|  |
| --- |
| HỌC VIỆN KỸ THUẬT MẬT MÃ  **KHOA CÔNG NGHỆ THÔNG TIN**  ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯  Logo HvKTMM |
| BÀI TẬP MÔN HỌC NGUYÊN LÝ HỆ ĐIỀU HÀNH  **VIẾT CHƯƠNG TRÌNH BẰNG NGÔN NGỮ TỰ CHỌN MÔ TẢ THUẬT TOÁN LẬP LỊCH CHO CPU KHÔNG ƯU TIÊN THEO THUẬT TOÁN RR (ROUND ROBIN)** |
| Ngành: Công nghệ thông tin  Chuyên ngành: Kỹ thuật phần mềm nhúng và di động  Sinh viên thực hiện:  Phương Văn Sơn  Trịnh Đức Trường  Vũ Hồng Phúc  Lớp: AT160245  Người hướng dẫn:  **TS. Nguyễn Đào Trường**  Khoa Công nghệ thông tin – Học viện Kỹ thuật mật mã  Hà Nội, 2021 |

**NHẬN XÉT VÀ CHO ĐIỂM CỦA GIÁO VIÊN**

# mục lục

[LỜI NÓI ĐẦU 1](#_Toc78549787)

[Chương 1 TỔNG QUAN 1](#_Toc78549788)

[1.1 Đặt lịch CPU là gì?. 1](#_Toc78549789)

[1.1.1 Nội dung nhỏ thứ nhất của 1.1 1](#_Toc78549790)

[1.1.2 Nội dung nhỏ thứ hai của 1.1 1](#_Toc78549791)

[1.2 Các loại thuật toán đặt lịch. 1](#_Toc78549792)

[1.2.1 Nội dung nhỏ thứ nhất của 1.2 1](#_Toc78549793)

[1.2.2 Nội dung nhỏ thứ hai của 1.2 1](#_Toc78549794)

[Chương 2 THUẬT TOÁN LẬP LỊCH CHO CPU KHÔNG ƯU TIÊN ROUND ROBIN 3](#_Toc78549796)

[2.1 Giới thiệu về thuật toán đặt lịch RR. 3](#_Toc78549797)

[2.1.1 Nội dung nhỏ thứ nhất của 2.1 3](#_Toc78549798)

[2.1.2 Nội dung nhỏ thứ hai của 2.1 3](#_Toc78549799)

[2.2 Giải thích thuật toán. 3](#_Toc78549800)

[2.2.1 Nội dung nhỏ thứ nhất của 2.2 3](#_Toc78549801)

[2.2.2 Nội dung nhỏ thứ hai của 2.2 4](#_Toc78549802)

[KẾT LUẬN 6](#_Toc78549803)

[TÀI LIỆU THAM KHẢO 7](#_Toc78549804)

LỜI NÓI ĐẦU

Trong phần này sẽ trình bày về sự cần thiết của đề tài. Tóm tắt nội dung mà đề tài yêu cầu, và những phần sẽ trình tiếp theo.

Bài viết phải được đánh máy, copy tài liệu phải chỉnh sửa theo văn phong của riêng mình.

**Trình bày trên khổ giấy A4; Canh lề trái 3 cm; phải 1.5 cm; trên 2cm; dưới 2.5 cm (có đánh số trang ở chính giữa dòng ở phần dưới).**

**Cỡ chữ 14; dãn dòng 1.5 line.**

**Chữ canh đều 2 bên; hình vẽ có đánh số hình theo chương và được canh giữa xem chi tiết trong các chương;**

**Số trang được đánh bắt đầu từ trang lời nói đầu.**

Chương 1 TÊN NỘI DUNG CỦA CHƯƠNG

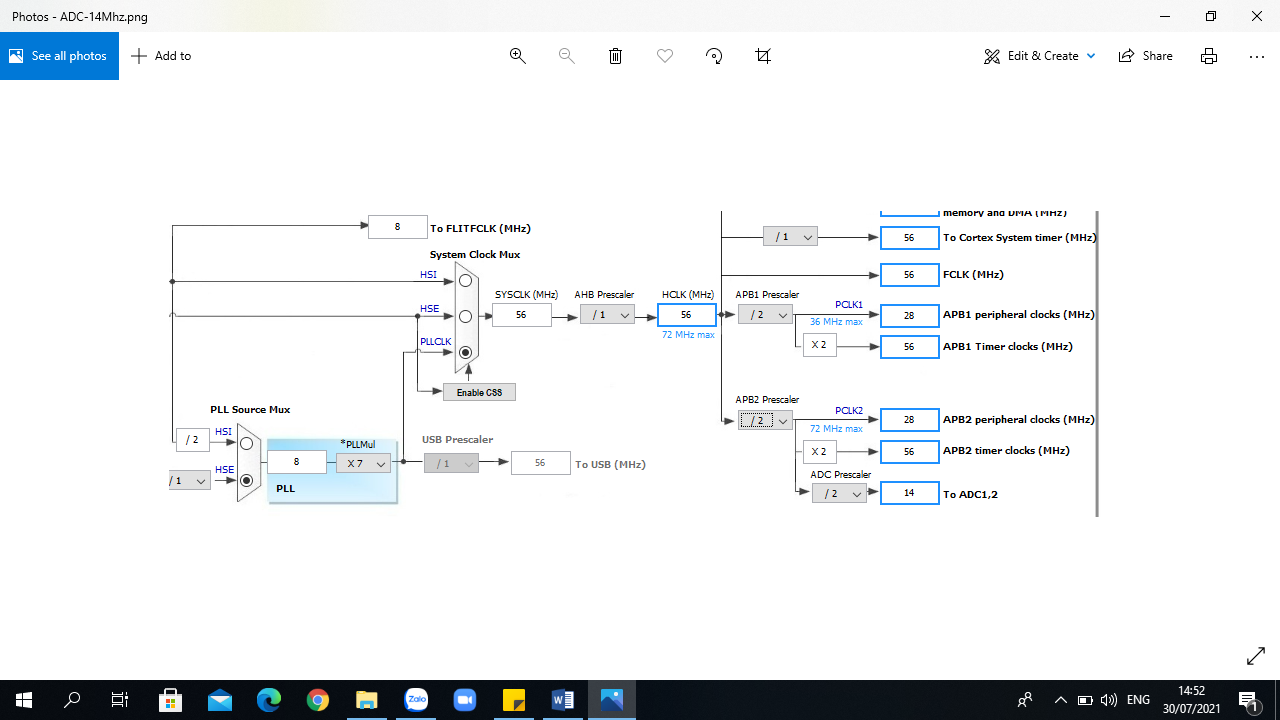
Chương này trình bày toàn bộ những kiến thức phần lý thuyết liên quan tới nội dung thực hành, bài tập của chủ đề.

1.1 Nội dung lớn 1.

………

1.1.1 Nội dung nhỏ thứ nhất của 1.1

…………



Hình 1.1 Tần số làm việc của Bộ vi xử lý ARM C8T6….

1.1.2 Nội dung nhỏ thứ hai của 1.1

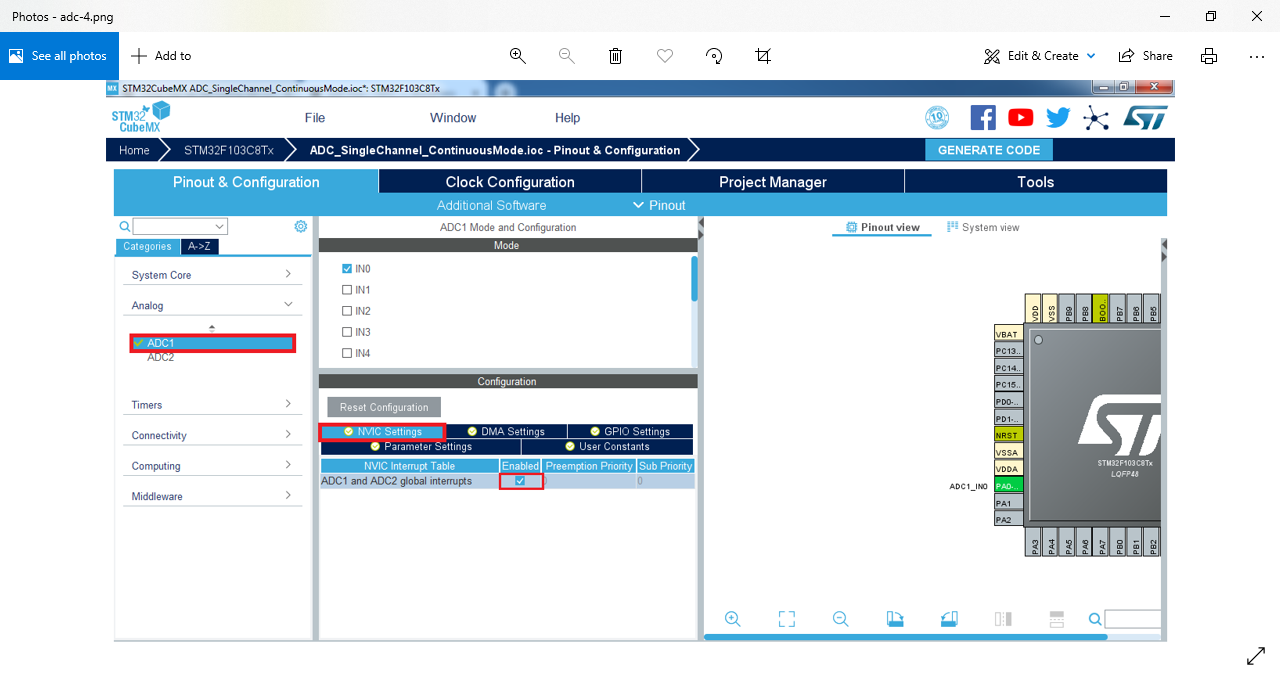
………….

1.2 Nội dung lớn 2.

…………..

1.2.1 Nội dung nhỏ thứ nhất của 1.2

………….



Hình 1.2 Lựa chọn chân chíp cho Bộ vi xử lý

1.2.2 Nội dung nhỏ thứ hai của 1.2

……………..

1.3 ……………

Chương 2 THUẬT TOÁN LẬP LỊCH CHO CPU KHÔNG ƯU TIÊN ROUND ROBIN

Chương này trình bày chi tiết toàn bộ mô tả thuật toán bằng sơ đồ, hình vẽ; Và mã nguồn chương trình cho từng module (Không lấy cả chương trình); Toàn bộ mã nguồn chương trình để trong phần Phụ lục của quyển. Có cả những nhận xét đánh giá về thực nghiệm mà nhóm thực hiện. Xu thế so với trong thực tiễn là gì? (Gắn với nội dung của từng nhóm)

2.1 Giới thiệu về thuật toán đặt lịch RR

…………..

2.1.1 Ưu điểm

………….

2.1.2 Nhược điểm

………….

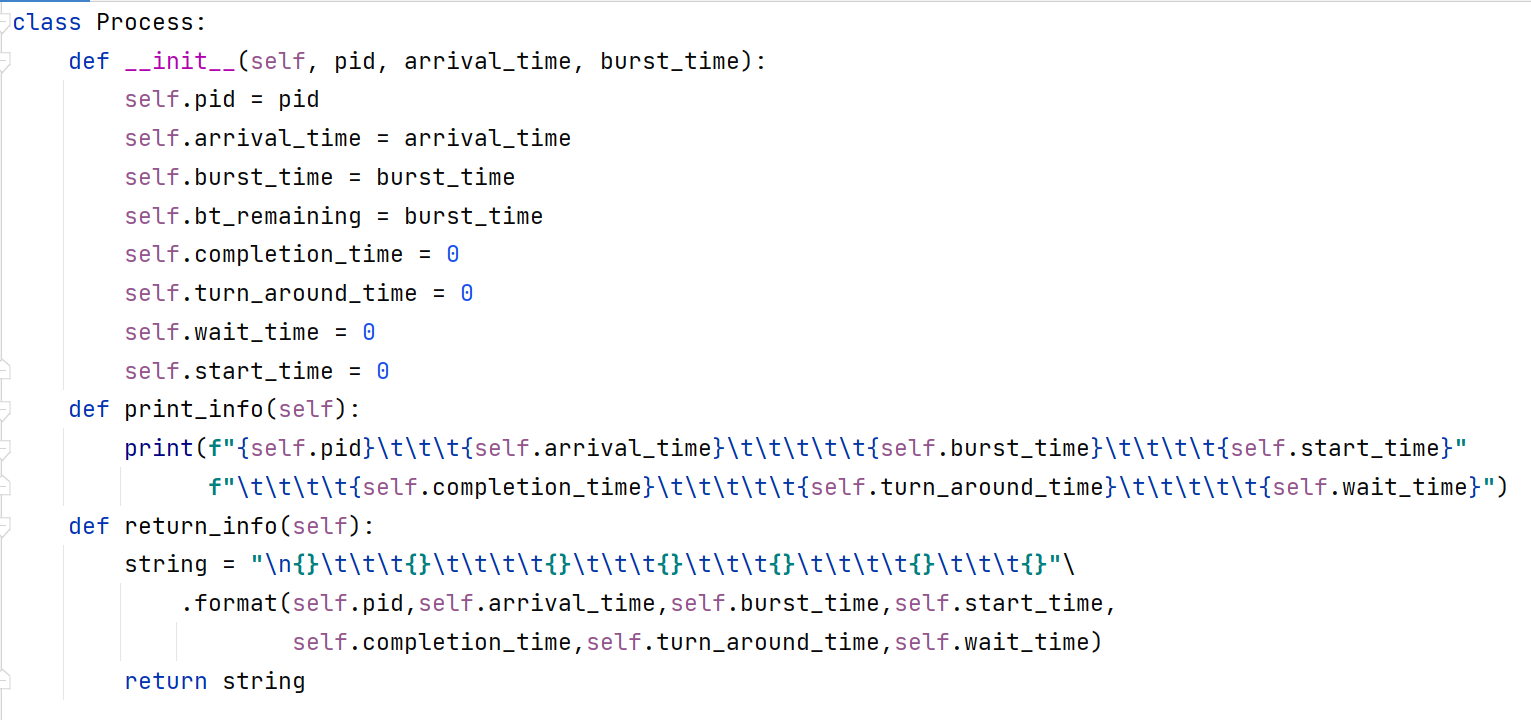
2.2 Giải thích thuật toán

2.2.1 Sơ lược về thuật toán

Thuật toán Round Robin làm việc với các tiến trình nên trước hết chúng ta sẽ tạo Class Process ứng với các tiến trình. Class này bao gồm các thuộc tính như: pid (ID của tiến trình), arrival\_time (thời điểm xuất hiện của tiến trình), burst\_time (thời gian thực thi của tiến trình).

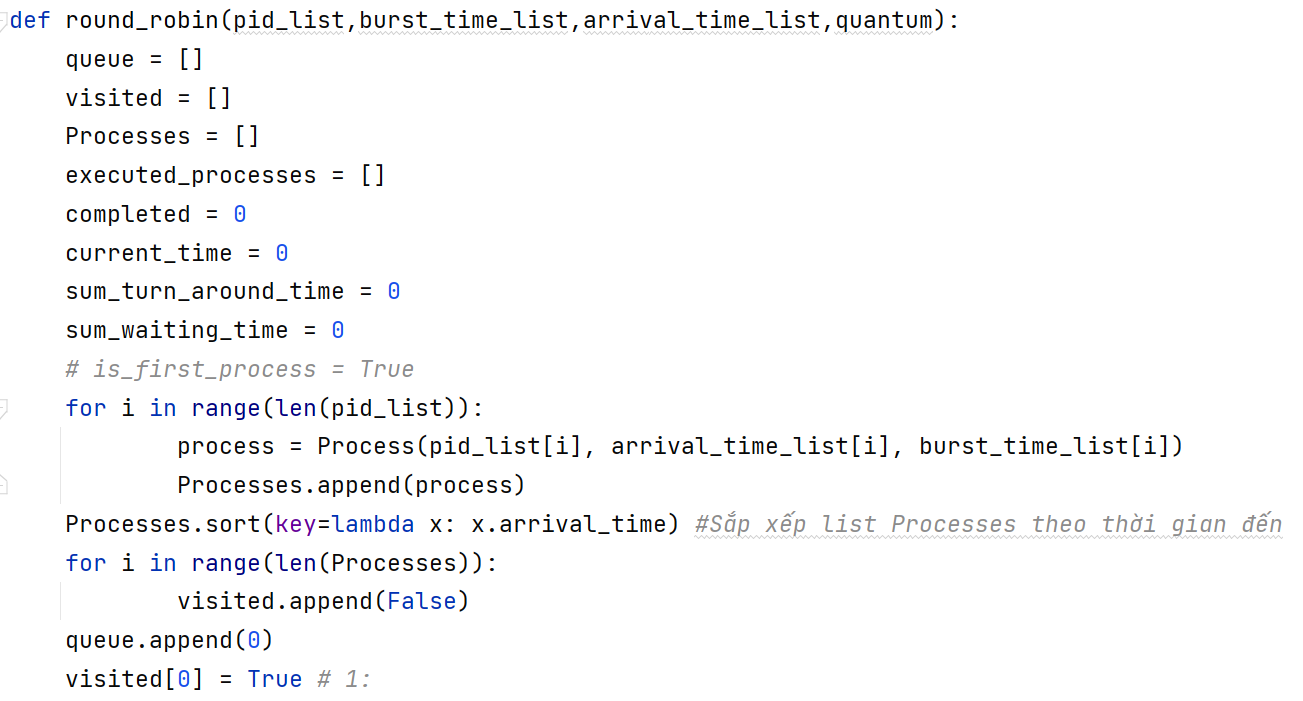
Ngoài các thuộc tính chính trên, các tiến trình còn có các thuộc tính được tạo tự động như bt\_remaining (thời gian thực thi còn lại), completion\_time (thời gian đã thực thi được trên tổng thời gian thực thi của tiến trình), turn\_around\_time (thời gian quay vòng để 1 tiến trình hoàn thành), wait\_time (Thời gian một tiến trình ở trong hàng đợi ready), start\_time (thời gian một tiến trình mới thực thi)

Các phương thức mà class Process có là hai phương thức trả về thông tin và in thông tin của process.



Hình 2.2.1 Class Process

Hàm chính của thuật toán Round Robin được truyền vào các tham số như pid\_list (danh sách ID của các tiến trình), burst\_time\_list (danh sách thời gian thực thi), arrival\_time\_list (danh sách thời điểm đến) và quatum (đơn vị thời gian)



Hình 2.2.2 Def round\_robin (1)

Định nghĩa các biến:

- Biến queue: là biến danh sách hàng đợi chứa thứ tự thực thi của các tiến trình

- Biến visited: là biến danh sách chứa các giá trị đúng sai, đúng khi tiến trình được thực thi còn sai khi vẫn ở không được thực thi

- Biến Processes: là biến danh sách chứa tấ cả các tiến trình, mỗi tiến trình là 1 class

- Biến executed\_processes: là biến danh sách sẽ chứa thứ tự các tiến trình được thực thi

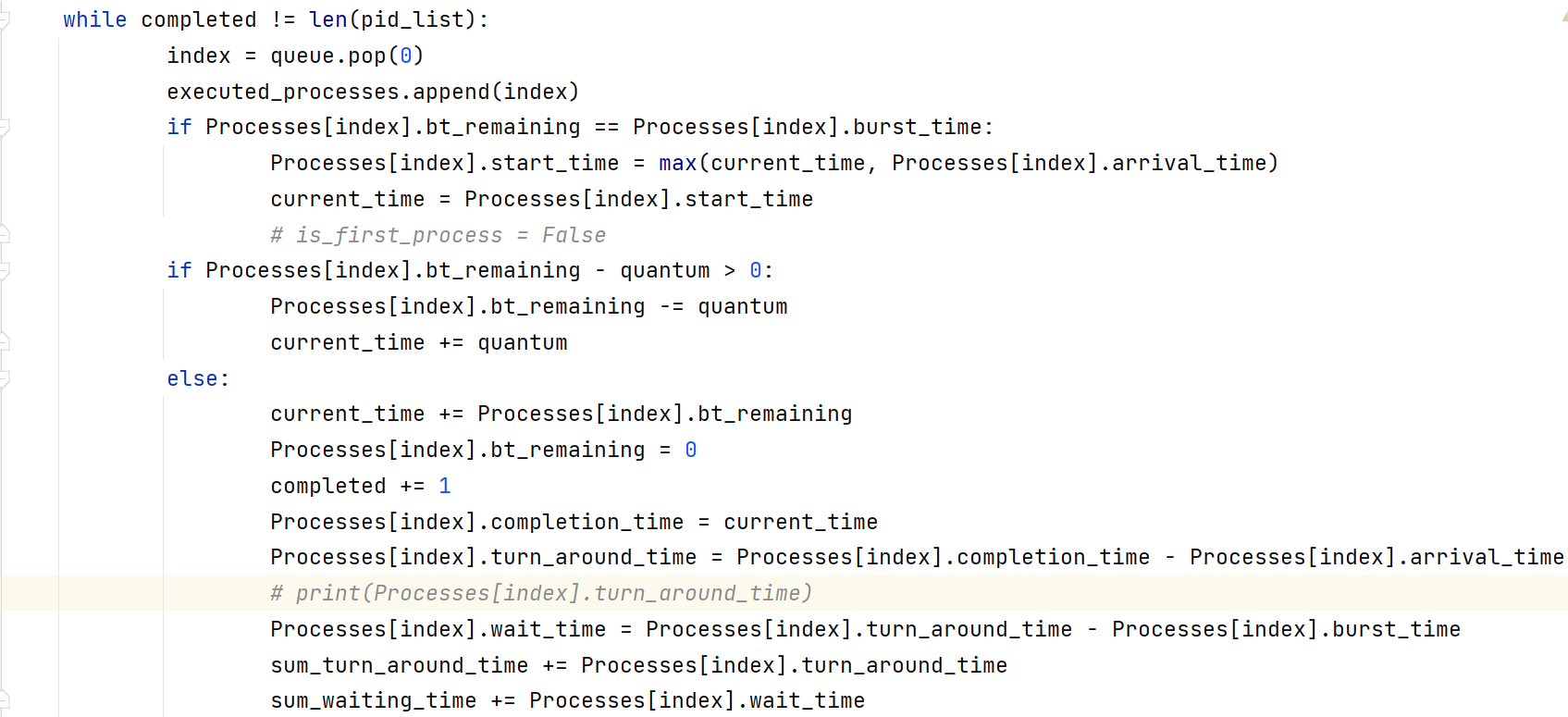
- Biến completed: là biến biểu diễn cho số lượng tiến trình đã thực thi xong

- Biến current\_time: trả về thời gian hiện tại

- Biến sum\_turn\_around\_time: là tổng thời gian để hoàn tất tất cả tiến trình

- Biến sum\_waiting\_time: là tổng thời gian các tiến trình khi ở trong hàng đợi

Vòng for đầu tiên, ta thực hiện nhập thông tin các tiến trình vào danh sách Processes. Vòng for thứ hai, khi tất cả các tiến trình chưa được thực thi, ta set visited cho từng tiến trình là False. Nhập thủ công tiến trình đầu tiên vào hàng đợi với queue.append(0), 0 là id của tiến trình đầu tiên, set visited bằng True.

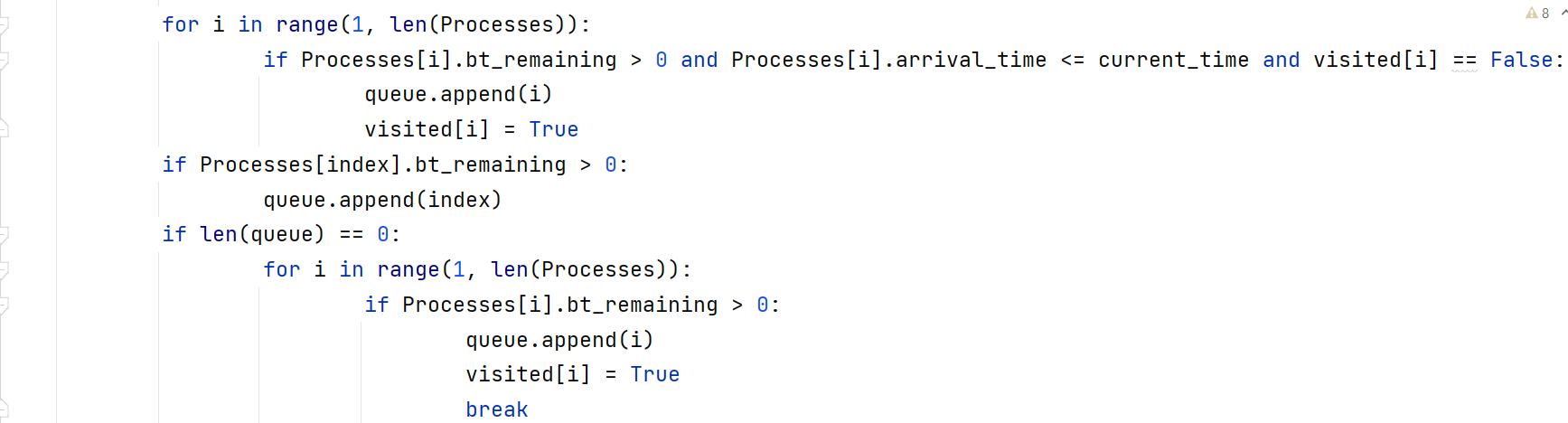


Hình 2.2.3 Def round\_robin (2)

Ta sẽ cần một vòng lặp while để chạy chương trình với điều kiện completed != len(pid\_list) tức là vòng lặp vẫn sẽ chạy khi biến danh sách completed khác độ dài danh sách các tiến trình. Biến index là biến local được tạo để lấy giá trị đầu tiên của hàng đợi queue loại ra. Sau đó biến tiến trình với ID bằng index sẽ được thực thi tức là thêm vào biến danh sách executed.

Câu lệnh điều kiện đầu tiền kiểm tra xem nếu thời gian thực thi còn lại của 1 tiến trình bằng với thời gian cần phải thực thi của nó tức là chưa thực thi được gì, tiến trình này mới vào thì start\_time của tiến trình đó bằng max của thời gian hiện tại và thời điểm tiến trình đó xuất hiện. Để giải thích tại sao lại lấy max của current\_time và arrival\_time là vì: Giả sử có 1 tiến trình đang chạy và trong thời gian nó chạy thì có một tiến trình khác được thêm vào hàng đợi, vậy thời gian start\_time để tiến trình đó chạy là sau khi tiến trình trước chạy xong tức là thời điểm lúc đó là current\_time chứ không phải là arrival\_time của tiến trình sau. Nhưng có trường hợp ngoại lệ là thời điểm xuất hiện và thời gian thực thi của 1 tiến trình trước nhỏ hơn so với thời điểm xuất hiện của tiến trình sau thì lúc này start\_time của tiến trình sau phải bằng arrival\_time của nó. Sau khi thực thi thì lúc này biến current\_time sẽ được gán bằng giá trị start\_time.

Câu lệnh điều kiện thứ hai kiểm tra xem nếu thời gian thực thi còn lại của tiến trình lớn hơn đơn vị thời gian quantum thì khi thực thi sẽ giảm thời gian thực thi còn lại đi lượng thời gian bằng quantum, và set thời gian hiện tại cộng thêm quantum. Trong trường hợp thời gian thực thi còn lại nhỏ hơn quantum thì thời gian hiện tại current\_time cộng thêm lượng thời gian bằng thời gian thực thi còn lại của tiến trình và thời gian thực thi còn lại của tiến trình bằng 0. Biến completed sẽ cộng thêm 1. Biến thời gian hoàn thành completion\_time sẽ được set bằng thời điểm hiện tại current\_time. Biến thời điểm quay vòng sẽ bằng thời điểm hiện tại khi mà tiến trình đã thực hiện được một phần xong vào hàng đợi trừ đi thời gian tiến trình xuất hiện, sau đó tất cả được cộng vào biến sum\_turn\_around\_time. Biến thời gian chờ của 1 tiến trình sẽ bằng thời gian lúc tiến trình đó hoàn thành thực thi trừ đi thời gian thực thi của tiến trình đó, sau đó tất cả được cộng vào biến sum\_waiting\_time.



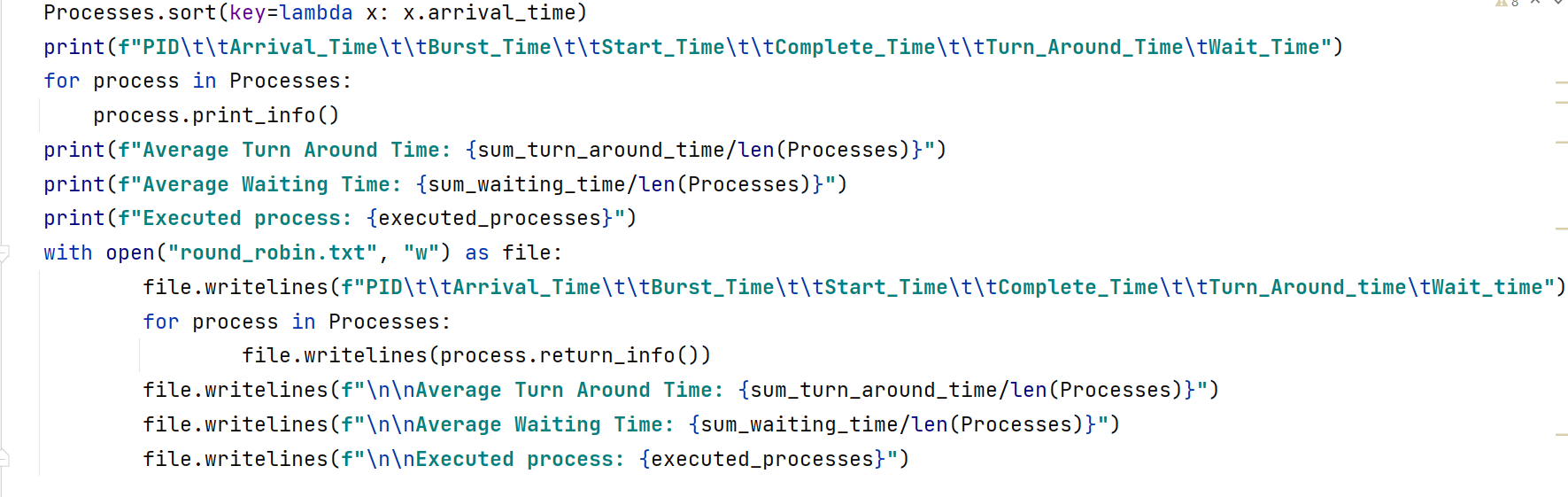
Hình 2.2.4 Def round\_robin (3)

Vòng lặp for trên kiểm tra xem trong danh sách các tiến trình, tiến trình nào đang có thời gian thực thi còn lại lớn hơn 0 và thời điểm xuất hiện của nó nhỏ hơn hoặc bằng thời gian hiện tại và chưa được thêm vào hàng đợi thì ID của tiến trình đo sẽ được thêm vào và set visited bằng True, do là ID 0 đã được thêm từ trước nên vòng lặp for này sẽ thêm bắt đầu từ 1.

Câu lệnh điều kiện đầu tiên kiểm tra nếu thời gian thực thi còn lại của tiến trình vẫn chưa hết, nó sẽ được thêm lại vào hàng đợi, do là bị pop từ đầu nên lúc này phải thêm lại.

Câu lệnh điều kiện thứ hai kiểm tra xem nếu hàng đợi đang rỗng, nó sẽ thêm một tiến trình mới với điều kiện thời gian thực thi còn lại của tiến trình đó lớn hơn 0.

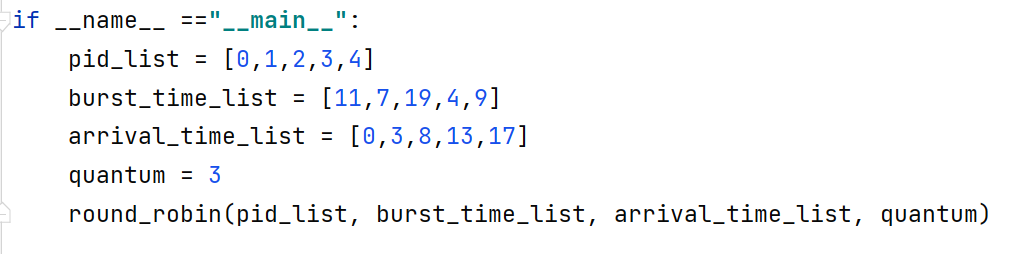
Dưới đây là phần in và lưu output vào file .txt



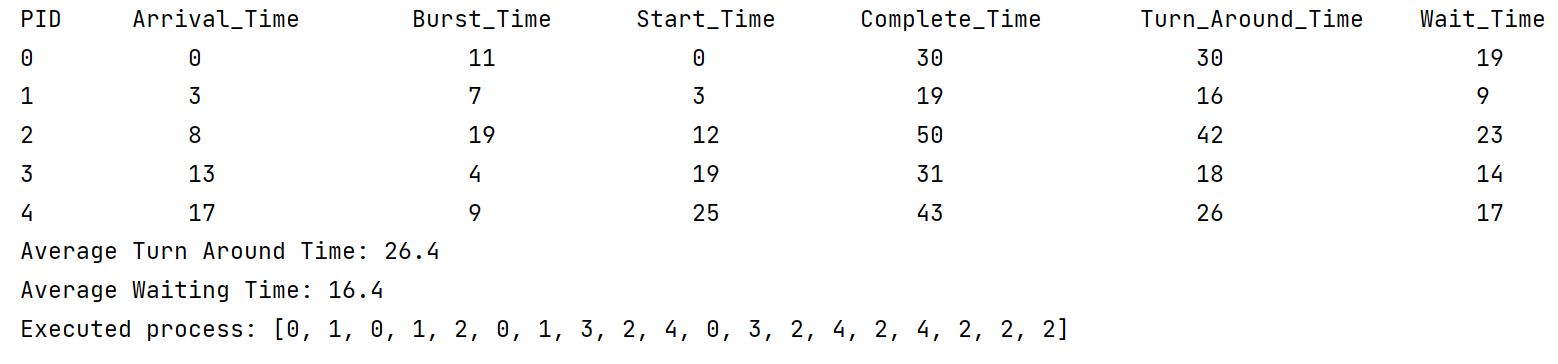
Hình 2.2.5 Def round\_robin (4)

2.2.2 Demo

Giả sử ta có Input như sau:



Và đây là Output:



Phụ lục

class Process:  
 def \_\_init\_\_(self, pid, arrival\_time, burst\_time):  
 self.pid = pid  
 self.arrival\_time = arrival\_time  
 self.burst\_time = burst\_time  
 self.bt\_remaining = burst\_time  
 self.completion\_time = 0  
 self.turn\_around\_time = 0  
 self.wait\_time = 0  
 self.start\_time = 0  
 def print\_info(self):  
 print(f"{self.pid}\t\t\t{self.arrival\_time}\t\t\t\t\t{self.burst\_time}\t\t\t\t{self.start\_time}"  
 f"\t\t\t\t{self.completion\_time}\t\t\t\t\t{self.turn\_around\_time}\t\t\t\t\t{self.wait\_time}")  
 def return\_info(self):  
 string = "\n{}\t\t\t{}\t\t\t\t{}\t\t\t{}\t\t\t{}\t\t\t\t{}\t\t\t{}"\  
 .format(self.pid,self.arrival\_time,self.burst\_time,self.start\_time,  
 self.completion\_time,self.turn\_around\_time,self.wait\_time)  
 return string  
  
def round\_robin(pid\_list,burst\_time\_list,arrival\_time\_list,quantium):  
 queue = []  
 visited = []  
 Processes = []  
 executed\_processes = []  
 completed = 0  
 current\_time = 0  
 sum\_turn\_around\_time = 0  
 sum\_waiting\_time = 0  
 *# is\_first\_process = True* for i in range(len(pid\_list)):  
 process = Process(pid\_list[i], arrival\_time\_list[i], burst\_time\_list[i])  
 Processes.append(process)  
 Processes.sort(key=lambda x: x.arrival\_time)  
 for i in range(len(Processes)):  
 visited.append(False)  
 queue.append(0)  
 visited[0] = True *# 1:* while completed != len(pid\_list):  
 index = queue.pop(0)  
 executed\_processes.append(index)  
 if Processes[index].bt\_remaining == Processes[index].burst\_time:  
 Processes[index].start\_time = max(current\_time, Processes[index].arrival\_time)  
 current\_time = Processes[index].start\_time  
 *# is\_first\_process = False* if Processes[index].bt\_remaining - quantium > 0:  
 Processes[index].bt\_remaining -= quantium  
 current\_time += quantium  
 else:  
 current\_time += Processes[index].bt\_remaining  
 Processes[index].bt\_remaining = 0  
 completed += 1  
 Processes[index].completion\_time = current\_time  
 Processes[index].turn\_around\_time = Processes[index].completion\_time - Processes[index].arrival\_time  
 Processes[index].wait\_time = Processes[index].turn\_around\_time - Processes[index].burst\_time  
 sum\_turn\_around\_time += Processes[index].turn\_around\_time  
 sum\_waiting\_time += Processes[index].wait\_time  
 for i in range(1, len(Processes)):  
 if Processes[i].bt\_remaining > 0 and Processes[i].arrival\_time <= current\_time and visited[i] == False:  
 queue.append(i)  
 visited[i] = True  
 if Processes[index].bt\_remaining > 0:  
 queue.append(index)  
 if len(queue) == 0:  
 for i in range(1, len(Processes)):  
 if Processes[i].bt\_remaining > 0:  
 queue.append(i)  
 visited[i] = True  
 break  
 Processes.sort(key=lambda x: x.arrival\_time)  
 print(f"PID\t\tArrival\_Time\t\tBurst\_Time\t\tStart\_Time\t\tComplete\_Time\t\tTurn\_Around\_time\tWait\_time")  
 for process in Processes:  
 process.print\_info()  
 print(f"Average Turn Around Time: {sum\_turn\_around\_time/len(Processes)}")  
 print(f"Average Waiting Time: {sum\_waiting\_time/len(Processes)}")  
 print(f"Executed process: {executed\_processes}")  
 *# with open("round\_robin.txt", "w") as file:  
 # file.writelines(f"PID\t\tArrival\_Time\t\tBurst\_Time\t\tStart\_Time\t\tComplete\_Time\t\tTurn\_Around\_time\tWait\_time")  
 # for process in Processes:  
 # file.writelines(process.return\_info())  
 # file.writelines(f"\n\nAverage Turn Around Time: {sum\_turn\_around\_time/len(Processes)}")  
 # file.writelines(f"\n\nAverage Waiting Time: {sum\_waiting\_time/len(Processes)}")  
 # file.writelines(f"\n\nExecuted process: {executed\_processes}")*if \_\_name\_\_ =="\_\_main\_\_":  
 pid\_list = [0,1,2,3,4]  
 burst\_time\_list = [2,7,19,4,9]  
 arrival\_time\_list = [0,3,8,13,17]  
 quantium = 3  
 round\_robin(pid\_list, burst\_time\_list, arrival\_time\_list, quantium)

KẾT LUẬN

A. Những kết quả đạt được của bài tập môn học

Nêu những kết quả đạt được của môn học, hiểu và nắm được những kiến thức gì, có những thực nghiệm gì cho sự chuẩn bị sau này làm đồ án tốt nghiệp hay ra trường…..

B. Hướng phát triển tiếp theo và đề xuất

Nêu những hướng phát triển tiếp theo của đề tài, trong ứng dụng vào thực tiễn hoặc trong đồ án tốt nghiệp sau này.

Đề xuất thêm gì đối với khoa và học viện.

TÀI LIỆU THAM KHẢO

Liệt kê những tài liệu tham khảo để thực hiện được đề tài.

[1]. Tên tác giả, tên tài liệu tham khảo, năm xuất bản;

[2]. …………………………………………………..

…..