

- I. Tìm kiếm
1. Định nghĩa không gian trạng thái.

Không gian trạng thái (state space) trong AI là tập hợp tất cả các **states** (trạng thái) có thể có của một bài toán, cùng với các **transitions** (chuyển đổi) giữa chúng thông qua các **legal moves** (hành động hợp lệ). Nó được biểu diễn như một **graph**, trong đó:

- **Nodes**: Đại diện cho các **states**, mỗi **state** là một cấu hình cụ thể của bài toán (ví dụ: vị trí các quân cờ trong **chess**, hoặc lượng nước trong bình trong **Water Jug Problem**).
- **Edges**: Đại diện cho các **legal moves**, tức là các hành động chuyển từ một **state** này sang **state** khác (ví dụ: di chuyển một quân cờ, đổ nước từ bình này sang bình kia).
- **Initial State**: Trạng thái bắt đầu của bài toán.
- **Goal State**: Trạng thái mục tiêu cần đạt được (giải pháp của bài toán).

Ví dụ: Trong **Water Jug Problem** (bình nước 4 lít và 3 lít):

- **State**: (x, y) , với x là lượng nước trong bình 4 lít, y là lượng nước trong bình 3 lít.
- **State Space**: Tất cả các cặp (x, y) khả thi (ví dụ: $(0,0)$, $(4,0)$, $(2,3)$, v.v.).
- **Legal Moves**: Các hành động như đổ đầy bình, đổ hết nước, hoặc chuyển nước giữa các bình.
- **Initial State**: $(0,0)$ (cả hai bình rỗng).
- **Goal State**: $(2,n)$ (bình 4 lít chứa 2 lít).

Ý nghĩa: State space cung cấp một cách biểu diễn bài toán để áp dụng **search algorithms** (như **BFS**, **DFS**, **A***), giúp tìm đường từ **initial state** đến **goal state**. Kích thước của state space (số lượng states) quyết định độ phức tạp của bài toán.

Problem solving = Searching for the goal state State?

Initial state:

Goal state:

Legal Moves: Current state → Next state

Problem solving = Searching for the goal state

- 1 Define a state space that contains all the possible configurations of the relevant objects.
- 2 Specify the initial states.
- 3 Specify the goal states.
- 4 Specify a set of rules.

Example:

Problem 2: Water Jug Problem

"You are given two jugs, a 4-litre one and a 3-litre one. Neither has any measuring markers on it. There is a pump that can be used to fill the jugs with water. How can you get exactly 2 litres of water into 4-litre jug."

- State: (x, y)
- Initial State: $(0, 0)$
- Goal State: $(2, n)$
- Legal Moves: Current state → Next State
 - ➊ $(x, y) \rightarrow (4, y)$, if $x < 4$
 - ➋ $(x, y) \rightarrow (x, 3)$, if $y < 3$
 - ➌ $(x, y) \rightarrow (0, y)$, if $x > 0$
 - ➍ $(x, y) \rightarrow (x, 0)$, if $y > 0$
 - ➎ $(x, y) \rightarrow (4, y - (4 - x))$, if $x + y \geq 4$, $y > 0$
 - ➏ $(x, y) \rightarrow (x - (3 - y), 3)$, if $x + y \geq 3$, $x > 0$
 - ➐ $(x, y) \rightarrow (x + y, 0)$, if $x + y \leq 4$, $y > 0$
 - ➑ $(x, y) \rightarrow (0, x + y)$, if $x + y \leq 3$, $x > 0$
 - ➒ $(0, 2) \rightarrow (2, 0)$

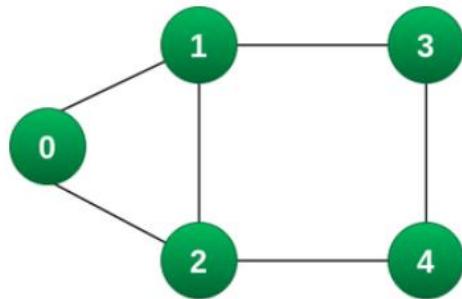
2. Tìm kiếm mù (BrFS, DFS): Vẽ cây minh họa quá trình duyệt.

2.1 Breath First Search (Tìm kiếm theo chiều rộng)

Bắt đầu từ gốc, tất cả các nút ở một cấp cụ thể được truy cập trước và sau đó các nút ở cấp tiếp theo được duyệt cho đến khi tất cả các nút được truy cập.

Để làm điều này một hàng đợi được sử dụng. Tất cả các nút chưa được truy cập liền kề ở cấp độ hiện tại sẽ được đẩy vào hàng đợi và các nút ở cấp độ hiện tại được đánh dấu là đã truy cập và xuất ra khỏi hàng đợi.

Hãy cho chúng tôi hiểu hoạt động của thuật toán với sự trợ giúp của ví dụ sau.



Bước 1: Ban đầu hàng đợi và mảng đã truy cập trống.

Bước 2: Đẩy nút 0 vào hàng đợi và đánh dấu nó đã truy cập.

Bước 3: Xóa nút 0 khỏi đầu hàng đợi và truy cập các hàng xóm chưa được thăm và đẩy chúng vào hàng đợi.

Bước 4: Xóa nút 1 khỏi đầu hàng đợi và truy cập các hàng xóm chưa được thăm và đẩy chúng vào hàng đợi.

Bước 5: Loại bỏ nút 2 khỏi phía trước hàng đợi và truy cập các hàng xóm chưa được thăm và đẩy chúng vào hàng đợi.

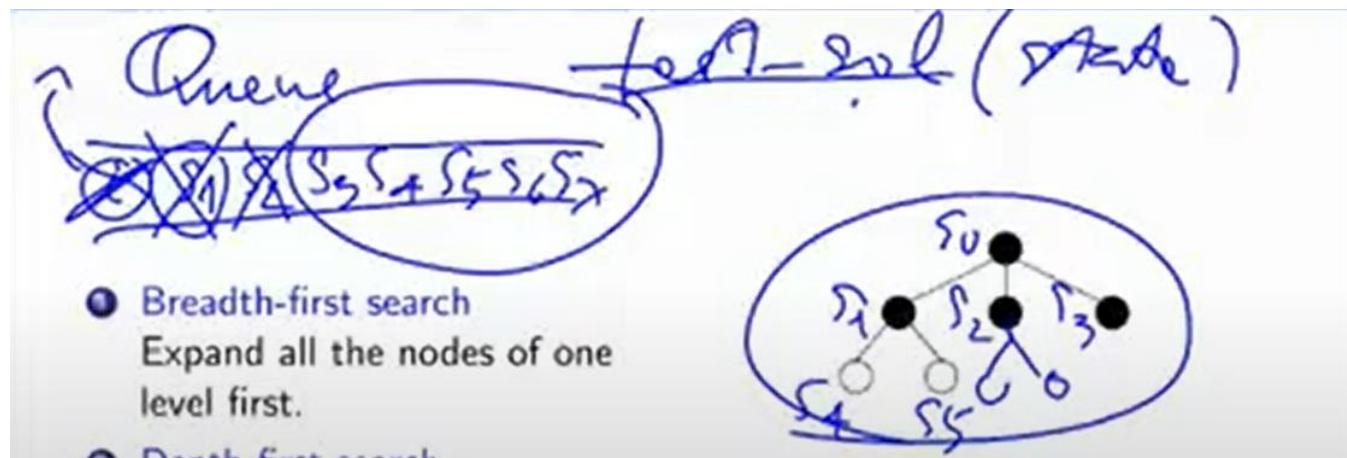
Bước 6: Xóa nút 3 khỏi phía trước hàng đợi và truy cập các hàng xóm chưa được thăm và đẩy chúng vào hàng đợi.

Như chúng ta có thể thấy rằng mọi hàng xóm của nút 3 đều được truy cập, vì vậy hãy chuyển sang nút tiếp theo ở phía trước hàng đợi.

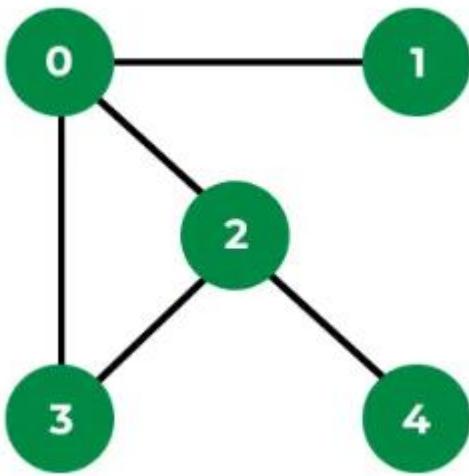
Bước 7: Xóa nút 4 khỏi phía trước hàng đợi và truy cập các hàng xóm chưa được thăm và đẩy chúng vào hàng đợi.

Như chúng ta có thể thấy rằng mọi hàng xóm của nút 4 đều được truy cập, vì vậy hãy chuyển sang nút tiếp theo ở phía trước hàng đợi.

Bây giờ Hàng đợi trở nên trống. Vì vậy, hãy chấm dứt quá trình lặp này.



2.2 Depth-first Search (Tìm kiếm theo chiều sâu)



Bước 1: Ban đầu, mảng ngăn xếp và mảng đã truy cập trống.

Bước 2: Truy cập 0 và đặt các nút lân cận chưa được truy cập vào ngăn xếp.

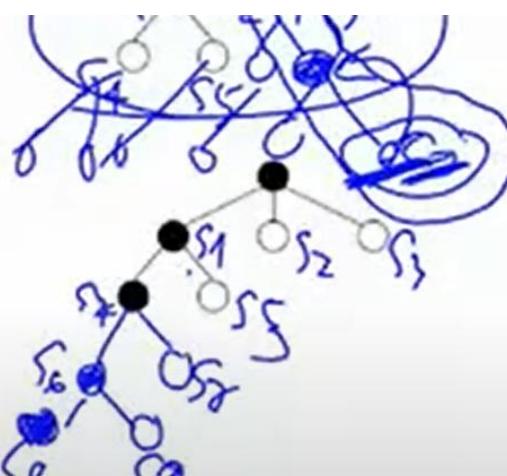
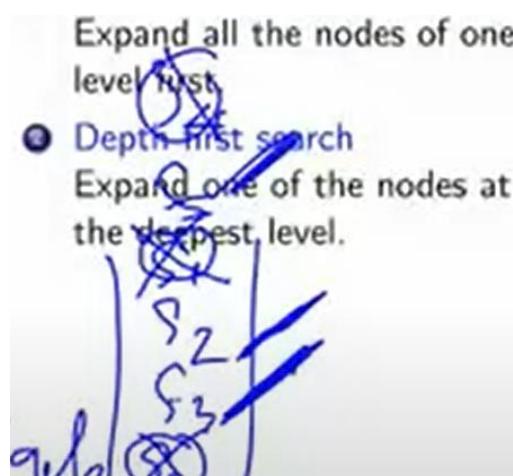
Bước 3: Nay, Nút 1 ở đầu ngăn xếp, vì vậy hãy truy cập nút 1 và lấy nó ra khỏi ngăn xếp và đặt tất cả các nút lân cận chưa được truy cập vào ngăn xếp.

Bước 4: Nay, Nút 2 ở đầu ngăn xếp, vì vậy hãy truy cập nút 2 và lấy nó ra khỏi ngăn xếp và đặt tất cả các nút lân cận chưa được truy cập (tức là 3, 4) vào ngăn xếp.

Bước 5: Nay, Nút 4 ở đầu ngăn xếp, vì vậy hãy truy cập nút 4 và lấy nó ra khỏi ngăn xếp và đặt tất cả các nút lân cận chưa được truy cập vào ngăn xếp.

Bước 6: Nay, Nút 3 ở đầu ngăn xếp, vì vậy hãy truy cập nút 3 và lấy nó ra khỏi ngăn xếp và đặt tất cả các nút lân cận chưa được truy cập vào ngăn xếp.

Nay, Ngăn xếp trống, điều đó có nghĩa là chúng tôi đã truy cập tất cả các nút và quá trình truyền tải DFS của chúng tôi kết thúc.

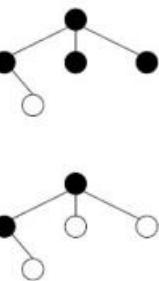


Search Strategies: Blind search

Criterion	Breadth-First	Depth-First
Time	b^d	b^m
Space	b^d	bm
Optimal?	Yes	No
Complete?	Yes	No

b: branching factor

d: solution depth



m: maximum depth

d: tầng sâu chưa đáp án

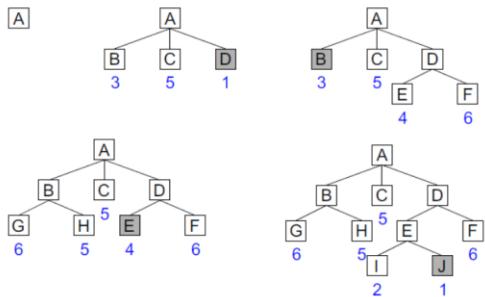
m: độ sâu tối đa để search

b: hệ số nhánh (ở một state có trung bình bao nhiêu state tiếp theo)

tồn ít biến đổi nhất

3. Best First Search.

Best-First Search



Best-First Search

- **OPEN:** nodes that have been generated, but have not examined. This is organized as a **priority queue**.
- **CLOSED:** nodes that have already been examined. Whenever a new node is generated, **check** whether it has been **generated before**. Hash table

Best-First Search

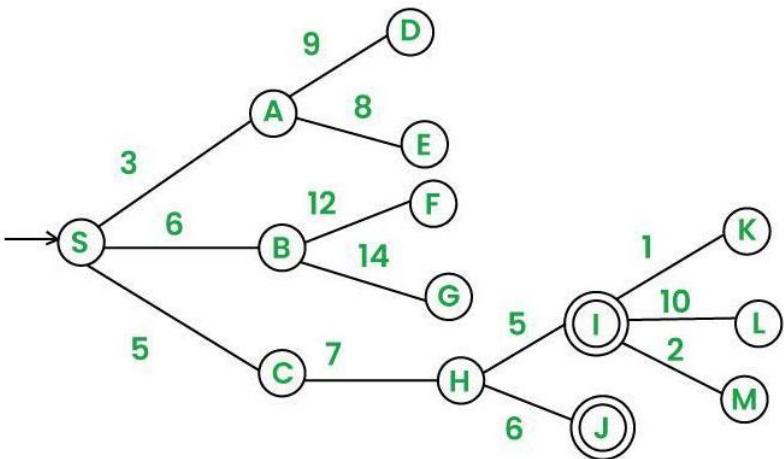
cũng ko đúng.

Algorithm

- ① $\text{OPEN} = \{\text{initial state}\}$.
- ② Loop until a goal is found or there are no nodes left in OPEN:
 - Pick the best node in OPEN
 - Generate its successors
 - For each successor:
 - new \rightarrow evaluate it, add it to OPEN, record its parent generated before
 - \rightarrow change parent, update successors

Chúng tôi sử dụng hàng đợi ưu tiên hoặc đống lưu trữ chi phí của các nút có giá trị hàm đánh giá thấp nhất. Vì vậy cách triển khai là một biến thể của BFS, chúng ta chỉ cần thay đổi Queue thành PriorityQueue.

Ví dụ:



Chúng tôi bắt đầu từ nguồn “S” và tìm kiếm mục tiêu “I” bằng cách sử dụng chi phí nhất định và tìm kiếm Tốt nhất Đầu tiên.

pq ban đầu chứa S

Chúng tôi loại bỏ S khỏi pq và xử lý các hàng xóm chưa được thăm của S thành pq.
 pq bây giờ chứa {A, C, B} (C được đặt trước B vì C có chi phí thấp hơn)

Chúng tôi loại bỏ A khỏi pq và xử lý các hàng xóm chưa được thăm của A thành pq.
 pq bây giờ chứa {C, B, E, D}

Chúng tôi loại bỏ C khỏi pq và xử lý các hàng xóm chưa được thăm của C thành pq.
 pq bây giờ chứa {B, H, E, D}

Chúng tôi loại bỏ B khỏi pq và xử lý các hàng xóm chưa được thăm của B thành pq.
 pq bây giờ chứa {H, E, D, F, G}

Chúng tôi loại bỏ H khỏi pq.

Vì mục tiêu “I” của chúng ta là hàng xóm của H nên chúng ta quay trở lại.

Best-First Search	Best-First Search
<ul style="list-style-type: none"> Greedy search: $h(n)$ = cost of the cheapest path from node n to a goal state. Neither optimal nor complete Uniform-cost search: $g(n)$ = cost of the cheapest path from the initial state to node n. 	<ul style="list-style-type: none"> Greedy search: $h(n)$ = cost of the cheapest path from node n to a goal state. Neither optimal nor complete Uniform-cost search: $g(n)$ = cost of the cheapest path from the initial state to node n.

Best-First Search	Best-First Search
<ul style="list-style-type: none"> Greedy search: $h(n)$ = cost of the cheapest path from node n to a goal state. Neither optimal nor complete Uniform-cost search: $g(n)$ = cost of the cheapest path from the initial state to node n. Optimal and complete, but very inefficient 	<p>Algorithm: A* (Hart et al., 1968)</p> $f(n) = g(n) + h(n)$ <p>$h(n)$ = cost of the cheapest path from node n to a goal state. $g(n)$ = cost of the cheapest path from the initial state to node n.</p>

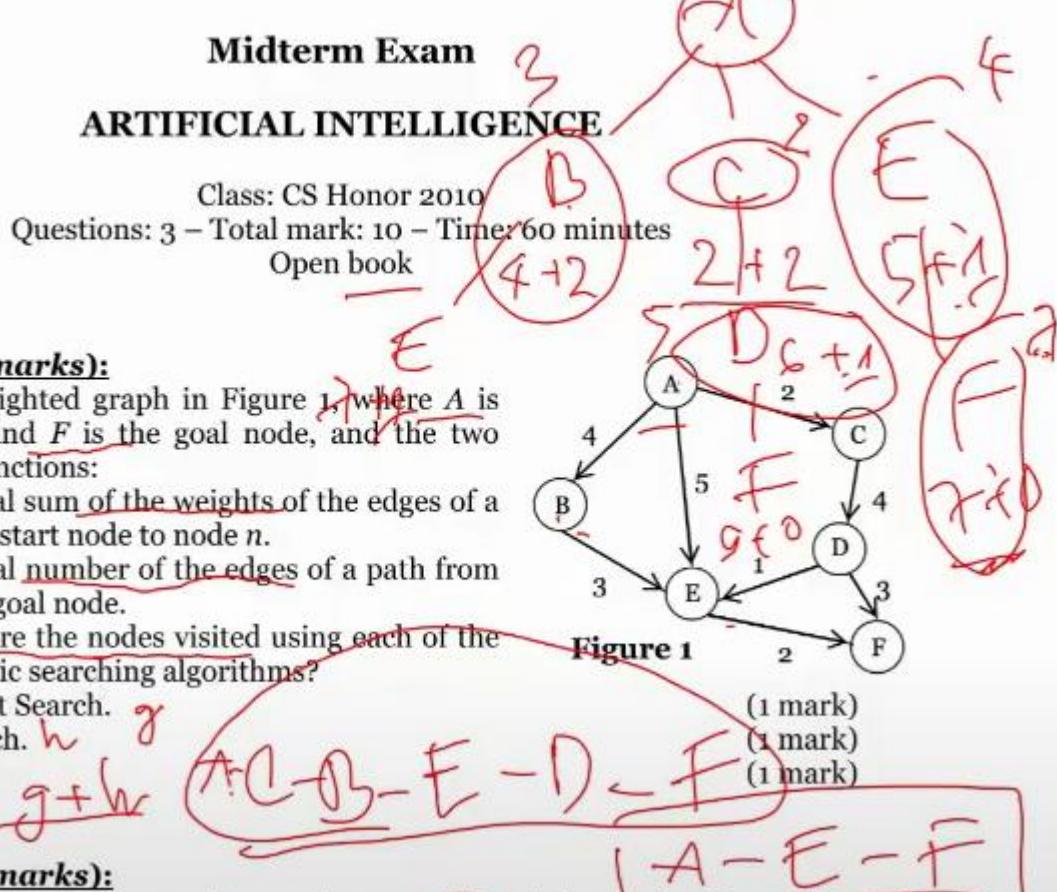
Best-First Search

Algorithm: A* (Hart et al., 1968)

$$f^*(n) = g^*(n) + h^*(n)$$

$h^*(n)$ (heuristic factor) = estimate of $h(n)$.

$g^*(n)$ (depth factor) = approximation of $g(n)$ found by A* so far.



4. Heuristic
a. Leo đồi: F \rightarrow đơn giản/ dốc nhất/ khắc phục.

Simple Hill Climbing

Algorithm

- ① Evaluate the initial state
- ② Loop until a solution is found or there are no new operators left to be applied:
 - Select and apply a operator
 - Evaluate the new state:
 - goal → quit
 - better than current state → new current state

not try all possible new states!

Hill Climbing: Disadvantages

- Hill climbing is a **local method**: Decides what to do next by looking only at the “immediate” consequences of its choices.
- **Global information** might be encoded in heuristic functions.

Hill Climbing: Block World

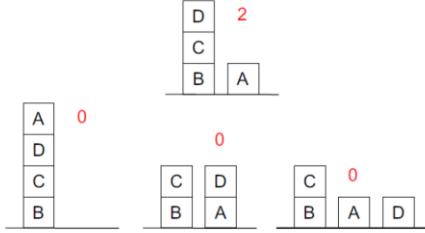
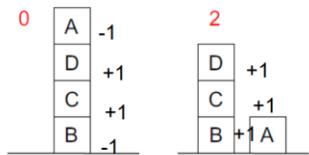


Local heuristic:

- +1 for each block that is resting on the thing it is supposed to be resting on.
- -1 for each block that is resting on a wrong thing.

Hill Climbing: Block World

Hill Climbing: Block World

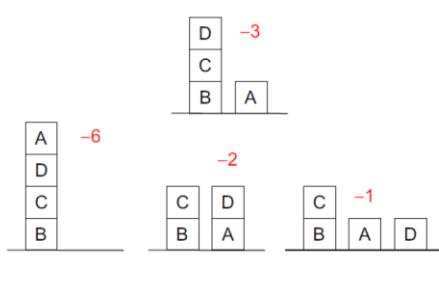
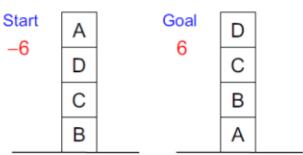


Cục bộ: Thêm một điểm cho mỗi khối nằm trên vật mà nó được cho là nằm trên đó. Trừ một điểm cho mỗi khối đặt sai vị trí.

Khi sử dụng chức năng này, trạng thái mục tiêu có điểm là 4. Trạng thái ban đầu có điểm là 0 (vì nó được cộng một điểm cho các khối A, D, C và bị trừ một điểm cho các khối A). Chỉ có một bước di chuyển từ trạng thái ban đầu, đó là di chuyển khối A vào bàn. Điều đó tạo ra trạng thái có số điểm là 2. Quá trình leo đồi sẽ chấp nhận nước đi đó. Từ trạng thái mới, có ba bước đi có thể xảy ra, dẫn đến ba trạng thái đó. Các bang này có số điểm: (a) 0, (b) 0 và (c) 0 Việc leo đồi sẽ tạm dừng vì tất cả các bang này đều có điểm thấp hơn trạng thái hiện tại. Quá trình đã đạt đến mức tối đa cục bộ nhưng không phải là mức tối đa toàn cầu.

Hill Climbing: Block World

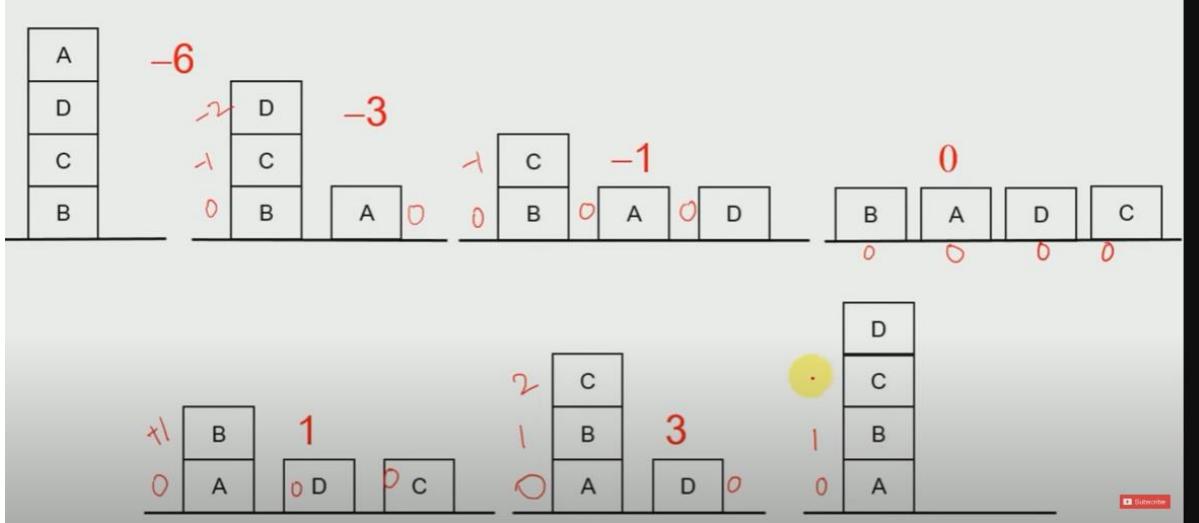
Hill Climbing: Block World



Global heuristic:

- For each block that has the correct support structure: +1 to every block in the support structure.
- For each block that has a wrong support structure: -1 to every block in the support structure.

Hill Climbing: Global Heuristic function



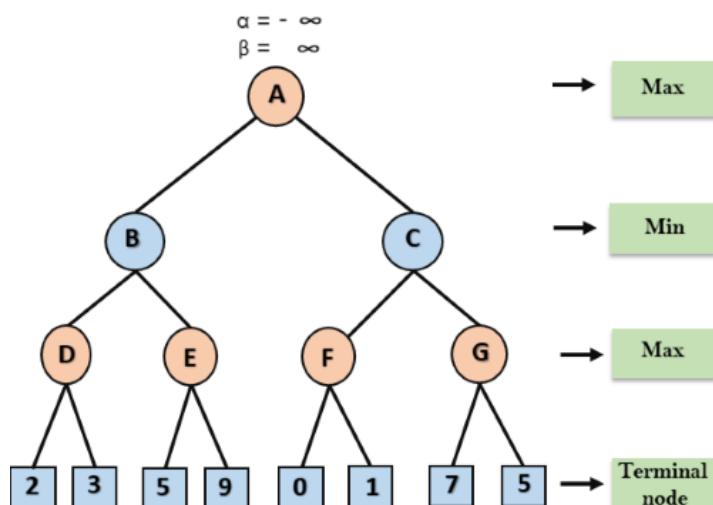
Dạng 2. Giải thuật Minmax.

Max \rightarrow Trong các node con dẫn đến node cha, cái nào lớn hơn thì chọn.

Min \rightarrow Ngược lại với max.

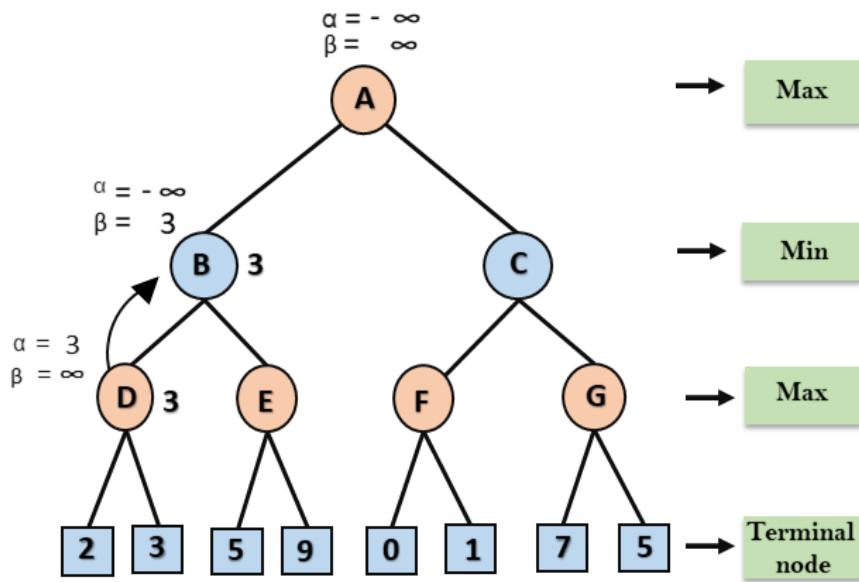
Dạng 3. Alpha-beta cutoff/pruning

Bước 1: Ở bước đầu tiên, người chơi Max sẽ bắt đầu di chuyển đầu tiên từ nút A trong đó $\alpha = -\infty$ và $\beta = +\infty$, các giá trị alpha và beta này được truyền xuống nút B trong đó lại là $\alpha = -\infty$ và $\beta = +\infty$ và Nút B chuyển cùng một giá trị cho nút con D của nó.



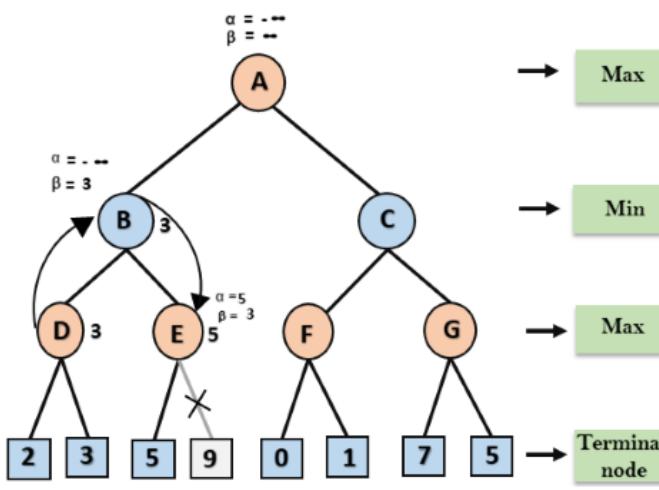
Bước 2: Tại Nút D, giá trị của α sẽ được tính lần lượt cho Max. Giá trị của α trước tiên được so sánh với 2 rồi đến 3, và giá trị tối đa ($2, 3$) = 3 sẽ là giá trị của α tại nút D và giá trị nút cũng sẽ là 3.

Bước 3: Bây giờ thuật toán quay lại nút B, trong đó giá trị của β sẽ thay đổi vì đây là lượt của Min, Bây giờ $\beta = +\infty$, sẽ so sánh với giá trị các nút tiếp theo có sẵn, từ c là min ($\infty, 3$) = 3, do đó tại nút B bây giờ $\alpha = -\infty$ và $\beta = 3$.



Ở bước tiếp theo, thuật toán duyệt qua nút kế thừa tiếp của Nút B là nút E và các giá trị của $\alpha = -\infty$ và $\beta = 3$ cũng sẽ được chuyển.

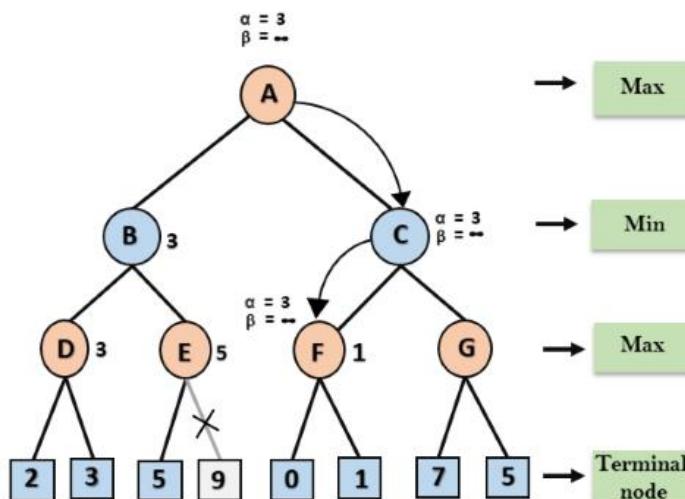
Bước 4: Tại nút E, Max sẽ đến lượt mình và giá trị của alpha sẽ thay đổi. Giá trị hiện tại của alpha sẽ được so sánh với 5 nên $\max(-\infty, 5) = 5$, do đó tại nút E $\alpha = 5$ và $\beta = 3$, trong đó $\alpha \geq \beta$ nên thừa kế bên phải của E sẽ bị lược bỏ, và thuật toán sẽ không duyệt qua nó và giá trị tại nút E sẽ là 5.



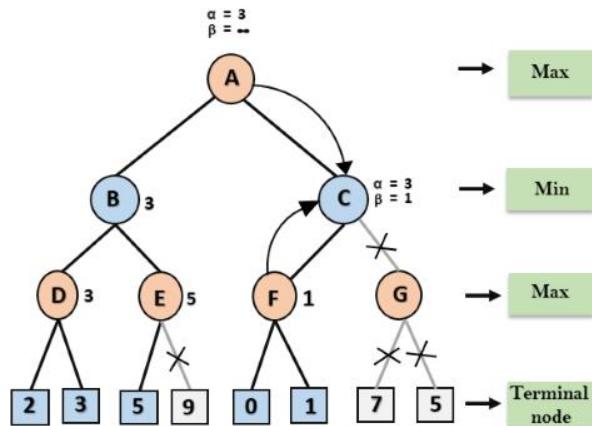
Bước 5: Ở bước tiếp theo, thuật toán lại dò ngược lại cây, từ nút B đến nút A. Tại nút A, giá trị của alpha sẽ được thay đổi, giá trị khả dụng tối đa là 3 khi $\max(-\infty, 3) = 3$ và $\beta = +\infty$, hai giá trị này hiện được chuyển sang nút kế tiếp bên phải của A là Nút C.

Tại nút C, $\alpha = 3$ và $\beta = +\infty$ và các giá trị tương tự sẽ được chuyển đến nút F.

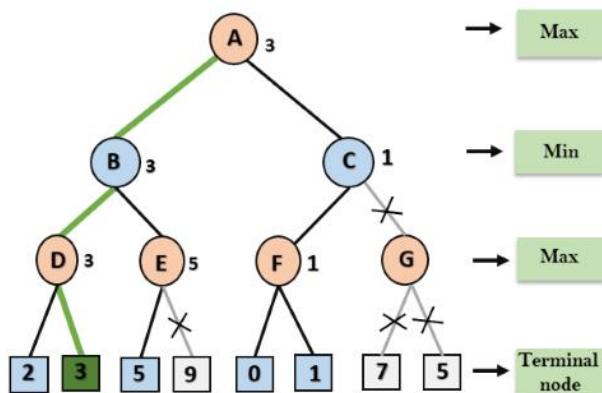
Bước 6: Tại nút F, một lần nữa giá trị của α sẽ được so sánh với con trái là 0 và $\max(3, 0) = 3$, sau đó so sánh với con phải là 1 và $\max(3, 1) = 3$ vẫn giữ nguyên là 3, nhưng giá trị nút của F sẽ trở thành 1.



Bước 7: Nút F trả về giá trị nút 1 cho nút C, tại C $\alpha = 3$ và $\beta = +\infty$, tại đây giá trị beta sẽ được thay đổi so sánh với 1 nên $\min(\infty, 1) = 1$. Bây giờ tại C, $\alpha = 3$ và $\beta = 1$, đồng thời thỏa mãn điều kiện $\alpha \geq \beta$ nên cây con tiếp theo của C là G sẽ bị lược bỏ và thuật toán sẽ không tính toán toàn bộ cây con G.

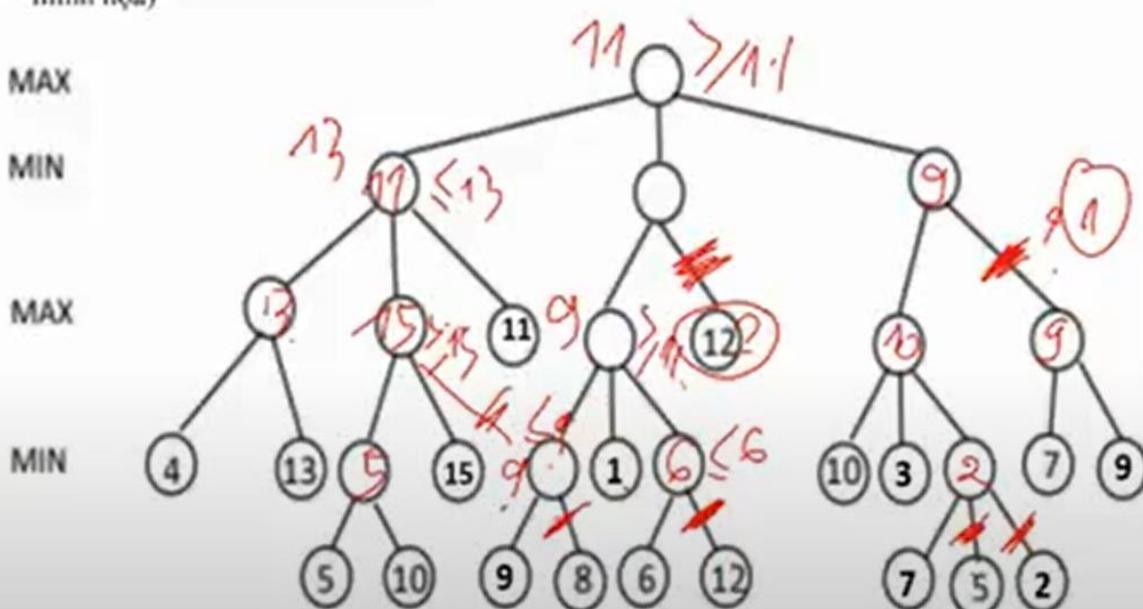


Bước 8: C bây giờ trả về giá trị từ 1 đến A ở đây giá trị tốt nhất cho A là $\max(3, 1) = 3$. Sau đây là cây trò chơi cuối cùng hiển thị các nút được tính toán và các nút chưa bao giờ được tính toán. Do đó, giá trị tối ưu cho bộ tối đa hóa là 3 cho ví dụ này.

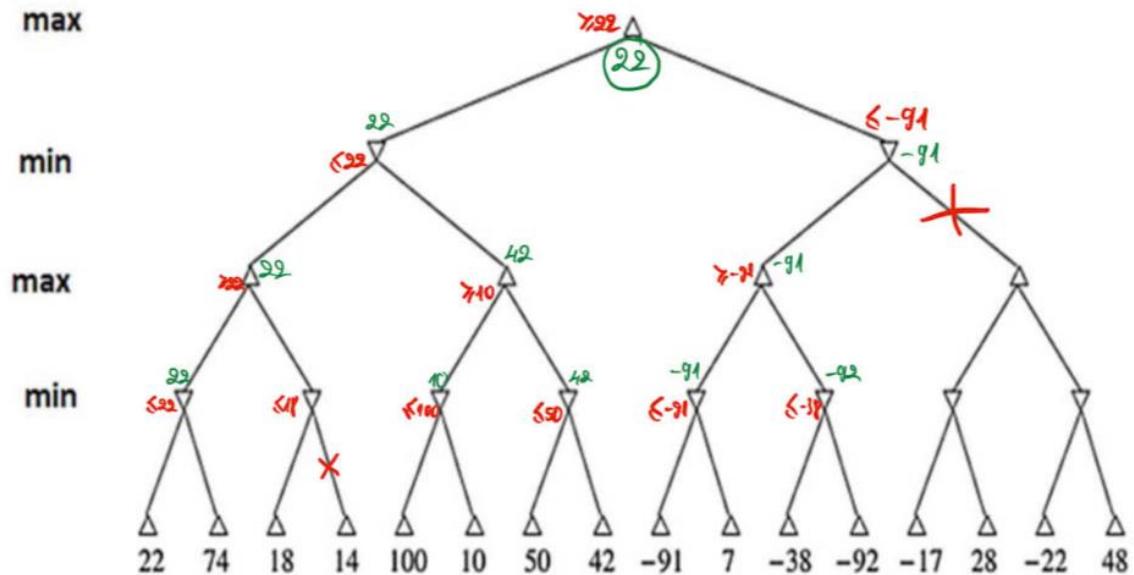


Điều này chứng minh giá càng lớn càng tốt.

- Điền vào những ô còn trống khi áp dụng giải thuật Minimax (vẽ hình cây kết quả tính toán cuối cùng)
- Khi áp dụng alpha-beta cutoff, các nhánh nào sẽ bị cắt nếu duyệt cây từ trái sang phải (vẽ hình minh họa)



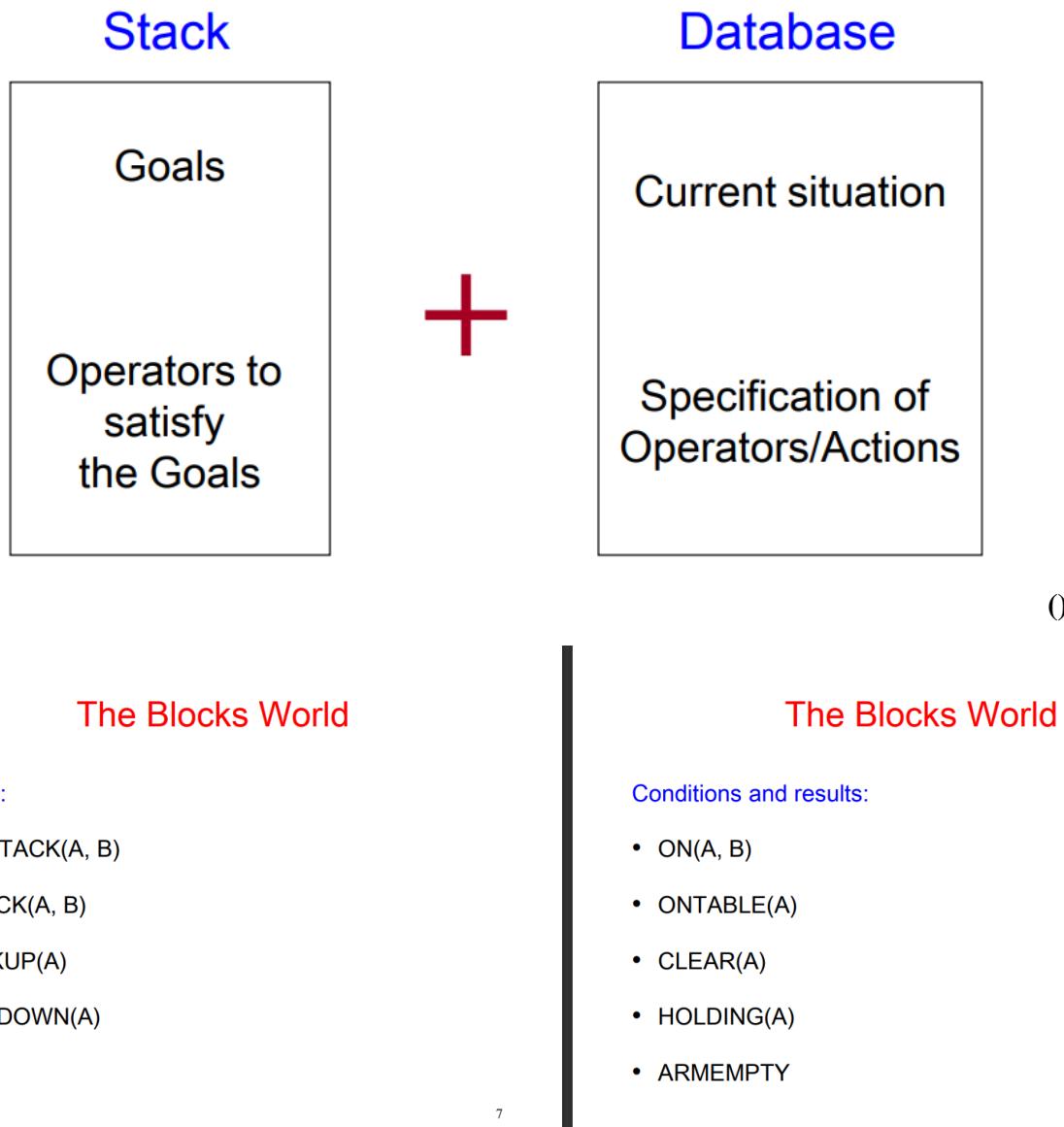
Hình 2. Cây trạng thái của trò chơi 2-người



Hình 1. Cây trạng thái trò chơi 2-người.

5. Goal Stack Planning

Goal Stack Planning



Lập kế hoạch ngăn xếp mục tiêu

Đây mục tiêu ban đầu vào ngăn xếp. Lặp lại cho đến khi ngăn xếp trống:

- Nếu định ngăn xếp là mục tiêu phức hợp, hãy đẩy các mục tiêu phụ chưa thỏa mãn của nó vào ngăn xếp.
 - Nếu định ngăn xếp là một mục tiêu duy nhất không được thỏa mãn, hãy thay thế nó bằng một toán tử khiến mục tiêu đó hài lòng và đẩy điều kiện tiên quyết của người vận hành đó vào ngăn xếp.
 - Nếu định ngăn xếp là toán tử, hãy lấy nó ra khỏi ngăn xếp, thực thi và thay đổi cơ sở dữ liệu theo ảnh hưởng của thao tác.
 - Nếu mục tiêu ở trên cùng của ngăn xếp là thỏa mãn, hãy lấy nó ra khỏi ngăn xếp
- STACK(X,Y): Chồng X lên Y. (Trên Y không được có)

UNSTACK(X,Y): Bỏ X ra khỏi Y. (X phải nằm trên Y)

PICKUP (X): Lấy X lên

PUTD(X): Bỏ X xuống.

The Blocks World

Specification of actions:

PICKUP(x):

P: CLEAR(x) \wedge ONTABLE(x) \wedge ARMEMPTY

D: ONTABLE(x) \wedge ARMEMPTY

A: HOLDING(x)

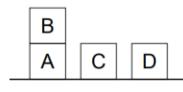
PUTDOWN(x):

P: HOLDING(x)

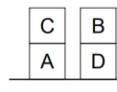
D: HOLDING(x)

A: ONTABLE(x) \wedge ARMEMPTY

The Blocks World



start: ON(B, A) \wedge
ONTABLE(A) \wedge
ONTABLE(C) \wedge
ONTABLE(D) \wedge
ARMEMPTY



goal: ON(C, A) \wedge
ON(B, D) \wedge
ONTABLE(A) \wedge
ONTABLE(D) \wedge

11

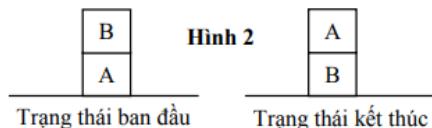
12 CL ©; CL (D)

P: tiền điều kiện

D: delete

A: hậu

Câu 2 (1 đ): Sử dụng phương pháp chòn mục tiêu (*goal stack*) và các tác vụ STACK, UNSTACK, PICKUP, PUTDOWN để lập kế hoạch đưa các khối từ trạng thái ban đầu về trạng thái kết thúc như ở Hình 2.



Bước 1. Goals. Cái nào càng dễ thoả mãn càng đưa lên cao.

Đưa các tiền điều kiện lên trên hành động chuẩn bị thực thi.

STACK	DB
CL(A) (clear A: A không có ai đè lên hoặc đang được cầm trên tay)	ARM(E) : cánh tay không cầm nắm
ONT(B) (B on table)	CL(B)
ON(A,B) (A nằm trên B)	ON(B,A)
goal	ONT(A)

Bước 2. Ta gỡ B ra khỏi A. UNSTACK(B,A) . Ta có bảng (nháp) sau.

STACK	DB
PRESTACK(B,A)	ARM(E)
UNSTACK(B,A)	CL(B)

CL(A)	ON(B,A)
ONT(B) -> Bỏ , thay bằng hành động PUTD(B), HOLD(B) ở dưới đây.,	ONT(A)
ON(A,B)	
goal	

Prestack(B,A) cần ON(B,A), ARM (E), CL(B). – Nháp.

Sau khi thực hiện hành động UNSTACK(B,A), ta có:

STACK	DB
PRESTACK(B,A)	ARM(E)
UNSTACK(B,A)	CL(B)
CL(A)	ON(B,A)
ONT(B) -> Bỏ , thay bằng hành động PUTD(B), HOLD(B) ở dưới đây.,	ONT(A)
ON(A,B)	
goal	

Bước 3. Sau khi PUTD(B), ta có.

STACK	DB
PRESTACK(A,B)	CL(B)
STACK(A,B)	ONT(A)
ON(A,B) -> Nháp, gạch đi, thay thế bằng 2 hành động trên cùng nó.	CL(A)
goal	ARM(E)
	ONT(B)

Prestack(A,B) cần HOLD (A), CL(B). – Nháp.

Xét trong DB, thấy HOLD(A) chưa có -> Bỏ tiếp vào bảng trên

STACK	DB
PICKUP(A)	CL(B)
HOLD(A) -> BỎ , ghi PICKUP(A) ở trên	ONT(A)
PRESTACK(A,B)	CL(A)
STACK(A,B)	ARM(E)

ON(A,B) -> Nhập, gạch đi, thay thế bằng 2 hành động trên cùng nó. goal	ONT(B)
---	--------

Để pickup(A) thì cần CL(A), ONT(A), ARM(E) . Xét trong DB, thấy thoả -> Pop Pickup(A) ra khỏi stack.

Bước 4. Thực hiện PICKUP(A)

STACK	DB
PRESTACK(A,B)	CL(B)
STACK(A,B)	CL(A)
goal	ONT(B) HOLD(A)

Nhận thấy PRESTACK(A,B) thoả mãn. Ta thực hiện STACK(A,B)

Bước 5. Thực hiện STACK(A,B). Ta nhận thấy giống với bảng mục tiêu -> Hoàn thành kế hoạch.

STACK	DB
goal	CL(A) ONT(B) ON(A,B) ARM (E)

II. Knowledge Base.

1. Logic vị từ (Slide - Ko kip)

Câu 3 (2 đ): Xét các phát biểu sau đây:

- Mỗi người La Mã hoặc là ghét Caesar hoặc là trung thành với ông ấy.
 - Chỉ có người La Mã nào không trung thành với Caesar mới muốn ám sát ông ấy.
 - Marcus là người La Mã và muốn ám sát Caesar.
 - Anthony là người La Mã và trung thành với Casear.
- (a) Biểu diễn các phát biểu trên bằng logic vị từ. (1 đ)
- b)

Có thể xem miền đang xét là tập hợp các người La Mã, khi đó các câu có thể viết đơn giản dạng :

- (1) $\text{ghetCaesar}(x) \vee \text{trungthanhCaesar}(x)$
 (2) $\neg \text{trungthanhCaesar}(x) \vee \neg \text{amsatCaesar}(x)$
 (3) $\text{amsatCaesar}(\text{Marcus})$
 (4) $\text{trungthanhCaesar}(\text{Anthony})$

(b) Sử dụng phương pháp phản chứng-phân giải để:

- Chứng minh Anthony không muốn ám sát Caesar. (0.5 đ)
- Tìm xem ai ghét Caesar.

b1. $\alpha = \neg \text{amsatCaesar}(\text{Anthony}) \neg \alpha = \text{amsatCaesar}(\text{Anthony})$ (5)

Kết hợp (2) và (5) : $\neg \text{trungthanhCaesar}(\text{Anthony})$ (6) với {Anthony/x}

Kết hợp (6) và (4) : đpcm

b2. $\alpha = \text{ghetCaesar}(y) \neg \alpha = \neg \text{ghetCaesar}(y)$

Dùng phản chứng chúng ta chỉ tìm được {Marcus/y}

Câu 4 (2 điểm)

Xét trong một thế giới đóng như sau:

1. An là một người đàn ông
2. Người đàn công là người giàu có chỉ khi nuôi cá Koi thành công
3. Thúy là một người đẹp
4. Người đẹp chỉ thích người giàu có
5. Thúy thích An

Hãy biểu diễn các câu trên bằng logic vị từ.

Và chứng minh An nuôi cá Koi thành công bằng phương pháp **phản chứng – phân giải**.

$3 + 4 \rightarrow 6$: Thúy thích người giàu có $6 + 5 \rightarrow 7$: An là người giàu có $1 + 7 \rightarrow 8$: An là người đàn ông giàu có $8 + 2$: An nuôi cá Koi thành công (đpcm) Tuy nhiên sinh viên phải biểu diễn bằng phương pháp phản chứng – phân giải để đi đến được kết luận trên

Bước 1: Biểu diễn các câu bằng logic vị từ

Định nghĩa các ký hiệu:

- $\text{man}(x)$: x là người đàn ông.
- $\text{rich}(x)$: x là người giàu có.
- $\text{successfulKoi}(x)$: x nuôi cá Koi thành công.
- $\text{beautiful}(x)$: x là người đẹp.
- $\text{likes}(x, y)$: x thích y .

Biểu diễn từng câu:

1. An là một người đàn ông.
 - $\text{man}(\text{An})$.
2. Người đàn ông là người giàu có chỉ khi nuôi cá Koi thành công.
 - **Điển giải:** Nếu x là người đàn ông và x giàu có, thì x nuôi cá Koi thành công.
 - **Logic vị từ:** $\forall x : \text{man}(x) \wedge \text{rich}(x) \rightarrow \text{successfulKoi}(x)$.
3. Thúy là một người đẹp.
 - $\text{beautiful}(\text{Thuy})$.
4. Người đẹp chỉ thích người giàu có.
 - **Điển giải:** Nếu x là người đẹp và x thích y , thì y phải giàu có.
 - **Logic vị từ:** $\forall x : \forall y : \text{beautiful}(x) \wedge \text{likes}(x, y) \rightarrow \text{rich}(y)$.
5. Thúy thích An.
 - $\text{likes}(\text{Thuy}, \text{An})$.

Tập tri thức KB:

- $\text{man}(\text{An})$
- $\forall x : \text{man}(x) \wedge \text{rich}(x) \rightarrow \text{successfulKoi}(x)$
- $\text{beautiful}(\text{Thuy})$
- $\forall x : \forall y : \text{beautiful}(x) \wedge \text{likes}(x, y) \rightarrow \text{rich}(y)$
- $\text{likes}(\text{Thuy}, \text{An})$

Mục tiêu cần chứng minh: $\text{successfulKoi}(\text{An})$.

Bước 2: Chuyển đổi thành dạng mệnh đề (Clause Form)

Để áp dụng phản chứng phân giải, ta cần chuyển các mệnh đề trong KB thành dạng mệnh đề (CNF - Conjunctive Normal Form).

1. $\text{man}(\text{An}) \rightarrow$ Giữ nguyên: $\text{man}(\text{An})$.
2. $\forall x : \text{man}(x) \wedge \text{rich}(x) \rightarrow \text{successfulKoi}(x)$
 - Chuyển \rightarrow : $\forall x : \neg(\text{man}(x) \wedge \text{rich}(x)) \vee \text{successfulKoi}(x)$
 - Áp dụng luật De Morgan: $\forall x : \neg\text{man}(x) \vee \neg\text{rich}(x) \vee \text{successfulKoi}(x)$
 - Bỏ \forall : $\neg\text{man}(x) \vee \neg\text{rich}(x) \vee \text{successfulKoi}(x)$.
3. $\text{beautiful}(\text{Thuy}) \rightarrow$ Giữ nguyên: $\text{beautiful}(\text{Thuy})$.
4. $\forall x : \forall y : \text{beautiful}(x) \wedge \text{likes}(x, y) \rightarrow \text{rich}(y)$
 - Chuyển \rightarrow : $\forall x : \forall y : \neg(\text{beautiful}(x) \wedge \text{likes}(x, y)) \vee \text{rich}(y)$
 - Áp dụng luật De Morgan: $\forall x : \forall y : \neg\text{beautiful}(x) \vee \neg\text{likes}(x, y) \vee \text{rich}(y)$
 - Bỏ \forall : $\neg\text{beautiful}(x) \vee \neg\text{likes}(x, y) \vee \text{rich}(y)$.
5. $\text{likes}(\text{Thuy}, \text{An}) \rightarrow$ Giữ nguyên: $\text{likes}(\text{Thuy}, \text{An})$.

Phù định mục tiêu:

- Mục tiêu: $\text{successfulKoi}(\text{An})$.
- Phù định: $\neg\text{successfulKoi}(\text{An})$.

Tập S ban đầu:

1. $\text{man}(\text{An})$
2. $\neg\text{man}(x) \vee \neg\text{rich}(x) \vee \text{successfulKoi}(x)$
3. $\text{beautiful}(\text{Thuy})$
4. $\neg\text{beautiful}(x) \vee \neg\text{likes}(x, y) \vee \text{rich}(y)$
5. $\text{likes}(\text{Thuy}, \text{An})$
6. $\neg\text{successfulKoi}(\text{An})$

Bước 3: Áp dụng phản chứng phân giải

Phân giải từng bước:

1. Từ (3), (5), và (4):
 - $\text{beautiful}(\text{Thuy}), \text{likes}(\text{Thuy}, \text{An}), \text{và } \neg\text{beautiful}(x) \vee \neg\text{likes}(x, y) \vee \text{rich}(y)$.
 - Unifier: $\{x/\text{Thuy}, y/\text{An}\}$.
 - Phân giải: $\text{rich}(\text{An})$ (mệnh đề 7).
2. Từ (1), (7), và (2):
 - $\text{man}(\text{An}), \text{rich}(\text{An}), \text{và } \neg\text{man}(x) \vee \neg\text{rich}(x) \vee \text{successfulKoi}(x)$.
 - Unifier: $\{x/\text{An}\}$.
 - Phân giải: $\text{successfulKoi}(\text{An})$ (mệnh đề 8).
3. Từ (8) và (6):
 - $\text{successfulKoi}(\text{An}) \text{ và } \neg\text{successfulKoi}(\text{An})$.
 - Phân giải: \square (mệnh đề rỗng).

Bước 4: Kết luận

- Quá trình phân giải đã dẫn đến mâu thuẫn (\square), nghĩa là $\text{KB} \wedge \neg\text{successfulKoi}(\text{An})$ không thỏa mãn.
- Do đó, $\text{KB} \models \text{successfulKoi}(\text{An})$.

Kết luận: An nuôi cá Koi thành công.

2. Chứng minh phản chứng – phân giải. (Slide - ko Kịp_

Lecture 16

Resolution for Predicate Logic

Instructor: Yu Zhen Xie

Rules of inference

- These patterns describe how new knowledge can be derived from existing knowledge, both in the form of propositional logic formulas (sentences).
- When describing an inference rule, the *premise* specifies the pattern that must match our knowledge base and the *conclusion* is the new knowledge inferred.

1

Revisit: main rules of inference in propositional logic

- Valid argument:
 $\text{AND of premises} \rightarrow \text{conclusion}$ is a tautology
- Modus ponens:
 $(p \rightarrow q) \wedge p \rightarrow q$ is a tautology
- Hypothetical syllogism:
 $(p \rightarrow q) \wedge (q \rightarrow r) \rightarrow (p \rightarrow r)$ is a tautology
- Disjunctive syllogism:
 $(A \vee B) \wedge \neg A \rightarrow B$ is a tautology
- Resolution:
 $(A \vee C) \wedge (B \vee \neg C) \rightarrow (A \vee B)$ is a tautology

2

Modus ponens, modus tollens, AND elimination, AND introduction, and universal instantiation

- If the sentences P and $P \rightarrow Q$ are known to be true, then **modus ponens** lets us infer Q .
- Under the inference rule **modus tollens**, if $P \rightarrow Q$ is known to be true and Q is known to be false, we can infer P .
- **AND elimination** allows us to infer the truth of either of the conjuncts from the truth of a conjunctive sentence. E.g. $P \wedge Q$ lets conclude both P and Q are true.
- **AND introduction** lets us infer the truth of a conjunction from the truth of its conjuncts. E.g. if both P and Q are true, then $P \wedge Q$ are true.
- **Universal instantiation** states that if any universally quantified variable in a true sentence is replaced by any appropriate term from the domain, the result is a true sentence. Thus, if a is from the domain of X , $\forall X p(X)$ lets us infer $p(a)$.

3

4

Definition

- A predicate logic (or calculus) expression X **logically follows** from a set S of predicate calculus expressions if every interpretation and variable assignment that satisfies S also satisfies X .
 - An *interpretation* is an assignment of specific values to domains and predicates.
- An inference rule is **sound** if every predicate calculus expressions also logically follows from S .
- An inference rule is **complete** if, given a set S of predicate calculus expressions, the rule can infer every expression that logically follows from S .

5

Logic and finding a proof

- Given
 - a knowledge base represented as a set of propositional sentences.
 - a goal stated as a propositional sentence
 - list of inference rules
- We can write a program to repeatedly apply inference rules to the knowledge base in the hope of deriving the goal.

6

Developing a proof procedure

- Deriving (or refuting) a goal from a collection of logic facts corresponds to a very large search tree.
- A large number of *rules of inference* could be utilized.
- The selection of which rules to apply and when would itself be non-trivial.

7

Resolution and CNF

- **Resolution** is a single rule of inference that can operate efficiently on a special form of sentences.
- The special form is called *conjunctive normal form* (CNF) or *clausal form*, and has these properties:
 - Every sentence is a disjunction (OR) of literals (clauses)
 - All sentences are implicitly conjoined (ANDed).

8

Predicate Logic Resolution

- We have to worry about the arguments to predicates, so it is harder to know when two literals match and can be used by resolution.
 - For example, does the literal `Father(Bill, Chelsea)` match `Father(x, y)`?
- The answer depends on how we substitute values for variables.

Proof procedure for predicate logic

- Same idea, but a few added complexities:
 - conversion to CNF is much more complex.
 - Matching of literals requires providing a matching of variables, constants and/or functions.
 - `Skates(x) ∨ LikesHockey(x)`
 - `LikesHockey(y)`

We can resolve these only if we assume x and y refer to the same object.

Predicate Logic and CNF

- Converting to CNF is harder - we need to worry about variables and quantifiers.
 - Eliminate all implications →
 - Reduce the scope of all \neg to single term
 - Make all variable names unique
 - Move quantifiers left (prenex normal form)
 - Eliminate Existential Quantifiers
 - Eliminate Universal Quantifiers
 - Convert to conjunction of disjuncts
 - Create separate clause for each conjunct.

Eliminate Existential Quantifiers

- Any variable that is **existentially quantified** means that
 - there is some value for that variable that makes the expression true.
- To eliminate the quantifier, we can **replace the variable with a function**.
- We don't know what the function is, we just know it exists.

Skolem functions

- Named after the Norwegian logician Thoralf Skolem
- **Example:** $\exists y \text{ President}(y)$
We replace `y` with a new function `func`:
`President(func())`
`func` is called a **Skolem function**.
- In general the function must have the same number of arguments as the number of **universal** quantifiers in the current scope.

Skolemization Example

- In general the function must have the *same number of arguments* as the number of **universal** quantifiers in the current scope.
- **Example:** $\forall x \exists y \text{ Father}(y, x)$
 - create a new function named `foo` and replace `y` with the function.
 - $\forall x \text{ Father}(\text{foo}(x), x)$

Unification

- Two formulas are said to **unify** if there are legal instantiations (assignments of terms to variables) that make the formulas in question *identical*.
- The act of unifying is called **unification**. The instantiation that unifies the formulas in question is called a **unifier**.
- There is a simple algorithm called the **unification algorithm** that does this.

Unification

- **Example:** Unify the formulas `Q(a, y, z)` and `Q(y, b, c)`
- **Solution:**
 - Since `y` in `Q(a, y, z)` is a different variable than `y` in `Q(y, b, c)`, rename `y` in the second formula to become `y1`.
 - This means that one must unify `Q(a, y, z)` with `Q(y1, b, c)`.
 - An instance of `Q(a, y, z)` is `Q(a, b, c)` and an instance of `Q(y1, b, c)` is `Q(a, b, c)`.
 - Since these two instances are identical, `Q(a, y, z)` and `Q(y, b, c)` unify.
 - The unifier is `y1 = a, y = b, z = c`.

Algorithm to convert to clausal form (1)

(1) Eliminate conditionals \rightarrow , using the equivalence

$$p \rightarrow q \equiv \neg p \vee q$$

e.g. $(\exists x)(p(x) \wedge (\forall y)(f(y) \rightarrow h(x, y)))$ becomes
 $(\exists x)(p(x) \wedge (\forall y)(\neg f(y) \vee h(x, y)))$

(2) Eliminate negations or reduce the scope of negation to one atom.

$$\neg(\neg p) \equiv p$$

$$\neg(p \wedge q) \equiv \neg p \vee \neg q$$

$$\neg(\forall x \in S, F(x)) \equiv \exists x \in S, \neg F(x)$$

$$\neg(\exists x \in S, F(x)) \equiv \forall x \in S, \neg F(x)$$

(3) Standardize variables within a well-formed formula so that the bound or free variables of each quantifier have unique names. e.g.

$(\exists x)\neg p(x) \vee (\forall x)p(x)$ is replaced by $(\exists x)\neg p(x) \vee (\forall y)p(y)$

19

Algorithm to convert to clausal form (2)

(4) Advanced step: if there are existential quantifiers, eliminate them by using Skolem functions

$$\text{e.g. } (\exists x)p(x) \text{ is replaced by } p(a)$$

$$(\forall x)(\exists y)k(x, y) \text{ is replaced by } (\forall x)k(x, f(x))$$

(5) Convert the formula to prenex form

$$\text{e.g. } (\exists x)(p(x) \wedge (\forall y)(\neg f(y) \vee h(x, y))) \text{ becomes}$$

$$(\forall y)(p(a) \wedge (\neg f(y) \vee h(a, y)))$$

(6) Convert the formulas to CNF, which is a conjunctive of clauses. Each clause is a disjunction.

$$\text{e.g. } p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$$

(7) Drop the universal quantifiers

e.g. the formula in (5) becomes $p(a) \wedge (\neg f(y) \vee h(a, y))$

20

Algorithm to convert to clausal form (3)

(8) Eliminate the conjunctive signs by writing the formula as a set of clauses

e.g. $p(a) \wedge (\neg f(y) \vee h(a, y))$ becomes $p(a)$,
 $(\neg f(y) \vee h(a, y))$

(9) Rename variables in clauses, if necessary, so that the same variable name is only used in one clause.

e.g. $p(x) \vee q(y) \vee k(x, y)$ and $\neg p(x) \vee q(y)$ becomes
 $p(x) \vee q(y) \vee k(x, y)$ and $\neg p(x) \vee q(y)$

21

Example: Resolution for predicate logic

Anyone passing his history exams and winning the lottery is happy.

$$\forall X (\text{pass}(X, \text{history}) \wedge \text{win}(X, \text{lottery}) \rightarrow \text{happy}(X))$$

Anyone who studies or is lucky can pass all his exams.

$$\forall X \vee Y (\text{study}(X) \vee \text{lucky}(X) \rightarrow \text{pass}(X, Y))$$

John did not study but he is lucky.

$$\neg \text{study}(\text{john}) \wedge \text{lucky}(\text{john})$$

Anyone who is lucky wins the lottery.

$$\forall X (\text{lucky}(X) \rightarrow \text{win}(X, \text{lottery}))$$

These four predicate statements are now changed to clause form (Section 12.2.2):

$$1. \neg \text{pass}(X, \text{history}) \vee \neg \text{win}(X, \text{lottery}) \vee \text{happy}(X)$$

$$2. \neg \text{study}(Y) \vee \text{pass}(Y, Z)$$

$$3. \neg \text{lucky}(W) \vee \text{pass}(W, V)$$

$$4. \neg \text{study}(\text{john})$$

$$5. \text{lucky}(\text{john})$$

$$6. \neg \text{lucky}(U) \vee \text{win}(U, \text{lottery})$$

Into these clauses is entered, in clause form, the negation of the conclusion:

$$7. \neg \text{happy}(\text{john})$$

$$\neg \text{pass}(X, \text{history}) \vee \neg \text{win}(X, \text{lottery}) \vee \text{happy}(X)$$

$$\neg \text{lucky}(U) \vee \text{win}(U, \text{lottery})$$

$$\{U/X\}$$

$$\neg \text{pass}(U, \text{history}) \vee \text{happy}(U) \vee \neg \text{lucky}(U)$$

$$\neg \text{happy}(\text{john})$$

$$\{\text{john}/U\}$$

$$\text{lucky}(\text{john})$$

$$\neg \text{pass}(\text{john}, \text{history}) \vee \neg \text{lucky}(\text{john})$$

$$\{\}$$

$$\neg \text{pass}(\text{john}, \text{history})$$

$$\neg \text{lucky}(V) \vee \text{pass}(V, W)$$

$$\{\text{john}/V, \text{history}/W\}$$

$$\neg \text{lucky}(\text{john})$$

$$\text{lucky}(\text{john})$$

$$\{\}$$



3. Graph. (Slide - Ko kịp)

4. Xác suất.

Dù sung các sự kiện нóp ý cho các suy luận may mắn.

Câu 2 (3 đ): Ở TP.HCM, nếu trời mưa to thì xác suất có cây đổ trên đường là 10% và xác suất đường bị ngập là 70%; ngược lại thì hai xác suất đó lần lượt là 1% và 20%. Việc đường bị ngập hoặc vào giờ cao điểm thường gây ra kẹt xe. Thống kê cho thấy kẹt xe xảy ra trong 90% trường hợp đường bị ngập vào giờ cao điểm, trong khi xác suất kẹt xe xảy ra chỉ là 30% trong trường hợp đường không bị ngập và không vào giờ cao điểm. Nếu đường bị ngập nhưng không vào giờ cao điểm thì xác suất đó là 40%, còn nếu vào giờ cao điểm nhưng đường không bị ngập thì xác suất đó là 60%. Bây giờ là mùa mưa nên xác suất trời mưa to là 80%.

(a) Xây dựng mạng Bayes từ các số liệu thống kê nói trên. (1 đ)

(b) Khi trời mưa to, tính xác suất để đường bị ngập nhưng không có cây đổ. (1 đ)

(c) Hai biến cỏ đường bị ngập và giờ cao điểm có độc lập với nhau hay không?

Chứng minh.

Ta nhận thấy xác suất cây đổ và đường ngập phụ thuộc vào xác suất mưa to.

Kẹt xe bị ảnh hưởng bởi đường ngập hoặc giờ cao điểm.

Đáp án:

a): Hãy đề xuất một mô hình bao gồm 10 người cho khái niệm "nhiệt độ trung bình" bằng cách sử dụng một tập hợp nhỏ tương ứng. Giả sử, miền giá trị của nhiệt độ là [-50, 50] độ F.

b): Cho bảng các thuộc tính-Phân loại về khái niệm Elephant như dưới đây:

Sample	GRAY?	MAMMAL?	LARGE?
1	0	1	1
2	1	0	1
3	0	1	0
4	1	1	1
5	0	0	0
6	1	1	0
7	0	0	1
8	1	1	1
9	0	0	0
10	1	1	1

c):

b) Mưa to đã xảy ra, cần tính xác suất đường ngập và không có cây đổ. Dấu phẩy ở đây có vai trò giống như dấu "&".

$$P(\text{ĐN}, \neg\text{CD} | \text{MT}) = P(\text{ĐN}, \neg\text{CD}, \text{MT}) / P(\text{MT})$$

Biến cỏ của đường ngập và cây đổ là mưa to

$$= P(\text{ĐN} | \text{MT}) * P(\neg\text{CD} | \text{MT}) * P(\text{MT}) / P(\text{MT})$$

$$= 0.7 * (1 - P(\text{CD} | \text{MT})) = 0.7 * 0.9 = 0.63$$

c) $P(\text{GCD}) = P(\text{GCD} | \text{ĐN})$

$$P(\text{GCD} | \text{DN}) = P(\text{GCD}, \text{DN}) / P(\text{DN}) = (P(\text{GCD}, \text{DN}, \text{MT}) + P(\text{GCD}, \text{DN}, \neg\text{MT})) / P(\text{DN})$$

$$= [P(\text{GCD}) * P(\text{DN}, \text{MT}) + P(\text{GCD}) * P(\text{DN}, \neg\text{MT})] / P(\text{DN})$$

$$= P(\text{GCD}) * (P(\text{DN}, \text{MT}) + P(\text{DN}, \neg\text{MT})) / P(\text{DN})$$

$$= P(\text{GCD}) * P(\text{DN}) / P(\text{DN})$$

$$= P(\text{GCD}) (\text{dpcm})$$

Câu 5 (2 điểm)

Xét bài toán liên quan đến dịch Covid ở Tp.HCM. Xác suất một người tiếp xúc gần với F0 là 30%. Nếu một người tiếp xúc gần với F0 thì xác suất bị nhiễm virus corona là 65%. Khi một người đã nhiễm virus Corona thì xác suất người đó dương tính khi test, mất vị giác hoặc khứu giác lần lượt là 90% và 60%. Ngược lại, nếu người đó không nhiễm virus corona thì xác suất người đó dương tính khi test, mất vị giác hoặc khứu giác lần lượt 1% và 3%. Khi một người bị nhiễm virus corona hoặc bị cúm thì người đó có thể bị ho. Nếu bị nhiễm virus corona và bị cúm thì xác suất ho là 99,9%, nếu không bị nhiễm virus corona cũng như không bị cúm thì xác suất ho là 1%. Nếu chỉ bị nhiễm virus corona thì xác suất ho là 80%, còn nếu chỉ bị cúm thì xác suất ho là 70%.

- Xây dựng mạng Bayes từ các số liệu thống kê trên.
- Khi bị mất vị giác hoặc khứu giác thì xác suất người đó nhiễm virus corona là bao nhiêu?

Với những thông tin từ đề bài cung cấp thì sinh viên sẽ xây dựng 1 mạng bayes gồm các biến cố: tiếp xúc gần F0, test covid dương tính, bị mất vị giác hoặc khứu giác, bị cúm, ho. Sau khi có những thông tin trên sinh viên có thể tính toán các xác suất, nếu thông tin xác suất còn thiếu thì có thể dùng biến để thay thế.

III. Học máy

- Bayesian Network (Xác suất/Định lý Bayes).
- Decision Tree

Câu b) Vẽ cây quyết định.

Sử dụng mảng toàn bộ, xây dựng cây quyết định cho tập dữ liệu huấn luyện sau:

Day	Outlook	Temp	Humidity	Wind	PlayTennis
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Chọn nút gốc của cây quyết định:

Tập dữ liệu hiện tại có kết quả Yes và 5 kết quả No, ta ký hiệu là S: [9+,5-]

Theo công thức tính Entropy (độ hỗn tạp dữ liệu) của một tập:

$$Entropy(S) = -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$

trong đó:

- $p\oplus$ là tỷ lệ các mẫu thuộc lớp dương trong S.
- $p\ominus$ là tỷ lệ các mẫu thuộc lớp âm trong S.

$$Entropy(S) = \sum_{i=1}^c -p_i \log_2 p_i$$

Tổng quát, nếu có c lớp trong tập S, ta có

$$Entropy(S) = Entropy([9+, 5-]) = -\frac{9}{14} \log_2 \frac{9}{14} - \frac{5}{14} \log_2 \frac{5}{14} = 0.94$$

Vậy ta có:

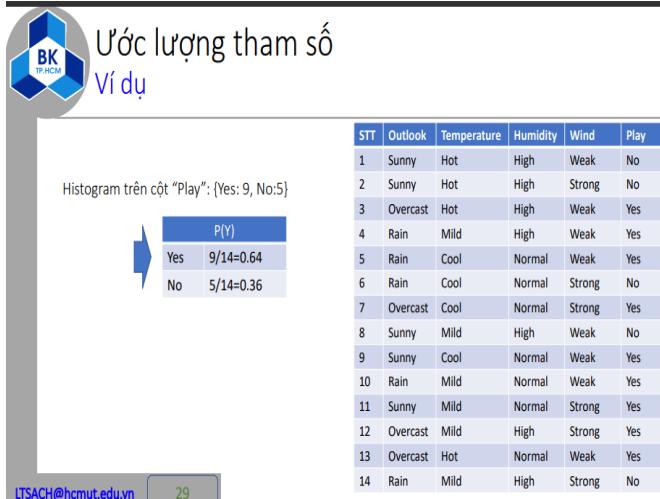
Lưu ý:

- Entropy là 0 nếu tất cả các thành viên của S đều thuộc về cùng một lớp.
- Entropy là 1 nếu tập hợp chứa số lượng bằng nhau các thành viên thuộc lớp âm và dương.

Tính toán IG và entropy cho tất cả đặc trưng (Có thể gộp lại khi tính)

Xét thuộc tính `Outlook`, thuộc tính này nhận 3 giá trị là Sunny, Overcast, Rain. Ứng với mỗi thuộc tính, ta có:

- SSunny: [2+,3-] (có nghĩa là trong tập dữ liệu hiện tại (S), có 2 kết quả Yes và 3 kết quả No tại Outlook = Sunny).



The figure shows a table with columns: STT, Outlook, Temperature, Humidity, Wind, Play. A note at the bottom says "Lựa chọn Play:Yes: có 9 hàng ($N_1 = 9$)". The background contains a large blue arrow pointing right.

STT	Outlook	Temperature	Humidity	Wind	Play
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Sunny	Cool	Normal	Weak	Yes
7	Rain	Mild	Normal	Weak	Yes
8	Overcast	Cool	Normal	Strong	Yes
9	Overcast	Hot	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

LTSACH@hcmut.edu.vn 30

Tương tự SOvercast: [4+,0-]. SRain: [3+,2-].

Tiếp theo tính Information Gain (độ lợi thông tin) của thuộc tính Outlook trên tập S. Thông số này phản ánh mức độ hiệu quả của một thuộc tính trong phân lớp. Đó là sự rút giảm mong muốn của Entropy gây ra bởi sự phân hoạch các mẫu dữ liệu theo thuộc tính này. Công thức tính IG của thuộc tính A trên tập S như sau:

$$Gain(S, A) = Entropy(S) - \sum_{v \in Value(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

trong đó:

Value(A) là tập các giá trị có thể cho thuộc tính A.

S_v là tập con của S mà A nhận giá trị v.

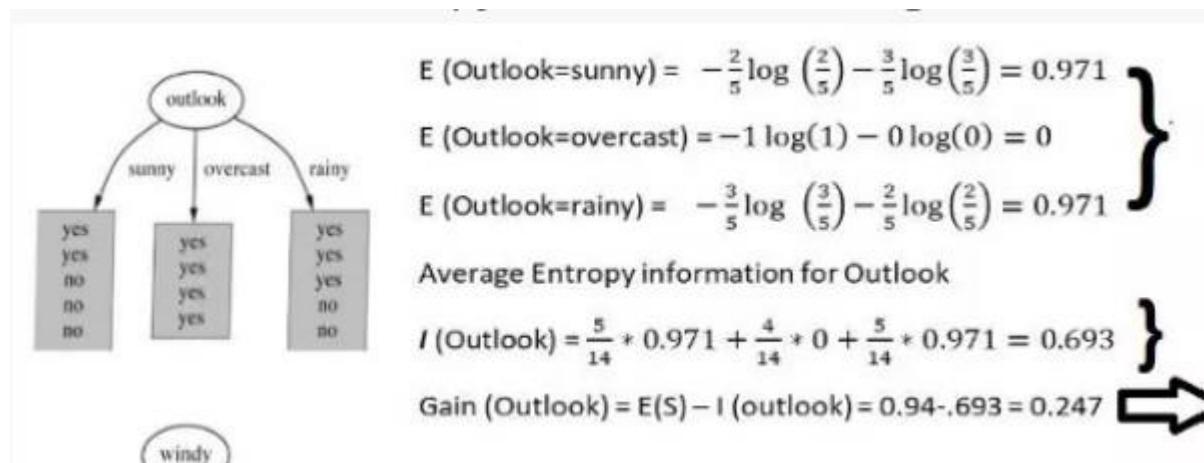
Lấy ví dụ với thuộc tính A = `Outlook`, ta có Value(A) = {Sunny, Overcast, Rain}, và $S_{Sunny} = [2+,3-]$ như đã tính ở trên

Từ công thức, dễ dàng tính được: (Trong thi có thể coi Entropy([9+;5+]) làm một hằng số, gọi là C.

$$Gain(S, Outlook) = Entropy([9+, 5-]) - \sum_{v \in Value(Outlook)} \frac{|S_v|}{14} Entropy(S_v) = 0.246$$

Giải thích con số 0.246.

Với $E(Outlook=Sunny)$, ta áp dụng công thức tính Entropy ở trên cho trường hợp $Sunny = False$ và $Sunny = True$. Ở đây, ta nhận thấy có 2 True và 3 False. Áp dụng tương tự với các trường hợp Overcast, Rainy.



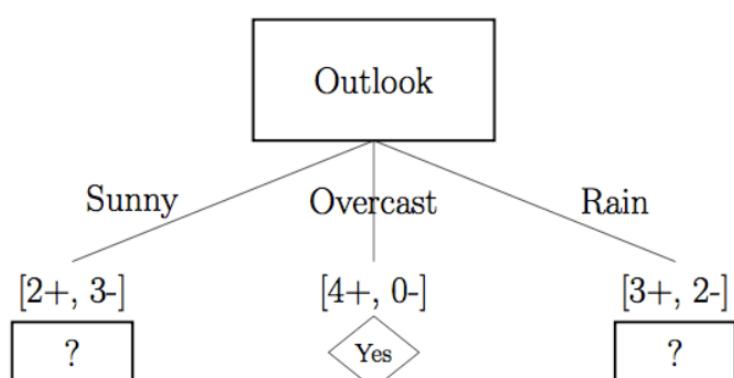
Sau đó, ta tính $I(\text{OutLook}) = \text{tổng} ([\text{số lượng phân tử của trường hợp} / \text{tổng số lượng mẫu}] * \text{entropy của trường hợp đó})$.

Ta lấy $Gain(S, OutLook) = Entropy(S) - I(\text{OutLook})$. I nào **nhỏ nhất** thì nhận trong trường hợp ta không muốn tính $Entropy(S)$ vì I càng nhỏ thì phép trừ trên cho ra kết quả Gain càng lớn.

Tương tự chúng ta có thể tính IG và entropy cho 2 đặc trưng còn lại. Chọn giá trị có gain cao nhất

Outlook	Temperature
Info: 0.693 Gain: 0.940-0.693	Info: 0.911 Gain: 0.940-0.911
Info: 0.788 Gain: 0.940-0.788	Info: 0.892 Gain: 0.940-0.892
	Windy

Thuộc tính **Outlook** có Information Gain cao nhất, chọn nó làm nút gốc. Các node con lần lượt là các trường hợp của thuộc tính đó. (Sunny, Overcast, Rain)



Overcast có $4+, 0-$ → chắc chắn Yes, khôi tính cho mêt.

Xây dựng tiếp cây quyết định:

Sau khi chọn được nút gốc là **Outlook**, tiếp theo ta tính tiếp các nút tại mỗi thuộc tính của nút vừa chọn. Trong hình 1:

Nhánh bên trái cùng ứng với **Outlook = Sunny**, có S_{Sunny} là $[2+, 3-]$, chưa phân lớp hoàn toàn nên vẫn phải tính toán chọn nút tại đây. Tương tự cho nhánh phải cùng.

Nhánh ở giữa ứng với **Outlook = Overcast**, tập dữ liệu tại nhánh này đã hoàn toàn phân lớp dương với $4+$ và $0-$. Tại đây đã có thể quyết định, khi **Outlook = Overcast** thì có thể đi chơi tennis.

Bây giờ ta sẽ thực hiện tính toán với nhánh trái cùng, trên tập $S_{Sunny} = [2+, 3-]$.

Hoàn toàn tương tự như cách tìm nút gốc, ta tính Information Gain cho 3 thuộc tính còn lại là **Temp**, **Humidity** và **Wind** (trên tập S_{Sunny}).

Xét thuộc tính **Humidity**, có: $S_{Normal} = [2+, 0-]$ (nghĩa là tại những dữ liệu có **Outlook = Sunny** và **Humidity = Normal**, có 2 dữ liệu, tất cả đều cho kết quả Yes).

$S_{High} = [0, 3-]$. Kết quả tương tự vậy.

Từ đó:

Ta có: $E(\text{Humidity} = \text{Normal} | \text{Outlook} = \text{Sunny}) = -2/2 * \log_2(2/2) = 0$.

$E(\text{Humidity} = \text{High} | \text{Outlook} = \text{Sunny}) = -3/3 * \log_2(3/3) = 0$

$I(\text{Humidity} | \text{Outlook} = \text{Sunny}) = 2/5 * E(\text{Humidity} = \text{Normal} | \text{Outlook} = \text{Sunny}) + 3/5 * E(\text{Humidity} = \text{High} | \text{Outlook} = \text{Sunny}) = 0$.

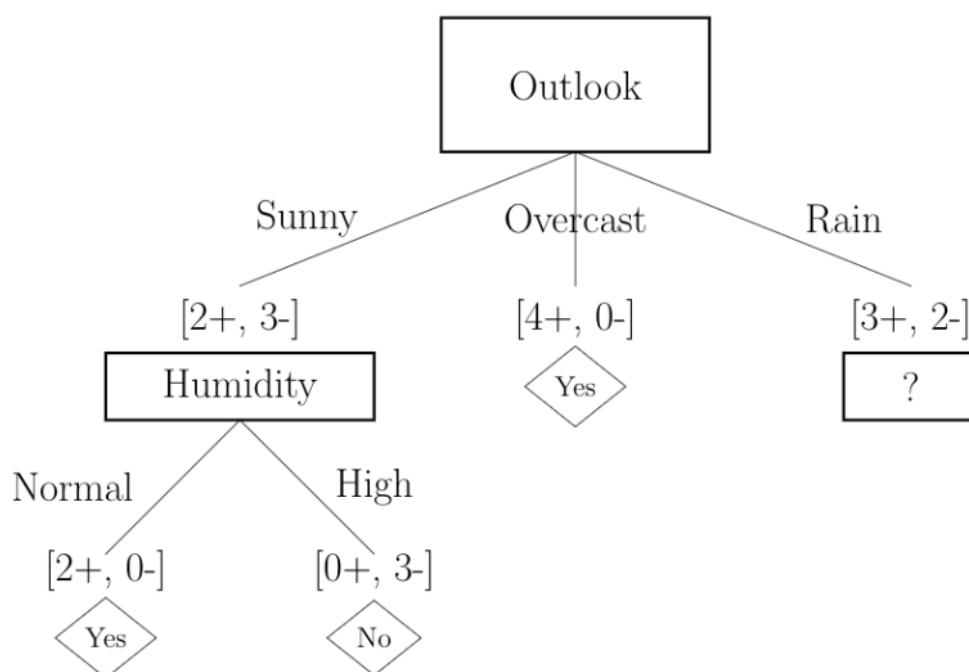
$\text{Gain}(S_{Sunny}, \text{Humidity}) = \text{Entropy}(2+, 3-) - I(\text{Humidity} | \text{Outlook} = \text{Sunny}) = 0.971$.

Tương tự:

$$\text{Gain}(S_{Sunny}, \text{Temp}) = 0.571$$

$$\text{Gain}(S_{Sunny}, \text{Wind}) = 0.019$$

Nhận thấy thuộc tính **Humidity** có Information Gain cao nhất, chọn thuộc tính này làm nút cho nhánh trái cùng.

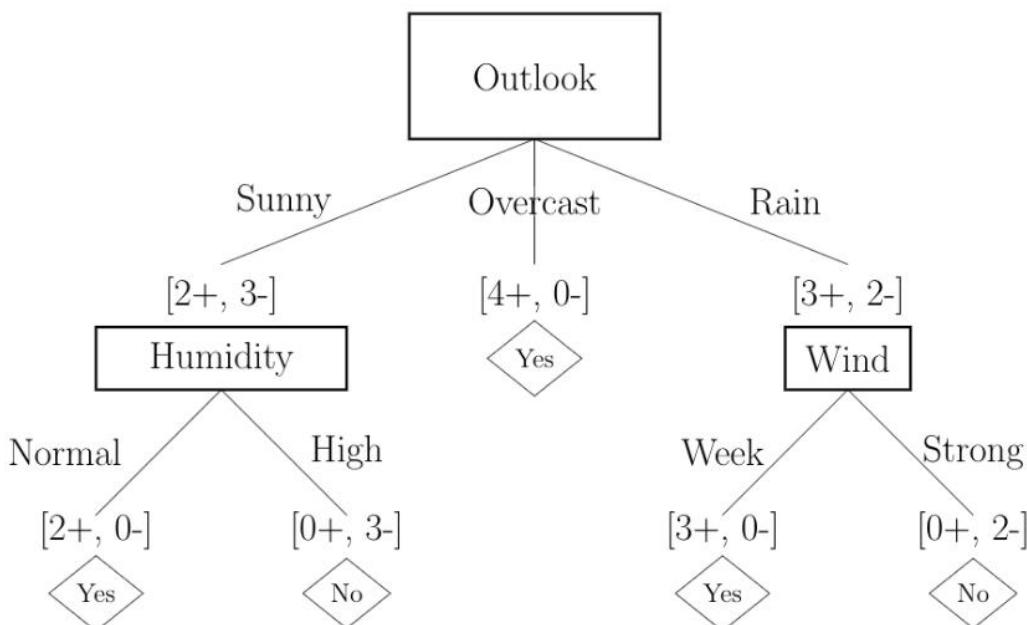


Hình 2. Cây quyết định sau khi chọn nút cho nhánh trái cùng.

Hình 2. Cây quyết định sau khi chọn nút cho nhánh trái cùng.

Cây quyết định hoàn chỉnh:

Làm tương tự cho nút tại nhánh phải ngoài (đến khi tất cả các nút lá của cây đều đã phân lớp), ta được cây quyết định hoàn chỉnh như sau:



Hình 3. Cây quyết định hoàn chỉnh.

b) Naive Bayes

- Bước 1. Vẽ mạng bayes.



Ước lượng tham số

Ví dụ

STT	Outlook	Temperature	Humidity	Wind	Play
1	Overcast	Hot	High	Weak	Yes
2	Rain	Mild	High	Weak	Yes
3	Rain	Cool	Normal	Weak	Yes
4	Overcast	Cool	Normal	Strong	Yes
5	Sunny	Cool	Normal	Weak	Yes
6	Rain	Mild	Normal	Weak	Yes
7	Sunny	Mild	Normal	Strong	Yes
8	Overcast	Mild	High	Strong	Yes
9	Overcast	Hot	Normal	Weak	Yes

Lựa chọn Play=Yes: có 9 hàng ($N_1 = 9$)

Histogram trên các cột:

- Outlook: {Overcast: 4; Rain: 3; Sunny: 2}
- Temperature: {Hot: 2; Mild: 4; Cool: 3}
- Humidity: {High: 3; Normal: 6}
- Wind: {Weak: 6; Strong: 3}

LTSACH@hcmut.edu.vn

32



Ước lượng tham số Ví dụ

Histogram trên cột "Play": {Yes: 9, No:5}

P(Y)	
Yes	9/14=0.64
No	5/14=0.36

STT	Outlook	Temperature	Humidity	Wind	Play
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

LTSACH@hcmut.edu.vn 29



Ước lượng tham số Ví dụ

STT	Outlook	Temperature	Humidity	Wind	Play
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Lựa chọn Play=No: có 5 hàng ($N_1 = 5$)

LTSACH@hcmut.edu.vn 31

STT	Outlook	Temperature	Humidity	Wind	Play
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Lựa chọn Play=Yes: có 9 hàng ($N_1 = 9$)

LTSACH@hcmut.edu.vn 32



Ước lượng tham số Ví dụ

Histogram trên các cột:

- Outlook: {Overcast: 0; Rain:2; Sunny:3}
- Temperature: {Hot: 2; Mild: 2; Cool: 1}
- Humidity: {High: 4; Normal: 1}
- Wind: {Weak:2; Strong: 3}

P(W Y=No)		
Y	W	
No	Weak	2/5=0.4
No	Strong	3/5=0.6

P(H Y=No)		
Y	T	
No	High	4/5=0.8
No	Normal	1/5=0.2

P(O Y=No)		
Y	O	
No	Overcast	0/5=0.0
No	Rain	2/5=0.4
No	Sunny	3/5=0.6

P(T Y=No)		
Y	T	
No	Hot	2/5=0.4
No	Mild	2/5=0.4
No	Cool	1/5=0.2

LTSACH@hcmut.edu.vn 35



Ước lượng tham số Ví dụ

STT	Outlook	Temperature	Humidity	Wind	Play
1	Overcast	Hot	High	Weak	Yes
2	Rain	Mild	High	Weak	Yes
3	Rain	Cool	Normal	Weak	Yes
4	Overcast	Cool	Normal	Strong	Yes
5	Sunny	Cool	Normal	Weak	Yes
6	Rain	Mild	Normal	Weak	Yes
7	Sunny	Mild	Normal	Strong	Yes
8	Overcast	Mild	High	Strong	Yes
9	Overcast	Hot	Normal	Weak	Yes

Lựa chọn Play=Yes: có 9 hàng ($N_1 = 9$)

LTSACH@hcmut.edu.vn 30

STT	Outlook	Temperature	Humidity	Wind	Play
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Overcast	Cool	Normal	Strong	Yes
7	Sunny	Cool	Normal	Weak	Yes
8	Sunny	Mild	Normal	Strong	Yes
9	Overcast	Mild	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Histogram trên các cột:

- Outlook: {Overcast: 4; Rain:3; Sunny:2}
- Temperature: {Hot: 2; Mild: 4; Cool: 3}
- Humidity: {High: 3; Normal: 6}
- Wind: {Weak:6; Strong: 3}

P(O Y=Yes)		
Y	O	
Yes	Overcast	4/9=0.44
Yes	Rain	3/9=0.33
Yes	Sunny	2/9=0.23
No	Overcast	0/5=0.0
No	Rain	2/5=0.4
No	Sunny	3/5=0.6

P(T Y=Yes)		
Y	T	
Yes	Hot	2/9=0.23
Yes	Mild	4/9=0.44
Yes	Cool	3/9=0.33
No	Hot	2/5=0.4
No	Mild	2/5=0.4
No	Cool	1/5=0.2

LTSACH@hcmut.edu.vn 36

LTSACH@hcmut.edu.vn 37



Ước lượng tham số Ví dụ

Histogram trên các cột:

- Outlook: {Overcast: 4; Rain:2; Sunny:3}
- Temperature: {Hot: 2; Mild: 4; Cool: 3}
- Humidity: {High: 3; Normal: 6}
- Wind: {Weak:6; Strong: 3}

P(T Y)		
Y	T	
Yes	Hot	2/9=0.23
Yes	Mild	4/9=0.44
Yes	Cool	3/9=0.33
No	Hot	2/5=0.4
No	Mild	2/5=0.4
No	Cool	1/5=0.2

P(O Y)		
Y	O	
Yes	Overcast	4/9=0.44
Yes	Rain	3/9=0.33
Yes	Sunny	2/9=0.23
No	Overcast	0/5=0.0
No	Rain	2/5=0.4
No	Sunny	3/5=0.6

LTSACH@hcmut.edu.vn 38

STT	Outlook	Temperature	Humidity	Wind	Play
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Overcast	Cool	Normal	Strong	Yes
7	Sunny	Cool	Normal	Weak	Yes
8	Sunny	Mild	Normal	Strong	Yes
9	Overcast	Mild	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Histogram trên các cột:

- Outlook: {Overcast: 0; Rain:2; Sunny:3}
- Temperature: {Hot: 2; Mild: 2; Cool: 1}
- Humidity: {High: 4; Normal: 1}
- Wind: {Weak:2; Strong: 3}



Ước lượng tham số

Ví dụ

Histogram trên các cột:

- Outlook: {Overcast: 4; Rain:3; Sunny:2}
- Temperature: {Hot: 2; Mild: 4; Cool: 3}
- Humidity: {High: 3; Normal: 6}
- Wind: {Weak:6; Strong: 3}

Histogram trên các cột:

- Outlook: {Overcast: 0; Rain:2; Sunny:3}
- Temperature: {Hot: 2; Mild: 2; Cool: 1}
- Humidity: {High: 4; Normal: 1}
- Wind: {Weak:2; Strong: 3}

LTSACH@hcmut.edu.vn

39

P(W Y)		
Y	H	P(W Y)
Yes	Weak	6/9=0.67
Yes	Strong	3/9=0.33
No	Weak	2/5=0.4
No	Strong	3/5=0.6

LTSACH@hcmut.edu.vn

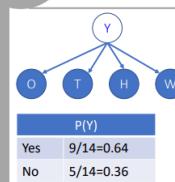
40



Ước lượng tham số

Ví dụ

<https://medium.com/@arunm8489/naive-bayes-for-machine-learning-238e8039edee>



Y	O	P(O Y)
Yes	Overcast	4/9=0.44
Yes	Rain	3/9=0.33
Yes	Sunny	2/9=0.23
No	Overcast	0/5=0.0
No	Rain	2/5=0.4
No	Sunny	3/5=0.6

Y	T	P(T Y)
Yes	Hot	2/9=0.23
Yes	Mild	4/9=0.44
Yes	Cool	3/9=0.33
No	Hot	2/5=0.4
No	Mild	2/5=0.4
No	Cool	1/5=0.2

Y	H	P(H Y)
Yes	High	3/9=0.33
Yes	Normal	6/9=0.67
No	High	4/5=0.8
No	Normal	1/5=0.2

Y	H	P(W Y)
Yes	Weak	6/9=0.67
Yes	Strong	3/9=0.33
No	Weak	2/5=0.4
No	Strong	3/5=0.6

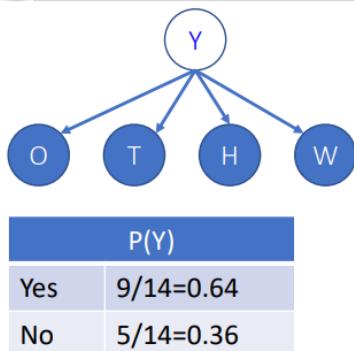
LTSACH@hcmut.edu.vn

39

Ước lượng tham số

Ví dụ

<https://medium.com/@arunm8489/naive-bayes-for-machine-learning-238e8039edee>



P(Y)	
Yes	9/14=0.64
No	5/14=0.36

LTSACH@hcmut.edu.vn

40

P(O Y)		
Y	O	P(O Y)
Yes	Overcast	4/9=0.44
Yes	Rain	3/9=0.33
Yes	Sunny	2/9=0.23
No	Overcast	0/5=0.0
No	Rain	2/5=0.4
No	Sunny	3/5=0.6

P(T Y)		
Y	T	P(T Y)
Yes	Hot	2/9=0.23
Yes	Mild	4/9=0.44
Yes	Cool	3/9=0.33
No	Hot	2/5=0.4
No	Mild	2/5=0.4
No	Cool	1/5=0.2

P(H Y)		
Y	H	P(H Y)
Yes	High	3/9=0.33
Yes	Normal	6/9=0.67
No	High	4/5=0.8
No	Normal	1/5=0.2

P(W Y)		
Y	H	P(W Y)
Yes	Weak	6/9=0.67
Yes	Strong	3/9=0.33
No	Weak	2/5=0.4
No	Strong	3/5=0.6



Naïve Bayes cho bài toán phân loại

Ví dụ 1

P(K)		P(P K)		
K	P	P(.)		
+	0.001			
-	0.999			
			0.95	
			0.05	
			0.2	
			0.8	

Triệu chứng:

- Thở dốc (B): +
- Đau ngực (P): +
- Ho (C) : -
- => Véc tơ đặc trưng: $x = [+, +, -]$

Hỏi: có khả năng ung thư phổi?

P(B K)			P(C K)		
K	B	P(.)	K	C	P(.)
+	+	0.8	+	+	0.7
+	-	0.2	+	-	0.3
-	+	0.3	-	+	0.4
-	-	0.7	-	-	0.6

$$P(C = + | x) = P(C = +) \times P(B = + | +) \times P(P = + | +) \times P(C = - | +)$$

$$K = 0.001 \times 0.8 \times 0.95 \times 0.3 = 0.000228$$

$$P(C = - | x) = P(C = -) \times P(B = + | -) \times P(P = + | -) \times P(C = - | -)$$

$$K = 0.999 \times 0.3 \times 0.2 \times 0.6 = 0.035964$$

$P(C = + | x) < P(C = - | x)$ Không ung thư phổi!

Probability

Probability

Example:

S = stiff neck

M = meningitis

$P(S | M) = 0.5$

$P(M) = 1/50000$

$P(S) = 1/20$

$P(M | S) = P(S | M).P(M)/P(S) = 1/5000$

Conditional probability: probability in the presence of some evidence

$P(\text{Dice} = 2 | \text{Dice is even}) = 1/3$

$P(\text{Dice} = 2 | \text{Dice is odd}) = 0$

$P(A | B) = P(A \wedge B)/P(B)$

$P(A \wedge B) = P(A | B).P(B)$

Probability

Probability

Joint probability distributions:

X: $\langle x_1, \dots, x_m \rangle$ Y: $\langle y_1, \dots, y_n \rangle$

$P(X = x_i, Y = y_j)$

Axioms:

- $0 \leq P(A) \leq 1$
- $P(\text{true}) = 1$ and $P(\text{false}) = 0$
- $P(A \vee B) = P(A) + P(B) - P(A \wedge B)$

Probability

Probability

Derived properties:

$$\bullet P(\neg A) = 1 - P(A)$$

$$\bullet P(U) = P(A_1) + P(A_2) + \dots + P(A_n)$$

$U = A_1 \vee A_2 \vee \dots \vee A_n$ collectively exhaustive

$A_i \wedge A_j = \text{false}$ mutually exclusive

Bayes' theorem:

$$P(H_i | E) = P(E | H_i).P(H_i) / \sum_i P(E | H_i).P(H_i)$$

H_i 's are collectively exhaustive & mutually exclusive

Probability

- Independence:

$$P(A \wedge B) = P(A).P(B)$$

$$P(A) = P(A | B)$$

- Conditional independence:

$$P(A \wedge B | E) = P(A | E).P(B | E)$$

$$P(A | E) = P(A | E \wedge B)$$

Example:

$$P(\text{Toothache} | \text{Cavity} \wedge \text{Catch}) = P(\text{Toothache} | \text{Cavity})$$

$$P(\text{Catch} | \text{Cavity} \wedge \text{Toothache}) = P(\text{Catch} | \text{Cavity})$$

Bayesian Networks

Semantics:

- An ordering on the nodes: X_i is a predecessor of $X_j \Rightarrow i < j$
- $P(X_1, X_2, \dots, X_n) = P(X_n | X_{n-1}, \dots, X_1)$
 $= P(X_n | X_{n-1}, \dots, X_1).P(X_{n-1} | X_{n-2}, \dots, X_1). \dots .P(X_2 | X_1).P(X_1)$
 $= \prod_i P(X_i | X_{i-1}, \dots, X_1) = \prod_i P(X_i | \text{Parents}(X_i))$

$P(X_i | X_{i-1}, \dots, X_1) = P(X_i | \text{Parents}(X_i)) \quad \text{Parents}(X_i) \subseteq \{X_{i-1}, \dots, X_1\}$

Each node is conditionally independent of its predecessors given its parents

$P(A, B) = P(A | B) + P(B)$

X_n, \sim Bayesian Networks

$$P(A, B) = P(A|B) \cdot P(B)$$

$$P(X_n, X_{n-1}, \dots, X_1) \quad \text{Bayesian Networks}$$

$$= P(X_n | X_{n-1}, \dots, X_1) \cdot P(X_{n-1} | X_{n-2}, \dots, X_1) \cdot \dots$$

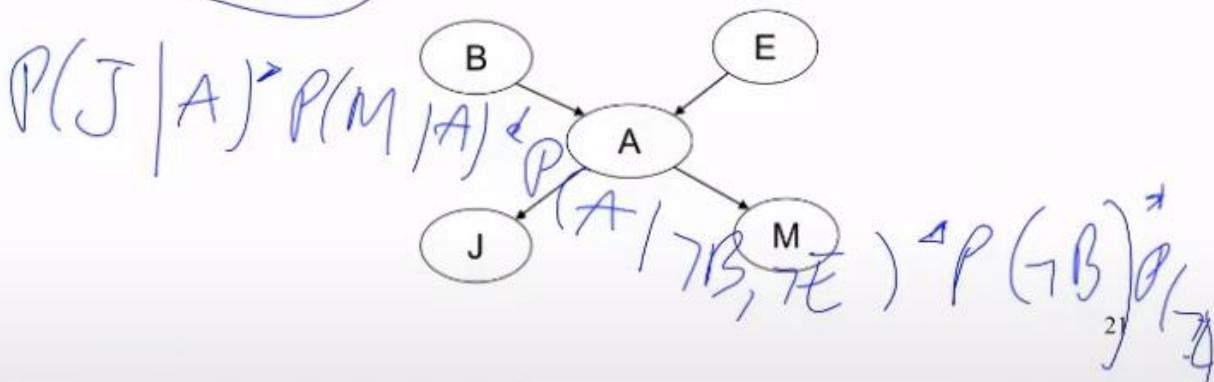
Example:

$$P(J \wedge M \wedge A \wedge \neg B \wedge \neg E)$$

$$= P(J|A) \cdot P(M|A) \cdot P(A|\neg B \wedge \neg E) \cdot P(\neg B) \cdot P(\neg E)$$

$$= 0.00062$$

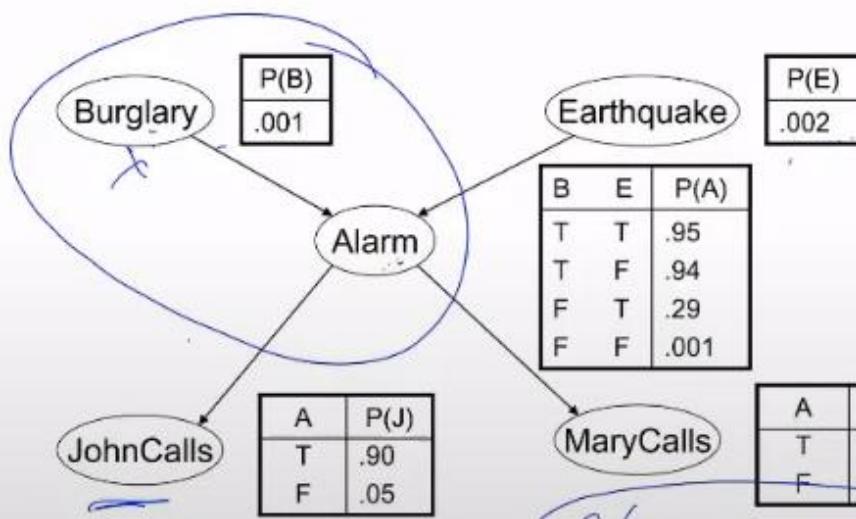
$$P(X_{n-1} | X_{n-2}, \dots, X_1)$$



$$P(B \mid A) = \frac{P(B, A)}{P(A)} = \frac{P(A, B, E) + P(A, B, \neg E)}{P(A)} \quad 17$$

$$P(J \mid \neg E) = \frac{P(J, \neg E)}{P(\neg E)}$$

Bayesian Networks



$$\begin{aligned} P(J, \neg E) &= P(J, A, \neg E) + P(J, \neg A, \neg E) \\ &= P(J, A, \neg E, B) + P(J, \neg A, \neg E, \neg B) \end{aligned} \quad 18$$

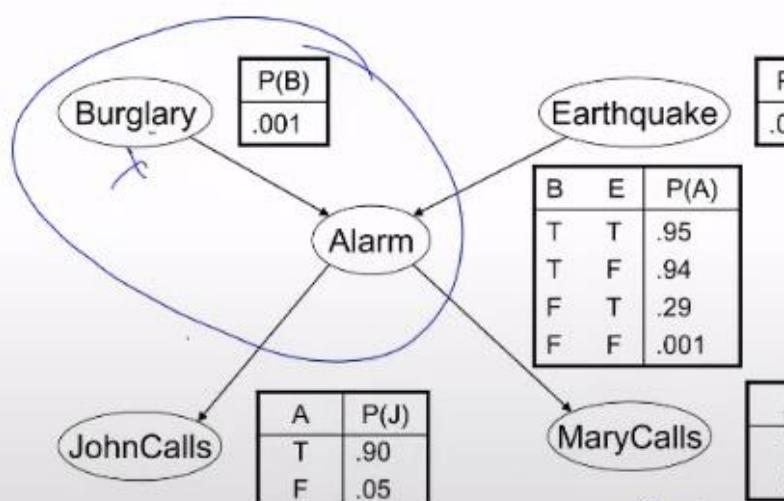
$$P(\neg J, \neg A, \neg E, \neg B) = P(\neg J | \neg A) \cdot P(\neg A | \neg E, \neg B)$$

18 70

$$P(\underline{B} \mid \underline{A}) = \frac{P(\underline{B}, \underline{A})}{P(\underline{A})} = \frac{P(A, B, E) + P(A, B, \bar{E})}{P(A)} \quad 17$$

$$P(\bar{J} \mid \bar{E}) =$$

Bayesian Networks



$$\begin{aligned} P(A) &= P(A, B) + P(A, \neg B, E) + P(A, \neg B, \neg E) \\ &= P(A, B, E) + P(A, B, \neg E) + P(A, \neg B, E) \end{aligned}$$

1. Bài toán này dùng để làm gì?

Bài toán này triển khai một trò chơi cờ Caro (Gomoku) trên máy tính, trong đó các tác nhân (agent) sử dụng các thuật toán trí tuệ nhân tạo để chơi hoặc thi đấu với nhau hoặc với người chơi. Cụ thể:

- **Mục đích chính:** Xây dựng các agent thông minh (Random, Minimax, Alpha-Beta) để chơi cờ Caro, với mục tiêu tìm ra nước đi tốt nhất nhằm chiến thắng hoặc hòa trong trò chơi.
- **Ứng dụng:**
 - **Chơi cờ Caro:** Cho phép người chơi đấu với máy (AI) hoặc máy đấu với máy.
 - **Đánh giá hiệu suất AI:** Chức năng evaluate_agents() trong main.py được dùng để so sánh hiệu suất của các agent (Random, Minimax, Alpha-Beta) ở các độ sâu tìm kiếm khác nhau, qua đó xác định agent nào hoạt động tốt nhất.
 - **Học thuật và nghiên cứu:** Mô phỏng các thuật toán tìm kiếm như Minimax và Alpha-Beta Pruning để nghiên cứu hiệu quả của chúng trong trò chơi đối kháng.

Tóm lại, bài toán này dùng để mô phỏng trò chơi cờ Caro, phát triển và đánh giá các thuật toán AI trong việc đưa ra quyết định chiến lược.

2. Hàm lượng giá trong bài toán là gì, tại sao lại dùng hàm lượng giá đó?

Hàm lượng giá là gì?

Hàm lượng giá (evaluation function) được sử dụng để đánh giá trạng thái của bàn cờ tại một thời điểm nhất định, nhằm giúp các agent (Minimax và Alpha-Beta) quyết định nước đi tốt nhất khi không thể tìm kiếm đến trạng thái kết thúc (ví dụ: khi độ sâu tìm kiếm đạt giới hạn hoặc bàn cờ đầy). Hàm này trả về một giá trị số, biểu thị mức độ "tốt" của trạng thái bàn cờ đối với agent.

- **Trong minimax_agent.py:** Hàm lượng giá chính là _evaluate_board(self, board), được tính toán dựa trên:
 - Điểm số từ các mẫu (patterns) trên bàn cờ, sử dụng _evaluate_patterns(board, symbol, weight).
 - Các mẫu được đánh giá bao gồm số quân liên tiếp (2, 3, 4, 5) và số đầu mở (open ends), với điểm cố định trong self.pattern_scores:
 - 5 quân liên tiếp: 10,000 điểm (thắng).
 - 4 quân liên tiếp: 500 (2 đầu mở), 100 (1 đầu mở).
 - 3 quân liên tiếp: 50 (2 đầu mở), 10 (1 đầu mở).
 - 2 quân liên tiếp: 5 (2 đầu mở), 2 (1 đầu mở).
 - Tổng điểm được tính bằng cách: Điểm của agent (symbol) trừ đi điểm của đối thủ (opponent_symbol) với trọng số 0.9 cho đối thủ.
- **Trong alphabeta_agent.py:** Hàm lượng giá cũng là _evaluate_board(self, board), nhưng phức tạp hơn:
 - Sử dụng hàm board.evaluate(self.symbol) làm điểm cơ bản (được cung cấp bởi lớp Board, không có chi tiết cụ thể trong code).
 - Đánh giá thêm các yếu tố:
 - **Tấn công và phòng thủ:** Đánh giá các dòng (lines) theo 4 hướng (ngang, dọc, chéo chính, chéo phụ) thông qua _count_consecutive() và _score_line().
 - 5 quân liên tiếp: 10,000 điểm.
 - 4 quân liên tiếp: 5,000 (2 đầu mở), 500 (1 đầu mở).
 - 3 quân liên tiếp: 200 (2 đầu mở), 50 (1 đầu mở).
 - 2 quân liên tiếp: 10 (2 đầu mở), 5 (1 đầu mở).
 - **Phòng thủ:** Áp dụng trọng số self.defense_weight = 1.2 để ưu tiên phòng thủ.
 - **Kiểm soát trung tâm:** Đánh giá thêm điểm cho việc kiểm soát trung tâm bàn cờ thông qua _evaluate_center_control().
 - Tổng điểm: base_score + attack_score + self.defense_weight * defense_score + center_score.

Tại sao dùng hàm lượng giá đó?

- **Cần thiết cho thuật toán tìm kiếm:**
 - Cả Minimax và Alpha-Beta Pruning đều cần một cách để đánh giá trạng thái bàn cờ khi không thể tìm kiếm đến trạng thái cuối (do độ sâu giới hạn hoặc bàn cờ chưa kết thúc).
Hàm lượng giá giúp định lượng trạng thái, từ đó chọn nước đi tối ưu.
 - Nếu không có hàm lượng giá, các thuật toán này chỉ có thể dựa vào kết quả thắng/thua (khi đến trạng thái cuối), điều này không khả thi với bàn cờ lớn (như 10x10 hoặc 15x15) vì không gian tìm kiếm quá lớn.
- **Phản ánh chiến lược chơi cờ Caro:**
 - **Số quân liên tiếp và đầu mở:** Trong cờ Caro, các mảng như 4 quân liên tiếp với 2 đầu mở là rất nguy hiểm (gần thắng), nên được cho điểm cao. Điều này giúp agent ưu tiên tạo ra hoặc chặn các mảng nguy hiểm.
 - **Phòng thủ và tấn công:** Trọng số phòng thủ (1.2 trong Alpha-Beta) giúp agent cân bằng giữa tấn công (tạo cơ hội thắng) và phòng thủ (ngăn đối thủ thắng), rất quan trọng trong trò chơi đối kháng.
 - **Kiểm soát trung tâm:** Trong cờ Caro, kiểm soát trung tâm bàn cờ giúp dễ dàng mở rộng các dòng (lines) theo nhiều hướng, nên được cộng điểm (trong Alpha-Beta Agent).
- **Tối ưu hiệu suất:**
 - Hàm lượng giá được thiết kế để tính toán nhanh, tránh đánh giá lặp lại (sử dụng checked trong Minimax, hay chỉ đánh giá các hướng cần thiết trong Alpha-Beta).
 - Điểm số cố định (như 10,000 cho thắng, 5,000 cho 4 quân 2 đầu mở) giúp đơn giản hóa việc so sánh các trạng thái mà không cần tính toán phức tạp.

Tóm lại, hàm lượng giá được thiết kế để phản ánh các chiến lược quan trọng trong cờ Caro (tấn công, phòng thủ, kiểm soát trung tâm), đồng thời đảm bảo hiệu suất tính toán để các thuật toán AI hoạt động hiệu quả.

3. Định nghĩa trạng thái trong bài toán?

Trong bài toán này, trạng thái (state) được định nghĩa dựa trên trạng thái của bàn cờ tại một thời điểm nhất định trong trò chơi cờ Caro. Cụ thể:

- **Trạng thái bàn cờ:**
 - **Bàn cờ (board.board):** Là một ma trận 2 chiều (kích thước board.size x board.size), trong đó mỗi ô có thể là:
 - ' ': Ô trống (chưa có quân cờ).
 - 'X': Quân cờ của người chơi 1 (hoặc agent 1).
 - 'O': Quân cờ của người chơi 2 (hoặc agent 2).
 - **Thông tin bổ sung:**
 - board.last_move: Nước đi cuối cùng (tọa độ (row, col)), được sử dụng để tạo hash cho bàn cờ (trong Alpha-Beta Agent).
 - board.moves_count: Số nước đi đã thực hiện, giúp xác định trạng thái ban đầu (ví dụ: nước đi đầu tiên).
 - board.size: Kích thước của bàn cờ (ví dụ: 10x10 hoặc 15x15).
- **Trạng thái trong thuật toán tìm kiếm:**
 - Trong thuật toán Minimax và Alpha-Beta, trạng thái bao gồm:
 - **Bàn cờ hiện tại:** Được sao chép (board.copy()) để mô phỏng các nước đi.
 - **Người chơi hiện tại:** Được biểu thị qua is_maximizing (True: lượt của agent, False: lượt của đối thủ).
 - **Độ sâu tìm kiếm còn lại (depth):** Xác định mức độ tìm kiếm sâu hơn.
 - **Alpha và Beta:** Các giá trị dùng trong Alpha-Beta Pruning để cắt tia nhánh.
 - Trạng thái được băm (hashed) để lưu vào bộ nhớ đệm (transposition_table hoặc position_cache) nhằm tránh tính toán lặp lại:
 - Trong Alpha-Beta: board_hash = ".join([".join(row) for row in board.board]) + str(board.last_move).
 - Trong Minimax: board_hash = hash(str(board.board)).

- **Trạng thái kết thúc:**

- Một trạng thái được coi là trạng thái kết thúc khi:
 - Có người chiến thắng (board.check_winner() trả về 'X' hoặc 'O').
 - Bàn cờ đầy (board.is_full() trả về True).
 - Độ sâu tìm kiếm đạt 0 (depth == 0).

Tóm lại, trạng thái trong bài toán là toàn bộ thông tin về bàn cờ tại một thời điểm, bao gồm ma trận quân cờ, nước đi cuối cùng, và các thông tin bổ sung (người chơi hiện tại, độ sâu, v.v.), được sử dụng để mô phỏng và đánh giá các nước đi trong trò chơi cờ Caro.

Nếu bạn có thêm câu hỏi hoặc cần giải thích chi tiết hơn, hãy cho tôi biết!

Công thức tính hàm lượng giá trong bài toán

Hàm lượng giá (evaluation function) được sử dụng để đánh giá trạng thái bàn cờ, giúp các agent (Minimax và Alpha-Beta) đưa ra quyết định khi không thể tìm kiếm đến trạng thái kết thúc. Tôi sẽ trình bày công thức cho cả hai agent:

1. Hàm lượng giá trong `MinimaxAgent (minimax_agent.py)`

Hàm lượng giá chính là `_evaluate_board(self, board)`, được tính như sau:

- Công thức tổng quát:

$$\text{Score} = \text{EvaluatePatterns}(board, \text{symbol}, 1.0) - \text{EvaluatePatterns}(board, \text{opponent_symbol}, 0.9)$$

- Chi tiết hàm `EvaluatePatterns(board, symbol, weight)`:

- Duyệt qua từng ô trên bàn cờ, kiểm tra các mẫu (patterns) của quân cờ `symbol` theo 4 hướng (ngang, dọc, chéo chính, chéo phụ).
- Với mỗi mẫu, tính số quân liên tiếp (consecutive) và số đầu mờ (open_ends).
- Điểm số cho mỗi mẫu được tính dựa trên bảng `self.pattern_scores`:
 - 5 quân liên tiếp: 10,000 điểm.
 - 4 quân liên tiếp: 500 (2 đầu mờ), 100 (1 đầu mờ).
 - 3 quân liên tiếp: 50 (2 đầu mờ), 10 (1 đầu mờ).
 - 2 quân liên tiếp: 5 (2 đầu mờ), 2 (1 đầu mờ).

- Tổng điểm cho `symbol`:

$$\text{PatternScore}(\text{symbol}) = \sum_{\text{mẫu}} \text{PatternScore}[\text{consecutive}][\text{open_ends}] \times \text{weight}$$

- Công thức cụ thể:

$$\text{Score} = \sum_{\text{mẫu của symbol}} \text{PatternScore}[\text{consecutive}][\text{open_ends}] \times 1.0 - \sum_{\text{mẫu của opponent_symbol}} \text{PatternScore}[\text{consecutive}][\text{open_ends}] \times 0.9$$

2. Hàm lượng giá trong AlphaBetaAgent (alphabeta_agent.py)

Hàm lượng giá chính là `_evaluate_board(self, board)`, phức tạp hơn và bao gồm nhiều yếu tố:

- Công thức tổng quát:

$$\text{TotalScore} = \text{BaseScore} + \text{AttackScore} + \text{DefenseWeight} \times \text{DefenseScore} + \text{CenterScore}$$

Trong đó:

- **BaseScore:** Điểm cơ bản từ `board.evaluate(self.symbol)` (không có chi tiết cụ thể trong code, giả định là một hàm có sẵn trong lớp `Board`).
- **AttackScore:** Điểm tấn công, tính từ các mẫu của agent.
- **DefenseScore:** Điểm phòng thủ, tính từ các mẫu của đối thủ.
- **DefenseWeight = 1.2:** Trọng số ưu tiên phòng thủ.
- **CenterScore:** Điểm kiểm soát trung tâm.
- **Chi tiết các thành phần:**

1. AttackScore và DefenseScore:

- Duyệt qua từng ô trên bàn cờ, kiểm tra các mẫu theo 4 hướng.
- Với mỗi mẫu, tính số quân liên tiếp (count) và số đầu mở (open_ends) bằng `_count_consecutive()`.
- Điểm số cho mỗi mẫu được tính bằng `_score_line(count_data)` :
 - 5 quân liên tiếp: 10,000 điểm.
 - 4 quân liên tiếp: 5,000 (2 đầu mở), 500 (1 đầu mở).
 - 3 quân liên tiếp: 200 (2 đầu mở), 50 (1 đầu mở).
 - 2 quân liên tiếp: 10 (2 đầu mở), 5 (1 đầu mở).
- Tổng điểm:

$$\text{AttackScore} = \sum_{\text{mẫu của symbol}} \text{ScoreLine}(\text{count}, \text{open_ends})$$

$$\text{DefenseScore} = - \sum_{\text{mẫu của opponent_symbol}} \text{ScoreLine}(\text{count}, \text{open_ends})$$

2. CenterScore (từ `_evaluate_center_control()`):

- Tính điểm cho các ô trong vùng trung tâm (bán kính `center_radius = board.size // 4`).
- Với mỗi ô có quân cờ:
 - `distance = |row - center| + |col - center|`
 - `weight = max(0, center_radius - distance + 1)`
 - Nếu ô có quân của `symbol`: `score+ = weight × 2`
 - Nếu ô có quân của đối thủ: `score- = weight`
- Tổng điểm:

$$\text{CenterScore} = \sum_{\text{đ trong vùng trung tâm}} \text{weight} \times (2 \text{ nếu là symbol, } -1 \text{ nếu là opponent})$$

- Công thức cụ thể:

$$\text{TotalScore} = \text{board.evaluate(self.symbol)} + \sum_{\text{mẫu của symbol}} \text{ScoreLine}(\text{count}, \text{open_ends}) - 1.2 \times \sum_{\text{mẫu của opponent_symbol}} \text{ScoreLine}(\text{count}, \text{open_ends}) + \sum_{\text{đ trong trung tâm}} \text{weight} \times (2 \text{ nếu là symbol, } -1 \text{ nếu là opponent})$$