MÔN HỌC: HỆ ĐIỀU HÀNH
CÂU HỎI VÀ BÀI TẬP CHƯƠNG 6

1. Deadlock là gì?

Trong một môi trường có nhiều process chạy đồng thời, các process có thể tranh giành nhau một số lượng hữu hạn các tài nguyên (resources).
Một process P1 yêu cầu một tài nguyên A, nếu tài nguyên A lúc đó chưa sẵn sàng thì process P1 sẽ vào trạng thái waiting. Đôi khi, process P1 đã vào trạng thái waiting không bao giờ có thể chuyển trạng thái sang Ready nữa, do tài nguyên A đang bị một process P2 nắm giữ, vì P2 cũng đang ở trạng thái waiting đang chờ một tài nguyên B nào đó đang bị nắm giữa cũng bởi 1 process nào đó (có thể là P1) đang ở trạng thái waiting.
Một trường hợp chờ đợi vòng như vậy, và khi đó hệ thống không thể tiến triển được, người ta gọi là trạng thái Deadlock.

2. Các điều kiện cần để xảy ra deadlock?

- Loại trừ tương hỗ (Mutual Exclusion): có ít nhất 1 tài nguyên đang bị chiến giữ theo cơ chế không chia sẻ (nonshareable mode). Hay nói cách khác, chỉ có một process P1 duy nhất được sử dụng tài nguyên A trong một thời điểm. Nếu một process P2 khác yêu cầu tài nguyên nói trên, process P2 phải chờ đến khi tài nguyên A đã được process P1 giải phóng.

- Giữ và chờ (Hold and wait): Một process P1 phải đang giữ ít nhất một tài nguyên A và đang chờ một hoặc nhiều tài nguyên B đang bị giữ bởi một process B2 khác.

- Không trưng dụng (Non-Preemption) : Tài nguyên không thể bị lấy lại, tức là tài nguyên chỉ có thể được giải phóng khi process đang giữ nó trả lại sau khi đã hoàn thành xong công việc. - Chu trình đợi (Circular Wait) : Tồn tại một tập {P0,…,Pn} các process đang đợi sao cho:

o P0 đợi một tài nguyên mà P1 giữ.

o P1 đợi một tài nguyên mà P2 giữ.

o …

o Pn đợi một tài nguyên mà P0 giữ.

3. Đồ thị cấp phát tài nguyên là gì? Mối liên hệ giữa đồ thị cấp phát tài nguyên và deadlock?

Mối liên hệ giữa đồ thị cấp phát tài nguyên và deadlock là RAG không chứa chu trình thì không có deadlock, ngược lại thì có thể xảy ra deadlock. Đồ thị cấp phát tài nguyên là một công cụ quan trọng trong việc phát hiện và giải quyết deadlock. Nó giúp cho việc phân tích các tài nguyên và các tiến trình trong hệ thống. Cụ thể, tập đỉnh V gồm 2 loại: P với các process và R với các loại tài nguyên. Tập cạnh E gồm 2 loại: P → R: cạnh yêu cầu và R → P: cạnh cấp phát

RAG không chứa chu trình → không có deadlock

RAG chứa một hay nhiều chu trình

+ Nếu mỗi loại tài nguyên chỉ có 1 thực thể → deadlock

+ Nếu mỗi loại tài nguyên có nhiều thực thể → có thể xảy ra deadlock

4. Có mấy phương pháp để giải quyết deadlock? Phân tích và đánh giá ưu, nhược điểm của từng phương pháp?

Dự đoán và Phòng ngừa (Prediction and Prevention):

* Ưu điểm:

- Được thực hiện trước khi deadlock xảy ra.

- Giảm nguy cơ xảy ra deadlock.

* Nhược điểm:

- Yêu cầu thông tin chi tiết về tất cả các tài nguyên được sử dụng.

- Khó khăn trong việc dự đoán một cách chính xác. 2. Phát hiện và Giải quyết (Detection and Resolution):

* Ưu điểm:

- Cho phép hệ thống tiếp tục hoạt động trong khi deadlock có thể xảy ra.

- Dễ dàng triển khai.

* Nhược điểm:

- Phải đợi đến khi deadlock xảy ra để giải quyết vấn đề.

- Chi phí thời gian và tài nguyên để phát hiện deadlock và giải quyết nó

5. Phân tích và đánh giá ưu, nhược điểm của các giải pháp đồng bộ busy waiting (cả phần cứng và phần mềm)?

Ưu điểm:

Phần Cứng (Hardware-Based):

- Đồng bộ Busy Waiting thực hiện trên phần cứng có thể đạt được hiệu suất cao hơn so với giải pháp phần mềm, do đặc tính thấp cấp của phần cứng.

- Phần cứng có thể phản ứng nhanh chóng khi có sự kiện đồng bộ xảy ra.

Phần Mềm (Software-Based):

- Giải pháp phần mềm có thể được triển khai trên nhiều nền tảng và kiến trúc khác nhau.

- Cài đặt và duy trì giải pháp phần mềm thường dễ dàng hơn so với giải pháp phần cứng.

Nhược Điểm:

Phần Cứng:

- Giải pháp phần cứng thường yêu cầu sử dụng các nguồn tài nguyên hơn, có thể làm giảm hiệu suất tổng thể của hệ thống.

- Các giải pháp phần cứng có thể khó mở rộng để đáp ứng yêu cầu mới mà không cần sự can thiệp sâu rộng vào phần cứng.

Phần Mềm:

- Giải pháp phần mềm thường yêu cầu thời gian xử lý lớn hơn so với giải pháp phần cứng, do đó có thể làm giảm hiệu suất hệ thống.

- Giải pháp phần mềm thường phụ thuộc vào hệ điều hành, điều này có thể tạo ra vấn đề tương thích khi chuyển đổi giữa các nền tảng.

6. Trạng thái an toàn là gì? Mối liên hệ giữa trạng thái an toàn và deadlock?

Một trạng thái của hệ thống được gọi là an toàn (safe) nếu tồn tại một chuỗi thứ tự an toàn

- Một chuỗi tiến trình là một chuỗi an toàn nếu

• Với mọi i = 1, …, n yêu cầu tối đa về tài nguyên của Pi có thể được thỏa bởi

• Tài nguyên mà hệ thống đang có sẵn sàng

• Cùng với tài nguyên mà tất cả các Pj (j < i) đang giữ

• Một trạng thái của hệ thống được gọi là không an toàn (unsafe) nếu không tồn tại một chuỗi an toàn

• Nếu hệ thống đang ở trạng thái safe -> không deadlock

• Nếu hệ thống đang ở trạng thái unsafe -> có thể dẫn đến deadlock

• Tránh deadlock bằng cách bảo đảm hệ thống không đi đến trạng thái unsafe

7. Mô tả cách thực hiện các giải thuật Banker: giải thuật an toàn, giải thuật yêu cầu tài nguyên và giải thuật phát hiện deadlock?

• Giải thuật an toàn:

- Khi một tiến trình vào hệ thống, tiến trình đó cần phải đưa ra số lượng thực thể tối đa mà nó cần (để từ đó hệ thống có thể xác định trạng thái an toàn, dựa vào việc thoả mãn yêu cầu của tất cả process). Con số này không được vượt quá số lượng tài nguyên hiện đang có trong hệ thống.

- Khi một process yêu cầu tài nguyên, hệ thống sẽ xem xét : Liệu sau khi cấp tài nguyên thì hệ thống có còn đang ở trạng thái an toàn hay không ? Nếu có thì hệ thống sẽ cấp, ngược lại hệ thống sẽ không cấp tài nguyên cho process. Lúc đó process phải chờ đến khi các process khác giải phóng đủ số lượng tài nguyên cần thiết.

• Giải thuật yêu cầu tài nguyên:

- Nếu Request(i) <= Need(i) thì đến bước 2. Nếu không, báo lỗi vì tiến trình đã vượt yêu cầu tối đa.

- Nếu Request(i) <= Available thì qua bước 3. Nếu không, P(i) phải chờ vì tài nguyên không còn đủ để cấp phát.

• Giải thuật phát hiện deadlock: Đầu tiên ta có các cấu trúc dữ liệu sau :

• Available : Một vector có độ dài m chỉ số lượng resources của từng loại.

• Allocation : Một ma trận n x m xác định số resources từng loại đang được cấp cho từng process (có tổng cộng n process).

• Request : Một ma trận n x m xác định số lượng resources mà từng process yêu cầu thêm ở từng loại resources.

Bước 1 : Gọi Work và Finish là hai vector có kích thước lần lượt là m và n. Khởi tạo : Work = Available. • Với mọi i = 1,2,…,n, nếu Allocation(i) khác 0 thì Finish[i] = false (tức là còn đang giữ resource thì chưa hoàn thành), ngược lại không giữ resource thì Finish[i] = true.

Bước 2 : Tìm i thoả mãn : • Finish[i] = false. • Request(i) <= Work. Nếu không tồn tại i như vậy, đến bước 4.

Bước 3 : Work = Work + Allocation. Finish[i] = true. Quay lại bước 2.

Bước 4 : nếu Finish[i] = false, với một số i = 1,2,…,n bất kỳ (tức là khi đến bước 4 vẫn có tiến trình nào đó chưa hoàn thành), thì hệ thống đang ở trạng thái deadlock. Hơn thế nữa,

Finish[i] = false thì process P(i) đang bị deadlocked. Thuật toán trên cần m x n^2 phép tính để xác định xem hệ thống có đang bị deadlock hay không.

8. Nêu ~~C~~các giải pháp để phục hồi hệ thống sau khi phát hiện có deadlock?

1. Chấm dứt tiến trình: Tìm và chấm dứt một hoặc nhiều tiến trình đang giữ tài nguyên và gây deadlock. Tài nguyên sau đó được giải phóng và có sẵn cho các tiến trình khác.

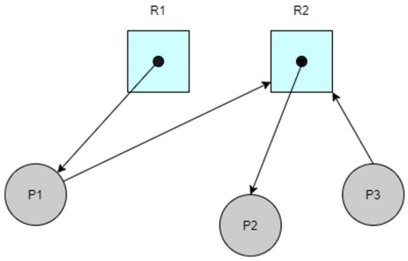
2. Khôi phục tự do tài nguyên: Giải phóng tất cả hoặc một số tài nguyên từ các tiến trình đang giữ và yêu cầu tài nguyên. Quá trình này có thể được thực hiện theo cách tuỳ thuộc vào chiến lược được chọn.

3. Rollback: Hệ thống hoặc mỗi tiến trình trong deadlock có thể rollback đến trạng thái trước khi yêu cầu tài nguyên cuối cùng.

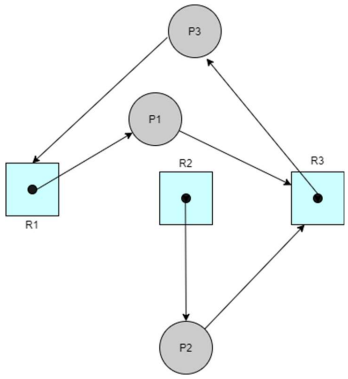
4. Recovery từ snapshot: Sử dụng các bản chụp (snapshot) trạng thái hệ thống được lưu trước khi xảy ra deadlock để phục hồi trạng thái hệ thống.

5. Hỗ trợ từ người quản trị: Một quản trị viên hệ thống có thể can thiệp để chấm dứt tiến trình, giải phóng tài nguyên, hoặc thực hiện các biện pháp phục hồi khác

9. (Bài tập mẫu) Cho các đồ thị cấp phát tài nguyên sau. Hỏi đồ thị nào có deadlock xảy ra?



(a)



(b)

Trả lời:

- Đồ thị (a) không có deadlock, do đồ thị này có chuỗi an toàn là: <P2, P1, P3> hoặc <P2, P3, P1>.
- Đồ thị (b) có deadlock.

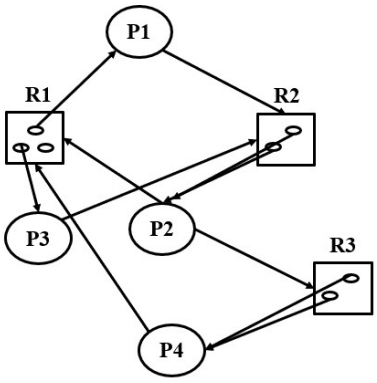
10. (Bài tập mẫu) Cho 1 hệ thống có 4 tiến trình P1, P2, P3, P4 và 3 loại tài nguyên R1 (3), R2 (2) R3 (2). P1 giữ 1 R1 và yêu cầu 1 R2; P2 giữ 2 R2 và yêu cầu 1 R1 và 1 R3; P3 giữ 1 R1 và yêu cầu 1 R2; P4 giữ 2 R3 và yêu cầu 1 R1.
    a. Vẽ đồ thị cấp phát tài nguyên của hệ thống
    b. Hệ thống có deadlock không?
    c. Tìm chuỗi an toàn (nếu có)

Trả lời:

a. Đồ thị cấp phát tài nguyên

Formatted: Underline

Formatted: Normal, Indent: Left: 0", Space After: 0 pt



~~b.~~

b. Hệ thống không bị deadlock do ~~nó~~ hệ thống có chuỗi an toàn.

c. Chuỗi an toàn là: <P4, P2, P3, P1> hoặc <P4, P2, P1, P3>.

11. Cho 1 hệ thống có 5 tiến trình P1, P2, P3, P4, P5 và 3 loại tài nguyên R1 (có 3 thực thể), R2 (có 3 thực thể) R3 (có 2 thực thể). P1 giữ 1 thực thể R1 và yêu cầu 1 thực thể R2; P2 giữ 2 thực thể R2 và yêu cầu 1 thực thể R1 và 1 thực thể R3; P3 giữ 1 thực thể R1 và yêu cầu 1 thực thể R2; P4 giữ 2 thực thể R3 và yêu cầu 1 thực thể R1; P5 đang giữ 1 thực thể của R2 và yêu cầu 1 thự thể của R1.

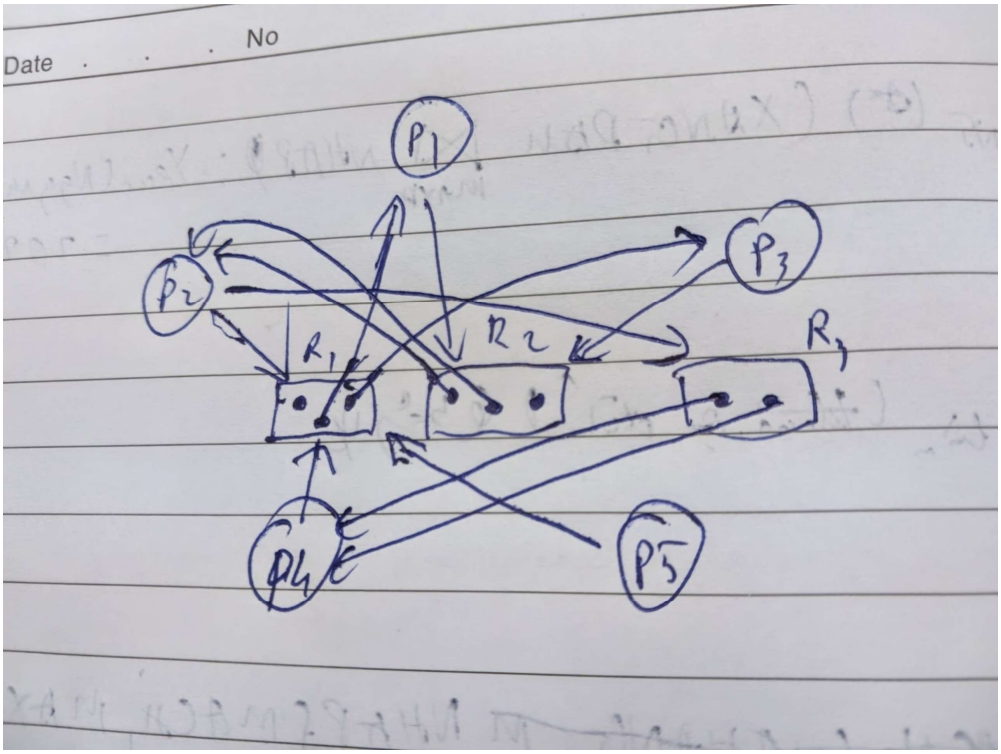
a. Vẽ đồ thị cấp phát tài nguyên

Formatted: Centered, Indent: Left: 0.45", No bullets or numbering

Formatted: Indent: Left: 0.2", Hanging: 0.25", Space After: 6 pt

Formatted: Indent: Left: 0.2", Hanging: 0.25", Space After: 6 pt, Numbered + Level: 1 + Numbering Style: a, b, c, … + Start at: 1 + Alignment: Left + Aligned at: 0.2" + Indent at: 0.45"

Formatted: fontstyle01, Font: Times New Roman, 13 pt



b. Có bao nhiêu chuỗi an toàn cho hệ thống trên?

Có 8 chuỗi an toàn

c. Viết ra tất cả các chuỗi an toàn (nếu có)

P4, P5, P1, P3, P2

P4, P5, P3, P1, P2

P5, P1, P3, P4, P2

P4, P2, P3, P1, P5

P4, P2, P5, P1, P3

P5, P3, P1, P4, P2

P4, P2, P1, P3, P5

P4, P2, P5, P3, P1

~~11.~~

12. (Bài tập mẫu) Xét một hệ thống máy tính có 5 tiến trình: P0, P1, P2, P3, P4 và 4 loại tài nguyên: A, B, C, D. Tại thời điểm t0, trạng thái của hệ thống như sau:

	Allocation				Max			
Tiến trình	**A**	**B**	**C**	**D**	**A**	**B**	**C**	**D**
P0	0	0	1	2	0	0	1	2
P1	1	0	0	0	1	7	5	0
P2	1	3	5	4	2	3	5	6
P3	0	6	3	2	0	6	5	2
P4	0	0	1	4	0	6	5	6

Available			
A	**B**	**C**	**D**
1	5	2	0

a. Tìm Need?

b. Hệ thống có an toàn không?

c. Nếu P1 yêu cầu (0,4,2,0) thì có thể cấp phát cho nó ngay không?

Trả lời:

a. Ma trận Need

	Need			
Tiến trình	**A**	**B**	**C**	**D**
P0	0	0	0	0
P1	0	7	5	0
P2	1	0	0	2
P3	0	0	2	0
P4	0	6	4	2

b. Thực hiện giải thuật an toàn

	Allocation				Max				Need				Available (Work)				
	A	**B**	**C**	**D**	**A**	**B**	**C**	**D**	**A**	**B**	**C**	**D**	**A**	**B**	**C**	**D**	
P0	0	0	1	2	0	0	1	2	0	0	0	0	1	5	2	0	P0
P1	1	0	0	0	1	7	5	0	0	7	5	0	1	5	3	2	P2
P2	1	3	5	4	2	3	5	6	1	0	0	2	2	8	8	6	P3
P3	0	6	3	2	0	6	5	2	0	0	2	0	2	14	11	8	P4
P4	0	0	1	4	0	6	5	6	0	6	4	2	2	14	12	12	P1

Hệ thống có chuỗi an toàn <P0, P2, P3, P4, P1> cho nên hệ thống an toàn.

c.

Formatted: Body Text First Indent, Indent: Left: 0.75", Line spacing: single, No bullets or numbering

Request P1 (0,4,2,0) ≤ Need P1 (0, 7, 5, 0).

Request P1 (0,4,2,0) ≤ Available (1, 5, 2, 0).

Giả sử hệ thống đáp ứng yêu cầu (0,4,2,0) của P1.

Trạng thái mới của hệ thống:

	Allocation				Max				Need				Available (Work)				
	A	**B**	**C**	**D**	**A**	**B**	**C**	**D**	**A**	**B**	**C**	**D**	**A**	**B**	**C**	**D**	
P0	0	0	1	2	0	0	1	2	0	0	0	0	1	1	0	0	P0
P1	1	4	2	0	1	7	5	0	0	3	3	0	1	1	1	2	P2
P2	1	3	5	4	2	3	5	6	1	0	0	2	2	4	6	6	P3
P3	0	6	3	2	0	6	5	2	0	0	2	0	2	10	9	8	P4
P4	0	0	1	4	0	6	5	6	0	6	4	2	2	10	7	12	P1

Hệ thống mới vẫn có chuỗi an toàn <P0, P2, P3, P4, P1> cho nên hệ thống đáp ứng yêu cầu cấp phát cho P1.

13. Xét hệ thống tại thời điểm t_0 có 5 tiến trình: P1, P2, P3, P4, P5; và 4 loại tài nguyên: R1, R2, R3, R4. Xét trạng thái hệ thống như sau:

	Allocation				Max			
Process	R1	R2	R3	R4	R1	R2	R3	R4
P1	0	0	1	2	0	0	3	2
P2	2	0	0	0	2	7	5	0
P3	0	0	3	4	6	6	5	6
P4	2	3	5	4	3	3	5	6
P5	0	3	3	2	0	6	5	2

Available			
R1	R2	R3	R4
2	1	2	0

a. Tại thời điểm t_0, áp dụng giải thuật banker tìm chuỗi an toàn của hệ thống?

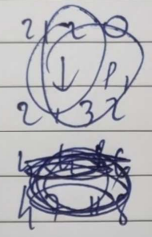
b. Tại thời điểm t1, tiến trình P3 yêu cầu thêm tài nguyên (1, 1, 0, 0) thì hệ thống có thể đáp ứng ngay được không? Tại sao?

Formatted: Indent: Left: 0.25", Hanging: 0.25", Line spacing: 1.5 lines

Date . . . No

	Need			
	R_1	R_2	R_3	R_4
P_1	0	0	2	0
P_2	0	7	5	0
P_3	6	6	2	2
P_4	1	0	0	2
P_5	0	3	2	0

a) 2 2 2



2 1 2 0

2 1 3 2

↓ P_1

4 4 8 6

~~6 7 11 8~~ P_5

4 7 11 8

↓ P_2

6 7 11 8

~~6 7 14 12~~ P_3

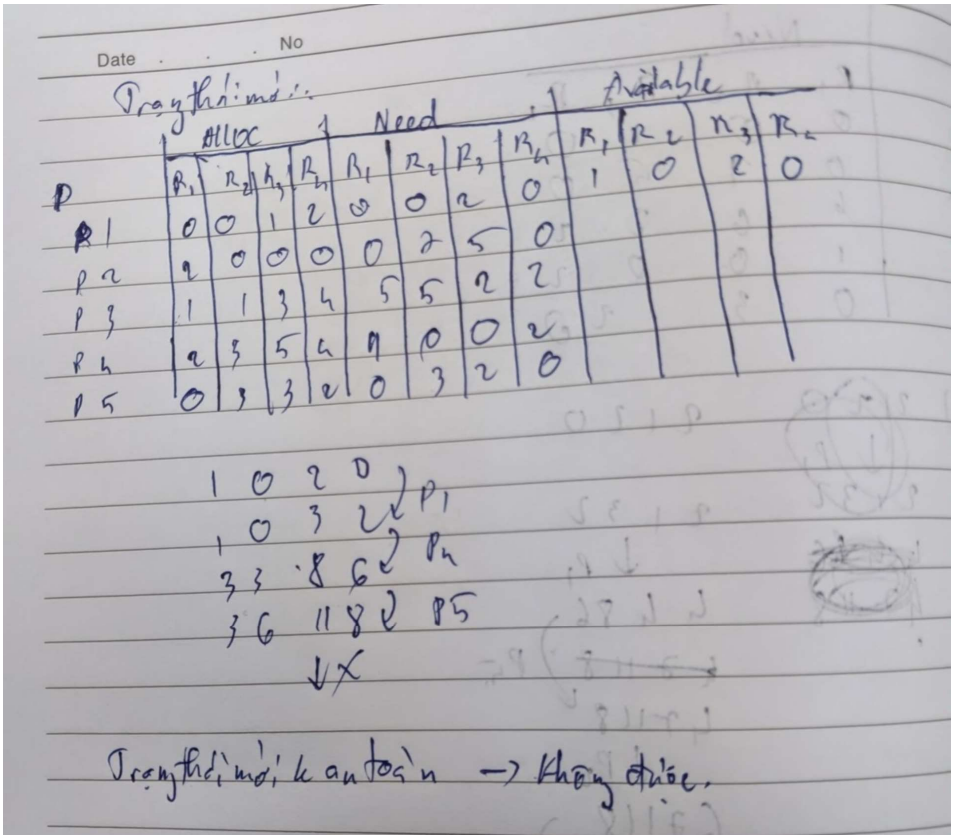
6 7 14 12

Chuỗi an toàn: P_1, P_4, P_5, P_2, P_3

b)

Request (1, 1, 0, 0) < Available (6, 6, 2, 2)

→ Đúng



14. Sử dụng giải thuật Banker để kiểm tra các trạng thái sau có an toàn hay không? Nếu có thì đưa ra chuỗi thực thi an toàn, nếu không thì nêu rõ lý do không an toàn?

a. Available = (0,3,0,1)

b. Available = (1,0,0,2)

	Allocation				Max			
Tiến trình	**A**	**B**	**C**	**D**	**A**	**B**	**C**	**D**
P0	3	0	1	4	5	1	1	7
P1	2	2	1	0	3	2	1	1
P2	3	1	2	1	3	3	2	1

P3	0	5	1	0	4	6	1	2
P4	4	2	1	2	6	3	2	5

Date . . . No

P	Alloc A	Alloc B	Alloc C	Alloc D	Max A	Max B	Max C	Max D	Need A	Need B	Need C	Need D
0	3	0	1	4	5	1	1	7	2	1	0	3
1	2	2	1	0	3	2	1	1	1	0	0	1
2	3	1	2	1	3	3	2	1	0	2	0	0
3	0	5	1	0	4	6	1	2	4	1	0	2
4	4	2	1	2	6	3	3	5	2	1	1	3

a) ~~Available~~ = (0, 3, 0, 1)

0 3 0 1
↓ P_2
3 4 2 2
↓ P_1
5 6 3 2
↓ P_3
5 11 4 2
↓ X
X

Ko có chuỗi an toàn → ~~không~~ trạng thái không an toàn

b) Available = (1, 0, 2)

1 0 2
3 2 1 2 → P_1
6 3 3 3 → P_2
6 8 4 3 → P_3
10 10 5 5 → P_4
13 10 6 9 → P_0

Chuỗi an toàn: (P_1, P_2, P_3, P_4, P_0) → Trạng thái an toàn

15. Trả lời các câu hỏi sau bằng cách sử dụng giải thuật Banker:

a. Hệ thống có an toàn không? Đưa ra chuỗi an toàn nếu có?

b. Nếu P1 yêu cầu (1,1,0,0) thì có thể cấp phát cho nó ngay không?

c. Nếu P4 yêu cầu (0,0,2,0) thì có thể cấp phát cho nó ngay không?

	Allocation				**Max**			
Tiến trình	**A**	**B**	**C**	**D**	**A**	**B**	**C**	**D**
P0	2	0	0	1	4	2	1	2
P1	3	1	2	1	5	2	5	2
P2	2	1	0	3	2	3	1	6
P3	1	3	1	2	1	4	2	4
P4	1	4	3	2	3	6	6	5

Available			
A	**B**	**C**	**D**
3	3	2	1

Date . . . No

	Alloc				Max				Need			
P	A	B	C	D	A	B	C	D	A	B	C	D
0	2	0	0	1	4	2	1	2	2	2	1	1
1	3	1	2	1	5	2	5	2	2	1	3	1
2	2	1	0	3	2	3	1	6	0	2	1	3
3	1	3	1	2	1	4	2	4	0	1	1	2
4	0	4	3	2	3	6	6	5	2	2	3	3

a)

Available = (3, 3, 2, 1)

3 3 2 1
↓ P_0
5 3 2 2
↓ P_3
6 6 3 4
↓
7 10 6 6
↓ P_1
10 11 8 7
↓ P_2
12 12 8 10

Chuỗi an toàn $< P_0, P_3, P_4, P_1, P_2 >$
→ An toàn

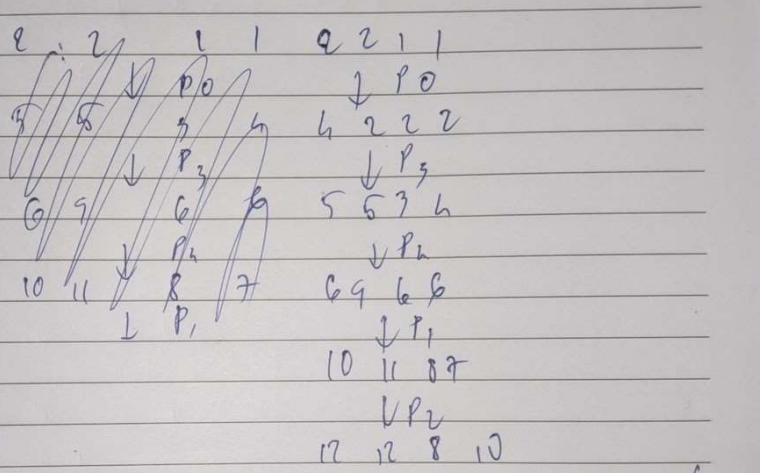
Date . . . No

b)

P_1 yêu cầu (1, 1, 0, 0) < available (3, 3, 2, 1)

Giả sử cấp phát P_1, thì trạng thái mới của P_1

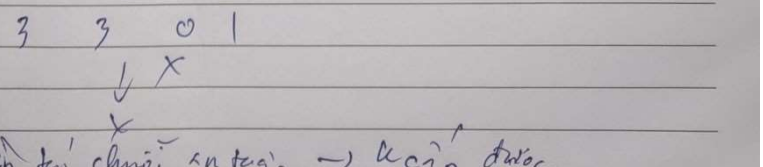
	Alloc				Need				Avail			
P_1	4	2	2	1	1	0	3	1	2	2	2	1



Chuỗi an toàn: (P_0, P_3, P_4, P_1, P_2) → an toàn → cấp được

c) P_4 yêu cầu (0, 0, 2, 0) < Avail

	Alloc				Need				Avail			
P_4	1	1	4	5	2	2	1	1	3	3	0	1



Không tồn tại chuỗi an toàn → Không cấp được