TRÍ TUỆ NHÂN TẠO

Artificial Intelligence

TS. Nguyễn Đình Thuân
Khoa Công nghệ Thông tin
Đại học Nha Trang
Email: thuanvinh122@gmail.com

Nha Trang 8-2007

Nội dung môn học

- Chương 1: **Giới thiệu**
  - Mở đầu
  - Lĩnh vực nghiên cứu của AI
  - Ứng dụng của AI
  - Các vấn đề đặt ra

Nội dung môn học (tiếp)

- Chương 2:**Tìm kiếm trên không gian trạng thái**
  - Bài toán tìm kiếm
  - Giải thuật tổng quát
  - Depth first search (DFS)
  - Breath first search (BFS)

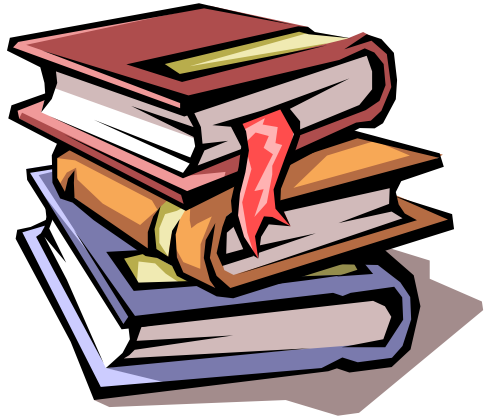
- Chương 3:**Tìm kiếm theo Heuristic**
  - Giới thiệu về Heuristic
  - Tìm kiếm theo heuristic
  - Giải thuật Best first search (BFS), Giải thuật A^T, A^{KT}, A*
  - Chiến lược Minimax, Alpha Beta

Nội dung môn học (tiếp)

- Chương 4:**Biểu diễn tri thức**
  - Bộ ba Đối tượng – Thuộc tính – Giá trị
  - Các luật dẫn
  - Mạng ngữ nghĩa
  - Frame
  - Logic mệnh đề, Logic vị từ
  - Thuật giải Vương Hạo, Thuật giải Robinson
- Chương 5: Máy học
  - Các hình thức học
  - Thuật giải Quinland
  - Học theo độ bất định

Thực hành &Tài liệu tham khảo

- **Thực hành** Prolog / C++ / Pascal
  - Các giải thuật tìm kiếm
  - Biểu diễn tri thức
  - Bài tập lớn



- **Tài liệu tham khảo**
  - Bài giảng "Trí tuệ nhân tạo" – TS Nguyễn Đình Thuân
  - Giáo trình "Trí tuệ nhân tạo" - GS Hoàng Kiếm– ĐHQGTPHCM
  - Trí tuệ nhận tạo–PGS Nguyễn Thanh Thủy–ĐH Bách Khoa HàNội
  - Artificial Inteligent – George F. Luget & Cilliam A. Stubblefied

Chương 1: GIỚI THIỆU

TS. Nguyễn Đình Thuân
Khoa Công nghệ Thông tin
Đại học Nha Trang
Email: thuanvinh122@gmail.com

1.1 Mở đầu

Trí tuệ là gì:

Theo từ điển Bách khoa toàn thư Webster:

- Trí tuệ là khả năng:
  - *Phản ứng một cách thích hợp lại những tình huống mới thông qua điều chỉnh hành vi một cách thích hợp.*
  - *Hiểu rõ mối liên hệ giữa các sự kiện của thế giới bên ngoài nhằm đưa ra những hành vi phù hợp để đạt được mục đích.*

Sự Thông Minh

Khái niệm về tính thông minh của một đối tượng thường biểu hiện qua các hoạt động:

- Sự hiểu biết và nhận thức được tri thức
- Sự lý luận tạo ra tri thức mới dựa trên tri thức đã có
- Hành động theo kết quả của các lý luận
- Kỹ năng (Skill)

TRI THỨC ???

Tri thức (Knowledge)

- Tri thức là những thông tin chứa đựng 2 thành phần
  - Các khái niệm:
    - Các khái niệm cơ bản: là các khái niệm mang tính quy ước
    - Các khái niệm phát triển: Được hình thành từ các khác niệm cơ bản thành các khái niệm phức hợp phức tạp hơn.
  - Các phương pháp nhận thức:
    - Các qui luật, các thủ tục
    - Phương pháp suy diễn, lý luận,..
- Tri thức là điều kiện tiên quyết của các hành xử thông minh hay "Sự thông minh"
- Tri thức có được qua sự thu thập tri thức và sản sinh tri thức
- Quá trình thu thập và sản sinh tri thức là hai quá trình song song và nối tiếp với nhau – không bao giờ chấm dứt trong một thực thể "Thông Minh"

Tri thức – Thu thập và sản sinh

- Thu thập tri thức:
  - Tri thức được thu thập từ thông tin, là kết quả của một quá trình thu nhận dữ liệu, xử lý và lưu trữ. Thông thường quá trình thu thập tri thức gồm các bước sau:
    - Xác định lĩnh vực/phạm vi tri thức cần quan tâm
    - Thu thập dữ liệu liên quan dưới dạng các trường hợp cụ thể.
    - Hệ thống hóa, rút ra những thông tin tổng quát, đại diện cho các trường hợp đã biết – Tổng quát hóa.
    - Xem xét và giữ lại những thông tin liên quan đến vấn đề cần quan tâm , ta có **các tri thức về vấn đề đó.**
- Sản sinh tri thức:
  - Tri thức sau khi được thu thập sẽ được đưa vào mạng tri thức đã có.
  - Trên cơ sở đó thực hiện các liên kết, suy diễn, kiểm chứng để sản sinh ra các tri thức mới.

Tri thức – Tri thức siêu cấp

- "Trí thức siêu cấp" (meta knowledge) hay "Tri thức về Tri thức"
  - Là các tri thức dùng để:
  - Đánh giá tri thức khác
  - Đánh giá kết quả của quá trình suy diễn
  - Kiểm chứng các tri thức mới

- Phương tiện truyền tri thức: ngôn ngữ tự nhiên

Hành xử thông minh – Kết luận

- Hành xử thông minh không đơn thuần là các hành động như là kết quả của quá trình thu thập tri thức và suy luận trên tri thức.
- Hành xử thông minh còn bao hàm
  - Sự tương tác với môi trường để nhận các phản hồi
  - Sự tiếp nhận các phản hồi để điều chỉnh hành động - Skill
  - Sự tiếp nhận các phản hồi để hiệu chỉnh và cập nhật tri thức
- Tính chất thông minh của một đối tượng là sự tổng hợp của cả 3 yếu tố: thu thập tri thức, suy luận và hành xử của đối tượng trên tri thức thu thập được. Chúng hòa quyện vào nhau thành một thể thống nhất " Sự Thông Minh"
- Không thể đánh giá riêng lẽ bất kỳ một khía cạnh nào để nói về tính thông minh.
  - ➔ **THÔNG MINH CẦN TRI THỨC**

1.2 Đối tượng nghiên cứu của AI

- AI là lĩnh vực của Công nghệ thông tin, có chức năng nghiên cứu và tạo ra các chương trình mô phỏng hoạt động tư duy của con người.
- Trí tuệ nhân tạo nhằm tạo ra "Máy người"?

Mục tiêu

- Xây dựng lý thuyết về thông minh để giải thích các hoạt động thông minh
- Tìm hiểu cơ chế sự thông minh của con người
  - Cơ chế lưu trữ tri thức
  - Cơ chế khai thác tri thức
- Xây dựng cơ chế hiện thực sự thông minh
- Áp dụng các hiểu biết này vào các máy móc phục vụ con người.

1.2 Đối tượng nghiên cứu của AI(tiếp)

- AI là ngành nghiên cứu về cách hành xử thông minh (intellgent behaviour) bao gồm: thu thập, lưu trữ tri thức, suy luận, hoạt động và kỹ năng.
- Đối tượng nghiên cứu là các "hành xử thông minh" chứ không phải là "sự thông minh".
- Giải quyết bài toán bằng AI là tìm cách **biểu diễn tri thức**, tìm cách **vận dụng tri thức** để giải quyết vấn đề và tìm cách **bổ sung tri thức** bằng cách "phát hiện" tri thức từ những thông tin sẵn có (máy học)

1.3 Lịch sử phát triển của AI : Giai đoạn cổ điển

- Giai đoạn cổ điển (1950 – 1965)

  Có 2 kỹ thuật tìm kiếm cơ bản:

  – Kỹ thuật **generate and test :** chỉ tìm được 1 đáp án/ chưa chắc tối ưu.

  – Kỹ thuật **Exhaustive search** (vét cạn): Tìm tất cả các nghiệm, chọn lựa phương án tốt nhất.

(Bùng nổ tổ hợp m^n với m>=10)

Lịch sử phát triển của AI : Giai đoạn viễn vông

- Giai đoạn viễn vông (1965 – 1975)
  - Đây là giai đoạn phát triển với tham vọng làm cho máy hiểu được con người qua ngôn ngữ tự nhiên.
  - Các công trình nghiên cứu tập trung vào việc biểu diễn tri thức và phương thức giao tiếp giữa người và máy bằng ngôn ngữ tự nhiên.
  - Kết quả không mấy khả quan nhưng cũng tìm ra được các phương thức biểu diễn tri thức vẫn còn được dùng đến ngày nay tuy chưa thật tốt như:
    - Semantic Network (mạng ngữ nghĩa)
    - Conceptial graph (đồ thị khái niệm)
    - Frame (khung)
    - Script (kịch bản)

Vấp phải trở ngại về năng lực của máy tính

Lịch sử phát triển của AI :
Giai đoạn hiện đại

- Giai đoạn hiện đại (từ 1975)
  - Xc định lại mục tiêu mang tính thực tiễn hơn của AI:
    - Tìm ra **lời giải tốt nhất** trong khoảng thời gian chấp nhận được.
    - Không cầu toàn tìm ra **lời giải tối ưu**
  - Tinh thần HEURISTIC ra đời và được áp dụng mạnh mẽ để khắc phục bùng nổ tổ hợp.
  - Khẳng định vai trò của tri thức đồng thời xác định 2 trở ngại lớn là biểu diễn tri thức và bùng nổ tổ hợp.
  - Nêu cao vai trò của Heuristic nhưng cũng khẳng định tính khó khăn trong đánh giá heuristic.

Better than nothing ⇨ Phát triển ứng dụng mạnh mẽ: Hệ chuyên gia, Hệ chuẩn đoán,..

1.4 Các lĩnh vực ứng dụng

- Game Playing: Tìm kiếm / Heuristic
- Automatic reasoning & Theorem proving: Tìm kiếm / Heuristic
- Expert System: là hướng phát triển mạnh mẽ nhất và có giá trị ứng dụng cao nhất.
- Planning & Robotic: các hệ thống dự báo, tự động hóa
- Machine learning: Trang bị khả năng học tập để giải quyết vấn đề kho tri thức:
  - Supervised : Kiểm soát được tri thức học được. Không tìm ra cái mới.
  - UnSupervised:Tự học, không kiểm soát. Có thể tạo ra tri thức mới nhưng cũng nguy hiểm vì có thể học những điều không mong muốn.

1.4 Các lĩnh vực ứng dụng(tiếp)

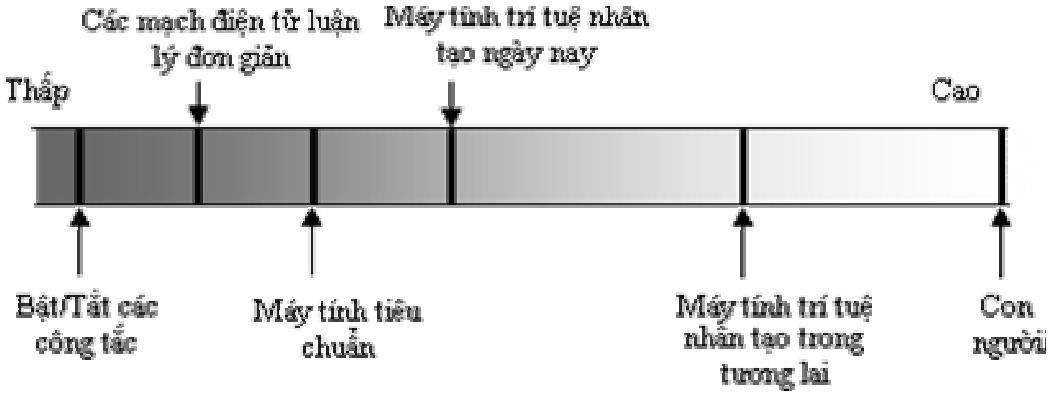
- Natural Language Understanding & Semantic modelling: Không được phát triển mạnh do mức độ phức tạp của bài toán cả về tri thức & khả năng suy luận.
- Modeling Human perfromance: Nghiên cứu cơ chế tổ chức trí tuệ của con người để áp dụng cho máy.
- Language and Environment for AI:Phát triển công cụ và môi trường để xây dựng các ứng dụng AI.
- Neural network / Parallel Distributed processing: giải quyết vấn đề năng lực tính toán và tốc độ tính toán bằng kỹ thuật song song và mô phỏng mạng thần kinh của con người.

Ứng dụng AI

- Mô hình ứng dụng AI hiện tại:

AI = Presentation & Search

- Mặc dù mục tiêu tối thượng của ngành TTNT là xây dựng một chiếc máy có năng lực tư duy tương tự như con người nhưng khả năng hiện tại của tất cả các sản phẩm TTNT vẫn còn rất khiêm tốn so với mục tiêu đã đề ra. Tuy vậy, ngành khoa học mới mẻ này vẫn đang tiến bộ mỗi ngày và đang tỏ ra ngày càng hữu dụng trong một số công việc đòi hỏi trí thông minh của con người. Hình ảnh sau sẽ giúp bạn hình dung được tình hình của ngành trí tuệ nhân tạo.



Các bài toán

– Xét các bài toán sau:

1. Đổi tiền (Vét cạn và Heuristic)
2. Tìm kiếm chiều rộng và sâu
3. Tic tac toe.
4. Đong dầu.
5. Bài toán TSP
6. 8 puzzle.
7. Cờ vua
8. Cờ tướng
9. Người nông dân qua sông.
10. Con thỏ và con cáo
11. Con khỉ và nải chuối

Chương 2: TÌM KIẾM TRÊN KHÔNG GIAN TRẠNG THÁI (State Space Search)

TS. Nguyễn Đình Thuân
Khoa Công nghệ Thông tin
Đại học Nha Trang
Email: thuanvinh122@gmail.com

Bài toán tìm kiếm

- Tìm kiếm cái gì?
- Biểu diễn và tìm kiếm là kỹ thuật phổ biến giải các bài toán trong lĩnh vực AI
- Các vấn đề khó khăn trong tìm kiếm với các bài toán AI
  - Đặc tả vấn đề phức tạp
  - Không gian tìm kiếm lớn
  - Đặc tính đối tượng tìm kiếm thay đổi
  - Đáp ứng thời gian thực
  - Meta knowledge và kết quả "tối ưu"
- Khó khăn về kỹ thuật

Cấu trúc chung của bài toán tìm kiếm

- Một cách chung nhất, nhiều vấn đề-bài toán phức tạp đều có dạng "tìm đường đi trong đồ thị" hay nói một cách hình thức hơn là *"xuất phát từ một đỉnh của một đồ thị, tìm đường đi hiệu quả nhất đến một đỉnh nào đó"*.
- Một phát biểu khác thường gặp của dạng bài toán này là:

  Cho trước hai trạng thái T_0 và T_G hãy xây dựng chuỗi trạng thái $T_0, T_1, T_2, ..., T_{n-1}, T_n = T_G$ sao cho :

$$\sum_1^n cost(T_{i-1}, T_i)$$

  thỏa mãn một điều kiện cho trước (thường là nhỏ nhất).

2.2 Giải thuật tổng quát

- Ký hiệu:

  s đỉnh xuất phát

  g: đỉnh đích

  n: đỉnh đang xét

  Γ(n): tập các đỉnh có thể đi trực tiếp từ đỉnh n

  Open: tập các đỉnh có thể xét ở bước kế tiếp

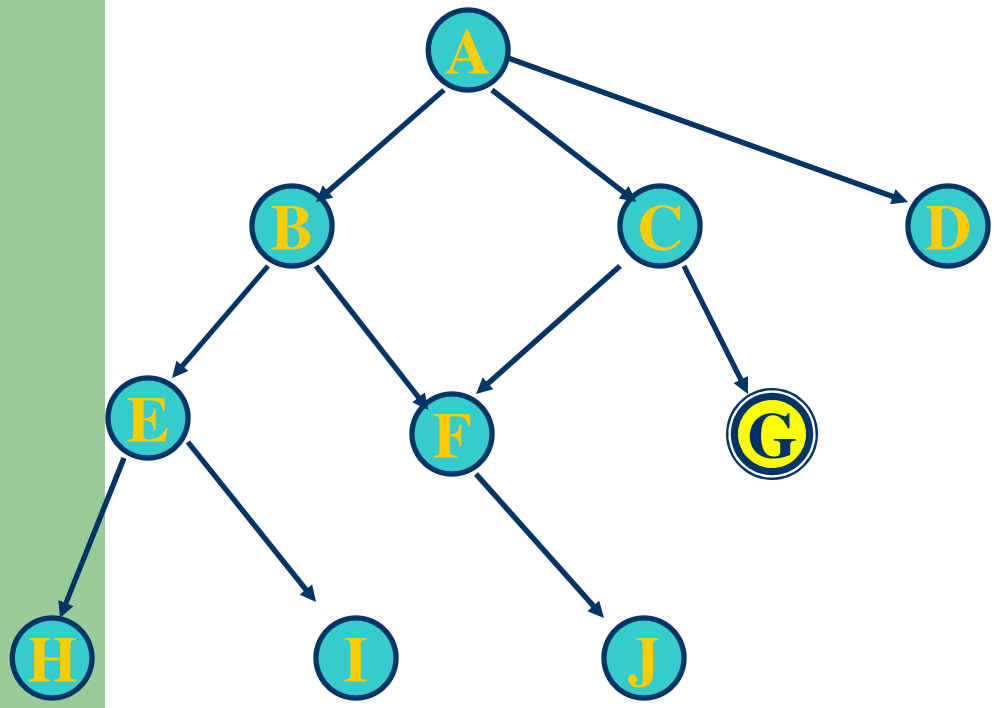
  Close: tập các đỉnh đã xét

2.2 Giải thuật tổng quát (tiếp)

```
Begin
    Open := {s};
    Close := ø;
    While (Open <> ø) do
        begin
            n:=Retrieve(Open);
            if (n=g) then Return True;
            Open := Open ∪ Γ(n); // (Γ(n) – Close)
            Close := Close ∪ {n};
        end;
    Return False;
End;
```

Ví dụ:

- Xét graph sau:

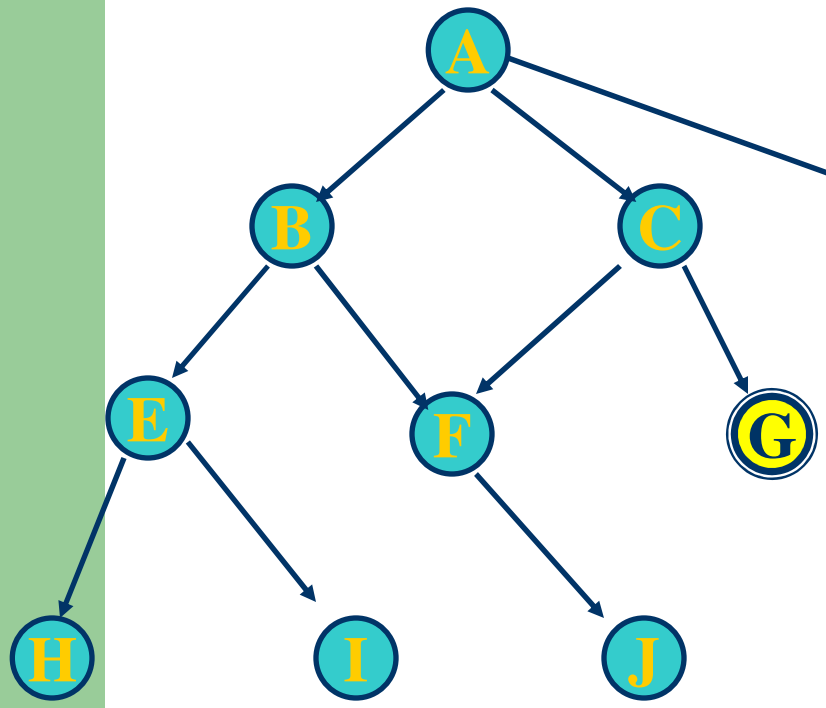


s = A là đỉnh bắt đầu

g= G là đỉnh đích

2.3 Breath First Search – Ví dụ

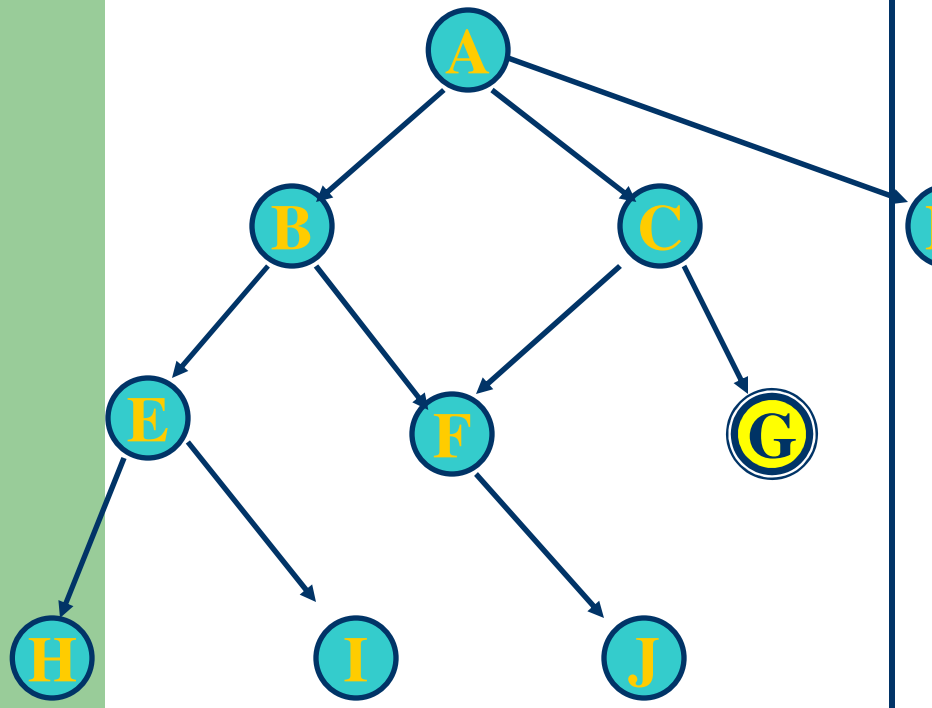
- Xét graph sau:



Lần lặp	n	Γ(n)	Open	Close
0			{A}	ø
1	A	{B, C, D}	{B, C, D}	{A}
2	B	{E, F}	{C, D, E, F}	{A, B}
3	C	{F, G}	{D, E, F, G}	{A, B, C}
4	D	ø	{E, F, G}	{A, B, C, D}
5	E	{H, I}	{F, G, H, I}	{A, B, C, D, E}
6	F	{J}	{G, H, I, J}	{A, B, C, D, E, F}
7	G			
		True		

2.3 Breath First Search – Ví dụ 1

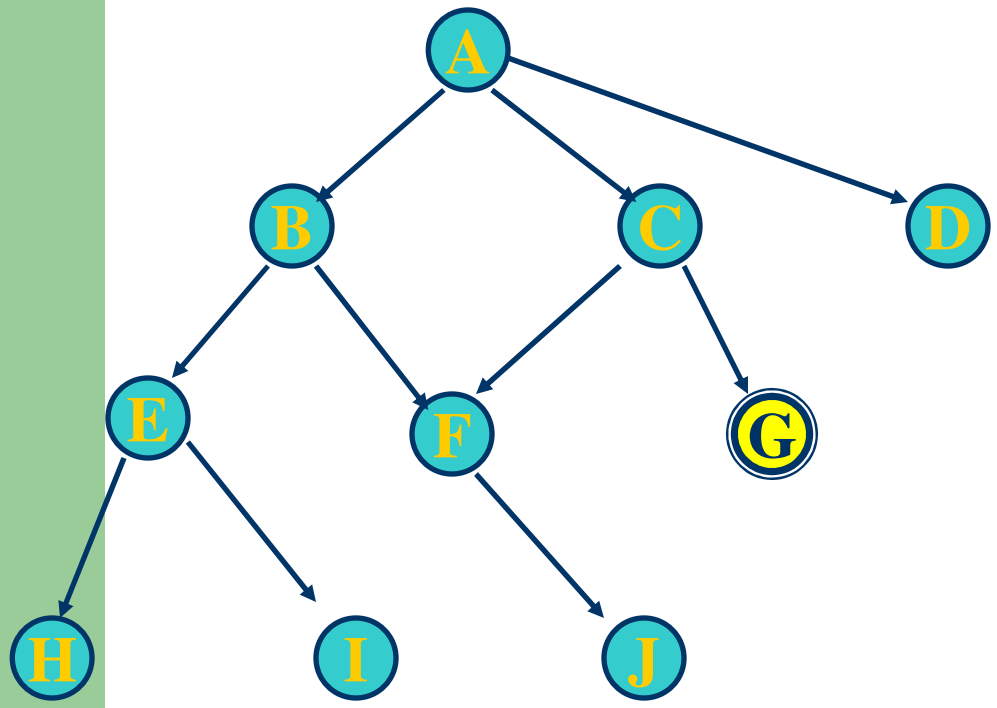
- Xét graph sau:A->U



Lần lặp	n	Γ(n)	Open	Close
0			{A}	ø
1	A	{B, C, D}	{B,C,D}	{A}
2	B	{E, F}	{C,D, E,F}	{A, B}
3	C	{F, G}	{D,E, F,G}	{A, B, C}
4	D	ø	{E, F, G}	{A, B, C, D}
5	E	{H, I}	{F, G, H, I}	{A, B, C, D, E}
6	F	{J}	{G, H, I, J}	{A, B, C, D, E, F}
7	G	ø	{H, I, J}	{A, B, C, D, E, F,G}
8	H	Ø	{I, J}	{A,B,C, D, E, F,G,H}
9	I	ø	{J}	{A,B,C, D, E, F,G,H,I}
10	J	Ø	Ø	{A,B,C, D, E, F,G,H,I,J}
		FALSE		

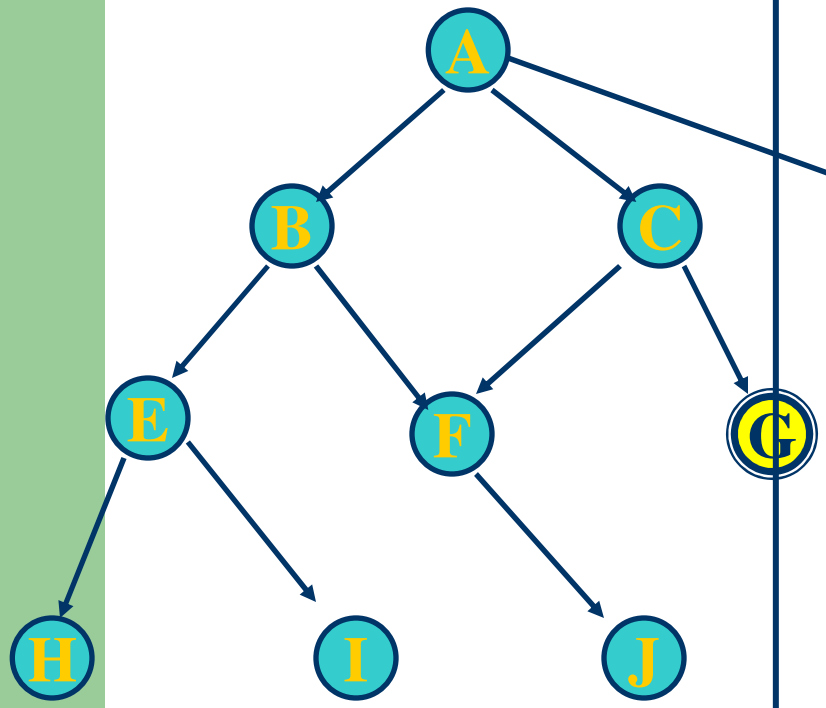
Ví dụ:

- Xét graph sau:



2.4 Depth First Search – Ví dụ

- Xét graph sau:



Lần lặp	n	Γ(n)	Open	Close
0			{A}	ø
1	A	{B, C, D}	{B, C, D}	{A}
2	B	{E, F}	{E, F, C, D}	{A, B}
3	E	{H, I}	{H, I, F, C, D}	{A, B, E}
4	H	Ø	{I, F, C, D}	{A, B, E, H}
5	I	Ø	{F, C, D}	{A, B, E, H, I}
6	F	{J}	{J, C, D}	{A, B, E, H, I, F}
7	J	Ø	{C, D}	{A, B, E, H, I, F,J}
8	C	{F, G}	{G, D}	{A,B,E,H,I, F,J,C}
9	G	True		

Breath First vs Depth First

- Breath First: open được tổ chức dạng FIFO
- Depth First: open được tổ chức dạng LIFO
- Hiệu quả
  - Breath First luôn tìm ra nghiệm có số cung nhỏ nhất
  - Depth First "thường" cho kết quả nhanh hơn.
- Kết quả
  - BFS, DFS chắc chắn tìm ra kết quả nếu có.
- Bùng nổ tổ hợp là khó khăn lớn nhất cho các giải thuật này.

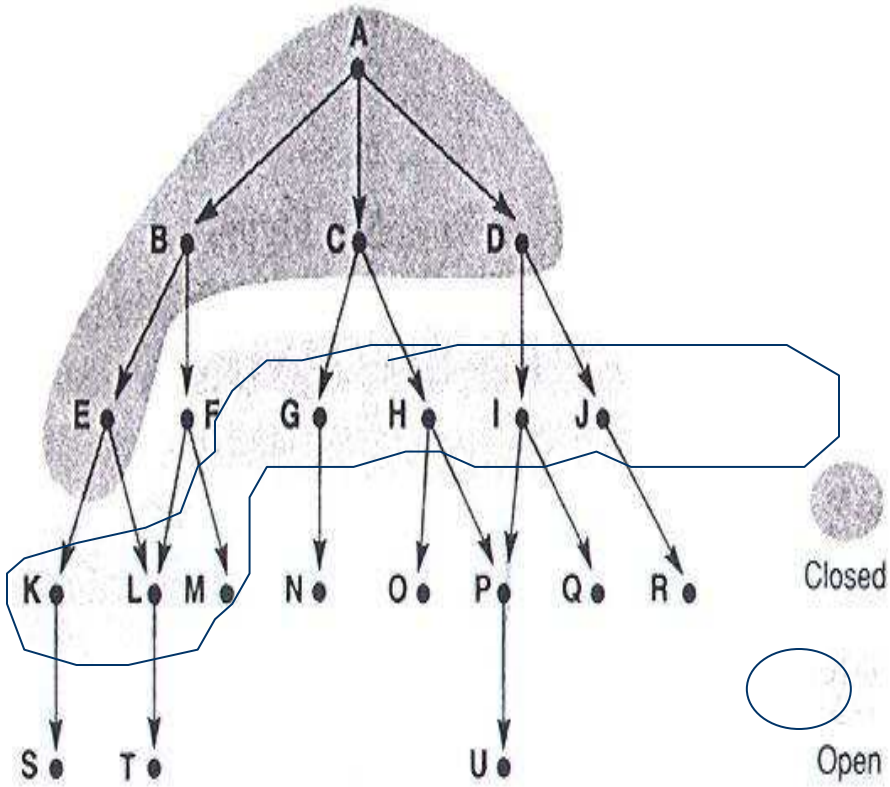
Giải Pháp cho bùng nổ tổ hợp??

Tìm Kiếm Rộng

1. Open = [A]; closed = []
2. Open = [B,C,D];
   closed = [A]
2. Open = [C,D,E,F];
   closed = [B,A]
3. Open = [D,E,F,G,H]; closed = [C,B,A]
4. Open = [E,F,G,H,I,J]; closed = [D,C,B,A]
5. Open = [F,G,H,I,J,K,L];closed = [E,D,C,B,A]
6. Open = [G,H,I,J,K,L,M];(vì L đã có trong open);

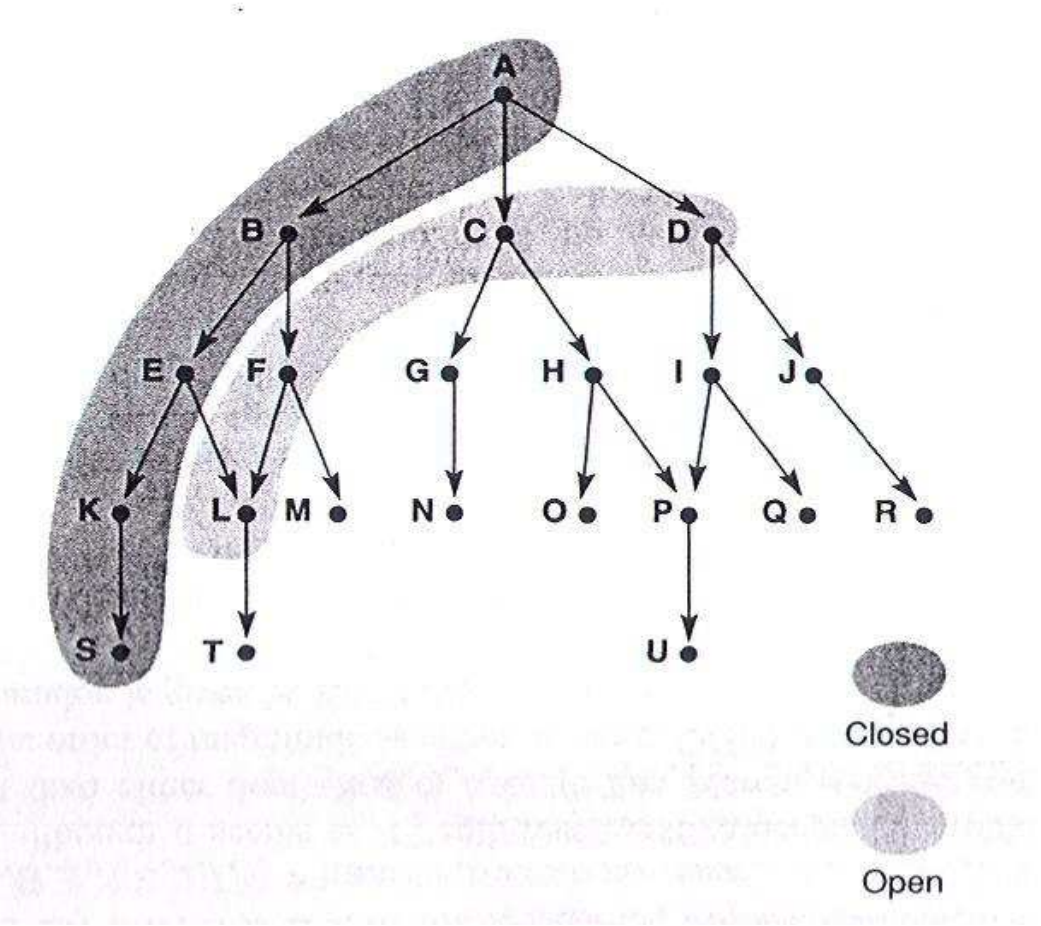
closed = [F,E,D,C,B,A]

…



Tìm kiếm Sâu

1. Open = [A]; closed = []
2. Open = [B,C,D]; closed = [A]
3. Open = [E,F,C,D];closed = [B,A]
4. Open = [K,L,F,C,D];
   closed = [E,B,A]
5. Open = [S,L,F,C,D];
   closed = [K,E,B,A]
6. Open = [L,F,C,D];
   closed = [S,K,E,B,A]
7. Open = [T,F,C,D];
   closed = [L,S,K,E,B,A]
8. Open = [F,C,D];
   closed = [T,L,S,K,E,B,A]
9. …



Depth first search có giới hạn

- Depth first search có khả năng lặp vô tận do các trạng thái con sinh ra liên tục. Độ sâu tăng vô tận.
- Khắc phục bằng cách giới hạn độ sâu của giải thuật.
- Sâu bao nhiêu thì vừa?
- Chiến lược giới hạn:
  - Cố định một độ sâu MAX, như các danh thủ chơi cờ tính trước được số nước nhất định
  - Theo cấu hình resource của máy tính
  - Meta knowledge trong việc định giới hạn độ sâu.
- Giới hạn độ sâu => co hẹp không gian trạng thái => có thể mất nghiệm.

Chương 3: HEURISTIC SEARCH

TS. Nguyễn Đình Thuân
Khoa Công nghệ Thông tin
Đại học Nha Trang
Email: thuanvinh122@gmail.com

3.1 Giới thiệu về Heuristic

- **Heuristic là gì?**
  - Heuristic là những tri thức được rút tỉa từ những kinh nghiệm, "trực giác" của con người.
  - Heuristic có thể là những tri thức "đúng" hay "sai".
  - Heuristic là những meta knowledge và "thường đúng".
- **Heuristic dùng để làm gì?**
  - Trong những bài toán tìm kiếm trên không gian trạng thái, có 2 trường hợp cần đến heuristic:
    - Vấn đề có thể không có nghiệm chính xác do các mệnh đề không phát biểu chặt chẽ hay thiếu dữ liệu để khẳng định kết quả.
    - Vấn đề có nghiệm chính xác nhưng phí tổn tính toán để tìm ra nghiệm là quá lớn (hệ quả của bùng nổ tổ hợp)
  - **Heuristic giúp tìm kiếm đạt kết quả với chi phí thấp hơn**

Heuristic (tiếp)

- Thuật giải Heuristic là một sự mở rộng khái niệm thuật toán. Nó thể hiện cách giải bài toán với các đặc tính sau:
  - Thường tìm được lời giải tốt (nhưng không chắc là lời giải tốt nhất)
  - Giải bài toán theo thuật giải Heuristic thường dễ dàng và nhanh chóng đưa ra kết quả hơn so với giải thuật tối ưu, vì vậy chi phí thấp hơn.
  - Thuật giải Heuristic thường thể hiện khá tự nhiên, gần gũi với cách suy nghĩ và hành động của con người.

Heuristic (tiếp)

Có nhiều phương pháp để xây dựng một thuật giải Heuristic, trong đó người ta thường dựa vào một số nguyên lý cơ bản như sau:

- Nguyên lý vét cạn thông minh: Trong một bài toán tìm kiếm nào đó, khi không gian tìm kiếm lớn, ta thường tìm cách giới hạn lại không gian tìm kiếm hoặc thực hiện một kiểu dò tìm đặc biệt dựa vào đặc thù của bài toán để nhanh chóng tìm ra mục tiêu.
- Nguyên lý tham lam (Greedy): Lấy tiêu chuẩn tối ưu (trên phạm vi toàn cục) của bài toán để làm tiêu chuẩn chọn lựa hành động cho phạm vi cục bộ của từng bước (hay từng giai đoạn) trong quá trình tìm kiếm lời giải.
- Nguyên lý thứ tự: Thực hiện hành động dựa trên một cấu trúc thứ tự hợp lý của không gian khảo sát nhằm nhanh chóng đạt được một lời giải tốt.
- Hàm Heuristic: Trong việc xây dựng các thuật giải Heuristic, người ta thường dùng các hàm Heuristic. Đó là các hàm đánh già thô, giá trị của hàm phụ thuộc vào trạng thái hiện tại của bài toán tại mỗi bước giải. Nhờ giá trị này, ta có thể chọn được cách hành động tương đối hợp lý trong từng bước của thuật giải.

Heuristic Greedy

- Bài toán đổi tiền: Đổi số tiền n thành các loại tiền cho trước sao cho số tờ là ít nhất
- Bài toán hành trình ngắn nhất (TSP): Hãy tìm một hành trình cho một người giao hàng đi qua n điểm khác nhau, mỗi điểm đi qua một lần và trở về điểm xuất phát sao cho tổng chiều dài đoạn đường cần đi là ngắn nhất. Giả sử rằng có con đường nối trực tiếp từ giữa hai điểm bất kỳ.
  - Vét cạn: (n-1)! (Với n lớn ???)
  - Greedy 1: Mỗi bước chọn i →j sao cho j gần i nhất trong những đỉnh nối với i còn lại
  - Greedy 2: Mỗi bước chọn i →j sao cho i gần j nhất trong những đỉnh nối với j còn lại

Ví dụ: TSP với n=8

	1	2	3	4	5	6	7	8
1	0	730	640	840	800	430	380	1010
2	730	0	710	1040	500	300	540	470
3	640	710	0	1420	1050	600	920	1160
4	840	1040	1420	0	740	950	570	900
5	800	500	1050	740	0	520	460	200
6	430	300	600	950	520	0	390	690
7	380	540	920	570	460	390	0	660
8	1010	470	1160	900	200	690	660	0

Ví dụ: TSP với n=8

*Với Greedy 1:

$1 \rightarrow 7 \rightarrow 6 \rightarrow 2 \rightarrow 8 \rightarrow 5 \rightarrow 4 \rightarrow 3 \rightarrow 1$

Tổng chi phí: 4540

*Với Greedy 2:

$1 \rightarrow 7 \rightarrow 4 \rightarrow 5 \rightarrow 8 \rightarrow 2 \rightarrow 6 \rightarrow 3 \rightarrow 1$

Tổng chi phí: 3900

Bài toán 3: Bài toán tô màu bản đồ

Heuristic (tt)

- Heuristic dùng như thế nào trong tìm kiếm?
  - Tìm kiếm trên không gian trạng thái theo chiều nào? Sâu hay rộng?
  - Tìm theo Heuristic : Heuristic định hướng quá trình tìm kiếm theo hướng mà "nó" cho rằng khả năng đạt tới nghiệm là cao nhất. Không "sâu" cũng không "rộng"
- Kết quả của tìm kiếm với Heuristic
  - Việc tìm kiếm theo định hướng của heuristic có kết quả tốt hay xấu tùy theo heuristic "đúng" hay "sai".
  - Heuristic có khả năng bỏ sót nghiệm
  - Heuristic càng tốt càng dẫn đến kết quả nhanh và tốt.
  - **Làm sao tìm được Heuristic tốt???**

3.2 Tìm kiếm tối ưu (Best First Search)

OPEN : tập chứa các trạng thái đã được sinh ra nhưng chưa được xét đến (vì ta đã chọn một trạng thái khác). Thực ra, OPEN là một loại hàng đợi ưu tiên (priority queue) mà trong đó, phần tử có độ ưu tiên cao nhất là phần tử *tốt nhất*.
CLOSE : tập chứa các trạng thái đã được xét đến. Chúng ta cần lưu trữ những trạng thái này trong bộ nhớ để đề phòng trường hợp khi một trạng thái mới được tạo ra lại trùng với một trạng thái mà ta đã xét đến trước đó.

Thuật giải BEST-FIRST SEARCH

1. Đặt OPEN chứa trạng thái khởi đầu.
2. Cho đến khi tìm được trạng thái đích hoặc không còn nút nào trong OPEN, thực hiện :
    2.a. Chọn trạng thái tốt nhất (Tmax) trong OPEN (và xóa Tmax khỏi OPEN)
    2.b. Nếu Tmax là trạng thái kết thúc thì thoát.
    2.c. Ngược lại, tạo ra các trạng thái kế tiếp T_k có thể có từ trạng thái Tmax.
    Đối với mỗi trạng thái kế tiếp T_k thực hiện : Tính $f(T_k)$; Thêm T_k vào OPEN

3.2 Tìm kiếm tối ưu (tiếp)

Thuật giải BEST-FIRST SEARCH

```
Begin
    open:={s};
    While (open<> ø) do
        begin
            n:= Retrieve(Open) //Chọn trạng thái tốt nhất từ Open.
            if (n=g) then return True
            else begin
                Tạo Γ(n)
                 for mỗi nút con m của Γ(n) do
                        Gán giá trị chi phí cho m
                                Open:=Open∪{m};
        end;
    Return False;
End;
```

```
Begin
        Open := {s};
        Close := ø;
        While (Open <> ø) do
                begin
                        n:=Retrieve(Open);
                        if (n=g) then Return True;
                        Open := Open ∪ Γ(n); // (Γ(n) – Close)
                        Close := Close ∪ {n};
                end;
        Return False;
End;
```

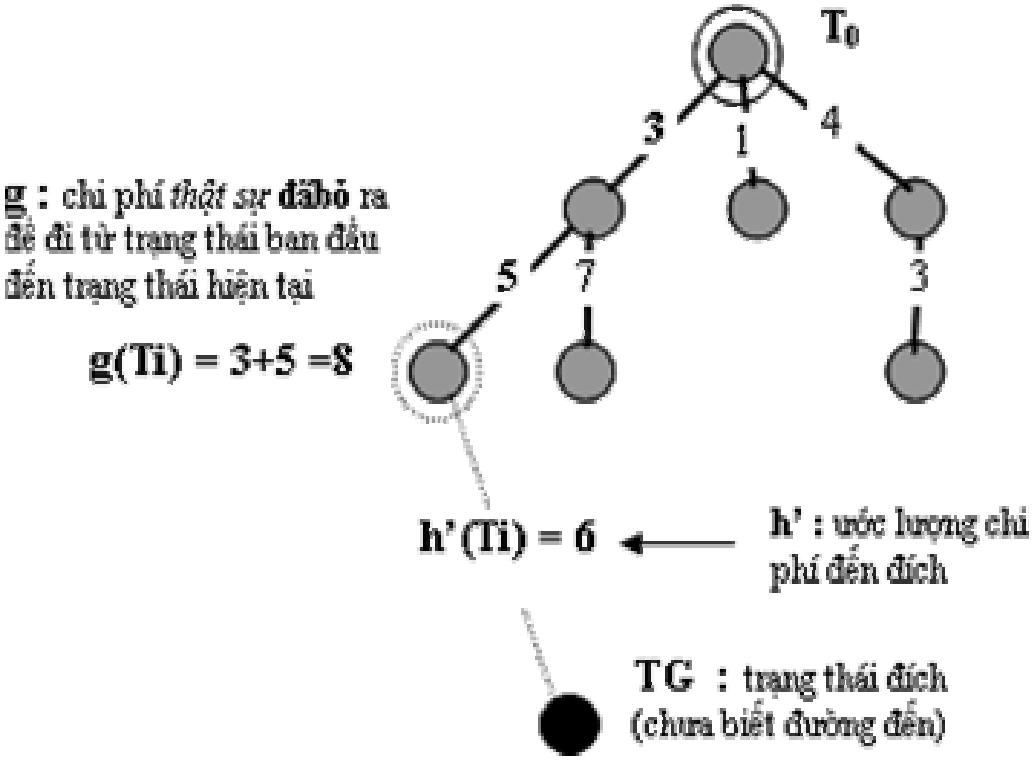
3.2 Tìm kiếm tối ưu (tiếp)

- BFS khá đơn giản. Tuy vậy, trên thực tế, cũng như tìm kiếm chiều sâu và chiều rộng, hiếm khi ta dùng BFS một cách trực tiếp. Thông thường, người ta thường dùng các phiên bản của BFS là AT, AKT và A*

Thông tin về quá khứ và tương lai

- Thông thường, trong các phương án tìm kiếm theo kiểu BFS, độ tốt **f** của một trạng thái được tính dựa theo 2 hai giá trị mà ta gọi là là **g** và **h'**. **h'** chúng ta đã biết, đó là một ước lượng về chi phí từ trạng thái hiện hành cho đến trạng thái đích (thông tin tương lai). Còn **g** là "chiều dài quãng đường" *đã* đi từ trạng thái ban đầu cho đến trạng thái hiện tại (thông tin quá khứ). Lưu ý rằng **g** là chi phí thực sự (không phải chi phí ước lượng).

3.3 Thuật giải A^T



Phân biệt khái niệm g và h'

3.3 Thuật giải A^T

Thuật giải A^T là một phương pháp tìm kiếm theo kiểu BFS với độ tốt của nút là giá trị hàm g – tổng chiều dài con đường đã đi từ trạng thái bắt đầu đến trạng thái hiện tại.

```
Begin
   open:={s};
   While (open<> ø) do
      begin
         n:= Retrieve(Open)      //Chọn n sao cho g(n) →nhỏ nhất từ Open.
         if (n=g) then return True
         else begin
            Tạo Γ(n)
             for mỗi nút con m của Γ(n) do
                     if (m∉ Open) then
                       Begin
                          g(m):=g(n)+Cost(n,m)
                         Open:=Open∪{m};
                       end
                     else So sánh g(m) va g_New (m) và cập nhật
      end;
   Return False;
```

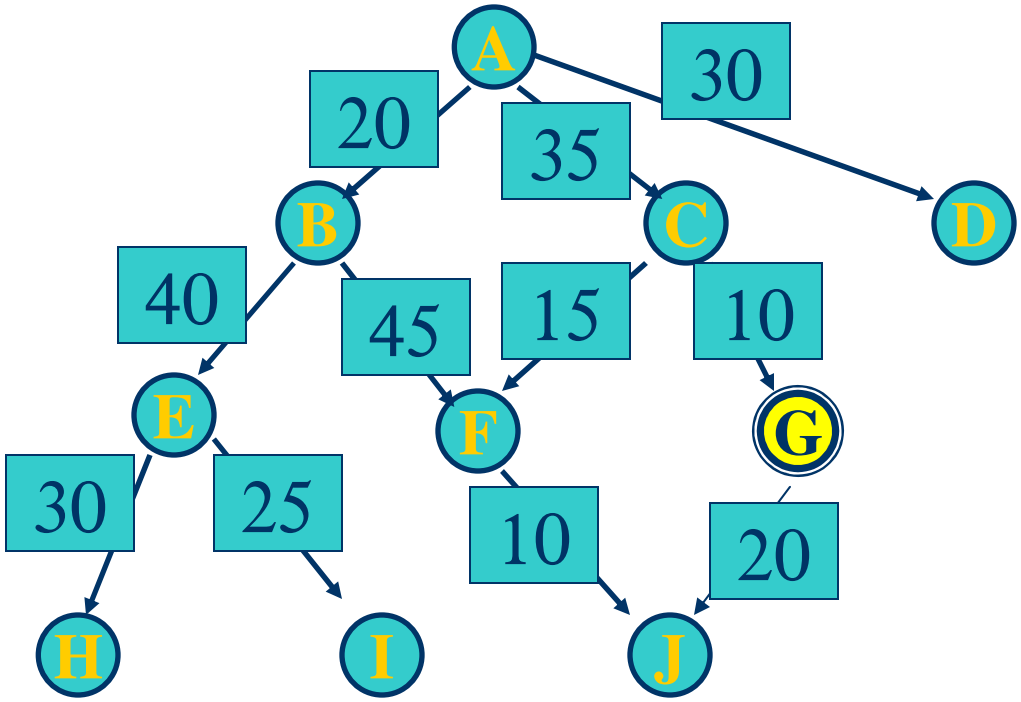
3.3 Thuật giải CMS (Cost Minimazation Search)

Thuật giải CMS là một phương pháp tìm kiếm theo kiểu BFS với độ tốt của nút là giá trị hàm g và bổ sung tập Close: tập đỉnh đã xét).

```
Begin
    open:={s}; close := ø
    While (open<> ø) do
        begin
            n:= Retrieve(Open)      //Chọn n sao cho g(n) →nhỏ nhất từ Open.
            if (n=g) then return True
            else begin
                Tạo Γ(n)
                for mỗi nút con m của Γ(n) do
                        if (m∉ Open) and (m∉ Close) then
                          Begin
                            g(m):=g(n)+Cost(n,m)
                           Open:=Open∪{m};
                          end
                        else So sánh g(m) va g_New (m) và cập nhật
            close = close ∪{n}
        end;
    Return False;
End;
```

Ví dụ:

- Xét graph sau:



s = A là đỉnh bắt đầu

g= J là đỉnh đích

Ví dụ:

- :

Trước	Sau
*	A
A	B
A	C
A	D
B	E
C	F
C	G
F	J

s = A là đỉnh bắt đầu

g= J là đỉnh đích

Lần lặp	n	Γ(n)	Open
0			{(A,0)}
1	A	{B,C,D}	{(B,20), (C,35), (D,30)}
	B	{E,F}	(C,35), (D,30),(E,60),(F,65)
	D	ø	(C,35),(E,60),(F,65)
	C	{F,G}	(E,60),(F,50),(G,45)
	G	{J}	(E,60),(F,50),(J,65)
	F	(J}	(E,60),(J,60)
	J		

3.4 Thuật giải A^{KT}

(Algorithm for Knowlegeable Tree Search)

Thuật giải A^{KT} mở rộng A^{T} bằng cách sử dụng thêm thông tin ước lượng h'. Độ tốt của một trạng thái f là tổng của hai hàm g và h'.

```
Begin
    open:={s};
    While (open<> ø) do
        begin
            n:= Retrieve(Open)      //Chọn n sao cho f(n) →nhỏ nhất từ Open.
            if (n=g) then return True
            else begin
                Tạo Γ(n)
                 for mỗi nút con m của Γ(n) do
                        Begin
                            g(m):=g(n)+Cost(n,m)
                            f(m):= g(m)+h'(m);
                           Open:=Open∪{m};
                        end;
        end;
    Return False;
End;
```

Thuật giải A*

A* là một phiên bản đặc biệt của A^{KT} áp dụng cho trường hợp đồ thị. Thuật giải A* có sử dụng thêm tập hợp CLOSE để lưu trữ những trường hợp đã được xét đến. A* mở rộng A^{KT} bằng cách bổ sung cách giải quyết trường hợp khi "mở" một nút mà nút này đã có sẵn trong OPEN hoặc CLOSE.

3.5 Thuật giải A* (tiếp)

```
Begin
 open:={s}; close:=ø;
While (open<> ø) do
 begin
    n:= Retrieve(Open) //sao cho f(n) min.
    if (n=g) then return path từ s đến g
    else begin
       Tạo Γ(n)
       for mỗi nút con m của Γ(n) do
          case m of
             m ∉ Open và m ∉ Close:
                begin
                   Gán giá trị heuristic cho m
                   Open:=Open∪{m};
                end;
             m ∈ Open:
                if đến được m bằng một path ngắn hơn
                then Cập nhật lại m trong Open.
             m ∈ Close
                if đến được m bằng một path ngắn hơn
                then begin
                        Close:=Close-{m}
                        Open:=Open∪{m}
                end;
          end; /*end case*/
       Close:=Close∪{n}
       end; / while/
       return false;
End;
```

Hàm lượng giá Heuristic

- Hàm lượng giá Heuristic là hàm ước lượng phí tổn để đi từ trạng thái hiện tại đến trạng thái goal.
- Cơ sở để xác định hàm lượng giá là dựa vào tri thức/kinh nghiệm thu thập được.
- Hàm lượng giá cho kết quả đúng (gần thực thế) hay sai (xa giá trị thực) sẽ dẫn đến kết quả tìm được tốt hay xấu.
- Không có chuẩn mực cho việc đánh giá một hàm lượng giá Heuristic. Lý do:
  - Không có cấu trúc chung cho hàm lượng giá
  - Tính đúng/sai thay đổi liên tục theo từng vấn đề cụ thể
  - Tính đúng/sai thay đổi theo từng tình huống cụ thể trong một vấn đề
- **Có thể dùng nhiều hàm lượng giá khác nhau theo tình huống → cần hàm lượng giá về các hàm lượng giá.**

Trò đố 8 ô hay 15 ô

Trạng thái ban đầu Trạng thái đích

- Trò đố 15 ô

11	14	4	7
10	6		5
1	2	13	15
9	12	8	3

1	2	3	4
12	13	14	5
11		15	6
10	9	8	7

- Trò đố 8 ô

	2	8
3	5	7
6	2	1

1	2	3
8		4
7	6	5

- Cần biểu diễn KGTT cho bài toán này như thế nào?

Thuật giải A* – Ví dụ

- Xét bài toán 8 pussle với goal là:

1	2	3
8		4
7	6	5

Heuristic 1: Tổng số miếng sai vị trí

Heuristic 2: Tổng khoảng cách sai vị trí của từng miếng.

2 8 3 1 6 4 7 5	5	6	
2 8 3 1 4 7 6 5	3	4	
2 8 3 1 6 4 7 5	5	6	

Việc chọn lựa hàm Heuristic là khó khăn và có ý nghĩa quyết định đối với tốc độ của giải thuật

Hàm lượng giá Heuristic – Cấu trúc

- Xét lại hoạt động của giải thuật Best First Search:
  - Khi có 2 nút cùng có giá trị kỳ vọng đạt đến mục tiêu bằng nhau thì nút có path từ nút bắt đầu đến nút đó ngắn hơn sẽ được chọn trước như vậy nút này có giá trị Heuristic tốt hơn.
  - Hay nói cách khác hàm lượng giá Heuristic cho nút gần start hơn là tốt hơn nếu kỳ vọng đến goal là bằng nhau.
  - Vậy chọn nút nào nếu kỳ vọng của 2 nút khác nhau? Nút kỳ vọng tốt hơn nhưng xa start hay nút kỳ vọng xấu hơn nhưng gần root

Hàm lượng giá bao gồm cả 2 và có cấu trúc:

F(n) := G(n) + H(n)

G(n): phí tổn thực từ root đến n

H(n): phí tổn ước lượng heuristic từ n đến goal.

Thuật giải A* – Ví dụ

- Xét ví dụ là bài toán 8 puzzle với:

2	8	3
1	6	4
7		5

Bắt đầu

1	2	3
8		4
7	6	5

Đích

Hàm lượng giá: F(n) = G(n) + H(n)

Với G(n): số lần chuyển vị trí đã thực hiện

H(n): Số miếng nằm sai vị trí

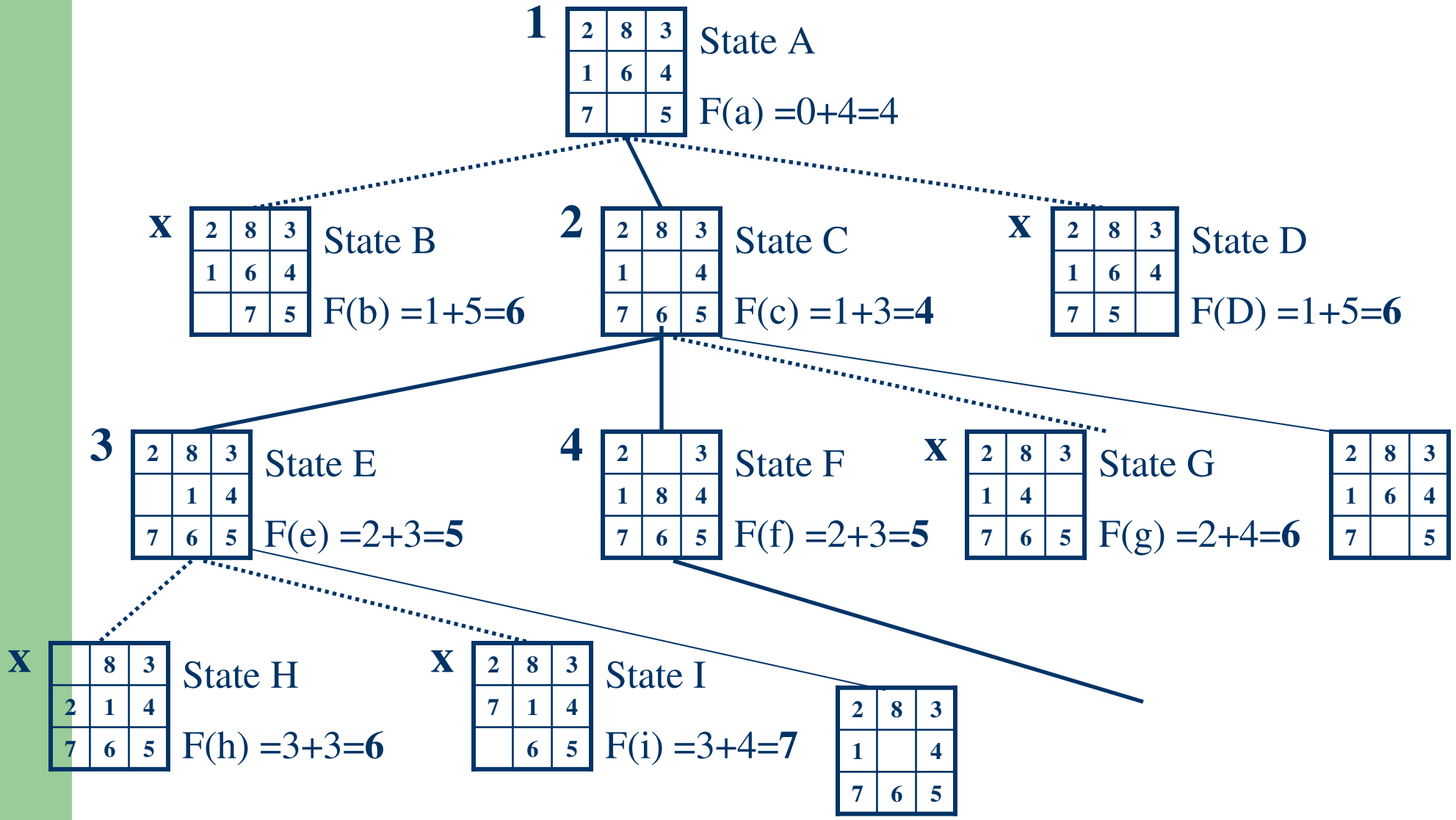
Nút X có giá trị heuristic tốt hơn nút Y nếu **F(x) < F(y)**.

Ta có hoạt động của giải thuật Best First search trên như hình sau:

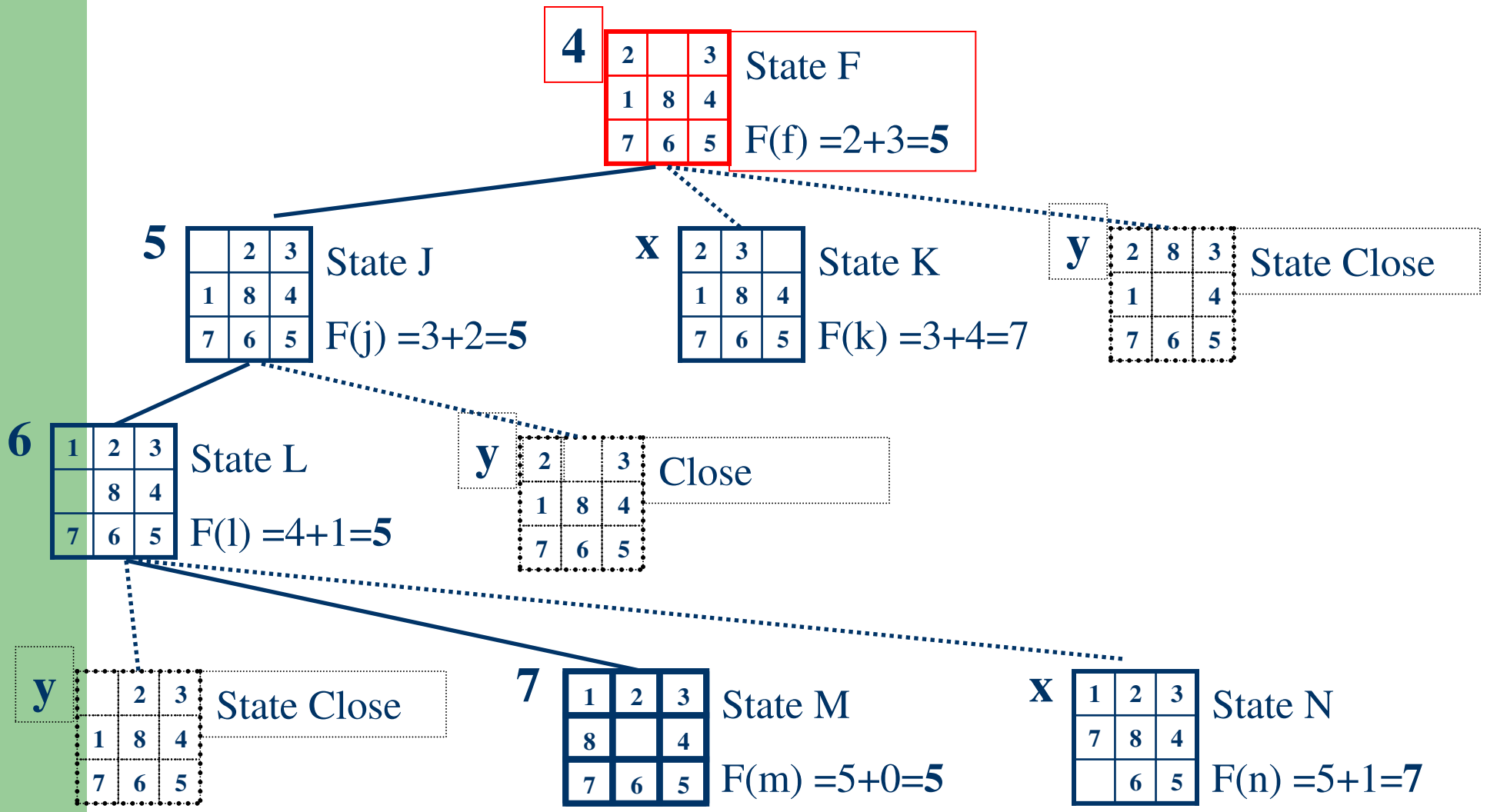
3.5 Thuật giải A* (tiếp)

```
Begin
 open:={s}; close:=ø;
While (open<> ø) do
 begin
    n:= Retrieve(Open) //sao cho f(n) min.
    if (n=g) then return path từ s đến g
    else begin
       Tạo Γ(n)
        for mỗi nút con m của Γ(n) do
          case m of
              m ∉ Open và m ∉ Close:
                 begin
                     Gán giá trị heuristic cho m
                     Open:=Open∪{m};
                 end;
              m ∈ Open:
                 if đến được m bằng một path ngắn hơn
                 then Cập nhật lại m trong Open.
              m ∈ Close
                 if đến được m bằng một path ngắn hơn
                 then begin
                             Close:=Close-{m}
                             Open:=Open∪{m}
                      end;
          end; /*end case*/
      Close:=Close∪{n}
     end; / while/
     return false;
End;
```

Ví dụ

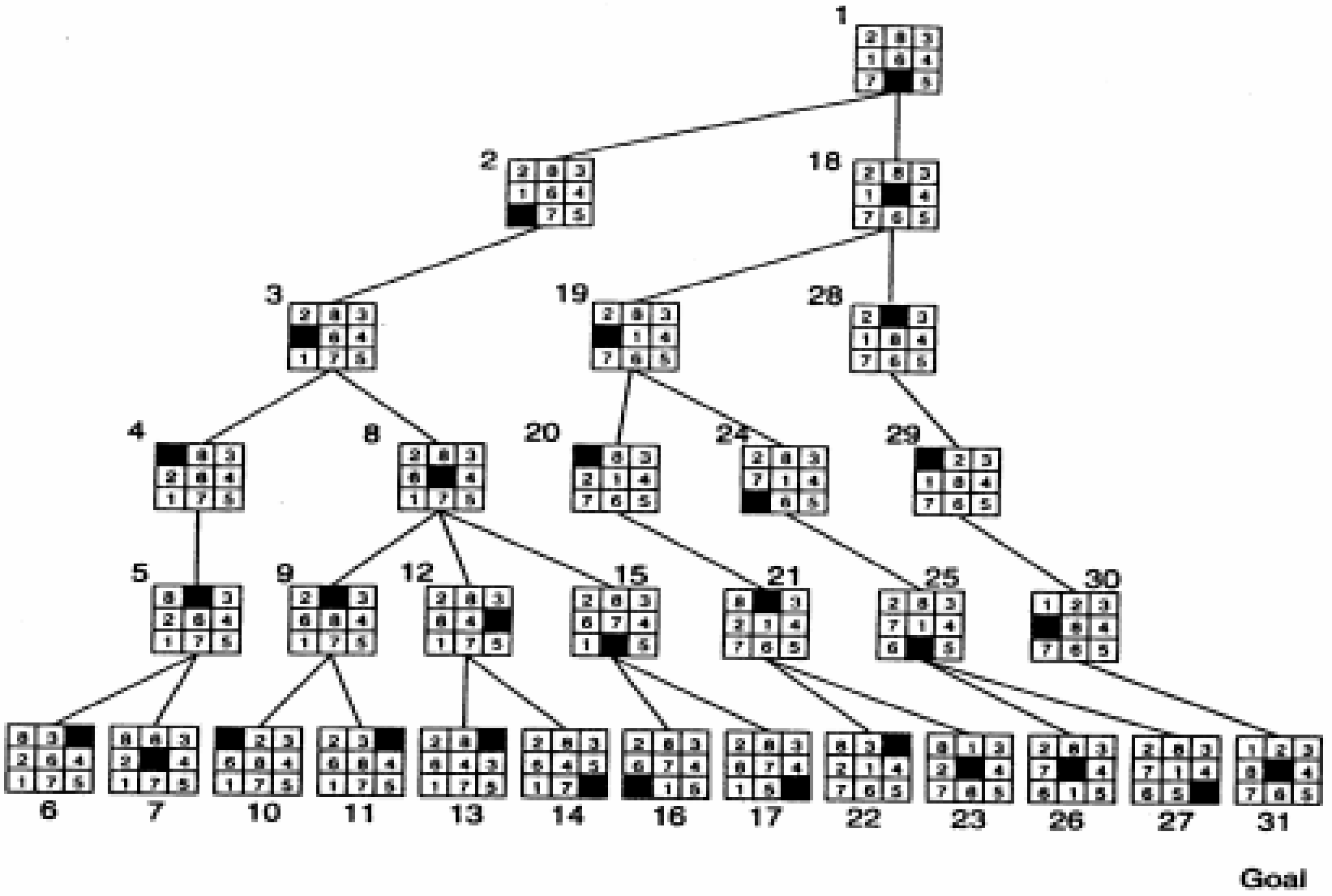

1
2 8 3
1 6 4
7 5
State A
F(a) =0+4=4
x
2 8 3
1 6 4
7 5
State B
F(b) =1+5=6
2
2 8 3
1 4
7 6 5
State C
F(c) =1+3=4
x
2 8 3
1 6 4
7 5
State D
F(D) =1+5=6
3
2 8 3
1 4
7 6 5
State E
F(e) =2+3=5
4
2 3
1 8 4
7 6 5
State F
F(f) =2+3=5
x
2 8 3
1 4
7 6 5
State G
F(g) =2+4=6
2 8 3
1 6 4
7 5
x
8 3
2 1 4
7 6 5
State H
F(h) =3+3=6
x
2 8 3
7 1 4
6 5
State I
F(i) =3+4=7
2 8 3
1 4
7 6 5

Ví dụ


4
2 3
1 8 4
7 6 5
State F
F(f) =2+3=5
5
2 3
1 8 4
7 6 5
State J
F(j) =3+2=5
x
2 3
1 8 4
7 6 5
State K
F(k) =3+4=7
y
2 8 3
1 4
7 6 5
State Close
6
1 2 3
8 4
7 6 5
State L
F(l) =4+1=5
y
2 3
1 8 4
7 6 5
Close
y
2 3
1 8 4
7 6 5
State Close
7
1 2 3
8 4
7 6 5
State M
F(m) =5+0=5
x
1 2 3
7 8 4
6 5
State N
F(n) =5+1=7

Trò chơi ô đố 8-puzzle

The 8-puzzle searched by a production system with loop detection and depth bound 5



Hoạt động theo giải thuật A*

Lần	n	Open	Close
0		{A4}	{}
1	A4	{C4,B6,D6}	{A4}
2	C4	{E5,F5,G6,B6,D6}	{A4,C4}
3	E5	{F5,H6,G6,B6,D6,I7}	{A4,C4,E5}
4	F5	{J5,H6,G6,B6,D6,K7,I7}	{A4,C4,E5,F5}
5	J5	{L5,H6,G6,B6,D6,K7,I7}	{A4,C4,E5,F5,J5}
6	l5	{M5,H6,G6,B6,D6,K7,I7,N7}	{A4,C4,E5,F5,J5,L5}
7	**m5**		

Đánh giá giải thuật Heuristic

- **Admissibility** – Tính chấp nhận
  - Một giải thuật Best first search với hàm đánh giá
  - F(n) = G(n) + H(n) với
    - N : Trạng thái bất kỳ
    - G(n) : Phí tổn đi từ nút bắt đầu đến nút n
    - H(n) : Phí tổn ước lượng heuristic đi từ nút **n** đến **goal**
  - Được gọi là giải thuật **A**
- Một giải thuật tìm kiếm được xem là admissible nếu đối với một đồ thị bất kỳ nó luôn dừng ở path nghiệm tốt nhất (nếu có).
- **Giải thuật A***: Là giải thuật A với hàm heuristic H(n)luôn luôn ≤ giá trị thực đi từ n đến goal.
- **Giải thuật A* là admissible**

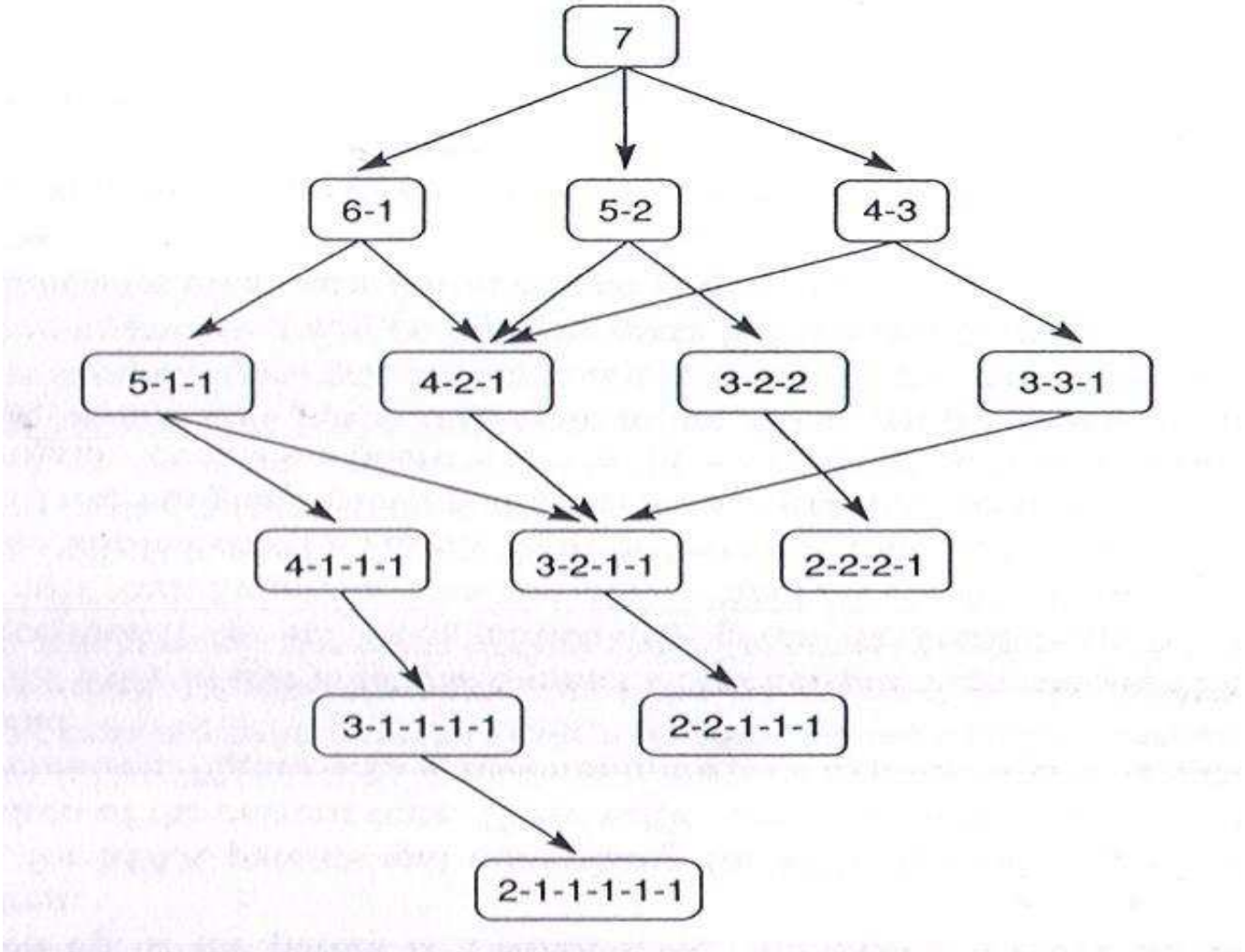
Đánh giá giải thuật Heuristic

- **Monotonicity** – Đơn điệu
  - Một hàm heuristic H(n) được gọi là monotone (đơn điệu) nếu:
  - $\forall n_i, n_j$: n_j là nút con cháu của ni ta có
    $H(n_i)-H(n_j) \leq$ phí tổn thật đi từ ni đến nj
  - Đánh giá heuristic của đích là 0 : H(goal) = 0.
- Giải thuật A có hàm H(n) monotone là giải thuật A* và Admissible
- **Informedness**
- Xét 2 hàm heuristic $H_1(n)$ và $H_2(n)$ nếu ta có $H_1(n) \leq H_2(n)$ với mọi trạng thái n thì $H_2(n)$ được cho là informed hơn $H_1(n)$.

Heuristic trong trò chơi đối kháng

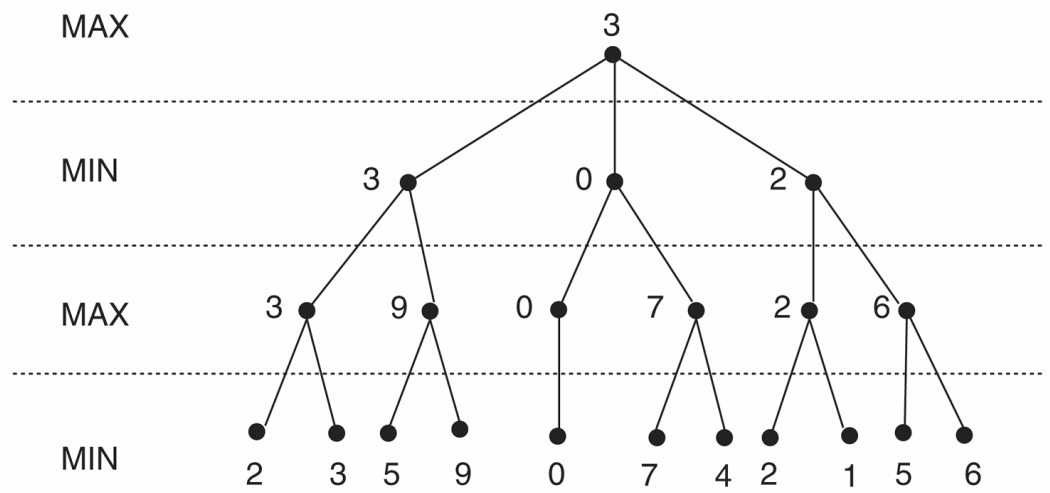
- **Giải thuật minimax**:
  - Hai đấu thủ trong trò chơi được gọi là **MIN** và **MAX**.
  - Mỗi nút lá có giá trị:
    - **1** nếu là MAX thắng,
    - **0** nếu là MIN thắng.
  - Minimax sẽ truyền các giá trị này lên cao dần trên đồ thị, qua các nút cha mẹ kế tiếp theo các luật sau:
    - Nếu trạng thái cha mẹ là **MAX**, gán cho nó giá trị **lớn nhất** có trong các trạng thái con.
    - Nếu trạng thái bố, mẹ là **MIN**, gán cho nó giá trị **nhỏ nhất** có trong các trạng thái con.

Hãy áp dụng GT Minimax vào Trò Chơi NIM



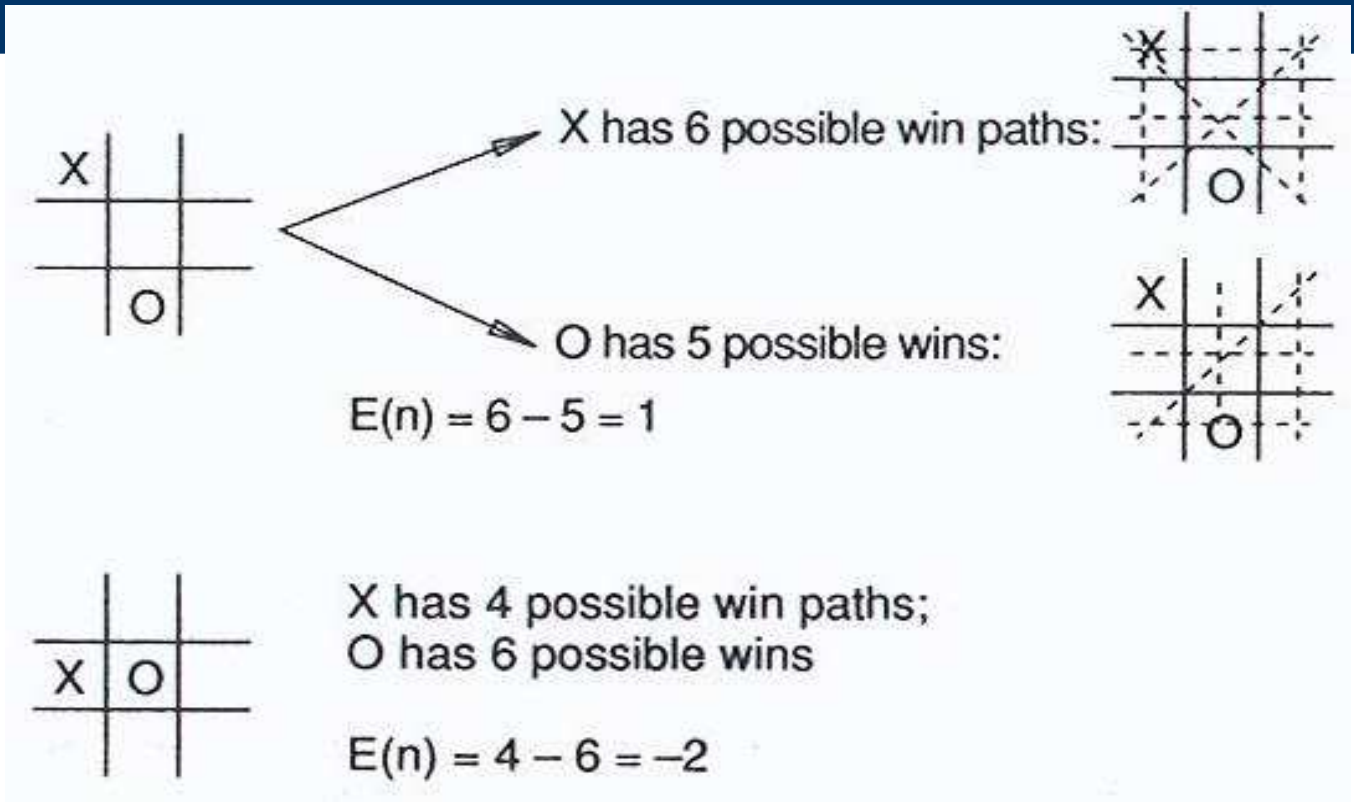
Minimax với độ sâu lớp cố định

- Minimax đối với một KGTT giả định.



- Các nút lá được gán các giá trị ***heuristic***
- Còn giá trị tại các nút trong là các giá trị nhận được dựa trên giải thuật Minimax

Heuristic trong trò chơi tic-tac-toe



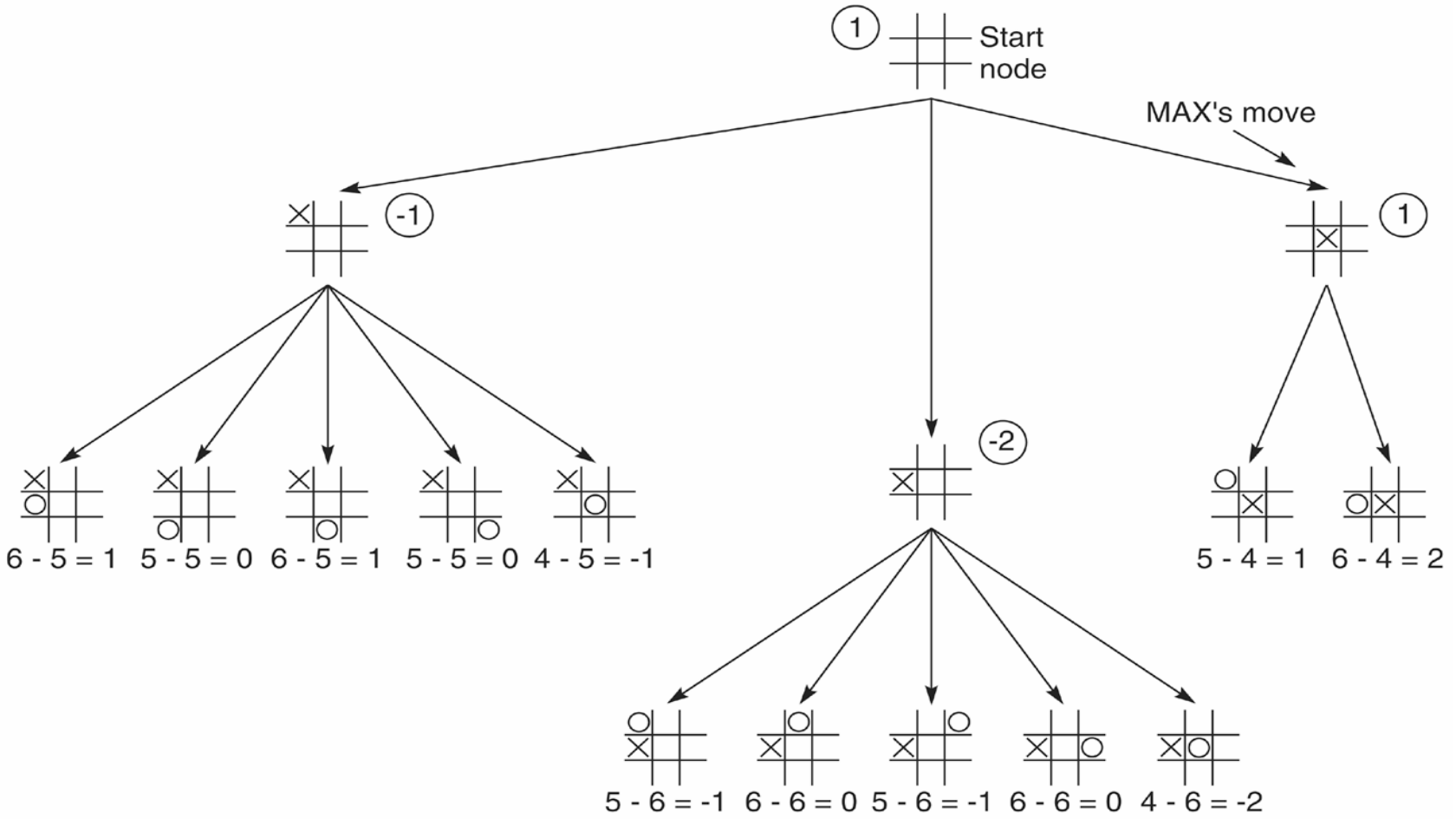
Hàm Heuristic: **E(n) = M(n) – O(n)**

Trong đó: M(n) là *tổng số đường thắng có thể của tôi*

O(n) là *tổng số đường thắng có thể của đối thủ*

E(n) là *trị số đánh giá tổng cộng cho trạng thái n*

Minimax 2 lớp trong tic-tac-toe

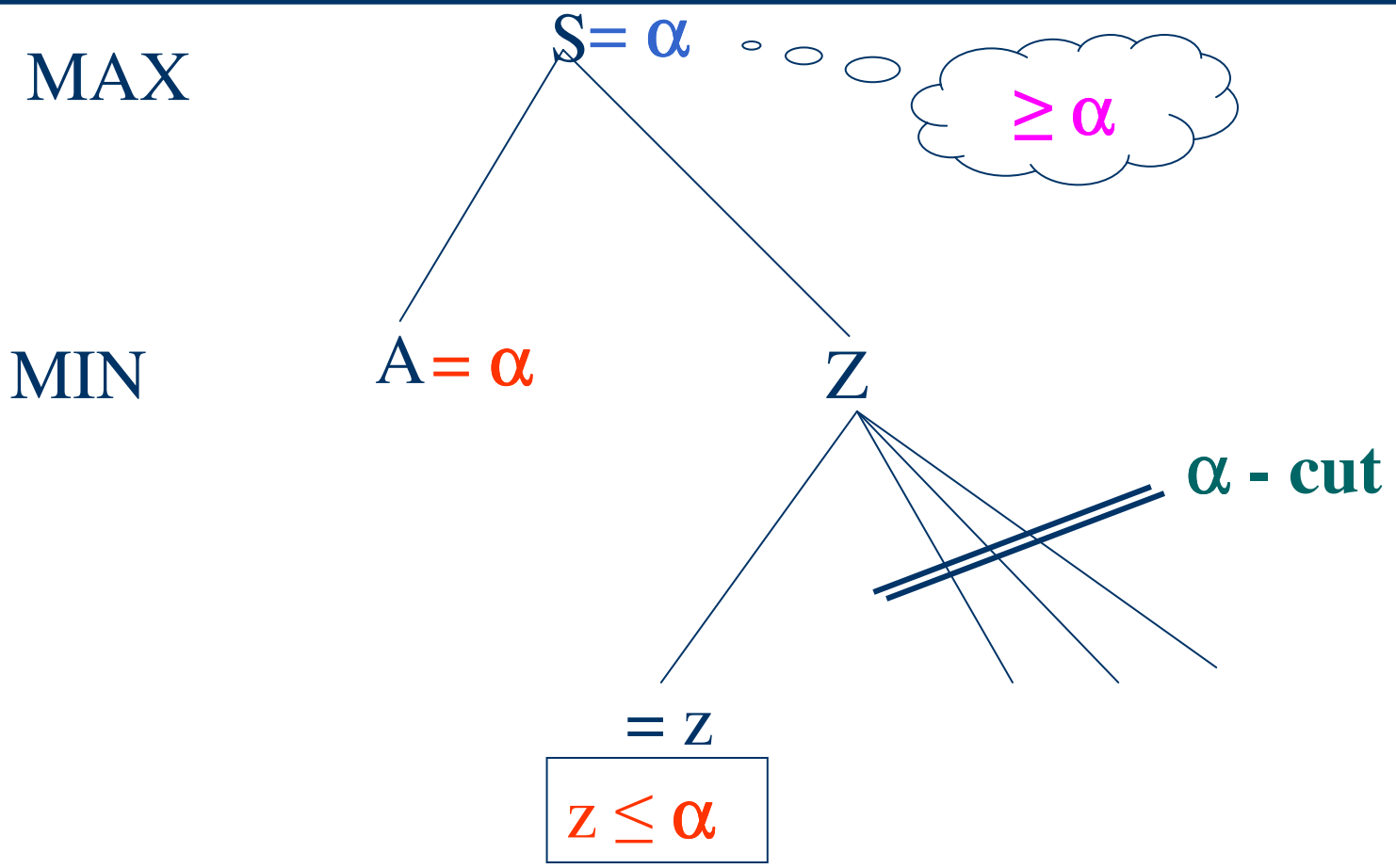


Trích từ Nilsson (1971).

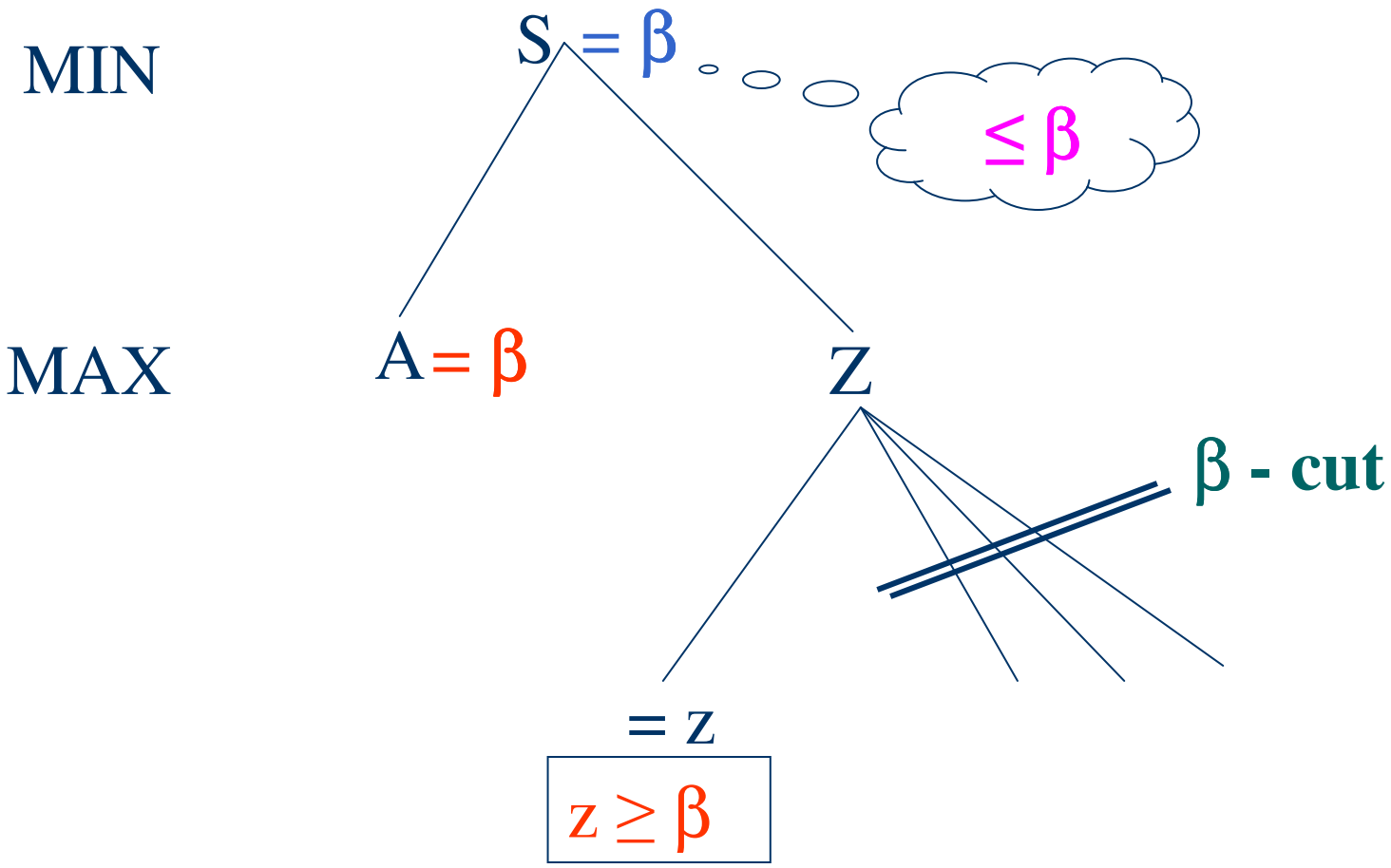
Giải thuật cắt tỉa α-β

- Tìm kiếm theo kiểu depth-first.
- Nút MAX có 1 giá trị α (luôn tăng)
- Nút MIN có 1 giá trị β (luôn giảm)
- **TK có thể kết thúc dưới bất kỳ**:
  - Nút MIN nào có $\beta \leq \alpha$ của bất kỳ nút cha MAX nào.
  - Nút MAX nào có $\alpha \geq \beta$ của bất kỳ nút cha MIN nào.
- Giải thuật cắt tỉa α-β thể hiện *mối quan hệ giữa các nút ở lớp n và n+2*, mà tại đó toàn bộ cây có gốc tại lớp n+1 có thể cắt bỏ.

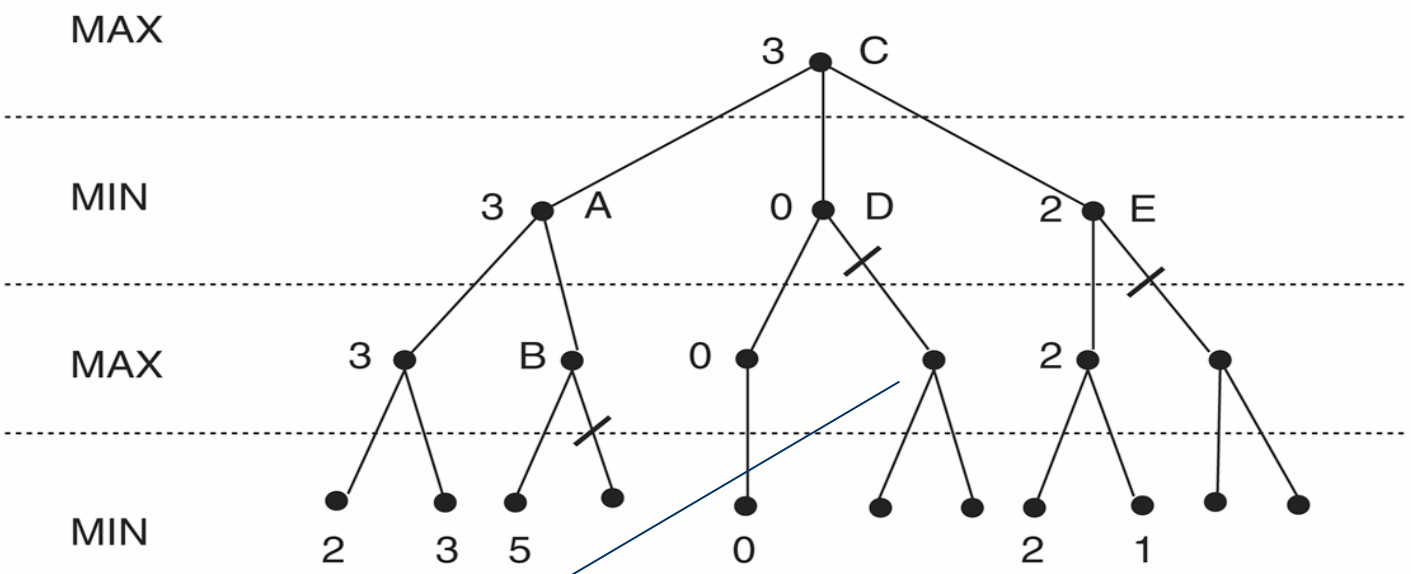
Cắt tỉa α



Cắt tỉa β



GT Cắt Tỉa α-β áp dụng cho KGTT giả định



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

A has β = 3 (A will be no larger than 3)
B is β pruned, since 5 > 3
C has α = 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

Chương 4: Biểu diễn và suy luận tri thức

TS. Nguyễn Đình Thuân
Khoa Công nghệ Thông tin
Đại học Nha Trang
Email: thuanvinh122@gmail.com

4.1. Mở đầu

- *tri thức*, *lĩnh vực* và *biểu diễn tri thức*.

4.2. Các loại tri thức: được chia thành 5 loại

1. Tri thức thủ tục: mô tả cách thức giải quyết một vấn đề. Loại tri thức này đưa ra giải pháp để thực hiện một công việc nào đó. Các dạng tri thức thủ tục tiêu biểu thường là các luật, chiến lược, lịch trình và thủ tục.
2. Tri thức khai báo: cho biết một vấn đề được thấy như thế nào. Loại tri thức này bao gồm các phát biểu đơn giản, dưới dạng các khẳng định logic đúng hoặc sai. Tri thức khai báo cũng có thể là một danh sách các khẳng định nhằm mô tả đầy đủ hơn về đối tượng hay một khái niệm nào đó.

4.2. Các loại tri thức (tiếp)

3. **Siêu tri thức:** mô tả *tri thức về tri thức*. Loại tri thức này giúp lựa chọn tri thức thích hợp nhất trong số các tri thức khi giải quyết một vấn đề. Các chuyên gia sử dụng tri thức này để điều chỉnh hiệu quả giải quyết vấn đề bằng cách hướng các lập luận về miền tri thức có khả năng hơn cả.
4. **Tri thức heuristic:** mô tả các "*mẹo*" để dẫn dắt tiến trình lập luận. Tri thức heuristic là *tri thức* không bảm đảm hoàn toàn 100% chính xác về kết quả giải quyết vấn đề. Các chuyên gia thường dùng các tri thức khoa học như sự kiện, luật, … sau đó chuyển chúng thành các tri thức heuristic để thuận tiện hơn trong việc giải quyết một số bài toán.
5. **Tri thức có cấu trúc:** mô tả tri thức theo cấu trúc. Loại tri thức này mô tả mô hình tổng quan hệ thống theo quan điểm của chuyên gia, bao gồm khái niệm, khái niệm con, và các đối tượng; diễn tả chức năng và mối liên hệ giữa các tri thức dựa theo cấu trúc xác định.

Ví dụ: Hãy phân loại các tri thức sau

1. Nha Trang là thành phố đẹp.
2. Bạn Lan thích đọc sách.
3. Thuật toán tìm kiếm BFS, DFS
4. Thuật giải Greedy
5. Một số cách chiếu tướng trong việc chơi cờ tướng.
6. Hệ thống các khái niệm trong hình học.
7. Cách tập viết chữ đẹp.
8. Tóm tắt quyển sách về Trí tuệ nhân tạo.
9. Chọn loại cổ phiếu để mua cổ phiếu.

4.3. CÁC KỸ THUẬT BIỂU DIỄN TRI THỨC

4.3.1 Bộ ba Đối tượng-Thuộc tính-Giá trị

4.3.2 Các luật dẫn

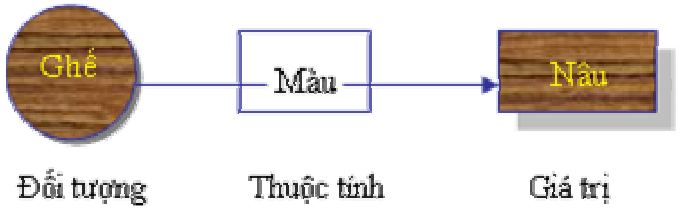
4.3.3 Mạng ngữ nghĩa

4.3.4 Frames

4.3.5 Logic

4.3.1 Bộ ba Đối tượng-Thuộc tính-Giá trị

- Một sự kiện có thể được dùng để xác nhận giá trị của một thuộc tính xác định của một vài đối tượng. Ví dụ, mệnh đề "quả bóng màu đỏ" xác nhận "đỏ" là giá trị thuộc tính "màu" của đối tượng "quả bóng". Kiểu sự kiện này được gọi là bộ ba Đối tượng-Thuộc tính-Giá trị (O-A-V – Object-Attribute-Value).



Hình 2.1. Biểu diễn tri thức theo bộ ba O-A-V

4.3.1 Bộ ba Đối tượng-Thuộc tính-Giá trị (tiếp)

- Trong các sự kiện O-A-V, một đối tượng có thể có nhiều thuộc tính với các kiểu giá trị khác nhau. Hơn nữa một thuộc tính cũng có thể có một hay nhiều giá trị. Chúng được gọi là các sự kiện *đơn trị* (single-valued) hoặc *đa trị* (multi-valued). Điều này cho phép các hệ tri thức linh động trong việc biểu diễn các tri thức cần thiết.
- Các sự kiện không phải lúc nào cũng bảo đảm là đúng hay sai với độ chắc chắn hoàn toàn. Ví thế, khi xem xét các sự kiện, người ta còn sử dụng thêm một khái niệm là *độ tin cậy*. Phương pháp truyền thống để quản lý thông tin không chắc chắn là sử dụng nhân tố chắc chắn CF (certainly factor). Khái niệm này bắt đầu từ hệ thống MYCIN (khoảng năm 1975), dùng để trả lời cho các thông tin suy luận. Khi đó, trong sự kiện O-A-V sẽ có thêm một giá trị xác định độ tin cậy của nó là CF.

4.3.2 Các luật dẫn

- Luật là cấu trúc tri thức dùng để liên kết thông tin đã biết với các thông tin khác giúp đưa ra các suy luận, kết luận từ những thông tin đã biết.
- Trong hệ thống dựa trên các luật, người ta thu thập các tri thức lĩnh vực trong một tập và lưu chúng trong cơ sở tri thức của hệ thống. Hệ thống dùng các luật này cùng với các thông tin trong bộ nhớ để giải bài toán. Việc xử lý các luật trong hệ thống dựa trên các luật được quản lý bằng một module gọi là *bộ suy diễn*.

4.3.2 Các luật dẫn*(tiếp)*

Các dạng luật cơ bản: 7 dạng

1. Quan hệ:

IF Bình điện hỏng

THEN Xe sẽ không khởi động được

2. Lời khuyên:

IF Xe không khởi động được

THEN Đi bộ

3. Hướng dẫn

IF Xe không khởi động được AND Hệ thống nhiên liệu tốt

THEN Kiểm tra hệ thống điện

4.3.2 Các luật dẫn*(tiếp)*

4. Chiến lược

IF Xe không khởi động được

THEN Đầu tiên hãy kiểm tra hệ thống nhiên liệu, sau đó kiểm tra hệ thống điện

5. Diễn giải

IF Xe nổ AND tiếng giòn

THEN Động cơ hoạt động bình thường

6. Chẩn đoán

IF Sốt cao AND hay ho AND Họng đỏ

THEN Viêm họng

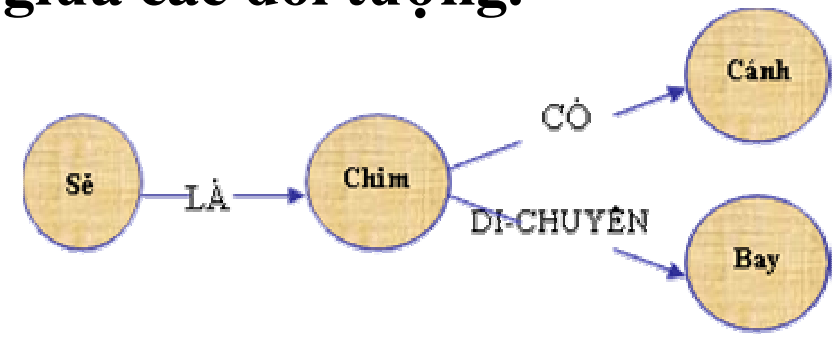
7. Thiết kế

IF Là nữ AND Da sáng

THEN Nên chọn Xe Spacy AND Chọn màu sáng

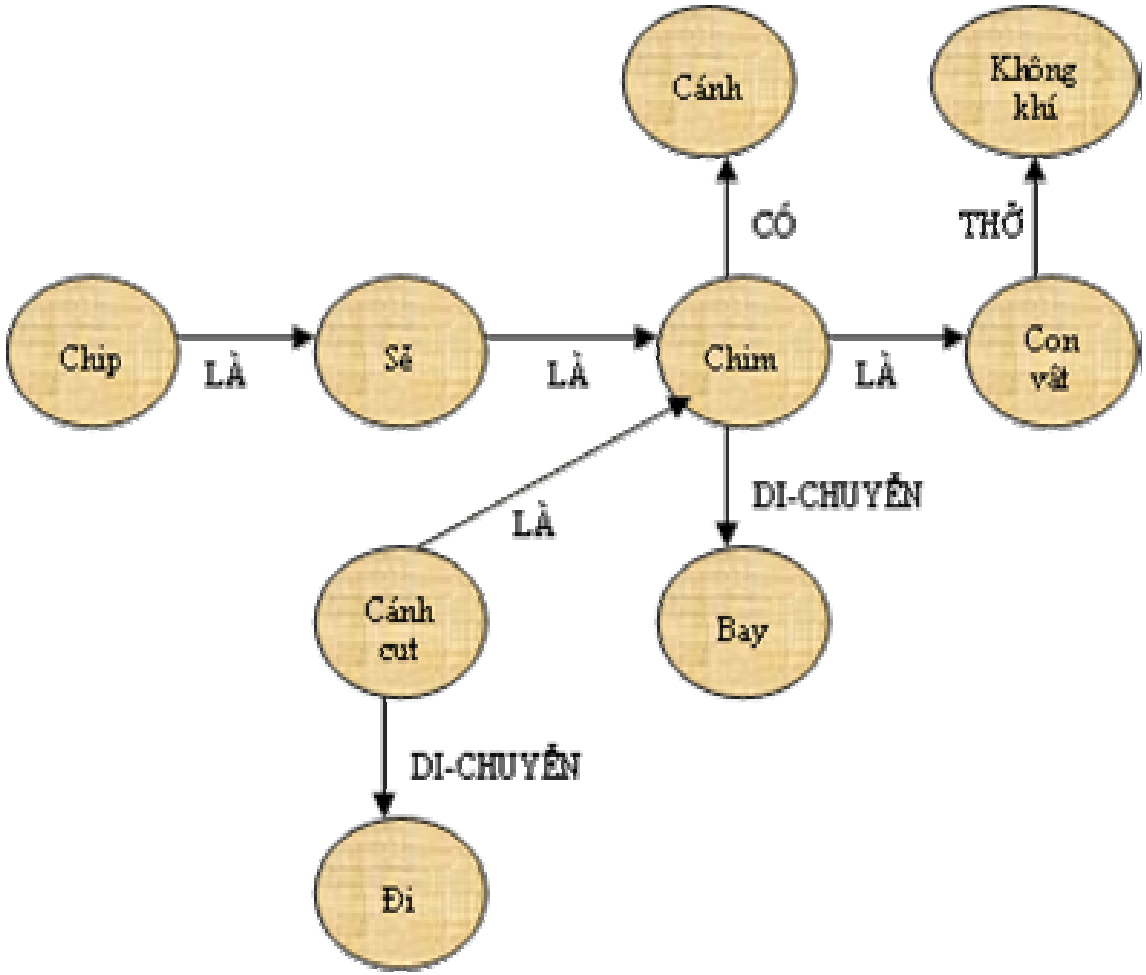
4.3.3 Mạng ngữ nghĩa

Mạng ngữ nghĩa là một phương pháp biểu diễn tri thức dùng đồ thị trong đó nút biểu diễn đối tượng và cung biểu diễn quan hệ giữa các đối tượng.



Hình 2.3. "Sẻ là Chim" thể hiện trên mạng ngữ nghĩa

4.3.3 Mạng ngữ nghĩa*(tiếp)*



Hình 4.4. Phát triển mạng ngữ nghĩa

Ví dụ: Giải bài toán tam giác tổng quát

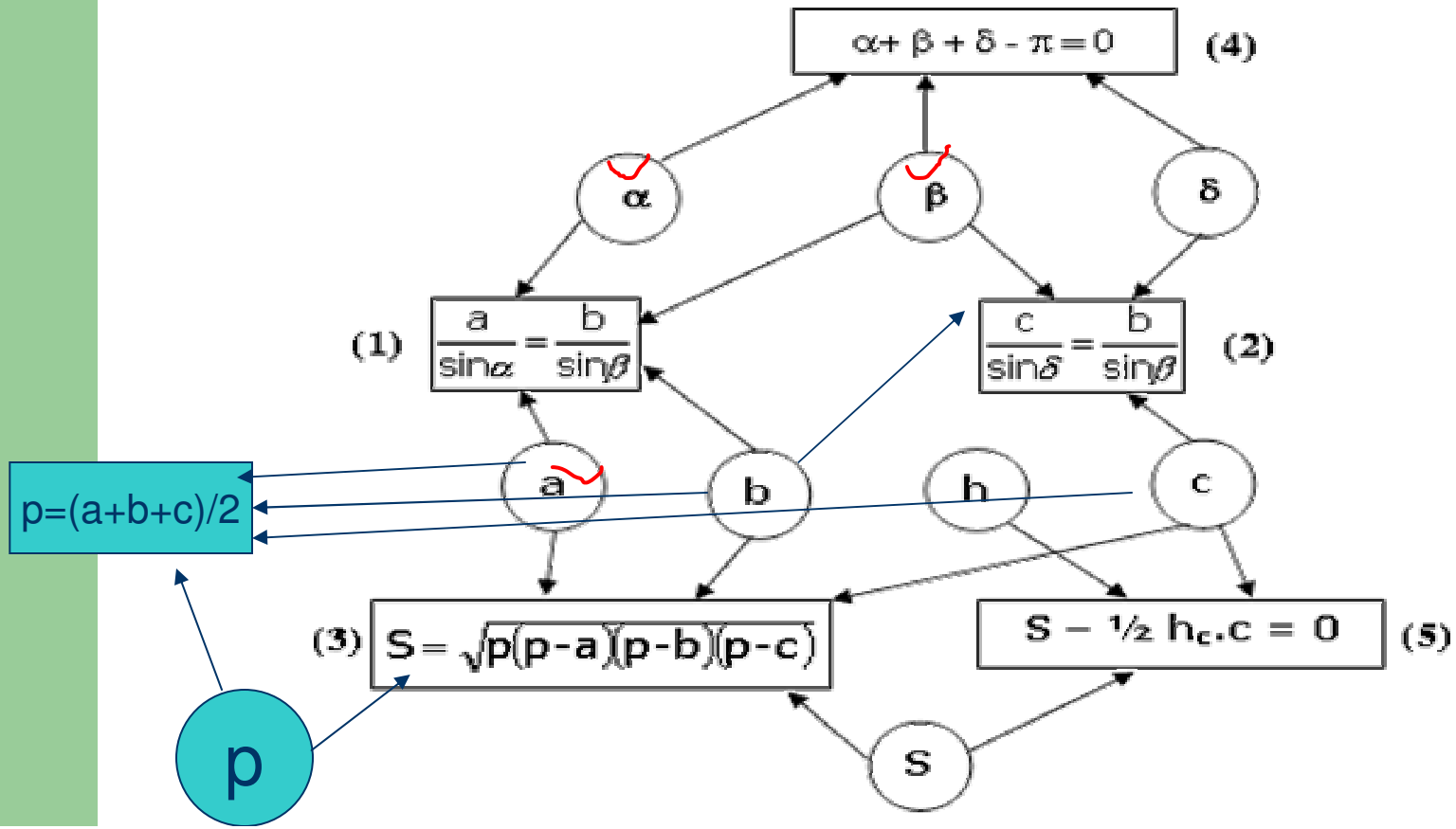
- Có 22 yếu tố của tam giác. Như vậy có C^3_{22} -1 cách để xây dựng hay xác định một tam giác.
- Theo thống kê, có khoảng 200 công thức liên quan đến cạnh và góc 1 tam giác.
- Để giải bài toán này bằng công cụ mạng ngữ nghĩa, sử dụng khoảng 200 đỉnh để chứa công thức và khoảng 22 đỉnh để chứa các yếu tố của tam giác. Mạng ngữ nghĩa cho bài toán này có cấu trúc như sau :
    - Đỉnh của đồ thị bao gồm hai loại :
        - Đỉnh chứa công thức (ký hiệu bằng hình chữ nhật)
        - Đỉnh chứa yếu tố của tam giác (ký hiệu bằng hình tròn)
    - Cung : chỉ nối từ đỉnh hình tròn đến đỉnh hình chữ nhật cho biết yếu tố tam giác xuất hiện trong công thức nào
- * Lưu ý : trong một công thức liên hệ giữa n yếu tố của tam giác, ta giả định rằng nếu đã biết giá trị của n-1 yếu tố thì sẽ tính được giá trị của yếu tố còn lại. Chẳng hạn như trong công thức tổng 3 góc của tam giác bằng 180^0 thì khi biết được hai góc, ta sẽ tính được góc còn lại.

Ví dụ: Giải bt tam giác tổng quát (tt)

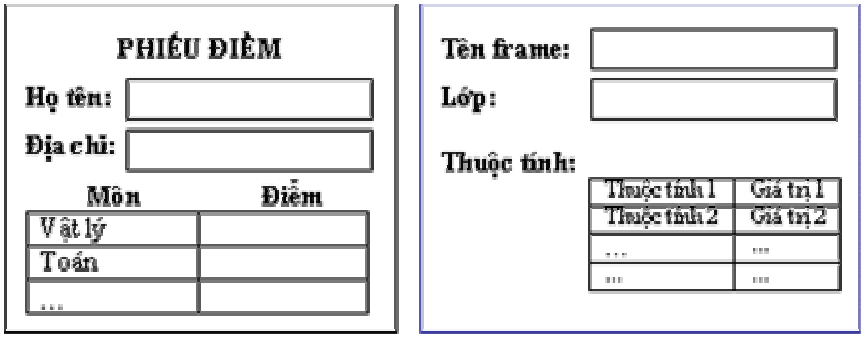
- **B1 :** Kích hoạt những **đỉnh hình tròn** đã cho ban đầu (những yếu tố đã có giá trị)
- **B2 :** Lặp lại bước sau cho đến khi kích hoạt được tất cả những đỉnh ứng với những yếu tố cần tính hoặc không thể kích hoạt được bất kỳ đỉnh nào nữa.
- **Nếu** một đỉnh hình chữ nhật có cung nối với **n** đỉnh hình tròn mà **n-1** đỉnh hình tròn đã được kích hoạt **thì** kích hoạt đỉnh hình tròn còn lại (và tính giá trị đỉnh còn lại này thông qua công thức ở đỉnh hình chữ nhật).

Ví dụ: Giải bt tam giác tổng quát (tt)

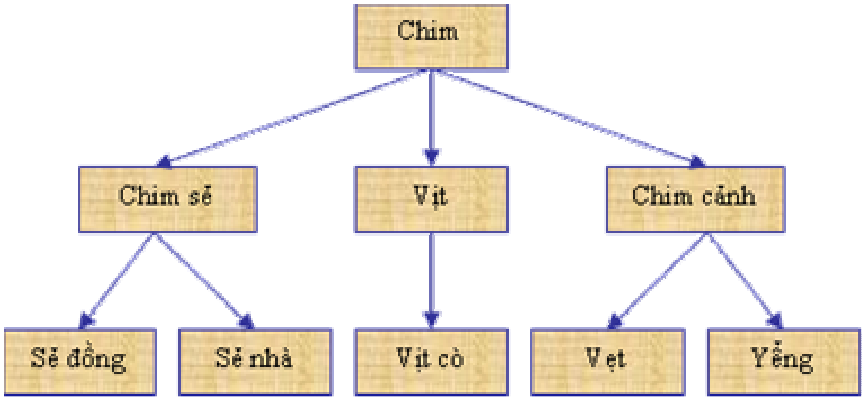
Ví dụ : *"Cho hai góc α, β và chiều dài cạnh a của tam giác. Tính chiều dài đường cao h_c".*



4.3.4 Frame



Hình 2.6. Cấu trúc frame



Hình 2.7. Nhiều mức của frame mô tả quan hệ phức tạp hơn

4.3.5 Logic

1. Logic mệnh đề

IF Xe không khởi động được (A)

AND Khoảng cách từ nhà đến chỗ làm là xa (B)

THEN Sẽ trễ giờ làm (C)

- Luật trên có thể biểu diễn lại như sau: $\mathbf{A \wedge B \Rightarrow C}$

2. Logic vị từ

- Logic vị từ, cũng giống như logic mệnh đề, dùng các ký hiệu để thể hiện tri thức. Những ký hiệu này gồm **hằng số**, **vị từ**, **biến** và **hàm**.

4.4 SUY DIỄN DỮ LIỆU

1. Modus ponens

1. E1
2. E1→ E2
3. E2

Nếu có tiên đề khác, có dạng **E2 → E3** thì **E3** được đưa vào danh sách.

2. Modus tollens

1. ¬ E2
2. E1→ E2
3. ¬ E1

4.5 Chứng minh mệnh đề

- Một trong những vấn đề khá quan trọng của logic mệnh đề là chứng minh tính đúng đắn của phép suy diễn ($a \rightarrow b$). Đây cũng chính là bài toán chứng minh thường gặp trong toán học.
- Với hai phép suy luận cơ bản của logic mệnh đề (Modus Ponens, Modus Tollens) cộng với các phép biến đổi hình thức, ta cũng có thể chứng minh được phép suy diễn. Tuy nhiên, thao tác biến đối hình thức là rất khó cài đặt được trên máy tính. Thậm chí điều này còn khó khăn với cả con người!
- Với công cụ máy tính, có thể cho rằng ta sẽ dễ dàng chứng minh được mọi bài toán bằng một phương pháp đã biết là lập bảng chân trị . Tuy về lý thuyết, phương pháp lập bảng chân trị luôn cho được kết quả cuối cùng nhưng độ phức tạp của phương pháp này là quá lớn, $O(2^n)$ với n là số biến mệnh đề. Sau đây chúng ta sẽ nghiên cứu hai phương pháp chứng minh mệnh đề với độ phức tạp chỉ có O(n).

4.5 Chứng minh mệnh đề

- Một trong những vấn đề khá quan trọng của logic mệnh đề là chứng minh tính đúng đắn của phép suy diễn ($a \rightarrow b$). Đây cũng chính là bài toán chứng minh thường gặp trong toán học.
- Với hai phép suy luận cơ bản của logic mệnh đề (Modus Ponens, Modus Tollens) cộng với các phép biến đổi hình thức, ta cũng có thể chứng minh được phép suy diễn. Tuy nhiên, thao tác biến đối hình thức là rất khó cài đặt được trên máy tính. Thậm chí điều này còn khó khăn với cả con người!
- Với công cụ máy tính, có thể cho rằng ta sẽ dễ dàng chứng minh được mọi bài toán bằng một phương pháp đã biết là lập bảng chân trị . Tuy về lý thuyết, phương pháp lập bảng chân trị luôn cho được kết quả cuối cùng nhưng độ phức tạp của phương pháp này là quá lớn, $O(2^n)$ với n là số biến mệnh đề. Sau đây chúng ta sẽ nghiên cứu hai phương pháp chứng minh mệnh đề với độ phức tạp chỉ có O(n).

4.5.1 Thuật giải Vương Hạo

B1 : Phát biểu lại giả thiết và kết luận của vấn đề theo dạng chuẩn sau :

$$GT_1, GT_2, ..., GT_n \rightarrow KL_1, KL_2, ..., KL_m$$

Trong đó các GT_i và KL_i là các mệnh đề được xây dựng từ các biến mệnh đề và 3 phép nối cơ bản : $\wedge, \vee, \neg$

B2 : Chuyển về các GT_i và KL_i có dạng phủ định.

Ví dụ :

$$p \vee q, \neg (r \wedge s), \neg g, p \vee r \rightarrow s, \neg p$$
$$\Rightarrow p \vee q, p \vee r, p \rightarrow (r \wedge s), g, s$$

B3 : Nếu GT_i có phép $\wedge$ thì thay thế phép $\wedge$ bằng dấu ","

Nếu KL_i có phép $\vee$ thì thay thế phép $\vee$ bằng dấu ","

Ví dụ :

$$p \wedge q, r \wedge (\neg p \vee s) \rightarrow \neg q, \neg s$$
$$\Rightarrow p, q, r, \neg p \vee s \rightarrow \neg q, \neg s$$

4.5.1 Thuật giải Vương Hạo

B4 : Nếu GT_i có phép $\vee$ thì tách thành hai dòng con.
Nếu ở KLi có phép $\wedge$ thì tách thành hai dòng con.
Ví dụ :

$$p, \neg p \vee q \rightarrow q$$
$$p, \neg p \rightarrow q$$
và $\quad p, q \rightarrow q$

B5 : Một dòng được chứng minh nếu tồn tại chung một mệnh đề ở ở cả hai phía.
Ví dụ :

$p, \mathbf{q} \rightarrow \mathbf{q}$ được chứng minh
$p, \neg p \rightarrow q \Rightarrow \mathbf{p} \rightarrow \mathbf{p}, q$

B6 :

a) Nếu một dòng không còn phép nối $\wedge$ và phép nối $\vee$ ở cả hai vế và ở 2 vế không có chung một biến mệnh đề thì dòng đó không được chứng minh.

b) Một vấn đề được chứng minh nếu tất cả dòng dẫn xuất từ dạng chuẩn ban đầu đều được chứng minh.

Ví dụ: i) $p \wedge (\neg p \vee q) \rightarrow q$
ii) $(p \vee q) \wedge (\neg p \vee r) \rightarrow q \vee r$

4.5.2 Thuật giải Robinson

- Thuật giải này hoạt động dựa trên phương pháp chứng minh phản chứng và phép hợp giải Robinson.
- Phương pháp chứng minh phản chứng:
  - Chứng minh phép suy luận ($a \rightarrow b$) là đúng (với a là giả thiết, b là kết luận).
  - Phản chứng : giả sử b sai suy ra $\neg$ b là đúng.
- Phép hợp giải Robinson:

  i) $p \wedge (\neg p \vee q) \rightarrow q$

  ii) $(p \vee q) \wedge (\neg p \vee r) \rightarrow q \vee r$

Bài toán được chứng minh nếu a đúng và $\neg$ b đúng sinh ra một mâu thuẫn.

4.5.2 Thuật giải Robinson (tiếp)

B1 : Phát biểu lại giả thiết và kết luận của vấn đề dưới dạng chuẩn như sau :

$$GT_1, GT_2, ..., GT_n \rightarrow KL_1, KL_2, ..., KL_m$$

Trong đó : GT_i và KL_j được xây dựng từ các biến mệnh đề và các phép toán : $\wedge, \vee, \neg$

B2 : Nếu GT_i có phép $\wedge$ thì thay bằng dấu ","

Nếu KL_i có phép $\vee$ thì thay bằng dấu ","

B3 : Biến đổi dòng chuẩn ở B1 về thành danh sách mệnh đề như sau :

$$\{ GT_1, GT_2, ..., GT_n , \neg KL_1, \neg KL_2, ..., \neg KL_m \}$$

B4 : Nếu trong danh sách mệnh đề ở bước 2 có 2 mệnh đề đối ngẫu nhau thì bài toán được chứng minh. Ngược lại thì chuyển sang B5. (a và ¬a gọi là hai mệnh đề đối ngẫu nhau)

4.5.2 Thuật giải Robinson (tiếp)

B6 : Áp dụng phép hợp giải

i) $p \wedge (\neg p \vee q) \rightarrow q$

ii) $(p \vee q) \wedge (\neg p \vee r) \rightarrow q \vee r$

B7 : Nếu không xây dựng được thêm một mệnh đề mới nào và trong danh sách mệnh đề không có 2 mệnh đề nào đối ngẫu nhau thì vấn đề không được chứng minh.

Ví dụ : Chứng minh rằng

$(\neg p \vee q) \wedge (\neg q \vee r) \wedge (\neg r \vee s) \wedge (\neg u \vee \neg s) \rightarrow \neg p \vee \neg u$

Chương 5 Máy học

5.1 MỞ ĐẦU

- Các chương trước đã thảo luận về biểu diễn và suy luận tri thức. Trong trường hợp này giả định đã có sẵn tri thức và có thể biểu diễn tường minh tri thức.
- Tuy vậy trong nhiều tinh huống, sẽ không có sẵn tri thức như:
  - Kỹ sư tri thức cần thu nhận tri thức từ chuyên gia lĩnh vực.
  - Cần biết các luật mô tả lĩnh vực cụ thể.
  - Bài toán không được biểu diễn tường minh theo luật, sự kiện hay các quan hệ.
- Có hai tiếp cận cho hệ thống học:
  - Học từ ký hiệu: bao gồm việc hình thức hóa, sửa chữa các luật tường minh, sự kiện và các quan hệ.
  - Học từ dữ liệu số: được áp dụng cho những hệ thống được mô hình dưới dạng số liên quan đến các kỹ thuật nhằm tối ưu các tham số. Học theo dạng số bao gồm mạng Neural nhân tạo, thuật giải di truyền, bài toán tối ưu truyền thông. Các kỹ thuật học theo số không tạo ra CSTT tường minh.

5.2 CÁC HÌNH THỨC HỌC

1. Học vẹt: Hệ tiếp nhận các khẳng định của các quyết định đúng. Khi hệ tạo ra một quyết định không đúng, hệ sẽ đưa ra các luật hay quan hệ đúng mà hệ đã sử dụng. Hình thức học vẹt nhằm cho phép chuyên gia cung cấp tri thức theo kiểu tương tác.
2. Học bằng cách chỉ dẫn: Thay vì đưa ra một luật cụ thể cần áp dụng vào tình huống cho trước, hệ thống sẽ được cung cấp bằng các chỉ dẫn tổng quát. Ví dụ: "gas hầu như bị thoát ra từ van thay vì thoát ra từ ống dẫn". Hệ thống phải tự mình đề ra cách biến đổi từ trừu tượng đến các luật khả dụng.
3. Học bằng qui nạp: Hệ thống được cung cấp một tập các ví dụ và kết luận được rút ra từ từng ví dụ. Hệ liên tục lọc các luật và quan hệ nhằm xử lý từng ví dụ mới.

5.2 CÁC HÌNH THỨC HỌC (Tiếp)

4. Học bằng tương tự: Hệ thống được cung cấp đáp ứng đúng cho các tác vụ tương tự nhưng không giống nhau. Hệ thống cần làm thích ứng đáp ứng trước đó nhằm tạo ra một luật mới có khả năng áp dụng cho tình huống mới.
5. Học dựa trên giải thích: Hệ thống phân tích tập các lời giải ví dụ (và kết quả) nhằm ấn định khả năng đúng hoặc sai và tạo ra các giải thích dùng để hướng dẫn cách giải bài toán trong tương lai.
6. Học dựa trên tình huống: Bất kỳ tính huống nào được hệ thống lập luận đều được lưu trữ cùng với kết quả cho dù đúng hay sai. Khi gặp tình hướng mới, hệ thống sẽ làm thích nghi hành vi đã lưu trữ với tình huống mới.
7. Khám phá hay học không giám sát: Thay vì có mục tiêu tường minh, hệ khám phá liên tục tìm kiếm các mẫu và quan hệ trong dữ liệu nhập. Các ví dụ về học không giám sát bao gồm gom cụm dữ liệu, học để nhận dạng các đặc tính cơ bản như cạnh từ các điểm ảnh.

Ví dụ về CÁC HÌNH THỨC HỌC

Ví dụ:

- **Hệ MYCIN**
- **Mạng Neural nhân tạo**
- **Thuật toán học Quinland**
- **Bài toán nhận dạng**
- **Máy chơi cờ carô, cờ tướng**

5.3 THUẬT GIẢI Quinlan

- Là thuật toán học theo quy nạp dùng luật, đa mục tiêu.
- Do Quinlan đưa ra năm 1979.
- Ý tưởng: Chọn thuộc tính quan trọng nhất để tạo cây quyết định.
- Thuộc tính quan trọng nhất là thuộc tính phân loại Bảng quan sát thành các bảng con sao cho từ mỗi bảng con này dễ phân tích để tìm quy luật chung.

5.3.1 THUẬT GIẢI A. Quinlan

STT	Size	Nationality	Family	Conclusion
1	Small	German	Single	A
2	Large	French	Single	A
3	Large	German	Single	A
4	Small	Italian	Single	B
5	Large	German	Married	B
6	Large	Italian	Single	B
7	Large	Italian	Married	B
8	Small	German	Married	B

Với mỗi thuộc tính của bảng quan sát:

- Xét vector V: có số chiều bằng số phân loại
  - $V_{(Size=Small)} = (A_{Small}, B_{Small})$
  - A_{Small}=Số quan sát A có Size là Small / Tổng số quan sát có Size=Small
  - B_{Small}= Số quan sát B có Size là Small / Tổng số quan sát có Size=Small

    $V_{(Size=Small)} = (1/3 , 2/3)$

    $V_{(Size=Large)} = (2/5 , 3/5)$
- Với thuộc tính Nationality

  $V_{(Nat = German)} = (2/4 , 2/4)$

  $V_{(Nat = French)} = (1 , 0)$

  $V_{(Nat = Italian)} = (0 , 1)$
- Thuộc tính Family:

  $V_{(Family=Single)} = (3/5 , 2/5)$

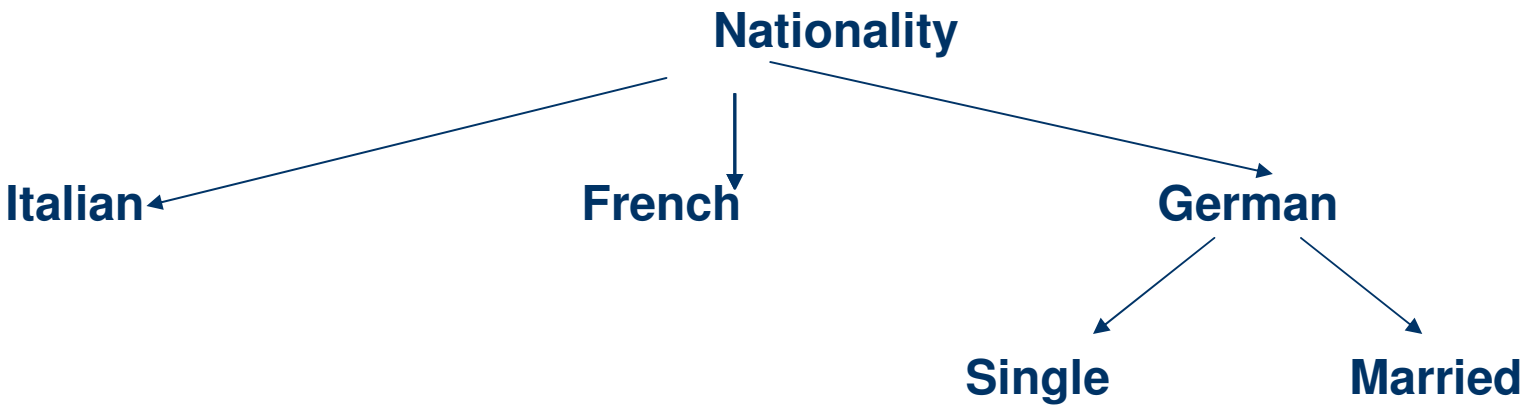
  $V_{(Family = Married)} = (0, 1)$

Với mỗi thuộc tính của bảng quan sát:

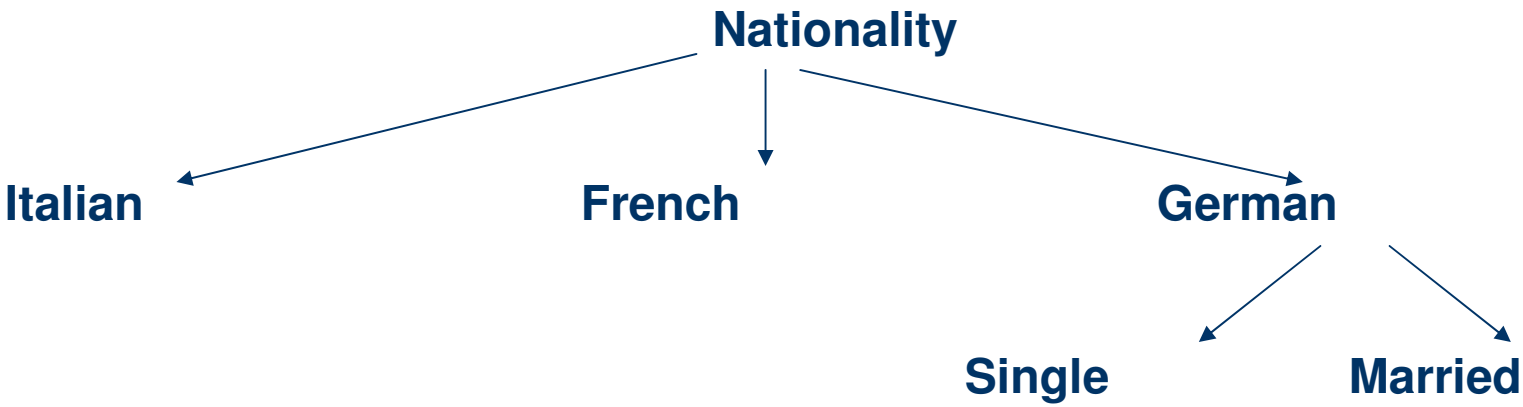
Chỉ còn xét German

- Thuộc tính Size:
  - $V_{(Size=Small)} = (1/2 , 1/2)$
  - $V_{(Size=Large)} = (1/2 , 1/2)$
- Thuộc tính Family:
  - $V_{(Family=Single)} = (1, 0)$
  - $V_{(Family=Married)} = (0,1)$

STT	Size	Family	Conclusion
1	Small	Single	A
2	Large	Single	A
3	Large	Married	B
4	Small	Married	B



Với mỗi thuộc tính của bảng quan sát(tiếp)



Rule 1: If (Nationality IS Italian) then (Conclusion IS B)

Rule 2: If (Nationality IS French) then (Conclusion IS A)

Rule 3: If (Nationality IS German) AND (Family IS Single)
then (Conclusion IS A)

Rule 4: If (Nationality IS German) AND (Family IS Married)
then (Conclusion IS B)

5.3.2 Thuật giải Học theo độ bất định

Stt	Age	Competition	Type	Profit
1	Old	No	Software	Down
2	Midle	Yes	Software	Down
3	Midle	No	Hardware	Up
4	Old	No	Hardware	Down
5	New	No	Hardware	Up
6	New	No	Software	Up
7	Midle	No	Software	Up
8	New	Yes	Software	Up
9	Midle	Yes	Hardware	Down
10	Old	Yes	Hardware	Down

Học theo độ bất định(tiếp)

- Độ bất định của X:

$$E(X) = -\sum_{i=1}^{k} p_i \log_2 p_i$$

- Tính Entropy cho mỗi thuộc tính và chọn thuộc tính có Entropy nhỏ nhất.

$$E(C / A) = -\sum_{i=1}^{k} p(c_i, a_i) \log_2 p(c_i, a_i)$$

$$E(C / Competitio\ \ n=No) = -\frac{4}{6}\log_2\frac{4}{6} - \frac{2}{6}\log_2\frac{2}{6} = 0.918$$

$$E(C / Competitio\ \ n=Yes) = -\frac{1}{4}\log_2\frac{1}{4} - \frac{3}{4}\log_2\frac{3}{4} = 0.811$$

$$E(C / Competitio\ \ n) = 0.6 * 0.918 + 0.4 * 0.811 = 0.8752$$

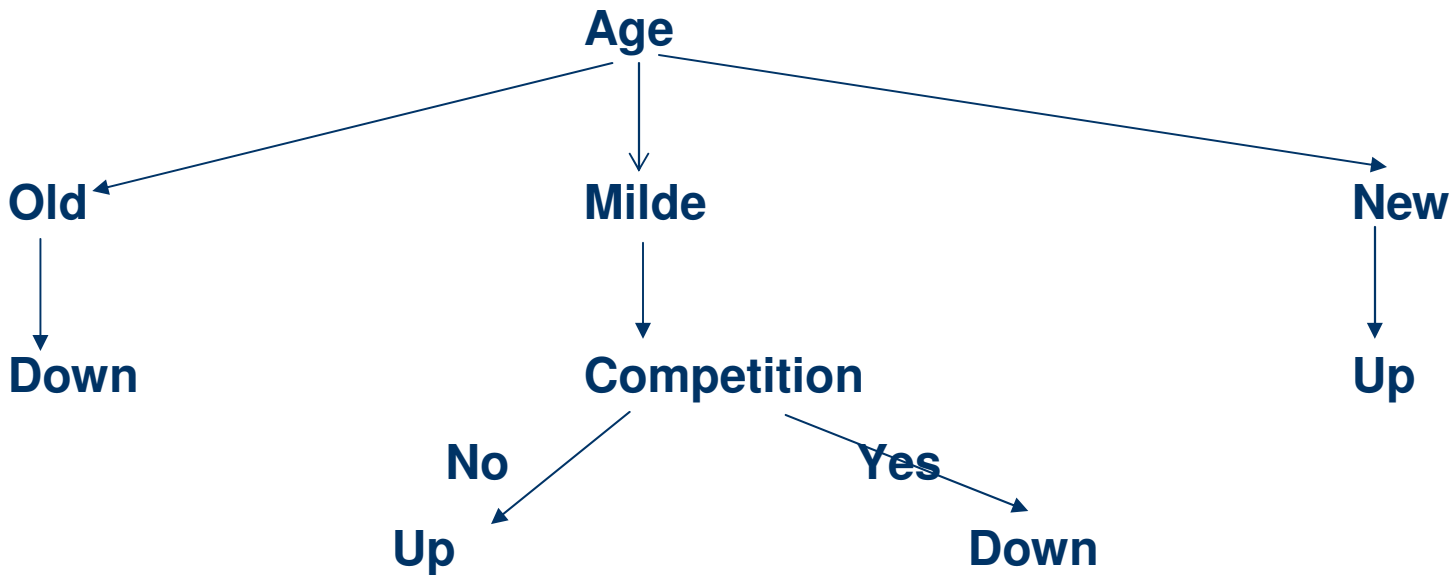
Học theo độ bất định(tiếp)

- Tương tự:

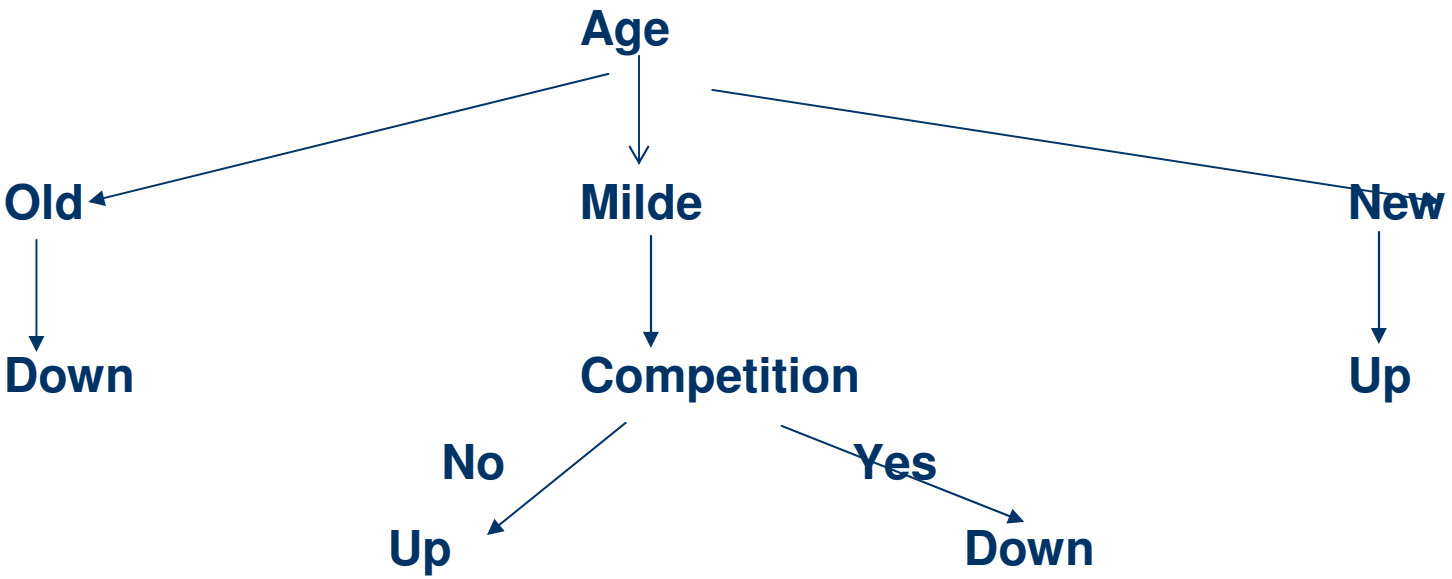
  E(C/Age) = 0.4

  E(C/Type) = 1
- Age cho nhiều thông tin nhất

STT	Competition	Type	Profit
1	Yes	Software	Down
2	No	Hardware	Up
3	No	Software	Up
4	Yes	Hardware	Down



Học theo độ bất định(tiếp)



Rule 1: If (Age IS Old) then (Profit IS Down)

Rule 2: If (Age IS New) then (Profit IS Up)

Rule 3: If (Age IS Midle) And (Competition IS No)
then (Profit IS Up)

Rule 4: If (Age IS Midle) And (Competition IS Yes)
then (Profit IS Down)