|  |  |
| --- | --- |
| **Name:** Lê Minh Nguyệt  **ID:** 21521211 | **Name:** Trần Trung Tín  **ID:** 21522679 |
| **Name:** Hồ Đức Trưởng  **ID:** 21522730 | **Name:** Nguyễn Công Nguyên  **ID:** 21521200 |
| **Class:** IT007.N11 | **Group Name:** Team |

**OPERATING SYSTEM  
LAB 04’S REPORT**

**SUMMARY**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Task** | | | **Status** | **Page** |
| Section 4.5 | Ex 1. Giải thuật SJF | * 1. Vẽ lưu đồ giải thuật | Done | 2 |
| * 1. Trình bày tính đúng đắn của lưu đồ bằng cách chạy tay ít nhất 01 test case tối thiểu 05 tiến trình | 3 |
| * 1. Thực hiện code cho giải thuật | 6 |
| * 1. Trình bày tính đúng đắn của code bằng cách chạy ít nhất 03 test case, mỗi test case 5 tiến trình | 13 |
| Ex 2. Giải thuật SRTF | 2.1. Vẽ lưu đồ giải thuật | Done | 16 |
| 2.2. Trình bày tính đúng đắn của lưu đồ bằng cách chạy tay ít nhất 01 test case tối thiểu 05 tiến trình | 18 |
| 2.3. Thực hiện code cho giải thuật | 21 |
| 2.4. Trình bày tính đúng đắn của code bằng cách chạy ít nhất 03 test case, mỗi test case 5 tiến trình | 27 |
| Ex 3. Giải thuật RR | 3.1. Vẽ lưu đồ giải thuật | Done | 30 |
| 3.2. Trình bày tính đúng đắn của lưu đồ bằng cách chạy tay ít nhất 01 test case tối thiểu 05 tiến trình | 33 |
| 3.3. Thực hiện code cho giải thuật | 36 |
| 3.4. Trình bày tính đúng đắn của code bằng cách chạy ít nhất 03 test case, mỗi test case 5 tiến trình | 42 |

```````````````````````````

**Self-scrores: 9,5**

**Section 4.5**

***Ex 1. Giải thuật SJF (Shortest Job First):***

* 1. **Vẽ lưu đồ giải thuật**

Diagram

Description automatically generated

Diagram

Description automatically generated

*Hình 1. Lưu đồ giải thuật SJF*

* 1. **Trình bày tính đúng đắn của lưu đồ bằng cách chạy tay ít nhất 01 test case tối thiểu 05 tiến trình**

*Test case:* n=5

|  |  |  |
| --- | --- | --- |
| Process name | Arrival Time | Burst time |
| 1 | 0 | 12 |
| 2 | 2 | 7 |
| 3 | 5 | 8 |
| 4 | 12 | 6 |
| 5 | 9 | 3 |

Trong minh họa giải thuật dưới đây, một process sẽ được trình bày các thuộc tính lần lượt theo thứ tự [process name, arrival time, burst time, remaining burst time, start, finish, wt, rt, tat]

*Thực hiện chạy tay test case trên:*

1. Sau quá trình nhập và khởi tạo:

plist=[[1,0,12,12], [2,2,7,7], [3,5,8,8], [4,12,6,6 ], [5,9,3,3]], n=5

1. Ta sử dụng sort theo arrvial time:

plist=[[1,0,12,12], [2,2,7,7], [3,5,8,8], [5,9,3,3], [4,12,6,6]]

1. Khởi tạo biến i =0, đồng thời sử dụng vòng lặp để đẩy những process có cùng arrvial time với plist[0] khi này vào hàng đợi ready mỗi lần đẩy vào ready tăng i lên 1 đơn vị.
2. Sau đó sắp xếp hàng đợi ready theo bur theo giải thuật insertion sort.

Ready=[[1,0,12,12]]

1. Gán current\_time=ready[0].arr=0, pi=vị trí của plist chưa được push vào ready=1, i=0,
2. Sử dụng vòng lặp khi i<5 (0<5)
   1. Tại currentime=0 kiểm ta thấy ready[0].rb==ready[0].bur

⇨ ready=[[1,0,12,12,0]], current\_time+=1,ready[0].rb--

* 1. Tại currentime=1, ready=[[1,0,12,11,0]], current\_time+=1, ready[0].rb-- tại đây do ta kiểm tra thấy current\_time=plist[pi].arr=2, ta sẽ push plist[1] vào ready đồng thời pi=pi+1=2,sử dụng sort(từ 1 đến vị trí cuối cùng trong hàng đợi ready) theo burst time

⇨ ready=[[1,0,12,10,0], [2,2,7,7]]

* 1. Tương tự như vậy, tại current\_time=4, ready=[[1,0,12,8,0], [2,2,7,7]], current\_time+=1, ready[0].rb--, tại đây do ta kiểm tra thấy current\_time=plist[pi].arr=5 ta sẽ push plist[2] vào ready đồng thời pi=pi+1=3, sử dụng sort(từ 1 đến vị trí cuối cùng trong hàng đợi ready) theo burstime

⇨ ready=[[1,0,12,7,0], [2,2,7,7], [3,5,8,8]]

* 1. Tượng tự như vậy ⇨ current\_time=8, ready=[[1,0,12,4,0], [2,2,7,7]], [3,5,8,8]], current\_time+=1 ready[0].rb--, do ta kiểm tra thấy current\_time=plist[pi].arr=9 ta sẽ push plist[3] vào ready đồng thời pi=pi+1=4, sử dụng sort(từ 1 đến vị trí cuối cùng trong hàng đợi ready) theo burst time ⇨ ready=[[1,0,12,3,0], [5,9,3,3], [2,2,7,7], [3,5,8,8]]
  2. Tương tự như vậy đến thời điểm current\_time =11, ready=[[1,0,12,1,0], [5,9,3,3], [2,2,7,7], [3,5,8,8]], current\_time+=1, ready[0].rb--, do ta kiểm tra thấy current\_time=plist[pi].arr=12 ta sẽ push plist[4] vào ready đồng thời pi=pi+1=5, sử dụng sort(từ 1 đến vị trí cuối cùng trong hàng đợi ready) theo burst time (do pi=5=n nên ta bỏ qua bước này trong vòng lặp

⇨ ready=[[1,0,12,0,0], [5,9,3,3], [4,12,6,6], [2,2,7,7], [3,5,8,8]]. Ở đây ta kiểm tra thấy ready[0].rb=0, do vậy pop ready[0] ra khỏi hàng đợi đồng thời tính toán finish, wt, rt, tat push ready[0] vào hàng đợi terminate ⇨ terminate=[[1,0,12,0,0,12,0,0,12]], i=i+1=1, ready=[[5,9,3,3], [4,12,6,6], [[2,2,7,7]], [3,5,8,8]].

* 1. Tại thời thời điểm current\_time=12, kiểm tra thấy ready[0].bur==ready[0]rb, ready=[[5,9,3,3,12], [4,12,6,6], [2,2,7,7], [3,5,8,8]], current\_time+=1, ready[0].rb--
  2. Tương tự như vậy đến thời điểm current\_time=14, ready=[[5,9,3,1,12], [4,12,6,6], [2,2,7,7], [3,5,8,8]], current\_time+=1, ready[0].rb--.

Ở đây ta kiểm tra thấy ready[0].rb=0 ,do vậy pop ready[0] ra khỏi hàng đợi đồng thời tính toán finish, wt, rt, tat push ready[0] vào hàng đợi terminate

⇨ terminate=[[1,0,12,0,0,12,0,0,12]], [5,9,3,0,12,15,3,3,6]], i=i+1=2

Ready=[[4,12,6,6], [2,2,7,7], [3,5,8,8]].

* 1. Tại thời điểm current\_time=15 ,kiểm tra thấy ready[0].bur==ready[0].rb ⇨ ready=[[4,12,6,6,15], [2,2,7,7], [3,5,8,8]], current\_time+=1, ready[0].rb--
  2. Tại thời điểm current\_time=20, ready=[[4,12,6,1,15], [2,2,7,7], [3,5,8,8]], current\_time+=1, ready[0].rb--.

Ở đây ta kiểm tra thấy ready[0].rb=0 ,do vậy pop ready[0] ra khỏi hàng đợi ready đồng thời tính toán fininsh, wt, rt, tat push ready[0] vào hàng đợi terminate

⇨ terminate=[[1,0,12,0,0,12,0,0,12], [5,9,3,0,12,15,3,3,6], [4,12,6,0,15,21,3,3,9]], i=i+1=3

ready=[[2,2,7,7], [3,5,8,8]].

* 1. Tại thời điểm current\_time=21, kiểm tra thấy ready[0].bur==ready[0].rb ⇨ ready=[[2,2,7,7,21], [3,5,8,8]] current\_time+=1, ready[0].rb--
  2. Tại thời điểm currentime=27, ready=[[2,2,7,1,21], [3,5,8,8]] current\_time+=1. Ở đây ta kiểm tra thấy ready[0].rb=0, do vậy pop ready[0] ra khỏi hàng đợi đồng thời tính toán fininsh, wt, rt, tat push ready[0] vào hàng đợi terminate

⇨ terminate=[[1,0,12,0,0,12,0,0,12], [5,9,3,0,12,15,3,3,6], [4,12,6,1,15,21,3,3,9], [2,2,7,0,21,28,19,19,26]], i=i+1=4

Ready=[[3,5,8,8]].

* 1. Tại thời điểm current\_time=28, kiểm tra thấy ready[0].bur==ready[0].rb => ready=[[3,5,8,8,28]], current\_time+=1, ready[0].rb--
  2. Tại thời điểm current\_time=35, ready=[[3,5,8,1,28]], current\_time+=1, ready[0].rb--. Ở đây ta kiểm tra thấy ready[0].rb=0 ,do vậy pop ready[0] ra khỏi hàng đợi đồng thời tính toán fininsh, wt, rt, tat push ready[0] vào hàng đợi terminate

⇨ terminate=[[1,0,12,0,0,12,0,0,12], [5,9,3,0,12,15,3,3,6], [4,12,6,0,15,21,3,3,9], [2,2,7,0,21,28,19,19,26], [3,5,8,0,28,36,23,23,31]], i=i+1=5

Ready=[]

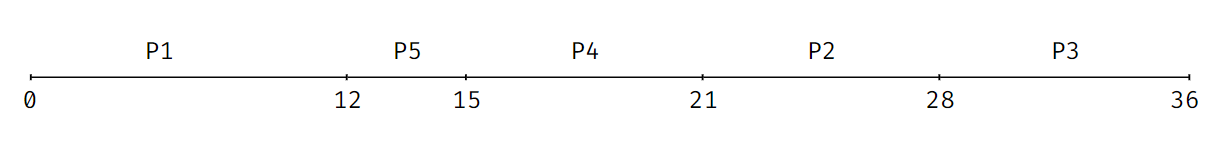
1. Ta kiểm tra i==n (5==5) thoát khỏi vòng lặp đồng thời tính avewt=9.6, avert=9.6, avetat=16.8
2. Xuất kết quả ra ngoài màn hình
3. Kết thúc

*Kết quả sau khi chạy tay (lưu trong term[]):*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Process name | Arrival time | Burst time | Remaining burst | Start | Finish | WT | RT | TAT |
| 1 | 0 | 12 | 0 | 0 | 12 | 0 | 0 | 12 |
| 5 | 9 | 3 | 0 | 12 | 15 | 3 | 3 | 6 |
| 4 | 12 | 6 | 0 | 15 | 21 | 3 | 3 | 9 |
| 2 | 2 | 7 | 0 | 21 | 28 | 19 | 19 | 26 |
| 3 | 5 | 8 | 0 | 28 | 36 | 23 | 23 | 31 |

avewt=9.6, avert=9.6, avetat=16.8

*So sánh với kết quả giải tay test case trên (thứ tự process name lần lượt từ 1 đến 5):*



start[ ]= { 0, 21, 28, 15, 12 }

fisnish[ ] ={12, 28, 36, 21, 15}

rt[ ]={ 0, 19, 23, 3, 3 }

* avert= 9.6

wt[ ]={ 0, 19, 23, 3, 3 }

* avewt= 9.6

tat [ ]={ 12,26, 31, 9, 6 }

* avert= 16.8

***⮚ Có thể thấy, kết quả thực hiện chạy tay trên lưu đồ giống với kết quả giải tay. Vì vậy, ta kết luận lưu đồ đúng.***

* 1. **Thực hiện code cho giải thuật**

Text

Description automatically generated

*Hình 2. Tạo cấu trúc Process và Ready Queue trong giải thuật SJF*

* Struct P chứa những thông tin của một process, gồm: tên, arrival time, burst time, start time, finish time, thời gian đáp ứng, thời gian đợi, thời gian hoàn thành và burst time còn lại (remaining burst time). Hai con trỏ \*print và \*set lần lượt thông qua hàm printP để xuất thông tin của một process ra màn hình, và setP để tính toán RT, WT, TAT.
* Struct ReadyQueue gồm một mảng process và kích cỡ của hàng đợi.

Text

Description automatically generated

*Hình 3. Hàm khởi tạo Ready Queue, hàm tính RT, WT, TAT và hàm in Process trong giải thuật SJF*

* Hàm init() khởi tạo hàng đợi Ready ban đầu không có phần tử nào.
* Các hàm setP() và printP() thực hiện tính toán và xuất kết quả như hình 3.

Text

Description automatically generated

*Hình 4. Hàm sắp xếp Process theo arrival time tăng dần trong giải thuật SJF*

Text

Description automatically generated

*Hình 5. Hàm sắp xếp Process theo burst time tăng dần trong giải thuật SJF*

* Hàm SortBur() sắp xếp một mảng process tăng dần theo burst time. Hàm này sử dụng giải thuật Insertion Sort thay vì sắp xếp thông thường, để khi một process xuất hiện mà có cùng burst time với một process khác đã ở sẵn trong hàng đợi, nó sẽ không tranh mất vị trí của process đến trước mà đứng đợi ngay sau process đến trước.

Text

Description automatically generated

*Hình 6. Hàm thêm một process và xóa process đầu tiên* *ra khỏi hàng đợi Ready trong giải thuật SJF*

* Hàm pushP() và popP() lần lượt thêm một process và xóa process đầu tiên ra khỏi hàng đợi Ready, khi đó kích cỡ của hàng đợi cũng sẽ tăng hoặc giảm đi một phần tử tương ứng.

Text

Description automatically generated

*Hình 7. Khởi tạo mảng process input, hàng đợi Ready, mảng terminated và các biến để tính tổng RT, WT, TAT trong giải thuật SJF*

Text

Description automatically generated

*Hình 8. Nhập input, khởi tạo giá trị ban đầu cho rb và sắp xếp mảng process input theo arrival time*

* Sau khi khởi tạo các mảng và hàng đợi, ta tiến hành nhập input gồm số lượng process n, tên, arrival time và burst time của từng process. Ta khởi tạo giá trị ban đầu cho rb (remaining burst time) bằng burst time của chính process đó.
* Các process nhập vào sẽ được lưu trong mảng plist, ta sắp xếp mảng này tăng dần theo arrival time.

Text

Description automatically generated

*Hình 9. Đưa tất cả những process cùng xuất hiện sớm nhất vào hàng đợi Ready*

* Sử dụng vòng lặp while để push tất cả những process có cùng arrival time với process đến sớm nhất vào hàng đợi Ready. Sau đó, ta tiến hành sắp xếp toàn bộ hàng đợi Ready theo burst time (bắt đầu từ vị trí 0 đến cuối hàng đợi).
* Sau khi push xong những process này, i sẽ chỉ mục đến process có arrival time nhỏ thứ 2 trong plist.

Text

Description automatically generated

*Hình 10. Khởi tạo giá trị ban đầu cho current\_time, pi, i và tính start time cho process đang chạy*

* Ta giả định hàng đợi Ready là một mảng các process, và phần tử đầu tiên trong mảng này luôn là process đang chạy.
* Biến current\_time chứa giá trị thời gian hiện tại, dùng để mô phỏng cách hàng đợi Ready làm việc trong thời gian thực. Ta khởi tạo giá trị ban đầu của current\_time là thời điểm xuất hiện của process đầu tiên được chọn để thực thi (process đến sớm nhất và có burst time nhỏ nhất so với các process đến cùng lúc).
* Biến pi chỉ mục đến các process trong plist chưa được thêm vào hàng đợi Ready (vì chúng chưa xuất hiện). pi bắt đầu từ vị trí i.
* Biến i đặt lại bằng 0, chỉ mục đến các process trong mảng term[] (là mảng chứa các process đã hoàn thành thực thi).
* Ta bắt đầu xử lý định thời CPU, dừng lại khi tất cả các process trong plist đều đã được thêm vào term[].
* Tại thời điểm current\_time, nếu process đầu tiên trong hàng đợi có bur bằng rb (tức nó chưa ở trong trạng thái running trước đó), current\_time chính là start time của nó.
* current\_time tăng thêm 1, cùng lúc burst time của process đang chạy giảm đi 1.

Text

Description automatically generated

*Hình 11. Đưa process vừa xuất hiện vào hàng đợi Ready và sắp xếp, thực hiện các thao tác đối với process đang chạy nếu nó hoàn thành thực thi*

* Tại thời điểm current\_time mới này, ta sử dụng vòng lặp while để thêm tất cả những process đến tại thời điểm này (arrival time == current\_time) vào hàng đợi Ready và tiến hành sắp xếp lại hàng đợi theo burst time của chúng. Ta *chỉ thực hiện sắp xếp từ vị trí 1 đến cuối hàng đợi, vì process đầu tiên vẫn đang thực thi và sẽ không bị ngắt cho đến khi hoàn thành trong giải thuật SJF.*
* Nếu tại current\_time, không có process nào đến, ta thoát khỏi vòng lặp để tiếp tục đếm current\_time.
* Cũng tại thời điểm này, nếu process đang chạy có rb == 0 (tức nó đã chạy xong), finish time của nó chính bằng thời điểm hiện tại. Ta thực hiện tính RT, WT, TAT cho nó thông qua hàm set() và thêm nó vào vị trí i của mảng term[]. Sau đó, ta tăng i lên 1 và xóa process đó ra khỏi hàng đợi Ready. Lúc này, process tiếp theo sẽ được đẩy lên vị trị đầu tiên trong hàng đợi và thực thi.

Text

Description automatically generated

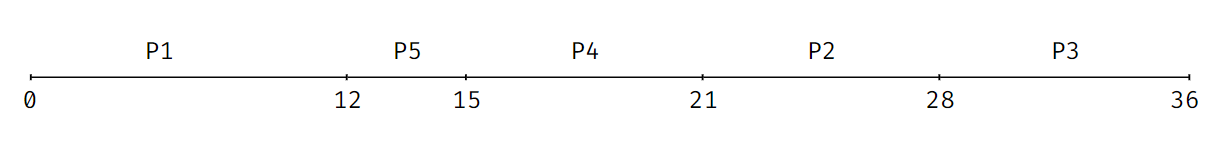
*Hình 12. Thực hiện tính toán RT, WT, TAT trung bình và xuất kết quả ra màn hình*

* 1. **Trình bày tính đúng đắn của code bằng cách chạy ít nhất 03 test case, mỗi test case 5 tiến trình**

*Test case 1:* n=5

|  |  |  |
| --- | --- | --- |
| Process name | Arrival Time | Burst time |
| 1 | 0 | 12 |
| 2 | 2 | 7 |
| 3 | 5 | 8 |
| 4 | 12 | 6 |
| 5 | 9 | 3 |

*Kết quả giải tay:*



*Hình 13. Giản đồ Gantt cho test case 1 giải thuật SJF*

start[ ]= { 0, 21, 28, 15, 12 }

fisnish[ ] ={12, 28, 36, 21, 15}

rt[ ]={ 0, 19, 23, 3, 3 }

* avert= 9.6

wt[ ]={ 0, 19, 23, 3, 3 }

* avewt= 9.6

tat [ ]={ 12,26, 31, 9, 6 }

* avert= 16.8

A picture containing calendar

Description automatically generated

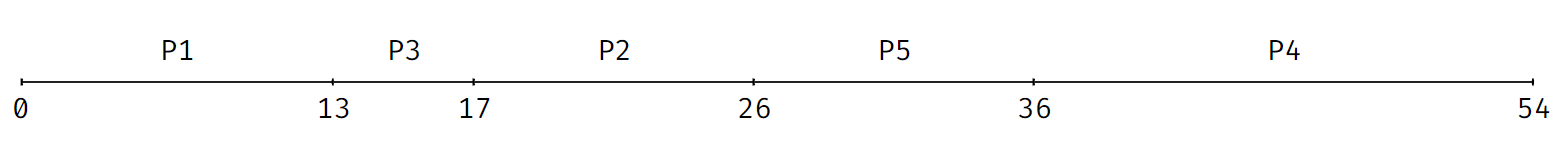
*Hình 14. Chạy test case 1 bằng code giải thuật SJF*

***⮚ Kết quả thực hiện bằng code giống với kết quả giải tay.***

*Test case 2:* n=5

|  |  |  |
| --- | --- | --- |
| Process name | Arrival Time | Burst time |
| 1 | 0 | 13 |
| 2 | 4 | 9 |
| 3 | 6 | 4 |
| 4 | 7 | 18 |
| 5 | 12 | 10 |

*Kết quả giải tay:*



*Hình 15. Giản đồ Gantt cho test case 2 giải thuật SJF*

start[ ]= { 0, 17, 13, 36, 26 }

fisnish[ ] ={13,26,17, 54, 36}

rt[ ]={ 0, 13,7, 29, 14 }

* avert= 12.6

wt[ ]={ 0, 13,7, 29, 14 }

* avert= 12.6

tat [ ]={ 13, 22, 11, 47, 24 }

* avert= 23.4

Calendar

Description automatically generated

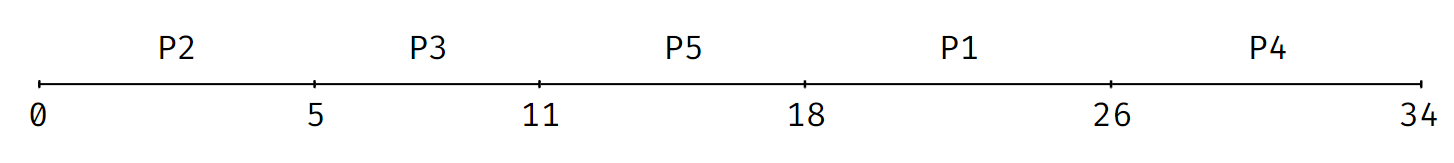
*Hình 16. Chạy test case 2 bằng code giải thuật SJF*

***⮚ Kết quả thực hiện bằng code giống với kết quả giải tay.***

*Test case 3:* n=5

|  |  |  |
| --- | --- | --- |
| Process name | Arrival Time | Burst time |
| 1 | 0 | 8 |
| 2 | 0 | 5 |
| 3 | 2 | 6 |
| 4 | 2 | 8 |
| 5 | 4 | 7 |

*Kết quả giải tay:*



*Hình 17. Giản đồ Gantt cho test case 3 giải thuật SJF*

start[ ]= { 18, 0, 5, 26, 11}

fisnish[ ] ={26, 5, 11 , 34, 18 }

rt[ ]={ 18, 0, 3, 24, 7 }

* avert= 10.4

wt[ ]={ 18, 0, 3, 24, 7 }

* avert= 10.4

tat [ ]={ 26, 5, 9, 32, 14}

* avert= 17.2

Calendar

Description automatically generated

*Hình 18. Chạy test case 3 bằng code giải thuật SJF*

***⮚ Kết quả thực hiện bằng code giống với kết quả giải tay.***

***⮚ Cả 3 test case đều cho kết quả giống với khi giải tay. Vì vậy, ta kết luận code đúng.***

***Ex 2. Giải thuật SRTF (Shortest Remaining Time First):***

* 1. **Vẽ lưu đồ giải thuật**

**A picture containing diagram

Description automatically generated**

**A picture containing diagram

Description automatically generated**

**A picture containing diagram

Description automatically generated**

*Hình 19. Lưu đồ giải thuật SRTF*

* 1. **Trình bày tính đúng đắn của lưu đồ bằng cách chạy tay ít nhất 01 test case tối thiểu 05 tiến trình**

*Test case:* n=5

|  |  |  |
| --- | --- | --- |
| Process name | Arrival Time | Burst time |
| 1 | 0 | 12 |
| 2 | 0 | 7 |
| 3 | 5 | 8 |
| 4 | 9 | 3 |
| 5 | 12 | 6 |

Trong minh họa giải thuật dưới đây, một process sẽ được trình bày các thuộc tính lần lượt theo thứ tự [process name, arrival time, burst time, remaining burst time, start, finish, wt, rt, tat]

*Thực hiện chạy tay test case trên:*

1. Sau quá trình nhập và khởi tạo plist=[[1,0,12,12], [2,0,7,7], [3,5,8,8], [4,9,3,3], [5,12,6,6]], n=5
2. Ta sử dụng sort theo arr plist=[[1,0,12,12], [2,0,7,7], [3,5,8,8], [4,9,3,3], [5,12,6,6]]
3. Khởi tạo biến i=0, đồng thời sử dụng vòng lặp để đẩy những process có cùng arr với plist[0] khi này vào hàng đợi ready mỗi lần đẩy vào ready tăng i lên 1 đơn vị cùng với đó sort ready queue theo reaming burst time
4. ready=[[2,0,7,7], [1,0,12,12]], i=2
5. Gán current\_time=0 (=ready[0].bur), pi=2(=i vị trí của plist chưa được push vào ready), gán lại i=0
6. Sử dụng vòng lặp khi i<5 (0<5)
   1. Tại current\_time=0, kiểm tra thấy ready[0].bur=ready[0].rb do đó ready[0].star=0 ⇨ ready=[[2,0,7,7,0], [1,0,12,12]], current\_time+=1, ready[0].rb--
   2. Tại current\_time=1->3, current\_time+=1, ready[0].rb--
   3. Tại current\_time=4, current\_time+=1, ready[0].rb--, đồng thời kiểm tra thấy current\_time==plist[2(=pi))].arr, ta đẩy plist[2] vào trong hàng đợi ready cùng với đó sort ready queue theo rb (sử dụng giải thuât insertion sort), pi++

* ready=[[2,0,7,2,0], [3,5,8,8], [1,0,12,12]]
  1. Tại current\_time=6, current\_time+=1, ready[0].rb-- . Ta kiểm tra thấy ready[0].rb==0. Do đó ready[0] vào hàng đợi terminate đồng thời tính toán finish, wt, rt, tat, pop ready[0] ra khỏi hàng đợi ready, i+=1
* terminate =[[2,0,7,0,0,7,0,0,7]], ready=[[3,5,8,8], [1,0,12,12]]
  1. Tại current\_time=7, kiểm tra thấy ready[0].bur=ready[0].rb(=8) do đó ready[0].star=7

⇨ ready=[[3,5,8,8,7], [1,0,12,12]], currren\_time++, ready[0].rb--

* 1. Tại current\_time=8, current\_time++, ready[0].rb-- , đồng thời kiểm tra thấy current\_time==plist[3(==pi)].arr, ta đẩy plist[3] vào trong hàng đợi ready cùng với đó sort ready queue theo rb, pi++

⇨ ready=[[4,9,3,3], [3,5,8,6,7], [1,0,12,12]]

* 1. Tại current\_time=9, kiểm tra thấy ready[0].bur=ready[0].rb(=3) do đó ready[0].star=9

⇨ ready=[[4,9,3,3,9], [3,5,8,6,7], [1,0,12,12]], current\_time++, ready[0].rb--

* 1. Tại current\_time=11, current\_time+=1, ready[0].rb--. Ta kiểm tra thấy current\_time==plist[4(==pi)].arr, ta đẩy plist[4] vào trong hàng đợi ready cùng với đó sort ready queue theo rb, pi++. Đồng thời kiểm tra thấy ready[0].rb==0. Do đó đồng thời tính toán finish, wt, rt, tat, push ready[0] vào terminate đồng thời pop ready[0] ra khỏi hàng đợi ready, i+=1

⇨ terminate=[[2,0,7,0,0,7,0,0,7], [4,9,3,0,9,12,0,0,3]], ready=[[3,5,8,6,7], [5,12,6,6], [1,0,12,12 ]]

* 1. Tại current\_time=17, current\_time+=1, ready[0].rb--. Ta kiểm tra thấy có ready[0].rb=0 ( ready[0]=[3,5,8,6,7] . Do đó đồng thời tính toán finish, wt, rt, tat, push ready[0] vào terminate đồng thời pop ready[0] ra khỏi hàng đợi ready, i+=1

⇨ terminate=[[2,0,7,0,0,7,0,0,7], [4,9,3,0,9,12,0,0,3], [3,5,8,0,7,18,5,2,13]], ready=[[5,12,6,6], [1,0,12,12]]

* 1. Tại current\_time=18. Ta kiểm tra thấy ready[0].bur=ready[0].rb(=6) do đó ready[0].star=18

⇨ ready=[[5,12,6,6,18], [1,0,12,12]], current\_time+=1, ready[0].rb--

* 1. Tại current\_time=23, current\_time+=1, ready[0].rb--. Ta kiểm tra thấy có ready[0].rb=0(ready[0]=[5,12,6,0,18]). Do đó đồng thời tính toán finish, wt, rt, tat, push ready[0] vào terminate đồng thời pop ready[0] ra khỏi hàng đợi ready, i+=1

⇨ terminate=[[2,0,7,0,0,7,0,0,7], [4,9,3,0,9,12,0,0,3], [3,5,8,0,7,18,5,2,13], [5,12,6,0,18,24,6,6,12]], ready=[1,0,12,12]

* 1. Tại current\_time=24. Ta kiểm tra thấy ready[0].bur=ready[0].rb(=12) do đó ready[0].star=24

⇨ ready=[[1,0,12,12,24]], current\_time+=1, ready[0].rb--

* 1. Tại current\_time=35, current\_time+=1, ready[0].rb--. Ta kiểm tra thấy ready=[ [1,0,12,0,24] ] thấy có ready[0].rb=0. Do đó đồng thời tính toán finish, wt, rt, tat, push ready[0] vào terminate đồng thời pop ready[0] ra khỏi hàng đợi ready, i+=1,

⇨ terminate=[[2,0,7,0,0,7,0,0,7], [4,9,3,0,9,12,0,0,3], [3,5,8,0,7,18,5,2,13], [5,12,6,0,18,24,6,6,12], [1,0,12,0,24,36,24,24,36]], ready=[]

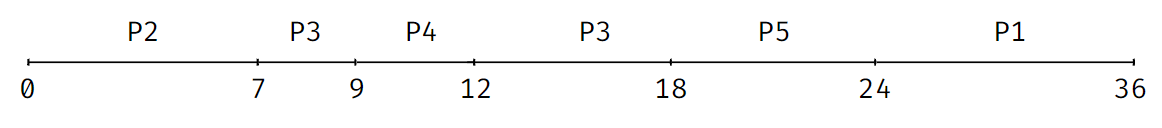
1. Ta kiểm tra i==n(5==5) thoát khỏi vòng lặp đồng thời tính avewt=7, avert=6.4, avetat=14.2
2. Xuất kết quả ra ngoài màn hình
3. Kết thúc

*Kết quả sau khi chạy tay (lưu trong term[]):*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Process name | Arrival time | Burst time | Remaining burst | Start | Finish | WT | RT | TAT |
| 2 | 0 | 7 | 0 | 0 | 7 | 0 | 0 | 7 |
| 4 | 9 | 3 | 0 | 9 | 12 | 0 | 0 | 3 |
| 3 | 5 | 8 | 0 | 7 | 18 | 5 | 2 | 13 |
| 5 | 12 | 6 | 0 | 18 | 24 | 6 | 6 | 12 |
| 1 | 0 | 12 | 0 | 24 | 36 | 24 | 24 | 36 |

avewt=7, avert=6.4, avetat=14.2

*So sánh với kết quả giải tay test case trên (thứ tự process name lần lượt từ 1 đến 5):*



start[ ]= { 24, 0, 7, 9, 18}

fisnish[ ] ={36, 7, 18, 12, 24}

rt[ ]={ 24, 0, 2, 0, 6 }

* avert= 6.4

wt[ ]={ 24, 0, 5, 0,6 }

* avewt= 7

tat [ ]={ 36,7, 13, 3, 12 }

avert= 14.2

***⮚ Có thể thấy, kết quả thực hiện chạy tay trên lưu đồ giống với kết quả giải tay. Vì vậy, ta kết luận lưu đồ đúng.***

* 1. **Thực hiện code cho giải thuật**

Text

Description automatically generated

*Hình 20. Tạo cấu trúc Process và Ready Queue trong giải thuật SRTF*

* Struct P chứa những thông tin của một process, gồm: tên, arrival time, burst time, start time, finish time, thời gian đáp ứng, thời gian đợi, thời gian hoàn thành và burst time còn lại (remaining burst time). Hai con trỏ \*print và \*set lần lượt thông qua hàm printP để xuất thông tin của một process ra màn hình, và setP để tính toán RT, WT, TAT.
* Struct ReadyQueue gồm một mảng process và kích cỡ của hàng đợi.

Text

Description automatically generated

*Hình 21. Hàm khởi tạo Ready Queue, hàm tính RT, WT, TAT và hàm in Process trong giải thuật SRTF*

* Hàm init() khởi tạo hàng đợi Ready ban đầu không có phần tử nào.
* Các hàm setP() và printP() thực hiện tính toán và xuất kết quả như hình 21.

Text

Description automatically generated

*Hình 22. Hàm sắp xếp Process theo arrival time tăng dần trong giải thuật SRTF*

Text

Description automatically generated

*Hình 23. Hàm sắp xếp Process theo remaining burst time tăng dần trong giải thuật SRTF*

* Hàm SortRB() sắp xếp một mảng process tăng dần theo remainingburst time. Hàm này sử dụng giải thuật Insertion Sort thay vì sắp xếp thông thường, để khi một process xuất hiện mà có burst time bằng với remaining burst time của process khác đã ở sẵn trong hàng đợi, nó sẽ không tranh mất vị trí của process đến trước mà đứng đợi ngay sau process đến trước đó.

Text

Description automatically generated

*Hình 24. Hàm thêm một process và xóa process đầu tiên* *ra khỏi hàng đợi Ready trong giải thuật SRTF*

* Hàm pushP() và popP() lần lượt thêm một process và xóa process đầu tiên ra khỏi hàng đợi Ready, khi đó kích cỡ của hàng đợi cũng sẽ tăng hoặc giảm đi một phần tử tương ứng.

Text

Description automatically generated

*Hình 25. Khởi tạo mảng process input, hàng đợi Ready, mảng terminated và các biến để tính tổng RT, WT, TAT trong giải thuật SRTF*

Text

Description automatically generated

*Hình 26. Nhập input, khởi tạo giá trị ban đầu cho rb và sắp xếp mảng process input theo arrival time*

* Sau khi khởi tạo các mảng và hàng đợi, ta tiến hành nhập input gồm số lượng process n, tên, arrival time và burst time của từng process. Ta khởi tạo giá trị ban đầu cho rb (remaining burst time) bằng burst time của chính process đó.
* Các process nhập vào sẽ được lưu trong mảng plist, ta sắp xếp mảng này tăng dần theo arrival time.

Text

Description automatically generated

*Hình 27. Đưa tất cả những process cùng xuất hiện sớm nhất vào hàng đợi Ready*

* Sử dụng vòng lặp while để push tất cả những process có cùng arrival time với process đến sớm nhất vào hàng đợi Ready. Sau đó, ta tiến hành sắp xếp toàn bộ hàng đợi Ready theo remaining burst time.
* Sau khi push xong những process này, i sẽ chỉ mục đến process có arrival time nhỏ thứ 2 trong plist.

Text

Description automatically generated

*Hình 28. Khởi tạo giá trị ban đầu cho current\_time, pi, i và tính start time cho process đang chạy*

* Ta giả định hàng đợi Ready là một mảng các process, và phần tử đầu tiên trong mảng này luôn là process đang chạy.
* Biến current\_time chứa giá trị thời gian hiện tại, dùng để mô phỏng cách hàng đợi Ready làm việc trong thời gian thực. Ta khởi tạo giá trị ban đầu của current\_time là thời điểm xuất hiện của process đầu tiên được chọn để thực thi (process đến sớm nhất và có burst time nhỏ nhất so với các process đến cùng lúc).
* Biến pi chỉ mục đến các process trong plist chưa được thêm vào hàng đợi Ready (vì chúng chưa xuất hiện). pi bắt đầu từ vị trí i.
* Biến i đặt lại bằng 0, chỉ mục đến các process trong mảng term[] (là mảng chứa các process đã hoàn thành thực thi).
* Ta bắt đầu xử lý định thời CPU, dừng lại khi tất cả các process trong plist đều đã được thêm vào term[].
* Tại thời điểm current\_time, nếu process đầu tiên trong hàng đợi có bur bằng rb (tức nó chưa ở trong trạng thái running trước đó), current\_time chính là start time của nó.
* current\_time tăng thêm 1, cùng lúc burst time của process đang chạy giảm đi 1.

Text

Description automatically generated

*Hình 29. Đưa process vừa xuất hiện vào hàng đợi Ready và sắp xếp, thực hiện các thao tác đối với process đang chạy nếu nó hoàn thành thực thi*

* Tại thời điểm current\_time mới này, ta sử dụng vòng lặp while để thêm tất cả những process đến tại thời điểm này (arrival time == current\_time) vào hàng đợi Ready và tiến hành sắp xếp lại hàng đợi theo burst time của chúng. Ta *thực hiện sắp xếp trên toàn bộ hàng đợi, vì process đầu tiên đang thực thi vẫn có thể bị ngắt trong giải thuật SRTF, nếu remaining burst time của nó lớn hơn burst time của một process khác trong hàng đợi*.
* Nếu tại current\_time, không có process nào đến, ta thoát khỏi vòng lặp để tiếp tục đếm current\_time.
* Cũng tại thời điểm này, nếu process đang chạy có rb == 0 (tức nó đã chạy xong), finish time của nó chính bằng thời điểm hiện tại. Ta thực hiện tính RT, WT, TAT cho nó thông qua hàm set() và thêm nó vào vị trí i của mảng term[]. Sau đó, ta tăng i lên 1 và xóa process đó ra khỏi hàng đợi Ready. Lúc này, process tiếp theo sẽ được đẩy lên vị trị đầu tiên trong hàng đợi và thực thi.

Text

Description automatically generated

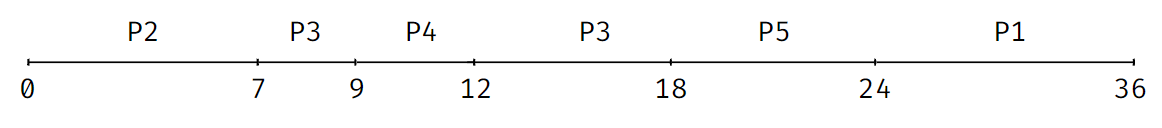
*Hình 30. Thực hiện tính toán RT, WT, TAT trung bình và xuất kết quả ra màn hình*

* 1. **Trình bày tính đúng đắn của code bằng cách chạy ít nhất 03 test case, mỗi test case 5 tiến trình**

*Test case 1:* n=5

|  |  |  |
| --- | --- | --- |
| Process name | Arrival Time | Burst time |
| 1 | 0 | 12 |
| 2 | 0 | 7 |
| 3 | 5 | 8 |
| 4 | 9 | 3 |
| 5 | 12 | 6 |

*Kết quả giải tay:*



*Hình 31. Giản đồ Gantt cho test case 1 giải thuật SRTF*

start[ ]= { 24, 0, 7, 9, 18}

fisnish[ ] ={36, 7, 18, 12, 24}

rt[ ]={ 24, 0, 2, 0, 6 }

* avert= 6.4

wt[ ]={ 24, 0, 5, 0,6 }

* avewt= 7

tat [ ]={ 36,7, 13, 3, 12 }

* avert= 14.2

A picture containing graphical user interface

Description automatically generated

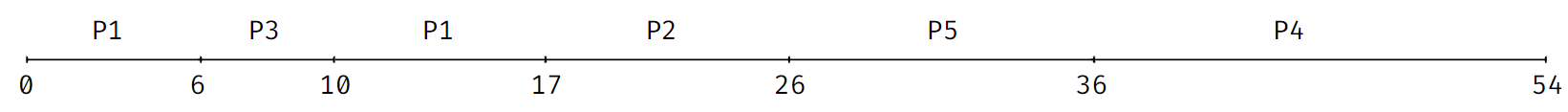
*Hình 32. Chạy test case 1 bằng code giải thuật SRTF*

***⮚ Kết quả thực hiện bằng code giống với kết quả giải tay.***

*Test case 2:* n=5

|  |  |  |
| --- | --- | --- |
| Process name | Arrival Time | Burst time |
| 1 | 0 | 13 |
| 2 | 4 | 9 |
| 3 | 6 | 4 |
| 4 | 7 | 18 |
| 5 | 12 | 10 |

*Kết quả giải tay:*



*Hình 33. Giản đồ Gantt cho test case 2 giải thuật SRTF*

start[ ]= { 0, 17, 6, 36, 26 }

fisnish[ ] ={17,26,10, 54, 36}

rt[ ]={ 0, 13,0, 29, 14 }

* avert= 11.2

wt[ ]={ 4, 13,0, 29, 14 }

* avert= 12

tat [ ]={ 17, 22, 4, 47, 24 }

* avert= 22.8

Calendar

Description automatically generated

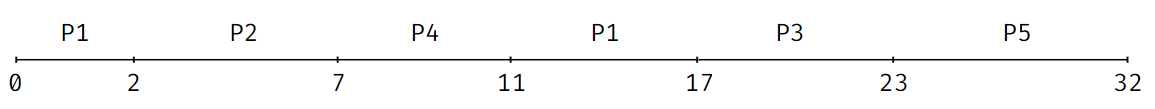
*Hình 34. Chạy test case 2 bằng code giải thuật SRTF*

***⮚ Kết quả thực hiện bằng code giống với kết quả giải tay.***

*Test case 3:* n=5

|  |  |  |
| --- | --- | --- |
| Process name | Arrival Time | Burst time |
| 1 | 0 | 8 |
| 2 | 2 | 5 |
| 3 | 3 | 6 |
| 4 | 5 | 4 |
| 5 | 7 | 9 |

*Kết quả giải tay:*



*Hình 35. Giản đồ Gantt cho test case 3 giải thuật SRTF*

start[ ]= {0,2,17,7,23}

finish[ ]= {17,7,23,11,32}

rt[ ]= {0,0,14,2,16}

* avert=6.4

wt[ ]= {9,0,14,2,16}

* averwt=8.2

tat[ ]= {17,5,20,6,25}

* avertat=14.6

Calendar

Description automatically generated

*Hình 36.* *Chạy test case 3 bằng code giải thuật SRTF*

***⮚ Kết quả thực hiện bằng code giống với kết quả giải tay.***

***⮚ Cả 3 test case đều cho kết quả giống với khi giải tay. Vì vậy, ta kết luận code đúng.***

***Ex 3. Giải thuật RR (Round Robin):***

* 1. **Vẽ lưu đồ giải thuật**

**Diagram

Description automatically generated with low confidence**

**Diagram

Description automatically generated with low confidence**

**Diagram

Description automatically generated with low confidence**

*Hình 37. Lưu đồ giải thuật RR*

* 1. **Trình bày tính đúng đắn của lưu đồ bằng cách chạy tay ít nhất 01 test case tối thiểu 05 tiến trình**

*Test case:* n=5

|  |  |  |
| --- | --- | --- |
| Process name | Arrival Time | Burst time |
| 1 | 0 | 11 |
| 2 | 6 | 4 |
| 3 | 2 | 8 |
| 4 | 5 | 16 |
| 5 | 11 | 6 |

Với q=7

Trong minh họa giải thuật dưới đây, một process sẽ được trình bày các thuộc tính lần lượt theo thứ tự [process name, arrival time, burst time, remaining burst time, start, finish, wt, rt, tat]

*Thực hiện chạy tay test case trên:*

1. Sau quá trình nhập và khởi tạo: plist=[[1,0,11,11], [2,6,4,4], [3,2,8,8], [4,5,16,16], [5,11,6,6]], n=5, q=7
2. Ta sử dụng sort theo arr plist=[[1,0,11,11], [3,2,8,8], [4,5,16,16], [2,6,4,4], [5,11,6,6]]
3. Khởi tạo biến i=0, đồng thời sử dụng vòng lặp để đẩy những plist[i] có cùng arr với plist[0] vào hàng đợi ready, đồng thời i+=1 ⇨ ready=[[1,0,11,11]], i=1
4. Khởi tạo current\_time=0 (=ready[0].arr), i=0, pi=1 (i=1), exe\_time=0
5. Sử dụng vòng lặp khi i<n (i<5):
   1. Tại current\_time =0, gán exe\_time=1, kiểm tra thấy ready[0].bur=ready[0].rb (11=11) ⇨ ready[0].start=0 (=current\_time )
   2. Thực hiện vòng lặp trong khi exe\_time<=q (1<=7) :
      * Khi exe\_time=1, current\_time++, ready[0].rb--, exe\_time++(\*)
      * Khi exe\_time=2, current\_time++, ready[0].rb--, exe\_time++. Lúc này current\_time= 2 kiểm tra điều kiện current\_time=plist[pi(1)].arr (2==2) ta đẩy plist[1] vào hàng đợi ready và pi++
        + ready=[[1,0,11,9,0], [3,2,8,8]]
      * Từ exe\_time=3 -> 4 thực hiện tương tự (\*)
      * Khi exe\_time=5, current\_time++, ready[0].rb--, exe\_time++. Lúc này current\_time=5 kiểm tra điều kiện current\_time=plist[pi(2)].arr (5==5) ta đẩy plist[2] vào hàng đợi ready và pi++ ⇨ ready=[[1,0,11,6,0], [3,2,8,8], [4,5,16,16]]
      * Khi exe\_time=6, current\_time++, ready[0].rb--, exe\_time++. Lúc này current\_time=6, kiểm tra điều kiện current\_time=plist[ pi(3)].arr đẩy vào hàng đởi ready và pi++
        + ready=[[1,0,11,5,0], [3,2,8,8], [4,5,16,16], [2,6,4,4]]
      * Khi exe\_time=7, thực hiện tương tự (\*)
      * Khi exe\_time=8, ta thoát khỏi vòng lặp do kiểm tra thấy điều kiện vòng lặp sai, cùng với đó ta đưa ready[0] về cuối hàng đợi ready
        + ready =[[3,2,8,8], [4,5,16,16], [2,6,4,4], [ 1,0,11,4,0]]
   3. Tại current\_time =7, gán exe\_time=1, kiểm tra thấy ready[0].bur=ready[0].rb (8=8) ⇨ ready[0].start=7 (=current\_time )
   4. Thực hiện vòng lặp trong khi exe\_time<=q (1<=7) :
      * Khi exe\_time=1->3, current\_time++, ready[0].rb--, exe\_time++.
      * Khi exe\_time=4 , current\_time++, ready[0].rb--, exe\_time++. Lúc này current\_time=11 kiểm tra điều kiện current\_time=plist[pi(4)] đẩy vào hàng đợi ready và pi++(pi=5)
        + ready=[[3,2,8,4,7], [4,5,16,16], [2,6,4,4], [1,0,11,4,0], [5,11,6,6]], không cần kiểm tra điều kiện để push plist vào ready nữa (vì pi=n)
      * Khi exe\_time=5->7, current\_time++, ready[0].rb--, exe\_time++.
      * Khi exe\_time=8, ta thoát khỏi vòng lặp, cùng với đó đưa ready[0] về cuối hàng đợi
        + ready=[[4,5,16,16], [2,6,4,4], [1,0,11,4,0], [5,11,6,6], [3,2,8,1,7]]
   5. Tại current\_time=14, gán exe\_time=1 kiểm tra thấy ready[0].bur=ready[0].rb (16=16)
      * + ready[0].start=14 (=current\_time ) .
   6. Thực hiện vòng lặp trong khi exe\_time<=q (1<=7):
      * Khi exe\_time=1->7 current\_time++, ready[0].rb--, exe\_time++.
      * Khi exe\_time=8, ta thoát khỏi vòng lặp, cùng với đó đưa ready[0] về cuối hàng đợi
        + ready=[[2,6,4,4], [1,0,11,4,0], [5,11,6,6], [3,2,8,1,7], [4,5,16,9,14 ]]
   7. Tại current\_time=21, gán exe\_time=1 kiểm tra thấy ready[0].bur=ready[0].rb(4=4)
      * + ready[0].start=21 (=current\_time)
   8. Thực hiện vòng lặp trong khi exe\_time<=q (1<=7):
      * Khi exe\_time=1->3 current\_time++, ready[0].rb--, exe\_time++.
      * Khi exe\_time=4 current\_time++, ready[0].rb--, exe\_time++. Khi này ready[0].rb=0 vì vậy ta thoát khỏi vòng lặp đồng thời tính finish, wt, rt, tat của ready[0] đồng thời pop ready[0] ra khỏi hàng đợi đưa nó vào hàng đợi terminate, i+=1
        + terminate=[[2,6,4,0,21,25,15,15,19]], ready=[[1,0,11,4,0], [5,11,6,6], [3,2,8,1,7], [4,5,16,9,14 ]]
   9. Tại current\_time=25, gán exe\_time=1
   10. Thực hiện vòng lặp trong khi exe\_time<=q (1<=7):
       * Khi exe\_time=1->3 current\_time++, ready[0].rb--, exe\_time++
       * Khi exe\_time=4 current\_time++, ready[0].rb--, exe\_time++. Khi này ready[0].rb=0 vì vậy ta thoát khỏi vòng lặp đồng thời tính finish, wt, rt, tat của ready[0] đồng thời pop ready[0] ra khỏi hàng đợi đưa nó vào hàng đợi terminate, i+=1

⇨ terminate=[[2,6,4,0,21,25,15,15,19], [1,0,11,4,0,29,18,0,29]], ready=[[5,11,6,6], [3,2,8,1,7], [4,5,16,9,14]]

* + - Tại current\_time=29, gán exe\_time=1 kiểm tra thấy ready[0].bur=ready[0].rb(6=6)
      * ready[0].start=29 (=current\_time)
  1. Thực hiện vòng lặp trong khi exe\_time<=q (1<=7):
  2. Khi exe\_time=1->5 current\_time++, ready[0].rb--, exe\_time++
     + Khi exe\_time=6 current\_time++, ready[0].rb--, exe\_time++. Khi này ready[0].rb=0 vì vậy ta thoát khỏi vòng lặp đồng thời tính finish, wt, rt, tat của ready[0] đồng thời pop ready[0] ra khỏi hàng đợi đưa nó vào hàng đợi terminate, i+=1

⇨ terminate=[[2,6,4,0,21,25,15,15,19], [1,0,11,4,0,29,18,0,29], [5,11,6,0,29,35,18,18,24]], ready=[[3,2,8,1,7], [4,5,16,9,14]]

* 1. Tại current\_time=35, gán exe\_time=1
  2. Thực hiện vòng lặp trong khi exe\_time<=q (1<=7):
     + Khi exe\_time=1 current\_time++, ready[0].rb--, exe\_time++. Khi này ready[0].rb=0 vì vậy ta thoát khỏi vòng lặp đồng thời tính finish, wt, rt, tat của ready[0] đồng thời pop ready[0] ra khỏi hàng đợi đưa nó vào hàng đợi terminate, i+=1

⇨ terminate=[ [2,6,4,0,21,25,15,15,19], [1,0,11,4,0,29,18,0,29], [5,11,6,0,29,35,18,18,24], [3,2,8,0,7,36,26,5,34]], ready=[[4,5,16,9,14]]

* 1. Tại current\_time=36, gán exe\_time=1
  2. Thực hiện vòng lặp trong khi exe\_time<=q (1<=7):
     + Khi exe\_time=1->7 current\_time++, ready[0].rb--, exe\_time++.
     + Khi exe\_time=8, ta thoát khỏi vòng lặp do kiểm tra thấy điều kiện vòng lặp sai, cùng với đó ta đưa ready[0] về cuối hàng đợi ready
       - ready=[[4,5,16,3,14]]
  3. Tại current\_time=42, gán exe\_time=1
  4. Thực hiện vòng lặp trong khi exe\_time<=q (1<=7):
     + Khi exe\_time=1 ->2 current\_time++, ready[0].rb--, exe\_time++.
     + Khi exe\_time=3 Khi này ready[0].rb=0 vì vậy ta thoát khỏi vòng lặp đồng thời tính finish, wt, rt, tat của ready[0] đồng thời pop ready[0] ra khỏi hàng đợi đưa nó vào hàng đợi terminate, i+=1
       - terminate=[[2,6,4,0,21,25,15,15,19], [1,0,11,4,0,29,18,0,29], [5,11,6,0,29,35,18,18,24], [3,2,8,0,7,36,26,5,34], [4,5,16,0,14,45,24,9,40]]

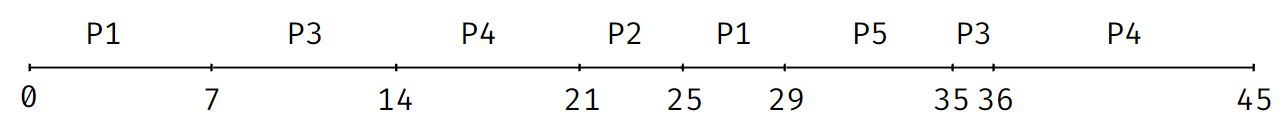
1. Ta kiểm tra i==n (5==5) thoát khỏi vòng lặp đồng thời tính avewt=20.2, avert=9.4, avetat =29.2
2. Xuất kết quả ra ngoài màn hình
3. Kết thúc

*Kết quả sau khi chạy tay (lưu trong term[]):*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Process name | Arrival time | Burst time | Remaining burst | Start | Finish | WT | RT | TAT |
| 2 | 6 | 4 | 0 | 21 | 25 | 15 | 15 | 19 |
| 1 | 0 | 11 | 4 | 0 | 29 | 18 | 0 | 29 |
| 5 | 11 | 6 | 0 | 29 | 35 | 18 | 18 | 24 |
| 3 | 2 | 8 | 0 | 7 | 35 | 26 | 5 | 34 |
| 4 | 5 | 16 | 0 | 14 | 45 | 24 | 9 | 36 |

avewt=20.2, avert=9.4, avetat =29.2

*So sánh với kết quả giải tay test case trên (thứ tự process name lần lượt từ 1 đến 5):*



start[ ]= { 0, 21, 7, 14, 29 }

fisnish[ ] ={29,25,36,45,35}

rt[ ]={ 0, 15, 5, 9, 18 }

* avert= 9.4

wt[ ]={ 18,15,26,24,18 }

* avewt= 20.2

tat [ ]={ 29,19,34,40,24 }

* avetat= 29.2

***⮚ Có thể thấy, kết quả thực hiện chạy tay trên lưu đồ giống với kết quả giải tay. Vì vậy, ta kết luận lưu đồ đúng.***

* 1. **Thực hiện code cho giải thuật**

Text

Description automatically generated

*Hình 38. Tạo cấu trúc Process và Ready Queue trong giải thuật RR*

* Struct P chứa những thông tin của một process, gồm: tên, arrival time, burst time, start time, finish time, thời gian đáp ứng, thời gian đợi, thời gian hoàn thành và burst time còn lại (remaining burst time). Hai con trỏ \*print và \*set lần lượt thông qua hàm printP để xuất thông tin của một process ra màn hình, và setP để tính toán RT, WT, TAT.
* Struct ReadyQueue gồm một mảng process và kích cỡ của hàng đợi.

Text

Description automatically generated

*Hình 39. Hàm khởi tạo Ready Queue, hàm tính RT, WT, TAT và hàm in Process trong giải thuật RR*

* Hàm init() khởi tạo hàng đợi Ready ban đầu không có phần tử nào.
* Các hàm setP() và printP() thực hiện tính toán và xuất kết quả như hình 39.

Text

Description automatically generated

*Hình 40. Hàm sắp xếp Process theo arrival time tăng dần trong giải thuật RR*

Text

Description automatically generated

*Hình 41. Hàm thêm một process và xóa process đầu tiên* *ra khỏi hàng đợi Ready trong giải thuật RR*

* Hàm pushP() và popP() lần lượt thêm một process và xóa process đầu tiên ra khỏi hàng đợi Ready, khi đó kích cỡ của hàng đợi cũng sẽ tăng hoặc giảm đi một phần tử tương ứng.

Text

Description automatically generated

*Hình 42. Hàm di chuyển process đầu tiên* *về cuối hàng đợi Ready trong giải thuật RR*

* Hàm moveP() di chuyển process đầu tiên xuống cuối hàng đợi, do hết quantum time để nhường CPU cho process tiếp theo. Kích cỡ hàng đợi qua hàm này không thay đổi.

Text

Description automatically generated

*Hình 43. Khởi tạo mảng process input, hàng đợi Ready, mảng terminated và các biến để tính tổng RT, WT, TAT trong giải thuật RR*

Text

Description automatically generated

*Hình 44. Nhập input, khởi tạo giá trị ban đầu cho rb và sắp xếp mảng process input theo arrival time*

* Sau khi khởi tạo các mảng và hàng đợi, ta tiến hành nhập input gồm số lượng process n, tên, arrival time và burst time của từng process. Ta khởi tạo giá trị ban đầu cho rb (remaining burst time) bằng burst time của chính process đó.
* Các process nhập vào sẽ được lưu trong mảng plist, ta sắp xếp mảng này tăng dần theo arrival time.
* Khai báo và nhập giá trị cho quantum time q.

Text

Description automatically generated

*Hình 45. Đưa tất cả những process cùng xuất hiện sớm nhất vào hàng đợi Ready*

* Sử dụng vòng lặp while để push tất cả những process có cùng arrival time với process đến sớm nhất vào hàng đợi Ready.
* Sau khi push xong những process này, i sẽ chỉ mục đến process có arrival time nhỏ thứ 2 trong plist.

Text

Description automatically generated

*Hình 46. Khởi tạo giá trị ban đầu cho current\_time, exe\_time, pi, i*

* Ta giả định hàng đợi Ready là một mảng các process, và phần tử đầu tiên trong mảng này luôn là process đang chạy.
* Biến current\_time chứa giá trị thời gian hiện tại, dùng để mô phỏng cách hàng đợi Ready làm việc trong thời gian thực. Ta khởi tạo giá trị ban đầu của current\_time là thời điểm xuất hiện của process đầu tiên được chọn để thực thi (process đến sớm nhất).
* Biến pi chỉ mục đến các process trong plist chưa được thêm vào hàng đợi Ready (vì chúng chưa xuất hiện). pi bắt đầu từ vị trí i.
* Biến i đặt lại bằng 0, chỉ mục đến các process trong mảng term[] (là mảng chứa các process đã hoàn thành thực thi).
* Ta bắt đầu xử lý định thời CPU, dừng lại khi tất cả các process trong plist đều đã được thêm vào term[].
* Biến exe\_time đếm thời gian process chạy, đảm bảo mỗi lần sử dụng CPU của nó không vượt quá quantum time. Mỗi khi một process chạy, ta đặt lại giá trị của exe\_time=1.

Text

Description automatically generated

*Hình 47. Tính start time, đếm exe\_time và đưa process vừa xuất hiện vào hàng đợi Ready.*

* Tại thời điểm current\_time, nếu process đầu tiên trong hàng đợi có bur bằng rb (tức nó chưa ở trong trạng thái running trước đó), current\_time chính là start time của nó.
* Tiến hành đếm exe\_time: current\_time tăng thêm 1, cùng lúc burst time của process đang chạy giảm đi 1, và exe\_time của nó tăng lên 1.
* Tại thời điểm current\_time mới này, ta sử dụng vòng lặp while để thêm tất cả những process đến tại thời điểm này (arrival time == current\_time) vào hàng đợi Ready.
* Nếu tại current\_time, không có process nào đến, ta thoát khỏi vòng lặp để tiếp tục đếm current\_time và exe\_time.
* Cũng tại thời điểm này, nếu process đang chạy có rb == 0 (tức nó đã chạy xong), ta thoát khỏi vòng lặp và thực hiện tính toán, dù exe\_time chưa vượt quá q.

Text

Description automatically generated

*Hình 48. Xóa process khỏi hàng đợi hoặc đưa về cuối hàng khi hết quantum time*

* finish time của process đang chạy chính bằng thời điểm hiện tại. Ta tính RT, WT, TAT cho nó thông qua hàm set() và thêm nó vào vị trí i của mảng term[]. Sau đó, ta tăng i lên 1 và xóa process đó ra khỏi hàng đợi Ready.
* Nếu process chưa chạy xong nhưng exe\_time của nó đã vượt quá q, ta thực hiện di chuyển nó về cuối hàng đợi. Lúc này, process tiếp theo sẽ được đẩy lên vị trị đầu tiên trong hàng đợi và thực thi.

Text

Description automatically generated

*Hình 49. Thực hiện tính toán RT, WT, TAT trung bình và xuất kết quả ra màn hình*

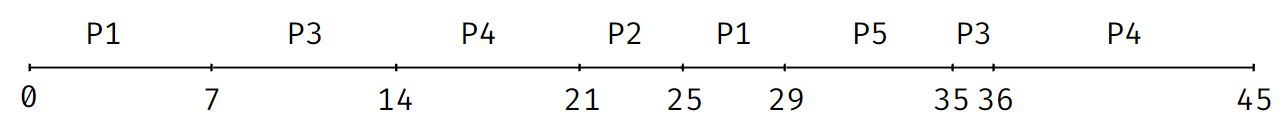
* 1. **Trình bày tính đúng đắn của code bằng cách chạy ít nhất 03 test case, mỗi test case 5 tiến trình**

*Test case 1:* n=5

|  |  |  |
| --- | --- | --- |
| Process name | Arrival Time | Burst time |
| 1 | 0 | 11 |
| 2 | 6 | 4 |
| 3 | 2 | 8 |
| 4 | 5 | 16 |
| 5 | 11 | 6 |

q=7

*Kết quả giải tay:*



*Hình 50. Giản đồ Gantt cho test case 1 giải thuật RR*

start[ ]= { 0, 21, 7, 14, 29 }

fisnish[ ] ={29,25,36,45,35}

rt[ ]={ 0, 15, 5, 9, 18 }

* avert= 9.4

wt[ ]={ 18,15,26,24,18 }

* avewt= 20.2

tat [ ]={ 29,19,34,40,24 }

* avetat= 29.2

A picture containing calendar

Description automatically generated

*Hình 51. Chạy test case 1 bằng code giải thuật RR*

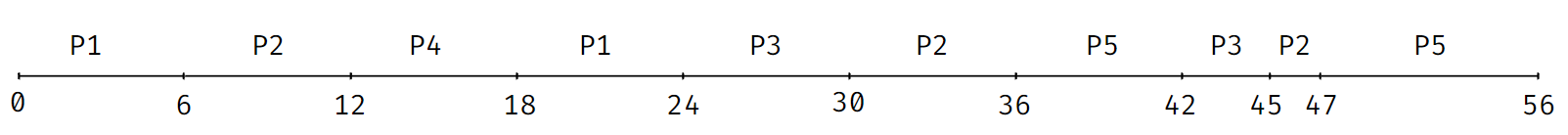
***⮚ Kết quả thực hiện bằng code giống với kết quả giải tay.***

*Test case 2:* n=5

|  |  |  |
| --- | --- | --- |
| Process name | Arrival Time | Burst time |
| 1 | 0 | 12 |
| 2 | 3 | 14 |
| 3 | 7 | 9 |
| 4 | 5 | 6 |
| 5 | 14 | 15 |

q=6

*Kết quả giải tay:*



*Hình 52. Giản đồ Gantt cho test case 2 giải thuật RR*

start[ ]= {0,6,24,12,36}

fisnish[ ] ={24,47,45,18,56}

rt[ ]={0,3,17,7,22}

* avert= 9.8

wt[ ]={ 12,30,29, 7, 27 }

* avewt= 21

tat [ ]={ 24,44,38,13,42}

* avetat= 32.2

Calendar

Description automatically generated

*Hình 53. Chạy test case 2 bằng code giải thuật RR*

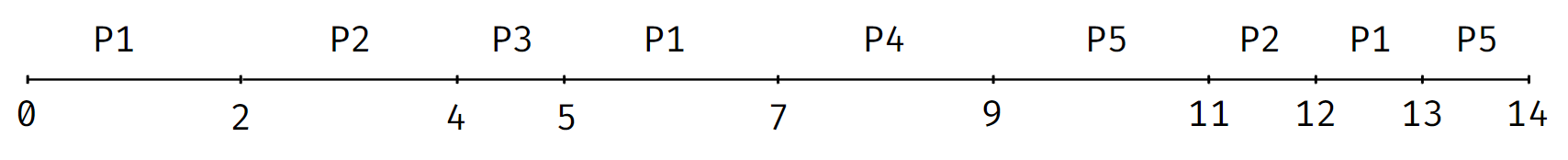
***⮚ Kết quả thực hiện bằng code giống với kết quả giải tay.***

*Test case 3:* n=5

|  |  |  |
| --- | --- | --- |
| Process name | Arrival Time | Burst time |
| 1 | 0 | 5 |
| 2 | 1 | 3 |
| 3 | 2 | 1 |
| 4 | 3 | 2 |
| 5 | 4 | 3 |

q=2

*Kết quả giải tay:*



*Hình 54. Giản đồ Gantt cho test case 3 giải thuật RR*

start[ ]= {0,2,4,7,9}

fisnish[ ] ={13,12,5,9, 14}

rt[ ]={0,1,2,4,5}

* avert= 2.4

wt[ ]={8,8,2,4,7}

* avewt=5.8

tat [ ]={13,11,3,6,10}

* avetat= 8.6

Calendar

Description automatically generated with medium confidence

*Hình 55. Chạy test case 3 bằng code giải thuật RR*

***⮚ Kết quả thực hiện bằng code giống với kết quả giải tay.***

***⮚ Cả 3 test case đều cho kết quả giống với khi giải tay. Vì vậy, ta kết luận code đúng.***