

Trí tuệ nhân tạo

TS. Nguyễn Văn Nam
(KHMT)

nvnam@tlu.edu.vn

Tài liệu

- 1) Stuart Russell and Peter Norvig, *Artificial Intelligence: A Modern Approach*, Prentice-Hall, 1994
- 2) Đinh Mạnh Tường, *Trí tuệ nhân tạo*, Nhà xuất bản khoa học kỹ thuật, 2002
- 3) Nguyễn Thanh Thủy, *Trí tuệ nhân tạo*, Nhà xuất bản giáo dục, 1997

Đánh giá môn học

- Thi cuối kỳ: 50% (thi viết)
- Điểm quá trình: 50%
 - Thi giữa kỳ
 - Thực hành
 - Ý thức học, chuyên cần

Nội dung

- Chương 1: Giới thiệu chung
- Chương 2: Giải quyết vấn đề bằng tìm kiếm
- Chương 3: Logic và suy diễn
- Chương 4: Biểu diễn tri thức
- Chương 5: Biểu diễn tri thức không chắc chắn
- Chương 6: Học máy

Chương 1

Giới thiệu chung

Giới thiệu về trí tuệ nhân tạo

- Khái niệm về trí tuệ nhân tạo
- Lịch sử phát triển của trí tuệ nhân tạo
- Các lĩnh vực ứng dụng

Trí tuệ nhân tạo là gì



- Trí tuệ là gì?

- Khả năng tính toán để đạt được mục tiêu
- Liên quan tới các nhiệm vụ đòi hỏi quá trình xử lý trí óc cao
 - Giải quyết bài toán; nhận dạng mẫu; phân loại; học; lập luận; suy diễn; xử lý ngôn ngữ, tri thức,...
 - http://www.myreaders.info/01_Introduction_to_Artificial_Intelligence.pdf

Trí tuệ nhân tạo là gì



- Trí tuệ nhân tạo là gì?
 - Artificial Intelligence (AI)
 - Là trí tuệ được biểu diễn bởi bất cứ một hệ thống nhân tạo nào
 - Là ngành khoa học kĩ thuật làm cho máy móc thông minh
 - Deep Blue đánh bại đại kiện tướng cờ vua Garry Kasparov 1997 (3 hòa, 2 thắng, 1 thua)
http://www.mycoders.info/01_Introduction_to_Artificial_Intelligence.pdf

Định nghĩa AI

- 4 nhóm định nghĩa, AI là ngành khoa

<ul style="list-style-type: none">Nghĩ giống con người	<ul style="list-style-type: none">Suy nghĩ hợp lý	 Lập luận
<ul style="list-style-type: none">Hành động giống con người	<ul style="list-style-type: none">Hành động hợp lý	 Hành động

 Hiệu suất

 Hợp lý

Định nghĩa AI

- (1) Nghĩ giống con người
 - Trí tuệ nhân tạo giúp tạo ra máy tính có khả năng **suy nghĩ**...máy tính có **trí tuệ** theo đầy đủ nghĩa của từ này (Haugeland, 1985)
 - Tự động hóa các hoạt động **phù hợp với suy nghĩ** con người như các hoạt động ra quyết định, giải bài toán, học, ... (Bellman 1978)
- (2) Suy nghĩ hợp lý
 - Sự nghiên cứu các **hoạt động trí não** thông qua việc sử dụng các mô hình tính toán (Charniak and McDermott, 1985)
 - Sự nghiên cứu các tính toán để máy **có thể nhận thức, lập luận** và hành động (Winston 1992)

Định nghĩa AI

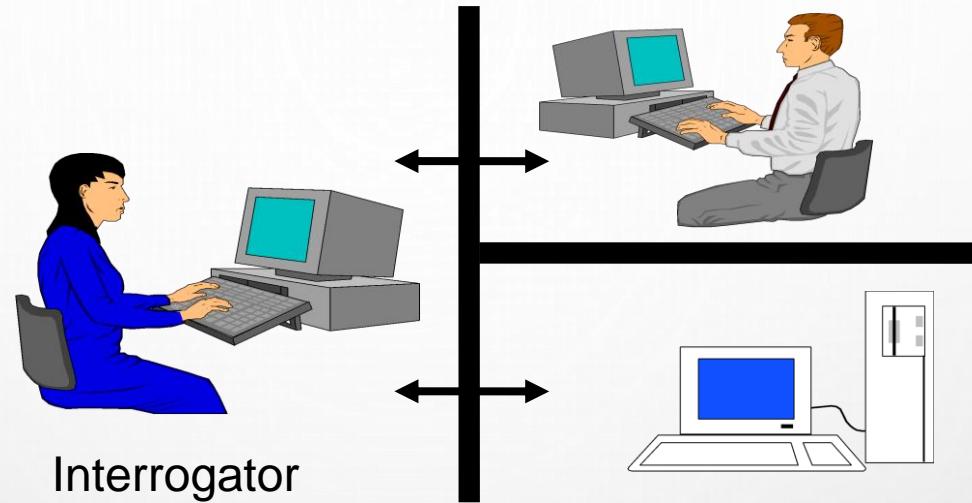
- (3) Hành động giống con người
 - Nghệ thuật tạo ra các máy **thực hiện** các chức năng đòi hỏi sự thông minh khi được thực hiện bởi con người (Kurzweil, 1990)
 - Sự nghiên cứu để làm cho máy tính **làm được** những việc mà con người còn làm tốt hơn máy tính (Rich và Knight, 1991)

Định nghĩa AI

- (4) Hành động hợp lý

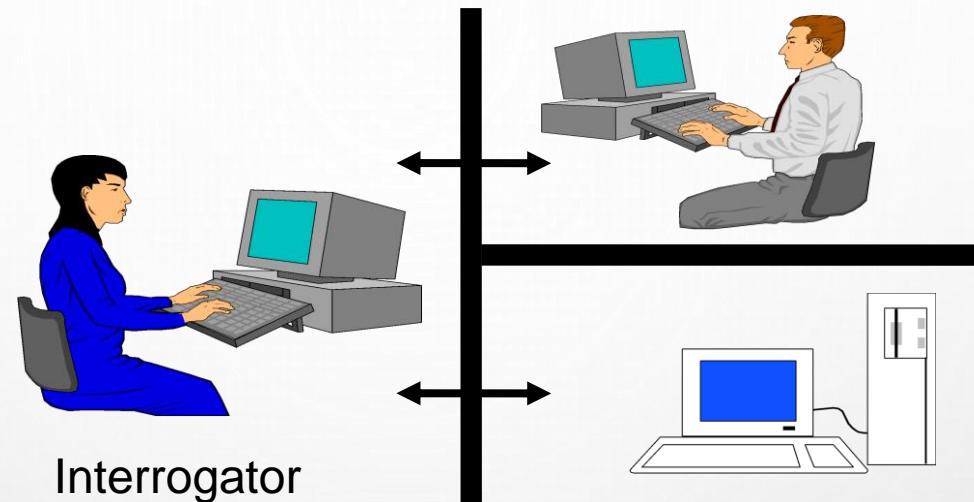
- Lĩnh vực nghiên cứu tìm cách giải quyết và mô phỏng các **hành vi thông minh** trong thuật ngữ các quá trình tính toán (Schalkoff, 1990)
- Một nhánh của khoa học máy tính liên quan tới sự tự động hóa các **hành vi thông minh** (Luger and Stubblefield, 1993)
- Nghiên cứu thiết kế các **tác nhân thông minh** (Pulle, Mackworth and Goebel, 1998)
- TTNT nghiên cứu các **hành vi thông minh** mô phỏng trong các vật thể nhân tạo (Nilsson 1998)

Hành động giống con người



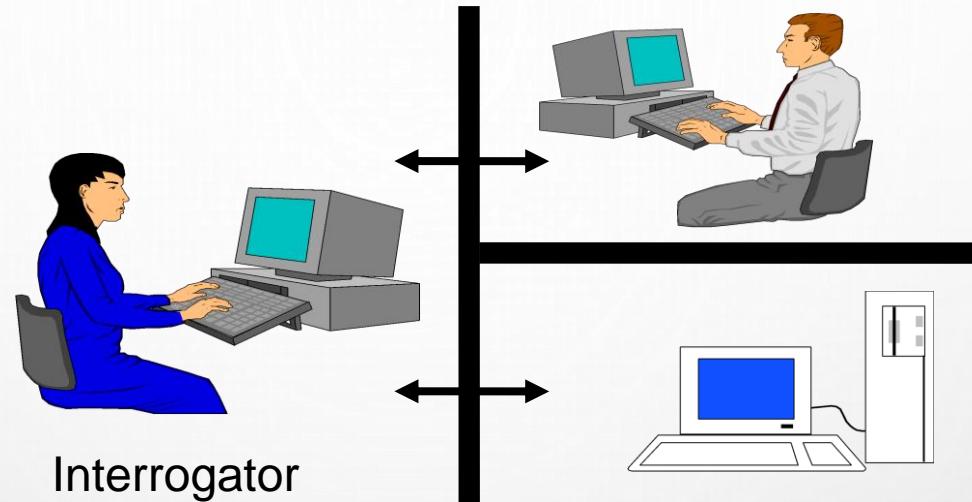
- Turing test
 - Liệu máy tính có thể suy nghĩ? → Máy tính có thể hành động thông minh (như con người)
 - Trò chơi bắt chước người để kiểm chứng hành động thông minh của máy tính

Hành động giống con người



- Turing test
 - Người thẩm vấn (interrogator) cần nhận biết mình đang chơi với người hay máy tính bằng cách đưa ra các câu hỏi và đánh giá các câu trả lời nhận được
 - Người chơi phải trả lời trung thực còn máy tính có thể nói dối để đánh lừa
 - Nếu không phát hiện được thì coi như máy tính thông minh như con người

Hành động giống con người

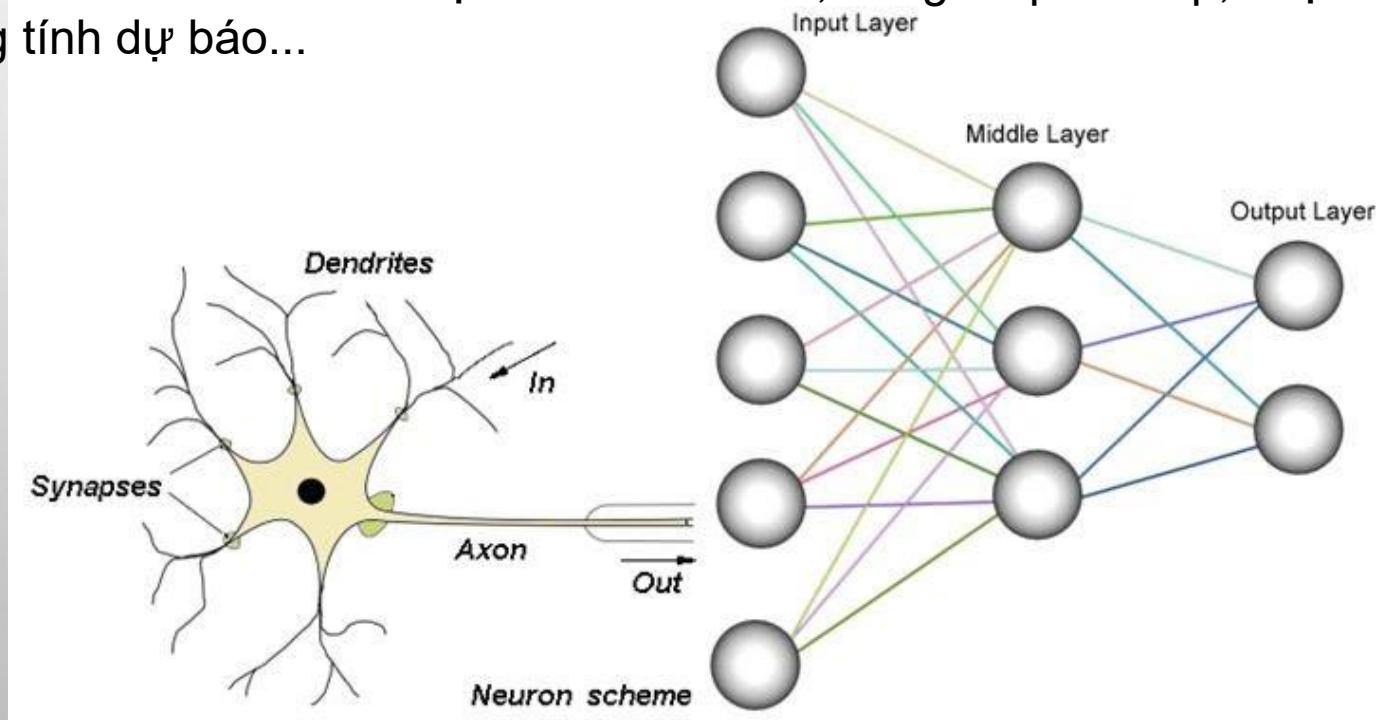


- Turing test

- **Ưu điểm:**
 - Loại trừ định kiến thiên vị của người thẩm vấn
- **Nhược điểm**
 - Chủ quan, phụ thuộc vào chất lượng câu hỏi của người hỏi

Suy nghĩ giống người

- Tìm hiểu cách suy nghĩ của con người
 - Xử lý ngôn ngữ, nghĩ, học, lập luận,... được thực hiện như thế nào
- Tạo các mô hình tính toán có cách thức suy nghĩ của con người, bao gồm 1 chuỗi các bước lập luận tương tự khi con người giải quyết cùng vấn đề
- Ví dụ: mạng nơron mô phỏng bộ não người bằng cách tạo ra các nơron nhân tạo, xây dựng cơ chế tính toán và học cho các nơron, dùng để phân lớp, nhận dạng, tính toán mang tính dự báo...



Suy nghĩ hợp lý

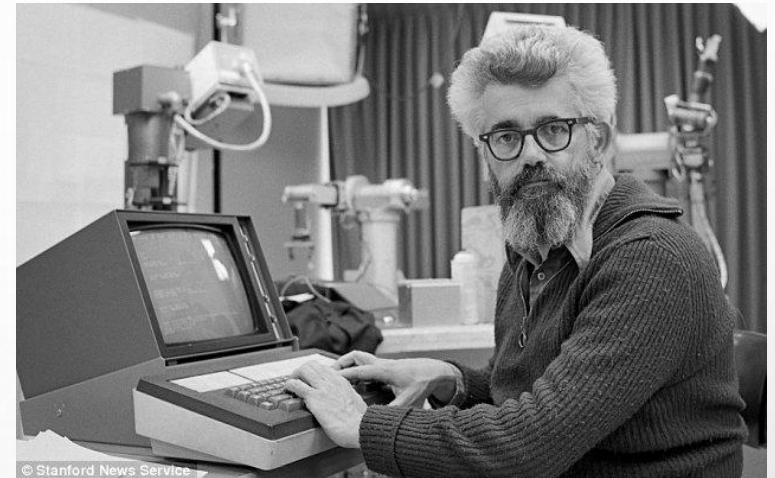
- Tập chung vào các cơ chế suy diễn mà có thể chính xác hoặc cho kết quả tối ưu
- Phát triển các hệ thống biểu diễn cho phép suy diễn kiểu như: “*Tất cả mọi người bị viêm răng đều đau răng. An bị viêm răng. Do đó, An bị đau răng*”
- Hình thức hóa quá trình lập luận dưới dạng hệ thống gồm các luật và thủ tục suy diễn
- Ví dụ: hệ chuyên gia (chuẩn đoán bệnh, xem tử vi, ...), hệ trợ giúp quyết định, ...
- Vấn đề: không phải vấn đề nào cũng có thể giải quyết bằng lập luận và suy diễn

Hành động hợp lý

- Thực hiện tốt công việc cần làm, cố gắng đạt được mục tiêu cao nhất (không nhất thiết tối ưu) từ những thông tin được cung cấp
- Thực thi các hành vi thông minh (là các hành vi làm tăng cơ hội đạt được các mục đích đề ra)
- Có thể tạm chấp nhận lập luận không hoàn hảo nếu thực hiện được công việc đề ra
- Ví dụ: multi-agent systems, robotics, ...

Lịch sử phát triển của AI

- Những năm 1950: khai sinh ra AI
 - 1950: Turing đề cập tới khái niệm về trí tuệ nhân tạo và đưa ra thí nghiệm Turing
 - 1956: thuật ngữ “trí tuệ nhân tạo” được chính thức đưa ra bởi John McCarthy và được công nhận tại hội nghị Dartmouth
 - Đầu 1950s, các chương trình AI đầu tiên
 - Chương trình chơi cờ của Samuel
 - Chương trình lý luận logic của Newell & Simon
 - Chương trình chứng minh các định lý hình học của Gelernter 1959
 - 1958: John McCarthy cho ra đời ngôn ngữ lập trình trí tuệ nhân tạo LISP



Lịch sử phát triển của AI

- Những năm 1960: khởi đầu của các đề án nghiên cứu AI
 - Bộ giải quyết bài toán tổng quát GPS (General Problem Solver- Newell và Simon)
 - 1965: chương trình chuyên gia phân tích tâm lý ELIZA
 - 1968: chương trình phân tích hình học (Tom Evans)

Lịch sử phát triển của AI

- Những năm 1970: sự phát triển của các hệ tri thức
 - Heuristic Programming Project (HPP): Feigenbaum
 - Mycin - hệ chuyên gia chuẩn đoán bệnh nhiễm trùng máu (Feigenbaum, Buchanan, and Dr. Edward Shortliffe)
 - 1972: ngôn ngữ lập trình logic Prolog (Alain Colmerauer và Robert Kowalski)
 - 1973: LUNAR-chương trình xử lý ngôn ngữ tự nhiên liên quan tới phân tích mẫu đá lấy từ mặt trăng
 - 1975: lược ngữ nghĩa Frame của Minsky

Lịch sử phát triển của AI

- Những năm 1980: AI đi vào các ngành kinh tế (AI thương mại) và công nghiệp (máy giặt, máy ảnh)
- Từ 1986 tới nay: sự phát triển trở lại của các ứng dụng mạng neuron, hệ chuyên gia
- 1995: xuất hiện hệ đa tác tử

Lĩnh vực ứng dụng của AI

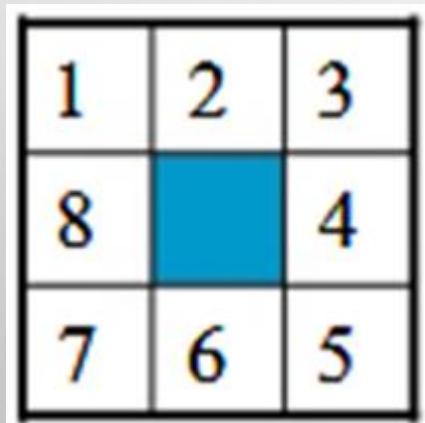
Game playing



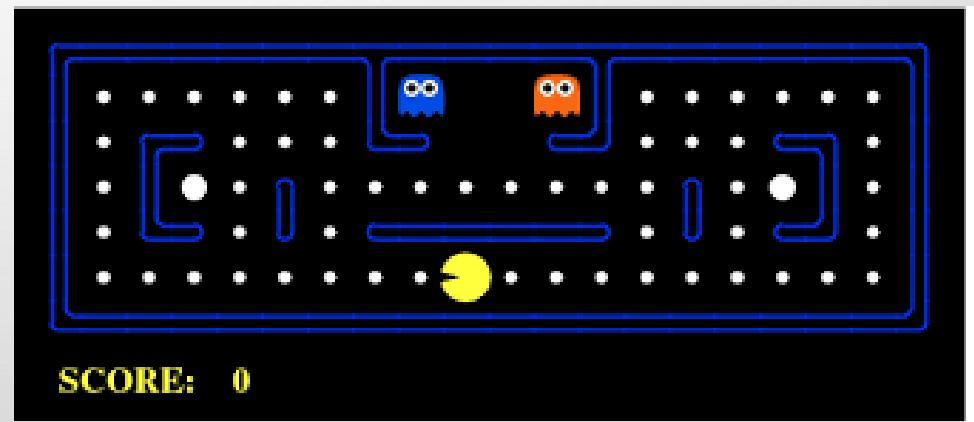
Chess playing



Hanoi tower



8-puzzle



Pacma

Muti-agent based games



Driver



Football



Haft-life



Sim city

Simulation

Crowd Simulation
in Airports



Flight simulator

Robotics



Dog robot

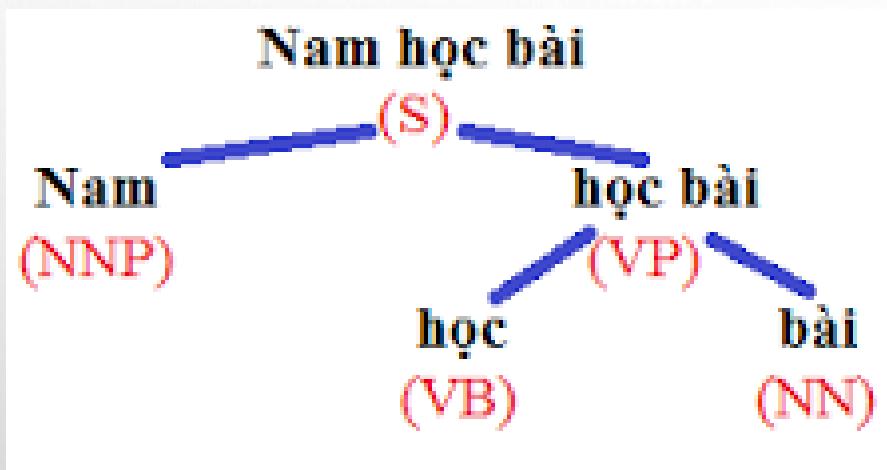


Kiva robots in Amazon



Self-driving cars

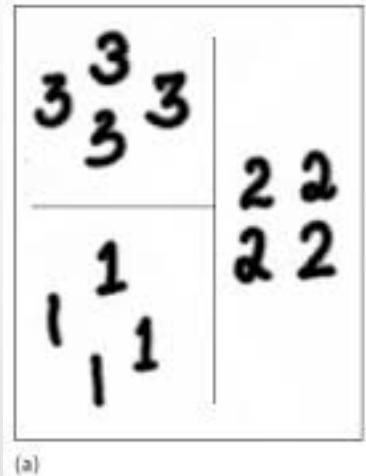
Language/Speech processing



Language processing

Speech
recognition

Pattern recognition



Find the O

Q Q Q
Q Q Q Q
Q Q Q Q Q
Q Q Q Q Q
Q Q Q Q Q
Q Q Q Q Q

Find the Q

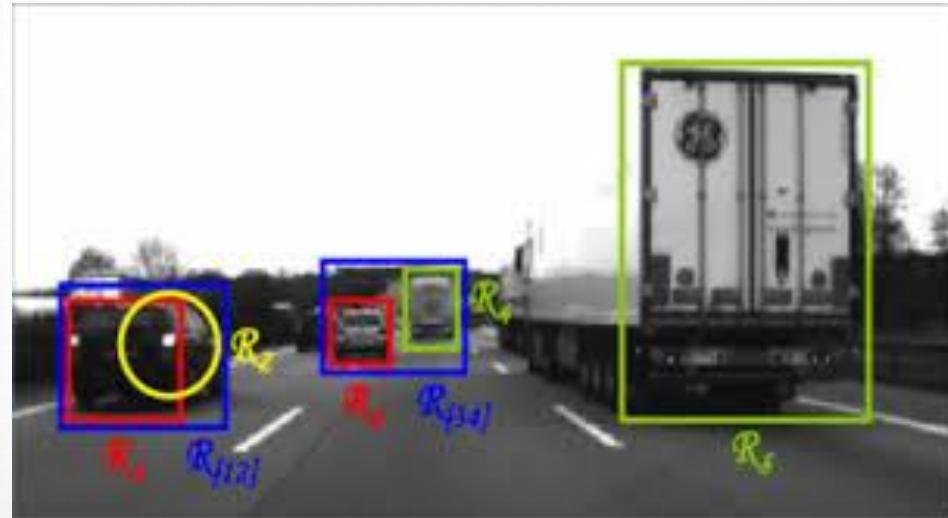
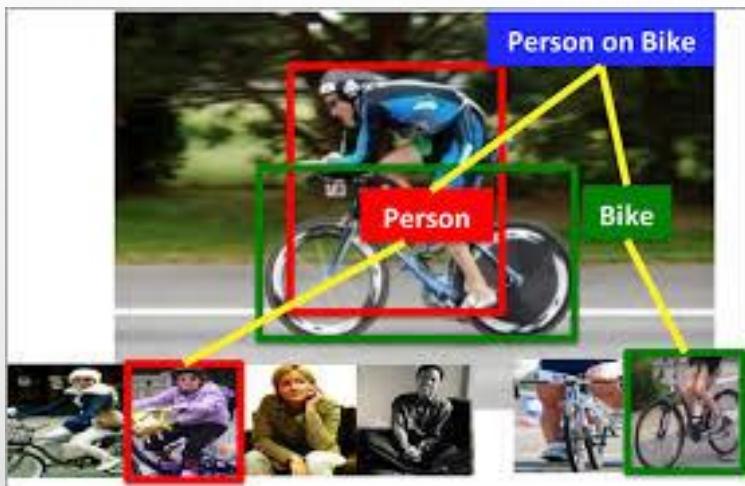
O O O
O O Q O
O O O O
O O O O
O O O

Text recognition



Facebook facial recognition

Computer vision



Expert systems

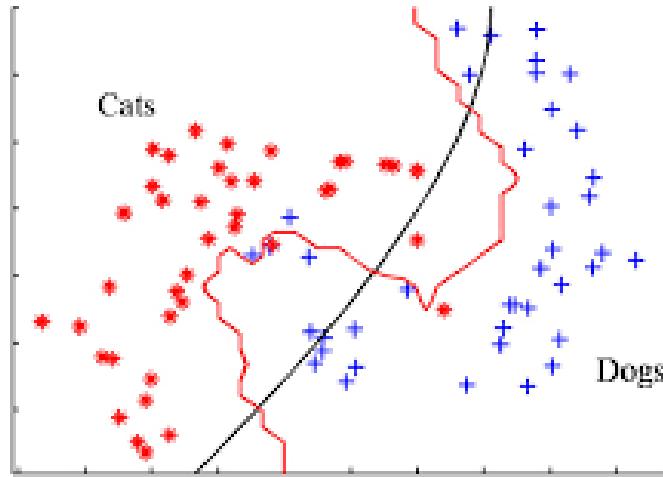


Mycin expert system

Decision support system



Machine learning



Phân lớp



Dự báo



Phân loại thư rác



Phát hiện thẻ tín dụng giả

Machine learning*



Chương 2

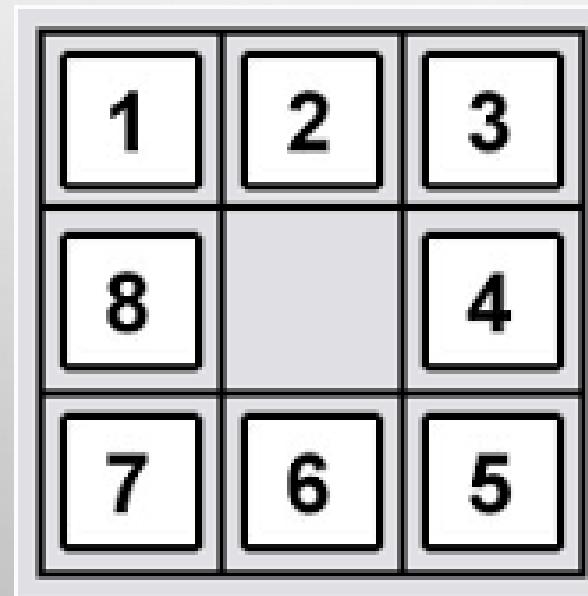
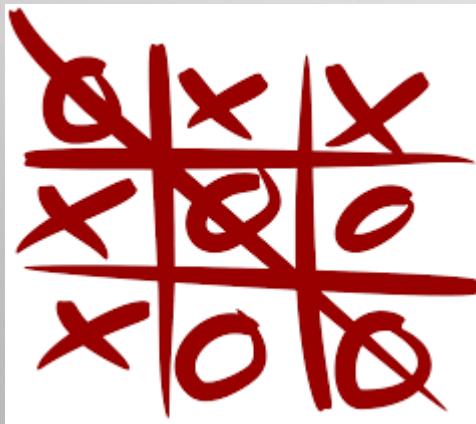
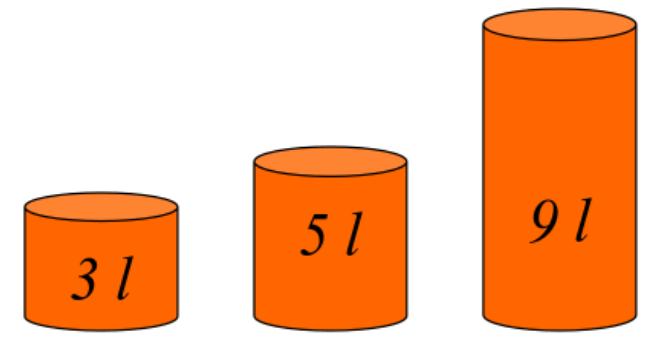
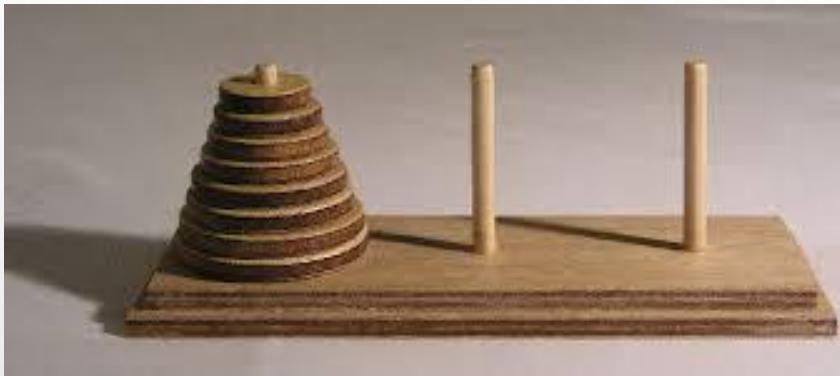
Giải quyết vấn đề bằng tìm kiếm

Chương 2: Giải quyết vấn đề bằng tìm kiếm

- Các trò chơi trên máy tính
- Biểu diễn bài toán bằng không gian trạng thái
- Giải quyết bài toán bằng tìm kiếm
- Tìm kiếm mù
- Tìm kiếm kinh nghiệm
- Tìm kiếm tối ưu
- Tìm kiếm có đối thủ

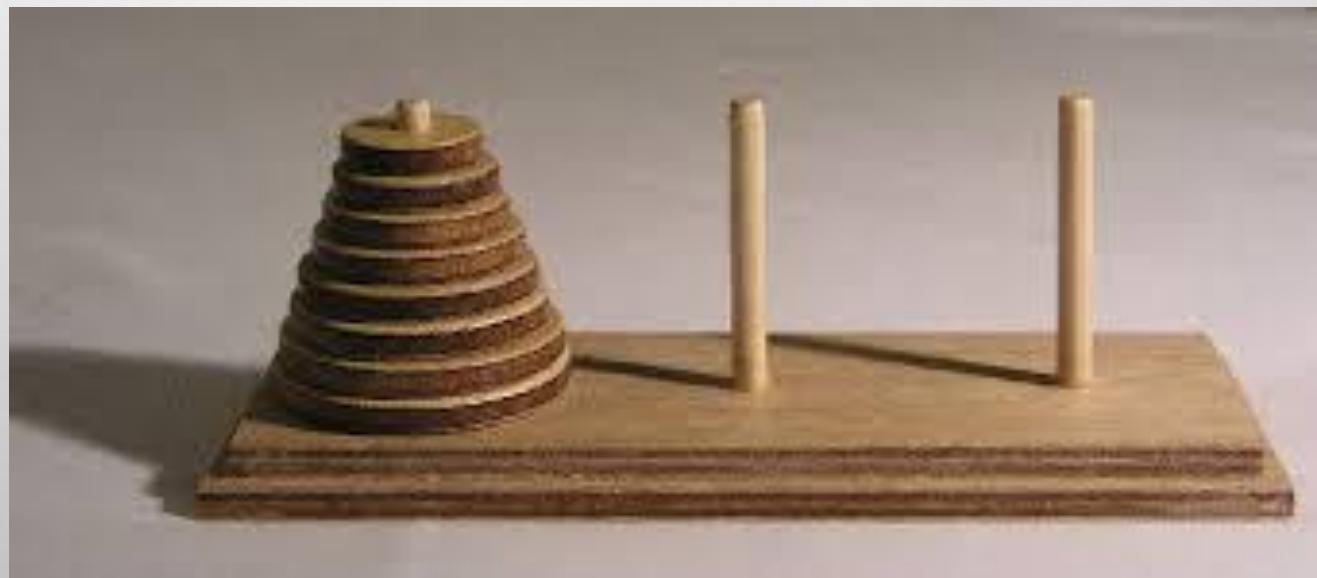
2.1 Các trò chơi trên máy tính

Các trò chơi trí tuệ



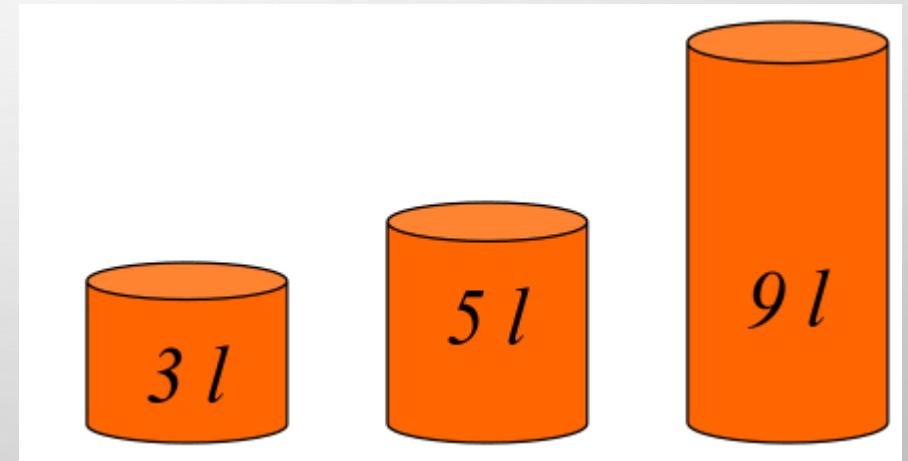
Bài toán tháp Hà Nội

- Cho n cọc và m đĩa ban đầu nằm trên cùng 1 cọc. Cần chuyển toàn bộ các đĩa sang một cọc đích.
- Điều kiện:
 - Mỗi bước chỉ nhấc 1 đĩa.
 - Đĩa nhỏ luôn ở trên đĩa to.



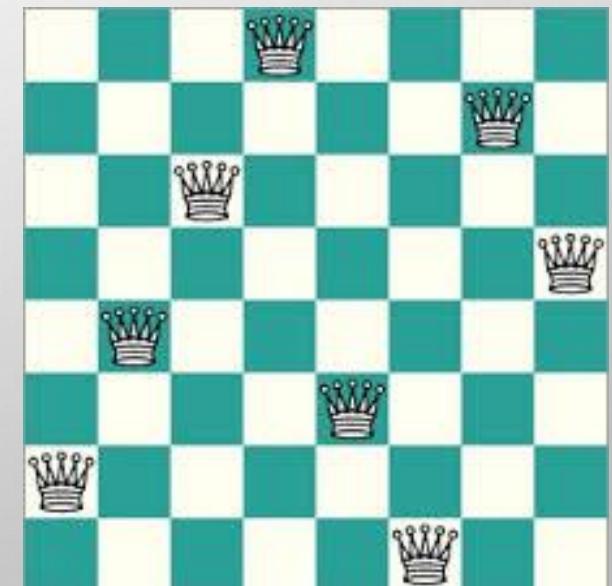
Bài toán rót nước

- Cho 3 bình dung tích 3 lít, 5 lít, 9 lít. Cần đong 7 lít
- Tổng quát: cho n bình dung tích m_1, m_2, \dots, m_n . Cần đong được k lít.



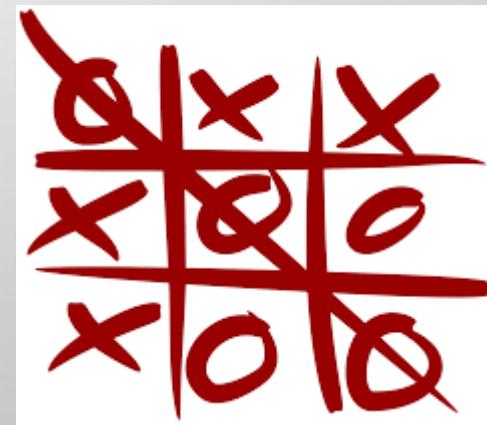
Bài toán 8 con hậu

- Đặt 8 con hậu vào bàn cờ 8×8
- Tổng quát: đặt n con hậu vào bàn cờ $n \times n$
- Điều kiện: không có 2 con hậu nào cùng hàng, cùng cột, cùng đường chéo



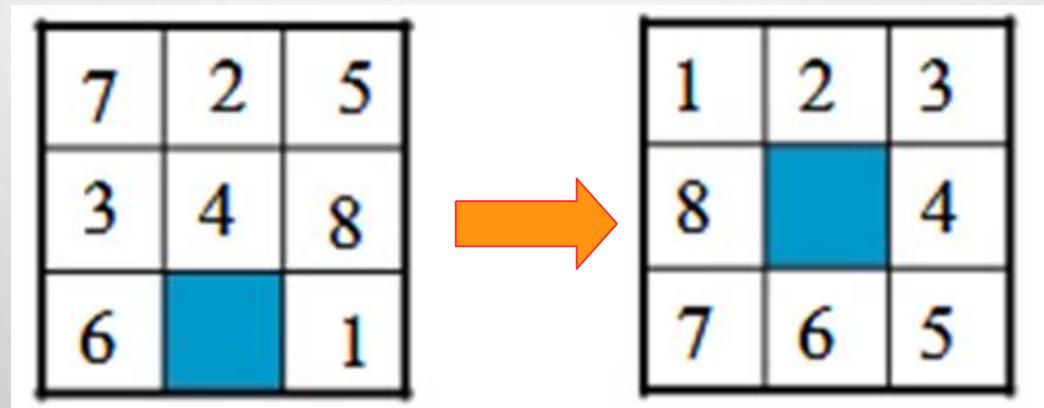
Bài toán cờ tic-tac-toe

- 2 người chơi lần lượt điền kí hiệu \times hoặc \circ của mình lên bàn cờ 3×3
- Người thắng tạo được dây 3 kí hiệu (thắng hàng, cột hay chéo) của mình trước



Bài toán 8-puzzle

- Cho 1 bảng 3×3 trong đó có 8 ô được đánh số từ 1 đến 8 và 1 ô trống
- Cần dịch chuyển liên tiếp ô trống sang các ô bên cạnh để dẫn đến 1 bàn cờ mong muốn cho trước.



2.2 Biểu diễn bài toán trong không gian trạng thái

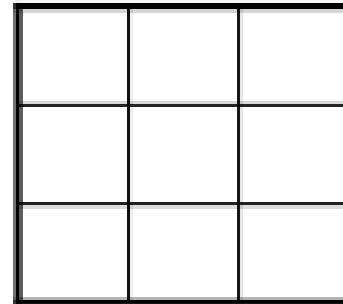
Trạng thái

- Trạng thái:

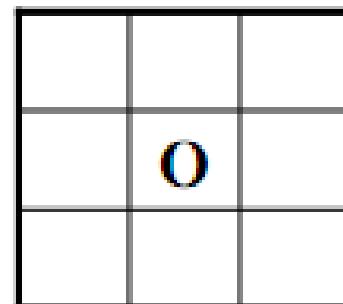
- Biểu diễn 1 bước nào đó của bài toán.
- Cách mã hóa bài toán trong máy tính.

- Ví dụ:

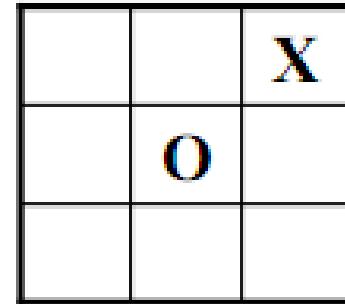
- Tic-tac-toe, 8 con hậu, puzzle: tình trạng bàn cờ
- Tháp Hà Nội: vị trí cọc của các đĩa



Trạng thái



Trạng thái

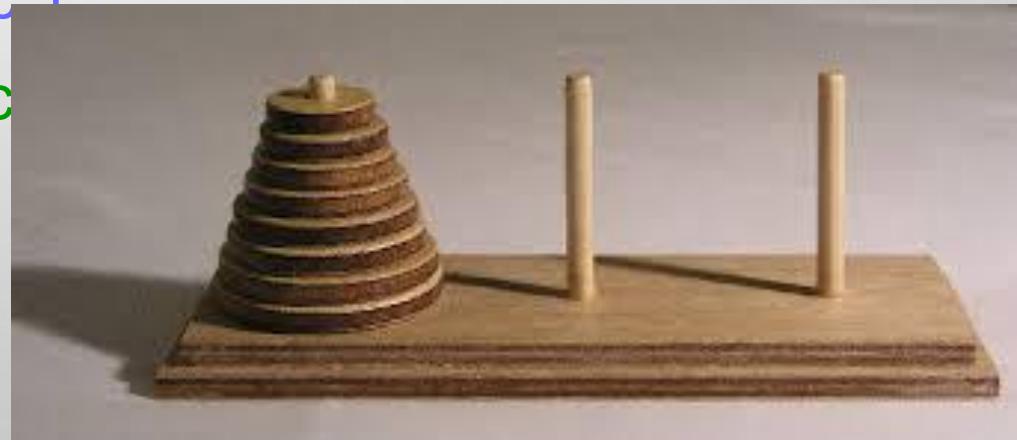


Trạng thái

Trạng thái

Bài toán tháp Hà Nội (3 cọc, 3 đĩa)

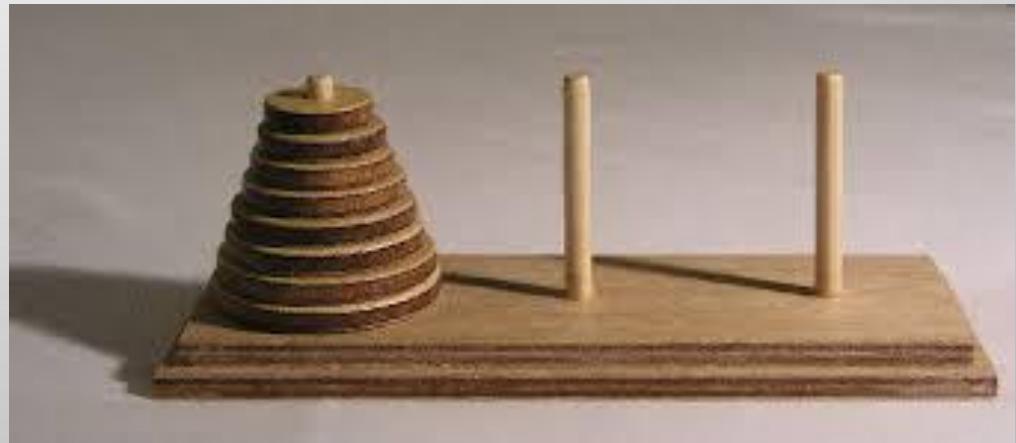
- Trạng thái: vị trí các đĩa
- Biểu diễn: (a,b,c)
 - a,b,c : lần lượt vị trí cọc của đĩa to nhất, thứ 2 và thứ 3.
 - Ví dụ $(3,1,2)$:
 - đĩa to nhất ở cọc 3
 - đĩa to thứ 2 ở cọc 1
 - đĩa to thứ 3 ở cọc
- Ban đầu: $(1,1,1)$
- Đích: $(3,3,3)$



Trạng thái

Bài toán tháp Hà Nội (n cọc, m đĩa)

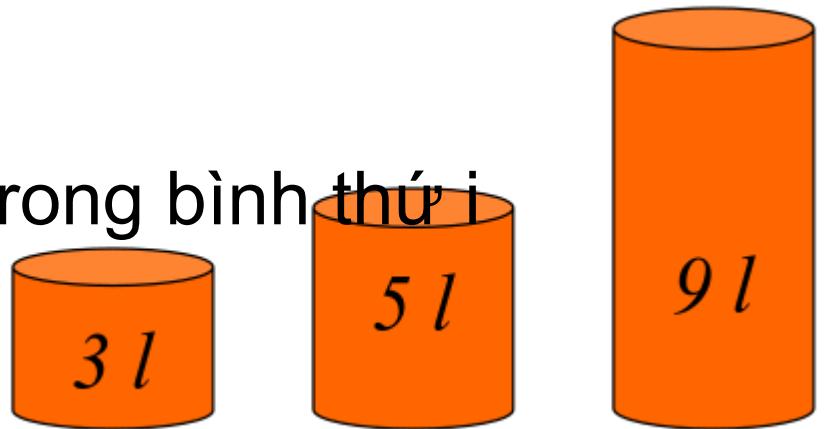
- Trạng thái: vị trí các đĩa
- Biểu diễn: (a_1, a_2, \dots, a_m)
 - a_i : vị trí cọc của đĩa to thứ i
 - $1 \leq a_i \leq n$
- Ban đầu: $(1, 1, \dots, 1)$
- Kết thúc: (n, n, \dots, n)



Trạng thái

Bài toán rót nước (cụ thể)

- Cho 3 bình dung tích 3 lít, 5 lít, 9 lít. Cần đong 7 lít
- Trạng thái: số lượng nước chứa trong mỗi bình
- Biểu diễn (a_1, a_2, a_3)
 - a_i : số lít nước đang chứa trong bình thứ i
- Ban đầu: $(0,0,0)$
- Đích: $(-, -, 7)$, ...



Trạng thái

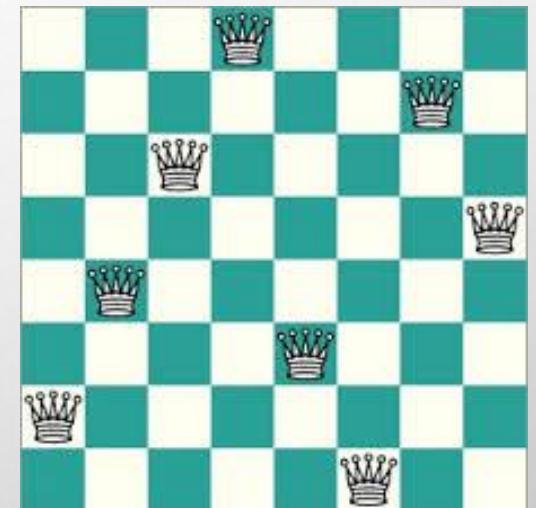
Bài toán rót nước (tổng quát)

- Cho n bình dung tích m_1, m_2, \dots, m_n . Cần đong được k lít.
- Trạng thái: số lượng nước chứa trong mỗi bình
- Biểu diễn: (a_1, a_2, \dots, a_n)
 - $0 \leq a_i \leq m_i$ là số lít nước đang chứa trong bình thứ i
- Ban đầu: $(0, 0, \dots, 0)$
- Đích: $(k, -, \dots, -)$ hoặc $(-, k, \dots, -)$, ..., hoặc $(-, -, \dots, k)$

Trạng thái

Bài toán n hậu

- Trạng thái: vị trí các con hậu trên bàn cờ
- Biểu diễn
- Cách 1: (a_1, a_2, \dots, a_n)
 - $1 \leq a_i \leq n$ là vị trí hàng của con hậu ở cột i
 - Ví dụ: $(7, 5, 3, 1, 6, 8, 2, 4)$
 - Ban đầu: bất kì tình trạng sắp xếp nào
 - Đích: trạng thái thỏa mãn điều kiện ràng buộc



Trạng thái

Bài toán n hậu

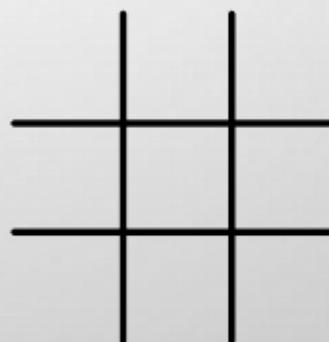
- Trạng thái: tình trạng của các ô trong bàn cờ
- Biểu diễn
- Cách 2: (a_1, a_2, \dots, a_n)
 - a_i là tình trạng ô thứ i
 - $a_i = \text{true}(T)/\text{false}(F)$ khi ô thứ i có hậu/trống
 - Ví dụ: $(F, F, F, \textcolor{red}{T}, F, F, F, F,$



Trạng thái

Tic-tac-toe

- Trạng thái: nội dung bàn cờ
- Biểu diễn: (a_1, a_2, \dots, a_9)
 - a_i : tình trạng của ô thứ i
 - $a_i = 0, 1$ hoặc 2 khi ô thứ i lần lượt trống, đánh dấu bởi \circ hoặc \times
- Ban đầu: $(0, 0, \dots, 0)$
- Đích: $(2, 0, 1, 0, 2, 1, 0, 1, 2), \dots$



Trạng thái đầu



Trạng thái đích

Trạng thái

8-puzzle

- Trạng thái: nội dung bảng
- Biểu diễn: (a_1, a_2, \dots, a_9)
 - a_i : nội dung điền vào ô thứ i
 - a_i thuộc $[0..8]$ trong đó 0 biểu diễn trường hợp ô trống
- Ban đầu: $(7, 2, 5, 3, 4, 8, 6, 0, 1)$
- Đích: $(1, 2, 3, 8, 0, 4, 7, 6, 5)$

7	2	5
3	4	8
6		1

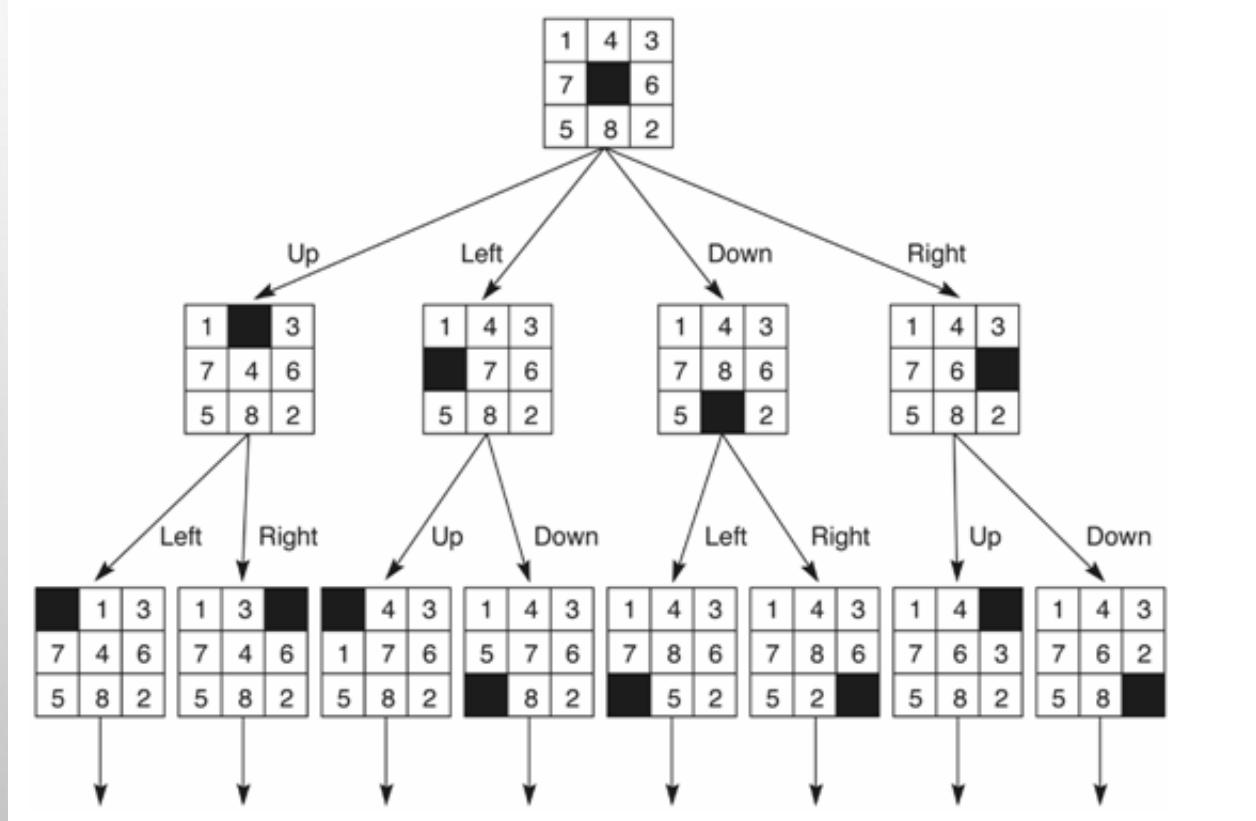
Trạng thái đầu

1	2	3
8		4
7	6	5

Trạng thái đích

Không gian trạng thái (KGTT)

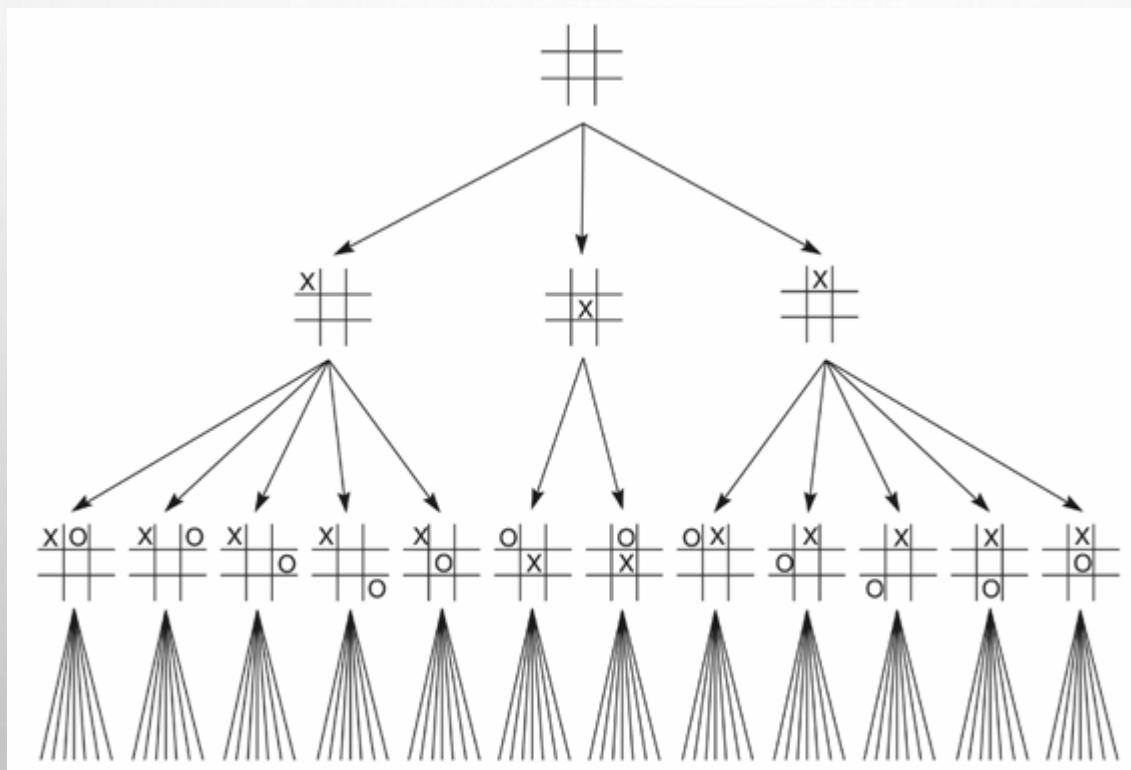
- Là đồ thị có hướng trong đó:
 - Mỗi nút là 1 trạng thái
 - Mỗi cạnh ứng với 1 phép chuyển đổi trạng thái



Không gian trạng thái của bài toán 8-puzzle
<https://voer.edu.vn/c/mo-dau/764b3239/70515552>

Không gian trạng thái (KGTT)

- Là đồ thị có hướng trong đó:
 - Mỗi nút là 1 trạng thái
 - Mỗi cạnh ứng với 1 phép chuyển đổi trạng thái



Không gian trạng thái của bài toán Tic-tac-toe
<https://voer.edu.vn/c/mo-dau/764b3239/2ab0f63d>

Không gian trạng thái (KGTT)

Để xây dựng không gian trạng thái cho 1 bài toán, ta cần xác định:

1) Trạng thái đầu

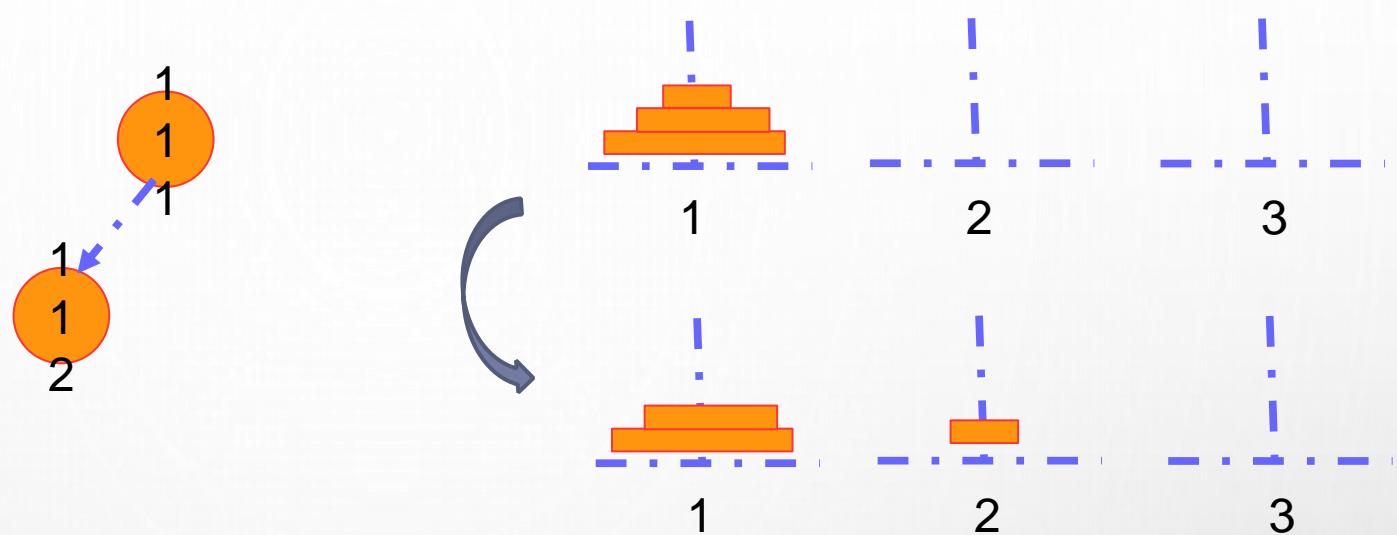
1) Trạng thái xuất phát. Một bài toán có thể có nhiều trạng thái xuất phát

2) Tập trạng thái đích

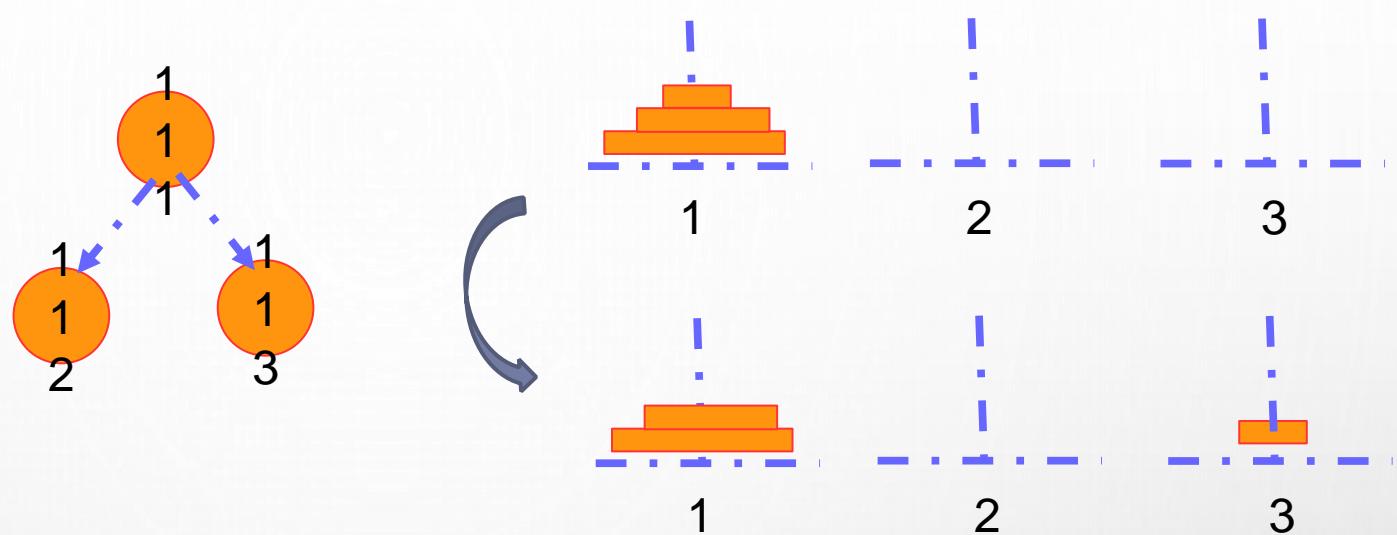
- Trạng thái mà bài toán được giải. Một bài toán có thể có nhiều trạng thái

3) Các phép chuyển

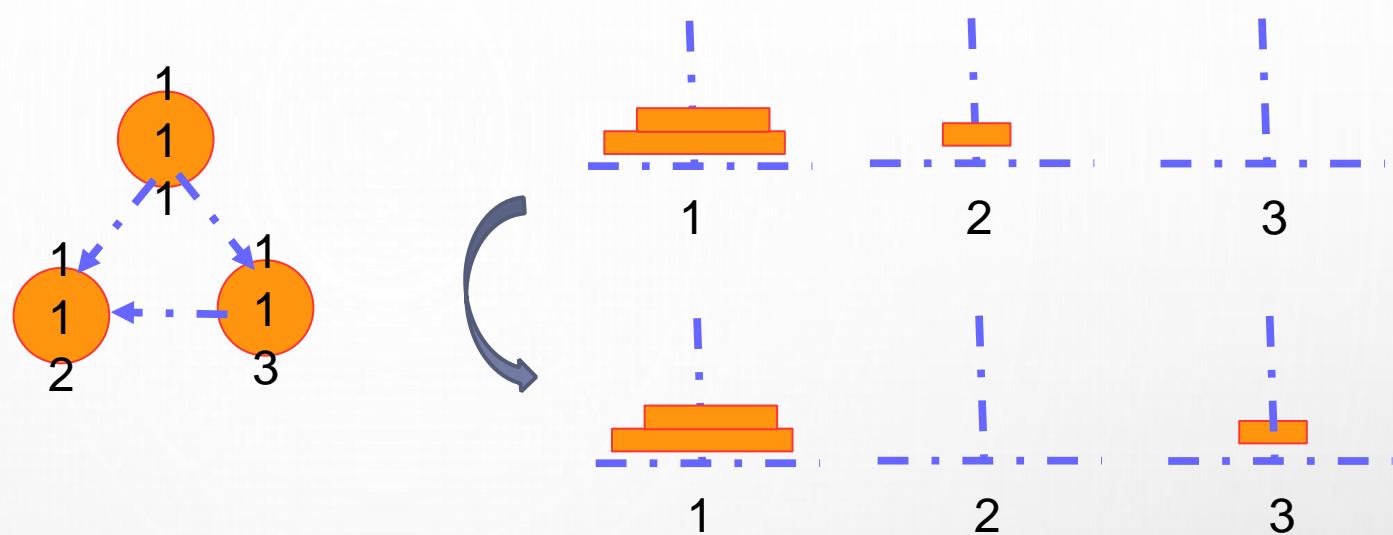
Không gian trạng thái của bài toán tháp Hà Nội



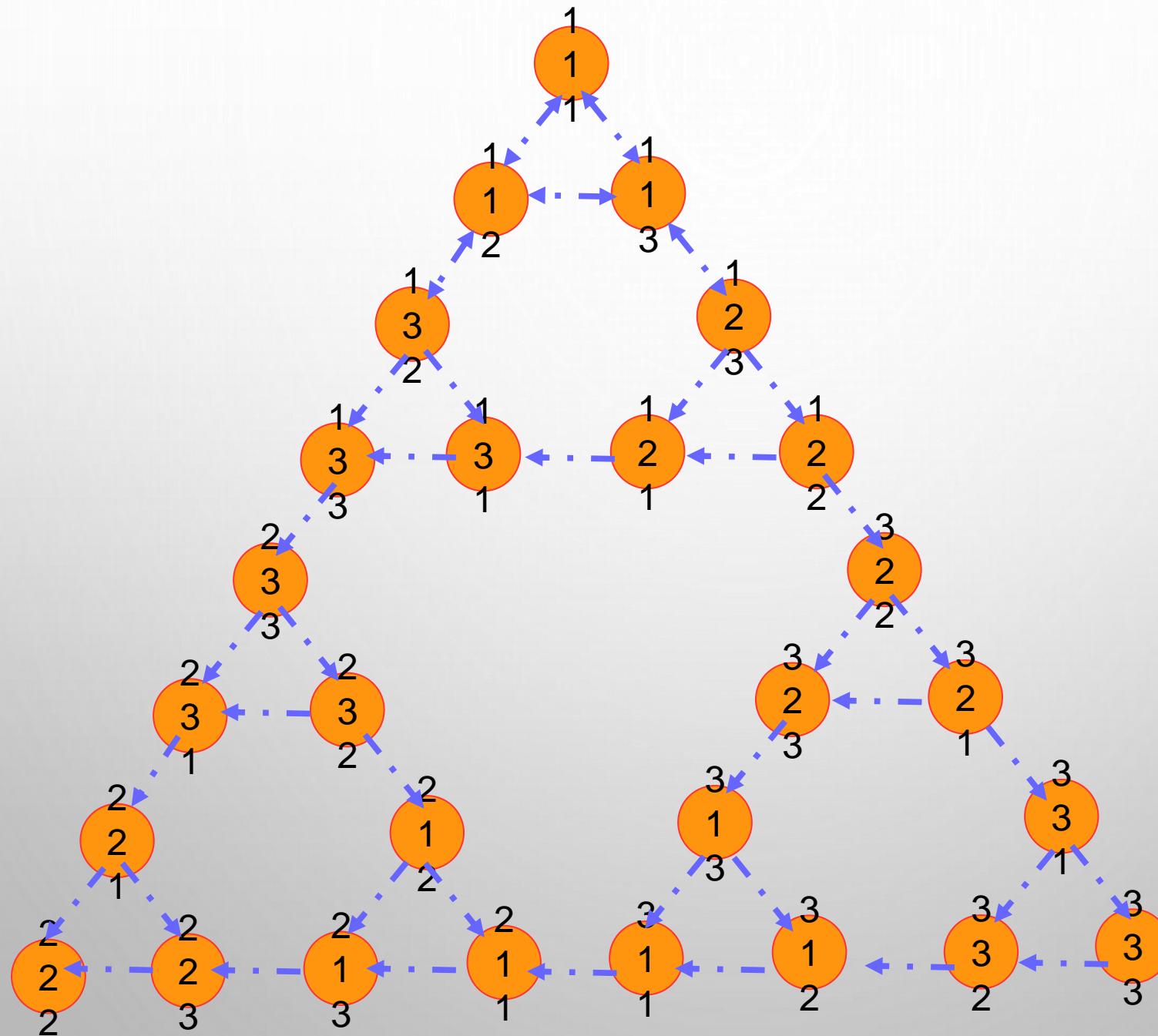
Không gian trạng thái của bài toán tháp Hà Nội



Không gian trạng thái của bài toán tháp Hà Nội

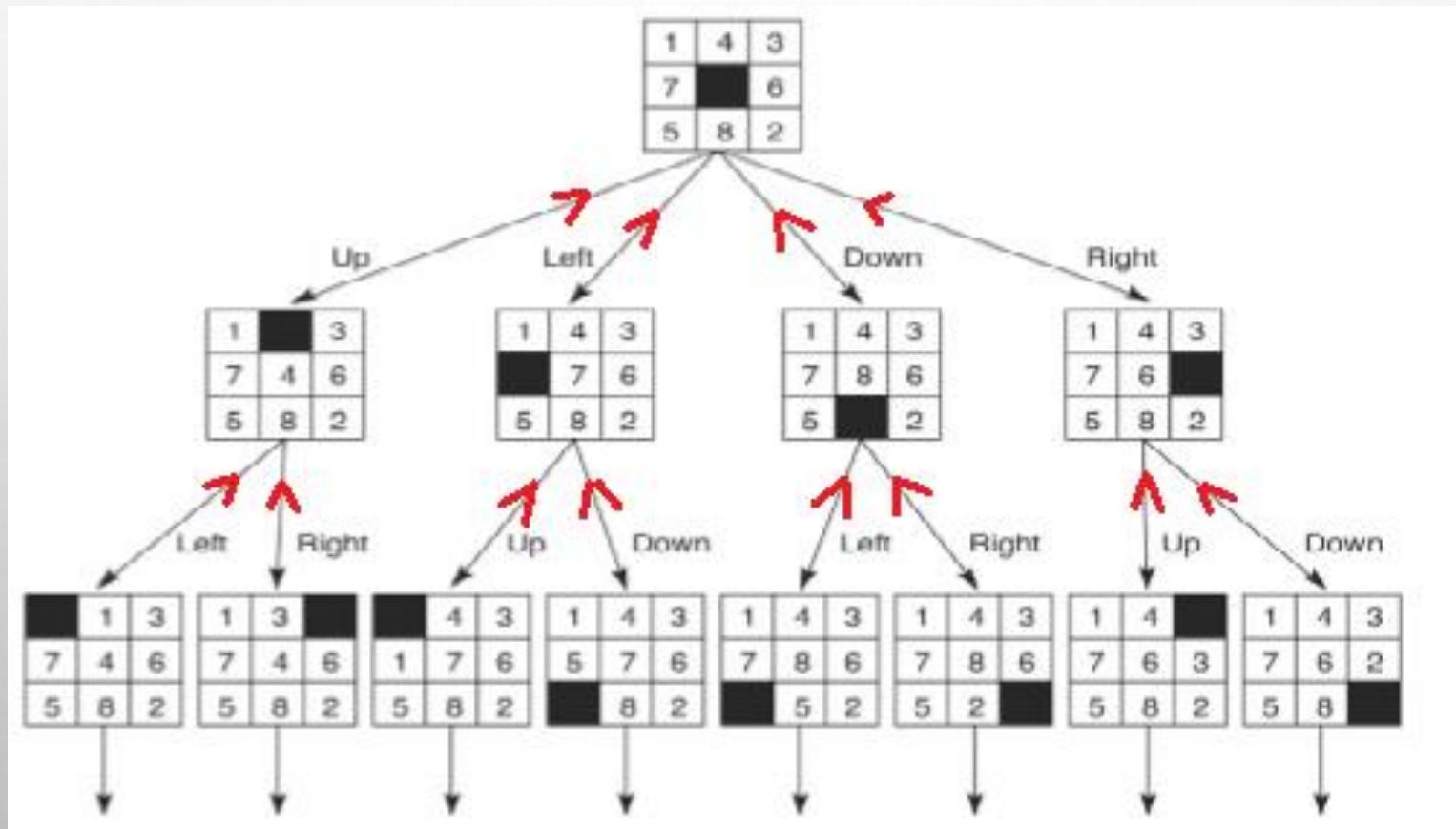


Không gian trạng thái của bài toán tháp Hà Nội (mũi tên 2 chiều)



Không gian trạng thái (KGTT)

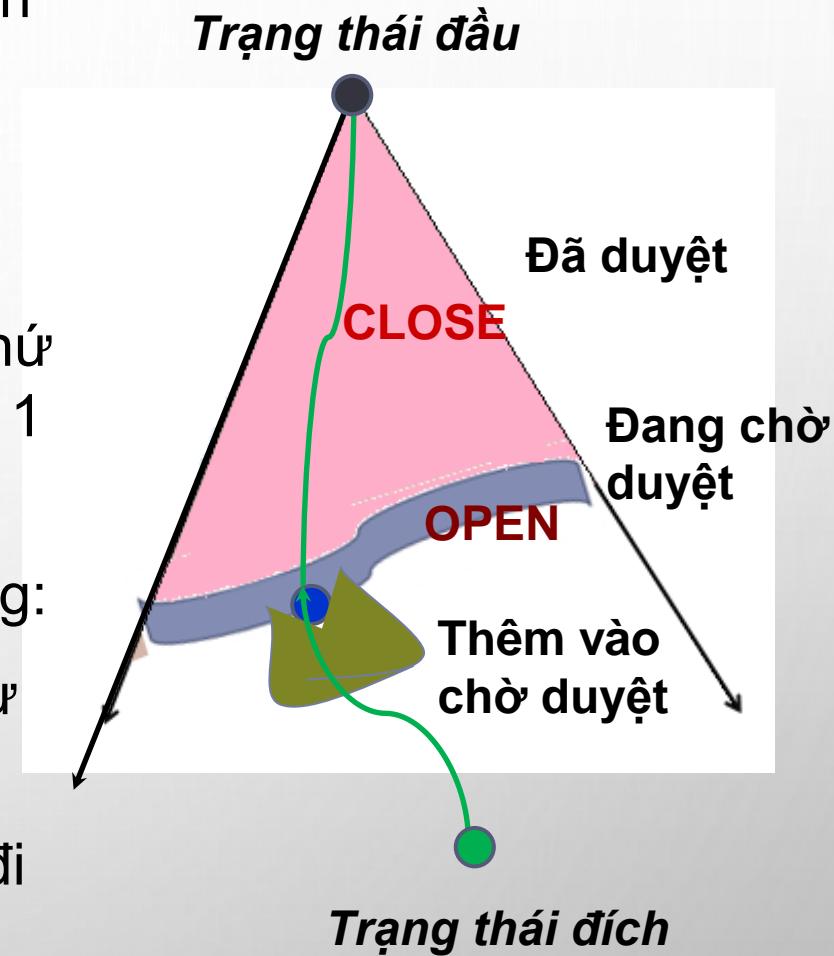
- Có thể là đồ thị chưa vòng lặp
- Không nhất thiết là cây



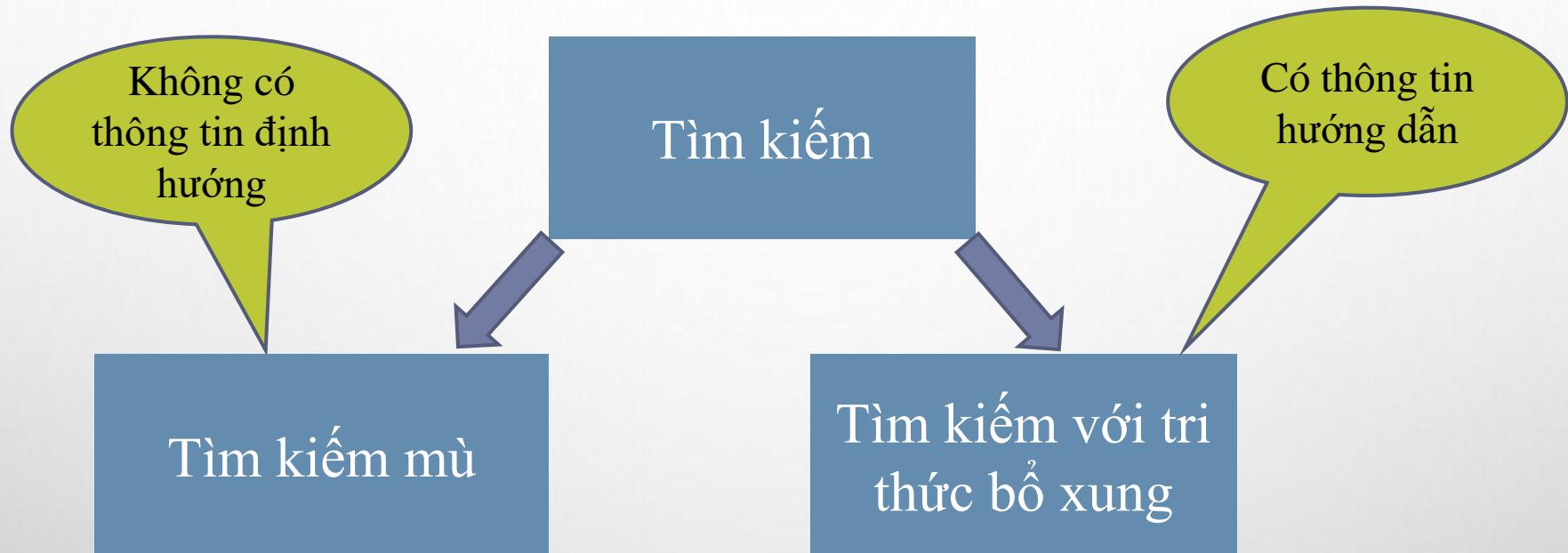
2.3 Giải quyết bài toán bằng tìm kiếm

Giải quyết bài toán bằng tìm kiếm

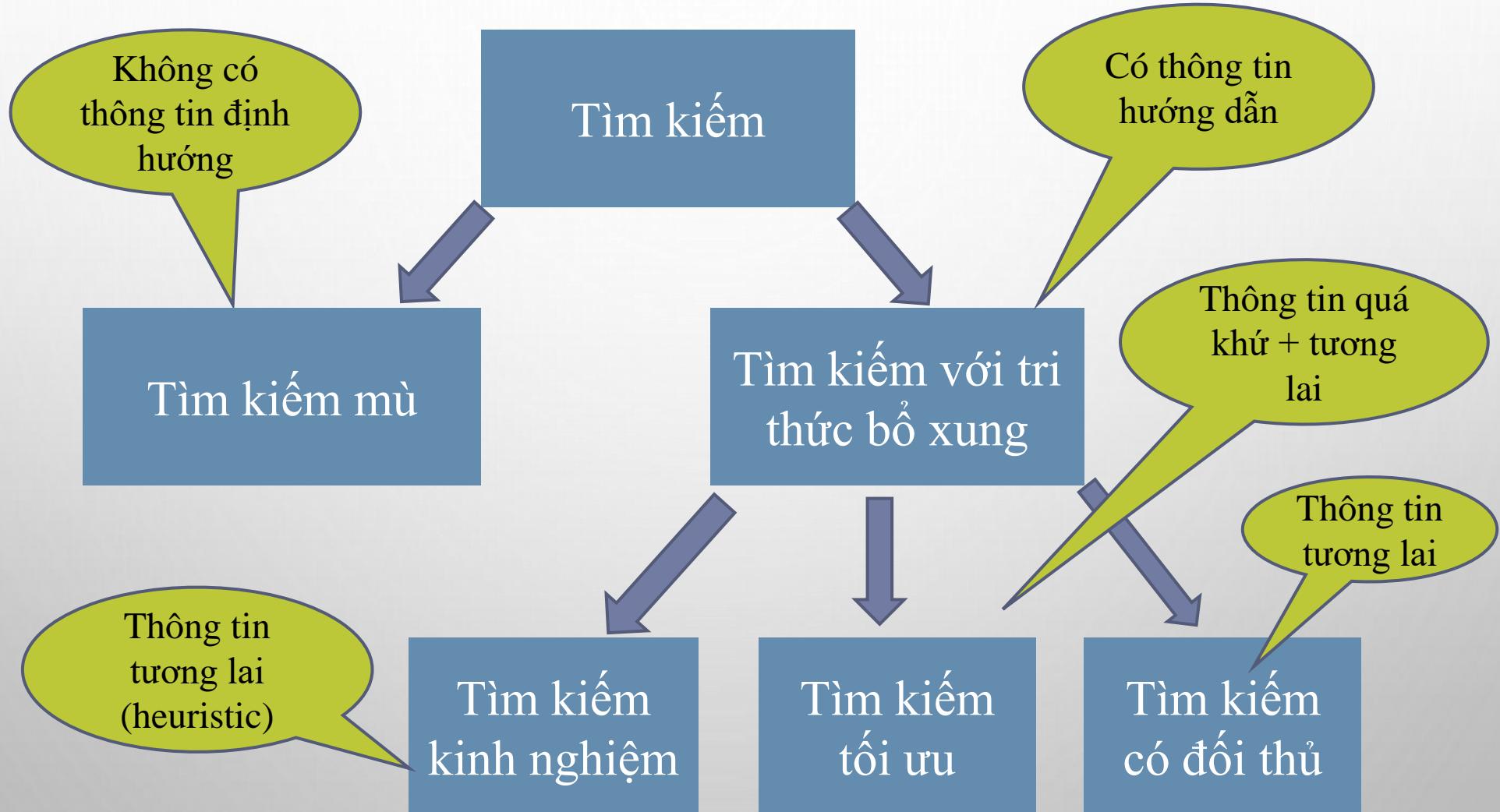
- Là tìm đường đi từ TT đầu tới một TT đích trong KGTT
 - Lời giải: tập các phép chuyển gắn với đường đi tìm được.
- Các chiến lược tìm kiếm khác nhau bởi thứ tự phát triển các nút để duyệt (cách chọn 1 trạng thái tiếp theo của để duyệt)
- Các thông tin được sử dụng để tìm đường:
 - Quá khứ: đánh giá chi phí đường đi từ gốc tới nút đang duyệt
 - Tương lai: ước lượng chi phí đường đi từ nút đang duyệt tới đích



Các chiến lược tìm kiếm



Các chiến lược tìm kiếm



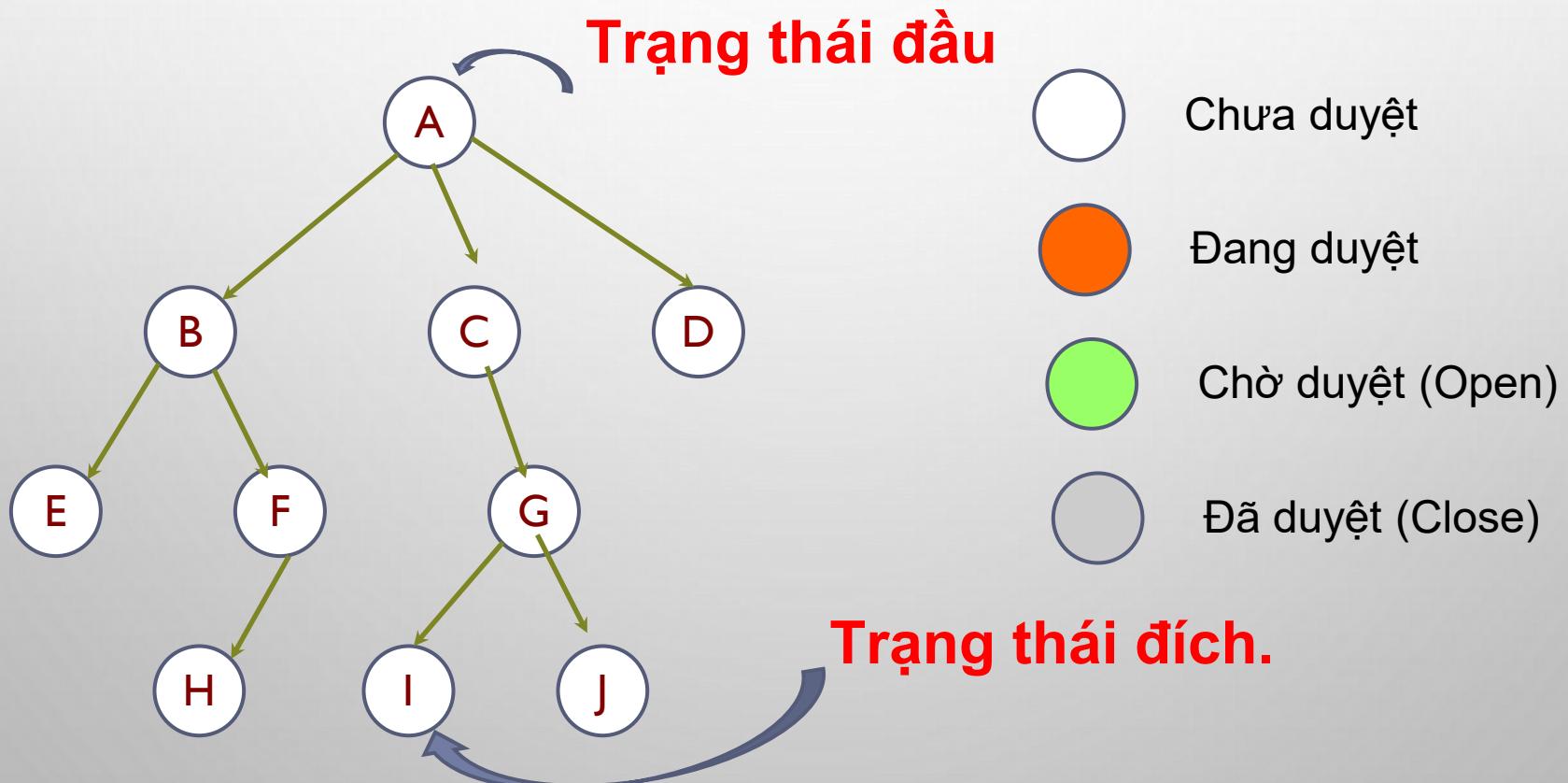
Tìm kiếm mù

Tìm kiếm mù

- Tìm kiếm không có định hướng: không có thông tin đánh giá (quá khứ, tương lai) của các trạng thái
- Từ trạng thái ban đầu, các trạng thái được phát triển theo 1 quy tắc lựa chọn cứng nhắc nào đó cho tới khi:
 - Gặp được 1 trạng thái đích → tìm kiếm thành công
 - Duyệt hết không gian trạng thái → tìm kiếm thất bại
- Các chiến lược tìm kiếm mù:
 - Tìm kiếm theo chiều rộng (Breadth-First-Search: BFS)

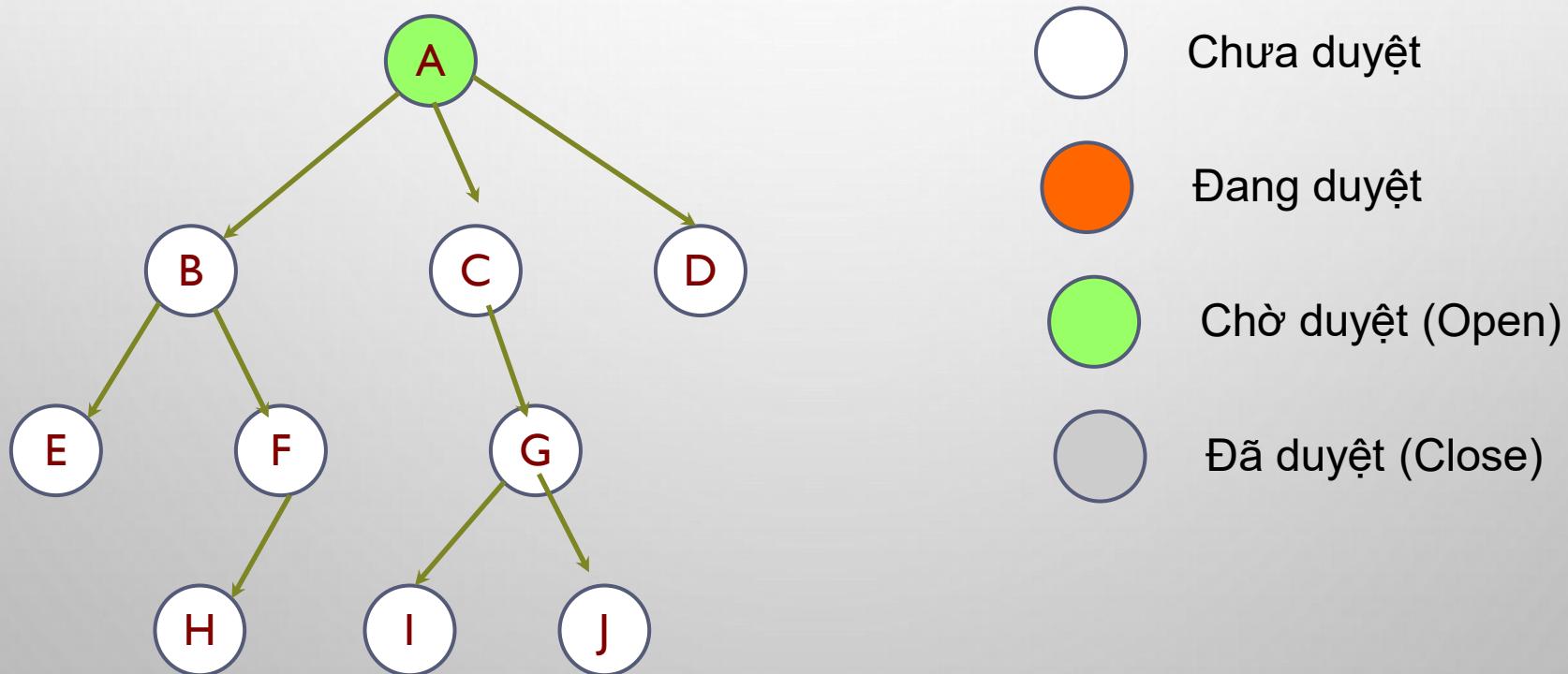
Tìm kiếm theo chiều rộng

- Tập trạng thái chờ duyệt “OPEN”: Queue
- Trạng thái nào được sinh ra trước thì được phát triển trước
- Duyệt hết các nút ở cùng độ sâu rồi mới duyệt tới các nút ở độ sâu tiếp theo



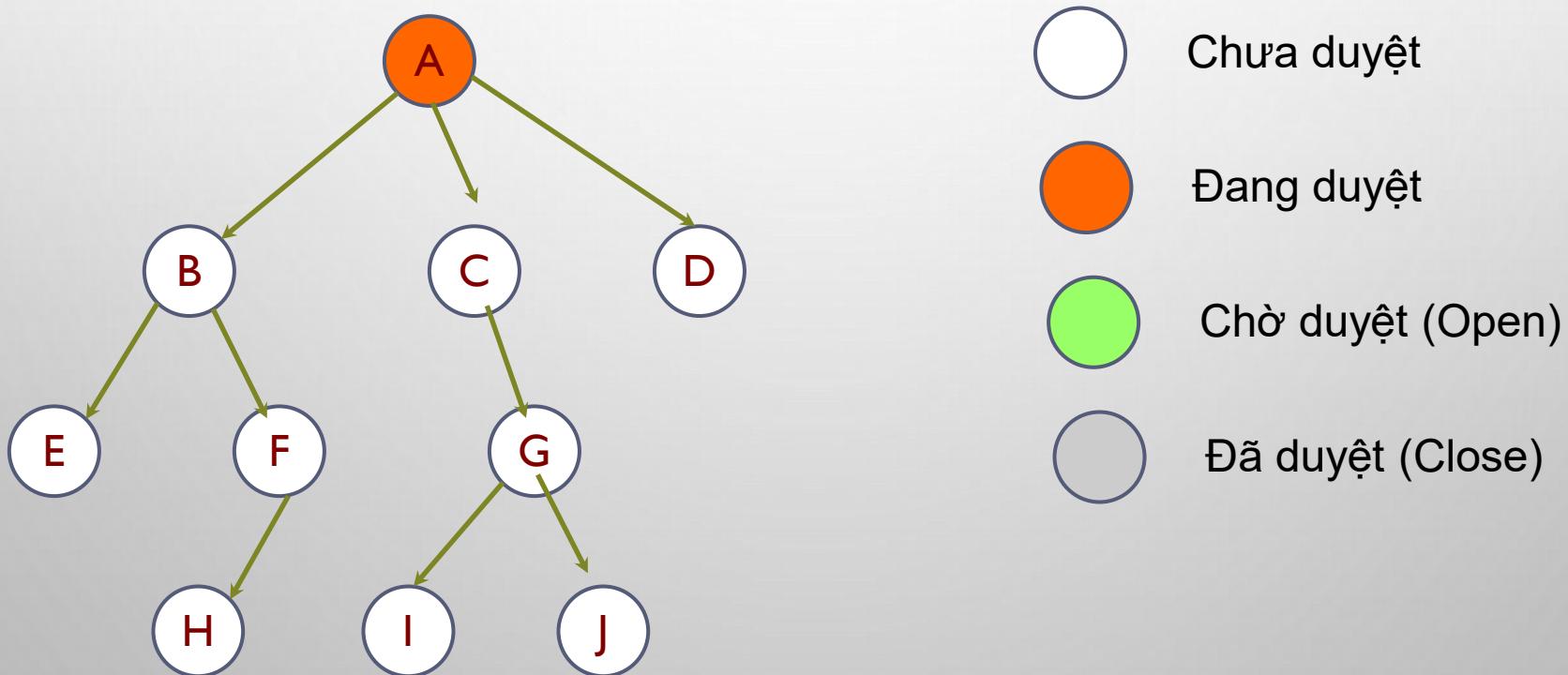
Tìm kiếm theo chiều rộng

- Tập trạng thái chờ duyệt “OPEN”: Queue
- Trạng thái nào được sinh ra trước thì được phát triển trước
- Duyệt hết các nút ở cùng độ sâu rồi mới duyệt tới các nút ở độ sâu tiếp theo



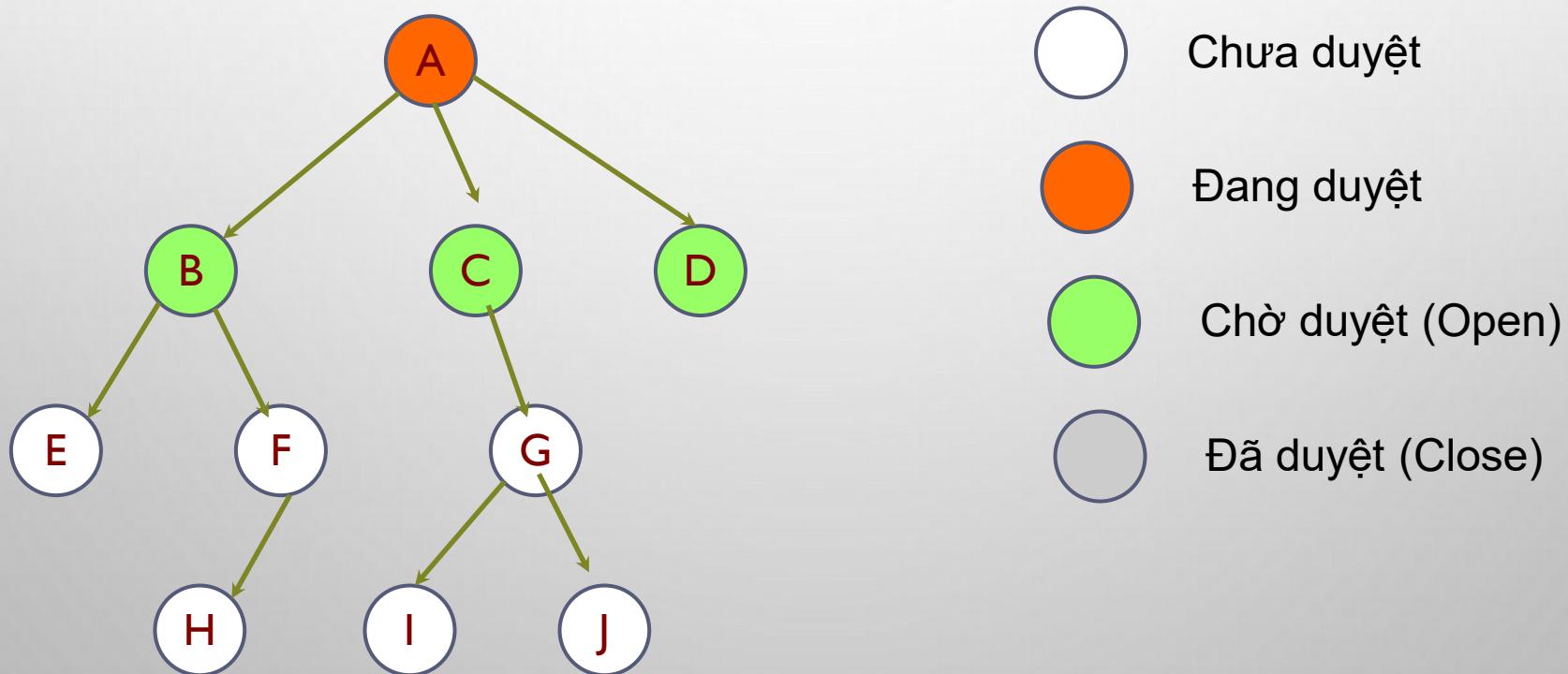
Tìm kiếm theo chiều rộng

- Tập trạng thái chờ duyệt “OPEN”: Queue
- Trạng thái nào được sinh ra trước thì được phát triển trước
- Duyệt hết các nút ở cùng độ sâu rồi mới duyệt tới các nút ở độ sâu tiếp theo



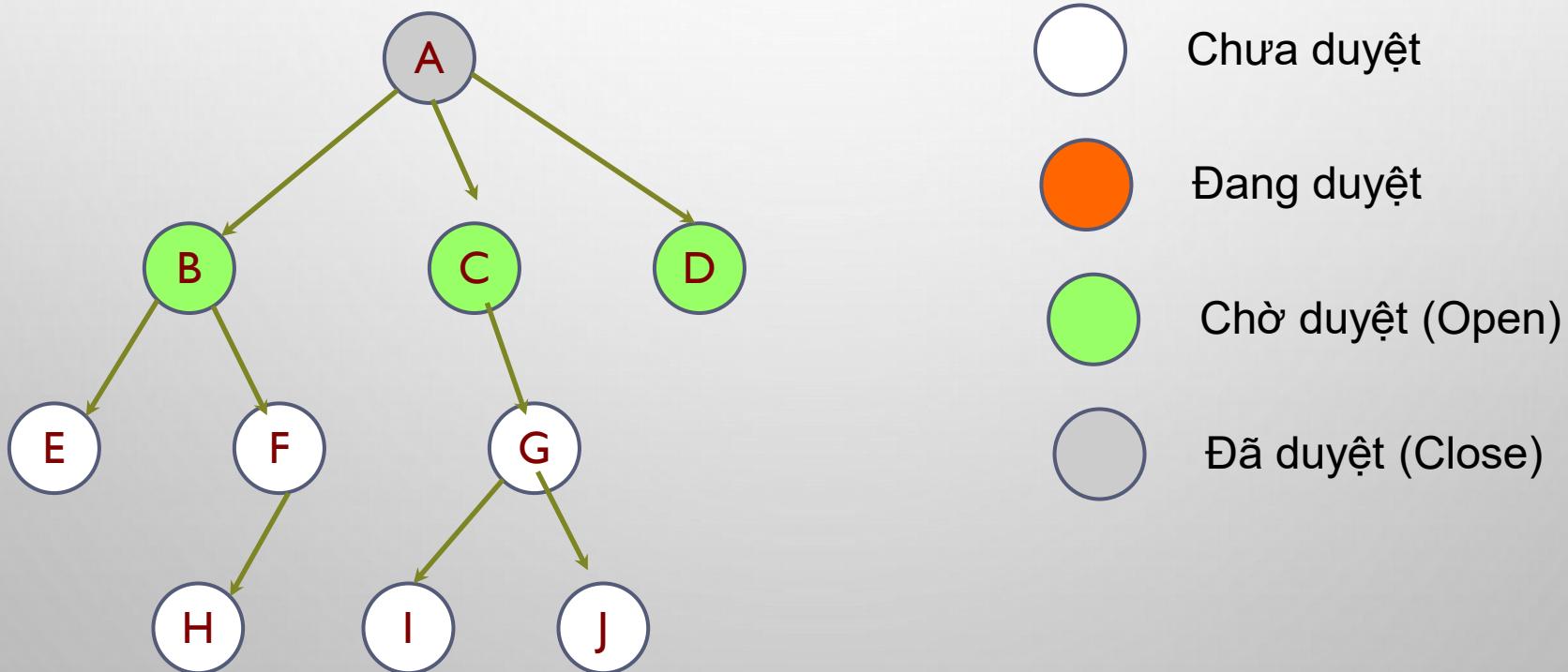
Tìm kiếm theo chiều rộng

- Tập trạng thái chờ duyệt “OPEN”: Queue
- Trạng thái nào được sinh ra trước thì được phát triển trước
- Duyệt hết các nút ở cùng độ sâu rồi mới duyệt tới các nút ở độ sâu tiếp theo



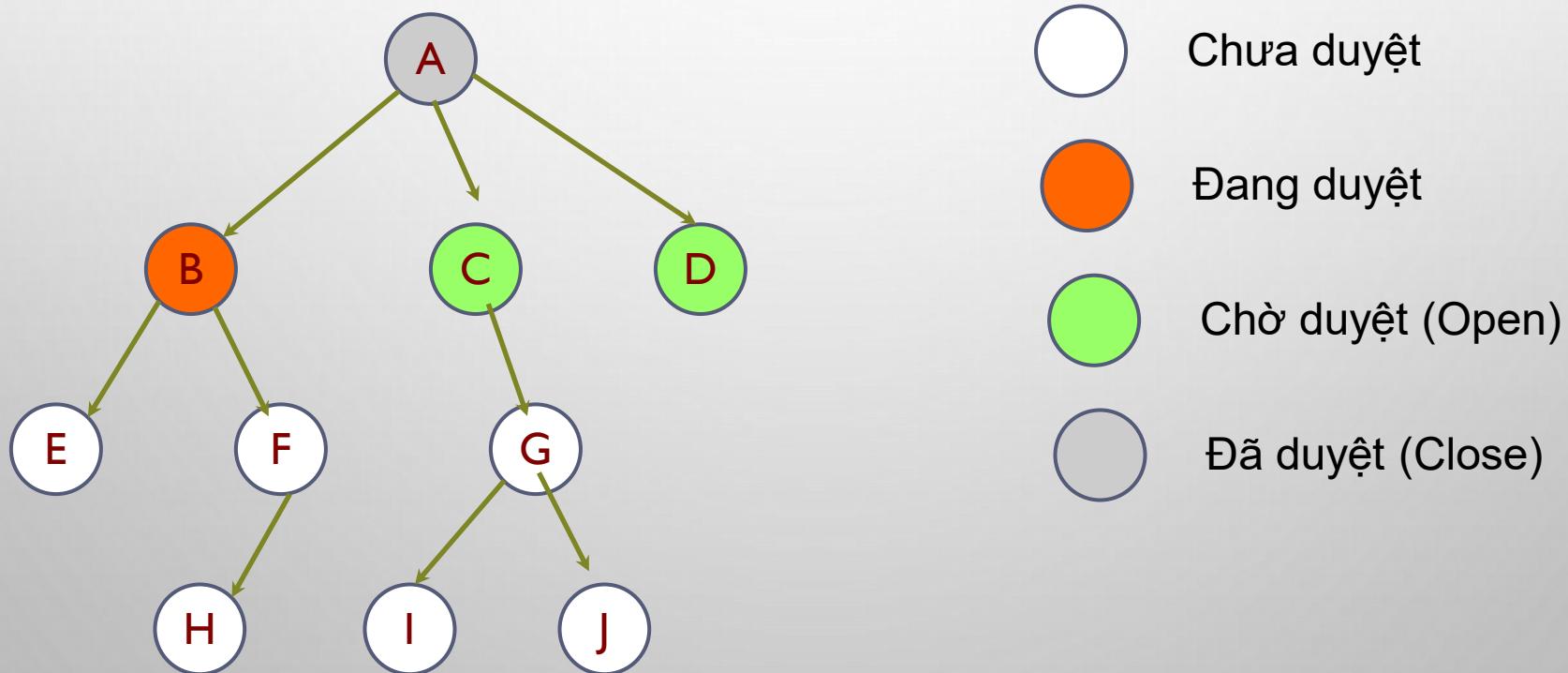
Tìm kiếm theo chiều rộng

- Tập trạng thái chờ duyệt “OPEN”: Queue
- Trạng thái nào được sinh ra trước thì được phát triển trước
- Duyệt hết các nút ở cùng độ sâu rồi mới duyệt tới các nút ở độ sâu tiếp theo



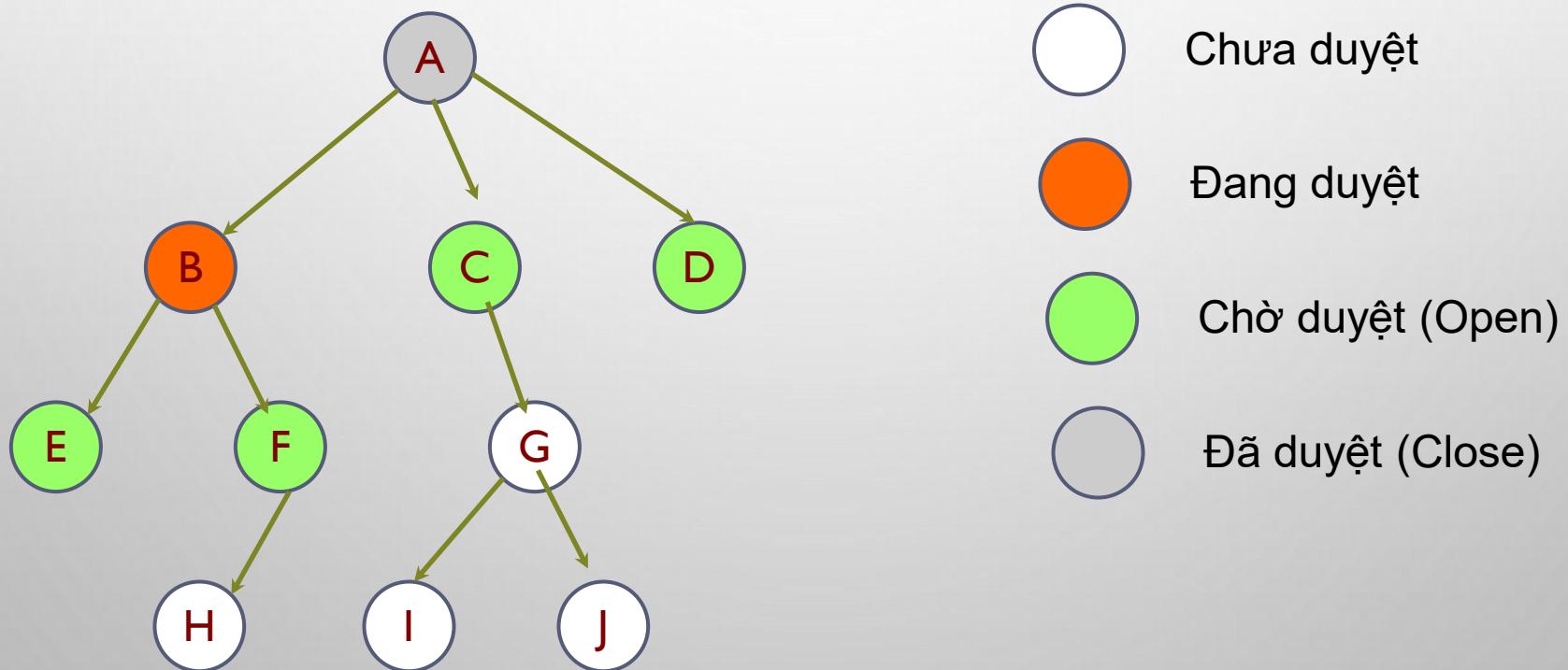
Tìm kiếm theo chiều rộng

- Tập trạng thái chờ duyệt “OPEN”: Queue
- Trạng thái nào được sinh ra trước thì được phát triển trước
- Duyệt hết các nút ở cùng độ sâu rồi mới duyệt tới các nút ở độ sâu tiếp theo



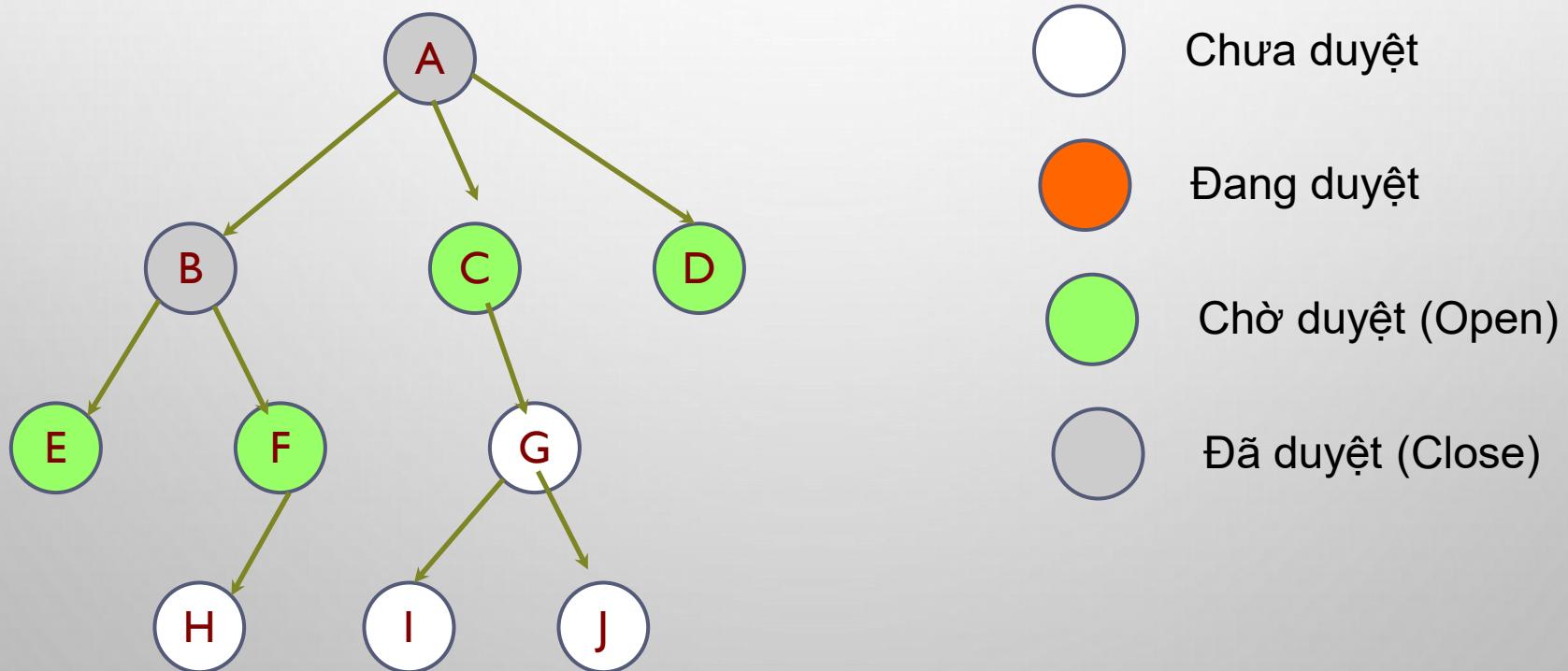
Tìm kiếm theo chiều rộng

- Tập trạng thái chờ duyệt “OPEN”: Queue
- Trạng thái nào được sinh ra trước thì được phát triển trước
- Duyệt hết các nút ở cùng độ sâu rồi mới duyệt tới các nút ở độ sâu tiếp theo



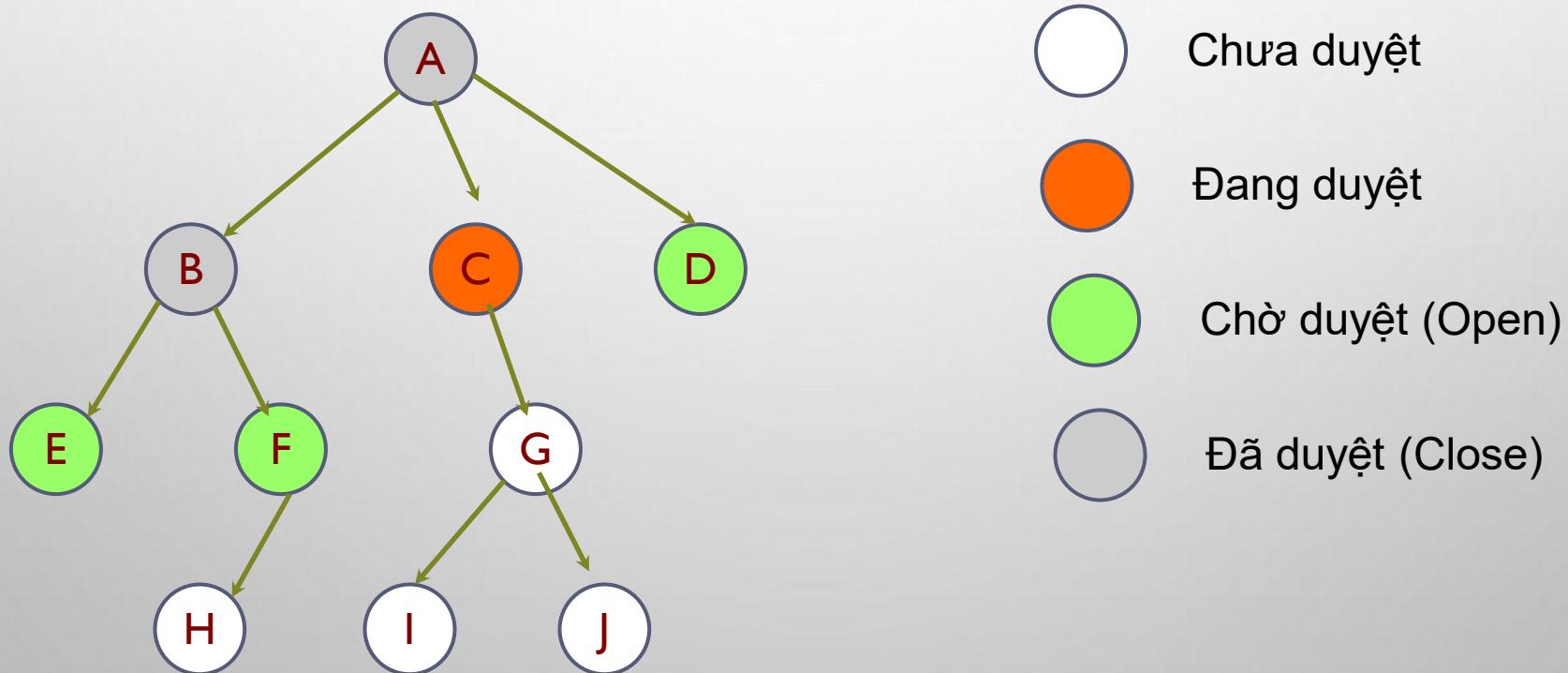
Tìm kiếm theo chiều rộng

- Tập trạng thái chờ duyệt “OPEN”: Queue
- Trạng thái nào được sinh ra trước thì được phát triển trước
- Duyệt hết các nút ở cùng độ sâu rồi mới duyệt tới các nút ở độ sâu tiếp theo



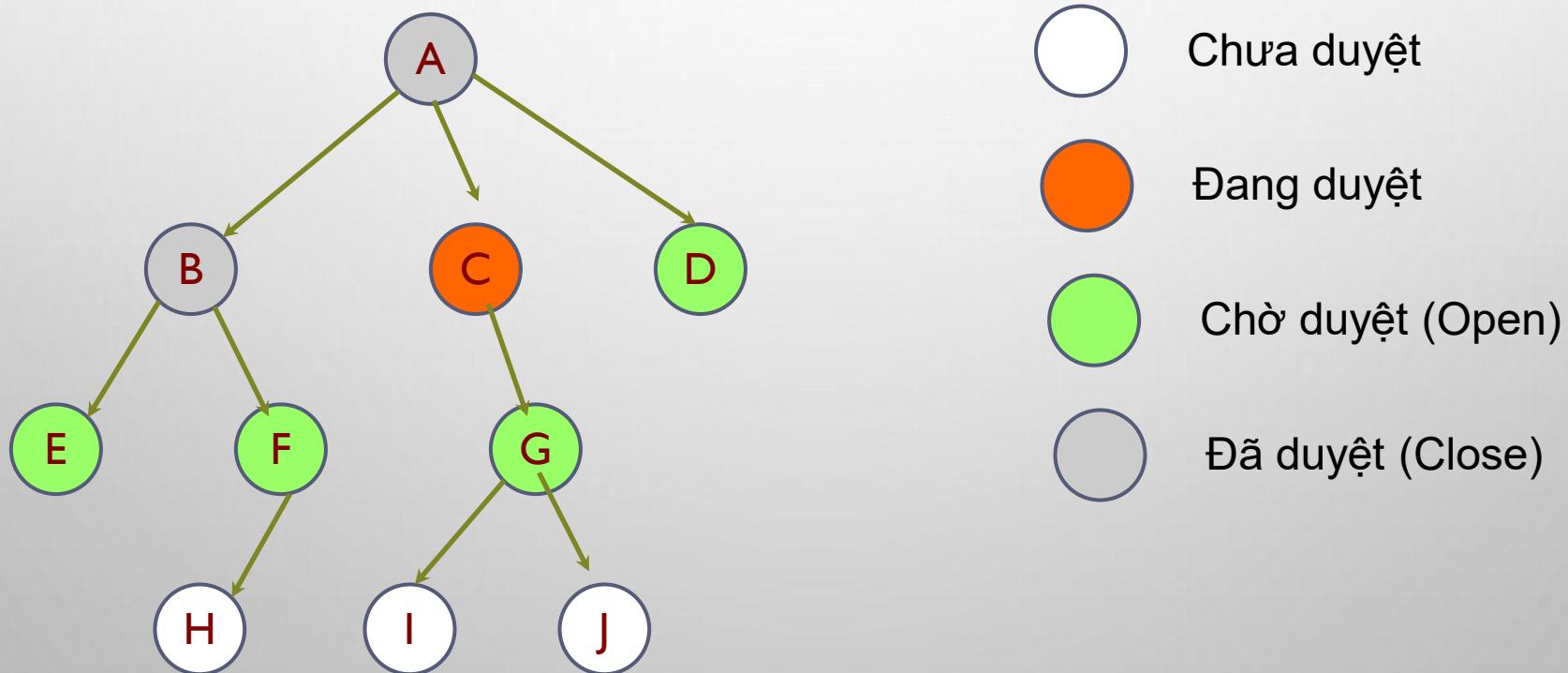
Tìm kiếm theo chiều rộng

- Tập trạng thái chờ duyệt “OPEN”: Queue
- Trạng thái nào được sinh ra trước thì được phát triển trước
- Duyệt hết các nút ở cùng độ sâu rồi mới duyệt tới các nút ở độ sâu tiếp theo



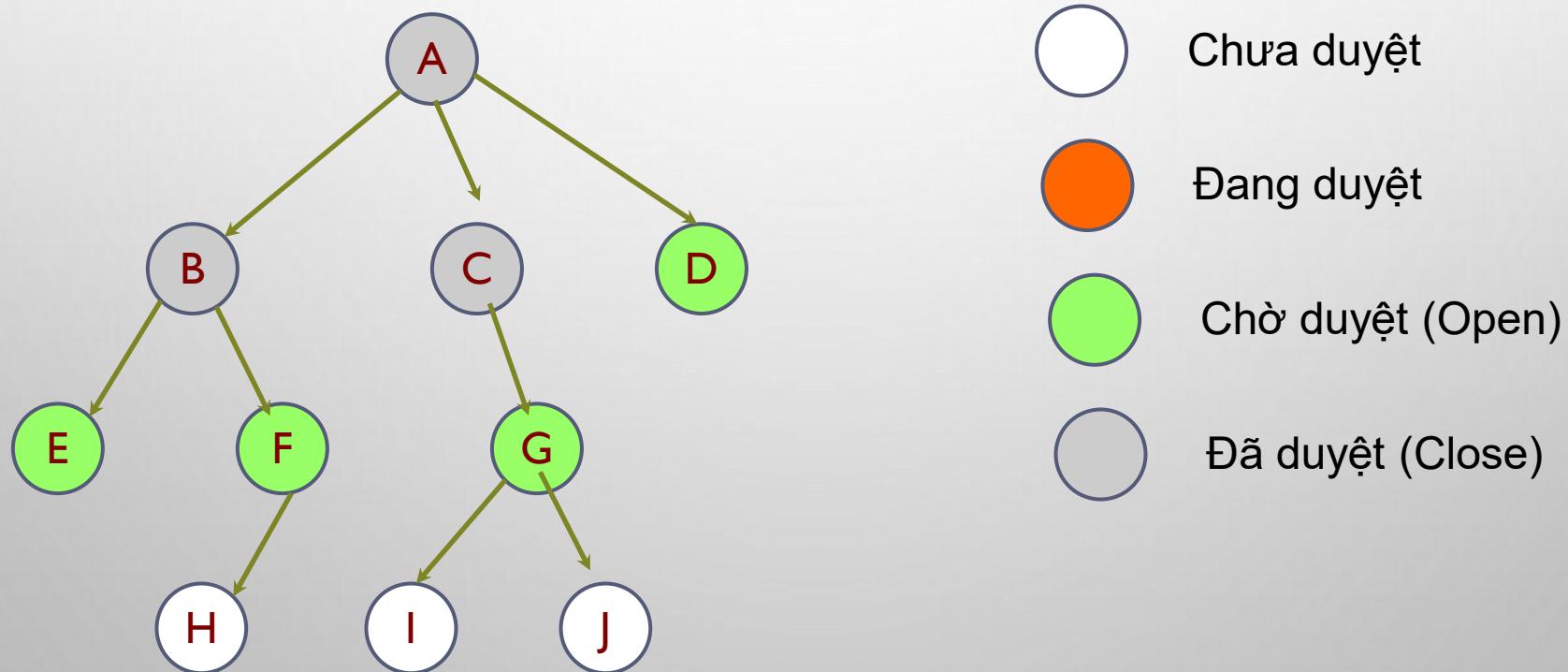
Tìm kiếm theo chiều rộng

- Tập trạng thái chờ duyệt “OPEN”: Queue
- Trạng thái nào được sinh ra trước thì được phát triển trước
- Duyệt hết các nút ở cùng độ sâu rồi mới duyệt tới các nút ở độ sâu tiếp theo



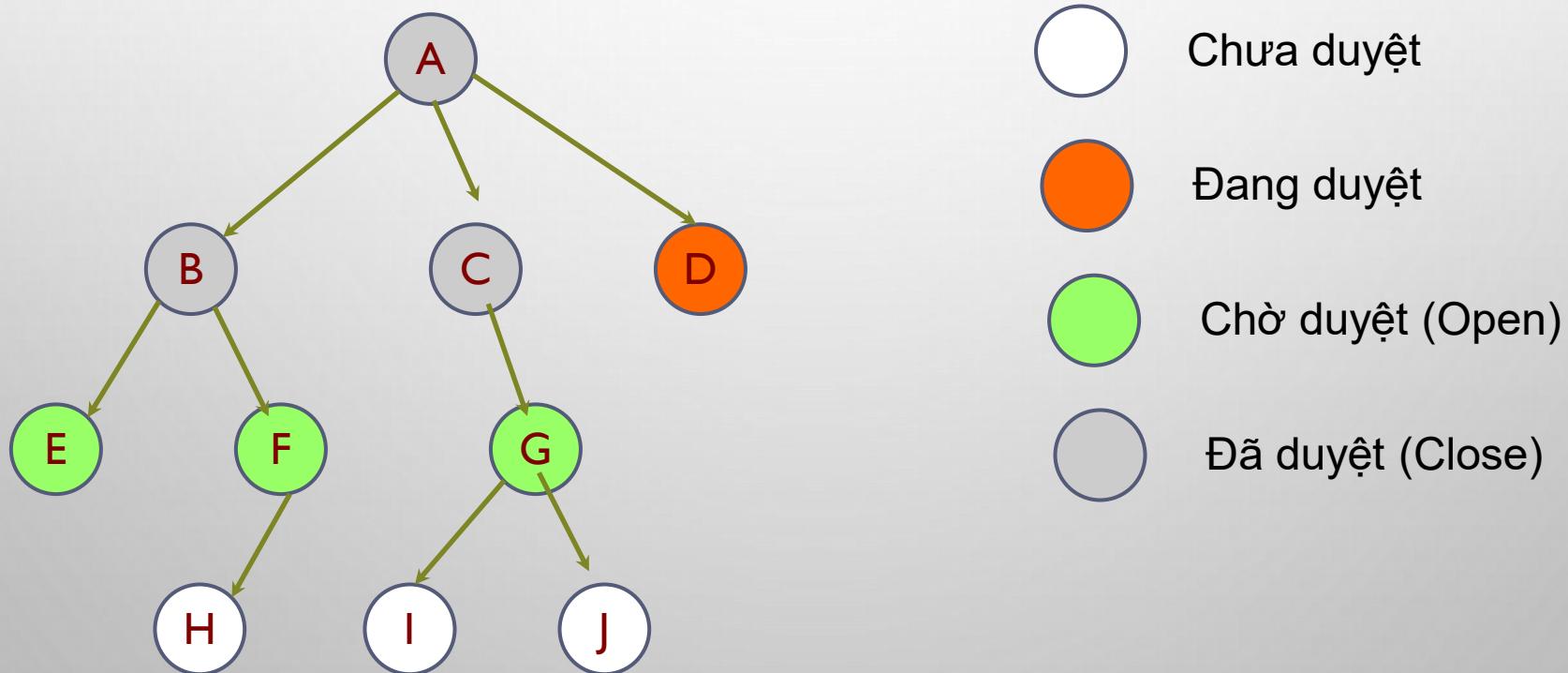
Tìm kiếm theo chiều rộng

- Tập trạng thái chờ duyệt “OPEN”: Queue
- Trạng thái nào được sinh ra trước thì được phát triển trước
- Duyệt hết các nút ở cùng độ sâu rồi mới duyệt tới các nút ở độ sâu tiếp theo



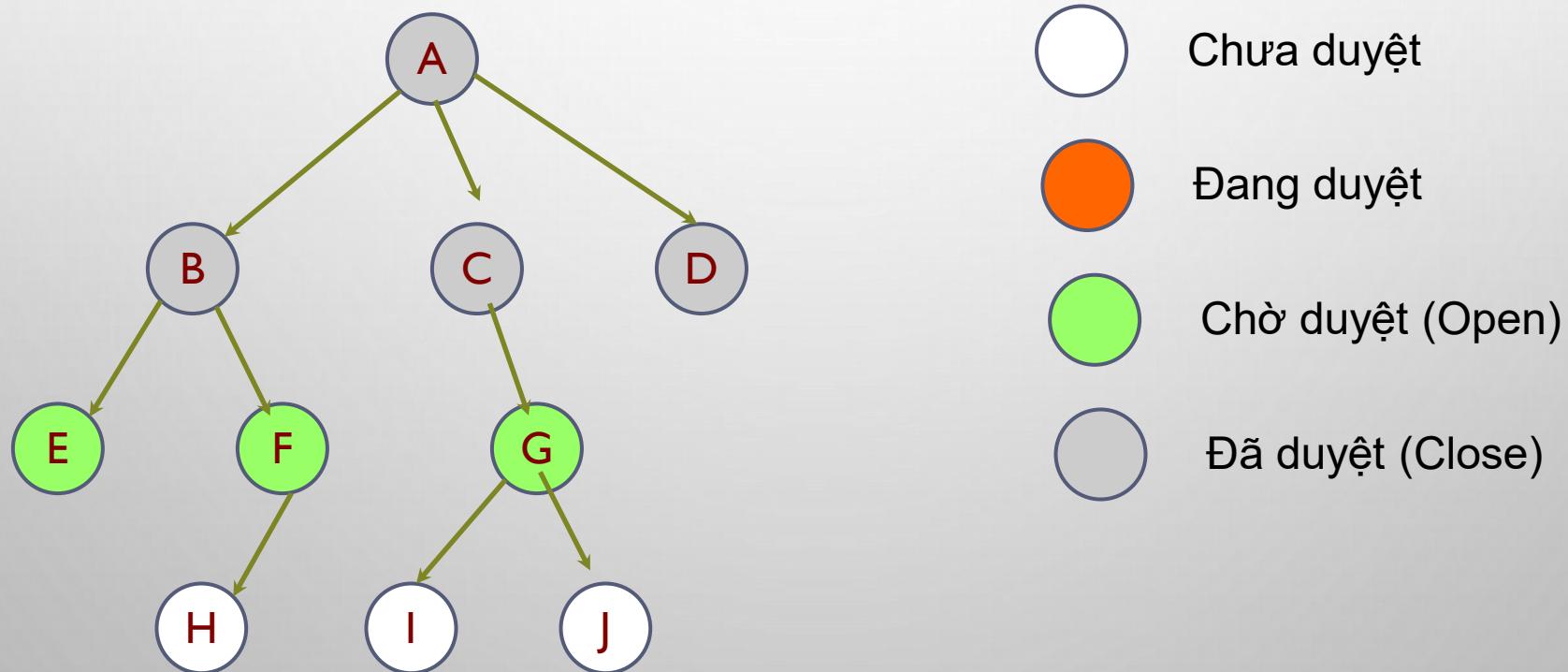
Tìm kiếm theo chiều rộng

- Tập trạng thái chờ duyệt “OPEN”: Queue
- Trạng thái nào được sinh ra trước thì được phát triển trước
- Duyệt hết các nút ở cùng độ sâu rồi mới duyệt tới các nút ở độ sâu tiếp theo



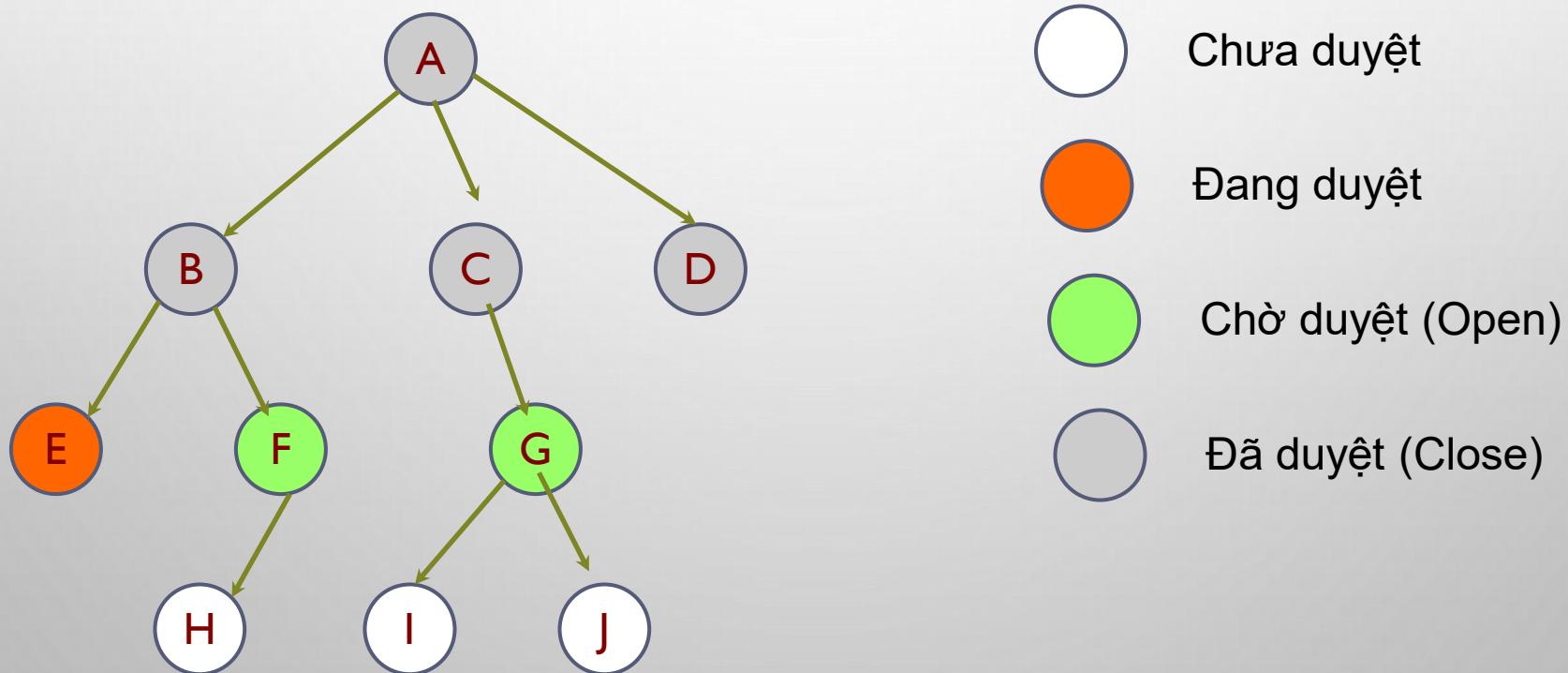
Tìm kiếm theo chiều rộng

- Tập trạng thái chờ duyệt “OPEN”: Queue
- Trạng thái nào được sinh ra trước thì được phát triển trước
- Duyệt hết các nút ở cùng độ sâu rồi mới duyệt tới các nút ở độ sâu tiếp theo



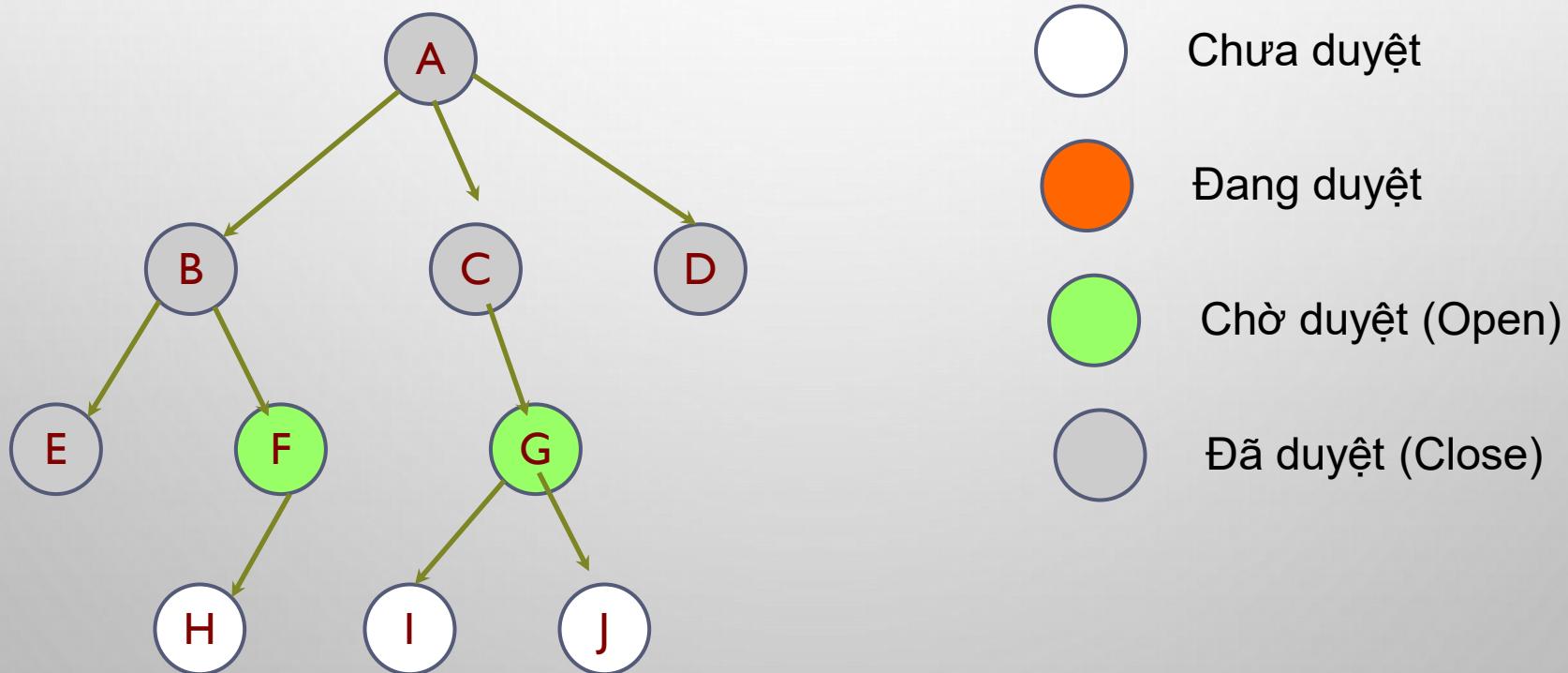
Tìm kiếm theo chiều rộng

- Tập trạng thái chờ duyệt “OPEN”: Queue
- Trạng thái nào được sinh ra trước thì được phát triển trước
- Duyệt hết các nút ở cùng độ sâu rồi mới duyệt tới các nút ở độ sâu tiếp theo



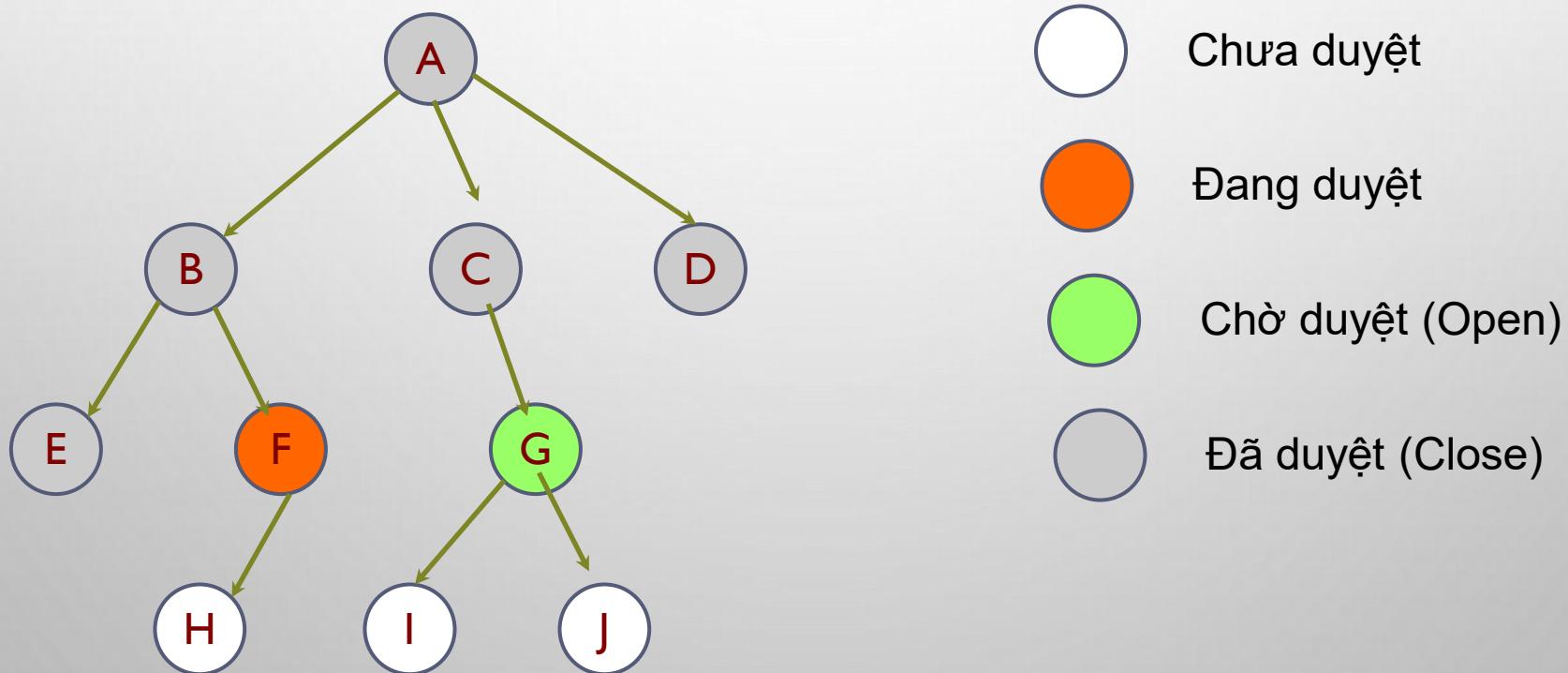
Tìm kiếm theo chiều rộng

- Tập trạng thái chờ duyệt “OPEN”: Queue
- Trạng thái nào được sinh ra trước thì được phát triển trước
- Duyệt hết các nút ở cùng độ sâu rồi mới duyệt tới các nút ở độ sâu tiếp theo



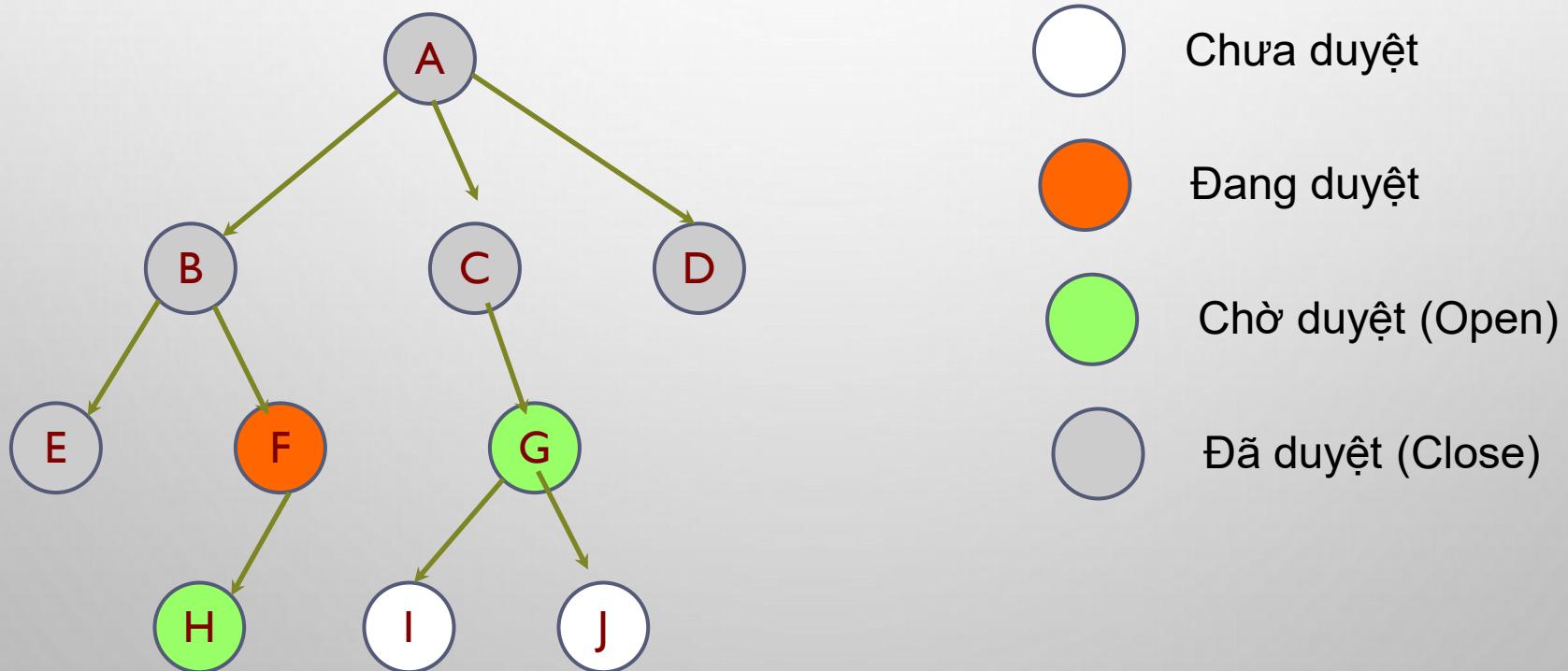
Tìm kiếm theo chiều rộng

- Tập trạng thái chờ duyệt “OPEN”: Queue
- Trạng thái nào được sinh ra trước thì được phát triển trước
- Duyệt hết các nút ở cùng độ sâu rồi mới duyệt tới các nút ở độ sâu tiếp theo



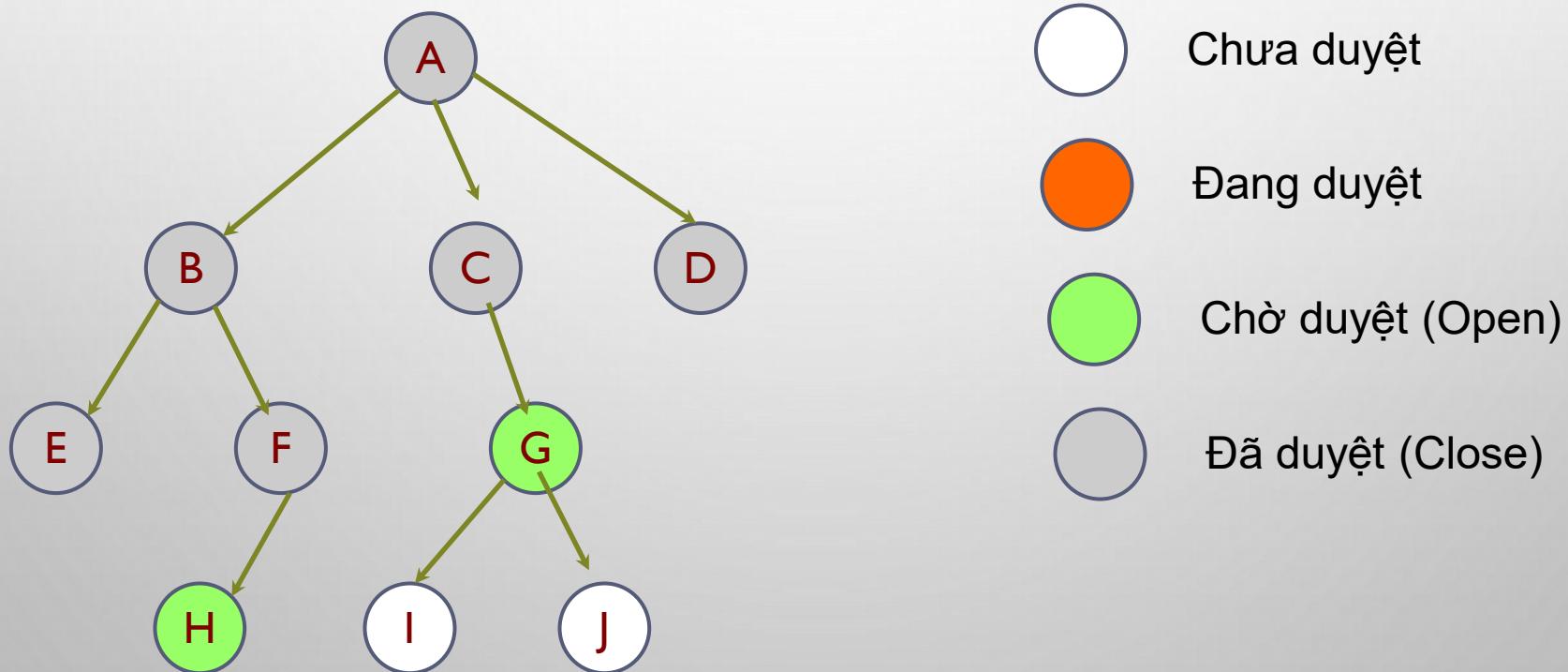
Tìm kiếm theo chiều rộng

- Tập trạng thái chờ duyệt “OPEN”: Queue
- Trạng thái nào được sinh ra trước thì được phát triển trước
- Duyệt hết các nút ở cùng độ sâu rồi mới duyệt tới các nút ở độ sâu tiếp theo



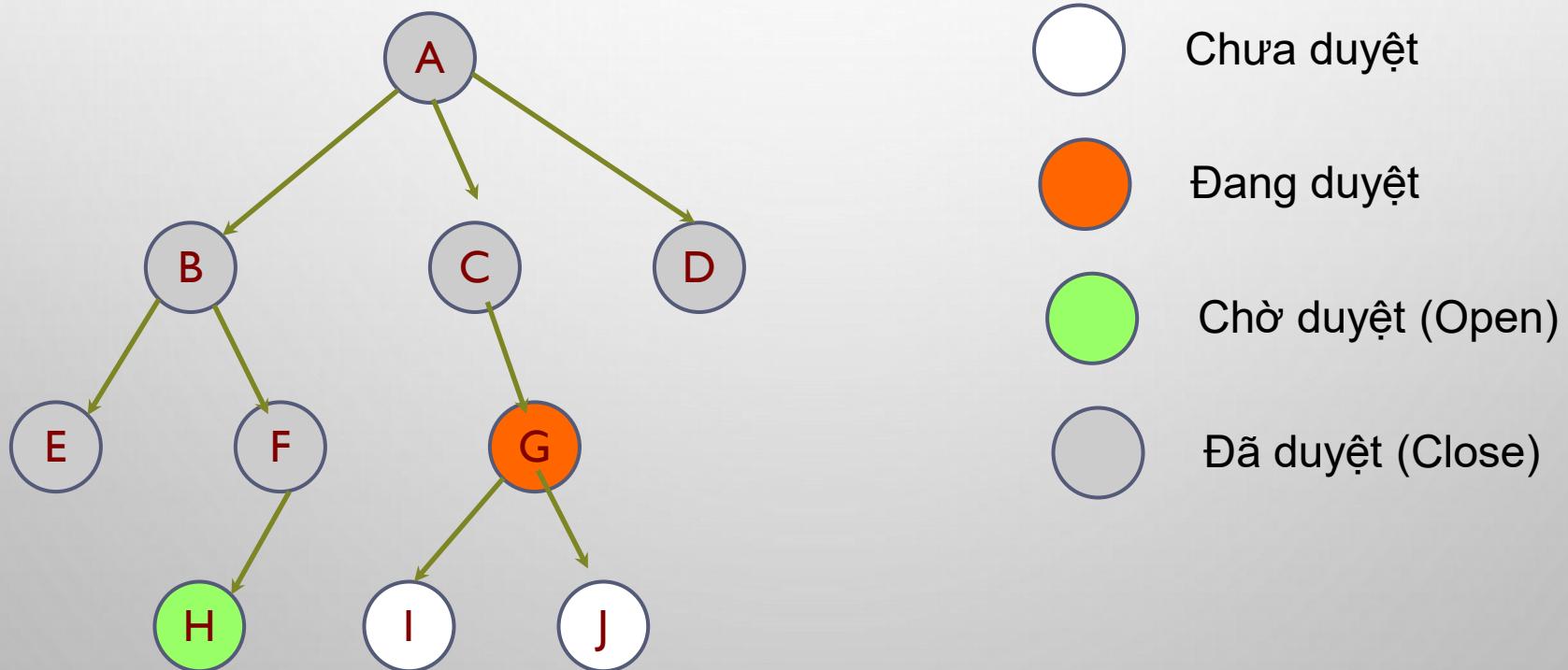
Tìm kiếm theo chiều rộng

- Tập trạng thái chờ duyệt “OPEN”: Queue
- Trạng thái nào được sinh ra trước thì được phát triển trước
- Duyệt hết các nút ở cùng độ sâu rồi mới duyệt tới các nút ở độ sâu tiếp theo



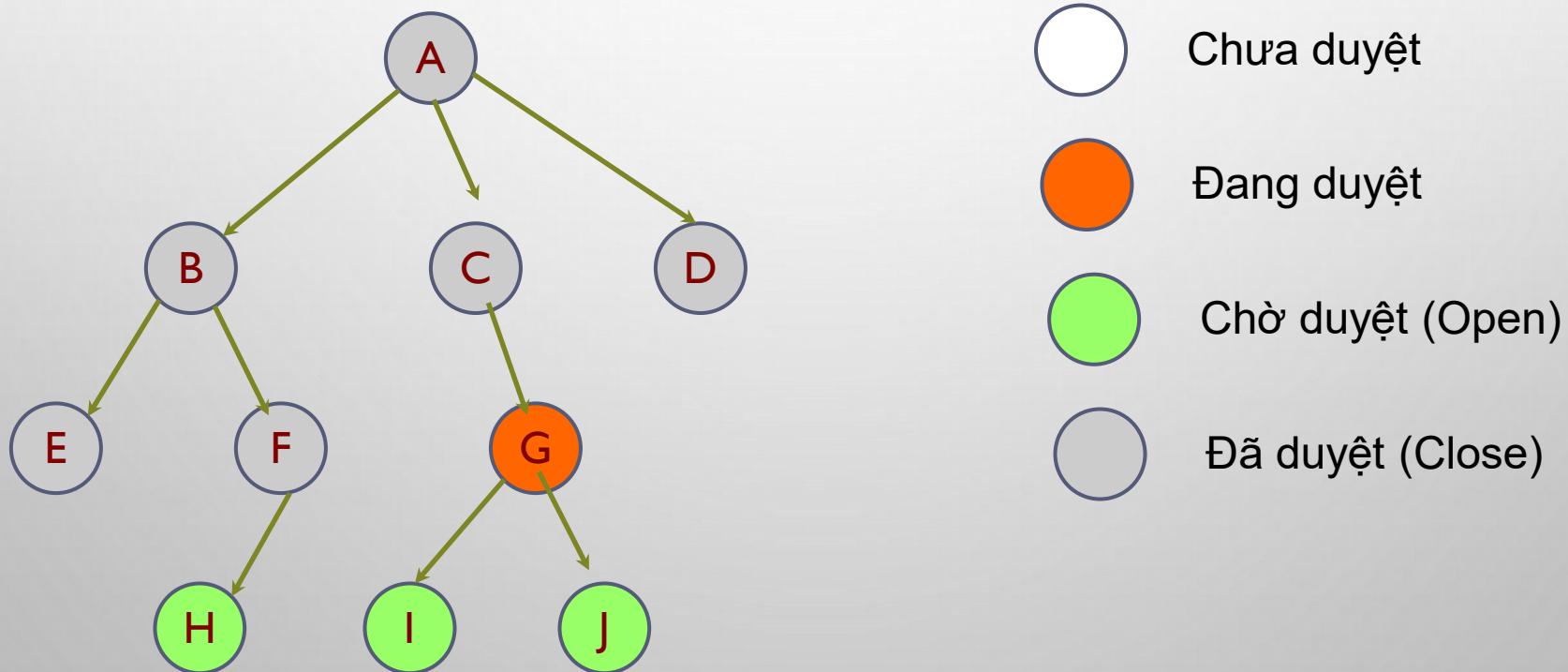
Tìm kiếm theo chiều rộng

- Tập trạng thái chờ duyệt “OPEN”: Queue
- Trạng thái nào được sinh ra trước thì được phát triển trước
- Duyệt hết các nút ở cùng độ sâu rồi mới duyệt tới các nút ở độ sâu tiếp theo



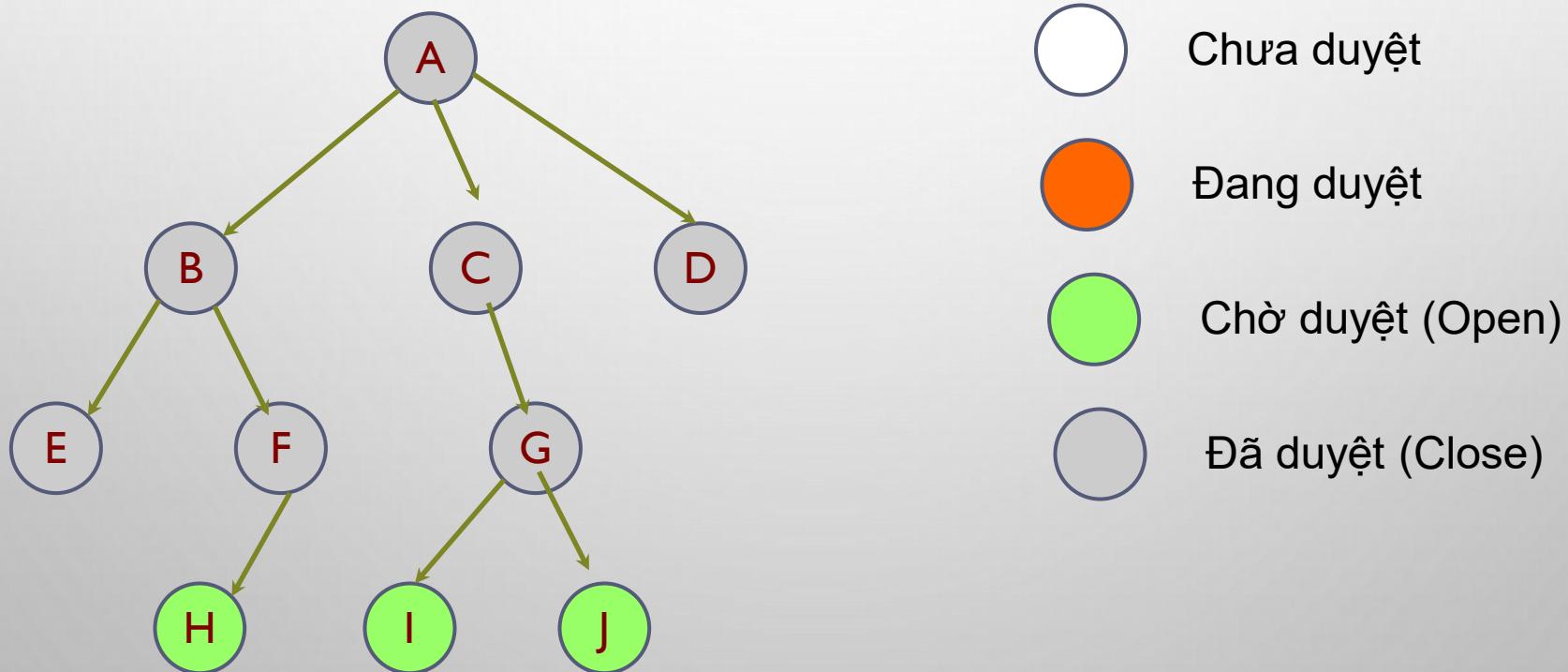
Tìm kiếm theo chiều rộng

- Tập trạng thái chờ duyệt “OPEN”: Queue
- Trạng thái nào được sinh ra trước thì được phát triển trước
- Duyệt hết các nút ở cùng độ sâu rồi mới duyệt tới các nút ở độ sâu tiếp theo



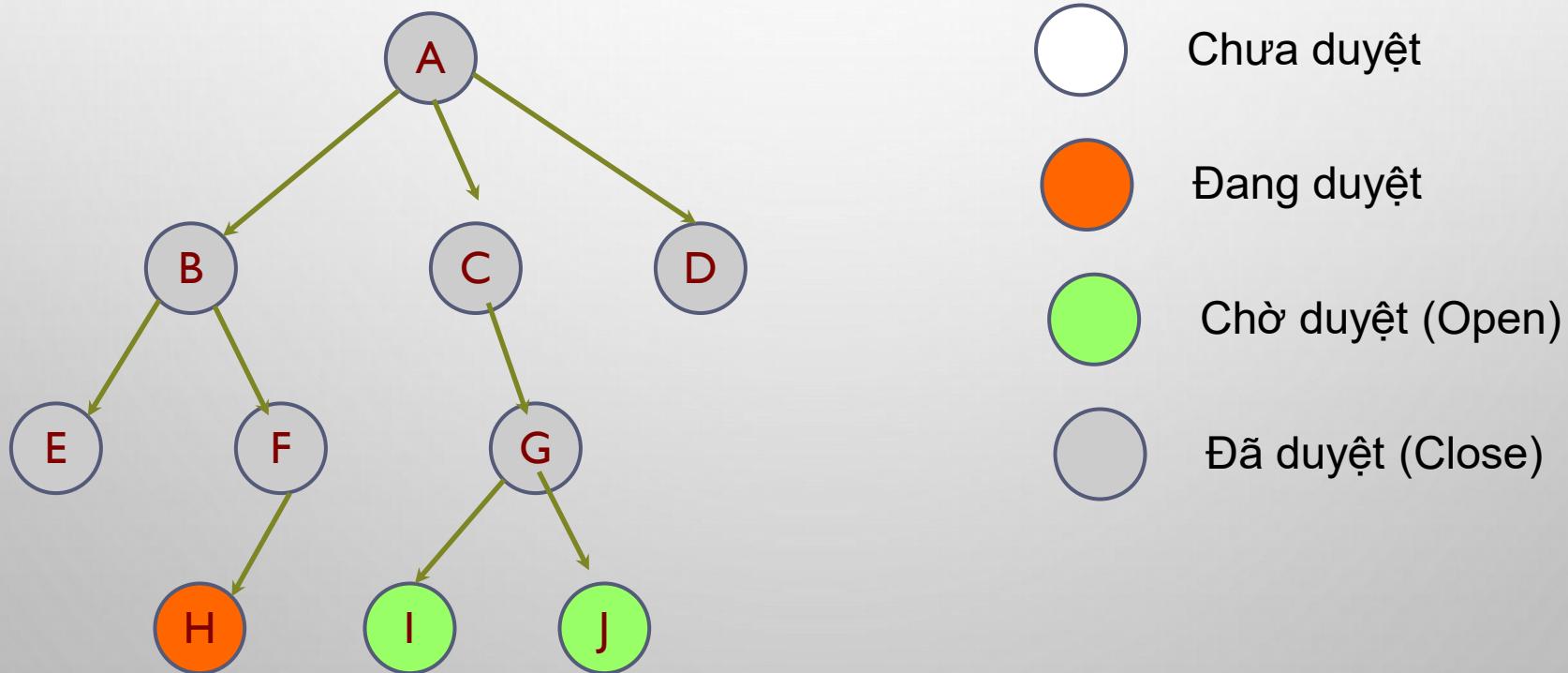
Tìm kiếm theo chiều rộng

- Tập trạng thái chờ duyệt “OPEN”: Queue
- Trạng thái nào được sinh ra trước thì được phát triển trước
- Duyệt hết các nút ở cùng độ sâu rồi mới duyệt tới các nút ở độ sâu tiếp theo



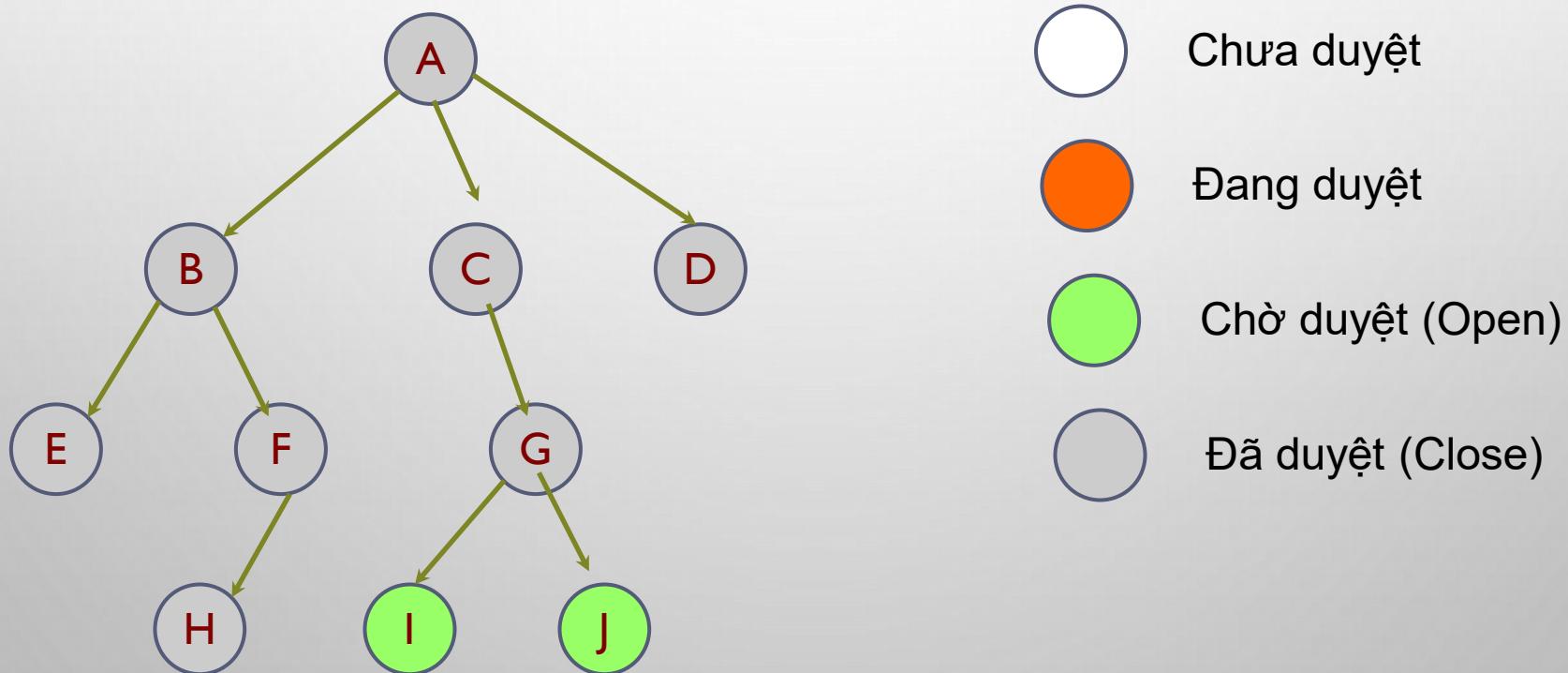
Tìm kiếm theo chiều rộng

- Tập trạng thái chờ duyệt “OPEN”: Queue
- Trạng thái nào được sinh ra trước thì được phát triển trước
- Duyệt hết các nút ở cùng độ sâu rồi mới duyệt tới các nút ở độ sâu tiếp theo



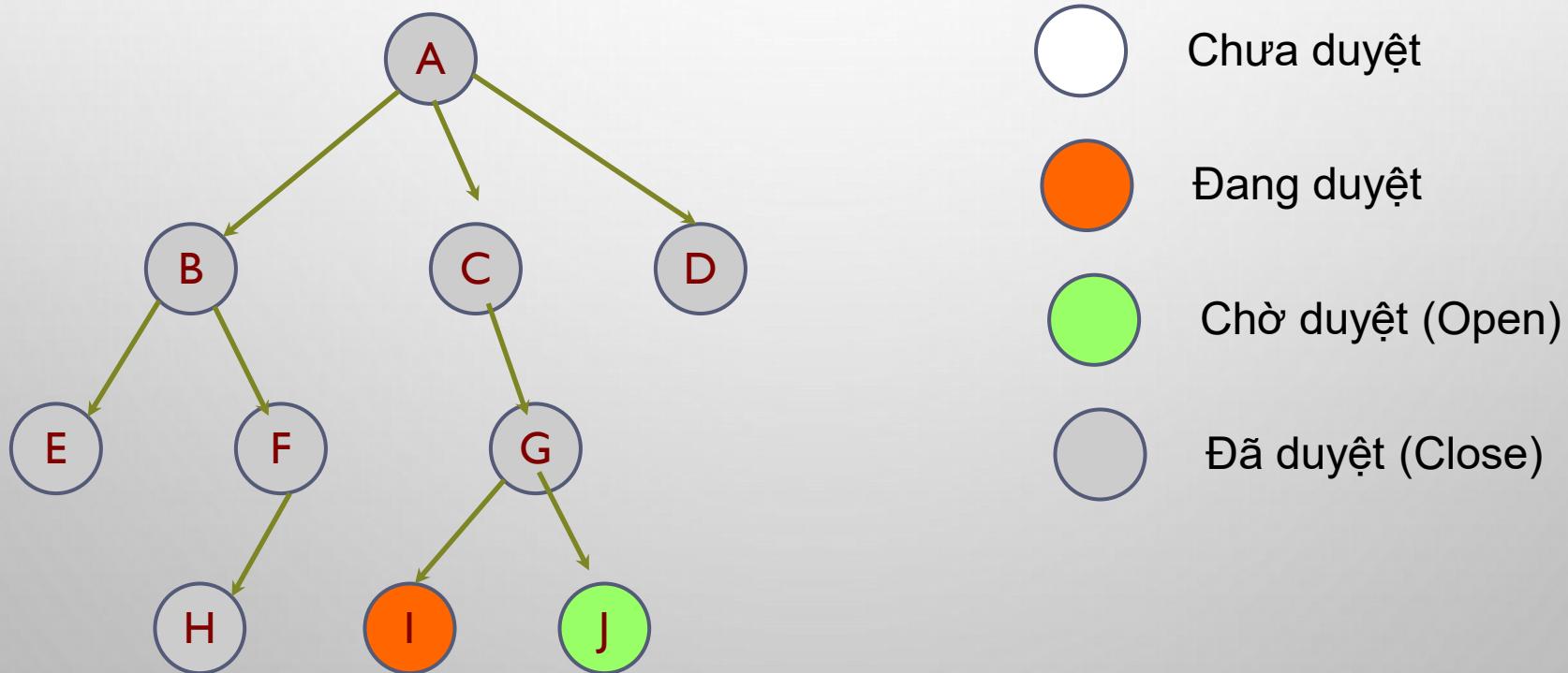
Tìm kiếm theo chiều rộng

- Tập trạng thái chờ duyệt “OPEN”: Queue
- Trạng thái nào được sinh ra trước thì được phát triển trước
- Duyệt hết các nút ở cùng độ sâu rồi mới duyệt tới các nút ở độ sâu tiếp theo



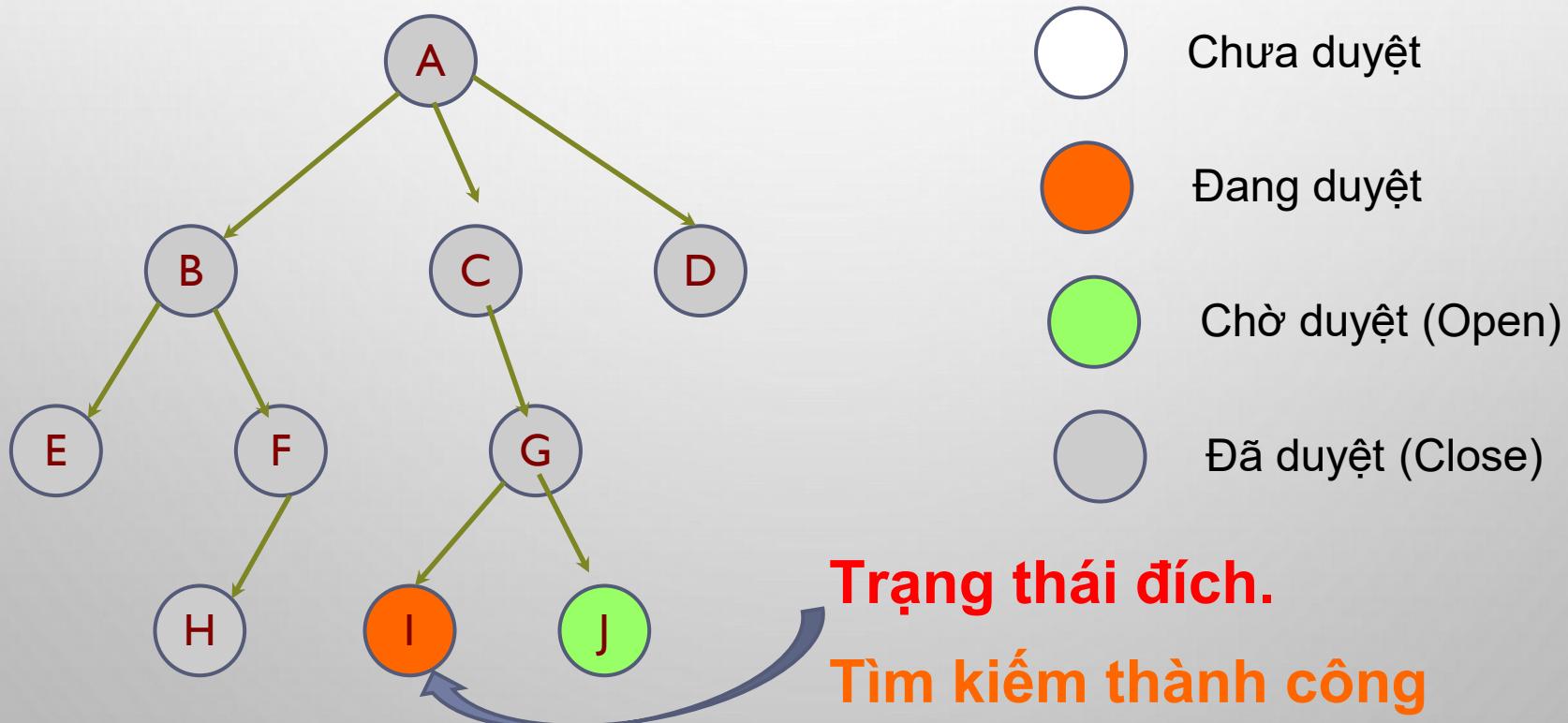
Tìm kiếm theo chiều rộng

- Tập trạng thái chờ duyệt “OPEN”: Queue
- Trạng thái nào được sinh ra trước thì được phát triển trước
- Duyệt hết các nút ở cùng độ sâu rồi mới duyệt tới các nút ở độ sâu tiếp theo



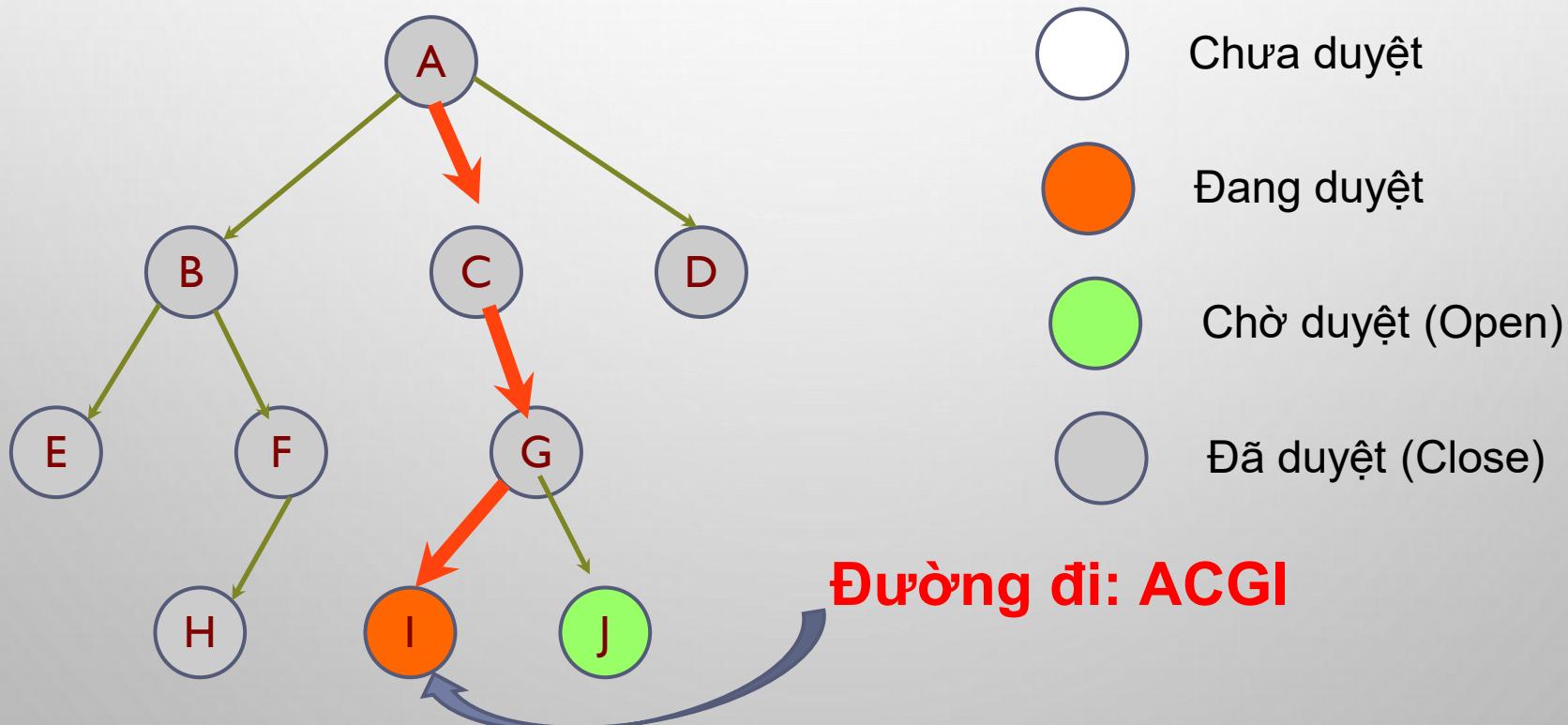
Tìm kiếm theo chiều rộng

- Tập trạng thái chờ duyệt “OPEN”: Queue
- Trạng thái nào được sinh ra trước thì được phát triển trước
- Duyệt hết các nút ở cùng độ sâu rồi mới duyệt tới các nút ở độ sâu tiếp theo



Tìm kiếm theo chiều rộng

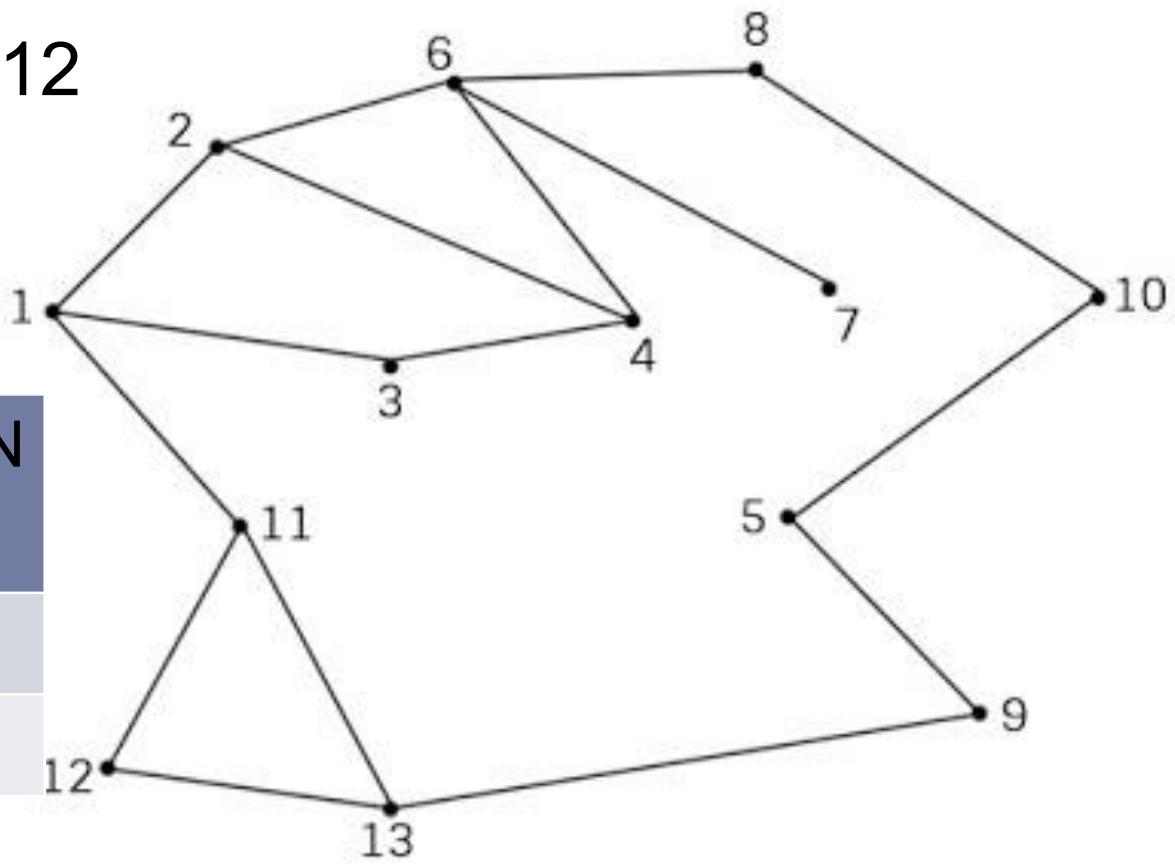
- Tập trạng thái chờ duyệt “OPEN”: Queue
- Trạng thái nào được sinh ra trước thì được phát triển trước
- Duyệt hết các nút ở cùng độ sâu rồi mới duyệt tới các nút ở độ sâu tiếp theo



Tìm kiếm theo chiều rộng

- Bài tập: Tìm đường đi và trình tự duyệt các đỉnh
 - Trạng thái đầu: 1
 - Trạng thái đích: 12

Bước lắp	u	kè(u)	OPEN
0	1	2,3,11	rỗng
...			



Tìm kiếm theo chiều rộng

```
procedure Breadth_First_Search
```

```
begin
```

```
1. Khởi tạo danh sách OPEN = {trạng thái ban đầu};
```

```
2. while true do
```

```
2.1 if (Open rỗng) then
```

```
{thông báo tìm kiếm thất bại; stop};
```

```
2.2 Loại trạng thái u ở đầu danh sách OPEN;
```

```
2.3 if u là trạng thái kết thúc then
```

```
{thông báo tìm kiếm thành công; stop};
```

```
2.4 Thêm các đỉnh trạng thái kè với u vào cuối danh sách OPEN;
```

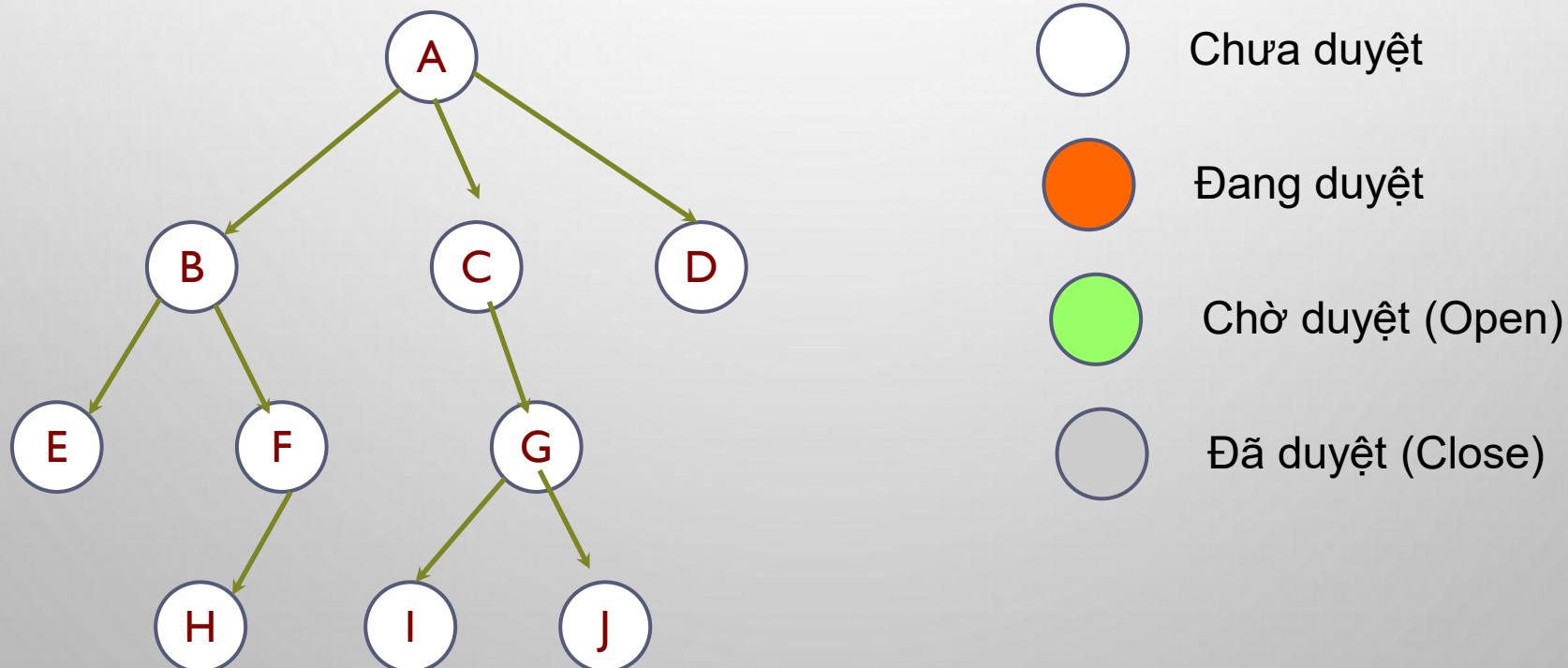
```
end
```

Tìm kiếm theo chiều rộng*

- Giả sử:
 - Cây tìm kiếm có nhân tố nhánh (số nút con tối đa của 1 đỉnh) là b
 - Lời giải của bài toán là đường đi có độ dài d
- Độ phức tạp thời gian:
 - Số nút duyệt $= 1 + b + b^2 + \dots + b^{d-1} + k = O(b^d)$
 - Trong đó k là số lượng nút ở độ sâu d tính đến nút trạng thái đích
- Độ phức tạp bộ nhớ:
 - Số nút lưu trong danh sách OPEN để chờ duyệt

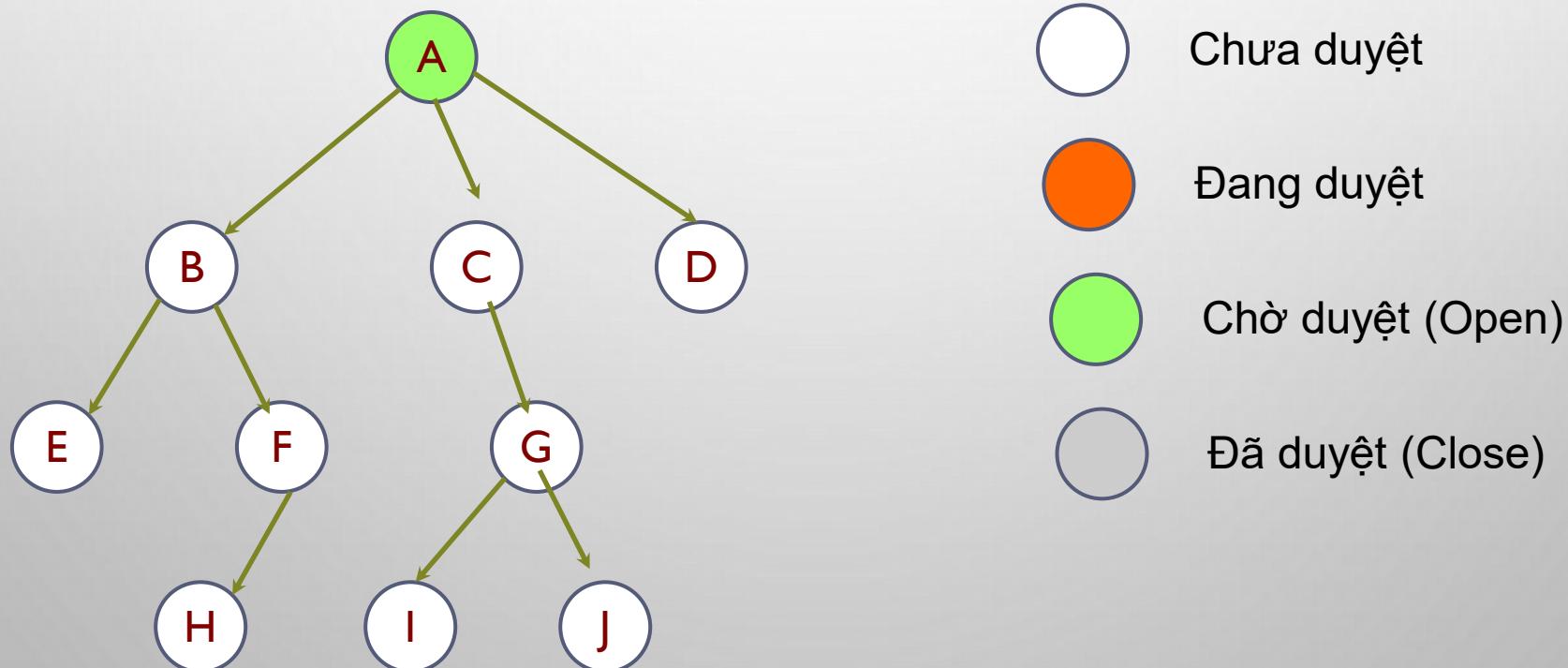
Tìm kiếm theo chiều sâu

- Tập trạng thái chờ duyệt “OPEN”: Stack
- Trạng thái sâu nhất (được đưa vào Open mới nhất) thì được phát triển trước
- Duyệt hết các nút con cháu, ... rồi mới đi lên duyệt các nút ở cùng độ sâu nếu chưa tìm thấy



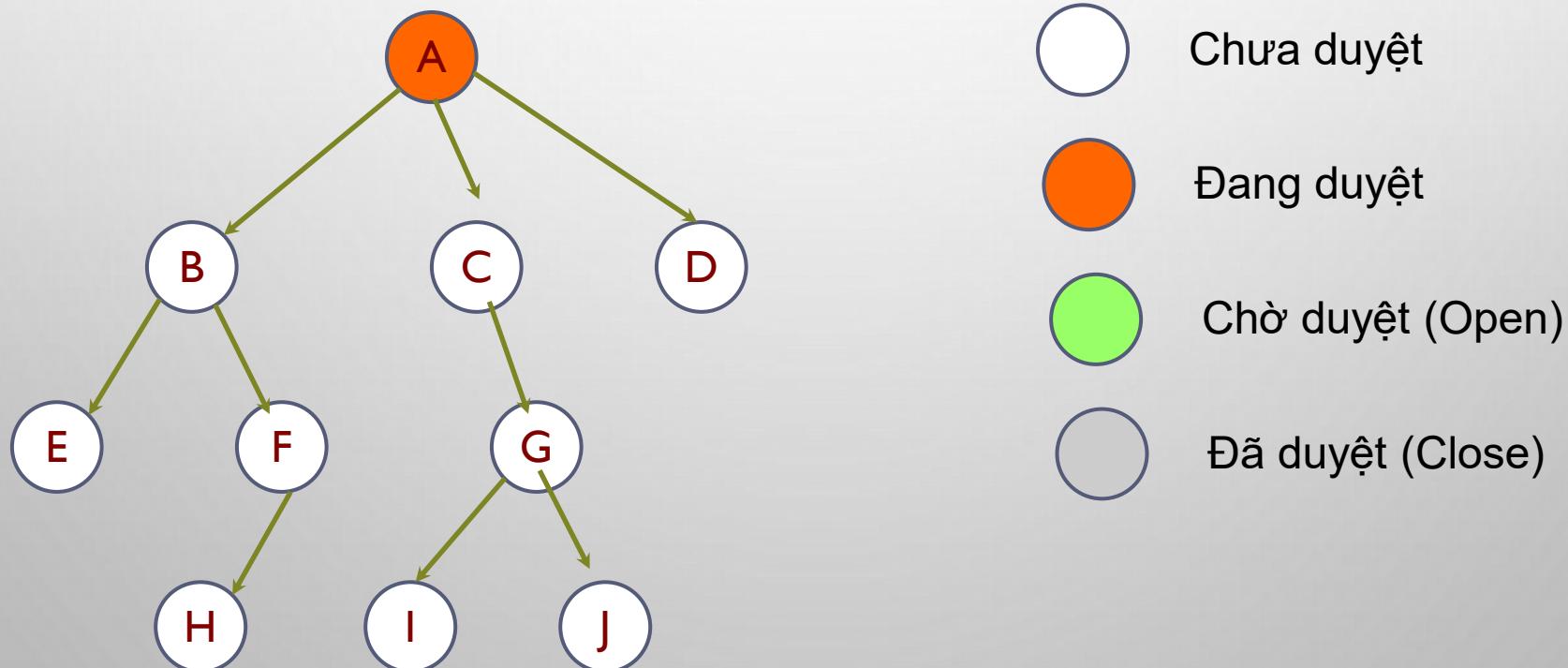
Tìm kiếm theo chiều sâu

- Tập trạng thái chờ duyệt “OPEN”: Stack
- Trạng thái sâu nhất (được đưa vào Open mới nhất) thì được phát triển trước
- Duyệt hết các nút con cháu, ... rồi mới đi lên duyệt các nút ở cùng độ sâu nếu chưa tìm thấy



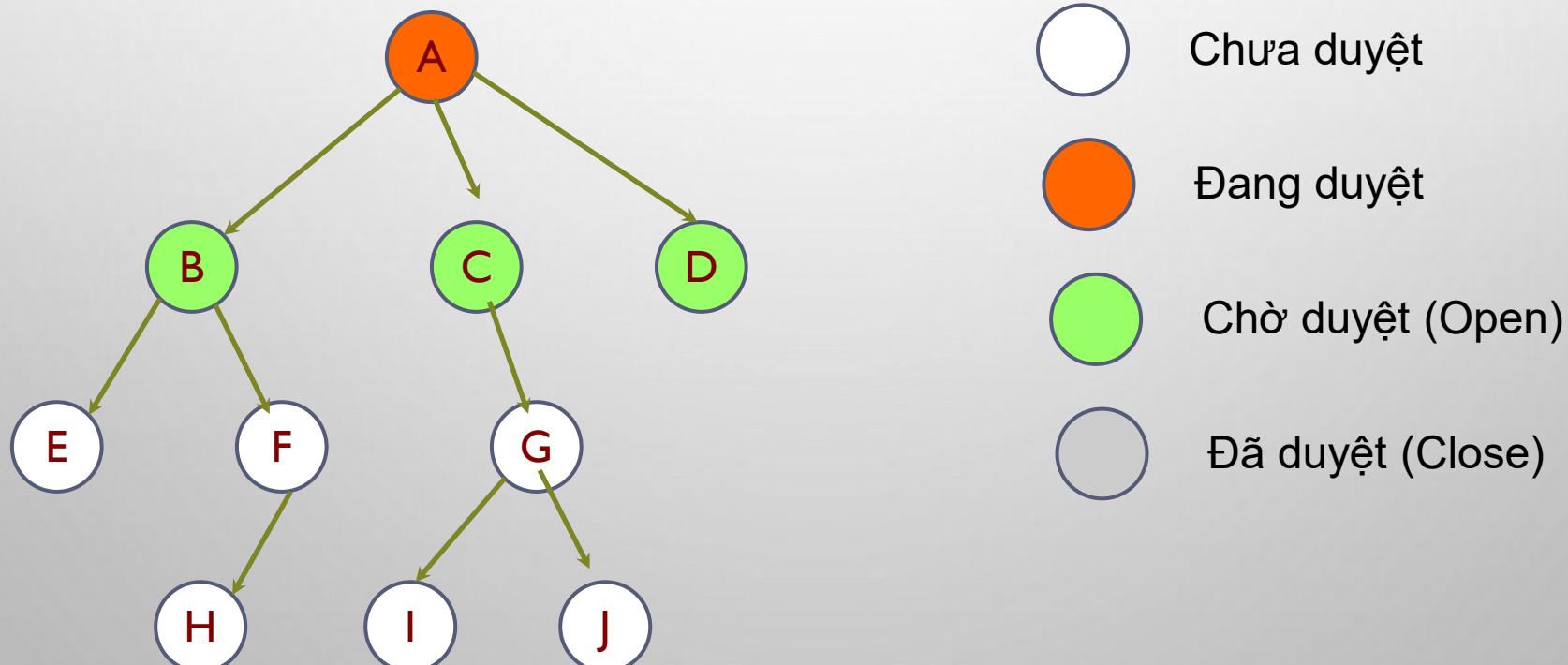
Tìm kiếm theo chiều sâu

- Tập trạng thái chờ duyệt “OPEN”: Stack
- Trạng thái sâu nhất (được đưa vào Open mới nhất) thì được phát triển trước
- Duyệt hết các nút con cháu, ... rồi mới đi lên duyệt các nút ở cùng độ sâu nếu chưa tìm thấy



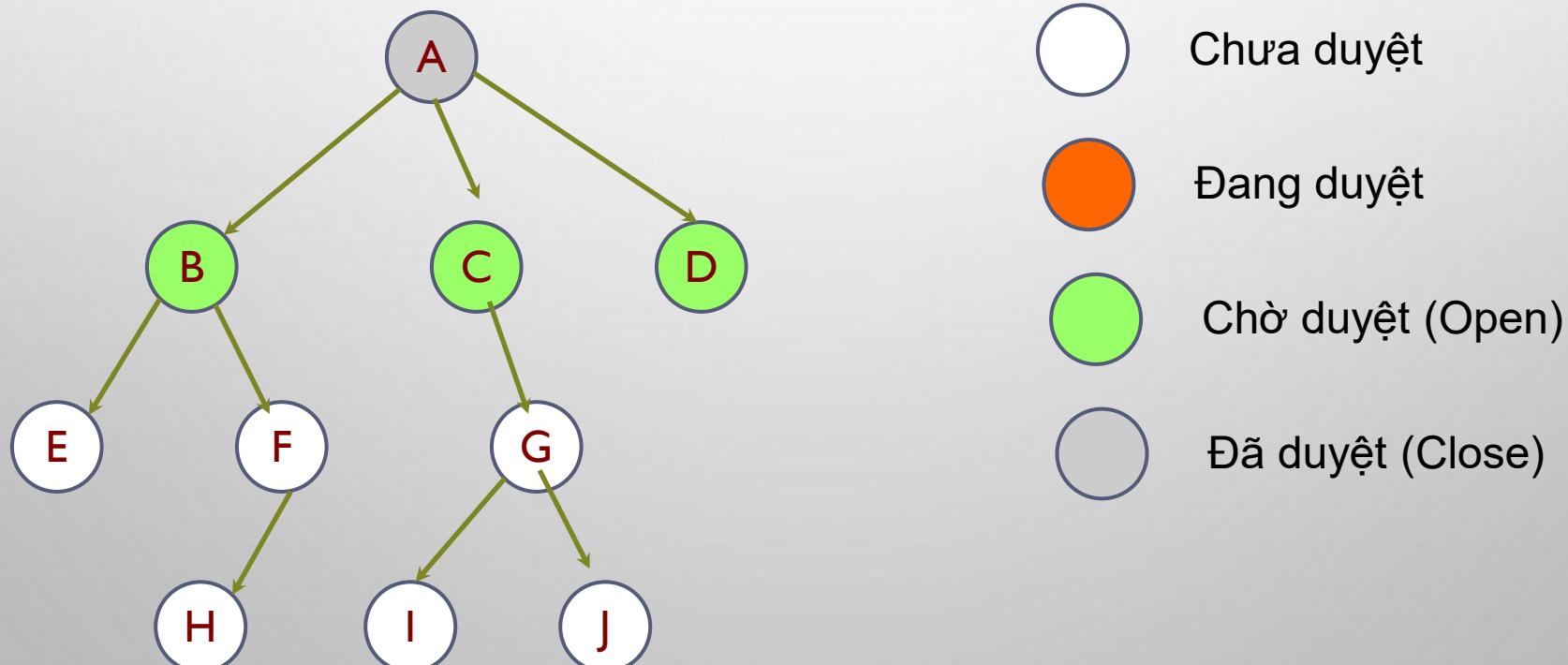
Tìm kiếm theo chiều sâu

- Tập trạng thái chờ duyệt “OPEN”: Stack
- Trạng thái sâu nhất (được đưa vào Open mới nhất) thì được phát triển trước
- Duyệt hết các nút con cháu, ... rồi mới đi lên duyệt các nút ở cùng độ sâu nếu chưa tìm thấy



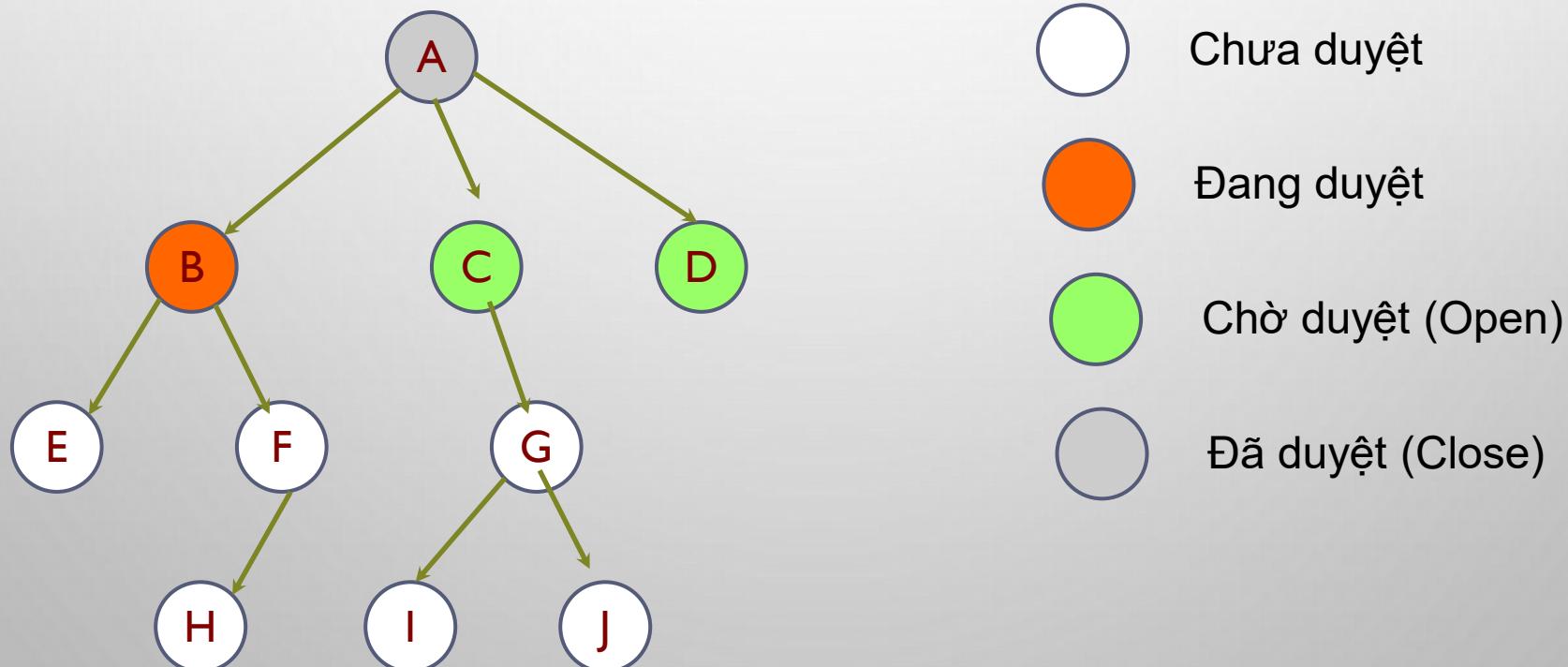
Tìm kiếm theo chiều sâu

- Tập trạng thái chờ duyệt “OPEN”: Stack
- Trạng thái sâu nhất (được đưa vào Open mới nhất) thì được phát triển trước
- Duyệt hết các nút con cháu, ... rồi mới đi lên duyệt các nút ở cùng độ sâu nếu chưa tìm thấy



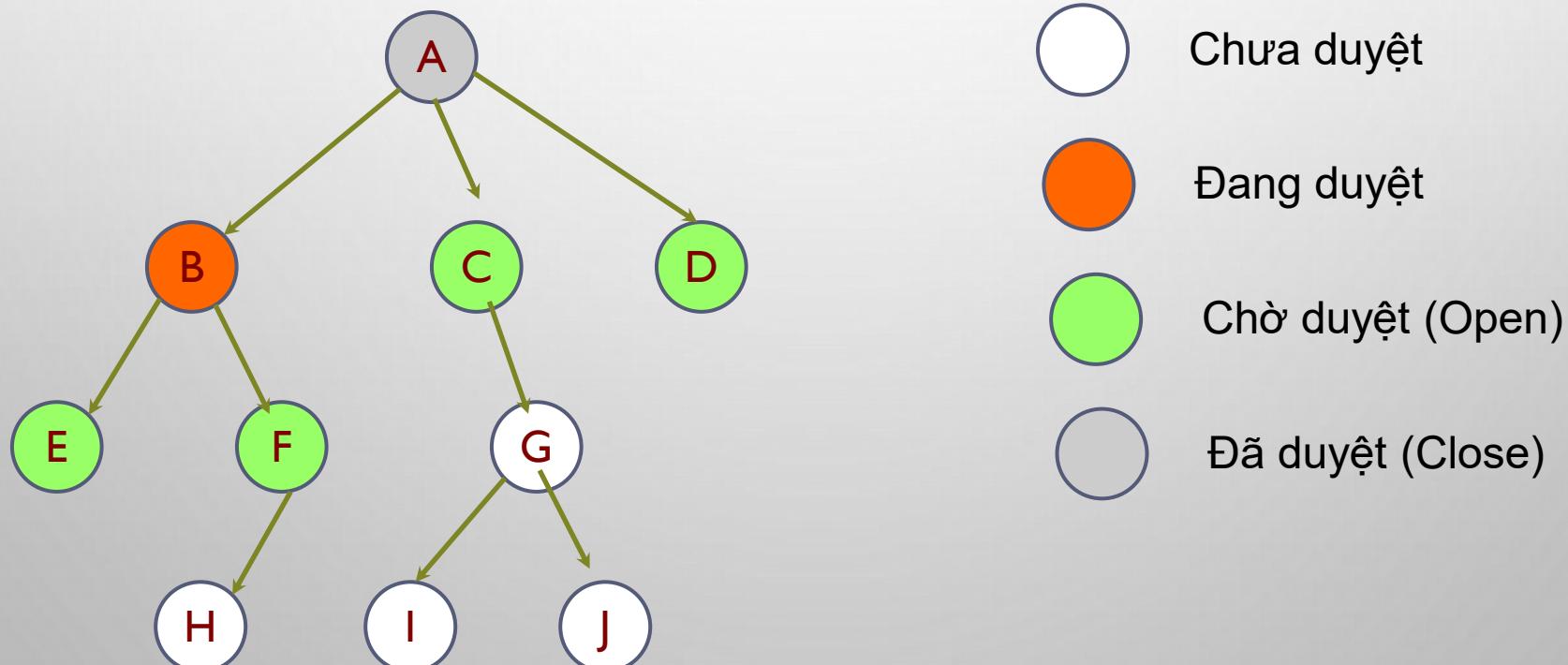
Tìm kiếm theo chiều sâu

- Tập trạng thái chờ duyệt “OPEN”: Stack
- Trạng thái sâu nhất (được đưa vào Open mới nhất) thì được phát triển trước
- Duyệt hết các nút con cháu, ... rồi mới đi lên duyệt các nút ở cùng độ sâu nếu chưa tìm thấy



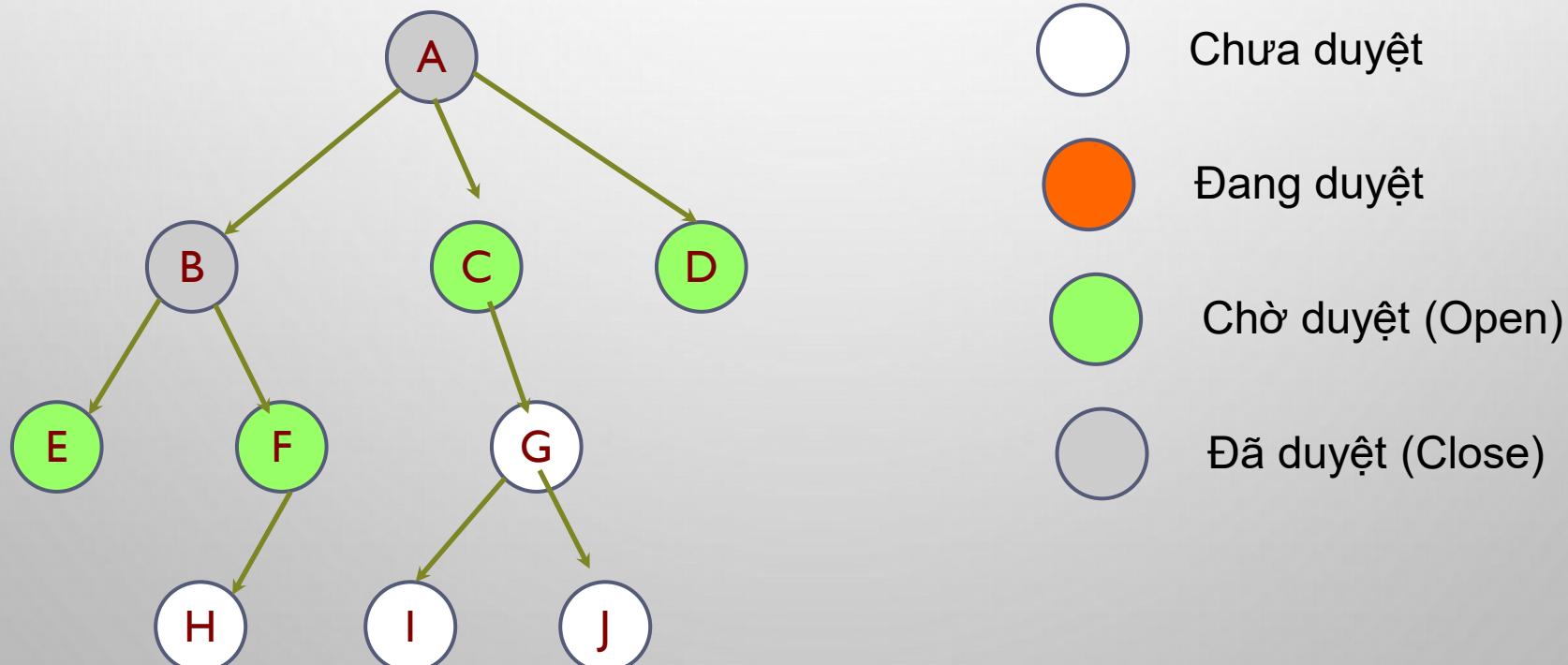
Tìm kiếm theo chiều sâu

- Tập trạng thái chờ duyệt “OPEN”: Stack
- Trạng thái sâu nhất (được đưa vào Open mới nhất) thì được phát triển trước
- Duyệt hết các nút con cháu, ... rồi mới đi lên duyệt các nút ở cùng độ sâu nếu chưa tìm thấy



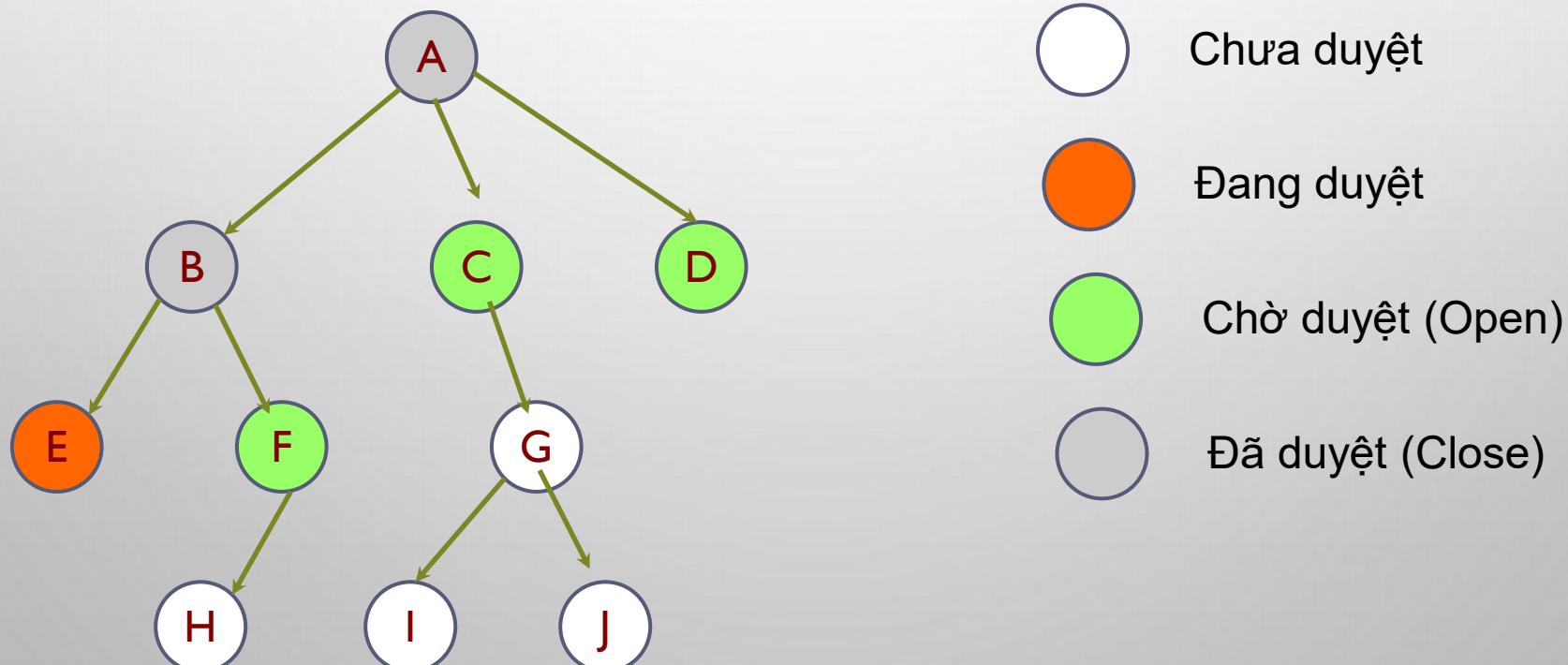
Tìm kiếm theo chiều sâu

- Tập trạng thái chờ duyệt “OPEN”: Stack
- Trạng thái sâu nhất (được đưa vào Open mới nhất) thì được phát triển trước
- Duyệt hết các nút con cháu, ... rồi mới đi lên duyệt các nút ở cùng độ sâu nếu chưa tìm thấy



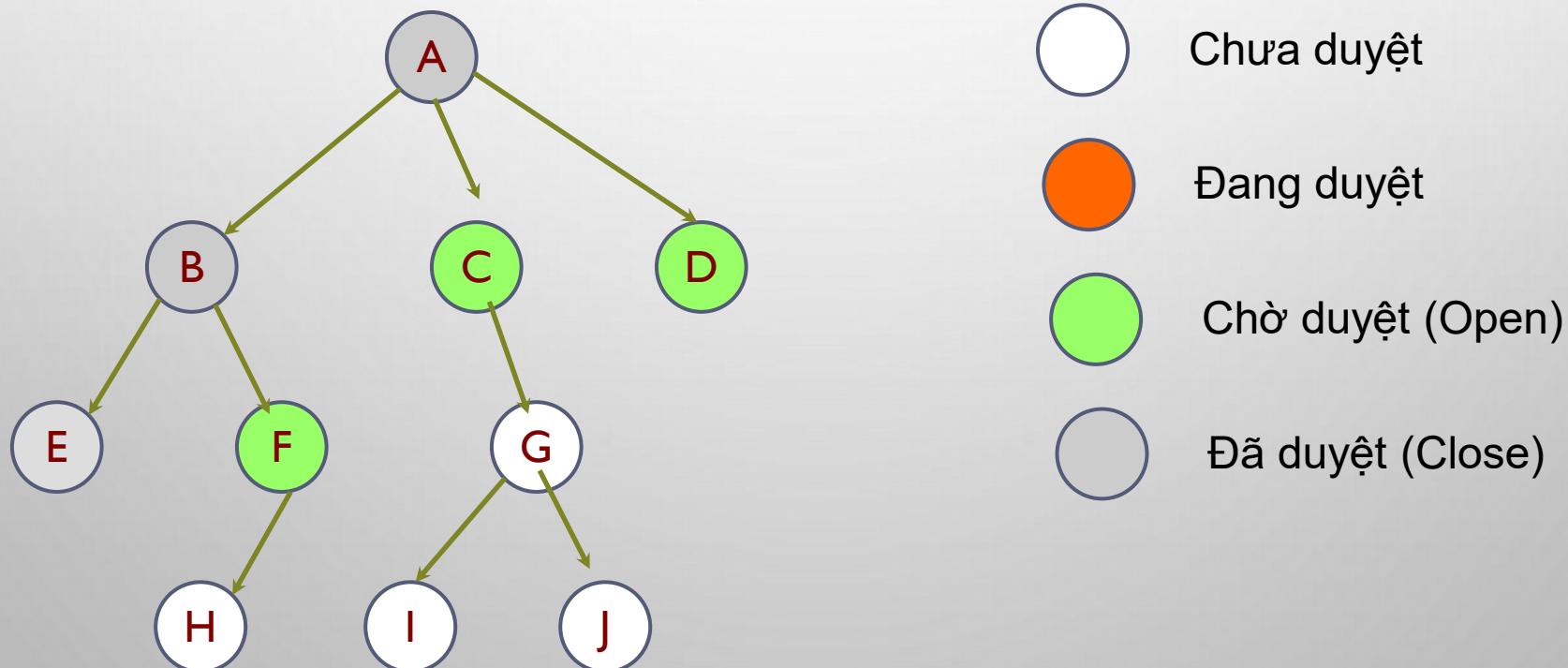
Tìm kiếm theo chiều sâu

- Tập trạng thái chờ duyệt “OPEN”: Stack
- Trạng thái sâu nhất (được đưa vào Open mới nhất) thì được phát triển trước
- Duyệt hết các nút con cháu, ... rồi mới đi lên duyệt các nút ở cùng độ sâu nếu chưa tìm thấy



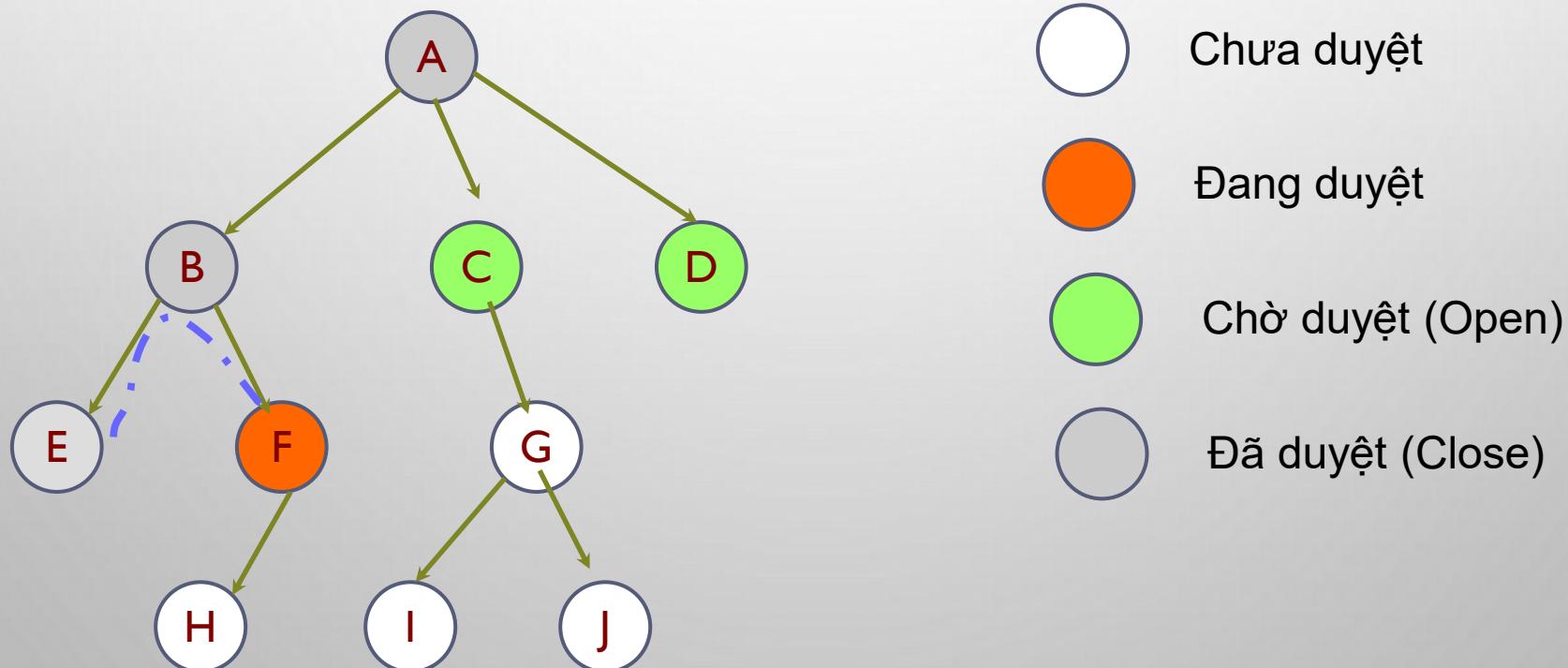
Tìm kiếm theo chiều sâu

- Tập trạng thái chờ duyệt “OPEN”: Stack
- Trạng thái sâu nhất (được đưa vào Open mới nhất) thì được phát triển trước
- Duyệt hết các nút con cháu, ... rồi mới đi lên duyệt các nút ở cùng độ sâu nếu chưa tìm thấy



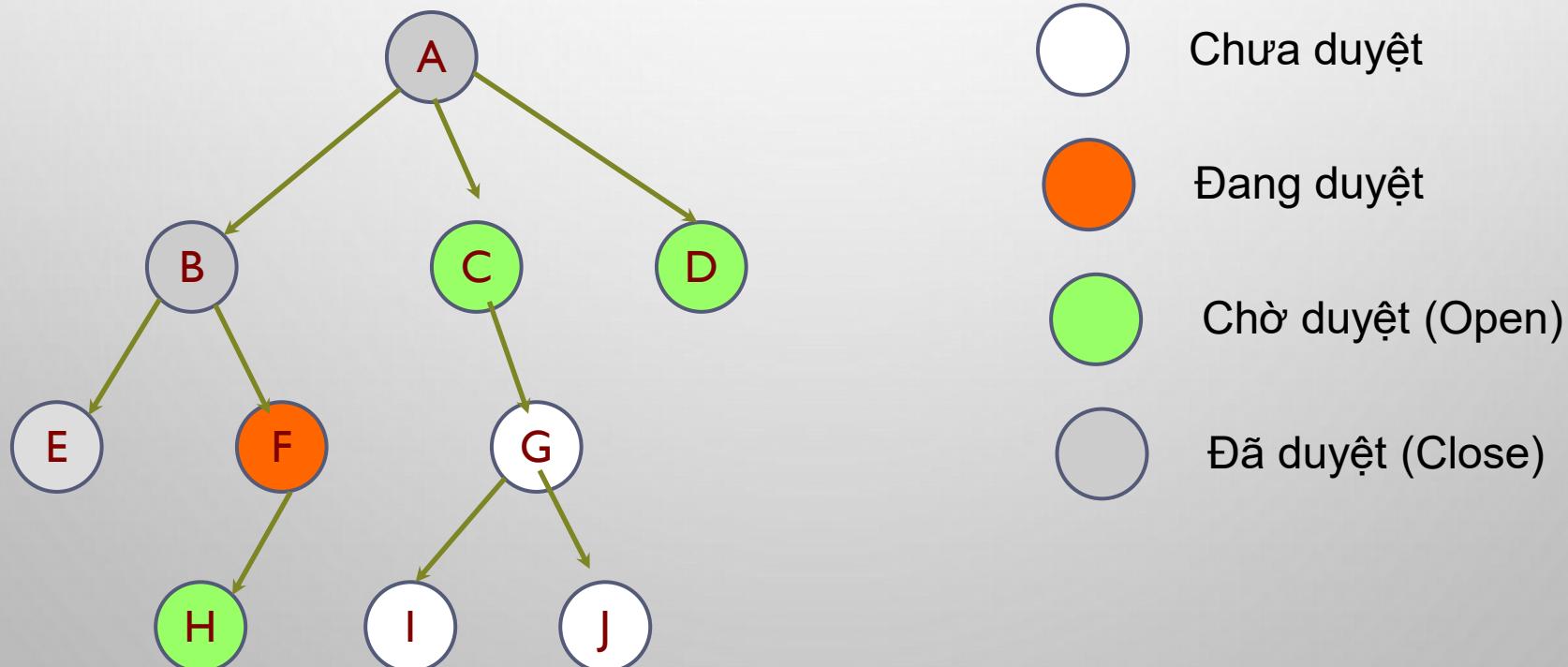
Tìm kiếm theo chiều sâu

- Tập trạng thái chờ duyệt “OPEN”: Stack
- Trạng thái sâu nhất (được đưa vào Open mới nhất) thì được phát triển trước
- Duyệt hết các nút con cháu, ... rồi mới đi lên duyệt các nút ở cùng độ sâu nếu chưa tìm thấy



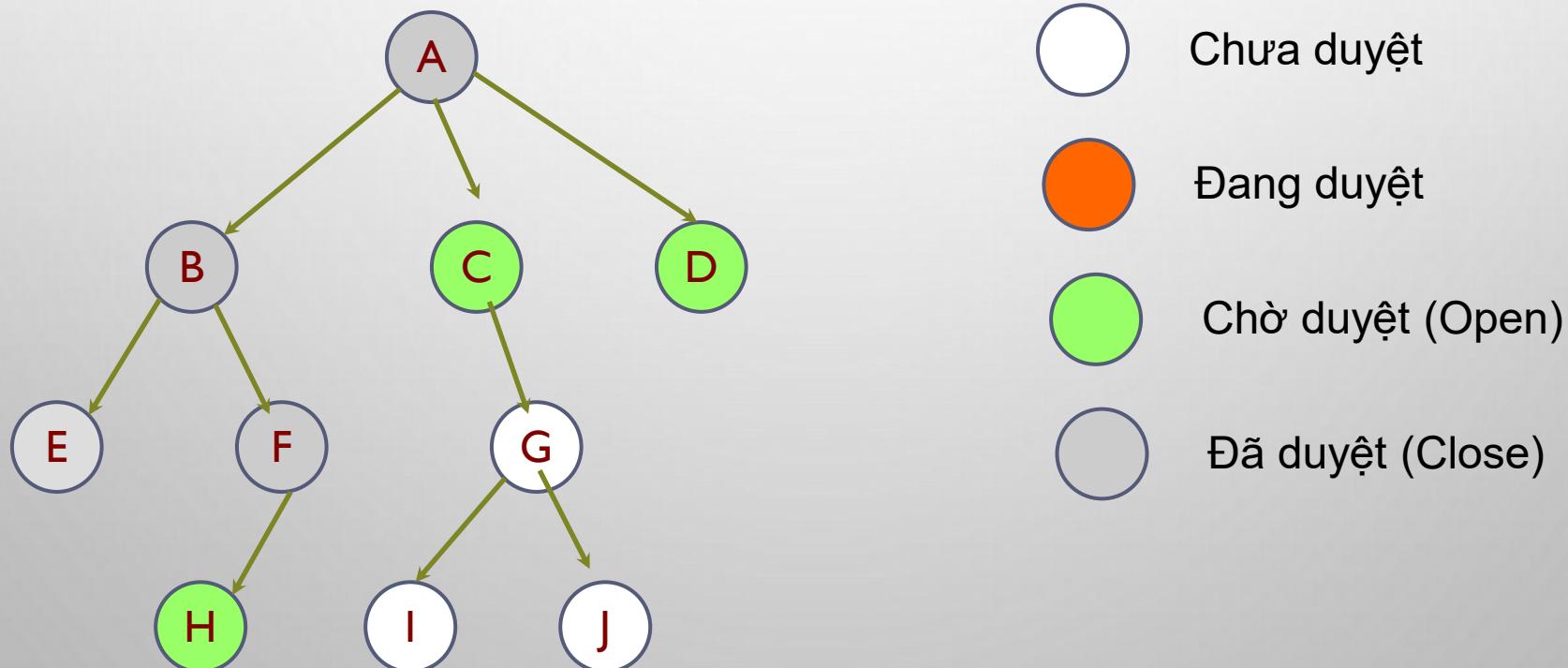
Tìm kiếm theo chiều sâu

- Tập trạng thái chờ duyệt “OPEN”: Stack
- Trạng thái sâu nhất (được đưa vào Open mới nhất) thì được phát triển trước
- Duyệt hết các nút con cháu, ... rồi mới đi lên duyệt các nút ở cùng độ sâu nếu chưa tìm thấy



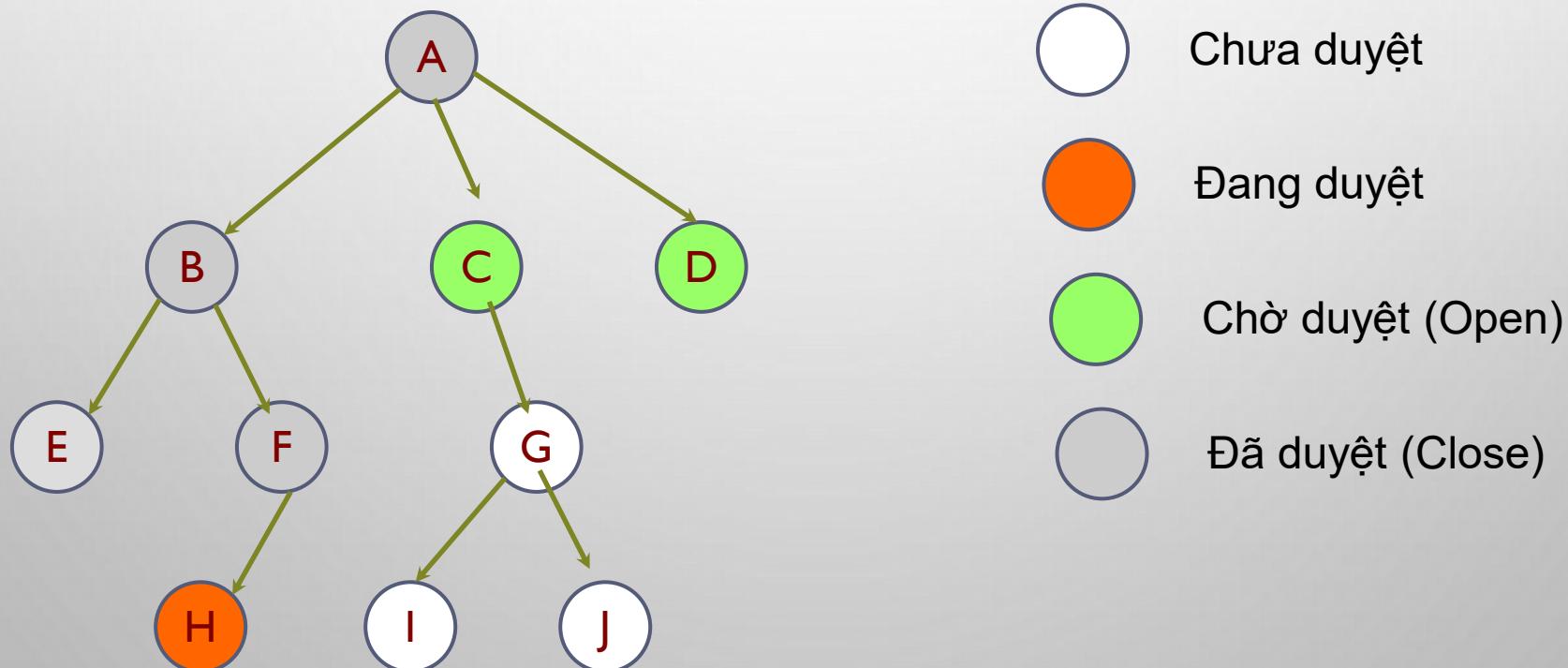
Tìm kiếm theo chiều sâu

- Tập trạng thái chờ duyệt “OPEN”: Stack
- Trạng thái sâu nhất (được đưa vào Open mới nhất) thì được phát triển trước
- Duyệt hết các nút con cháu, ... rồi mới đi lên duyệt các nút ở cùng độ sâu nếu chưa tìm thấy



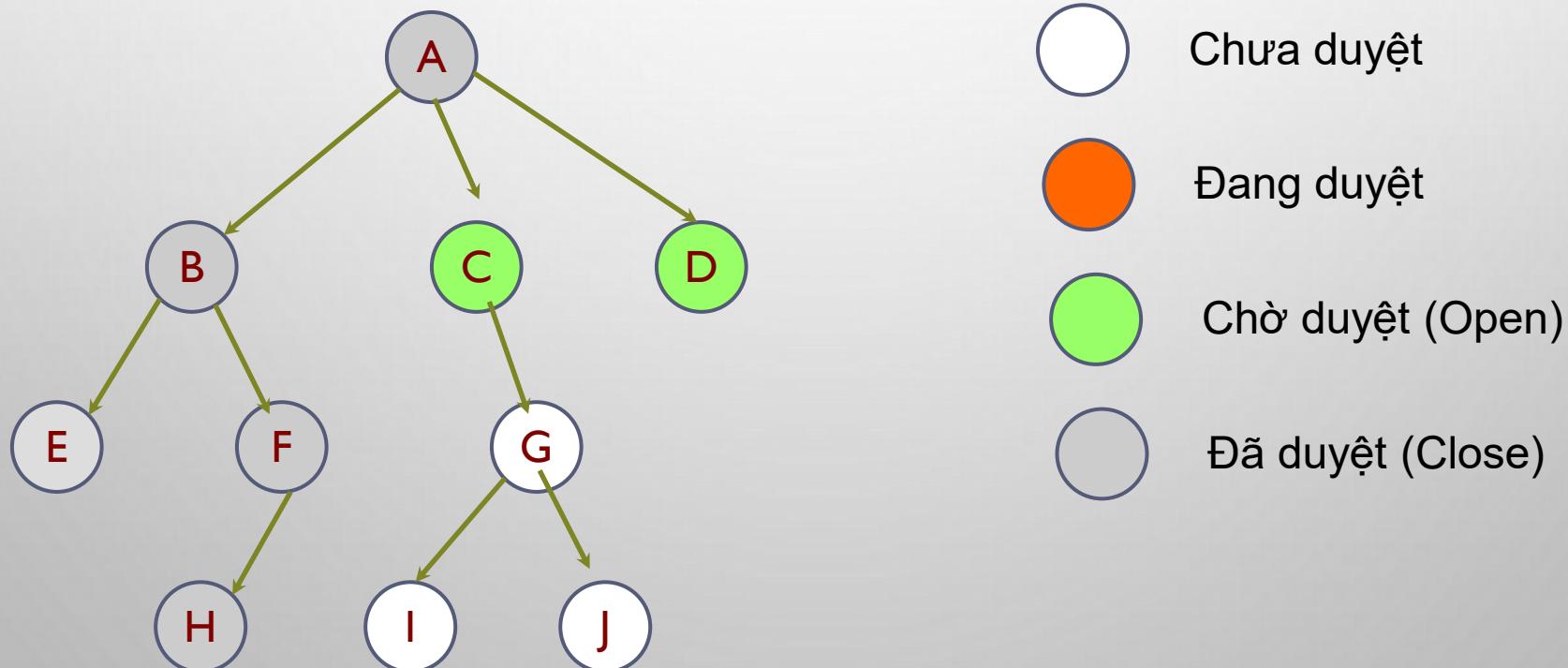
Tìm kiếm theo chiều sâu

- Tập trạng thái chờ duyệt “OPEN”: Stack
- Trạng thái sâu nhất (được đưa vào Open mới nhất) thì được phát triển trước
- Duyệt hết các nút con cháu, ... rồi mới đi lên duyệt các nút ở cùng độ sâu nếu chưa tìm thấy



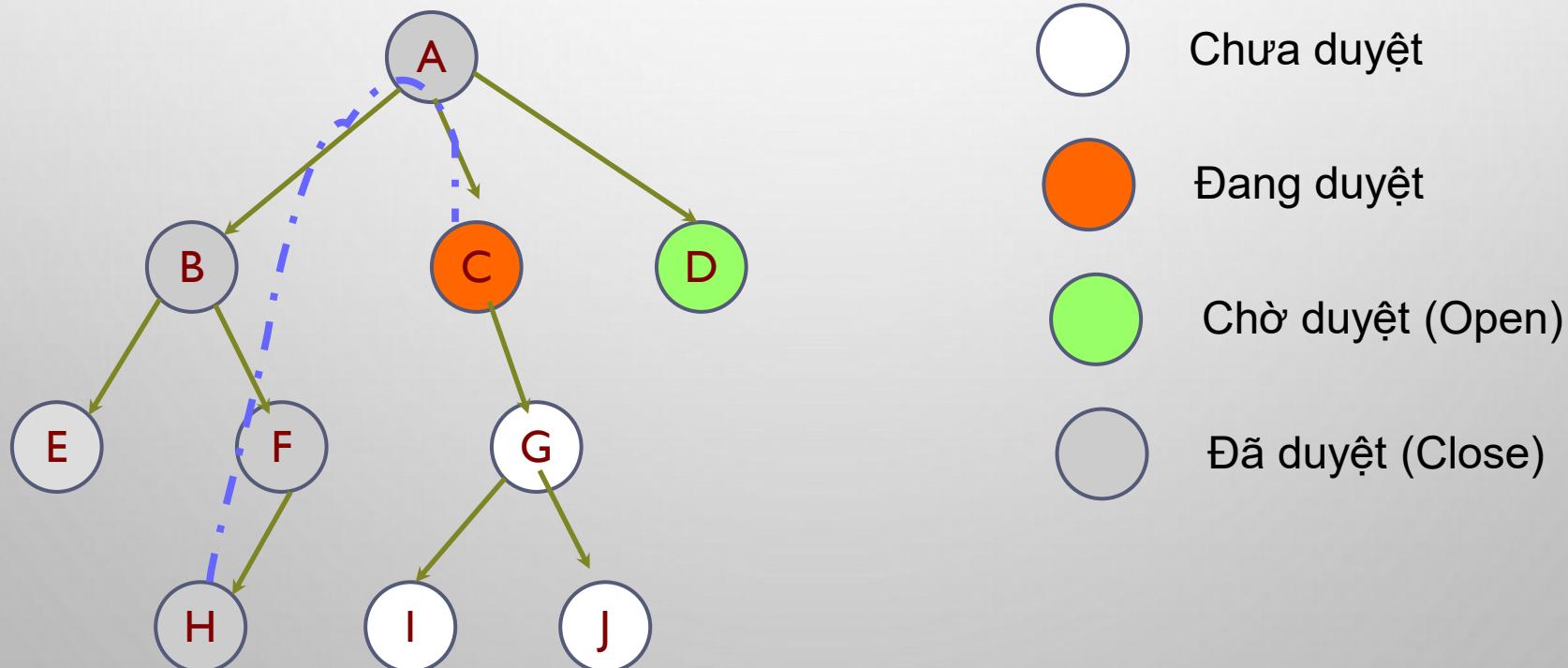
Tìm kiếm theo chiều sâu

- Tập trạng thái chờ duyệt “OPEN”: Stack
- Trạng thái sâu nhất (được đưa vào Open mới nhất) thì được phát triển trước
- Duyệt hết các nút con cháu, ... rồi mới đi lên duyệt các nút ở cùng độ sâu nếu chưa tìm thấy



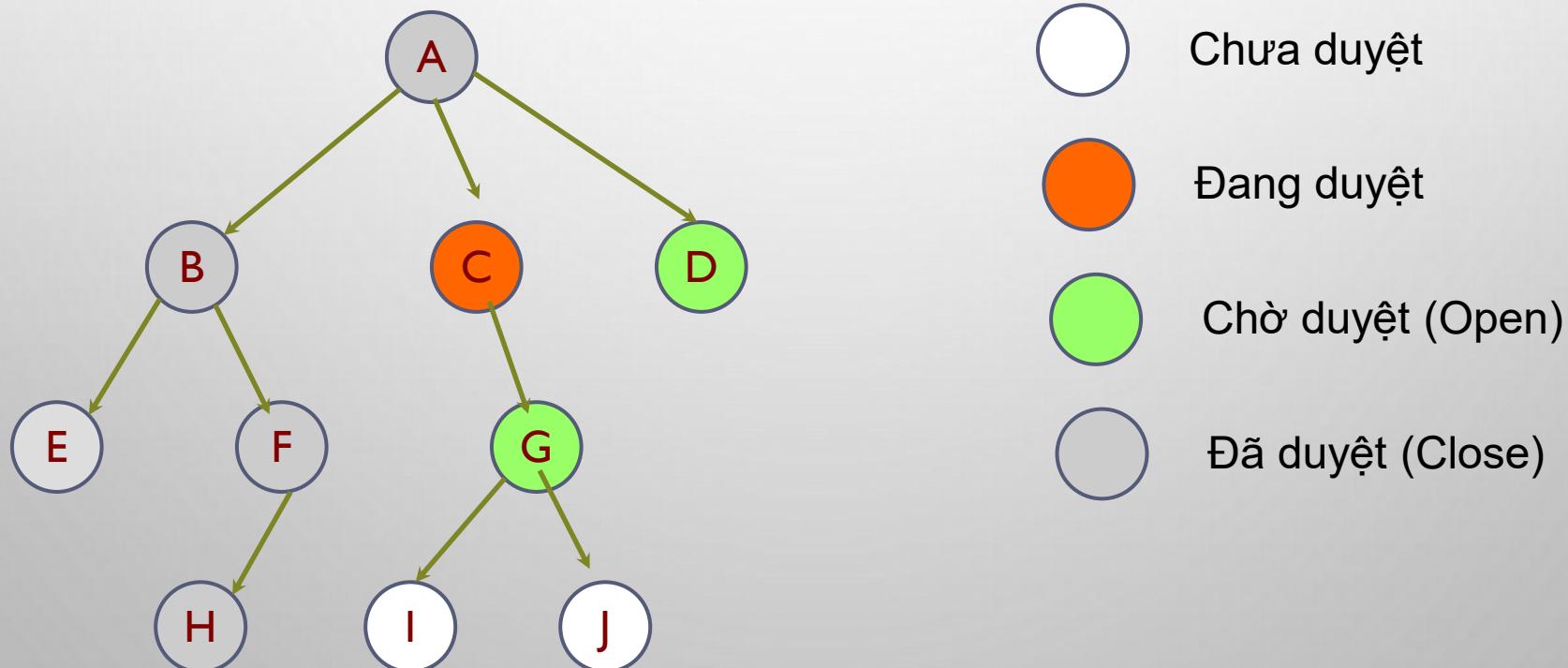
Tìm kiếm theo chiều sâu

- Tập trạng thái chờ duyệt “OPEN”: Stack
- Trạng thái sâu nhất (được đưa vào Open mới nhất) thì được phát triển trước
- Duyệt hết các nút con cháu, ... rồi mới đi lên duyệt các nút ở cùng độ sâu nếu chưa tìm thấy



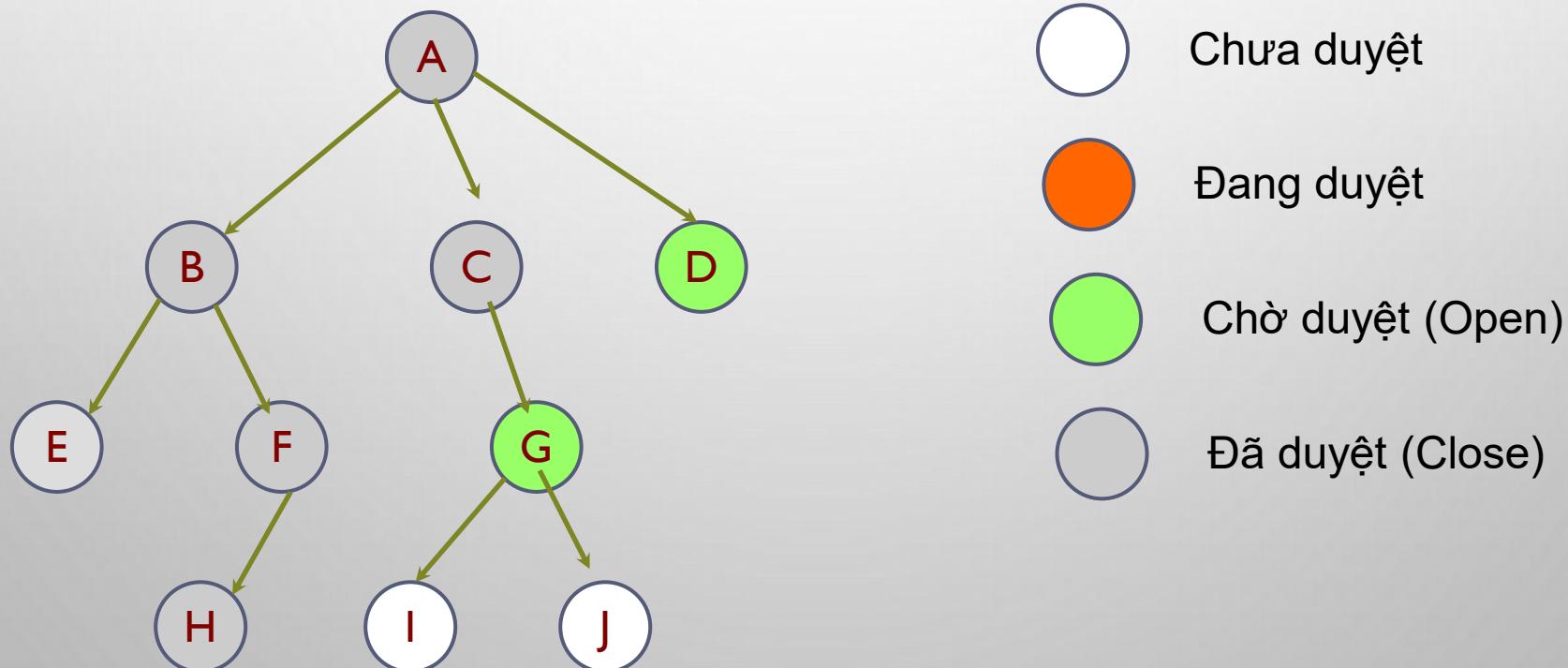
Tìm kiếm theo chiều sâu

- Tập trạng thái chờ duyệt “OPEN”: Stack
- Trạng thái sâu nhất (được đưa vào Open mới nhất) thì được phát triển trước
- Duyệt hết các nút con cháu, ... rồi mới đi lên duyệt các nút ở cùng độ sâu nếu chưa tìm thấy



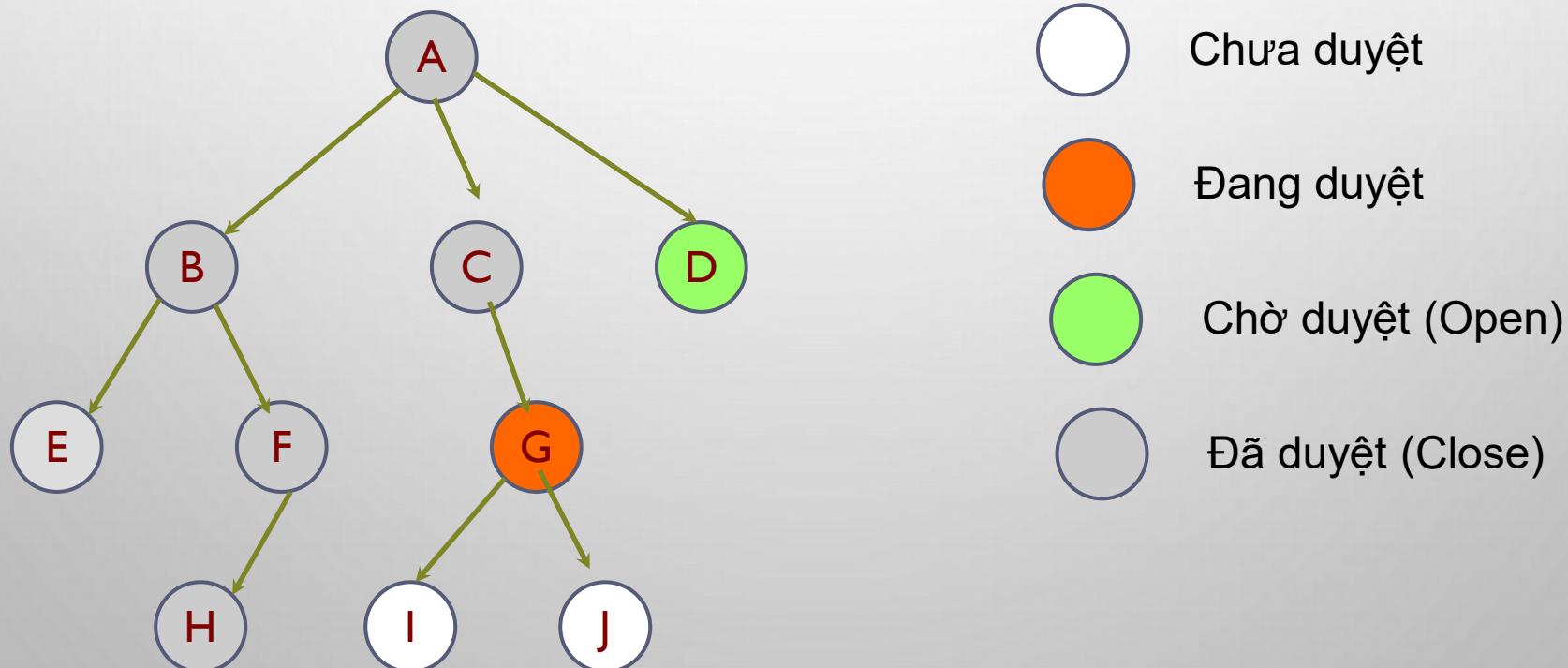
Tìm kiếm theo chiều sâu

- Tập trạng thái chờ duyệt “OPEN”: Stack
- Trạng thái sâu nhất (được đưa vào Open mới nhất) thì được phát triển trước
- Duyệt hết các nút con cháu, ... rồi mới đi lên duyệt các nút ở cùng độ sâu nếu chưa tìm thấy



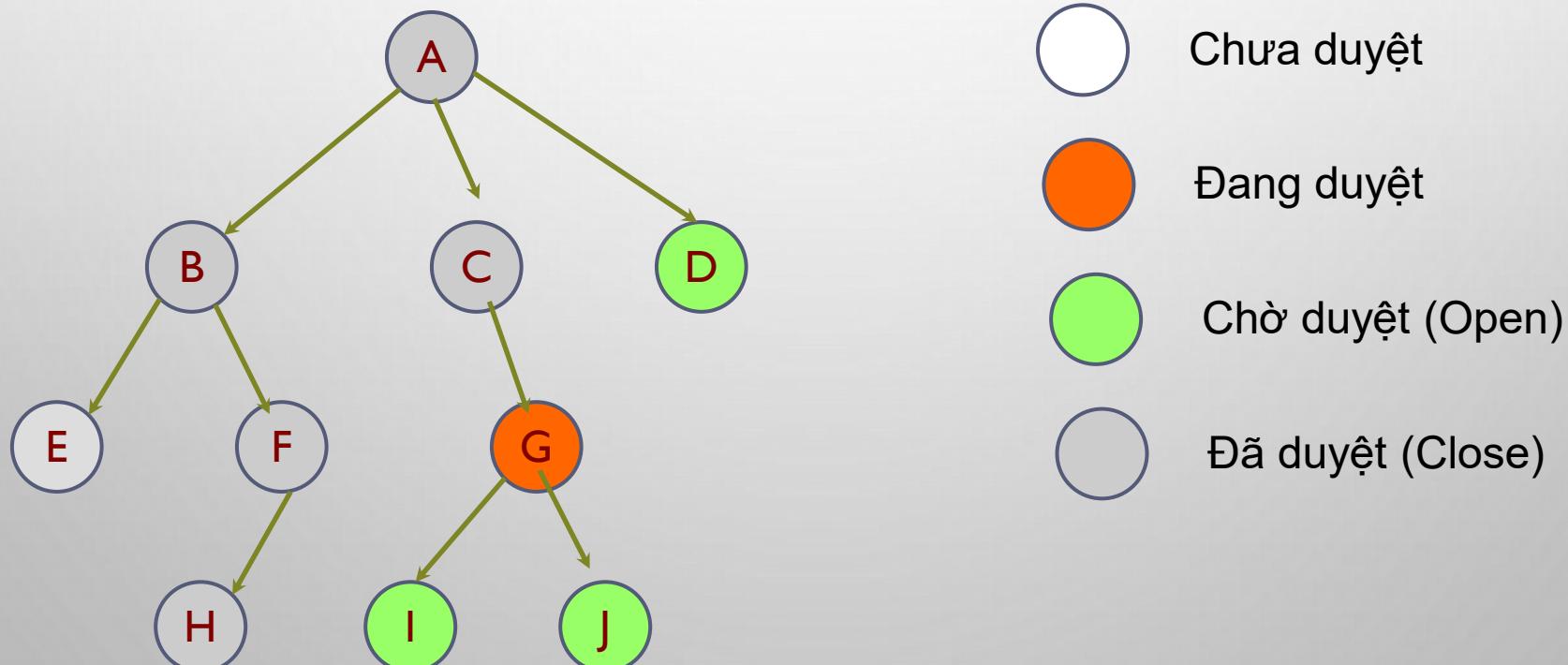
Tìm kiếm theo chiều sâu

- Tập trạng thái chờ duyệt “OPEN”: Stack
- Trạng thái sâu nhất (được đưa vào Open mới nhất) thì được phát triển trước
- Duyệt hết các nút con cháu, ... rồi mới đi lên duyệt các nút ở cùng độ sâu nếu chưa tìm thấy



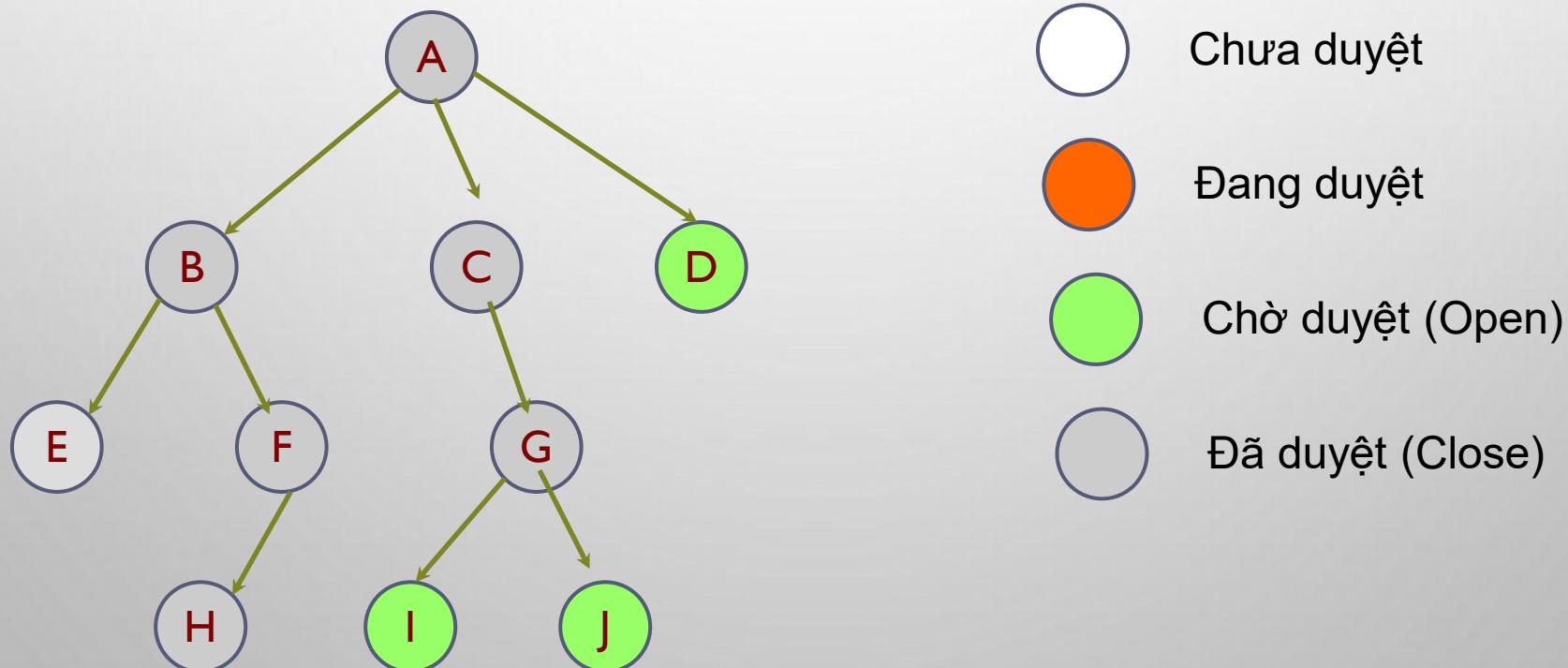
Tìm kiếm theo chiều sâu

- Tập trạng thái chờ duyệt “OPEN”: Stack
- Trạng thái sâu nhất (được đưa vào Open mới nhất) thì được phát triển trước
- Duyệt hết các nút con cháu, ... rồi mới đi lên duyệt các nút ở cùng độ sâu nếu chưa tìm thấy



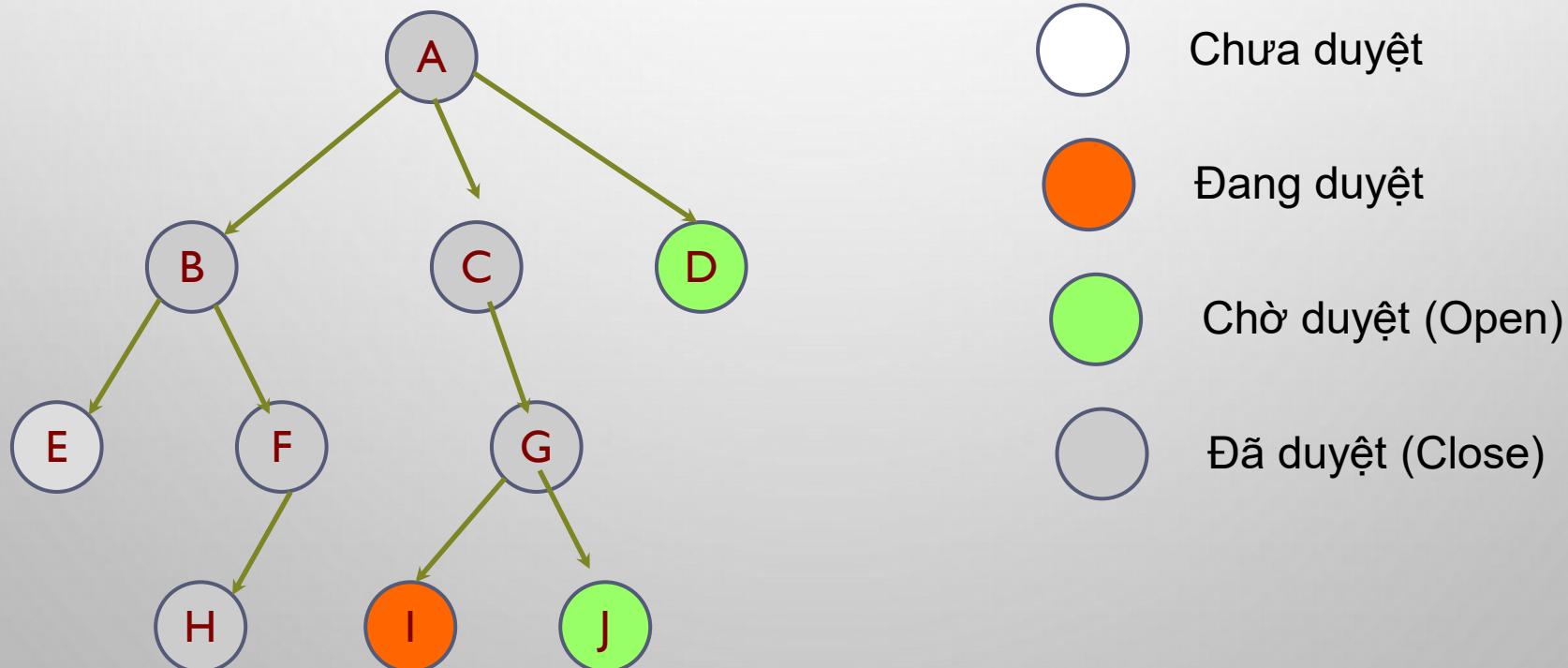
Tìm kiếm theo chiều sâu

- Tập trạng thái chờ duyệt “OPEN”: Stack
- Trạng thái sâu nhất (được đưa vào Open mới nhất) thì được phát triển trước
- Duyệt hết các nút con cháu, ... rồi mới đi lên duyệt các nút ở cùng độ sâu nếu chưa tìm thấy



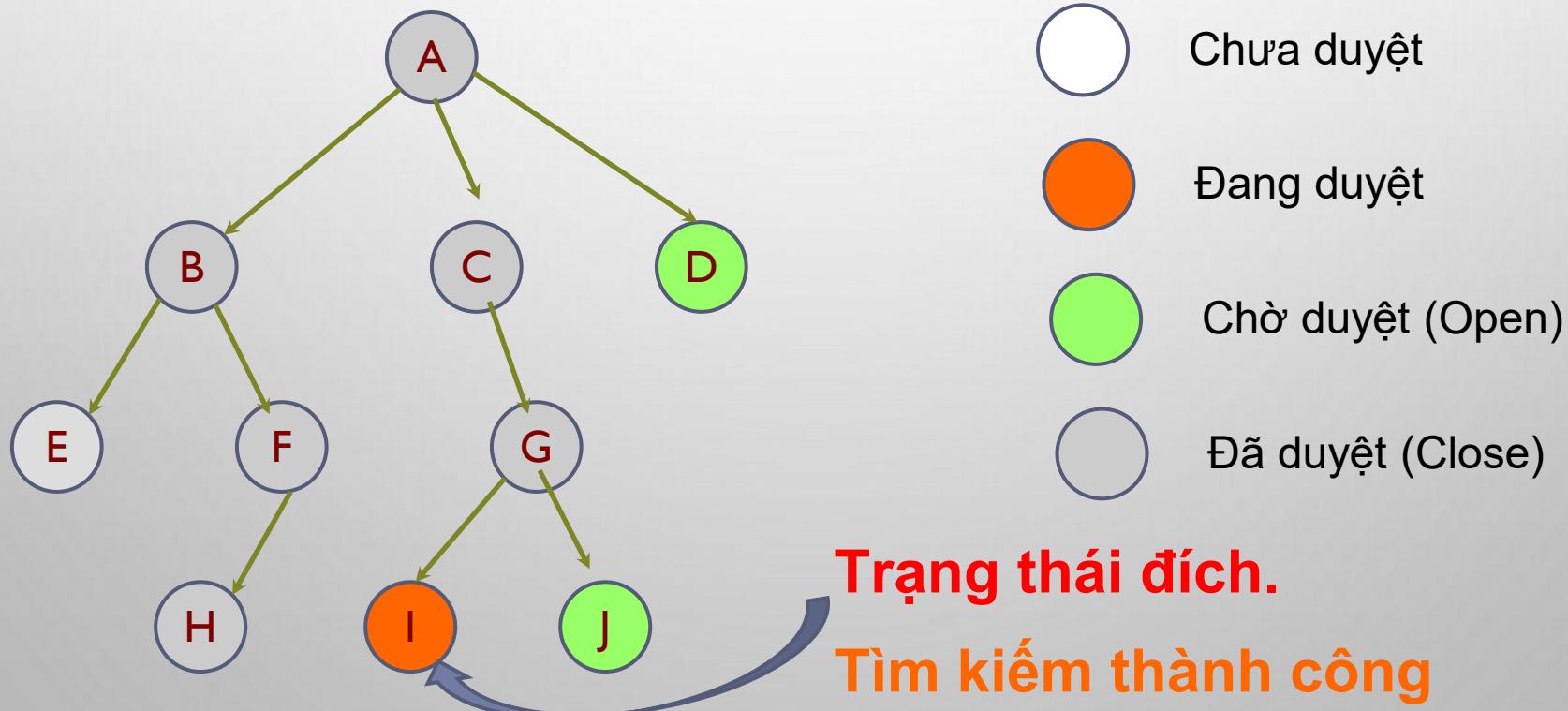
Tìm kiếm theo chiều sâu

- Tập trạng thái chờ duyệt “OPEN”: Stack
- Trạng thái sâu nhất (được đưa vào Open mới nhất) thì được phát triển trước
- Duyệt hết các nút con cháu, ... rồi mới đi lên duyệt các nút ở cùng độ sâu nếu chưa tìm thấy



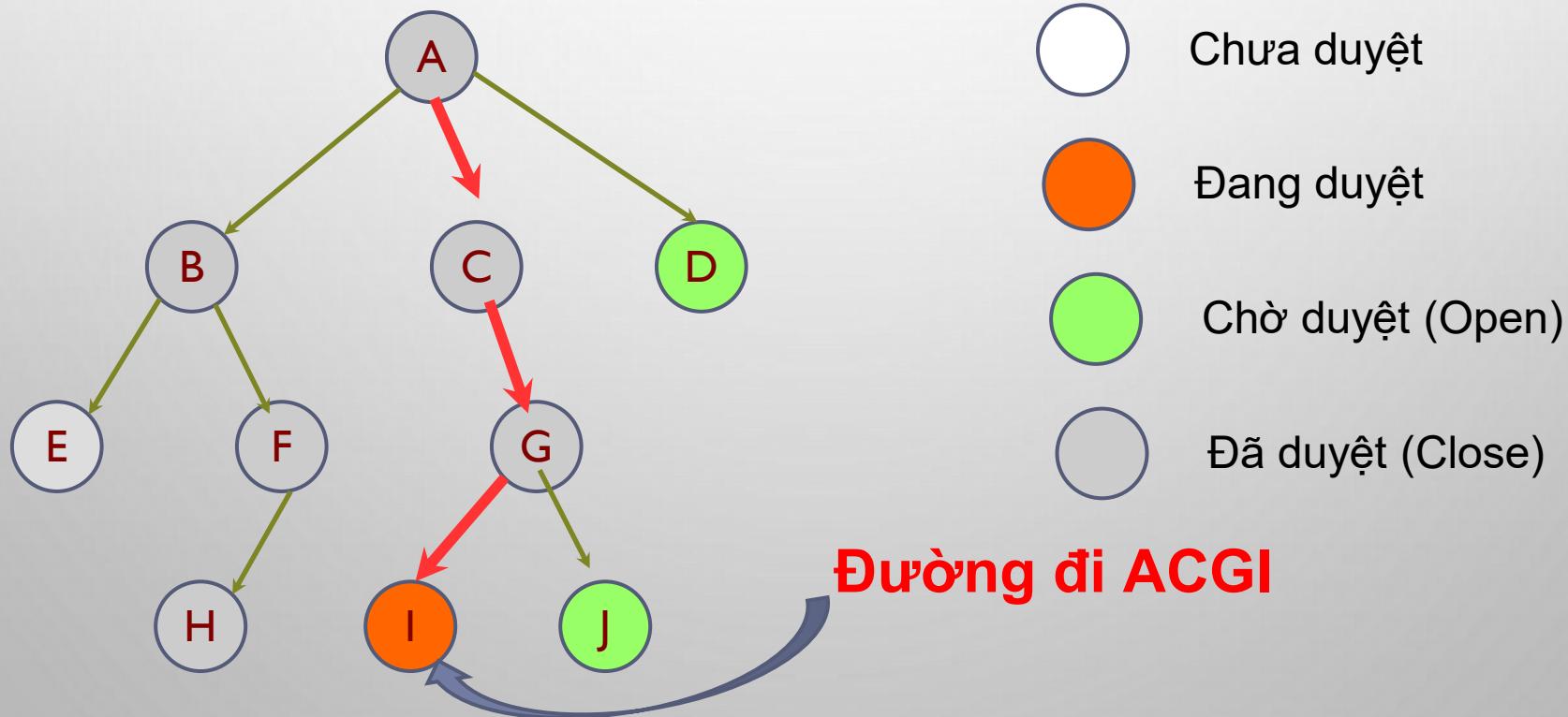
Tìm kiếm theo chiều sâu

- Tập trạng thái chờ duyệt “OPEN”: Stack
- Trạng thái sâu nhất (được đưa vào Open mới nhất) thì được phát triển trước
- Duyệt hết các nút con cháu, ... rồi mới đi lên duyệt các nút ở cùng độ sâu nếu chưa tìm thấy



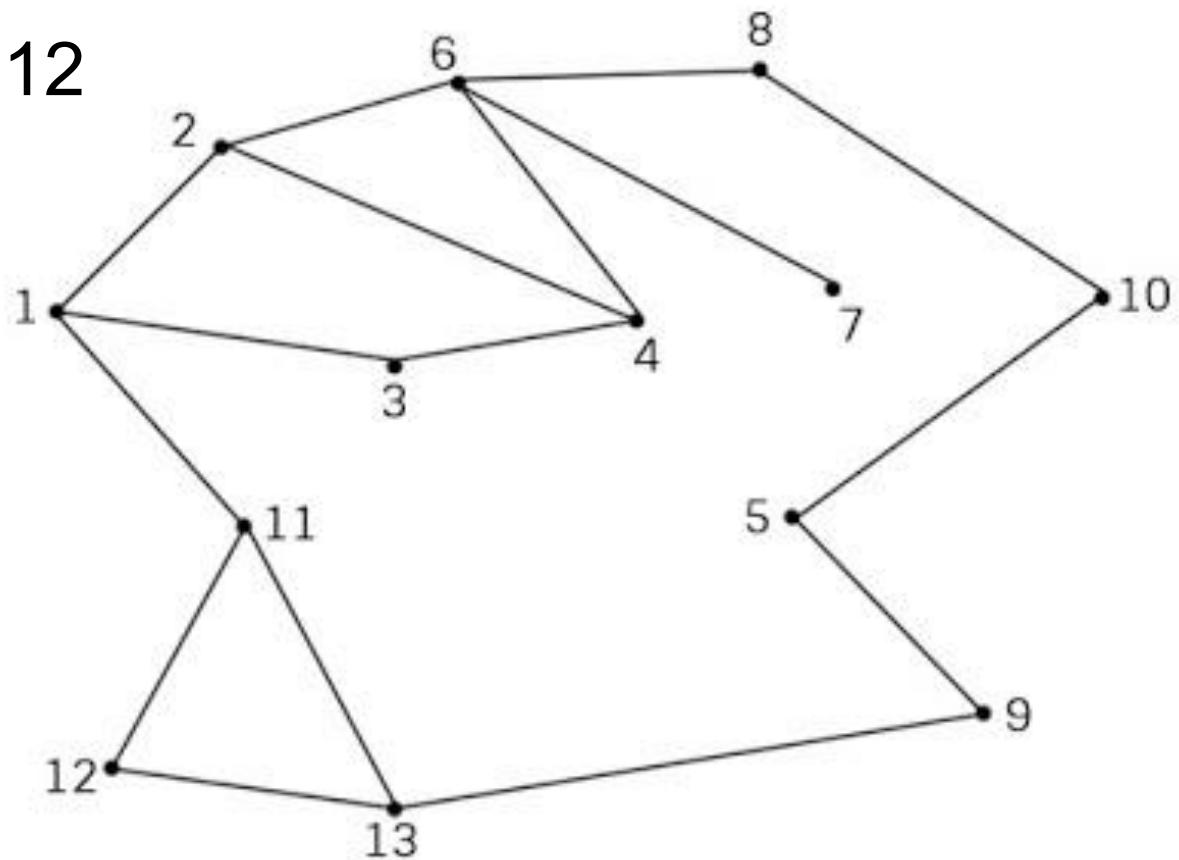
Tìm kiếm theo chiều sâu

- Tập trạng thái chờ duyệt “OPEN”: Stack
- Trạng thái sâu nhất (được đưa vào Open mới nhất) thì được phát triển trước
- Duyệt hết các nút con cháu, ... rồi mới đi lên duyệt các nút ở cùng độ sâu nếu chưa tìm thấy



Tìm kiếm theo chiều sâu

- Bài tập: Tìm đường đi và trình tự duyệt các đỉnh
 - Trạng thái đầu: 1
 - Trạng thái đích: 12



Tìm kiếm theo chiều sâu

- Procedure Depth_First_Search
 - 1. Khởi tạo Open = {trạng thái ban đầu};
 - 2. while true do
 - 2.1 If (Open rỗng) then
 - {thông báo thất bại; stop};
 - 2.2 Loại trạng thái u **ở đầu** danh sách Open;
 - 2.3 If (u là trạng thái kết thúc) then
 - {thông báo thành công; stop};
 - 2.4 Thêm các nút trạng thái kề với u **vào đầu** danh sách Open

Tìm kiếm theo chiều sâu

- Độ phức tạp bộ nhớ

- Chỉ cần lưu các nút con (kè) chưa được phát triển của các đỉnh nằm trên đường đi từ gốc tới đích
- Số nút cần lưu = $b + b + \dots + b = d.b = O(b.d)$

- Độ phức tạp thời gian

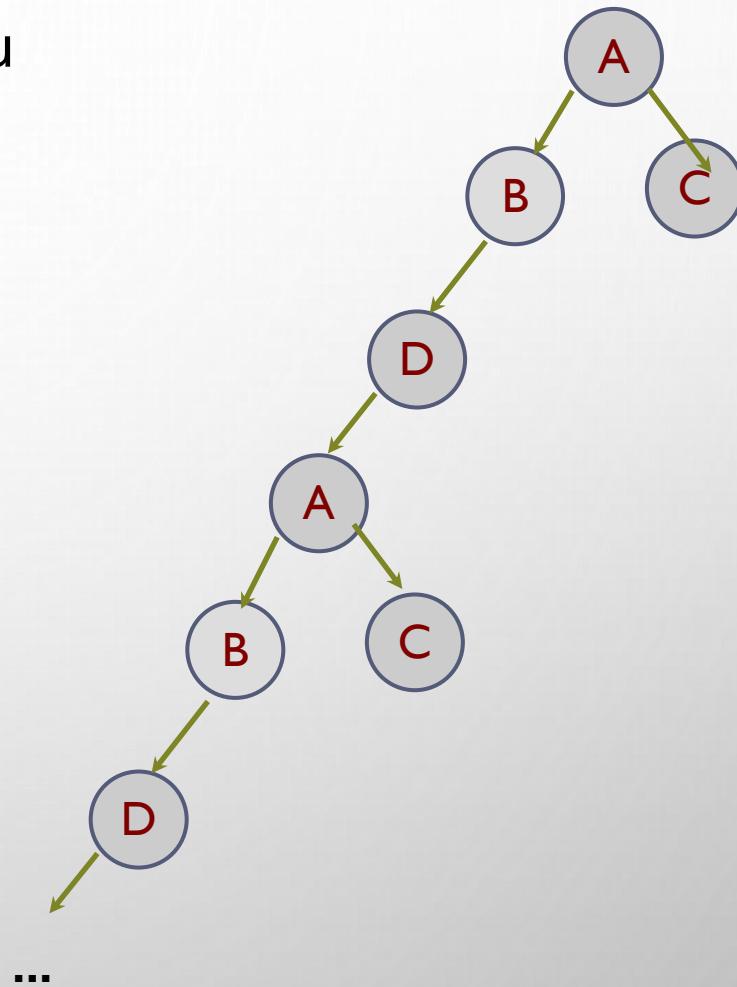
- Trường hợp tốt nhất (nghiệm là đỉnh ngoài cùng bên trái ở mức d):
 - . Số nút duyệt = d
- Trường hợp xấu nhất (nghiệm là đỉnh ngoài cùng bên phải ở mức d)
 - . Số nút duyệt = $1 + b + b^2 + \dots + b^{d-1} + k = O(b^d)$

So sánh

	Chiều rộng	Chiều sâu
Open	FIFO (Queue)	LIFO (Stack)
Hiệu quả	Khi trạng thái đích nằm gần gốc của cây tìm kiếm	Khi trạng thái đích nằm sâu trong cây tìm kiếm và có một phương án chọn hướng đi chính xác
Độ phức tạp	Tốn nhiều bộ nhớ hơn Có cùng độ phức tạp thời gian về mặt lý thuyết nhưng chậm hơn trong thực tế	
Kết quả	Chắc chắn tìm ra kết quả nếu có	Không tìm ra nghiệm trong không gian lặp hoặc không gian có độ sâu vô hạn

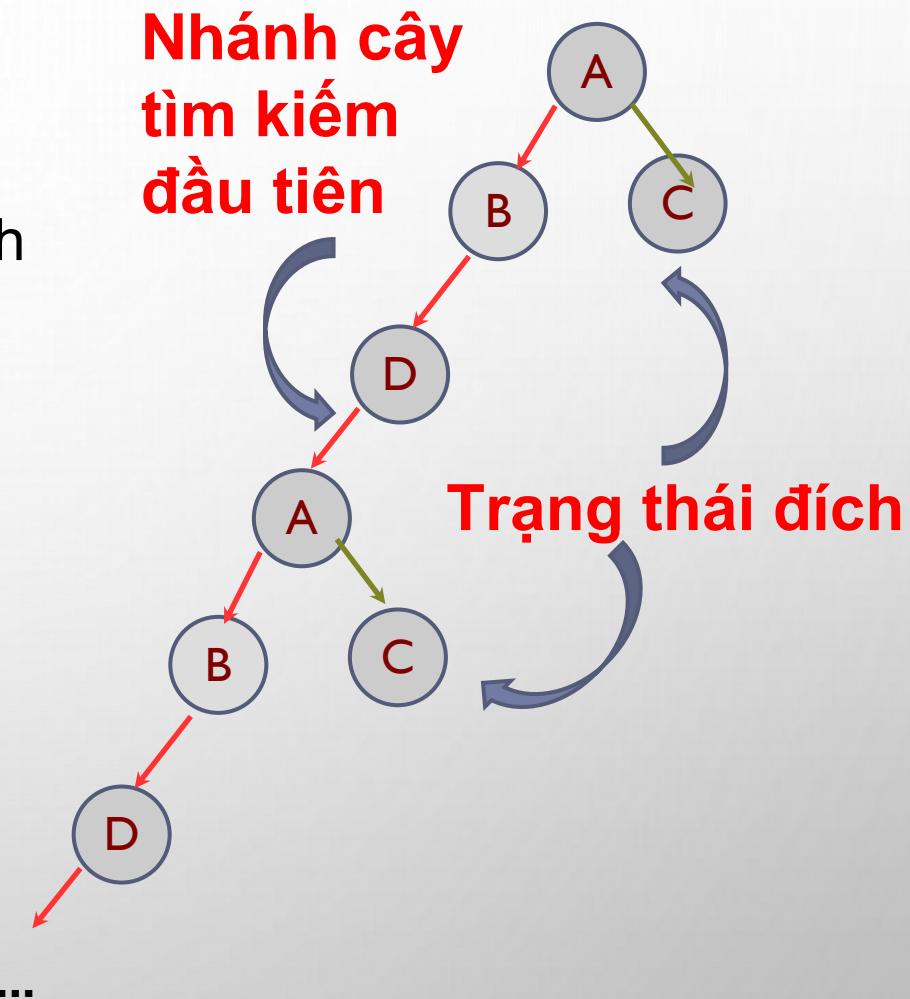
Vấn đề của tìm kiếm theo chiều sâu

- Không gian tìm kiếm có thể có độ sâu vô hạn hoặc lặp trạng thái.



Vấn đề của tìm kiếm theo chiều sâu

- Không gian tìm kiếm có thể có độ sâu vô hạn hoặc lặp trạng thái.
- Khi đó, DFS có thể bị mắc kẹt ở nhánh vô hạn và không tìm ra nghiệm.
- Khắc phục:
 - Chỉ phát triển các nút chưa được phát triển.
 - Thêm biến trạng thái boolean duyệt(u).
 - Hạn chế độ sâu của giải thuật: tìm kiếm theo chiều sâu hạn chế.



Tìm kiếm theo chiều sâu hạn chế

- Chỉ tìm kiếm (theo chiều sâu) trong 1 giới hạn độ sâu nào đó.
 - Các nút nằm ở độ sâu sâu hơn không được xét duyệt.
- Procedure Depth_Limited_Search(l)
 - begin
 - 1. Khởi tạo Open = {trạng thái ban đầu u0};
 - $\text{depth}(u0) = 0;$
 - 2. while true do
 - 2.1 If (Open rỗng) then
 - {thông báo thất bại; stop};
 - 2.2 Loại trạng thái u ở đầu danh sách Open;
 - 2.3 If (u là trạng thái kết thúc) then
 - {thông báo thành công; stop};
 - 2.4 If ($\text{depth}(u) < l$) then
 - for (mỗi trạng thái v kè u) do
 - Thêm v vào đầu danh sách Open;
 - $\text{depth}(v) = \text{depth}(u) + 1;$
 - end

Tìm kiếm sâu lấp

- Còn gọi là tìm kiếm sâu dần.
- Là tìm kiếm theo chiều sâu hạn chế với giới hạn độ sâu tăng dần từ 0 tới một mức max.
- Quá trình tìm kiếm:
 - Tìm kiếm theo độ sâu ở mức giới hạn d nào đó (bắt đầu từ $d=0$)
 - Nếu không tìm thấy nghiệm, tìm kiếm theo độ sâu ở mức giới hạn độ sâu $d+1$

Tìm kiếm sâu lặp

- Procedure Depth_Deepening_Search(max)
- begin
 - for (l = 0 to max) do{
 - Depth_Limited_Search(l);
 - if (thành công) then exit;
 - }
- end

Tìm kiếm sâu lặp

- Nếu nghiệm có độ sâu d và cây có nhân tố nhánh b thì ta đã thực hiện các thủ tục DLS(0), DLS(1), ..., DLS(max)
- Độ phức tạp thời gian
 - Số đỉnh phải phát triển ở mức 0 là 1, lặp max+1 lần
 - Số đỉnh phải phát triển ở mức 1 là b, lặp max lần
 - Số đỉnh phải phát triển ở mức 2 là b^2 , lặp $d=\max-1$ lần
 - ...
 - Số đỉnh phải phát triển ở mức max là b^{\max} , lặp 1 lần
 - Số đỉnh duyệt = $(\max+1).1 + \max.b + (\max-1).b^2 + \dots + 2.b^{\max-1} + 1.b^{\max} = O(b^{\max})$

So sánh các giải thuật tìm kiếm mù

	Chiều rộng	Chiều sâu	Chiều sâu hạn chế	Sâu lặp
Hoàn thành	Có	Không	Không	Không
Thời gian	$O(b^d)$	$O(b^d)$	$O(b^l)$	$O(b^{\max})$
Không gian	$O(b^{d+1})$	$O(b.d)$	$O(b.l)$	$O(b.\max)$

- Hoàn thành: là khả năng luôn tìm thấy nghiệm nếu có.
 - Chiều sâu: không tìm thấy nếu không gian lặp trạng thái hoặc có chiều sâu vô hạn
 - Chiều sâu hạn chế: không tìm thấy nếu nghiệm nằm sâu hơn giới hạn

Tìm kiếm với tri thức bổ xung

- Tìm kiếm dựa trên các thông tin đánh giá của các trạng thái
 - **Hàm đánh giá**
- Hiệu quả

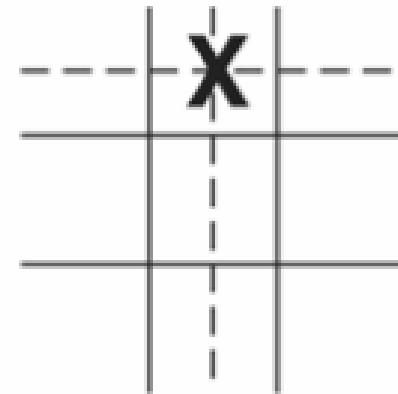
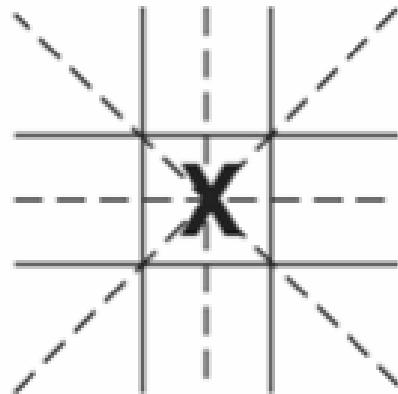
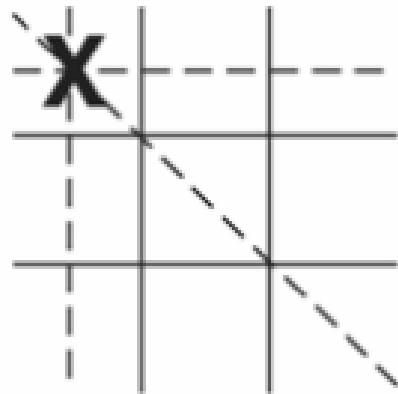
Hàm đánh giá

- Là hàm ước lượng, đánh giá mức độ tốt/xấu, khả năng về đích của mỗi trạng thái
- Dựa trên kinh nghiệm
- Với 1 trạng thái u ,:
 - $g(u)$ là chi phí đã đi từ xuất phát (thông tin **quá khứ**, đã xác định)
 - $h(u)$ là chi phí còn lại để đi tới đích (thông tin **tương lai** hay **heuristic**, ước lượng). Giá trị này càng nhỏ thì càng tốt
 - $g(u) + h(u)$ là tổng chi phí đi từ trạng thái xuất phát tới đích có qua trạng thái đó (thông tin **chung**, ước lượng). Giá trị này càng nhỏ thì càng tốt

Hàm đánh giá

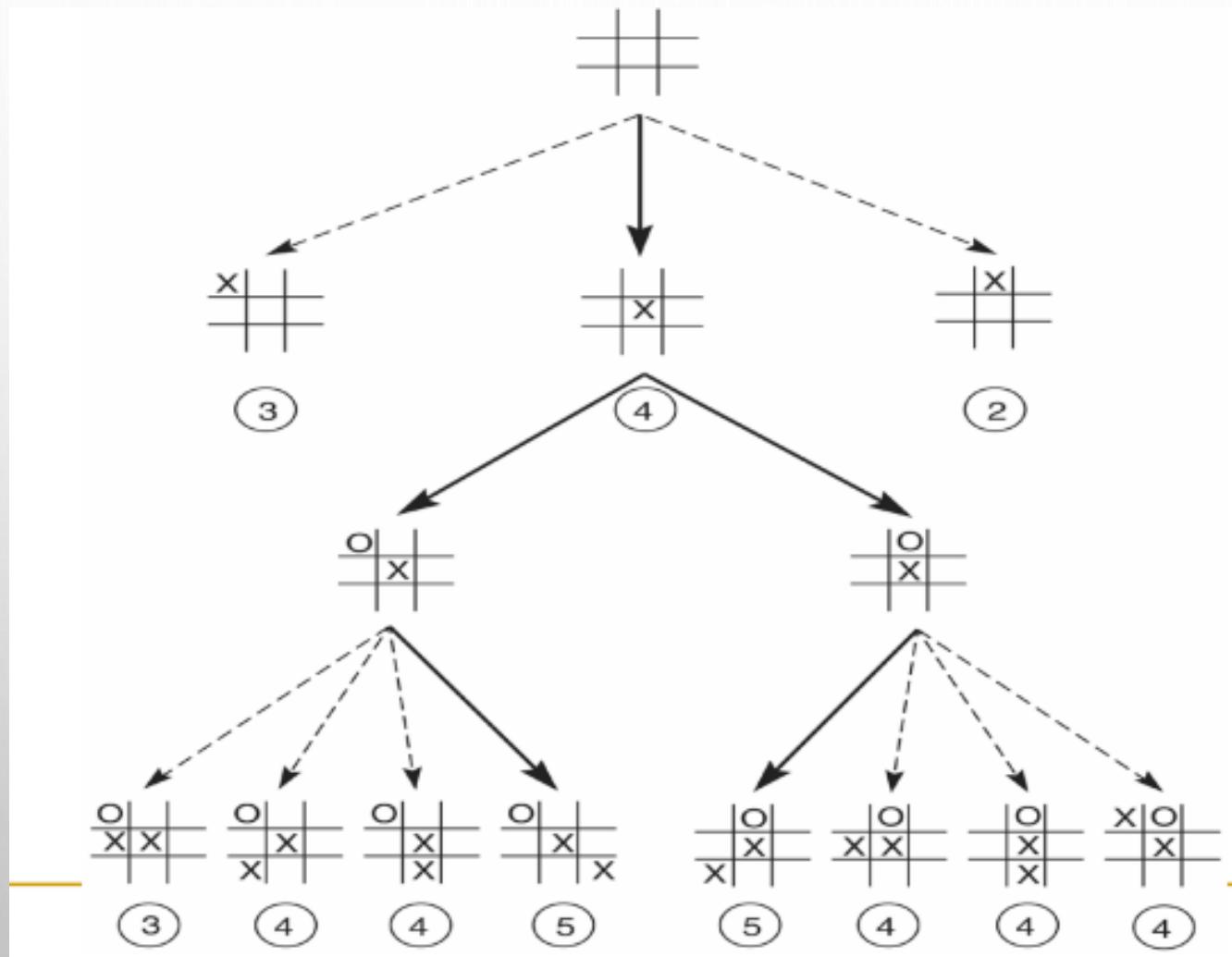
- Có thể chỉ khai thác 1 hoặc cả 2 thông tin đánh giá về quá khứ, tương lai
 - $f(u) = g(u) + h(u)$
 - $f(u) = g(u)$
 - $f(u) = h(u)$
- Các kĩ thuật TK sử dụng hàm đánh giá $h(u)$ gọi chung là TK kinh nghiệm - heuristic search
- Các kĩ thuật TK tối ưu sử dụng hàm đánh giá $f(u)=g(u)+h(u)$
- Hàm đánh giá càng tốt (sát với thực tế) thì tìm kiếm càng nhanh

Phép đo heuristic



- Heuristic $h(u)$: số đường thẳng đi qua các quân cờ của ta mà không bị chặn bởi đối phương
- Là số nước ăn hay khả năng thắng
- $h(u)$ càng lớn càng tốt

Phép đo heuristic



- Không gian trạng thái của trò chơi tic-tac-toe

- <https://voer.edu.vn/c/mo-dau/764b3239/2ab0f63d>

Phép đo heuristic

7	2	5
3	4	8
6		1

Trạng thái u

1	2	3
8		4
7	6	5

Goal

- Cách 1: tổng số ô sai khác so với trạng thái đích
 - $h_1(u) = 7$
- Cách 2: tổng số bước dịch chuyển để đưa các ô sai khác về trạng thái đích (tổng khoảng cách Manhattan)
 - $h_2(u) = 2+0+2+3+1+2+1+4=15$
- $h(u)$ càng nhỏ càng tốt

Phép đo heuristic*

- Tính giá trị heuristic h_1 , h_2 cho các trạng thái sau

2	8	6
3	4	7
1		5

$h = ?$

2	5	3
8	6	4
7		1

$h = ?$

1	2	3
8		4
7	6	5

goal

Phép đo heuristic

- Thông thường, $h(u)$ biểu diễn ước lượng chi phí để đi từ u tới đích
- Khi $h(u)$ biểu diễn ước lượng khả năng thắng trong game (như tic-tac-toe) thì ta sử dụng $-h(u)$
- Một ước lượng $h(u)$ là chấp nhận được nếu với mọi u , $h(u) \leq h^*(u)$, trong đó $h^*(u)$ là chi phí thực sự để đi từ u tới đích

Hàm đánh giá (sửa lại hình)

- **Bài toán tic-tac-toe**

- **Quá khứ**

- $g(u)$ = khoảng cách thực sự từ trạng thái đầu tới u

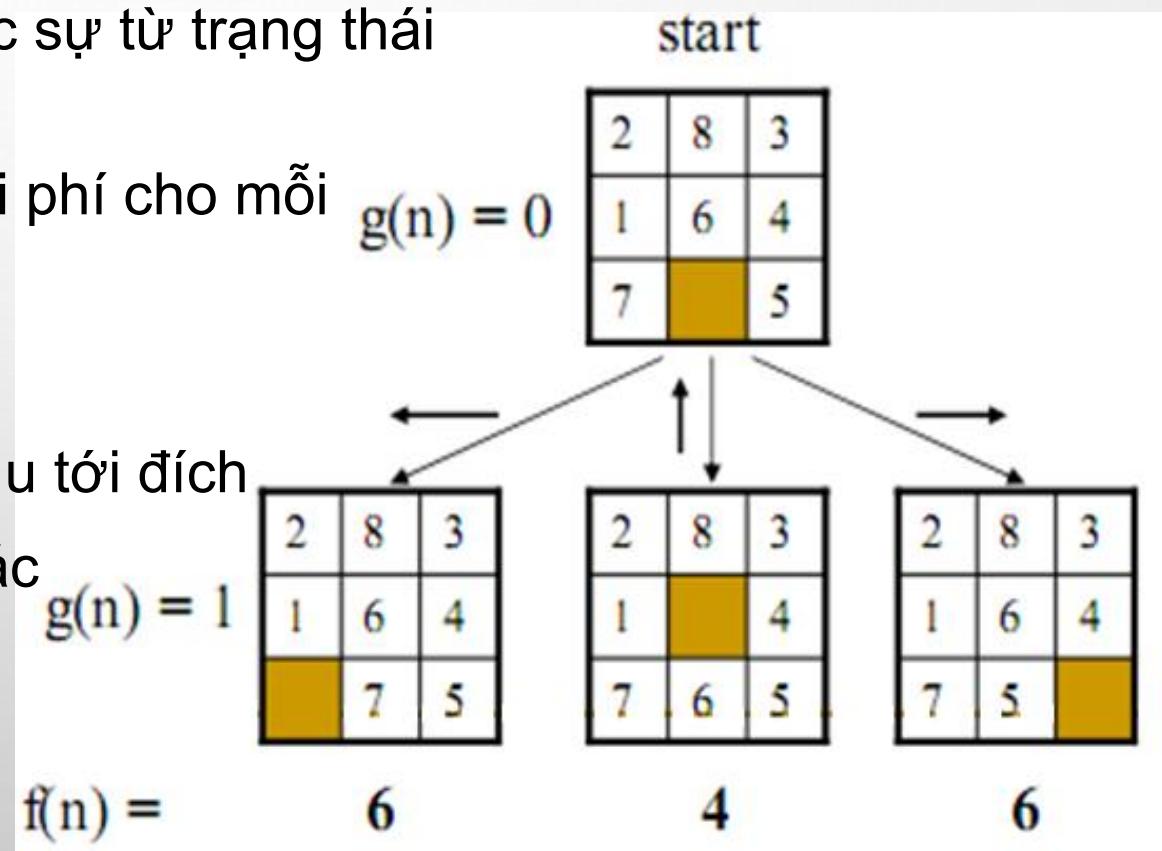
- . = số bước đã đi (chi phí cho mỗi bước đi là 1)

- **Tương lai**

- $h_1(u)$ = khoảng cách từ u tới đích
 - . = số ô còn sai khác

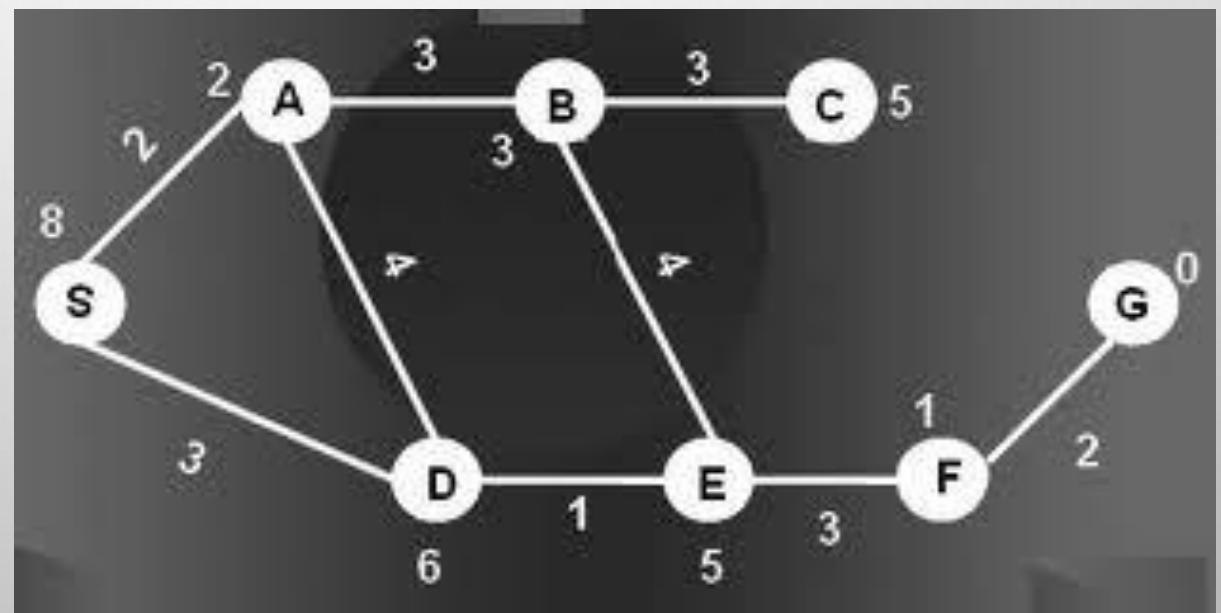
- **Hàm đánh giá**

- $f(u) = g(u) + h_1(u)$



Hàm đánh giá

- Bài toán người du lịch: tìm đường đi ngắn nhất từ 1 thành phố ban đầu đến 1 thành phố đích
- Hàm đánh giá
 - Quá khứ: $g(u) = \text{khoảng cách đã đi từ thành phố xuất phát}$
 - Tương lai: $h(u) = \text{khoảng cách theo đường chim bay từ 1 thành phố đến thành phố đích}$



Tìm kiếm với hàm đánh giá

- TK kinh nghiệm (TK heuristic): sử dụng hàm đánh giá $f(u) = h(u)$
 - TK tốt nhất đầu tiên = TK bề rộng + $h(u)$
 - TK leo đồi = TK chiều sâu + $h(u)$
 - Tìm kiếm tối ưu: sử dụng hàm đánh giá $f(u) = g(u) + h(u)$
 - TK A* = TK tốt nhất đầu tiên + $f(u)$
 - TK nhánh cận = TK leo đồi + $f(u)$
- ...

Tìm kiếm kinh nghiệm

Tìm kiếm tốt nhất đầu tiên

- Là TK theo bề rộng được định hướng bởi hàm đánh giá $h(u)$
- OPEN : tập các đỉnh chờ phát triển được sắp xếp theo thứ tự tăng dần của hàm đánh giá
- Tại mỗi bước của tìm kiếm:
 - Chọn đỉnh u trong OPEN có giá trị hàm đánh giá nhỏ nhất để duyệt
 - Đưa các đỉnh kề của u vào OPEN sao cho OPEN được sắp xếp theo thứ tự **tăng dần của hàm đánh giá**

Tìm kiếm tốt nhất đầu tiên

```
procedure Best_First_Search
```

```
begin
```

```
1. Khởi tạo danh sách OPEN = {trạng thái ban đầu};
```

```
2. while true do
```

```
2.1 if (OPEN rỗng) then
```

```
{thông báo tìm kiếm thất bại; stop};
```

```
2.2 Loại trạng thái u ở đầu danh sách OPEN;
```

```
2.3 if u là trạng thái kết thúc then
```

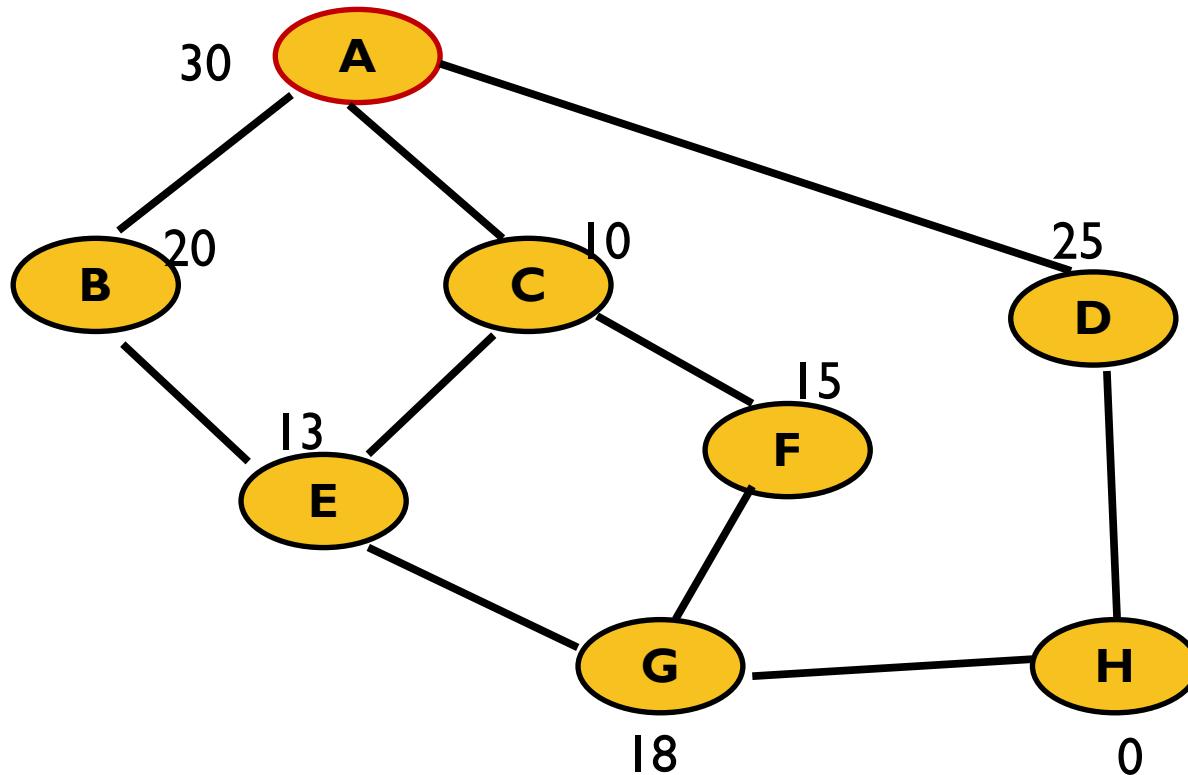
```
{thông báo tìm kiếm thành công; stop};
```

```
2.4 Chèn các đỉnh kề của u vào OPEN sao cho OPEN được sắp xếp theo thứ tự  
tăng dần của hàm đánh giá
```

```
end
```

Tìm kiếm tốt nhất đầu tiên

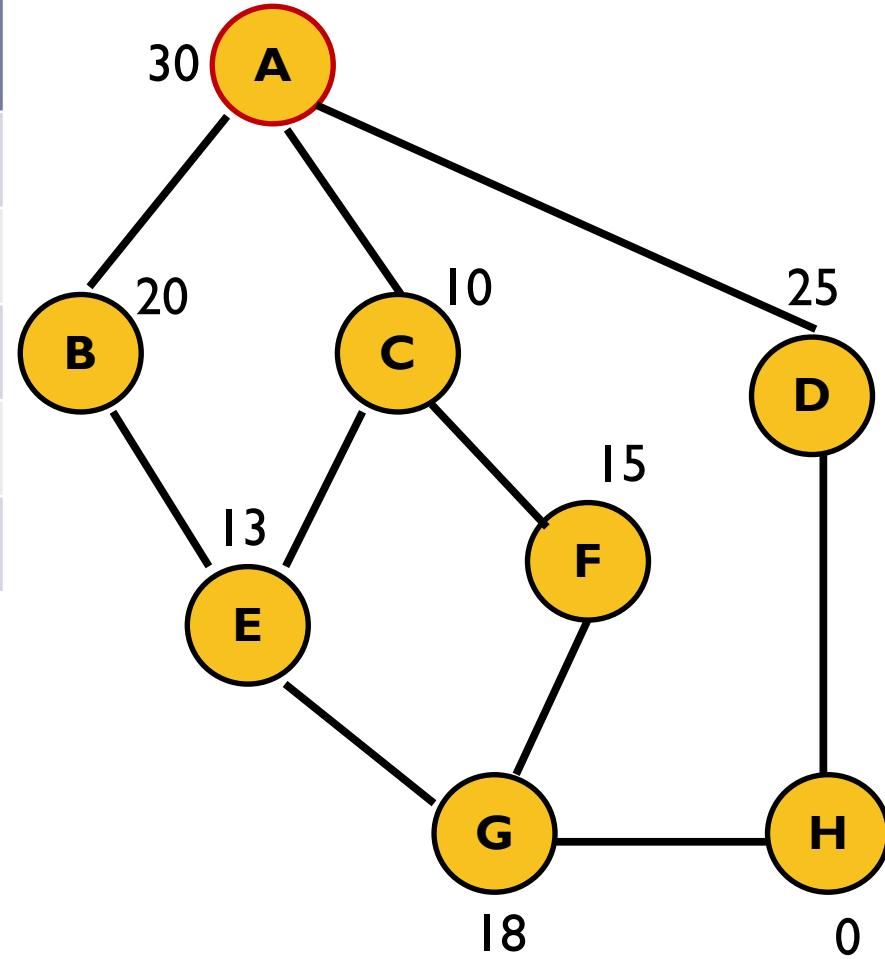
Ví dụ: tìm đường đi ngắn nhất từ $A \rightarrow H$ (sửa lại mũi tên)



Giá trị gắn với mỗi nút là đánh giá heuristic $h(u)$

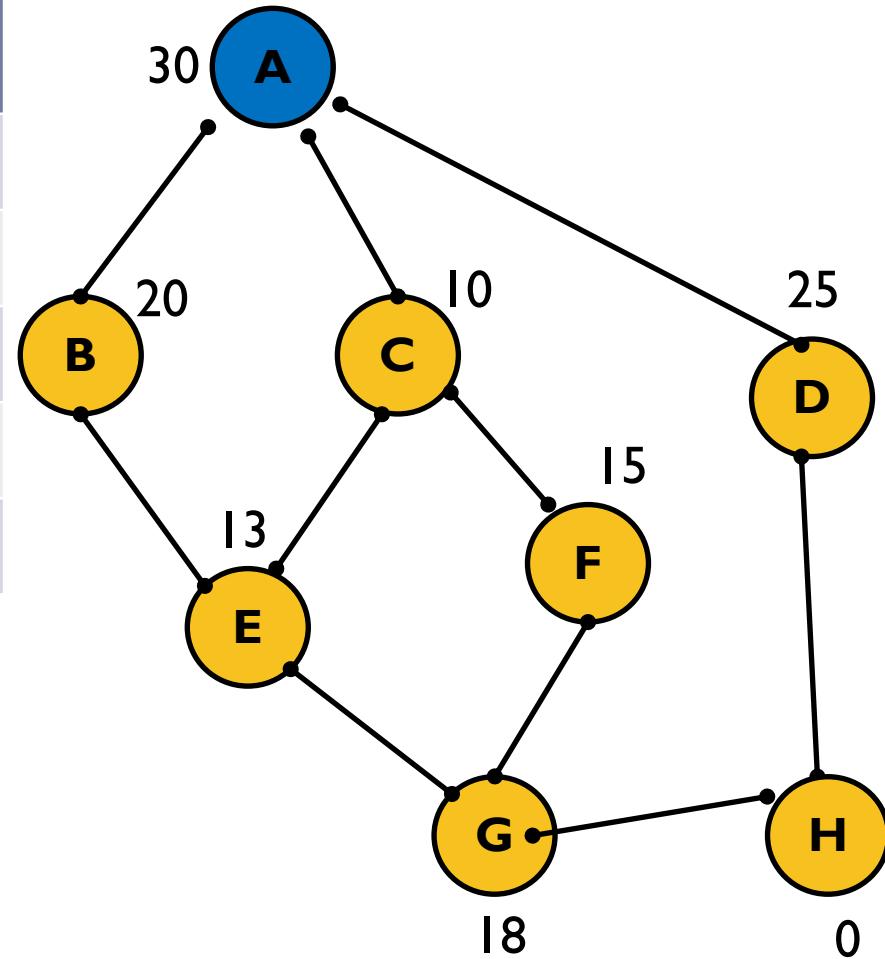
Tìm kiếm tốt nhất đầu tiên

Bước lắp	u	kề(u)	OPEN
0			A ³⁰



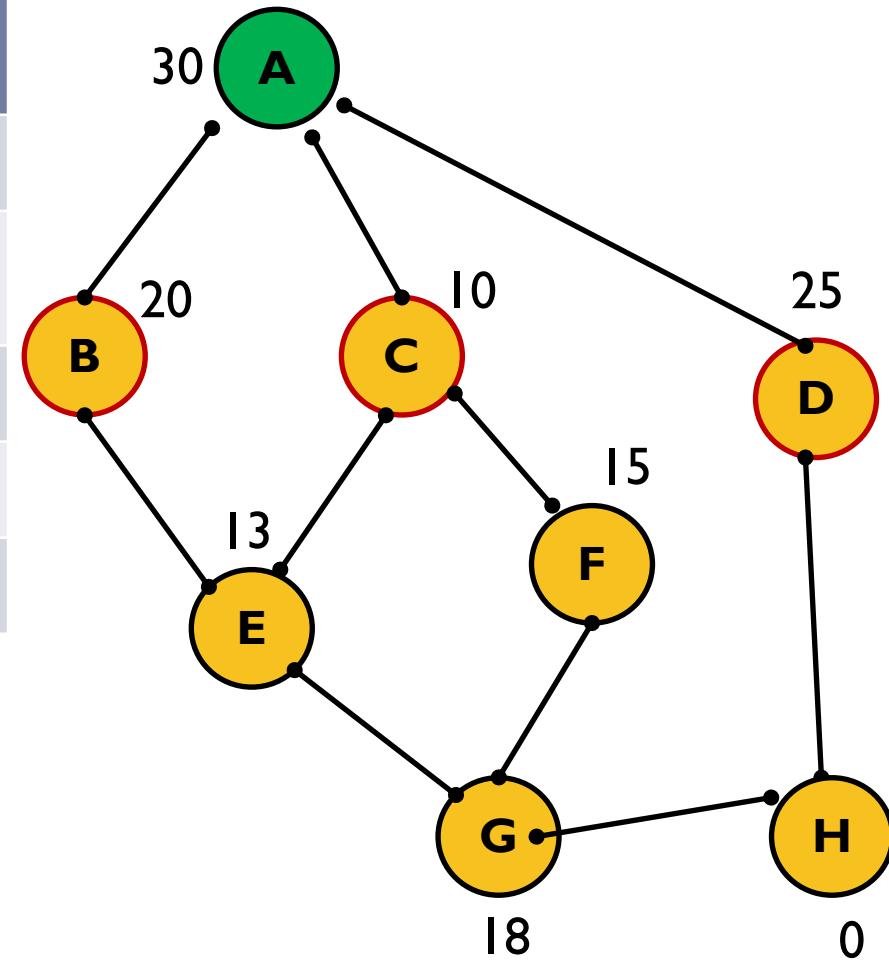
Tìm kiếm tốt nhất đầu tiên

Bước lắp	u	kề(u)	OPEN
0			A^{30}
I	A		



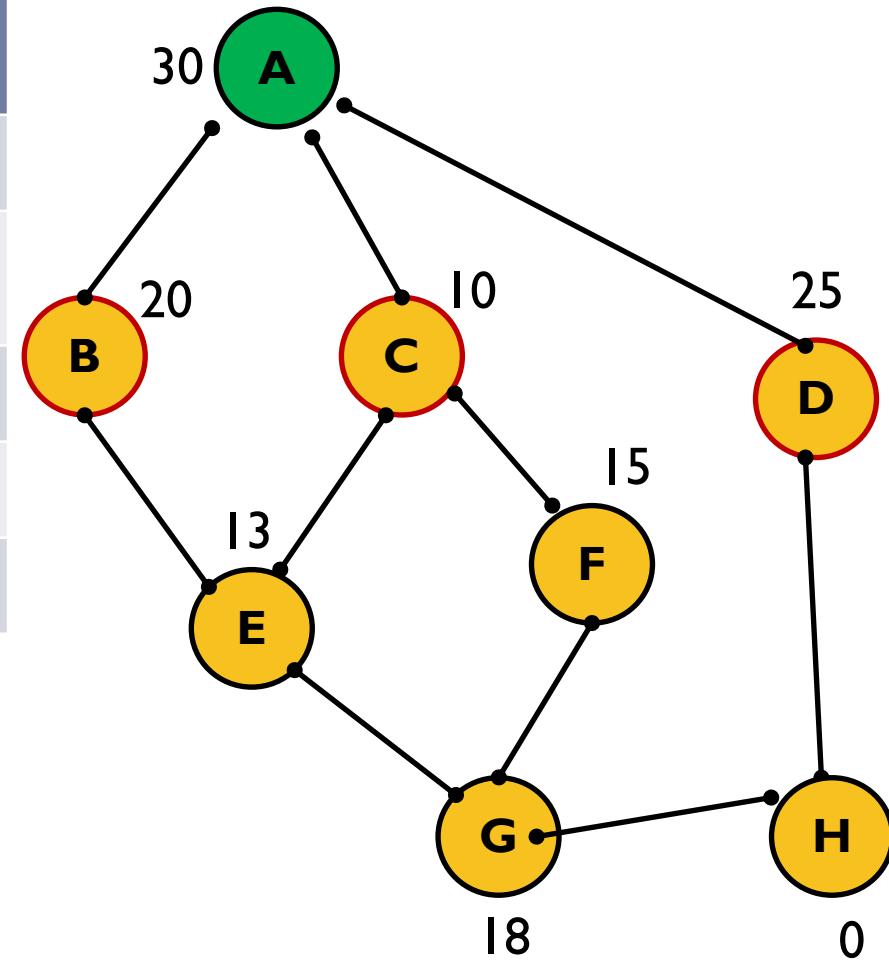
Tìm kiếm tốt nhất đầu tiên

Bước lắp	u	kề(u)	OPEN
0			A^{30}
I	A	B^{20}, C^{10}, D^{25}	



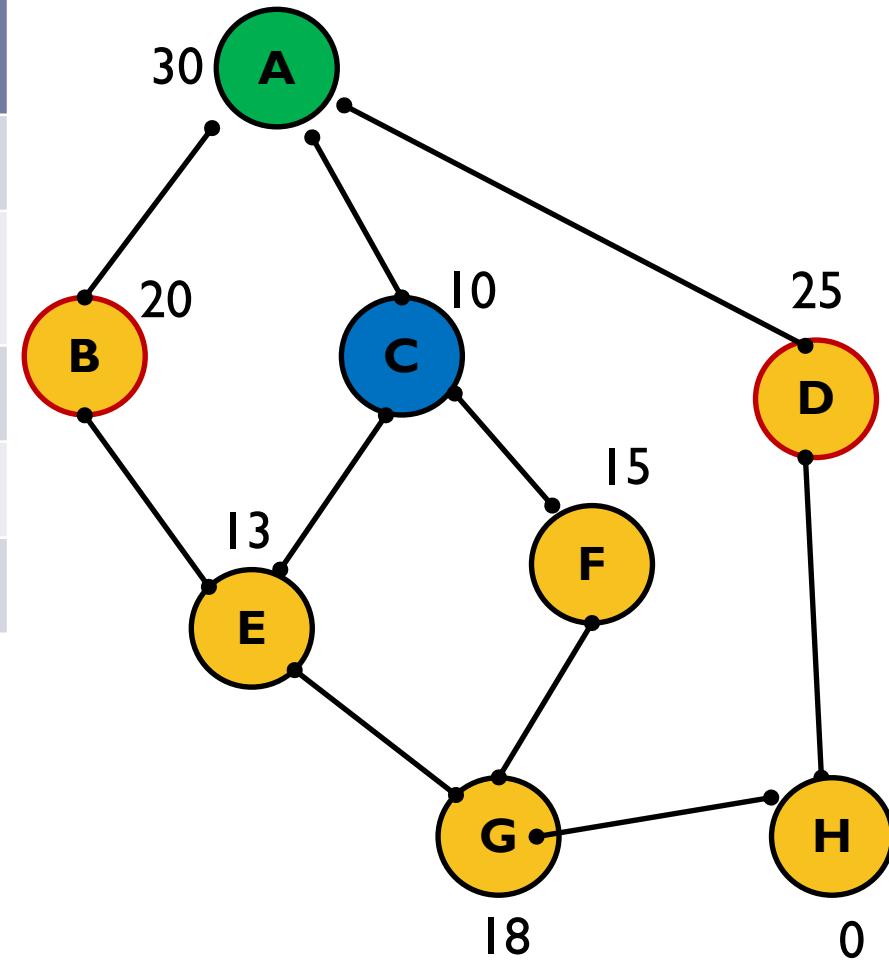
Tìm kiếm tốt nhất đầu tiên

Bước lặp	u	kề(u)	OPEN
0			A^{30}
I	A	B^{20}, C^{10}, D^{25}	C^{10}, B^{20}, D^{25}



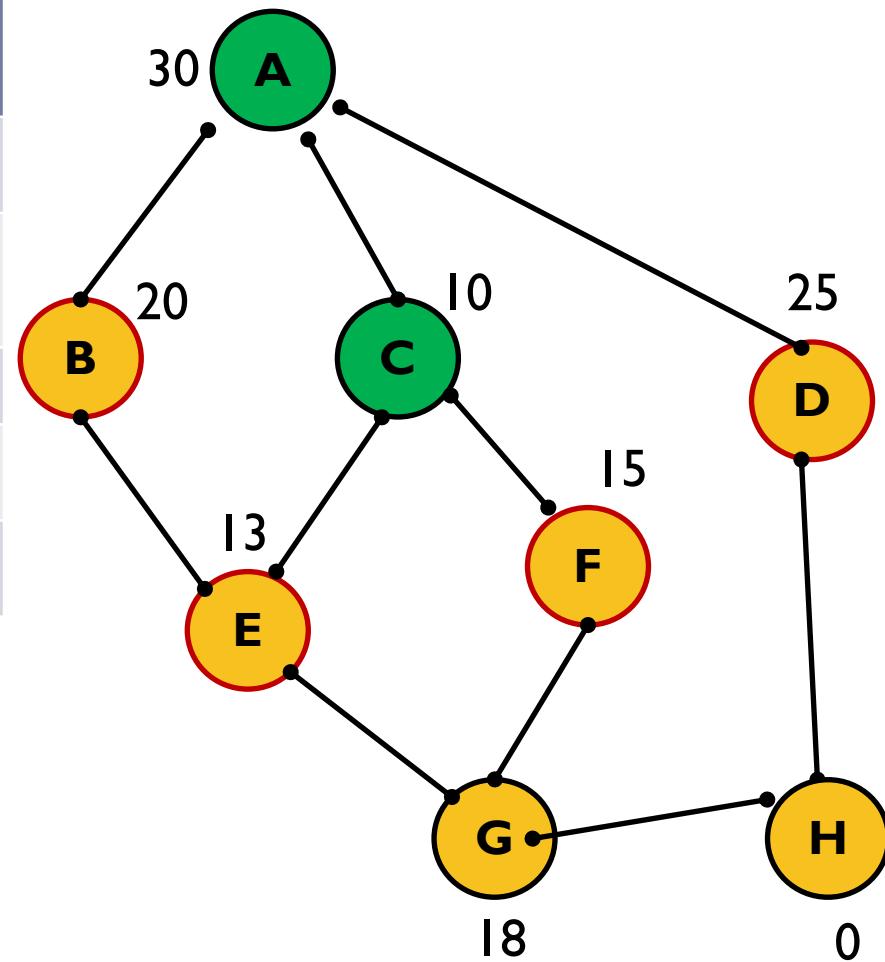
Tìm kiếm tốt nhất đầu tiên

Bước lặp	u	kề(u)	OPEN
0			A ³⁰
1	A	B ²⁰ , C ¹⁰ , D ²⁵	C ¹⁰ , B ²⁰ , D ²⁵
2	C		B ²⁰ , D ²⁵



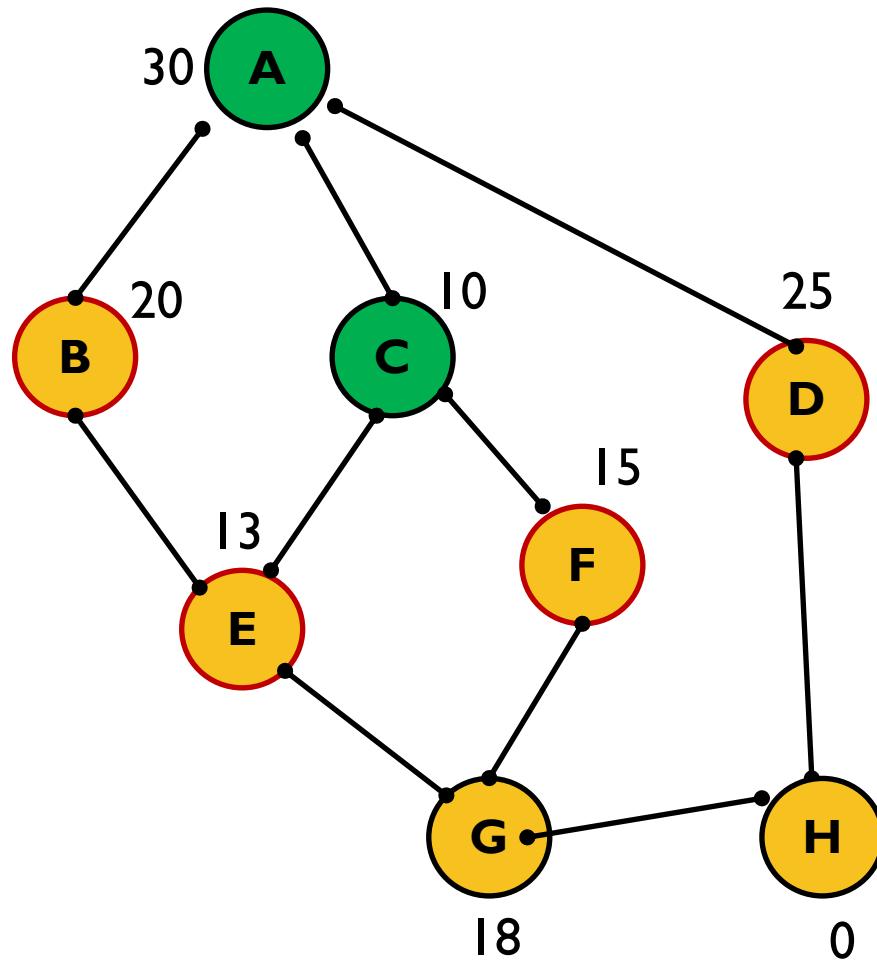
Tìm kiếm tốt nhất đầu tiên

Bước lặp	u	kề(u)	OPEN
0			A^{30}
1	A	B^{20}, C^{10}, D^{25}	C^{10}, B^{20}, D^{25}
2	C	E^{13}, F^{15}	B^{20}, D^{25}



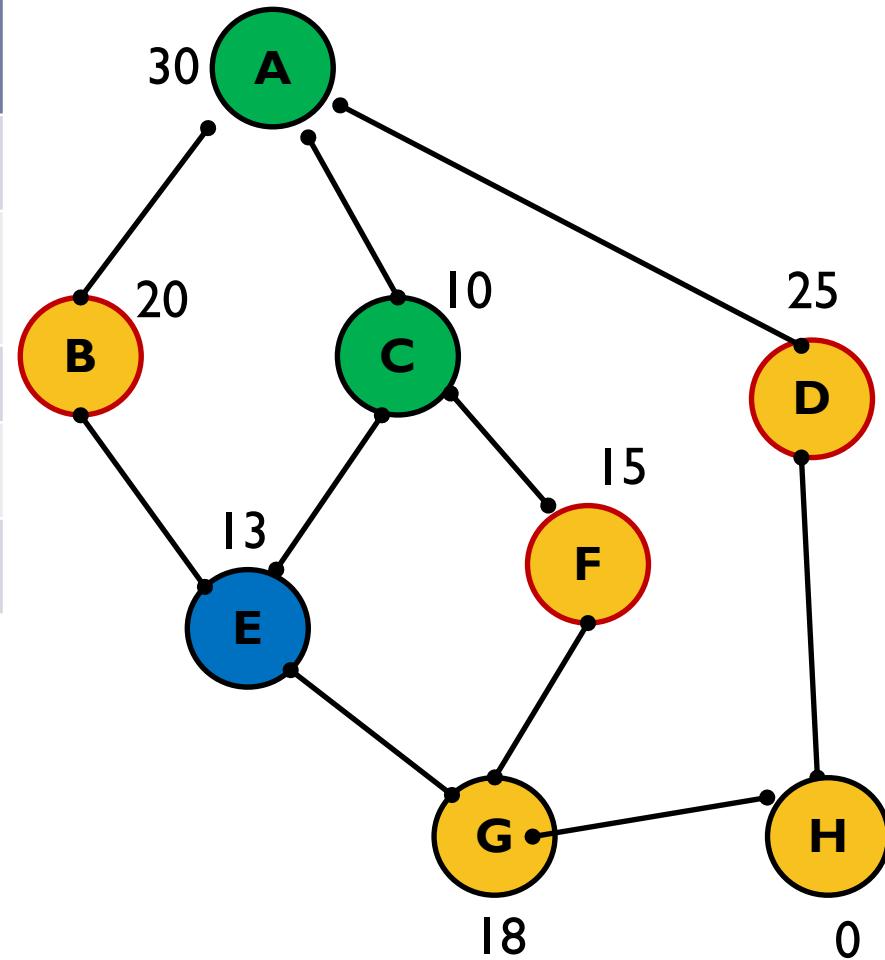
Tìm kiếm tốt nhất đầu tiên

Bước lặp	u	kề(u)	OPEN
0			A^{30}
1	A	B^{20}, C^{10}, D^{25}	C^{10}, B^{20}, D^{25}
2	C	E^{13}, F^{15}	$E^{13}, F^{15}, B^{20}, D^{25}$



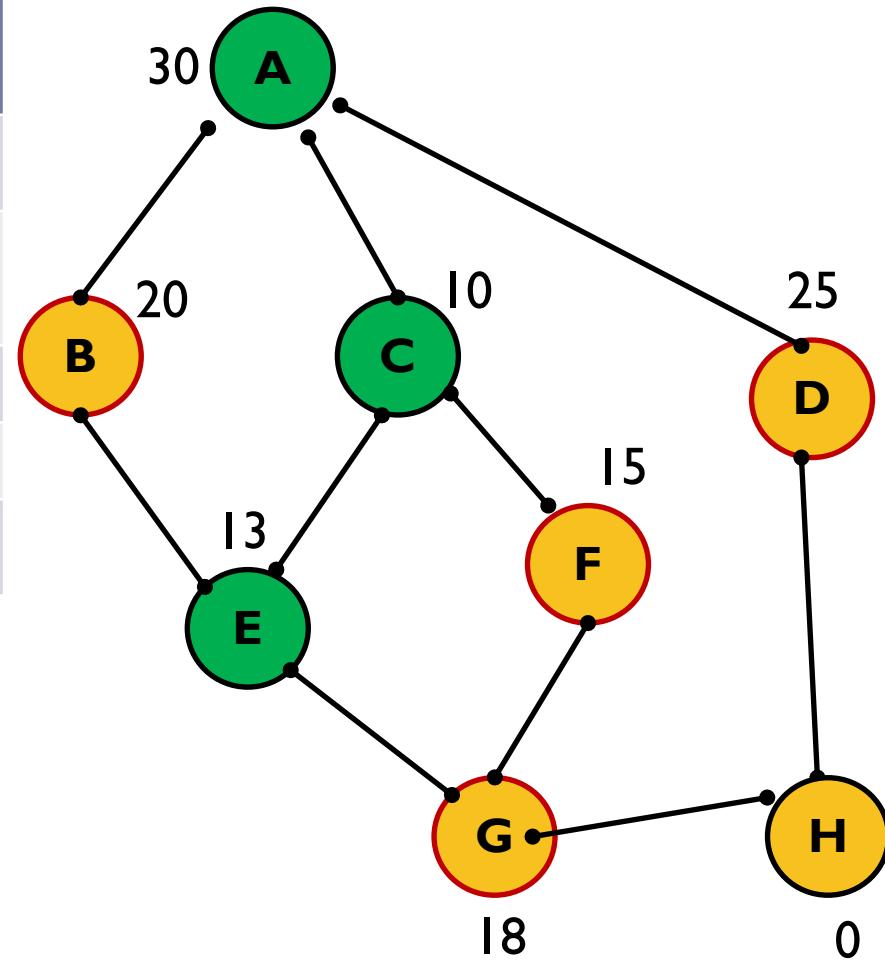
Tìm kiếm tốt nhất đầu tiên

Bước lắp	u	kề(u)	OPEN
0			A ³⁰
1	A	B ²⁰ , C ¹⁰ , D ²⁵	C ¹⁰ , B ²⁰ , D ²⁵
2	C	E ¹³ , F ¹⁵	E ¹³ , F ¹⁵ , B ²⁰ , D ²⁵
3	E	G ¹⁸ , B ²⁰	F ¹⁵ , B ²⁰ , D ²⁵



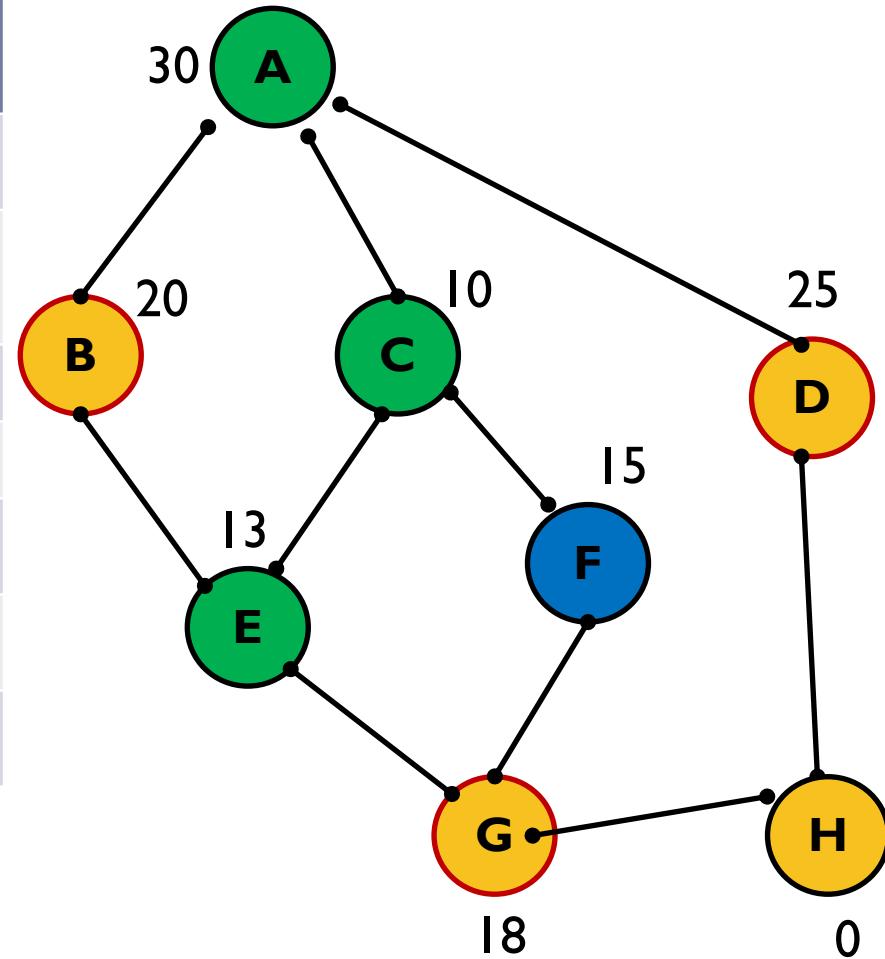
Tìm kiếm tốt nhất đầu tiên

Bước lắp	u	kề(u)	OPEN
0			A ³⁰
1	A	B ²⁰ , C ¹⁰ , D ²⁵	C ¹⁰ , B ²⁰ , D ²⁵
2	C	E ¹³ , F ¹⁵	E ¹³ , F ¹⁵ , B ²⁰ , D ²⁵
3	E	G ¹⁸ , B ²⁰	F ¹⁵ , G ¹⁸ , B ²⁰ , D ²⁵



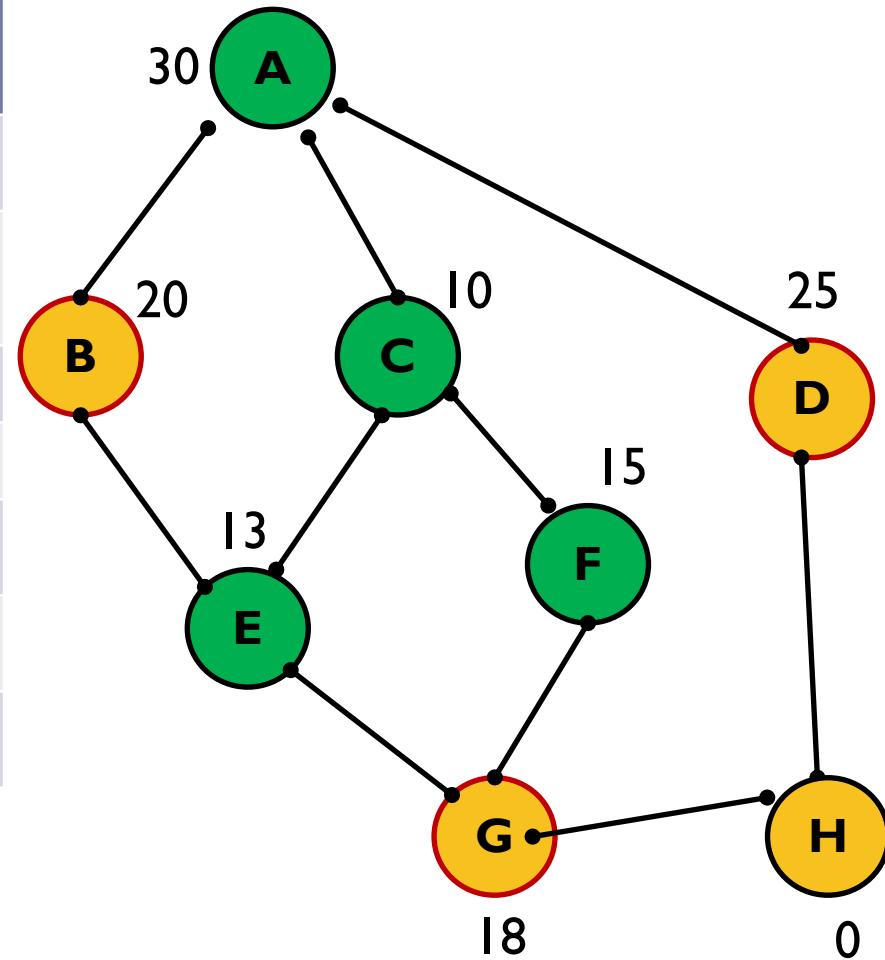
Tìm kiếm tốt nhất đầu tiên

Bước lắp	u	kề(u)	OPEN
0			A^{30}
1	A	B^{20}, C^{10}, D^{25}	C^{10}, B^{20}, D^{25}
2	C	E^{13}, F^{15}	$E^{13}, F^{15}, B^{20}, D^{25}$
3	E	G^{18}, B^{20}	$F^{15}, G^{18}, B^{20}, D^{25}$
4	F		G^{18}, B^{20}, D^{25}



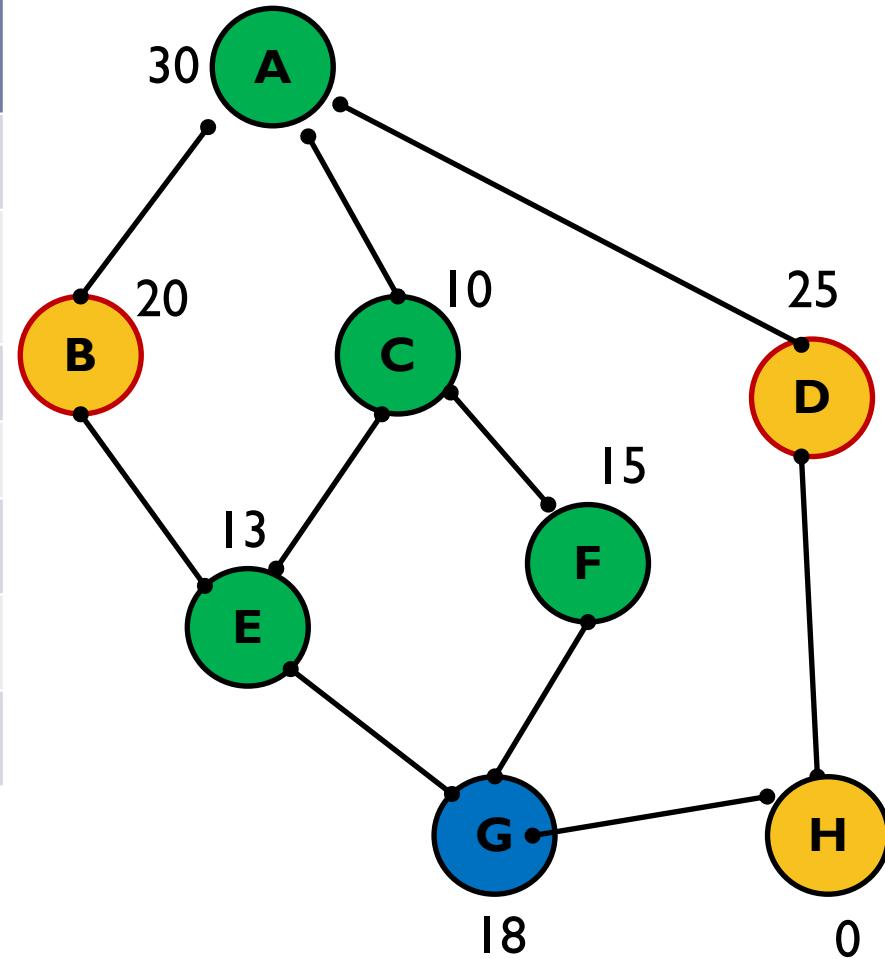
Tìm kiếm tốt nhất đầu tiên

Bước lắp	u	kề(u)	OPEN
0			A^{30}
1	A	B^{20}, C^{10}, D^{25}	C^{10}, B^{20}, D^{25}
2	C	E^{13}, F^{15}	$E^{13}, F^{15}, B^{20}, D^{25}$
3	E	G^{18}, B^{20}	$F^{15}, G^{18}, B^{20}, D^{25}$
4	F		G^{18}, B^{20}, D^{25}



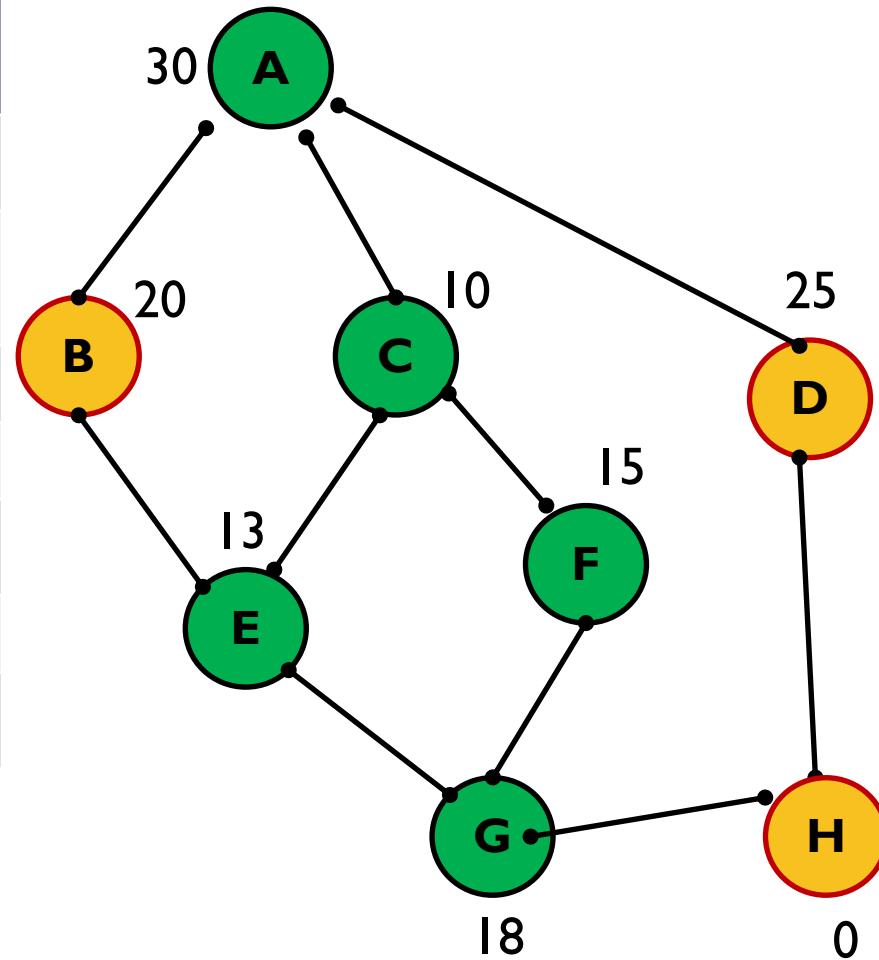
Tìm kiếm tốt nhất đầu tiên

Bước lắp	u	kề(u)	OPEN
0			A ³⁰
1	A	B ²⁰ ,C ¹⁰ ,D ²⁵	C ¹⁰ , B ²⁰ ,D ²⁵
2	C	E ¹³ ,F ¹⁵	E ¹³ ,F ¹⁵ ,B ²⁰ ,D ²⁵
3	E	G ¹⁸ , B ²⁰	F ¹⁵ , G ¹⁸ ,B ²⁰ ,D ²⁵
4	F	G ¹⁸ ,	G ¹⁸ ,B ²⁰ ,D ²⁵
5	G		B ²⁰ ,D ²⁵



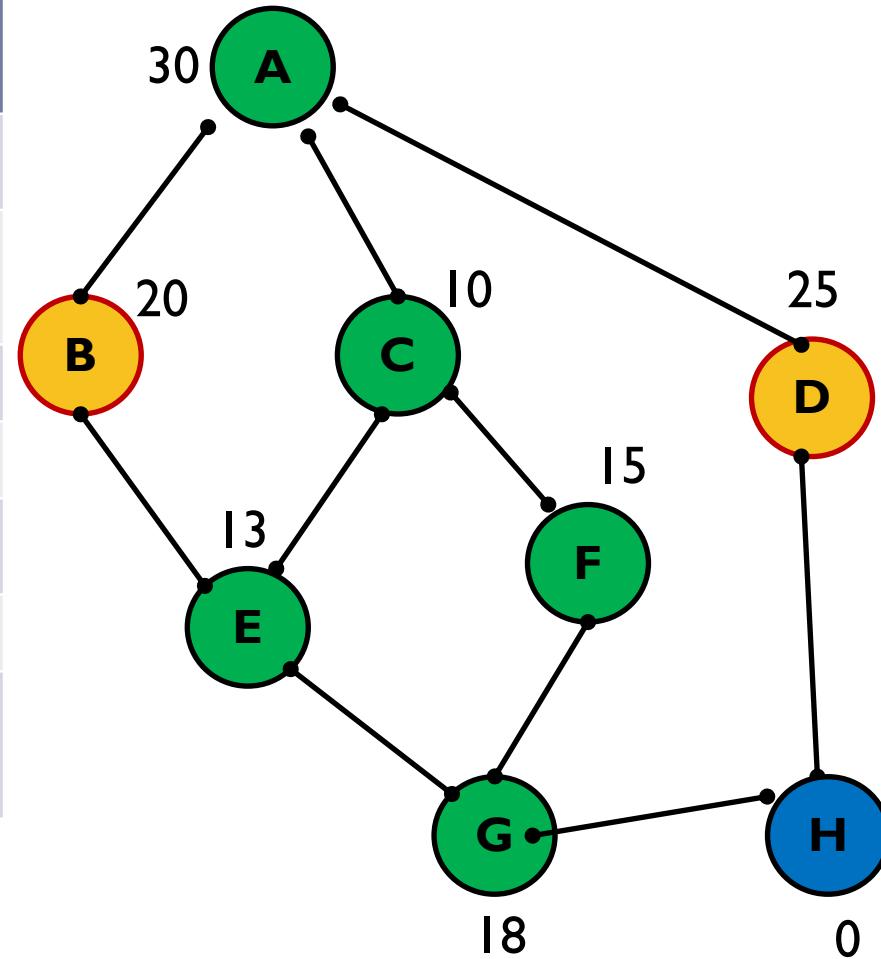
Tìm kiếm tốt nhất đầu tiên

Bước lắp	u	kề(u)	OPEN
0			A^{30}
1	A	B^{20}, C^{10}, D^{25}	C^{10}, B^{20}, D^{25}
2	C	E^{13}, F^{15}	$E^{13}, F^{15}, B^{20}, D^{25}$
3	E	G^{18}, B^{20}	$F^{15}, G^{18}, B^{20}, D^{25}$
4	F	G^{18}	G^{18}, B^{20}, D^{25}
5	G	H^0	H^0, B^{20}, D^{25}



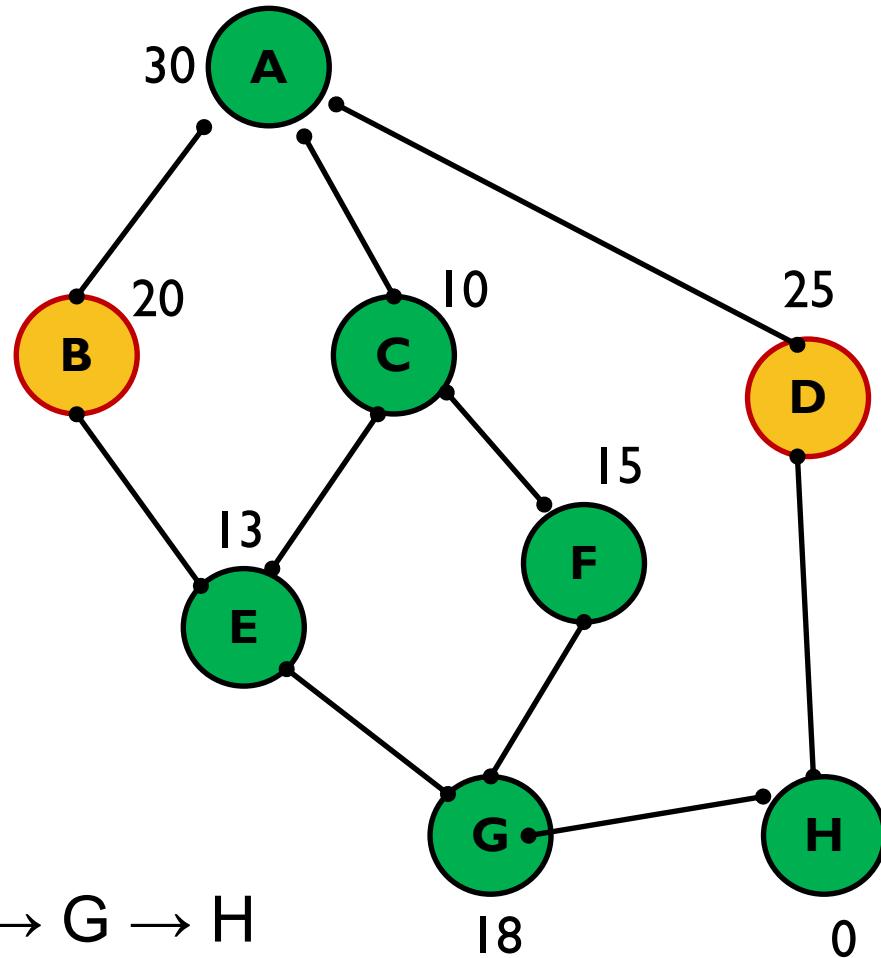
Tìm kiếm tốt nhất đầu tiên

Bước lắp	u	kề(u)	OPEN
0			A ³⁰
1	A	B ²⁰ , C ¹⁰ , D ²⁵	C ¹⁰ , B ²⁰ , D ²⁵
2	C	E ¹³ , F ¹⁵	E ¹³ , F ¹⁵ , B ²⁰ , D ²⁵
3	E	G ¹⁸ , B ²⁰	F ¹⁵ , G ¹⁸ , B ²⁰ , D ²⁵
4	F	G ¹⁸	G ¹⁸ , B ²⁰ , D ²⁵
5	G	H ⁰	H ⁰ , B ²⁰ , D ²⁵
6	H ≡ đích		B ²⁰ , D ²⁵



Tìm kiếm tốt nhất đầu tiên

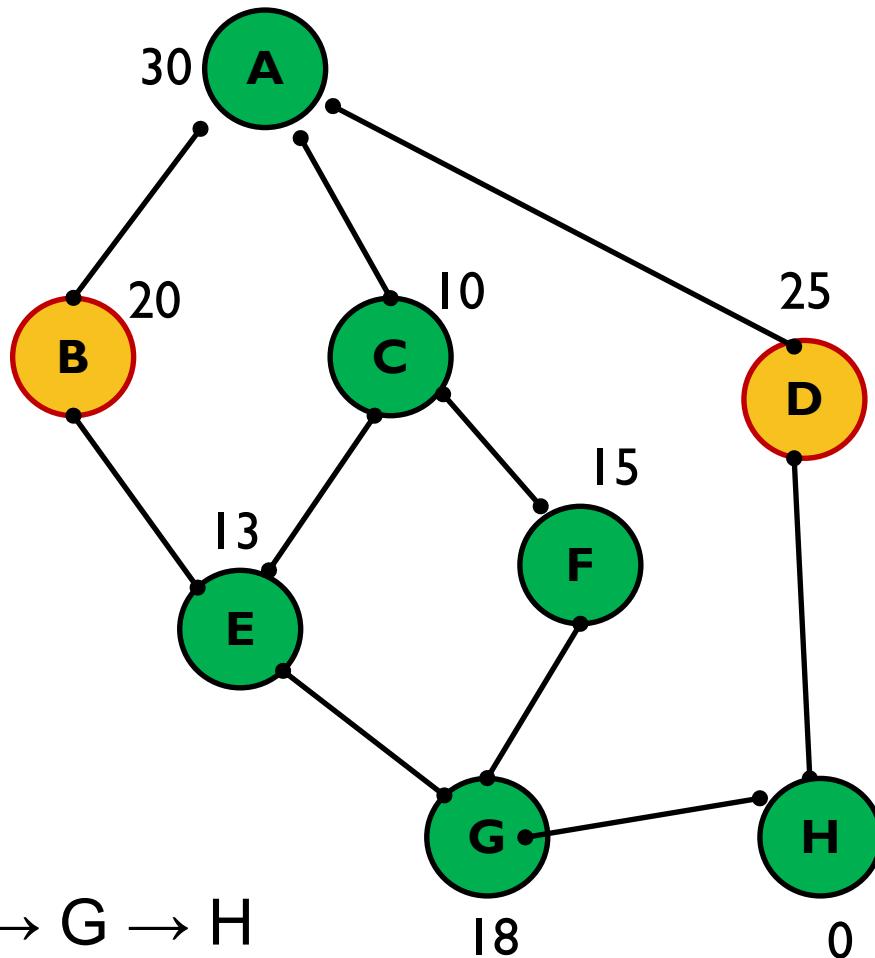
Bước lặp	u	kề(u)	OPEN
0			A^{30}
1	A	B^{20}, C^{10}, D^{25}	C^{10}, B^{20}, D^{25}
2	C	E^{13}, F^{15}	$E^{13}, F^{15}, B^{20}, D^{25}$
3	E	G^{18}, B^{20}	$F^{15}, G^{18}, B^{20}, D^{25}$
4	F	G^{18}	G^{18}, B^{20}, D^{25}
5	G	H^0	H^0, B^{20}, D^{25}
6	H ≡ đích		B^{20}, D^{25}



Quá trình duyệt : $A \rightarrow C \rightarrow E \rightarrow F \rightarrow G \rightarrow H$

Tìm kiếm tốt nhất đầu tiên

Bước lặp	u	kề(u)	OPEN
0			A^{30}
1	A	B^{20}, C^{10}, D^{25}	C^{10}, B^{20}, D^{25}
2	C	E^{13}, F^{15}	$E^{13}, F^{15}, B^{20}, D^{25}$
3	E	G^{18}, B^{20}	$F^{15}, G^{18}, B^{20}, D^{25}$
4	F	G^{18}	G^{18}, B^{20}, D^{25}
5	G	H^0	H^0, B^{20}, D^{25}
6	H ≡ đích		B^{20}, D^{25}



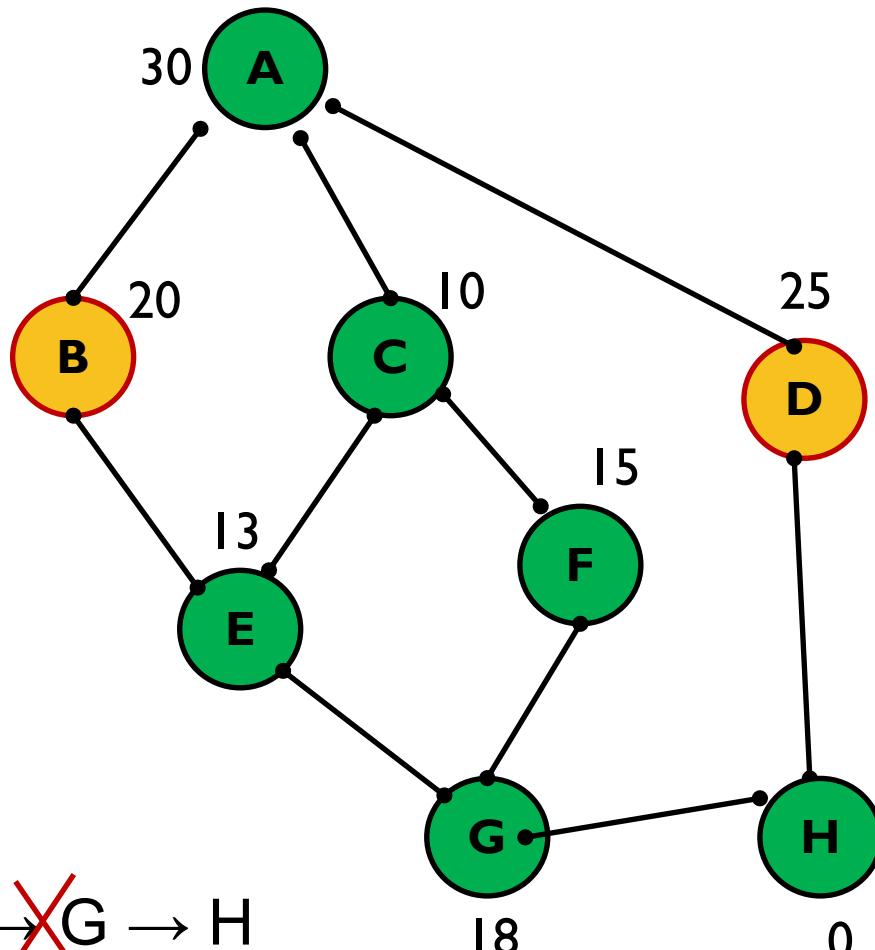
Quá trình duyệt : $A \rightarrow C \rightarrow E \rightarrow F \rightarrow G \rightarrow H$



parent

Tìm kiếm tốt nhất đầu tiên

Bước lặp	u	kề(u)	OPEN
0			A^{30}
1	A	B^{20}, C^{10}, D^{25}	C^{10}, B^{20}, D^{25}
2	C	E^{13}, F^{15}	$E^{13}, F^{15}, B^{20}, D^{25}$
3	E	G^{18}, B^{20}	$F^{15}, G^{18}, B^{20}, D^{25}$
4	F	G^{18}	G^{18}, B^{20}, D^{25}
5	G	H^0	H^0, B^{20}, D^{25}
6	H ≡ đích		B^{20}, D^{25}



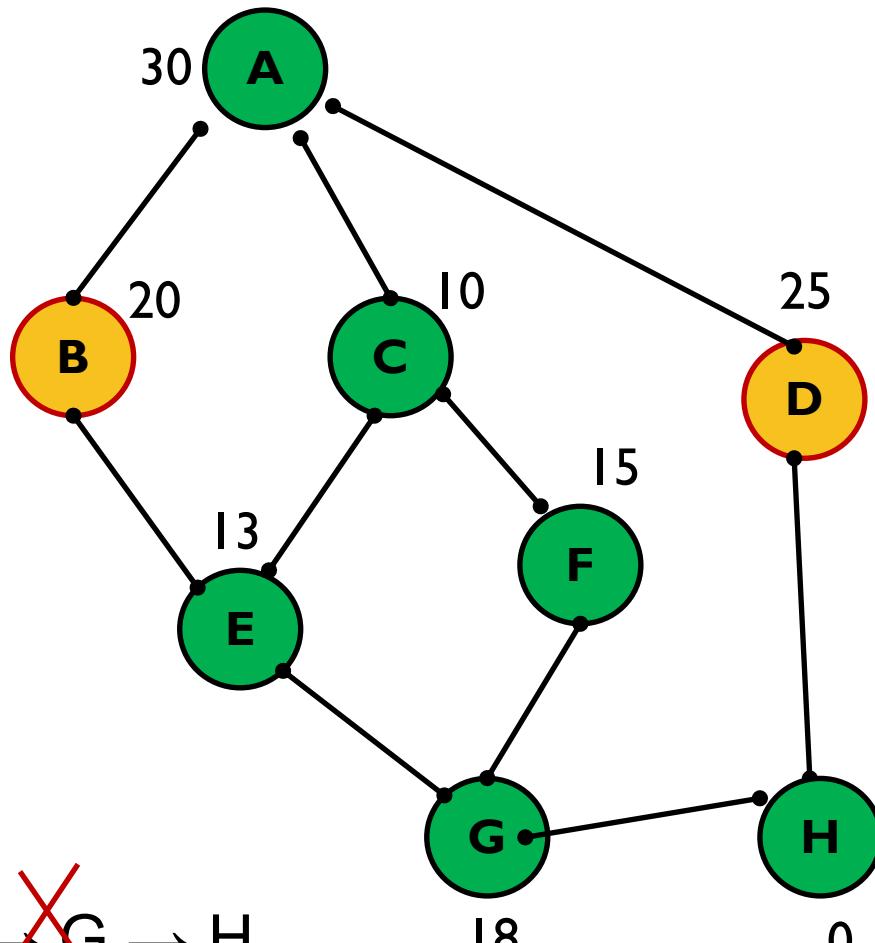
Quá trình duyệt : $A \rightarrow C \rightarrow E \rightarrow F \rightarrow G \rightarrow H$



parent

Tìm kiếm tốt nhất đầu tiên

Bước lắp	u	kề(u)	OPEN
0			A ³⁰
1	A	B ²⁰ , C ¹⁰ , D ²⁵	C ¹⁰ , B ²⁰ , D ²⁵
2	C	E ¹³ , F ¹⁵	E ¹³ , F ¹⁵ , B ²⁰ , D ²⁵
3	E	G ¹⁸ , B ²⁰	F ¹⁵ , G ¹⁸ , B ²⁰ , D ²⁵
4	F	G ¹⁸	G ¹⁸ , B ²⁰ , D ²⁵
5	G	H ⁰	H ⁰ , B ²⁰ , D ²⁵
7	H ≡ đích	rỗng	B ²⁰ , D ²⁵



Quá trình duyệt : A → C → E → F → G → H

Đường đi : A → C → E → G → H hoặc A → C → F → G → H

Phương pháp: lần ngược từ đích về nút xuất phát theo các nút cha

Tìm kiếm tốt nhất đầu tiên

Bước lặp	u	kè(u)	OPEN
0			A ³⁰
1	A	B ²⁰ ,C ¹⁰ , D ²⁵	C ¹⁰ , B ²⁰ ,D ²⁵
2	C	E ¹³ ,F ¹⁵	E ¹³ ,F ¹⁵ ,B ²⁰ ,D ²⁵ ,
3	E	G ¹⁸ , B ²⁰	F ¹⁵ , G ¹⁸ ,B ²⁰ ,D ²⁵
4	F	G ¹⁸	G ¹⁸ ,B ²⁰ ,D ²⁵
5	G	H ⁰	H ⁰ ,B ²⁰ ,D ²⁵
7	H ≡ đích	rỗng	B ²⁰ ,D ²⁵

Loại bỏ nút dư thừa để tìm ra vết đường đi: từ bảng tìm kiếm, lần ngược từ đích về nút xuất phát theo quan hệ cha (cột u) - con (cột kè(u)):

- Cha của H là G (bước 5)
- Cha của G là F (bước 4) và/hoặc E (bước 3)
- Cha của F và E đều là C (bước 2)
- Cha của C là A (bước 1)

Phương pháp: lần ngược từ đích về nút xuất phát theo các nút cha

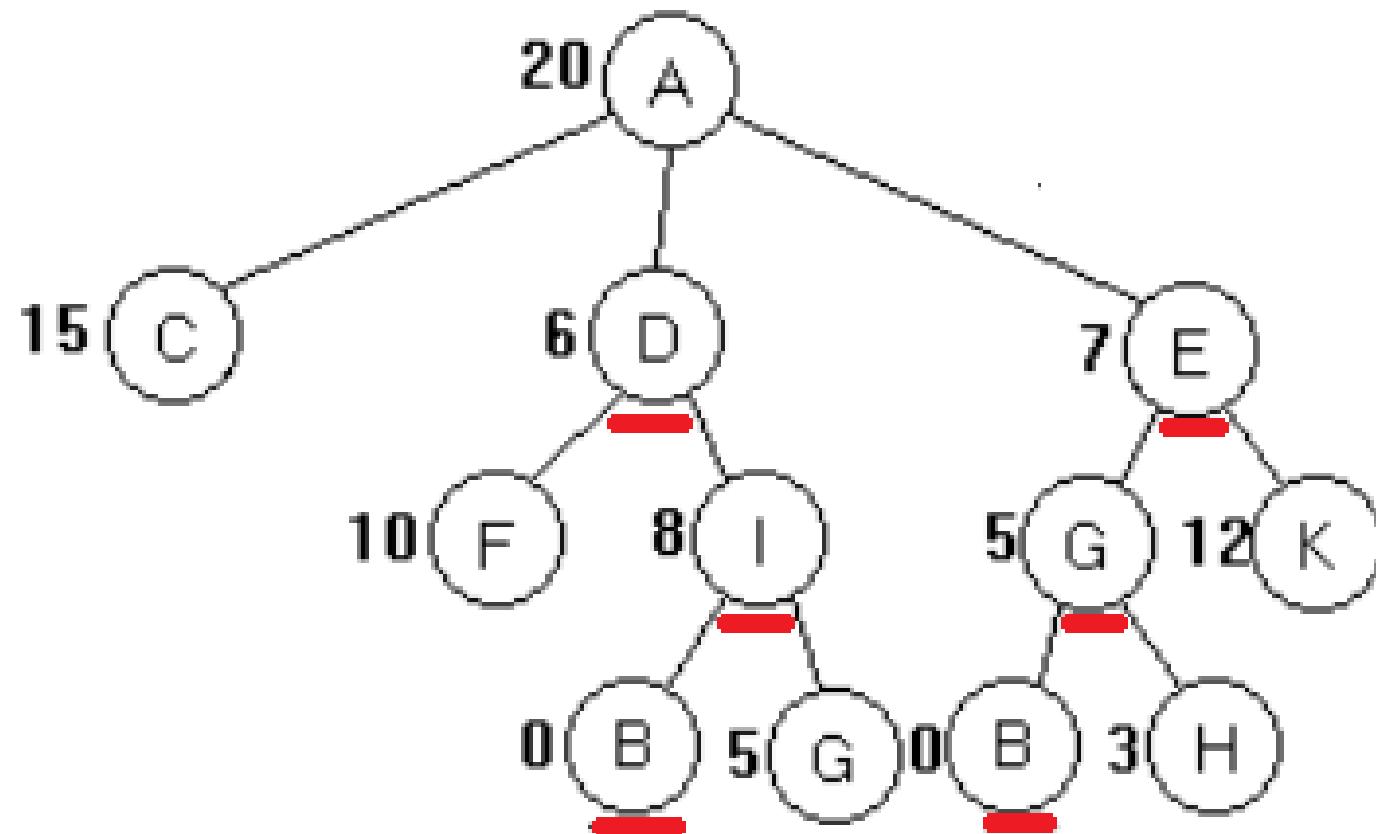
Tìm kiếm beam

- ❑ Giống tìm kiếm tốt nhất đầu tiên
- ❑ Hạn chế chỉ phát triển k đỉnh tốt nhất ở mỗi độ sâu

Tốt nhất đầu tiên BFS	Beam
Duyệt u, phát triển tất cả các đỉnh kề v của u (đưa tất cả vào OPEN)	Duyệt u, phát triển 1 số đỉnh kề v của u sao cho tổng số đỉnh ở cùng độ sâu trong OPEN là k (đưa những đỉnh n vào OPEN)

Tìm kiếm beam

💡 Ví dụ: tìm đường đi từ A → B, k = 2



Tìm kiếm leo đồi

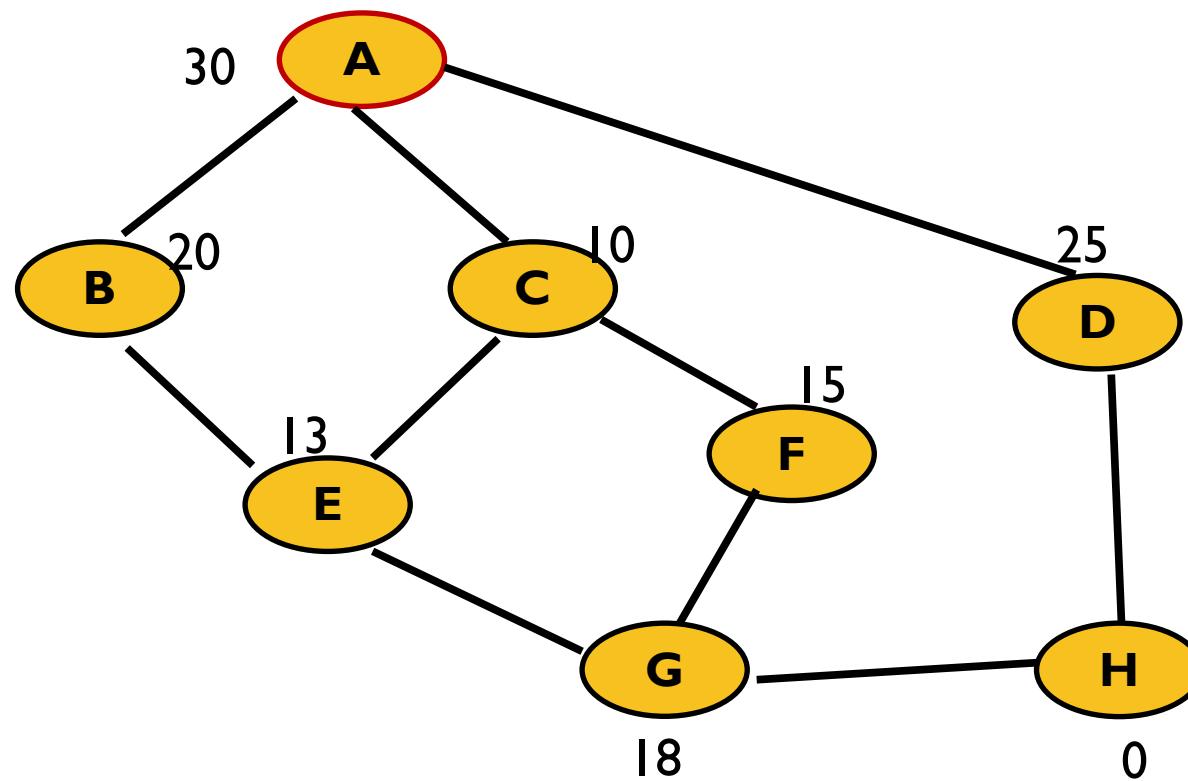
- ❑ Tìm kiếm theo độ sâu dưới định hướng của hàm đánh giá $h(u)$
- ❑ Tại mỗi bước của tìm kiếm
 - Chọn đỉnh u ở đầu danh sách OPEN để duyệt
 - Sau khi duyệt đỉnh u :
 - Sắp xếp danh sách các đỉnh kề của u theo thứ tự tăng dần của hàm đánh giá
 - Chèn danh sách kề này vào đầu OPEN

Tìm kiếm leo đồi

- Procedure Hill_Climbing_Search
- begin
 - 1. Khởi tạo Open = {trạng thái ban đầu};
 - 2. while true do
 - 2.1 If (Open rỗng) then
 - {thông báo thất bại; stop};
 - 2.2 Loại trạng thái u ở đầu danh sách Open;
 - 2.3 If (u là trạng thái kết thúc) then
 - {thông báo thành công; stop};
 - 2.4 for (mỗi v kề u) do thêm v vào danh sách L;
 - 2.5 Sắp xếp danh sách L theo thứ tự tăng dần của hàm đánh giá;
 - 2.5 Chèn L vào đầu OPEN;

Tìm kiếm leo đồi

Ví dụ: tìm đường đi ngắn nhất từ A→H bằng tìm kiếm leo đồi?



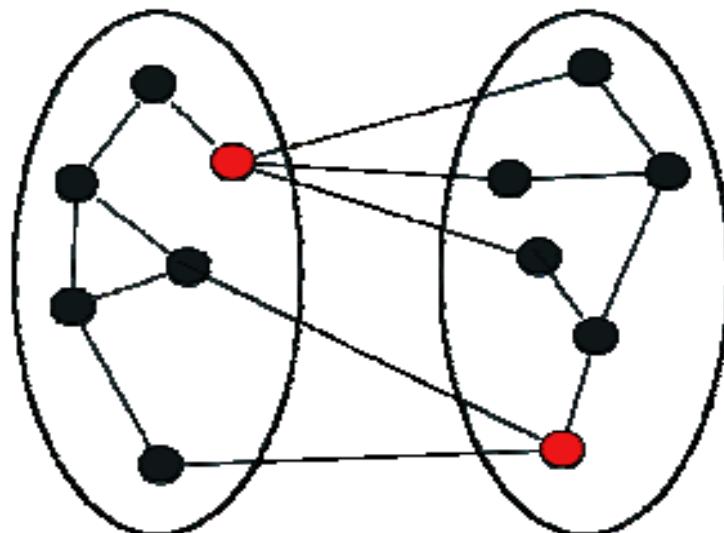
Tìm kiếm tối ưu

Tìm kiếm tối ưu

- ❑ Trong thực tế, ta thường không chỉ quan tâm đến việc tìm ra nghiệm mà còn phải quan tâm đến việc nghiệm đó có tối ưu hay không.
 - Tìm đường đi ngắn nhất : tính tới chi phí một hành trình
 - Trò chơi 8 số yêu cầu thời gian ngắn nhất để đưa về trạng thái đích : tính cả số bước đã đi
- ❑ Trong các chiến lược tìm kiếm mù và tìm kiếm kinh ngh, chúng ta chưa quan tâm đến độ dài hay chi phí của đường đi nghiệm.

Tìm kiếm tối ưu

- Bài toán Graph partition
- Cho 1 đồ thi, hãy tìm cách phân chia đồ thị thành n phần có kích thước (số đỉnh) bằng nhau sao cho số cạnh nối giữa các phần là ít nhất



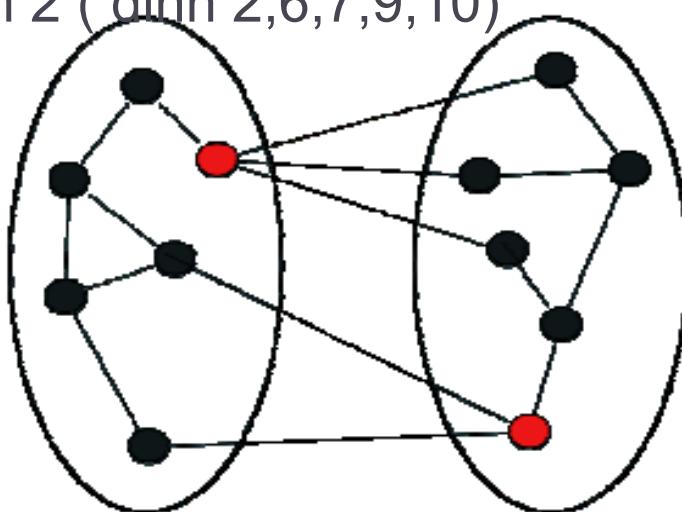
Tìm kiếm tối ưu

- ℳ Mỗi cách phân chia $G(V, E) \rightarrow \{G_1(V_1, E_1), G_2(V_2, E_2)\}$ là 1 trạng thái

Biểu diễn trạng thái bởi 1 mảng $[a_1, a_2, \dots, a_n]$

$[0100011011] \equiv$ nhóm 1 (đỉnh 1,3,4,5,8) – nhóm 2 (đỉnh 2,6,7,9,10)

- ℳ Hàm đánh giá mỗi cách phân chia
 $F() = |V_1 - V_2| +$ số cạnh nối



- ℳ Tìm kiếm cách phân chia tốt nhất là tìm ra trạng thái u sao cho hàm $f(u)$ đạt cực tiểu

Tìm kiếm tối ưu

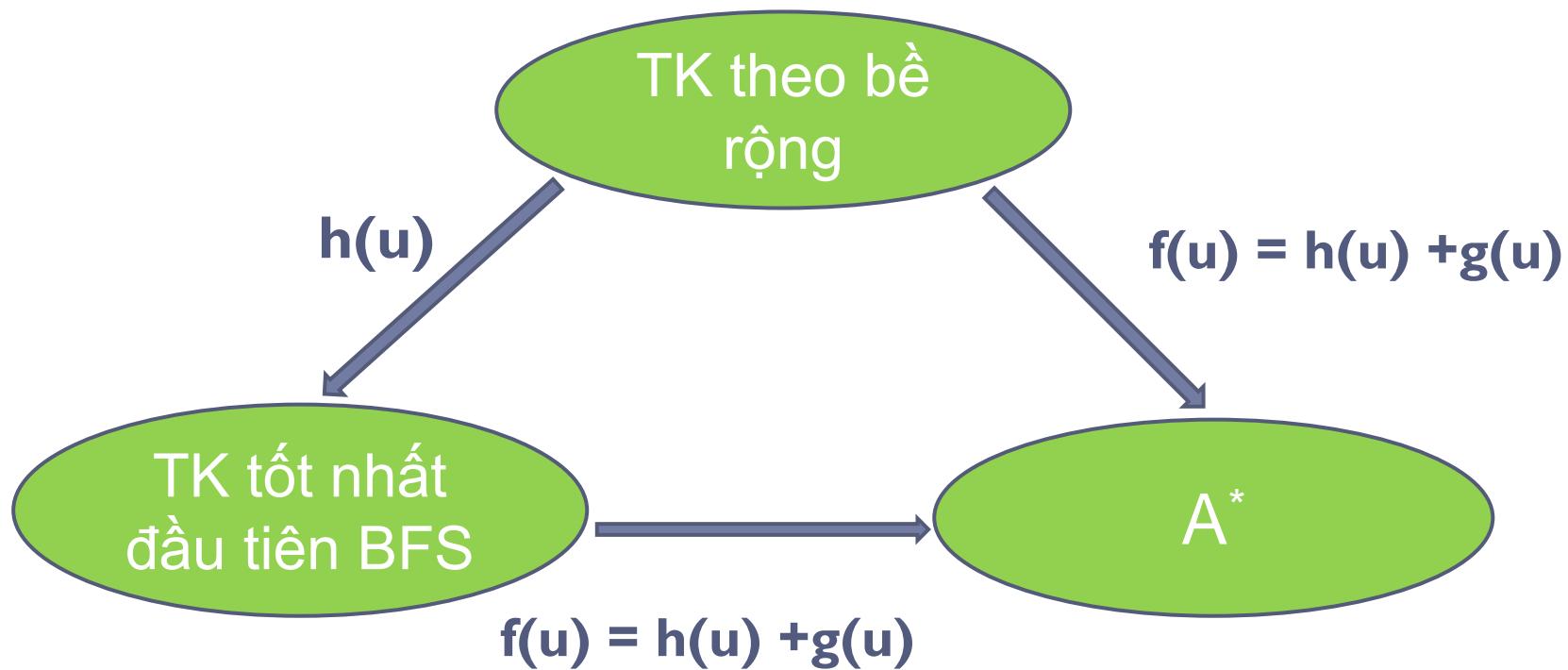
Tìm kiếm kinh nghiệm	Tìm kiếm tối ưu
Hàm đánh giá : $f(u) = h(u)$	Hàm đánh giá đầy đủ : $f(u) = h(u) + g(u)$
Hàm đánh giá định hướng việc tìm kiếm (chọn đỉnh, rẽ nhánh)	Tối ưu hóa hàm đánh giá Hàm đánh giá $\rightarrow \min(\max)$

Tìm kiếm tối ưu

- Các kỹ thuật tìm đường đi ngắn nhất : A*, nhánh cận
- Các kỹ thuật tìm kiếm cục bộ: TK leo đồi, gradient, mô phỏng luyễn kim
- Tìm kiếm bắt chước sự tiến hóa : thuật toán di truyền

Thuật toán A*

- Là thuật toán tìm kiếm tốt nhất đầu tiên với hàm đánh giá $h(u) + g(u)$ thay vì $h(u)$



Procedure A*

begin

1. Khởi tạo danh sách OPEN = {trạng thái ban đầu};

2. while true do

2.1 if (Open rỗng) then

{thông báo tìm kiếm thất bại; **stop**};

2.2 Loại trạng thái u ở đầu danh sách OPEN;

2.3 if u là trạng thái kết thúc then

{thông báo tìm kiếm thành công; **stop**};

2.4 for mỗi v kề u

$g(v) = g(u) + k(u,v)$; // $k(u,v)$ là chi phí thực tế để đi từ u tới v, được cho trước

$f(v) = g(v) + h(v)$; // **ưu tiên g nhỏ, cho phép duyệt lại v nếu f mới tốt hơn**

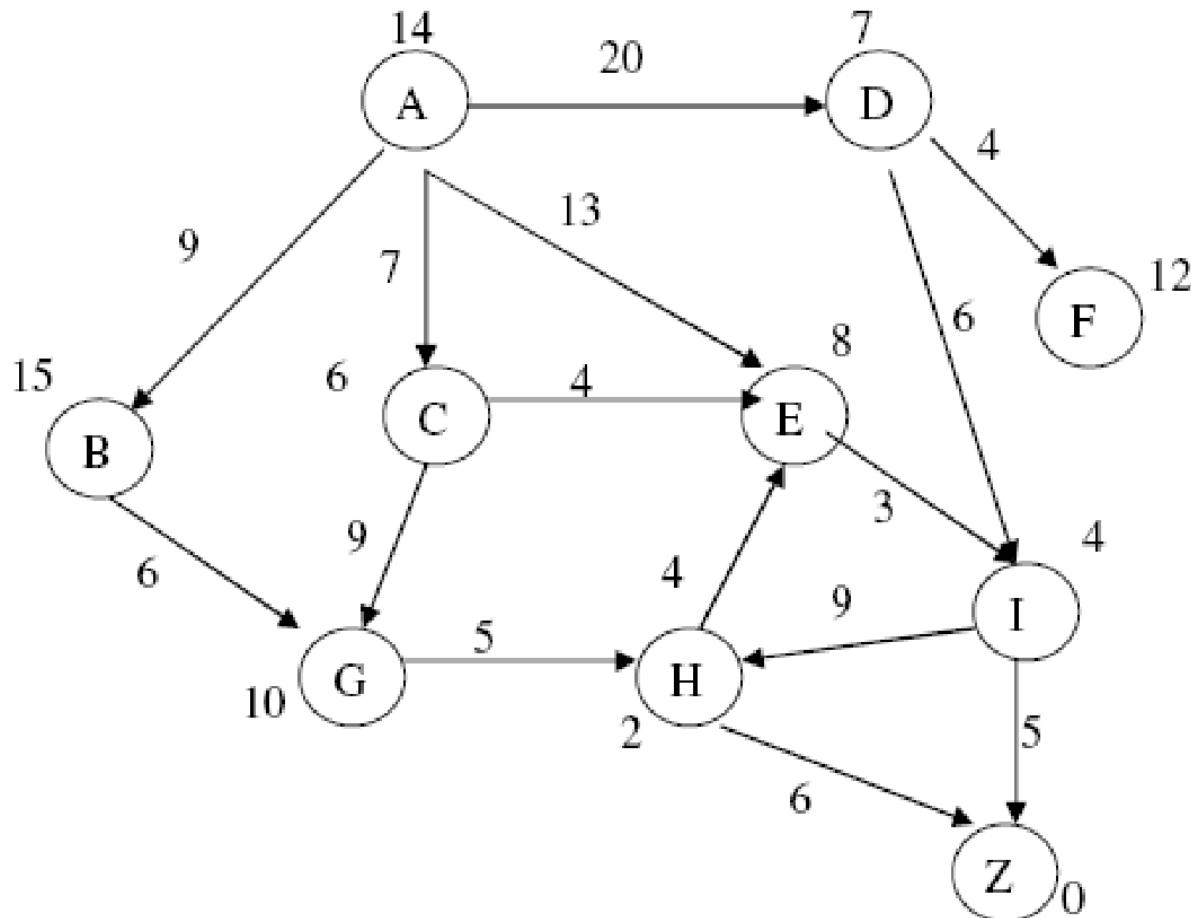
Chèn v vào OPEN sao cho OPEN được sắp xếp theo thứ tự tăng dần của f;

end

Thuật toán A*

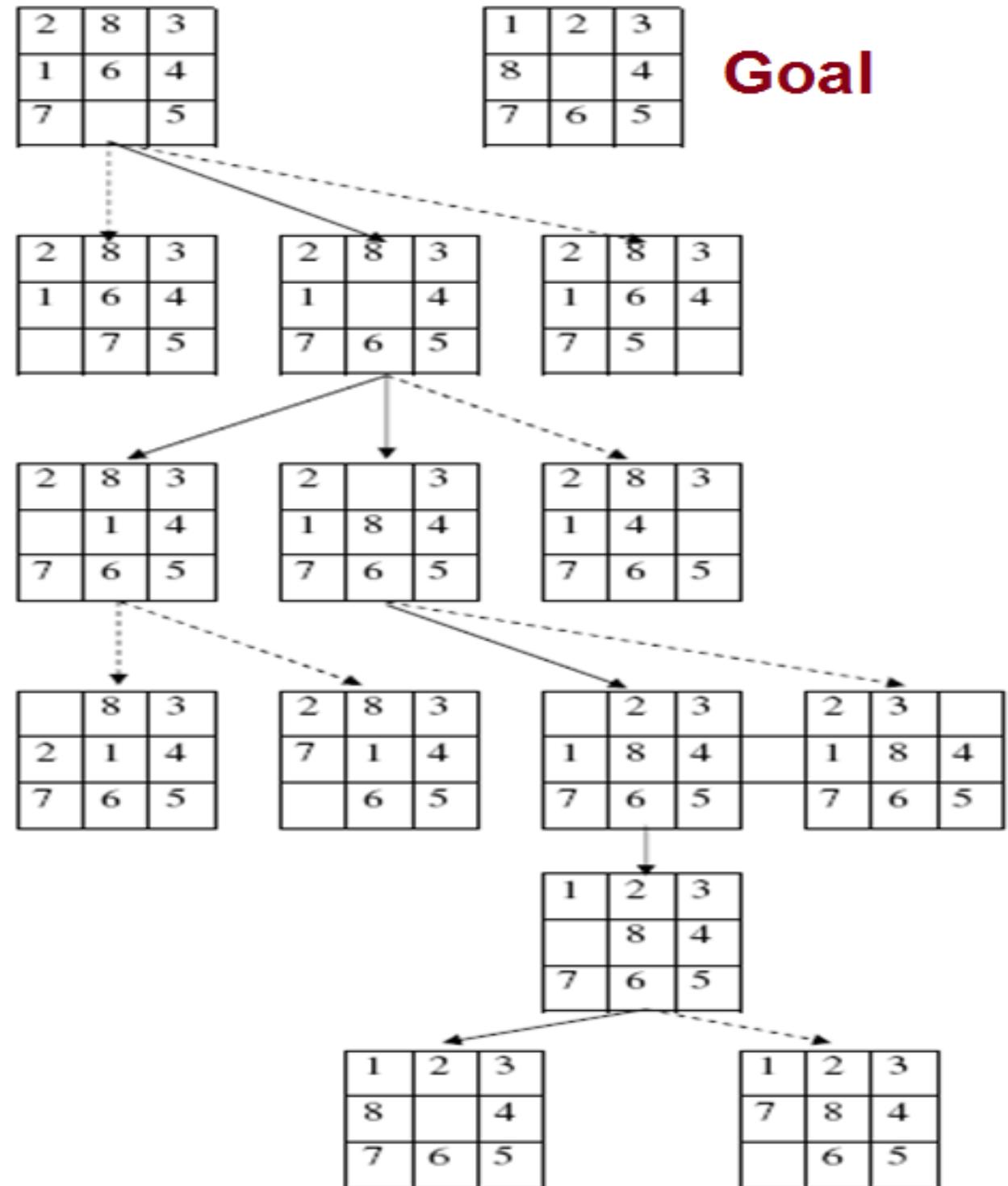
Thuật toán A*

- Tìm đường đi ngắn nhất từ A tới Z bằng A*:
 - Giá trị gắn tại mỗi đỉnh là $h(u)$
 - Giá trị gắn tại mỗi cạnh là chi phí để chuyển trạng thái $k(u,v)$
 - Lưu ý: khi $g+h$ như nhau thì ưu tiên g bé hơn



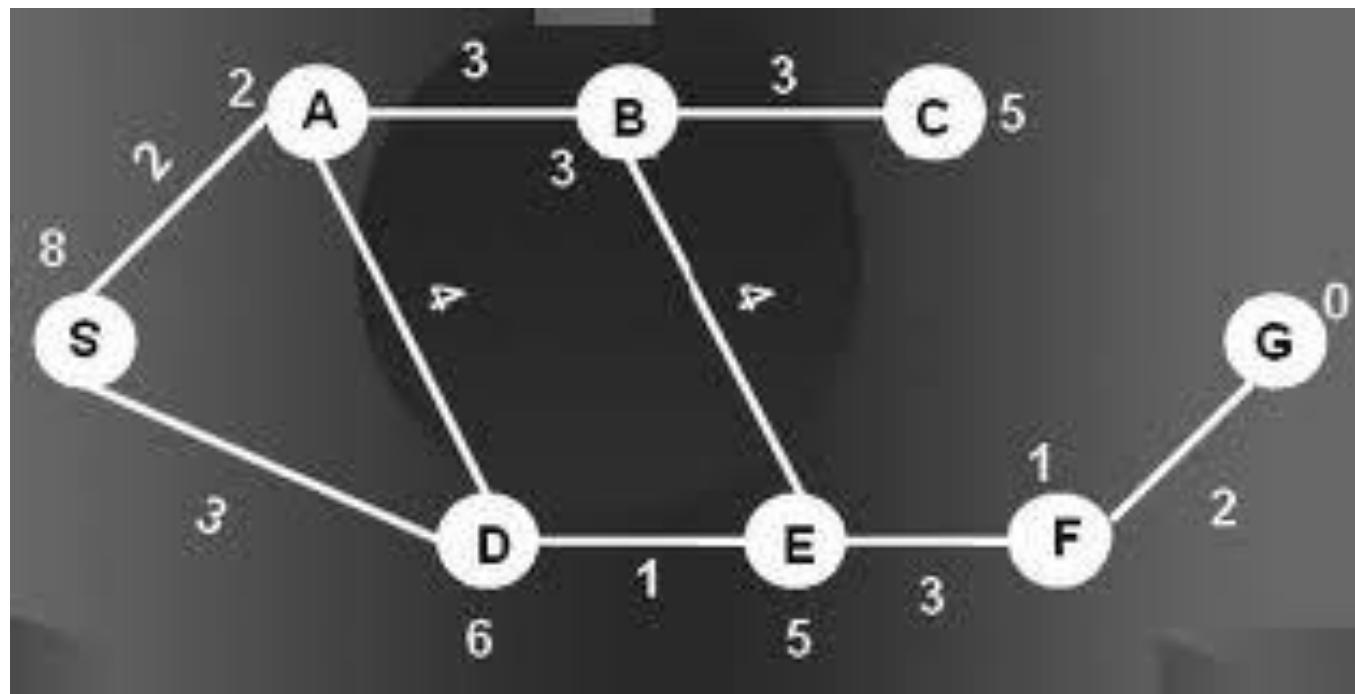
Thuật toán A*

- Bài tập I:
- Tìm cách chơi nhanh nhất với hàm heuristic h1,h2



Thuật toán A*

- Bài tập 2: Cho sơ đồ nối các thành phố. Tìm đường đi từ S tới G bằng A*



Thuật toán A*

- ② Trường hợp đặc biệt : $h(u) = 0$ với mọi u
 - ③ $f(u) = g(u) + h(u) = g(u)$
 - ④ Khi đó, A* là TK tốt nhất đầu tiên với hàm đánh giá $g(u)$
- ② Định lý: Nếu $h(u)$ là chấp nhận được, thì A* là TK tối ưu
- ② Chứng minh:
 - ③ Giả sử OPEN chứa:
 - Đích tối ưu cục bộ G' và
 - Trạng thái u nằm trên đường tới đích tối ưu toàn cục G
 - ④ Ta có: $g(u)+h(u) < g(u)+h^*(u)$ vì h là chấp nhận được
 - ④ Suy ra $f(u) < g(G)$
 - ④ Mà $g(G) < g(G')$ vì G' là tối ưu cục bộ
 - ④ Suy ra $f(u) < g(G')$
 - ④ Hay $f(u) < f(G')$ vì $h(G') = 0$
 - ④ A* không bao giờ chọn G' để phát triển

Thuật toán nhánh cận

- Branch-and-bound search
- Thuật toán nhánh cận là sự cải tiến của thuật toán tìm kiếm leo đồi
 - TK leo đồi: gặp nghiệm thì dừng tìm kiếm. Kết quả là 1 tối ưu cục bộ
 - TK nhánh cận: gặp nghiệm thì vẫn tiếp tục tìm kiếm nghiệm tốt hơn. Kết quả là nghiệm tối ưu
- Mục đích: chuyển từ tìm kiếm nghiệm tối ưu cục bộ sang tìm kiếm tốt nhất toàn cục

Thuật toán nhánh cành

Khi sử dụng hàm đánh giá $h(u)$ chấp nhận được, dọc theo 1 đường đi, f luôn tăng dần

Thật vậy, ta có :

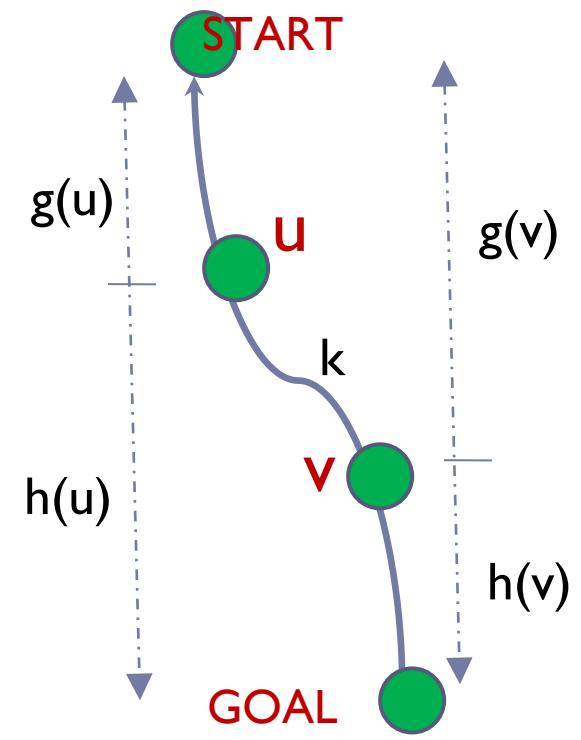
$$\text{Mà } h(u \rightarrow \text{GOAL}) \leq h(u \rightarrow v) + h(v \rightarrow \text{GOAL})$$

$$\text{Tức } h(u) \leq k + h(v)$$

$$\Leftrightarrow g(u) + h(u) \leq g(u) + k + h(v)$$

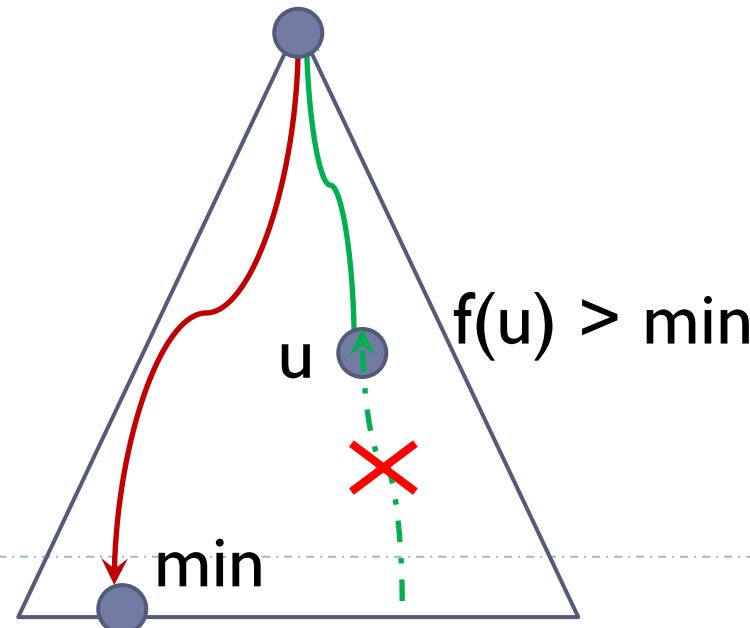
$$\Leftrightarrow g(u) + h(u) \leq g(v) + h(v)$$

$$\Leftrightarrow f(u) \leq f(v)$$



Thuật toán nhánh cận

- min : chi phí ngắn nhất tạm thời tìm thấy từ lúc bắt đầu tìm kiếm
- Khi xét nút u, nếu $f(u) > \text{min}$ thì sẽ cắt bỏ nhánh con của u
 - Toàn bộ các nút con/cháu v của u đều có $f(v) > f(u) > \text{min}$ nên không thể là nghiệm tốt hơn (tối ưu hơn)
- Nếu tìm thấy 1 đường đi mới tốt hơn đường đi tốt nhất tạm thời (có chi phí min), cập nhật lại min và đường đi tốt nhất tạm thời đó



Thuật toán nhánh cận

- Branch-and-bound search
- Thuật toán nhánh cận là sự cải tiến của thuật toán tìm kiếm leo đồi
 - TK leo đồi: gặp nghiệm thì dừng tìm kiếm. Kết quả là 1 tối ưu cục bộ
 - TK nhánh cận: gặp nghiệm thì vẫn tiếp tục tìm kiếm nghiệm tối ưu. Kết quả là nghiệm tối
- Mục đích: chuyển từ tìm kiếm nghiệm tối ưu cục bộ sang tìm kiếm tốt nhất toàn cục

Thuật toán nhánh cận

begin

1. Khởi tạo danh sách OPEN chỉ chứa trạng thái ban đầu;

Gán giá trị ban đầu cho min $\leftarrow +\infty$;

2. loop do

 2.1 if OPEN rỗng then stop;

 2.2 Loại trạng thái u ở đầu danh sách OPEN;

 2.3 if u là trạng thái kết thúc then

 if $f(u) < \text{min}$ then { $\text{min} \leftarrow f(u)$; Quay lại 2.1}; // cập nhật lại min

 2.4 if $f(u) > \text{min}$ then Quay lại 2.1; // cắt bỏ nhánh con

 2.5 for mỗi trạng thái v kề u do

 {
 $g(v) \leftarrow g(u) + k(u,v);$
 $f(v) \leftarrow g(v) + h(v);$
 Đặt v vào danh sách L
 }

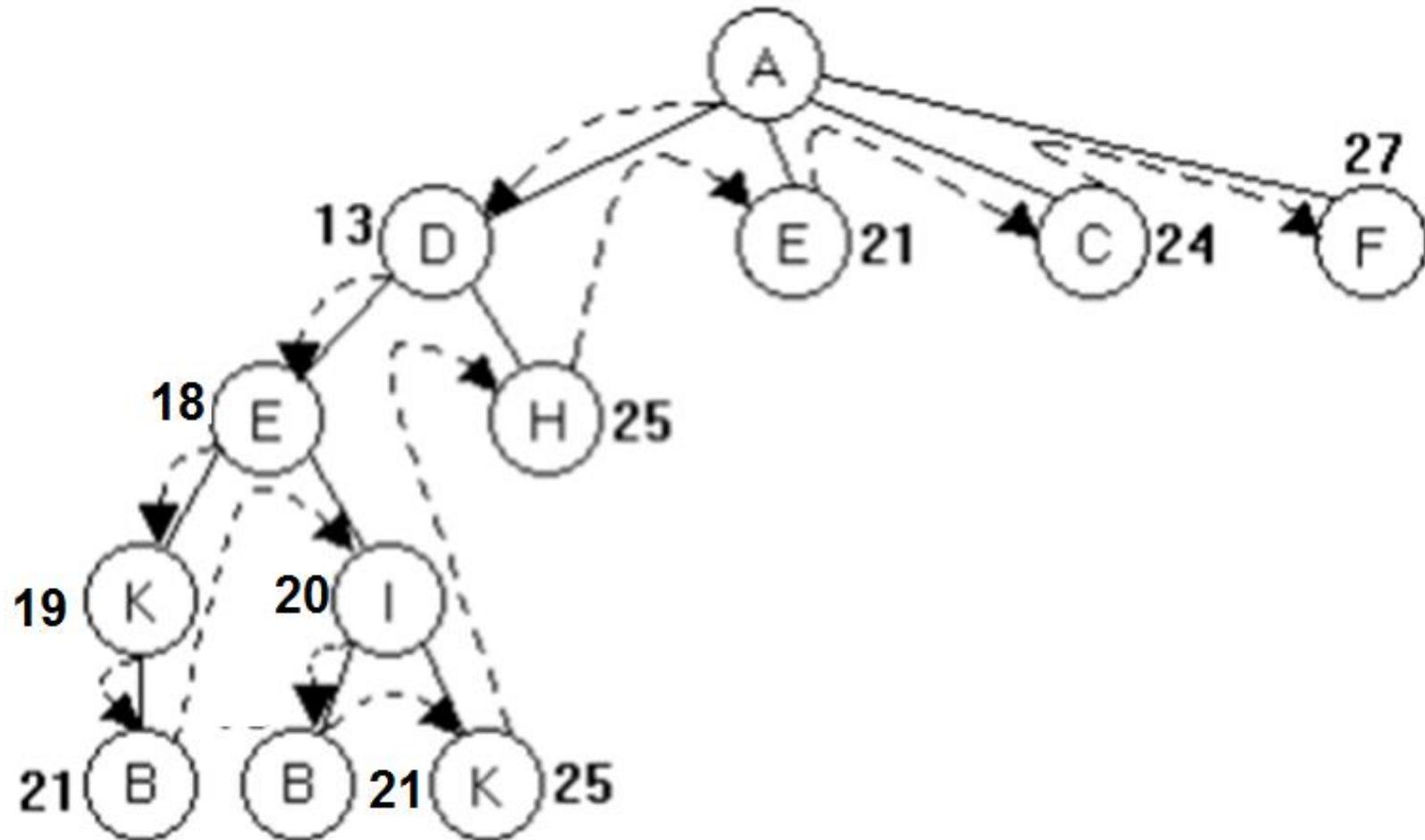
 2.6 Sắp xếp L theo thứ tự tăng của hàm f;

 2.7 Chèn L vào đầu danh sách OPEN

end;

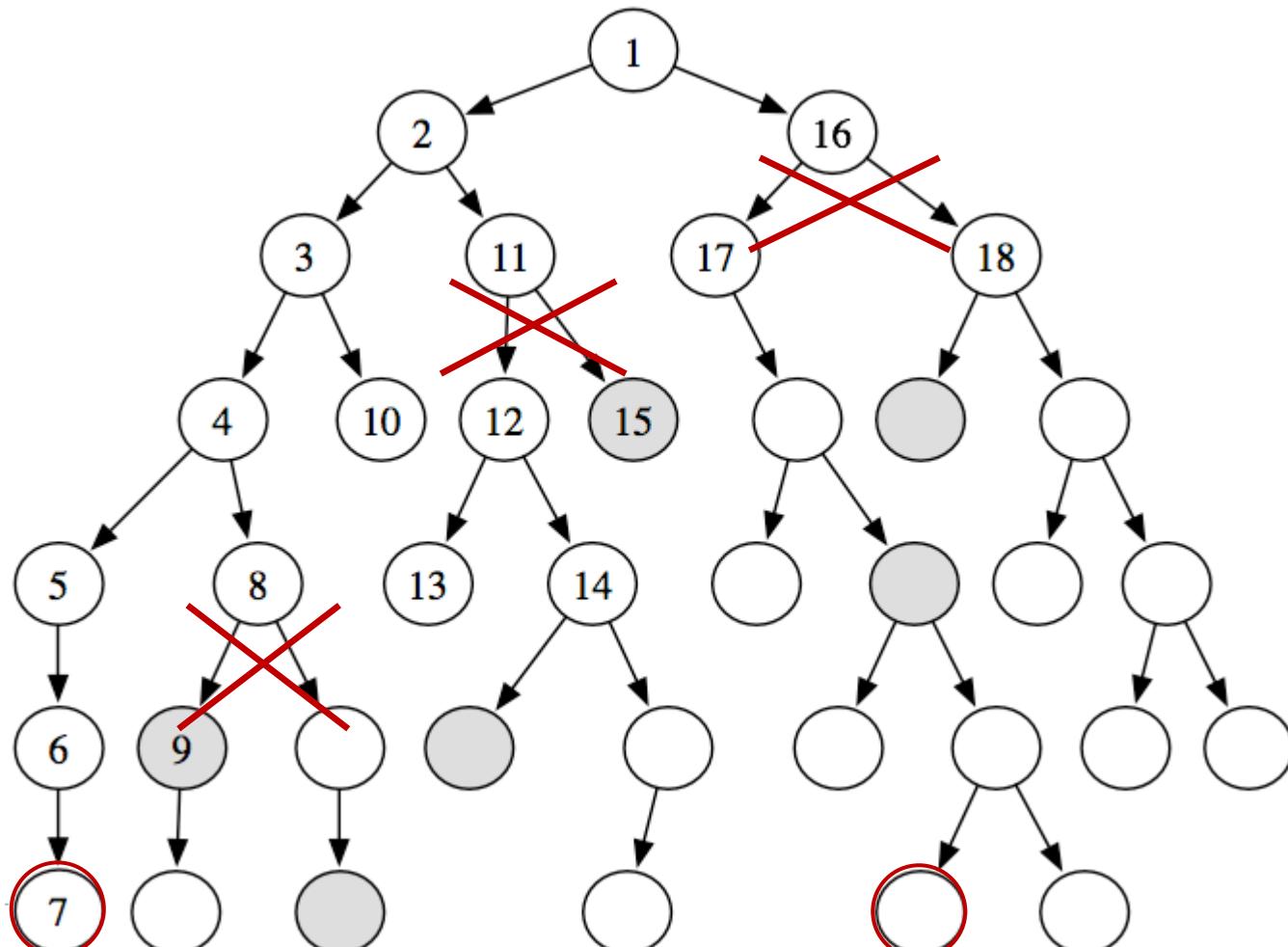
Thuật toán nhánh cận

- Khuyết $h(u)$. Thông tin trên mỗi nút là chi phí $g(u)$



Thuật toán nhánh cận

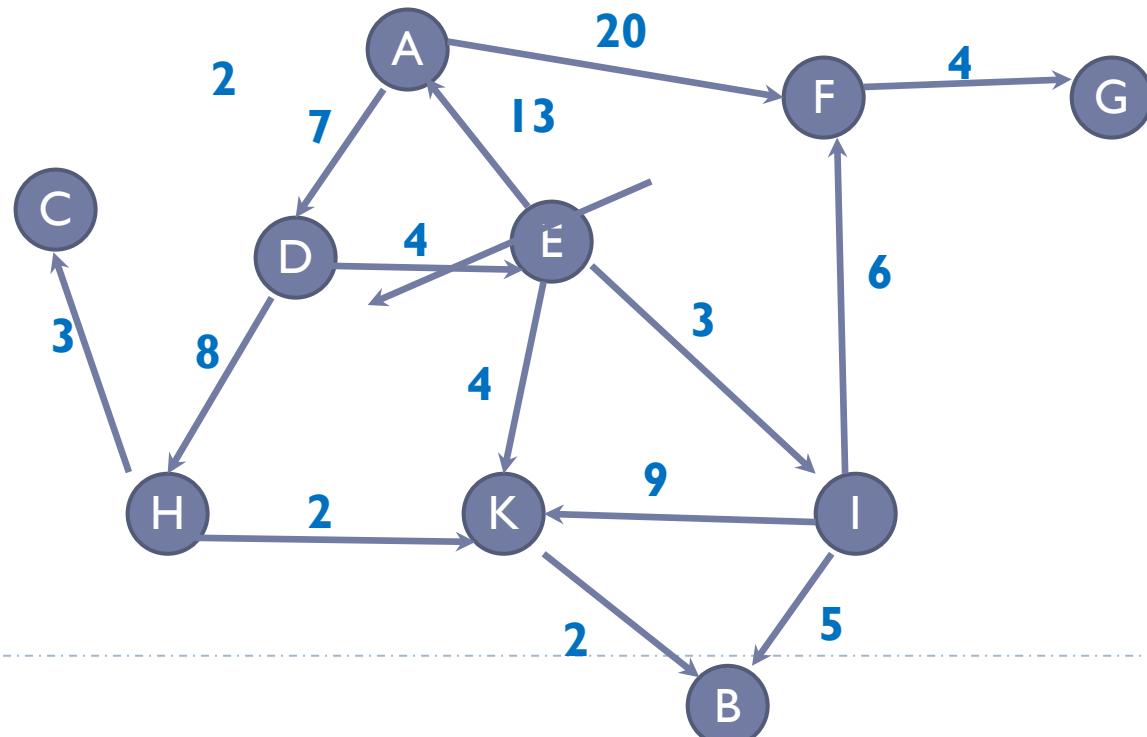
- Khuyết $h(u)$. Thông tin ghi trên mỗi nút là chi phí tổng chi phí $g(u)$ từ nút start đến u .



Thuật toán nhánh cận

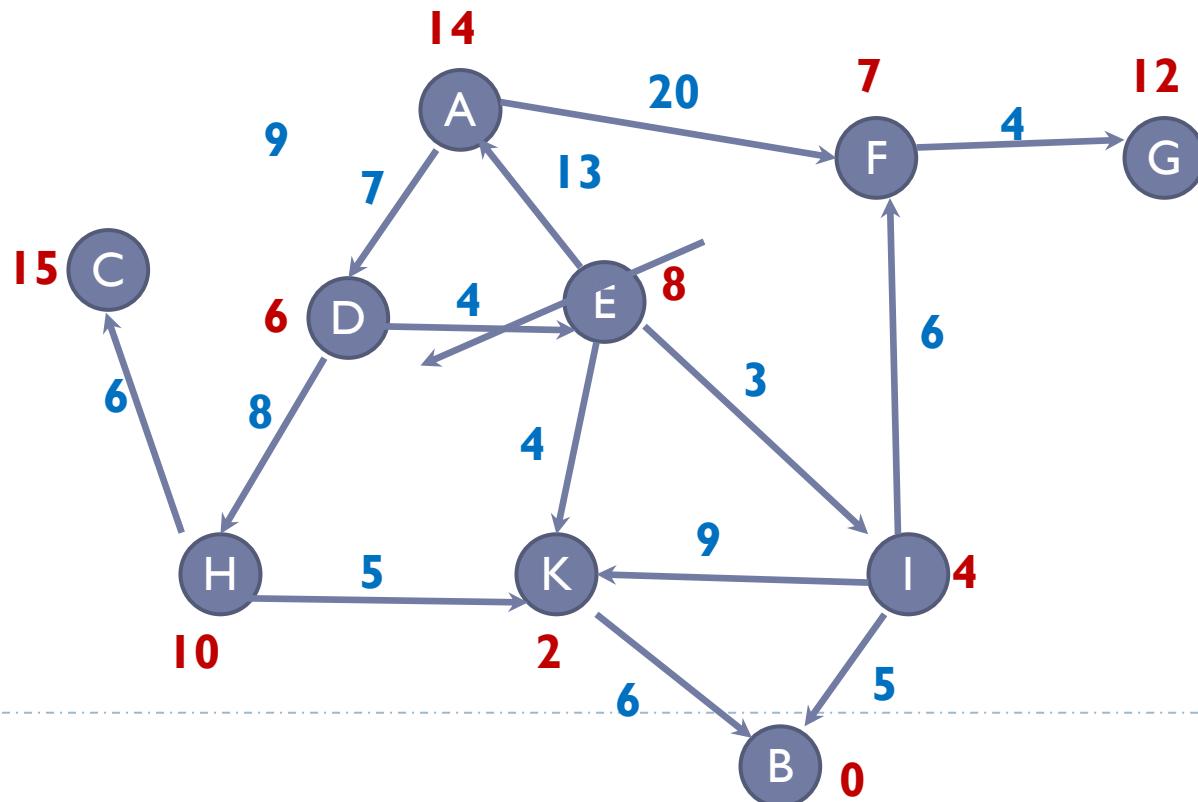
■ Tìm đường đi từ A → B bằng giải thuật nhánh cận

- Thông tin trên các cạnh là k_{uv}
- Thuật toán nhánh cận cũng thường áp dụng với bài toán chỉ có thông tin g



Thuật toán nhánh cành

- Tìm đường đi từ A → B bằng giải thuật nhánh cành
 - Có thêm thông tin $h(u)$ gắn tại các đỉnh)



Thuật toán nhánh cận

A*	Nhánh cận
<ul style="list-style-type: none"><input type="checkbox"/> Tìm thấy thì dừng<input type="checkbox"/> Phương án tìm được là tối ưu chỉ khi hàm h là hàm chấp nhận được<input type="checkbox"/> Khi hàm h không phải là chấp nhận được, phương án tìm được chỉ là phương án tìm thấy đầu tiên, chưa chắc đã là tối ưu nhất	<ul style="list-style-type: none"><input type="checkbox"/> Tìm thấy thì vẫn tiếp tục tìm các phương án khác để so sánh<input type="checkbox"/> Luôn tối ưu<input type="checkbox"/> Kết hợp đánh chặn (chặt bỏ) trước các nhánh không tốt ngay khi có thể (<i>phát hiện ra nhánh không tốt so với đường đi tốt nhất tạm thời tìm được</i>)<input type="checkbox"/> Cập nhật lại đường đi tốt nhất tạm thời nếu tìm thấy có đường đi khác tốt hơn

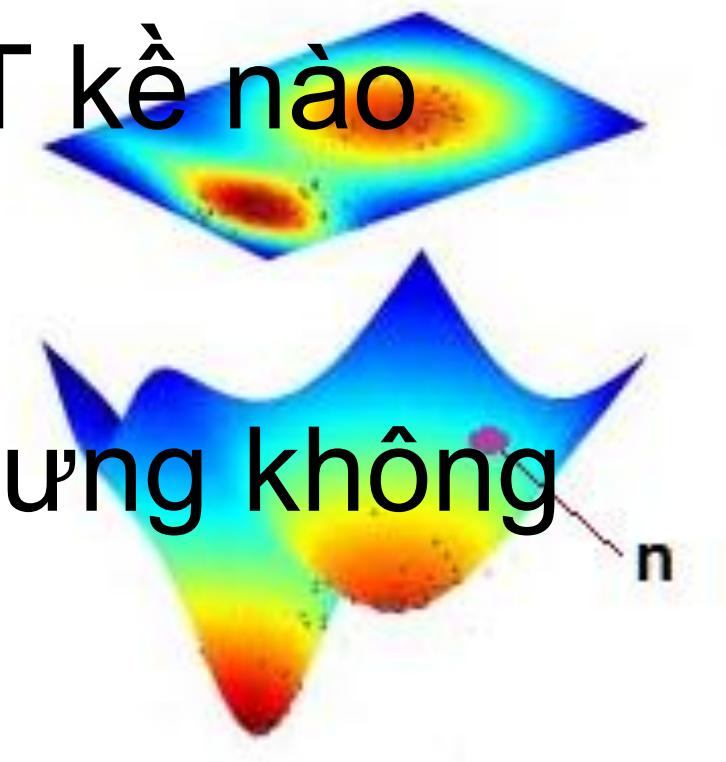
Tìm kiếm tối ưu cục bộ

Tìm kiếm tối ưu cục bộ

- Mỗi trạng thái u được đánh giá bằng 1 giá trị $\text{cost}(u)$.
- Ví dụ
 - n-queens: số vi phạm (số lượng các cặp quân hậu thằng hàng/cột/chéo)
 - 8-puzzle: $h(u)$
- Hàm mục tiêu: $\text{cost}(u) \rightarrow \min$
- TK cục bộ (local search) là phương pháp TK xấp xỉ
 - Ko đi tìm nghiệm tối ưu toàn cục. Không xét hết KGTT.
 - Tìm trạng thái tốt nhất có thể trong 1 khoảng thời gian chấp nhận được
- Các phương pháp tìm kiếm cục bộ:
 - Greedy search
 - Tabu search

Greedy search

- ☒ Từ 1 TT xuất phát nào đấy, di chuyển sang TT kèt tốt hơn cho đến khi không tìm TT kèt nào tốt hơn
- ☒ Giống TK leo đồi nhưng không có quay lui



Greedy search

- Các yếu tố của thuật toán tìm kiếm cục bộ
 - Tập trạng thái kề của 1 trạng thái: nhiều định nghĩa
 - Phép chọn: Khi duyệt 1 trạng thái, ta sẽ lựa chọn di chuyển tới 1 trạng thái kề của nó để duyệt trong bước lặp tiếp theo. Có nhiều chiến lược lựa chọn
 - . Chọn ngẫu nhiên 1 TT với 1 xác suất nào đó
 - . Chọn TT tốt nhất trong số các TT kề (có thể xấu đi)
 - . Chọn TT tốt nhất trong số các TT kề tốt hơn (greedy search)
 -

Greedy search

- Giả sử thuật toán dừng lại tại phương án u^* , u^* là nghiệm tối ưu địa phương
 - $\text{Cost}(u^*)$ nhỏ hơn cost của tất cả các trạng thái nằm trên đường đi từ u tới u^*
- Cải tiến :
 - Chạy nhiều lần giải thuật tại các điểm xuất phát khác nhau
 - Chọn phương án tốt nhất trong số các phương án tìm được từ các lần chạy
 - Tăng khả năng tìm được tối ưu toàn cục

Tabu search

- Mục đích: tránh TK rơi vào vòng lặp, di chuyển quanh quẩn tại 1 số trạng thái
- Đánh dấu thời điểm duyệt 1 trạng thái
 - tabu[u]: thời gian u được duyệt
- Cấm duyệt lại 1 trạng thái vừa mới được duyệt trong 1 giới hạn thời gian nào đó
- Thường sử dụng greedy search để tạo trạng thái xuất phát
- **Thuật toán**

Tìm kiếm có đối thủ

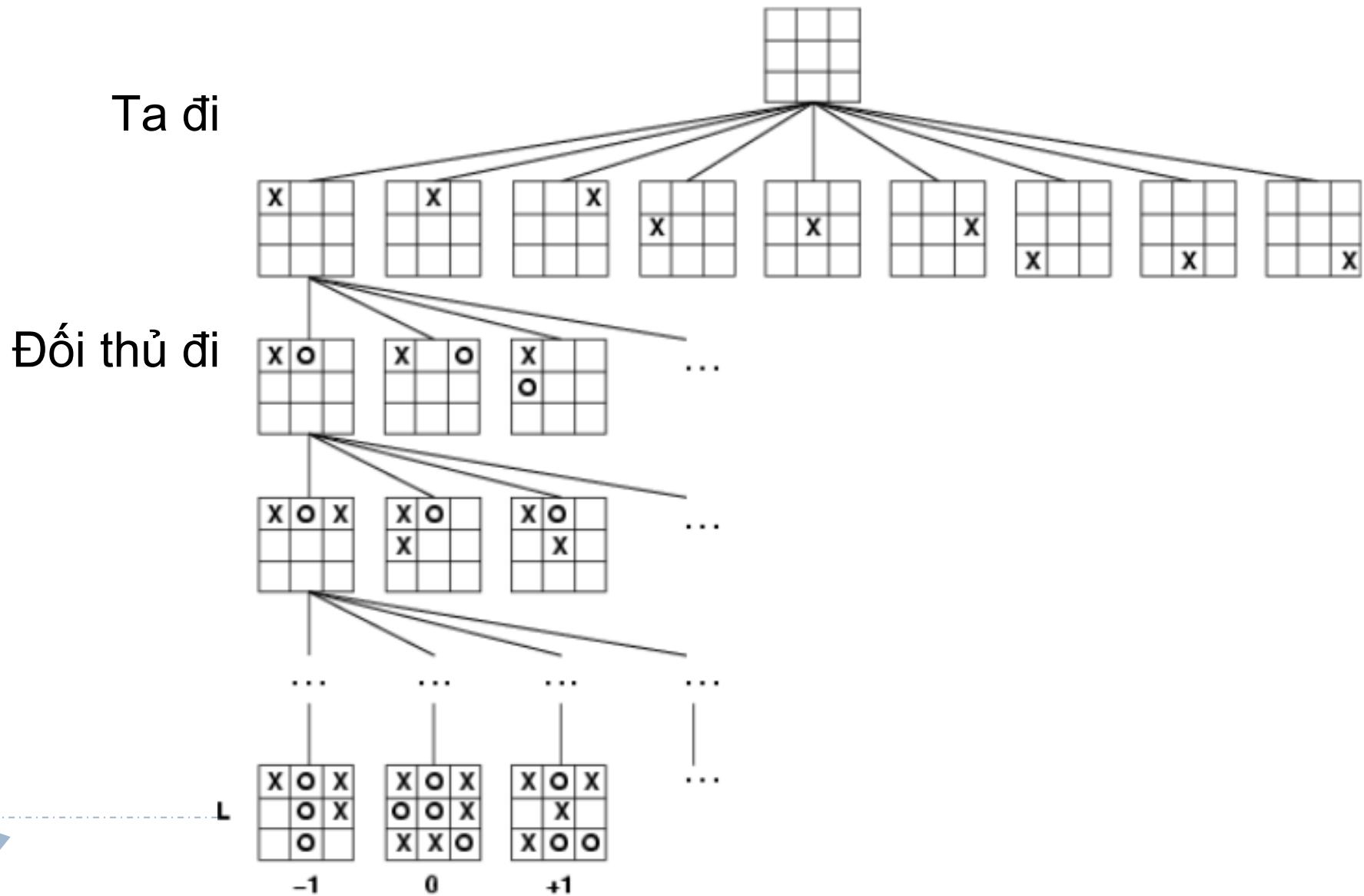
Trò chơi có đối thủ

- Là những trò chơi gồm:
 - Hai người thay phiên (xen kẽ) chơi như cờ vua, cờ tướng, cờ caro, ...
 - Các luật chơi là như nhau với 2 người
 - Hai người biết thông tin đầy đủ về nhau
 - Nhưng mỗi người không biết người còn lại sẽ đi nước nào tiếp theo
 - Mục tiêu: tại mỗi bước đi, người chơi cần chọn ra 1 nước đi trong số các nước đi hợp lệ sao cho qua 1 dãy các nước đi của mình và đối thủ, người chơi giành phần thắng

Cây trò chơi

- Gốc: trạng thái bắt đầu cuộc chơi
- Các phép chuyển: các nước đi hợp lệ
- Các nút lá: các trạng thái kết thúc, ứng với các tình thế mà cuộc chơi dừng (thắng, thua, hòa)
 - Giá trị của mỗi trạng thái kết thúc thường được xác định bằng 1 hàm kết cuộc
 - Ví dụ :
 - Cờ vua, cờ caro : thắng = 1, hòa = 0, thua = -1
 - Ô ăn quan : [-k, k]

Cây trò chơi



Cây trò chơi

- Mục tiêu: người chơi cần tìm ra một dãy các nước đi xen kẽ với các nước đi của đối thủ từ trạng thái xuất phát tới trạng thái kết thúc mà mình thắng
- Tìm kiếm có đối thủ quy về tìm kiếm trên cây trò chơi sao cho người chơi có điểm càng lớn càng tốt, còn đối thủ có điểm càng nhỏ càng tốt

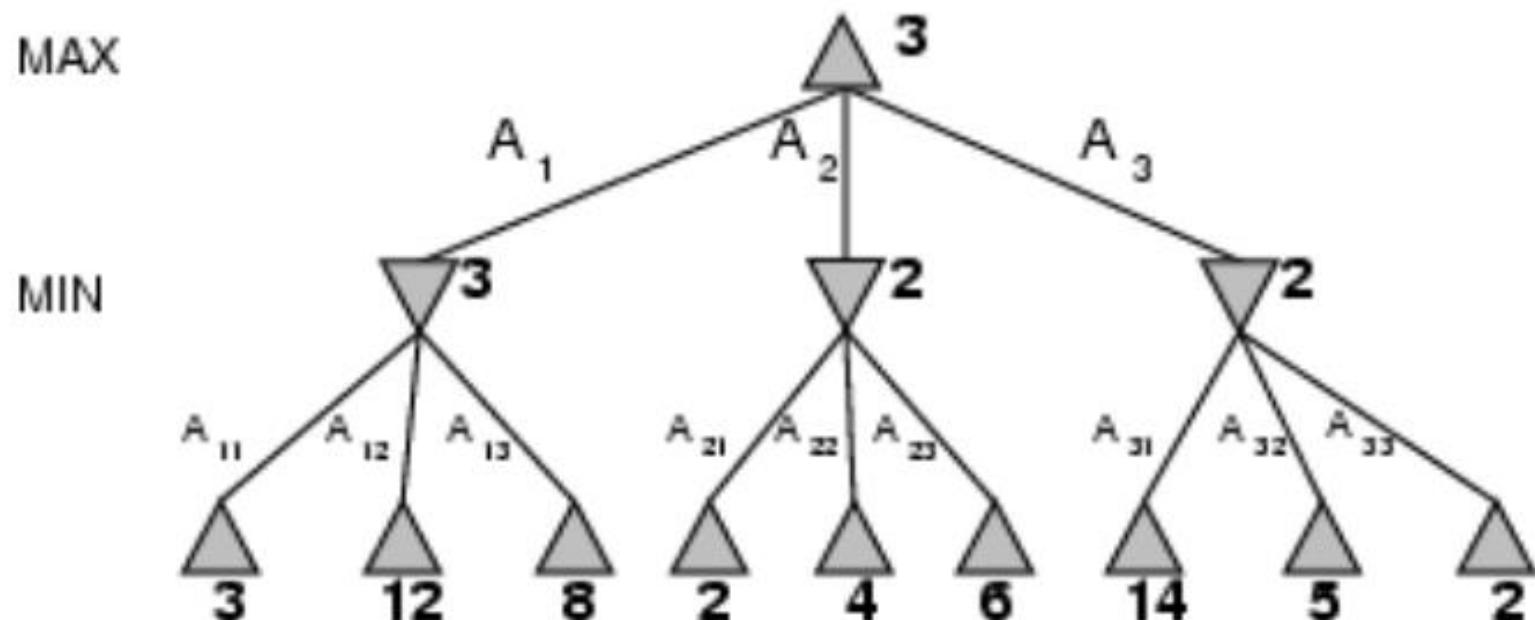
Thuật toán min-max

- Hai đấu thủ: MAX(ta) và MIN(đối thủ)
 - MAX luôn tìm cách tối đa ưu thế của mình
 - MIN luôn tìm cách đưa MAX vào thế khó khăn nhất.
- Mỗi mức trên cây trò chơi ứng với một đấu thủ
 - Đỉnh MAX ứng với trạng thái MAX đưa ra nước đi
 - Đỉnh MIN ứng với trạng thái MIN đưa ra nước đi
- Các mức MAX và MIN xen kẽ nhau
 - Con của 1 nút MAX là các nút MIN và ngược lại

Thuật toán min-max

- ℳ Mỗi nút u của cây trò chơi sẽ gắn với một giá trị $\text{val}(u)$ phản ảnh lợi thế của người chơi (bất lợi với đối thủ).
- ℳ Giá trị của các nút lá xác định bởi hàm kết cục
- ℳ Giá trị của các nút trong được xác định như sau:
 - Giá trị của 1 nút MAX là max của giá trị của các nút con của nó
 - Giá trị của 1 nút MIN là min của các giá trị của các nút con của nó

?



Thuật toán min-max

■ Ví dụ trò chơi Nim

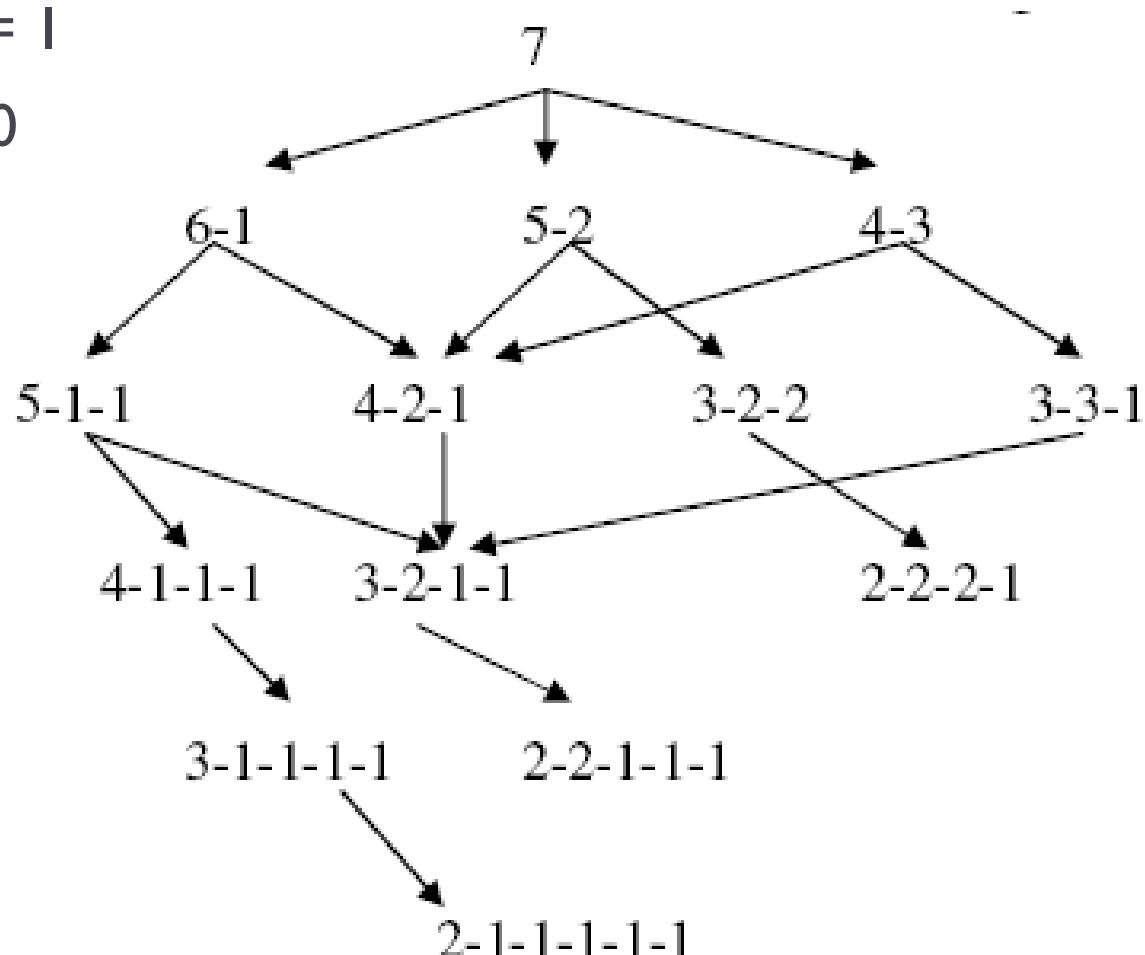
- Có n ($n > 2$) đồng xu, ban đầu xếp thành 1 đống.
- Mỗi bước, người chơi phải chọn 1 đống đồng xu để chia thành hai đống con có số lượng đồng xu khác nhau.
- Người thua sẽ là người cuối cùng không chia được theo yêu cầu của bài toán.

Thuật toán min-max

■ Ví dụ trò chơi Nim (n=7)

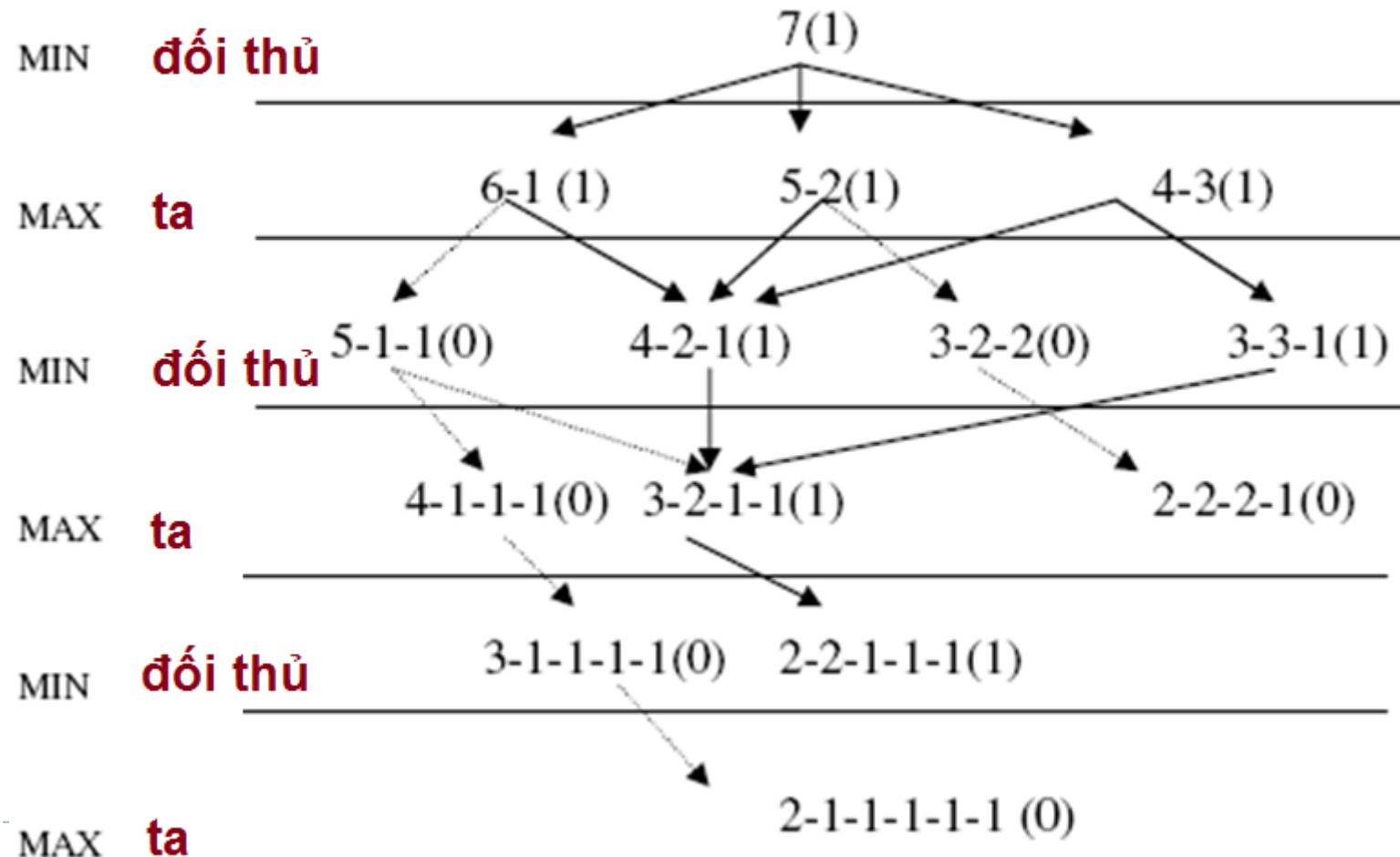
- Hàm kết cuộc :

- Ta thắng = 1
- Ta thua = 0

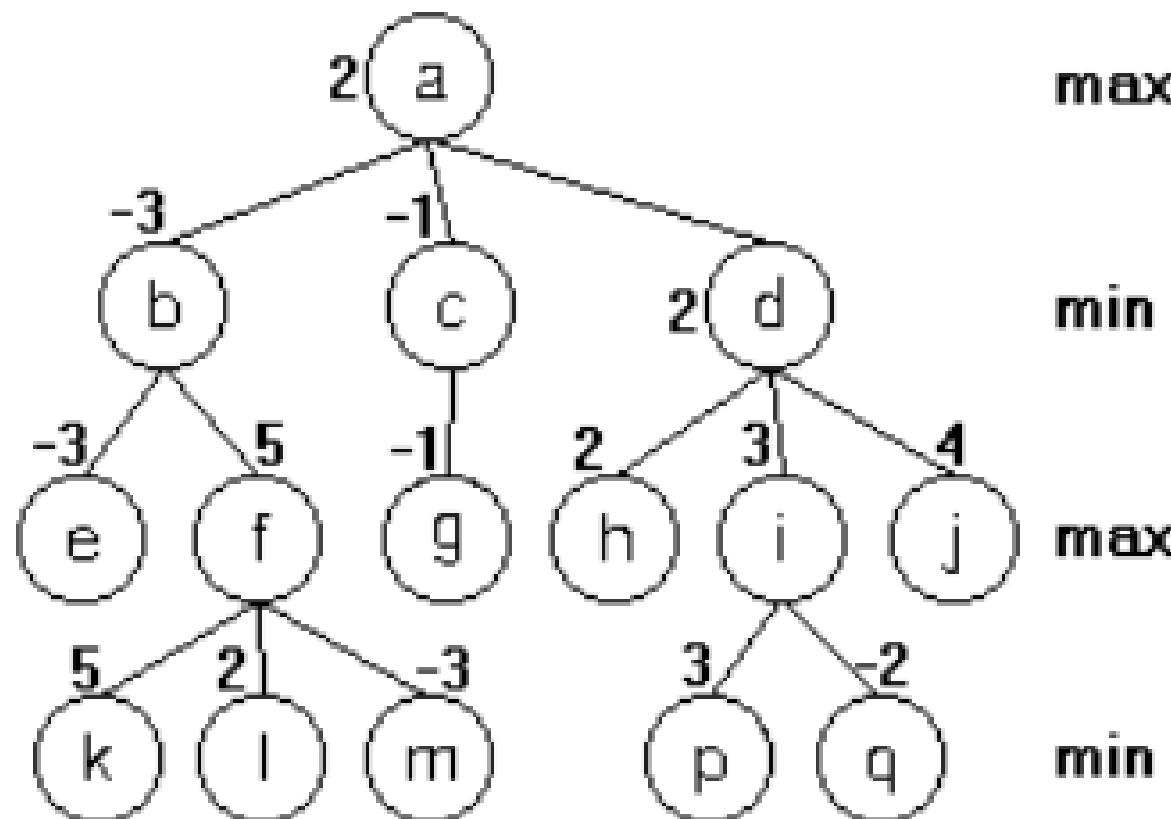


Thuật toán min-max

■ Ví dụ : trò chơi Nim ($n=7$)



Thuật toán min-max



Hình 4.3 Gán giá trị cho các đỉnh của cây trò chơi.

Thuật toán min-max

?

Tại một trạng thái u:

- Nếu đến lượt MAX (ta) đi:
 - . Cần tìm nước đi tốt nhất với ta trong số các nút con của u
 - . Chọn nút con có giá trị lớn nhất để **tối đa hóa** lợi thế của ta
- Nếu đến lượt MIN (đối thủ) đi:
 - . Cần tìm nước đi tốt nhất với địch trong số các nút con của u
 - . Địch chọn nút con có giá trị nhỏ nhất để **tối thiểu hóa** lợi thế của ta, tức tối đa hóa lợi thế của địch



Thuật toán min-max

Function MaxVal(u); // Định giá cho nút MAX

Begin

If u là nút lá then

 Val(u):=f(u) // f(u) là hàm kết cuộc

Else

 MaxVal(u):=max{MinVal(v) | v là các con của u}

End;

Function MinVal(u) // Định giá cho nút MIN

Begin

If u là nút lá then

 MinVal(u):=f(u); // f(u) là hàm kết cuộc

Else

 MinVal(u):=min{MaxVal(v) | v là các con của u};

End;

Thuật toán min-max

Thủ tục chọn nước đi cho MAX

Procedure Minimax(u, var v);

Begin

 Val:= -∞;

 For mỗi con w của u do

 If val<=MinVal(w) then begin

 Val:=MinVal(w);

 v:=w;

 End;

End;

Thuật toán min-max

Thủ tục chọn nước đi cho MIN

Procedure Minimax(u, var v);

Begin

 Val:= +∞;

 For mỗi con w của u do

 If val>=MaxVal(w) then begin

 Val:=MaxVal(w);

 v:=w;

 End;

End;

Bài tập

- ❑ Biểu diễn KGTT trò chơi Nim với $n=6$
- ❑ Đánh giá các nút
- ❑ Giả sử đối thủ chọn cách đi là con bên phải
- ❑ Máy thắng hay thua?

Minimax với độ sâu hạn chế

- Về mặt lý thuyết, khi có đầy đủ nội dung cây trò chơi, ta có thể tính trước toàn bộ nước đi và tìm được nước đi tối ưu (cho MAX)
- Trên thực tế, ta không thể tìm được nước đi tối ưu vì không đủ thời gian để xem xét toàn bộ các nút của cây
- Giải pháp để tìm nhanh nước đi tốt tại nút u
 - Chỉ tính trước d bước nào đó
 - Chỉ xem xét các nút con/cháu của u trong giới hạn độ sâu d

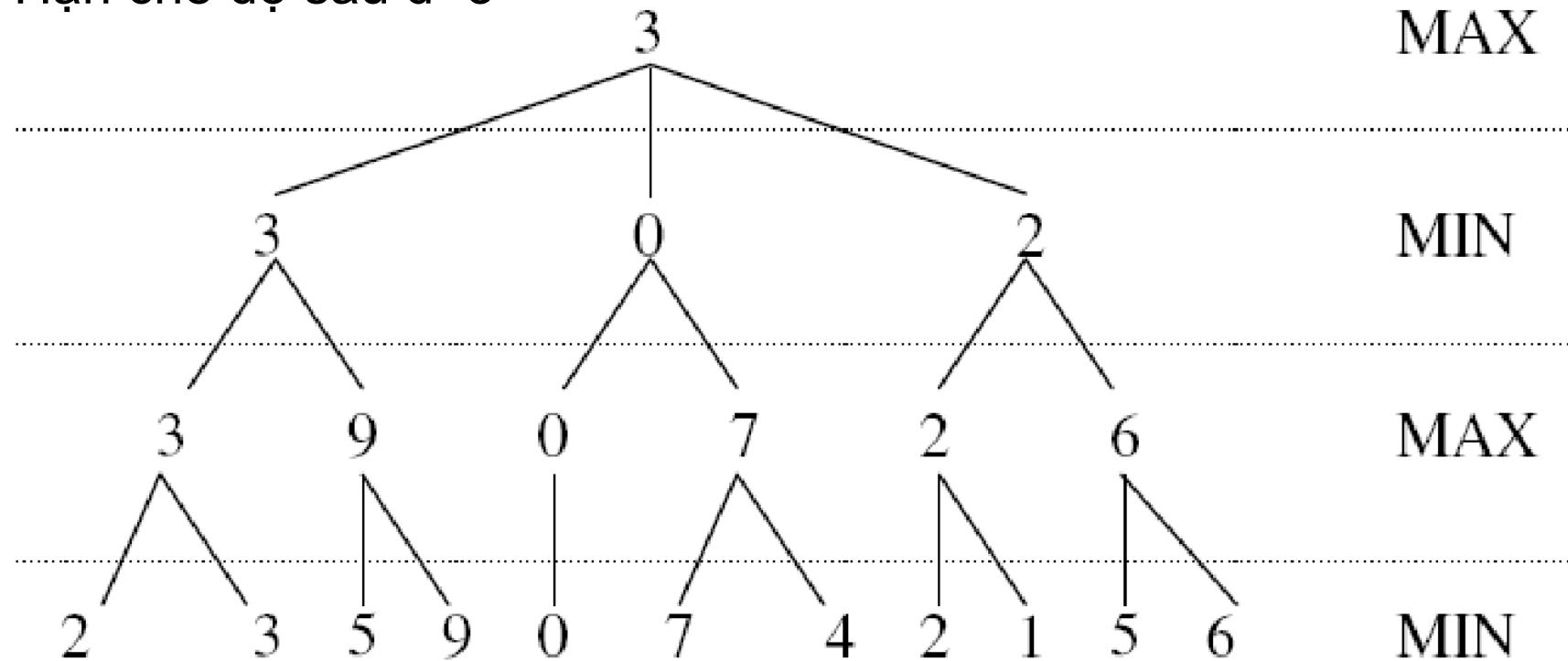
?

Minimax với độ sâu hạn chế

- Không thể gán giá trị thắng/thua một cách chính xác cho các nút mà chỉ đánh giá khả năng thắng thua – tức là một heuristic nào đó.
- Các nút lá được gán các giá trị heuristic (ước lượng) thay vì hàm kết cuộc (chính xác)
- Các nút trong được tính giá trị dựa trên giải thuật Minimax

Minimax với độ sâu hạn chế

Hạn chế độ sâu d=3



Minimax với độ sâu hạn chế

Độ sâu d, u thuộc lớp MAX

Function MaxVal(u, d);

Begin

If $d=0$ hoặc u là đỉnh lá then
 MaxVal(u):= $h(u)$

Else

 MaxVal(u):= $\max\{\text{MinVal}(v, d-1) \mid v \text{ là các con của } u\}$

End;

Minimax với độ sâu hạn chế

Độ sâu d, u thuộc lớp MIN

Function MinVal(u,d);

Begin

If $d=0$ hoặc u là đỉnh lá then

 MinVal(u):=**h(u)**

Else

 MinVal(u):=min{MaxVal(v,d-1) | v là các con của u}

End;

Minimax với độ sâu hạn chế

Thủ tục chọn nước đi cho MAX

Procedure Minimax(u, var v, d);

Begin

 Val:= -∞;

 For mỗi con w của u do

 If MinVal(w, d-1) >= Val then begin

 Val:=MinVal(w,d-1);

 v:=w;

 End;

End;

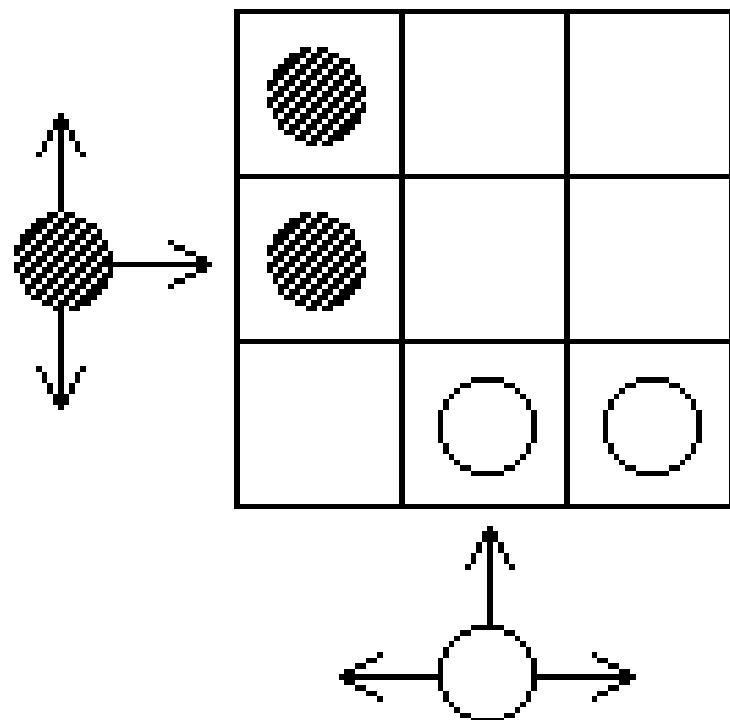
Định giá nút thế nào?

■ Ví dụ: cờ vua

- Dựa vào sự chênh lệch **số lượng** các quân
 - Trọng số cho **loại** quân cờ
 - **Vị trí** của quân cờ.
-
- Quân trắng: tốt hệ số 1; mã, tượng hệ số 3 ; xe hệ số 5, hậu hệ số 9.
 - Quân đen nhận giá trị âm ngược lại đối với quân trắng.
 - Tổng giá trị cả hai quân dùng để đánh giá trạng thái đó.
 - Cách đánh giá này chưa tính đến vị trí của chúng.

Định giá nút lá thế nào?

Trò chơi Dodgem

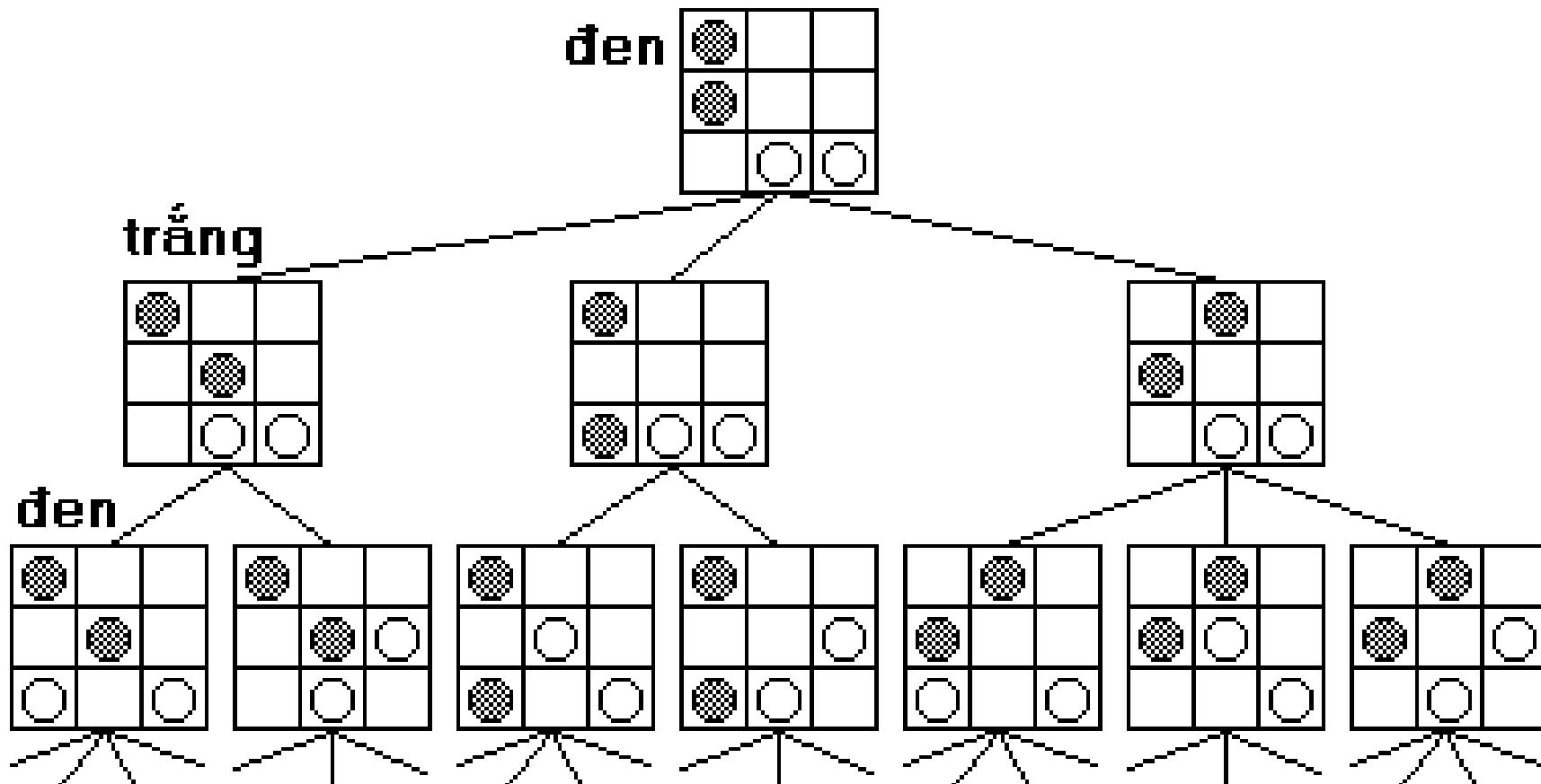


Hình 4.1 Trò chơi Dodgem.

Luật chơi

- Có hai quân trắng và hai quân đen trên bàn cờ 3x3.
- Quân đen không qua được bên trái mà chỉ di chuyển lên trên, xuống dưới hoặc qua phải.
- Quân trắng không đi xuống dưới mà chỉ lên trên qua trái hoặc qua phải.
- Quân đen chỉ được ra khỏi bàn cờ nếu đang đứng ở cột cuối cùng, quân trắng chỉ được ra khỏi bàn cờ nếu đang đứng ở dòng trên cùng.
- Đấu thủ thắng cuộc nếu đưa hết 2 quân của mình ra khỏi bàn cờ hoặc làm cho đối phương không đi được.

Không gian trạng thái



Hình 4.2 Cây trò chơi Dodgem với Đen đi trước.

Định giá cho nút ở độ sâu d?

30	35	40
15	20	25
0	5	10

Giá trị quân Trắng.

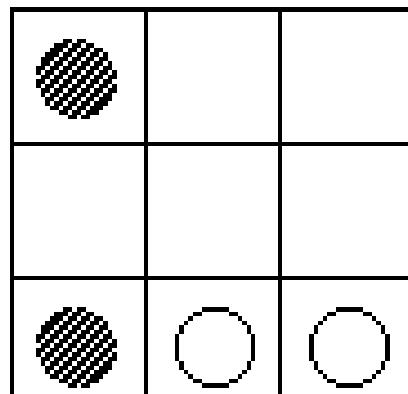
-10	-25	-40
-5	-20	-35
0	-15	-30

Giá trị quân Đen.

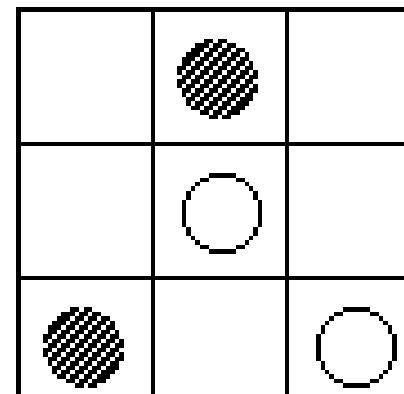
Đánh giá các quân trong trò chơi Dodgem.

Chính xác hơn?

- ℳ Mỗi quân trắng cản trực tiếp quân đen cộng 40 điểm, cản gián tiếp cộng 30 điểm và ngược lại cho quân đen.



75



-5

Giá trị của một số trạng thái trong trò chơi Dodgem.

Số quân?

- 在线咨询
- Ngoài ra, do số quân cờ càng ít thì khả năng thắng càng cao nên nếu chỉ còn một quân trắng thì được cộng thêm như 50 chặng hạn, ngược lại chỉ còn một quân đen thì - 50.

Bài tập 1

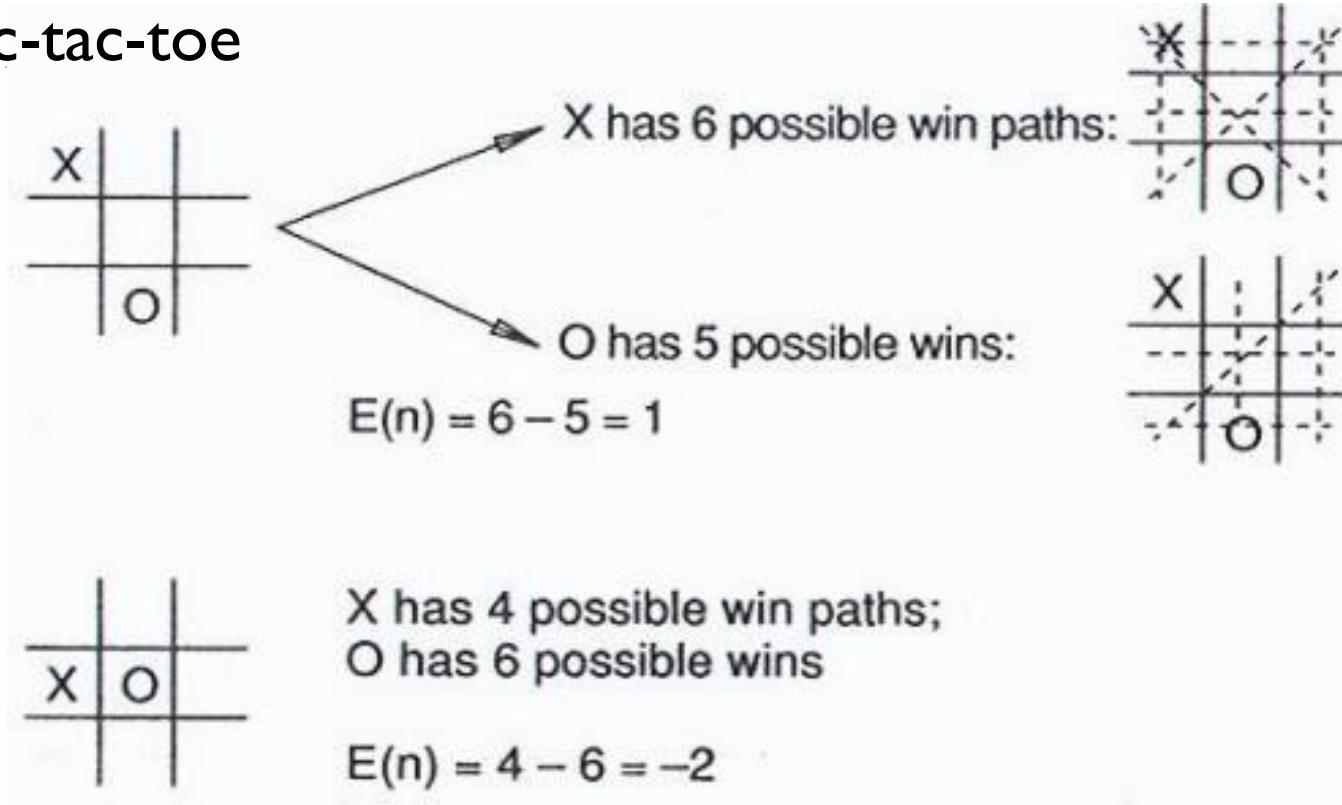
- Vẽ cây trò chơi dodgem với độ sâu $d = 4$
- Tìm nước đi tốt nhất cho bên xuất phát trước

Bài tập 2 :

- ❑ Trò chơi tic-tac-toe
 - ❑ Vẽ cây trò chơi với $d = 4$
 - ❑ Nếu người chơi là người đi xuất phát thì nên chọn nước đi như thế nào ?

Định giá nút lá thế nào?

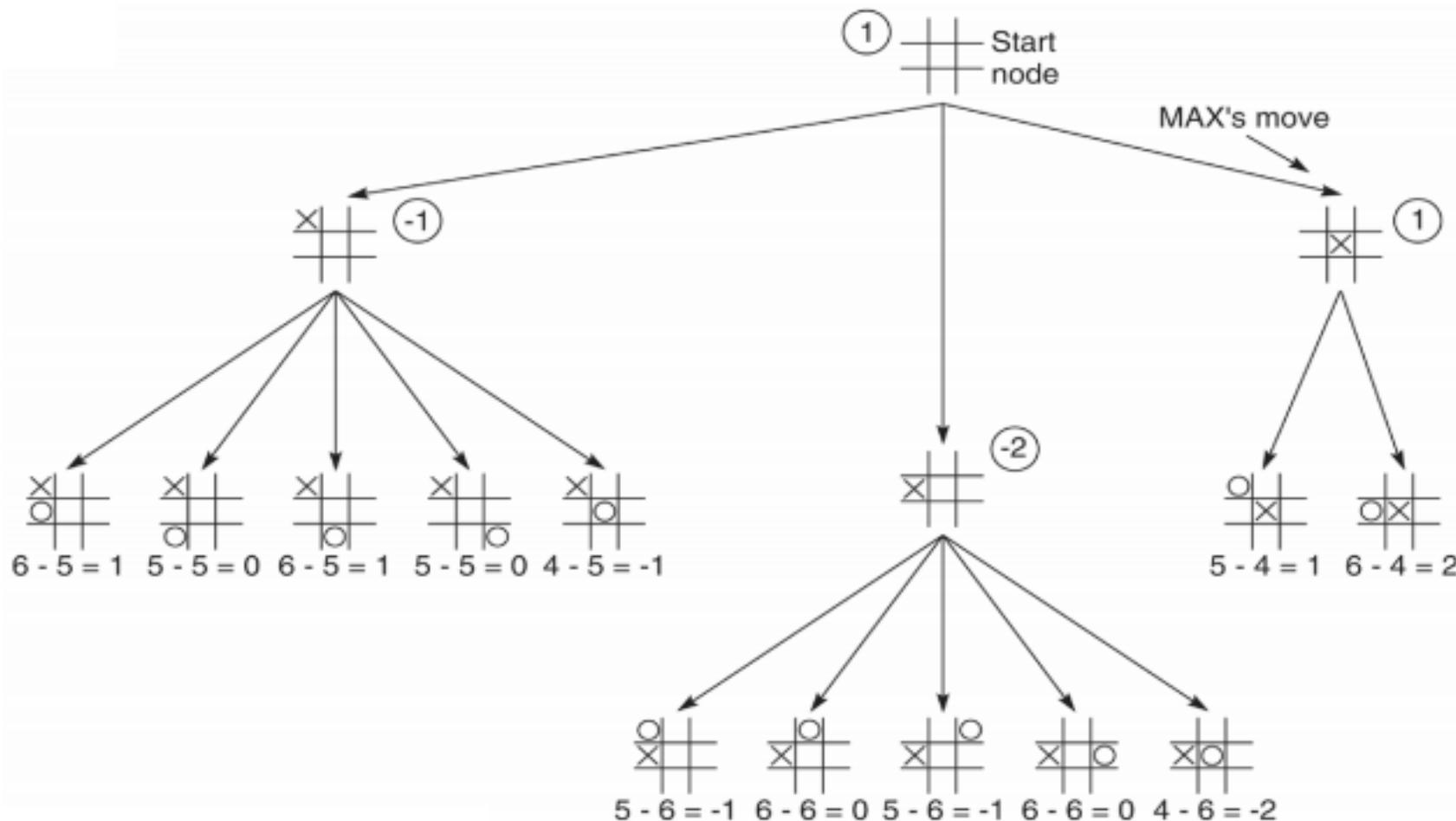
Trò chơi tic-tac-toe



Hàm Heuristic: $E(n) = M(n) - O(n)$. Trong đó:

- ▶ 229- $M(n)$ là tổng số đường thắng có thể của ta
 - $O(n)$ là tổng số đường thắng có thể của đối thủ
 - $E(n)$ trị số đánh giá tổng cộng cho trạng thái n

Minimax 2 lớp trong tic-tac-toe



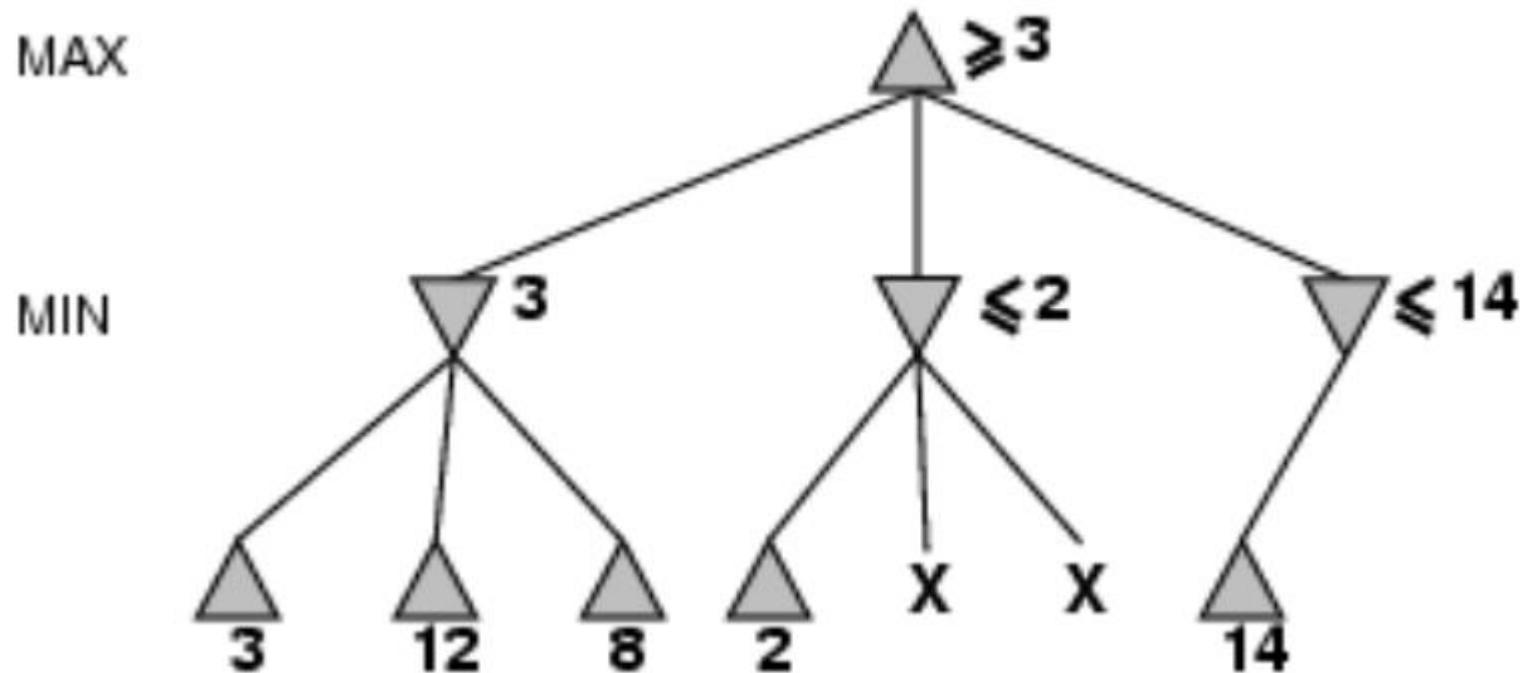
Trích từ Nilsson (1971).

Bài tập 2

QUESTION MARK **Xác định minimax trong tic-tac-toe 4 lớp ?**

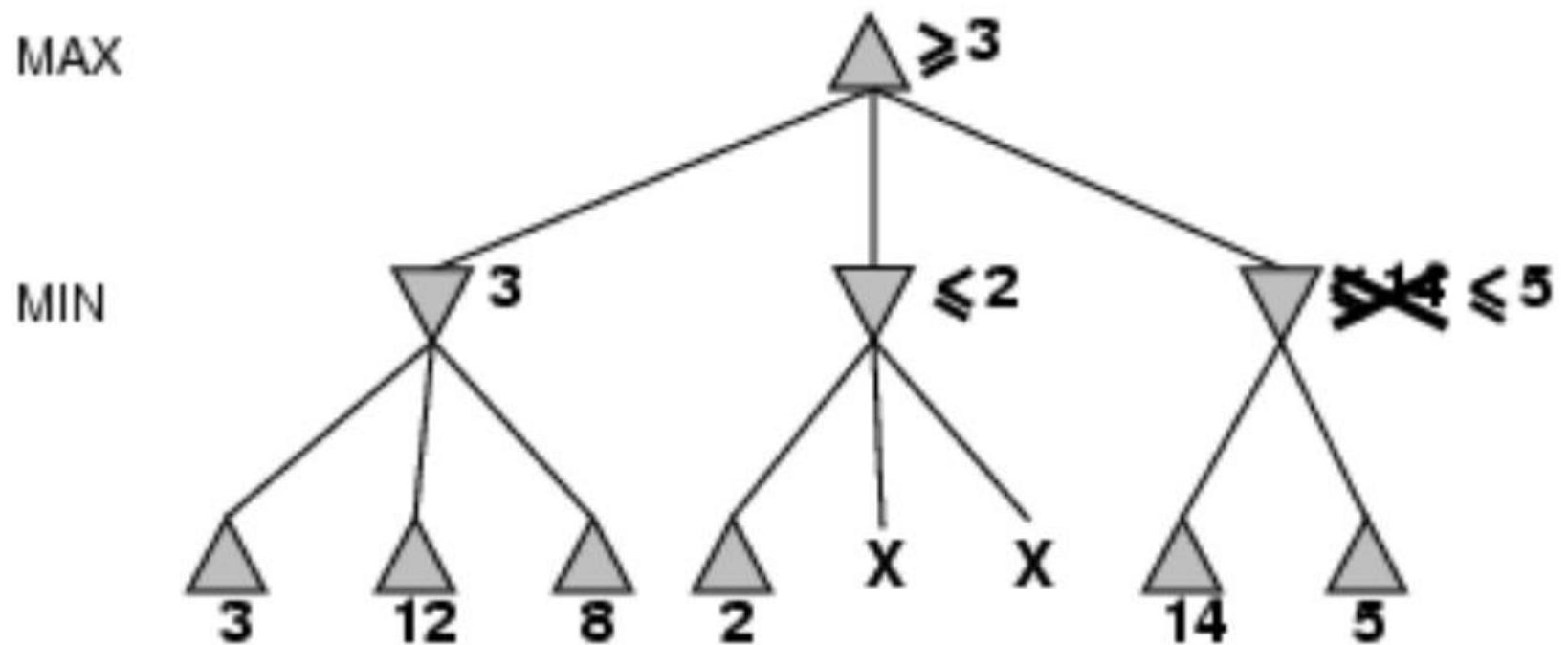
Thủ tục cắt nhánh alpha-beta

- Cắt bỏ bớt các nhánh không cần thiết cho việc đánh giá giá trị heuristic tại một đỉnh
- Ví dụ



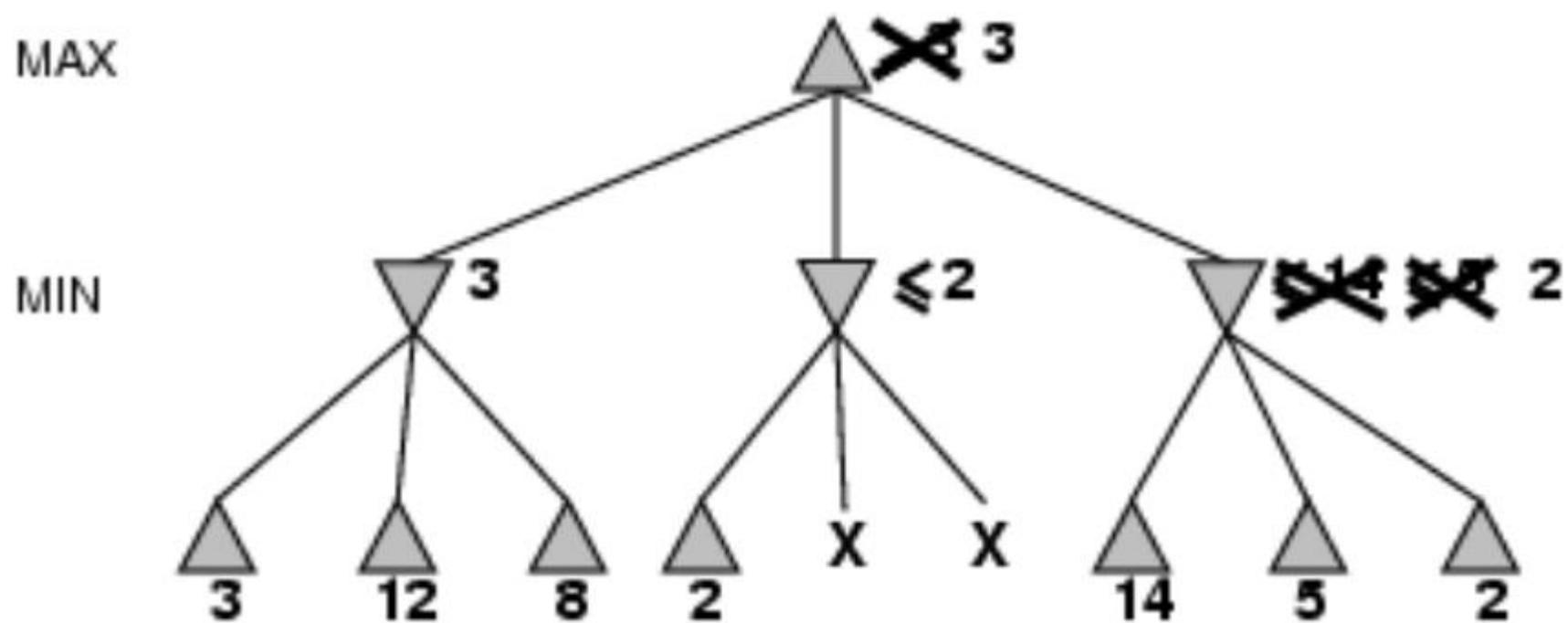
Thủ tục cắt nhánh alpha-beta

- Cắt bỏ bớt các nhánh không cần thiết cho việc đánh giá giá trị heuristic tại một đỉnh
- Ví dụ



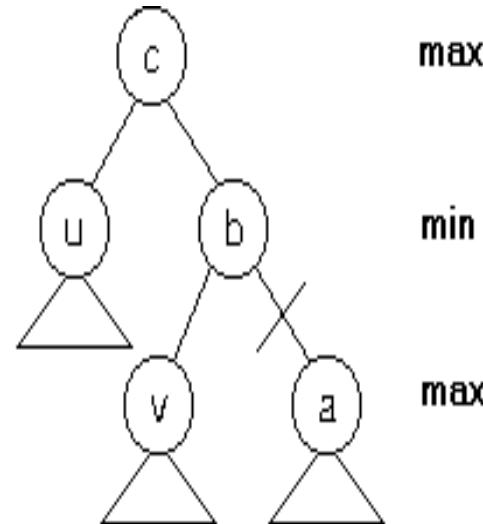
Thủ tục cắt nhánh alpha-beta

- Cắt bỏ bớt các nhánh không cần thiết cho việc đánh giá giá trị heuristic tại một đỉnh
- Ví dụ



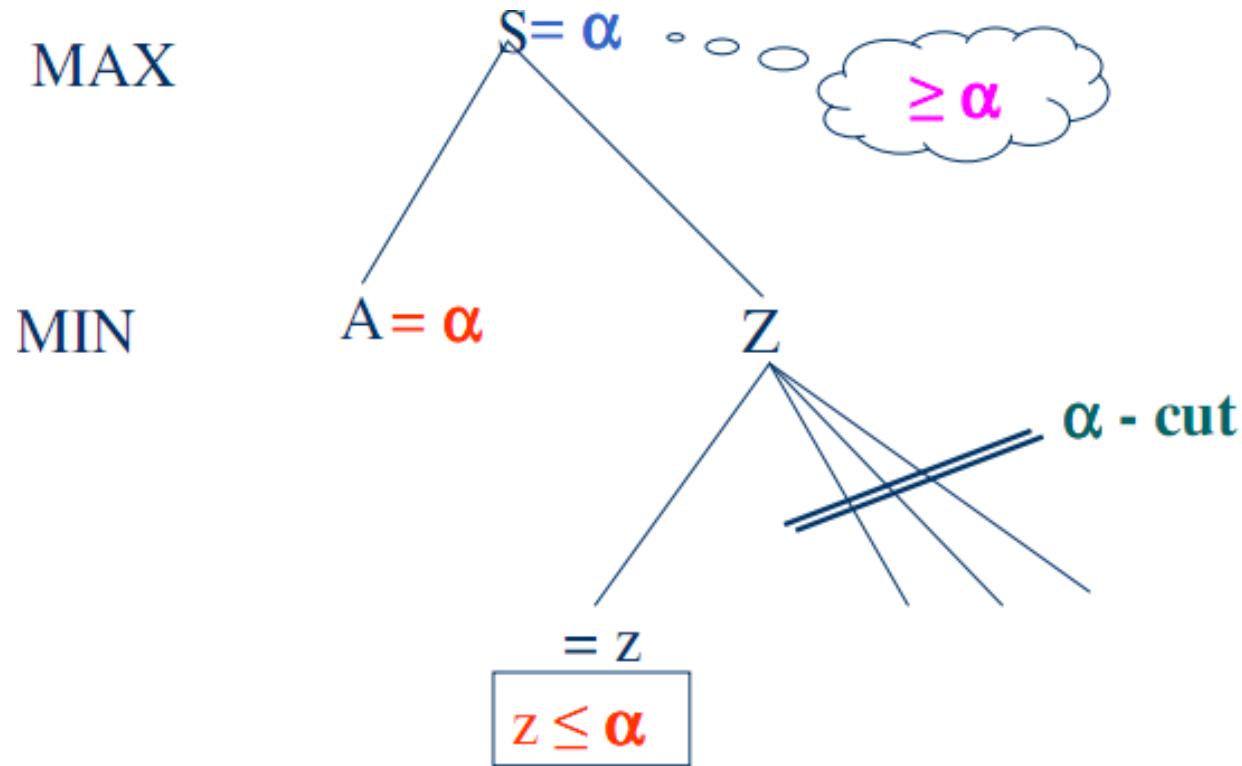
Cơ sở của thủ tục

- ❑ Nếu $\text{val}(v) < \text{val}(u)$ thì bất kể $\text{val}(a)$ bằng bao nhiêu thì $\text{val}(c)$ cũng bằng $\text{val}(u)$.



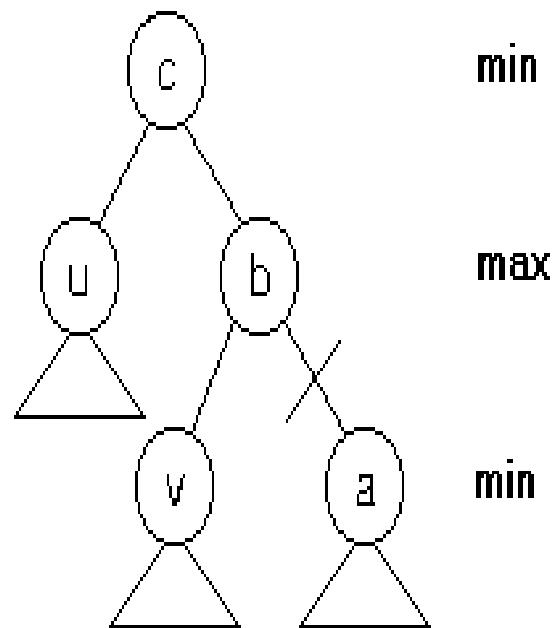
Hình 4.7 Cắt bỏ cây con gốc a, nếu $\text{eval}(u) > \text{eval}(v)$.

Cắt tỉa α



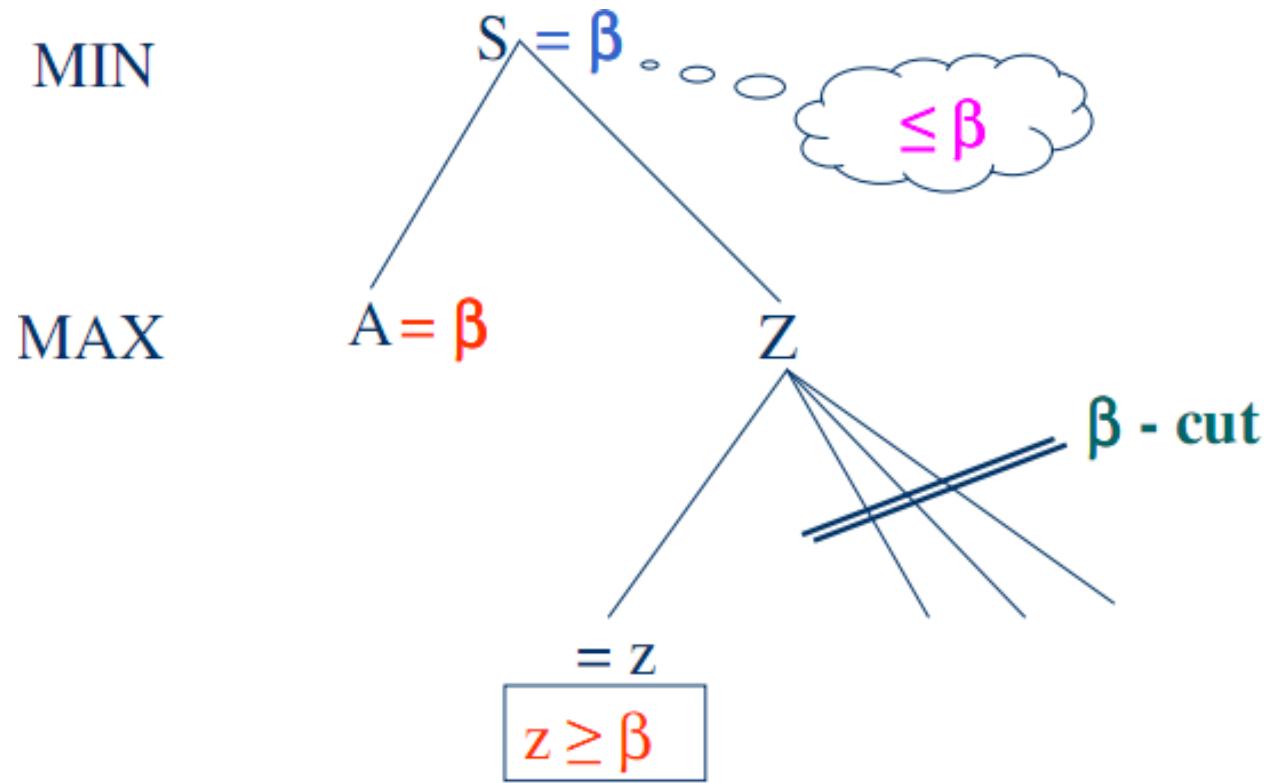
Cơ sở của thủ tục

- ❑ Nếu $\text{val}(u) < \text{val}(v)$ thì bất kể $\text{val}(a)$ bằng bao nhiêu thì $\text{val}(c)$ cũng bằng $\text{val}(u)$. (sửa lại hình)

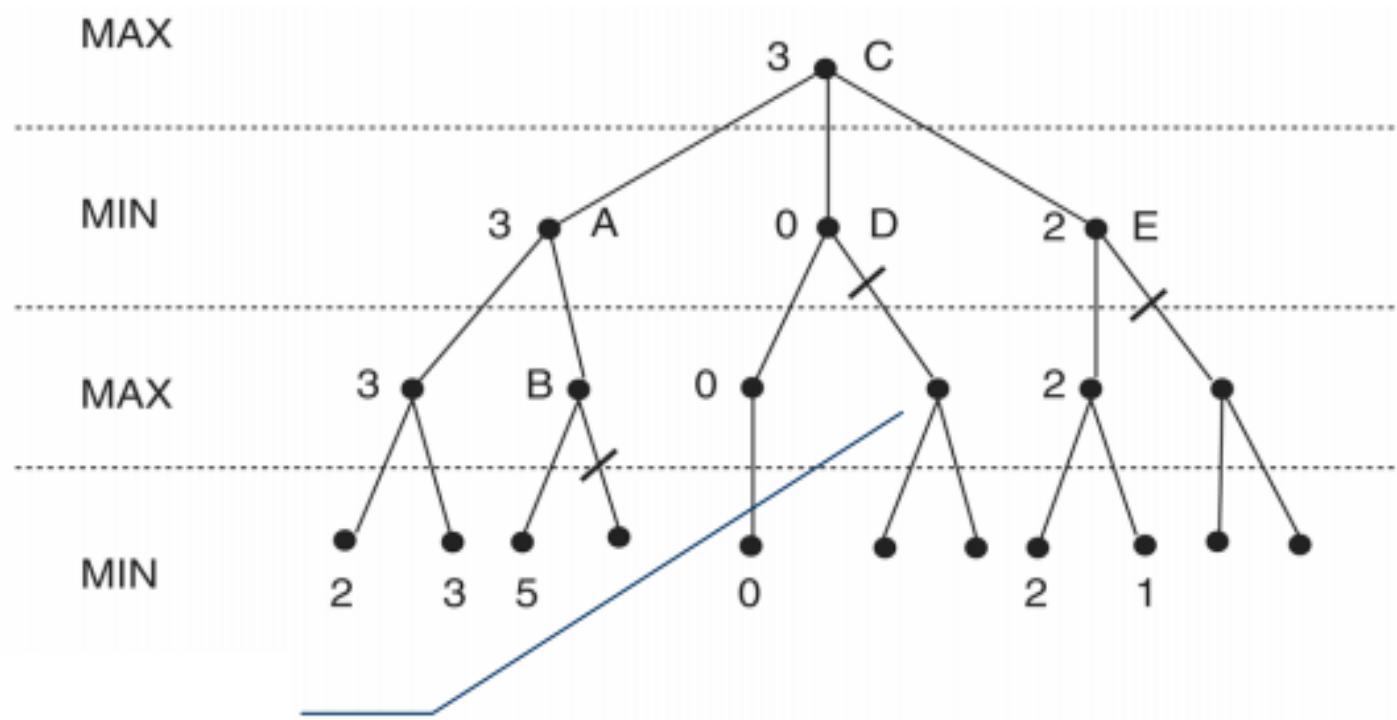


' Cắt bỏ cây con gốc a, nếu $\text{eval}(u) > \text{eval}(v)$.

Cắt tỉa β



Cách duyệt bài toán áp dụng cho max - giá định



Các nút không có giá trị là các nút không được duyệt qua

A has $\beta = 3$ (A will be no larger than 3)

B is β pruned, since $5 > 3$

C has $\alpha = 3$ (C will be no smaller than 3)

D is α pruned, since $0 < 3$

E is α pruned, since $2 < 3$

C is 3

Thủ tục cắt nhánh α - β

- ❑ Tìm kiếm theo kiểu depth-first.
- ❑ Nút MAX có 1 giá trị α (luôn tăng)
- ❑ Nút MIN có 1 giá trị β (luôn giảm)
- ❑ Giải thuật cắt tỉa α - β thể hiện mối quan hệ giữa các nút ở lớp n và lớp $n+2$, mà tại đó toàn bộ cây có gốc tại lớp $n+1$ có thể cắt bỏ.

Thủ tục cắt nhánh α - β

- Khi cài đặt, ta dùng một cặp biến alpha, beta.
- Alpha - giá trị lớn nhất trong các giá trị của các nút con đã đánh giá của một nút max
- Beta – giá trị bé nhất trong các giá trị của các nút con đã đánh giá của một nút min
- **Anpha>=beta cắt nhánh đang xét**

Định giá cho nút u lớp Max

```
Function MaxVal(u, alpha, beta);  
begin  
    if (u là nút lá của cây hoặc cây hạn chế) then Maxval:=f(u)  
    else  
        for w con của u do  
            begin  
                alpha:=max{alpha, MinVal(w, alpha, beta)}  
                if alpha>beta then exit;  
            end;  
        MaxVal:=alpha;  
    end;
```

Định giá cho nút u - lớp Min

Function MinVal(u, anpha, beta);

begin

if (u là nút lá của cây hoặc cây hạn chế) then Minval:=f(u)

else

for w con của u do

begin

beta:=min{beta, MaxVal(w, alpha, beta)}

if anpha>=beta then exit;

end;

MinVal:=beta;

end;

Thủ tục hướng dẫn nước đi cho Max

Procedure Alpha_beta(u, var v)

begin

 alpha:=- ∞ ; beta:=+ ∞;

 for (w con u) do

 if alpha<MinVal(w, alpha, beta) then

 begin

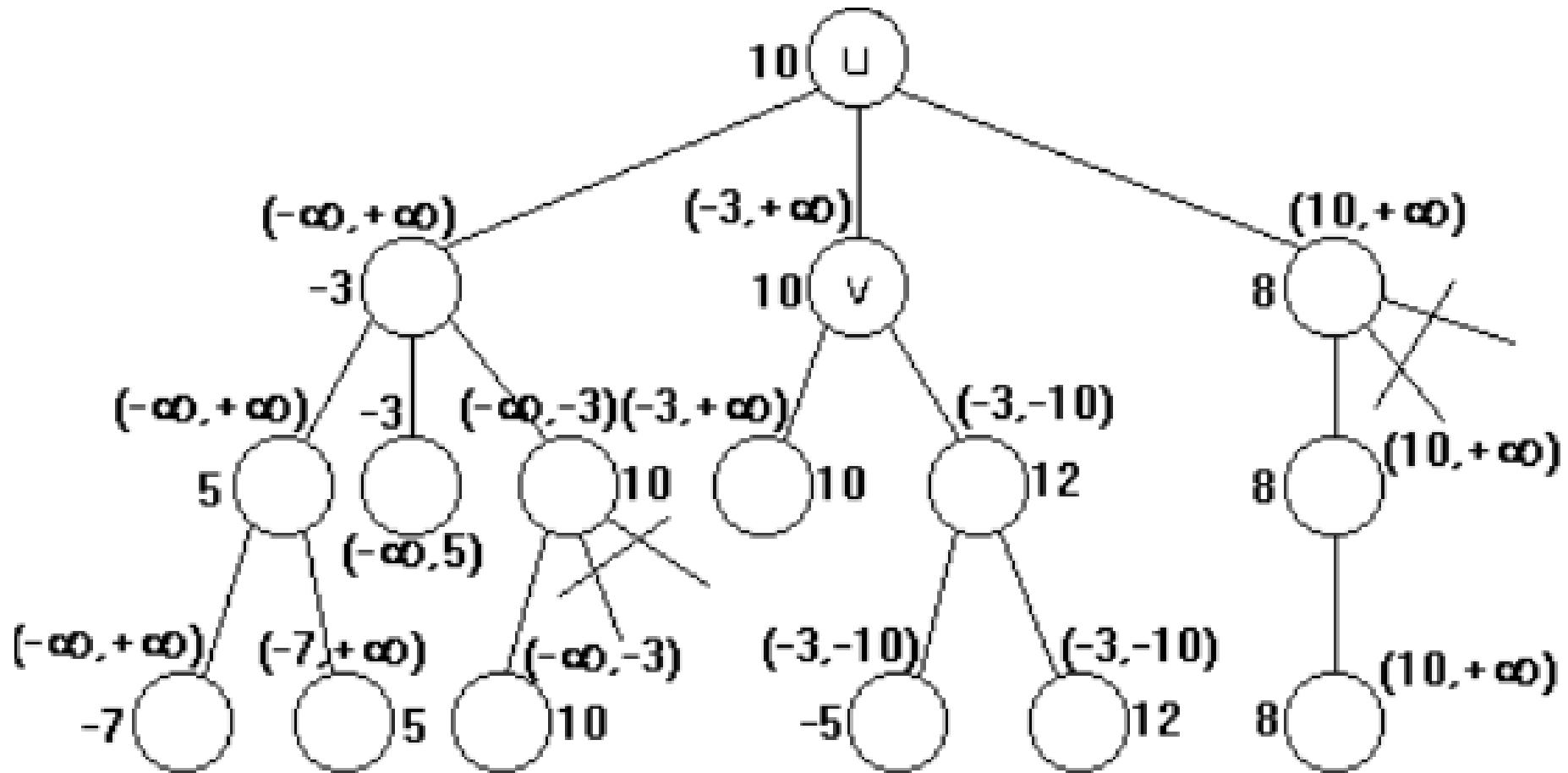
 alpha:=MinVal(w, alpha, beta);

 v:=w;

 end;

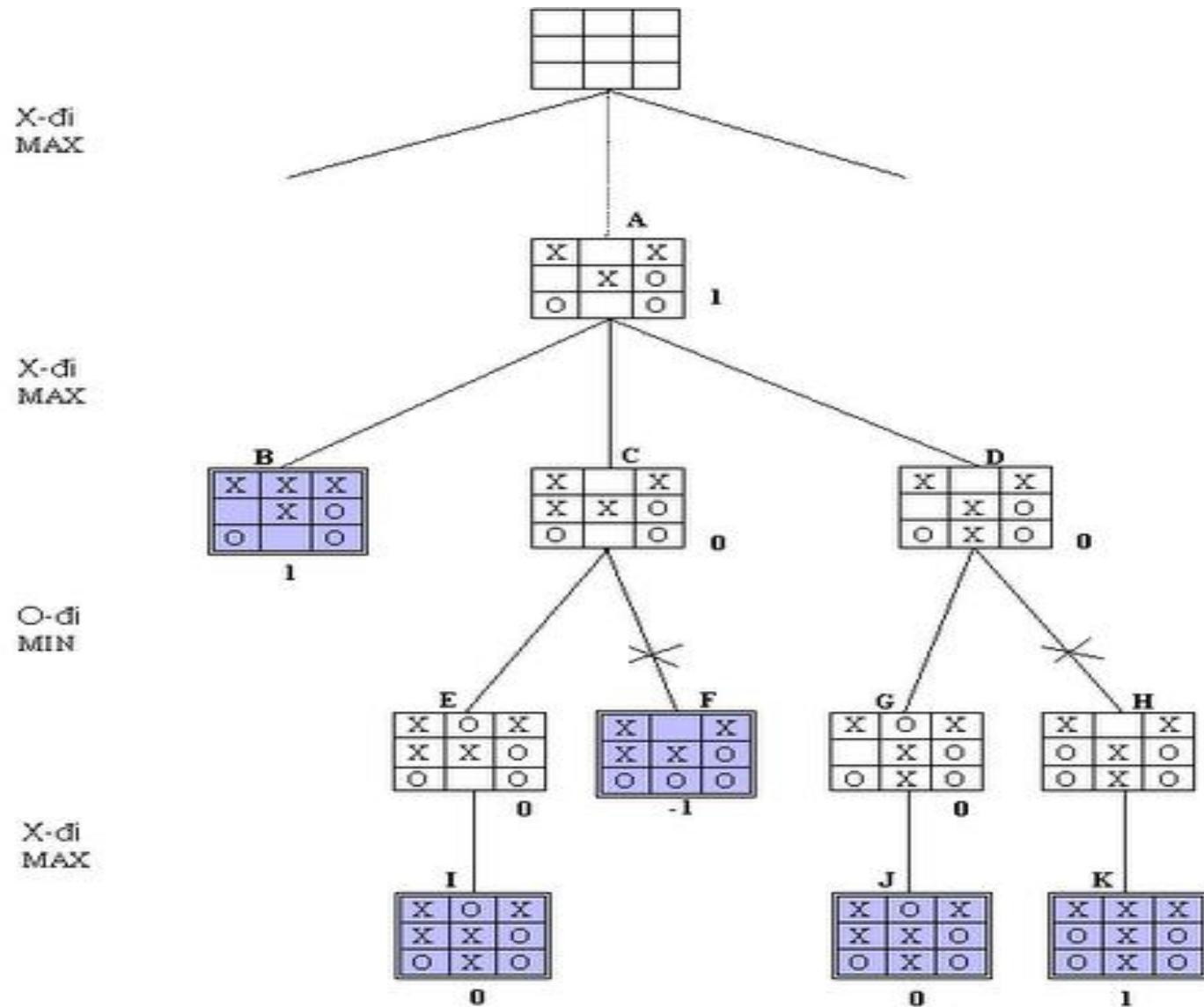
 end;

Xác định giá trị các đỉnh bằng a-β cắt cùt



Bài tập 1

■ Giải bài toán tic-tac-toe 9 ô bằng giải thuật α - β cut ?



Định trị cây trò chơi bằng kỹ thuật cắt tia alpha-beta

Bài tập 2

 **Bài I – trang 36**

Chương 3

Logic và suy diễn

Chương 3. Logic và suy diễn

- Biểu diễn bài toán bằng logic
- Logic mệnh đề
- Logic vị từ

Biểu diễn tri thức

- Tri thức?
 - Thông tin, hiểu biết về lĩnh vực nào đấy
 - Biểu diễn dưới dạng câu
- Cơ sở tri thức (CSTT)
 - Tập tri thức
 - Được biểu diễn dưới dạng nào đấy tạo thành ngôn ngữ tri thức
- Suy diễn
 - Liên kết các sự kiện thu nhận được từ môi trường với các tri thức đã có trong CSTT để rút ra kết luận và hành động hơn lý

Logic mệnh đề

Logic mệnh đề

- Mệnh đề là các phát biểu chỉ có thể nhận giá trị chân lý **đúng** (true, T, 1) hoặc **sai** (false, F, 0)
- Ví dụ
 - 2 nhân 2 bằng 4
 - Hà Nội là thủ đô của Việt Nam
 - Chiều cao trung bình của người Việt Nam là 1m65
 - Nếu trời mưa thì Nam không đi chơi
 - ...
- Ký hiệu mệnh đề bởi các chữ cái: a,b,p,q,...

Logic mệnh đề

- Các phép toán logic
 - Phép hội (and, và) \wedge
 - Phép tuyễn (or, hoặc) \vee
 - Phép phủ định (not) \neg
 - Kéo theo \rightarrow
 - Tương đương «»
- Bảng chân lý ý nghĩa của các phép toán logic

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
F	F	T	F	F	T	T
F	T	T	F	T	T	F
T	F	F	F	T	F	F
T	T	F	T	T	T	T

Logic mệnh đề

- Các phép logic hội (và), tuyễn(hoặc) có tính chất giao hoán, kết hợp, phân phối
- Thứ tự ưu tiên: phủ định, kéo theo, tương đương, hội, tuyễn.
 - Dùng () nhóm các mệnh đề để quy định thứ tự ưu tiên
 - Ví dụ: $(A \wedge B \vee C) \rightarrow D \wedge E$

Logic mệnh đề

- Biểu thức mệnh đề là sự kết hợp của các mệnh đề bởi các phép toán logic
- Giá trị của 1 biểu thức logic mệnh đề được xác định bởi bảng chân lý
 - Ví dụ: thiết lập bảng chân lý cho biểu thức $(A \vee B) \wedge (B \vee \neg D)$
- Biểu thức hằng: luôn đúng, luôn sai trong mọi trường hợp
 - $A \vee \neg A$ luôn đúng
 - $A \wedge \neg A$ luôn sai

Ý nghĩa của phép kítса \wedge , \neg

Logic mệnh đề

- Chỉ cần 2 phép toán (and, not) hay (or, not) là có thể biểu diễn mọi biểu thức logic
- **Dạng chuẩn tắc hội**
 - Hội của các tuyễn
 $(p_1 \vee \dots \vee p_n) \wedge \dots \wedge (q_1 \vee \dots \vee q_m)$
- **Dạng chuẩn tuyễn**
 - Tuyễn của $(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D) \wedge \dots \wedge (p_n \vee \neg q_m)$
- **Dạng Horn**
 - Hội của các câu Horn (số mệnh đề dương ≤ 1)

Chuẩn hóa về dạng chuẩn hội

- 2 biểu thức A và B là tương đương nhau ($A \equiv B$) nếu chúng có cùng giá trị chân lý (cùng đúng hoặc cùng sai) trong mọi trường hợp
- Luật Demorgan

$$\neg(A \wedge B) \equiv \neg A \vee \neg B$$

$$\neg(A \vee B) \equiv \neg A \wedge \neg B$$

- Luật giao hoán
 - $A \vee B \equiv B \vee A$
 - $A \wedge B \equiv B \wedge A$
- Luật kết hợp

$$A \wedge (B \wedge C) \equiv (A \wedge B) \wedge C$$

$$A \vee (B \vee C) \equiv (A \vee B) \vee C$$

Suy diễn trong logic mệnh đề

- Bài toán

- Cho biết tập giả thiết $\{gt_1, gt_2, \dots, gt_n\}$, mỗi giả thiết là 1 biểu thức mệnh đề đúng
- Cần suy ra kết luận kl (tức chứng minh kết luận kl đúng)
- Giải bài toán tương đương với việc chứng minh
 - $gt_1 \wedge gt_2 \wedge \dots \wedge gt_n \rightarrow kl$

- Phương pháp chứng minh

- Cách 1: sử dụng bảng chân lý

Chứng minh bằng bảng chân lý

- Cho giả thiết GT = $\{(p \vee r) \wedge (q \vee \neg r)\}$
- Chứng minh $p \vee q$

p	q	r	$p \vee r$	$q \vee \neg r$	$(p \vee r) \wedge (q \vee \neg r)$	$p \vee q$
0	0	0	0	1	0	0
0	0	1	1	0	0	0
0	1	0	0	1	0	1
0	1	1	1	1	1	1
1	0	0	1	1	1	1
1	0	1	1	0	0	1
1	1	0	1	1	1	1
1	1	1	1	1	1	1

Phương pháp Robinson

● Phương pháp chứng minh phản chứng

1) Giả sử kết luận sai ($\neg kl$) thì $(gt_1 \wedge gt_2 \wedge \dots \wedge gt_n \wedge \neg kl) \rightarrow$ vô lý

1) Bước 0: đưa thêm $\neg kl$ vào tập giả thiết gt

1) Bước 1: chuẩn hóa về dạng chẩn hội

2) Sử dụng 7 phép biến đổi sau để chuẩn hóa:

1. $a \rightarrow b \equiv \neg a \vee b$
2. $(a \wedge b) \rightarrow c \equiv \neg a \vee \neg b \vee c$
3. $a \rightarrow (b \wedge c) \equiv (\neg a \vee b) \wedge (\neg a \vee c)$
4. $(a \vee b) \rightarrow c \equiv (\neg a \vee c) \wedge (\neg b \vee c)$
5. $a \rightarrow (b \vee c) \equiv \neg a \vee b \vee c$
6. $a \rightarrow (b \rightarrow c) \equiv \neg a \vee \neg b \vee c$
7. $(a \rightarrow b) \rightarrow c \equiv (a \vee c) \wedge (\neg b \vee c)$

Phương pháp Robinson

- **Bước 2: tách các dòng chuẩn thành các dòng con (mỗi dòng con là 1 biểu thức nhận giá trị đúng)**
 $(p_1 \vee \dots \vee p_n) \wedge \dots \wedge (q_1 \vee \dots \vee q_m)$ tách thành:
 $(p_1 \vee \dots \vee p_n)$
...
 $(q_1 \vee \dots \vee q_m)$
- Nhận xét:
 - Bước 1 và 2 là tiền xử lý
 - Mỗi bài toán được biến đổi thành 1 tập hữu hạn các dòng chuẩn. Mỗi dòng chuẩn chỉ có phép hợp \vee của các mệnh đề, mỗi mệnh đề có thể ở dạng khẳng định (p) hoặc phủ định ($\neg p$)

Phương pháp Robinson

- **Bước 3: quá trình hợp giải.** Là quá trình lặp, mỗi bước lặp gồm:
 - 3.1. Tìm 2 dòng bất kì trong số 2 dòng đã có chứa
 - $p \vee q$ (dòng i)
 - $\neg p \vee r$ (dòng j)
 - Cơ chế chọn: FIFO, LIFO, mềm dẻo
 - 3.2. Ghép 2 dòng đó tạo thành dòng k
 - $q \vee r$
- Lặp cho đến khi ta có:

Phương pháp Robinson

- Tóm lại phương pháp chứng minh Robinson là
 - $gt_1 \wedge gt_2 \wedge \dots \wedge gt_n \rightarrow kl$
 - $\equiv gt_1 \wedge gt_2 \wedge \dots \wedge gt_n \wedge \neg kl \rightarrow \{u, \neg u\}$ vô lý

● Robinson procedure

Begin

1. Thêm $\neg kl$ vào tập giả thiết G
 2. Chuyển các biểu thức trong G thành dạng chuẩn hội
 3. Tách các câu chuẩn hội trong G thành các câu con chuẩn tuyễn
 4. Repeat
 - 4.1. Chọn 2 câu A, B thuộc G
 - 4.2. Nếu A và B phân giải được thì
 - 4.2.1. Tính Res(A,B)
 - 4.2.2. Nếu Res(A,B) sinh ra câu mới thì thêm Res(A,B) vào G
- until nhận được câu rỗng hoặc không có câu mới xuất hiện

End

Phương pháp Robinson

. Ví dụ

1. Sắp Tết
2. Có thưởng
3. Sắp Tết ∧ có thưởng → vui
4. Vui → chơi
5. Chơi → thăm nơi đẹp và mới
6. Thăm nơi đẹp và mới → thăm Mỹ đình
7. Thăm Mỹ đình ∧ Mỹ đình gần nhà bà ngoại
→ thăm bà ngoại
8. Thăm bà ngoại → thăm bà nội

Phương pháp Robinson

- Giả sử điều cần chứng minh là sai. Ta có \neg Mỹ đình v \neg bà nội v \neg bà ngoại
- Bước 1 và 2: chuẩn hóa và tách dòng

1. Tết
2. Thưởng
3. \neg Tết v \neg Thưởng v Vui
4. \neg Vui v Chơi
5. \neg Chơi v Đẹp
6. \neg Chơi v Mới
7. \neg Đẹp v \neg Mới v Mỹ Đình
8. \neg Mỹ Đình v \neg Gần v Bà Ngoại
9. \neg Bà ngoại v Bà nội
10. Gần
11. \neg Mỹ đình v \neg bà nội v \neg bà ngoại

Phương pháp Robinson

- Bước 3: hợp giải

Hàng i	hàng j
1	3
2	3
3	4
4	5
4	6
5	7
6	7
7	8,11
8	9
8	10
9	11

Danh sách cặp hàng có
thể hợp giải (IIFO > FIFO))

Phương pháp Robinson

- Bước 3: hợp giải

- Res(9,11): 12. \neg Mỹ đình v \neg Bà ngoại

Hàng i	hàng j
1	3
2	3
3	4
4	5
4	6
5	7
6	7
7	8,11
8	9
8	10
9	11

Danh sách cặp hàng có
thể hợp giải (LIFO > FIFO)

Phương pháp Robinson

- Bước 3: hợp giải

- Res(9,11): 12. \neg Mỹ đình v \neg Bà ngoại

Hàng i	hàng j
1	3
2	3
3	4
4	5
4	6
5	7
6	7
7	8,11
8	9
8	10
12	7
12	8

Danh sách cặp hàng có
thể hợp giải (LIFO > FIFO)

Phương pháp Robinson

• Bước 3: hợp giải

Res(9,11): 12. \neg Mỹ đình v \neg Bà ngoại

Res(12,8): 13. \neg Mỹ Đình v \neg Gần

Hàng i	hàng j
1	3
2	3
3	4
4	5
4	6
5	7
6	7
7	8,11
8	9
8	10
12	7
12	8

Danh sách cặp hàng có
thể hợp giải (LIFO > FIFO)

Phương pháp Robinson

- **Bước 3: hợp giải**

- Res(9,11): 12. \neg Mỹ đình v \neg Bà ngoại

- Res(12,8): 13. \neg Mỹ Đình v \neg Gần

Hàng i	hàng j
1	3
2	3
3	4
4	5
4	6
5	7
6	7
7	8,11
8	9
8	10
12	7
13	7
13	10

Phương pháp Robinson

• Bước 3: hợp giải

- Res(9,11): 12. \neg Mỹ đình v \neg Bà ngoại
- Res(12,8): 13. \neg Mỹ Đình v \neg Gần
- Res(13,10): 14. \neg Mỹ Đình

Hàng i	hàng j
1	3
2	3
3	4
4	5
4	6
5	7
6	7
7	8,11
8	9
8	10
12	7
13	7
13	10

Phương pháp Robinson

• Bước 3: hợp giải

- Res(9,11): 12. \neg Mỹ đình v \neg Bà ngoại
- Res(12,8): 13. \neg Mỹ Đình v \neg Gần
- Res(13,10): 14. \neg Mỹ Đình

Hàng i	hàng j
1	3
2	3
3	4
4	5
4	6
5	7
6	7
7	8,11
8	9
8	10
12	7
13	7
14	7

Phương pháp Robinson

• Bước 3: hợp giải

- Res(9,11): 12. \neg Mỹ đình v \neg Bà ngoại
- Res(12,8): 13. \neg Mỹ Đình v \neg Gần
- Res(13,10): 14. \neg Mỹ Đình
- Res(14,7): 15. \neg Đẹp v \neg Mới

Hàng i	hàng j
1	3
2	3
3	4
4	5
4	6
5	7
6	7
7	8,11
8	9
8	10
12	7
13	7
14	7

Phương pháp Robinson

• Bước 3: hợp giải

- Res(9,11): 12. \neg Mỹ đình v \neg Bà ngoại
- Res(12,8): 13. \neg Mỹ Đình v \neg Gần
- Res(13,10): 14. \neg Mỹ Đình
- Res(14,7): 15. \neg Đẹp v \neg Mới

Hàng i	hàng j
1	3
2	3
3	4
4	5
4	6
5	7
6	7
7	8,11
8	9
8	10
12	7
13	7
15	5
15	6

Phương pháp Robinson

• Bước 3: hợp giải

- Res(9,11): 12. \neg Mỹ đình v \neg Bà ngoại
- Res(12,8): 13. \neg Mỹ Đình v \neg Gần
- Res(13,10): 14. \neg Mỹ Đình
- Res(14,7): 15. \neg Đẹp v \neg Mới
- Res(15,6): 16. \neg Đẹp v \neg Chơi

Hàng i	hàng j
1	3
2	3
3	4
4	5
4	6
5	7
6	7
7	8,11
8	9
8	10
12	7
13	7
15	5
15	6

Phương pháp Robinson

• Bước 3: hợp giải

- Res(9,11): 12. \neg Mỹ đình v \neg Bà ngoại
- Res(12,8): 13. \neg Mỹ Đình v \neg Gần
- Res(13,10): 14. \neg Mỹ Đình
- Res(14,7): 15. \neg Đẹp v \neg Mới
- Res(15,6): 16. \neg Đẹp v \neg Chơi

Hàng i	hàng j
1	3
2	3
3	4
4	5
4	6
5	7
6	7
7	8,11
8	9
8	10
12	7
13	7
15	5
16	4
16	5

Phương pháp Robinson

• Bước 3: hợp giải

- Res(9,11): 12. \neg Mỹ đình v \neg Bà ngoại
- Res(12,8): 13. \neg Mỹ Đình v \neg Gần
- Res(13,10): 14. \neg Mỹ Đình
- Res(14,7): 15. \neg Đẹp v \neg Mới
- Res(15,6): 16. \neg Đẹp v \neg Chơi
- Res(16,5): 17. \neg Chơi

Hàng i	hàng j
1	3
2	3
3	4
4	5
4	6
5	7
6	7
7	8,11
8	9
8	10
12	7
13	7
15	5
16	4
16	5

Phương pháp Robinson

• Bước 3: hợp giải

- Res(9,11): 12. \neg Mỹ đình v \neg Bà ngoại
- Res(12,8): 13. \neg Mỹ Đình v \neg Gần
- Res(13,10): 14. \neg Mỹ Đình
- Res(14,7): 15. \neg Đẹp v \neg Mới
- Res(15,6): 16. \neg Đẹp v \neg Chơi
- Res(16,5): 17. \neg Chơi

Hàng i	hàng j
1	3
2	3
3	4
4	5
4	6
5	7
6	7
7	8,11
8	9
8	10
12	7
13	7
15	5
16	4
17	4

Phương pháp Robinson

• Bước 3: hợp giải

- Res(9,11): 12. \neg Mỹ đình v \neg Bà ngoại
- Res(12,8): 13. \neg Mỹ Đình v \neg Gần
- Res(13,10): 14. \neg Mỹ Đình
- Res(14,7): 15. \neg Đẹp v \neg Mới
- Res(15,6): 16. \neg Đẹp v \neg Chơi
- Res(16,5): 17. \neg Chơi
- Res(17,4): 18. \neg Vui

Hàng i	hàng j
1	3
2	3
3	4
4	5
4	6
5	7
6	7
7	8,11
8	9
8	10
12	7
13	7
15	5
16	4
17	4

Phương pháp Robinson

• Bước 3: hợp giải

- Res(9,11): 12. \neg Mỹ đình v \neg Bà ngoại
- Res(12,8): 13. \neg Mỹ Đình v \neg Gần
- Res(13,10): 14. \neg Mỹ Đình
- Res(14,7): 15. \neg Đẹp v \neg Mới
- Res(15,6): 16. \neg Đẹp v \neg Chơi
- Res(16,5): 17. \neg Chơi
- Res(17,4)": 18. \neg Vui

Hàng i	hàng j
1	3
2	3
3	4
4	5
4	6
5	7
6	7
7	8,11
8	9
8	10
12	7
13	7
15	5
16	4
18	3

Phương pháp Robinson

• Bước 3: hợp giải

- Res(9,11): 12. \neg Mỹ đình v \neg Bà ngoại
- Res(12,8): 13. \neg Mỹ Đình v \neg Gần
- Res(13,10): 14. \neg Mỹ Đình
- Res(14,7): 15. \neg Đẹp v \neg Mới
- Res(15,6): 16. \neg Đẹp v \neg Chơi
- Res(16,5): 17. \neg Chơi
- Res(17,4): 18. \neg Vui
- Res(18,3): 19. \neg Sắp tết v \neg Có thưởng

Hàng i	hàng j
1	3
2	3
3	4
4	5
4	6
5	7
6	7
7	8,11
8	9
8	10
12	7
13	7
15	5
16	4
18	3

Phương pháp Robinson

• Bước 3: hợp giải

- Res(9,11): 12. \neg Mỹ đình v \neg Bà ngoại
- Res(12,8): 13. \neg Mỹ Đình v \neg Gần
- Res(13,10): 14. \neg Mỹ Đình
- Res(14,7): 15. \neg Đẹp v \neg Mới
- Res(15,6): 16. \neg Đẹp v \neg Chơi
- Res(16,5): 17. \neg Chơi
- Res(17,4): 18. \neg Vui
- Res(18,3): 19. \neg Sắp tết v \neg Có thưởng

Hàng i	hàng j
1	3
2	3
3	4
4	5
4	6
5	7
6	7
7	8,11
8	9
8	10
12	7
13	7
15	5
16	4
19	1
16	2

Phương pháp Robinson

• Bước 3: hợp giải

- Res(9,11): 12. \neg Mỹ đình v \neg Bà ngoại
- Res(12,8): 13. \neg Mỹ Đình v \neg Gần
- Res(13,10): 14. \neg Mỹ Đình
- Res(14,7): 15. \neg Đẹp v \neg Mới
- Res(15,6): 16. \neg Đẹp v \neg Chơi
- Res(16,5): 17. \neg Chơi
- Res(17,4): 18. \neg Vui
- Res(18,3): 19. \neg Sắp tết v \neg Có thưởng
- Res(19,2): 20. \neg Sắp tết

Hàng i	hàng j
1	3
2	3
3	4
4	5
4	6
5	7
6	7
7	8,11
8	9
8	10
12	7
13	7
15	5
16	4
19	1

Phương pháp Robinson

• Bước 3: hợp giải

- Res(9,11): 12. \neg Mỹ đình v \neg Bà ngoại
- Res(12,8): 13. \neg Mỹ Đình v \neg Gần
- Res(13,10): 14. \neg Mỹ Đình
- Res(14,7): 15. \neg Đẹp v \neg Mới
- Res(15,6): 16. \neg Đẹp v \neg Chơi
- Res(16,5): 17. \neg Chơi
- Res(17,4): 18. \neg Vui
- Res(18,3): 19. \neg Sắp tết v \neg Có thưởng
- Res(19,2): 20. \neg Sắp tết

Hàng i	hàng j
1	3
2	3
3	4
4	5
4	6
5	7
6	7
7	8,11
8	9
8	10
12	7
13	7
15	5
16	4
19	1
20	1

Phương pháp Robinson

• Bước 3: hợp giải

Res(9,11): 12. \neg Mỹ đình v \neg Bà ngoại

Res(12,8): 13. \neg Mỹ Đình v \neg Gần

Res(13,10): 14. \neg Mỹ Đình

Res(14,7): 15. \neg Đẹp v \neg Mới

Res(15,6): 16. \neg Đẹp v \neg Chơi

Res(16,5): 17. \neg Chơi

Res(17,4): 18. \neg Vui

Res(18,3): 19. \neg Sắp tết v \neg Có thưởng

Res(19,2): 20. \neg Sắp tết

Res(20,1): 21. vô lý

Hàng i	hàng j
1	3
2	3
3	4
4	5
4	6
5	7
6	7
7	8,11
8	9
8	10
12	7
13	7
15	5
16	4
19	1
20	1

Phương pháp Robinson

- Bài tập: cho các giả thiết sau đây là đúng

$$1. v \Rightarrow w$$

$$2. (d \wedge m \wedge b) \Rightarrow w$$

$$3. u \Rightarrow (v \vee \neg f)$$

$$4. (b \wedge c \wedge a) \Rightarrow v$$

$$5. (u \wedge e) \Rightarrow (\neg m \vee a)$$

$$6. e \Rightarrow (f \wedge m)$$

$$7. (e \wedge f) \Rightarrow u$$

Chứng minh bằng các luật suy diễn

• Các luật suy diễn trong logic mệnh đề

$$1) \frac{\alpha}{\beta} \quad \alpha \Rightarrow \beta$$

Modus Ponens

$$2) \frac{\neg\beta}{\neg\alpha} \quad \alpha \Rightarrow \beta$$

Modus Tolens

$$3) \frac{\beta \Rightarrow \gamma}{\alpha \Rightarrow \gamma} \quad \alpha \Rightarrow \beta$$

Luật bắc cầu

$$4) \frac{\begin{array}{c} \alpha \\ \beta \end{array}}{\alpha \wedge \beta} \text{ and } \frac{\beta}{\alpha \vee \beta}$$

**Đưa vào hội, đưa
vào tuyễn**

$$5) \frac{\alpha \wedge \beta}{\alpha(\text{or } \beta)}$$

Loại bỏ hội

$$6) \frac{\neg\beta \vee \gamma}{\alpha \vee \gamma} \quad \alpha \vee \beta$$

Luật phân giải

Chứng minh bằng các luật suy diễn

Ví dụ: KB gồm các câu:

1. $Q \wedge S \Rightarrow G \wedge H$
2. $P \Rightarrow Q$
3. $R \Rightarrow S$
4. P
5. R

hãy chứng minh G

Giải:

(2)(4) $\Rightarrow Q$ (6) (Áp dụng luật MP)

(3)(5) $\Rightarrow S$ (7) (Áp dụng luật MP)

(6)(7) $\Rightarrow Q \wedge S$ (8) (Áp dụng luật đưa vào hội)

(1)(8) $\Rightarrow G \wedge H$ (9) (Áp dụng luật MP)

(9) $\Rightarrow G$ (Áp dụng luật loại bỏ hội)

Chứng minh bằng suy diễn tiến, lùi

- Câu chuẩn Horn có dạng

$$Q \vee \neg P_1 \vee \neg P_2 \vee \dots \vee \neg P_m$$

$$\text{Hay } P_1 \wedge P_2 \wedge \dots \wedge P_m \rightarrow Q$$

- Câu Horn còn gọi là luật nếu-thì (if-then) và viết lại dưới dạng:

Nếu P_1 và P_2 ... và P_m thì Q

- Trong đó:

P_1, P_2, \dots, P_m được gọi là các điều kiện của luật

Q là kết luận của luật

- Luật Modus Ponens trên các câu Horn

$$\alpha_1, \dots, \alpha_n, \quad \alpha_1 \wedge \dots \wedge \alpha_n \Rightarrow \beta$$

$$\beta$$

Suy diễn tiến, lùi

- Cho cơ sở tri thức dưới dạng luật

- $R = \{r_1, r_2, \dots, r_n\}$ trong đó mỗi r_i là một luật

- $r_i: (p_{i_1}^i \wedge p_{i_2}^i \wedge \dots \wedge p_{i_m}^i) \rightarrow q^i$

Suy diễn tiến, lùi

- Trả lời cho bài toán
 - Yes (suy ra được kết luận KL): cần đưa ra vết suy diễn $VET=\{r_1, r_2, \dots, r_k\}$ là tập con của R
 - $GT \rightarrow_{r_1} TG_1 \rightarrow_{r_2} TG_2 \rightarrow \dots \rightarrow_{r_k} TG_k$ chứa KL
 - No (không suy ra được kết luận KL): áp dụng mọi luật r_i trong R và mọi giả thiết trong GT nhưng không có cách nào suy diễn $GT \rightarrow^* KL$
- Thủ tục suy diễn
 - Procedure SD(R : tập luật; GT, KL : tập sự kiện, var kq : boolean; VET : tập luật);
- Qui tắc suy diễn
 - Xuất phát từ giả thiết dần đi đến kết luận: suy diễn tiến (forward inference)
 - Xuất phát từ kết luận lùi dần đến giả thiết: suy diễn lùi (backward inference)

Suy diễn tiến

- Áp dụng luật MP (Modus Ponens) thể hiện cơ chế tăng trưởng tri thức về tình huống



Dạng tổng quát

- Tức là {a, a → b} →^{MP} {a, a → b, b}
- Hay {a} →^{MP} {a, b}

Suy diễn tiến

- Cho tập luật R biểu diễn các mối quan hệ trong tam giác như sau:

$$1. a \wedge b \wedge C \rightarrow c$$

$$3. a \wedge b \wedge mb \rightarrow c$$

$$4. A \wedge B \rightarrow C$$

$$6. b \wedge hc \rightarrow A$$

$$7. a \wedge R \rightarrow A$$

$$9. a \wedge b \wedge c \rightarrow P$$

$$10. a \wedge b \wedge c \rightarrow p$$

$$12. a \wedge ha \rightarrow S$$

$$13. b \wedge a \wedge C \rightarrow S$$

$$15. b \wedge S \rightarrow hb$$

$$2. a \wedge b \wedge ma \rightarrow c$$

$$5. a \wedge hc \rightarrow B$$

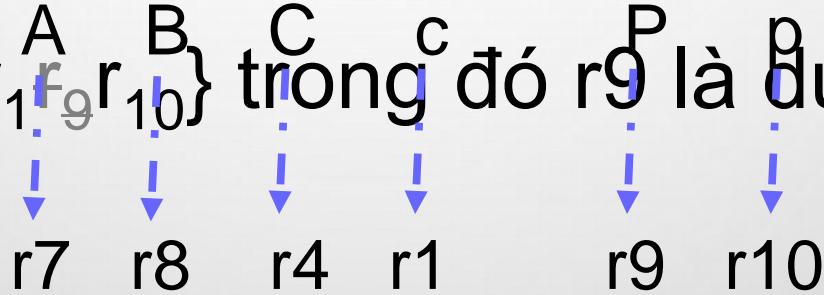
$$8. b \wedge R \rightarrow B$$

$$11. a \wedge b \wedge c \rightarrow mc$$

$$14. a \wedge b \wedge c \wedge p \rightarrow S$$

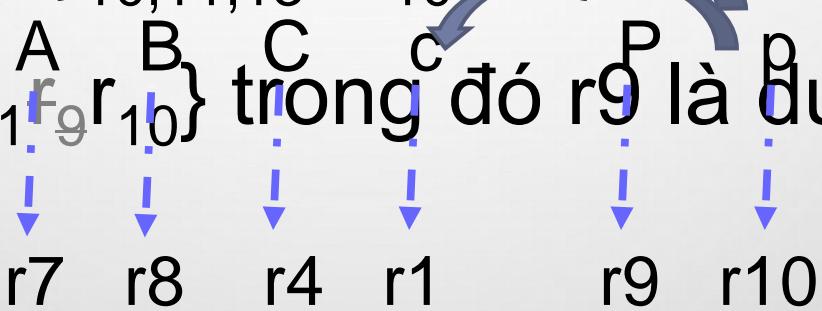
Suy diễn tiến

- $\{a,b,R\}_{7,8} \xrightarrow{7}^{\text{MP}} \{a,b,R,A\}_8 \xrightarrow{8}^{\text{MP}}$
 $\{a,b,R,A,B\}_4 \xrightarrow{4}^{\text{MP}} \{a,b,R,A,B,C\}_{1,13} \xrightarrow{1}^{\text{MP}}$
 $\{a,b,R,A,B,C,c\}_{9,10,11,13} \xrightarrow{9}^{\text{MP}}$
 $\{a,b,R,A,B,C,P\}_{10,11,13} \xrightarrow{10}^{\text{MP}} \{a,b,R,A,B,C,P,p\}$
- $VET = \{r_7^a r_8^b r_4^R r_1^A r_9^B r_{10}^C\}$ trong đó r_9^P là dư thừa



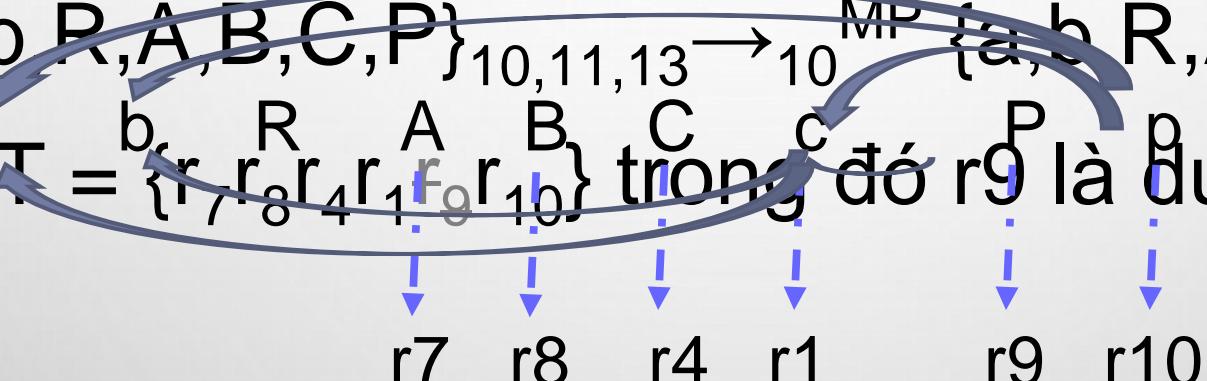
- Kết luận:

Suy diễn tiến

- $\{a,b,R\}_{7,8} \xrightarrow{7}^{MP} \{a,b,R,A\}_8 \xrightarrow{8}^{MP}$
 $\{a,b,R,A,B\}_4 \xrightarrow{4}^{MP} \{a,b,R,A,B,C\}_{1,13} \xrightarrow{1}^{MP}$
 $\{a,b,R,A,B,C,c\}_{,9,10,11,13} \xrightarrow{9}^{MP}$
 ~~$\{a,b,R,A,B,C,P\}_{10,11,13} \xrightarrow{10}^{MP} \{a,b,R,A,B,C,P,p\}$~~
- VET = $\{r_7 r_8 r_4 r_1 r_9 r_{10}\}$ trong đó r_9 là dư thừa


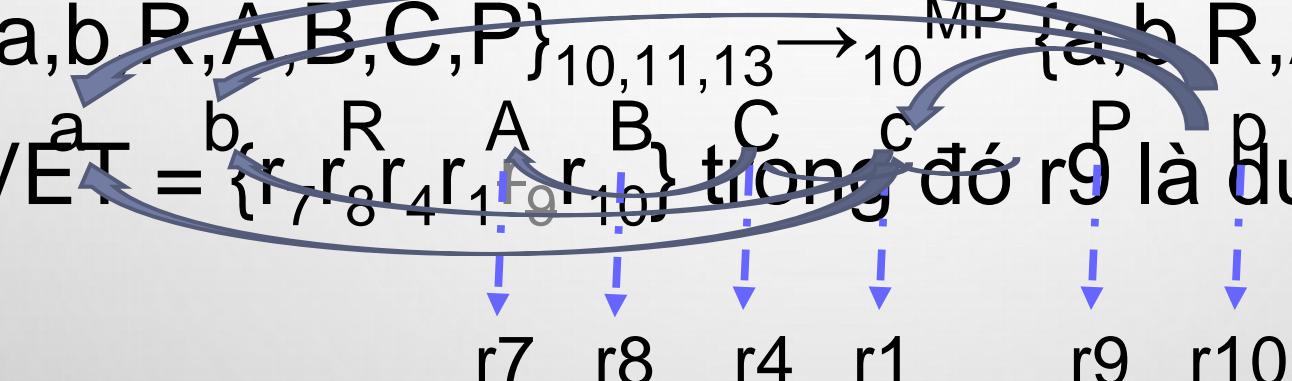
- Kết luận:

Suy diễn tiến

- $\{a,b,R\}_{7,8} \xrightarrow{7}^{MP} \{a,b,R,A\}_8 \xrightarrow{8}^{MP}$
 $\{a,b,R,A,B\}_4 \xrightarrow{4}^{MP} \{a,b,R,A,B,C\}_{1,13} \xrightarrow{1}^{MP}$
 $\{a,b,R,A,B,C,c\}_{,9,10,11,13} \xrightarrow{9}^{MP}$
 $\{a,b,R,A,B,C,P\}_{10,11,13} \xrightarrow{10}^{MP} \{a,b,R,A,B,C,P,p\}$
 - $VET = \{r_7, r_8, r_4, r_1, r_9, r_{10}\}$ trong đó r_9 là dư thừa
- 

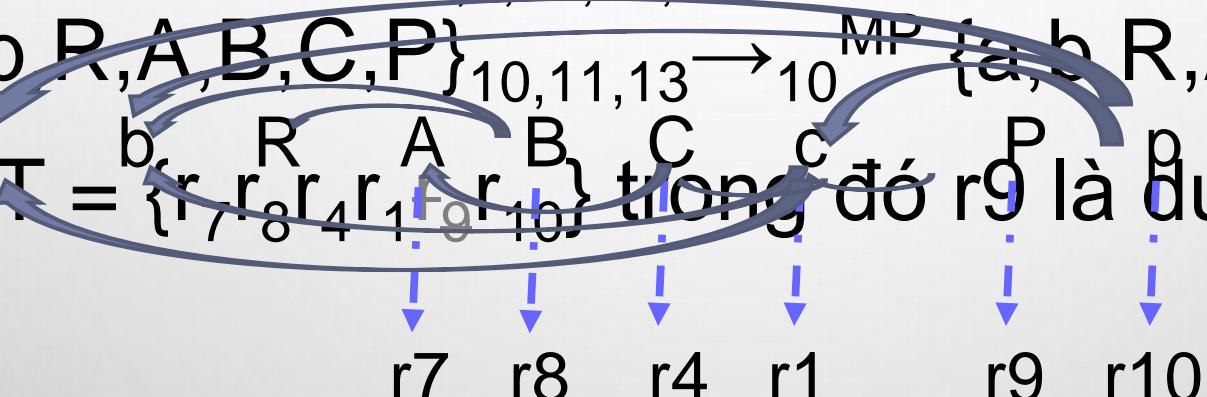
- Kết luận:

Suy diễn tiến

- $\{a,b,R\}_{7,8} \xrightarrow{7}^{MP} \{a,b,R,A\}_8 \xrightarrow{8}^{MP}$
 $\{a,b,R,A,B\}_4 \xrightarrow{4}^{MP} \{a,b,R,A,B,C\}_{1,13} \xrightarrow{1}^{MP}$
 $\{a,b,R,A,B,C,c\}_{,9,10,11,13} \xrightarrow{9}^{MP}$
 $\{a,b,R,A,B,C,P\}_{10,11,13} \xrightarrow{10}^{MP} \{a,b,R,A,B,C,P,p\}$
 - VET = $\{r_7, r_8, r_4, r_1, r_9, r_{10}\}$ trong đó r_9 là dư thừa
- 

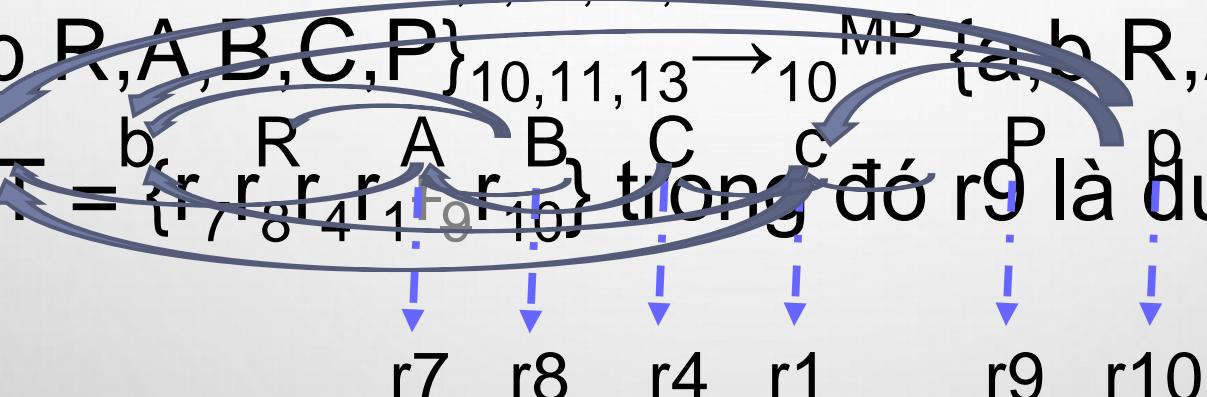
- Kết luận:

Suy diễn tiến

- $\{a,b,R\}_{7,8} \xrightarrow{7}^{MP} \{a,b,R,A\}_8 \xrightarrow{8}^{MP}$
 $\{a,b,R,A,B\}_4 \xrightarrow{4}^{MP} \{a,b,R,A,B,C\}_{1,13} \xrightarrow{1}^{MP}$
 $\{a,b,R,A,B,C,c\}_{,9,10,11,13} \xrightarrow{9}^{MP}$
 $\{a,b,R,A,B,C,P\}_{10,11,13} \xrightarrow{10}^{MP} \{a,b,R,A,B,C,P,p\}$
 - VET = $\{r_7, r_8, r_4, r_1, r_9, r_{10}\}$ trong đó r_9 là dư thừa
- 

- Kết luận:

Suy diễn tiến

- $\{a,b,R\}_{7,8} \xrightarrow{7}^{\text{MP}} \{a,b,R,A\}_8 \xrightarrow{8}^{\text{MP}}$
 $\{a,b,R,A,B\}_4 \xrightarrow{4}^{\text{MP}} \{a,b,R,A,B,C\}_{1,13} \xrightarrow{1}^{\text{MP}}$
 $\{a,b,R,A,B,C,c\}_{,9,10,11,13} \xrightarrow{9}^{\text{MP}}$
 $\{a,b,R,A,B,C,P\}_{10,11,13} \xrightarrow{10}^{\text{MP}} \{a,b,R,A,B,C,P,p\}$
 - $VET = \{r_7, r_8, r_4, r_1, r_9, r_{10}\}$ trong đó r_9 là dư thừa
- 
- Kết luận:

Suy diễn tiến

- Loại bỏ dư thừa trong vết?
 - Lần ngược mối quan hệ phụ thuộc của kết luận vào các trung gian cho tới giả thiết.
 - Những trung gian TGi nào không có quan hệ với KL (dù là gián tiếp) sẽ là dư thừa trong vết
- Từ giả thiết đến kết luận có nhiều vết đúng ứng với các cơ chế lựa chọn các luật (từ tập các luật thỏa mãn THOA) để áp dụng
 - Mềm dẻo: chọn tùy ý
 - Cứng nhắc: min, max, LIFO, FIFO

Suy diễn tiến

- Cơ chế max

- TG = {a,b,R}
- THOA = {r₇,r₈,
- VET = {

Suy diễn tiến

- Cơ chế max

- TG = {a,b,R,B}
- THOA = {r₇, r₈,
- VET = {r₈,

Suy diễn tiến

- Cơ chế max

- TG = {a,b,R,B,A}
- THOA = { $r_7, r_8,$
- VET = { $r_8, r_7,$

Suy diễn tiến

- Cơ chế max

- TG = {a,b,R,B,A}
- THOA = { $r_7, r_8, r_4,$
- VET = { $r_8, r_7,$

Suy diễn tiến

- Cơ chế max

- TG = {a,b,R,B,A,C}
- THOA = {f₇, f₈, f₄,
- VET = {r₈, r₇, r₄,

Suy diễn tiến

- Cơ chế max

- TG = {a,b,R,B,A,C}
- THOA = { $r_7, r_8, r_4, r_1, r_{13},$
- VET = { $r_8, r_7, r_4,$

Suy diễn tiến

- Cơ chế max

- TG = {a,b,R,B,A,C,S}
- THOA = { $r_7, r_8, r_4, r_1, r_{13},$
- VET = { $r_8, r_7, r_4, r_{13},$

Suy diễn tiến

- Cơ chế max

- TG = {a,b,R,B,A,C,S}
- THOA = {r₇, r₈, r₄, r₁, r₁₃, r₁₅, r₁₆,
- VET = {r₈, r₇, r₄, r₁₃,

Suy diễn tiến

- Cơ chế max

- TG = {a,b,R,B,A,C,S,r,
- THOA = {r₇,r₈,r₄,r₁,r₁₃,r₁₅,r₁₆,
- VET = {r₈,r₇,r₄,r₁₃,r₁₆,

Suy diễn tiến

- Cơ chế max

- TG = {a,b,R,B,A,C,S,r,h_b,
- THOA = {r₇,r₈,r₄,r₁,r₁₃,r₁₅,r₁₆,
- VET = {r₈,r₇,r₄,r₁₃,r₁₆,r₁₅,

Suy diễn tiến

- Cơ chế max

- TG = {a,b,R,B,A,C,S,r,h_b,c}
- THOA = {f₇,f₈,f₄,f₁,f₁₃,f₁₅,f₁₆}
- VET = {r₈,r₇,r₄,r₁₃,r₁₆,r₁₅,r₁,

Suy diễn tiến

- Cơ chế max

- TG = {a,b,R,B,A,C,S,r,h_b,c}
- THOA = {r₇, r₈, r₄, r₁, r₁₃, r₁₅, r₁₆, r₉, r₁₀, r₁₁,
- VET = {r₈, r₇, r₄, r₁₃, r₁₆, r₁₅, r₁,

Suy diễn tiến

- Cơ chế max

- TG = { $a, b, R, B, A, C, S, r, h_b, c, m_c,$
- THOA = { $r_7, r_8, r_4, r_1, r_{13}, r_{15}, r_{16}, r_9, r_{10}, r_{11},$
- VET = { $r_8, r_7, r_4, r_{13}, r_{16}, r_{15}, r_1, r_{11},$

Suy diễn tiến

- Cơ chế max

- TG = { $a, b, R, B, A, C, S, r, h_b, c, m_c, p$ }
- THOA = { $r_7, r_8, r_4, r_1, r_{13}, r_{15}, r_{16}, r_9, r_{10}, r_{11}$,
- VET = { $r_8, r_7, r_4, r_{13}, r_{16}, r_{15}, r_1, r_{11}, r_{10}$,

Suy diễn tiến

- Cơ chế max

- TG = { $a, b, R, B, A, C, S, r, h_b, c, m_c, p$ }
- THOA = { $r_7, r_8, r_4, r_1, r_{13}, r_{15}, r_{16}, r_9, r_{10}, r_{11}$ }
- VET = { $r_8, r_7, r_4, r_{13}, r_{16}, r_{15}, r_1, r_{11}, r_{10}$ }

Suy diễn tiến

- Cơ chế max?

- TG = {a,b,R,B,A,C,S,r,h_b,c,m_c,**p**}
- THOA = {f₇,f₈,f₄,f₁,f₁₃,f₁₅,f₁₆,r₉,f₁₀,f₁₁}
- VET = {r₈,r₇,r₄,r₁₃,r₁₆,r₁₅,r₁,r₁₁,r₁₀}

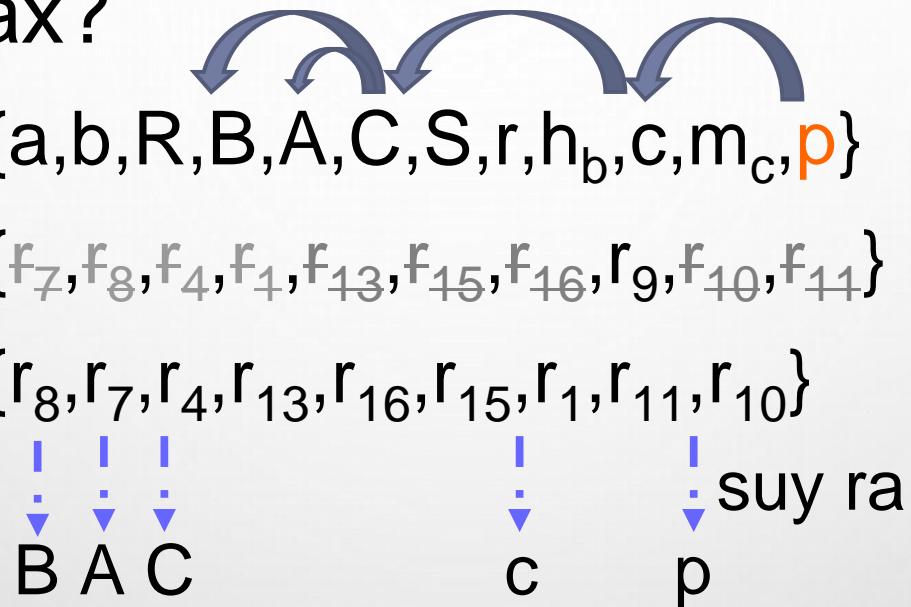
Suy diễn tiến

- ## • Cơ chế max?

- TG = {a,b,R,B,A,C,S,r,h_b,c,m_c,p}

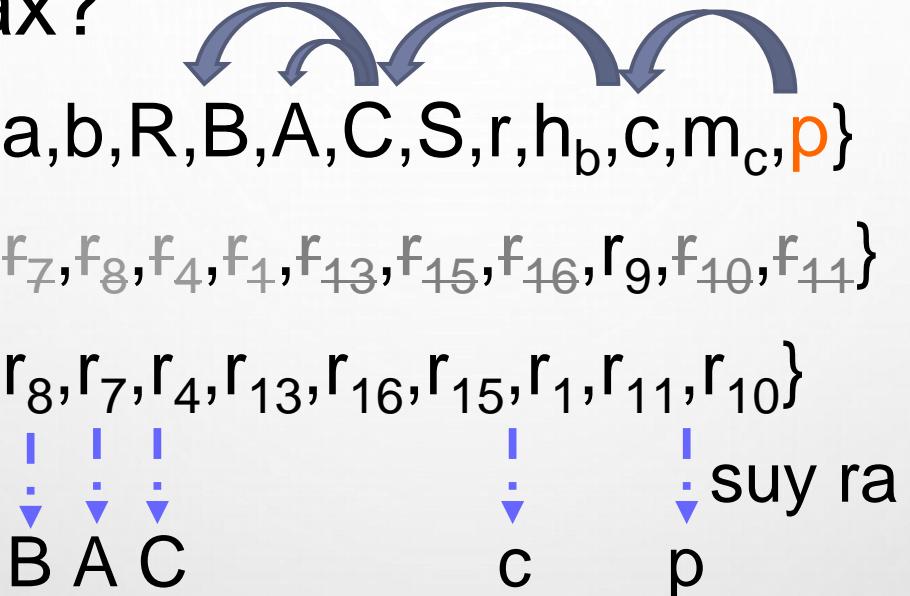
- THOA = {r₇, r₈, r₄, r₁, r₁₃, r₁₅, r₁₆, r₉, r₁₀, r₁₁}

$$- \text{ VET } = \{r_8, r_7, r_4, r_{13}, r_{16}, r_{15}, r_1, r_{11}, r_{10}\}$$



Suy diễn tiến

- Cơ chế max?

- TG = {a,b,R,B,A,C,S,r,h_b,c,m_c,**p**}
- THOA = {f₇,f₈,f₄,f₁,f₁₃,f₁₅,f₁₆,r₉,f₁₀,f₁₁}
- VET = {r₈,r₇,r₄,r₁₃,r₁₆,r₁₅,r₁,r₁₁,r₁₀}


Vết sau khi loại bỏ dư thừa: r₈,r₇,r₄,r₁,r₁₀

Suy diễn tiến

- Cơ chế max?



- TG = {a,b,R,B,A,C,S,r,h_b,c,m_c,**p**}
- THOA = {f₇,f₈,f₄,f₁,f₁₃,f₁₅,f₁₆,r₉,f₁₀,f₁₁}
- VET = {r₈,r₇,r₄,r₁₃,r₁₆,r₁₅,r₁,r₁₁,r₁₀}

- Cơ chế min?

- Cơ chế FIFO?

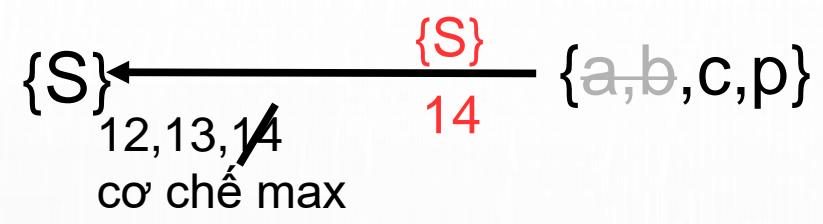
- Cơ chế LIFO?

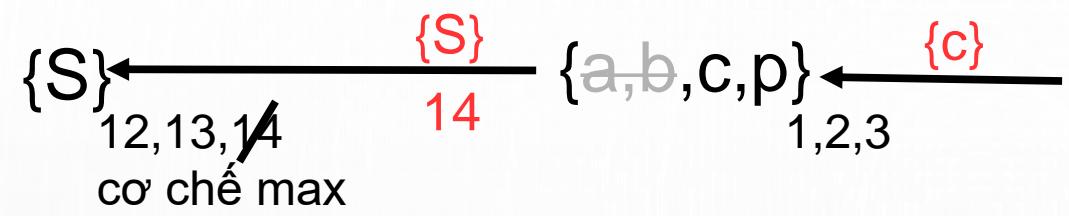
Suy diễn lùi

- Xuất phát từ kết luận đi ngược về giả thiết
- Ví dụ:
 - $GT = \{a, b, R\}$
 - $KL = \{S\}$

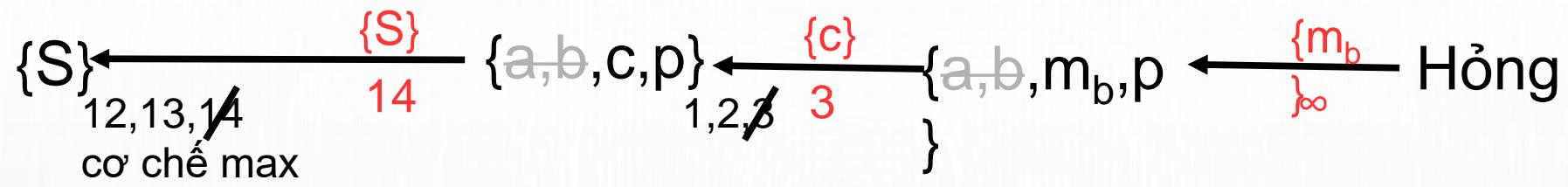
$\{S\}$ ← $\{S\}$
12,13,14

$\{S\}$ ← $\{S\}$
12,13,14
Cơ chế max

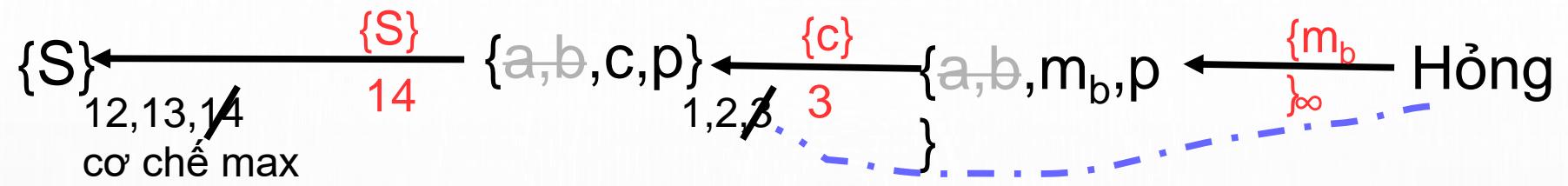


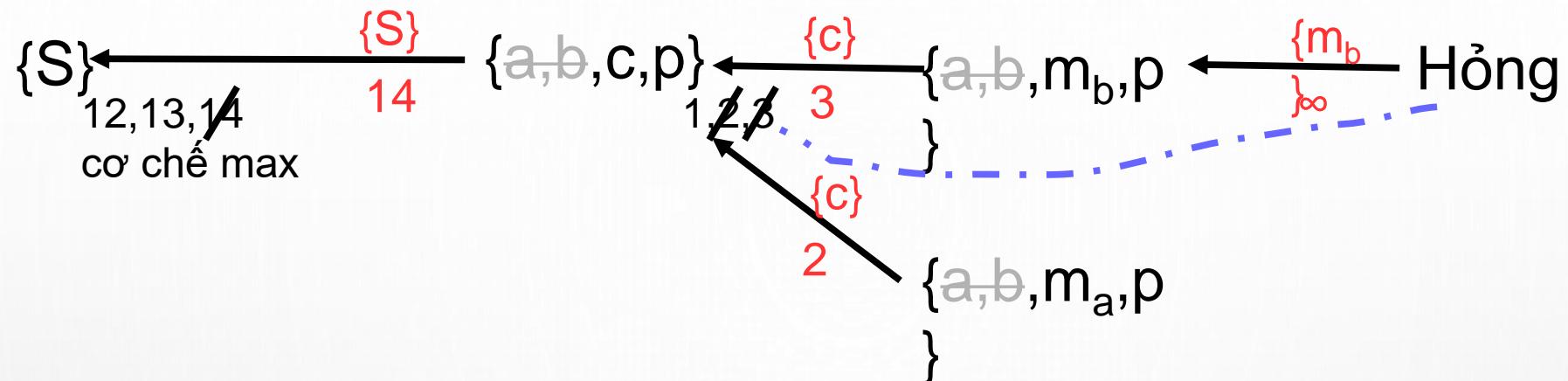


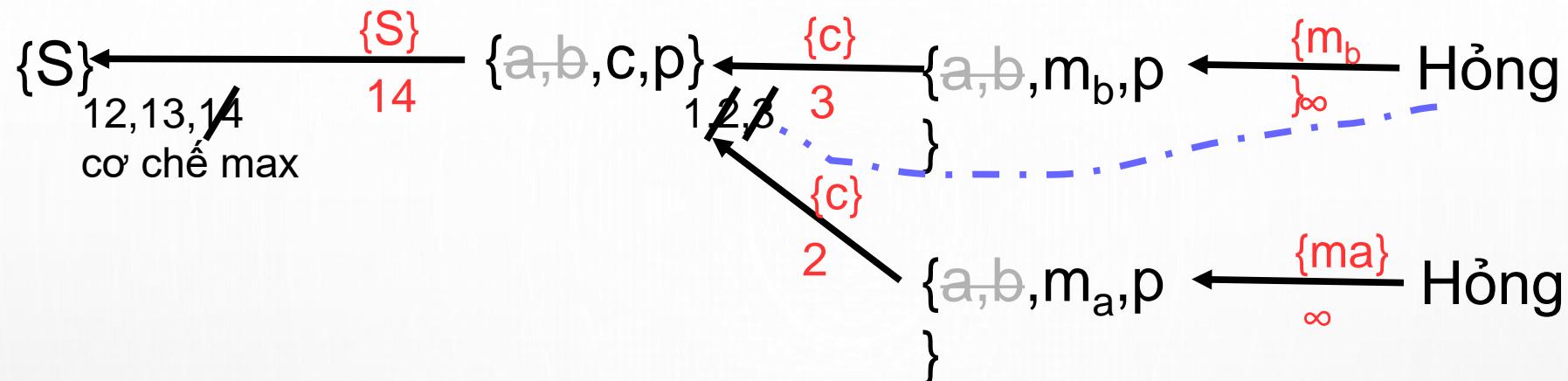


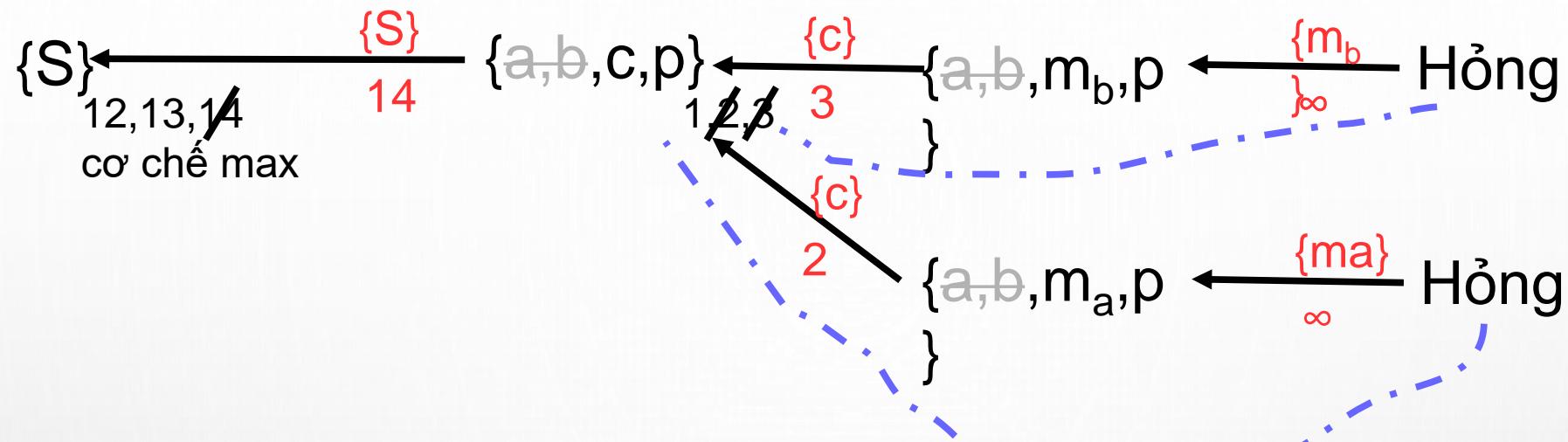


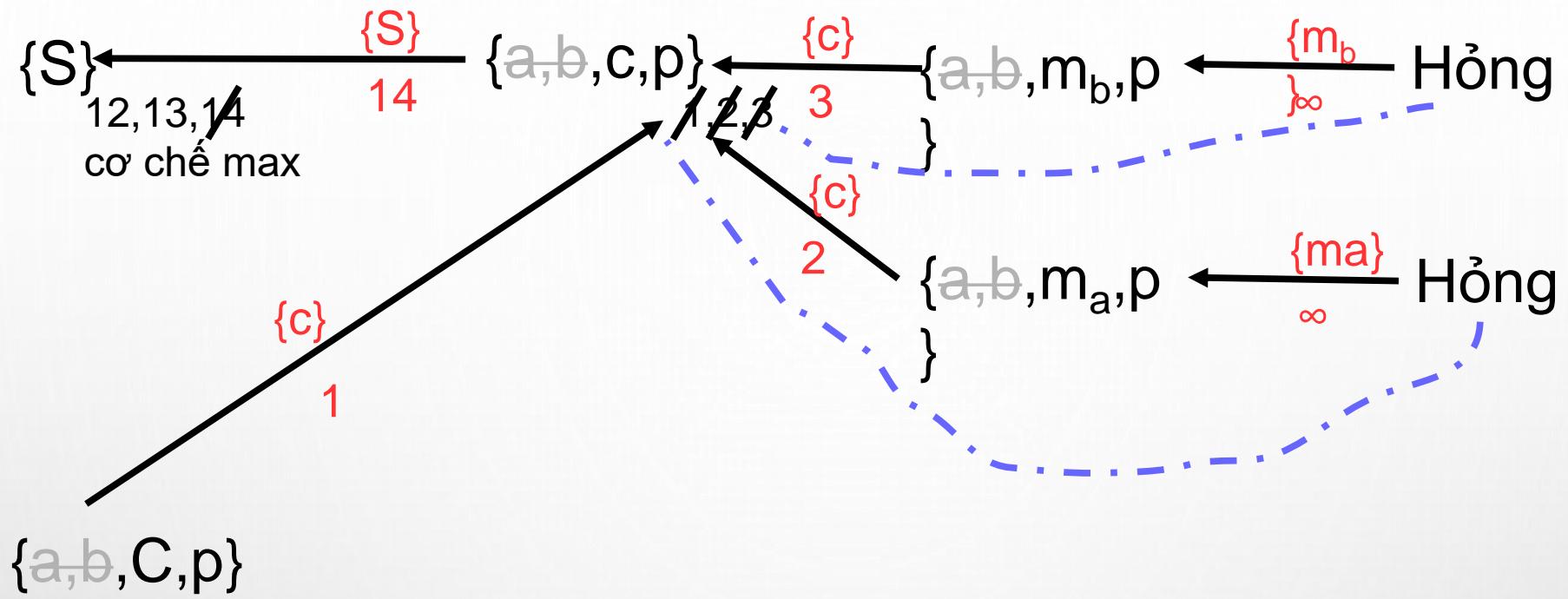
∞ : ký hiệu cho trường
 hợp không có luật nào

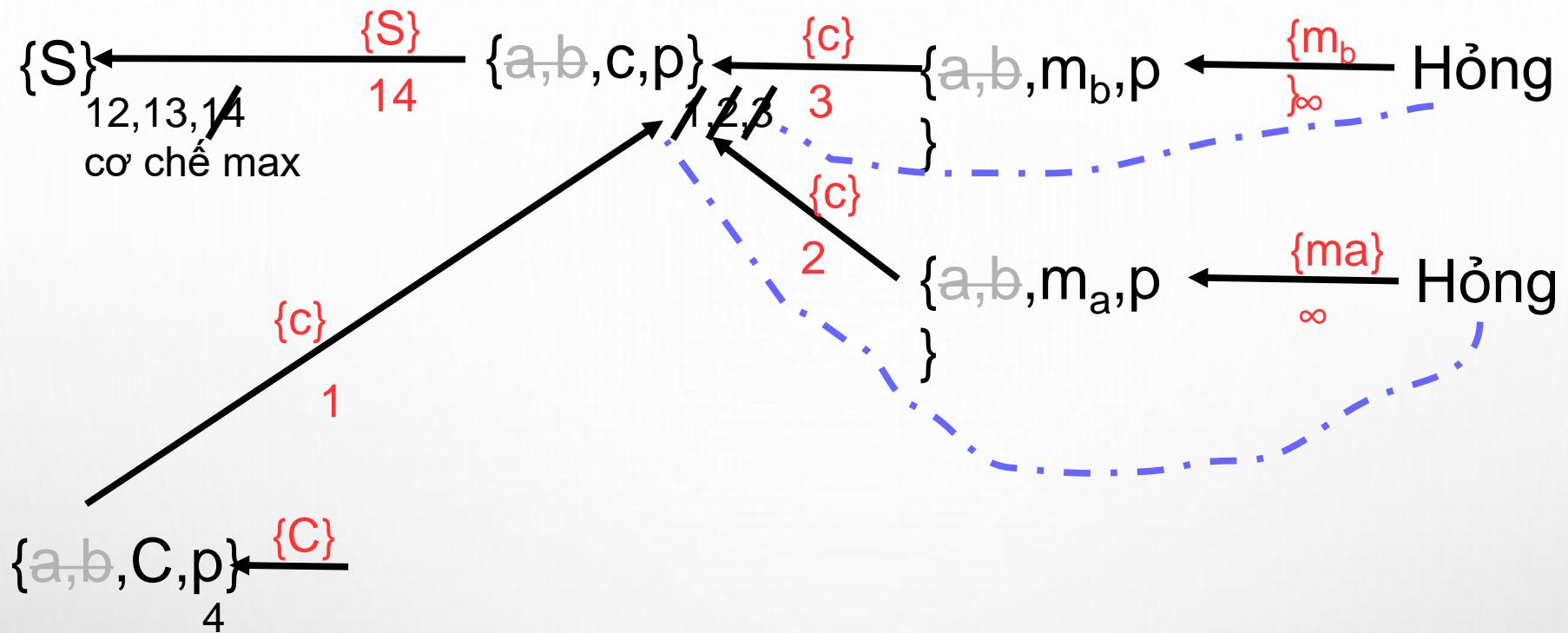


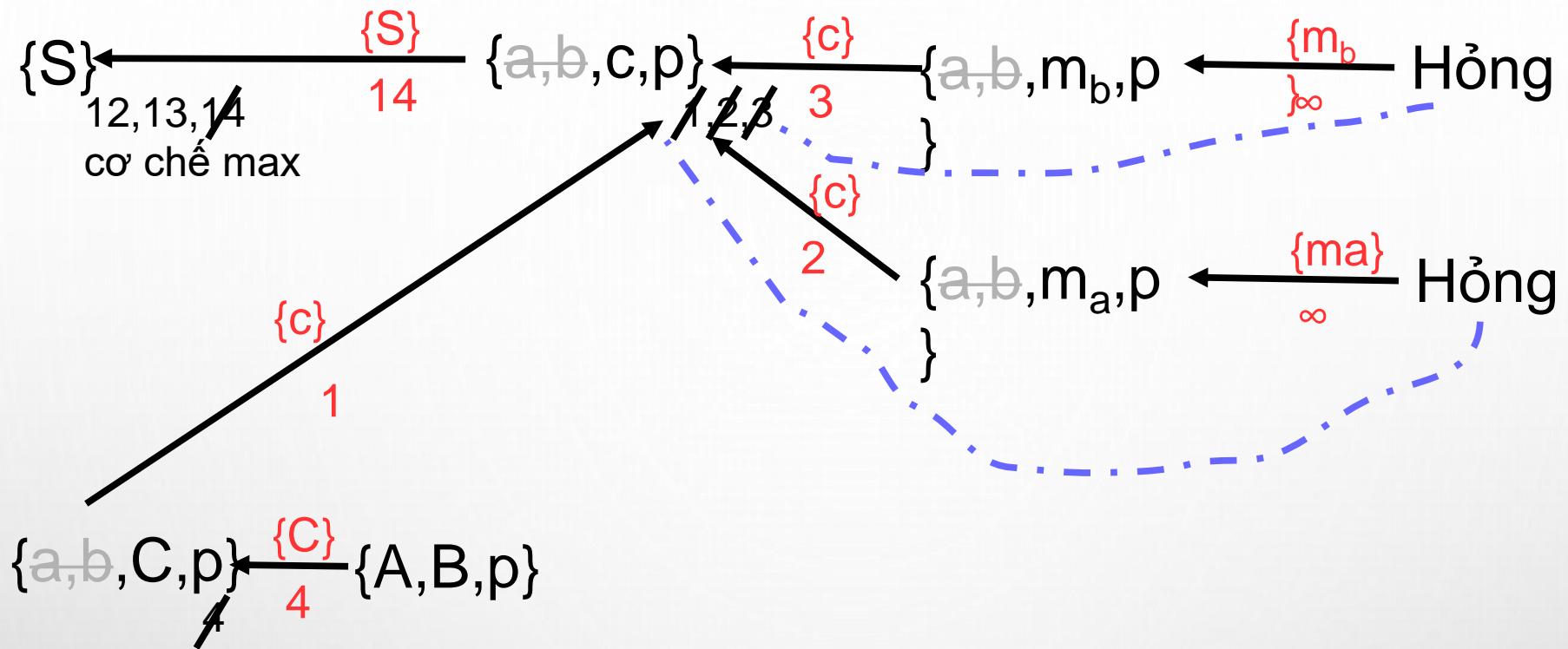


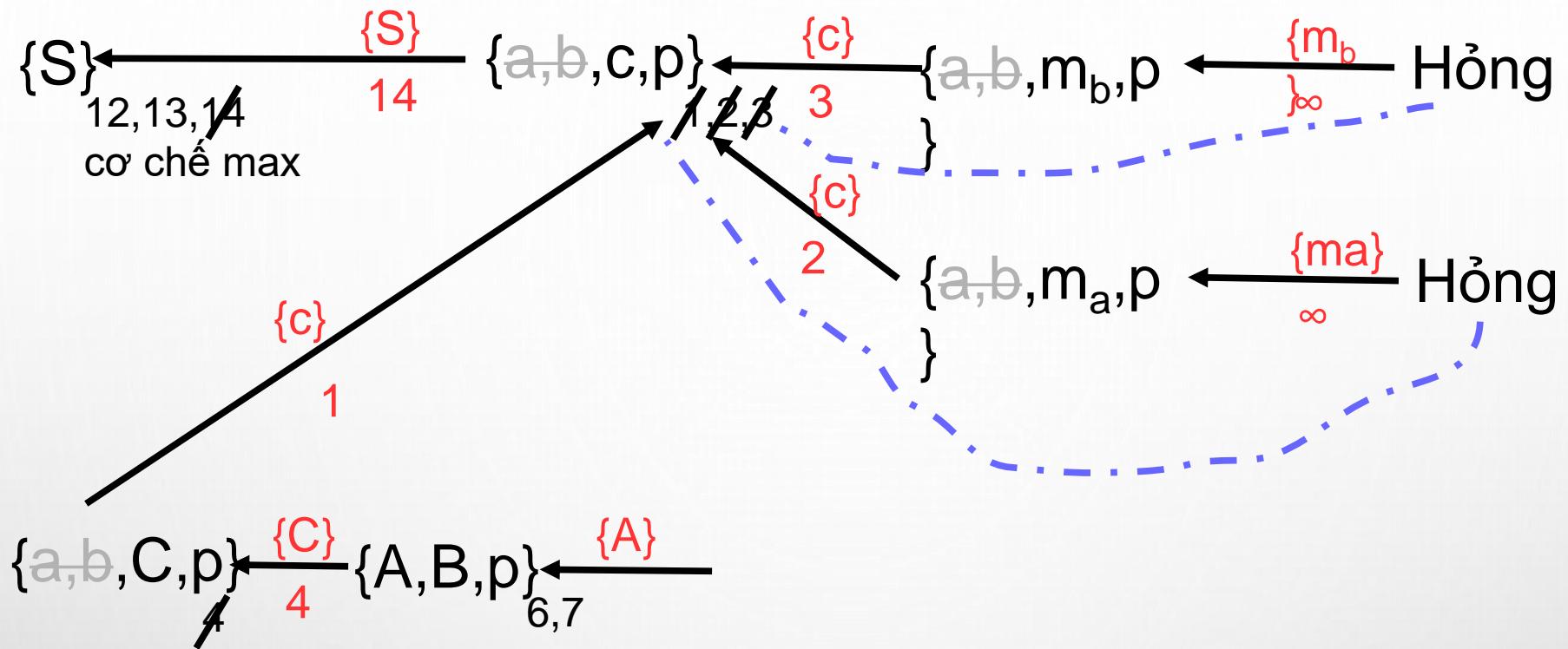


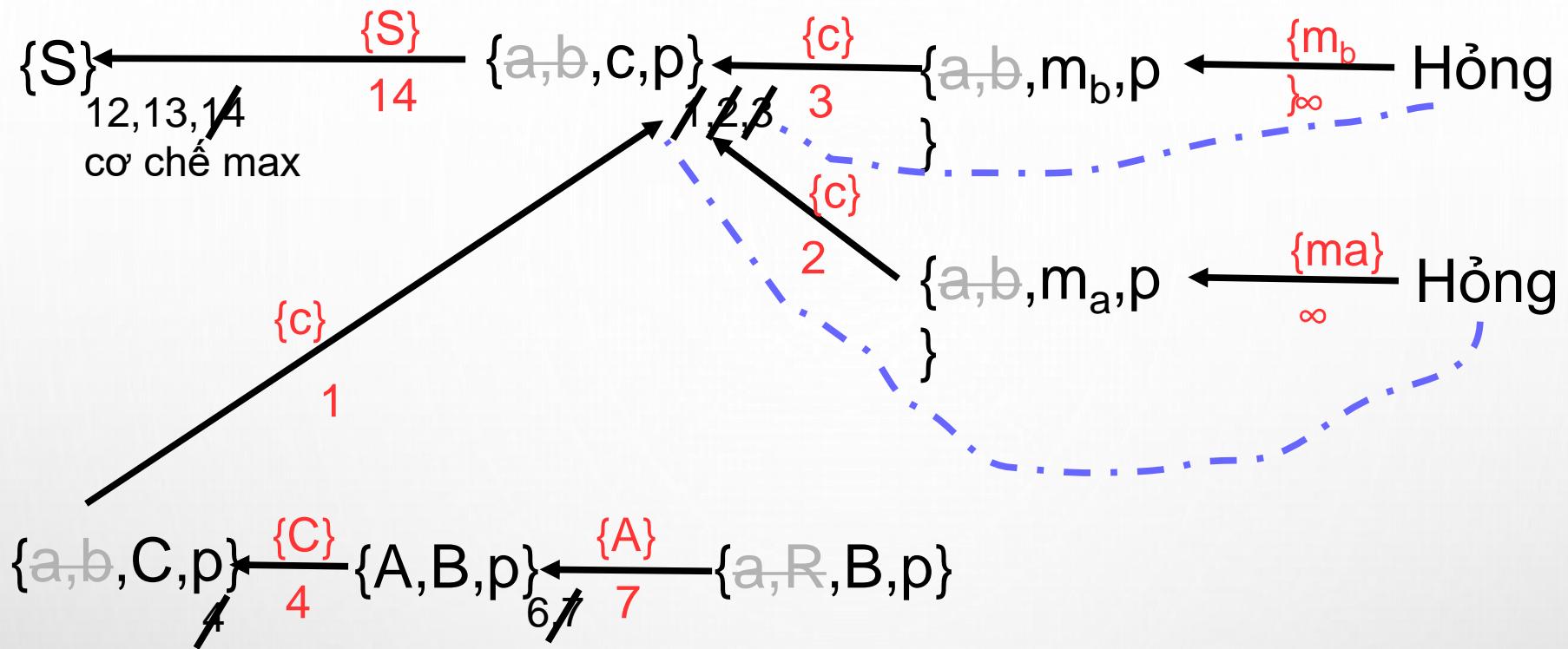


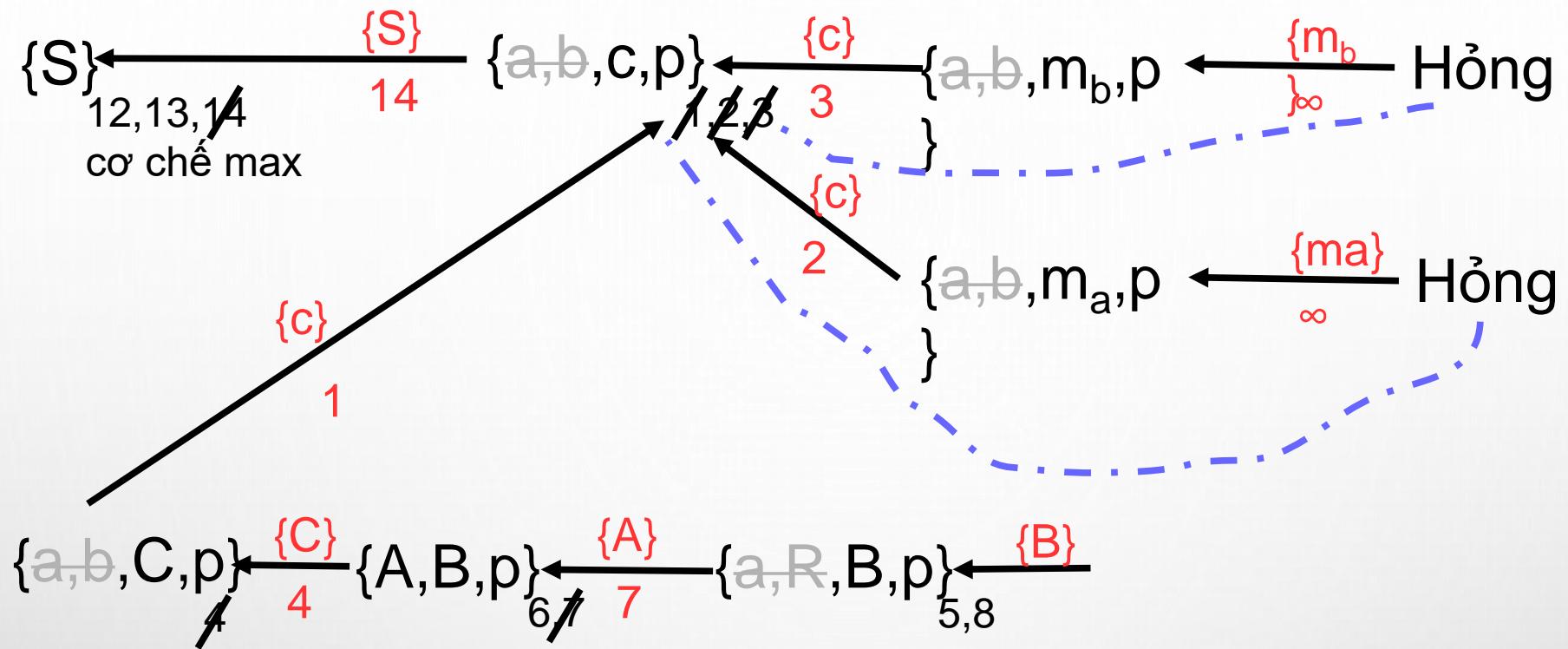


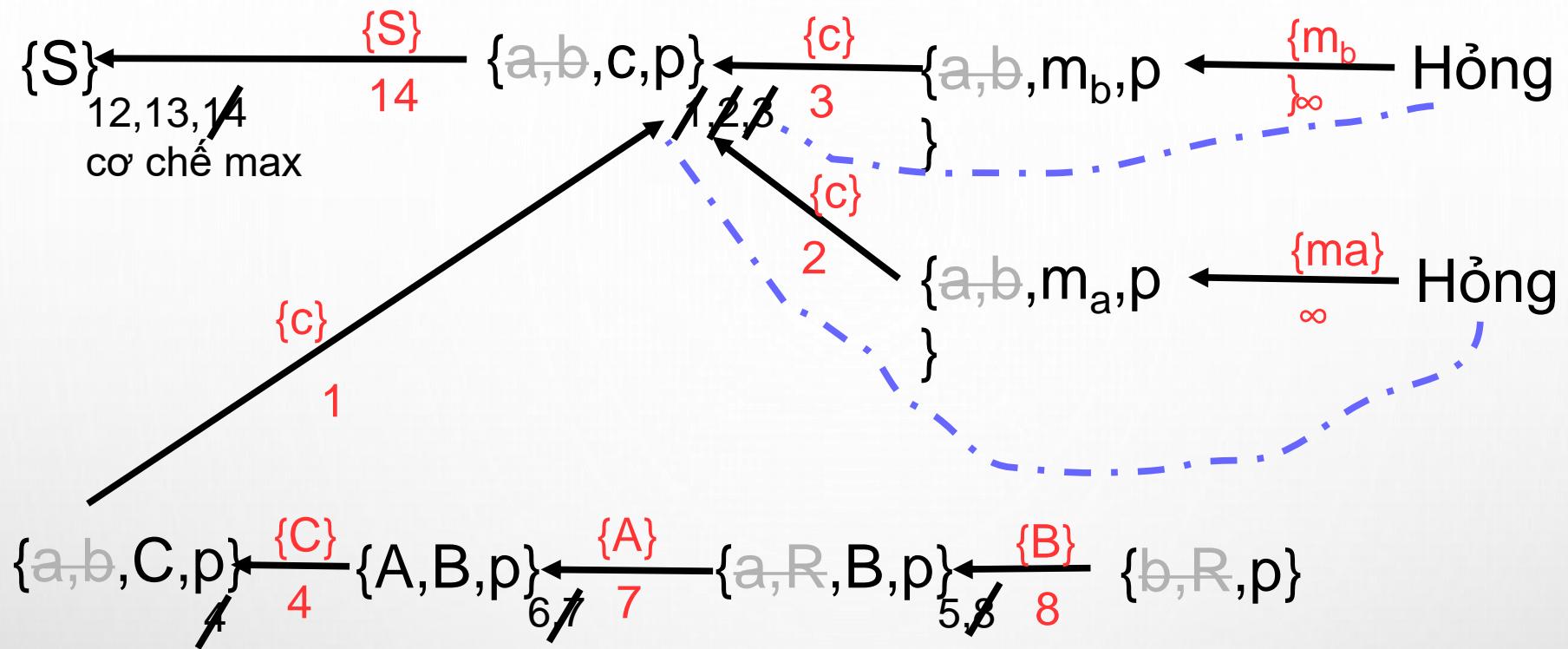


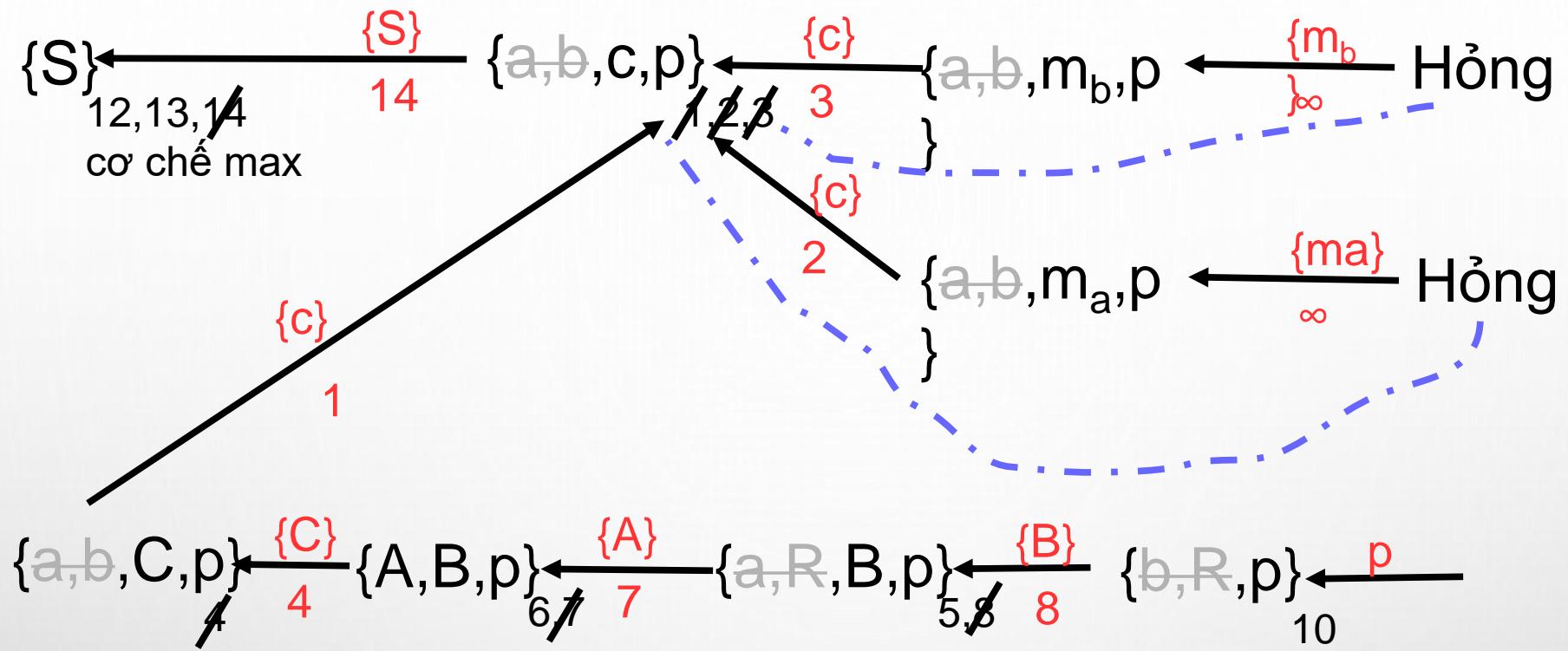


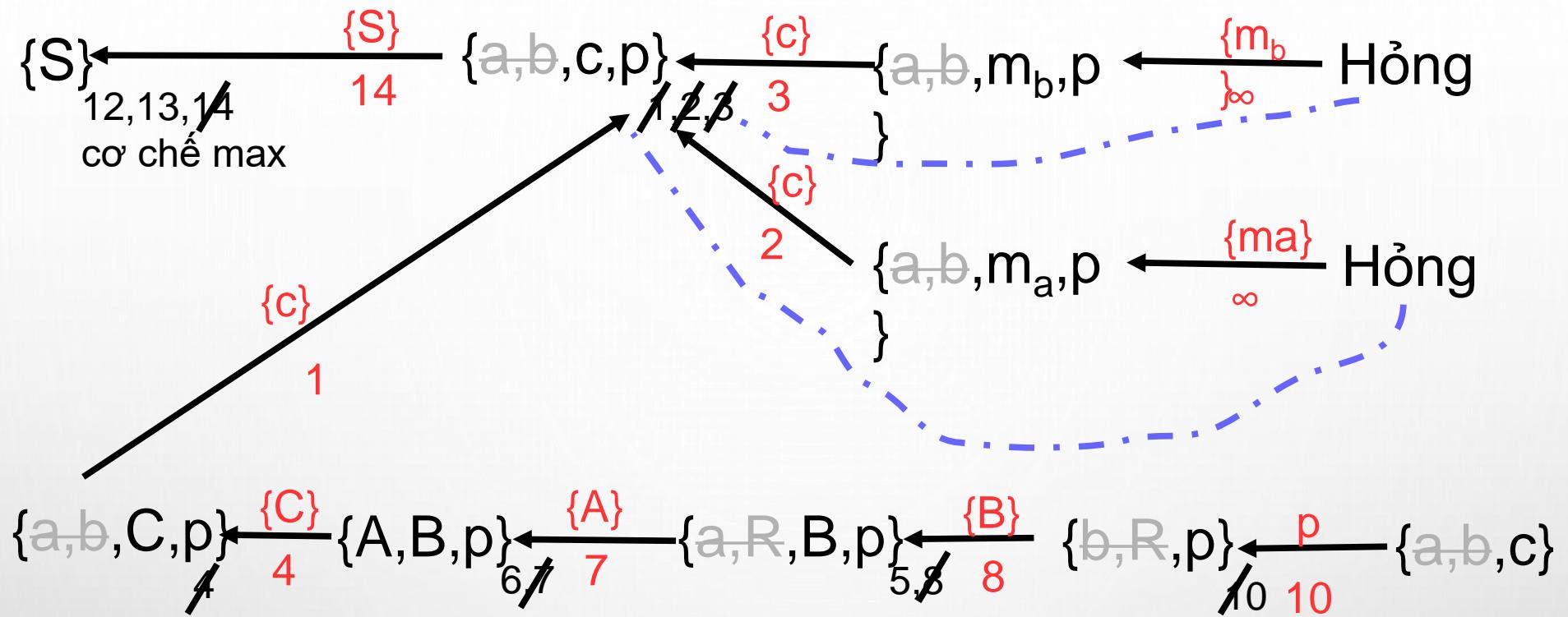


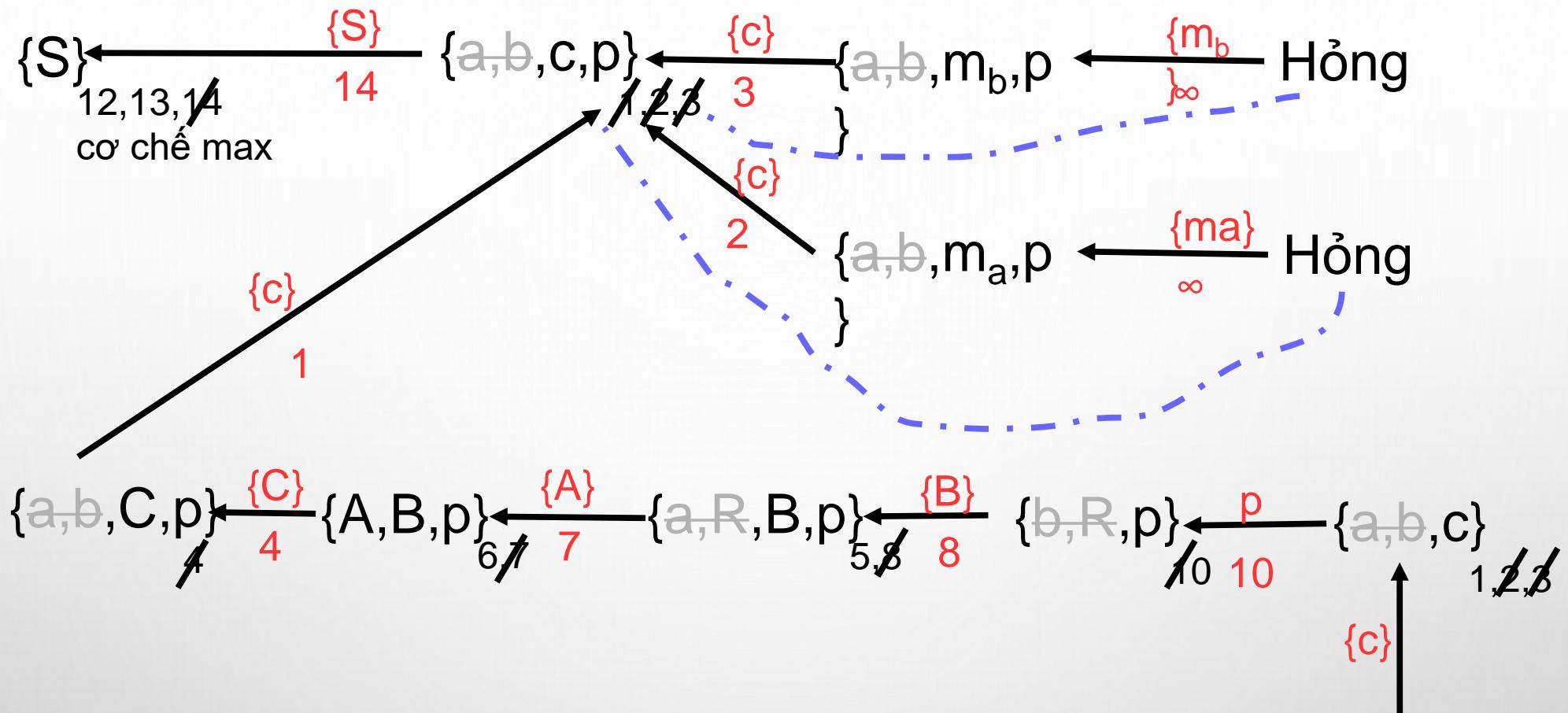


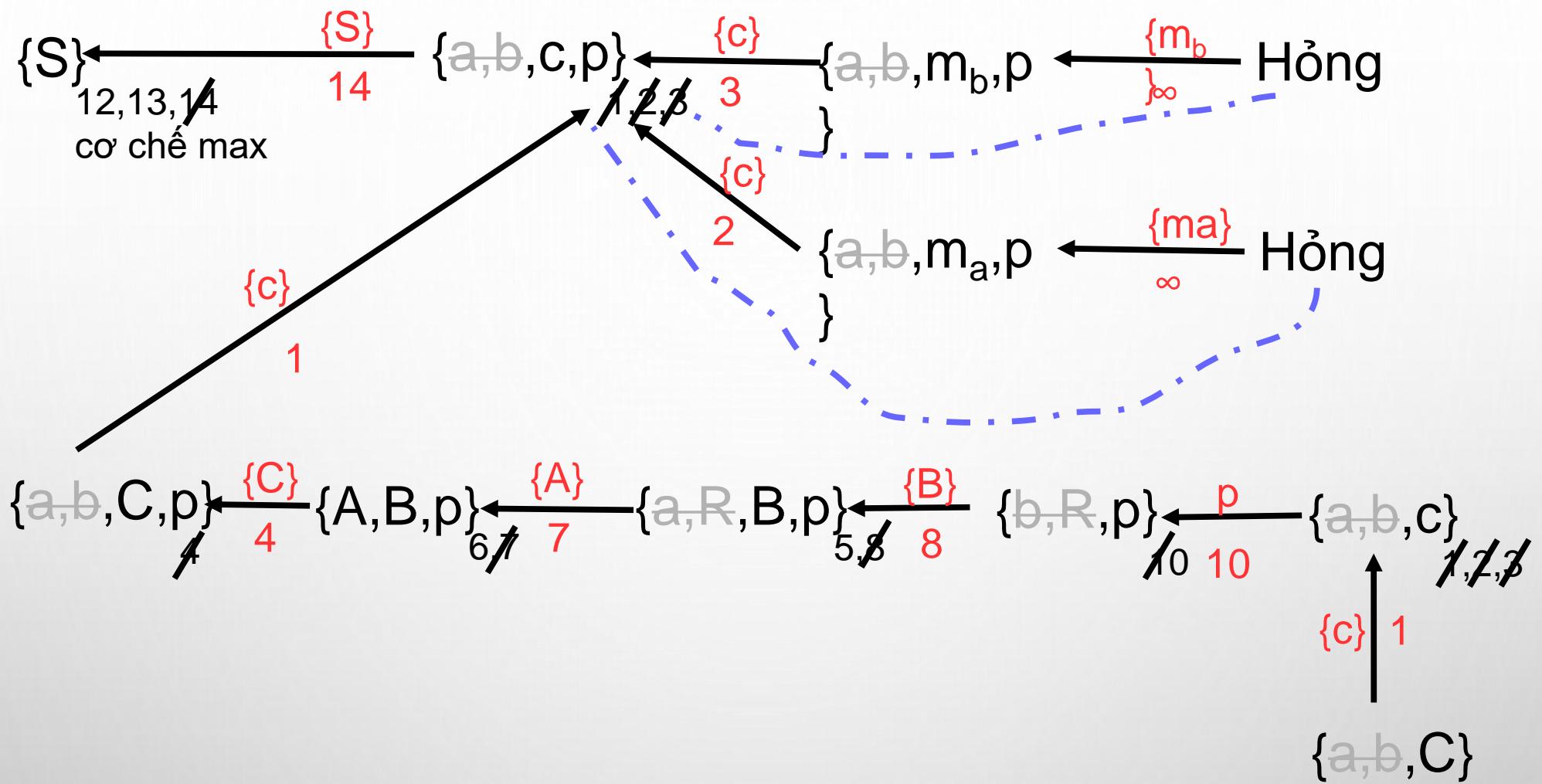


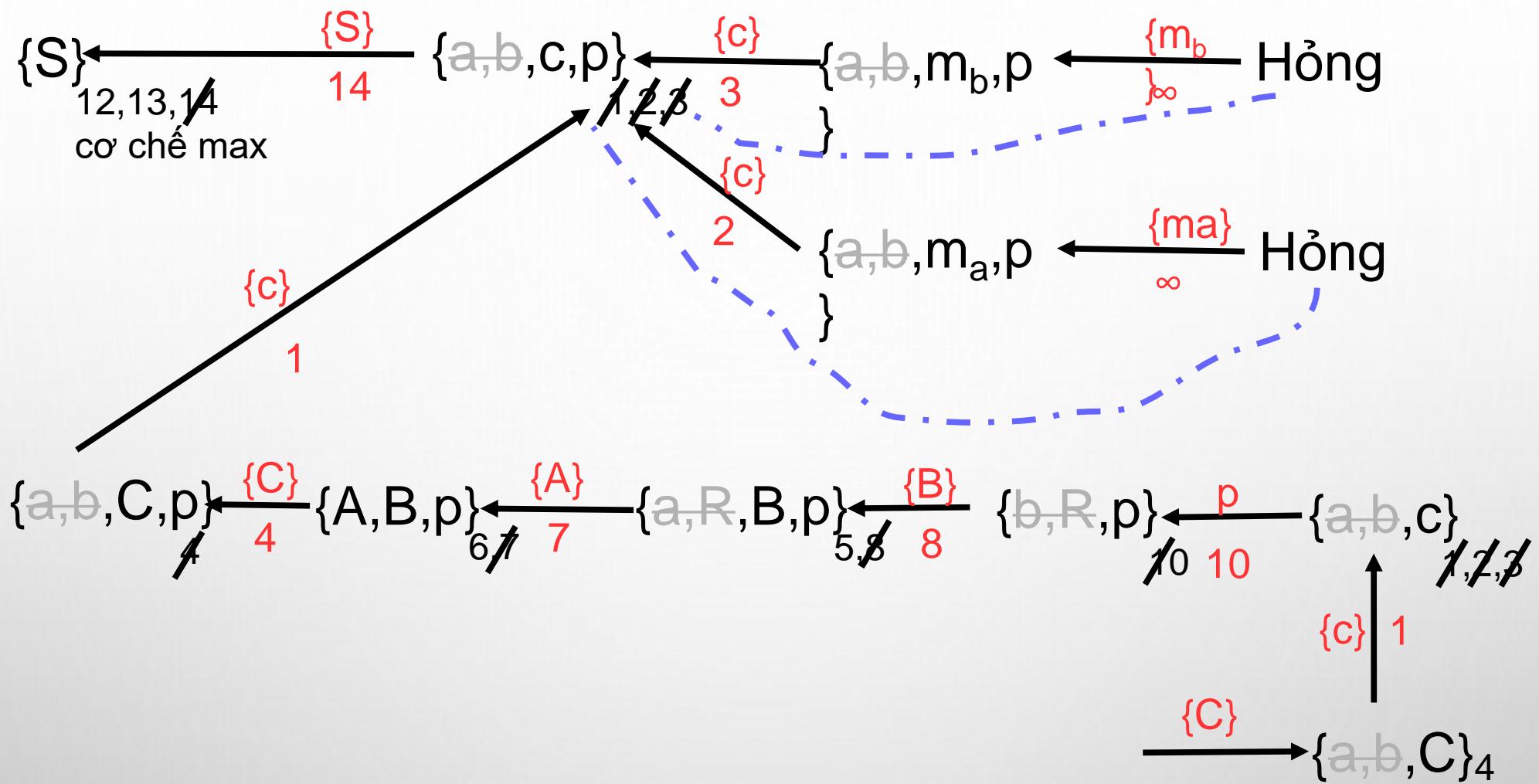


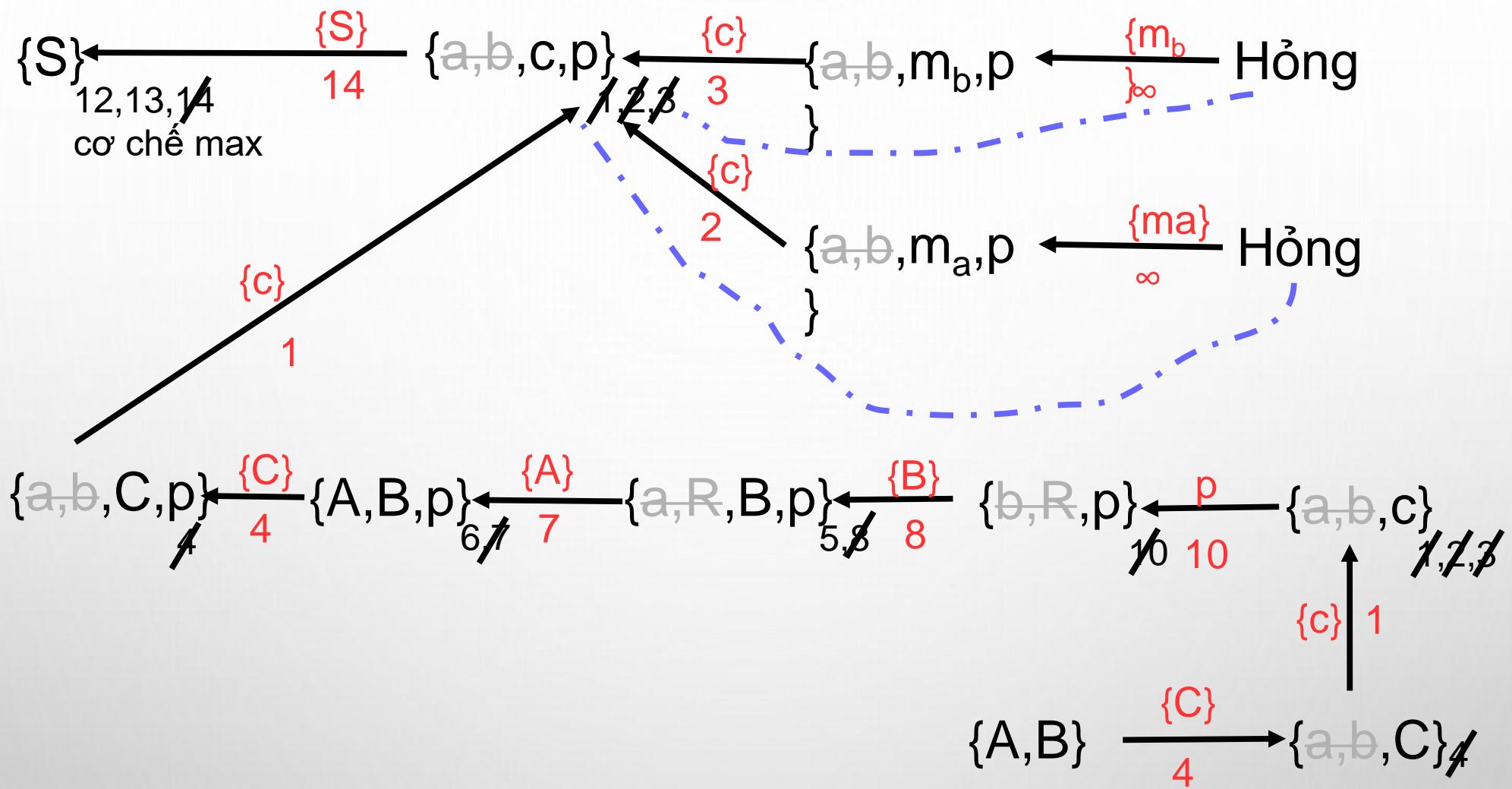


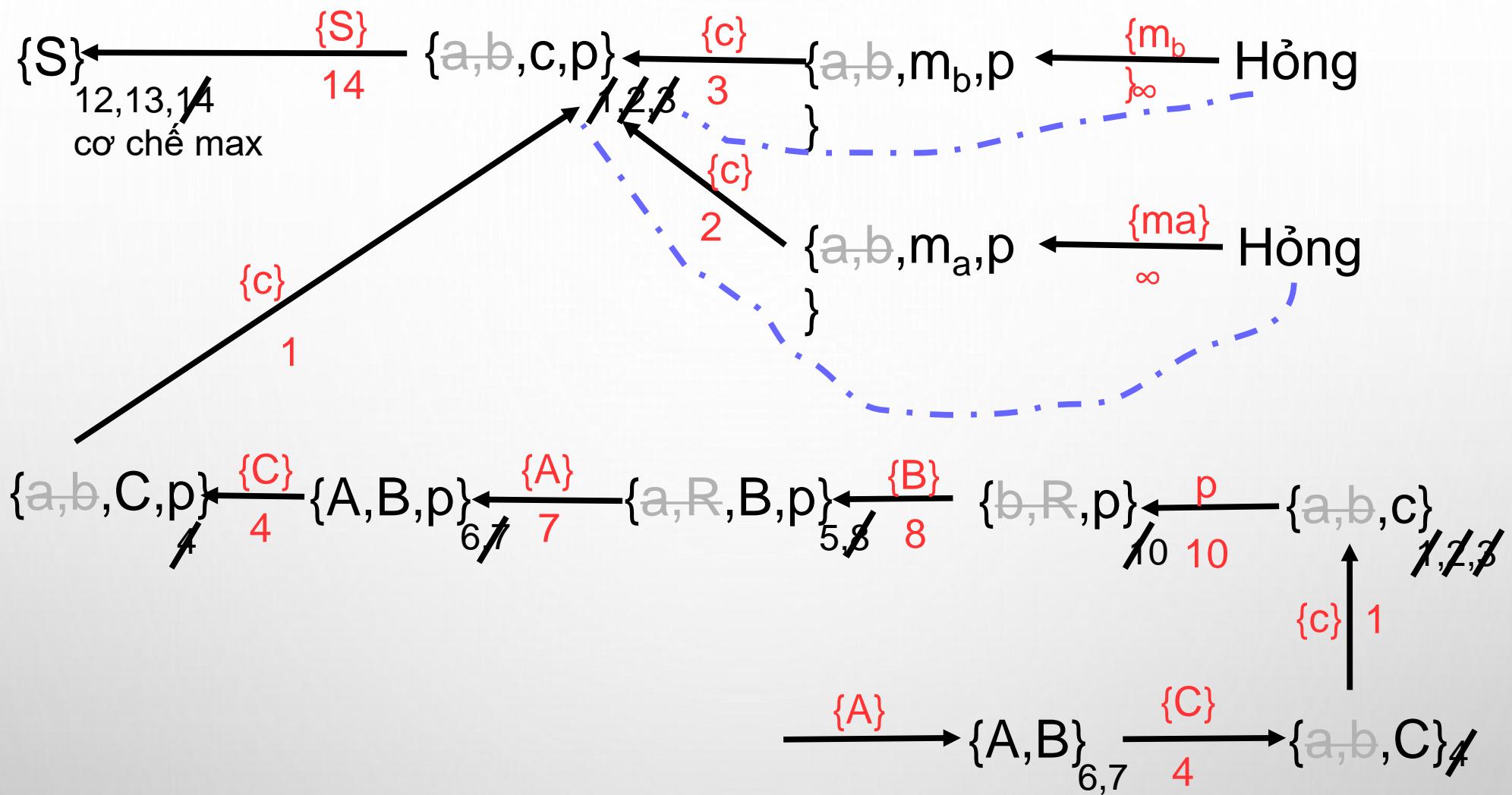


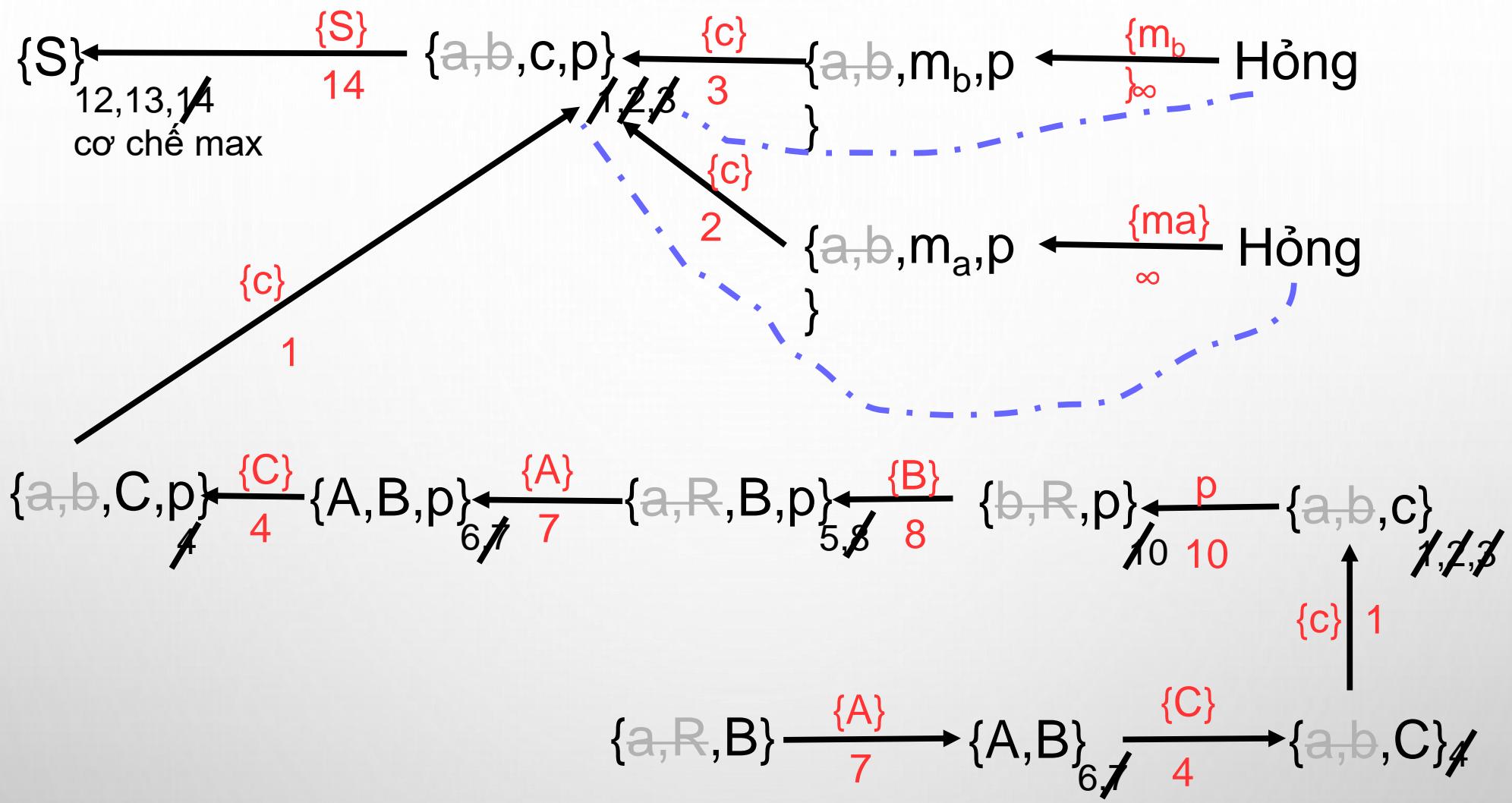


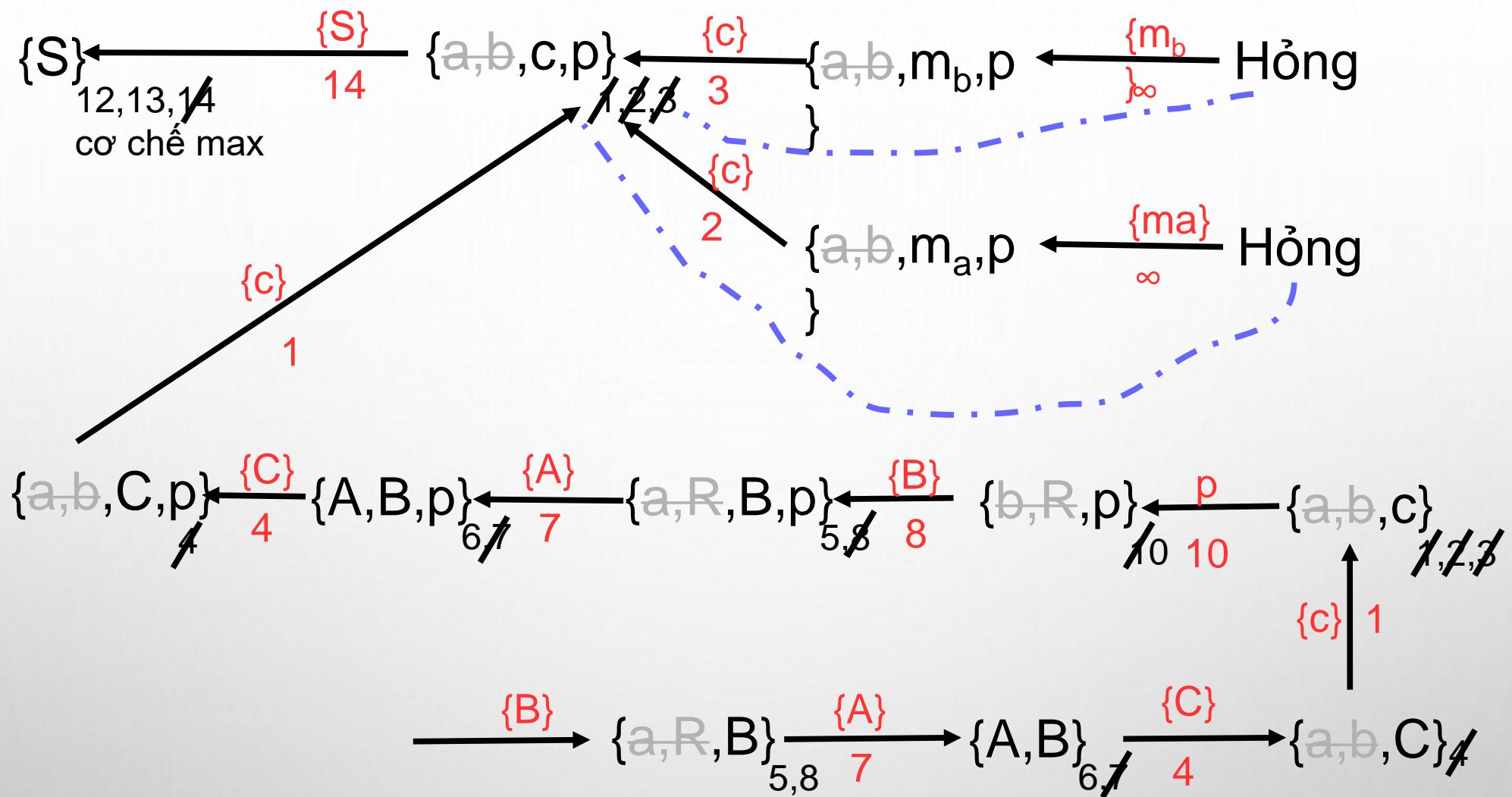


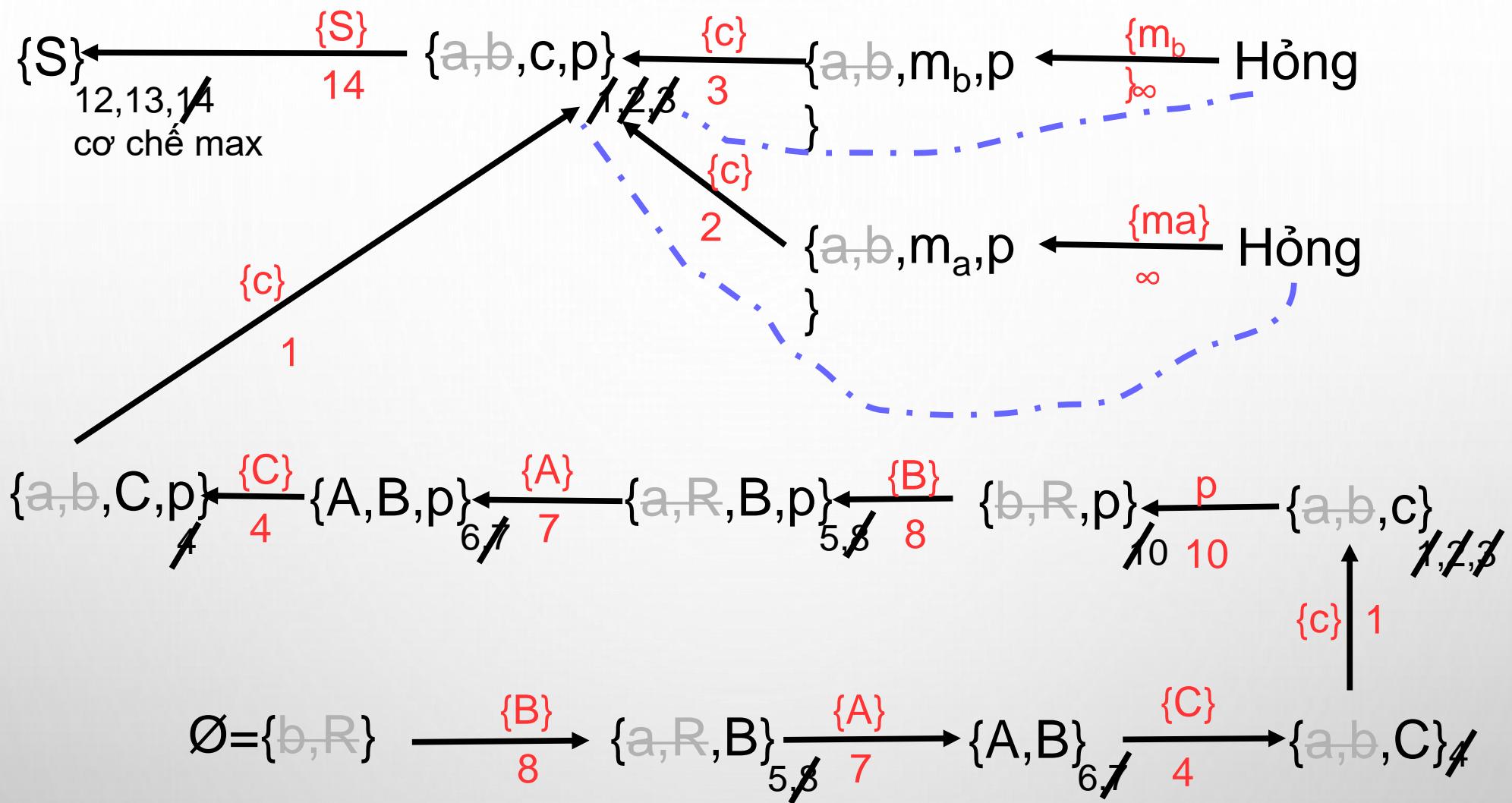


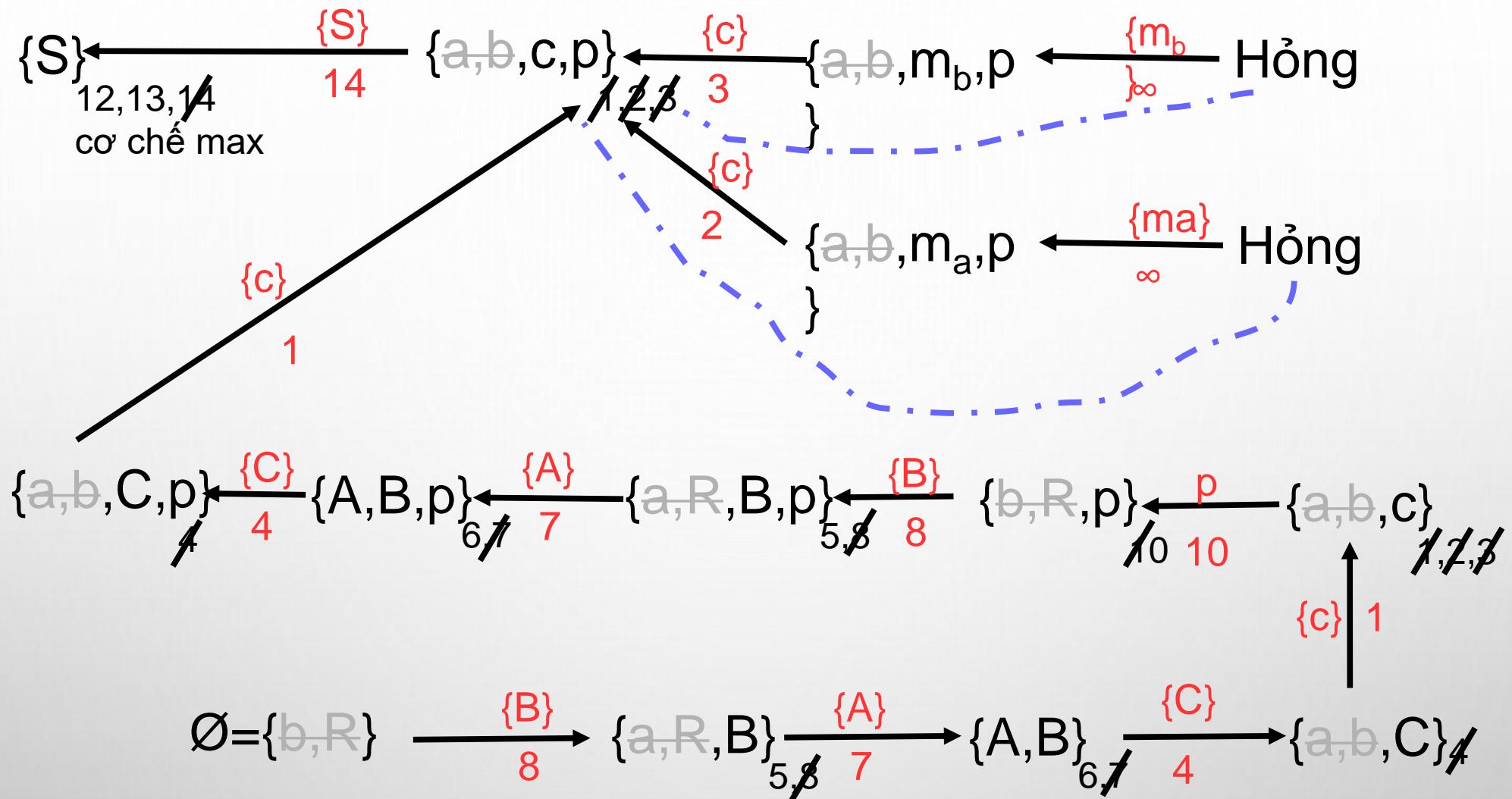










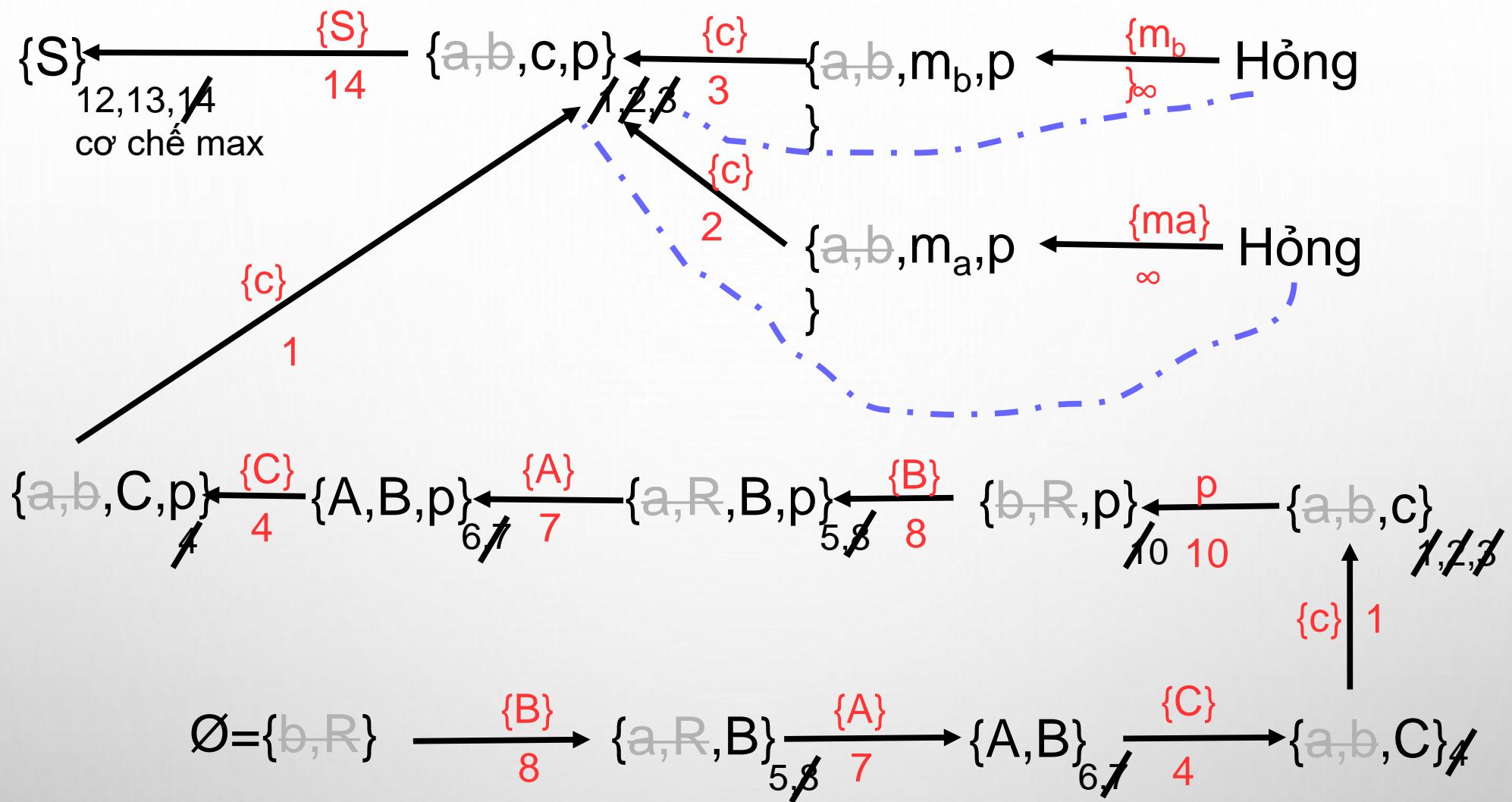


Nhận xét: tìm c đi theo đường vòng

$c \leftarrow X_1 \leftarrow \dots \leftarrow X_n \leftarrow c \leftarrow TG_1 \leftarrow \dots \leftarrow TG_m$

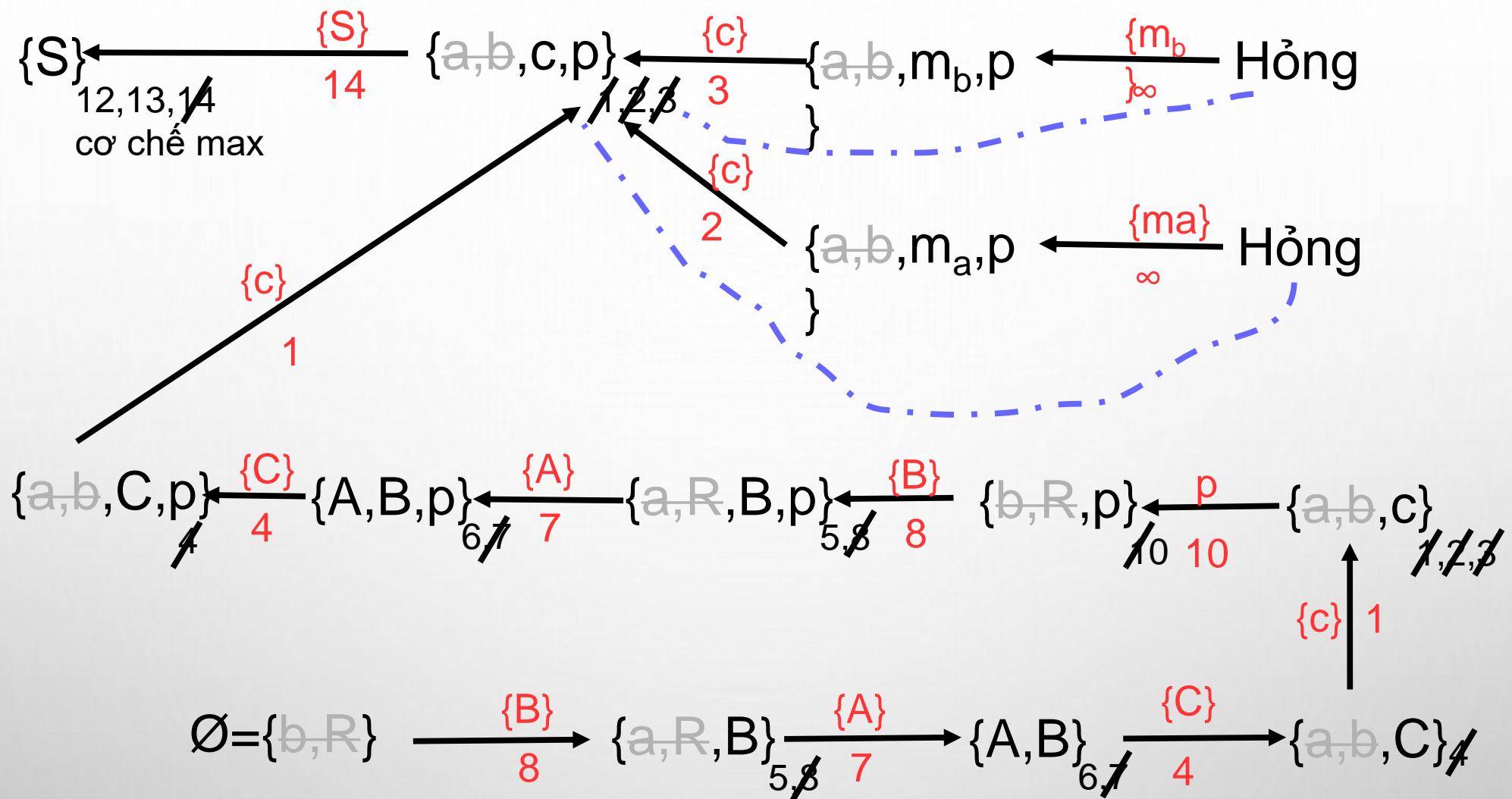
Đến lúc này phải tìm chính nó bởi quan hệ vòng

Loại bỏ dư thừa thành $c \leftarrow TG_1 \leftarrow \dots \leftarrow TG_m$



Vết =

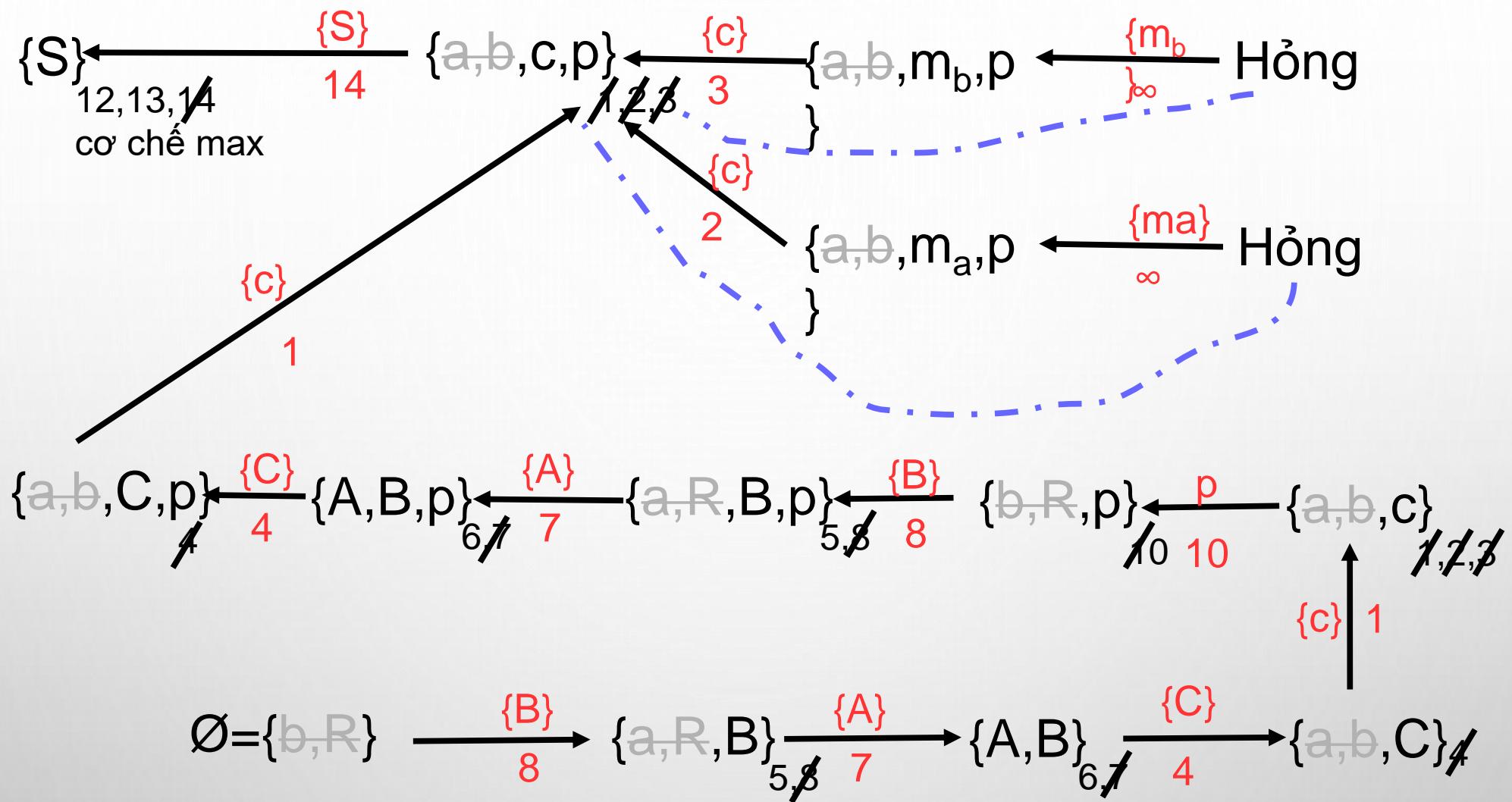
$$\{r_8, r_7, r_4, r_1, r_{10}, r_8, r_7, r_4, r_1, r_{14}\}$$



Vết =

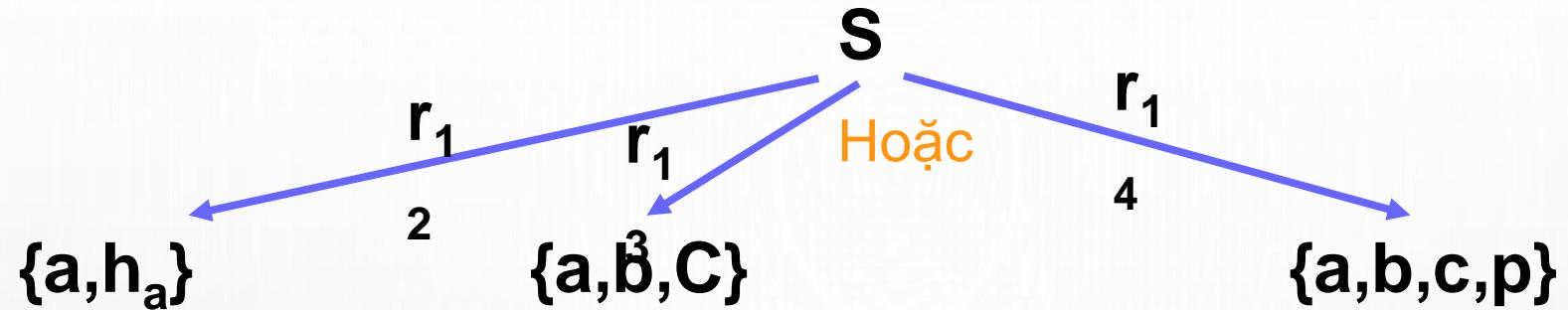
$\{r_8, r_7, r_4, r_1, r_{10}, r_8, r_7, r_4, r_1, r_{14}\}$

B A C c p
S

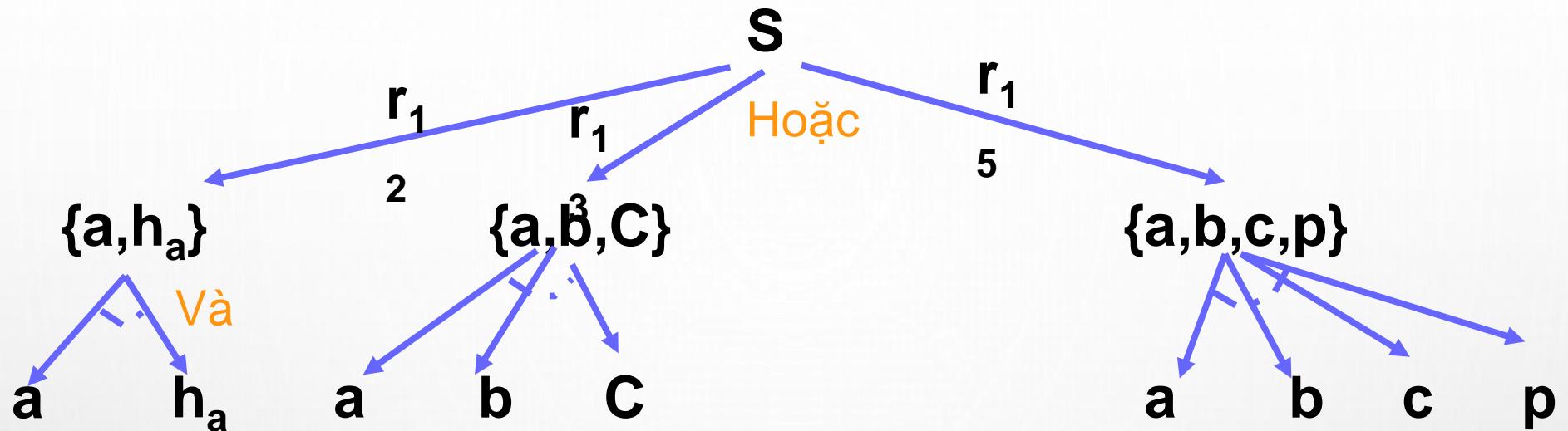


Cơ chế chọn min, LIFO, FIFO?

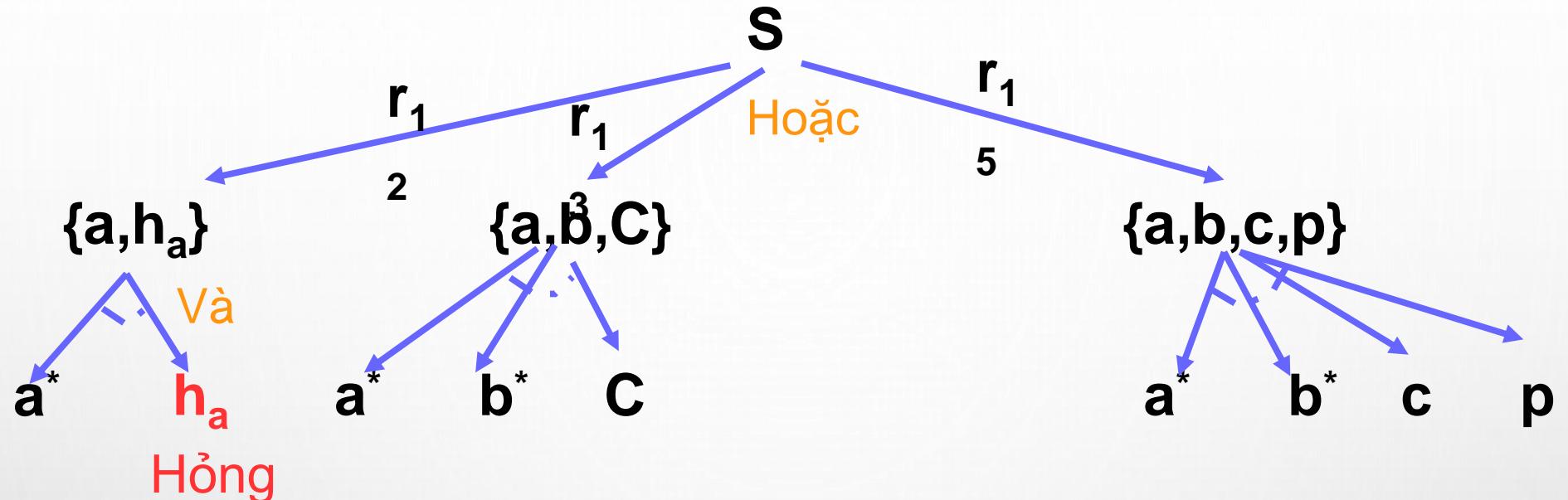
Đồ thị suy diễn lùi Và-Hoặc



Đồ thị suy diễn lùi Và-Hoặc

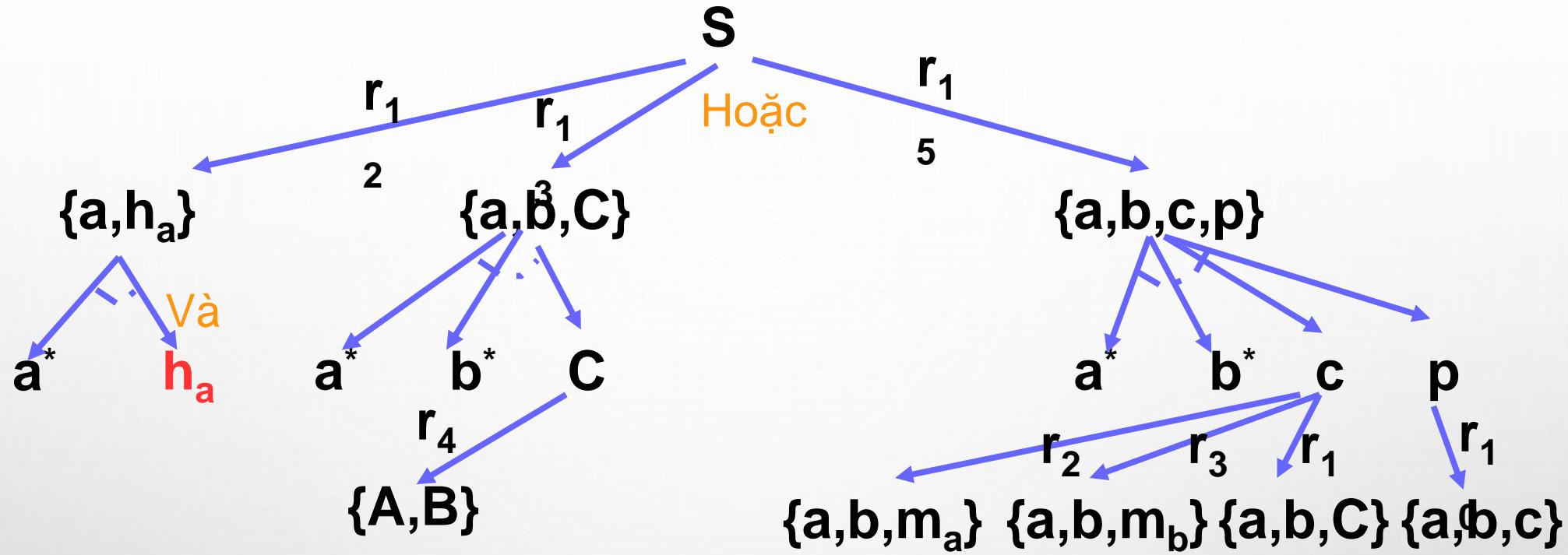


Đồ thị suy diễn lùi Và-Hoặc



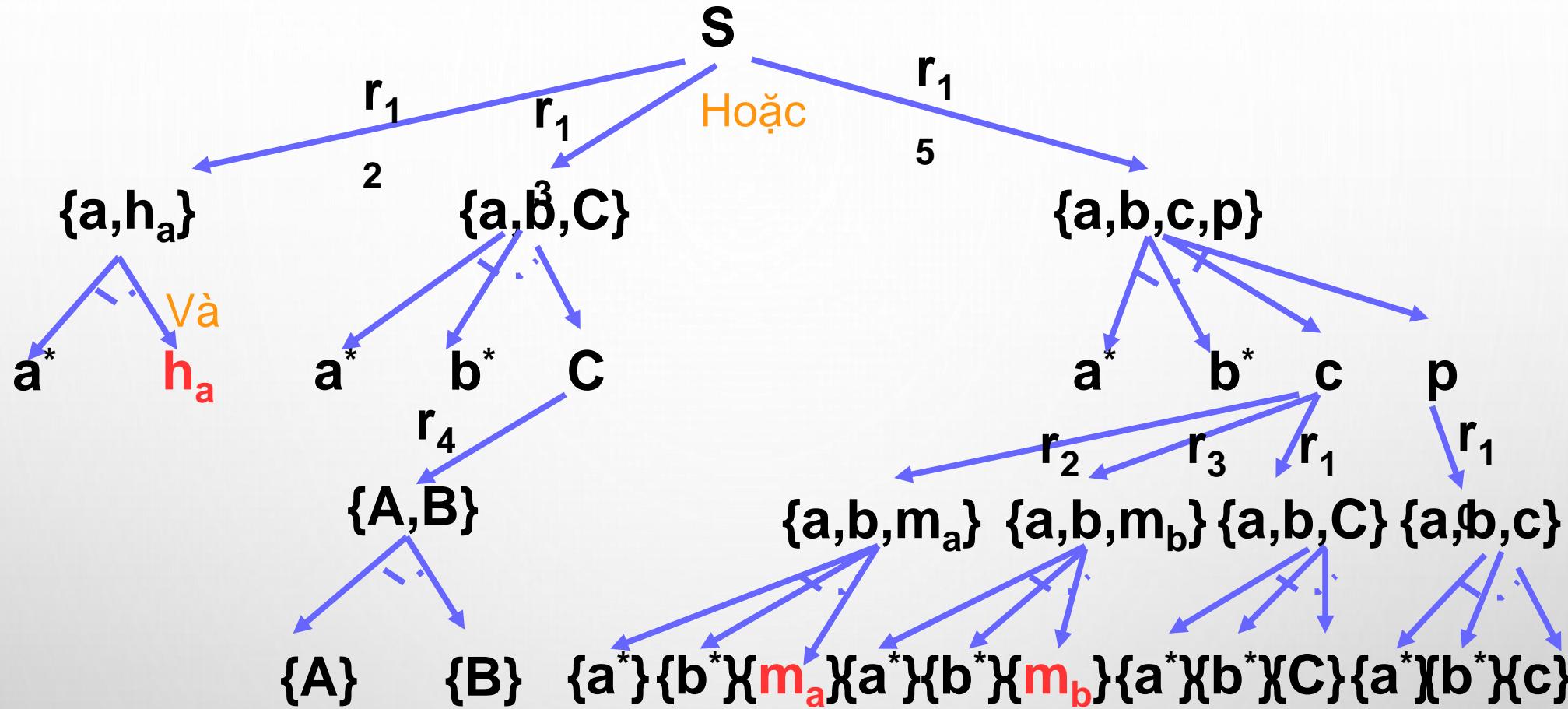
*: Nút lá, là 1 sự kiện cho trong giả thiết
f: Sự kiện không cho trong giả thiết và không nằm ở về phải của các luật

Đồ thị suy diễn lùi Và-Hoặc



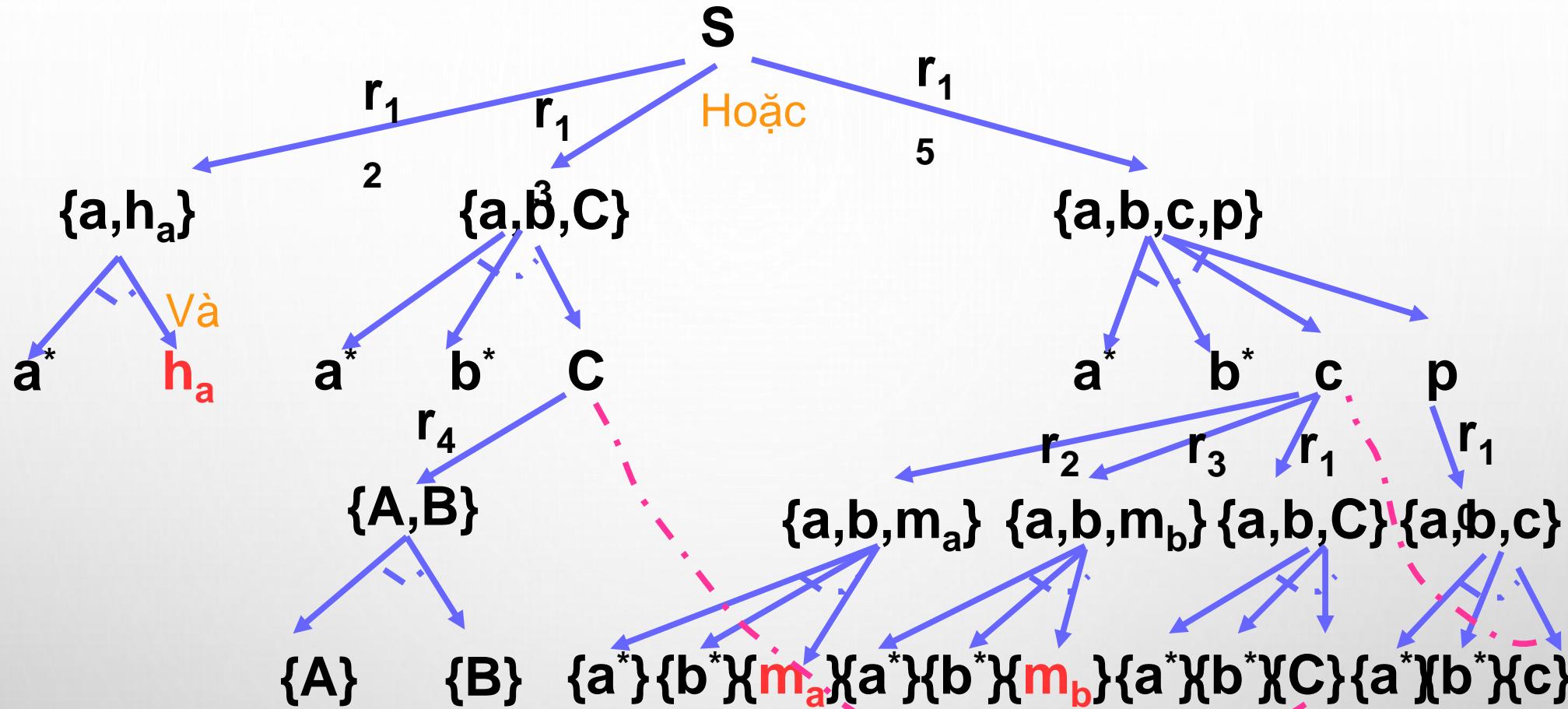
*: Nút lá, là 1 sự kiện cho trong giả thiết
 f: Sự kiện không cho trong giả thiết và không nằm ở về phải của các luật

Đồ thị suy diễn lùi Và-Hoặc



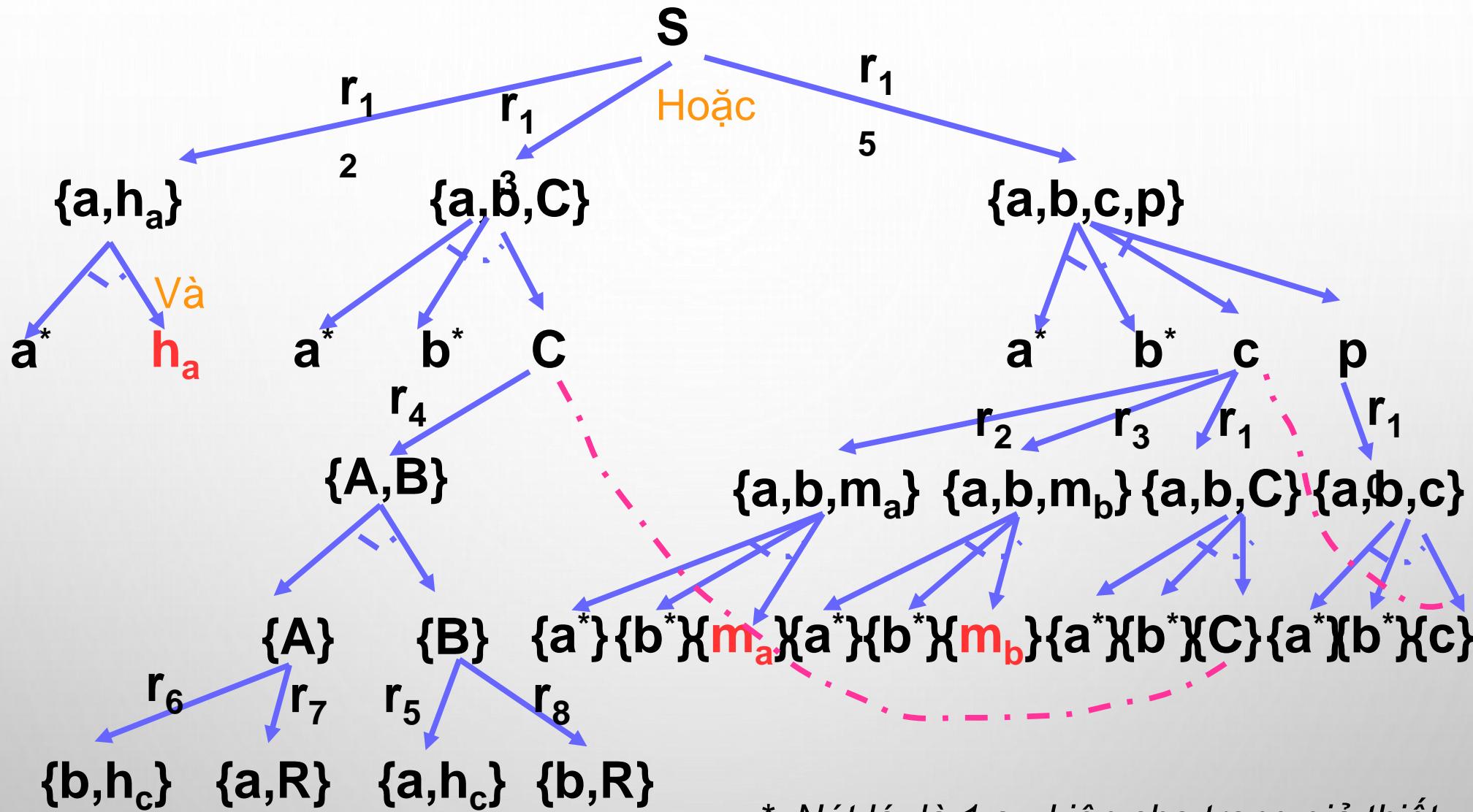
*: Nút lá, là 1 sự kiện cho trong giả thiết
f: Sự kiện không cho trong giả thiết và không nằm ở về phải của các luật

Đồ thị suy diễn lùi Và-Hoặc



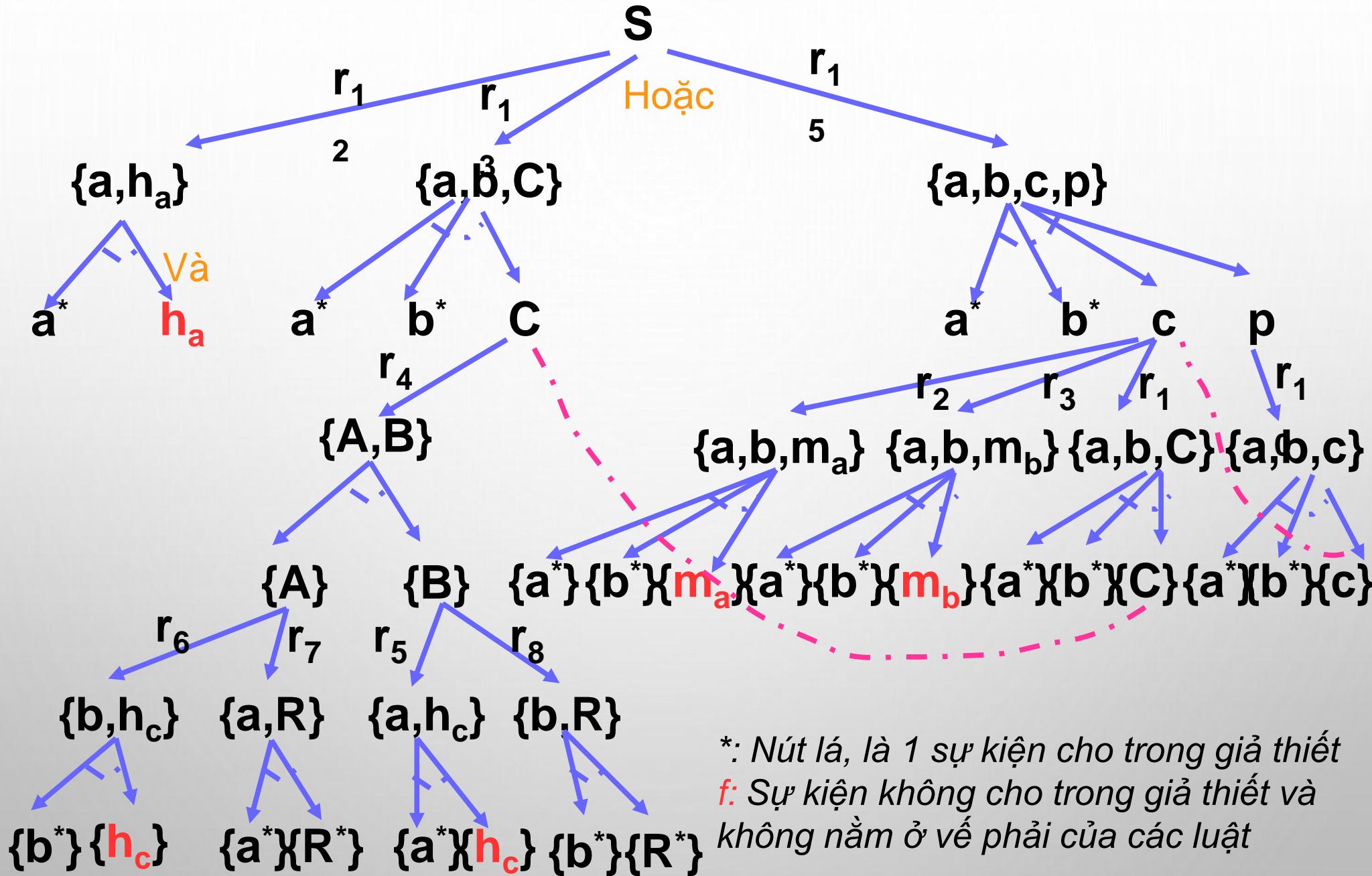
$*$: Nút lá, là 1 sự kiện cho trong giả thiết
 f : Sự kiện không cho trong giả thiết và không nằm ở về phải của các luật

Đồ thị suy diễn lùi Và-Hoặc



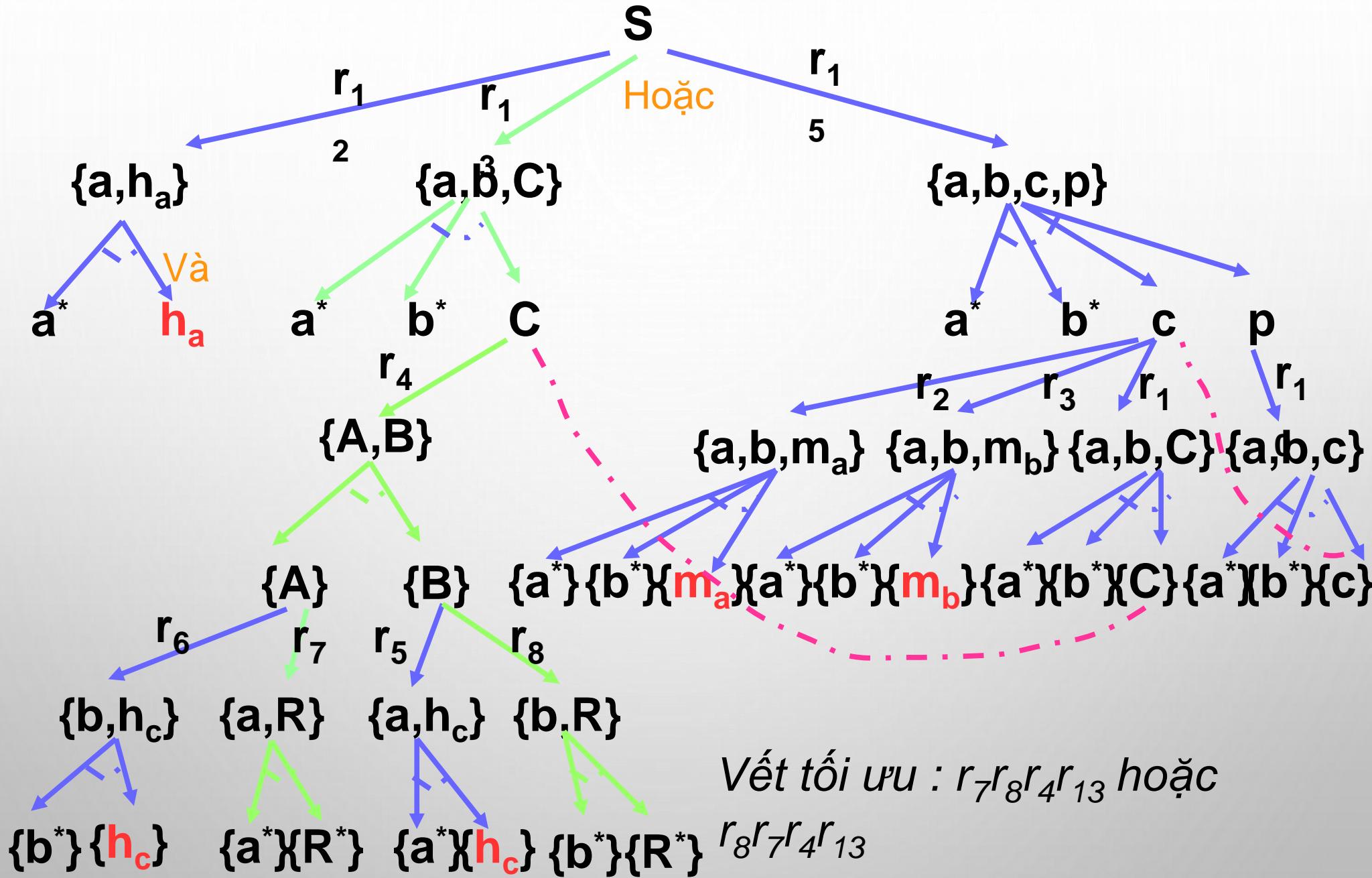
*: Nút lá, là 1 sự kiện cho trong giả thiết
f: Sự kiện không cho trong giả thiết và không nằm ở về phải của các luật

Đồ thị suy diễn lùi Và-Hoặc



*: Nút lá, là 1 sự kiện cho trong giả thiết
f: Sự kiện không cho trong giả thiết và không nằm ở về phải của các luật

Đồ thị suy diễn lùi Và-Hoặc



Suy diễn lùi

- Suy diễn lùi = thực hiện tìm kiếm sâu trên đồ thị suy diễn lùi VÀ-HOẶC
 - OPEN (tập mở): tập các sự kiện (mệnh đề) cần phải suy diễn
 - CLOSE (tập đóng): vết suy diễn
- Mô phỏng thuật toán suy diễn lùi trên đồ thị suy diễn lùi VÀ-HOẶC
 - Giải thích các đối tượng trong thuật toán:

. Thuật toán suy diễn lùi trên đồ thị suy diễn lùi VÀ-HOẶC

. Thuật toán suy diễn lùi trên đồ thị suy diễn lùi VÀ-HOẶC

• Thuật toán suy diễn lùi trên đồ thị suy diễn lùi VÀ-HOẶC

Sự kiện	luật	Đích	Vết	sự kiện hỏng	luật hỏng	luật thay thế	Quay lui
	S		\emptyset				False
S	r_{12}	h_a	(S, r_{12})				False
h_a	∞	\emptyset	\emptyset		S	r_{12}	r_{13}
	C		(S, r_{13})				True
C	r_4	A,B	$(S, r_{13}), (C, r_4)$				False
A	r_6	h_c, B	$(S, r_{13}), (C, r_4), (A, r_6)$				False
h_c	∞	B					

• Thuật toán suy diễn lùi trên đồ thị suy diễn lùi VÀ-HOẶC

Sự kiện	luật	Đích	Vết	sự kiện hỏng	luật hỏng	luật thay thế	Quay lui
	S		\emptyset				False
S	r_{12}	h_a	(S, r_{12})				False
h_a	∞	\emptyset	\emptyset	S	r_{12}	r_{13}	True
	C		(S, r_{13})				False
C	r_4	A,B	$(S, r_{13}), (C, r_4)$				False
A	r_6	h_c, B	$(S, r_{13}), (C, r_4), (A, r_6)$	A	r_6		False
h_c	∞	B	$(S, r_{13}), (C, r_4)$				

• Thuật toán suy diễn lùi trên đồ thị suy diễn lùi VÀ-HOẶC

Sự kiện	luật	Đích	Vết	sự kiện hỏng	luật hỏng	luật thay thế	Quay lui
	S		\emptyset				False
S	r_{12}	h_a	(S, r_{12})				False
h_a	∞	\emptyset	\emptyset	S	r_{12}	r_{13}	True
	C		(S, r_{13})				False
C	r_4	A,B	$(S, r_{13}), (C, r_4)$				False
A	r_6	h_c, B	$(S, r_{13}), (C, r_4), (A, r_6)$	A	r_6	r_7	False
h_c	∞	B	$(S, r_{13}), (C, r_4)$				True

• Thuật toán suy diễn lùi trên đồ thị suy diễn lùi VÀ-HOẶC

Sự kiện	luật	Đích	Vết	sự kiện hỏng	luật hỏng	luật thay thế	Quay lui
	S		\emptyset				False
S	r_{12}	h_a	(S, r_{12})				False
h_a	∞	\emptyset	\emptyset		S	r_{12}	r_{13}
	C		(S, r_{13})				True
C	r_4	A,B	$(S, r_{13}), (C, r_4)$				False
A	r_6	h_c, B	$(S, r_{13}), (C, r_4), (A, r_6)$				False
h_c	∞	B	$(S, r_{13}), (C, r_4)$		A	r_6	r_7
		B	$(S, r_{13}), (C, r_4), (A, r_7)$				True
							False

• Thuật toán suy diễn lùi trên đồ thị suy diễn lùi VÀ-HOẶC

Sự kiện	luật	Đích	Vết	sự kiện hỏng	luật hỏng	luật thay thế	Quay lui
	S		\emptyset				False
S	r_{12}	h_a	(S, r_{12})				False
h_a	∞	\emptyset	\emptyset	S	r_{12}	r_{13}	True
		C	(S, r_{13})				False
C	r_4	A,B	$(S, r_{13}), (C, r_4)$				False
A	r_6	h_c, B	$(S, r_{13}), (C, r_4), (A, r_6)$				False
h_c	∞	B	$(S, r_{13}), (C, r_4)$	A	r_6	r_7	True
		B	$(S, r_{13}), (C, r_4), (A, r_7)$				False
B	r_5	hc	$(S, r_{13}), (C, r_4), (A, r_7), (B, r_5)$				False

• Thuật toán suy diễn lùi trên đồ thị suy diễn lùi VÀ-HOẶC

Sự kiện	luật	Đích	Vết	sự kiện hỏng	luật hỏng	luật thay thế	Quay lui
	S		\emptyset				False
S	r_{12}	h_a	(S, r_{12})				False
h_a	∞	\emptyset	\emptyset	S	r_{12}	r_{13}	True
	C		(S, r_{13})				False
C	r_4	A,B	$(S, r_{13}), (C, r_4)$				False
A	r_6	h_c, B	$(S, r_{13}), (C, r_4), (A, r_6)$				False
h_c	∞	B	$(S, r_{13}), (C, r_4)$	A	r_6	r_7	True
		B	$(S, r_{13}), (C, r_4), (A, r_7)$				False
B	r_5	hc	$(S, r_{13}), (C, r_4), (A, r_7), (B, r_5)$				False
hc	∞	\emptyset					False

• Thuật toán suy diễn lùi trên đồ thị suy diễn lùi VÀ-HOẶC

Sự kiện	luật	Đích	Vết	sự kiện hỏng	luật hỏng	luật thay thế	Quay lui
	S		\emptyset				False
S	r_{12}	h_a	(S, r_{12})				False
h_a	∞	\emptyset	\emptyset	S	r_{12}	r_{13}	True
	C		(S, r_{13})				False
C	r_4	A,B	$(S, r_{13}), (C, r_4)$				False
A	r_6	h_c, B	$(S, r_{13}), (C, r_4), (A, r_6)$				False
h_c	∞	B	$(S, r_{13}), (C, r_4), (A, r_6)$	A	r_6	r_7	True
	B		$(S, r_{13}), (C, r_4), (A, r_7)$				False
B	r_5	hc	$(S, r_{13}), (C, r_4), (A, r_7), (B, r_5)$				False
hc	∞	\emptyset	$(S, r_{13}), (C, r_4), (A, r_7)$	B	r_5		False

• Thuật toán suy diễn lùi trên đồ thị suy diễn lùi VÀ-HOẶC

Sự kiện	luật	Đích	Vết	sự kiện hỏng	luật hỏng	luật thay thế	Quay lui
	S		\emptyset				False
S	r_{12}	h_a	(S, r_{12})				False
h_a	∞	\emptyset	\emptyset	S	r_{12}	r_{13}	True
	C		(S, r_{13})				False
C	r_4	A,B	$(S, r_{13}), (C, r_4)$				False
A	r_6	h_c, B	$(S, r_{13}), (C, r_4), (A, r_6)$				False
h_c	∞	B	$(S, r_{13}), (C, r_4), (A, r_6)$	A	r_6	r_7	True
	B		$(S, r_{13}), (C, r_4), (A, r_7)$				False
B	r_5	hc	$(S, r_{13}), (C, r_4), (A, r_7), (B, r_5)$				False
hc	∞	\emptyset	$(S, r_{13}), (C, r_4), (A, r_7)$	B	r_5	r_8	True

• Thuật toán suy diễn lùi trên đồ thị suy diễn lùi VÀ-HOẶC

Sự kiện	luật	Đích	Vết	sự kiện hỏng	luật hỏng	luật thay thế	Quay lui
	S		\emptyset				False
S	r_{12}	h_a	(S, r_{12})				False
h_a	∞	\emptyset	\emptyset	S	r_{12}	r_{13}	True
	C		(S, r_{13})				False
C	r_4	A,B	$(S, r_{13}), (C, r_4)$				False
A	r_6	h_c, B	$(S, r_{13}), (C, r_4), (A, r_6)$				False
h_c	∞	B	$(S, r_{13}), (C, r_4), (A, r_6)$	A	r_6	r_7	True
	B		$(S, r_{13}), (C, r_{14}), (A, r_7)$				False
B	r_5	hc	$(S, r_{13}), (C, r_4), (A, r_7), (B, r_5)$				False
hc	∞	\emptyset	$(S, r_{13}), (C, r_4), (A, r_7)$	B	r_5	r_8	True
	\emptyset		$(S, r_{13}), (C, r_4), (A, r_7), (B, r_8)$				False

• Thuật toán suy diễn lùi trên đồ thị suy diễn lùi VÀ-HOẶC

Sự kiện	luật	Đích	Vết	skiện hỏng	luật hỏng	luật thế	Quay lui
	S	\emptyset					False
S	r_{12}	h_a	(S, r_{12})				False
h_a	∞	\emptyset	\emptyset	S	r_{12}	r_{13}	True
		C	(S, r_{13})				False
C	r_4	A,B	$(S, r_{13}), (C, r_4)$				False
A	r_6	h_c, B	$(S, r_{13}), (C, r_4), (A, r_6)$				False
h_c	∞	B	$(S, r_{13}), (C, r_4), (A, r_6)$	A	r_6	r_7	True
		B	$(S, r_{13}), (C, r_4), (A, r_7)$				False
B	r_5	hc	$(S, r_{13}), (C, r_4), (A, r_7), (B, r_5)$				False
hc	∞	\emptyset	$(S, r_{13}), (C, r_4), (A, r_7), (B, r_5)$	B	r_5		False
		\emptyset	$(S, r_{13}), (C, r_4), (A, r_7), (B, r_8)$				False

Logic vị từ

Hạn chế logic mệnh đề

- Khả năng diễn đạt hạn chế
 - Biểu diễn các sự kiện
 - Không thể mô tả đầy đủ thế giới các đối tượng, tính chất và mối quan hệ giữa các đối tượng
 - Ví dụ
 - 1) Nam là một sinh viên của Thủy Lợi
 - 2) Mọi sinh viên của Thủy Lợi đều học tin đại cương
 - 3) Vì Nam là sinh viên của Thủy Lợi nên Nam học tin đại cương
 - . Trong logic mệnh đề: 3 không thể suy ra từ 1 và 2

Logic vị từ

- Mở rộng của logic mệnh đề
- Vị từ là các phát biểu có chứa biến
 - $\text{TL_Student}(x)$: x là sinh viên của thủy lợi
 - $\text{TL_Student}(\text{Nam})$: Nam là sinh viên của thủy lợi
 - $\text{Studies_Tin}(x)$: x học tin
 - $\text{Studies_Tin}(\text{Nam})$: Nam học tin
 - $\forall x: \text{TL_Student}(x) \rightarrow \text{Studies_Tin}(x)$: Mọi sinh viên của Thủy lợi đều học môn Tin đại cương

Thành phần cú pháp

- **Hằng:** chỉ các đối tượng cụ thể trong lĩnh vực chuyên môn nào đó
 - a, b, c, Mai, Nam, John,...
- **Biến:** chỉ đối tượng chung chung
 - x,y,z,u,v,w,...
- **Hàm:** chỉ sự chuyển đổi đối tượng
 - best_friend(x), father(x), distance(x,y), ...
- **Vị từ:** chỉ các mối quan hệ giữa các đối tượng hoặc tính chất của đối tượng
 - friend(x,y), father(x,y), love(x,y), good(x),...

Lượng tử logic “Với mọi”

- \forall (biến₁, biến₂, ..., biến_n): <mệnh đề>
- Ví dụ:
 - An là bạn của tất cả mọi người
 - $\forall x: \text{friend}(An, x)$
 - Tất cả những người nuôi động vật đều yêu động vật
 - $\forall x, \forall y: \text{rear}(x, y) \wedge \text{animal}(y) \rightarrow \text{love_animal}(x)$
 - Tất cả các bạn sinh viên công nghệ thông tin k56 thủy lợi đều chăm học
 - $\forall x: \text{tlu_cntt56_student}(x) \rightarrow \text{chăm}(x)$

Lượng tử logic “Với mọi”

- Mệnh đề $(\forall x:P)$ đúng khi và chỉ khi P đúng với mọi đối tượng trong thế giới
- Ví dụ:
 - An là bạn của tất cả mọi người đúng khi
 - friend(An, Nam) \wedge friend (An, Mai) \wedge friend(An,Son)
 - Tất cả các bạn sinh viên công nghệ thông tin k56 thủy lợi đều chăm học đúng khi:
 - . tlu_cntt56_student(Vân) \rightarrow chăm(Vân)
 - . \wedge tlu_cntt56_student(Hùng) \rightarrow chăm(Hùng)
 - . \wedge tlu_cntt56_student(Bình) \rightarrow chăm(Bình)

Lượng tử logic “Tồn tại”

- \exists (biến₁, biến₂, ..., biến_n): <mệnh đề>
- Ví dụ:
 - Có một người là bạn của An
 - . $\exists x: \text{friend}(\text{An}, x)$
 - Tồn tại một người nuôi một con động vật nào đấy nhưng lại không yêu động vật
 - . $\exists x, \exists y: \text{rear}(x, y) \wedge \text{animal}(y) \wedge \neg \text{love_animal}(x)$
 - Tồn tại một bạn sinh viên công nghệ thông tin k56 thủy lợi và chăm học
 - . $\exists x: \text{tlu_cntt56_student}(x) \wedge \text{cham}(x)$
 - Tồn tại 1 bạn sinh viên học tất cả các môn học của

Lượng tử logic “Tồn tại”

- Mệnh đề ($\exists x:P$) đúng khi P đúng với ít nhất một đối tượng trong thế giới
- Ví dụ:
 - An là bạn của ít nhất 1 người đúng khi friend(An, Nam) v friend (An, Mai) v friend(An,Son)
 - Ít nhất một bạn sinh viên công nghệ thông tin k56 thủy lợi đều chăm học đúng khi:
 - $\text{tlu_cntt56_student(Vân)} \rightarrow \text{chăm(Vân)}$
 - v $\text{tlu_cntt56_student(Hùng)} \rightarrow \text{chăm(Hùng)}$
 - v $\text{tlu_cntt56_student(Bình)} \rightarrow \text{chăm(Bình)}$

- Đặc điểm của logic lượng tử
- Tính hoán vị
 - $(\forall x \forall y) \equiv (\forall y \forall x)$
 - $(\exists x \exists y) \equiv (\exists y \exists x)$
- Tuy nhiên, $(\exists x \forall y)$ không tương đương với $(\forall x \exists y)$
 - $\exists x \forall y \text{ Know}(x,y)$: tồn tại 1 người biết tất cả các lĩnh vực
 - $\forall x \exists y \text{ Know}(x,y)$: tồn tại 1 lĩnh vực mà tất cả mọi người cùng biết
- Đưa các phép lượng tử vào từng vị trí

• Đặc điểm của logic lượng tử

- Đặt lại tên biến

$$\forall x G(x) \equiv \forall y G(y)$$

$$\exists x G(x) \equiv \exists y G(y)$$

- Loại bỏ \neg

$$\neg(\forall x G(x)) \equiv \exists x (\neg G(x))$$

$$\neg(\exists x G(x)) \equiv \forall x (\neg G(x))$$

- Hệ quả: mỗi lượng tử (\forall, \exists) đều có thể biểu diễn bởi lượng tử kia

- $[\forall x \text{ friend}(An, x)] \equiv [\neg \exists x \neg \text{friend}(An, x)]$

Chứng minh trong logic vị từ

- Tương tự trong logic mệnh đề, sử dụng thêm phép gán giá trị
- Phương pháp chứng minh bằng luật hợp giải
 - Giả sử kết luận sai
 - Chuẩn hóa về dạng chuẩn hội và tách dòng
 - Hợp giải: tìm 2 dòng để hợp giải cho đến khi gặp ><
 - dòng i chứa $G(x, \dots)$
 - dòng j chứa $\neg G(A, \dots)$
 - Bằng phép gán giá trị $[x|A]$, hợp nhất 2 dòng i, j và triệt tiêu G

Chứng minh trong logic vị từ

- Ví dụ 1: cho cơ sở tri thức sau:
 - 1)Ông Ba **nuôi** 1 con chó
 - 2)Hoặc ông Ba hoặc ông An **đã giết** con mèo Bibi
 - 3)Mọi người nuôi chó đều **yêu quý động vật**
 - 4)Ai yêu quý động vật cũng ko giết động vật
 - 5)**Chó mèo** đều là **động vật**
- Câu hỏi: ai **đã giết** Bibi?

Chứng minh trong logic vị từ

- Khai báo các vị từ

- Rear(x,y): x nuôi y

- Kill(x,y): x giết y

- Animal(x): x là động vật

- AnimalLover(x): x là

- người yêu động vật

- Dog(x): x là loài chó

- Cat(x): x

- là loài mèo

- Biểu diễn tri thức

- 1)Dog(D) \wedge Rear(Ba,D)

- 2)Cat(Bibi) \wedge (Kill(Ba,Bibi) \vee Kill(An,Bibi))

- 3) $\forall x (\forall y \text{Rear}(x,y) \wedge \text{Dog}(y) \rightarrow \text{AnimalLover}(x))$

Chứng minh trong logic vị từ

- Chứng minh ông An giết con mèo Bibi
- Chuẩn hóa và tách
 - 1)Dog(D)
 - 2)Rear(Ba,D)
 - 3)Cat(Bibi)
 - 4)Kill(Ba,Bibi) v Kill(An,Bibi)
 - 5) \neg Rear(x,y) v \neg Dog(y) v AnimalLover(x)
 - 6) \neg AnimalLover(x) v \neg Animal(y) v \neg Kill(x,y)
 - 7) \neg Dog(x) v Animal(x)
 - 8) \neg Cat(x) v Animal(x)

Chứng minh trong logic vị từ

- Hợp giải

10) Kill(Ba,Bibi) : Res(4,9)

11) $\neg \text{AnimalLover}(\text{Ba}) \vee \neg \text{Animal}(\text{Bibi})$: Res(10,6),
[$x^6|\text{Ba}, y^6|\text{Bibi}$]

12) $\neg \text{AnimalLover}(\text{Ba}) \vee \neg \text{Cat}(\text{Bibi})$: Res (11,8),
[$x^8|\text{Bibi}$]

13) $\neg \text{AnimalLover}(\text{Ba})$: Res (12,3)

14) $\neg \text{Rear}(\text{Ba}, y) \vee \neg \text{Dog}(y)$: Res (13,5), [$x^{14}|\text{Ba}$]

15) $\neg \text{Dog}(\text{D})$: Res (14,2), [$y^{14}|\text{D}$]

16) >< : Res (15,1)

Chứng minh trong logic vị từ

- Chứng minh ông Ba giết con mèo Bibi?
 - Phương pháp hợp giải không dẫn tới câu rỗng → không chứng minh được
- Chứng minh ông Ba không giết con mèo Bibi?
 - Chứng minh được bằng phương pháp hợp giải

Chứng minh trong logic vị từ

- Ví dụ 2: cho cơ sở tri thức sau
 - 1. Hùng thích tất cả các loại **thực phẩm**
 - 2. Táo là thực phẩm
 - 3. Gà là thực phẩm
 - 4. Bất cứ thứ gì mọi người **ăn** và không bị hại đó là thực phẩm
 - 5. Phong ăn đậu phộng và vẫn **còn sống**
 - 6. Lan ăn bất cứ thứ gì Phong ăn
- Sử dụng phương pháp hợp giải để:
- Chứng minh Hùng thích đậu phộng
 - Trả lời câu hỏi “Lan ăn thực phẩm nào”

Chứng minh trong logic vị từ

- Ví dụ 3: cho cơ sở tri thức
 - 1. Bất kì ai đi **lừa dối** người khác đều bị coi là **bịp bợm**
 - 2. Dù vô tình hay cố ý, đã có lần **đồng tình** với bạn đi lừa dối người khác, cũng sẽ bị coi là bịp bợm
 - 3. Do tính tình **nhút nhát** nên có thể có người trong 1 hoàn cảnh nào đó trót đồng tình với người khác để lừa dối.
 - 4. Từ đó có thể thấy rằng không phải ai bị coi là bịp bợm cũng không nhút nhát (Kết luận)

Chứng minh trong logic vị từ

- Khai báo các vị từ
 - $Ld(x)$: x là người lừa dối
 - $Bb(x)$: x là người bịp bợm
 - $Dt(x,y)$: x đồng tình với y
 - $Nn(x)$: x là người nhút nhát
- Biểu diễn tri thức
 1. $Ld(x) \rightarrow Bb(x)$
 2. $Dt(x,y) \wedge Bb(y) \rightarrow Bb(x)$
 3. $\exists x (Nn(x) \wedge \exists y(Dt(x,y) \wedge Ld(y)))$
 4. Kết luận $\neg(\forall x Bb(x) \rightarrow \neg Nn(x))$

Chứng minh trong logic vị từ

- Thêm \neg (Kết luận) là $(\forall x Bb(x) \rightarrow \neg Nn(x))$
- Chuẩn hóa và tách
 - 1. $\neg Ld(x) \vee Bb(x)$
 - 2. $\neg Dt(x,y) \vee \neg Bb(y) \vee Bb(x)$
 - 3. $Nn(A)$
 - 4. $Dt(A,B)$
 - 5. $Ld(B)$
 - 6. $\neg Bb(x) \vee \neg Nn(x)$
- Hợp giải

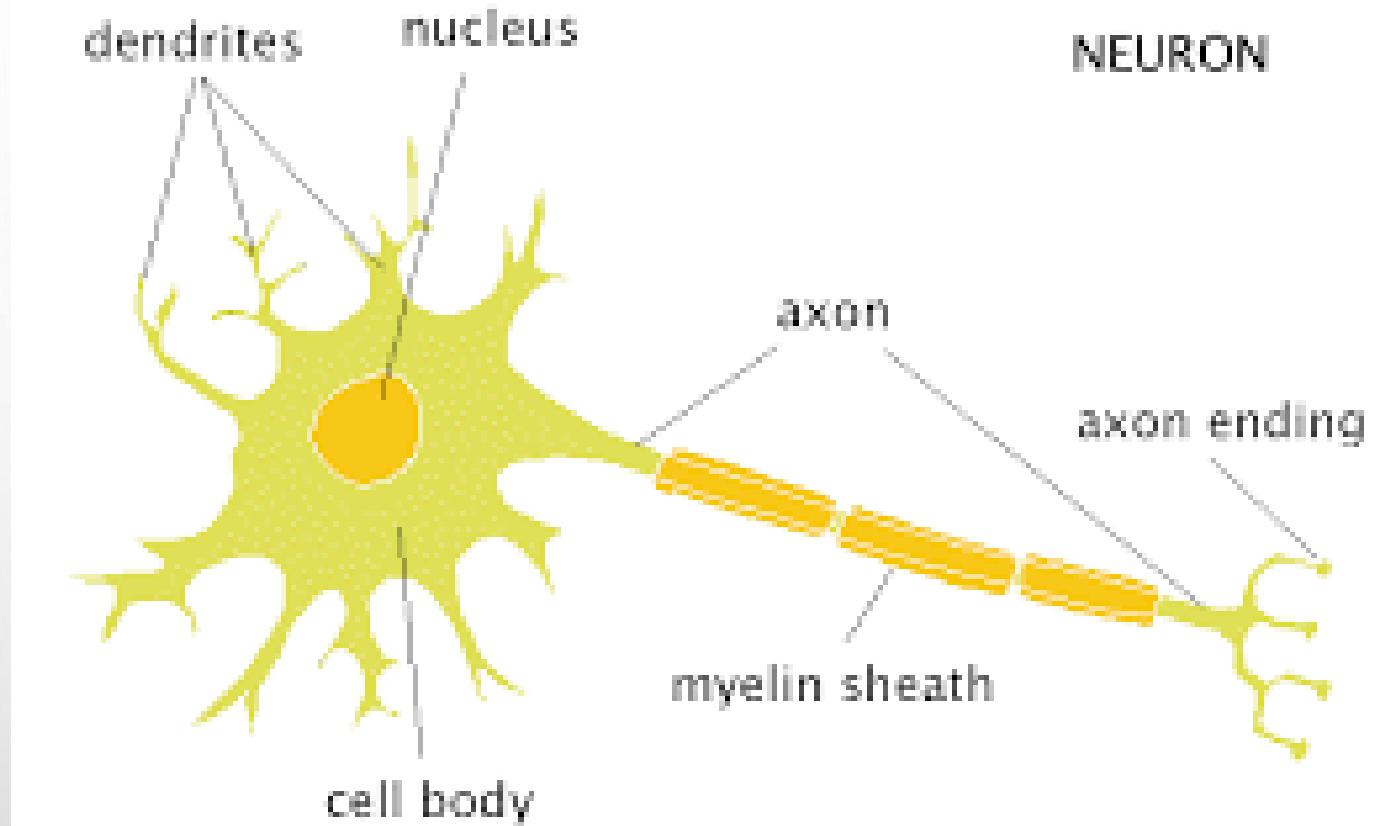
Chương 4

Mạng nơron nhân tạo

Mạng nơron nhân tạo

- Kỹ thuật tính toán mềm dựa trên mô phỏng hoạt động của não bộ (mạng nơron thần kinh)

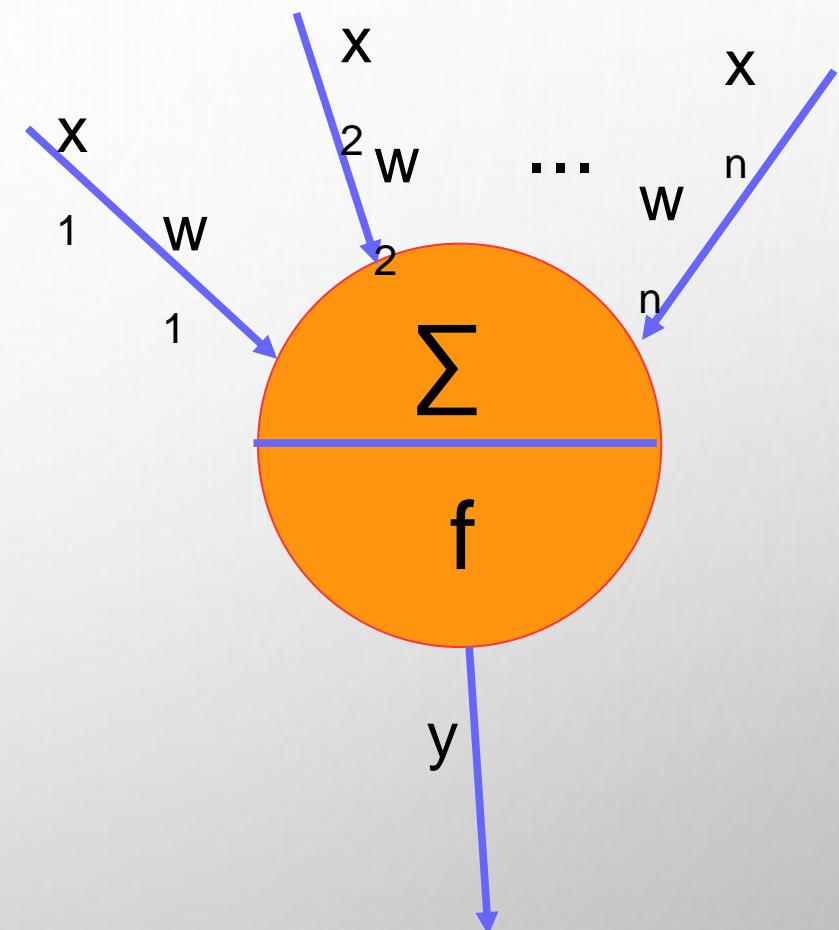
Nơron thần kinh



- Soma: thân nơron, tiếp nhận hoặc phát ra các xung động thần kinh
- Dendrites: dây thần kinh vào, đưa tín hiệu tới nơron
- Axon: đầu dây thần kinh ra, nối với dây thần kinh vào hoặc tới nhân tế bào của nơron khác thông qua khớp nối
- Synapse: khớp để kích hoạt hoặc kích thích thông tin

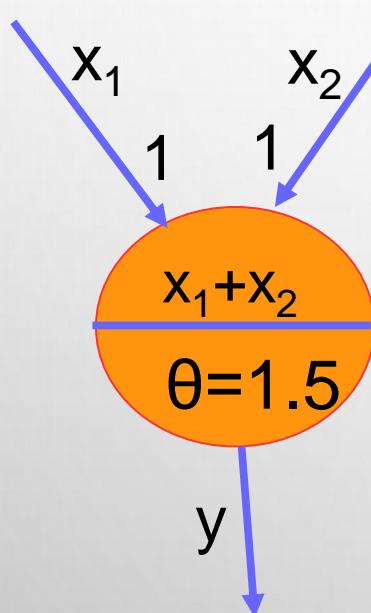
Nơron nhân tạo

- Tổng thông tin vào của 1 nơron
- $\text{Net} = \sum w_i x_i$
- Đầu ra
- $y = f(\text{Net}) = f(\sum w_i x_i)$
- f được gọi là hàm truyền (transfer function) hay kích hoạt (activation function)

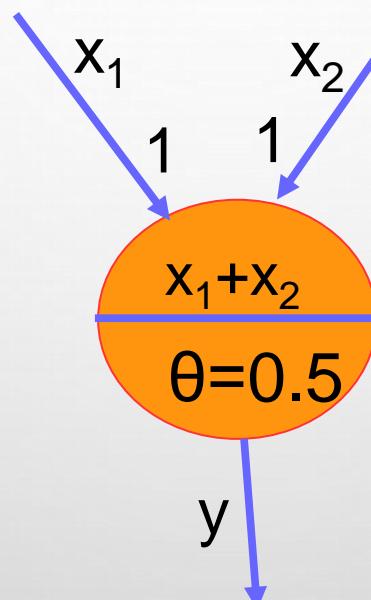


Nơron nhân tạo

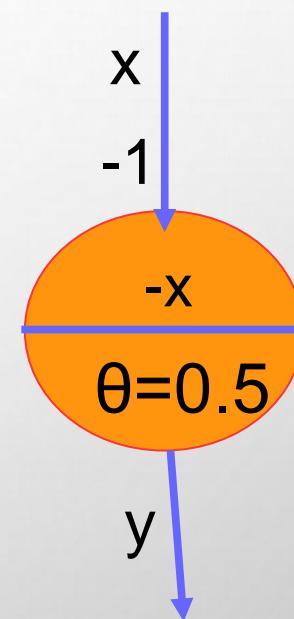
- Bản thân mỗi nơron như 1 hàm tính và tính mạnh hơn cỗng logic cơ bản



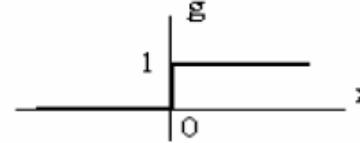
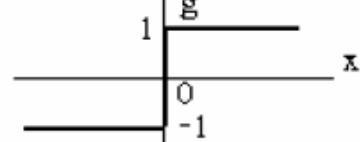
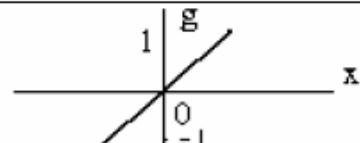
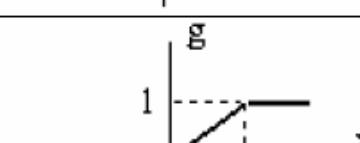
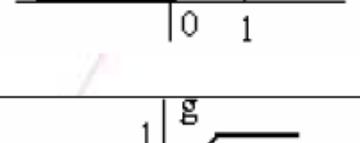
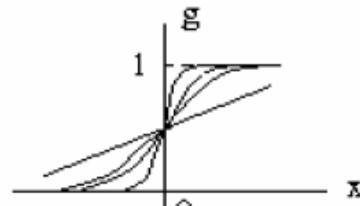
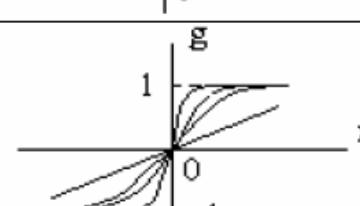
$$y = x_1 \text{ and } x_2$$



$$y = x_1 \text{ or } x_2$$

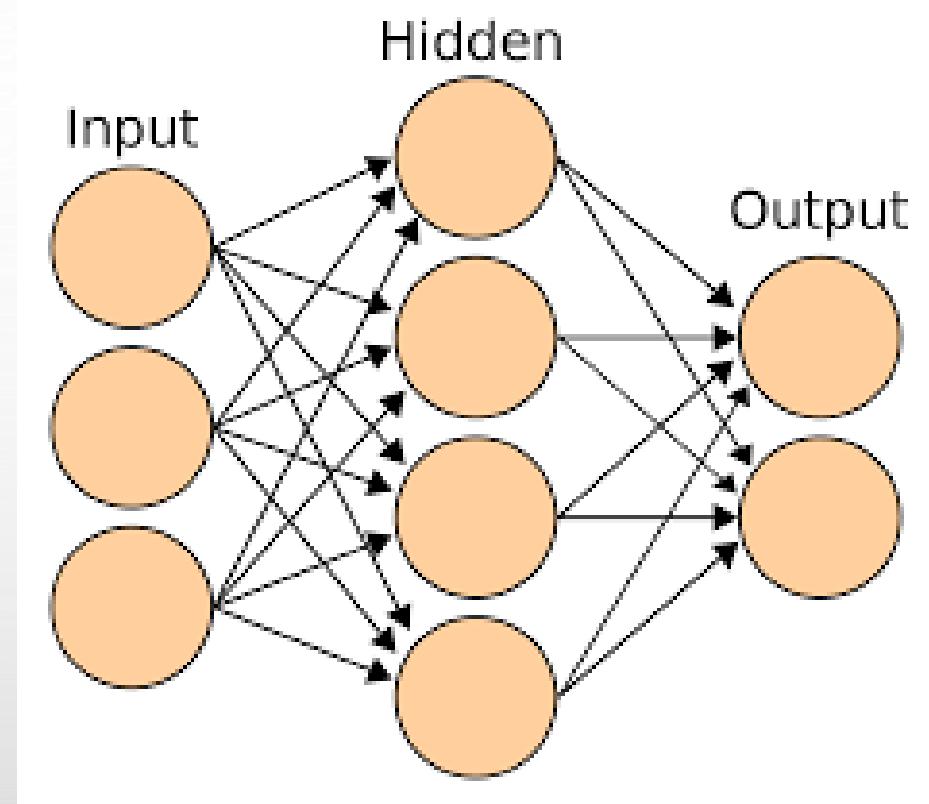


$$y = \text{not } x$$

Tên hàm	Công thức	Đặc tính
Bước nhảy đơn vị	$g(x) = \begin{cases} 1 & \text{nếu } x \geq 0 \\ 0 & \text{nếu } x < 0 \end{cases}$	
Hard limiter (sgn)	$g(x) = \begin{cases} 1 & \text{nếu } x \geq 0 \\ -1 & \text{nếu } x < 0 \end{cases}$	
Hàm tuyến tính	$g(x) = x$	
Hàm tuyến tính bão hòa	$g(x) = \begin{cases} 1 & \text{nếu } x > 1 \\ x & \text{nếu } 0 \leq x \leq 1 \\ 0 & \text{nếu } x < 0 \end{cases}$	
Hàm tuyến tính bão hòa đối xứng	$g(x) = \begin{cases} 1 & \text{nếu } x > 1 \\ x & \text{nếu } -1 \leq x \leq 1 \\ -1 & \text{nếu } x < -1 \end{cases}$	
Hàm Sigmoid đơn cực (Unipolar Sigmoid)	$g(x) = \frac{1}{1 + e^{-\lambda x}}$	
Hàm Sigmoid lưỡng cực (Bipolar Sigmoid)	$g(x) = \frac{2}{1 + e^{-\lambda x}} - 1$	

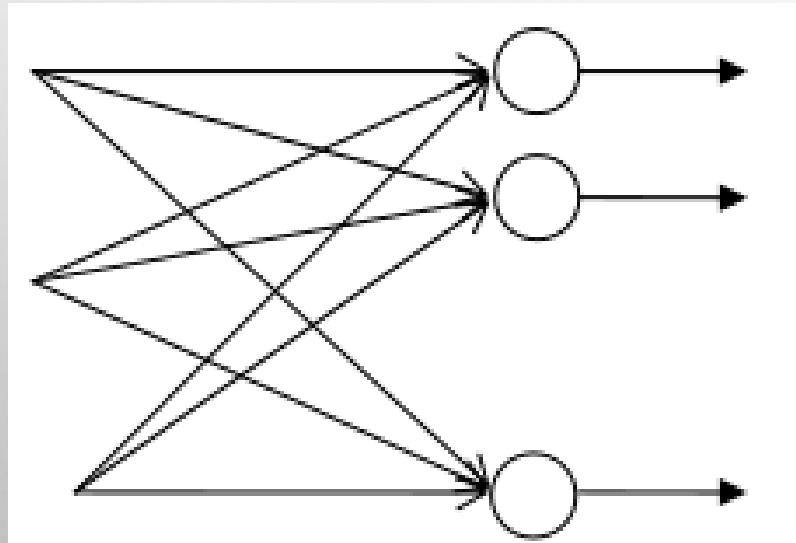
Cấu trúc mạng nơron

- Số lượng tín hiệu vào/ra
- Số lớp ẩn
- Số nơron của mỗi lớp ẩn
- Trọng số các liên kết
- Ngưỡng của các nơron
- Hàm kích hoạt



Mô hình mạng nơron

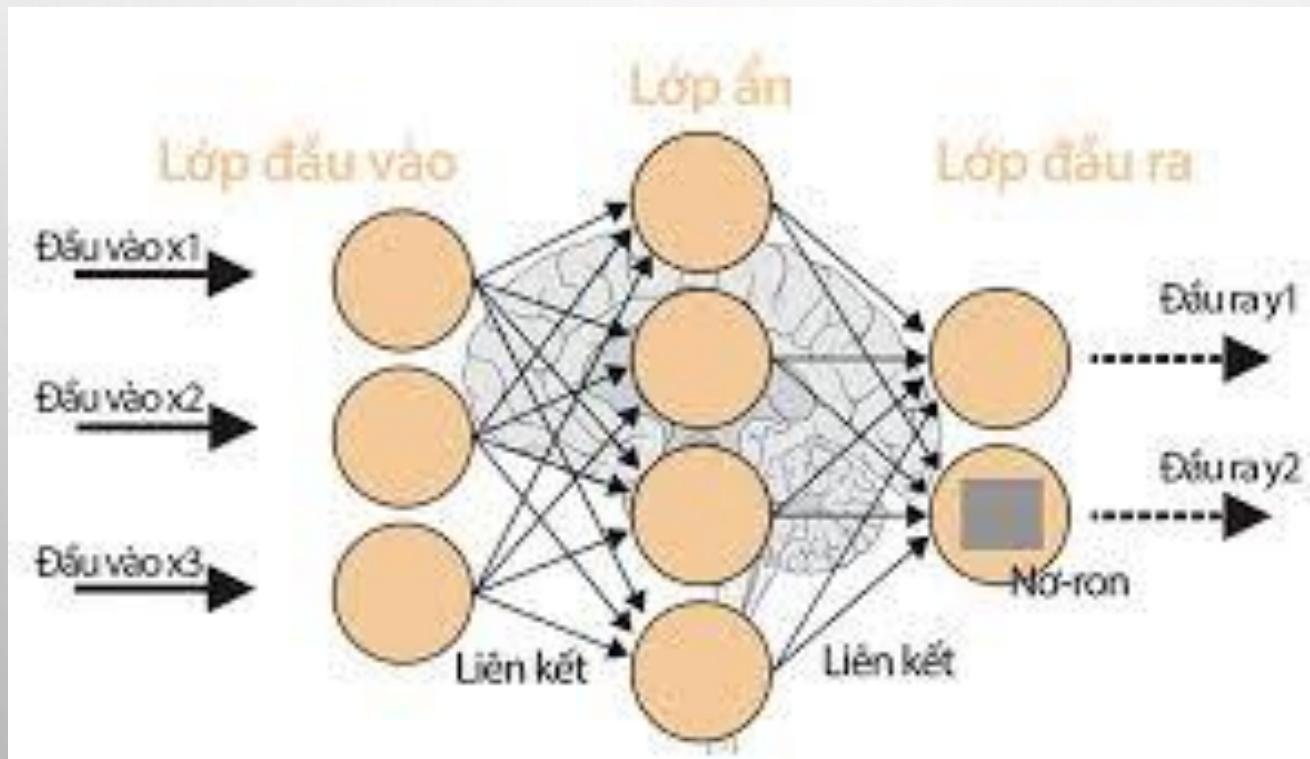
- 2 loại:
 - Không chu trình: mạng truyền thẳng (feed forward NN)
 - Có chu trình: mạng hồi quy (recurrent NN)



Mạng truyền thẳng 1 lớp

Mô hình mạng nơron

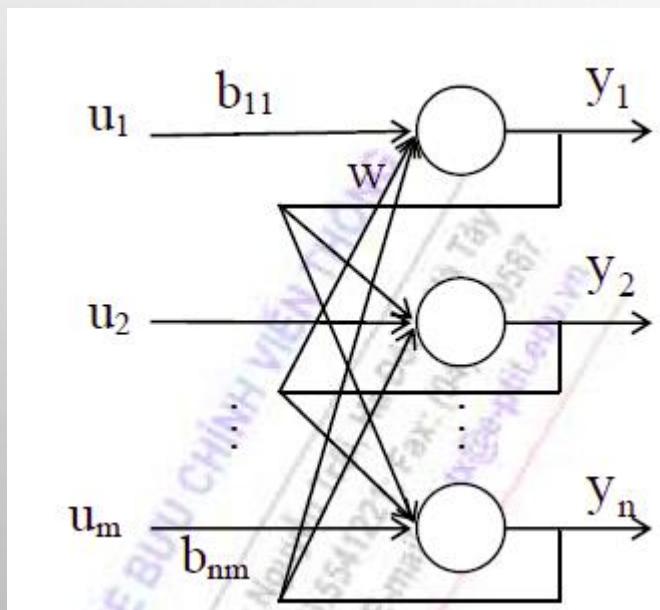
- 2 loại:
 - Không chu trình: mạng truyền thẳng (feed forward NN)
 - Có chu trình: mạng hồi quy (recurrent NN)



Mạng truyền
thẳng 3 lớp

Mô hình mạng nơron

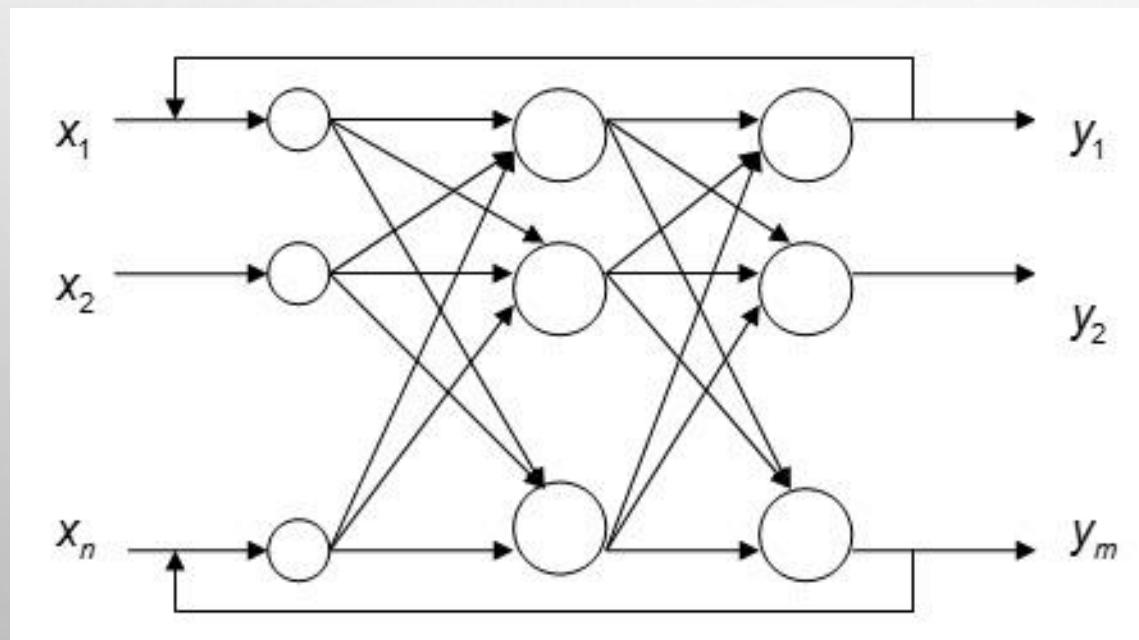
- 2 loại:
 - Không chu trình: mạng truyền thẳng (feed forward NN)
 - Có chu trình: mạng hồi quy (recurrent NN)



Mạng hồi quy 1 lớp

Mô hình mạng nơron

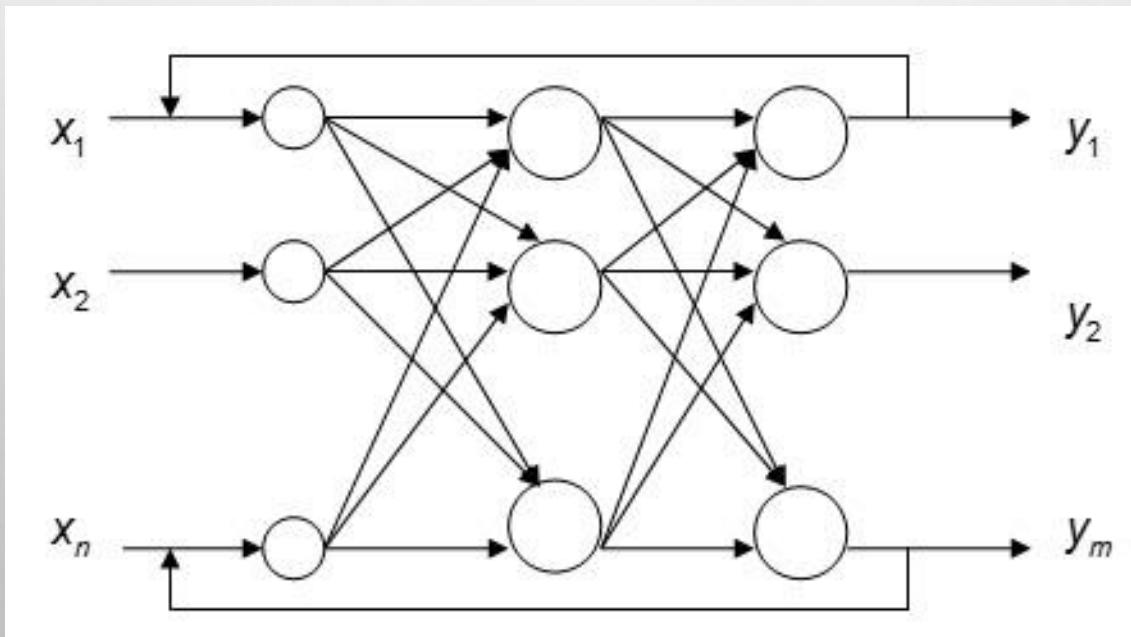
- 2 loại:
 - Không chu trình: mạng truyền thẳng (feed forward NN)
 - Có chu trình: mạng hồi quy (recurrent NN)



*Mạng hồi quy
nhiều lớp*

Mô hình mạng nơron

- Mạng hồi quy nhiều lớp(recurrent neural network)
 - chứa các liên kết ngược có sự kết nối giữa neural đầu ra với neural đầu vào
 - lưu lại các trạng thái đầu ra trước đó
 - trạng thái đầu ra tiếp theo không chỉ phụ thuộc vào các tín hiệu đầu vào mà còn phụ thuộc vào các trạng thái trước đó của mạng



Thiết kế mạng nơron

- Thiết kế mạng
 - Thủ công
 - Tự động: bằng thuật toán học để tín hiệu đầu ra của mạng thu được giống như mong muốn
 - Học cấu trúc: tìm ra cấu trúc mạng (số lớp, số nơron/lớp) hợp lý
 - Học tham số: tìm ra các trọng số liên kết hợp lý (giả sử cấu trúc của mạng là cố định)
- Các kiểu học cho học tham số
 - Có thầy, có giám sát (supervised learning)
 - Không thầy, không có giám sát (unsupervised learning)

Các phương pháp học

- Học có giám sát:

- Biết được đầu ra mong muốn
- Tập mẫu để huấn luyện mạng $\{(X^i, D^i)\}$ trong đó
 - $X^i = (x_1^i, x_2^i, \dots, x_m^i)$ là tín bộ hiệu đầu vào thứ i trong tập huấn luyện
 - $D^i = (d_1^i, d_2^i, \dots, d_n^i)$ là bộ tín hiệu đầu ra mong muốn cho tín hiệu đầu vào X^i

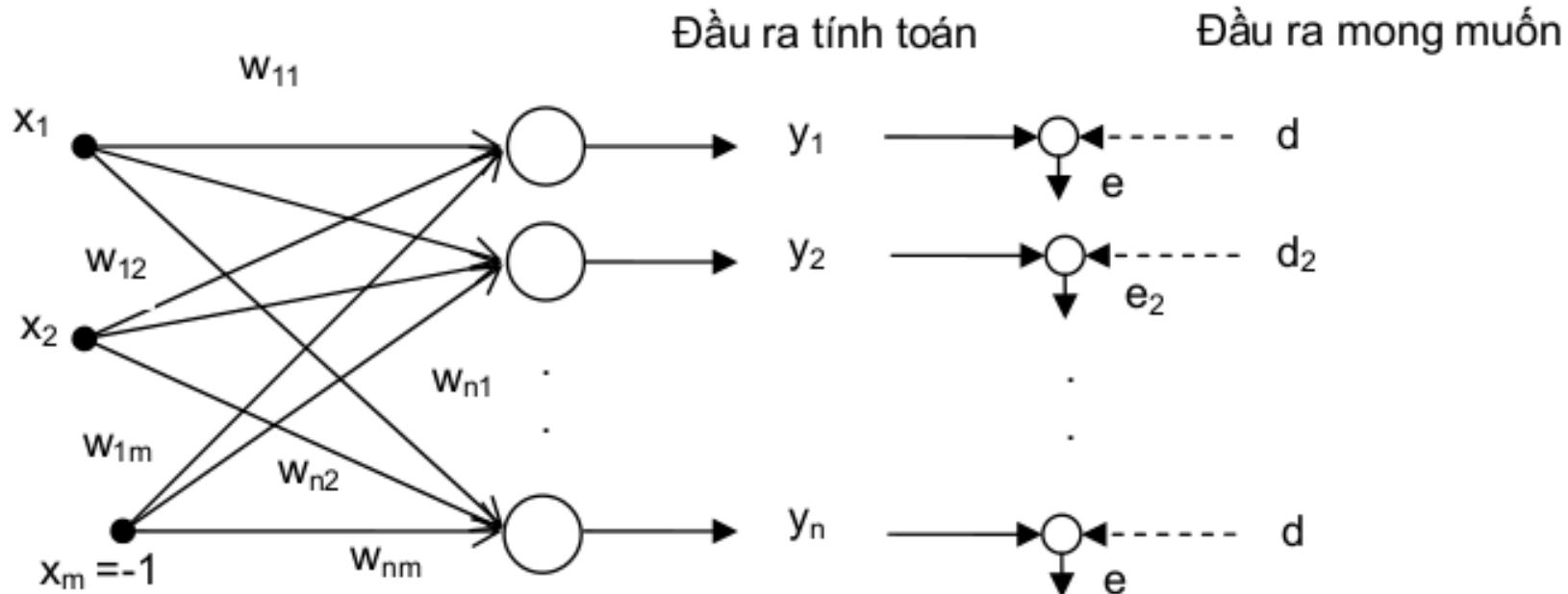
- Học không có giám sát

- Không biết đầu ra mong muốn
- Có 1 hàm chi phí (hàm bất kì của dữ liệu đầu vào X và đầu ra Y của mạng) cần được cực tiểu

Giải thuật học lan truyền ngược sai số

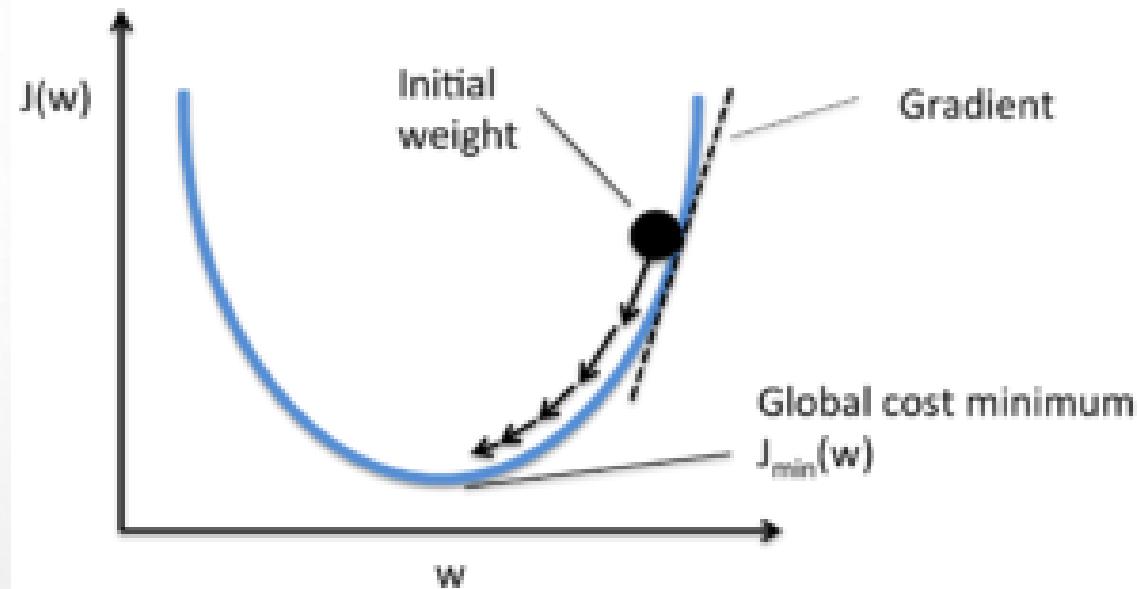
- Học tham số có giám sát
- Đã biết trước cấu trúc mạng
 - Số nơron lớp vào/ra
 - Số lớp ẩn
 - Số nơron/lớp ẩn
 - Các đường liên kết giữa các nơron (đầy đủ)
- Tập mẫu: lấy được từ quan sát, đo đạc, thực địa, thực nghiệm

Giải thuật học lan truyền ngược sai số



- Với 1 bộ dữ liệu đầu vào $X=(x_1, x_2, \dots, x_m)$
- Đầu ra mong muốn $D=(d_1, d_2, \dots, d_n)$
- Đầu ra thu được $Y=(y_1, y_2, \dots, y_n)$
- Sai số giữa đầu ra thu được so với đầu ra

Giải thuật học lan truyền ngược sai số



- Phương pháp giảm gradient

- Hàm lỗi (tổng chêch lệch giữa đầu ra thu được và đầu ra mong muốn) là 1 hàm $f(w)$ của các trọng số liên kết
- Cần tìm ra 1 trọng số w mà tại đó hàm lỗi là nhỏ nhất
- Hàm lỗi sẽ giảm dần mỗi khi học 1 dữ liệu mẫu. Tức là đầu ra thu được của mạng sẽ tiến sát dần đến đầu ra mong muốn.

Giải thuật học lan truyền ngược sai số

- Khởi tạo các trọng số liên kết nơron 1 cách ngẫu nhiên tương đối nhỏ, ví dụ € [0,1]
- Quá trình học
 - Với mỗi mẫu đầu vào X^k , tính đầu ra Y^k . Việc tính toán được thực hiện song song trong cùng 1 lớp, và tuần tự theo thứ tự các lớp
 - Xét nơron thứ q của lớp ẩn với dữ liệu đưa vào mạng X (viết lược giản cho X^k)
 - **Đầu vào** $\text{net}_q = \sum_{j=0}^m v_{qj}x_j$ trong đó v_{qj} là trọng số liên kết giữa nơron thứ j của lớp vào với nơron thứ q của lớp ẩn
 - **Đầu ra** $z_q = f(\text{net}_q) = f(\sum_{j=0}^m v_{qj}x_j)$
 - Xét nơron thứ i ở lớp ra

Giải thuật học lan truyền ngược sai số

- Nếu $Y^k = D^k$ nghĩa là mạng đã học được mẫu (X^k, D^k) , bỏ qua và xét mẫu tiếp theo
- Ngược lại $Y^k \neq D^k$, hiệu chỉnh lại trọng số W dựa theo sai số của đầu ra
 - $E(w) = \frac{1}{2} [\sum_{i=0}^l (d_i - y_i)]^2$
 - $= \frac{1}{2} [\sum_{i=0}^l (d_i - f(\text{net}_i))]^2$
 - $= \frac{1}{2} [\sum_{i=0}^n (d_i - f(\sum_{q=0}^l w_{iq} z_q))]^2$
 - Để tối thiểu hàm sai số $E(w)$, sử dụng phương pháp giảm gradient để điều chỉnh các trọng số liên kết. E sẽ đạt cực tiểu tại bộ giá trị nào đó của tham số W
 - $\Delta w = w^{(\text{new})} - w^{(\text{old})} = -\eta \cdot \frac{\partial E}{\partial w}$ trong đó η là hệ số h

Giải thuật học lan truyền ngược sai số

• Hiệu chỉnh lại

- Áp dụng phương pháp giảm gradient đối với các trọng số liên kết giữa các nơron trong lớp ẩn tới các nơ-ron của lớp ra ta có:
- $\Delta w_{iq} = -\eta \frac{\partial E}{\partial w_{iq}}$
- $= -\eta |\frac{\partial E}{\partial y_i}| \cdot |\frac{\partial y_i}{\partial net_i}| \cdot |\frac{\partial net_i}{\partial w_{iq}}|$
- $= -\eta |d_i - y_i| \cdot |g'(net_i)| \cdot z_q$
- $= \eta \delta_{oi} z_q$ trong đó đặt $\delta_{oi} = -|d_i - y_i| \cdot |g'(net_i)|$
- $w_{iq}^{\text{new}} = w_{iq}^{\text{old}} + \Delta w_{iq}$
- Tương tự áp dụng phương pháp giảm gradient đối với các trọng số liên kết giữa nơron lớp vào và lớp

Ứng dụng của mạng nơron

- Phân lớp
 - Nhận dạng chữ viết, chữ số
- Dự báo
 - Biết giá ngoại tệ của n ngày trước đó, dự báo giá ngoại tệ trong ngày hôm nay
- Điều khiển
 - Ra quyết định