Chương 3: Các chiến lược tìm kiếm Heuristics

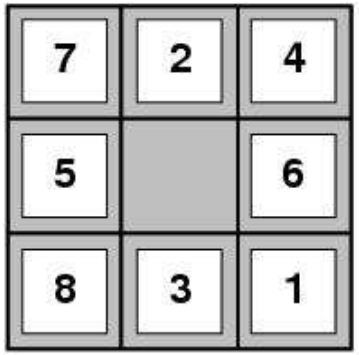
Nội dung

- Khái niệm
- Tìm kiếm tốt nhất trước
- Phương pháp leo đồi
- Cài đặt hàm đánh giá
- Thu giảm ràng buộc
- Giải thuật cắt tỉa α-β

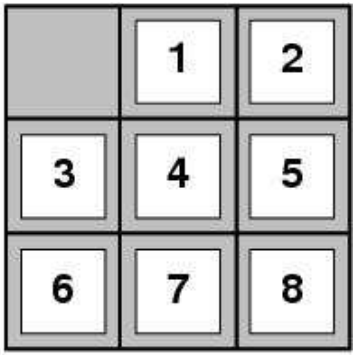
Giới hạn không gian hệ thống

- 8-puzzle
  - Lời giải cần trung bình 22 cấp (depth)
  - Độ rộng của bước ~ 3
  - Tìm kiếm vét cạn cho 22 cấp cần
    - 3.1 x 10^{10} states
  - Nếu chỉ giới hạn ở d=12, cần trung bình 3.6 triệu trạng thái

    [24 puzzle có 10^{24} trạng thái]
- $\Rightarrow$ Cần chiến lược tìm kiếm heuristic



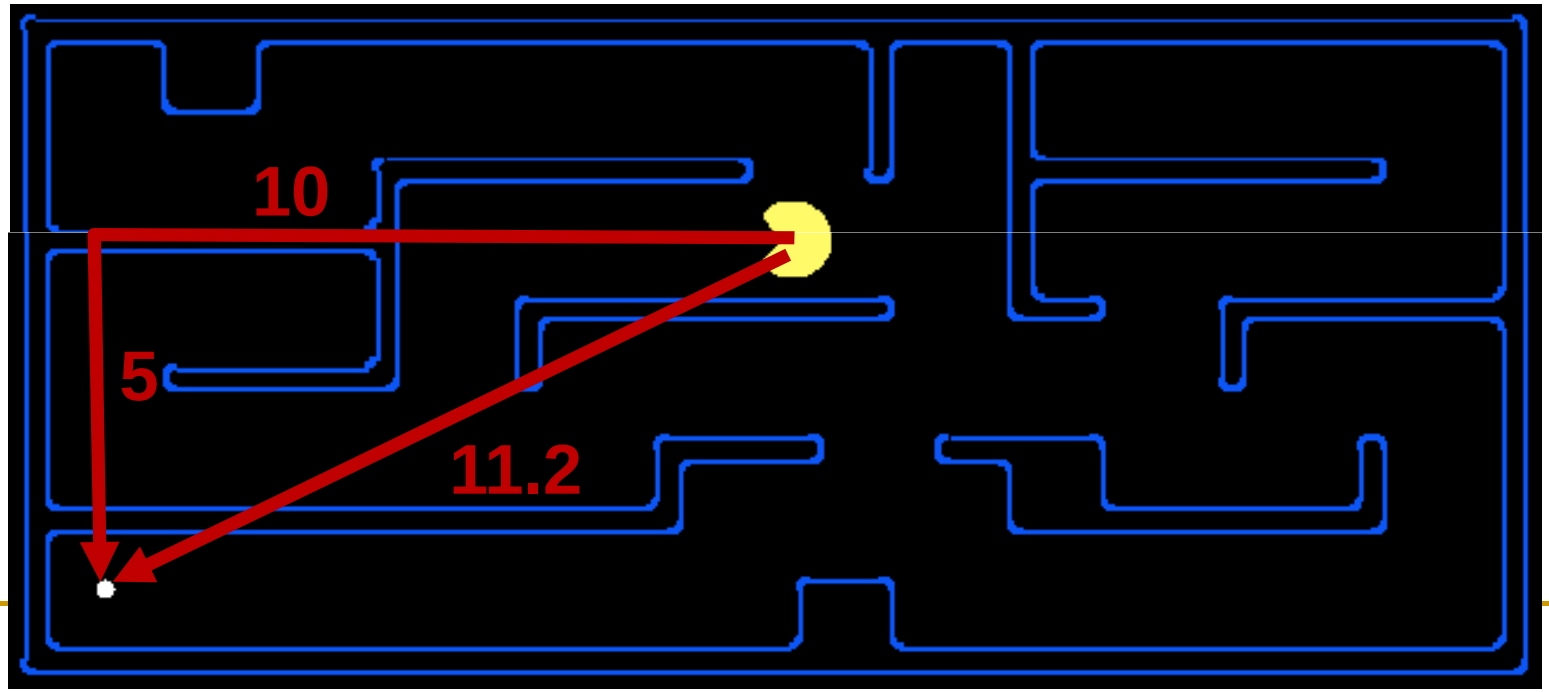
Start State



Goal State

Tìm kiếm Heuristics

- Any *estimate* of how close a state is to a goal
- Designed for a particular search problem
- Examples: Manhattan distance, Euclidean distance



Tìm kiếm Heuristic (tt)

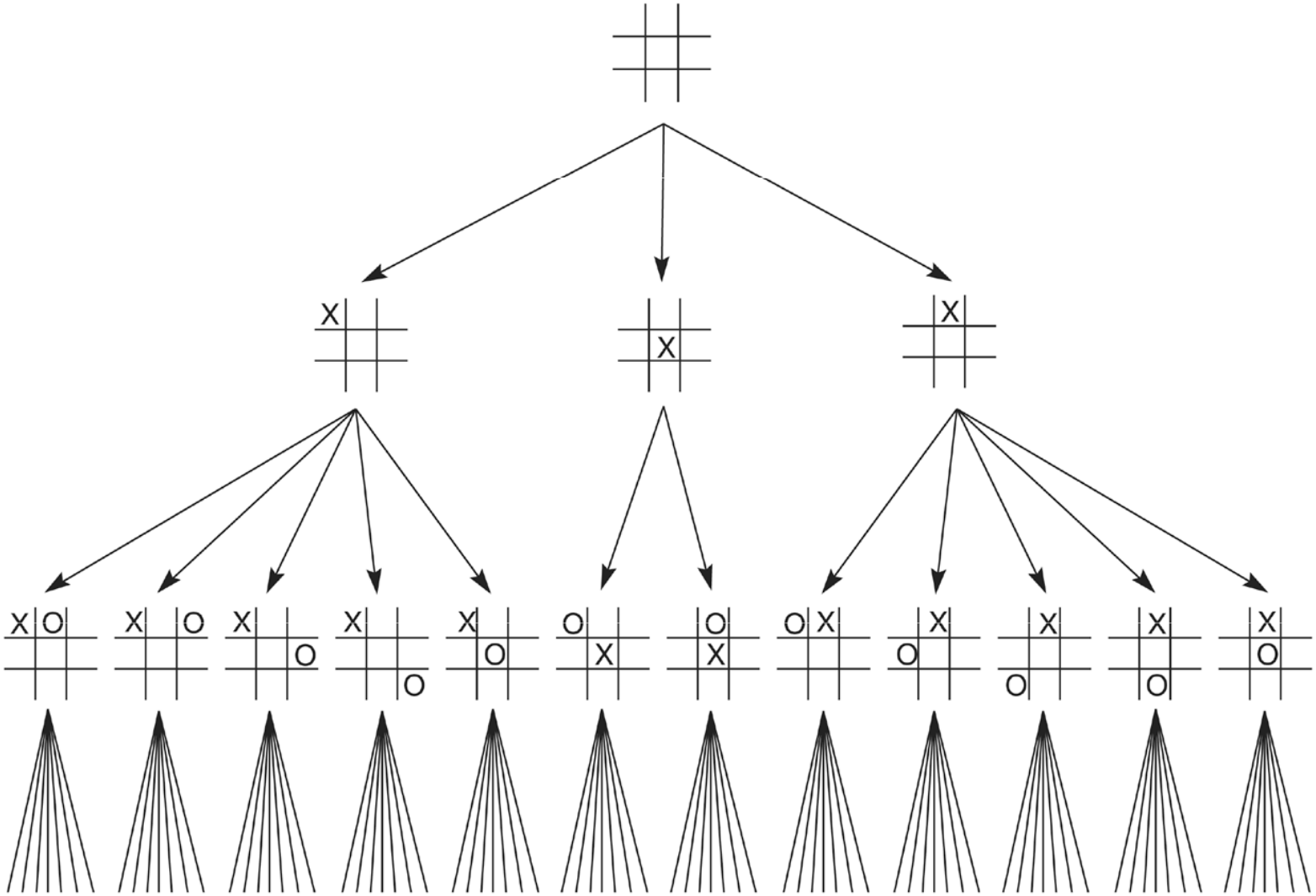
- Có nhiều phương pháp để xây dựng một thuật giải Heuristic, trong đó người ta thường dựa vào một số nguyên lý cơ bản như sau:
  - **Nguyên lý vét cạn thông minh:** Trong một bài toán tìm kiếm nào đó, khi không gian tìm kiếm lớn, ta thường tìm cách giới hạn lại không gian tìm kiếm hoặc thực hiện một kiểu dò tìm đặc biệt dựa vào đặc thù của bài toán để nhanh chóng tìm ra mục tiêu.
  - **Nguyên lý tham lam (Greedy):** Lấy tiêu chuẩn tối ưu (trên phạm vi toàn cục) của bài toán để làm tiêu chuẩn chọn lựa hành động cho phạm vi cục bộ của từng bước (hay từng giai đoạn) trong quá trình tìm kiếm lời giải.

Tìm kiếm Heuristic (tt)

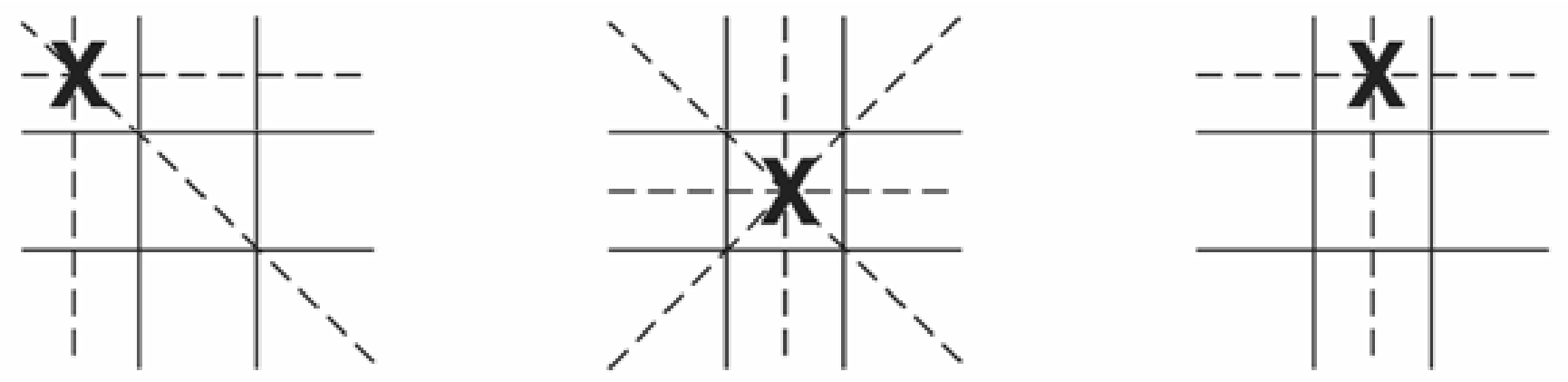
- **Nguyên lý thứ tự:** Thực hiện hành động dựa trên một cấu trúc thứ tự hợp lý của không gian khảo sát nhằm nhanh chóng đạt được một lời giải tốt.
- **Hàm Heuristic:** các hàm đánh giá thô, giá trị của hàm phụ thuộc vào trạng thái hiện tại của bài toán tại mỗi bước giải, giúp chọn được cách hành động tương đối hợp lý trong từng bước của thuật giải.

- Giải thuật tìm kiếm heuristic:
  - *Giải thuật leo núi (hill-climbing)*
  - *Greedy, TK tốt nhất (best-first search)*
  - *A* search, …*

Ví dụ phép đo Heuristics

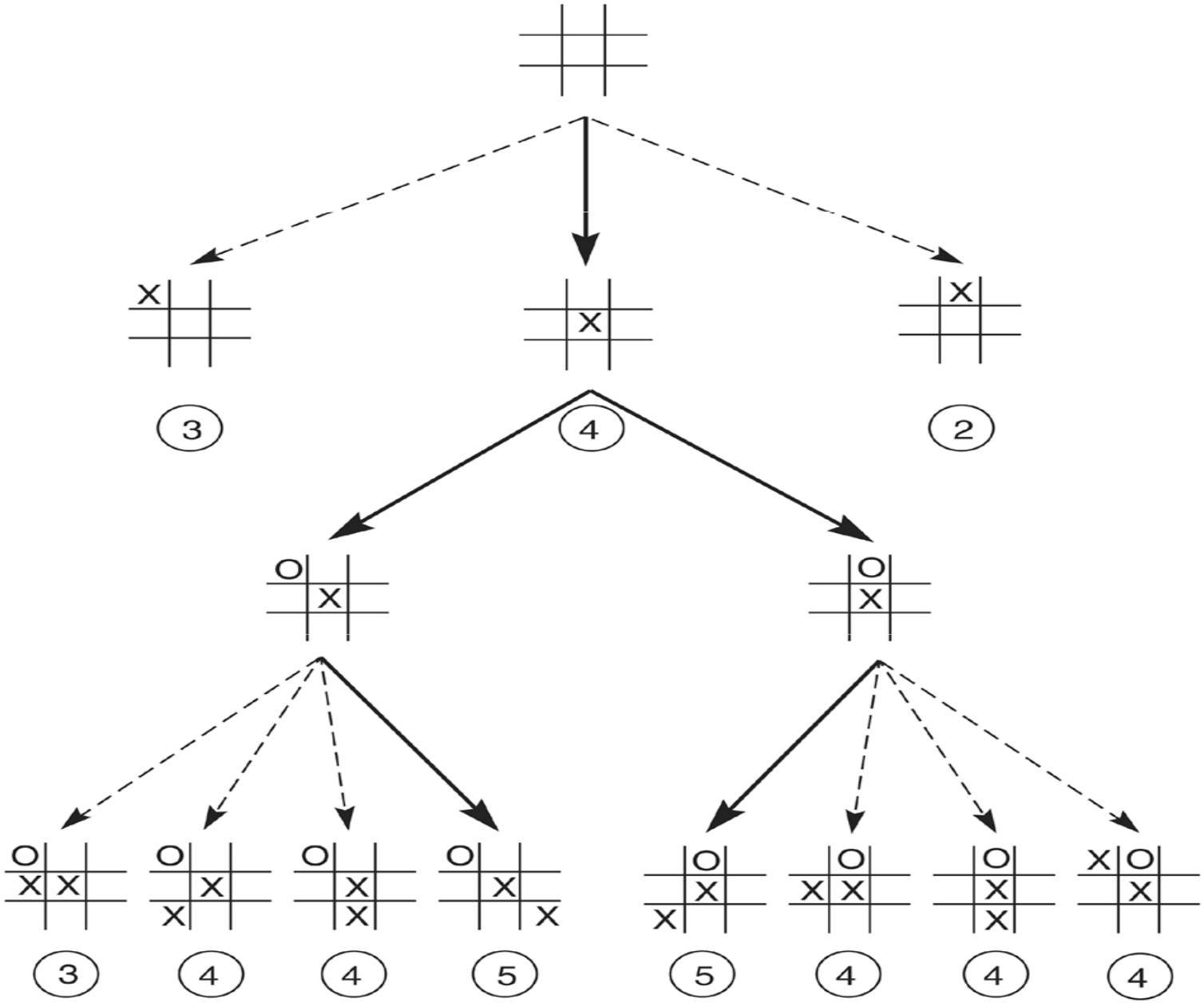


Ví dụ phép đo Heuristics (tt)



Heuristic "*Số đường thắng nhiều nhất*" (*theo các đường chéo trên bàn cờ)* áp dụng cho các con cờ đầu tên đặt vào bàn cờ trong bàn cờ tic-tac-toe

Ví dụ phép đo Heuristics (tt)



Tìm kiếm leo đồi – Hill Climbing Search (Pearl, 1984)

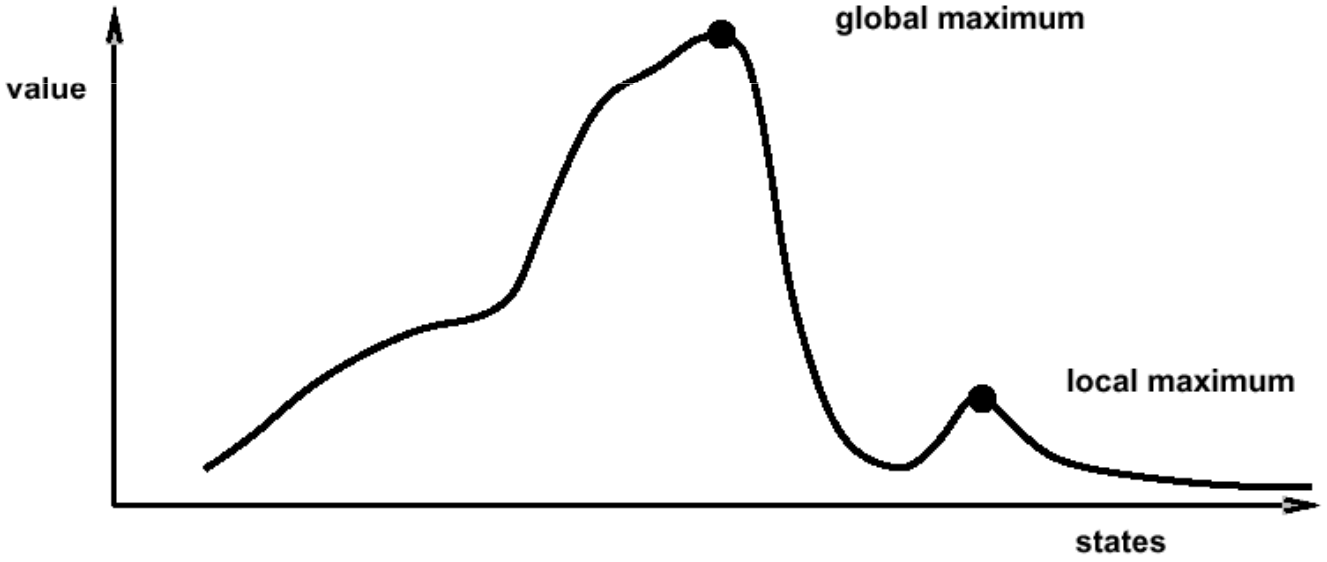
- Chọn một trạng thái tốt hơn trạng thái đang khảo sát để phát triển. Nếu không có thuật tóan phải dừng.
- Nếu chỉ chọn một trạng thái tốt hơn: leo đồi đơn giản, trạng thái tốt nhất: leo đồi dốc đứng.
- Sử dụng hàm H để biết trạng thái nào tốt hơn.
- Khác với tìm kiếm sâu, leo đồi không lưu tất cả các con mà chỉ lưu đúng một trạng thái được chọn nếu có.

Tìm kiếm leo đồi (tt)

- Giải thuật
  - Xét trạng thái bắt đầu
    - Nếu là đích ⇒ dừng
    - Ngược lại: thiết lập khởi đầu như TT hiện tại
  - Lặp đến khi: gặp đích hoặc không còn luật nào chưa được áp dụng vào TT hiện tại
    - Lựa một luật để áp dụng vào TT hiện tại để sinh ra một TT mới
    - Xem xét TT mới này
      - Nếu là đích ⇒ dừng
      - Nếu không là đích, nhưng tốt hơn TT hiện tại ⇒ thiết lập TT mới là TT hiện tại
      - Nếu không tốt hơn thì lặp tiếp tục

Tìm kiếm leo đồi (tt)

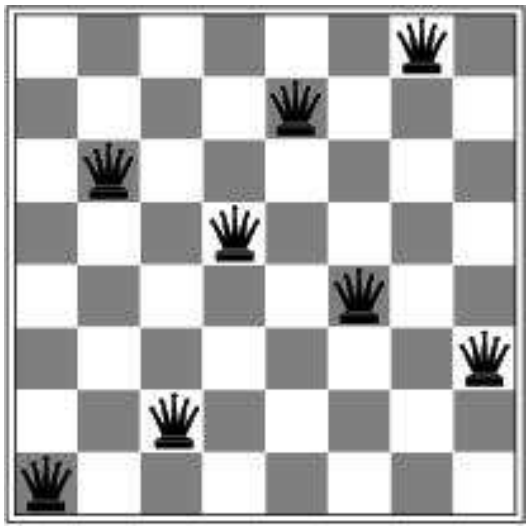
- Hiệu quả nếu có được hàm (H) đánh giá tốt
- Giới hạn
  - Có khuynh hướng bị sa lầy ở những cực đại cục bộ
    - Lời giải tìm được không tối ưu
    - Không tìm được lời giải mặc dù có tồn tại lời giải
  - Có thể gặp vòng lặp vô hạn do không lưu giữ thông tin về các trạng thái đã duyệt



Tìm kiếm leo đồi (tt)

- Bài toán 8 Hậu
  - Trạng thái bắt đầu: mỗi Hậu trên 1 cột
  - Trạng thái đích: các Hậu không thể tấn công nhau
  - Phép đo Heuristic h(n): số lượng các cập hậu đối kháng nhau

H(n) = 17



h(n) = 1

Tìm kiếm tốt nhất (BFS)

- Là phương pháp dung hoà của BrFS và DFS
- Có sử dụng để đánh giá ưu thế của mỗi trạng thái, có thể là ước lượng từ nó đến TT đích
- Tại mỗi bước, giải thuật sẽ chọn trạng thái mà nó cho rằng là có ưu thế nhất trong số các trạng thái đã sinh ra được đến thời điểm đó
- Khác với giải thuật leo đồi có cải tiến ở chổ: có lưu tất cả những TT đã phát sinh đến thời điểm chọn TT để xét tiếp
- Dùng hai danh sách:
  - **OPEN: chứa các TT sẽ được xét.**
  - **CLOSED: chứa các TT đã xét qua.**

Tìm kiếm tốt nhất (BFS)

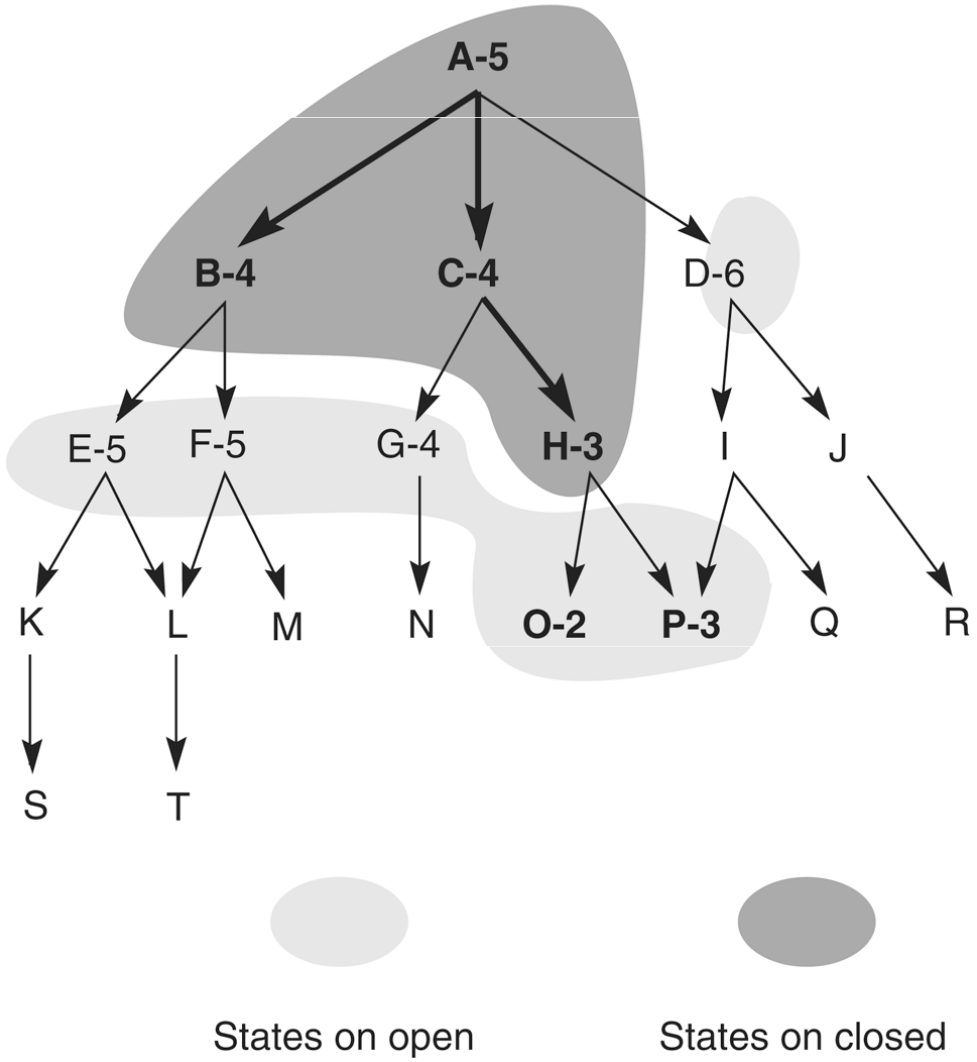
- Giải thuật
  - OPEN = [TT đầu]
  - Lặp đến khi: gặp đích hoặc OPEN rỗng
    - **Lấy TT tốt nhất từ OPEN**
    - **Phát sinh các con của nó**
    - **Với mỗi con:**
      - Nếu nó chưa được phát sinh: gán nó trị đánh giá, đưa vào OPEN, ghi nhận TT cha của nó
      - Nếu đã được phát sinh trước: Nếu đạt đến bởi đường khác tốt hơn

        $\Rightarrow$ ghi nhận lại TT cha của nó, cập nhật lại trị đánh giá của nó và của các con của nó

Tìm kiếm tốt nhất (BFS)

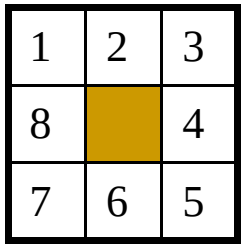
1. open = [A5]; closed = [ ]
2. Đánh giá A5; open = [B4,C4,D6];
   closed = [A5]
3. Đánh giá B4;
   open = [C4,E5,F5,D6];
   closed = [B4,A5]
4. Đánh giá C4;
   open = [H3,G4,E5,F5,D6];
   closed = [C4,B4,A5]
5. Đánh giá H3;
   open = [O2,P3,G4,E5,F5,D6];
   closed = [H3,C4,B4,A5]
6. Đánh giá O2;
   open = [P3,G4,E5,F5,D6];
   closed = [O2,H3,C4,B4,A5]
7. Đánh giá P3; tìm được lời giải!

Open là queue, xếp theo Heuristic tăng dần



Cài đặt hàm đánh giá (EF)

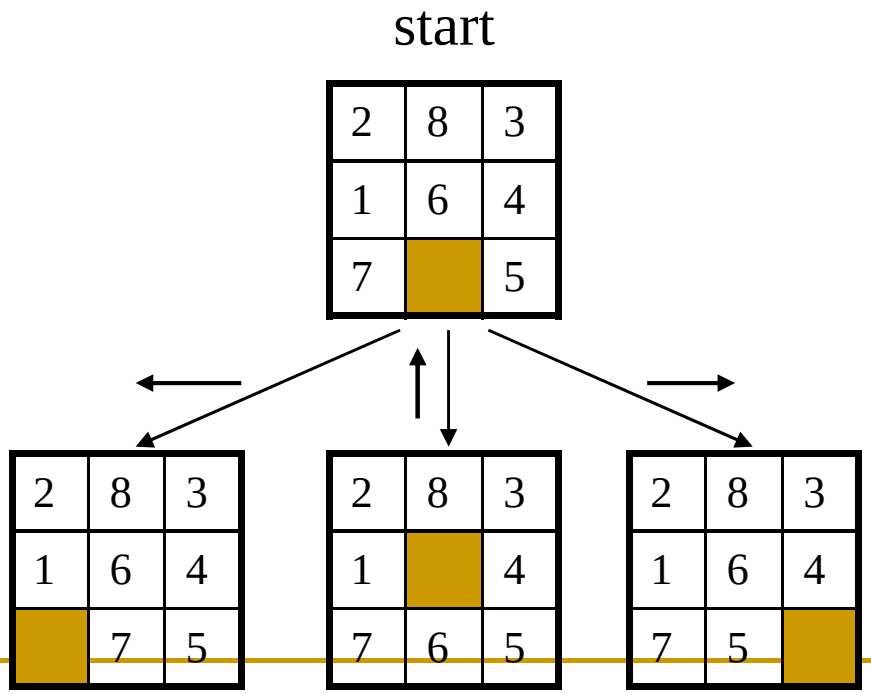
- Xét trò chơi 8-Ô, mỗi trạng thái n, một giá trị f(n):
  f(n) = g(n) + h(n)
  - **g(n)** = khoảng cách thực sự từ n đến trạng thái bắt đầu
  - **h(n)** = hàm heuristic đánh giá khoảng cách từ trạng thái n đến mục tiêu.



goal

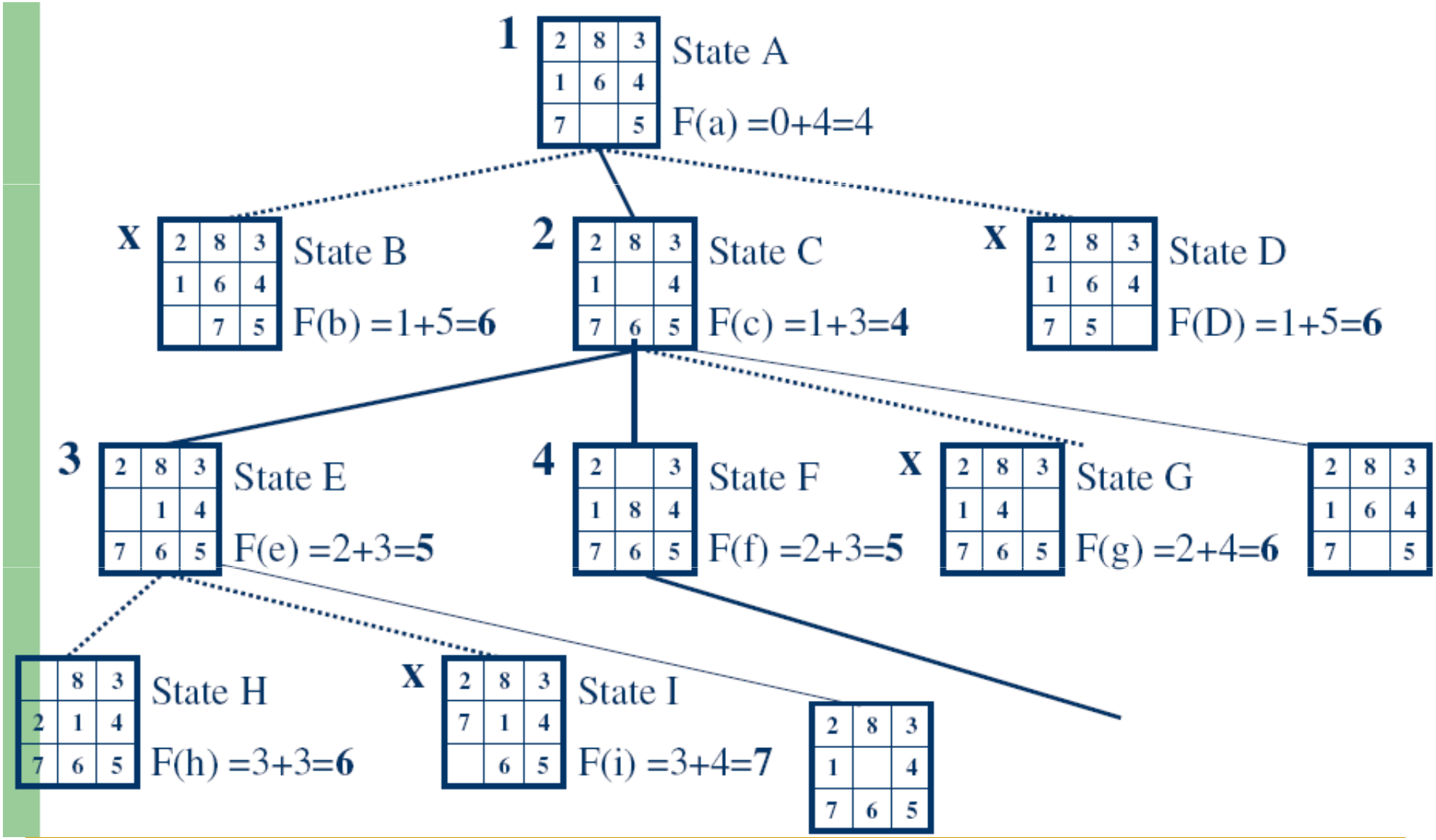
h(n): số lượng các vị trí còn sai; g(n) = 1

g(n) = 0

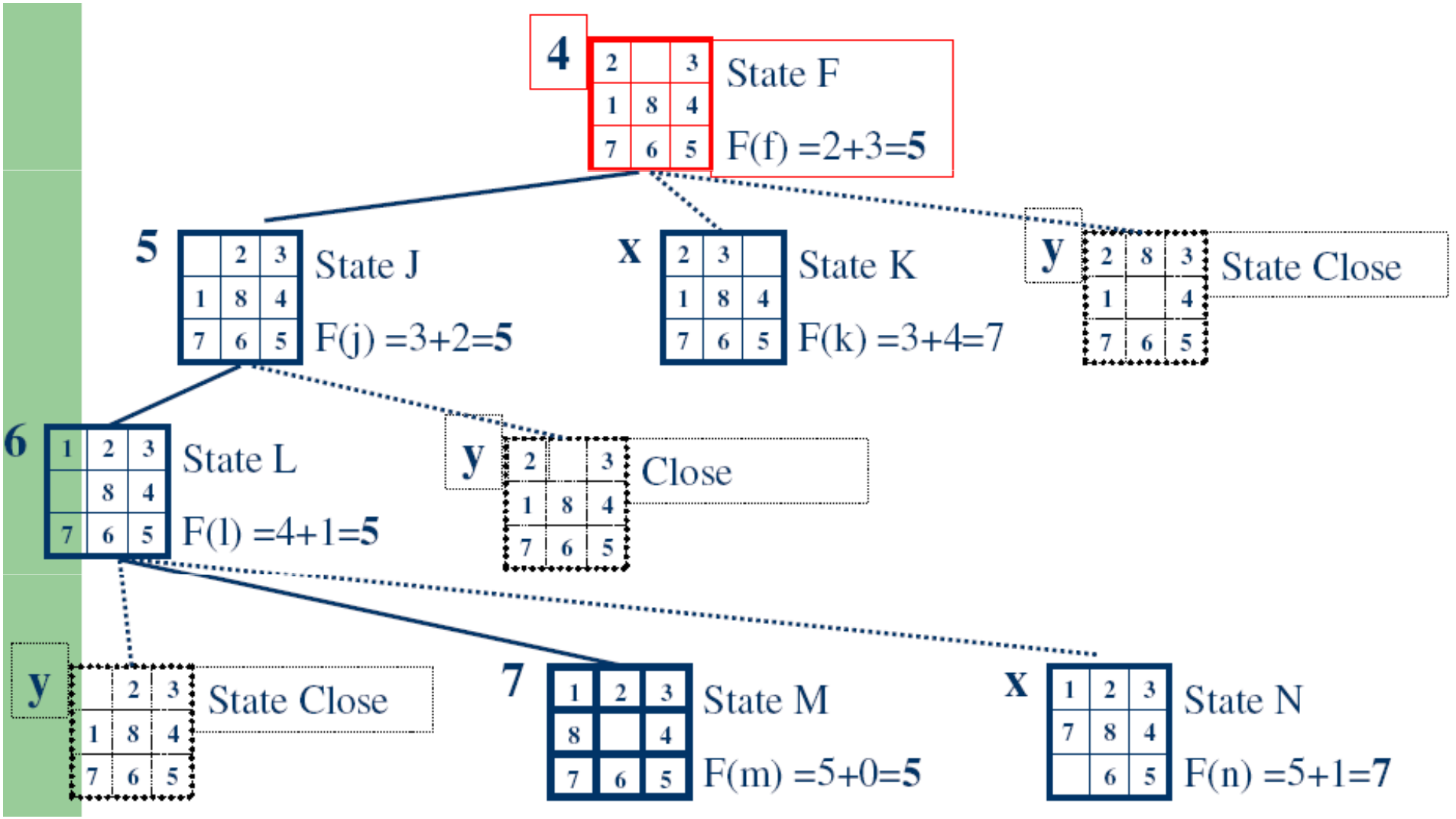


f(n) = **6** **4** **6**

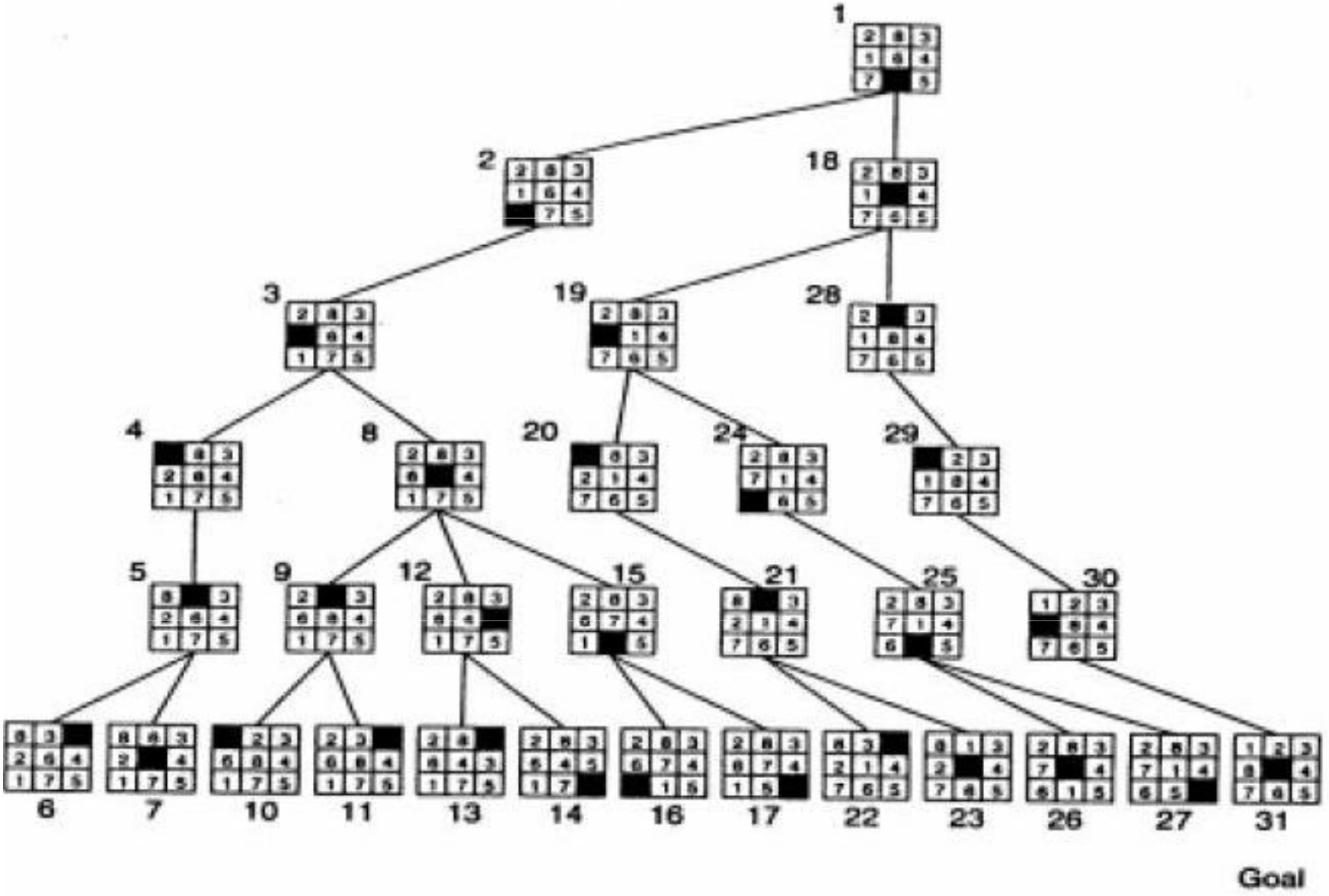
Ví dụ


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

Ví dụ …


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

Ví dụ …


1
2
18
3
19
28
4
8
20
24
29
5
9
12
15
21
25
30
6
7
10
11
13
14
16
17
22
23
26
27
31
Goal

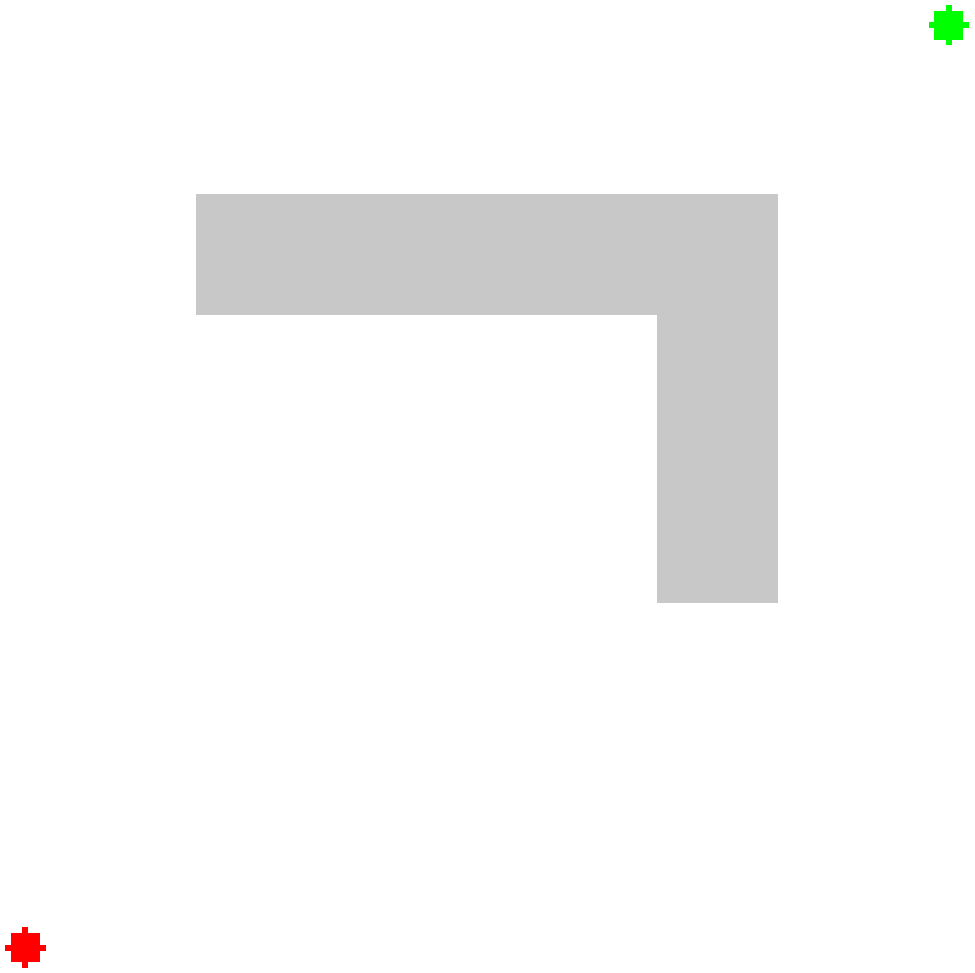
Giải thuật A*

- A* là giải thuật tổng quát hơn BestFS, nó tìm kiếm trên KGTT là đồ thị.
- Vì là đồ thị nên phát sinh nhiều vấn đề khi tìm đường đi tối ưu.
- Để ý rằng nghiệm là đường đi nên ta phải lưu lại vết của đường đi này.
- Trong các giải thuật trước, để tập trung cho tư tưởng chính của các giải thuật đó chúng ta bỏ qua chi tiết này, nhưng trong giải thuật này chi tiết này được đề cập vì nó liên quan đến nghiệm một cách trực tiếp.

Thông tin mỗi nút

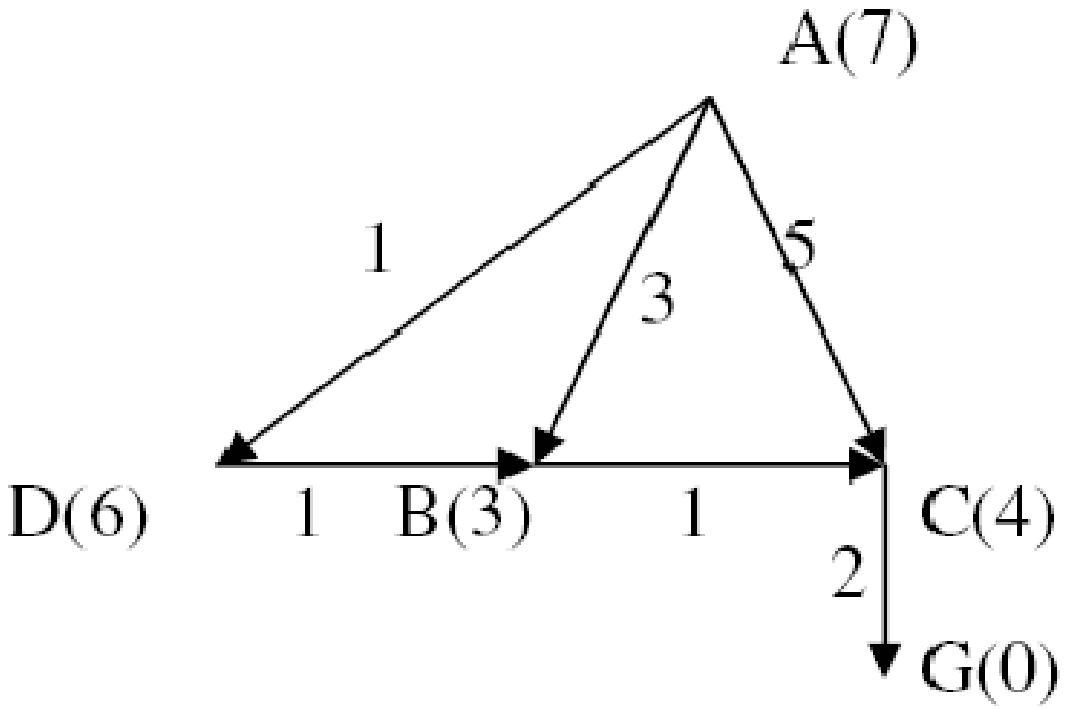
- Mỗi trạng thái u tùy ý sẽ gồm bốn yếu tố (g(n), h(n), f(n), cha(n)). Trong đó:
- G(n), h(n), f(n) đã biết
- Cha(n) là nút cha của nút đang xét n

A* Search Progress



source: wikipedia page for A* Algorithm; by Subh83

Mô tả hoạt động của A*



Lưu các trạng thái

Bước	Open	closed
1	A(0,7,0,-)	
2	B(3,3,6,A), D(1,6,7,A), C(5,4,9,A)	A(0,7,0,-)
3	D(1,6,7,A), C(4,4,8,B)	B(3,3,6,A)
4	B(2,3,5,D), C(4,4,8,B)	D(1,6,7,A)
5	C(3,4,7,B)	B(2,3,5,D)
6	G(5,0,5,C)	C(3,4,7,B)

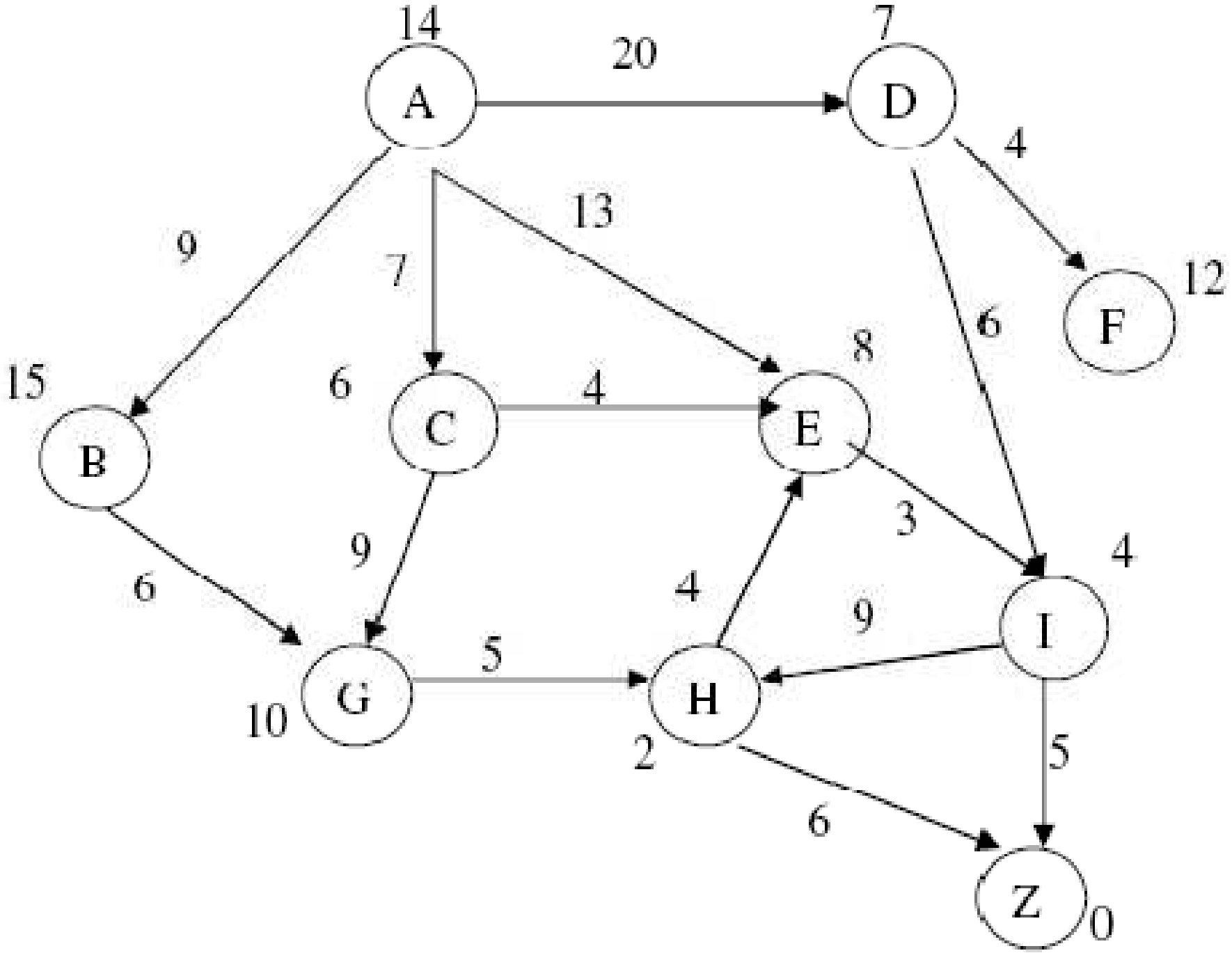
Giải thuật dừng ở bước 6 và đường đi thu được độ dài 5 như sau A-D-B-C-G.

Chi tiết các bước

- Ở bước 2, mọi việc xảy ra bình thường
- Ở bước 3, tìm được đường đi đến C qua B ngắn hơn nên các giá trị của C trong open phải được sửa đổi.
- Ở bước 4, mặc dù B đã nằm trong closed, tức đã xét xong nhưng đường đi mới qua D đến B ngắn hơn nên B phải được lấy khỏi closed chuyển qua open chờ xét lại với giá trị mới.
- Ở bước 5, lại xảy ra việc chỉnh sửa các giá trị của C như ở bước 3.
- Giải thuật dừng ở bước 6 và đường đi thu được độ dài 5 như sau A-D-B-C-G.

Mã giả giải thuật A*

$g(n_o)=0$; $f(n_o)=h(n_o)$;
open:=[n_o]; closed:=[];
while open<>[] do
    loại n bên trái của open và dưa n vào closed;
    if (n là một đích) then thành công, thoát
    else
        Sinh các con m của n;
        For m thuộc con(n) do
            g(m):=g(n)+c[n,m];
            If m không thuộc open hay closed
                f(m):=g(m)+h(m); cha(m):=n; Bỏ m vào open;
            If m thuộc open (tồn tại m' thuộc open, sao cho m=m')
                If g(m)<g(m') then
                    g(m'):=g(m); f(m'):=g(m')+h(m'); Cha(m'):=n;
            If m thuộc closed (tồn tại m' thuộc closed, sao cho m=m')
                 If g(m)<g(m') then
                     f(m):=g(m)+h(m); cha(m):=n; Đưa n vào open; Loại n' khỏi closed;
    Xếp open để t.thái tốt nhất nằm bên trái;


14
A
20
7
D
4
9
7
13
12
F
6
8
15
6
C
4
E
B
3
4
9
6
4
9
I
5
10
G
H
2
5
6
Z
0

Đánh giá giải thuật A*

- Một hàm đánh giá h(n) được gọi là chấp nhận được hay là hàm đánh giá thấp nếu như h(n)<=h*(n) với mọi n, ở đây h*(n) là đường đi ngắn nhất từ n đến đích.
- Nếu hàm đánh giá h(n) là chấp nhận được thì thuật toán A* là tối ưu.
- A* là thuật tóan hiệu quả nhất trong các thuật toán đầy đủ và tối ưu.

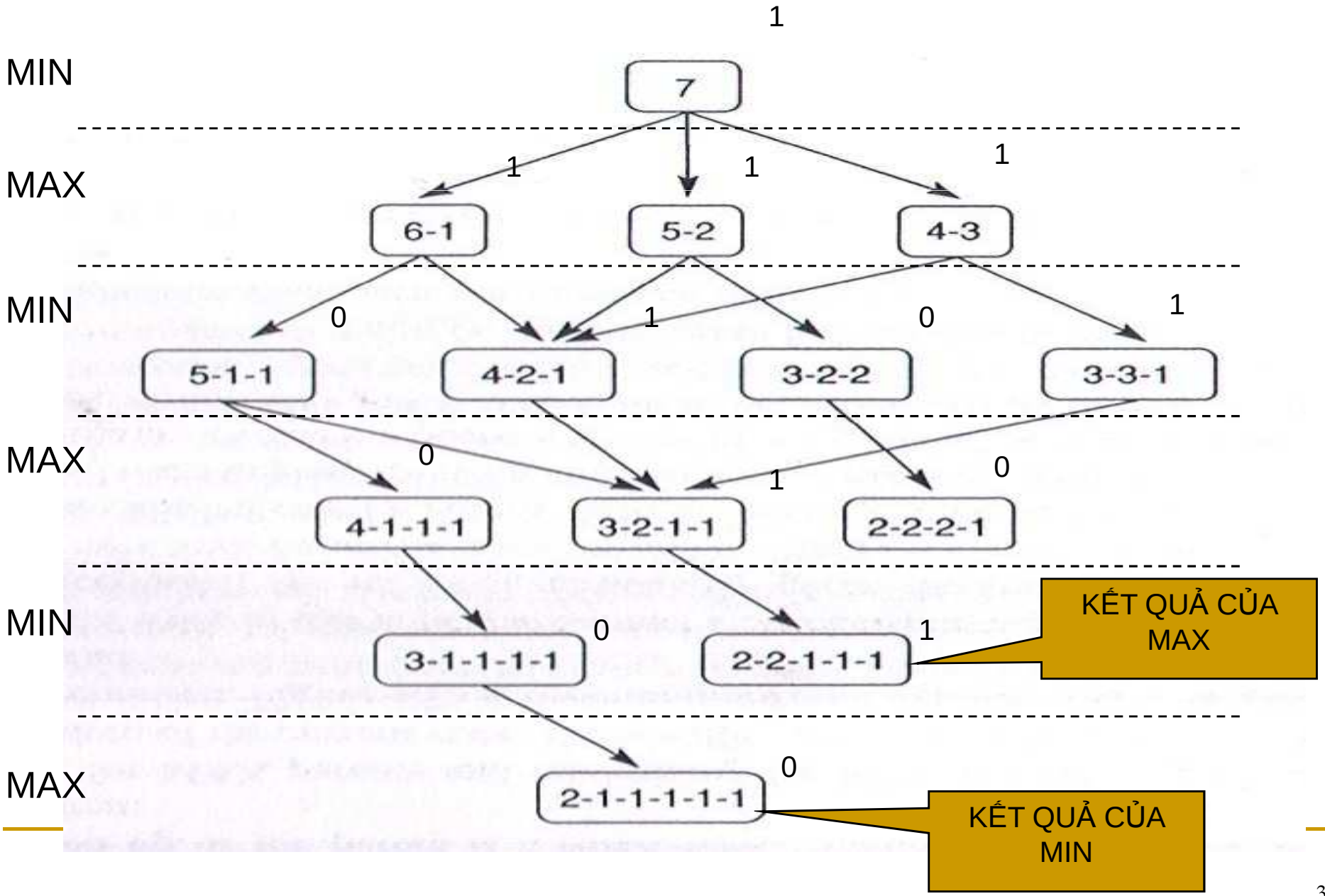
Chiến lược tìm kiếm có đối thủ

- Đặc điểm
  - Hai người thay phiên đi (xen kẽ)
  - Hai người biết thông tin đầy đủ về nhau
  - Mỗi người tìm kiếm nước đi
  - Nước đi tốt nhất là nước đi dẫn đến phần thắng.
  - Biểu diễn KGTT bằng cây: cây trò chơi.
- Giải thuật tiêu biểu: MiniMax

Thủ tục Minimax cơ bản

- Ví dụ: trò chơi Nim
  - Có n (n>2) đồng xu.
  - Mỗi nước đi, người chơi chia các đồng xu này thành hai đống nhỏ có số lượng mỗi đống khác nhau.
  - Người thua sẽ là người cuối cùng không chia được theo yêu cầu của bài toán.
- Phân tích
  - Tính toán phản ứng của đối thủ là khó khăn chủ yếu của bài toán này
  - Cách giải quyết là giả thiết đối thủ cũng sử dụng kiến thức về không gian trạng thái

Trò Chơi NIM: giá trị tại các nút

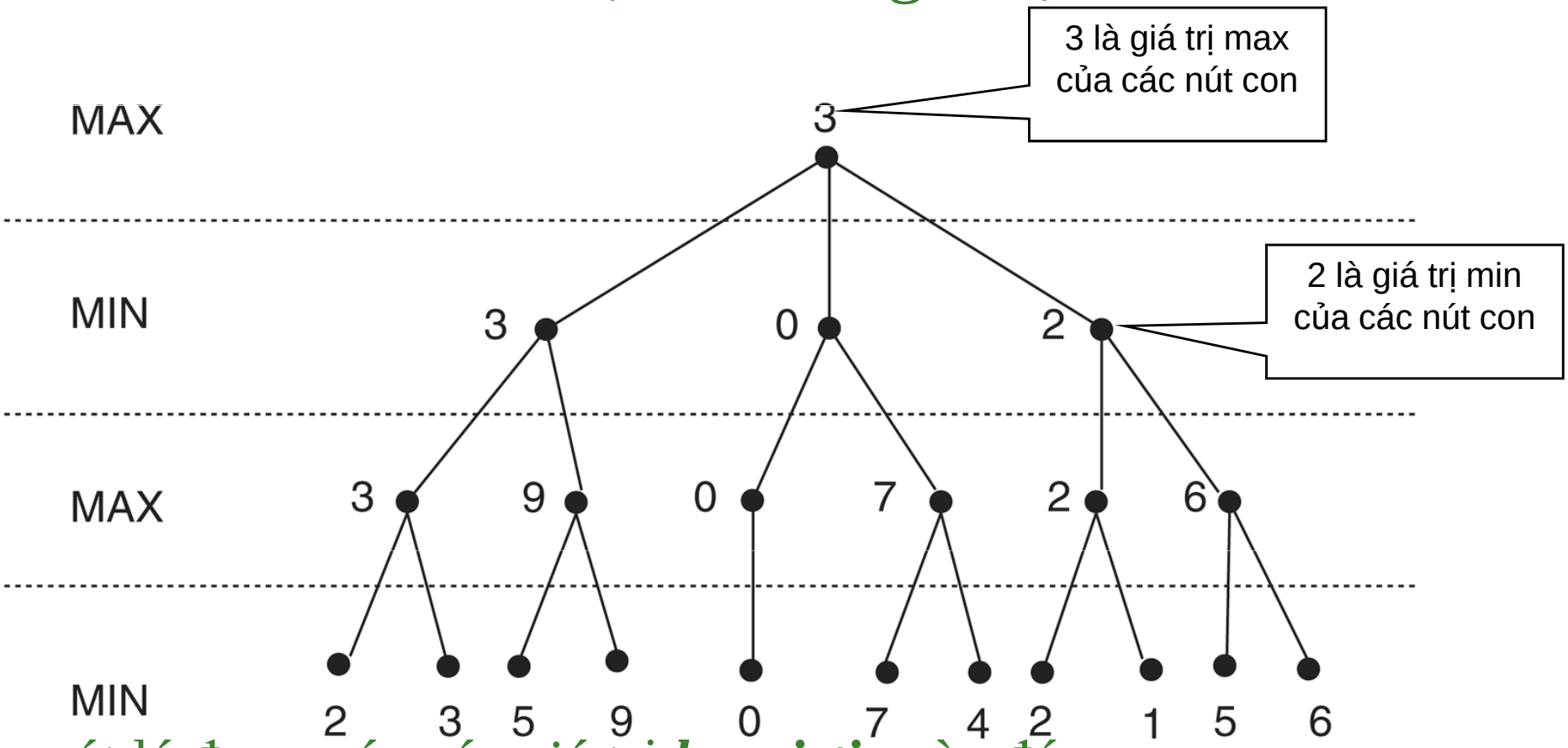


Trò Chơi NIM

- Hai đấu thủ: MIN và MAX.
- Trong đó MAX luôn tìm cách tối đa ưu thế của mình và MIN tìm mọi cách để đưa MAX vào thế khó khăn nhất.
- Mỗi mức trên KGTT ứng với một đấu thủ.
- Để chỉ dẫn được cách đi, chúng ta sẽ gán cho các nút lá là 1 nếu MAX thắng, là 0 nếu MIN thắng.
- Gán giá trị cho các nút
  - Truyền ngược các trị từ các nút lá về gốc theo qui tắc:
    - Nếu đỉnh ở mức MAX, gán trị cho đỉnh này bằng giá trị lớn nhất trong các giá trị của các con của nó
    - Nếu đỉnh ở mức MIN, gán trị cho đỉnh này bằng giá trị bé nhất trong các trị của các con của nó.

Minimax với độ sâu lớp cố định

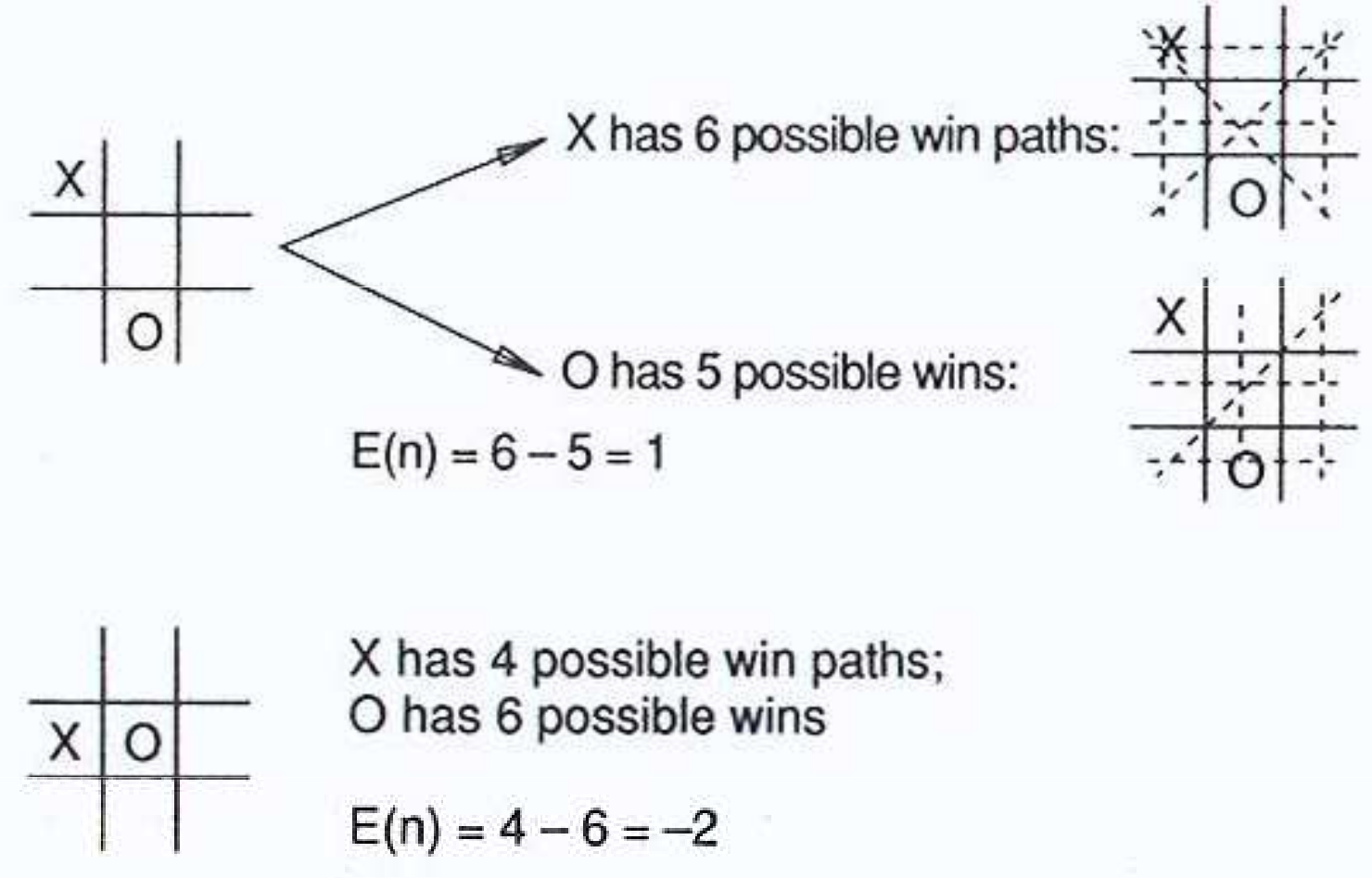
- Minimax đối với một KGTT giả định



Các nút lá được gán các giá trị ***heuristic*** nào đó

Còn giá trị tại các nút trong là các giá trị nhận được dựa trên giải thuật Minimax *(min hay max cua các nút con)*

Heuristic trong trò chơi tic-tac-toe



Hàm Heuristic: **E(n) = M(n) – O(n)**

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

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

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

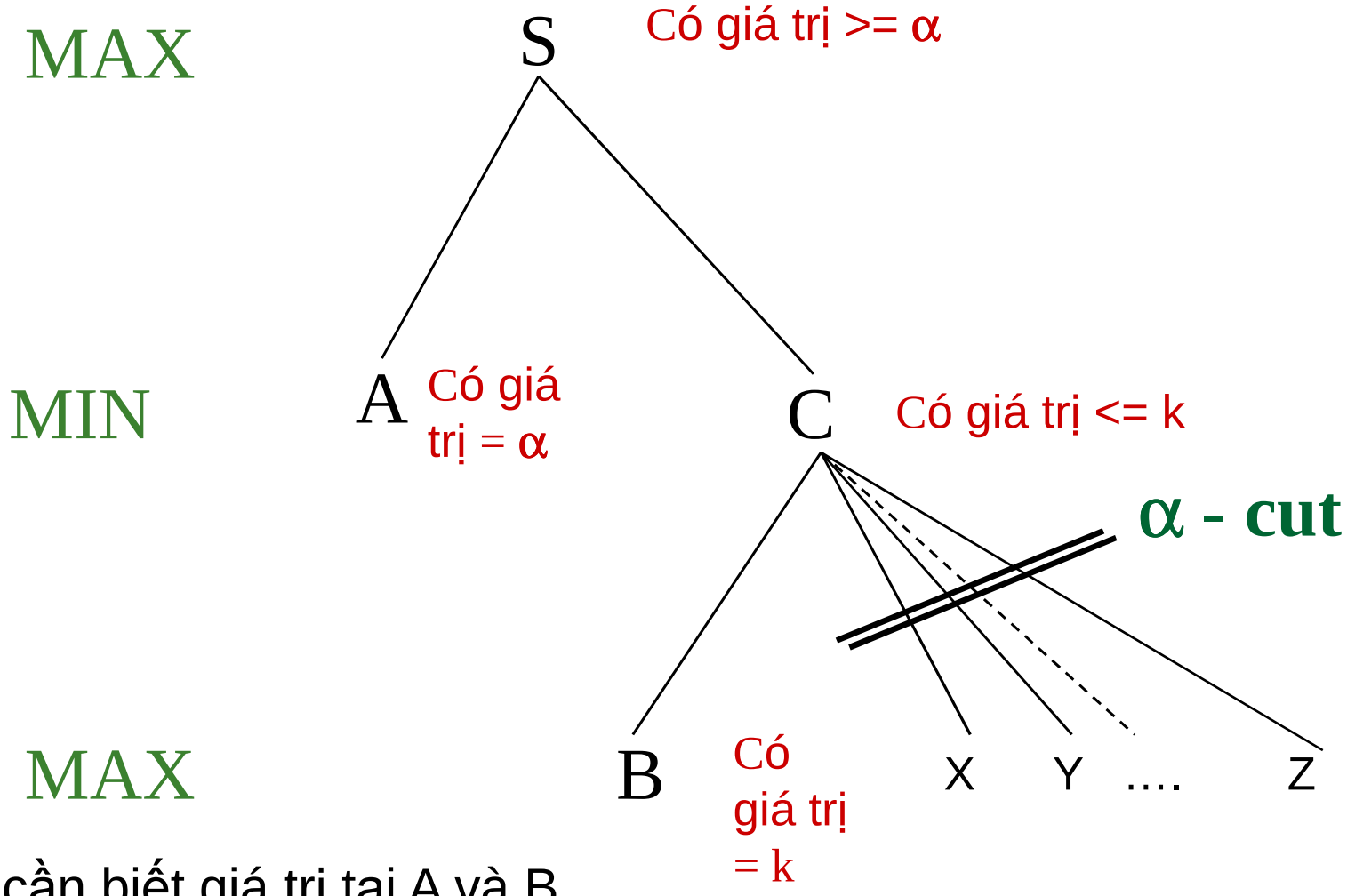
Giải thuật cắt nhánh alpha-beta

- Hạn chế với số mức d đi nữa thì số trạng thái đã rất lớn.
- Cờ vua: nhân tố nhánh b=35; d=3 có 35*35*35=42.785 trạng thái
- Giảm bớt các trạng thái cần khảo sát mà vẫn không ảnh hưởng gì đến việc giải quyết bài toán.
- Cắt bỏ các nhánh không cần khảo sát.

Giải thuật cắt nhánh alpha-beta

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

Cắt nhánh α (vị trí MAX)



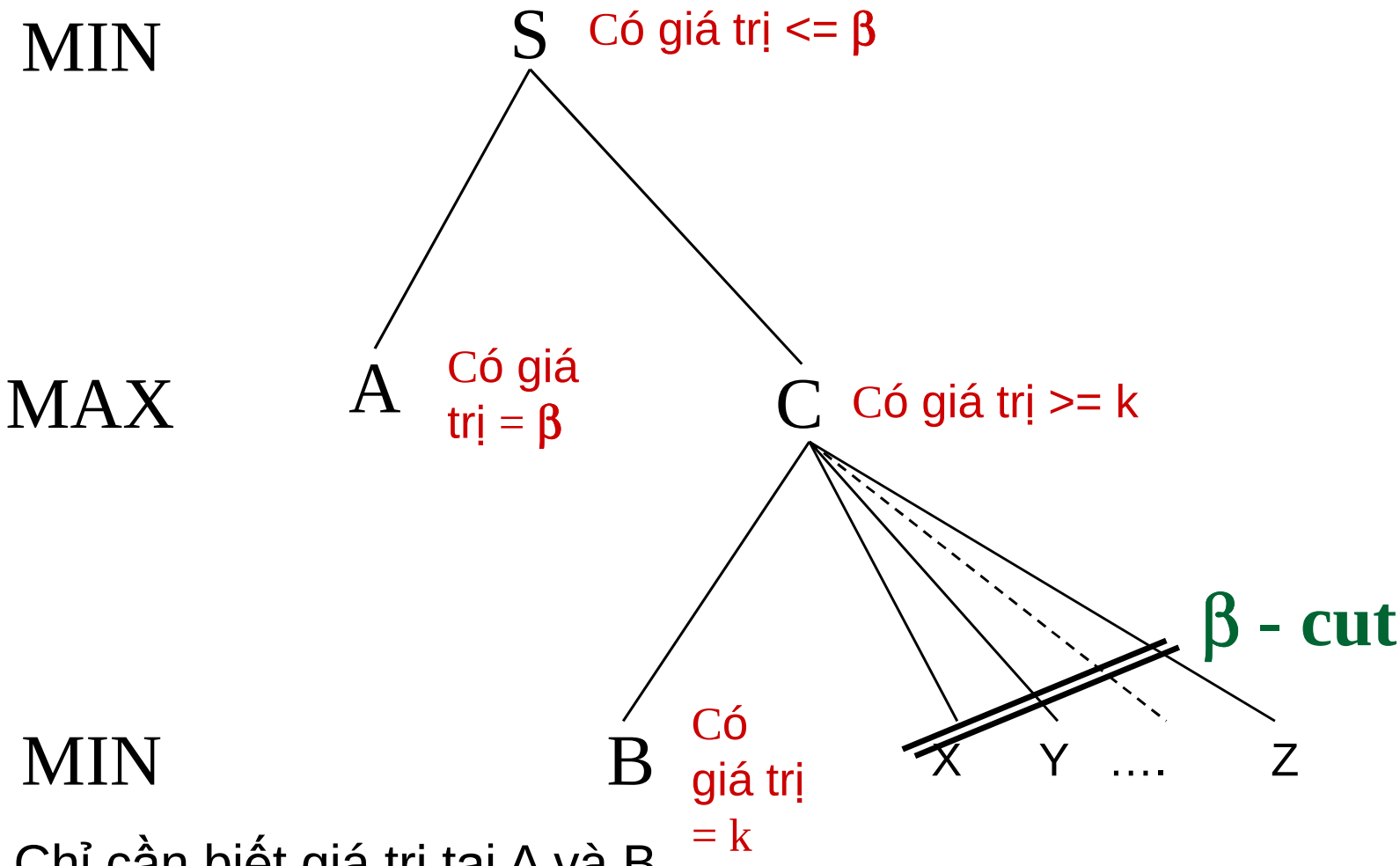
Điều kiện 1: Chỉ cần biết giá trị tại A và B

Điều kiện 2: Giá trị A > giá trị B

Điều kiện 3: X, Y, .. , Z ở vị trí Max

Bỏ những cây con có gốc là X,Y,…, Z

Cắt nhánh β (vị trí Min)



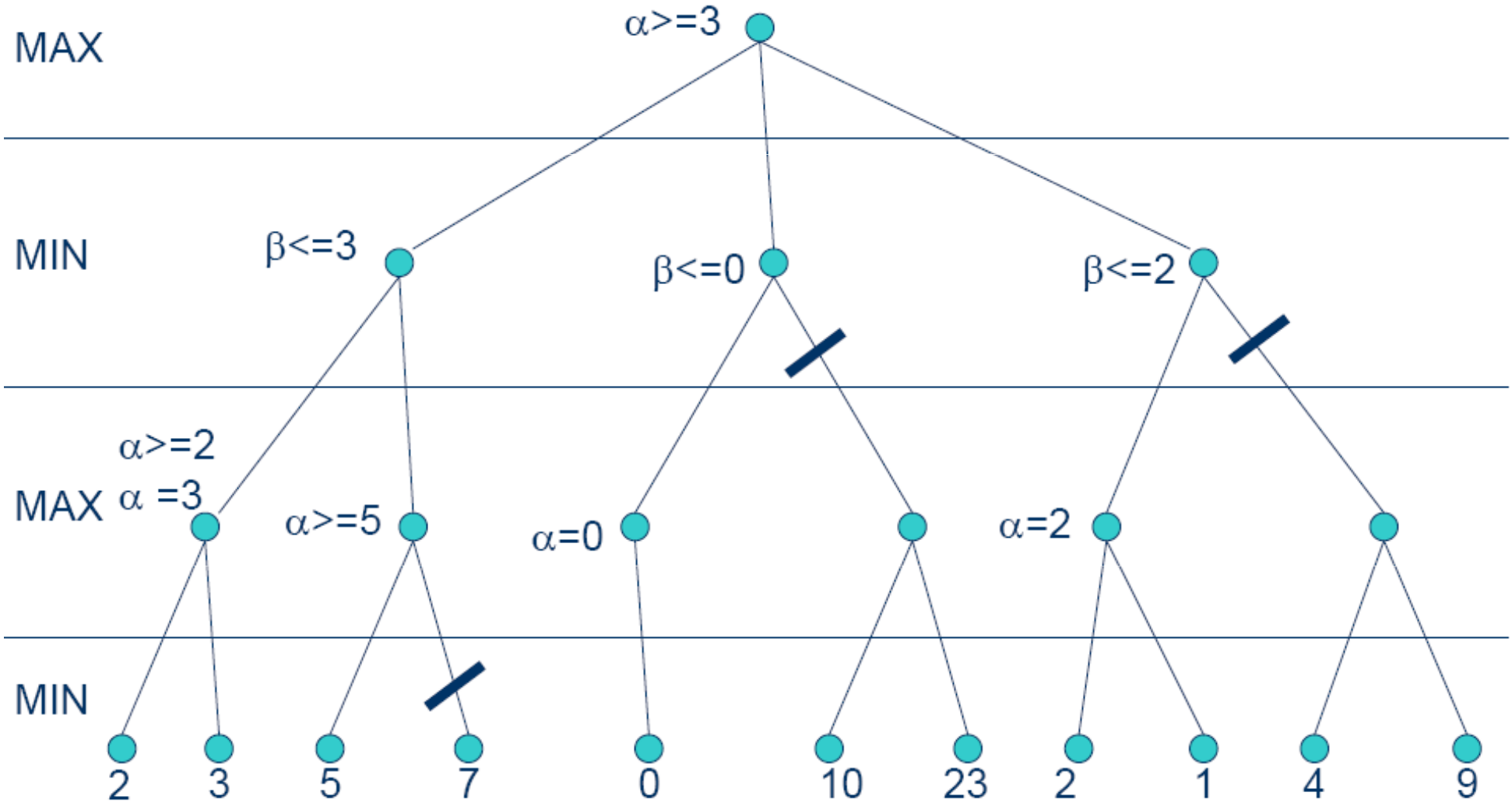
Điều kiện 1: Chỉ cần biết giá trị tại A và B

Điều kiện 2: Giá trị A < giá trị B

Điều kiện 3: X, Y, .. , Z ở vị trí Min

Bỏ những cây con có gốc là X,Y,…, Z

Cắt nhánh α-β: ví dụ



α-β pruning

Bài tập: bài 1 (trò đố 8 ô như)

Start

1	2	3
6	7	
8	5	4

Goal

1	2	3
8		4
7	6	5

Dùng các hàm lượng giá heuristic sau

- h1 = số lượng các vị trí sai khác so với trạng thái goal.
- h2 = tổng số độ dời ngắn nhất của các ô về vị trí đúng (khoảng cách Manhattan)

Hãy triển khai không gian trạng thái của bài toán đến mức 5 (nếu chưa tìm được goal):

a) Theo giải thuật leo núi

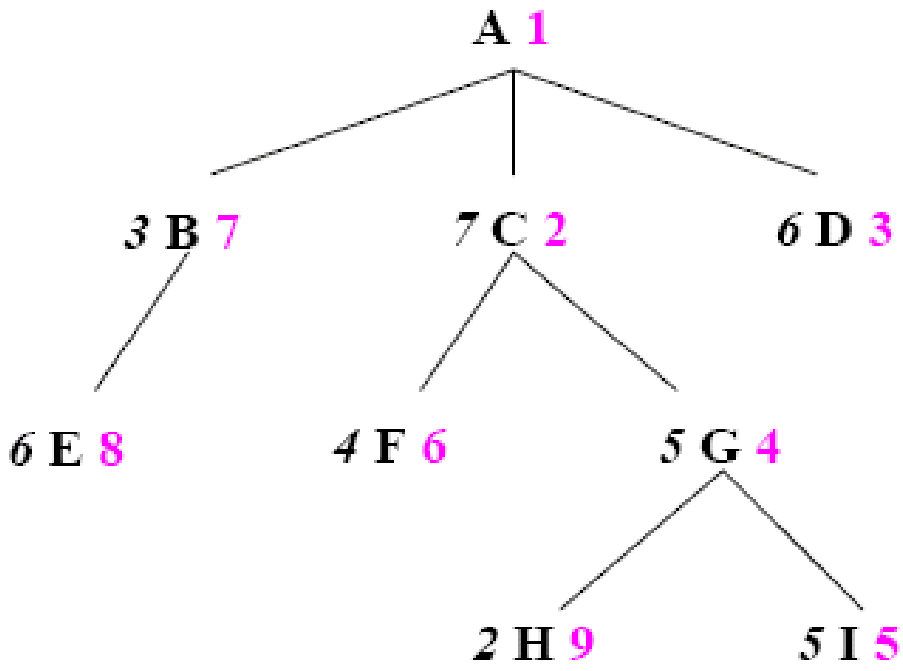
b) Theo giải thuật tìm kiếm rộng

c) Theo giải thuật tìm kiếm sâu

d) Theo giải thuật tìm kiếm tốt nhất đầu tiên

Bài tập: bài 2 (duyệt đồ thị)

Trong cây tìm kiếm dưới đây, mỗi nút có 2 giá trị đi kèm: giá trị bên trái của nút (in nghiêng) thể hiện giá trị heuristic của nút, và giá trị bên phải nút thể hiện thứ tự nút được duyệt qua. Với mỗi chiến lược tìm kiếm dưới đây, hãy viết danh sách thứ tự các nút được duyệt, so sánh và cho biết ta đã dùng giải thuật tìm kiếm nào trên cây :



a) Tìm kiếm rộng BrFS

b) Tìm kiếm sâu DFS

c) Tìm kiếm tốt nhất đầu tiên BFS

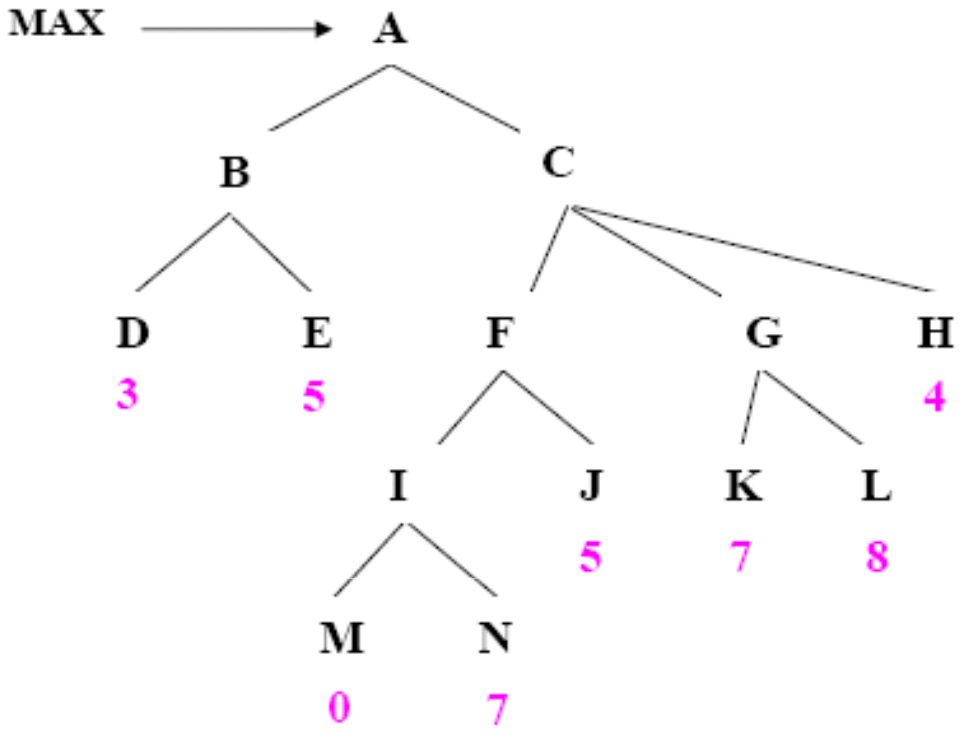
d) Tìm kiếm leo núi

f) A*

Bài tập: bài 3 (minimax)

Liệt kê danh sách các nút được duyệt theo tìm kiếm DFS.

- Thực hiện giải thuật Minimax trên cây.



- Sẽ có gì khác biệt nếu như ta dùng giải thuật cắt tỉa alpha – beta để định trị nút gốc cho cây?