Thời gian hồi đáp (Response time): Tối thiểu hóa thời gian từ khi gửi một yêu cầu tới hệ thống cho tới khi nhận được phúc đáp từ hệ thống.

- Thời gian xoay vòng (Turnaround time) nhỏ: Tối thiểu hóa thời gian hoàn tất các tác vụ xử lý theo lô .
- Thông lượng (Throughput) cao: Tối đa số tiến trình kết thúc trong một đơn vị thời gian.

1.2. Các thuật toán lập lịch

1.2.1. First-Come First-Served (FCFS)

- Đây là thuật toán điều phối theo nguyên tắc độc quyền.
- Tiến trình nào đến trước sẽ được xử lý trước, theo thứ tự chúng đến hệ thống.
- Không có ưu tiên dựa trên bất kỳ tiêu chí nào khác ngoài thời điểm đến.
- Hàng đợi các tiến trình được tổ chức theo kiểu FIFO. Mọi tiến trình đều được phục vụ theo trình tự xuất hiện cho đến khi kết thúc hoặc bị ngắt.
- Các bước thực hiện:
  - Bước 1: Sắp xếp các tiến trình theo thời gian đến (arrival time), là thời điểm khi chúng yêu cầu được thực hiện. Thời gian đến sẽ quyết định thứ tự tiến trình trong hàng đợi.
  - Bước 2: Tiến trình được chọn sẽ được thực hiện. Tiến trình này sẽ chạy cho đến khi hoàn thành công việc.
  - Bước 3: Sau khi tiến trình hoàn thành, các thông tin như thời gian chờ đợi (wait time = turnaround time – burst time), thời gian quay vòng (turnaround time = completion time – arrive time), và thời gian hoàn thành (completion time) của tiến trình đó sẽ được tính toán và cập nhật.
  - Bước 4: Lặp lại bước 2 và 3 cho đến khi tất cả các tiến trình được lập lịch.

Example:

Process	Arrival time	Burst time
P1	0 ms	18 ms
P2	2 ms	7 ms
P3	2 ms	10 ms

Gantt Chart

P1		P2		P3	
0 ms	18 ms	18 ms	25 ms	25 ms	35 ms

Hình 2: Ví dụ minh hoạ thuật toán FCFS

1.2.2. Shortest Job First (SJF)

- Là thuật toán điều phối theo nguyên tắc độc quyền (non-preemptive), nghĩa là tiến trình sẽ chạy đến khi hoàn thành mà không bị gián đoạn.
- Trong lĩnh vực quản lý tài nguyên và lập kế hoạch cho các tiến trình trong hệ thống máy tính, Shortest Job First (SJF) là một trong những giải thuật quan trọng nhất.
- Giải thuật SJF dựa trên nguyên tắc sắp xếp các tiến trình theo thời gian thực thi ngắn nhất. Tiến trình có thời gian thực hiện (burst time) ngắn hơn sẽ được ưu tiên thực hiện trước để tối ưu hóa thời gian đáp ứng và hiệu suất hệ thống.
- Các bước thực hiện:
    - Bước 1: Sắp xếp: Các tiến trình được sắp xếp theo thời gian đến (arrival time), là thời điểm khi chúng yêu cầu được thực hiện. Thời gian đến sẽ quyết định thứ tự tiến trình trong hàng đợi.
    - Bước 2: Lựa Chọn Tiến Trình: Khi một tiến trình kết thúc hoặc mới đến, hệ thống sẽ chọn tiến trình có thời gian thực hiện (burst time) ngắn nhất từ trong hàng đợi.
    - Bước 3: Thực Hiện Tiến Trình: Tiến trình được chọn sẽ được thực hiện. Tiến trình này sẽ chạy cho đến khi hoàn thành công việc.
    - Bước 4: Tính Toán: Sau khi tiến trình kết thúc, các thông tin như thời gian chờ đợi (wait time), thời gian quay vòng (turnaround time), và thời gian hoàn thành (completion time) của tiến trình đó sẽ được cập nhật.
    - Bước 5: Lặp Lại: Quá trình lựa chọn và thực hiện tiến trình sẽ tiếp tục cho đến khi tất cả các tiến trình hoàn thành.

Example:

Process	Arrival time	Burst time
P1	3 ms	5 ms
P2	0 ms	4 ms
P3	4 ms	2 ms
P4	5 ms	4 ms

Gantt Chart

P2		P3		P4		P1	
0ms	4ms	4ms	6ms	6ms	10ms	10ms	15ms

Hình 3: Ví dụ minh hoạ thuật toán SJF

1.2.3. Shortest Remaining Time First (SRTF)

- Đây là trường hợp mở rộng của SJF, theo cơ chế không độc quyền.
- Nguyên tắc: Chọn tiến trình có thời gian còn lại (Remaining time) ngắn nhất để thực hiện trước. Nếu một tiến trình mới đến và có thời gian còn lại ngắn hơn tiến trình đang thực hiện, hệ thống sẽ tạm dừng tiến trình đang thực hiện và chuyển sang thực hiện tiến trình mới.
- Các bước thực hiện:
  - Bước 1: Khởi tạo danh sách các tiến trình chưa được thực hiện. Đặt thời gian thực hiện ban đầu (thời gian của CPU) là 0.
  - Bước 2: Xác định tiến trình nào sẽ được thực hiện tiếp theo, bắt đầu từ danh sách các tiến trình chưa được thực hiện. So sánh thời gian còn lại để thực hiện của từng tiến trình và chọn tiến trình có thời gian còn lại nhỏ nhất.
  - Bước 3: Thực hiện tiến trình đã được chọn trong một khoảng thời gian ngắn nhất. Cập nhật thời gian của CPU.
  - Bước 4: Kiểm tra xem tiến trình đã hoàn thành hay chưa. Nếu đã hoàn thành, đánh dấu tiến trình đã kết thúc. Tiến hành tính toán và cập nhật các các thông tin như thời gian chờ đợi (wait time), thời gian quay vòng (turnaround time), và thời gian hoàn thành (completion time) của tiến trình đó .
  - Bước 5: Lặp lại bước 2 đến bước 4 cho đến khi tất cả các tiến trình được lập lịch.

Example:

Process	Arrival time	Burst time
P1	1 ms	6 ms
P2	1 ms	8 ms
P3	2 ms	7 ms
P4	3 ms	3 ms

Gantt Chart

P1		P4		P1		P3		P2	
1	3	3	6	6	10	10	17	17	25

Hình 4: Ví dụ minh hoạ thuật toán SRTF

1.2.4. Round Robin

- Thuật toán Round Robin (RR) là một phương pháp lập lịch trong hệ điều hành, nơi mỗi tiến trình được thực hiện trong một khoảng thời gian cố định được gọi là "quantum".
- Các bước thực hiện:
  - Bước1: Sắp xếp: Các tiến trình được xếp theo thứ tự thời gian đến (arrival time), là thời điểm khi chúng yêu cầu được thực hiện. Thời gian đến sẽ quyết định thứ tự tiến trình trong hàng đợi.
  - Bước 2: Cấp phát thời gian: Một đơn vị thời gian cố định, gọi là "quantum", được cho trước. Mỗi tiến trình sẽ được thực hiện trong một khoảng thời gian này.
  - Bước 3: Thực Hiện Tiến Trình: Tiến trình đầu tiên trong hàng đợi sẽ được thực hiện trong khoảng thời gian được quyết định là "quantum". Nếu tiến trình không kết thúc sau khi đã sử dụng hết quantum, nó sẽ được chuyển xuống cuối hàng đợi.
  - Bước 4: Chuyển Tiến Trình: Nếu một tiến trình kết thúc hoặc đã sử dụng hết quantum, hệ thống sẽ kiểm tra có tiến trình mới đến hay không sau đó sẽ chuyển sang tiến trình đó trong hàng đợi thực hiện.
  - Bước 5: Tính Toán: Sau khi một tiến trình kết thúc, các thông tin như thời gian chờ đợi (wait time), thời gian quay vòng (turnaround time), và thời gian hoàn thành (completion time) của tiến trình đó sẽ được cập nhật.
  - Bước 6: Lặp Lại: Quá trình thực hiện và chuyển đổi tiến trình sẽ lặp đi lặp lại cho đến khi tất cả các tiến trình hoàn thành.

Example:

Process	Arrival time	Burst time
P1	0 ms	10 ms
P2	0 ms	5 ms
P3	0 ms	8 ms

Gantt Chart

P1		P2		P3		P1		P2		P3	
1	2	2	4	4	6	6	8	8	10	10	12

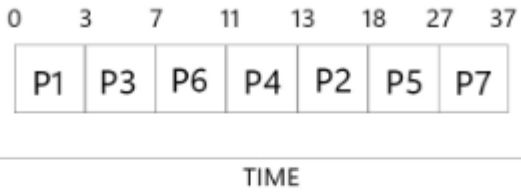
P1		P2		P3		P1		P3		P1	
12	14	14	15	15	17	17	19	19	21	21	23

Hình 5: Ví dụ minh hoạ thuật toán Round Robin

1.2.5. Priority Non-Preemptive

- Là thuật toán điều phối theo nguyên tắc độc quyền.
- Thuật toán này không cho phép gián đoạn thực hiện một tiến trình khi nó đang chạy. Nghĩa là, một khi một tiến trình bắt đầu thực hiện, nó sẽ chạy đến khi hoàn thành hoặc có sự can thiệp từ hệ thống.
- Mức ưu tiên (Priority): Mỗi tiến trình được gắn một mức ưu tiên. Mức ưu tiên này thường được thiết lập dựa trên các yếu tố như độ quan trọng của tiến trình, thời gian đã chờ đợi, hoặc các yếu tố khác quan trọng đối với hệ thống. Trong chương trình này, tiến trình nào có mức ưu tiên (priority) càng nhỏ thì sẽ có độ ưu tiên càng cao.
- Quyết Định Lịch Trình: Khi một tiến trình mới được tạo hoặc một tiến trình đã hoàn thành, hệ thống sẽ quyết định xem tiến trình nào sẽ được thực hiện tiếp theo dựa trên mức ưu tiên của chúng.
- Các bước thực hiện:
    - Bước 1: Sắp xếp các tiến trình theo thứ tự tăng dần của thời gian đến (arrival time) - thời điểm khi chúng yêu cầu được thực hiện và mức ưu tiên (priority). Điều này có nghĩa là tiến trình có mức ưu tiên nhỏ nhất sẽ được đặt ở đầu danh sách.
    - Bước 2: Bắt đầu từ đầu danh sách, thực hiện tiến trình ưu tiên cao nhất đầu tiên. Tiến trình này sẽ chạy đến khi hoàn thành.

- Bước 3: Sau khi một tiến trình kết thúc, đánh dấu tiến trình đã hoàn thành trong danh sách. Tiến hành tính toán và cập nhật các thông tin như thời gian chờ đợi (wait time), thời gian quay vòng (turnaround time), và thời gian hoàn thành (completion time) của tiến trình đó.
- Bước 4: Lặp lại bước 2 và 3 cho đến khi tất cả các tiến trình đều hoàn thành.



Process	P1	P2	P3	P4	P5	P6	P7
Burst Time	3	5	4	2	9	4	10
Priority	3	6	3	5	7	4	10
Arrival Time	0	2	1	4	6	5	7

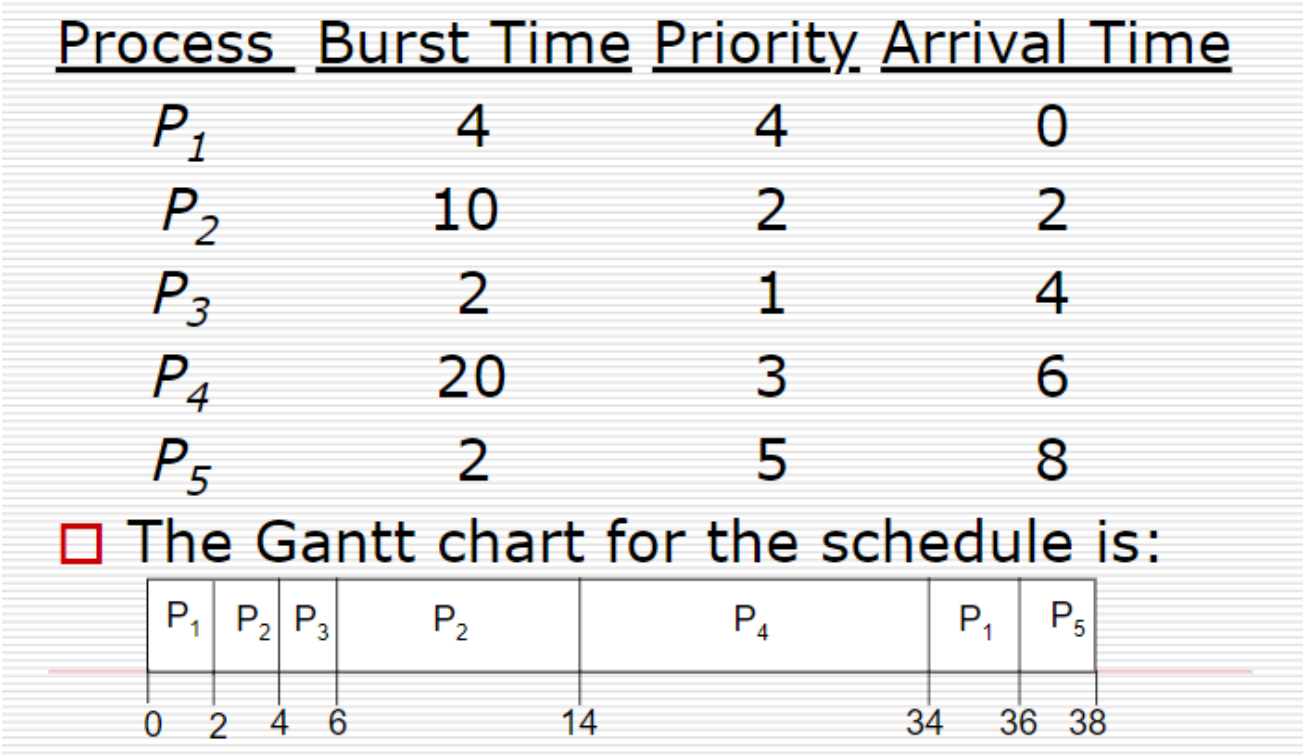
Hình 6: Ví dụ minh hoạ thuật toán Priority Non-Preemptive

1.2.6. Priority Preemptive

- Là thuộc toán điều phối theo nguyên tắc không độc quyền.
- Thường được sử dụng trong các hệ điều hành thời gian thực, nơi cần đảm bảo các tiến trình quan trọng được xử lý trước.
- Là một thuật toán phân phối thời gian thực trong hệ điều hành. Thuật toán này hoạt động bằng cách cấp phát CPU cho tiến trình có mức ưu tiên cao nhất. Nếu một tiến trình có mức ưu tiên cao hơn được đưa vào hàng đợi trong khi một tiến trình đang chạy, thì tiến trình đang chạy sẽ bị ngắt và tiến trình có mức ưu tiên cao hơn sẽ được cấp phát CPU.
- Các bước thực hiện:
    - Bước1: Sắp xếp: Các tiến trình được xếp theo thứ tự thời gian đến (arrival time), là thời điểm khi chúng yêu cầu được thực hiện. Thời gian đến sẽ quyết định thứ tự tiến trình trong hàng đợi.
    - Bước 2: Lựa Chọn Tiến Trình: hệ thống sẽ chọn tiến trình có độ ưu tiên cao nhất trong hàng đợi.
    - Bước 3: Thực Hiện Tiến Trình: Thực hiện tiến trình đã chọn cho đến khi hoàn thành hoặc bị gián đoạn bởi tiến trình mới đến có độ ưu tiên cao hơn.
    - Bước 4: Chuyển Tiến Trình: Nếu có tiến trình mới đến và có độ ưu tiên cao hơn, chuyển ngay lập tức sang thực hiện tiến trình mới.

- Bước 5: Tính Toán: Sau khi một tiến trình kết thúc, các thông tin như thời gian chờ đợi (wait time), thời gian quay vòng (turnaround time), và thời gian hoàn thành (completion time) của tiến trình đó sẽ được cập nhật.
- Bước 6: Lặp Lại: Quá trình thực hiện và chuyển đổi tiến trình sẽ lặp đi lặp lại cho đến khi tất cả các tiến trình hoàn thành.

Example



Hình 7: Ví dụ minh hoạ thuật toán Priority Preemptive

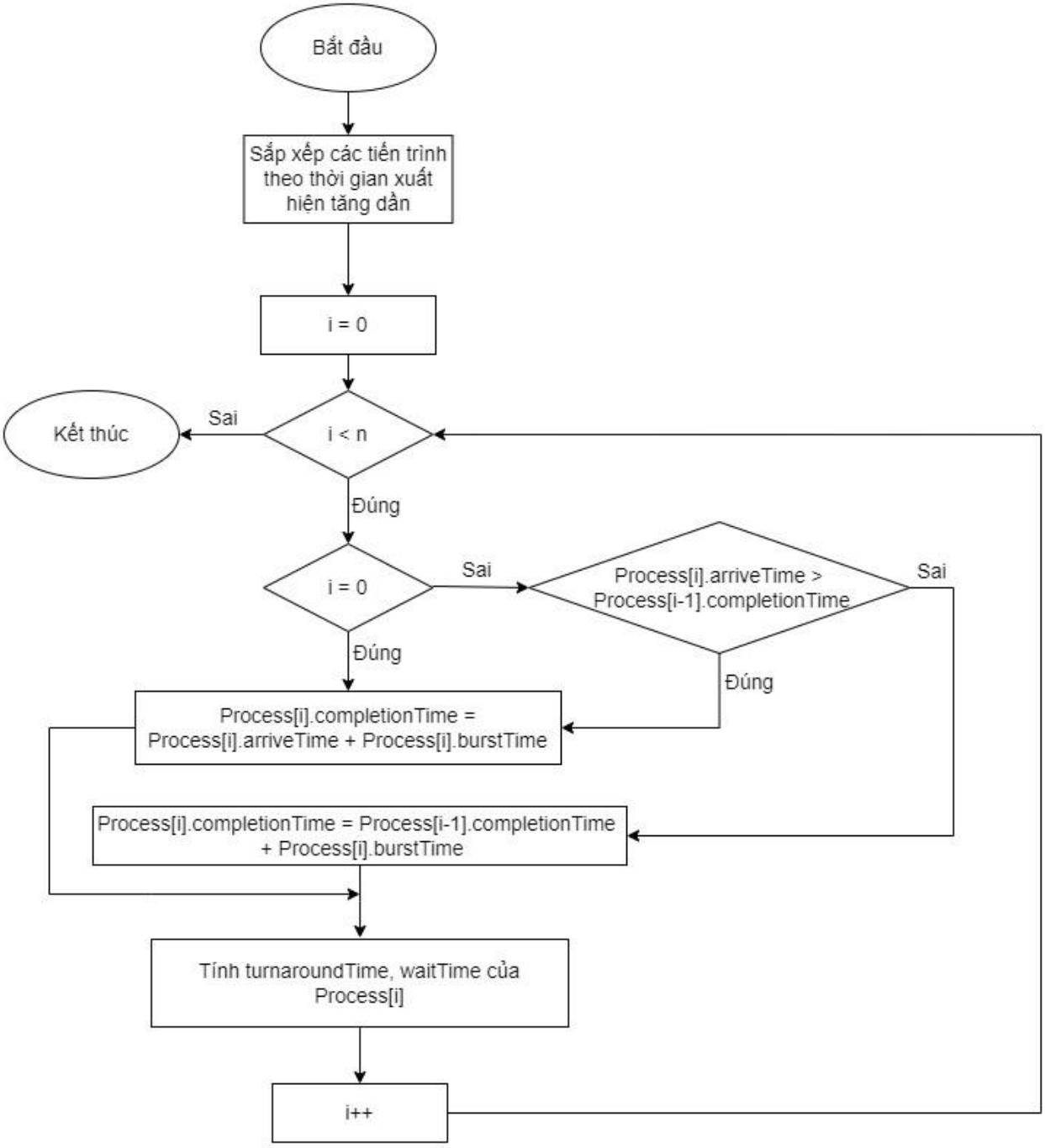
CHƯƠNG 2. THIẾT KẾ VÀ XÂY DỰNG HỆ THỐNG

2.1. Mô hình thuật toán lập lịch

2.1.1. First-Come First-Served (FCFS)

- Nguyên tắc hoạt động: lựa chọn tiến trình nào xuất hiện trước thì được ưu tiên thực hiện trước.
- Input:
    - Thời gian đến (Arrival Time): Thời điểm mà mỗi tiến trình xuất hiện để yêu cầu được thực hiện.
    - Thời gian thực hiện (Burst Time): Thời gian mà mỗi tiến trình yêu cầu để thực hiện công việc của mình.
- Output:
    - Thứ tự hoàn thành của các tiến trình.
    - Biểu đồ gantt mô tả quá trình thực hiện của các tiến trình.
    - Thời gian chờ đợi (Waiting Time): Thời gian mà mỗi tiến trình phải đợi từ khi đến đến khi bắt đầu thực hiện.
    - Thời gian quay vòng (Turnaround Time): Tổng thời gian mà mỗi tiến trình tồn tại trong hệ thống, từ khi đến đến khi hoàn thành.
    - Thời gian hoàn thành (Completion Time): Thời điểm mà mỗi tiến trình hoàn thành thực hiện công việc của mình.
    - Thống kê các thông số: Các thông số tổng quát như thời gian chờ đợi trung bình (avarage wating time), thời gian quay vòng trung bình (avarage turnaround time).

- Sơ đồ khối



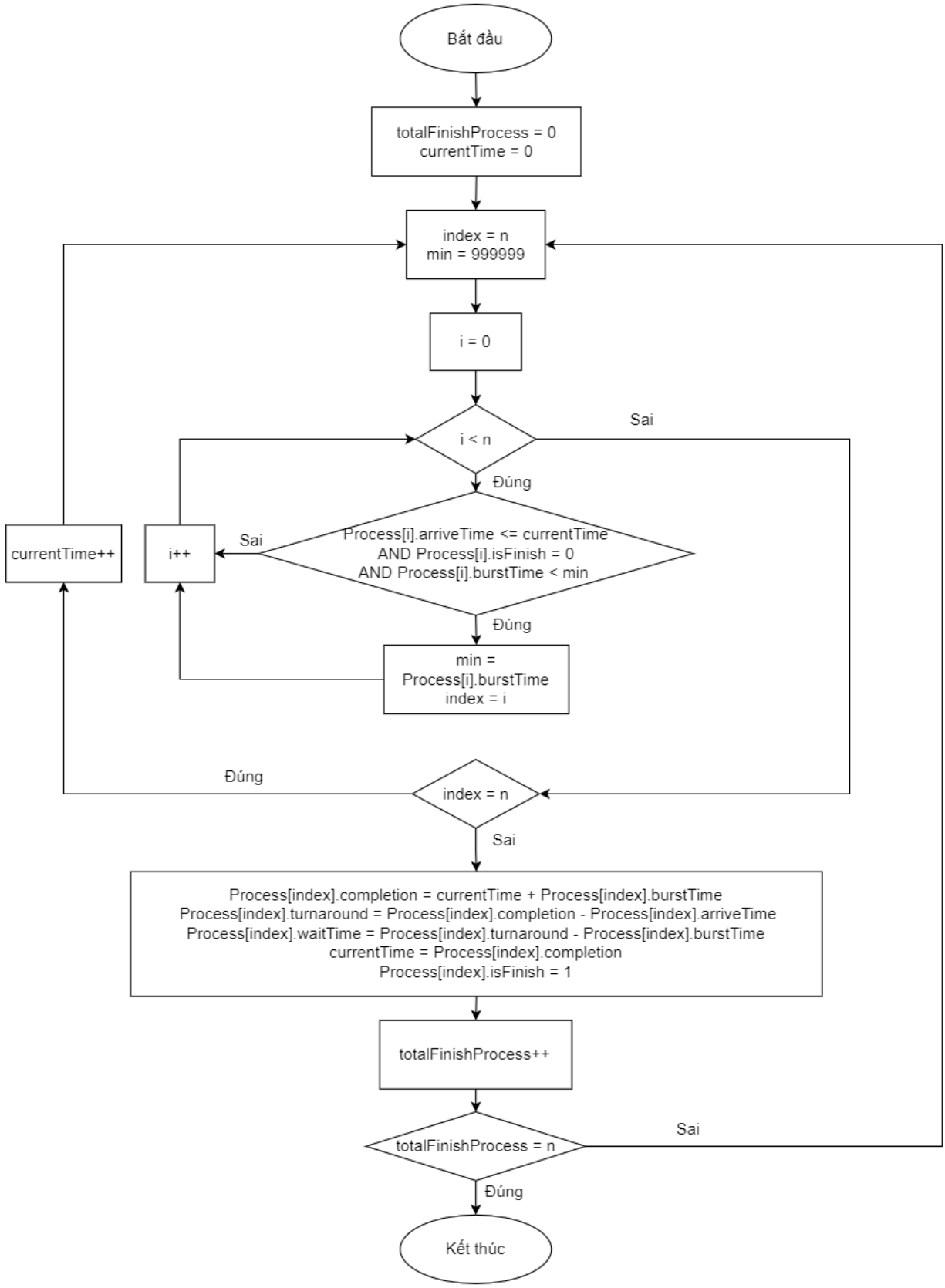
Hình 8: Sơ đồ khối thuật toán FCFS

- Độ phức tạp: O(nlogn) với n là số lượng tiến trình đầu vào.

2.1.2. Shortest Job First (SJF)

- Nguyên tắc hoạt động: lựa chọn tiến trình nào có thời gian thực hiện (burst time) ngắn nhất được ưu tiên thực hiện trước.
- Input:
    - Thời gian đến (Arrival Time): Thời điểm mà mỗi tiến trình xuất hiện để yêu cầu được thực hiện.
    - Thời gian thực hiện (Burst Time): Thời gian mà mỗi tiến trình yêu cầu để thực hiện công việc của mình.
- Output:
    - Thứ tự hoàn thành của các tiến trình.
    - Biểu đồ gantt mô tả quá trình thực hiện của các tiến trình.
    - Thời gian chờ đợi (Waiting Time): Thời gian mà mỗi tiến trình phải đợi từ khi đến đến khi bắt đầu thực hiện.
    - Thời gian quay vòng (Turnaround Time): Tổng thời gian mà mỗi tiến trình tồn tại trong hệ thống, từ khi đến đến khi hoàn thành.
    - Thời gian hoàn thành (Completion Time): Thời điểm mà mỗi tiến trình hoàn thành thực hiện công việc của mình.
    - Thống kê các thông số: Các thông số tổng quát như thời gian chờ đợi trung bình (avarage wating time), thời gian quay vòng trung bình (avarage turnaround time).

- Sơ đồ khối:



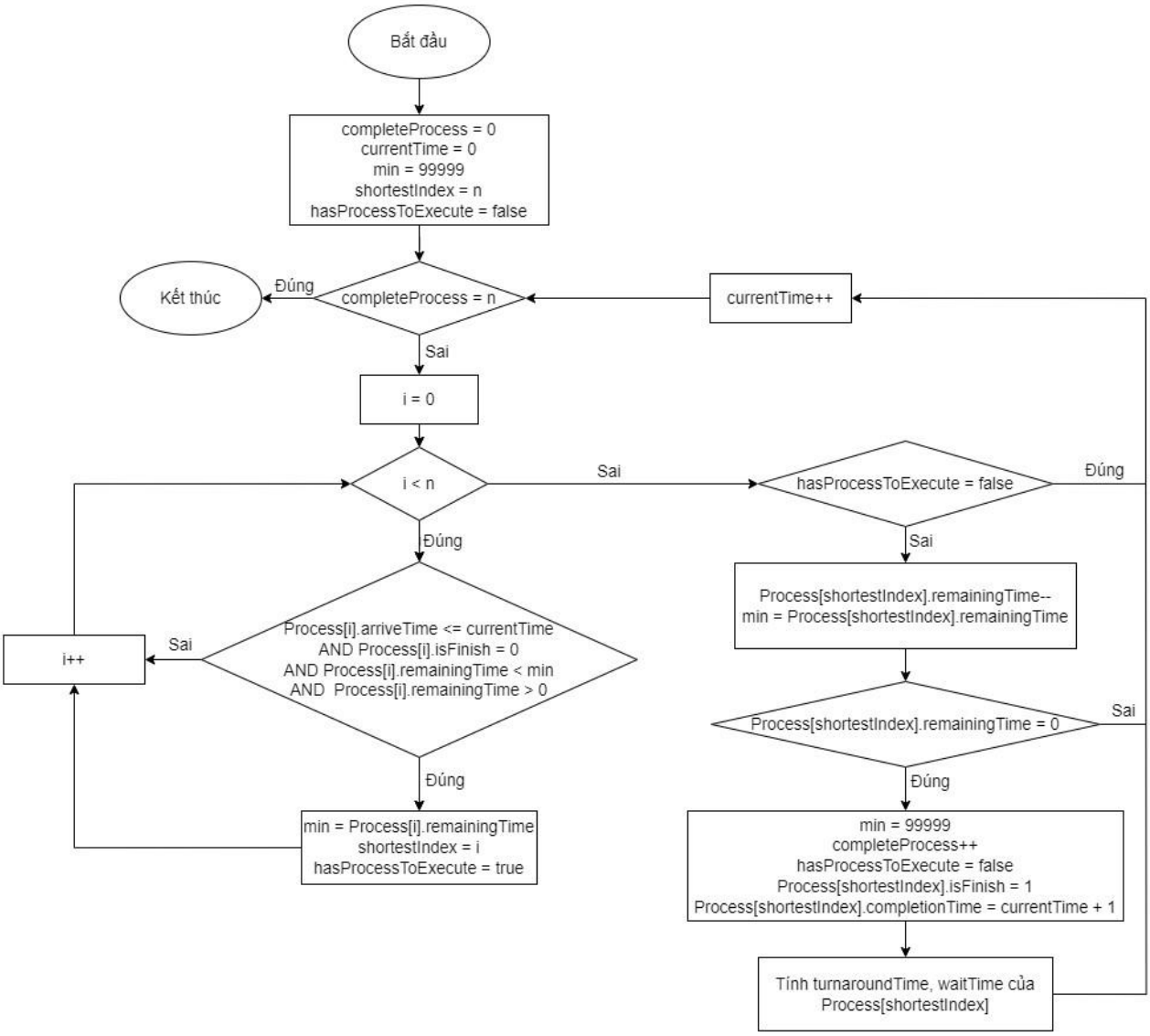
Hình 9: Sơ đồ khối thuật toán Short Job First

- Độ phức tạp: $O(n^2)$ với n là số lượng tiến trình đầu vào

2.1.3. Shortest Remaining Time First (SRTF)

- Nguyên tắc hoạt động: Nếu một tiến trình có thời gian còn lại (remaining time) nhỏ hơn được đưa vào hàng đợi trong khi một tiến trình khác đang chạy, thì tiến trình đang chạy sẽ bị ngắt và tiến trình có thời gian còn lại (remaining time) nhỏ hơn sẽ được ưu tiên thực hiện tiếp theo.
- Input:
    - Thời gian đến (Arrival Time): Thời điểm mà mỗi tiến trình xuất hiện để yêu cầu được thực hiện.
    - Thời gian thực hiện (Burst Time): Thời gian mà mỗi tiến trình yêu cầu để thực hiện công việc của mình.
- Output:
    - Thứ tự hoàn thành của các tiến trình.
    - Biểu đồ gantt mô tả quá trình thực hiện của các tiến trình.
    - Thời gian chờ đợi (Waiting Time): Thời gian mà mỗi tiến trình phải đợi từ khi đến đến khi bắt đầu thực hiện.
    - Thời gian quay vòng (Turnaround Time): Tổng thời gian mà mỗi tiến trình tồn tại trong hệ thống, từ khi đến đến khi hoàn thành.
    - Thời gian hoàn thành (Completion Time): Thời điểm mà mỗi tiến trình hoàn thành thực hiện công việc của mình.
- Thống kê các thông số: Các thông số tổng quát như thời gian chờ đợi trung bình (avarage wating time), thời gian quay vòng trung bình (avarage turnaround time).

- Sơ đồ khối:



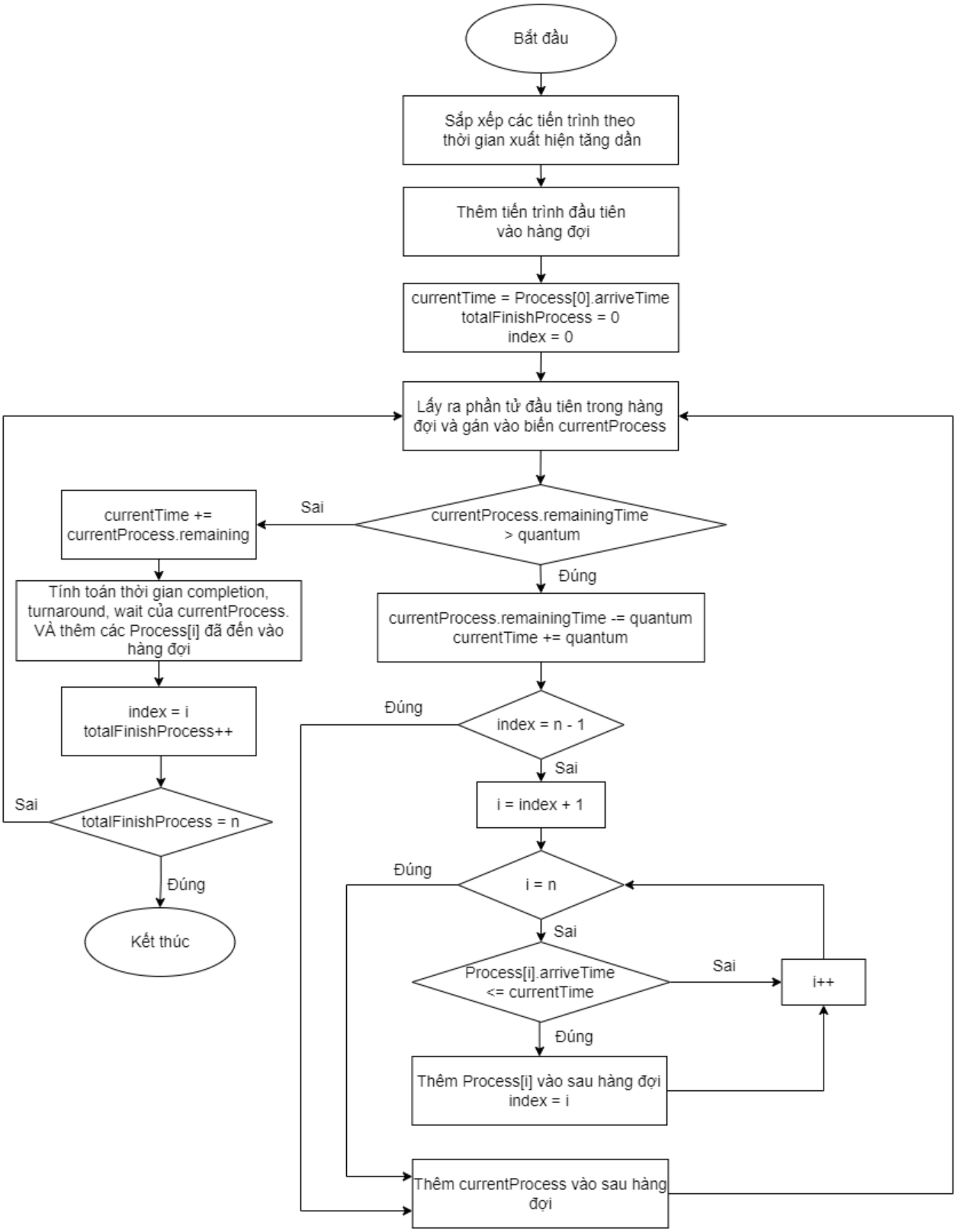
Hình 10: Sơ đồ khối thuật toán SRTF

- Độ phức tạp: $O(n^2)$ với n là số lượng tiến trình đầu vào.

2.1.4. Round Robin

- Nguyên tắc hoạt động: hoạt động bằng cách cấp phát thời gian CPU cho mỗi tiến trình trong một hàng đợi theo vòng tròn. Mỗi tiến trình được thực hiện trong một khoảng thời gian cố định gọi là "quantum" trước khi chuyển sang tiến trình tiếp theo trong hàng đợi. Điều này giúp đảm bảo công bằng và tránh tình trạng ưu tiên quá mức đối với một tiến trình cụ thể.
- Input:
    - Thời gian đến (Arrival Time): Thời điểm mà mỗi tiến trình xuất hiện để yêu cầu được thực hiện.
    - Thời gian thực hiện (Burst Time): Thời gian mà mỗi tiến trình yêu cầu để thực hiện công việc của mình.
    - Thời gian chia tỉ lệ (Time Quantum): Là khoảng thời gian nhỏ nhất mà mỗi tiến trình được phép chạy trước khi hệ thống chuyển qua tiến trình khác.
- Output:
    - Thứ tự hoàn thành của các tiến trình.
    - Biểu đồ gantt mô tả quá trình thực hiện của các tiến trình.
    - Thời gian chờ đợi (Waiting Time): Thời gian mà mỗi tiến trình phải đợi từ khi đến đến khi bắt đầu thực hiện.
    - Thời gian quay vòng (Turnaround Time): Tổng thời gian mà mỗi tiến trình tồn tại trong hệ thống, từ khi đến đến khi hoàn thành.
    - Thời gian hoàn thành (Completion Time): Thời điểm mà mỗi tiến trình hoàn thành thực hiện công việc của mình.
    - Thống kê các thông số: Các thông số tổng quát như thời gian chờ đợi trung bình (avarage wating time), thời gian quay vòng trung bình (avarage turnaround time).

- Sơ đồ khối:



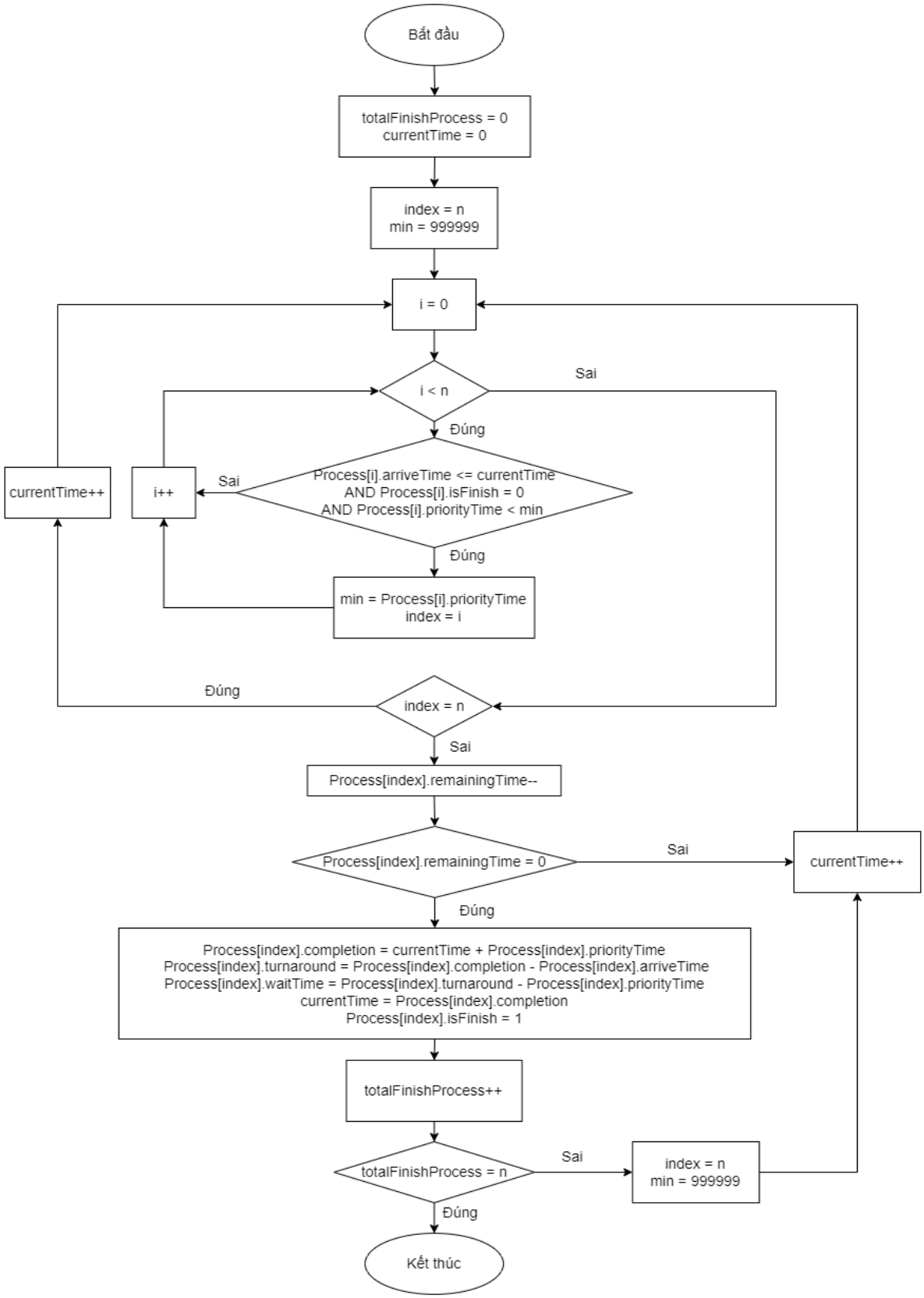
Hình 11: Sơ đồ khối thuật toán Round Robin

- Độ phức tạp: $O(n^2)$ với n là số lượng tiến trình đầu vào.

2.1.5. Priority Preemptive

- Nguyên tắc hoạt động: Nếu một tiến trình có mức ưu tiên cao hơn được đưa vào hàng đợi trong khi một tiến trình khác đang chạy, thì tiến trình đang chạy sẽ bị ngắt và tiến trình có mức ưu tiên cao hơn sẽ được ưu tiên thực hiện tiếp theo.
- Input:
    - Thời gian đến (Arrival Time): Thời điểm mà mỗi tiến trình xuất hiện để yêu cầu được thực hiện.
    - Thời gian thực hiện (Burst Time): Thời gian mà mỗi tiến trình yêu cầu để thực hiện công việc của mình.
    - Độ ưu tiên (Priority): Mức độ ưu tiên của mỗi tiến trình. Độ ưu tiên càng nhỏ, tiến trình sẽ càng được ưu tiên thực hiện.
- Output:
    - Thứ tự hoàn thành của các tiến trình.
    - Biểu đồ gantt mô tả quá trình thực hiện của các tiến trình.
    - Thời gian chờ đợi (Waiting Time): Thời gian mà mỗi tiến trình phải đợi từ khi đến đến khi bắt đầu thực hiện.
    - Thời gian quay vòng (Turnaround Time): Tổng thời gian mà mỗi tiến trình tồn tại trong hệ thống, từ khi đến đến khi hoàn thành.
    - Thời gian hoàn thành (Completion Time): Thời điểm mà mỗi tiến trình hoàn thành thực hiện công việc của mình.
    - Thống kê các thông số: Các thông số tổng quát như thời gian chờ đợi trung bình (avarage wating time), thời gian quay vòng trung bình (avarage turnaround time).

- Sơ đồ khối:



Hình 12: Sơ đồ khối thuật toán Priority Preemptive

- Độ phức tạp: $O(n^2)$ với n là số lượng tiến trình đầu vào.

2.1.6. Priority Non-Preemptive

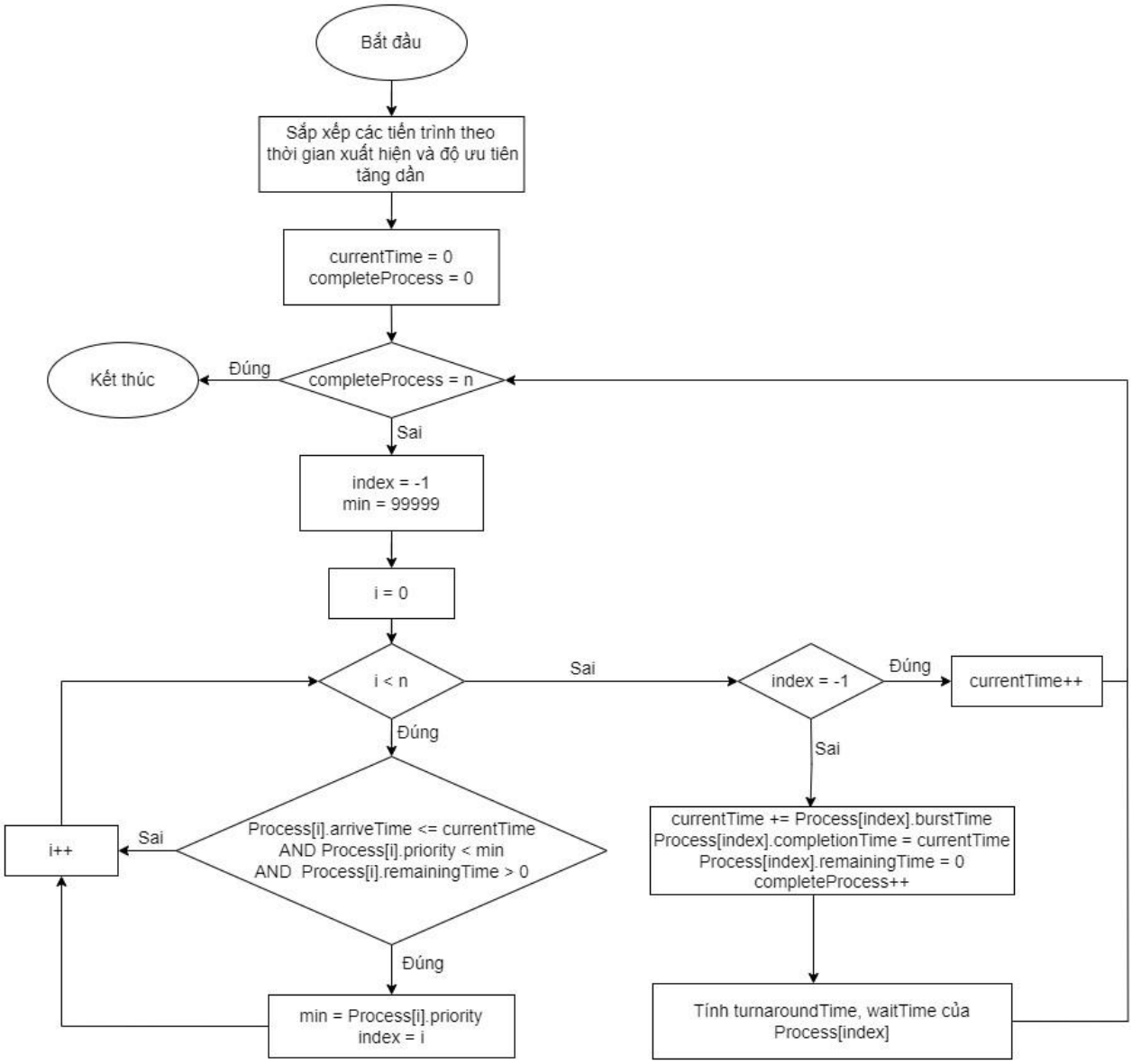
- Nguyên tắc hoạt động: Nếu một tiến trình có mức ưu tiên cao hơn được đưa vào hàng đợi trong khi một tiến trình khác đang chạy, thì tiến trình đang chạy sẽ bị ngắt và tiến trình có mức ưu tiên cao hơn sẽ được ưu tiên thực hiện tiếp theo.
- Input:

- Thời gian đến (Arrival Time): Thời điểm mà mỗi tiến trình xuất hiện để yêu cầu được thực hiện.
- Thời gian thực hiện (Burst Time): Thời gian mà mỗi tiến trình yêu cầu để thực hiện công việc của mình.
- Độ ưu tiên (Priority): Mức độ ưu tiên của mỗi tiến trình. Độ ưu tiên càng nhỏ, tiến trình sẽ càng được ưu tiên thực hiện.

- Output:

- Thứ tự hoàn thành của các tiến trình.
- Biểu đồ gantt mô tả quá trình thực hiện của các tiến trình.
- Thời gian chờ đợi (Waiting Time): Thời gian mà mỗi tiến trình phải đợi từ khi đến đến khi bắt đầu thực hiện.
- Thời gian quay vòng (Turnaround Time): Tổng thời gian mà mỗi tiến trình tồn tại trong hệ thống, từ khi đến đến khi hoàn thành.
- Thời gian hoàn thành (Completion Time): Thời điểm mà mỗi tiến trình hoàn thành thực hiện công việc của mình.
- Thống kê các thông số: Các thông số tổng quát như thời gian chờ đợi trung bình (avarage wating time), thời gian quay vòng trung bình (avarage turnaround time).

- Sơ đồ khối:



Hình 13: Sơ đồ khối thuật toán Priority Non-Preemptive

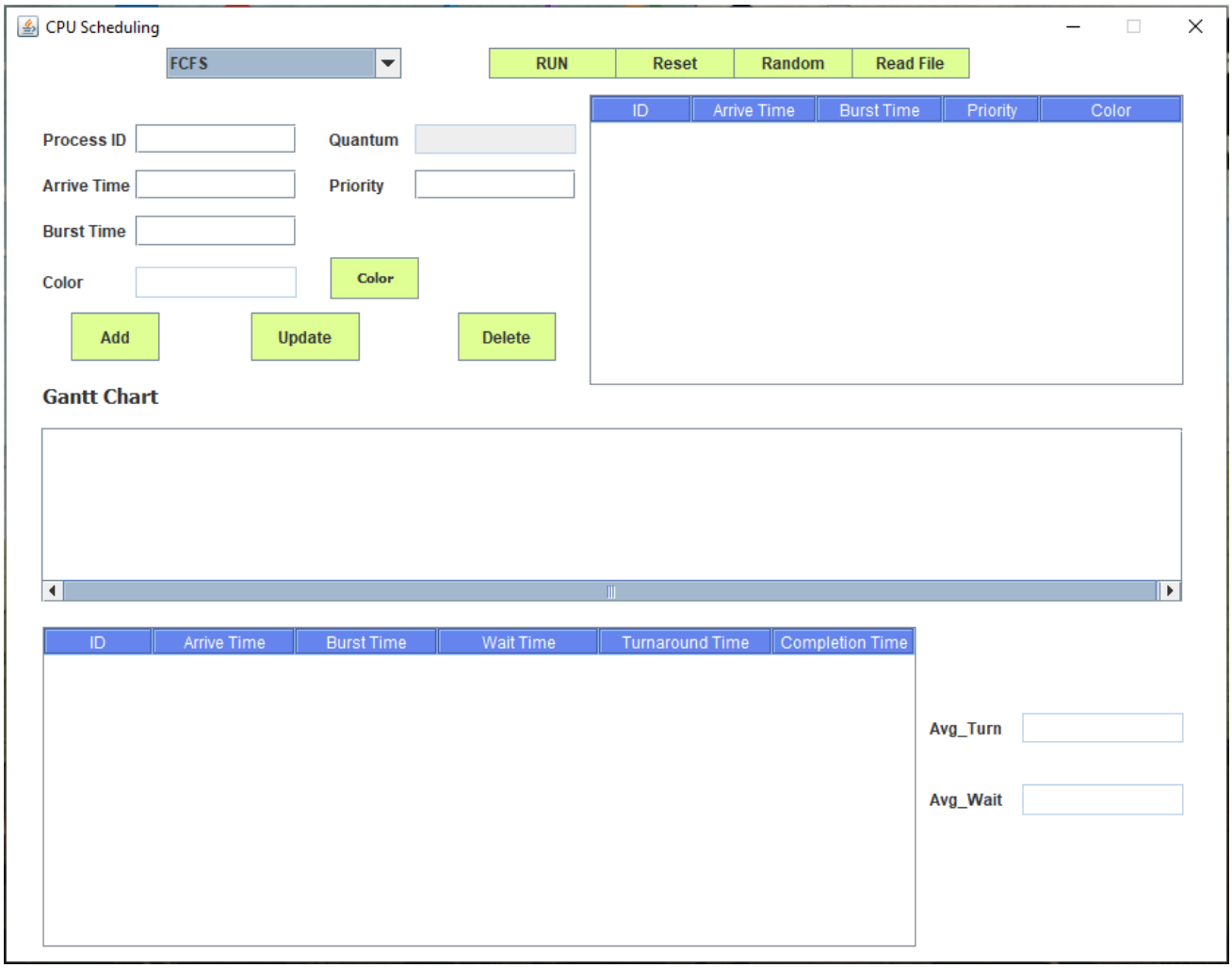
- Độ phức tạp: $O(n^2)$ với n là số lượng tiến trình đầu vào.

CHƯƠNG 3. DEMO ỨNG DỤNG VÀ ĐÁNH GIÁ KẾT QUẢ

3.1. Demo ứng dụng

3.1.1. Giao diện người dùng

- Khi người dùng truy cập vào ứng dụng, dưới đây là giao diện đầu vào của chương trình "mô phỏng các thuật toán lập lịch". Giao diện này được thiết kế dễ sử dụng, các chức năng được sắp xếp.



Hình 14: Giao diện người dùng

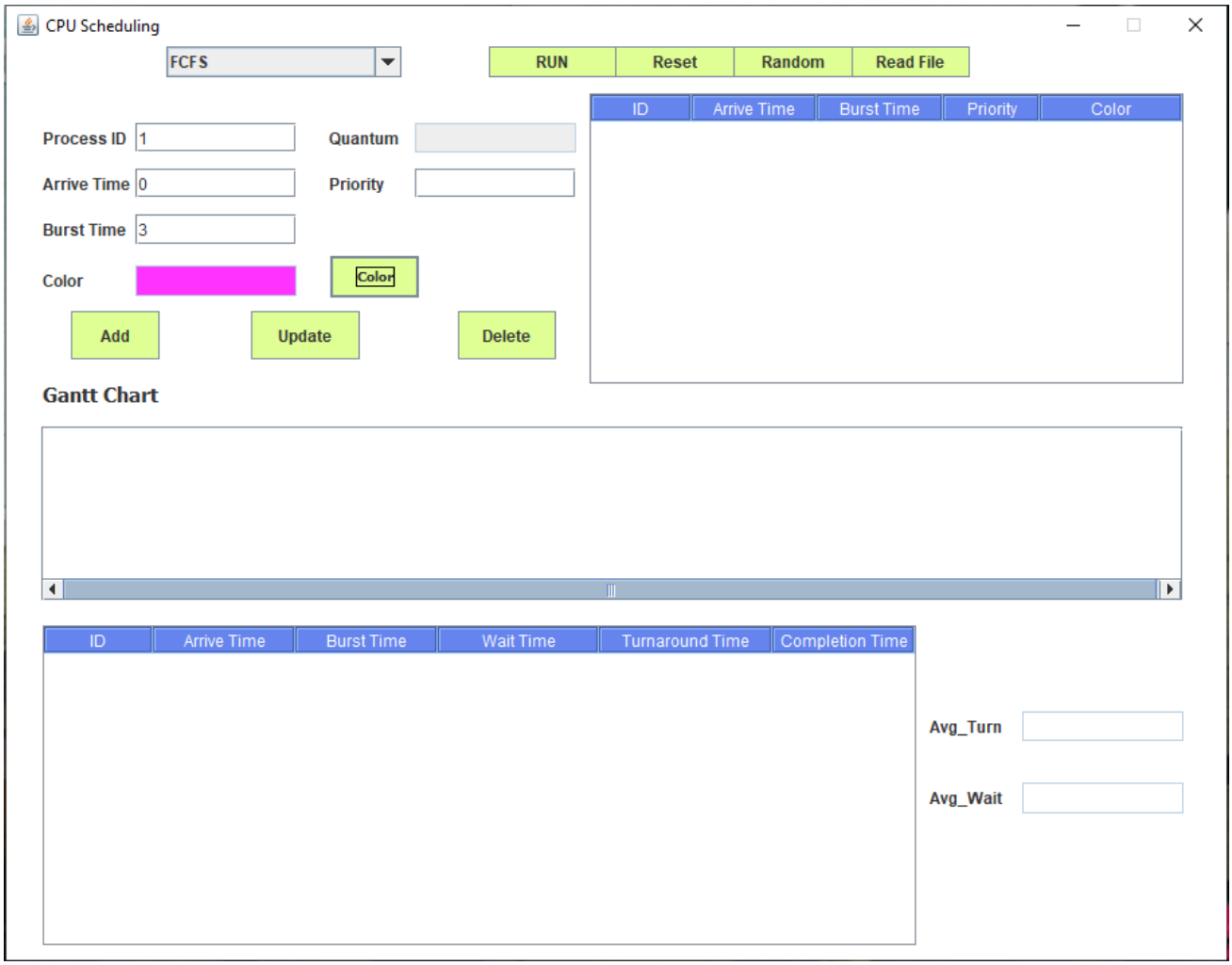
3.1.2. Thêm mới các tiến trình trước khi thực hiện mô phỏng

3.1.2.1. Tự nhập các thông tin

- Người dùng sẽ nhập các thông tin quan trọng của tiến trình. Cụ thể, họ cần nhập các dữ liệu sau:
  - ID: Nhập ID của tiến trình. ID của mỗi tiến trình là khác nhau giúp phân biệt giữa các tiến trình.

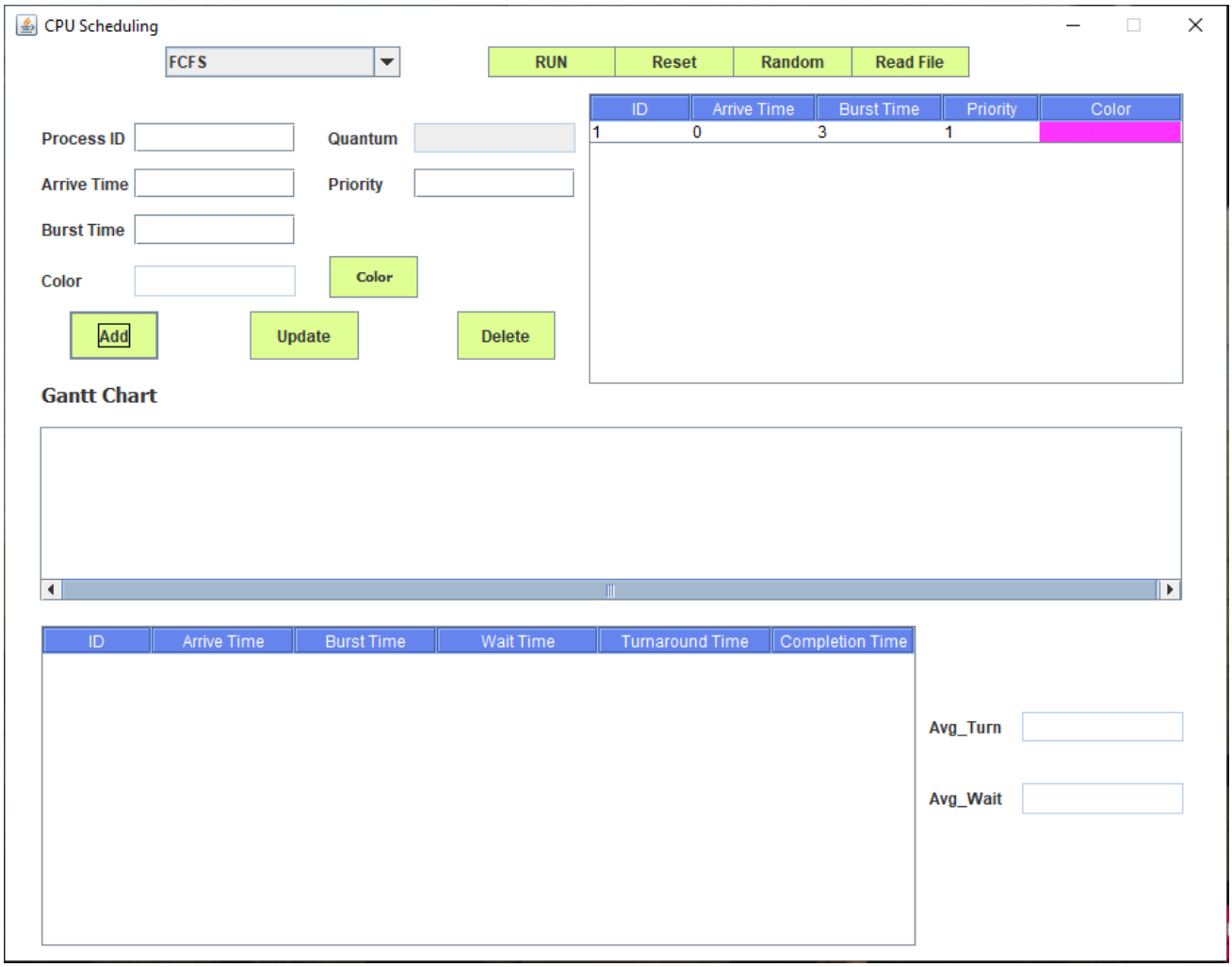
- Thời gian đến (Arrive Time): Thời điểm mà mỗi tiến trình xuất hiện để yêu cầu được thực hiện. Thời gian được nhập theo định dạng là một số nguyên dương (đơn vị giây).
- Thời gian thực hiện (Burst Time): Thời gian mà mỗi tiến trình yêu cầu để thực hiện công việc của mình. Thời gian được nhập theo định dạng là một số nguyên dương (đơn vị giây).
- Màu sắc cho tiến trình (color): Nhấn vào nút color để chọn màu sắc khác nhau cho mỗi tiến trình. Giúp phân biệt rõ ràng giữa các tiến trình khác nhau và làm cho biểu đồ gantt trở nên dễ đọc hơn.
- Quantum: Thời gian tối đa mà một tiến trình có thể sử dụng CPU trước khi bị chuyển đổi với tiến trình khác. Người dùng sẽ được yêu cầu nhập nếu chọn thuật toán Round Robin.
- Priority: độ ưu tiên của một tiến trình (có thể bằng nhau hoặc không giữa các tiến trình). Người dùng sẽ được yêu cầu nếu chọn thuật toán Priority.

- Sau đó, người dùng sẽ nhấn nút "Add" để tiến hành thêm tiến trình.



Hình 15: Giao diện nhập thông tin của tiến trình

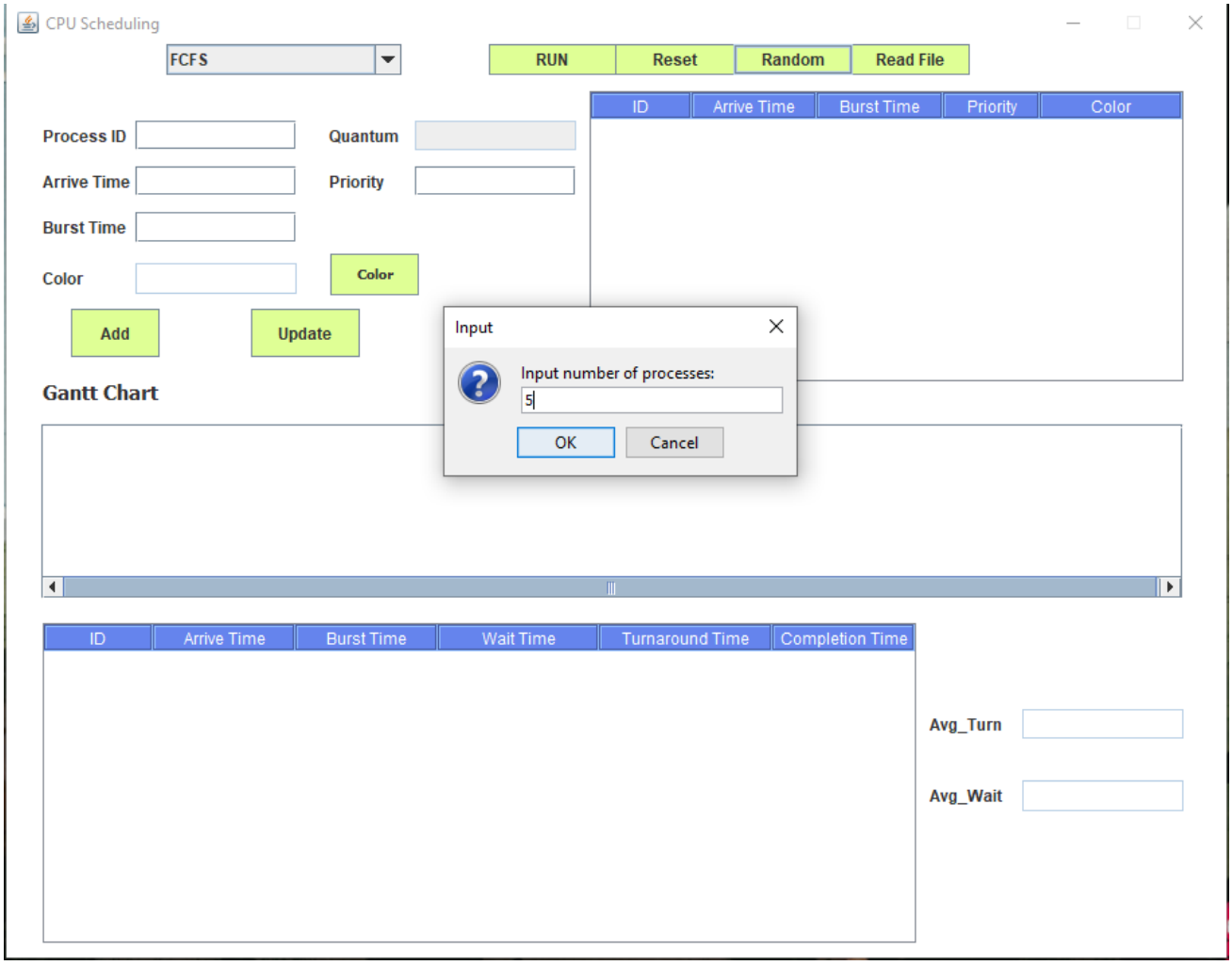
- Sau khi người dùng nhấn nút "Add", hệ thống sẽ kiểm tra thông tin người dùng đã hợp lệ chưa. Nếu nhập thiếu thông tin, nhập sai định dạng, hoặc là nhập trùng ID của tiến trình đã có trước thì người dùng sẽ nhận được thông báo lỗi yêu cầu nhập lại
- Nếu các thông tin nhập hợp lệ thì một tiến trình mới nhập sẽ được thêm vào, người dùng có thể tiếp tục tạo thêm nhiều tiến trình khác bằng cách nhập tương tự.



Hình 16: Giao diện sau khi thêm tiến trình

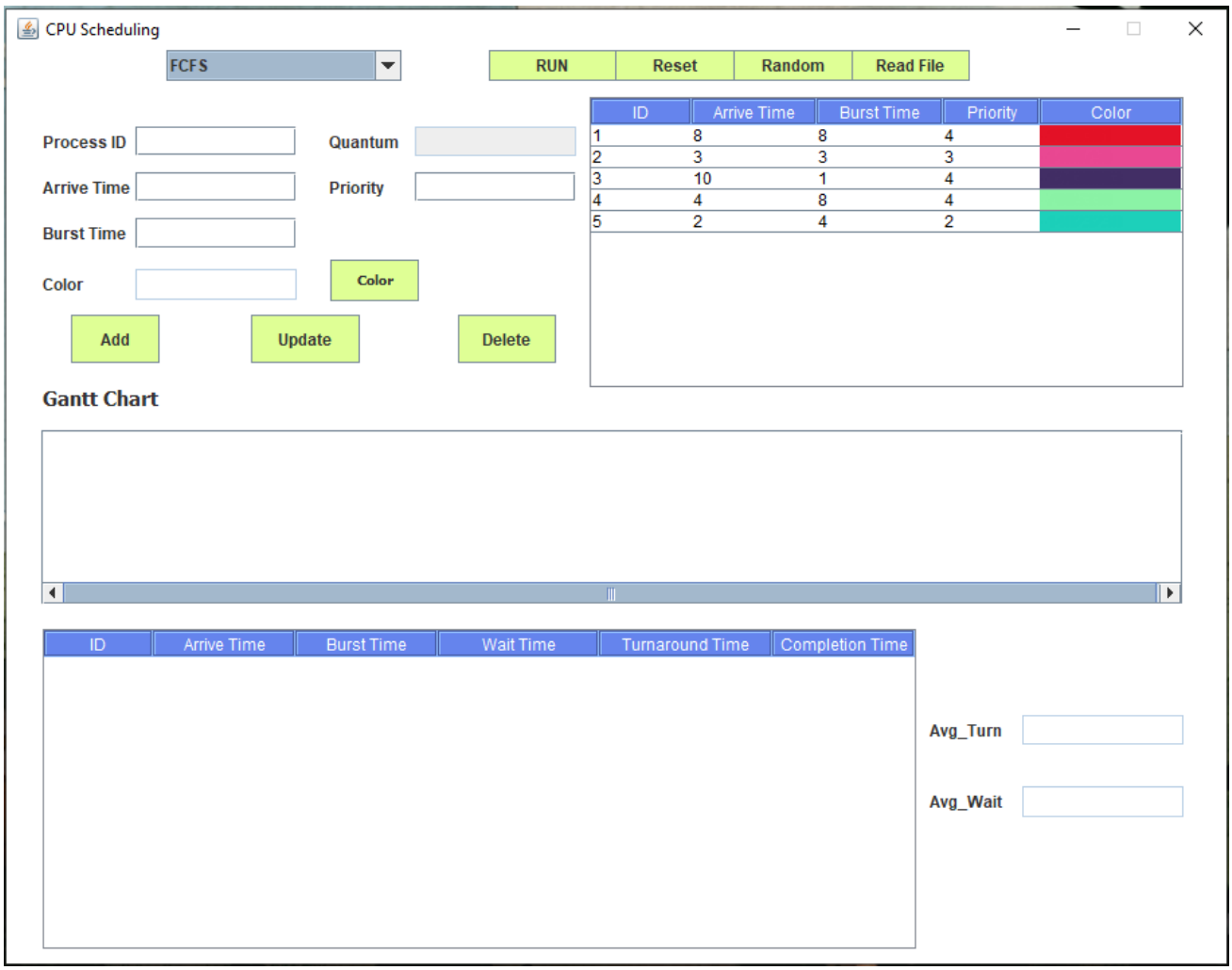
3.1.2.2. Random các thông tin

- Người dùng nhấn nút "Random", hệ thống sẽ hiển thị "message" yêu cầu người dùng nhập số tiến trình muốn tạo (là một số nguyên dương).



Hình 17: Giao diện nhập số lượng tiến trình cần random

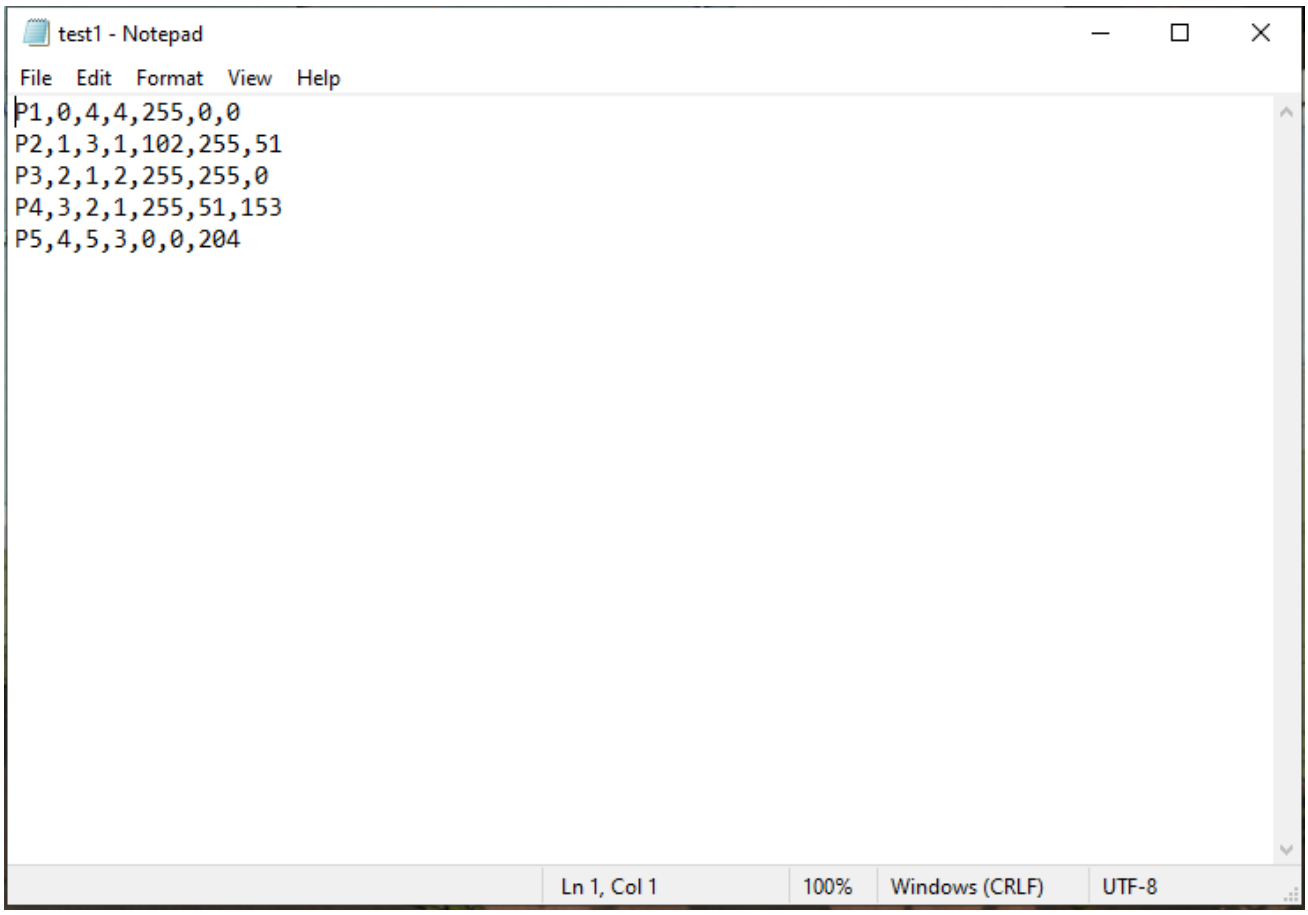
- Nếu thông tin nhập hợp lệ hệ thống sẽ tự động random các thông tin như: ID, Arrive Time, Burst Time, Color, Priority của n tiến trình (n là số tiến trình mà người dùng nhập vào) vào bảng đầu vào như hình dưới đây.



Hình 18: Giao diện sau khi random thành công

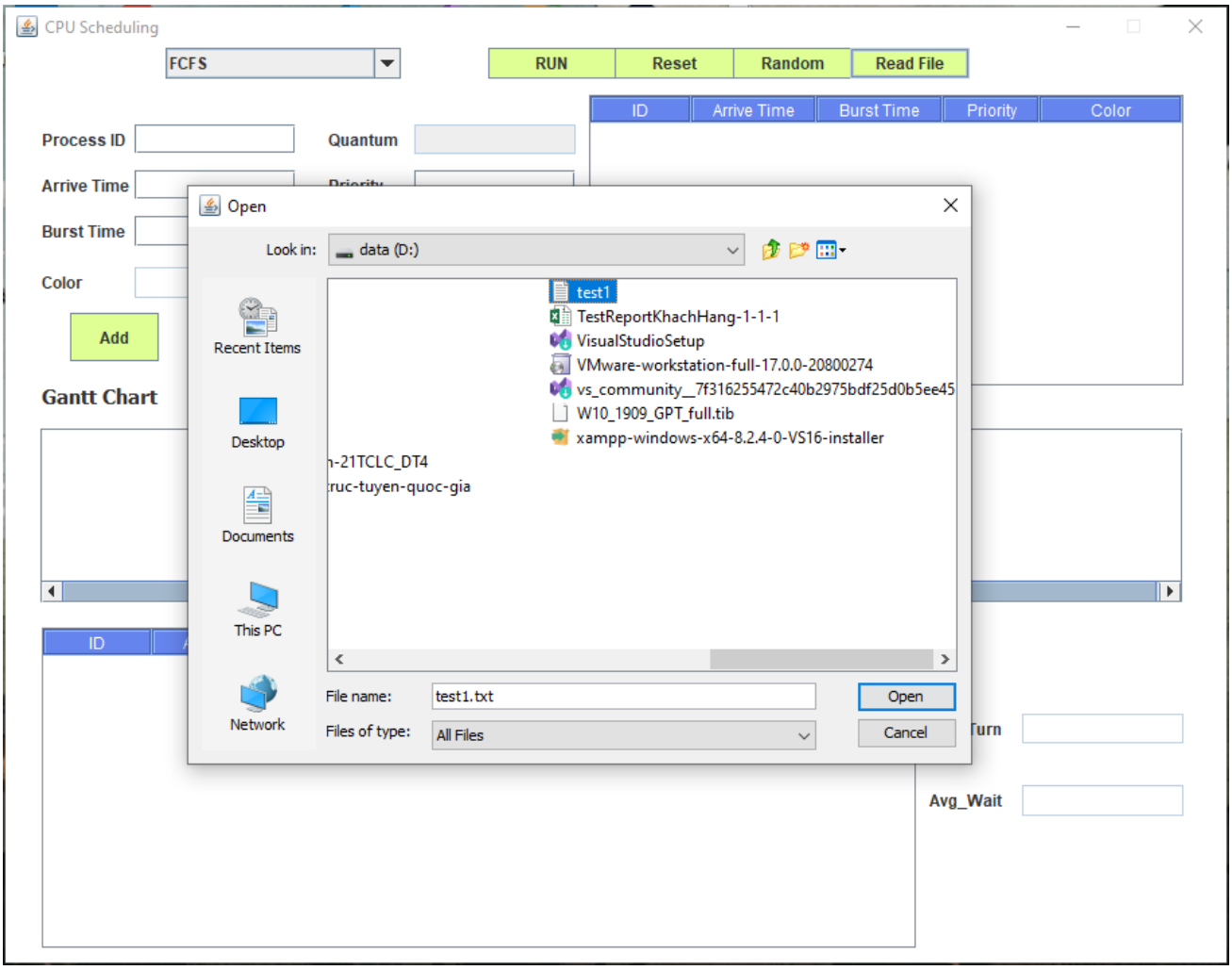
3.1.2.3. Đọc thông tin từ file

- Tạo mới một file sẵn có với các thông tin cần thiết của tiến trình theo thứ tự: Process ID, Arrive Time, Burst Time, Priority, Red, Green, Blue.
- 3 thông số cuối tương ứng với màu RGB của tiến trình.



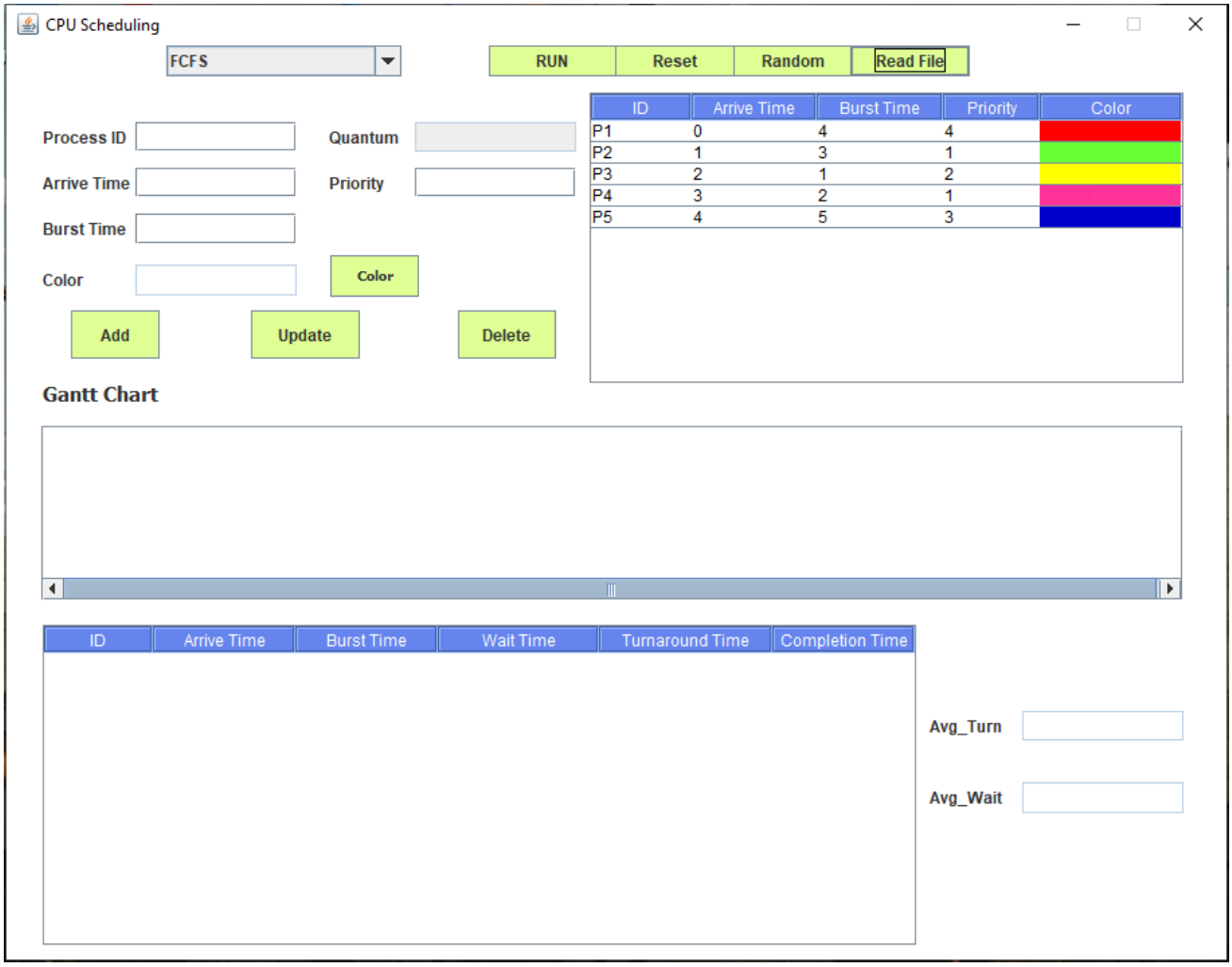
Hình 19: Thông tin file cần đọc

- Người dùng nhấn vào nút "Read File" để chọn một file có sẵn trong máy tính của mình, sau đó nhấn "Open". Nếu chọn file không đúng định dạng, tiến trình sẽ không được thêm vào.



Hình 20: Giao diện chọn file

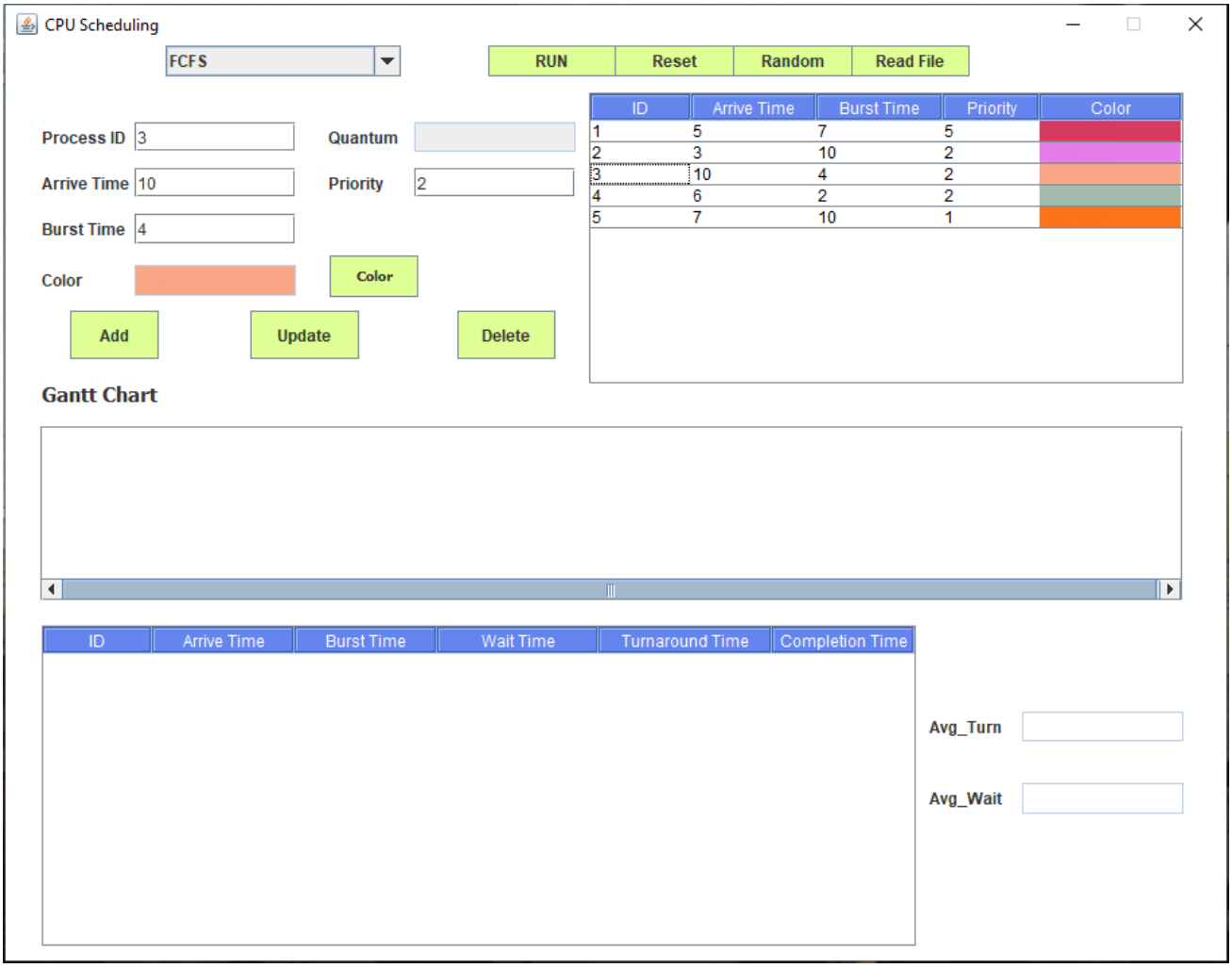
- Nếu chọn file đúng định dạng thông tin của các tiến trình, thì các tiến trình sẽ được thêm vào bảng tiến trình đầu vào.



Hình 21: Giao diện đọc file thành công

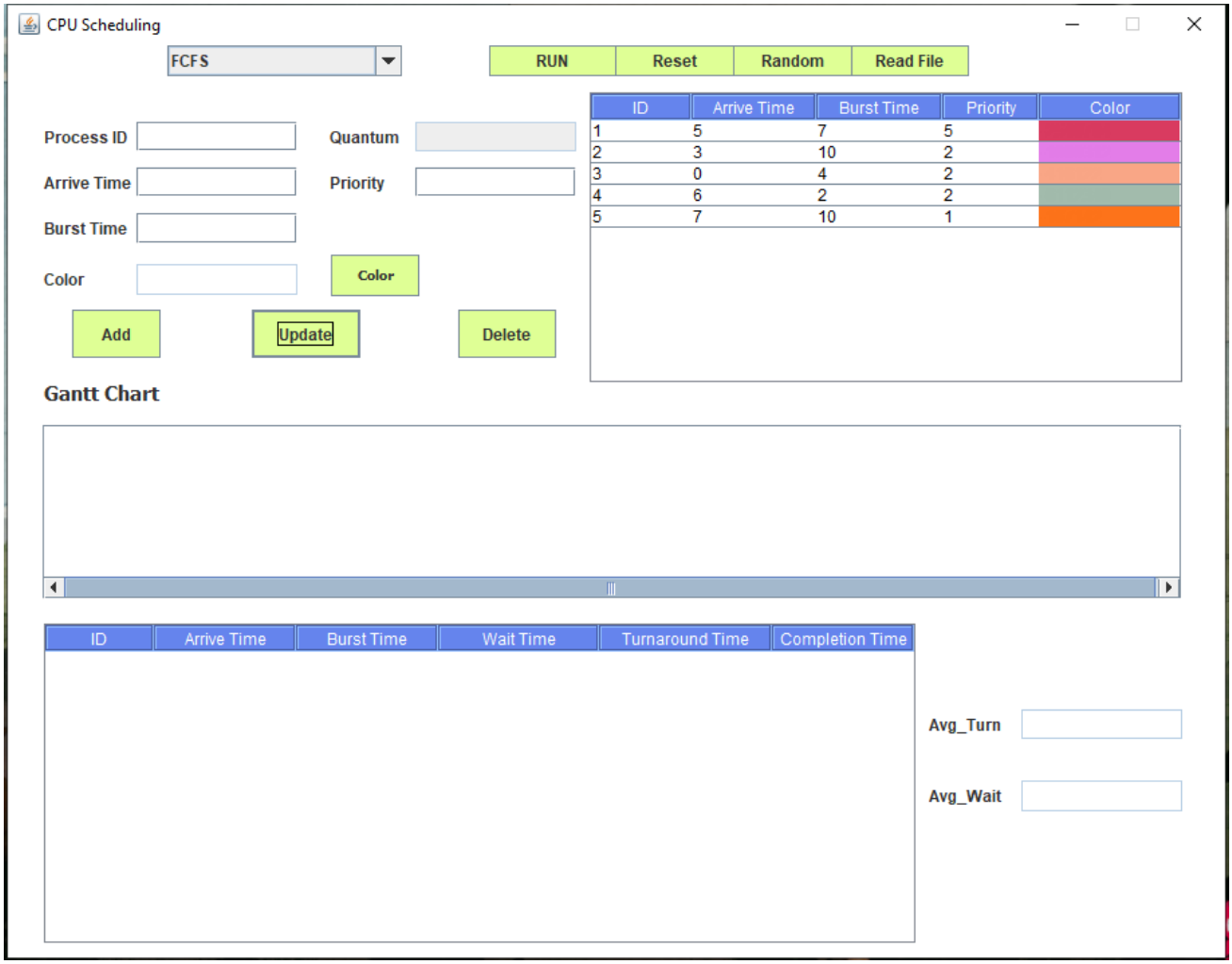
3.1.3. Sửa các thông tin của tiến trình

- Khi chọn một hàng trong bảng tiến trình đầu vào, các thông tin của tiến trình như ID, Arrive Time, Burst Time, Priority, Color sẽ được hiển thị để người dùng chỉnh sửa.



Hình 22: Giao diện chọn một tiến trình cần cập nhật

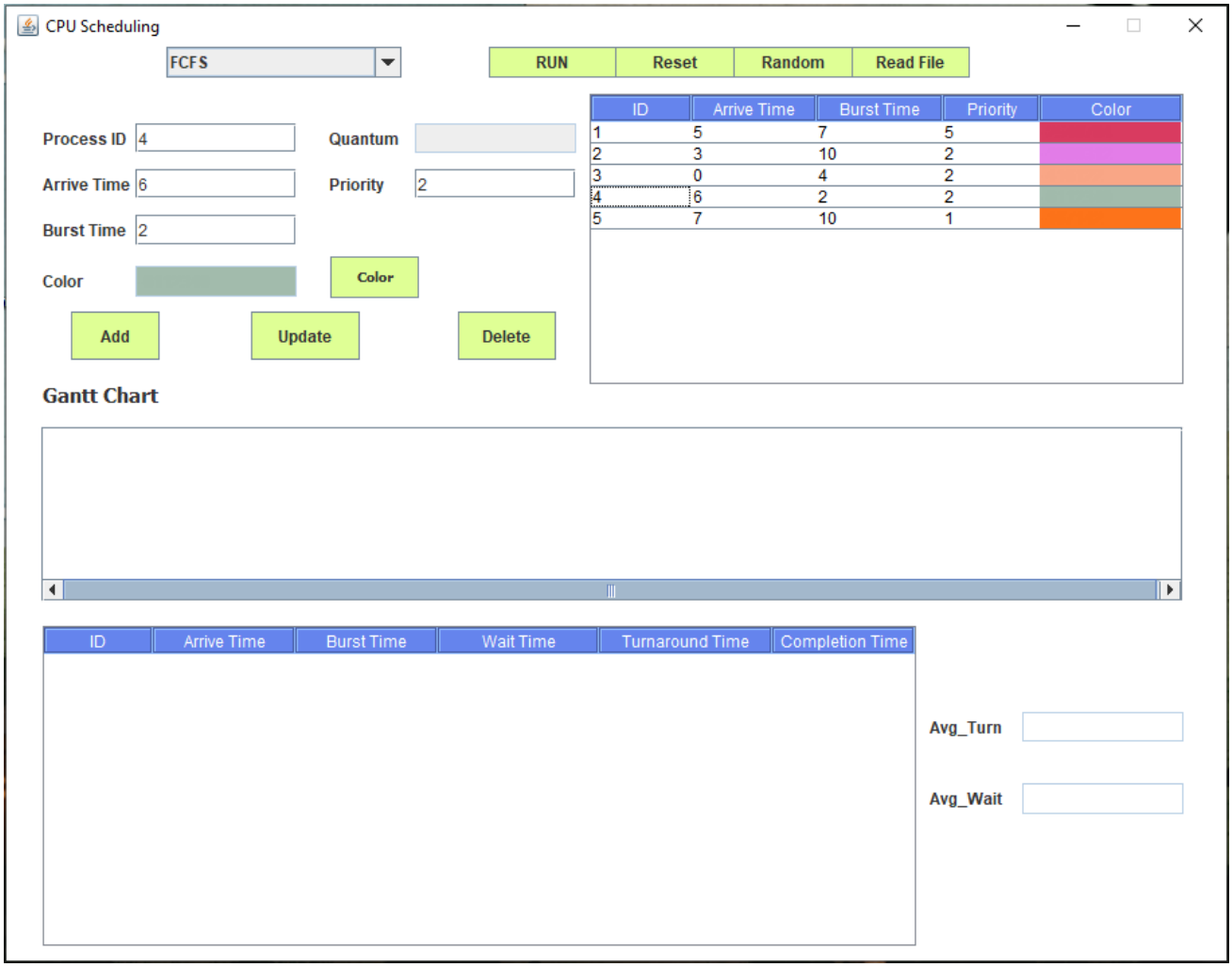
- Sau khi chỉnh sửa các thông tin mong muốn thì người dùng sẽ nhấn vào nút “Update” để cập nhật tiến trình. Thông tin tiến trình sẽ được cập nhật lại trên bảng các tiến trình đầu vào.



Hình 23: Giao diện cập nhật thông tin thành công

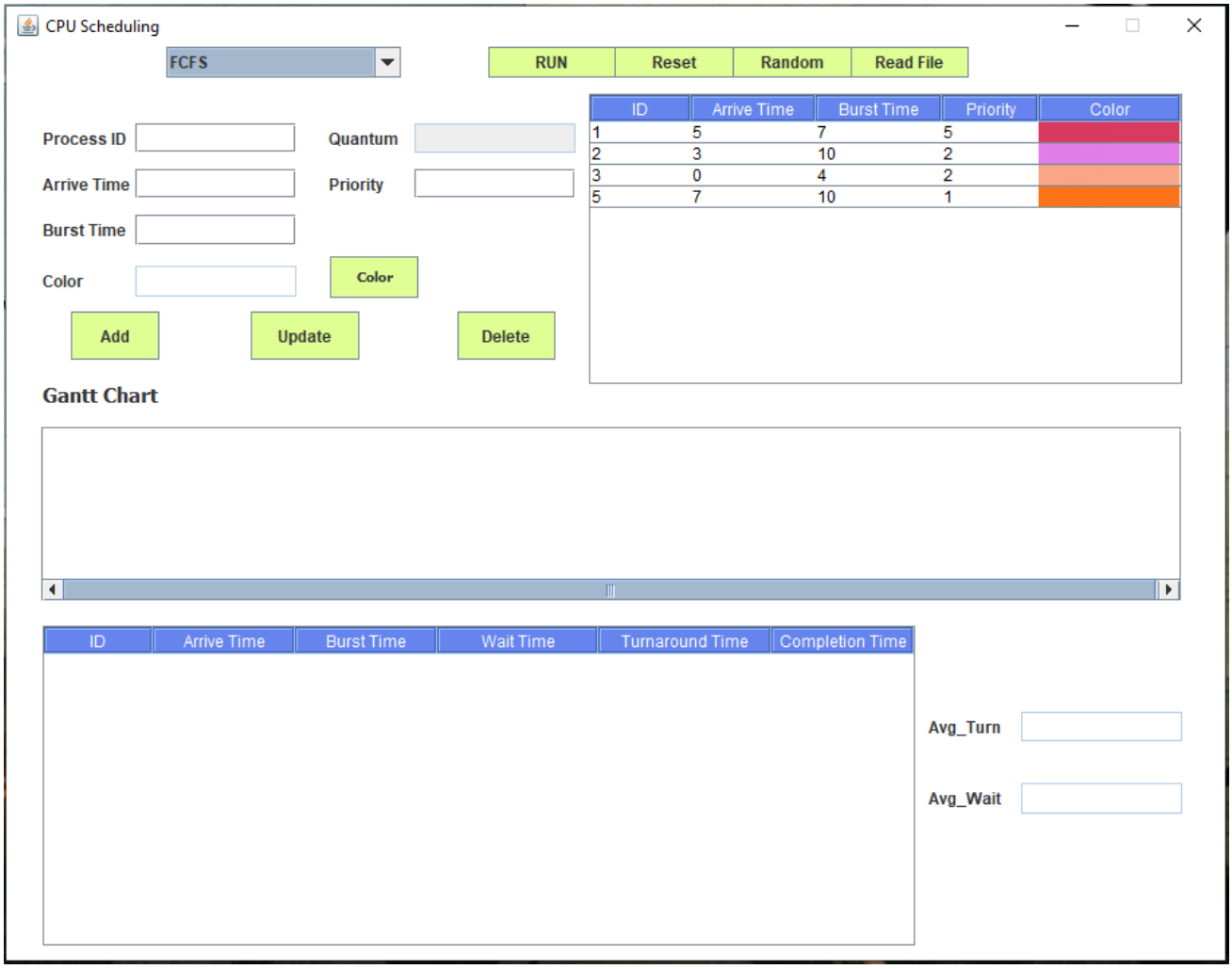
3.1.4. Xoá tiến trình

- Khi muốn xóa một hoặc nhiều tiến trình, người dùng sẽ chọn một hoặc nhiều hàng tương ứng với tiến trình muốn xóa trên bảng các tiến trình đầu vào.



Hình 24: Giao diện chọn tiến trình cần xoá

- Sau đó, người dùng nhấn vào nút “Delete” để xóa các tiến trình đã chọn. Bảng chứa tiến trình đầu vào sẽ tự động xóa các tiến trình đã chọn.

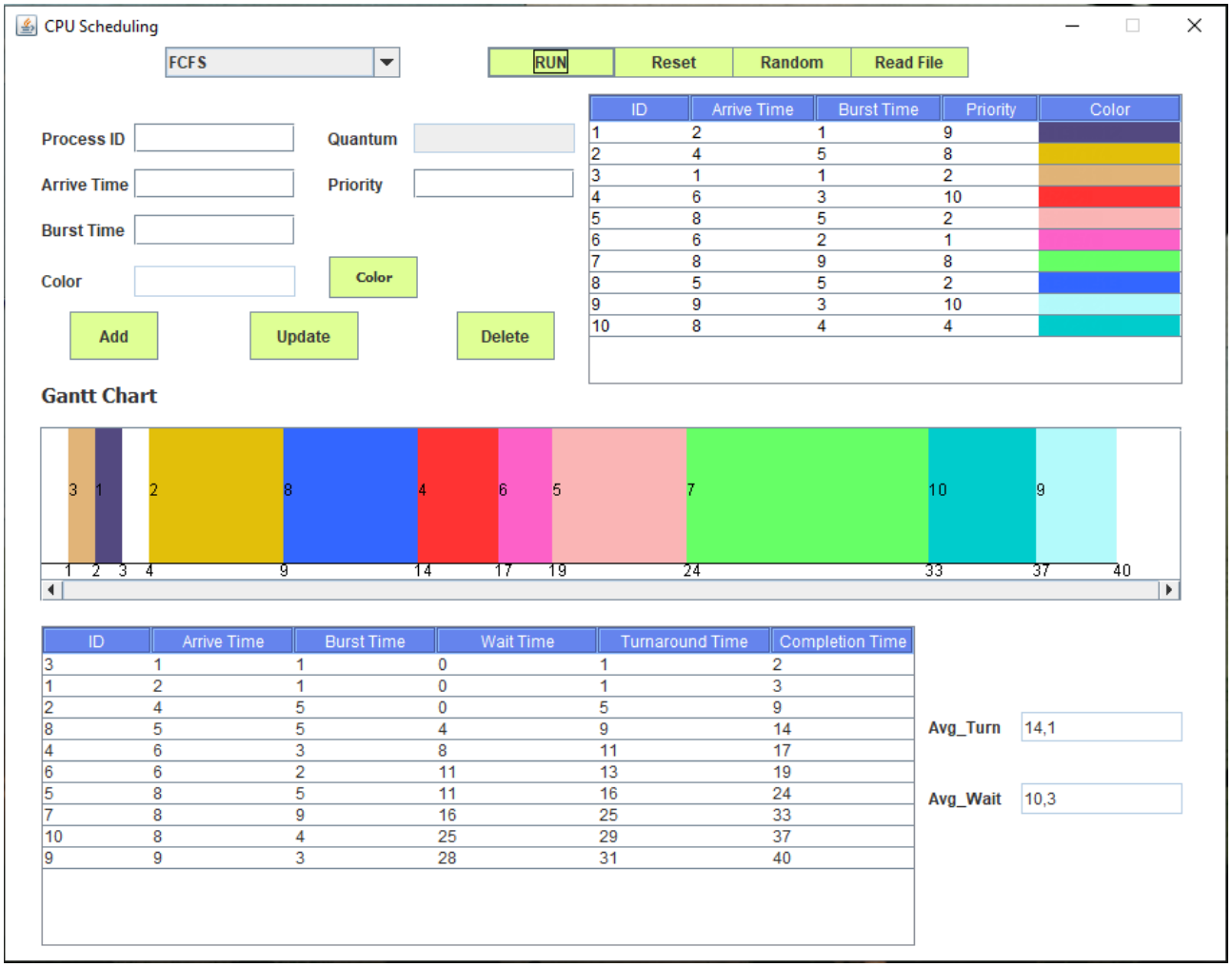


Hình 25: Giao diện sau khi xoá thành công

3.1.5. Mô phỏng các thuật toán lập lịch

3.1.5.1. FCFS

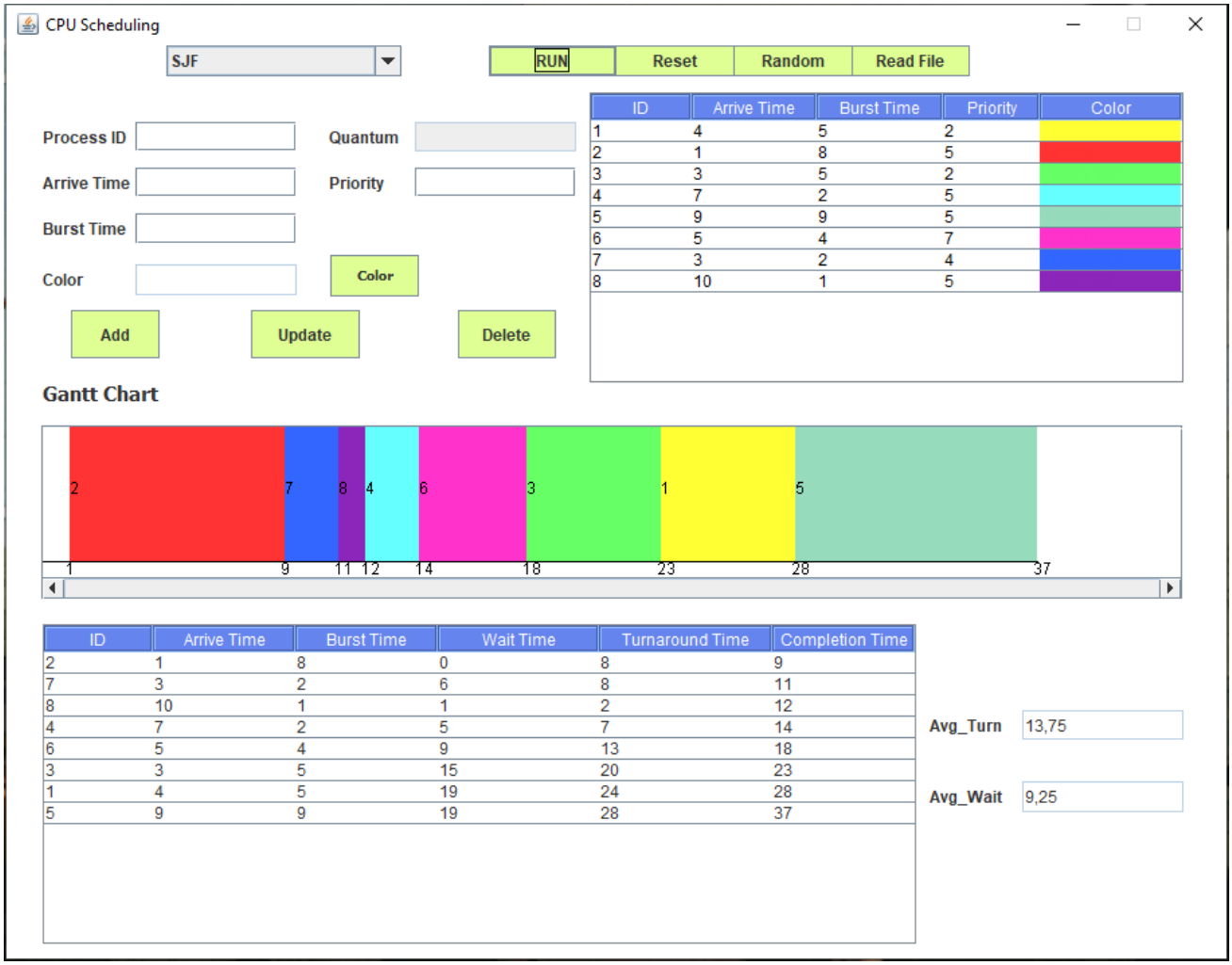
- Người dùng sẽ chọn "FCFS" tương ứng với thuật toán First-Come First Served trong JComboBox.
- Sau đó, nhấn vào nút "RUN" để tiến hành mô phỏng theo thuật toán FCFS.
- Quá trình lập lịch cho tiến trình sẽ được mô phỏng theo thời gian trong JPanel GanttChart.
- Bảng Output bên dưới sẽ tính toán và hiển thị các thông tin của các tiến trình theo thứ tự hoàn thành của chúng như ID, Arrive Time, Burst Time, Wait Time, Turnaround Time, Completion Time.
- Đồng thời, các thông số như thời gian chời đợi trung bình, thời gian xoay vòng trung bình cũng sẽ được tính toán và hiển thị.



Hình 26: Giao diện mô phỏng thuật toán FCFS

3.1.5.2. SJF

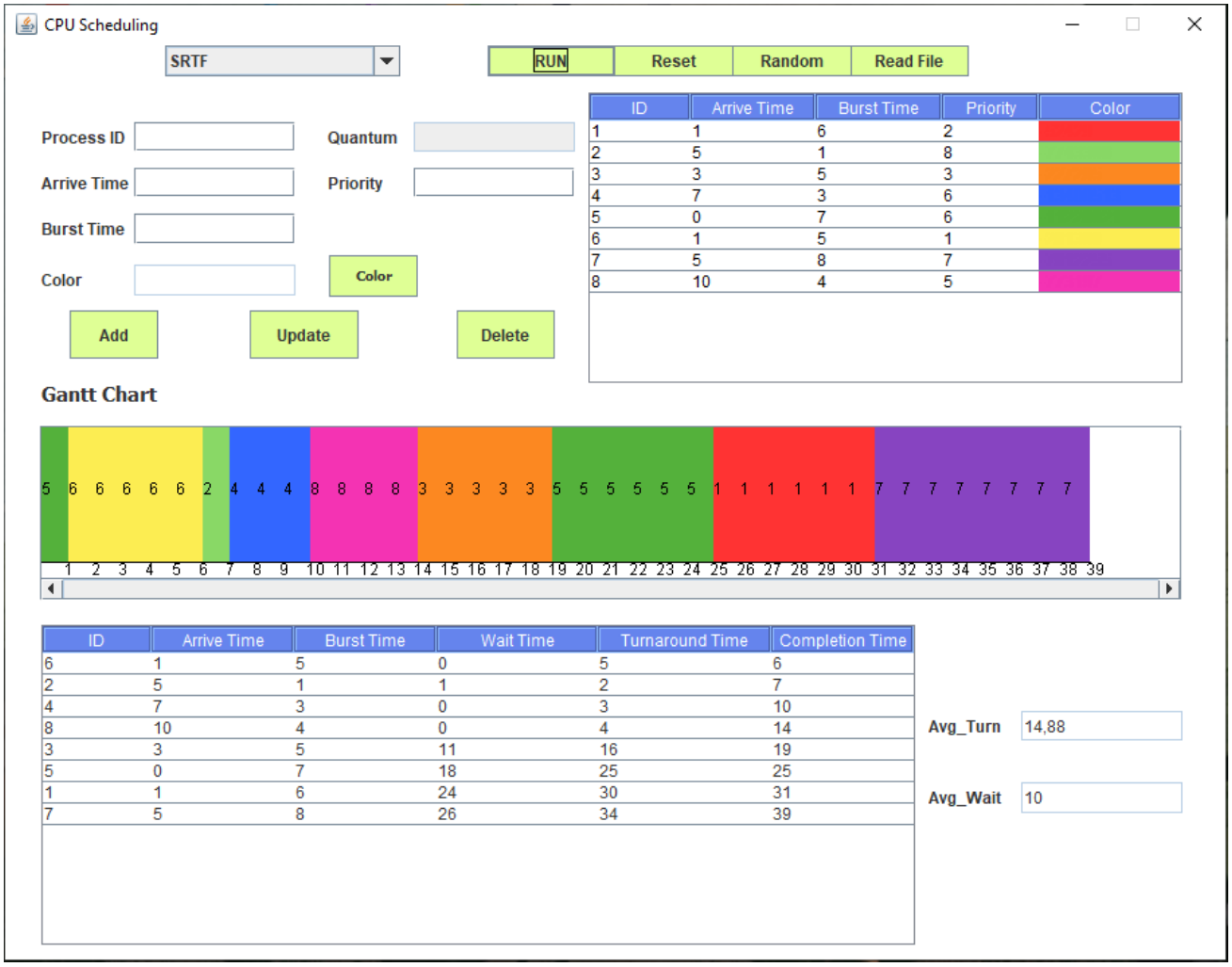
- Người dùng sẽ chọn "SJF" tương ứng với thuật toán Shortest Job First trong JComboBox.
- Sau đó, nhấn vào nút "RUN" để tiến hành mô phỏng theo thuật toán SJF.
- Quá trình lập lịch cho tiến trình sẽ được mô phỏng theo thời gian trong JPanel GanttChart.
- Bảng Output bên dưới sẽ tính toán và hiển thị các thông tin của các tiến trình theo thứ tự hoàn thành của chúng như ID, Arrive Time, Burst Time, Wait Time, Turnaround Time, Completion Time.
- Đồng thời, các thông số như thời gian chời đợi trung bình, thời gian xoay vòng trung bình cũng sẽ được tính toán và hiển thị.



Hình 27: Giao diện mô phỏng thuật toán SJF

3.1.5.3. SRTF

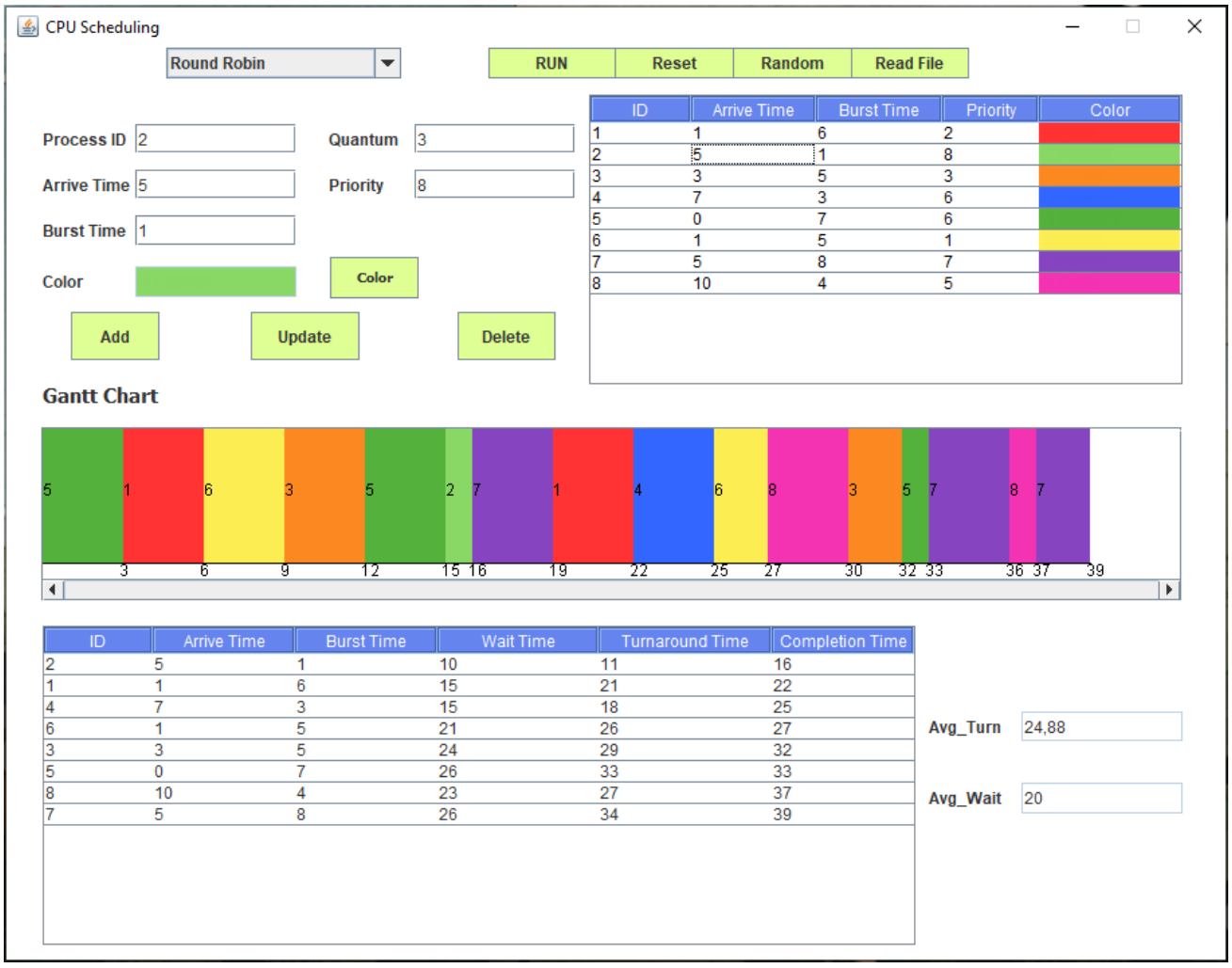
- Người dùng sẽ chọn "SRTF" tương ứng với thuật toán Shortest Remaining Time First trong JComboBox.
- Sau đó, nhấn vào nút "RUN" để tiến hành mô phỏng theo thuật toán SRTF.
- Quá trình lập lịch cho tiến trình sẽ được mô phỏng theo thời gian trong JPanel GanttChart.
- Bảng Output bên dưới sẽ tính toán và hiển thị các thông tin của các tiến trình theo thứ tự hoàn thành của chúng như ID, Arrive Time, Burst Time, Wait Time, Turnaround Time, Completion Time.
- Đồng thời, các thông số như thời gian chời đợi trung bình, thời gian xoay vòng trung bình cũng sẽ được tính toán và hiển thị.



Hình 28: Giao diện mô phỏng thuật toán SRTF

3.1.5.4. Round Robin

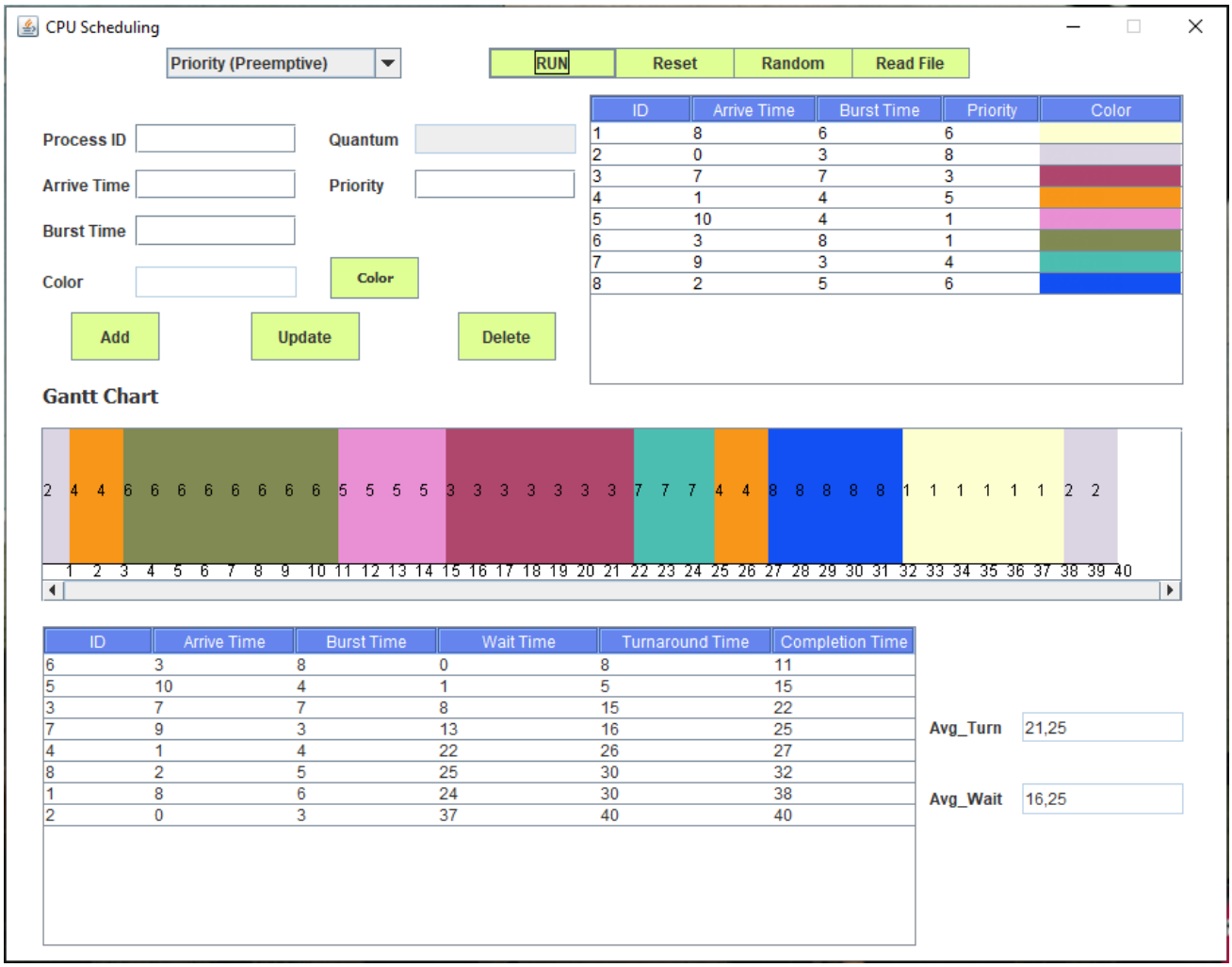
- Người dùng cần nhập thời gian "quantum" cho thuật toán Round Robin.
- Người dùng sẽ chọn "Round Robin" tương ứng với thuật toán Round Robin trong JComboBox.
- Sau đó, nhấn vào nút "RUN" để tiến hành mô phỏng theo thuật toán Round Robin.
- Quá trình lập lịch cho tiến trình sẽ được mô phỏng theo thời gian trong JPanel GanttChart.
- Bảng Output bên dưới sẽ tính toán và hiển thị các thông tin của các tiến trình theo thứ tự hoàn thành của chúng như ID, Arrive Time, Burst Time, Wait Time, Turnaround Time, Completion Time.
- Đồng thời, các thông số như thời gian chời đợi trung bình, thời gian xoay vòng trung bình cũng sẽ được tính toán và hiển thị.



Hình 29: Giao diện mô phỏng thuật toán Round Robin

3.1.5.5. Priority Preemptive

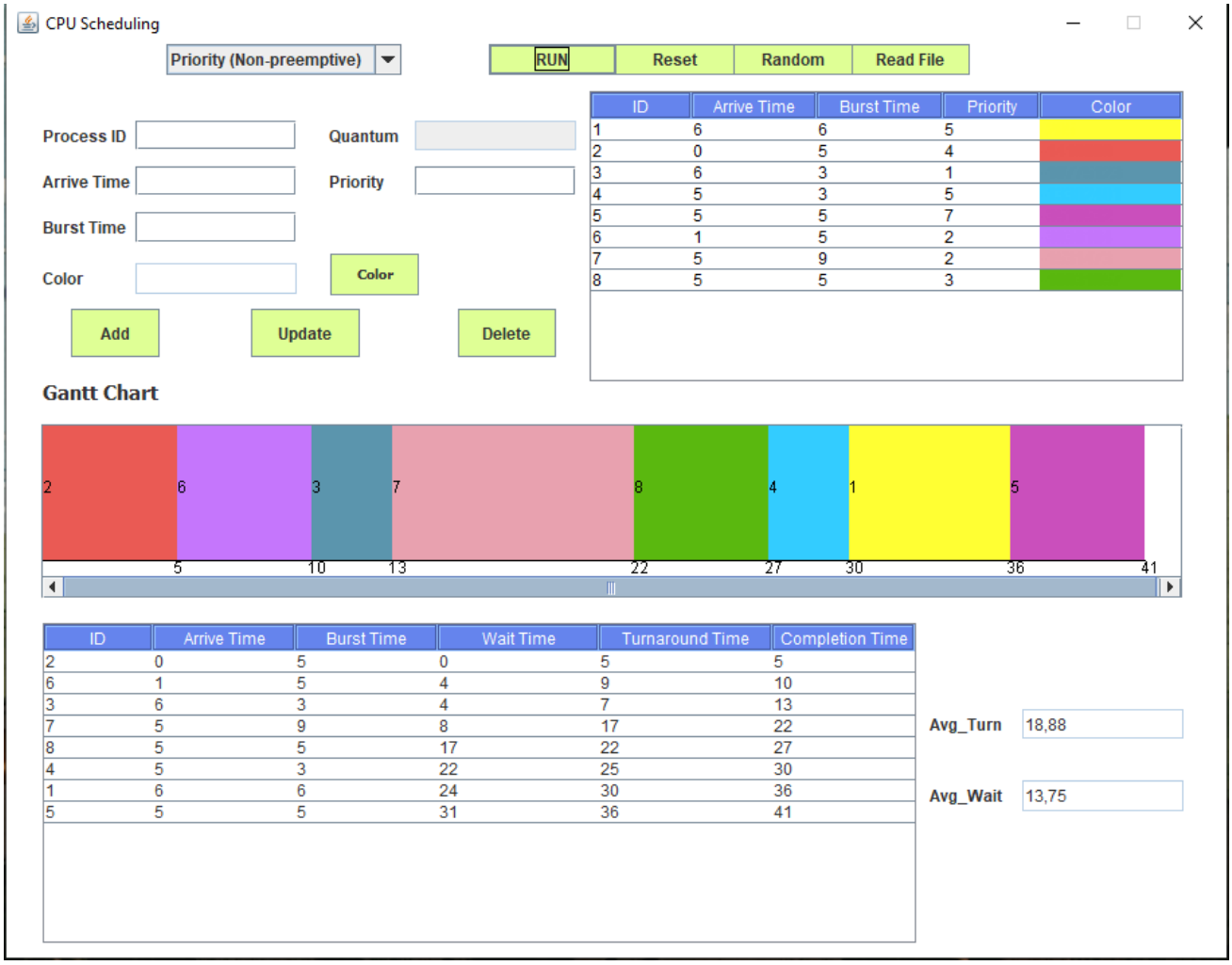
- Người dùng sẽ chọn "Priority Preemptive" tương ứng với thuật toán Priority Preemptive trong JComboBox.
- Sau đó, nhấn vào nút "RUN" để tiến hành mô phỏng theo thuật toán Priority Preemptive.
- Quá trình lập lịch cho tiến trình sẽ được mô phỏng theo thời gian trong JPanel GanttChart.
- Bảng Output bên dưới sẽ tính toán và hiển thị các thông tin của các tiến trình theo thứ tự hoàn thành của chúng như ID, Arrive Time, Burst Time, Wait Time, Turnaround Time, Completion Time.
- Đồng thời, các thông số như thời gian chời đợi trung bình, thời gian xoay vòng trung bình cũng sẽ được tính toán và hiển thị.



Hình 30: Giao diện mô phỏng thuật toán Priority Preemptive

3.1.5.6. Priotity Non-Preemptive

- Người dùng sẽ chọn "Priority Non-Preemptive" tương ứng với thuật toán Priority Non-Preemptive trong JComboBox.
- Sau đó, nhấn vào nút "RUN" để tiến hành mô phỏng theo thuật toán Priority Non-Preemptive.
- Quá trình lập lịch cho tiến trình sẽ được mô phỏng theo thời gian trong JPanel GanttChart.
- Bảng Output bên dưới sẽ tính toán và hiển thị các thông tin của các tiến trình theo thứ tự hoàn thành của chúng như ID, Arrive Time, Burst Time, Wait Time, Turnaround Time, Completion Time.
- Đồng thời, các thông số như thời gian chời đợi trung bình, thời gian xoay vòng trung bình cũng sẽ được tính toán và hiển thị.



Hình 31: Giao diện mô phỏng thuật toán Priority Non-Preemptive

3.2. Đánh giá kết quả

3.2.1. First-Come First-Served (FCFS)

- Ưu điểm:
  - Đơn giản, dễ triển khai.
  - Giờ CPU không bị phân phối lại (không bị ngắt).
  - Chi phí thực hiện thấp vì không phải thay đổi thứ tự ưu tiên thực hiện, thứ tự ưu tiên là thứ tự của tiến trình trong danh sách.
- Nhược điểm:
  - Nếu các tiến trình ở đầu danh sách cần nhiều thời gian để thực hiện thì các tiến trình ở sau sẽ phải chờ đợi rất lâu mới được thực hiện.
  - Mặc dù dễ thực hiện, nhưng nó có hiệu suất kém vì thời gian chờ đợi trung bình cao hơn so với các thuật toán lập lịch trình khác. Có thể xảy ra tình trạng "convoy effect" nếu một tiến trình dài hạn đến trước, làm tăng thời gian chờ cho các tiến trình ngắn hạn khác.
  - Không linh hoạt với các tiến trình có thời gian xử lý khác nhau.
  - Không phù hợp với hệ thống phân chia thời gian.
  - Thích hợp được sử dụng trong các hệ thống xử lí theo lô (batch system).

3.2.2. Shortest Job First (SJF)

- Ưu điểm:
  - Tối ưu hóa thời gian chờ đợi trung bình cho các tiến trình ngắn: Thuật toán SJF sẽ ưu tiên cho các tiến trình có thời gian thực thi ngắn nhất được ưu tiên thực hiện trước. Điều này đảm bảo rằng các tiến trình ngắn sẽ không phải chờ đợi quá lâu để được xử lý, do đó giảm thiểu thời gian chờ đợi trung bình của các tiến trình ngắn.
  - Đảm bảo các tiến trình quan trọng sẽ được xử lý nhanh chóng, ngay cả khi có các tiến trình không quan trọng đang chờ xử lý.
  - Đạt hiệu suất cao với các tiến trình ngắn.
  - Thời gian chờ đợi và quay vòng trung bình tối thiểu.
- Nhược điểm:
  - Có thể dẫn đến tình trạng bỏ đói (starvation), không công bằng đối với các tiến trình dài hạn. Nó phải đợi lâu vì nó đứng sau một chuỗi các tiến trình ngắn hạn, gây giảm hiệu suất.
  - Cần phải biết trước thời gian thực thi của các tiến trình.

- Phức tạp để triển khai vì cần phải cập nhật liên tục thời gian thực thi còn lại của các tiến trình.

3.2.3. Shortest Remaining Time First (SRTF)

- Ưu điểm:
  - Thời gian chờ đợi, tồn tại trong hệ thống của mỗi tiến trình đều ngắn.
  - Tối ưu hóa thời gian chờ: Giảm thiểu thời gian chờ của các tiến trình bằng cách ưu tiên thực hiện tiến trình có thời gian còn lại ngắn nhất.
  - Linh hoạt: Phản ứng nhanh với các tiến trình có thời gian xử lý ngắn.
  - Hiệu suất cao với công việc ngắn: Đối với các công việc có thời gian thực hiện ngắn, SRTF thường cho kết quả tốt vì nó chọn lựa tiến trình có thời gian còn lại ngắn nhất để thực hiện trước.
- Nhược điểm:
  - Chi phí thuật toán cao: Yêu cầu quản lý thời gian còn lại cho mỗi tiến trình và thực hiện kiểm tra liên tục, làm tăng chi phí thuật toán.
  - Không công bằng: Có thể gây ra tình trạng "starvation" cho các tiến trình yêu cầu thời gian xử lý lớn nếu liên tục có các tiến trình ngắn thời gian đến.
  - Phức tạp khi quản lý thời gian còn lại: Việc cập nhật và theo dõi thời gian còn lại cho từng tiến trình đòi hỏi quản lý phức tạp và tăng độ phức tạp của thuật toán.
  - Không phù hợp cho hệ thống thời gian thực cứng: Trong các hệ thống đòi hỏi đáp ứng nhanh chóng và chính xác theo thời gian, SRTF có thể không phù hợp vì nó tập trung vào việc giảm thời gian chờ mà không quan tâm đến đảm bảo đúng thời gian kết thúc.

3.2.4. Round Robin

- Ưu điểm:
  - Công bằng giữa các tiến trình: chia sẻ thời gian CPU đều đặn, không ưu tiên cao hơn cho bất kỳ tiến trình nào.
  - Tính linh hoạt và tương tác cao: phù hợp với nhiều loại tiến trình, bao gồm cả tiến trình ngắn và tiến trình dài, giúp giảm thiểu thời gian chờ đợi.
  - Với việc sử dụng "quantum", giúp giữ cho các tiến trình chuyển đổi liên tục, tạo ra một trải nghiệm tương tác đáng kể.
  - Dễ hiểu: Thuật toán Round Robin dựa trên nguyên tắc đơn giản là cấp phát CPU cho các tiến trình theo vòng.

- Không có nguy cơ "Starvation" nếu quantum không quá lớn: mọi tiến trình đều có cơ hội được thực hiện.

- Nhược điểm:
  - Thời gian chờ đợi có thể tăng: Nếu "quantum" được chọn quá lớn, thời gian chờ đợi của các tiến trình có thể tăng lên.
  - Hiệu suất kém đối với tiến trình dài hạn: Với các tiến trình dài hạn, RR có thể không hiệu quả bằng các thuật toán lập lịch khác như SJF, dẫn đến thời gian chờ đợi lâu.
  - Với các tiến trình dài hạn, RR có thể không hiệu quả bằng các thuật toán lập lịch khác như SJF, dẫn đến thời gian chờ đợi lâu.
  - Không thể ưu tiên các tiến trình quan trọng.
  - Khả năng "Starvation" nếu "quantum" quá lớn.

3.2.5. Priority Preemptive

- Ưu điểm:
  - Đáng tin cậy hơn: Vì một tiến trình không thể chiếm dụng hoàn toàn bộ xử lý.
  - Thời gian phản hồi trung bình cải thiện.
  - Tối ưu hóa thời gian chờ đợi trung bình: đảm bảo rằng các tiến trình quan trọng sẽ không phải chờ đợi quá lâu để được xử lý, do đó giảm thiểu thời gian chờ đợi trung bình của các tiến trình.
  - Có thể ưu tiên các tiến trình quan trọng.
  - Đáng tin cậy hơn: Vì một tiến trình không thể chiếm dụng hoàn toàn bộ xử lý, phương pháp này đáng tin cậy hơn.
  - Giảm độ trễ đáng kể: Với việc chấp nhận và thực hiện tiến trình quan trọng ngay lập tức khi chúng có độ ưu tiên cao, giảm độ trễ và đảm bảo tính nhất quán.

- Nhược điểm:
  - Tiến trình ít được ưu tiên có thể phải bị chờ đợi: luôn bị đẩy xuống đáy hàng đợi do sự xuất hiện liên tục của các tiến trình được ưu tiên hơn.
  - Có thể gây ra lệch lạc thứ tự (aging) khi các tiến trình ưu tiên thấp liên tục nhận được cơ hội.
  - Tốn thời gian: Việc tạm dừng tiến trình đang chạy, điều phối tiến trình mới đến ưu tiên hơn sẽ mất nhiều thời gian hơn so với các phương pháp lập lịch khác.

3.2.6. Priority Non-Preemptive

- Ưu điểm:
  - Đơn giản, dễ triển khai.
  - Ít tính toán phức tạp.
  - Ưu tiên theo mức độ quan trọng: Cho phép ưu tiên thực hiện các tiến trình quan trọng hơn trước, giúp tối ưu hóa mức độ ưu tiên trong hệ thống.
  - Tốc độ thông lượng (throughput) cao.
  - Phản ánh độ ưu tiên của công việc: Phản ánh mức độ quan trọng của công việc trong hệ thống, giúp đảm bảo rằng các công việc quan trọng được thực hiện trước.
  - Phù hợp cho các ứng dụng đặc biệt: Thuận lợi cho các hệ thống yêu cầu ưu tiên công việc theo mức độ quan trọng như trong các ứng dụng y tế, quản lý tài nguyên, v.v...
  - Hiệu suất cao với công việc ngắn: Đối với các công việc có thời gian thực hiện ngắn, SRTF thường cho kết quả tốt vì nó chọn lựa tiến trình có thời gian còn lại ngắn nhất để thực hiện trước.
- Nhược điểm:
  - Nguy cơ đối với "starvation": Các tiến trình có độ ưu tiên thấp có thể bị dời đi liên tục nếu có các tiến trình có độ ưu tiên cao thường xuyên xuất hiện.
  - Không linh hoạt: Thuật toán này không linh hoạt khi đưa ra quyết định và không thể thay đổi ưu tiên của một tiến trình đang thực hiện một khi đã bắt đầu.
  - Không có sự chấm dứt (preemption): Thiếu tính năng chấm dứt có thể dẫn đến việc không tận dụng tối ưu CPU khi xuất hiện tiến trình có độ ưu tiên cao hơn trong quá trình thực hiện.

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

❖ Kết luận:

- Đề tài "Mô phỏng các thuật toán lập lịch" đã tạo nên một hệ thống mô phỏng linh hoạt và chi tiết, mang lại cái nhìn sâu sắc về cách các thuật toán quản lý tiến trình trong hệ điều hành. Nhóm đã tiến hành một hành trình khám phá mô hình lập lịch từ những cơ bản như FCFS, SJF, SRTF, Round Robin đến những mô hình đòi hỏi sự ưu tiên như Priority Non-Preemptive và Priority Preemptive.
- Hệ thống mô phỏng không chỉ giúp hiểu rõ cách mà các thuật toán tương tác với tiến trình, mà còn cung cấp một nền tảng kiểm thử để đánh giá hiệu suất của chúng trong nhiều tình huống. Điều này mang lại giá trị lớn cho cả những người nghiên cứu, những người phát triển hệ điều hành và sinh viên đang tìm hiểu về quản lý tiến trình.
- Hệ thống mô phỏng đã được thiết kế một cách toàn diện và linh hoạt, bao gồm nhiều thuật toán lập lịch quan trọng và biến thể. Điều này mang lại cái nhìn đầy đủ về cách các thuật toán tương tác với tiến trình trong hệ điều hành.
- Giao diện người dùng được thiết kế một cách trực quan, giúp người dùng tương tác một cách dễ dàng với hệ thống. Tính năng mô phỏng cung cấp một trải nghiệm thực tế và hỗ trợ người học hiểu rõ về lập lịch.
- Các thuật toán lập lịch là một trong những vấn đề quan trọng nhất trong hệ điều hành, quyết định hiệu quả sử dụng tài nguyên CPU và chất lượng dịch vụ của hệ thống.
- Mỗi thuật toán lập lịch đều có những ưu điểm và nhược điểm riêng. Việc lựa chọn thuật toán lập lịch phụ thuộc vào các yêu cầu cụ thể của hệ thống, bao gồm các yếu tố như:
    - Thời gian thực thi của các tiến trình.
    - Mức độ ưu tiên của các tiến trình.
    - Khả năng ưu tiên của các tiến trình.

❖ Hướng phát triển

- Mở rộng môi trường mô phỏng:
  - Phát triển khả năng mô phỏng môi trường đa người dùng để hiển thị tương tác giữa các hàng đợi và tiến trình.
- Tích hợp nhiều thuật toán tới:
  - Thêm vào hệ thống các thuật toán mới và biến thể để mở rộng phạm vi mô phỏng.
- Tích hợp công nghệ thực tế:
  - Kết hợp với các công nghệ hiện đại như trí tuệ nhân tạo và machine learning để tối ưu hóa tự động việc lựa chọn thuật toán dựa trên điều kiện thực tế.
- Cải tiến giao diện người dùng:
  - Tối ưu hóa giao diện người dùng để cung cấp thông tin chi tiết và biểu đồ tương tác.
- Tương tác với các ngôn ngữ lập trình:
  - Tích hợp với các ngôn ngữ lập trình phổ biến khác nhau, giúp người dùng thực hành và hiểu rõ cách lập trình ảnh hưởng đến quá trình lập lịch.
- Hỗ trợ tính năng đa nền tảng:
  - Xây dựng ứng dụng có khả năng chạy trên nhiều hệ điều hành và môi trường, từ máy tính cá nhân đến thiết bị di động, tối ưu hóa trải nghiệm người dùng.
- Kết hợp với ứng dụng thực tế:
  - Tích hợp hệ thống với các dự án hoặc ứng dụng thực tế để mô phỏng lập lịch trong môi trường thực tế.
- Hỗ trợ ngôn ngữ đa dạng:
  - Mở rộng hệ thống để hỗ trợ đa ngôn ngữ, giúp người dùng từ các quốc gia và vùng lãnh thổ khác nhau sử dụng ứng dụng mô phỏng một cách thuận tiện.

TÀI LIỆU THAM KHẢO

[1] Andrew S. Tanenbaum, Modern Operating Systems 4th Edition, Pearson, 2014

[2] Vũ Lê Hùng, Giáo Trình Nguyên Lý Hệ Điều Hành, Đại học Bách Khoa Hồ Chí Minh

[3] Trần Hồ Thủy Tiên, Giáo Trình Nguyên Lý Hệ Điều Hành, Đại học Đà Nẵng, Trường đại học Bách Khoa, Khoa Công Nghệ Thông Tin