1. Tìm kiếm
2. Định nghĩa không gian trạng thái.

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

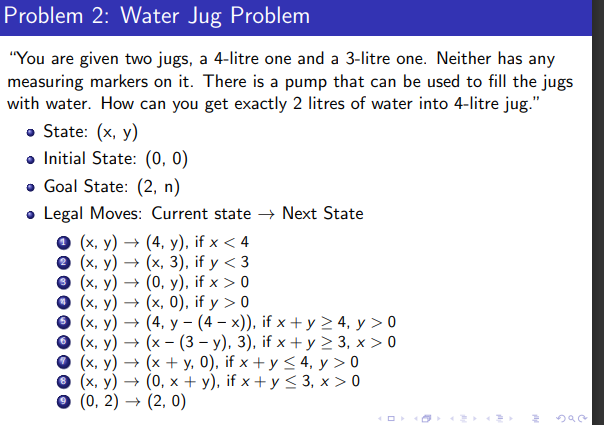
1 Define a state space that contains all the possible configurations of the relevant objects.

2 Specify the initial states.

3 Specify the goal states.

4 Specify a set of rules.

Example:



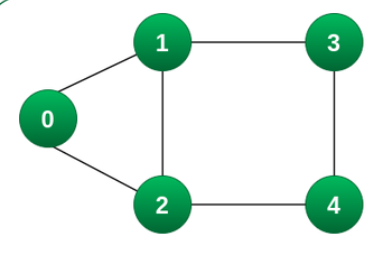
1. Tìm kiếm mù (BrFS, DFS): Vẽ cây minh hoạ quá trình duyệt.

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.

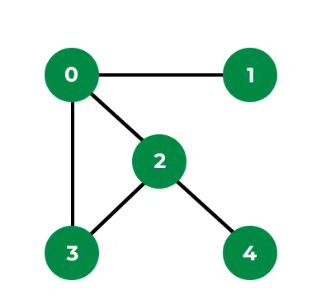
Như chúng ta có thể thấy rằng mọi hàng xóm của nút 3 đều được truy cập, vì vậy hãy chuyển sang nút tiếp theo ở phía trước hàng đợi.

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.

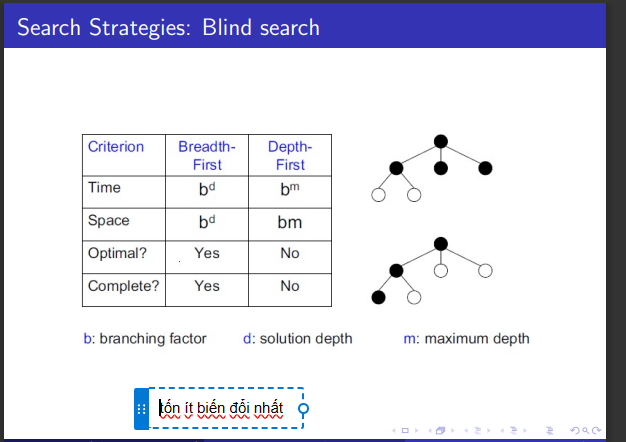
Bước 3: Bây giờ, Nút 1 ở đầu ngăn xếp, vì vậy hãy truy cập nút 1 và lấy nó ra khỏi ngăn xếp và đặt tất cả các nút lân cận chưa được truy cập vào ngăn xếp.

Bước 4: Bây giờ, 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 liền kề chưa được truy cập (tức là 3, 4) vào ngăn xếp.

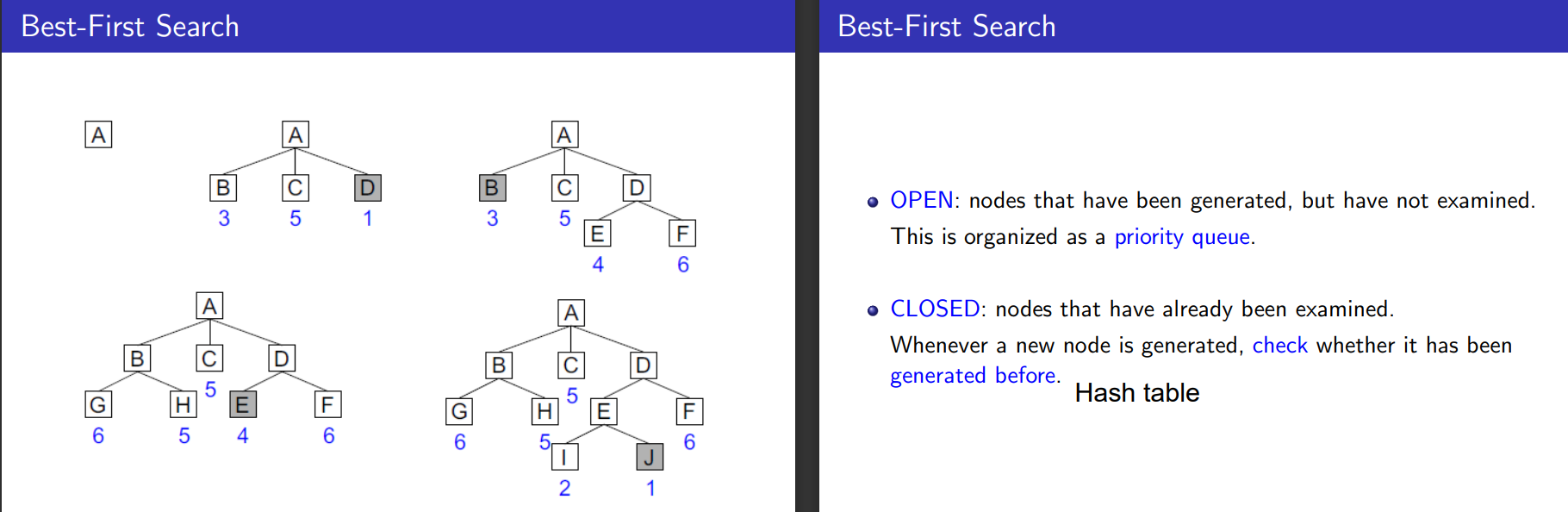
Bước 5: Bây giờ, Nút 4 ở đầu ngăn xếp, vì vậy hãy truy cập nút 4 và lấy nó ra khỏi ngăn xếp và đặt tất cả các nút lân cận chưa được truy cập vào ngăn xếp.

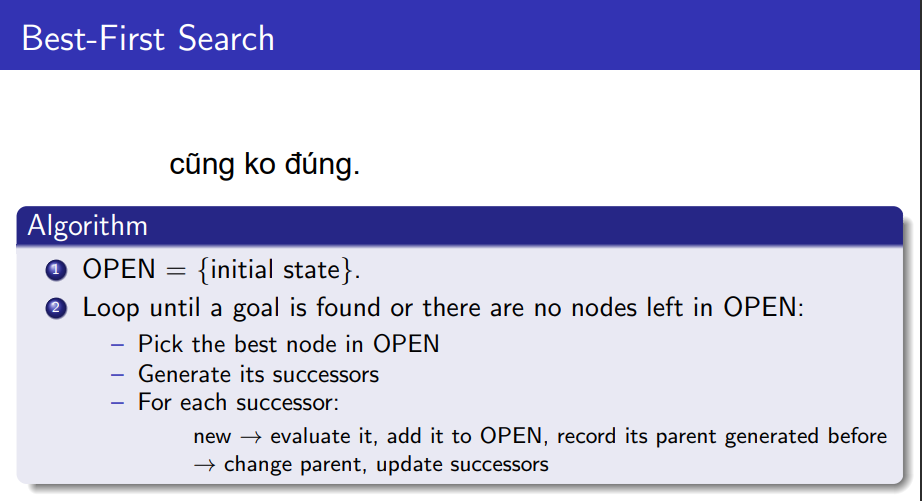
Bước 6: Bây giờ, Nút 3 ở đầu ngăn xếp, vì vậy hãy truy cập nút 3 và lấy nó ra khỏi ngăn xếp và đặt tất cả các nút lân cận chưa được truy cập vào ngăn xếp.

Bây giờ, Ngăn xếp trở nên 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.



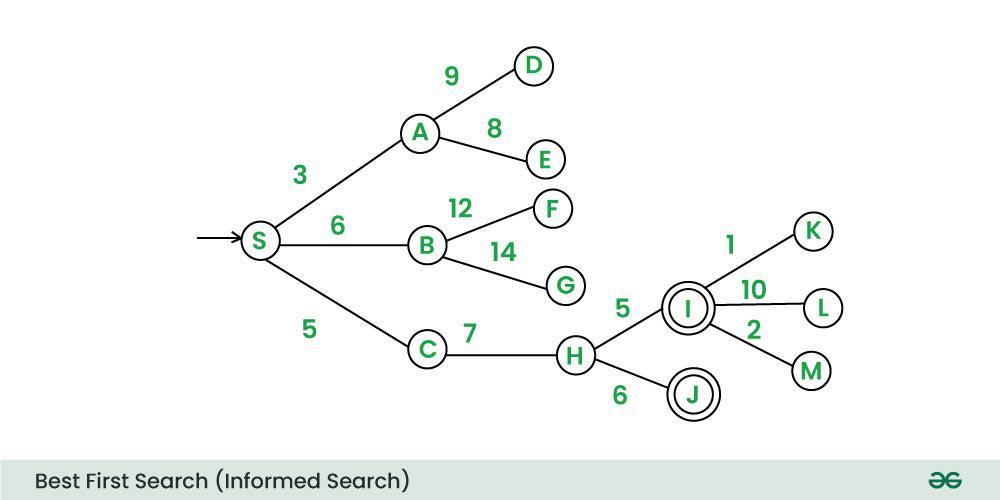
1. Best First Search.





Chúng tôi sử dụng hàng đợi ưu tiên hoặc đống để 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.

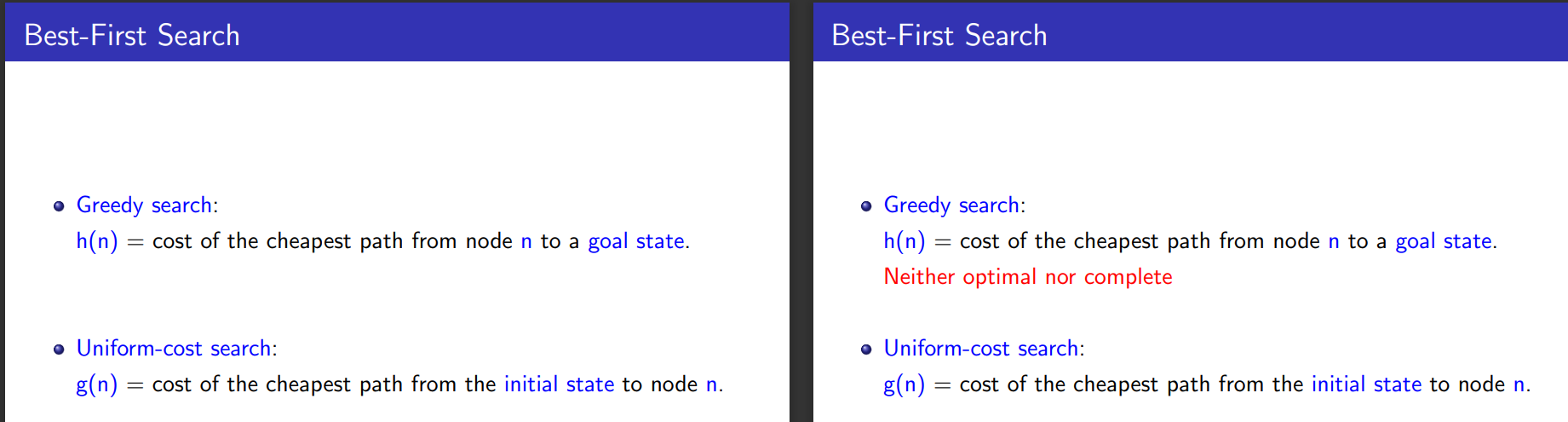
pq bây giờ chứa {B, H, E, D}

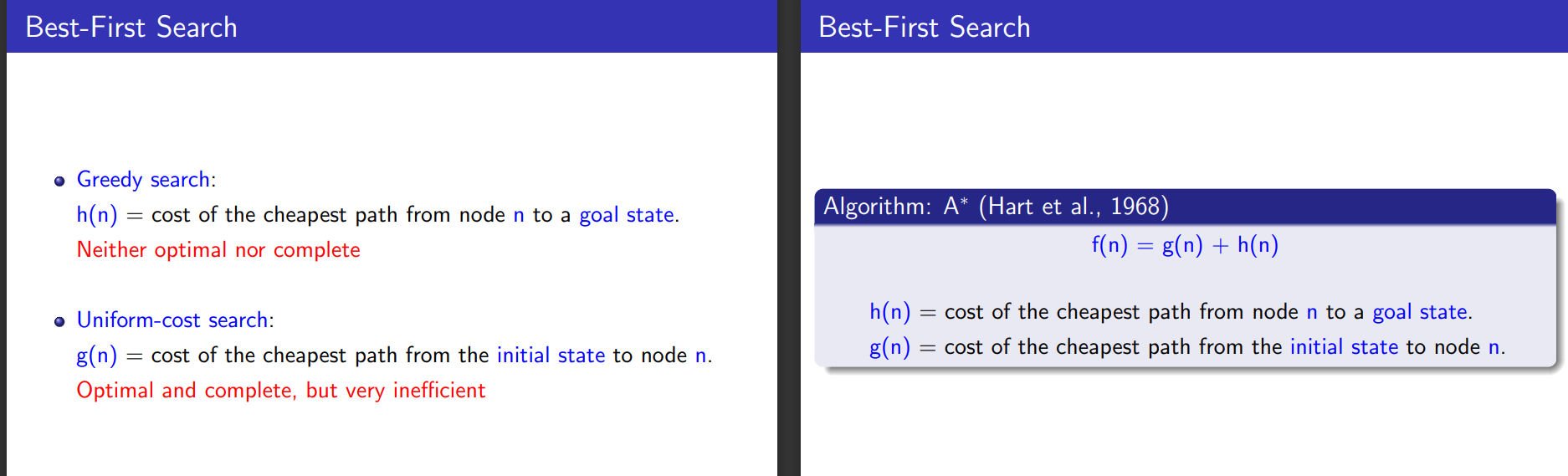
Chúng tôi loại bỏ B khỏi pq và xử lý các hàng xóm chưa được thăm của B thành pq.

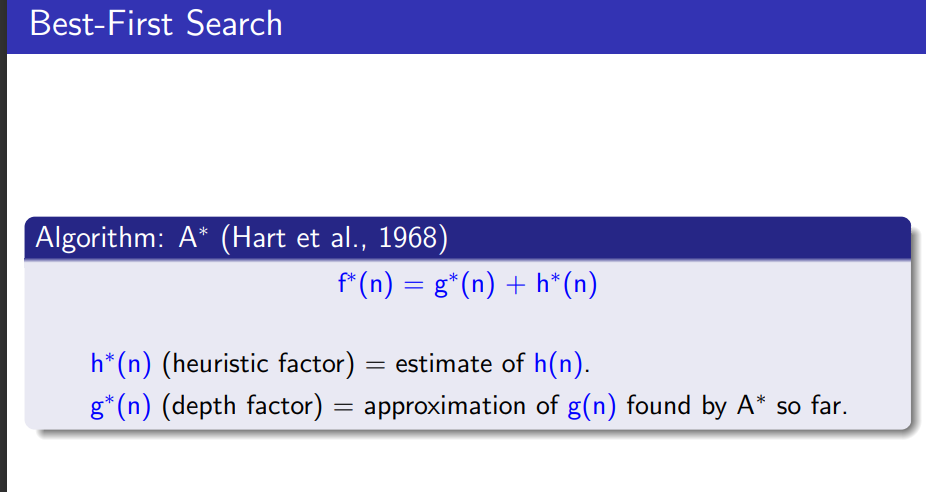
pq bây giờ chứa {H, E, D, F, G}

Chúng tôi loại bỏ H khỏi pq.

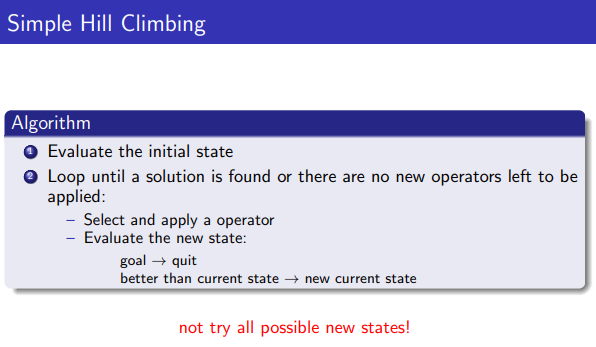
Vì mục tiêu “I” của chúng ta là hàng xóm của H nên chúng ta quay trở lại.



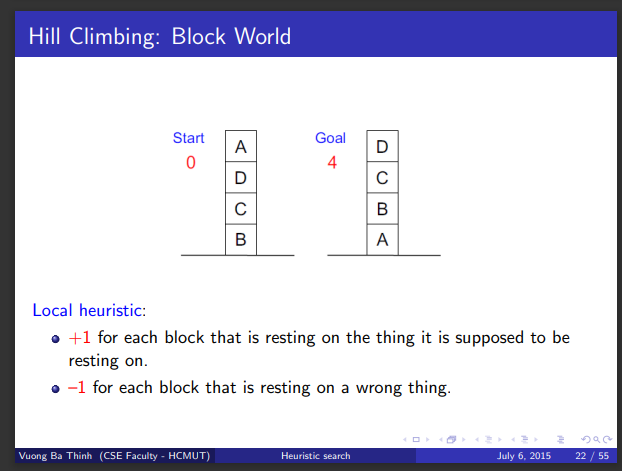


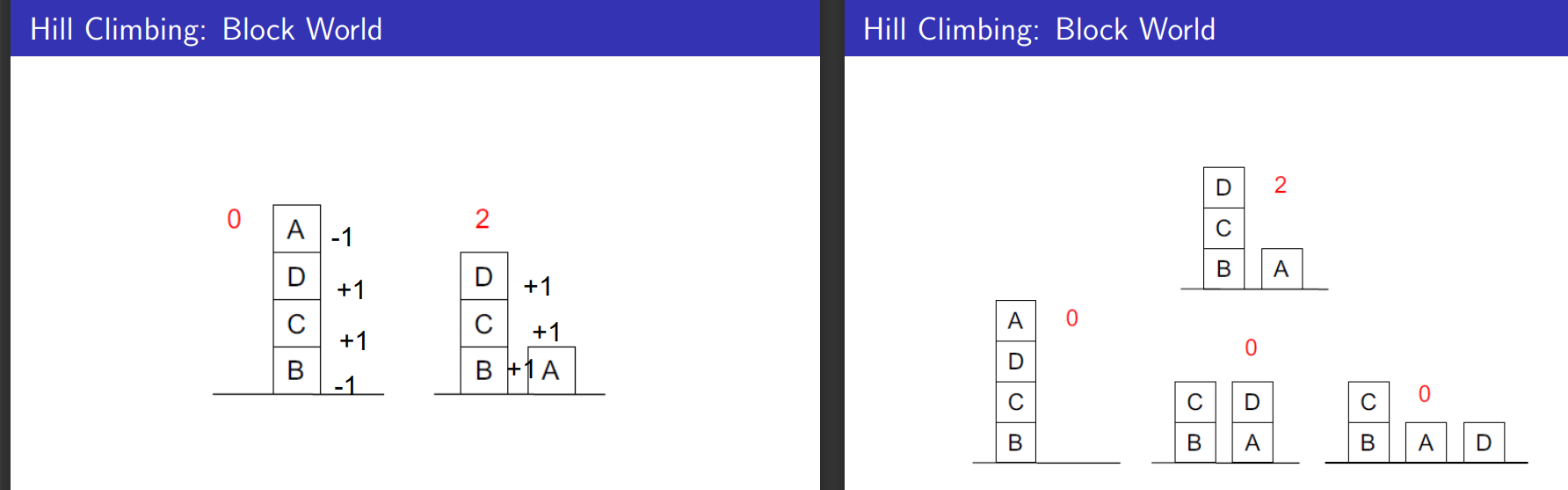


1. Heuristic
   1. Leo đồi: F -> đơn giản/ dốc nhất/ khắc phục.



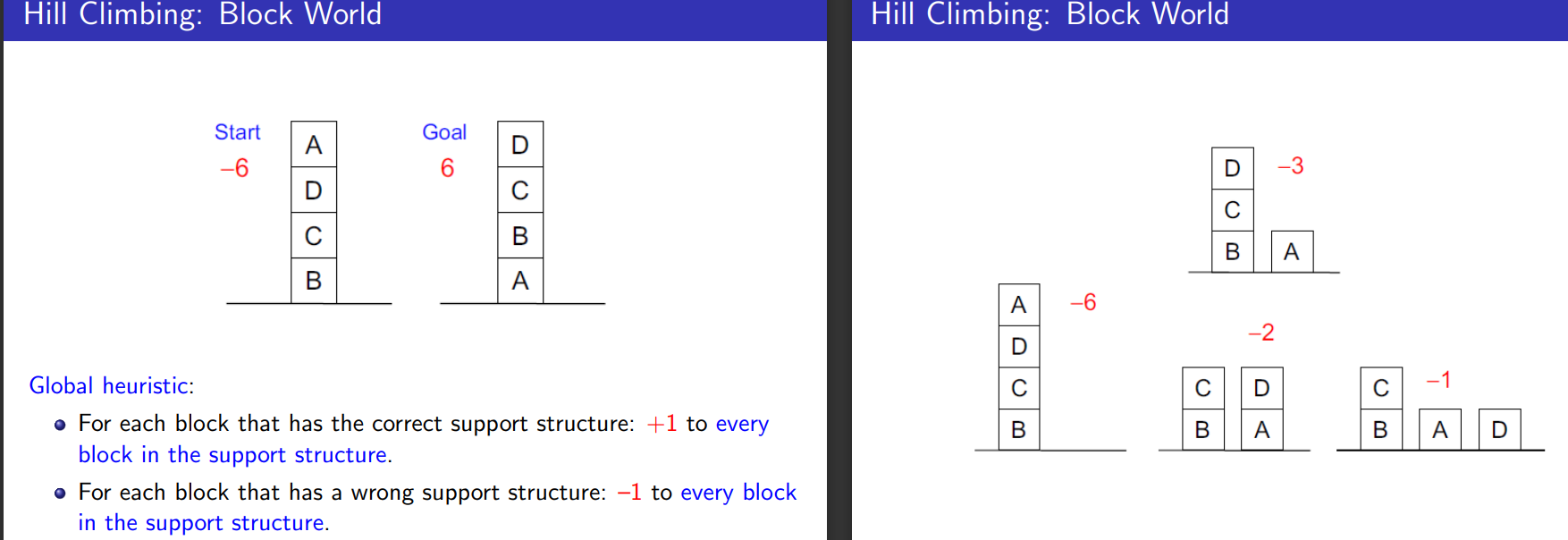


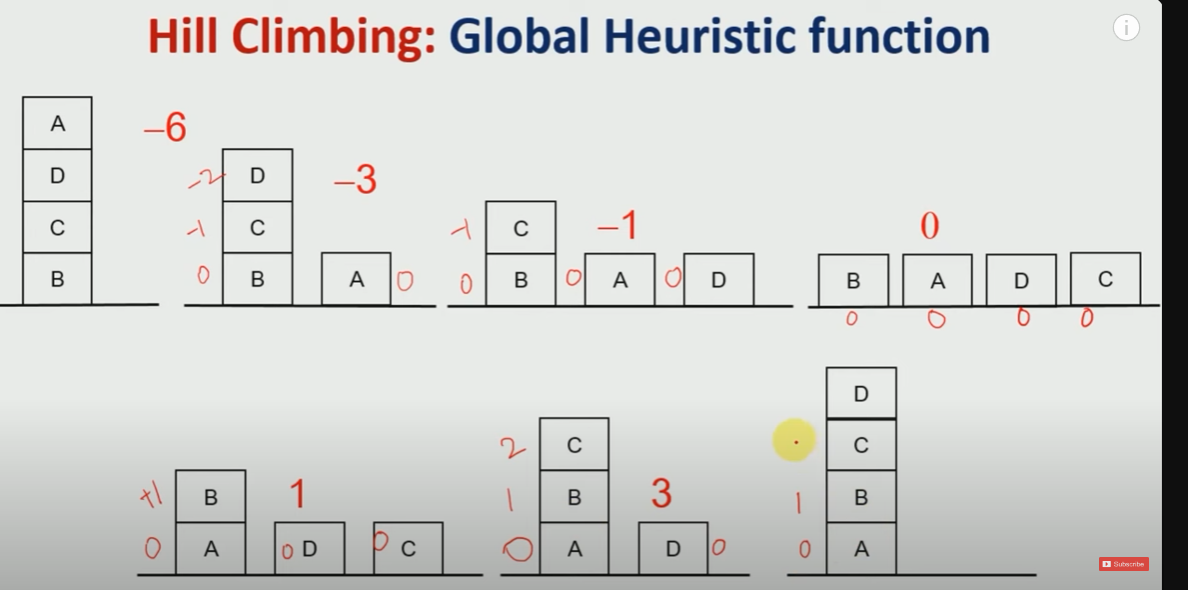




Cục bộ: Thêm một điểm cho mỗi khối nằm trên vật mà nó được cho là nằm trên đó. Trừ một điểm cho mỗi khối đặt sai vị trí.

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.





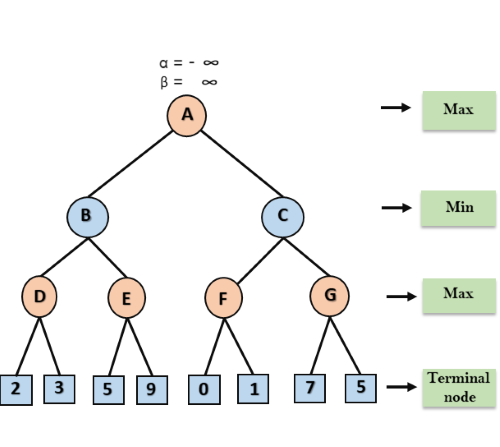
Dạng 2. Giải thuật Minmax.

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

Min -> Ngược lại với max.

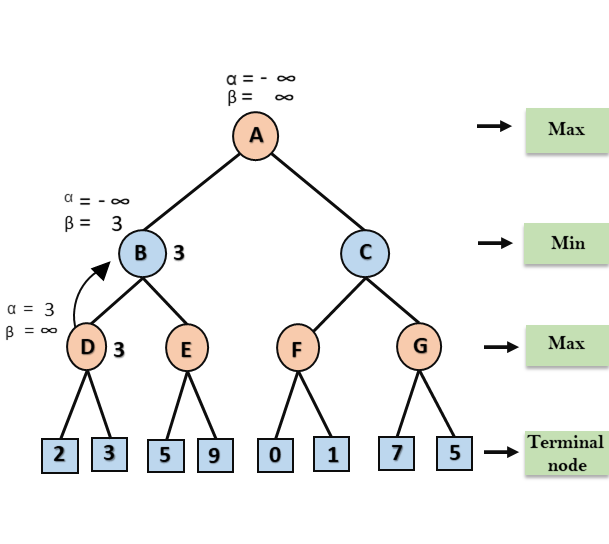
Dạng 3. Alpha-beta cutoff/pruning

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



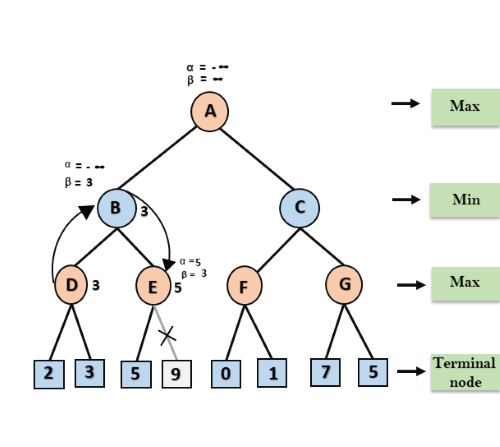
Bước 2: Tại Nút D, giá trị của α sẽ được tính lần lượt cho Max. Giá trị của α trước tiên được so sánh với 2 rồi đến 3, và giá trị tối đa (2, 3) = 3 sẽ là giá trị của α tại nút D và giá trị nút cũng sẽ là 3.

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 β= +∞, sẽ so sánh với giá trị các nút tiếp theo có sẵn, tức là min (∞, 3) = 3, do đó tại nút B bây giờ α= -∞ và β= 3.



Ở bước tiếp theo, thuật toán duyệt qua nút kế thừa tiếp theo của Nút B là nút E và các giá trị của α= -∞ và β= 3 cũng sẽ được chuyển.

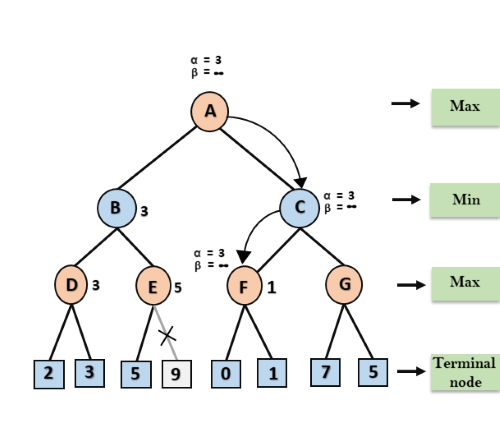
Bước 4: Tại nút E, Max sẽ đến lượt mình và giá trị của alpha sẽ thay đổi. Giá trị hiện tại của alpha sẽ được so sánh với 5 nên max (-∞, 5) = 5, do đó tại nút E α= 5 và β= 3, trong đó α>=β nên thừa kế bên phải của E sẽ bị lược bỏ, và thuật toán sẽ không duyệt qua nó và giá trị tại nút E sẽ là 5.



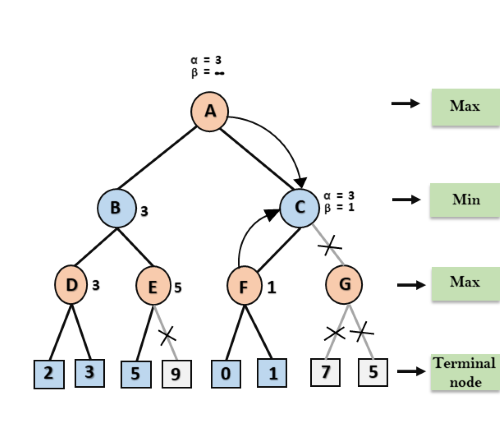
Bước 5: Ở bước tiếp theo, thuật toán lại dò ngược lại cây, từ nút B đến nút A. Tại nút A, giá trị của alpha sẽ được thay đổi, giá trị khả dụng tối đa là 3 khi max (-∞, 3)= 3 và β = +∞, hai giá trị này hiện được chuyển sang nút kế tiếp bên phải của A là Nút C.

Tại nút C, α=3 và β= +∞ và các giá trị tương tự sẽ được chuyển đến nút F.

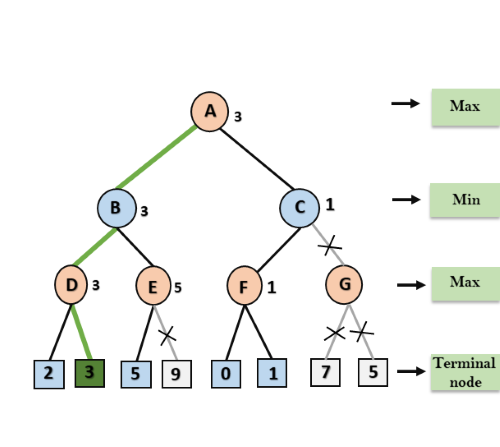
Bước 6: Tại nút F, một lần nữa giá trị của α sẽ được so sánh với con trái là 0 và max(3,0)= 3, sau đó so sánh với con phải là 1 và max(3,1)= 3 vẫn giữ nguyên là 3, nhưng giá trị nút của F sẽ trở thành 1.



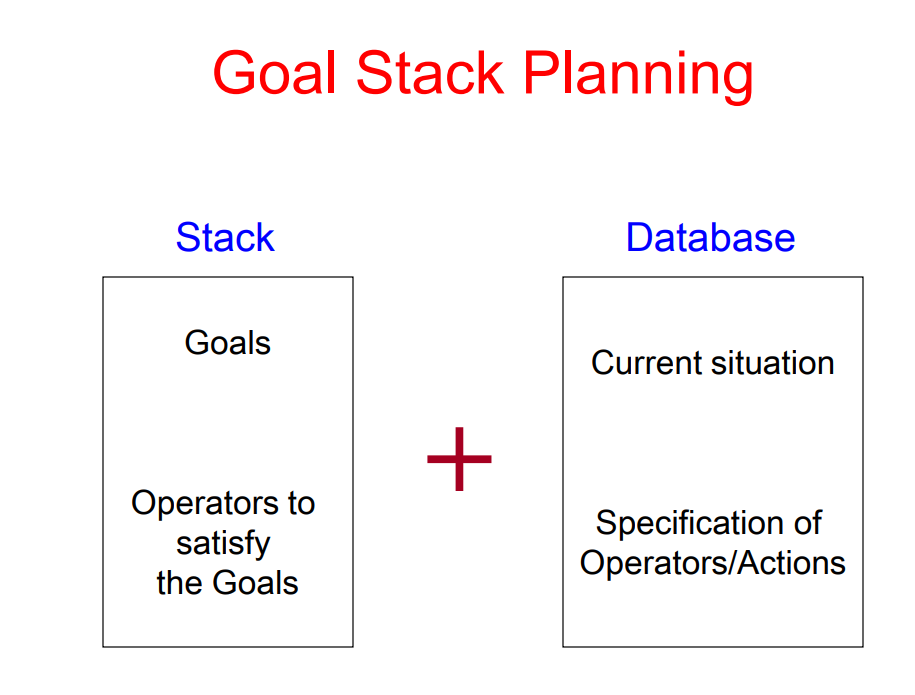
Bước 7: Nút F trả về giá trị nút 1 cho nút C, tại C α= 3 và β= +∞, tại đây giá trị beta sẽ được thay đổi sẽ so sánh với 1 nên min (∞, 1) = 1. Bây giờ tại C, α=3 và β= 1, đồng thời thỏa mãn điều kiện α>=β nên cây con tiếp theo của C là G sẽ bị lược bỏ và thuật toán sẽ không tính toán toàn bộ cây con G.



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.



1. **Goal Stack Planning**

****

**0**

Lập kế hoạch ngăn xếp mục tiêu

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

– Nếu đỉnh ngăn xếp là mục tiêu phức hợp, hãy đẩy các mục tiêu phụ chưa thỏa mãn của nó vào ngăn xếp.

– Nếu đỉnh 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 đỉnh 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

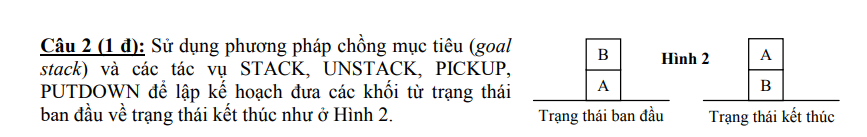
STACK(X,Y): Chồng X lên Y. (Trên Y không được có

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

PICKUP (X): Lấy X lên

PUTD(X): Bỏ X xuống.





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)  ONT(B)  ON(A,B)  goal | ARM(E)  CL(B)  ON(B,A)  ONT(A) |

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

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

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

Bước 3. Sau khi PUTD(B), ta có.

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

1. **Knowledge Base.**
2. **Logic vị từ (Slide - Ko kịp)**

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) ghetCaesar(x) ∨ trungthanhCaesar(x)

(2) ┐trungthanhCaesar(x) ∨┐ amsatCaesar(x)

(3) amsatCaesar(Marcus)

(4) trungthanhCaesar(Anthony)

(b) S**ử dụng phương pháp phản chứng-phân giải để:**

- Chứng minh Anthony không muốn ám sát Caesar. (0.5 đ)

- Tìm xem ai ghét Caesar.

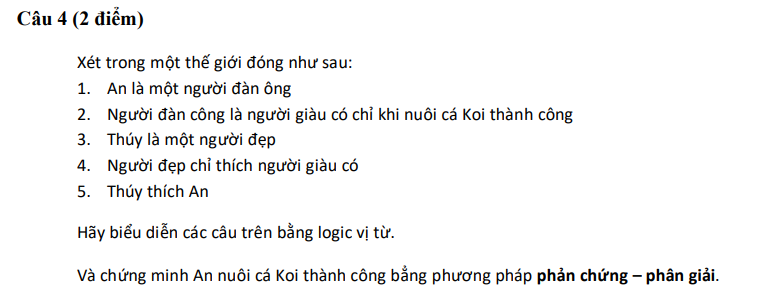
b1. α = ┐ amsatCaesar(Anthony) ┐α = amsatCaesar(Anthony) (5)

Kết hợp (2) và (5) : ┐trungthanhCaesar(Anthony) (6) với {Anthony/x}

Kết hợp (6) và (4) : đpcm

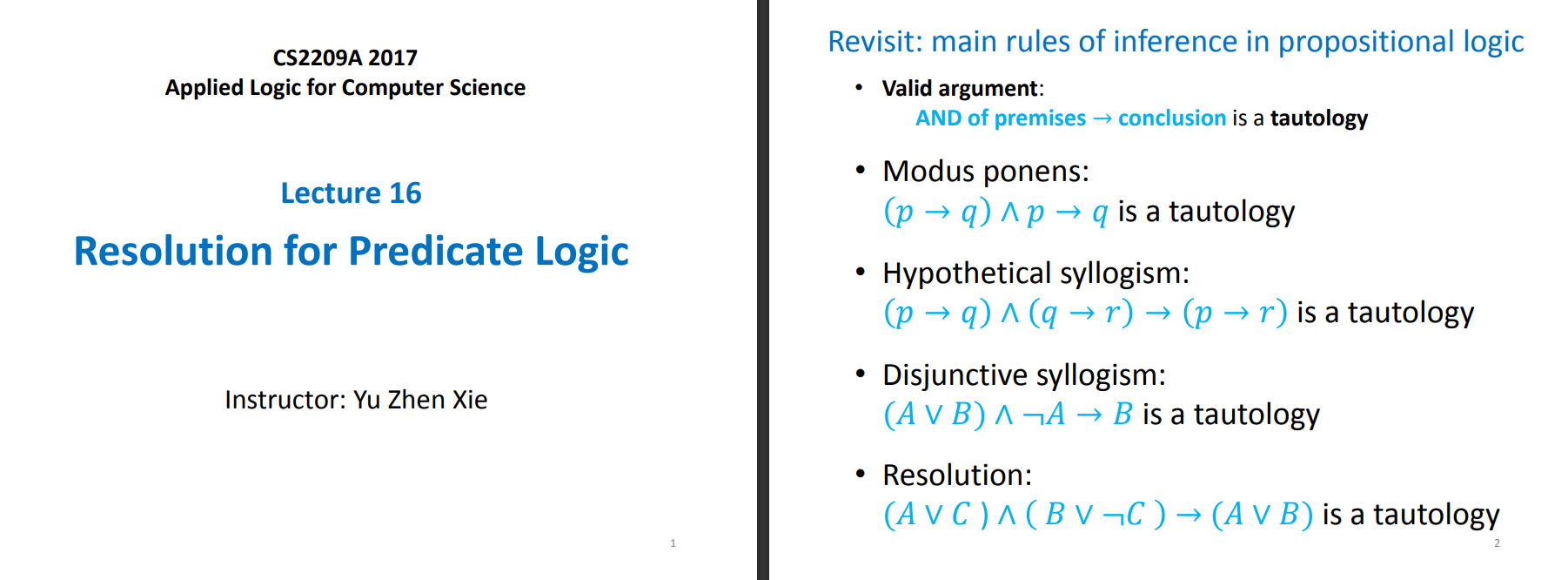
b2. α = ghetCaesar(y) ┐α = ┐ ghetCaesar(y)

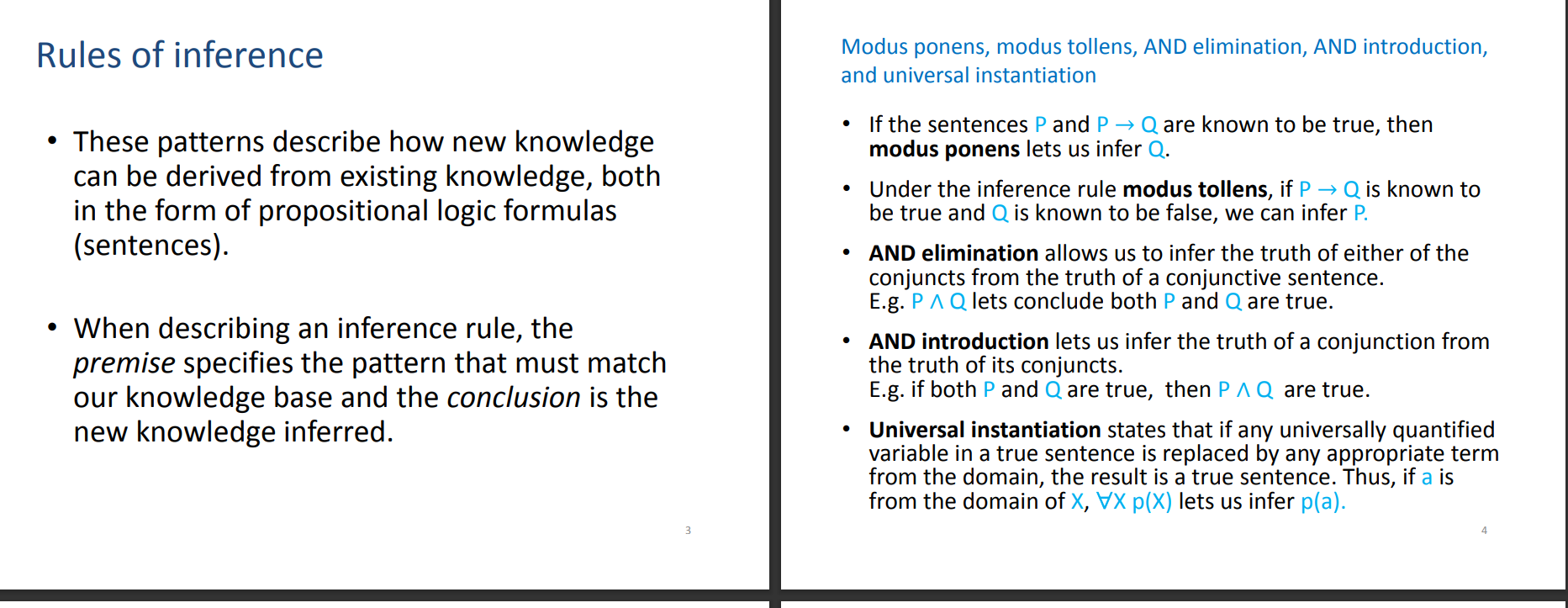
Dùng phản chứng chúng ta chỉ tìm được {Marcus/y}

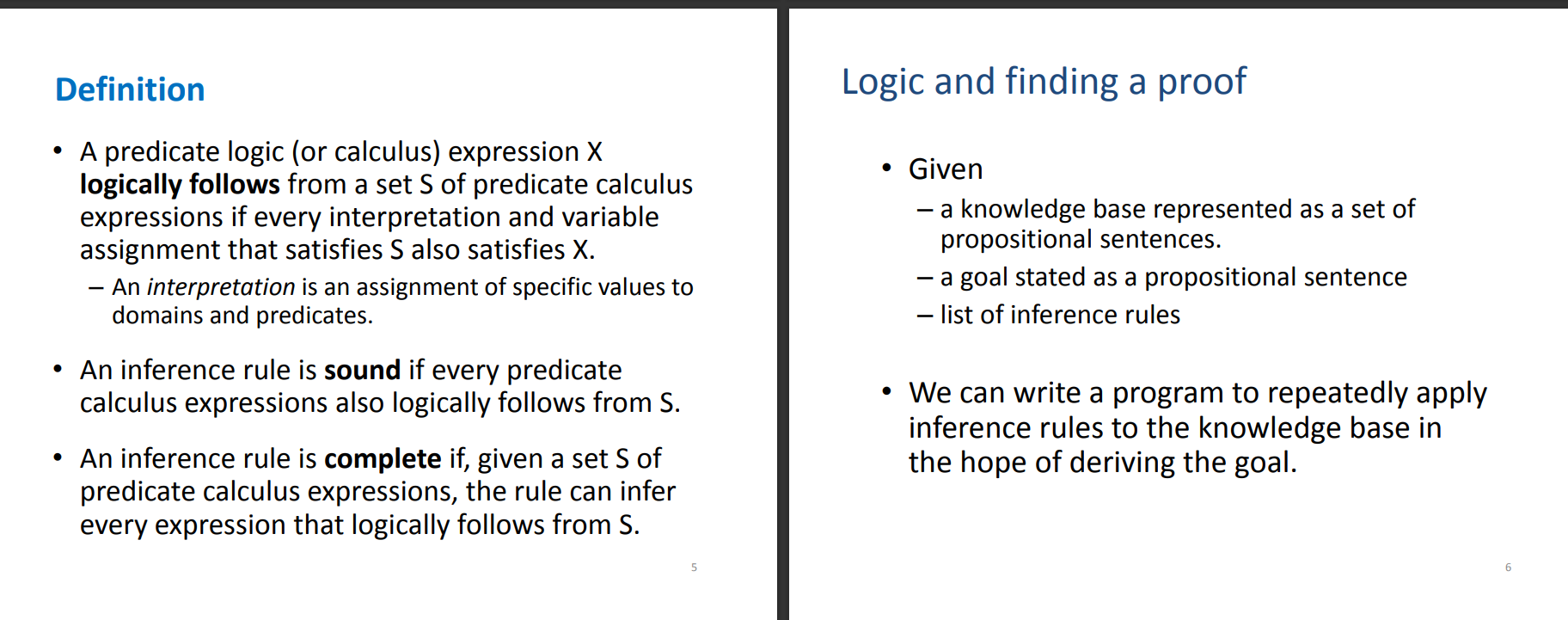
****

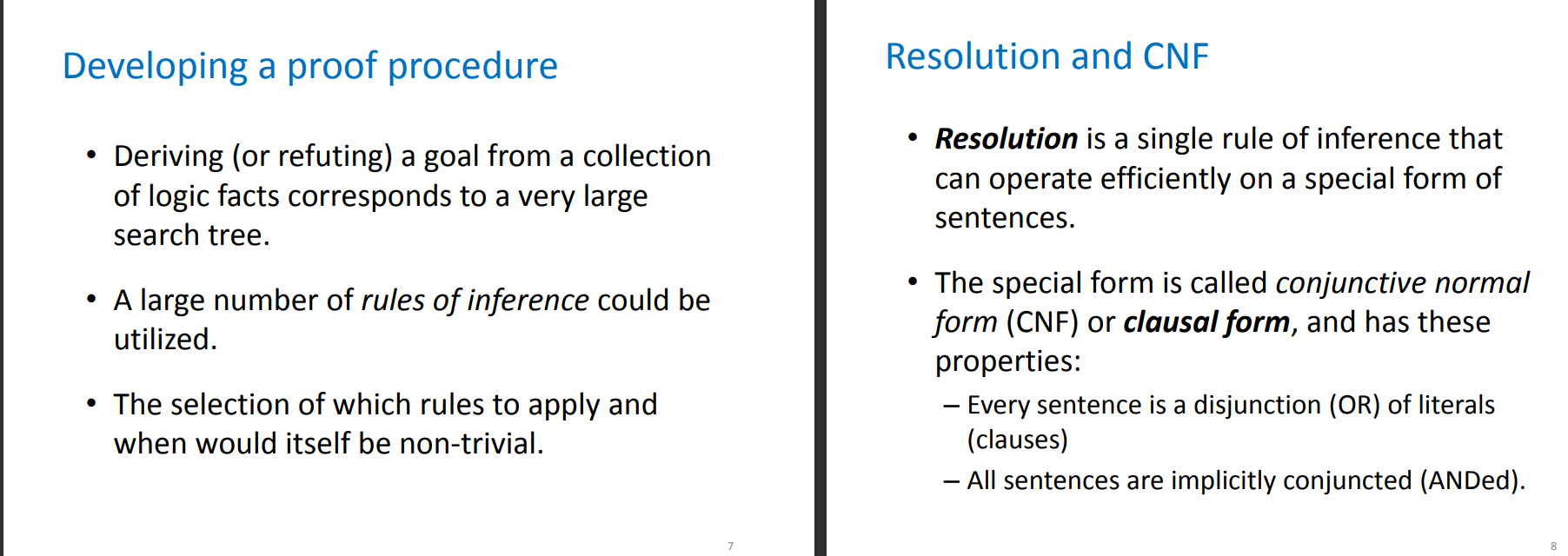
**3** + 4 -> 6: Thúy thích người giàu có 6 + 5 -> 7: An là người giàu có 1 + 7 -> 8: An là người đàn ông giàu có 8 + 2: An nuôi cá Koi thành công (đpcm) Tuy nhiên sinh viên phải biểu diễn bằng phương pháp phản chứng – phân giải để đi đến được kết luận trên

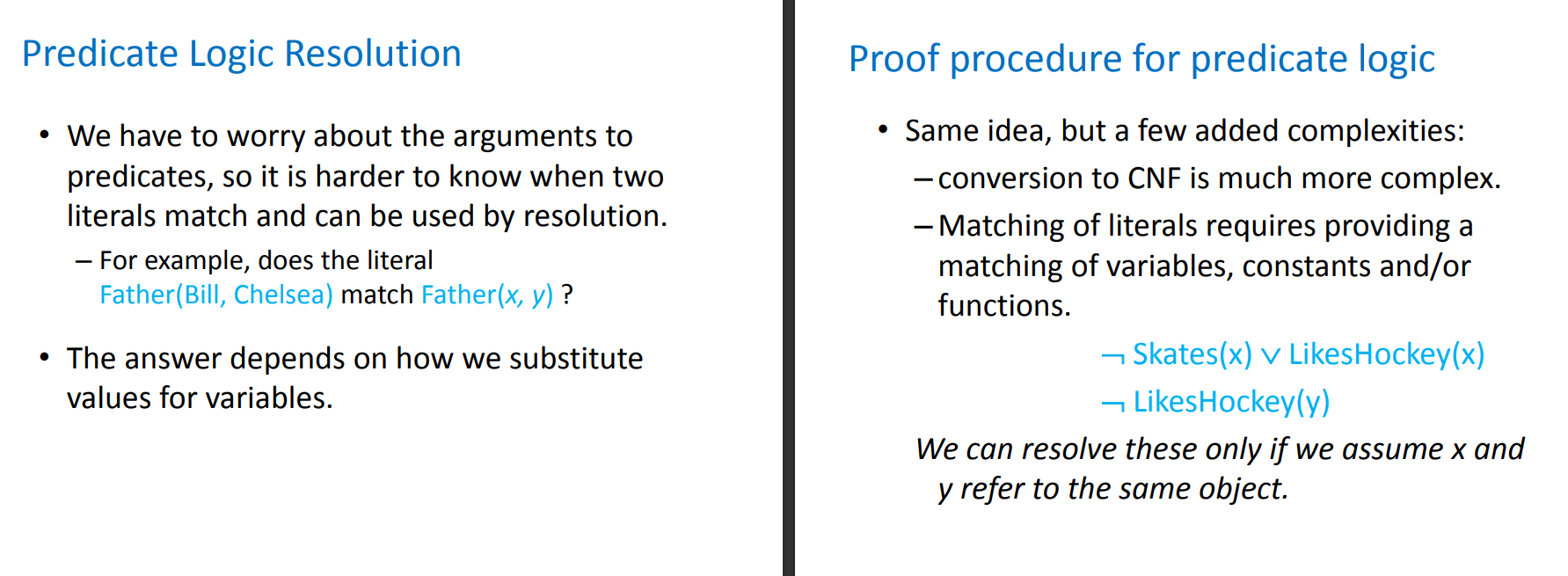
1. **Chứng minh phản chứng – phân giải. (Slide - ko Kịp\_**

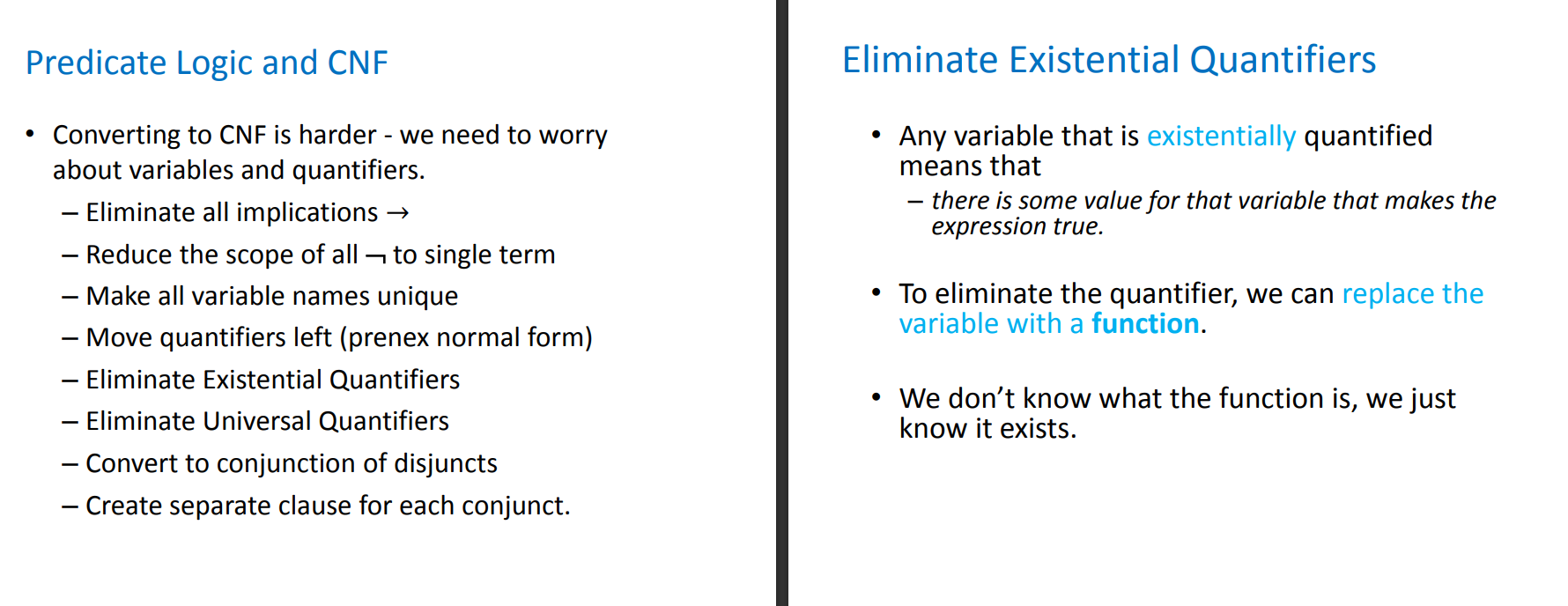
****

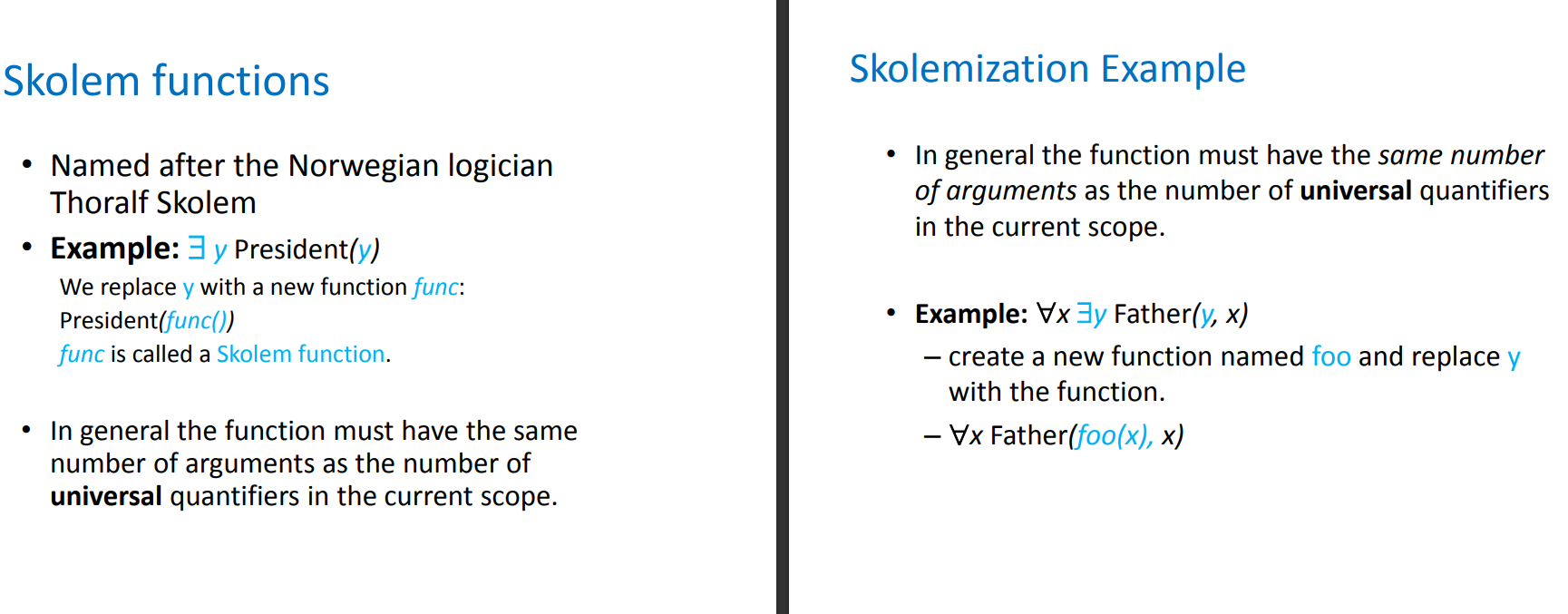
****

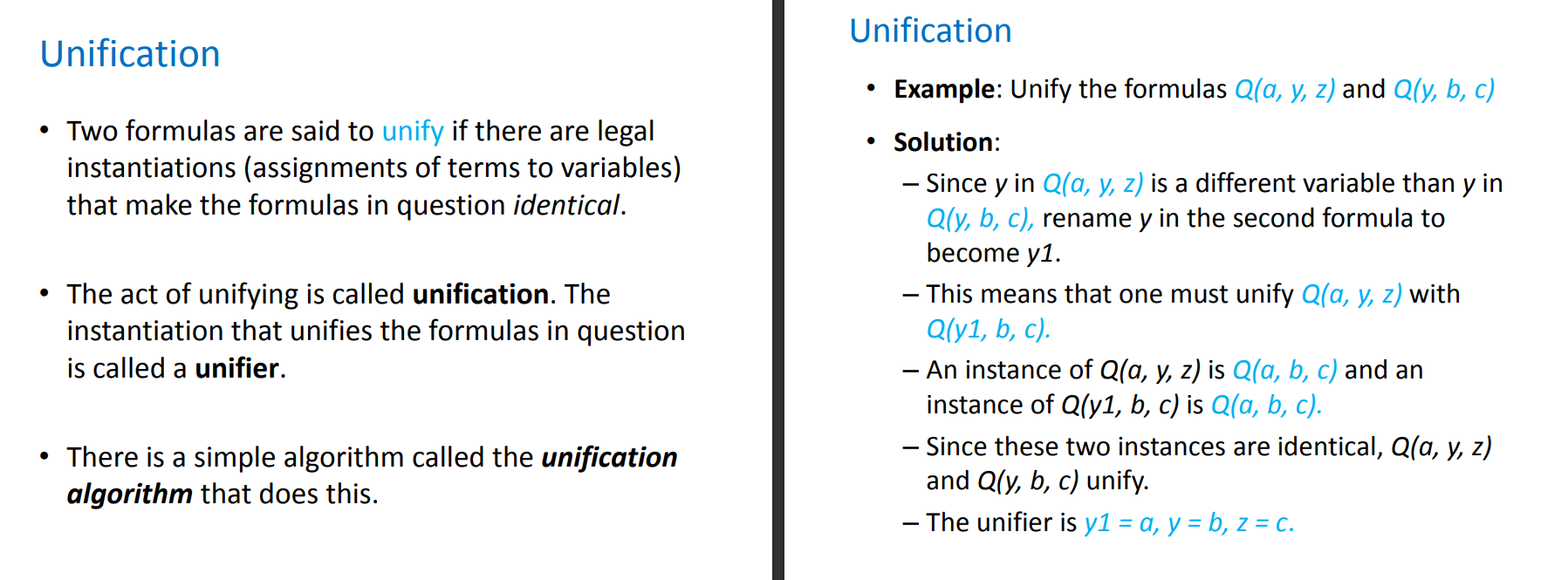
****

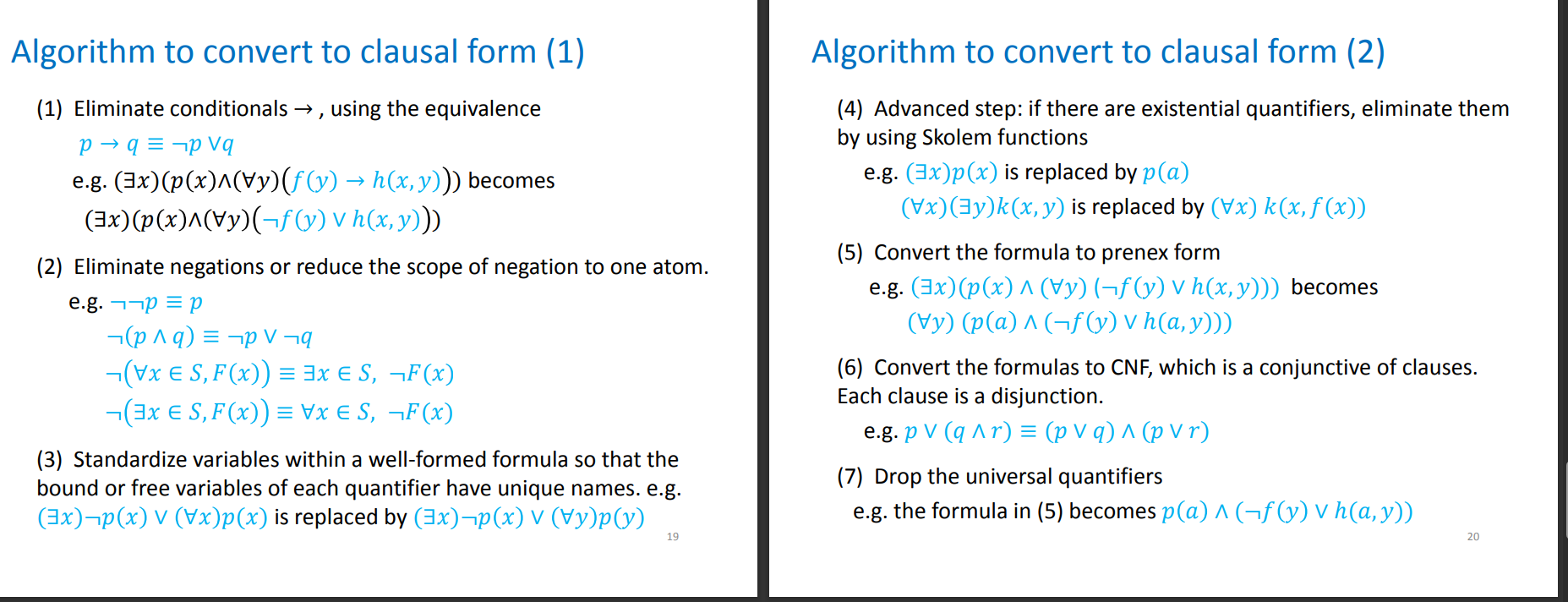
****

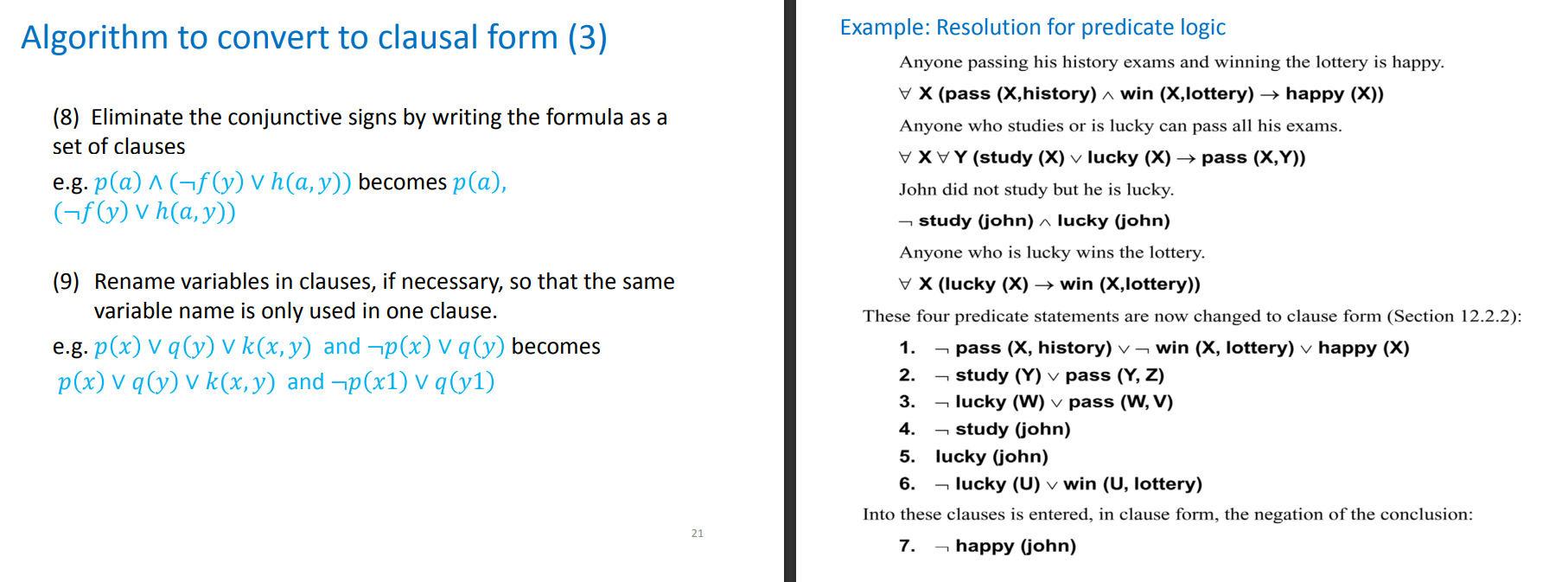
****

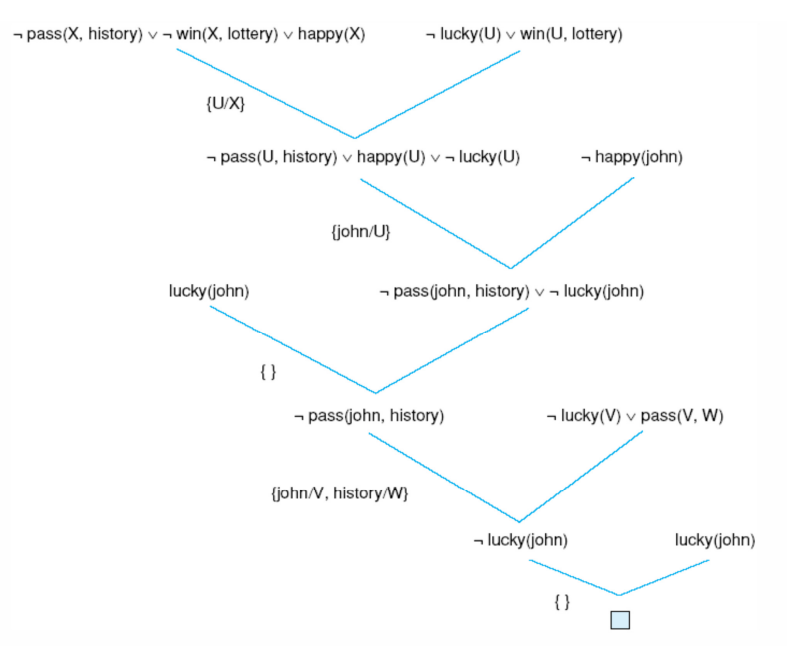
****

****

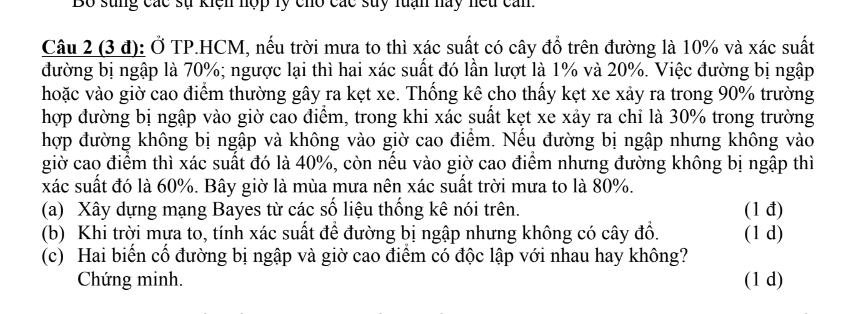
****

****

****

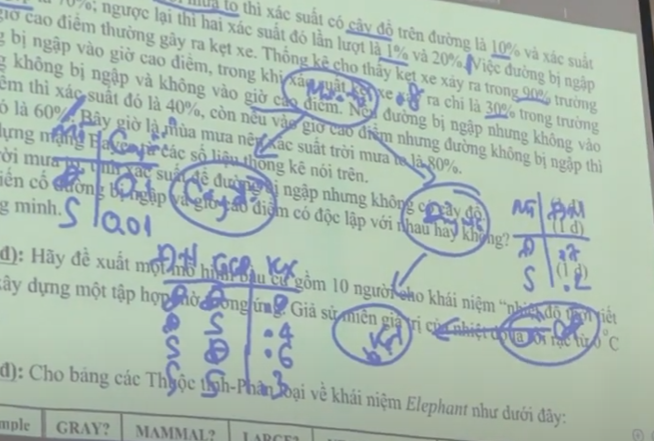
****

1. **Graph. (Slide - Ko kịp)**
2. **Xác suất.**

****

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.



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(ĐN, -CĐ | MT) = P(ĐN,-CĐ, MT) / P(MT)

Biến cố cha của đường ngập và cây đổ là mưa to

= P(ĐN | MT) \* P(-CĐ | MT) \* P(MT) / P(MT)

= 0.7 \* (1 - P(CĐ|MT)) = 0.7 \* 0.9 = 0.63

c) P(GCD) = P(GCD | ĐN)

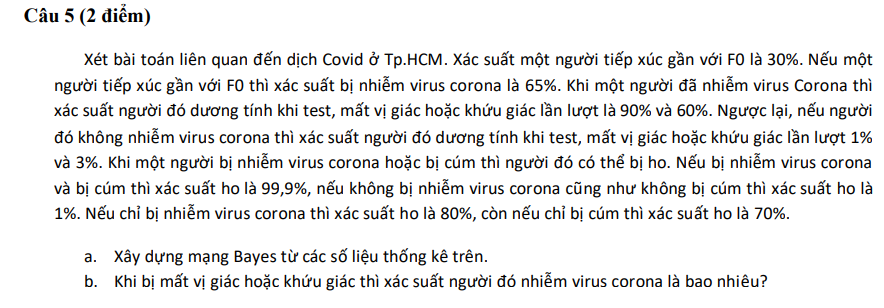
P(GCD|DN) = P(GCD, DN) / P(DN) = ( P(GCD, DN, MT) + P (GCD, DN, -MT) )/ P(DN)

= [P(GCD) \* P(DN,MT) + P(GCD) \* P(DN,-MT)] / P(DN)

= P(GCD) \* (P(DN,MT) + P(DN,-MT) ) / P(DN)

= P(GCD) \* P(DN) / P(DN)

= P(GCD) (dpcm)



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.

1. **Học máy**
2. **Bayesian Network (Xác suất/Định lý Bayes).**
3. **Decision Tree**

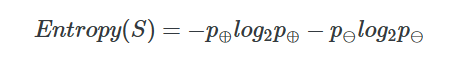
Câu b) Vẽ cây quyết định.



Chọn nút gốc của cây quyết định:

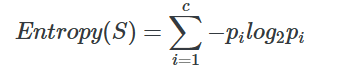
Tập dữ liệu hiện tại có kết 9 quả Yes và 5 kết quả No, ta kí hiệu là S: [9+,5−]

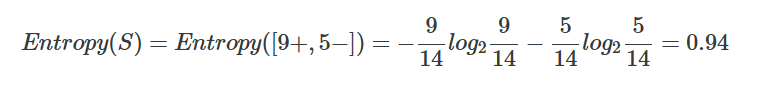
Theo công thức tính Entropy (độ hỗn tạp dữ liệu) của một tập:



trong đó:

* p⊕ là tỷ lệ các mẫu thuộc lớp dương trong S.
* p⊖ là tỷ lệ các mẫu thuộc lớp âm trong S.

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

Vậy ta có:

Lưu ý:

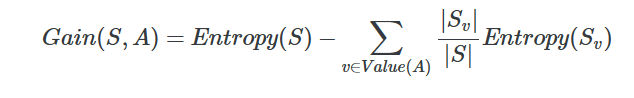
* Entropy là 0 nếu tất cả các thành viên của S đều thuộc về cùng một lớp.
* Entropy là 1 nếu tập hợp chứa số lượng bằng nhau các thành viên thuộc lớp âm và dương.

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).
* 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:

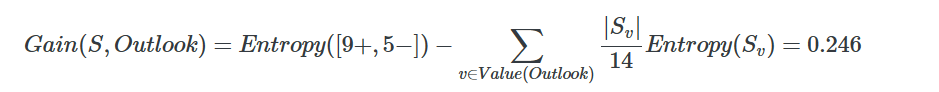


trong đó:

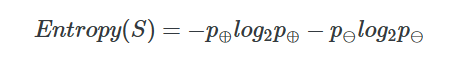
* Value(A) là tập các giá trị có thể cho thuộc tính A.
* Sv là tập con của S mà A nhận giá trị v.

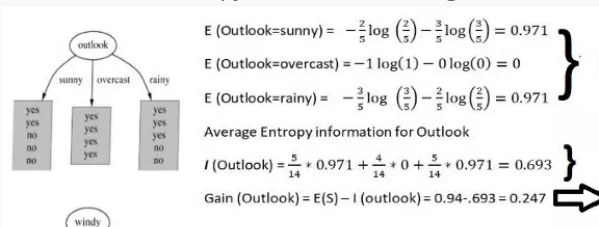
*Lấy ví dụ với thuộc tính A = Outlook, ta có Value(A) = {Sunny, Overcast, Rain}, và SSunny =* [2+,3−] *như đã tính ở trên*

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



Giải thích con số 0.246.

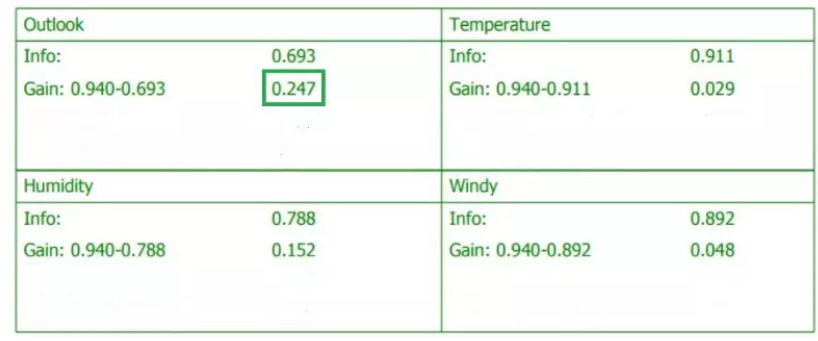
Với E(Outlook=Sunny), ta áp dụng công thức tính Entropy ở trên cho trường hợp Sunny = False và Sunny = True. Ở đây, ta nhận thấy có 2 True và 3 False. Áp dụng tương tự với các trường hợp Overcast, Rainy.



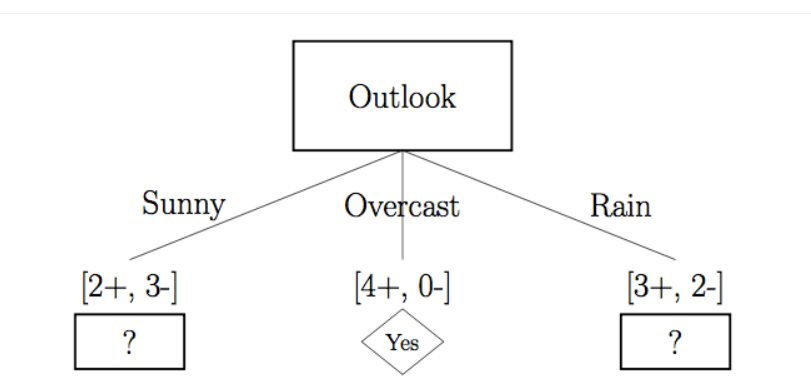
Sau đó, ta tính I(OutLook) = tổng ([số lượng phân tử của trường hợp / tổng số lượng mẫu] \* entropy của trường hợp đó) .

Ta lấy Gain(S, OutLook) = Entropy(S) - I(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



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, khỏi tính cho mệt.

**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 SSunny 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 SSunny = [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 SSunny).

Xét thuộc tính Humidity, có: SNormal: = [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).

SHigh: [0,3−]. Kết quả tương tự vậy.

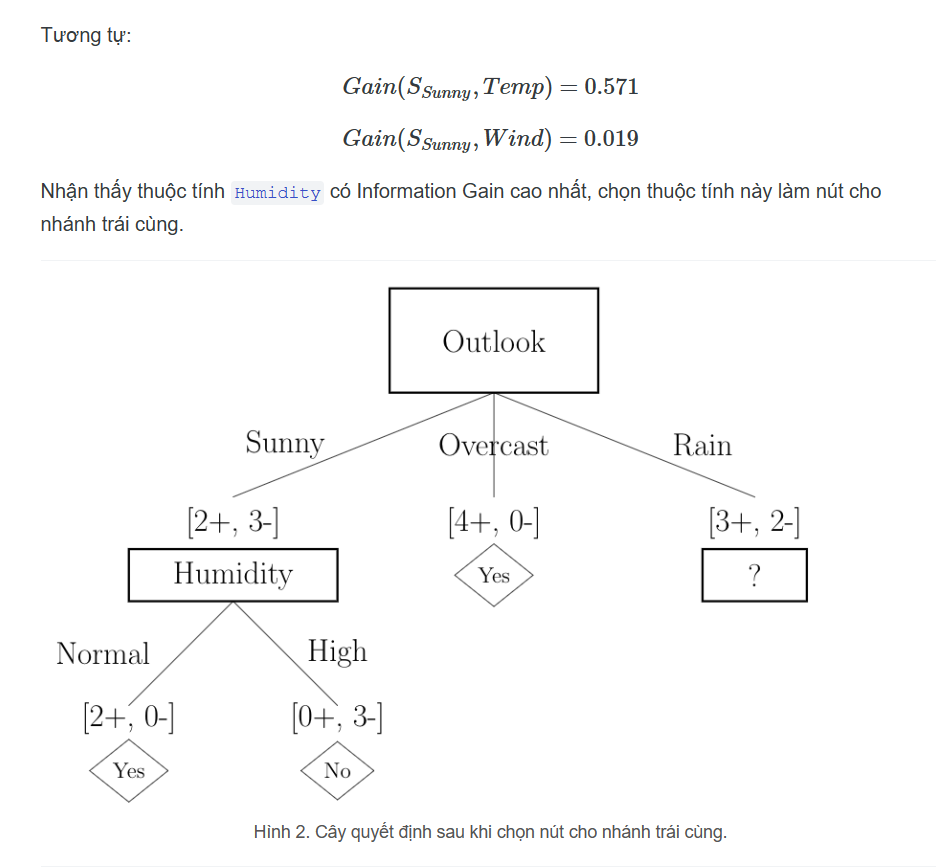
Từ đó:

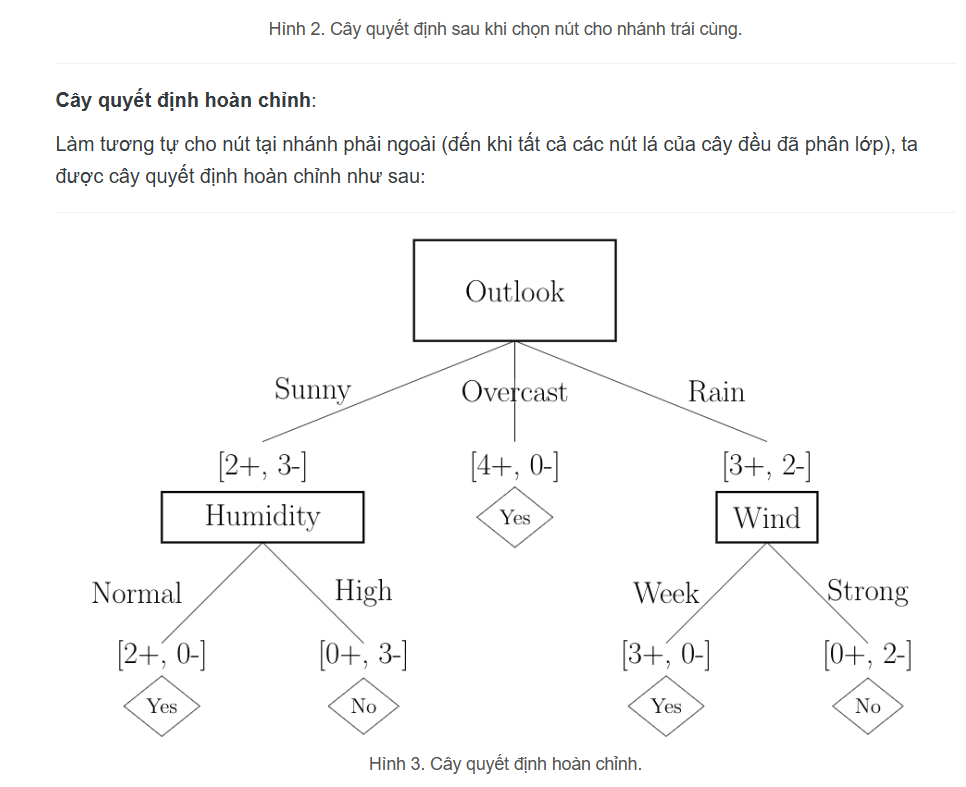
Ta có: E(Humidity = Normal |Outlook = Sunny) = - 2/2 \* log\_2(2/2) = 0.

E(Huminity = High | Outlook = Sunny) = = 3/3 \* log\_2(3/3) = 0

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

Gain(S\_Sunny, Humindity) = Entropy(2+,3-) -I(Huminity | Outlook = Sunny) = 0.971.





b) Naive Bayes

* Bước 1. Vẽ mạng bayes.

