ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH

BÀI GIẢNG
TRÍ TUỆ NHÂN TẠO

Trương Hải Bằng, PhD

NỘI DUNG

1. **Tổng quan về Trí tuệ nhân tạo**
2. **Thuật toán và thuật giải .**
3. **Biểu diễn và xử lý tri thức .**
4. **Giới thiệu về máy học.**

Tổng quan về Trí tuệ nhân tạo

1. **Đối tượng và mục tiêu nghiên cứu của trí tuệ nhân tạo.**
2. **Vai trò của Trí Tuệ Nhân Tạo.**
3. **Các kỹ thuật Trí tuệ nhân tạo**
4. **Các khái niệm cơ bản**

Đối tượng và mục tiêu nghiên cứu của trí tuệ nhân tạo.

Trí tuệ nhân tạo nghiên cứu về cách hành xử thông minh (intelligent behaviour) với mục tiêu là xây dựng lý thuyết đầy đủ về thông minh để có thể giải thích được hoạt động thông minh của sinh vật và áp dụng được các hiểu biết vào các máy móc nói chung, nhằm phục vụ cho con người.

Đối tượng và mục tiêu nghiên cứu của trí tuệ nhân tạo *(tt)*

Theo Winton: mục đích chính của trí tuệ nhân tạo là hướng tới việc xây dựng các máy tính thông minh hơn, giúp ích cho việc khám phá các quy luật hoạt động sáng tạo và khả năng trí tuệ của con người.

Vai trò của Trí Tuệ Nhân Tạo

–Trí tuệ nhân tạo bao quát rất nhiều lĩnh vực nghiên cứu. Nó nghiên cứu từ các lĩnh vực tổng quát như máy nhận biết, suy luận logic, đến các bài toán như chơi cờ, chứng minh định lý.

–Trong các lĩnh vực khác trí tuệ nhân tạo được dùng kỹ thuật hệ thống hoá và tự động hoá các xử lý tri thức cũng như các phương pháp thuộc lĩnh vực mang tính con người.

Vai trò của Trí Tuệ Nhân Tạo *(tt)*

Trí tuệ nhân tạo nghiên cứu kỹ thuật làm cho máy tính có thể "suy nghĩ một cách thông minh" và mô phỏng quá trình suy nghĩ của con người khi đưa ra những quyết định, lời giải. Trên cơ sở đó, thiết kế các chương trình cho máy tính để giải quyết bài toán.

Các kỹ thuật Trí tuệ nhân tạo.

- **Lý thuyết giải bài toán và suy diễn thông minh ;**
- **Lý thuyết tìm kiếm may rủi;**
- **Các ngôn ngữ về Trí tuệ nhân tạo ;**
- **Lý thuyết thể hiện tri thức và hệ chuyên gia;**
- **Lý thuyết nhận dạng và xử lý tiếng nói;**
- **Người máy;**
- …

Các khái niệm cơ bản

Trí tuệ con người (Human Intelligence): Cho đến nay có hai khái niệm về trí tuệ con người được chấp nhận và sử dụng nhiều nhất, đó là:

Khái niệm trí tuệ theo quan điểm của Turing

"Trí tuệ là những gì có thể đánh giá được thông qua các trắc nghiệm thông minh"

Các khái niệm cơ bản *(tt)*

Khái niệm trí tuệ đưa ra trong tụ điển bách khoa toàn thư:

Trí tuệ là khả năng:

"*Phản ứng một cách thích hợp những tình huống mới thông qua hiệu chỉnh hành vi một cách thích đáng.*

Hiểu rõ những mối liên hệ qua lại của các sự kiện của thế giới bên ngoài nhằm đưa ra những hành động phù hợp đạt tới một mục đích nào đó".

Các khái niệm cơ bản *(tt)*

Trí tuệ máy: cũng không có một định nghĩa tổng quát, nhưng cũng có thể nêu các đặc trưng chính:

- Khả năng học.
- Khả năng mô phỏng hành vi của con người.
- Khả năng trừu tượng hoá, tổng quát hoá và suy diễn .
- Khả năng tự giải thích hành vi.

Các khái niệm cơ bản *(tt)*

- Khả năng thích nghi tình huống mới kể cả thu nạp tri thức và dữ liệu.
- Khả năng xử lý các biểu diễn hình thức như các ký hiệu tượng trưng.
- Khả năng sử dụng tri thức heuristic.
- Khả năng xử lý các thông tin không đầy đủ, không chính xác

THUẬT TOÁN, THUẬT GIẢI
MỘT SỐ PHƯƠNG PHÁP GIẢI QUYẾT VẤN ĐỀ

Nội dung

- **Vấn đề, giải quyết vấn đề**
- **Khái niệm về thuật toán, thuật giải**
- **Các nguyên lý của thuật giải heuristic**
- **Các chiến lược tìm kiếm và Thuật giải AT,AKT, A***

Vấn đề?

Những vướng mắc khó khăn cần giải quyết

Một yêu cầu tìm kiếm xử lý trong một ngữ cảnh cụ thể

Bao gồm:

- các sự kiện;
- các thông tin ;
- những ràng buộc nhất định.

vấn đề = bài toán

Mô hình vấn đề

A ➔ B

A: giả thiết, điều kiện ban đầu

B: kết luận cần đạt đến

➔: suy luận hay giải pháp cần xác định = một số hữu hạn bước

Phân loại vấn đề

Xác định rõ

- A, B đều rõ

Chưa rõ

- A rõ, B chưa rõ
- A chưa rõ, B rõ
- A, B đều chưa rõ

Thuật toán

Thuật toán:

Là chuỗi hữu hạn các công việc trình tự xác định các thao tác để giải các bài toán.

Tính chất:

1)Tính xác định.

2)Tính đúng đắn.

3)Tính dừng

Thuật toán

Thuật toán có thể được thể hiện qua:

Ngôn ngữ tự nhiên

Lưu đồ

Mã giả

NN lập trình

Ngoài ra thuật toán còn phải đạt hiệu quả cao hay có độ phức tạp thấp

Thuật toán

$O(\log_2 n)$
$O(n)$
$O(n\log_2 n)$
$O(n^2)$
$O(n^k)$

Độ phức tạp đa thức $\Rightarrow$ chấp nhận được

$O(2^n)$
$n!$

Độ phức tạp cao $\Rightarrow$ khả chấp nhận

Một số ví dụ về bài toán có độ phức tạp cao

Bài toán phân công công việc

Một đề án gồm n công việc và các việc sẽ được thực hiên bởi m máy như nhau.

Giả sử biết thời gian để 1 máy thực hiện viêc thứ j là tj.

Yêu cầu: Tìm phương án phân công sao cho thời gian hoàn thành toàn bộ công việc là thấp nhất.

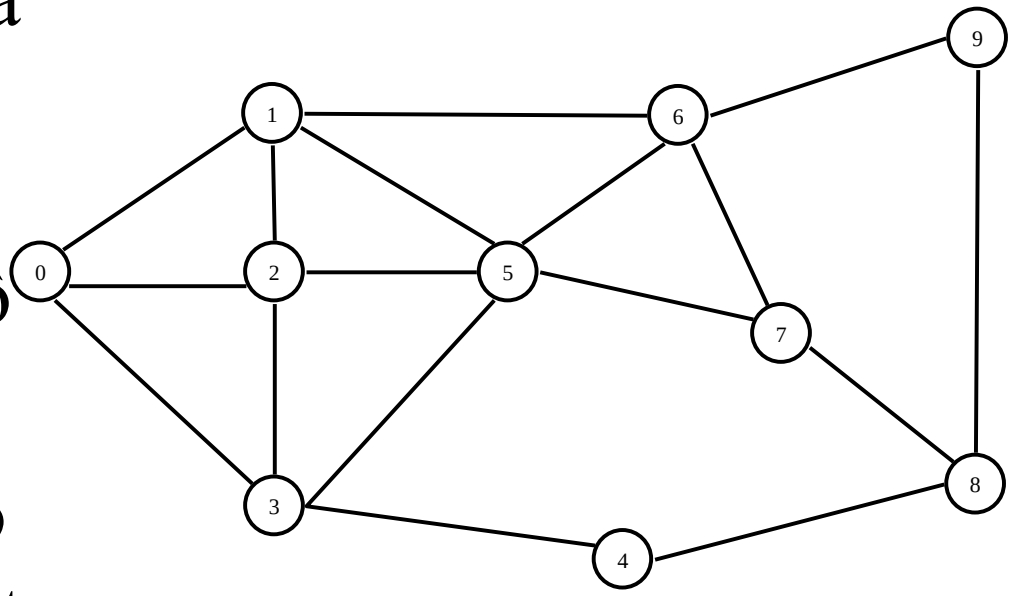
Mẫu số liệu: n=10, m=3, tj = 4 9 5 2 7 6 10 8 7 5

Một số ví dụ về bài toán có độ phức tạp cao

Bài toán tô màu

Giả sử ta có bản đồ các quốc gia trên thế giới, ta muốn tô màu các quốc gia này sao cho các nước khác nhau được tô khác màu.

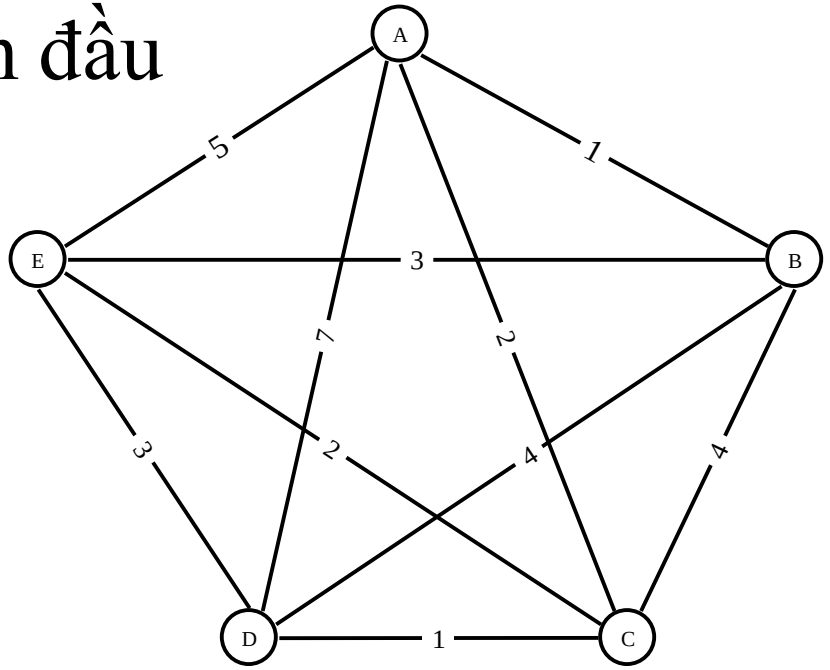
Yêu cầu tìm cách tô sao cho số màu sử dụng là ít nhất.



Một số ví dụ về bài toán có độ phức tạp cao

Bài toán người đưa thư

Giả sử có một đồ thị có trọng số dương, tìm đường đi nhắn nhất qua tất cả các đỉnh của đồ thị rồi trở về đỉnh ban đầu



Thuật giải

Thuật giải:

Giải pháp được viết dưới dạng thủ tục tương tự như thuật toán nhưng không đòi hỏi các tiêu chuẩn như thuật toán.

Tính đúng: chấp nhận các thuật giải đơn giản có thể cho kết quả đúng hay gần đúng nhưng có khả năng thành công cao hơn.

Thuật giải *(tt)*

Để có thể được chấp nhận thuật giải phải thể hiện một giải pháp hợp lý nhất có thể trong tình huống hiện tại bằng cách:

- –Tận dụng mọi thông tin hữu ích
- –Sử dụng tri thức, kinh nghiệm trực giác của con người
- –Tự nhiên đơn giản nhưng cho kết quả chấp nhận được

Thuật giải Heuristic:

Là mở rộng khái niệm thuật toán.

- Thường tìm lời giải tốt nhưng không tốt nhất.
- Nhanh chóng tìm ra kết quả hơn so với giải thuật tối ưu, vì vậy chi phí thấp hơn.
- Thường thể hiện khá tự nhiên, gần gũi với cách suy nghĩ và hành động của con người.

Các nguyên lý của thuật giải heuristic

Vét cạn thông minh

Nguyên lý thứ tự

Nguyên lý tham lam

Hàm heuristic

Kỹ thuật Heuristics

Theo Từ điển tiếng Anh Oxford: “Heuristics là nghệ thuật tìm kiếm chân lý. Nói riêng, heuristics là đặc trưng của quá trình học nhờ đó các học sinh học được cách tự tìm ra cách giải thích các hiện tượng tự nhiên”.

Từ “Heuristics” có cùng một gốc tiếng Hy Lạp như từ Eureka. Feigenbaum Feldman đã đưa ra định nghĩa :

“Heuristics (Các quy tắc heuristics, các phương pháp heuristics) là các quy tắc, phương pháp, chiến lược, mẹo giải hay phương cách nào đó nhằm làm giảm khối lượng tìm kiếm lời giải trong không gian bài tóan cực lớn”.

Các nguyên lý của thuật giải heuristic

1.Vét cạn thông minh

Hạn chế vùng không gian tìm kiếm và có sự định hướng để nhanh chóng tìm đến mục tiêu.

Tạo miền D’ rất nhỏ so với D

Vét cạn trên D’

Các nguyên lý của thuật giải heuristic

2.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.

a)Thuật giải GTS1: (Greedy-Traveling Saleman)

Xây dựng một lịch trình du lịch có chi phí Cost tối thiểu cho bài toán trong trường hợp phải qua n thành phố với ma trận chi phí C và bắt đầu tại một đỉnh U nào đó.

Các nguyên lý của thuật giải heuristic

Thuật giải:
Bước 1: {Khởi đầu}
Đặt Tour := {};
Cost := 0;
V := U; {V là đỉnh hiện tại đang làm việc}
Bước 2: {Thăm tất cả các thành phố}
For k := 1 To n Do
qua bước 3;

Bước 3: {Chọn cung kế tiếp}
Đặt (V, W) là cung có chi phí nhỏ nhất tình từ V đến các đỉnh W chưa dùng:
Tour := Tour + {(V,W)};
Cost := Cost + Cost(V,W);
Nhãn W được sử dụng
Đặt V := W; {Gán để xét bước kế tiếp}
Bước 4: {Chuyến đi hoàn thành}
Đặt Tour := Tour + {(V,U)};
Cost := Cost + Cost(V,U);
Dừng.

Ví dụ :

$$C = \begin{bmatrix} \infty & 1 & 2 & 7 & 5 \\ 1 & \infty & 4 & 4 & 3 \\ 2 & 4 & \infty & 1 & 2 \\ 7 & 4 & 1 & \infty & 3 \\ 5 & 3 & 2 & 3 & \infty \end{bmatrix}$$

U = A

Tour = {}

Cost = 0

V = A

W $\in$ {B, C, D, E}{Các đỉnh có thể đến từ A}

$\rightarrow$ W = B{Vì qua B có giá thành bé nhất}

Tour = {(A, B)}

Cost = 1

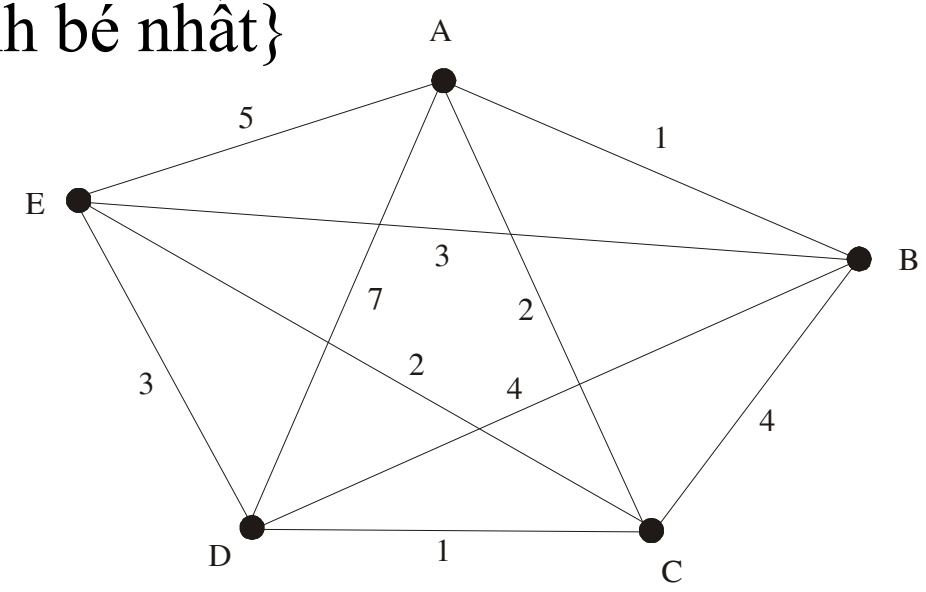
V = B

W $\in$ {C, D, E}$\rightarrow$ W = E

Tour = {(A, B),(B, E)}

Cost = 1 + 3 = 4

V = E



Ví dụ :

Tour = {(A, B), (B, E), (E, C)}

Cost = 4 + 2 = 6

V = C

$W \in \{D\}$

$\rightarrow W = D$

Tour = {(A, B), (B, E), (E, C), (C, D)}

Cost = 6 + 1 = 7

V = D

Tour = {(A, B), (B, E), (E, C), (C, D), (D, A)}

Cost = 7 + 7 = 14

Kết quả:Tour du lịch $A \rightarrow B \rightarrow E \rightarrow C \rightarrow D \rightarrow A$ với giá thành Cost = 14.

Nhận xét: Tuy nhiên kết quả nhỏ nhất sẽ là $A \rightarrow B \rightarrow D \rightarrow C \rightarrow E \rightarrow A$ với Cost=13. Sở dĩ không tối ưu do "háu ăn": cứ hướng nào có chi phí thấp thì đi, bất chấp về sau.

b)Thuật giải GTS2:

Tạo ra lịch trình từ p thành phố xuất phát riêng biệt. Tìm chu trình của người bán hàng qua n thành phố (1<p<n) và p chu trình được tạo ra và chỉ chu trình tốt nhất trong p chu trình được giữ lại mà thôi (thuật giải này đòi hỏi phải nhập n, p và C)

Thuật giải:

Bước 1: {Khởi đầu}

k := 0; {Đếm số thành phố đi qua}

Best := {}; {Ghi nhớ chu trình tốt nhất tìm thấy có chi phí là Cost}

Cost := ∞;

b)Thuật giải GTS2:(tt)

Bước 2: {Bắt đầu chu trình mới}

Chuyển qua bước 3 khi k<p, ngược lại dừng.

Bước 3: {Tạo chu trình mới}

k := k + 1;

Call (GTS1(Vk)) : Trả về một chu trình T(k) ứng với chi phí C(k).

Bước 4: {Cập nhật chu trình tốt nhất}

Nếu C(k)< Cost thì

Best := T(k);

Cost := C(k);

b)Thuật giải GTS2:(tt)

Ví dụ: (Giải lại ví dụ trên) p=3

$$C = \begin{bmatrix} \infty & 25 & 40 & 31 & 27 \\ 5 & \infty & 17 & 30 & 25 \\ 19 & 15 & \infty & 6 & 1 \\ 9 & 30 & 24 & \infty & 6 \\ 22 & 8 & 7 & 10 & \infty \end{bmatrix}$$

k=0 Best={} Cost=∞

k=0 < p V_0=A Call(GTS1(A))

=>T(0) = {(A,B),(B,E),(E,C),(C,D),(D,A)},

C(0) = 14 < Cost =>Best = T(0) Cost = 14

k=1 < p V_1=B Call(GTS1(B))

=>T(1) = {(B,A),(A,C),(C,D),(D,E),(E,B)},

C(1) = 10 < Cost=>Best = T(1) Cost = 10

k=2 < p V_2 = C Call(GTS1(C))

=>T(2) = {(C,D),(D,E),(E,B),(B,A),(A,C)},

C(2) = 10 >= Cost

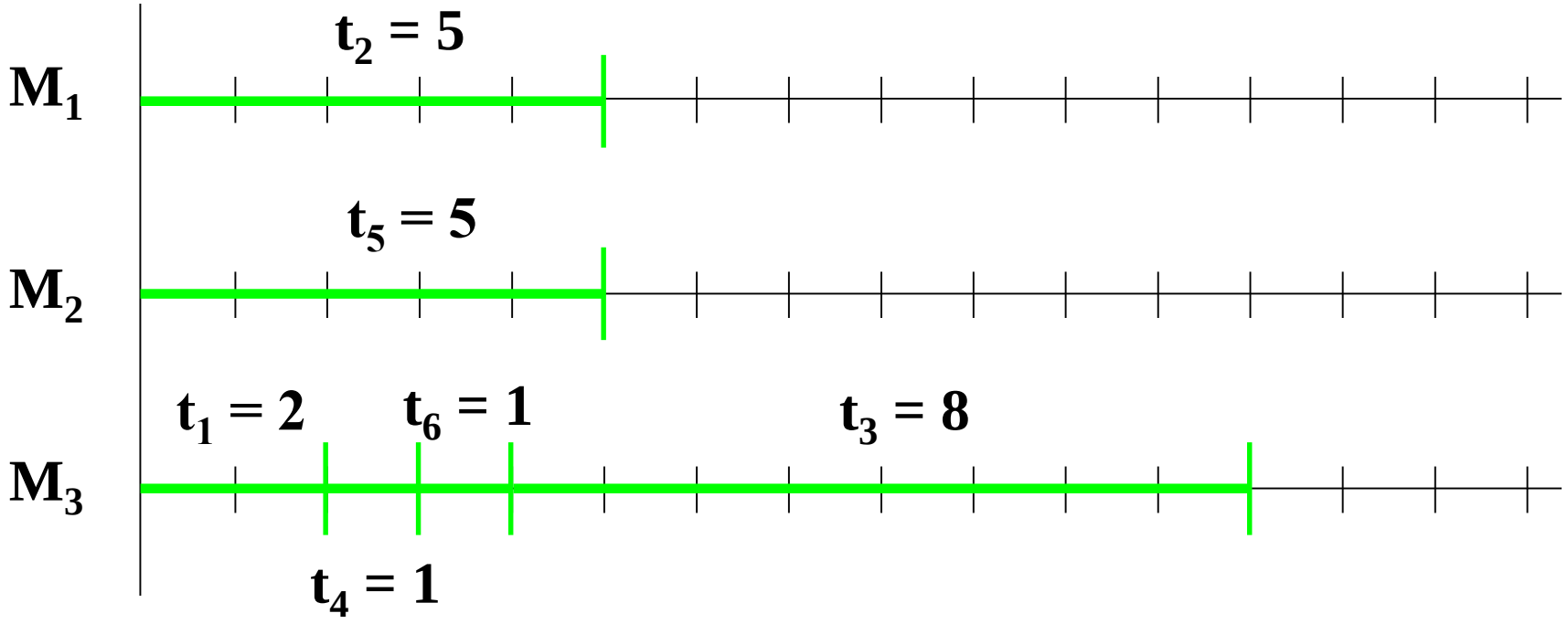
k=3 >= p Dừng.

3.Nguyên lý thứ tự

Bài toán phân công : Cho M máy có cùng công suất như nhau và n công việc . Thực hiện công việc i trên bất kỳ máy nào cũng tốn thời gian là ti. Hãy phân công các công việc trên các máy sao cho tổng thời gian để hoàn thành tất cả công việc là thấp nhất**.**

Ví dụ:

Có 3 máy M1, M2, M3 và 6 công việc t1 = 2, t2 = 5, t3 = 8, t4 = 1, t5 = 5, t6 = 1.



Nhận xét độ phức tạp

Thứ tự của các công việc trên một máy là không quan trọng *(vì không làm thay đổi tổng thời gian thực hiện trên máy đó)*

Công việc	**1**	**2**	**3**	**4**	**...**	**n**
Máy	**1**	**1**	**3**	**2**	**...**	

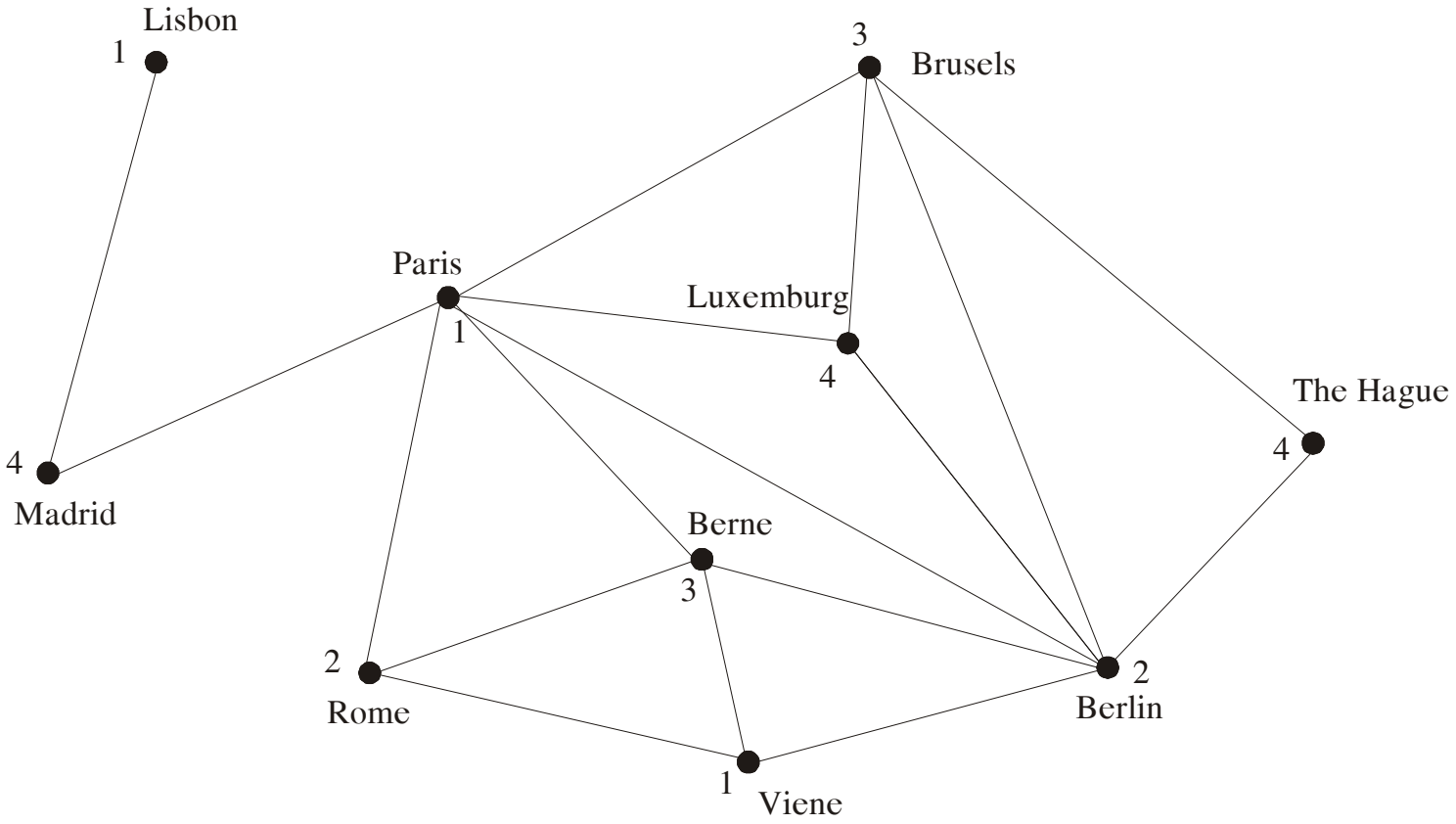
Độ phức tạp : $O(M^n)$

4.Thuật toán tô màu tối ưu trên đồ thị

Giả thiết 4 màu: Chúng ta nói 2 nước trên bản đồ vẽ trên mặt cầu hoặc là mặt phẳng là láng giềng của nhau nếu như chúng có chung đường biên giới (chỉ xét những nước có đường biên giới là một đường cong khép kín).

Yêu cầu: tô toàn bộ bản đồ mà chỉ sử dụng 4 màu sao cho không có bất kỳ 2 nước láng giềng nào có cùng chung một màu

Đỉnh	Lisbon L	Madrid M	Paris P	Berne Be	Rome R	Viene V	Berlin Ber	Luxemburg Lx	Brusen Bru	Hague H
Bậc	1	2	6	4	3	3	6	3	4	2



4.Thuật toán tô màu tối ưu trên đồ thị (tt)

Thuật toán: Lặp lại các bước sau cho đến khi nào tô màu hết các đỉnh

Bước 1: Chọn đỉnh có bậc lớn nhất tô màu i.

Bước 2: Hạ bậc:

- Đỉnh đã tô màu: bậc = 0
- Những đỉnh có liên hệ: bậc := bậc – 1

Bước 3: Đánh dấu các đỉnh liên hệ (bậc vừa trừ đi 1) cấm tô màu i.

4. Thuật toán tô màu tối ưu trên đồ thị (tt)

Màu tô

Màu tô lần 7 (Các nước có bậc là 0 mà chưa tô màu)		**2**				**1**		**4**		**1**
Màu tô lần 6				**3**		**3**				
Màu tô lần 5	**1**	**1**								
Màu tô lần 4								**3**	**3**	**3**
Màu tô lần 3				**2**	**2**	**2**				
Màu tô lần 2				**2**		**2**	**2**	**2**	**2**	**2**
Màu tô lần 1		**1**	**1**	**1**	**1**		**1**	**1**	**1**	
Đỉnh	**L**	**M**	**P**	**Be**	**R**	**V**	**Ber**	**Lx**	**Bru**	**H**
Bậc	**1**	**2**	**6***	**4**	**3**	**3**	**6**	**3**	**4**	**2**
Hạ bậc lần 1	**1**	**1**	**0**	**3**	**2**	**3**	**5***	**2**	**3**	**2**
Hạ bậc lần 2	**1**	**1**	**0**	**2**	**2***	**2**	**0**	**1**	**2**	**1**
Hạ bậc lần 3	**1**	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**2***	**1**
Hạ bậc lần 4	**1***	**1**	**0**	**1**	**0**	**1**	**0**	**0**	**0**	**0**
Hạ bậc lần 5	**0**	**0**	**0**	**1***	**0**	**1**	**0**	**0**	**0**	**0**
Hạ bậc lần 6	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

Ví dụ 2: Phân công, lịch công tác, lịch thi đấu:

- Có một cuộc hội thảo khoa học với 9 chủ đề khác nhau, mỗi chủ đề diễn ra trong một buổi.
- Các chủ đề sau không được đồng thời: AE, BC, CD, ED, ABD, AHI, BHI, DFI, DHI, FGH.
- Xây dựng lịch sao cho số buổi diễn ra là ít nhất.
- Gợi ý: số màu = số buổi.

AE, BC, CD, ED, ABD, AHI, BHI, DFI, DHI, FGH.

	-1	-1	-1	1	-1	-1		-1	-1
dinh	A	B	C	D	E	F	G	H	I
bac	5	5	2	7	2	4	2	6	5
	4	4	1	0	1	3	2	5	4

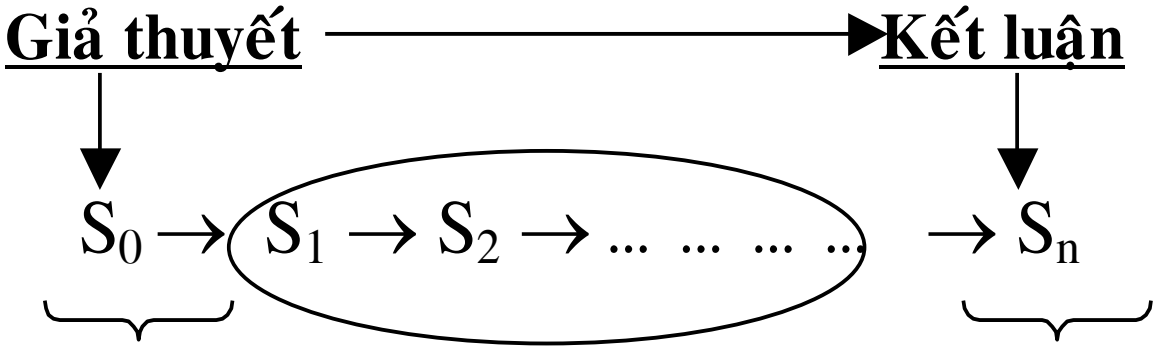
Hàm heuristic

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.

CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

Vấn đề = Tìm kiếm mục tiêu

Biểu diễn bài toán



START

Trạng thái bắt đầu

GOAL

Trạng thái kết thúc

Biểu diễn bài toán

Hầu hết các bài toán đều có thể phát biểu dưới dạng sau: từ một trạng thái xuất phát hãy tìm đường dẫn đến một trạng thái kết thúc mong muốn. Việc tìm đường đi này là một nghệ thuật để giải quyết vấn đề, bao gồm các bước sau:

Chọn được không gian tìm kiếm thích hợp.

Tiến hành tìm kiếm có hệ thống và có hiệu quả trong không gian tìm kiếm.

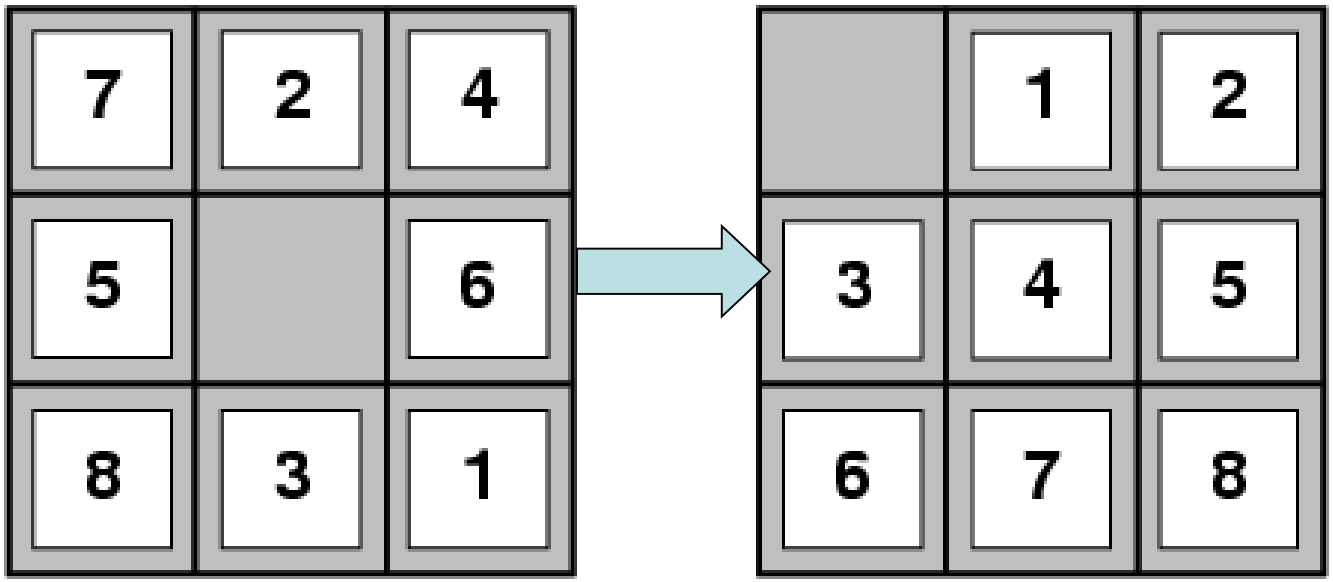
Sử dụng triệt để các nguồn tri thức có liên quan trong quá trình tìm kiếm tương ứng với miền đại lượng cụ thể.

Biểu diễn bài toán

Không gian tìm kiếm của một vấn đề giải trên máy tính thường được biểu diễn bởi một đồ thị hoặc một dạng đặc biệt của đồ thị (cây). Sau khi bài toán được biểu diễn dưới dạng đồ thị (hoặc cây) thì:

- Mỗi đỉnh là một giai đoạn của quá trình giải (hay là trạng thái).
- Mỗi cung là một tác động biến đổi quá trình từ giai đoạn này sang giai đoạn khác.

Bài toán Taci



Các đặc điểm của bài toán

Khả năng phân rã ?

Khả năng lờ đi và quay lui.

Khả năng dự đoán toàn cục.

Mục tiêu là trạng thái hay con đường.

Lượng tri thức cần để giải bài toán.

Có cần sự can thiệp của con người trong quá trình giải không?

Khả năng lờ đi và quay lui

Có thể lờ đi : như BT chứng minh định lý.

Vì: định lý vẫn đúng sau một vài bước áp dụng các luật.

Có thể quay lui: như BT 8-puzzle.

Vì: có thể di chuyển theo hướng ngược lại để về TT trước.

Không thể quay lui: như BT chơi cờ.

Vì: game over!

Các vấn đề trong thiết kế các chương trình tìm kiếm

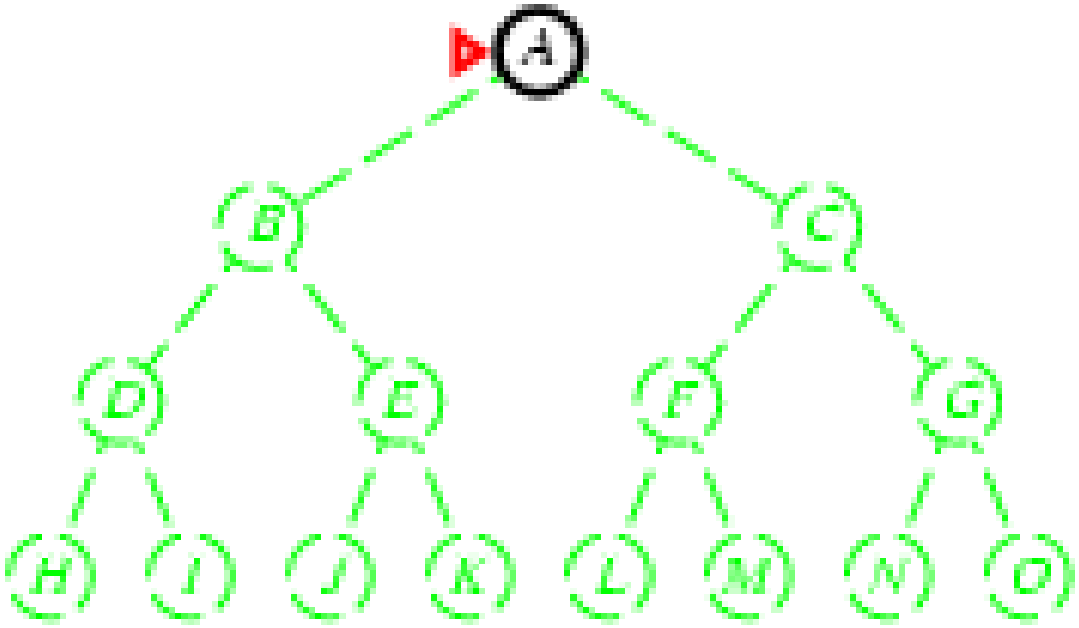
- Xác định hướng tìm (forward hay backward reasoning).
- Cách lựa chọn luật để áp dụng (matching)
- Cách biểu diễn NODE của quá trình tìm:

Các NODE trong đồ thị có thể được phát sinh nhiều lần, và có thể đã được xem xét trước đó trong quá trình duyệt → cần loại bỏ những NODE lặp lại. → cần lưu lại các NODE đã xét.

Các chiến lược tìm kiếm mù

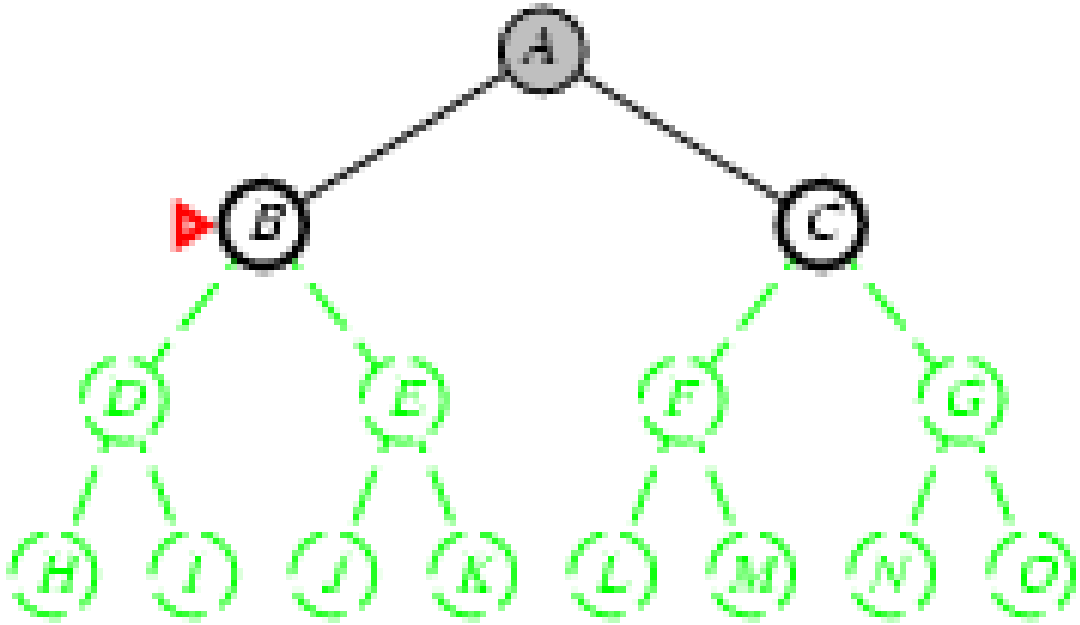
1.Tìm kiếm theo chiều sâu: Depth-first search

Hiện thực: LIFO queue



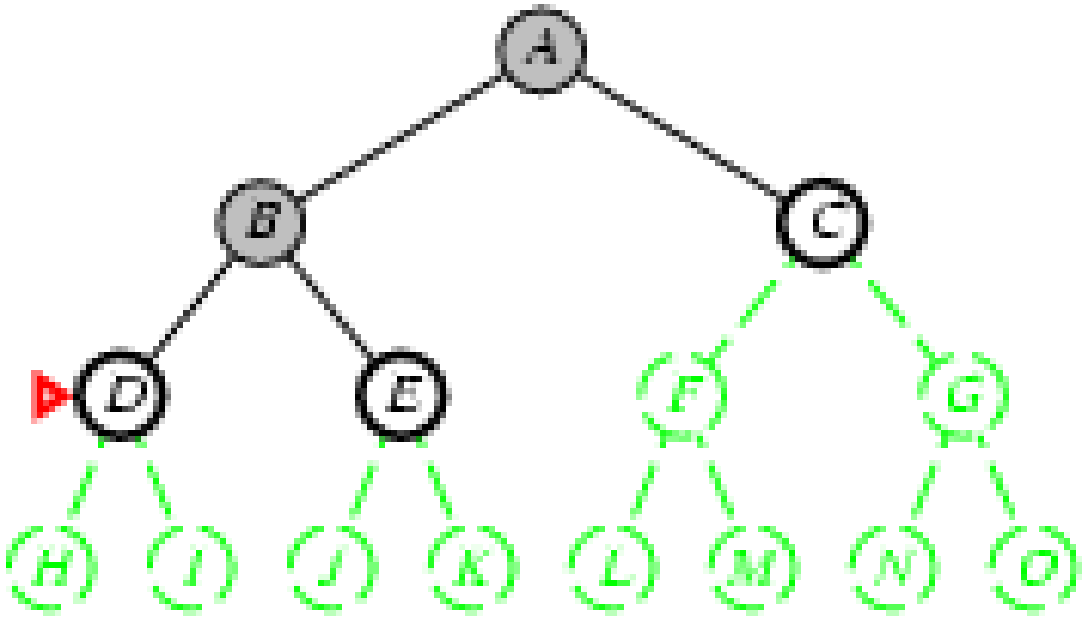
Depth-first search

Hiện thực: LIFO queue



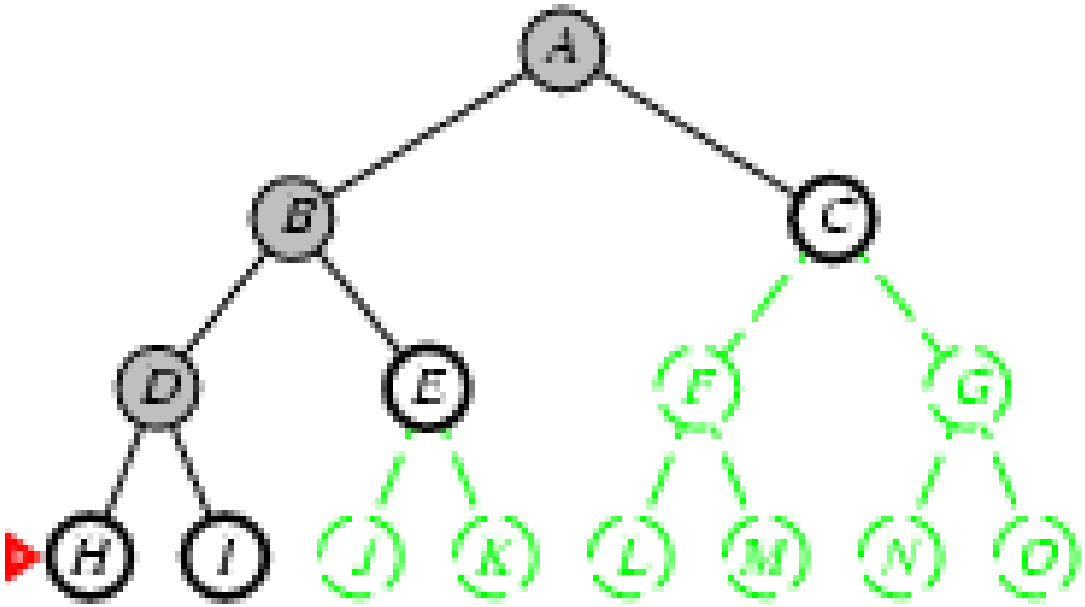
Depth-first search

Hiện thực: LIFO queue



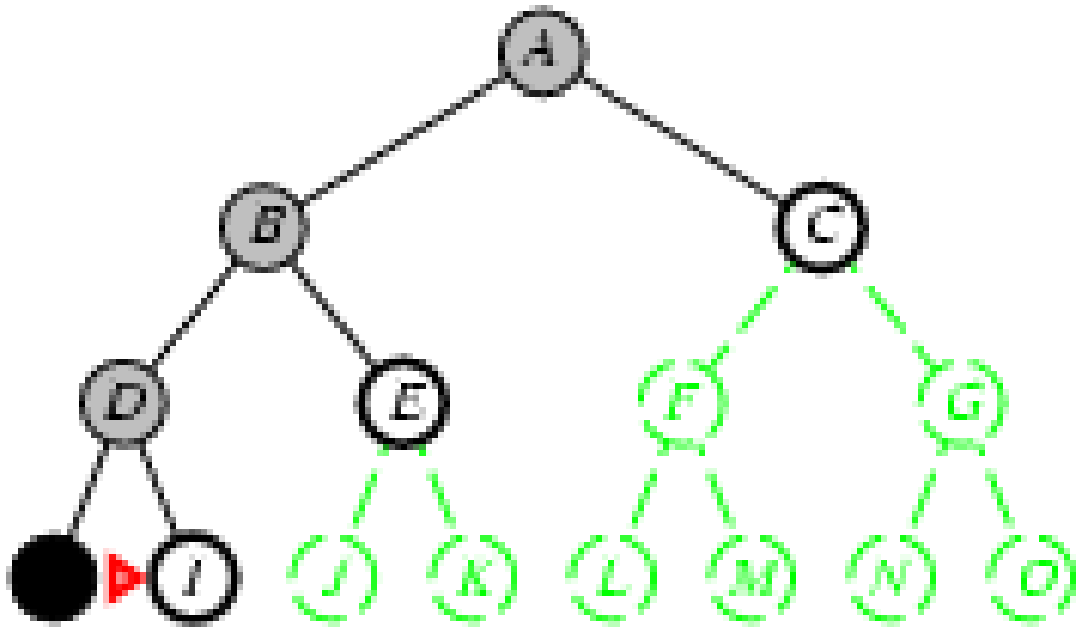
Depth-first search

Hiện thực: LIFO queue



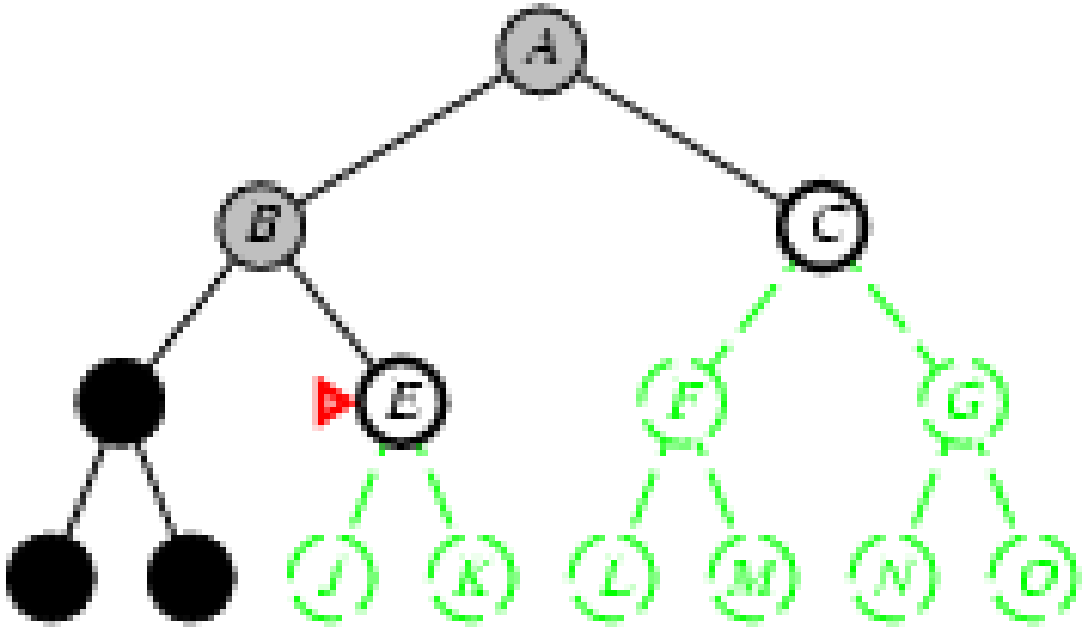
Depth-first search

Hiện thực: LIFO queue



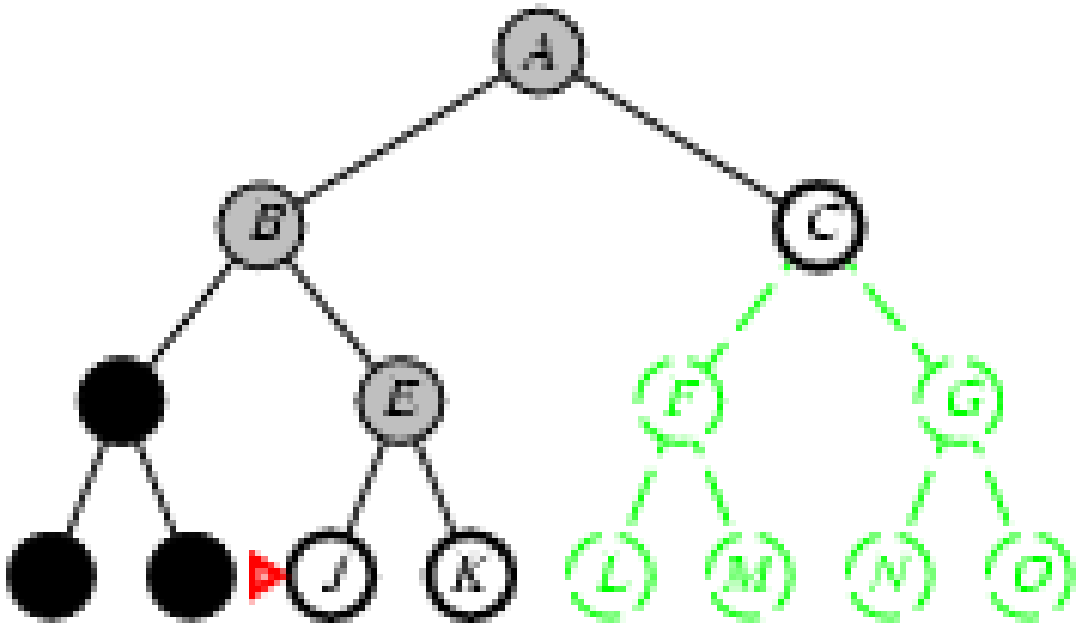
Depth-first search

Hiện thực: LIFO queue



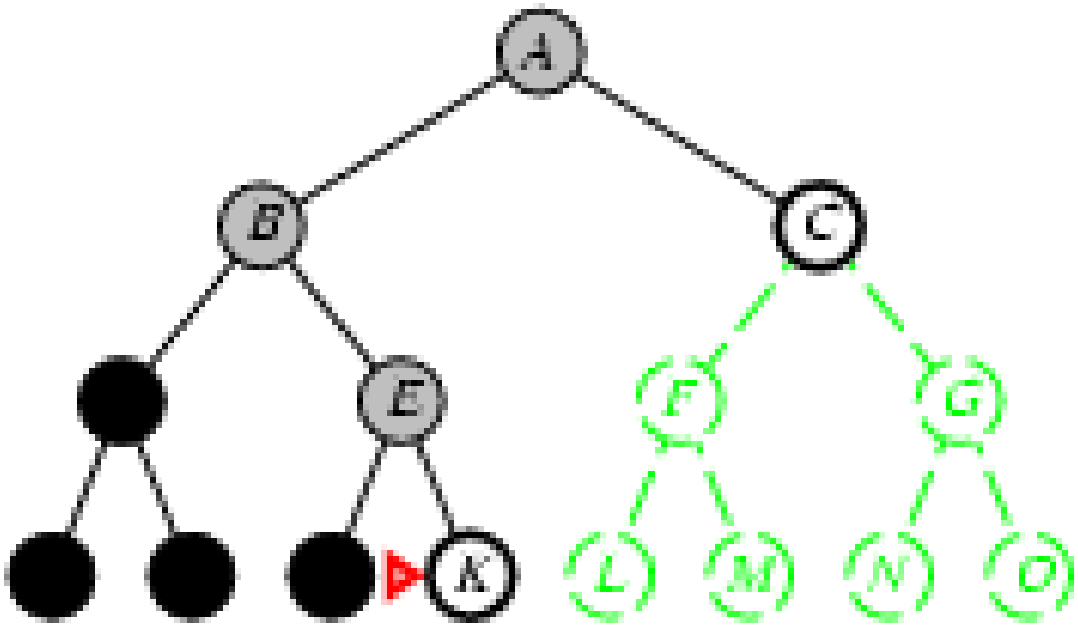
Depth-first search

Hiện thực: LIFO queue



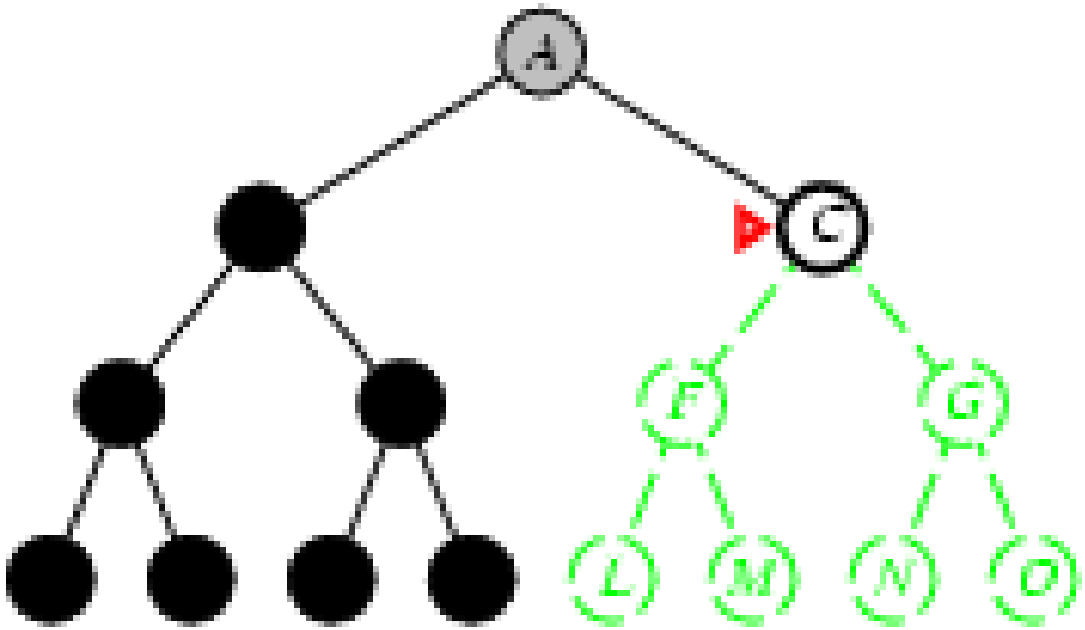
Depth-first search

Hiện thực: LIFO queue



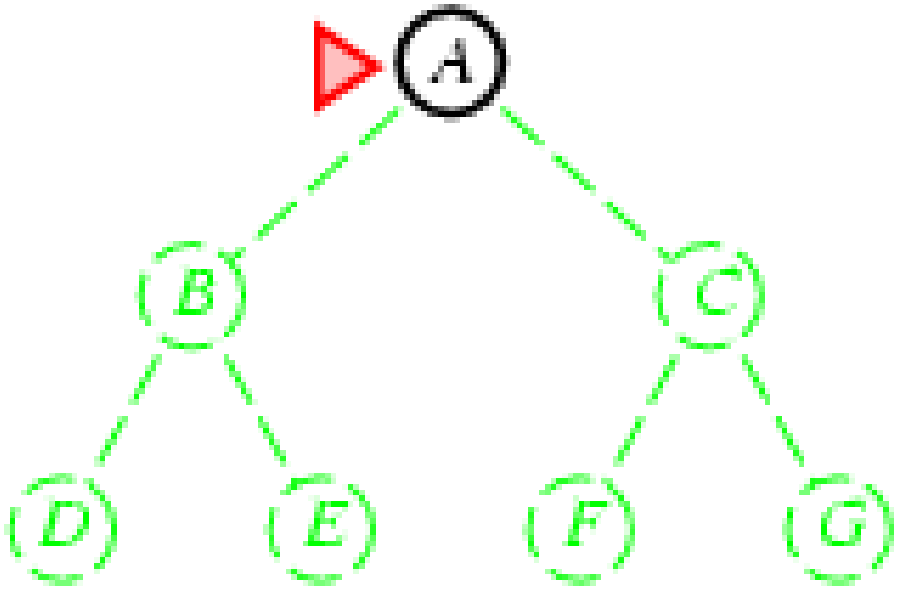
Depth-first search

Hiện thực: LIFO queue



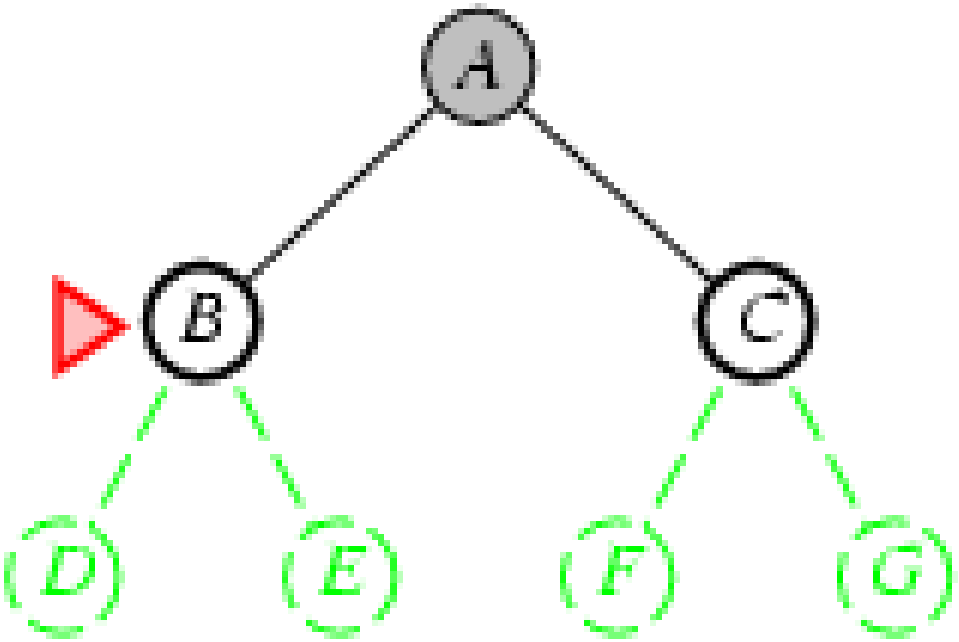
2.Tìm kiếm rộng (Breadth-first search)

Hiện thực: FIFO queue.



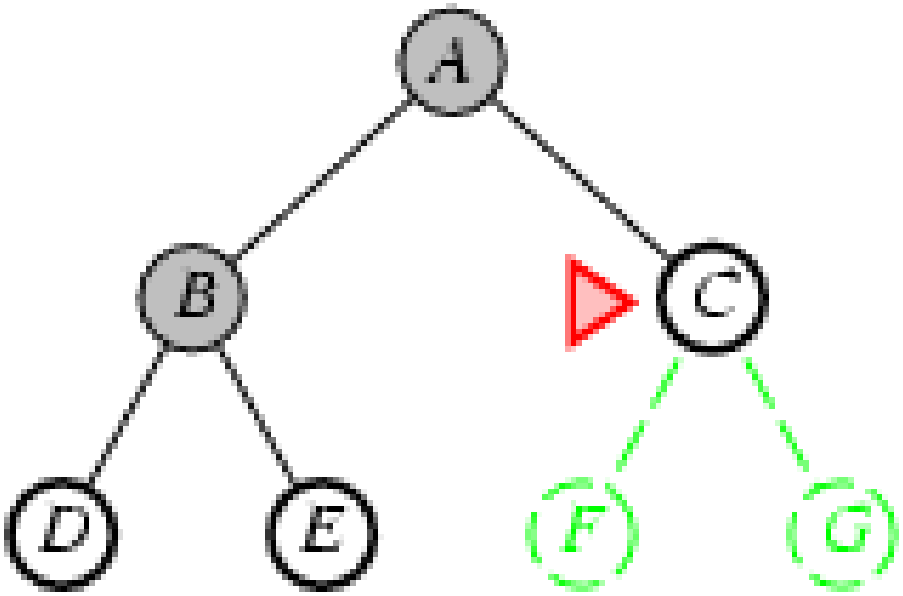
Breadth-first search

Hiện thực: FIFO queue.



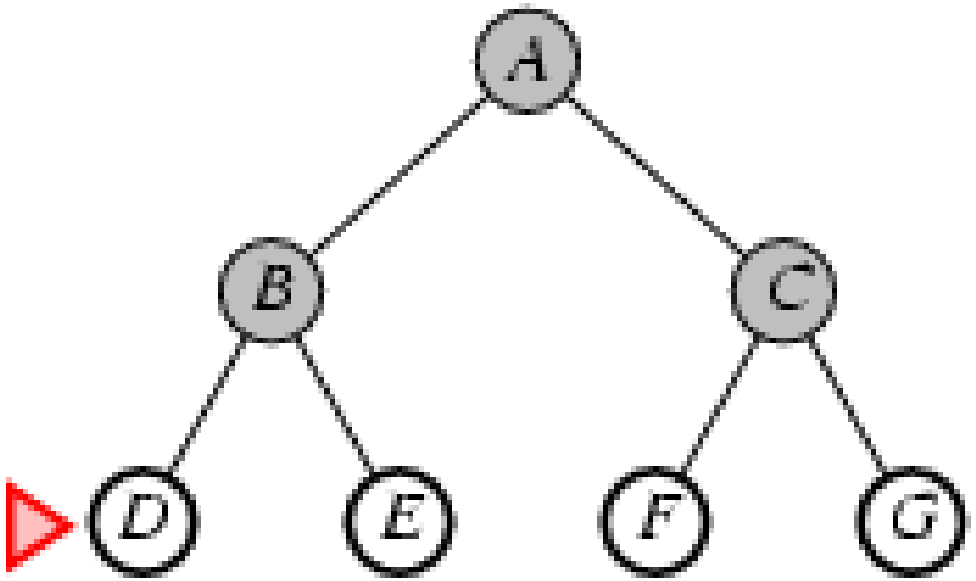
Breadth-first search

Hiện thực: FIFO queue.



Breadth-first search

Hiện thực: FIFO queue.



Tìm kiếm sâu dần
(Iterative deepening search)

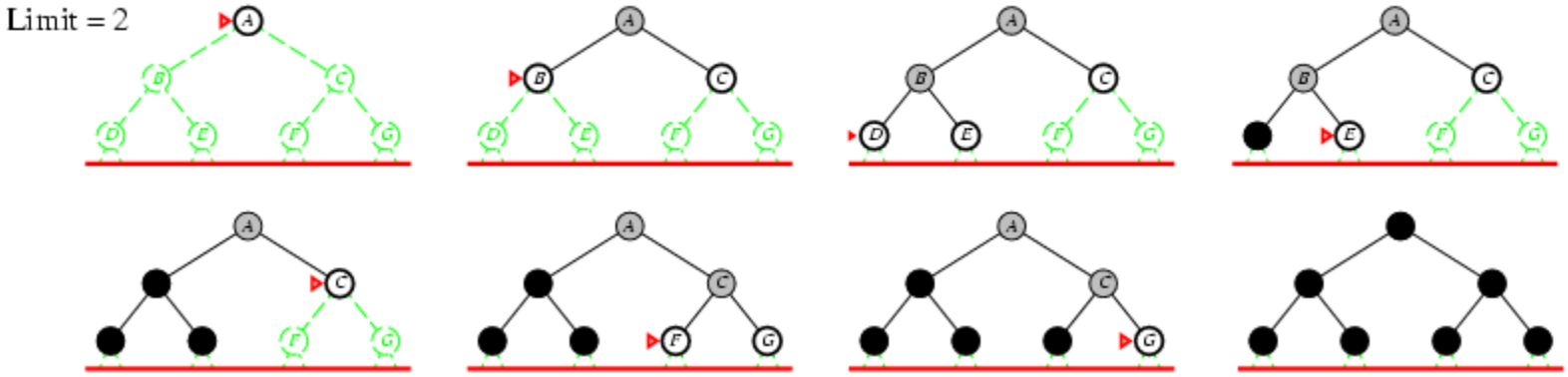
Kết hợp của tìm kiếm rộng và tìm kiếm sâu trên cơ sở cho biết mức sâu n rồi tìm kiếm rộng ứng mới mức sâu đó.

Tìm kiếm sâu dần / =0

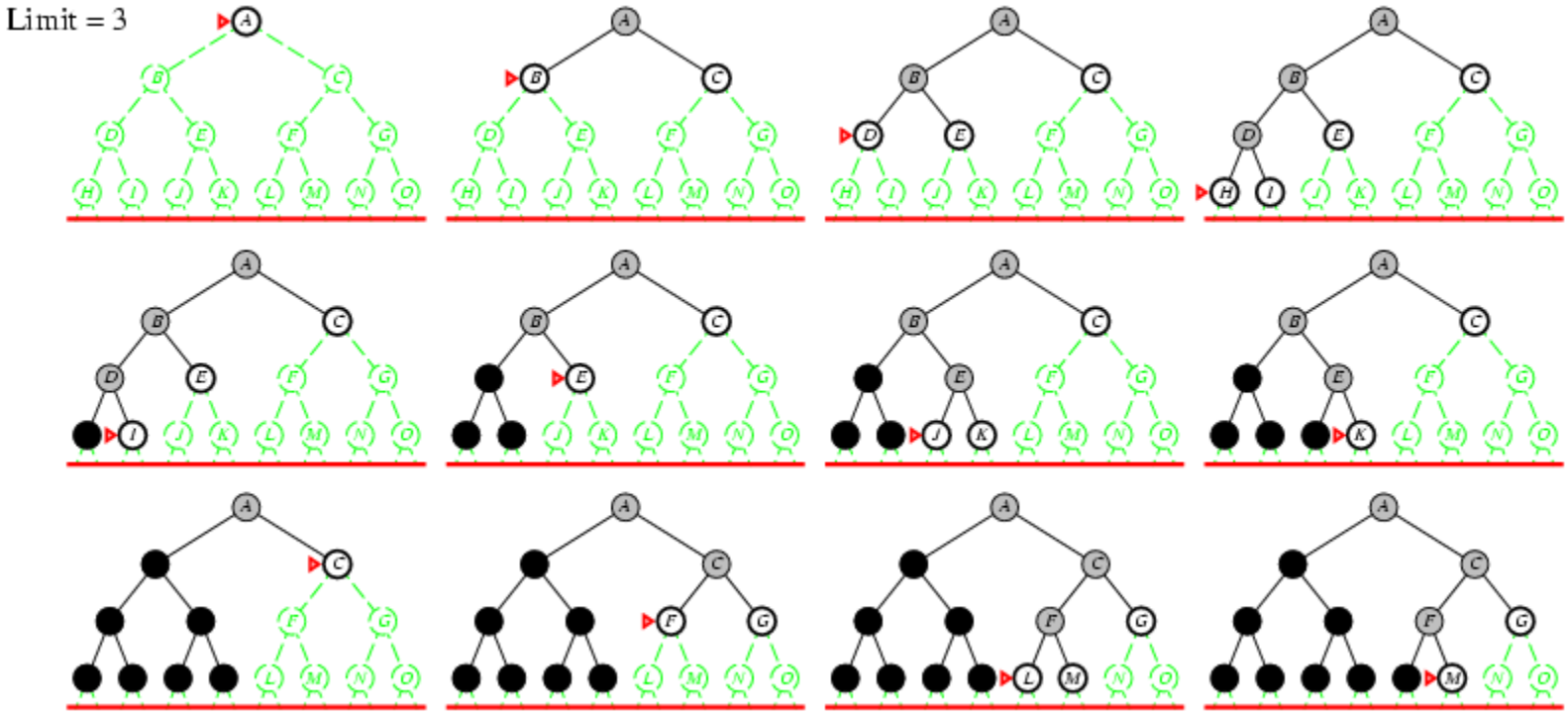
Limit = 0

Tìm kiếm sâu dần / =1

Tìm kiếm sâu dần / =2



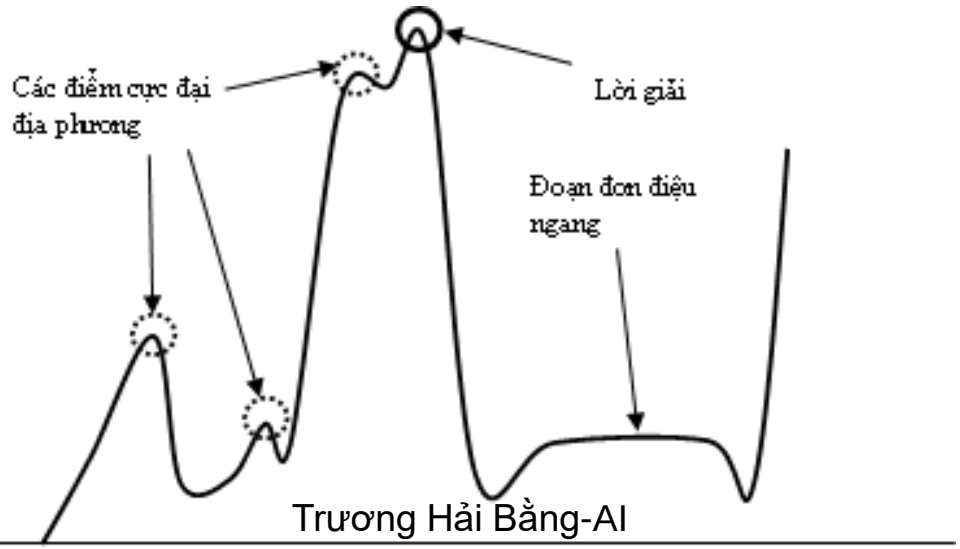
Tìm kiếm sâu dần / =3



Tìm kiếm Heuristics

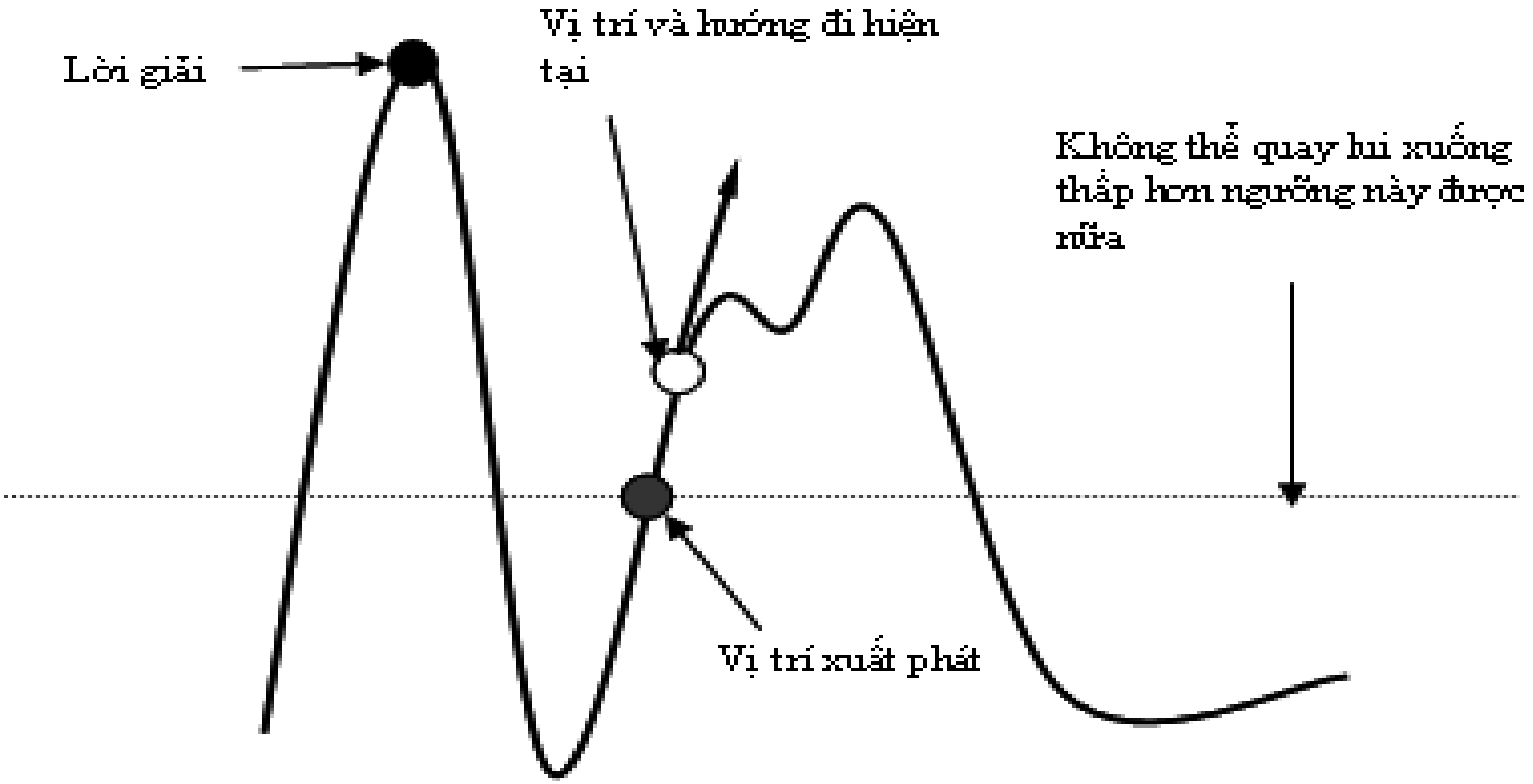
Tìm kiếm leo đồi:

Tìm kiếm leo đồi theo đúng nghĩa, nói chung, thực chất chỉ là một trường hợp đặc biệt của tìm kiếm theo chiều sâu nhưng không thể quay lui. Trong tìm kiếm leo đồi, việc lựa chọn trạng thái tiếp theo được quyết định dựa trên một hàm Heuristic.



Tìm kiếm leo đồi : (tt)

Một trường hợp thất bại của leo đèo kết hợp quay lui



Tìm kiếm leo đồi : (tt)

Bước 1: n:=Startnode;

Bước 2: Nếu n là đích thì dừng (Success).

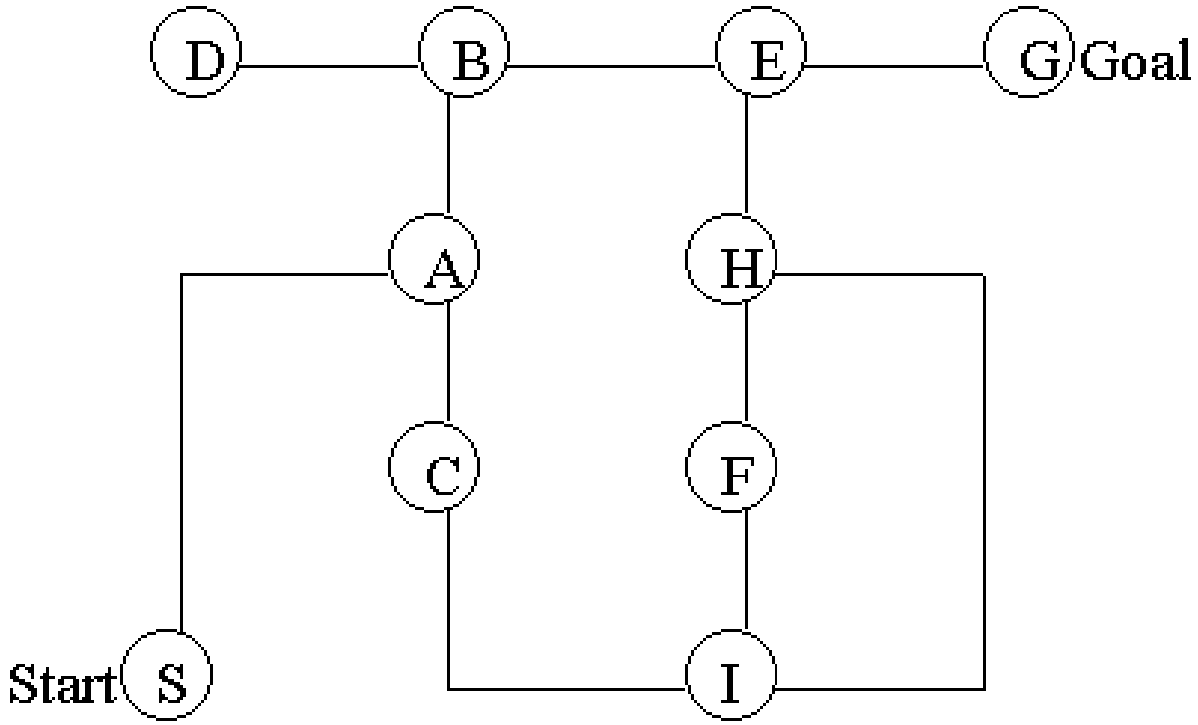
Bước 3: Triển khai n, Tính hàm , với ni là con của n. Chọn ni tương ứng với nhỏ nhất và gọi là nextn.

Bước 4: Nếu thì thoát (Fail).

Bước 5: n:=nextn;

Bước 6: Nhảy sang bước 2.

Vi dụ



H(n)= |Tọa độ x của đích – Tọa độ x của n| + |Tọa độ y của đích – Tọa độ y của n|

h(S)=|4-1|+|4-1|=6

n:=S h(A)=|4-2|+|4-3|=3 < h(S)

NextS = A

n:=A h(B)=|4-2|+|4-4|=2 (min) < h(A)

h(C)=|4-2|+|4-2|=4

NextA = B

n:=B h(D)=|4-1|+|4-4|=3

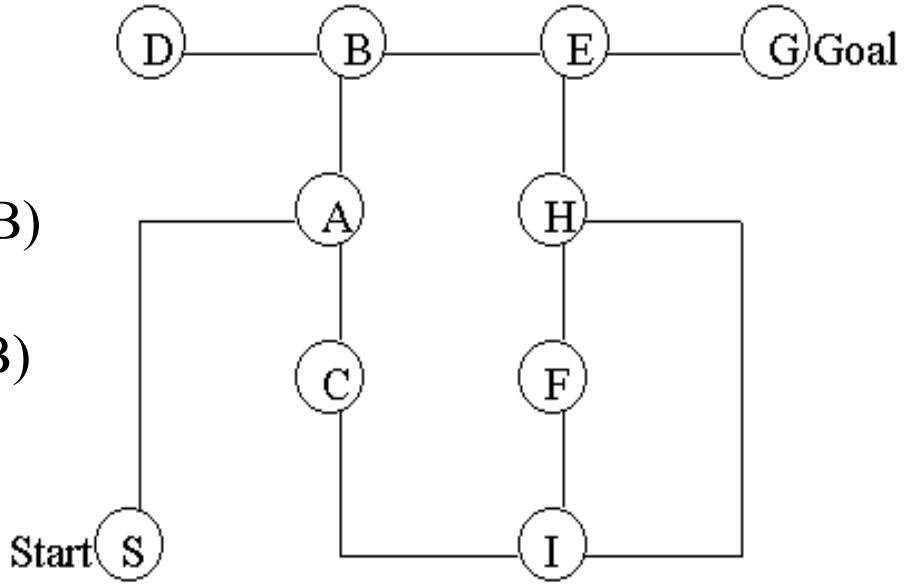
h(E)=|4-3|+|4-4|=1 (min) < h(B)

NextB = E

n:=E h(G)=|4-4|+|4-4|=0(min) < h(B)

h(H)=|4-3|+|4-3|=2

NextE = G (Đích- Dừng)



Thuật giải A^T

Thuật giải A^T (Algorithm for Tree):

Mỗi đỉnh n tương ứng với một số g(n): giá thành của đường đi từ đỉnh ban đầu đến đỉnh n.

Đỉnh:

+ Đỉnh đóng (Closed) : là những đỉnh đã được xem xét.

+Đỉnh mở (Open) : là những đỉnh giả thiết sẽ được xem xét ở bước sau.

+ Đỉnh ẩn (Hiden) : là những đỉnh mà tại đó hàm g(n) chưa được xác định.

Thuật giải A^T

Bước 1:

+ Mọi đỉnh n, mọi giá trị g(n) đều là ẩn.

+ Mở đỉnh đầu tiên và gọi đó là đỉnh S. Đặt g(S) = 0.

Bước 2 : Chọn đỉnh mở với giá thành g tương ứng là nhỏ nhất và gọi đó là đỉnh N.

+ Nếu N là mục tiêu: đường đi từ đỉnh ban đầu đến N là đường đi ngắn nhất và bằng g(N). Dừng (Success).

+ Nếu không tồn tại một đỉnh mở nào nữa: cây biểu diễn vấn đề không có đường đi tới mục tiêu. Dừng (Fail).

+ Nếu tồn tại nhiều hơn 1 đỉnh N (nghĩa là có 2 đỉnh N trở lên) mà có cùng giá thành g(N) nhỏ nhất. Kiểm tra xem trong số đó có đỉnh nào là đích hay không.

Nếu có: đường đi từ đỉnh ban đầu đến đỉnh N là ngắn nhất và bằng g(N), dừng (Success).

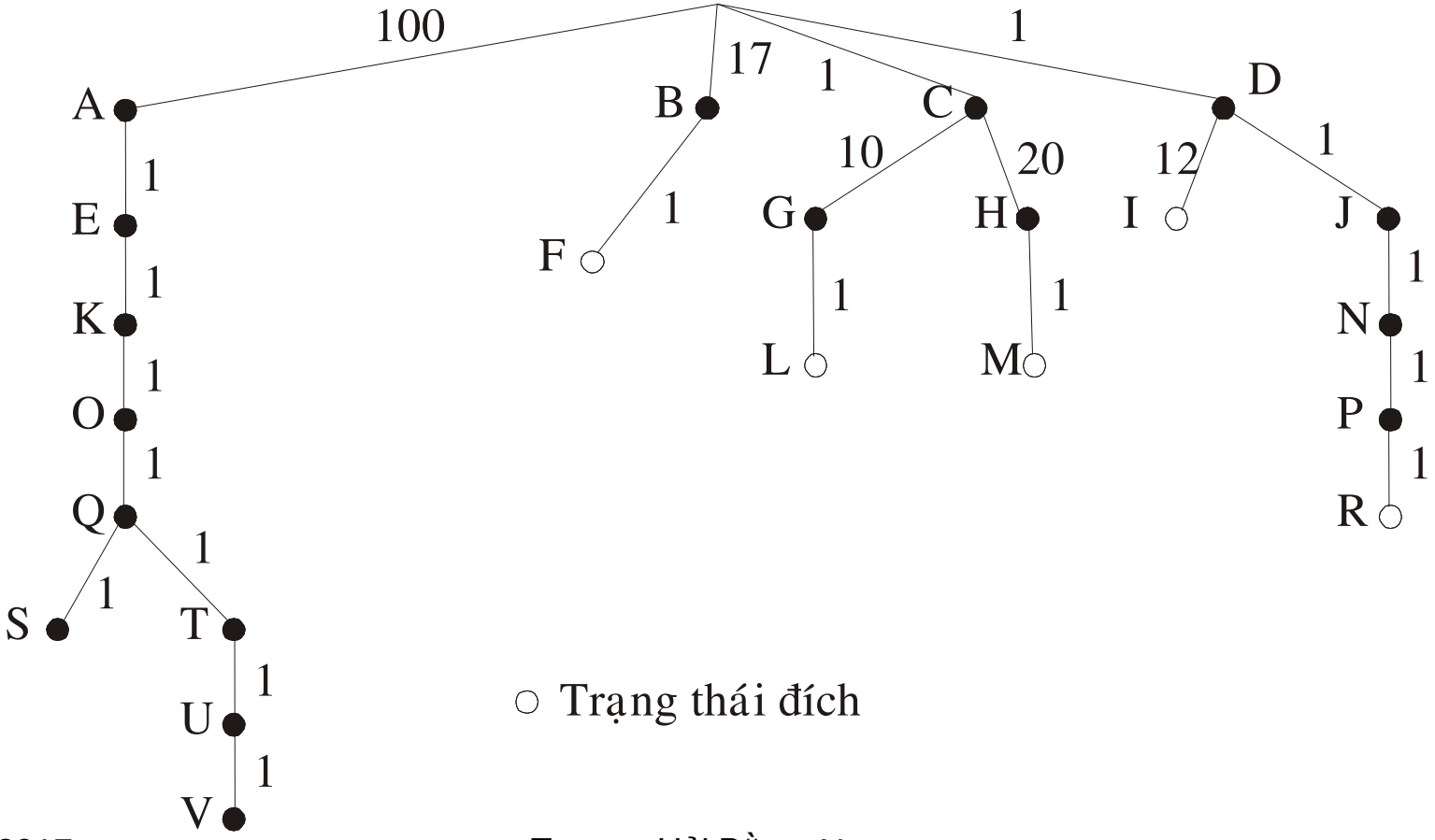
Nếu không có: Chọn ngẫu nhiên một trong các đỉnh đó và gọi là đỉnh N.

Bước 3: Đóng đỉnh N và mở các đỉnh sau N (là những đỉnh có cung hướng từ N tới). Tại mọi đỉnh S sau N tính :

$g(S) = g(N) + cost(N \rightarrow S)$

Bước 4: Quay lại bước 2

Thuật giải A^{T-} Ví dụ



Thuật giải A^{T-} Ví dụ

Mọi đỉnh n, g(n) chưa biết.

B1: Mở S, đặt g(S) = 0.

B2: Đóng S; mở A, B, C, D

g(A) = g(S) + gt(S→A) = 0 + 100 = 100

g(B) = 0 + 17 = 17

g(C) = g(D) = 0 + 1 = 1 (min)

Chọn ngẫu nhiên giữa C, D: chọn C

B3: Đóng C, mở G, H:

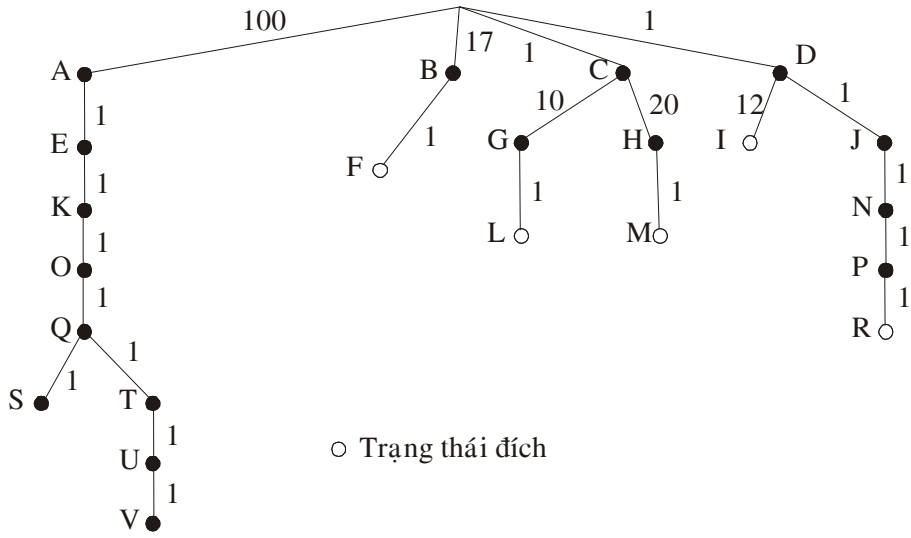
g(A) = 100

g(B) = 17

g(D) = 1 (min)

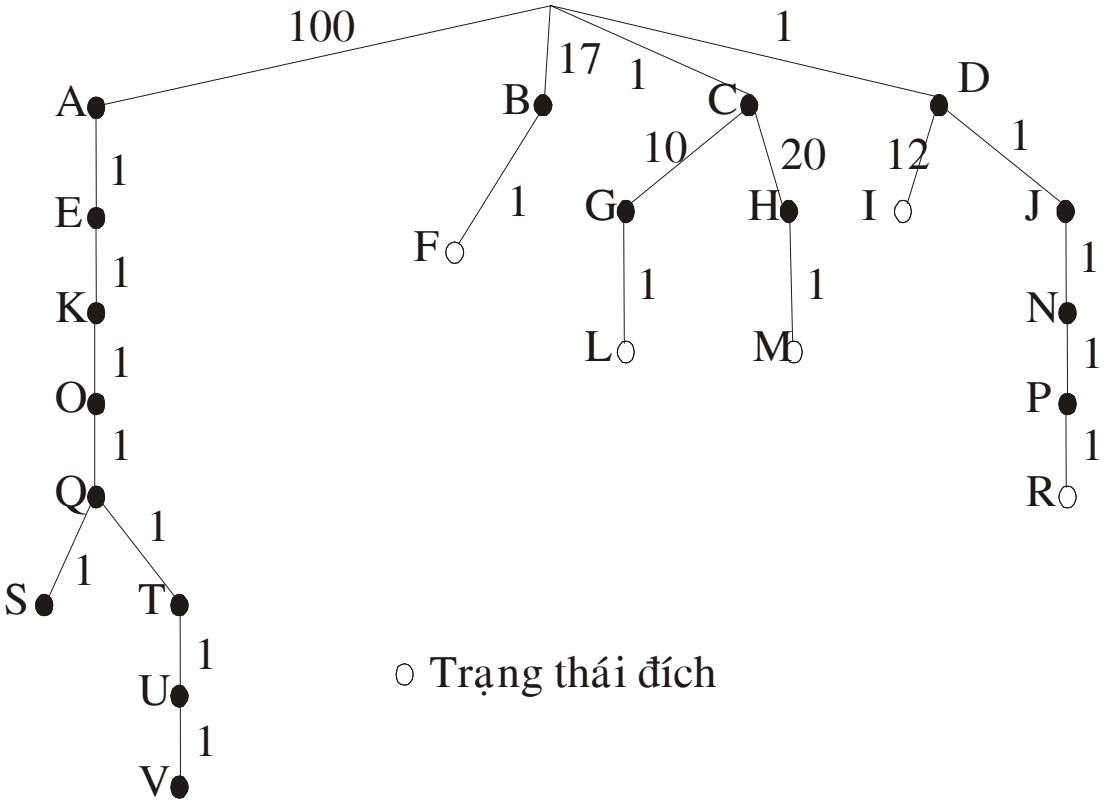
g(G) = 11

g(H) = 21



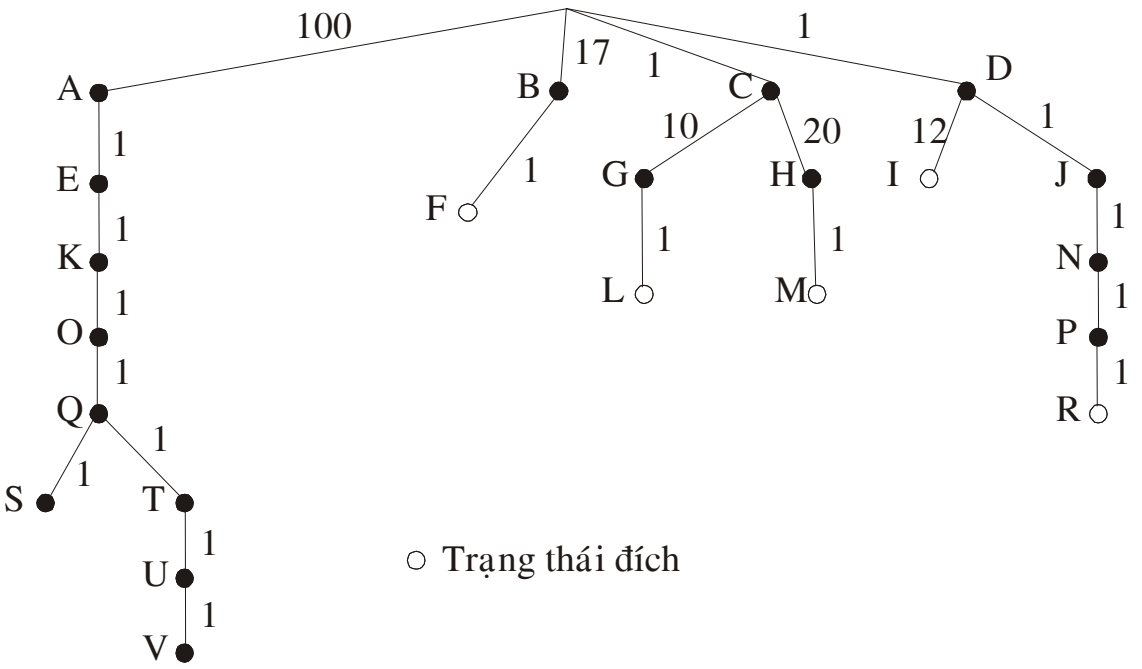
Thuật giải A^{T-} Ví dụ

B4: Đóng D, mở I, J:
g(A) = 100
g(B) = 17
g(I) = 12
g(J) = 2 (min)
g(G) = 11
g(H) = 21
B5: Đóng J, mở N:
g(A) = 100
g(B) = 17
g(I) = 12
g(G) = 11
g(H) = 21
g(N) = 3 (min)



Thuật giải A^{T-} Ví dụ

B6: Đóng N, mở P:
g(A) = 100
g(B) = 17
g(I) = 12
g(G) = 11
g(H) = 21
g(P) = 4 (min)
B7: Đóng P, mở R:
g(A) = 100
g(B) = 17
g(I) = 12
g(G) = 11
g(H) = 21
g(R) = 5 (min)
R là đích. Vậy đường đi là:



Nhận xét: Thuật toán này chỉ sử dụng 3 thông tin: đỉnh, cung và giá thành của cung.

$$S \xrightarrow{1} D \xrightarrow{1} J \xrightarrow{1} N \xrightarrow{1} P \xrightarrow{1} R$$

Thuật giải A^{KT} – Tìm kiếm với tri thức bổ sung (Algorthm for Knowled geable Tree Search):

Thuật giải A^T là thuật giải tìm kiếm đường đi tốt nhất đối với cây chỉ có các thông tin về đỉnh, cung và giá trị của cung. Trong nhiều trường hợp việc tìm kiếm đường đi sẽ được định hướng rõ thêm nếu sử dụng các tri thức thu được dựa trên các hiểu biết về tình huống vấn đề ở mỗi bước.

Tri thức bổ sung ở mỗi đỉnh được tượng ứng với một giá trị h(n). Chẳng hạn đó là ước lượng giá thành đường đi từ n đến mục tiêu. Ở ví dụ của giải thuật AT, ở bước đầu tiên :

g(c) = g(d) = 1

A^T chọn tùy ý một trong hai đỉnh c và d để xét tiếp. Nhưng thay vì chọn tùy ý chúng ta có thể đặt câu hỏi "Đỉnh nào trong các đỉnh c và d gần mục tiêu hơn", chúng ta ước lượng được:

h(c) = 11

h(d) = 4

thì việc chọn đỉnh kế tiếp sẽ là d chứ không phải c.

Do vậy tri thức bổ sung sẽ dựa trên cơ sở cực tiểu hóa giá thành f ở mỗi bước :

f(n) = g(n) + h(n)

Thuật giải A^{KT}

Bước 1:

Mọi đỉnh, cũng như các hàm g, h, f chưa biết.

Mở đỉnh đầu tiên S, gán g(S) = 0

Sử dụng tri thức bổ sung để ước tính hàm h(S)

Tính f(S) = g(S) + h(S)

Bước 2: Chọn đỉnh mở có f là nhỏ nhất và gọi là đỉnh N

Nếu N là đích: đường đi từ đỉnh ban đầu đến đỉnh N là ngắn nhất và và bằng g(N). Dừng (Success).

Nếu không tồn tại đỉnh mở nào: cây biểu diễn vấn đề không tồn tại đường đi tới mục tiêu. Dừng (Fail).

Nếu có 2 đỉnh mở trở lên có cùng giá trị f nhỏ nhất: Chúng ta phải kiểm tra xem những đỉnh đó có đỉnh nào là đích hay không.

+ Nếu có: đường đi từ đỉnh ban đầu đến đỉnh N là ngắn nhất và bằng g(N). Dừng (Success).

+ Nếu không có: chọn ngẫu nhiên một trong các đỉnh đó và gọi đỉnh đó là N.

Bước 3:

Đóng đỉnh N, mở mọi đỉnh sau N. Với mỗi đỉnh S sau N, tính:

$g(S) = g(N) + cost(S \rightarrow N)$

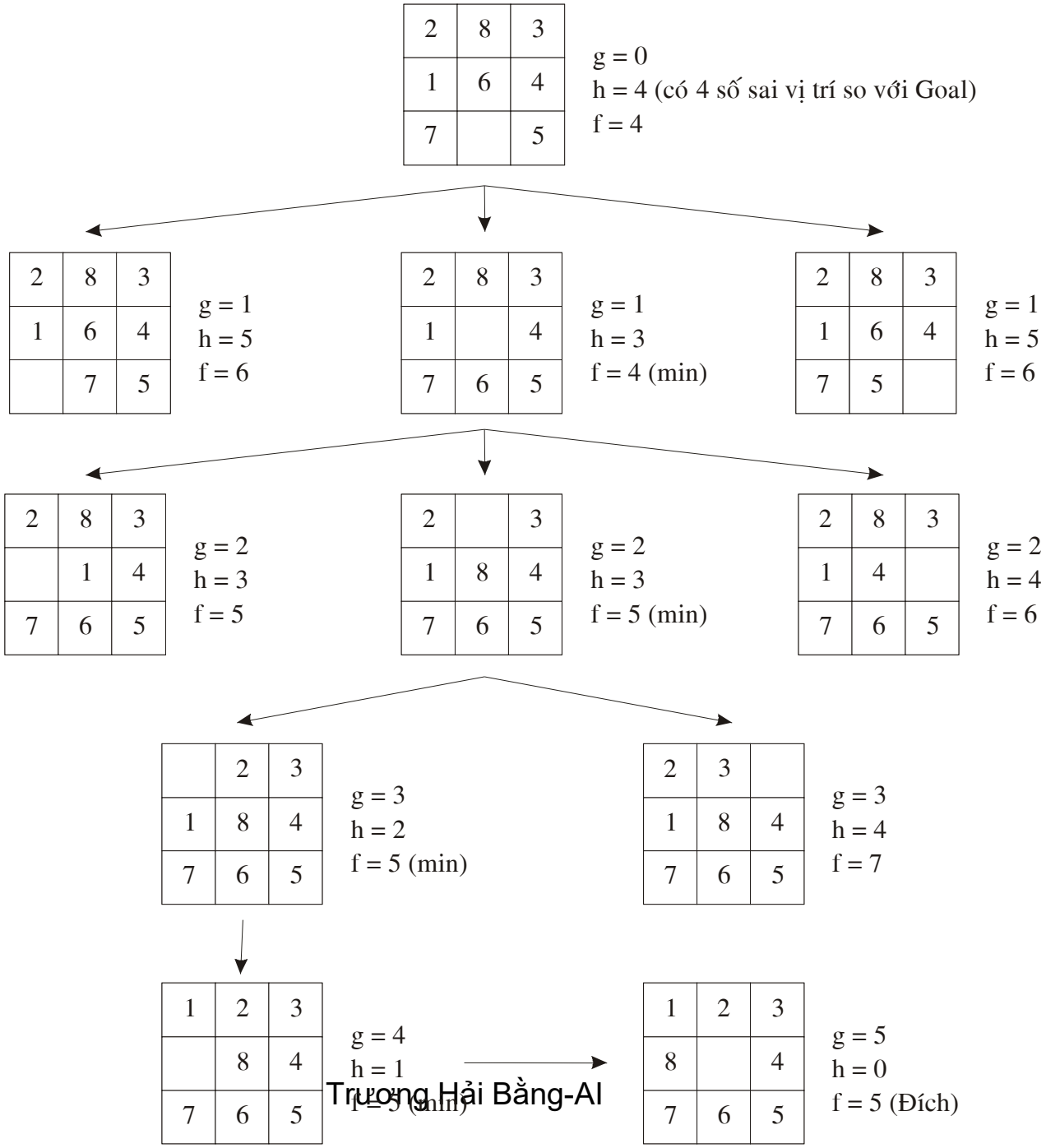
Sử dụng tri thức bổ sung để tính h(S) và f(S): f(S) = g(S) + h(S)

Bước 4: Quay lại bước 2.

Vd: **Bài toán taci**

☞**Cách 1**:

$$H = \sum_{i=1}^{t} \delta(a_i, b_i) \text{ với } \delta(a_i, b_i) = \begin{cases} 0 \text{ nếu } a_i = b_i \\ 1 \text{ nếu } a_i \neq b_i \end{cases}$$


2 8 3
1 6 4
7 5
g = 0
h = 4 (có 4 số sai vị trí so với Goal)
f = 4
2 8 3
1 6 4
7 5
g = 1
h = 5
f = 6
2 8 3
1 4
7 6 5
g = 1
h = 3
f = 4 (min)
2 8 3
1 6 4
7 5
g = 1
h = 5
f = 6
2 8 3
1 4
7 6 5
g = 2
h = 3
f = 5
2 3
1 8 4
7 6 5
g = 2
h = 3
f = 5 (min)
2 8 3
1 4
7 6 5
g = 2
h = 4
f = 6
2 3
1 8 4
7 6 5
g = 3
h = 2
f = 5 (min)
2 3
1 8 4
7 6 5
g = 3
h = 4
f = 7
1 2 3
8 4
7 6 5
g = 4
h = 1
f = 5 (min)
1 2 3
8 4
7 6 5
g = 5
h = 0
f = 5 (Đích)

Vd: Bài toán taci

☞**Cách 2:** $$H = \sum_{i=1}^{t} \eta(a_i, b_i)$$

với $\eta(a_i,b_i)$ là số lần ít nhất phải đẩy ô $a_i = a$ theo chiều dọc hay ngang về đúng vị trí $b_i = b$.

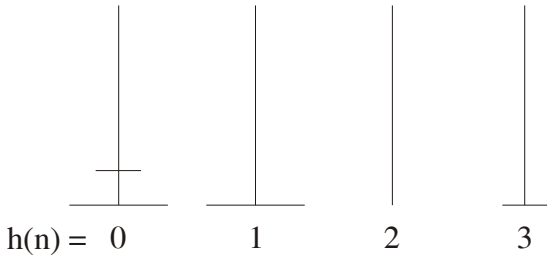
Giả sử:

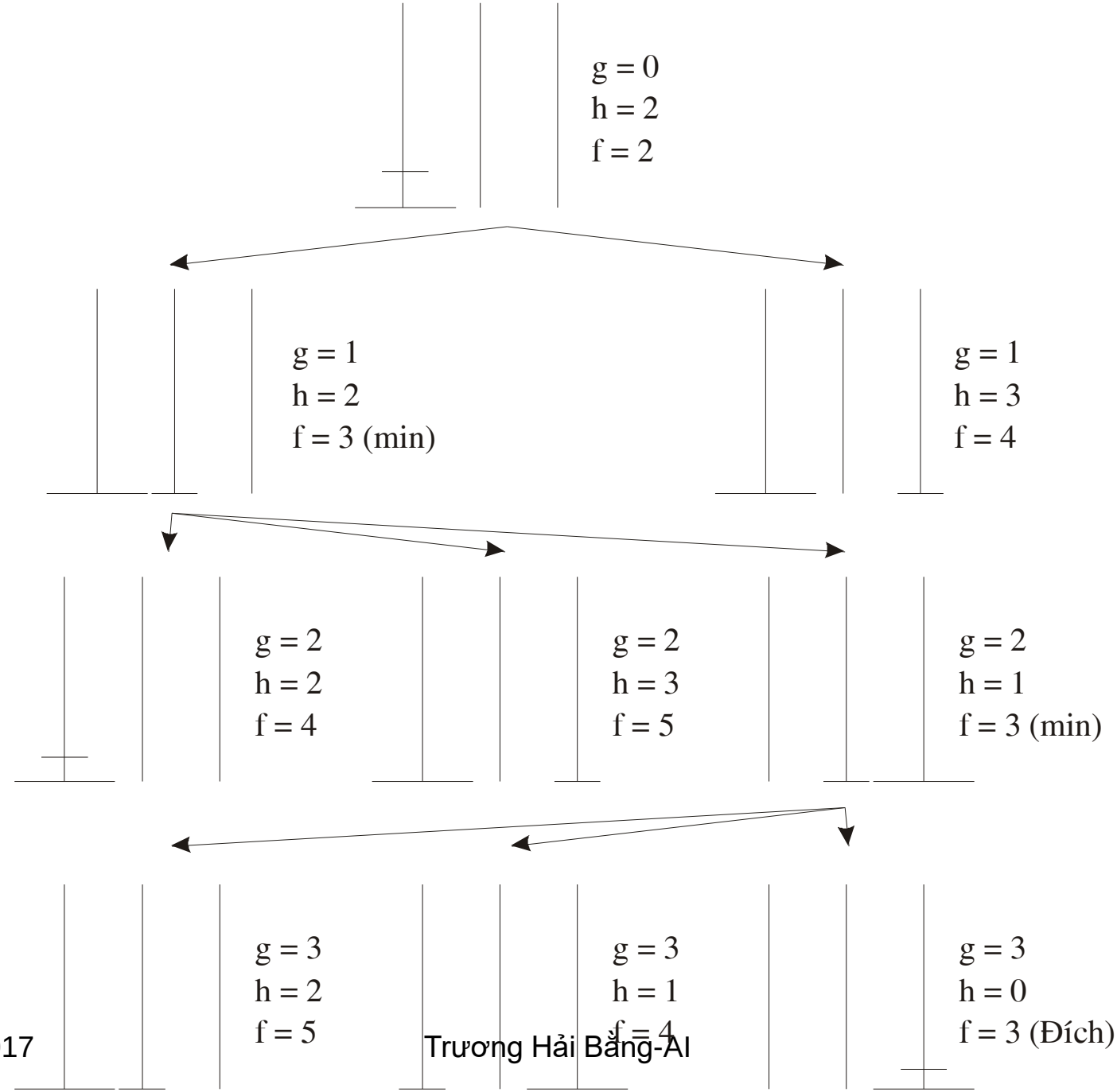
2	8	3
1	6	4
7		5

Số	1	2	3	4	5	6	7	8	Cộng
Vị trí	1	1	0	0	0	1	0	2	5

Bài toán Tháp Hà Nội với n = 2

Các trường hợp của bài toán là với trạng thái cột thứ ba:




g = 0
h = 2
f = 2
g = 1
h = 2
f = 3 (min)
g = 1
h = 3
f = 4
g = 2
h = 2
f = 4
g = 2
h = 3
f = 5
g = 2
h = 1
f = 3 (min)
g = 3
h = 2
f = 5
g = 3
h = 1
f = 4
g = 3
h = 0
f = 3 (Đích)

Thuật giải A*

(*Tìm kiếm đường đi trên đồ thị tổng quát)*

Mở rộng thuật giải A^{KT} thành thuật giải A* như sau:

Bước 1: Mở đỉnh đầu tiên:

S0 = E; {Trang thái ban đầu}

g(S0) = 0;

f(S0) = g(S0)+ h(S0) ;

θ= {S0}; {Gán S0 cho tập đỉnh mở}

C={}; {Gán tập đóng C bằng rỗng}

while θ ≠ {} do

Bước 2: Chọn một S trong θ với f(S) nhỏ nhất:

θ = θ - {S}

C = C + {S} {Đóng đỉnh S}

Nếu S là đích thì dừng.

Ngược lại qua bước 3.

*7.Thuật giải A** - tìm kiếm đường đi trên đồ thị tổng quát (tt)

Bước 3: Xây dựng các đỉnh Si có thể đến từ S nhờ các hành động có thể chọn để thực hiện.$\forall S_i$ sau S:

- Tính g(Si) ứng với mỗi i: g(Si) = g(S) + cost(S->Si).
- Ước lượng h(Si) G
- Gán f(Si)=g(Si)+h(Si)

Bước 4: Đặt vào trong θ những Si không có trong θ lẫn trong C. Với các Si đã có trong θ hoặc trong C thì gán:

f(Si) = Min($f_{cũ}$(Si), $f_{mới}$(Si)).

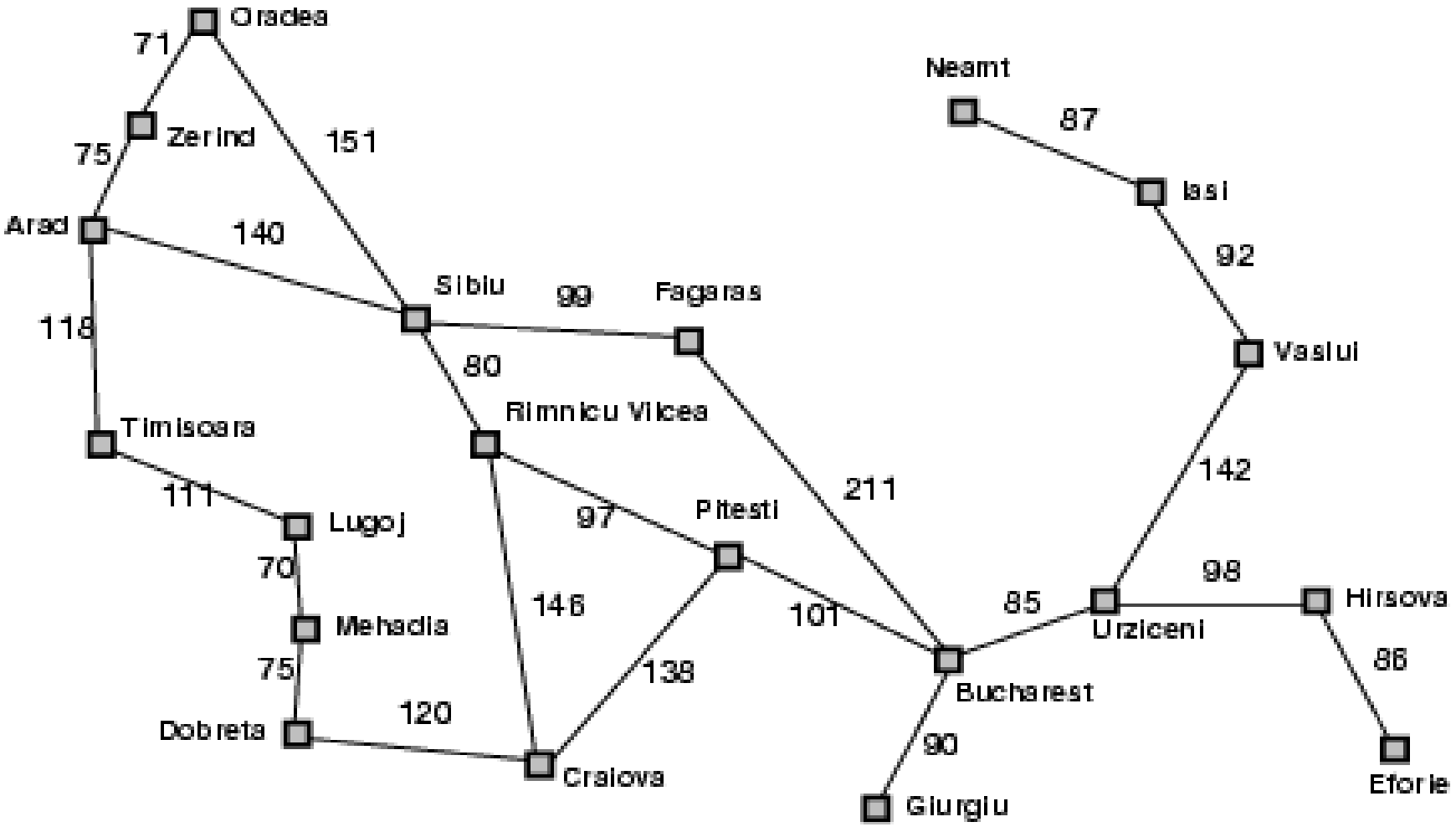
If Si có trong C and $f_{cũ}$(Si)< $f_{mới}$(Si) then

C := C – {Si}

θ := θ + {Si} {Mở Si}

End A*

*Vd1:*Bản đồ của Romania với khoảng cách tính theo km



Khoảng cách đường chim bay từ một thành phố đến Bucharest.

Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Dobreta	242	Pitesti	98
Eforie	161	R.Vilcea	193
Fagaras	178	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

Ban đầu ***(bước 1)*** :

OPEN = {(Arad,g= 0,h'= 0,f'= 0)}

CLOSE = {}

Do trong OPEN chỉ chứa một thành phố duy nhất nên thành phố này sẽ là thành phố tốt nhất. Nghĩa là S_0 = Arad.Ta lấy Arad ra khỏi OPEN và đưa vào CLOSE***(bước 2)***.

OPEN = {}

CLOSE = {(Arad,g= 0,h'= 0,f'= 0)}

Từ Arad có thể đi đến được 3 thành phố là Sibiu, Timisoara và Zerind. Ta lần lượt tính giá trị f', g và h' của 3 thành phố này ***(bước 3)***.

h’(Sibiu) = 253
g(Sibiu) = g(Arad)+cost(Arad,Sibiu) = 0+140= 140
f’(Sibiu) = g(Sibiu)+h’(Sibiu) = 140+253 = 393
h’(Timisoara) = 329
g(Timisoara) = g(Arad)+cost(Arad, Timisoara) = 0+118= 118
f’(Timisoara) = g(Timisoara)+ h’(Timisoara) = 118+329 = 447
h’(Zerind) = 374
g(Zerind) = g(Arad)+cost(Arad, Zerind) = 0+75= 75
f’(Zerind) = g(Zerind)+h’(Zerind) = 75+374 = 449
Do cả 3 nút Sibiu, Timisoara, Zerind đều không có trong cả OPEN và CLOSE nên ta bổ sung 3 nút này vào OPEN ***(bước 4)***.
OPEN = {(Sibiu,g= 140,h’= 253,f’= 393)
(Timisoara,g= 118,h’= 329,f’= 447)
(Zerind,g= 75,h’= 374,f’= 449)}
CLOSE = {(Arad,g= 0,h’= 0,f’= 0)}

Trong tập OPEN, nút Sibiu là nút có giá trị f’ nhỏ nhất nên ta sẽ chọn S_i = Sibiu. Ta lấy Sibiu ra khỏi OPEN và đưa vào CLOSE.

OPEN = {(Timisoara,g= 118,h’= 329,f’= 447)
(Zerind,g= 75,h’= 374,f’= 449)}

CLOSE = {(Arad,g= 0,h’= 0,f’= 0)
(Sibiu,g= 140,h’= 253,f’= 393)}

Từ Sibiu có thể đi đến được 4 thành phố là : Arad, Fagaras, Oradea, Rimnicu. Ta lần lượt tính các giá trị g, h’, f’ cho các nút này.

h’(Arad) = 366

g(Arad) = g(Sibiu)+cost(Sibiu,Arad)= 140+140= 280

f’(Arad) = g(Arad)+h’(Arad)= 280+366 = 646

h’(Fagaras) = 178
g(Fagaras) = g(Sibiu)+cost(Sibiu, Fagaras) = 140+99= 239
f’(Fagaras) = g(Fagaras)+ h’(Fagaras) = 239+178= 417
h’(Oradea) = 380
g(Oradea) = g(Sibiu)+cost(Sibiu, Oradea) = 140+151 = 291
f’(Oradea) = g(Oradea)+ h’(Oradea) = 291+380 = 671
h’(R.Vilcea) = 193
g(R.Vilcea) = g(Sibiu)+cost(Sibiu, R.Vilcea) = 140+80 = 220
f’(R.Vilcea) = g(R.Vilcea)+ h’(R.Vilcea) = 220+193 = 413
Nút Arad đã có trong CLOSE. Tuy nhiên, do g(Arad) mới được tạo ra (có giá trị 280) lớn hơn g(Arad) lưu trong CLOSE (có giá trị 0) nên ta sẽ không cập nhật lại giá trị g và f’ của Arad lưu trong CLOSE. 3 nút còn lại : Fagaras, Oradea, Rimnicu đều không có trong cả OPEN và CLOSE nên ta sẽ đưa 3 nút này vào OPEN, đặt cha của chúng là Sibiu. Như vậy, đến bước này OPEN đã chứa tổng cộng 5 thành phố.

Nhận xét

A^T	A^{KT}	A^*
đỉnh	đỉnh	đỉnh
Cung	Cung	Cung
Giá thành cung	Giá thành cung	Giá thành cung
	Tri thức bổ sung	Tri thức bổ sung
Thao tác trên cây	Thao tác trên cây	Thao tác trên đồ thị

Mối quan hệ giữa A^T, A^{KT}, A^*:

$f(S) = (1 - \alpha)\, g(S) + \alpha\, h(S)$ với $0 \le \alpha \le 1$

- Nếu $\alpha = 0$ $\rightarrow$ A^T (không có tri thức bổ sung)

- Nếu $\alpha = 1$ $\rightarrow$ A^{KT} (Phụ thuộc vào tri thức bổ sung)

- Nếu $\alpha = ½$ $\rightarrow$ A^*

GAME

Năm 2016 có thể được xem là năm Trí tuệ nhân tạo khi mà rất nhiều các sự kiện xảy ra và các kỷ lục bị phá vỡ. Điển hình nhất là chiến thắng áp đảo trong bộ môn cờ vây giữa kỳ thủ thế giới Lee Se-dol và hệ thống **AlphaGo** của Google vào ngày 15/3 vừa qua.

Al Demis Hassabis (trái) và Lee Sedol

Lee thảm bại với tỉ số 4-1 trước Alpha Go, đây là lần đầu tiên hệ thống trí tuệ nhân tạo chiến thắng 1 kỳ thủ cờ vây thế giới mà không cần chấp

AlphaGo được phát triển bởi DeepMind, một công ty con thuộc Google. DeepMind được thành lập vào năm 2010 bởi nhà nghiên cứu, thần đồng cờ vua, Al Demis Hassabis. Google mua lại DeepMind vào năm 2014. Cho tới nay, lượng dữ liệu các trận đấu cờ vây mà AlphaGo nhập vào khoảng 100.000 trận đấu cờ vây của con người

Chiến lược minimax

Giải thuật tìm kiếm Heuristic với các hàm heuristic chỉ thích hợp cho các bài toán không có tính đối kháng. Như các trò chơi chỉ có một người chơi: Puzzle, tìm lối ra mê cung, bài toán n quân hậu,…

Các trò chơi có tính đối kháng cao, thường là các trò chơi 2 người chơi như: tic tac toa, caro, cờ quốc tế,… giải thuật trên không có tác dụng vì: Đối phương không bao giờ đi theo con đường cho ta có thể đi đến goal

Giải thuật minimax

Bài toán que diêm

Một tập que diêm ban đầu đặt giữa 2 người chơi. Lần lượt đi xen kẽ. Người đến lượt đi phải chia nhóm que diêm theo nguyên tắc:

Chọn nhóm bất kỳ có số que >2

Chia thành 2 nhóm có số que khác nhau

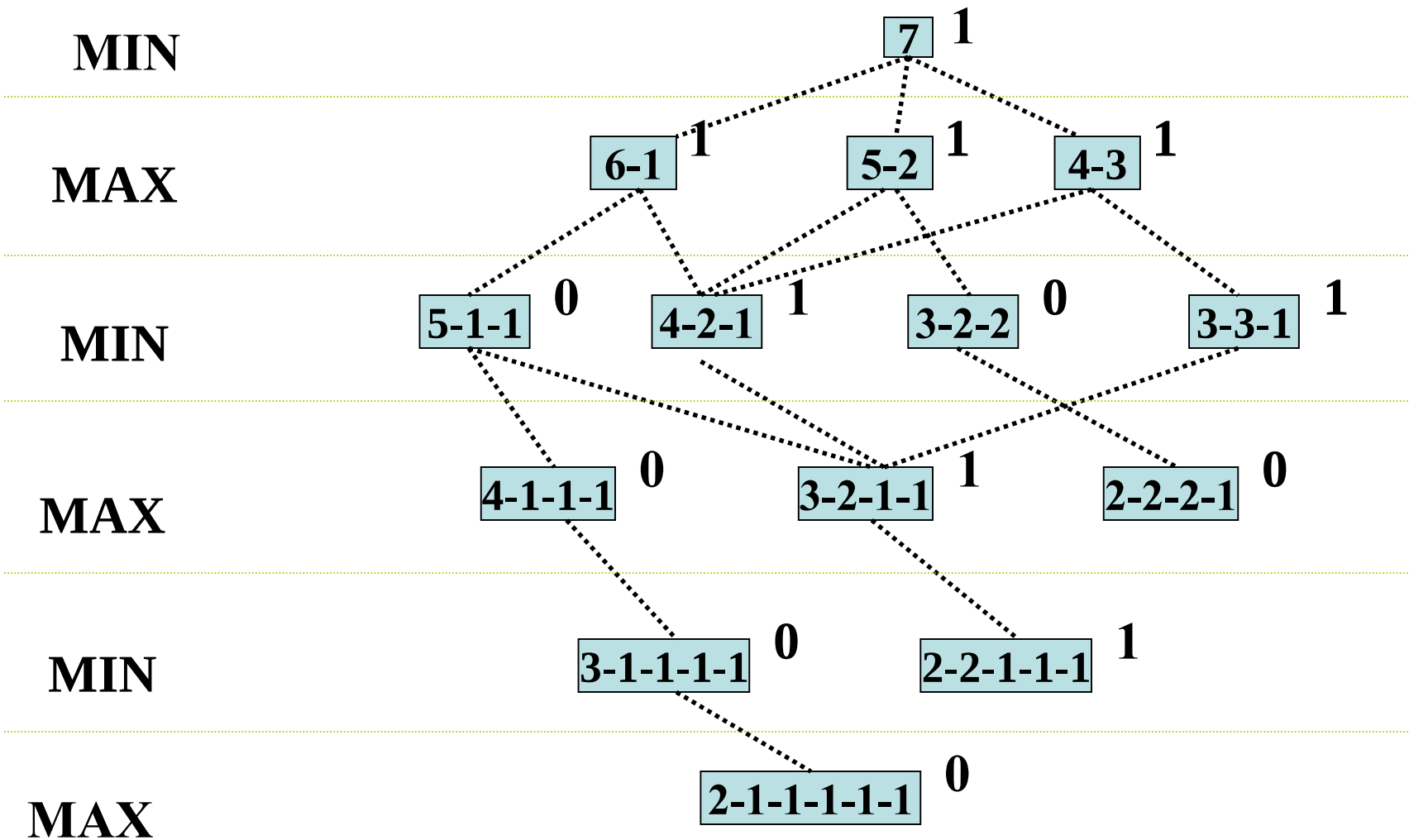
Goal: người nào đến lượt mà không chia được là thua.

Giải thuật minimax

Với một node bất kỳ nếu thuộc lớp MAX gán cho nó giá trị lớn nhất của các node con. Nếu thuộc lớp MIN gán cho nó giá trị nhỏ nhất của các node con.

Không gian trạng thái của trò chơi được phát triển toàn bộ, các node lá được gán giá trị 1 nếu là MAX thắng và 0 nếu là MIN thắng.

Minimax – bài toán que diêm



Minimax với độ sâu giới hạn

Minimax như đã xét buộc phải có toàn bộ không gian trạng thái đã được triển khai để có thể gán trị cho các nút lá và tính ngược lại ➔ Không khả thi với các bài toán lớn vì không gian trạng thái là quá lớn.

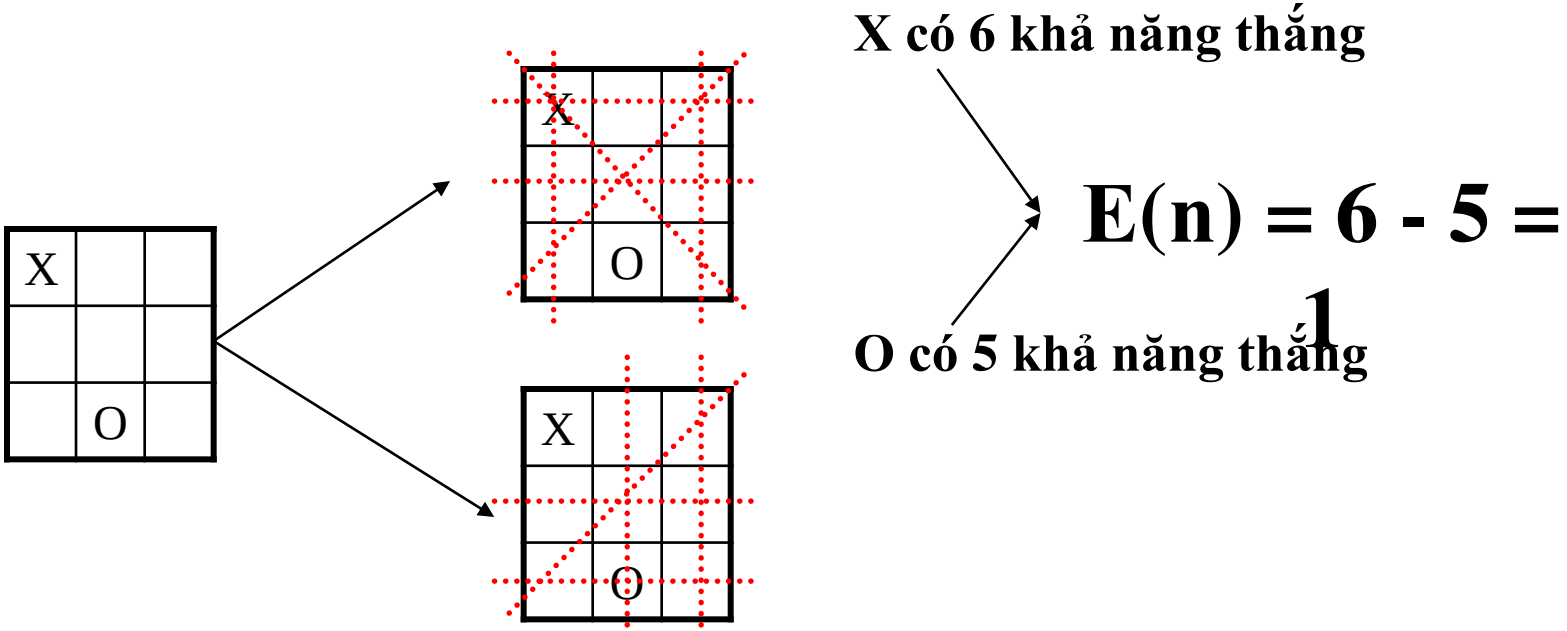
Đánh giá cho các nút này như là nút lá bằng một hàm lượng giá Heuristic.

áp dụng chiến lược minimax cho việc đánh giá các nút cấp trên.

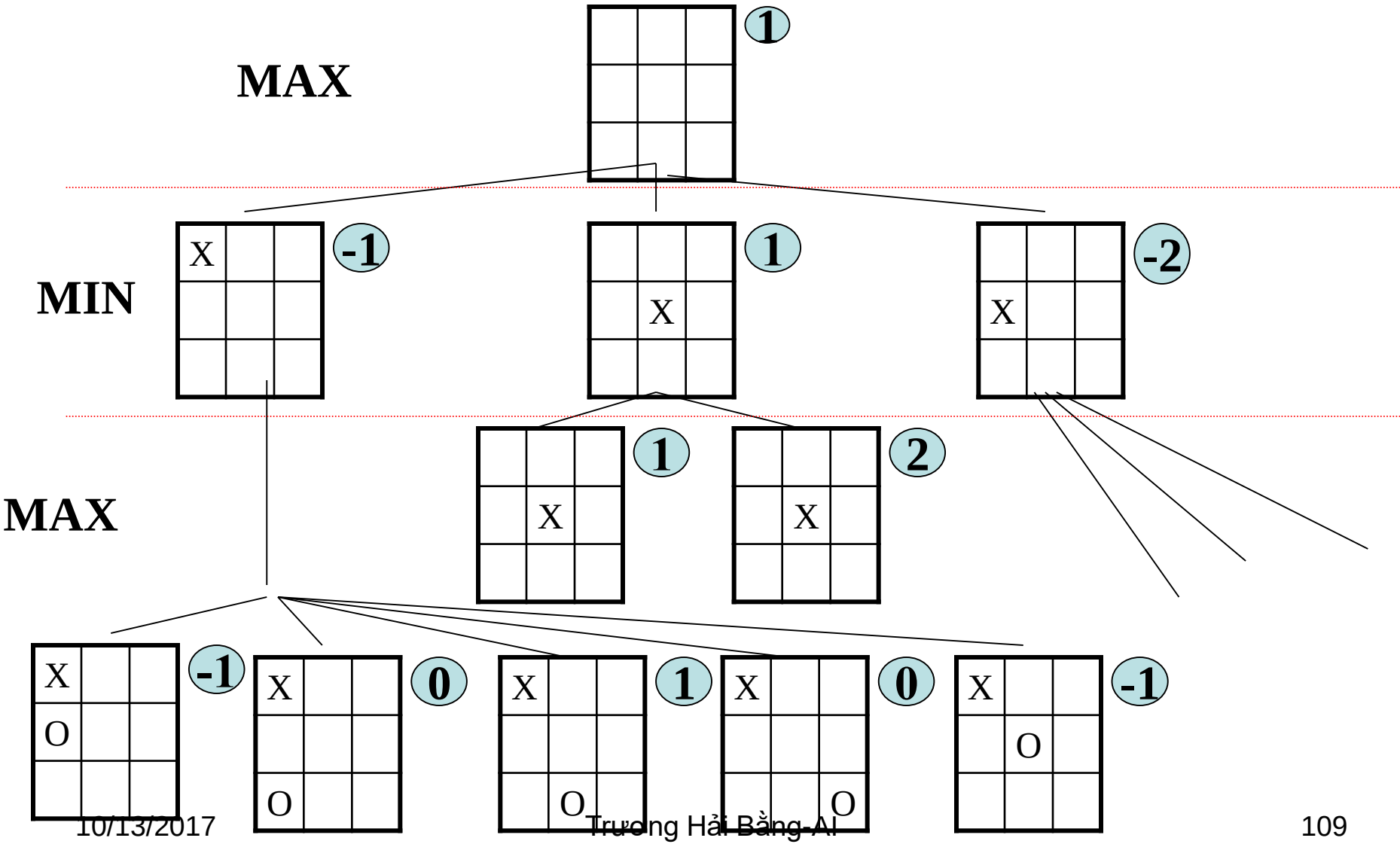
Ví dụ: Bài toán Tic Tac Toa

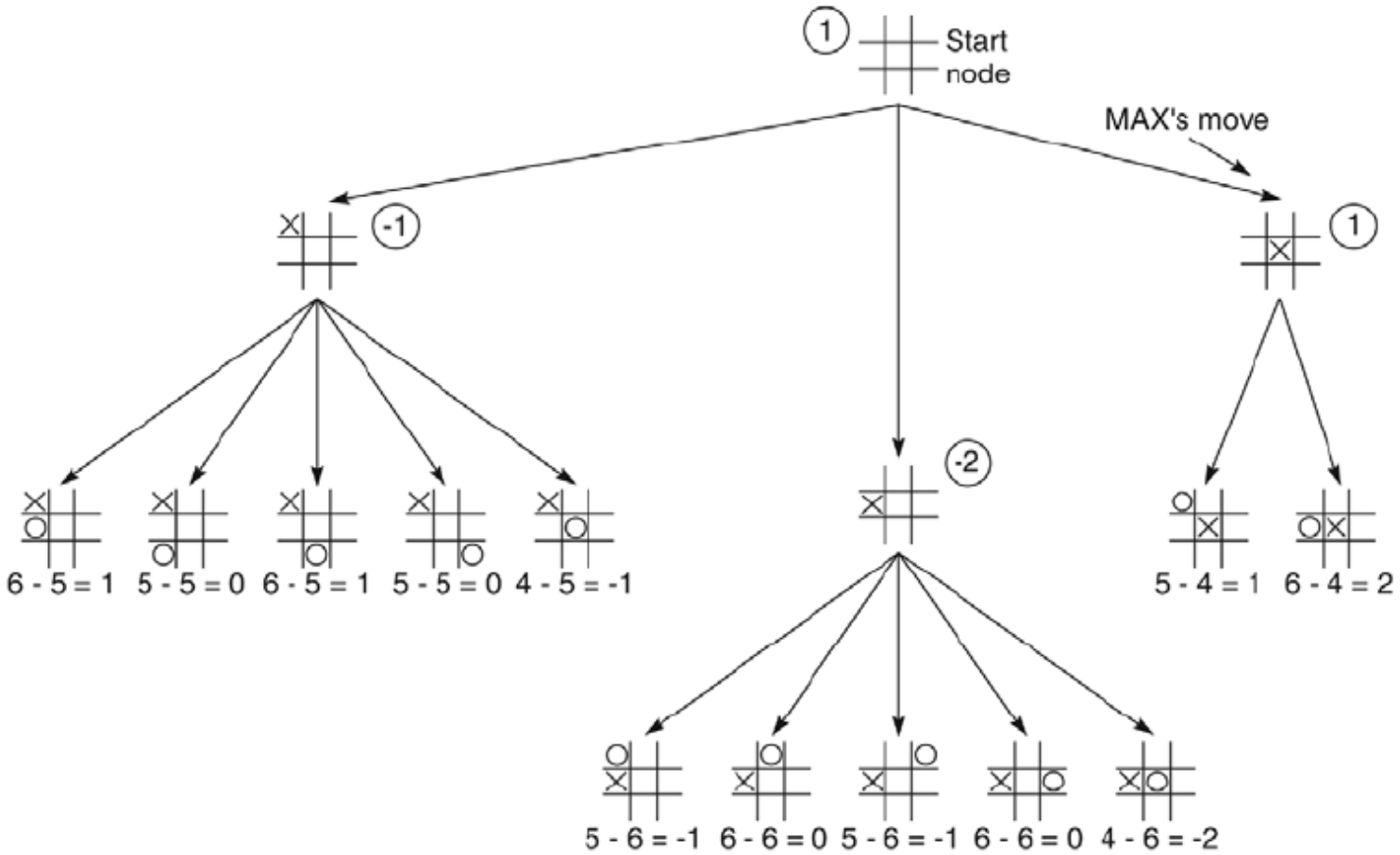
Hàm lượng giá :

f(x) = (Số dòng, số cột, số đường chéo còn mở đối với Max)-(Số dòng, số cột, số đường chéo còn mở đối với Min)



Ví dụ: Bài toán Tic Tac Toa




1
Start
node
MAX's move
-1
1
-2
6 - 5 = 1
5 - 5 = 0
6 - 5 = 1
5 - 5 = 0
4 - 5 = -1
5 - 4 = 1
6 - 4 = 2
5 - 6 = -1
6 - 6 = 0
5 - 6 = -1
6 - 6 = 0
4 - 6 = -2

Thủ tục Minimax

```
khai báo hàm minimax(nút, độ sâu của nút, người chơi hiện tại)
        nếu như độ sâu = 0 hoặc nút là nút lá cuối cùng
                trả về giá trị của nút đó
        nếu như người chơi hiện tại là MAX
        {
                gán biến bestValue = -∞
                chạy vòng lặp con của các nút
                {
                        gán giá trị val = minimax (giá trị con, độ sâu - 1 và khi dừng lại ở giá trị MIN)
                        bestValue = max (bestValue,val)
                }
                        trả về giá trị bestValue
        nếu như người chơi hiện tại là MIN
        {
                gán biến bestValue = +∞
                chạy vòng lặp con của các nút
                {
                        gán giá trị val = minimax(giá trị con, độ sâu - 1 và khi dừng lại ở giá trị MAX)
                        bestValue = min (bestValue,val)
                }
                        trả về giá trị bestValue
```

- Nút là trạng thái của bàn cờ.
- Độ sâu là số bước nhảy
- Người chơi hiện tại.

Thủ tục Minimax

- Nếu trường hợp nút là nút lá; trả về giá trị đã được gán cho nút lá.
- Nếu trường hợp người chơi là *max* gán giá trị tạm *bestValue*=-∞, trong vòng lặp for của các nút con tiếp gán giá trị *val* bằng cách gọi đệ quy hàm này cho đối thủ. Cuối cùng tìm giá trị lớn nhất giữa *bestValue* và *val* vừa tìm được gán ngược lại cho biến *bestValue*.

Thủ tục Minimax

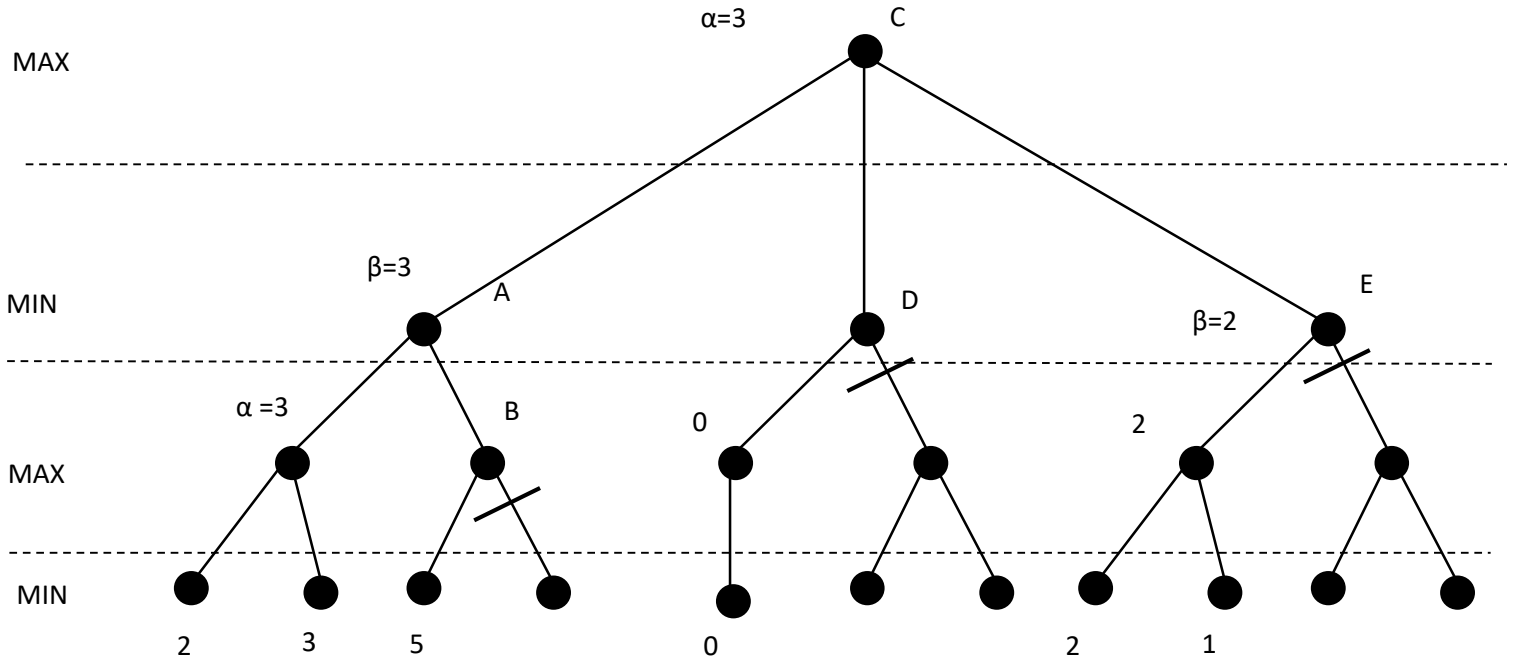
- Nếu trường hợp người chơi là min: *bestValue*=+∞
- Trong vòng lặp for con của các nút con gán giá trị *val* bằng cách gọi đệ quy hàm này cho đối thủ. Cuối cùng tìm giá trị nhỏ nhất giữa *bestValue* và *val* vừa tìm được gán ngược lại cho biến *bestValue*.

Thủ tục Alpha - Beta

Thuật toán Minimax yêu cầu có hai bước đối với không gian tìm kiếm: bước đầu truyền xuống đến độ sâu của lớp áp dụng heuristic và bước sau để truyền ngược các giá trị trên cây. Minimax lần theo tất cả các nhánh trong không gian bao gồm cả những nhánh mà một thuật toán thông minh hơn có thể bỏ qua hay tỉa bớt. Thủ tục cắt tỉa Alpha-beta nhằm nâng cao hiệu quả thủ tục

Ví dụ

A có β=3 (Giá trị nút A sẽ không lớn hơn 3). B bị cắt tỉa β, vì 5>3. C có α=3 (Giá trị nút C sẽ không nhỏ hơn 3). D bị cắt tỉa α, vì 0<3. E bị cắt tỉa α, vì 2<3. Giá trị nút C là 3.



Thủ tục Alpha - Beta

```
function alphabeta(node, depth, α, β, maximizingPlayer)
{
        if (depth == 0 or node là nút lá)
                return giá trị lượng giá của thế cờ
        if (lượt chơi là maximizingPlayer)
        {
                v= -∞;
                for (con của node)
                {
                        v := max(v, alphabeta( con , depth - 1, α, β, FALSE)); {Gán giá trị lớn nhất cho v}
                        α := max(α, v); {Gán giá trị lớn nhất cho alpha}
                        if (β ≤ α)
                                break; (* cắt α *)
                }
                return v;
        }
        if (lượt chơi là minimizingPlayer)
        {
                v := ∞;
                for   (con của node)
                {
                        v := min(v, alphabeta( con , depth - 1, α, β, TRUE)); { Gán giá trị nhỏ nhất cho v}
                        β := min(β, v) {Gán giá trị nhỏ nhất cho beta}
                        if β ≤ α
                               break (* cắt β*)
                }
                return v;
        }
}
```

Bài toán 8 con hậu

Ví dụ: Bài toán 8 hậu:

+ Cho 3 quân hậu đặt trước vào bàn cờ A0 {(1,4), (2,2), (3,8)}

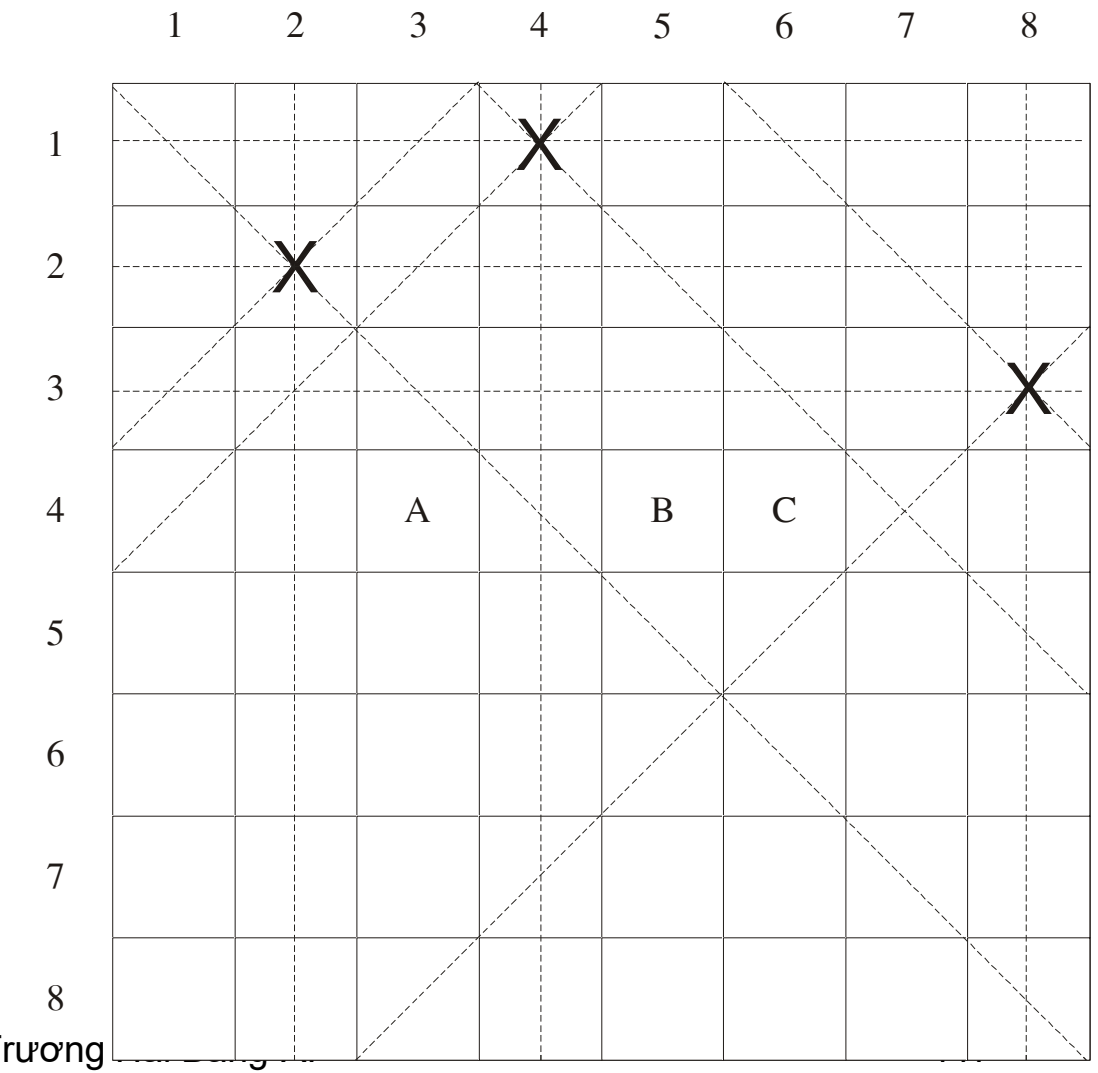
+ Hãy đặt tiếp 5 quân hậu khác sao cho các con hậu không ăn nhau:

+ Gợi ý: Có thể đặt tại dòng 4 một con hậu ở một trong ba vị trí A,B,C:

Nếu chọn A: sẽ còn 8 vị trí có thể đặt tiếp quân hậu

Nếu chọn B: sẽ còn 9 vị trí có thể đặt tiếp quân hậu

Nếu chọn C: sẽ còn 10 vị trí có thể đặt tiếp quân hậu



Bài toán 8 con hậu(tt)

+ Hàm lượng giá:

f1(0): số vị trí có thể đặt quân hậu tiếp trên không gian còn lại

f1(A) = 8; f1(B) = 9; f1(C) = 10

$\Rightarrow$ Max f1(0) = C

f2(0): số ít nhất các vị trí có thể đặt quân hậu tiếp trên một hàng

f2(A) = 1; f2(B) = 1; f2(C) = 2

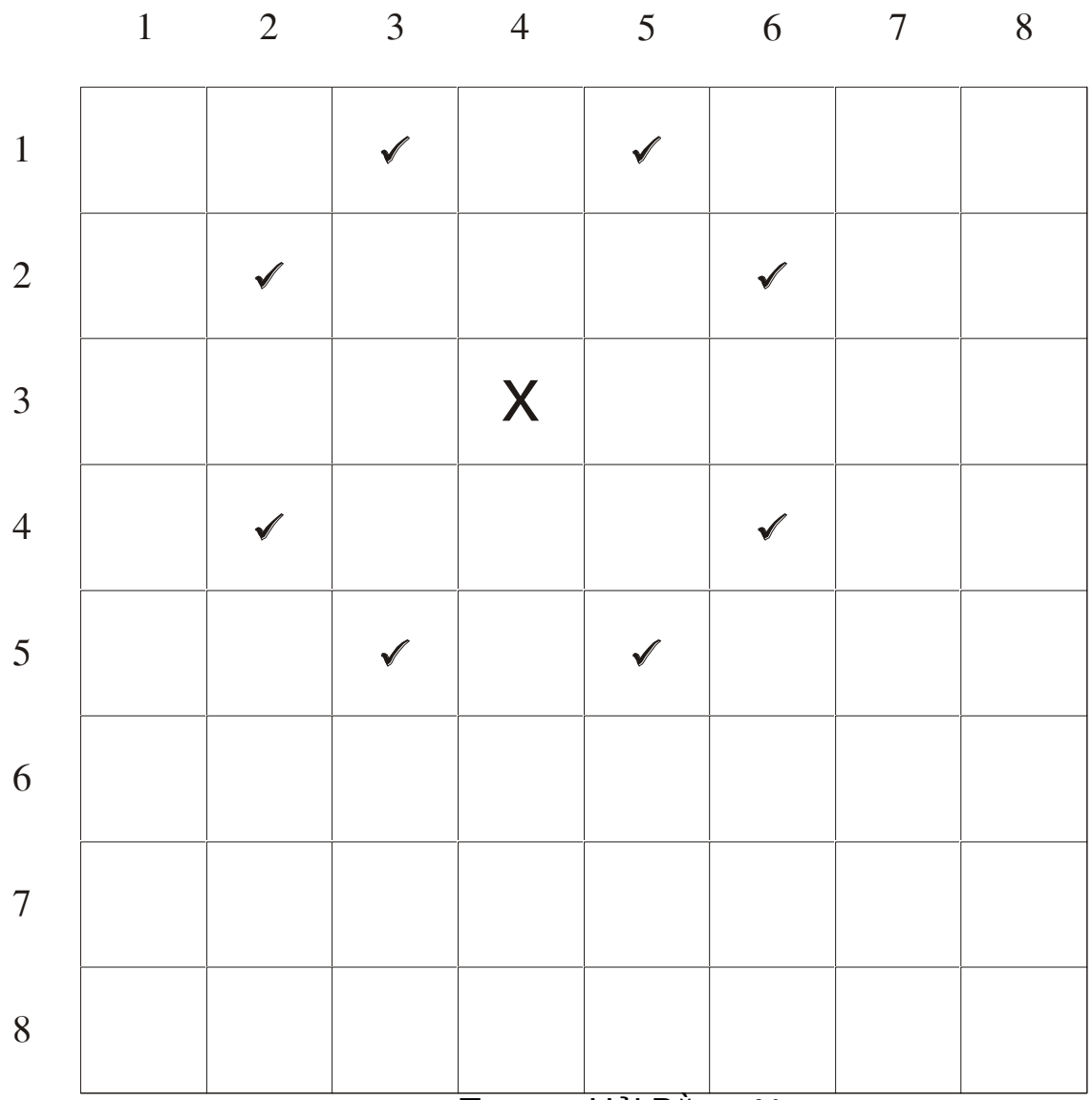
$\Rightarrow$ Max f2(0) = C

Có thể sử dụng cùng lúc cả 2 hàm f1(0) và f2(0). Nếu 2 hàm này tính ra có vị trí sai lệch nhau, ta chọn: $\text{Max}\left(\frac{f_1 + f_2}{2}\right)$

Mã đi tuần

+ Cách 1: Viết chương trình để mã đi qua hết 64 ô trong bàn cờ mà mỗi ô chỉ qua một lần.
+ Cách 2: Viết chương trình để mã đi qua hết 64 ô trong bàn cờ mà mỗi ô chỉ qua một lần và bước cuối cùng phải quay về đúng vị trí xuất phát.

Mã đi tuần



Thuật giải Heuristic

Định nghĩa: Một thuật giải Heuristic có hai đặc tính sau:
Thường tìm được lời giải tốt, mặc dù không phải là lời giải tốt nhất.
Thực tiễn dễ dàng và nhanh chóng so với thuật giải tối ưu.
Cách tiếp cận chung cho việc thiết kế một thuật giải Heuristic:
Liệt kê tất cả các yêu cầu của một thuật toán và chia chúng thành 2 lớp:
Lớp 1: những yêu cầu dễ dàng thỏa mãn.
Lớp 2: những yêu cầu không dễ dàng thỏa mãn.
Hoặc 2 lớp khác như sau:
Lớp 1: những yêu cầu cần phải thỏa mãn.
Lớp 2: những yêu cầu có thể sẵn lòng thỏa hiệp.

Mở rộng khái niệm thuật toán:

Khái quát: Đối với thuật toán, chúng ta có ba tính chất bắt buộc:

- Tính xác định (đơn định).
- Tính dừng (hữu hạn).
- Tính đúng (kết quả).

Tuy nhiên trong quá trình nghiên cứu giải bài toán, các nhà toán học đã nhận thấy các tình huống như sau:

Có các bài toán cho đến nay vẫn chưa tìm ra được cách giải theo thuật toán và cũng không biết có tồn tại thuật toán để giải bài toán này hay không?

Có các bài toán đã tìm ra được thuật toán nhưng không thể thực hiện được hoặc khó thực hiện vì thời gian giải theo thuật toán đó quá dài hoặc các điều kiện đặt ra cho thuật toán là khó đáp ứng được.

Có các bài toán được giải theo các cách giải mà vi phạm các tính chất của thuận toán nhưng lời giải được thực tiễn

Mở rộng khái niệm thuật toán(tt)

Từ các tính huống (3) gợi mở các đổi mới cho khái niệm thuật toán giải các bài toán thuộc tình huống (1) và (2).

Mở rộng tính xác định : Tính xác định của thuật toán được hiểu theo trực quan là tính tự nhiên tới từng bước nhỏ nhất (đó là các thao tác sơ cấp) của thuật toán được hiểu đơn trị và rõ ràng dù là người hay máy chỉ cần biết thực hiện đúng các bước (các thao tác sơ cấp) theo đúng trình tự quy định sẽ đi tới kết quả mong muốn.

Tuy nhiên có nhiều cách giải các **bài** toán trong thực tế không cần phải giải theo cách này. Thay cho việc xác định toàn bộ quá trình giải chỉ cần chỉ ra cách chuyển (không bắt buộc phải đơn trị và rõ ràng) từ bước i đến bước i+1 là được

Mở rộng khái niệm thuật toán(tt)

Mở rộng tính đúng: Được hiểu theo nghĩa kết quả nhận được sau khi thực hiện thuật toán phải là kết quả đúng. Thật ra yêu cầu về kết quả đúng đã vi phạm từ khi con người phải liên quan đến số thực. Có thể nói với mọi thuật toán trên máy tính có nội dung tính toán với số thực đều chỉ cho kết quả gần đúng. Nhiều bài toán thực tiễn cũng như vậy. Nếu chấp nhận kết quả không bắt buộc là kết quả đúng mà chỉ gần đúng thì có thể tồn tại nhiều cách giải đỡ phức tạp và hiệu quả hơn. Tuy nhiên mở rộng về tính đúng đắn của thuật toán được khởi nguồn từ cách giải quyết vấn đề bài toán heuristic. Đó là các cách giải đơn giản thường cho kết quả đúng hay gần đúng, cách giải này không phải lúc nào cũng dẫn đến kết quả nhưng do tính đơn giản và khả năng thành công nhiều nên xét về mặt hiệu quả vẫn có vai trò quan trọng.

CÁC PHƯƠNG PHÁP BIỂU DIỄN VÀ XỬ LÝ TRI THỨC

Thuật toán Vương Hạo và Robinson
Biểu diễn tri thức bằng mạng ngữ nghĩa
Biểu diễn tri thức bằng luật sinh
Biểu diễn Tri thức bằng Frame
Biểu diễn tri thức bằng Script
Phối hợp nhiều phương pháp biểu diễn tri thức

I. Tổng quan

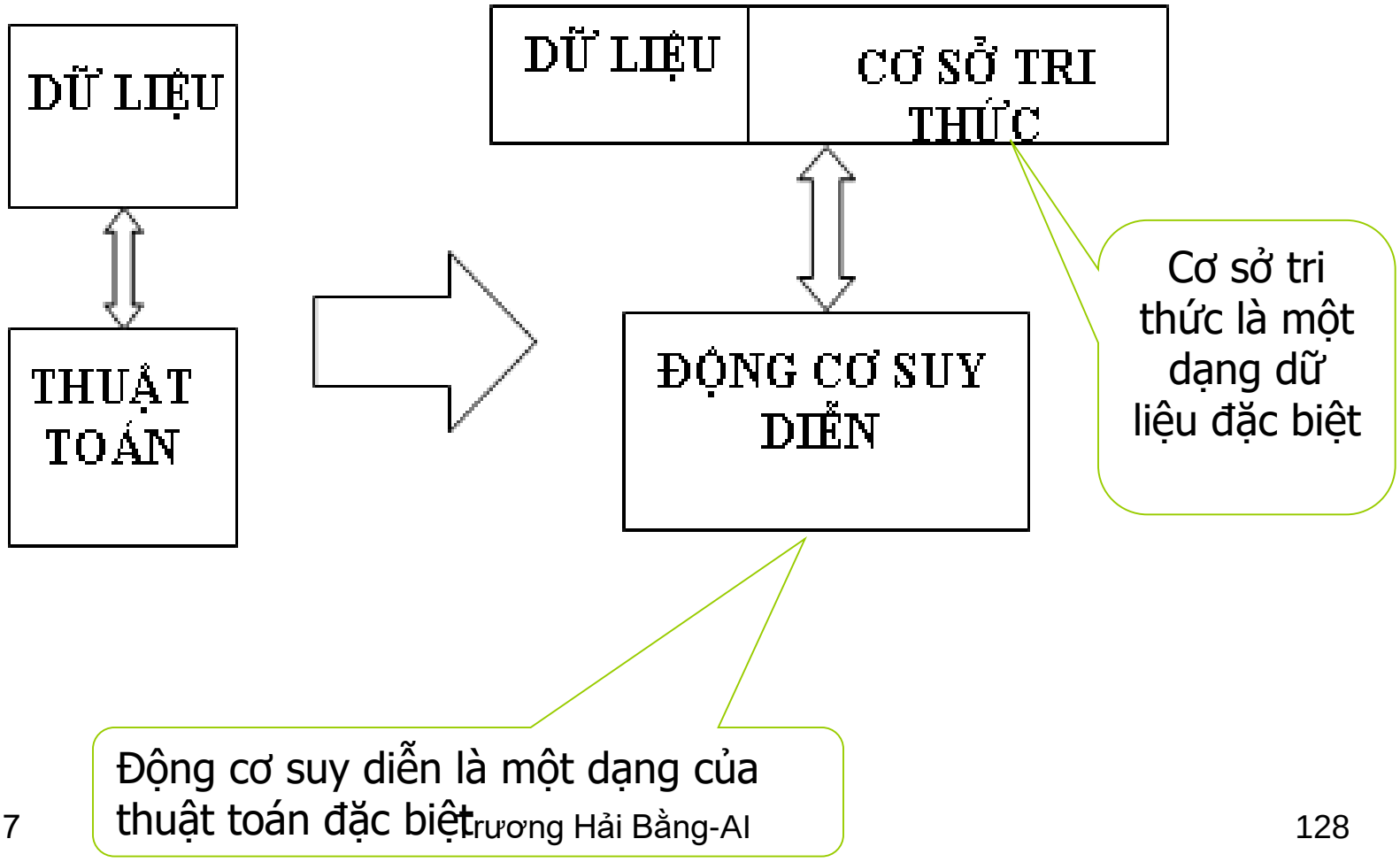
So với chương trình truyền thống (được cấu tạo từ hai "chất liệu" cơ bản là ***dữ liệu*** **và** ***thuật toán*****), chương trình trí tuệ nhân tạo được cấu tạo từ hai thành phần là** ***cơ sở tri thức*** **(knowledge base) và** ***động cơ suy diễn*** ***(inference engine).***

1.Giới thiệu(tt):

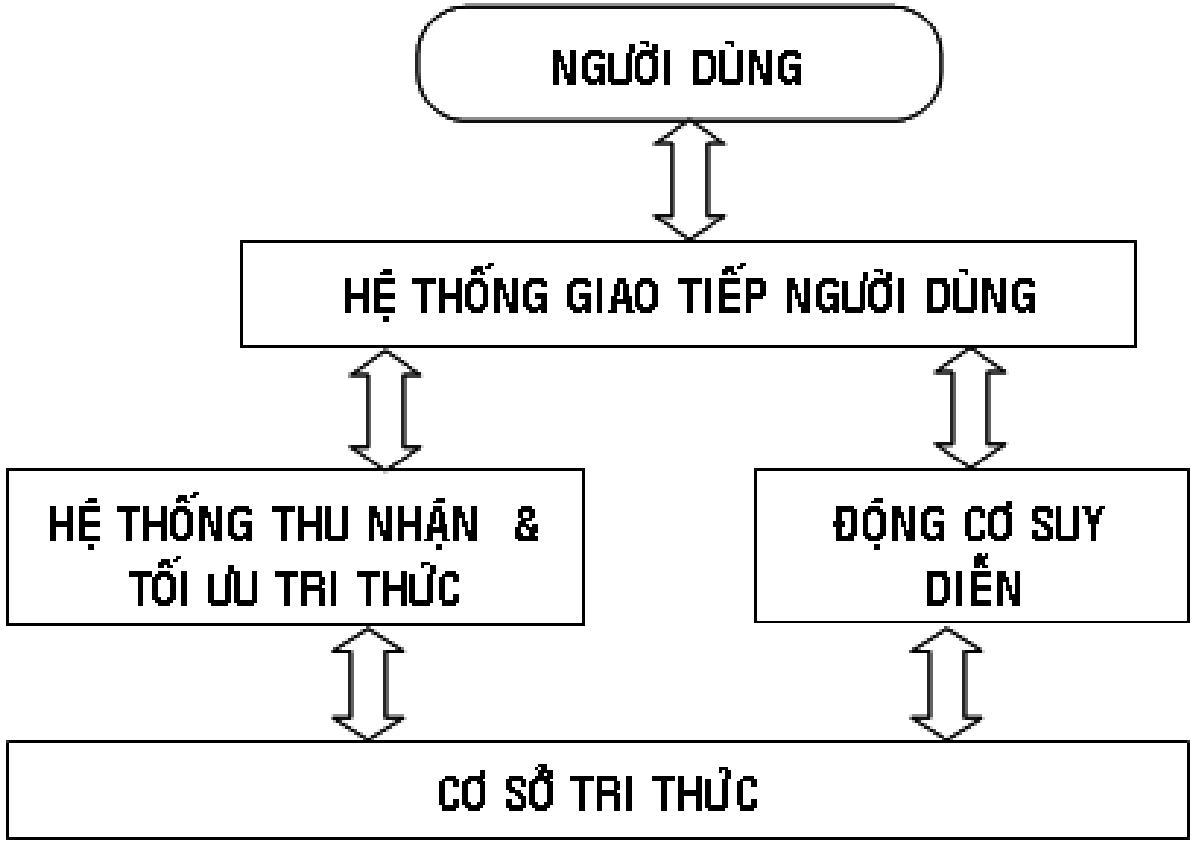
Cơ sở tri thức : là tập hợp các tri thức liên quan đến vấn đề mà chương trình quan tâm giải quyết.

Động cơ suy diễn : là phương pháp vận dụng tri thức trong cơ sở tri thức để giải quyết vấn đề.

1.Giới thiệu(tt):



1.Giới thiệu(tt):



Cấu trúc của một chương trình trí tuệ nhân tạo.

2. Phân loại tri thức:

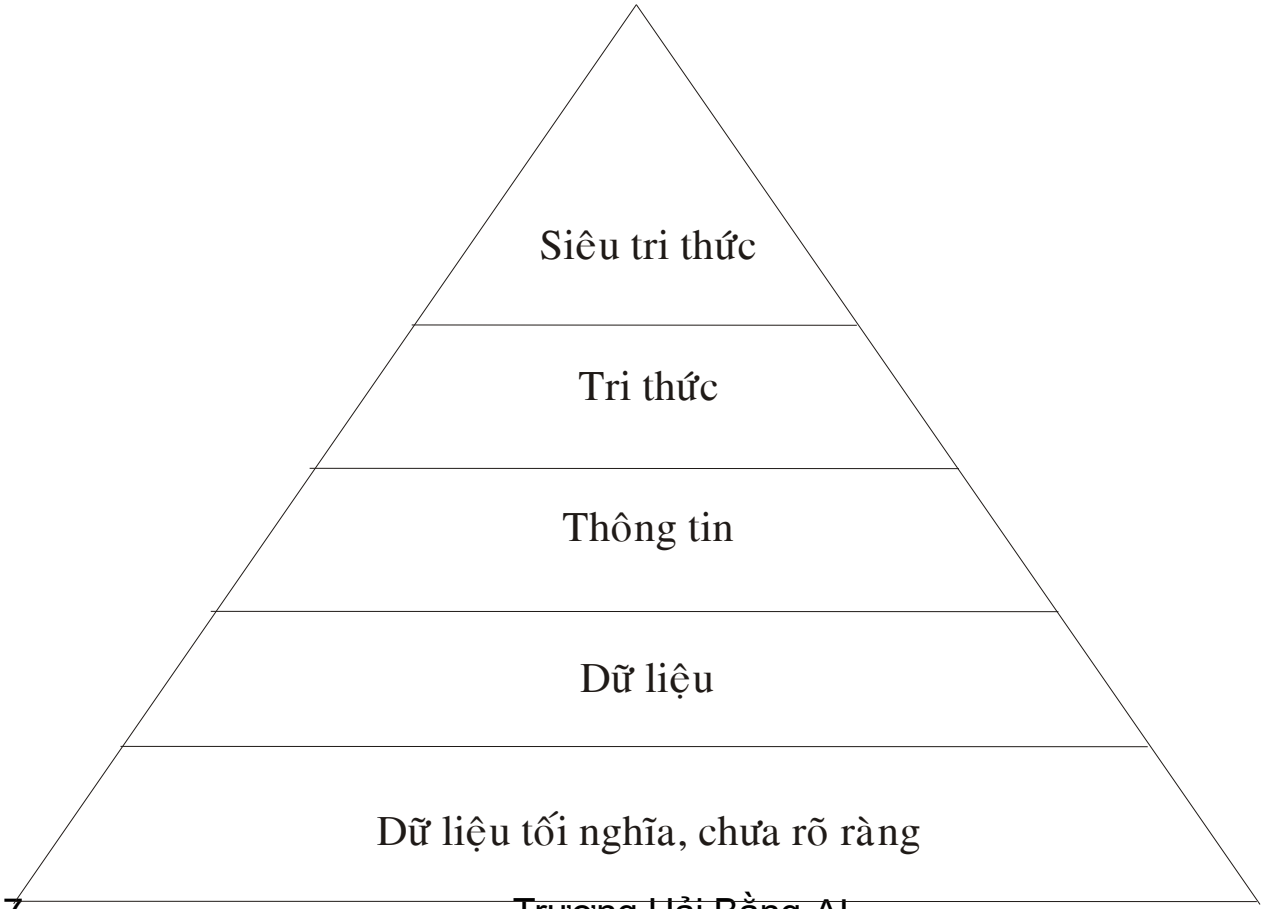
Tri thức sự kiện : là các khẳng định về một sự kiện, khái niệm nào đó (trong một phạm vi xác định). Các định luật vật lý, toán học, ... thường được xếp vào loại này. (Chẳng hạn : mặt trời mọc ở đằng đông, tam giác đều có 3 góc 60^0, ...)

Tri thức thủ tục : thường dùng để diễn tả phương pháp, các bước cần tiến hành, trình từ hay ngắn gọn là cách giải quyết một vấn đề. Thuật toán, thuật giải là một dạng của tri thức thủ tục.

Tri thức mô tả : cho biết một đối tượng, sự kiện, vấn đề, khái niệm, ... được thấy, cảm nhận, cấu tạo như thế nào (một cái bàn thường có 4 chân, con người có 2 tay, 2 mắt,...)

Tri thức Heuristic : là một dạng tri thức cảm tính. Các tri thức thuộc loại này thường có dạng ước lượng, phỏng đoán, và thường được hình thành thông qua kinh nghiệm.

3.Sự phân lớp của tri thức:



4. Đặc điểm của tri thức:

Làm thế nào để phân biệt thông tin vào máy tính là dữ liệu hoặc tri thức. Giữa tri thức và dữ liệu có một số đặc trưng khác nhau.

Tự giải thích nội dung: Tri thức tự giải thích nội dung còn dữ liệu không tự giải thích được. Chỉ có người lập trình mới hiểu được nội dung ý nghĩa các dữ liệu.

Ví dụ:

Dữ liệu là số 7.

Tri thức là số 7: là số lẻ, là số nguyên tố, là số dương,…

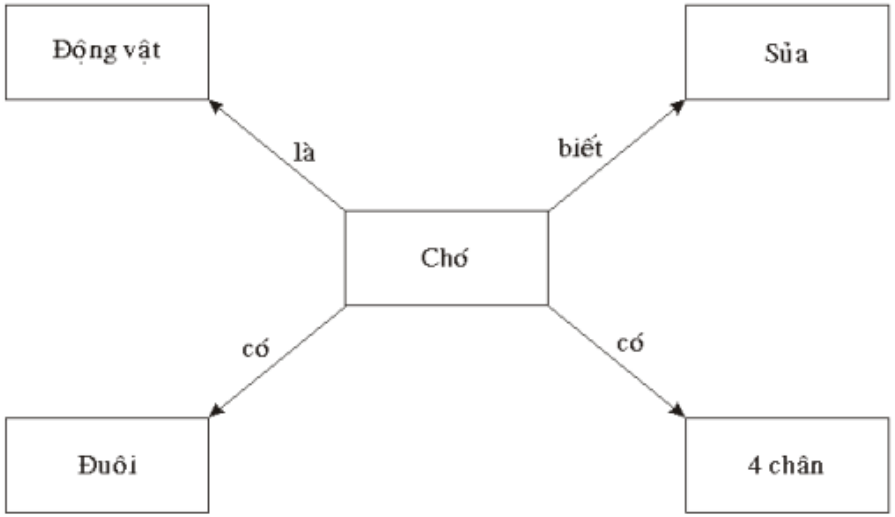
Tính cấu trúc: Một trong những đặc trưng cơ bản của hoạt động nhận thức con người đối với thế giới xung quanh là khả năng phân tích cấu trúc các đối tượng.Ở mức đơn giản nhất là cấu trúc: là một bộ phận của toàn thể, là một giống của một loài nào đó, là phần tử của lớp nào đó.

Tri thức đưa vào máy cũng cần có khả năng tạo được phân cấp giữa các khái niệm và quan hệ giữa chúng.

4. Đặc điểm của tri thức (tt):

Tính liên hệ: Ngoài các quan hệ về cấu trúc của mỗi tri thức (khái niệm, quá trình, sự kiện, hiện tượng,…) giữa các đơn vị tri thức còn có nhiều mối liên hệ khác (không gian, thời gian, nhân-quả, …)

Ví dụ: Các khái niệm: chó, sủa, động vật, bốn chân, đuôi.



4. Đặc điểm của tri thức (tt):

-<u>Có tính chủ động:</u>

Dữ liệu hoàn toàn bị động do con người khai thác, còn tri thức thì có tính chủ động. Khi hoạt động bất kỳ ở đâu trong lĩnh vực nào, con người cũng bị điều khiển bởi tri thức của mình. Các tri thức biểu diễn trong máy tính cũng vậy, chúng chủ động hướng người dùng biết cách khai thác dữ liệu.

II. CÁC PHƯƠNG PHÁP BIỂU DIỄN TRI THỨC

1. Logic mệnh đề:

-Định nghĩa: Mệnh đề là một khẳng định có thể nhận giá trị đúng hoặc sai**.**

2. Logic vị từ :

-Định nghĩa: là sự mở rộng của logic mệnh đề bằng cách đưa vào các khái niệm vị từ và các lượng từ phổ thông dụng **(∀, ∃).**

Trong logic vị từ, một mệnh đề được cấu tạo bởi hai thành phần là các đối tượng tri thức và mối liên hệ giữa chúng (gọi là vị từ). Các mệnh đề sẽ được biểu diễn dưới dạng :

Vị từ (<đối tượng 1>, <đối tượng 2>, …, <đối tượng n>)

Như vậy để biểu diễn vị của các trái cây, các mệnh đề sẽ được viết lại thành :

Cam có vị Ngọt ~ Vị (Cam, Ngọt)

Cam có màu Xanh ~ Màu (Cam, Xanh)

II. CÁC PHƯƠNG PHÁP BIỂU DIỄN TRI THỨC (tt)

VD: Câu cách ngôn "Không có vật gì là lớn nhất và không có vật gì là bé nhất!" có thể được biểu diễn dưới dạng vị từ như sau :

LớnHơn(x,y) = x>y

NhỏHơn(x,y) = x<y

" ∀ x ∃ y : LớnHơn(y,x) và " ∀ x ∃ y : NhỏHơn(y,x)

Câu châm ngôn "Gần mực thì đen, gần đèn thì sáng" được hiểu là "chơi với bạn xấu nào thì ta cũng sẽ thành người xấu" có thể được biểu diễn bằng vị từ như sau :

NgườiXấu (x) = ∃ y : Bạn(x,y) và NgườiXấu(y)

Công cụ vị từ đã được nghiên cứu và phát triển thành một ngôn ngữ lập trình đặc trưng cho trí tuệ nhân tạo. Đó là ngôn ngữ PROLOG. Phần đọc thêm của chương sẽ giới thiệu tổng quan với các bạn về ngôn ngữ này.

3.MỘT SỐ THUẬT GIẢI LIÊN QUAN ĐẾN LOGIC MỆNH ĐỀ

a)Thuật toán Vương Hạo (Havard – 1960):

Bước 1: Phát biểu lại giả thiết và kết luận của vấn đề dưới dạng chuẩn như sau:

GT1, GT2, …, GTn $\rightarrow$ KL1, KL2, … KLm

Trong đó các GTi và KLj được xây dựng từ các biến mệnh đề và các phép toán $\wedge$, $\vee$, $\neg$.

Bước 2: Chuyển về các GTi và KLj có dạng phủ định.

Ví dụ:

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

$\Rightarrow p \vee q, p \vee r, p \rightarrow s, r \wedge s, q$

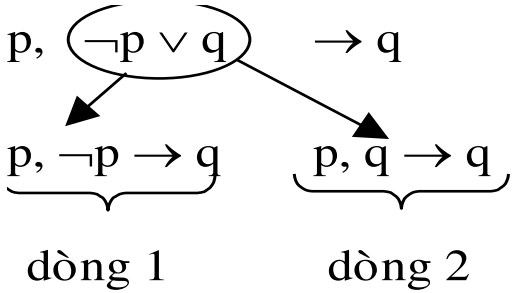
Bước 3: Thay dấu "$\wedge$" ở trong GTi và dấu "$\vee$" ở trong KLj bằng dấu "," (phẩy).

Ví dụ:

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

$\Rightarrow p, q, r, \neg p \vee s \rightarrow \neg q, \neg r$

a)Thuật toán Vương Hạo (Havard – 1960) (tt):



Bước 4: Nếu GTi còn dấu "$\vee$" và KLj còn dấu "$\wedge$" thì dòng đó được tách thành hai dòng con.

Ví dụ:

Bước 5: Nếu một dòng được chứng minh: nếu tồn tại chung một mệnh đề ở cả 2 vế thì coi như đúng.

Ví dụ: p, q $\rightarrow$ p: mệnh đề đúng

Bước 6:

+ Nếu một dòng không còn dấu liên kết tuyển và hội mà cả ở hai vế đều không có chung biến mệnh đề nào thì dòng đó không được chứng minh.

Ví dụ: p, $\neg$q $\rightarrow$ q

+ Một vấn đề được giải quyết một cách trọn vẹn nếu mọi dòng dẫn xuất từ dạng chuẩn được chứng minh.

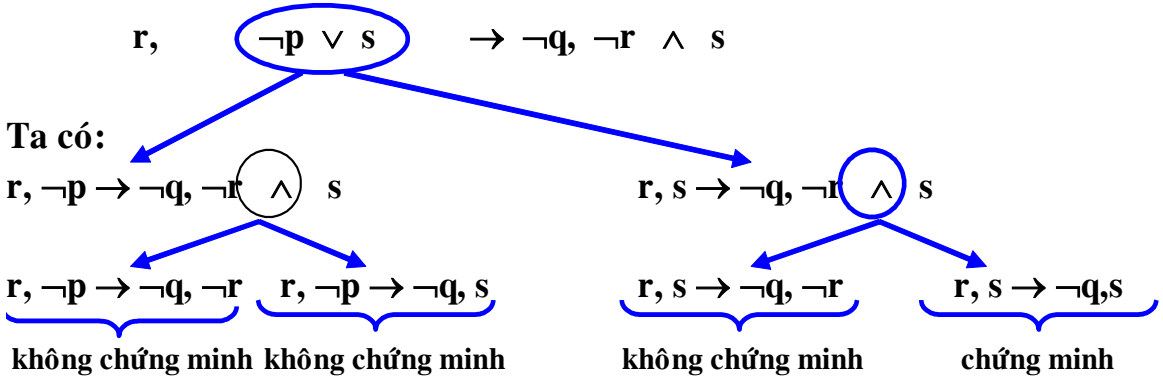
Lưu ý:

Từ bước 2 đến bước 4 không cần làm theo thứ tự.

a)Thuật toán Vương Hạo (Havard – 1960) (tt):

Khi một vấn đề được phân thành n vấn đề con, ta phải chứng minh tất cả các mệnh đề con đều đúng thì mệnh đề đầu mới đúng. Nếu chứng minh được một mệnh đề con sai thì mệnh đề chính sai.

Ví dụ: Giả sử có một vấn đề được hiểu dưới dạng chuẩn sau, hãy chứng minh vấn đề này đúng hay sai.



Kết luận: Vấn đề trên sai.

a)Thuật toán Vương Hạo (Havard – 1960) (tt):

Đánh giá giải thuật: Nếu ở một dòng có n dấu $\wedge$, $\vee$ thì:

+ Để lập bảng chân trị cần 2n cột để xét giá trị.

+ Nếu dùng thuật toán thì phải tách ra 2n dòng.

$\Rightarrow$ Độ phức tạp của thuật toán đơn giản hơn phương pháp lập bảng chân trị.

b)Thuật toán Robinson (1961):

Bước 1: Phát biểu lại giả thiết và kết luận của vấn đề dưới dạng chuẩn như sau:

GT1, GT2, …, GTn $\rightarrow$ KL1, KL2, … KLm

Trong đó các GTi và KLj được xây dựng từ các biến mệnh đề và các phép toán $\wedge$, $\vee$, $\neg$.

Bước 2: Biến đổi dòng trên thành danh sách các mệnh đề:

{GT1, GT2, …, GTn, $\neg$KL1, $\neg$KL2, …, $\neg$KLm}

Bước 3: Nếu trong danh sách mệnh đề có 2 mệnh đề đối ngẫu nhau thì vấn đề được giải quyết xong. Nếu không thì chuyển sang bước 4.

b)Thuật toán Robinson (1961) (tt):

Bước 4: Xây dựng một mệnh đề mới bằng cách tuyển một cặp mệnh đề từ danh sách mệnh đề ở bước 2. Nếu mệnh đề mới có các biến mệnh đề đối ngẫu thì các biến mệnh đề đó được loại bỏ.

Ví dụ:

$(p \vee \neg q) \vee (\neg r \vee s \vee q)$

$\Rightarrow p \vee \neg r \vee s$

Bước 5: Bổ sung mệnh đề mới này vào danh sách các mệnh đề và loại bỏ 2 mệnh đề được tuyển thành mệnh đề mới đó.

Bước 6: Nếu không xây dựng thêm được mệnh đề mới nào và trong danh sách các mệnh đề không có 2 mệnh đề nào đối ngẫu nhau thì vấn đề phát biểu ở dạng chuẩn của bước 1 là sai..

Ví dụ 1:

Có một vấn đề phát biểu ở dạng chuẩn như sau, hãy chứng minh vấn đề đúng hay sai:

$\neg p \vee q, \neg q \vee r, \neg r \vee s, \neg u \vee \neg s \rightarrow \neg p, \neg u$

$\{\neg p \vee q, \neg q \vee r, \neg r \vee s, \neg u \vee \neg s, p, u\}$

$\{\neg p \vee r, \neg r \vee s, \neg u \vee \neg s, p, u\}$

$\{\neg p \vee s, \neg u \vee \neg s, p, u\}$

$\{\neg p \vee \neg u, p, u\}$

$\{\neg u, u\}$

Kết luận: Điều phải chứng minh là đúng

VÍ DỤ 2

(a) Cam là thức ăn.

(b) Vịt là thức ăn.

(c) Món ăn mà ăn không chết gọi là thức ăn.

(d) Ông An ăn táo.

(e) Ông An còn sống.

Hỏi Táo có phải là thức ăn không?

VÍ DỤ 2:(tt)

Diễn đạt phát biểu dưới dạng vị từ:

Thứcăn(X)
Sống(Y)
Ăn(Y, X)

(a) Cam là thức ăn.
(b) Vịt là thức ăn.
(c) Món ăn mà ăn
không chết gọi là thức ăn.
(d) Ông An ăn táo.
(e) Ông An còn sống.
Hỏi Táo có phải là thức ăn không

(a) Thứcăn(Cam)
(b) Thứcăn(Vịt)
(c) Ăn(Y, X) ∧ Sống(Y) → Thứcăn(X)
≡¬Ăn(Y, X) ∨ ¬Sống(Y) ∨ Thứcăn(X)
(d) Ăn(An, Táo)
(e) Sống(An)
(f) Thứcăn(Táo)?

Bài tóan được phát biểu lại: a,b,c,d,e->f

VÍ DỤ 2:(tt)

Bài toán: a,b,c,d,e ->f

Bước 2:a,b,c,d,e, ¬ f

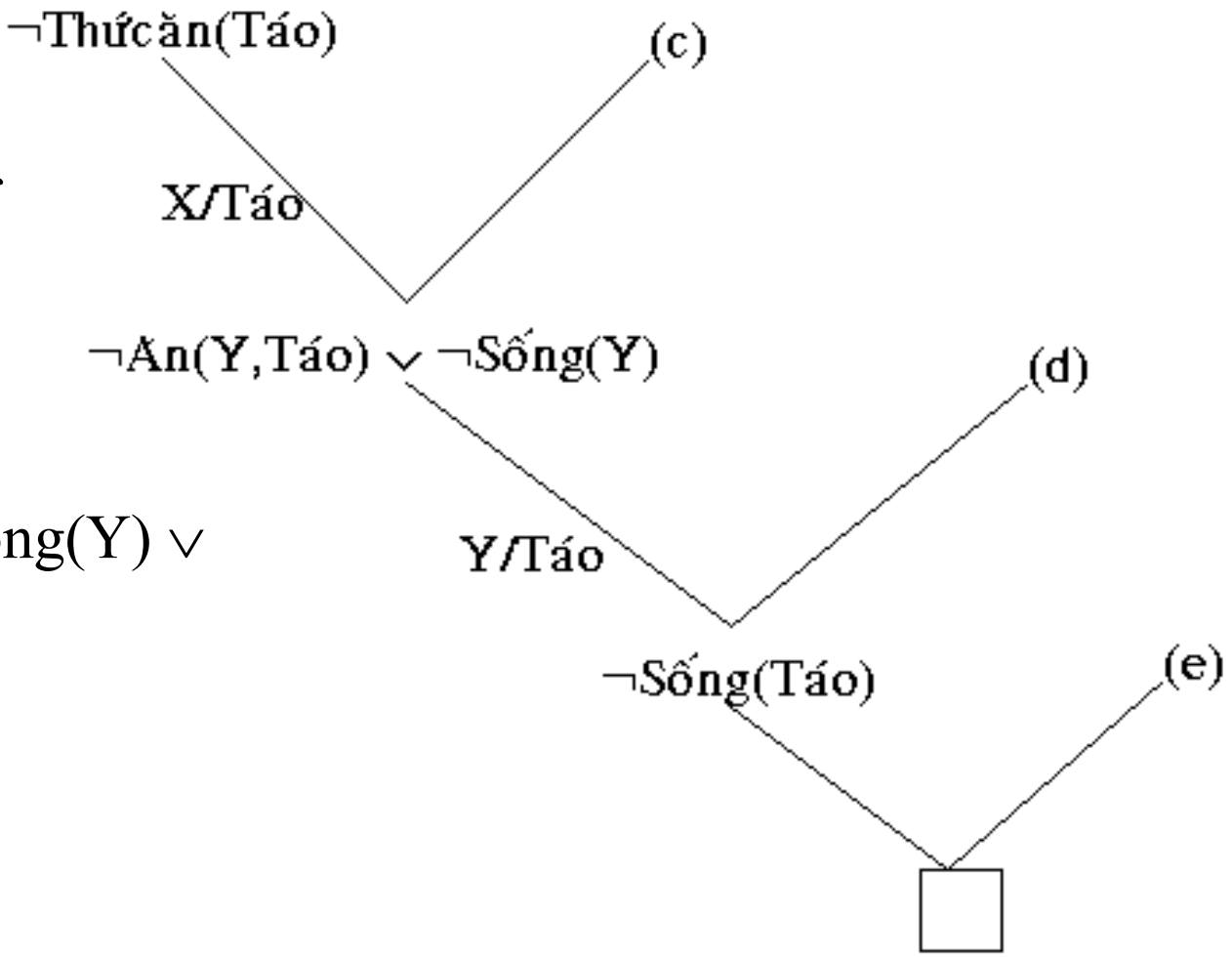
(a) Thứcăn(Cam)

(b) Thứcăn(Vịt)

(c) ¬Ăn(Y, X) ∨ ¬Sống(Y) ∨ Thứcăn(X)

(d) Ăn(An, Táo)

(e) Sống(An)



4. BIỂU DIỄN TRI THỨC BẰNG LUẬT SINH

a.Khái niệm:

Các luật sinh có dạng: **P1 ∧ P2 ∧ P3 ∧ … ∧ Pm → Q**.

Tùy thuộc vào bản chất của lĩnh vực đang quan tâm mà có những ngữ nghĩa khác nhau về luật sinh:

Trong logic vị từ:

P1, P2, …, Pm, Q : là những biểu thức logic

→ : phép kéo theo

Trong ngôn ngữ lập trình: if P1 and P2 and … and Pm then Q

Trong ngôn ngữ tự nhiên. Ví dụ: one → một.

Trong hệ chuyên gia (Expert System):

+ Cơ sở dữ liệu các sự kiện: F = $\{f_1, f_2, \ldots, f_k\}$ (F: Fact – Sự kiện)

+ Cơ sở luật sinh: $f_{i1} \wedge f_{i2} \wedge \ldots \wedge f_{ik} \rightarrow Q$ (R: Rule – Luật)

4. BIỂU DIỄN TRI THỨC BẰNG LUẬT SINH (tt)

Ví dụ: Cho một cơ sở tri thức sau:

+ Cơ sở sự kiện: H, K

+ Tập các luật (quy tắc):

(R1): $A \rightarrow E$

(R2): $B \rightarrow D$

(R3): $H \rightarrow A$

(R4): $E \wedge G \rightarrow C$

(R5): $E \wedge K \rightarrow B$

(R6): $D \wedge E \wedge K \rightarrow C$

(R7): $G \wedge K \wedge F \rightarrow A$

b.Cơ chế suy luận trên các luật sinh:

****Suy luận tiến:***

là quá trình suy luận xuất phát từ một số sự kiện ban đầu, xác định các sự kiện có thể được "sinh" ra từ sự kiện này.

Sự kiện ban đầu : H, K

Ta có: {H, K}

Từ (R3): $H \rightarrow A$ thì {A, H, K}

(R1): $A \rightarrow E$ thì {A, E, H, K}

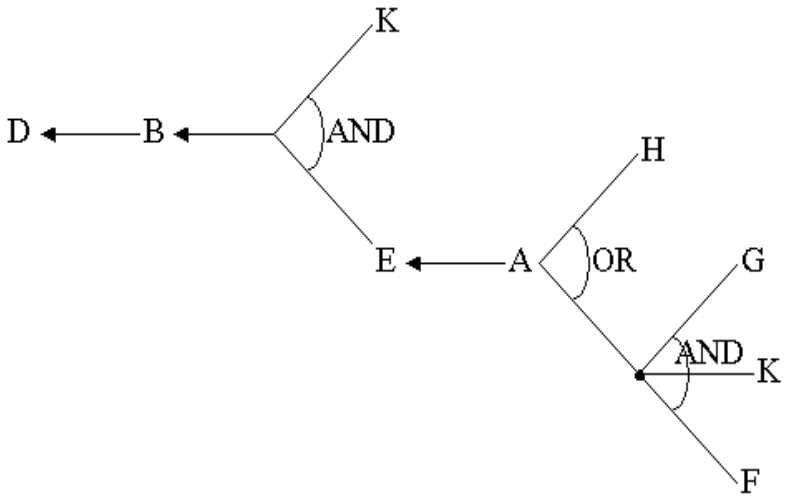
(R5): $R \wedge K \rightarrow B$ thì {A, B, E, H, K}

(R2): $B \rightarrow D$ thì {A, B, D, E, H, K}

(R6): $D \wedge E \wedge K \rightarrow C$ thì {A, B, C, D, E, H, K}

b.Cơ chế suy luận trên các luật sinh:

***Suy diễn lùi :** là quá trình suy luận ngược xuất phát từ một số sự kiện ban đầu, ta tìm kiếm các sự kiện đã "sinh" ra sự kiện này.



(R1): $A \rightarrow E$
(R2): $B \rightarrow D$
(R3): $H \rightarrow A$
(R4): $E \wedge G \rightarrow C$
(R5): $E \wedge K \rightarrow B$
(R6): $D \wedge E \wedge K \rightarrow C$
(R7): $G \wedge K \wedge F \rightarrow A$

c.Vấn đề tối ưu luật :

Rút gọn bên phải

Luật sau hiển nhiên đúng :

A V B -> A

Do đó luật

A V B -> A -> C

Là hoàn toàn tương đương với

A V B -> C

Quy tắc rút gọn : Có thể loại bỏ những sự kiện bên vế phải nếu những sự kiện đó đã xuất hiện bên vế trái. Nếu sau khi rút gọn mà vế phải trở thành rỗng thì luật đó là luật hiển nhiên. Ta có thể loại bỏ các luật hiển nhiên ra khỏi tri thức.

c.Vấn đề tối ưu luật (tt):

Rút gọn bên trái

Xét các luật :

(L1) A, B -> C {sự kiện B trong luật là dư thừa, và có thể loại bỏ được}

(L2) A -> X

(L3) X -> C

Luật A, B -> C có thể được thay thế bằng luật A -> C mà không làm ảnh hưởng đến các kết luận.

Phân rã và kết hợp luật

Ví dụ: Luật A V B -> C

Tương đương với hai luật

A -> C

B -> C

c.Vấn đề tối ưu luật (tt):

Luật thừa

Một luật dẫn A -> B được gọi là thừa nếu có thể suy ra luật này từ những luật còn lại.

Ví dụ : trong tập các luật gồm {A -> B, B -> C, A -> C} thì luật thứ 3 là luật thừa vì nó có thể được suy ra từ 2 luật còn lại.

Thuật toán tối ưu tập luật

B1 : Rút gọn vế phải

B2 : Phân rã các luật

B3 : Loại bỏ luật thừa

B4 : Rút gọn vế trái

Nhận xét

Ưu điểm

Biểu diễn tri thức bằng luật đặc biệt hữu hiệu trong những tình huống hệ thống cần đưa ra những hành động dựa vào những sự kiện có thể quan sát được. Nó có những ưu điểm chính yếu sau đây :

- Các luật rất dễ hiểu nên có thể dễ dàng dùng để trao đổi với người dùng (vì nó là một trong những dạng tự nhiên của ngôn ngữ).
- Có thể dễ dàng xây dựng được cơ chế suy luận và giải thích từ các luật.
- Việc hiệu chỉnh và bảo trì hệ thống là tương đối dễ dàng.
- Có thể cải tiến dễ dàng để tích hợp các luật mờ.
- Các luật thường ít phụ thuộc vào nhau.

Nhận xét (tt)

Nhược điểm

- Các tri thức phức tạp đôi lúc đòi hỏi quá nhiều (hàng ngàn) luật sinh. Điều này sẽ làm nảy sinh nhiều vấn đề liên quan đến tốc độ lẫn quản trị hệ thống.
- Thống kê cho thấy, người xây dựng hệ thống trí tuệ nhân tạo thích sử dụng luật sinh hơn tất cả phương pháp khác (dễ hiểu, dễ cài đặt) nên họ thường tìm mọi cách để biểu diễn tri thức bằng luật sinh cho dù có phương pháp khác thích hợp hơn! Đây là nhược điểm mang tính chủ quan của con người.
- Cơ sở tri thức luật sinh lớn sẽ làm giới hạn khả năng tìm kiếm của chương trình điều khiển. Nhiều hệ thống gặp khó khăn trong việc đánh giá các hệ dựa trên luật sinh cũng như gặp khó khăn khi suy luận trên luật sinh.

5. BIỂU DIỄN TRI THỨC SỬ DỤNG MẠNG NGỮ NGHĨA

Khái niệm

Mạng ngữ nghĩa là một phương pháp biểu diễn tri thức đầu tiên và cũng là phương pháp dễ hiểu nhất đối với chúng ta. Phương pháp này sẽ biểu diễn tri thức dưới dạng một đồ thị, trong đó đỉnh là các đối tượng (khái niệm) còn các cung cho biết mối quan hệ giữa các đối tượng (khái niệm) này.

Mạng ngữ nghĩa sử dụng công cụ là đồ thị nên nó thừa hưởng tất cả những mặt mạnh của công cụ đồ thị. Các thuật toán đã được cài đặt và phát triển trên máy tính, khi áp dụng chúng ta có thể giải quyết nhiều vấn đề khác nhau ở trên mạng. Cho đến nay mạng ngữ nghĩa được ứng dụng nhiều trong hai lĩnh vực:

+Xử lý ngữ nghĩa tự nhiên.

+ Giải các bài toán thông minh.

Ví dụ:Xây dựng mạng ngữ nghĩa để giải tam giác

Đặt vấn đề:

Có 22 yếu tố liên quan đến cạnh và góc của tam giác. Để xác định một tam giác hay để xây dựng một 1 tam giác ta cần có 3 yếu tố trong đó phải có yếu tố cạnh. Như vậy có khoảng C^3_{22} -1 (khoảng vài ngàn) 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, ta phải 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 :

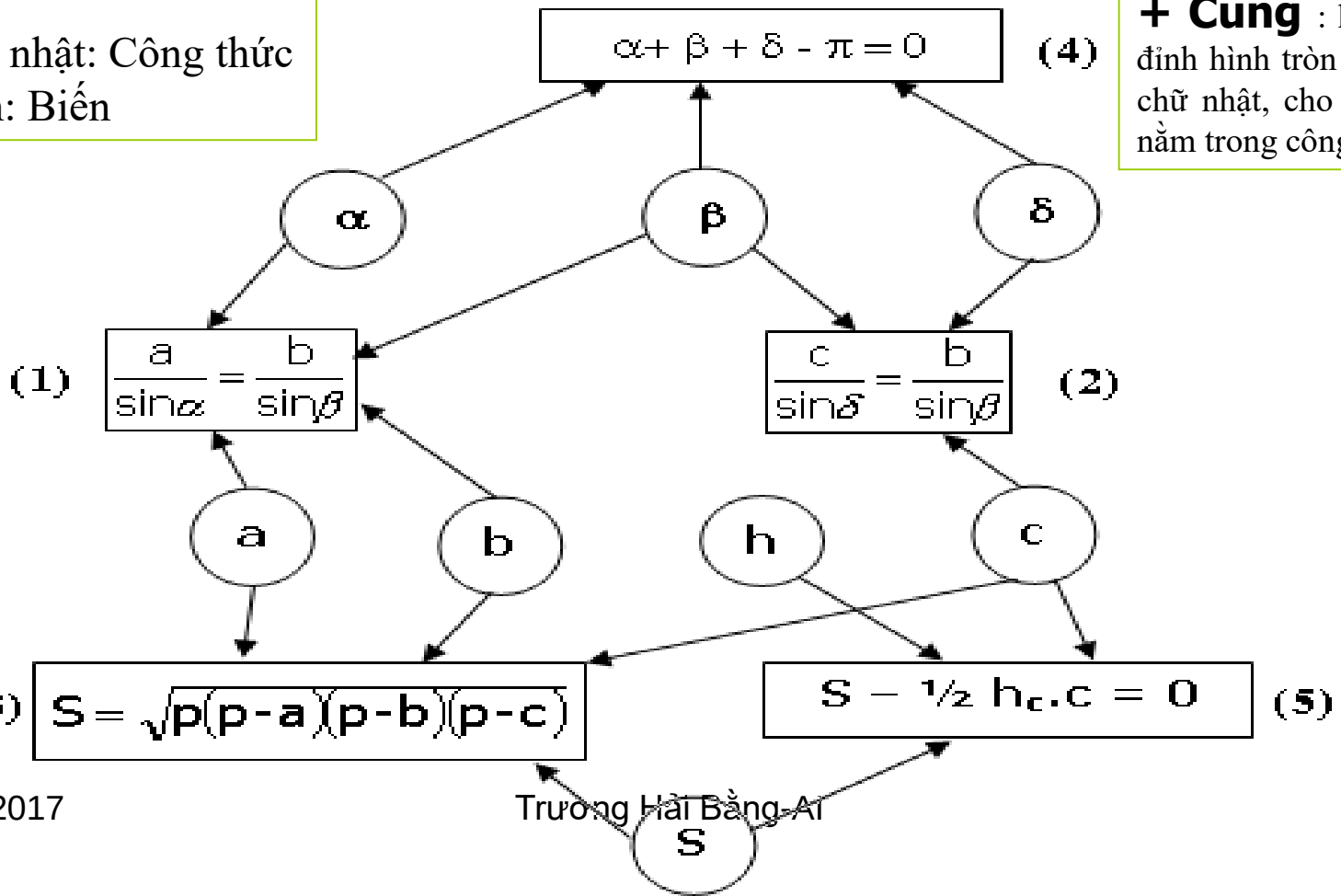
Bài toán:"Cho hai góc α, β và chiều dài cạnh a của tam giác. Tính đường cao h_C".

Cơ chế suy diễn thực hiện theo thuật toán "loang" đơn giản sau :
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:Xây dựng mạng ngữ nghĩa để giải tam giác (tt)

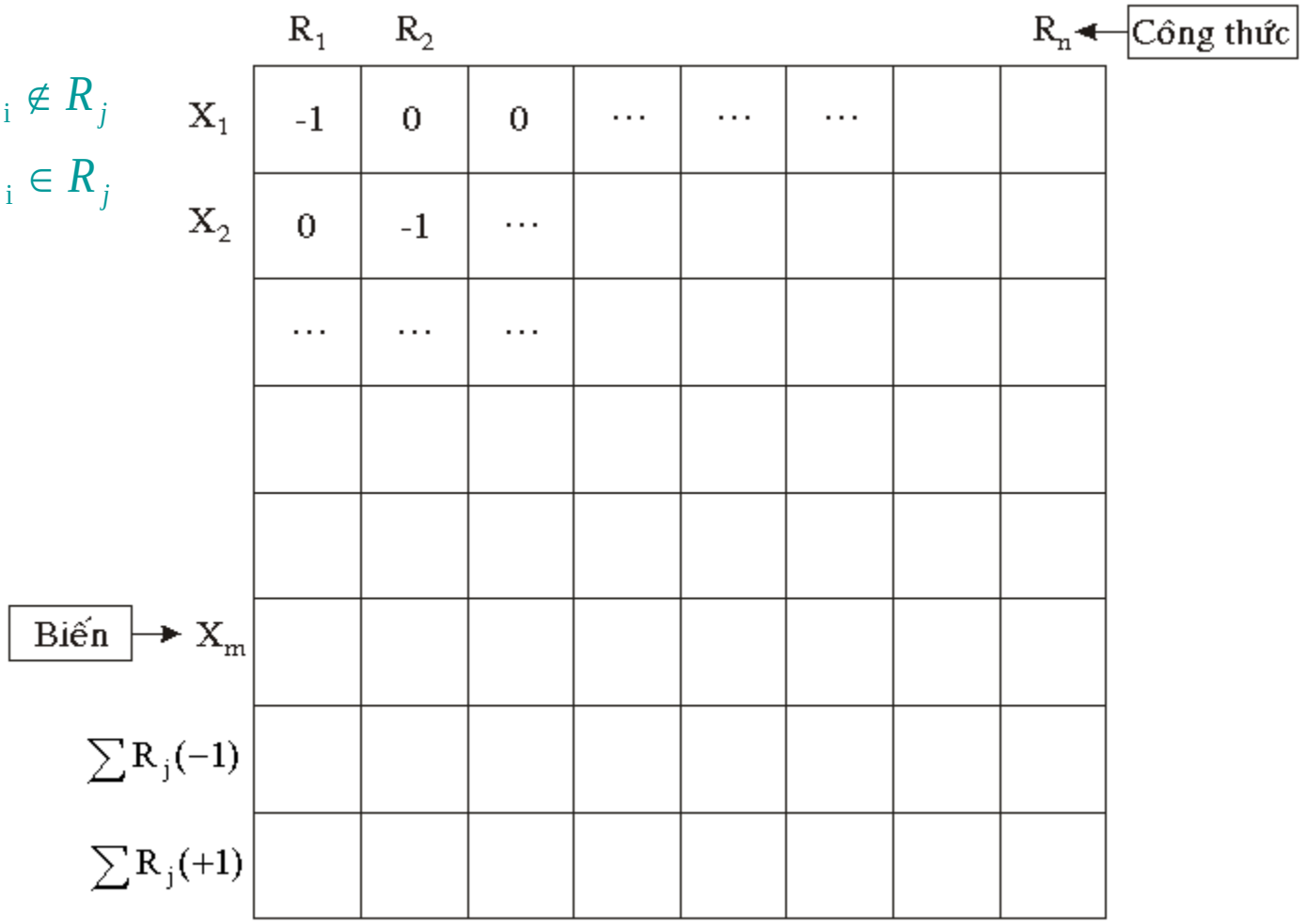
+ Đỉnh:
Hình chử nhật: Công thức
Hình tròn: Biến

+ Cung : luôn hướng từ đỉnh hình tròn lên đỉnh hình chữ nhật, cho biết biến nào nằm trong công thức nào.



Cài đặt thuật toán:

$$X_i R_j = \begin{cases} 0 & \text{if } X_i \notin R_j \\ -1 & \text{if } X_i \in R_j \end{cases}$$



Cài đặt thuật toán:

+ Nhập các biến X_i cho trước (kích hoạt): khi đó những công thức nào có chứa biến này thì cho giá trị là 1 (đổi từ –1 thành 1).
+ Tính $\sum R_j(+1)$: Số biến đã biết trong công thức.
+ Tính: IF ($\sum R_j(-1) - \sum R_j(+1) = 1$): công thức R_j đã biết
ELSE Công thức chưa được biết.
Nếu toàn bộ đều ≠ 1 thì dữ liệu chưa đủ.
+ Nếu công thức = 1 $\Rightarrow$ công thức đó được kích hoạt. Các biến liên hệ với công thức này (duyệt theo cột) sẽ được kích hoạt từ –1 sang 1.
+ Duyệt tiếp để xác định tiếp các công thức liên quan.

Ban đầu

	(1)	(2)	(3)	(4)	(5)
α	-1	0	0	-1	0
β	-1	-1	0	-1	0
δ	0	-1	0	-1	0
a	-1	0	-1	0	0
b	-1	-1	-1	0	0
c	0	-1	-1	0	-1
S	0	0	-1	0	-1
hC	0	0	0	0	-1

đỉnh **α, β , a** của đồ thị được kích hoạt.

	(1)	(2)	(3)	(4)	(5)
α	1	0	0	1	0
β	1	1	0	1	0
δ	0	-1	0	-1	0
a	1	0	1	0	0
b	-1	-1	-1	0	0
c	0	-1	-1	0	-1
S	0	0	-1	0	-1
hC	0	0	0	0	-1

Trên cột **(1)**, hiệu (1+1+1 – (-1)) = 4 nên dòng **b** sẽ được kích hoạt

	(1)	(2)	(3)	(4)	(5)
α	1	0	0	1	0
β	1	1	0	1	0
δ	0	-1	0	-1	0
a	1	0	1	0	0
b	1	1	1	0	0
c	0	-1	-1	0	-1
S	0	0	-1	0	-1
hC	0	0	0	0	-1

Trên cột (4), hiệu (1+1 – (-1)) = 3 nên dòng **d** sẽ được kích hoạt.

	(1)	(2)	(3)	(4)	(5)
α	1	0	0	1	0
β	1	1	0	1	0
δ	0	1	0	1	0
a	1	0	1	0	0
b	1	1	1	0	0
c	0	-1	-1	0	-1
S	0	0	-1	0	-1
hC	0	0	0	0	-1

Trên cột (2), hiệu (1+1+1 – (1)) = 4 nên dòng **c** được kích hoạt.

	(1)	(2)	(3)	(4)	(5)
α	1	0	0	1	0
β	1	1	0	1	0
δ	0	1	0	1	0
a	1	0	1	0	0
b	1	1	1	0	0
c	0	1	1	0	1
S	0	0	-1	0	-1
hC	0	0	0	0	-1

Trên cột (3), hiệu (1+1+1 – (-1)) = 4 nên dòng **S** được kích hoạt.

	(1)	(2)	(3)	(4)	(5)
α	1	0	0	1	0
β	1	1	0	1	0
δ	0	1	0	1	0
a	1	0	1	0	0
b	1	1	1	0	0
c	0	1	1	0	1
S	0	0	1	0	1
hC	0	0	0	0	-1

Trên cột (5), hiệu (1+1 – (-1)) = 3 nên dòng **hC** được kích hoạt.

	(1)	(2)	(3)	(4)	(5)
α	1	0	0	1	0
β	1	1	0	1	0
δ	0	1	0	1	0
a	1	0	1	0	0
b	1	1	1	0	0
c	0	1	1	0	1
S	0	0	1	0	1
hC	0	0	0	0	1

6. BIỂU DIỄN TRI THỨC BẰNG FRAME

a. Khái niệm

•Frame là một cấu trúc dữ liệu chứa đựng tất cả những tri thức liên quan đến một đối tượng cụ thể nào đó.

•Frame là nguồn gốc của lập trình hướng đối tượng.

•Một frame bao hàm trong nó một khối lượng tương đối lớn tri thức về một đối tượng, sự kiện, vị trí, tình huống hoặc những yếu tố khác

6. BIỂU DIỄN TRI THỨC BẰNG FRAME (tt)

b. Cấu trúc của frame

Mỗi một frame mô tả một *đối tượng (object)*. Một frame bao gồm 2 thành phần cơ bản là **slot** và **facet.** Một **slot** là một thuộc tính đặc tả đối tượng được biểu diễn bởi frame. Ví dụ : trong frame mô tả xe hơi, có hai slot là *trọng lượng* và *loại*

Ví dụ :

Một số facet thường gặp.
Value: giá trị. Cho biết giá trị của thuộc tính đó
Default :giá trị mặc định
Range: miền giá trị
If added : mô tả một hành động sẽ được thi hành khi một giá trị trong slot được thêm vào (hoặc được hiệu chỉnh). Thủ tục thường được viết dưới dạng một script.
If needed : được sử dụng khi slot không có giá trị nào. Facet mô tả một hàm để tính ra giá trị của slot.

Frame MÁY
Xy-lanh : 3.19 inch
Tỷ lệ nén : 3.4 inche
Xăng : TurboCharger
Mã lực : 140 hp

7. BIỂU DIỄN TRI THỨC BẰNG SCRIPT

Script là một cách biểu diễn tri thức tương tự như frame nhưng thay vì đặc tả một đối tượng, nó mô tả *một chuỗi các sự kiện*. Để mô tả chuỗi sự kiện, script sử dụng một dãy các **slot** chứa thông tin về các con người, đối tượng và hành động liên quan đến sự kiện đó.

Các thành phần của Script

Điều kiện vào (entry condition): mô tả những tình huống hoặc điều kiện cần được thỏa mãn trước khi các sự kiện trong script có thể diễn ra.

Role (diễn viên): là những con người có liên quan trong script.

Prop (tác tố): là tất cả những đối tượng được sử dụng trong các chuỗi sự kiện sẽ diễn ra.

Scene(Tình huống) : là chuỗi sự kiện thực sự diễn ra.

Result (Kết quả) : trạng thái của các Role sau khi script đã thi hành xong.

Track (phiên bản) : mô tả một biến thể (hoặc trường hợp đặc biệt) có thể xảy ra trong đoạn script.

***Ví dụ* Script "nhà hàng"**

Phiên bản : Nhà hàng bán thức ăn nhanh.
Diễn viên : Khách hàng,Người phục vụ.
Tác tố : Bàn phục vụ. Chỗ ngồi. Khay đựng thức ăn …
Điều kiện vào : Khách hàng đóiKhách hàng có đủ tiền để trả.
Tình huống 1 : Vào nhà hàng
Tình huống 2: Kêu món ăn.
Tình huống 3: Khách hàng dùng món ăn
Tình huống 4 : Ra về
Kết quả : Khách hàng không còn đói.
Khách hàng còn ít tiền hơn ban đầu.
Khách hàng vui vẻ *
Khách hàng bực mình *
Khách hàng quá no

8. *PHỐI HỢP NHIỀU CÁCH BIỂU DIỄN TRI THỨC*

P.Pháp	Ưu điểm	Nhược điểm
Luật sinh	Cú pháp đơn giản, dễ hiểu, diễn dịch đơn giản, tính đơn thể cao, linh động (dễ điều chỉnh).	Rất khó theo dõi sự phân cấp, không hiệu quả trong những hệ thống lớn, không thể biểu diễn được mọi loại tri thức, rất yếu trong việc biểu diễn các tri thức dạng mô tả, có cấu trúc.
Mạng ngữ nghĩa	Dễ theo dõi sự phân cấp, sẽ dò theo các mối liên hệ, linh động	Ngữ nghĩa gắn liền với mỗi đỉnh có thể nhập nhằng, khó xử lý các ngoại lệ, khó lập trình.

8. *PHỐI HỢP NHIỀU CÁCH BIỂU DIỄN TRI THỨC (tt)*

Frame	Có sức mạnh diễn đạt tốt, dễ cài đặt các thuộc tính cho các slot cũng như các mối liên hệ, dễ dàng tạo ra các thủ tục chuyên biệt hóa, dễ đưa vào các thông tin mặc định và dễ thực hiện các thao tác phát hiện các giá trị bị thiếu sót.	Khó lập trình, khó suy diễn, thiếu phần mềm hỗ trợ.
Logic hình thức	Cơ chế suy luận chính xác (được chứng minh bởi toán học).	Tách rời việc biểu diễn và xử lý, không hiệu quả với lượng dữ liệu lớn, quá chậm khi cơ sở dữ liệu lớn

MỘT SỐ VÍ DỤ VỀ MÁY HỌC

1. Giới thiệu

Một số phương pháp máy học để tiếp thu tri thức hay tạo ra tri thức

- Học vẹt
- Học cách đề xuất
- Học bằng cách thu thập các trường hợp
- Học bằng cách xây dựng cây định danh
- Học không giám giám sát và bài tóm gom nhóm dữ liệu
- Học giám sát và bài toán phân lớp dữ liệu

1. Giới thiệu

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.

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.

1. Giới thiệu

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.

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.

1. Giới thiệu

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.

Học dựa trên tình huống

Bấy 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.

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.

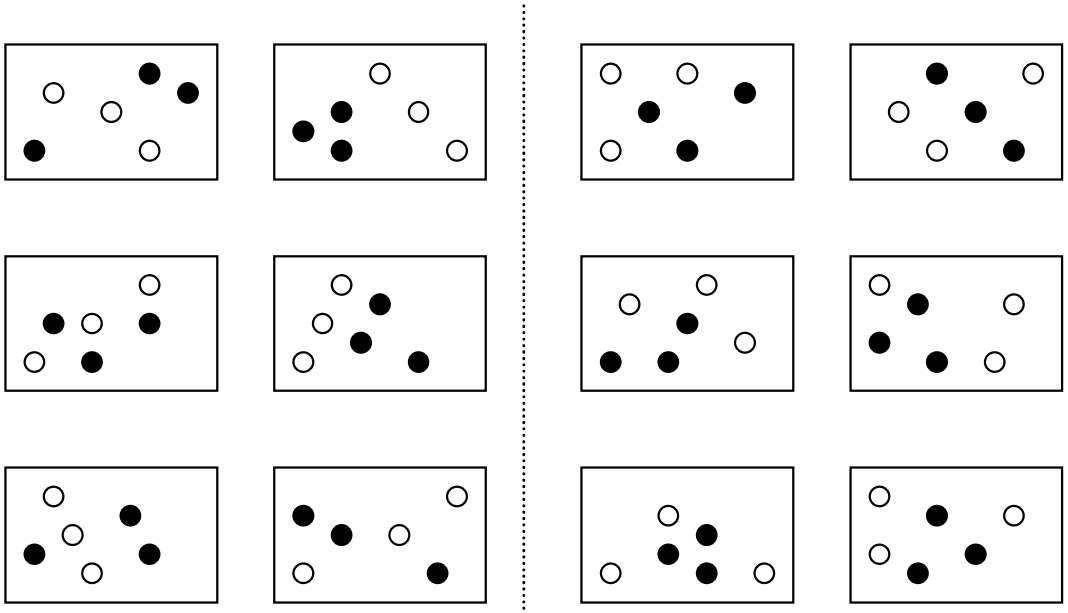
2. Một số ví dụ

Học qua logic:

Bongard (1970) là người đầu tiên ứng dụng các toán tử logic để học và nhận dạng các đối tượng hình ảnh.

Ý tưởng: Tìm quan hệ đơn giản nhất trong số các quan hệ có thể sử dụng để học và nhận dạng các hình ảnh.

2. Một số ví dụ



Chúng ta có thể quan sát thấy các hình vẽ thuộc lớp A có 3 vòng trắng luôn luôn nằm trên một đường thẳng.

2. Một số ví dụ

<u>Vấn đề đặt ra</u>:

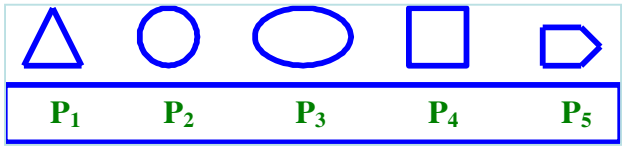
-Tìm quan hệ đơn giản nhất có thể phân biệt được các hình ảnh.

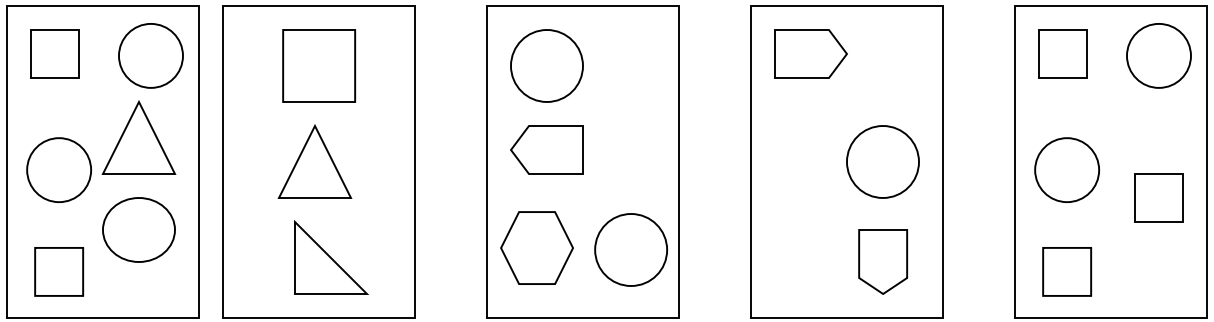
Bongard đã dùng bảng logic "mô tả – quan hệ" để dẫn xuất ra các mệnh đề logic:

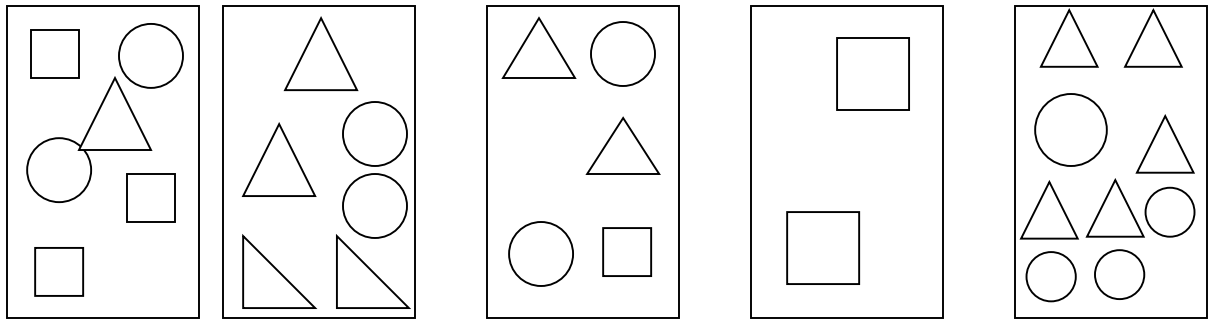
$$\varphi = \vee(\varphi_1 \wedge \varphi_2 \wedge \ldots \wedge \varphi_n)$$

ϕ có thể dùng để phân biệt 2 lớp E và E' nếu $\phi(E)$ và $\phi(E')$ đối ngẫu nhau.

2. Một số ví dụ







2. Một số ví dụ

Các đối tượng trong mẫu:

	P_1	P_2	P_3	P_4	P_5	
1	1	1	1	1	0	$P_1P_2P_3P_4\overline{P_5}$
2	1	0	0	1	0	$P_1\overline{P_2}\,\overline{P_3}P_4\overline{P_5}$
3	0	1	0	0	1	$\overline{P_1}P_2\overline{P_3}\,\overline{P_4}P_5$
4	0	1	0	0	1	$\overline{P_1}P_2\overline{P_3}\,\overline{P_4}P_5$
5	0	1	0	1	0	$\Rightarrow \overline{P_1}P_2\overline{P_3}P_4\overline{P_5}$
6	1	1	0	1	0	$P_1P_2\overline{P_3}P_4\overline{P_5}$
7	1	1	0	0	0	$P_1P_2\overline{P_3}\,\overline{P_4}\,\overline{P_5}$
8	1	0	0	1	0	$P_1\overline{P_2}\,\overline{P_3}P_4\overline{P_5}$
9	0	0	0	1	0	$\overline{P_1}\,\overline{P_2}\,\overline{P_3}P_4\overline{P_5}$
10	1	1	0	0	0	$P_1P_2\overline{P_3}\,\overline{P_4}\,\overline{P_5}$

2. Một số ví dụ

Sau khi tính tổng và rút gọn lại được:

$$\overline{P_1}.P_2 + P_1.(P_2.P_3 + \overline{P_2}.\overline{P_3})$$

$$x \in \varphi(A)\begin{cases}\overline{P_1}.P_2 \\ P_1.P_2.P_3 \\ P_1.\overline{P_2}.\overline{P_3}\end{cases}$$

3. Học bằng cách xây dựng cây định danh

Cây định danh: Là một dạng của cây quyết định, trong đó mỗi tập các kết luận có thể xảy ra được thiết lập một cách ngầm định bởi một danh sách các mẫu mà chúng được phân vào một lớp đã biết.

3. Học bằng cách xây dựng cây định danh

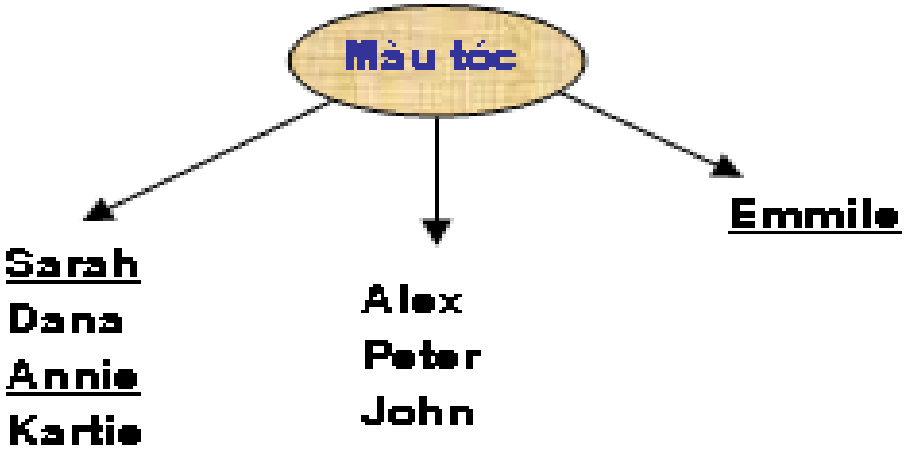
Ví dụ có bảng dữ liệu quan sát

Tên	Tóc	Ch.Cao	Cân Nặng	Dùng kem?	Kết quả
Sarah	Vàng	T.Bình	Nhẹ	Không	Cháy
Dana	Vàng	Cao	T.Bình	Có	Không
Alex	Nâu	Thấp	T.Bình	Có	Không
Annie	Vàng	Thấp	T.Bình	Không	Cháy
Emilie	Đỏ	T.Bình	Nặng	Không	Cháy
Peter	Nâu	Cao	Nặng	Không	Không
John	Nâu	T.Bình	Nặng	Không	Không
Kartie	Vàng	Thấp	Nhẹ	Có	Không

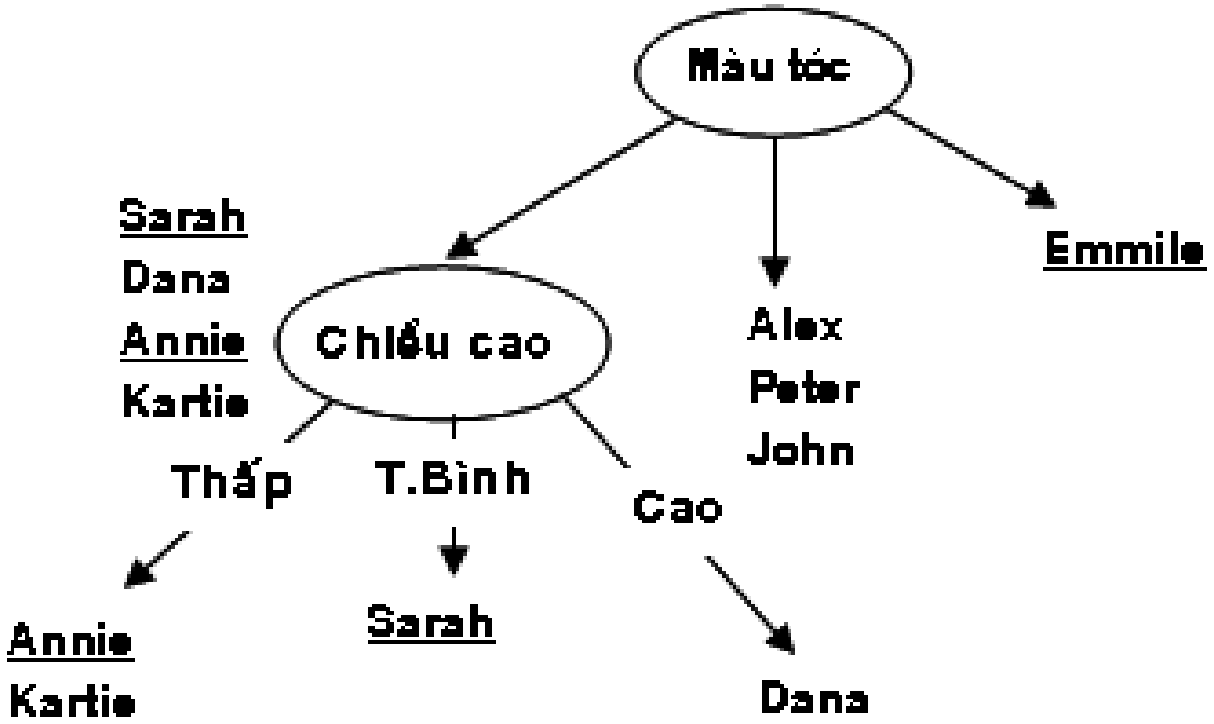
3. Học bằng cách xây dựng cây định danh

Ta gọi tính chất cháy nắng hay không cháy nắng là thuộc tính quan tâm ***(thuộc tính mục tiêu).*** Như vậy, trong trường hợp này, tập R của chúng ta chỉ gồm có hai phần tử {"cháy nắng", "bình thường"}. Còn tập P là tất cả những người được liệt kê trong bảng dưới (8 người) Chúng ta quan sát hiện tượng cháy nắng dựa trên 4 thuộc tính sau : ***chiều cao (cao, trung bình, thấp), màu tóc (vàng, nâu, đỏ) cân nặng (nhẹ, TB, nặng),*** dùng kem ***(có, không),.*** Ta gọi các thuộc tính này gọi là ***thuộc tính dẫn xuất.***

3.1. Đâm chồi



3.1. Đâm chồi


Màu tóc
Sarah
Dana
Annie
Kartie
Chiều cao
Emmile
Alex
Peter
John
Thấp
T.Bình
Cao
Annie
Kartie
Sarah
Dana

3.2. Phương án chọn thuộc tính phân hoạch

Vấn đề mà chúng ta gặp phải cũng tương tự như bài toán tìm kiếm : "Đứng trước một ngã rẽ, ta cần phải đi vào hướng nào?". Hai phương pháp đánh giá dưới đây sẽ giúp ta chọn được thuộc tính phân hoạch tại mỗi bước xây dựng cây định danh.

Thuật toán Quinlan

•Quinlan quyết định thuộc tính phân hoạch bằng cách xây dựng các ***vector đặc trưng*** cho mỗi giá trị của từng thuộc tính dẫn xuất và thuộc tính mục tiêu.

• Cho một bảng quan sát là tập hợp các mẫu với các thuộc tính nhất định của các đối tượng nào đó.

• Sử dụng một độ đo để định lượng và đề ra một tiêu chuẩn nhằm chọn lựa một thuộc tính mang tính chất "phân loại" để phân bảng này thành các bảng con nhỏ hơn sao cho từ mỗi bảng con này dễ dàng phân tích tìm ra quy luật chung .

Thuật toán Quinlan (tt)

$$\text{Cháy nắng} = \frac{\text{Tổng số người tóc vàng cháy nắng}}{\text{Tổng số người tóc vàng}}$$

$$\text{Không cháy nắng} = \frac{\text{Tổng số người tóc vàng không cháy nắng}}{\text{Tổng số người tóc vàng}}$$

Thuật toán Quinlan (tt)

VTóc (vàng) = (cháy nắng, không cháy nắng)

Số người tóc vàng là : **4**

Số người tóc vàng và cháy nắng là : **2**

Số người tóc vàng và không cháy nắng là : **2**

Do đó

$$VTóc(vàng) = (2/4 , 2/4) = (0.5, 0.5)$$

Tương tự

$$VTóc(nâu) = (0/3, 3/3) = (0,1) \text{ (vector đơn vị)}$$

$$VTóc(đỏ) = (1/1, 0/1) = (1,0) \text{ (vector đơn vị)}$$

Tổng số vector đơn vị của thuộc tính tóc là 2

Thuật toán Quinlan (tt)

Các thuộc tính khác được tính tương tự, kết quả như sau :

VC.Cao(Cao) = (0/2,2/2) = **(0,1)**

VC.Cao(T.B) = (2/3,1/3)

VC.Cao(Thấp) = (1/3,2/3)

VC.Nặng (Nhẹ) = (1/2,1/2)

VC.Nặng (T.B) = (1/3,2/3)

VC.Nặng (Nặng) = (1/3,2/3)

VKem (Có) = (3/3,0/3) = **(1,0)**

VKem (Không) = (3/5,2/5)

Như vậy thuộc tính màu tóc có số vector đơn vị nhiều nhất nên sẽ được chọn để phân hoạch.

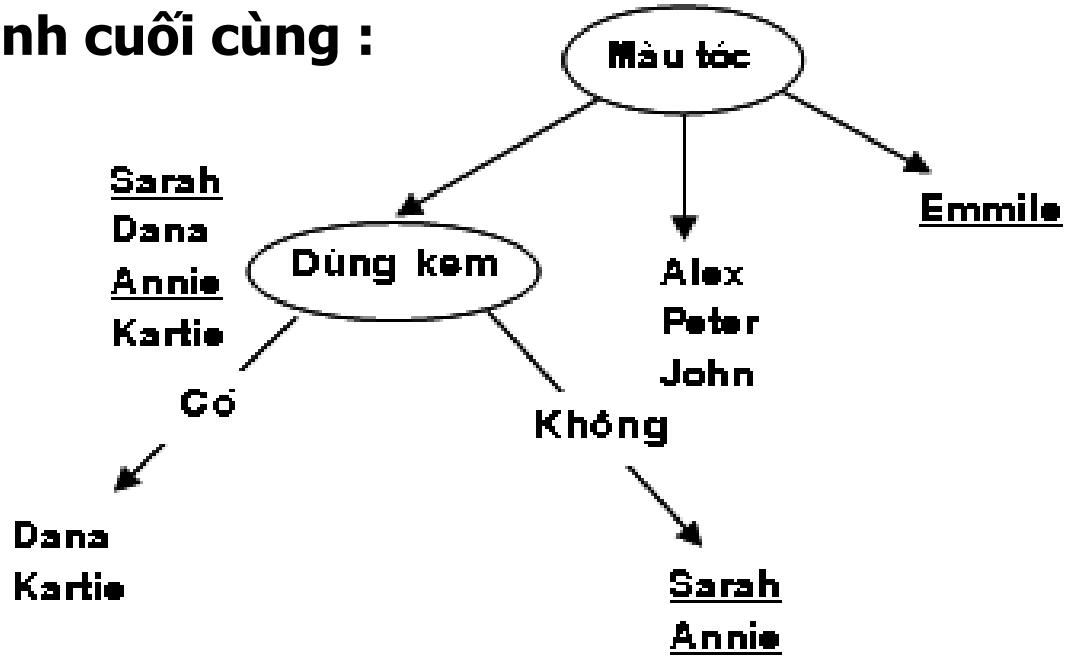
Thuật toán Quinlan (tt)

Sau khi phân hoạch theo màu tóc xong, chỉ có phân hoạch theo tóc vàng (Pvàng) là còn chứa những người cháy nắng và không cháy nắng nên ta sẽ tiếp tục phân hoạch tập này. Ta sẽ thực hiện thao tác tính vector đặc trưng tương tự đối với các thuộc tính còn lại *(chiều cao, cân nặng, dùng kem)*. Trong phân hoạch Pvàng, tập dữ liệu của chúng ta còn lại là :

Tên	**Ch.Cao**	**Cân Nặng**	**Dùng kem?**	**Kết quả**
Sarah	T.Bình	Nhẹ	Không	Cháy
Dana	Cao	T.Bình	Có	Không
Annie	Thấp	T.Bình	Không	Cháy
Kartie	Thấp	Nhẹ	Có	Không

Thuật toán Quinlan (tt)

Kết quả Cây định danh cuối cùng :



Phương pháp độ đo hỗn loạn

Thay vì phải xây dựng các vector đặc trưng như phương pháp của Quinlan, ứng với mỗi thuộc tính dẫn xuất ta chỉ cần tính ra độ đo hỗn loạn và lựa chọn thuộc tính nào có độ đo hỗn loại là thấp nhất. Công thức tính như sau :

$$T_A = \sum_j \left(\frac{b_j}{b_t} \sum_i \left(\frac{-b_{ji}}{b_j} \log_2 \left(\frac{-b_{ji}}{b_j} \right) \right) \right)$$

Phương pháp độ đo hỗn loạn

Trong đó:

A: thuộc tính cần tính độ hỗn loạn

bt : tổng số phần tử có trong phân hoạch

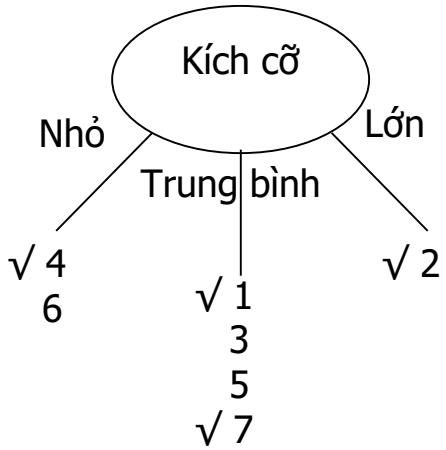
bj : tổng số phần tử có thuộc tính A với giá tri của thuộc tính là j

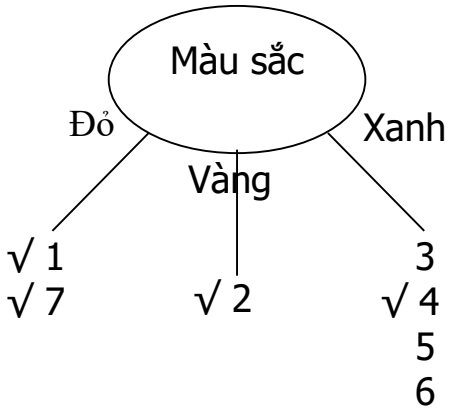
bji: tổng số phần tử có thuộc tính A với giá tri của thuộc tính là j và thuộc tính mục tiêu là i

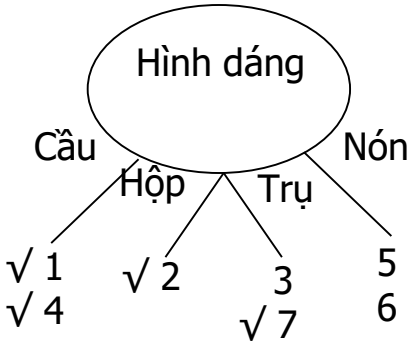
Ví dụ 1: cho bảng quan sát

STT	Kích cỡ	Màu sắc	Hình dáng	Quyết định
1	Trung bình	Đỏ	Cầu	Mua
2	Lớn	Vàng	Hộp	Mua
3	Trung bình	Xanh	Trụ	Không mua
4	Nhỏ	Xanh	Cầu	Mua
5	Trung bình	Xanh	Nón	Không mua
6	Nhỏ	Xanh	Nón	Không mua
7	Trung bình	Đỏ	Trụ	Mua

Ví dụ (tt)


Kích cỡ
Nhỏ
Trung bình
Lớn
√ 4
6
√ 1
3
5
√ 7
√ 2


Màu sắc
Đỏ
Vàng
Xanh
√ 1
√ 7
√ 2
3
√ 4
5
6


Hình dáng
Cầu
Hộp
Trụ
Nón
√ 1
√ 4
√ 2
3
√ 7
5
6

Ví dụ (tt)

Độ hỗn loạn TB kích cỡ:

$$= \frac{2}{7}\left(-\frac{1}{2}\times\log_2\frac{1}{2}-\frac{1}{2}\times\log_2\frac{1}{2}\right)+\frac{4}{7}\left(-\frac{2}{4}\times\log_2\frac{2}{4}-\frac{2}{4}\times\log_2\frac{2}{4}\right)+\frac{1}{7}\left(-\frac{1}{1}\times\log_2\frac{1}{1}-0\right)=\frac{2}{7}+\frac{4}{7}=\frac{6}{7}$$

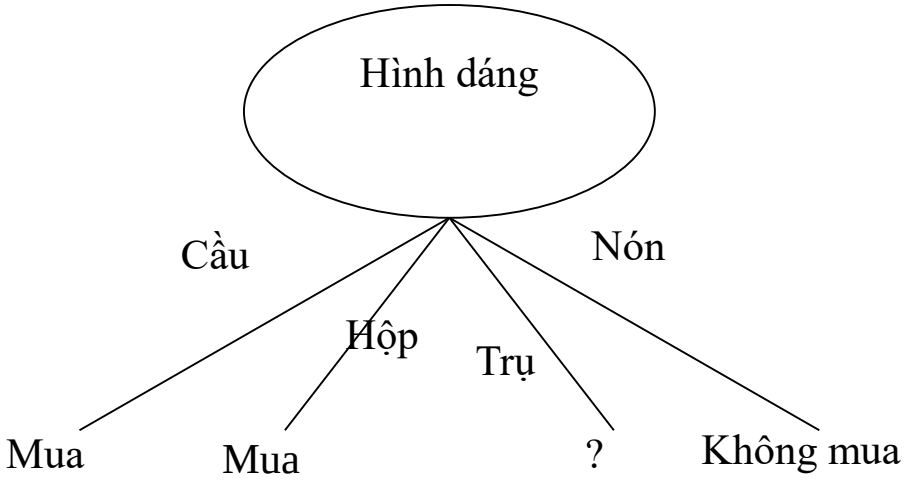
Độ hỗn loạn TB màu sắc: $= 0+0+\frac{4}{7}\left(-\frac{1}{4}\times\log_2\frac{1}{4}-\frac{3}{4}\times\log_2\frac{3}{4}\right)\approx 0.46$

Độ hỗn loạn TB hình dáng:

$$= 0+0+\frac{2}{7}\left(-\frac{1}{2}\times\log_2\frac{1}{2}-\frac{1}{2}\times\log_2\frac{1}{2}\right)+0=\frac{2}{7}$$

Ví dụ (tt)

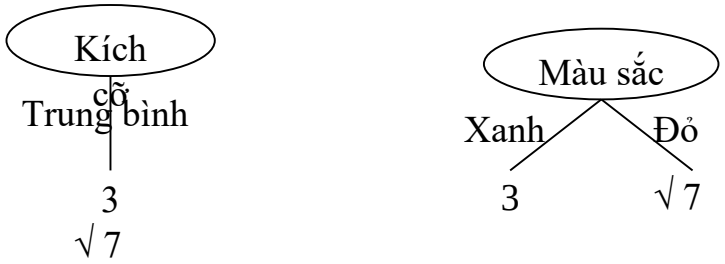
Chọn thuộc tính hình dáng vì có độ hỗn loạn TB nhỏ nhất:



Ví dụ (tt)

Sau khi test lần 1 xong, ta đã loại ra 5 mẫu ổn định
=> có 1 bảng nhỏ hơn:

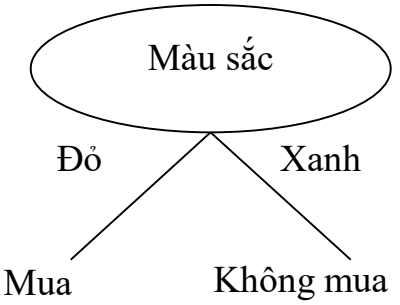
STT	Kích cỡ	Màu sắc	Quyết định
3	Trung bình	Xanh	Không mua
7	Trung bình	Đỏ	Mua



Độ hỗn loạn trung bình kích cỡ:=1
Độ hỗn loạn trung bình màu sắc:=0

Ví dụ (tt)

Chọn thuộc tính màu sắc vì có độ hỗn loạn TB nhỏ nhất:



Cây quyết định: