BẾ TẮC (DEADLOCK)

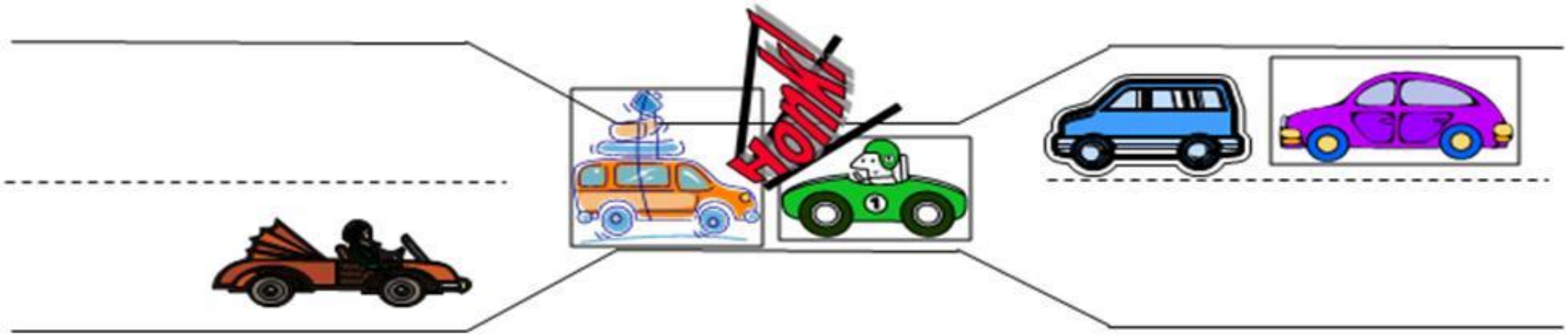
ThS. Nguyễn Thị Hải Bình

Khoa CNTT, ĐH Giao thông vận tải

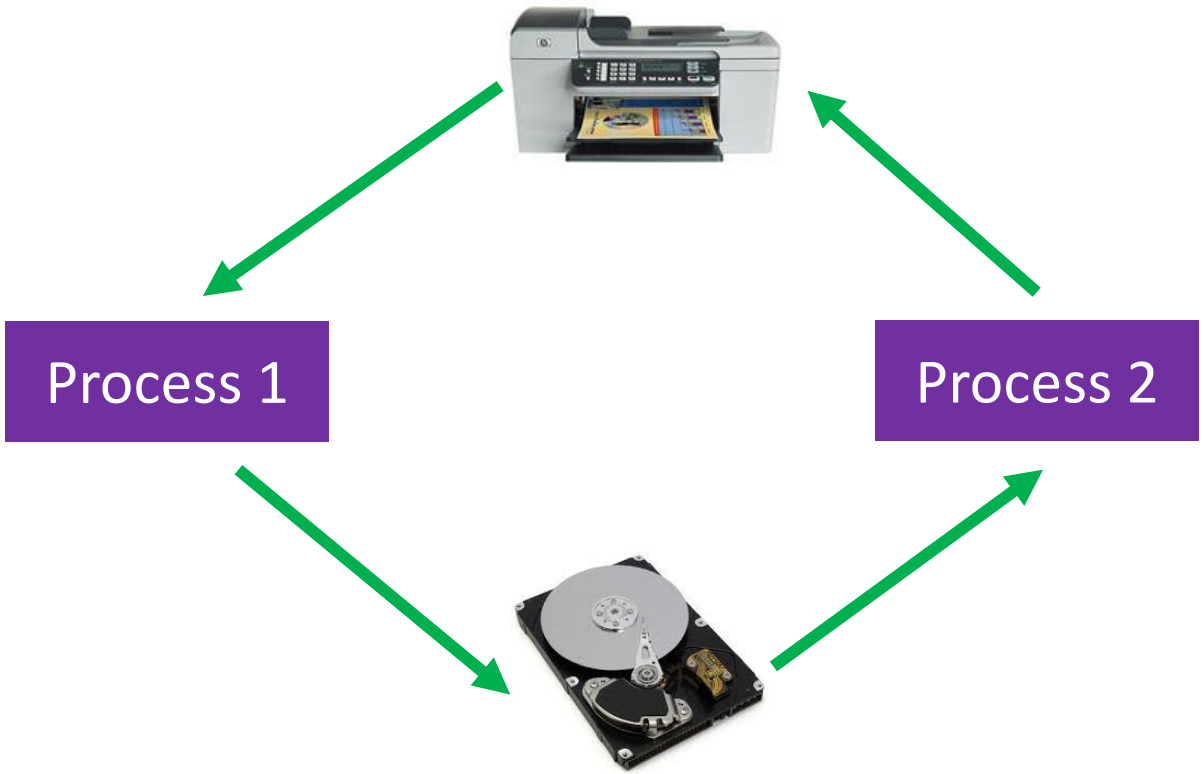
Email: calmseahn@gmail.com

Website: calmseahn.weebly.com

BRIDGE CROSSING EXAMPLE



DEADLOCK EXAMPLE



1. Process 1 requests the printer, gets it
2. Process 2 requests the tape unit, gets it
3. Process 1 requests the tape unit, waits
4. Process 2 requests the printer, waits

Bế tắc là tình huống xuất hiện khi hai tiến trình phải chờ đợi nhau giải phóng tài nguyên hoặc nhiều tiến trình chờ sử dụng các tài nguyên theo một “vòng tròn” (circular chain).

MÔ HÌNH HỆ THỐNG

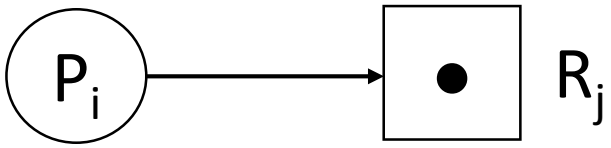
- Xem hệ thống như một tập hợp có giới hạn các tài nguyên
- Kiểu tài nguyên (type)
  - Các tài nguyên được chia thành các kiểu, ví dụ: memory, printers, CPUs, open files, tape drives, CD-ROMS, …
  - Hệ thống có 2 CPU thì kiểu tài nguyên CPU có 2 đối tượng
- Các đối tượng (instances) trong cùng một kiểu tài nguyên có vai trò như nhau
- Tiến trình sử dụng tài nguyên theo trình tự
  - Yêu cầu (Request)
  - Sử dụng (Use)
  - Giải phóng (Release)
- Một tập hợp các tiến trình ở tình trạng bế tắc khi mỗi tiến trình đều chờ tài nguyên từ một tiến trình khác trong tập hợp

ĐIỀU KIỆN CẦN ĐỂ CÓ BẾ TẮC

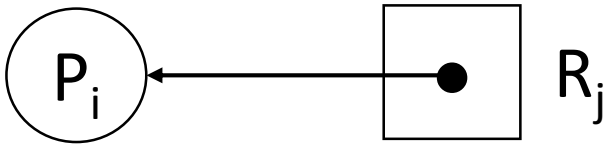
- Bế tắc xuất hiện nếu 4 điều kiện sau đồng thời xuất hiện
  - Độc quyền truy xuất (Mutal exclusion): ít nhất một tài nguyên bị nắm giữ thuộc kiểu không thể dùng chung
  - Giữ và chờ (Hold and wait): tồn tại tiến trình đang nắm giữ tài nguyên, đồng thời lại chờ tài nguyên bị giữ bởi tiến trình khác
  - Không chiếm đoạt (No preemption): hệ thống không thể chiếm tài nguyên của tiến trình
  - Vòng đợi (Circular wait): Tồn tại tập hợp các tiến trình $\{P_0, P_1, \ldots, P_n\}$, mà P_0 chờ P_1, P_1 chờ P_2, …, P_n chờ P_0

ĐỒ THỊ PHÂN PHỐI TÀI NGUYÊN

- Tập đỉnh V
  - $P = \{P_1, P_2, \ldots, P_n\}$ - ứng với các tiến trình
  - $R = \{R_1, R_2, \ldots, R_n\}$ - ứng với các kiểu tài nguyên của hệ thống
- Tập cung E
  - Cung yêu cầu

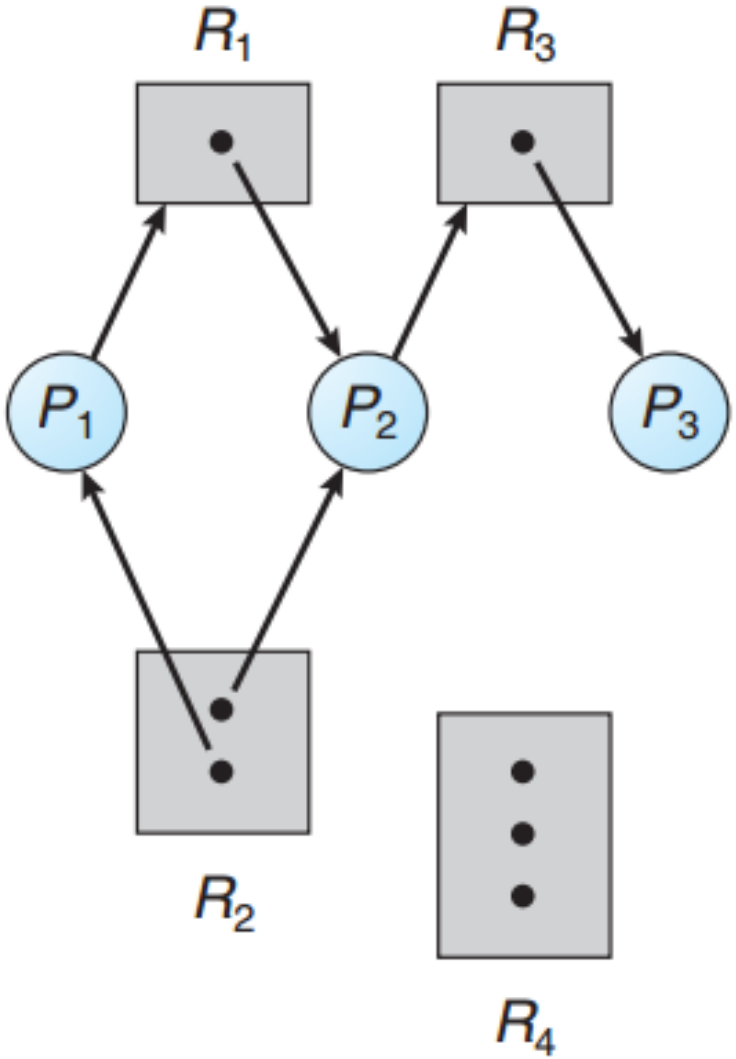


  - Cung phân phối



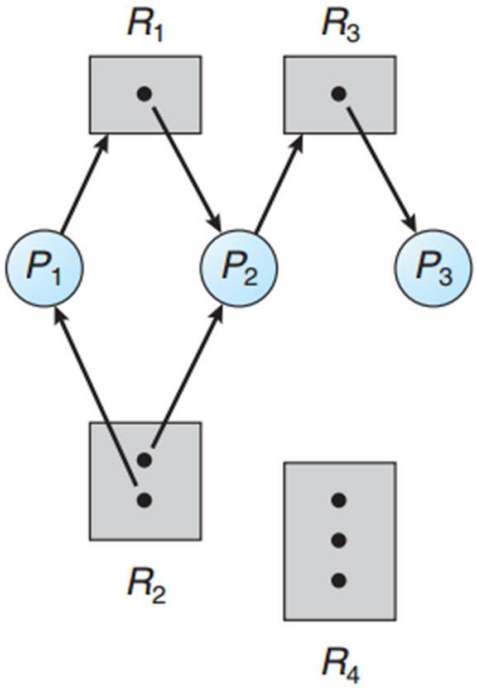
ĐỒ THỊ PHÂN PHỐI TÀI NGUYÊN

- Ví dụ
  - Tập đỉnh: $V = P \cup R$
    - $P = \{P_1, P_2, P_3\}$
    - $R = \{R_1, R_2, R_3, R_4\}$
  - Tập cung
    - $E = \{P_1 \rightarrow R_1, P_2 \rightarrow R_3, R_1 \rightarrow P_2, R_2 \rightarrow P_2, R_2 \rightarrow P_1, R_3 \rightarrow P_3\}$



PHÁT HIỆN BẾ TẮC BẰNG ĐTPPTN

- Nếu đồ thị không có chu trình, thì không có tiến trình nào bị bế tắc
- Nếu đồ thì có chu trình
  - Nếu mỗi kiểu tài nguyên trong chu trình có đúng một đối tượng, thì các tiến trình trong chu trình rơi vào trạng thái bế tắc
  - Nếu mỗi kiểu tài nguyên trong chu trình có nhiều hơn một đối tượng, thì bế tắc *có thể* xảy ra



Thêm vào cung $P_3 \rightarrow R_3$

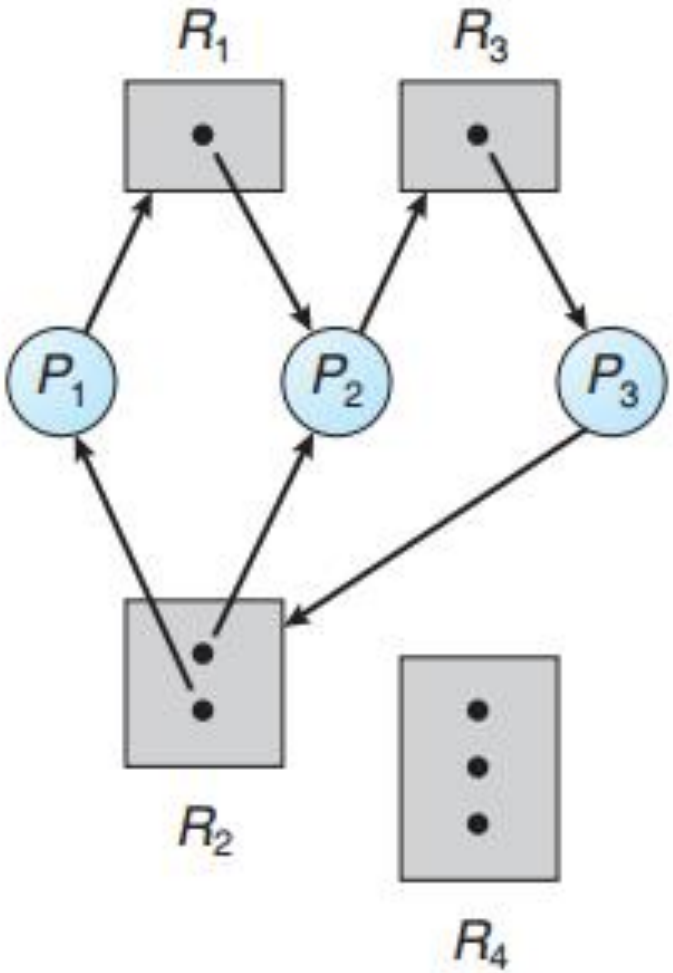


Figure 7.2 Resource-allocation graph with a deadlock.

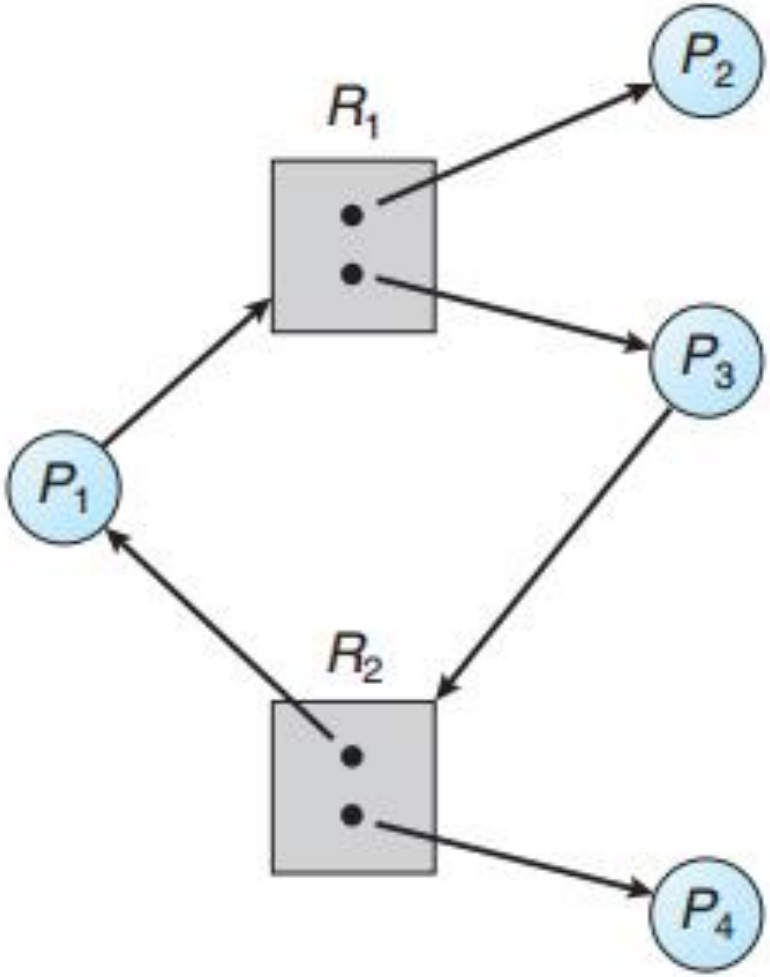


Figure 7.3 Resource-allocation graph with a cycle but no deadlock.

GIẢI QUYẾT BẾ TẮC

- Ba giải pháp
  - Đảm bảo hệ thống không rơi vào trạng thái bế tắc
    - Sử dụng một trong hai giao thức sau: ngăn chặn bế tắc (deadlock prevention) hoặc tránh bế tắc (deadlock avoidance)
  - Cho phép hệ thống rơi vào trạng thái bế tắc, sau đó phát hiện (deadlock detection) và khắc phục (recovery)
  - Bỏ qua mọi vấn đề, xem như bế tắc không bao giờ xuất hiện

NGĂN CHẶN BẾ TẮC

- Ý tưởng: đảm bảo ít nhất một trong bốn điều kiện cần của bế tắc không xảy ra
  - Độc quyền truy xuất (Mutal exclusion): ít nhất một tài nguyên bị nắm giữ thuộc kiểu không thể dùng chung
  - Giữ và chờ (Hold and wait): tồn tại tiến trình đang nắm giữ tài nguyên, đồng thời lại chờ tài nguyên bị giữ bởi tiến trình khác
  - Không chiếm đoạt (No preemption): hệ thống không thể chiếm tài nguyên của tiến trình
  - Vòng đợi (Circular wait): Tồn tại tập hợp các tiến trình $\{P_0, P_1, \ldots, P_n\}$, mà P_0 chờ P_1, P_1 chờ P_2, …, P_n chờ P_0

NGĂN CHẶN BẾ TẮC

- Điều kiện "độc quyền truy xuất"
  - Liên quan tới bản chất của tài nguyên
  - Điều kiện này luôn đúng với các tài nguyên không thể chia sẻ (ví dụ máy in)
  - Không thể loại bỏ điều kiện này
- Điều kiện "giữ và chờ"
  - Đảm bảo tiến trình không nắm giữ bất kỳ tài nguyên nào khi yêu cầu tài nguyên khác
  - Giải pháp
    - Tiến trình yêu cầu tất cả các tài nguyên trước khi thực thi
    - Trước khi yêu cầu thêm tài nguyên, tiến trình phải giải phóng tất cả tài nguyên đang giữ
    - "Nạn đói"

NGĂN CHẶN BẾ TẮC

- Điều kiện “không chiếm đoạt”
  - Giải pháp 1: Nếu tiến trình phải chờ một tài nguyên, thì hệ thống sẽ thu hồi tất cả tài nguyên mà tiến trình đó đang giữ
  - Giải pháp 2:
    - Nếu tiến trình P yêu cầu một tài nguyên, và tài nguyên đó đang bị giữ bởi tiến trình Q.
    - Nếu Q đang bị phong toả (i.e. đang chờ tài nguyên khác) thì tài nguyên của Q bị chiếm bởi P
    - Nếu ngược lại thì P phải chờ

NGĂN CHẶN BẾ TẮC

- Điều kiện "Vòng đợi"
  - Đánh số tất cả các tài nguyên
  - Các tiến trình yêu cầu cấp phát tài nguyên theo thứ tự tăng dần
- Ưu điểm
  - Dễ cài đặt
- Nhược điểm
  - Giảm hiệu suất sử dụng của tài nguyên
  - Giảm thông lượng của hệ thống

TRÁNH BẾ TẮC

- Ý tưởng
  - Nếu biết trước được tất cả nhu cầu sử dụng tài nguyên của tiến trình, hệ thống có thể quyết định việc cấp phát tài nguyên để tránh bế tắc
  - Hiểu biết về nhu cầu sử dụng tài nguyên có thể là
    - Số lượng cực đại các tài nguyên mà tiến trình cần
    - Hoặc thứ tự các tài nguyên mà tiến trình cần

TRÁNH BẾ TẮC

- Khái niệm "trạng thái cấp phát tài nguyên"
  - Xác định bởi số lượng tài nguyên rảnh rỗi, các tài nguyên đã cấp phát, và tất cả nhu cầu sử dụng tài nguyên của các tiến trình
- Trạng thái an toàn (Safe state)
  - Hệ thống có thể cấp phát tài nguyên cho tất cả các tiến trình mà không xảy ra bế tắc
  - Hệ thống ở trong tình trạng an toàn khi tồn tại một dãy an toàn
  - { P_0, P_1, P_2, ..., P_N } là dãy an toàn nếu yêu cầu tài nguyên của tiến trình P_i có thể được đáp ứng bởi các tài nguyên chưa cấp phát và các tài nguyên đang bị giữ bởi P_j với $j<i$

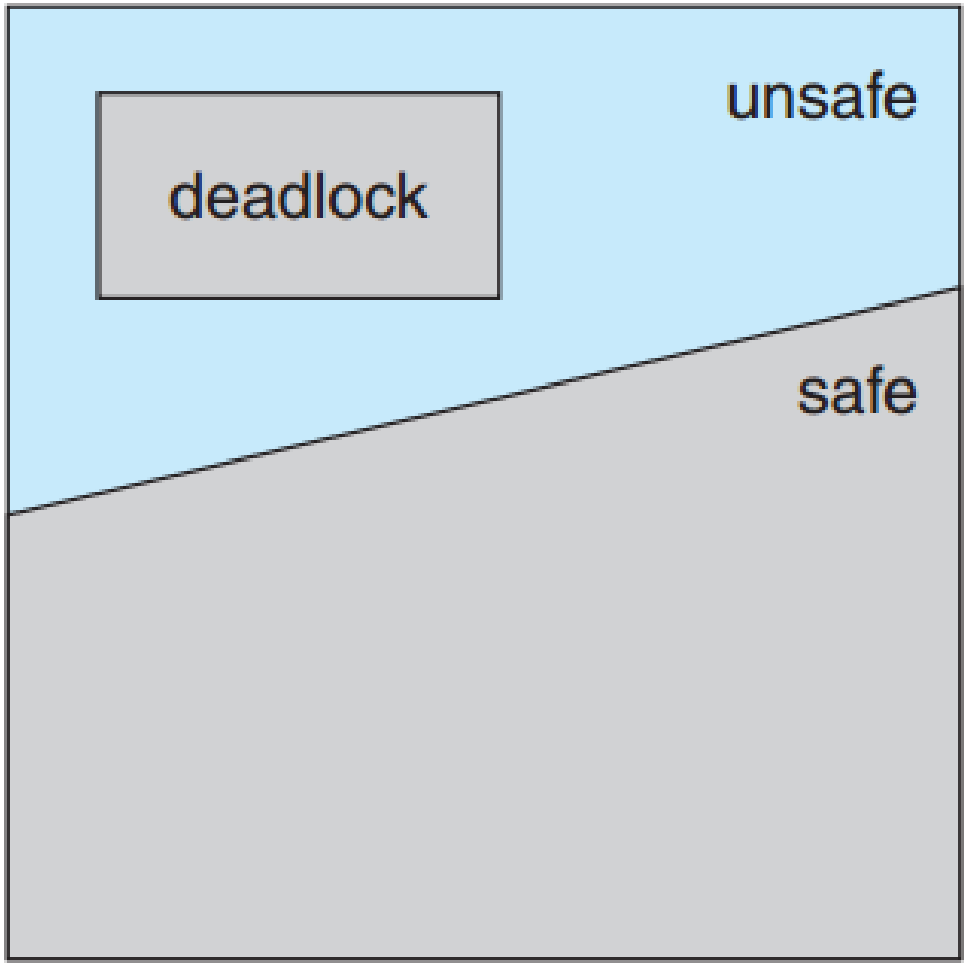


Figure 7.6 Safe, unsafe, and deadlocked state spaces.

VÍ DỤ VỀ TRẠNG THÁI AN TOÀN

- Xét hệ thống có 12 tài nguyên là 12 băng từ
  - Tại thời điểm t_0 hệ thống ở trạng thái như sau:

Process	Max needs	Allocated	Current needs
P0	10	5	5
P1	4	2	2
P2	9	2	7

  - Hệ thống ở trạng thái an toàn vì tồn tại $\{P_1, P_0, P_2\}$ là dãy an toàn

VÍ DỤ VỀ TRẠNG THÁI AN TOÀN

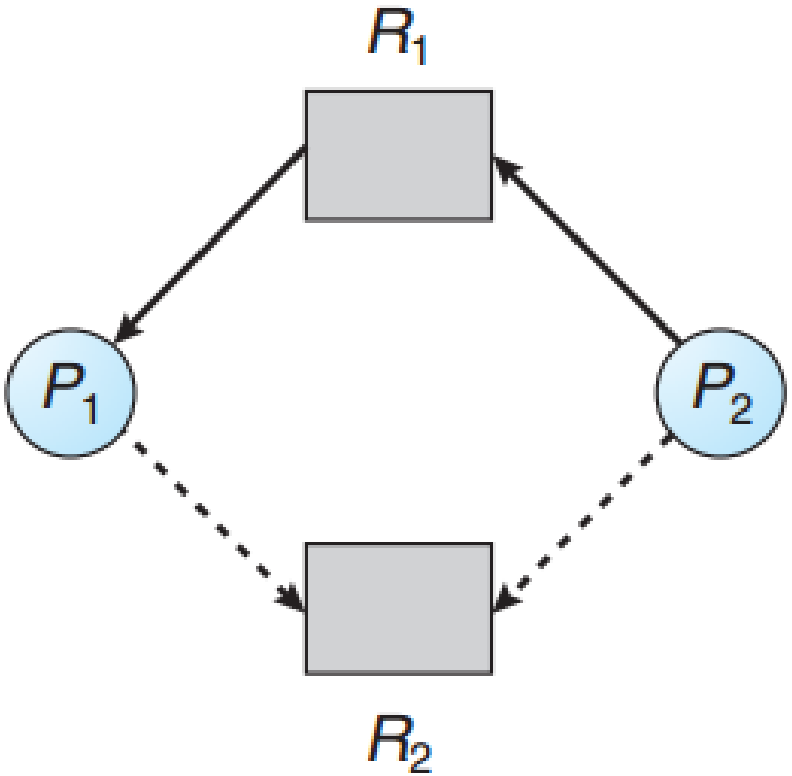
- Xét hệ thống có 12 tài nguyên là 12 băng từ
  - Tại thời điểm t_1, P2 xin cấp phát 1 tài nguyên và được chấp nhận. Hệ thống chuyển sang trạng thái như sau:

Process	Max needs	Allocated	Current needs
P0	10	5	5
P1	4	2	2
P2	9	3	6

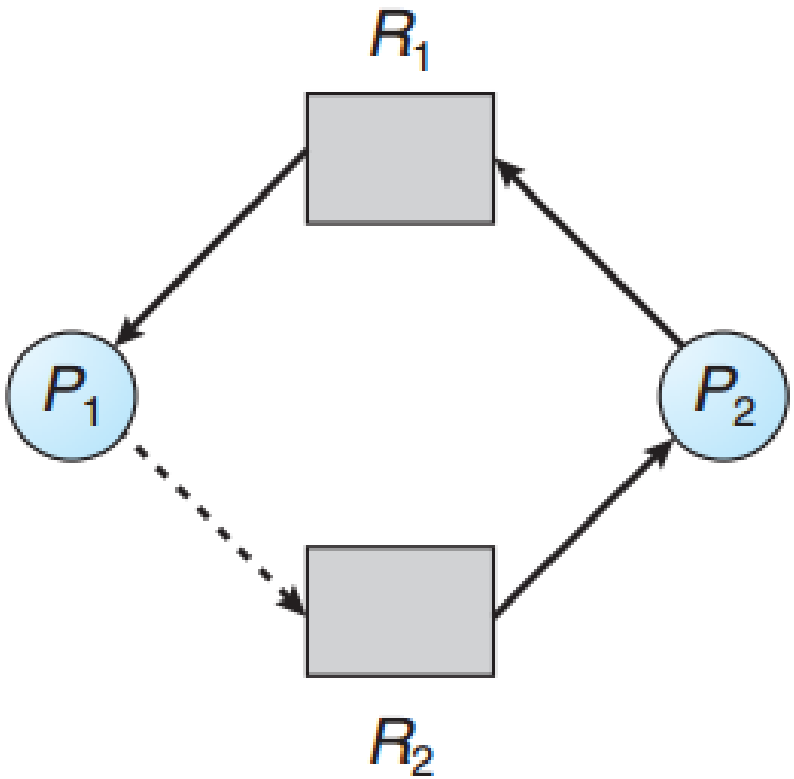
  - Hệ thống ở trạng thái không an toàn

THUẬT TOÁN ĐỒ THỊ CẤP PHÁT TÀI NGUYÊN

- Giả sử mỗi kiểu tài nguyên chỉ có một đối tượng
- Đồ thị gồm có:
  - Tập đỉnh: $P \cup R$
  - Tập cung
    - Cung yêu cầu
    - Cung phân phối
    - Cung nhu cầu (cung báo trước)



Resource-allocation graph for deadlock avoidance



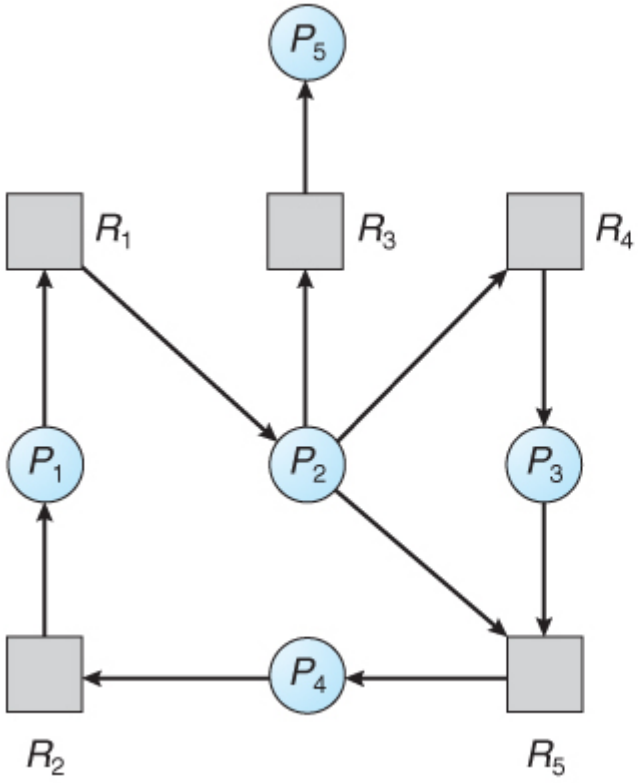
An unsafe state in a resource-allocation graph

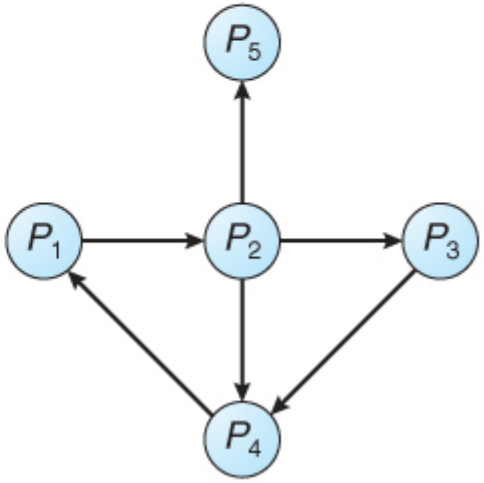
THUẬT TOÁN NGÂN HÀNG (BANKER'S ALGORITHM)

- Tự đọc

PHÁT HIỆN BẾ TẮC

- Kiểu tài nguyên có một đối tượng
  - Sử dụng đồ thị đợi chờ (wait-for graph)
  - Nếu tồn tại chu trình thì có bế tắc

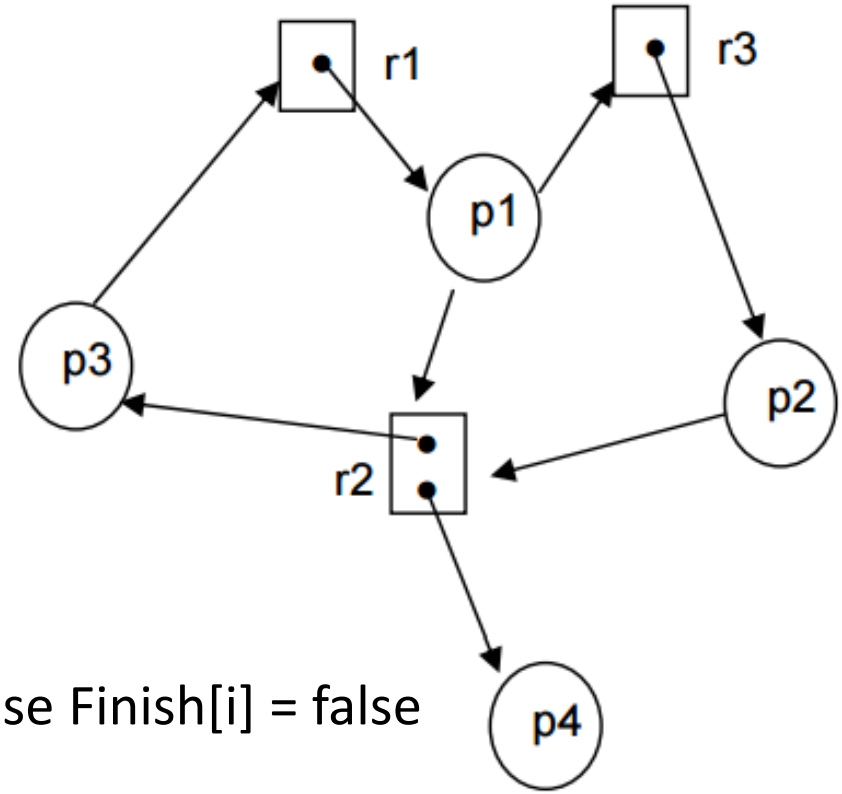




PHÁT HIỆN BẾ TẮC

- Kiểu tài nguyên có nhiều đối tượng
  - Giả sử hệ thống có m kiểu tài nguyên và n tiến trình
  - Thuật toán phát hiện bế tắc sử dụng các biến sau
    - Mảng available[m]
    - Mảng allocation[n,m]
    - Mảng request[n,m]
    - Mảng work[m]
    - Mảng finish[n]

Available [m]
Allocation [n,m]
Request [n,m]
Work[m]
Finish[n]
The algorithm is as follows:



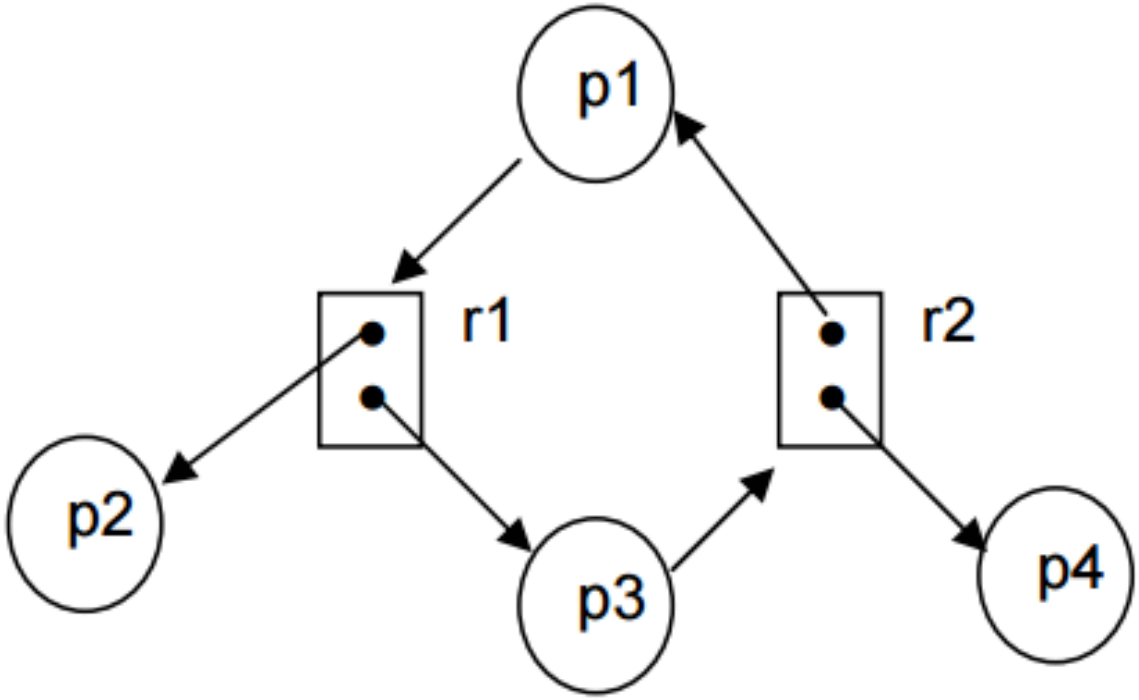
1. Initialize Work = Available
 For i = 1 to n do
 If Allocation(i) = 0 then Finish[i] = true else Finish[i] = false
2. Search an i such that
 Finish[i] = false and Request(i) ≤ Work
 If no such i can be found, go to step 4.
3. For that i found in step 2 do:
 Work = Work + Allocation(i)
 Finish[i] = true
 Go to step 2.
4. If Finish[i] ≠ true for a some i then the system is in deadlock state else the system is safe

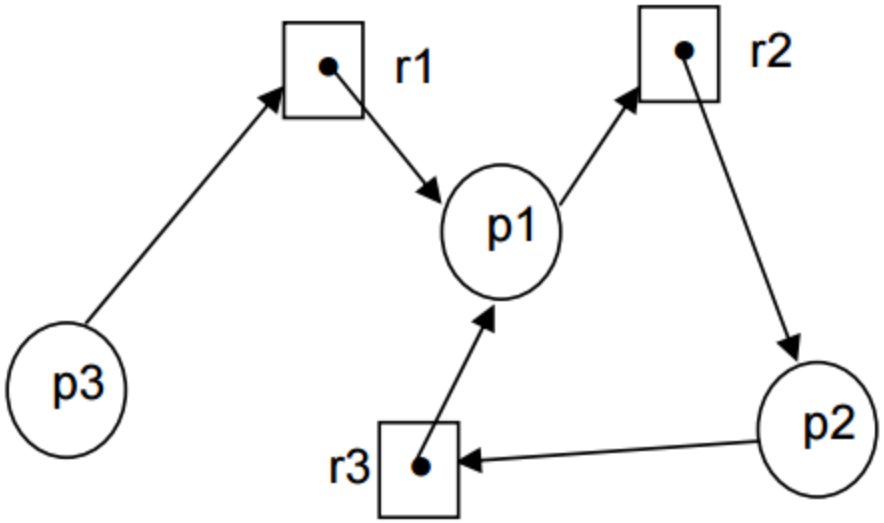
KHẮC PHỤC BẾ TẮC

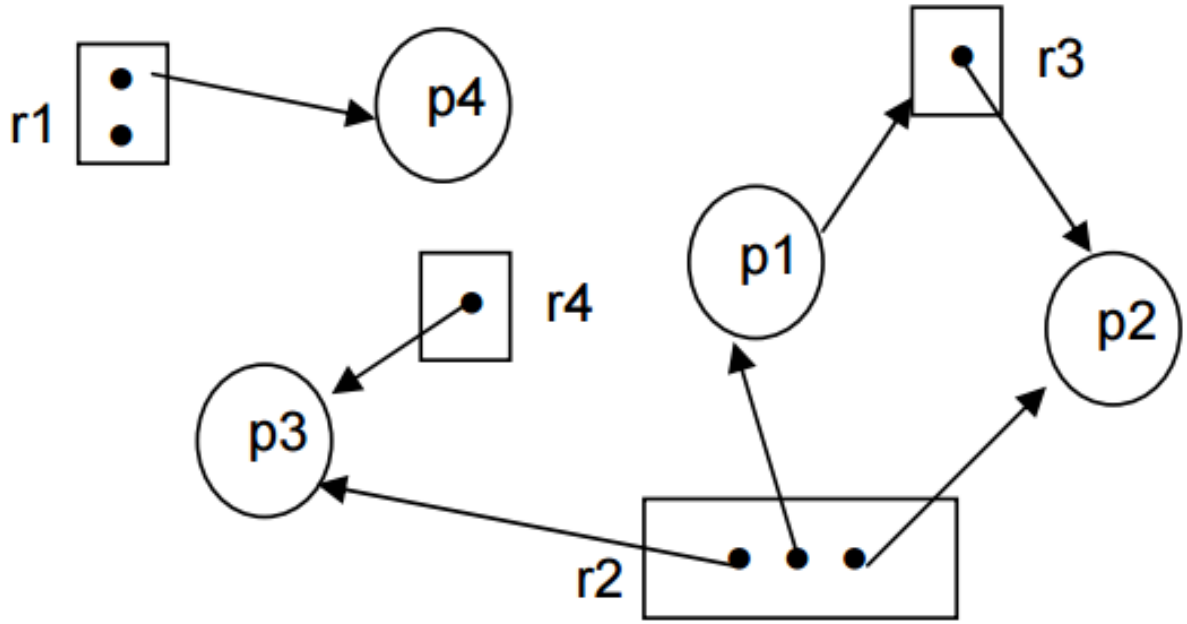
- Tự đọc

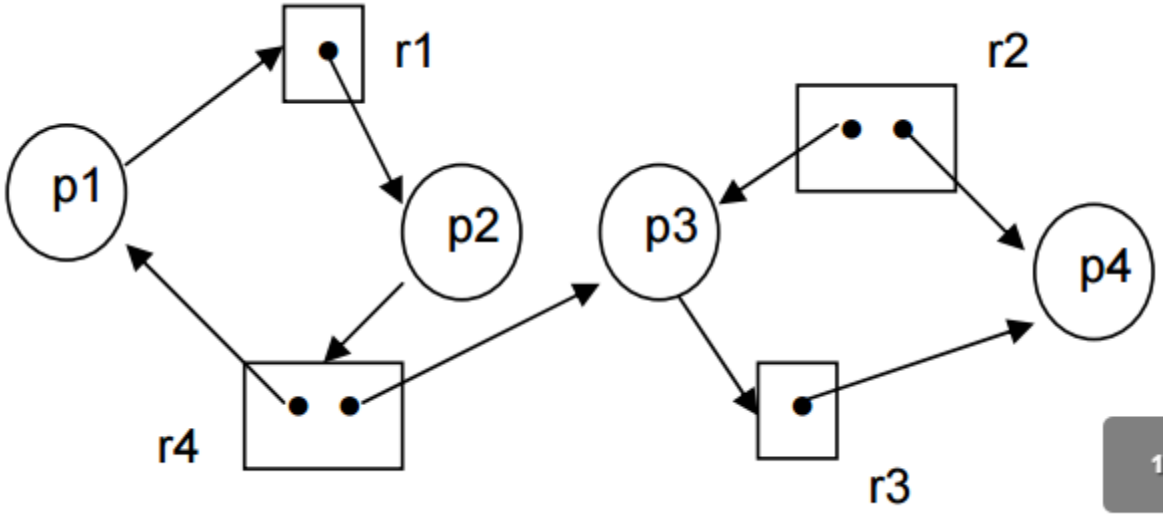
BÀI TẬP

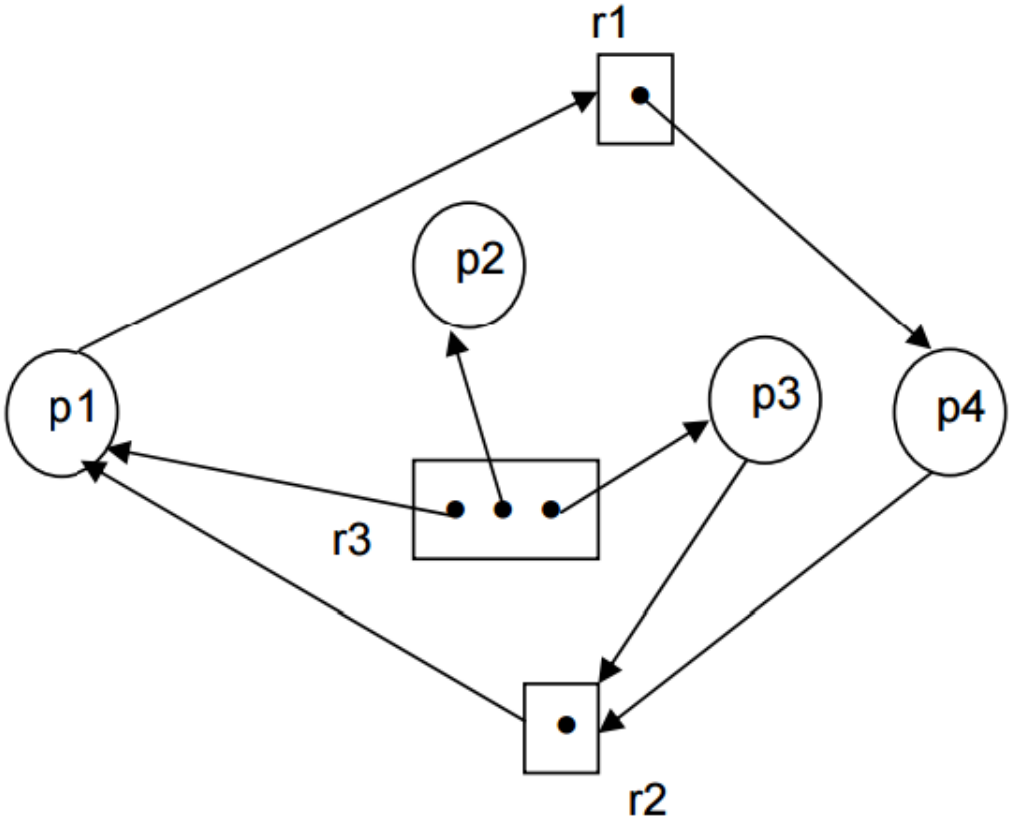
- Đồ thị phân phối tài nguyên sau có xuất hiện bế tắc hay không?

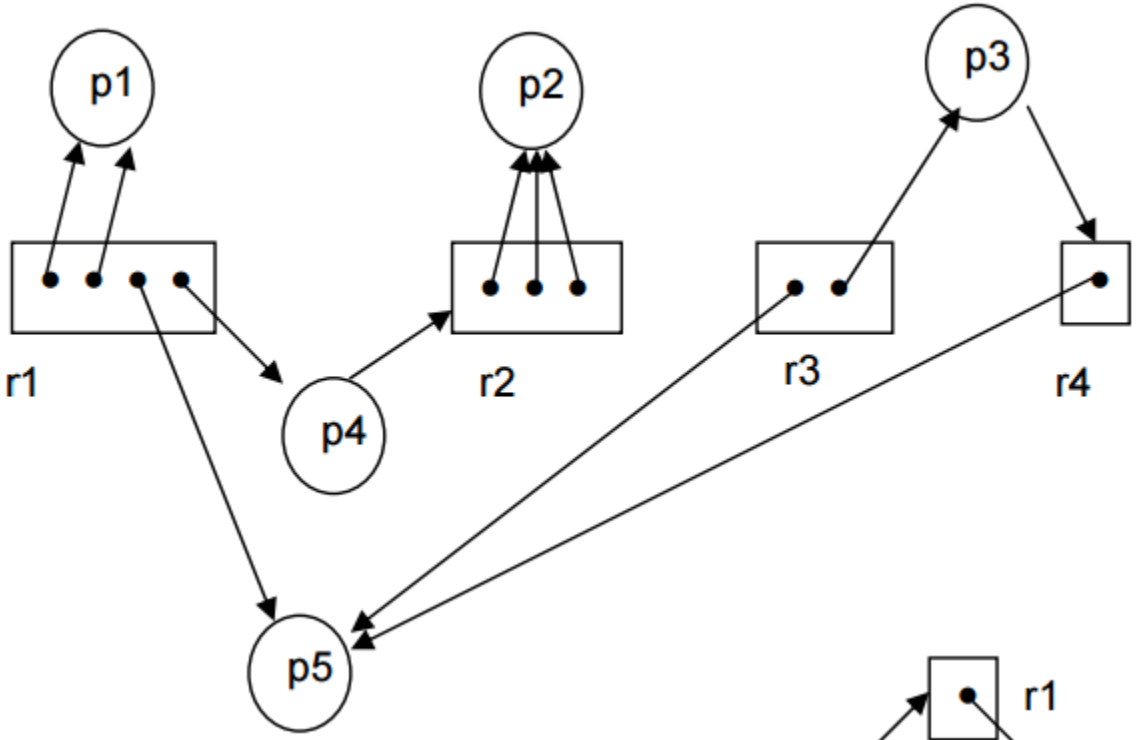


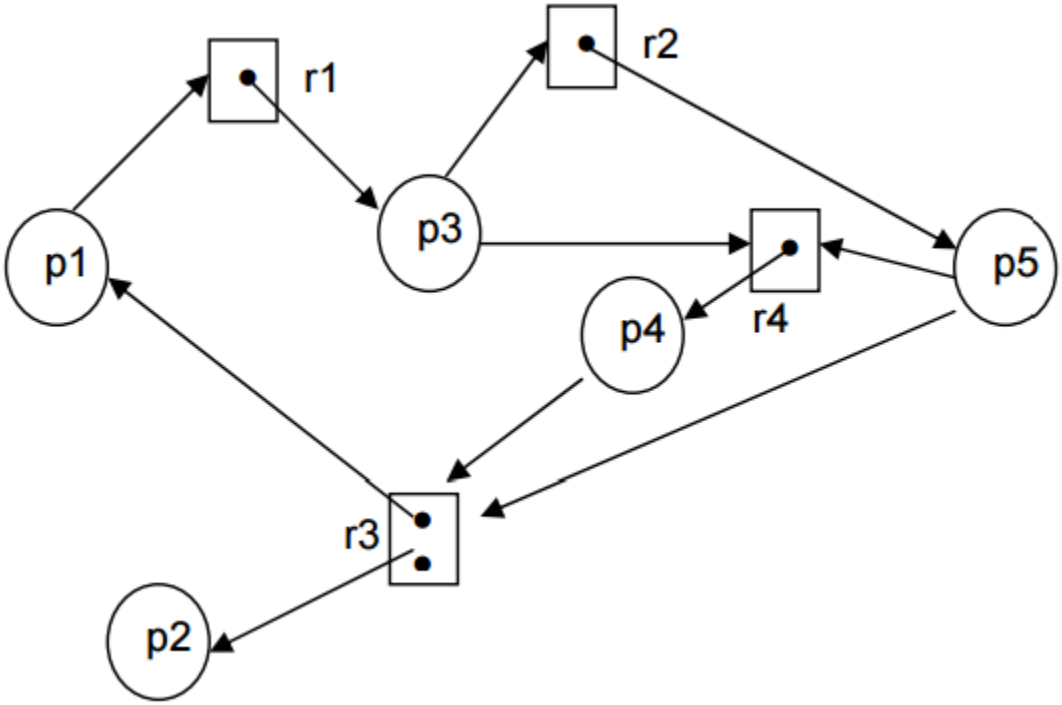

r1
r2
p1
p3
p2
r3

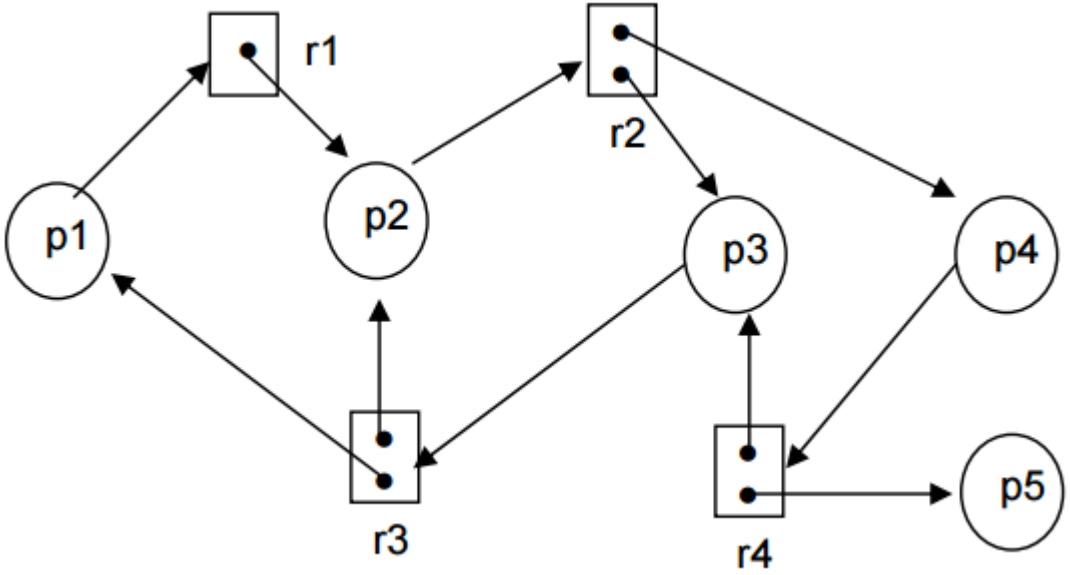

r1
p4
r3
p1
r4
p2
p3
r2


r1
r2
p1
p2
p3
p4
r4
r3


r1
p2
p1
p3
p4
r3
r2


p1
p2
p3
r1
r2
r3
r4
p4
p5


r1
r2
p1
p3
p5
p4
r4
r3
p2


r1
r2
p1
p2
p3
p4
p5
r3
r4

- Xét hệ thống có 9 tài nguyên
- Trạng thái hiện thời của hệ thống như sau

process	current	maximum
p1	3	6
p2	1	2
p3	4	9
p4	0	2

- Hệ thống có ở trạng thái an toàn hay không?