Chương 3: Chiến lược tìm kiếm có thông tin heuristic

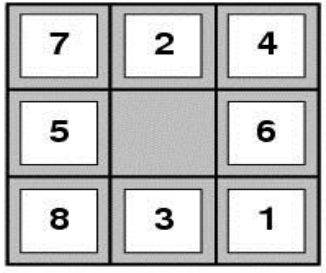
Giảng viên: Nguyễn Văn Hòa

Khoa CNTT - ĐH An Giang

Nội dung

- Khái niệm
- Tìm kiếm tốt nhất trước
- Phương pháp leo đồi
- Tìm kiếm Astar (A*)
- Cài đặt hàm đánh giá

Không gian tìm kiếm

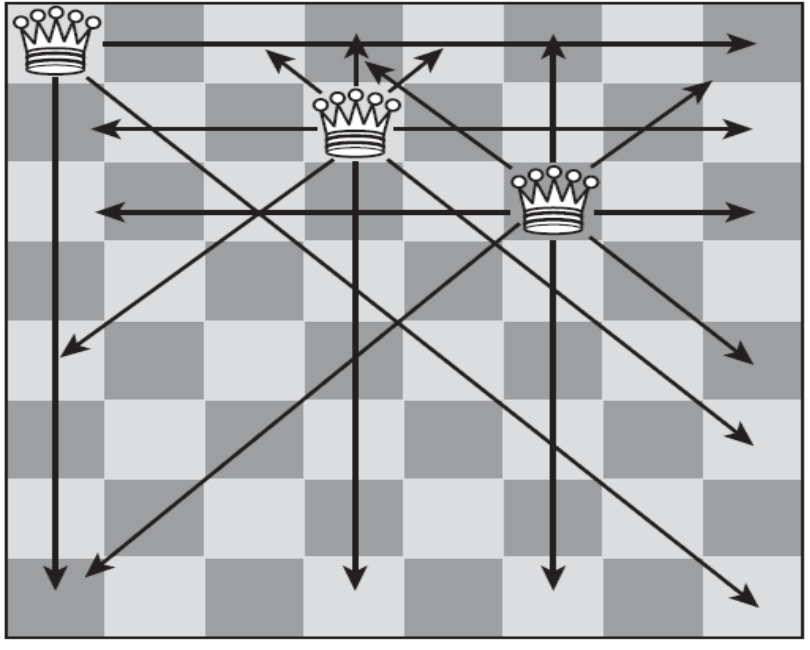


- 8-puzzle
  - Lời giải cần trung bình 22 cấp (depth)
  - Độ rộng của bước ~ 3
  - Tìm kiếm vét cạn cho 22 cấp cần
    - 3.1 x 10^{10} states
  - Nếu chỉ giới hạn ở d=12, cần trung bình 3.6 triệu trạng thái

    [24 puzzle có 10^{24} trạng thái]

Không gian tìm kiếm (tt)

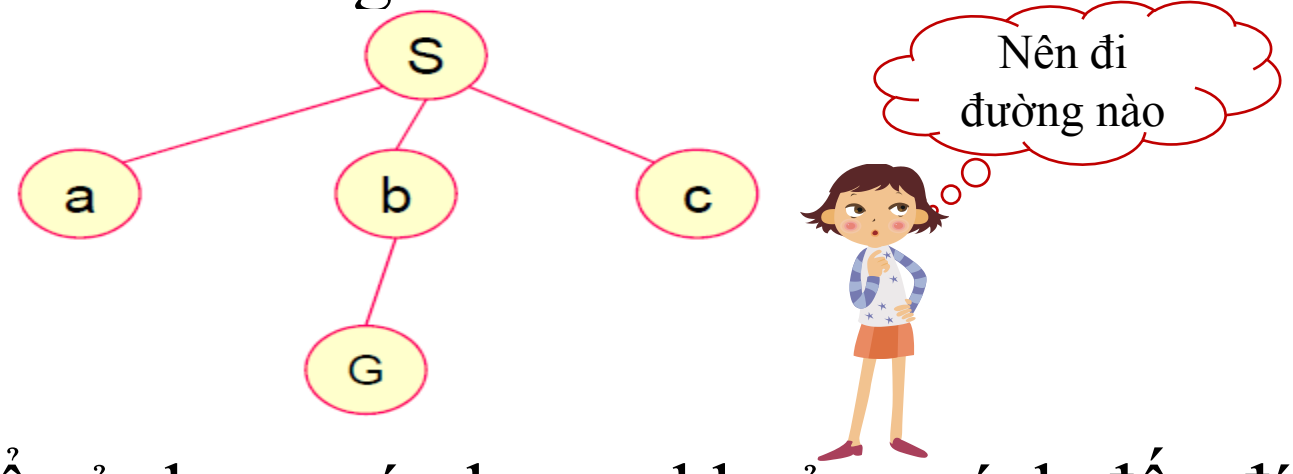
- Bài toán 8 hậu
  - Hậu đầu tiên có thể đặt ở một trong 64 ô
  - Hậu thứ hai có thể đặt ở một trong 63 ô
  - ..
  - Tổng số trạng thái la $64x63\ldots x57 = 1.8 \times 10^{14}$



$\Rightarrow$ Cần chiến lược tìm kiếm có thông tin heuristic

Tìm kiếm có thông tin heuristic (tt)

- Trong tìm kiếm mù, thông tin về TT đích không đóng vai trò trong tìm kiếm

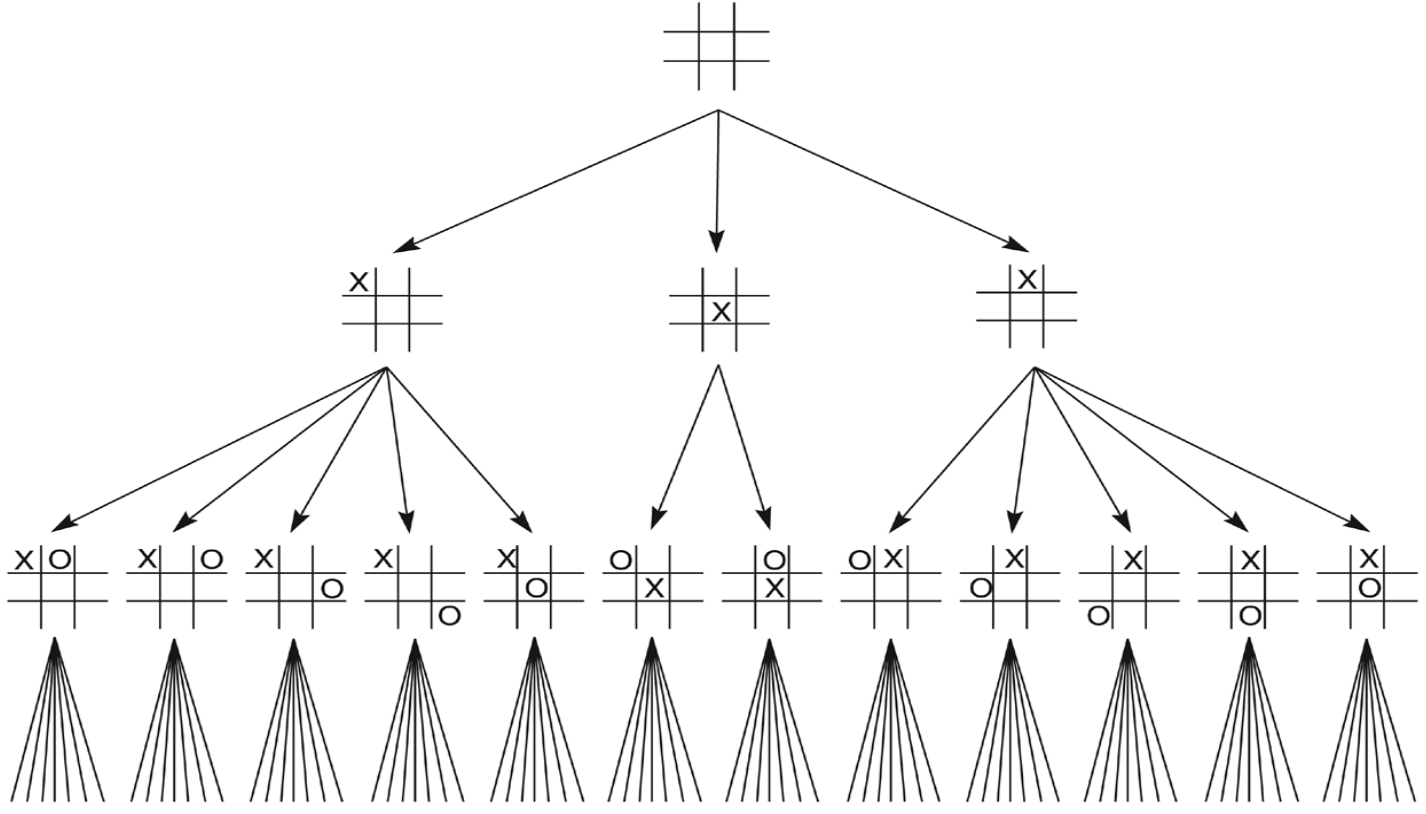


- Có thể sử dụng ước lượng khoảng cách đến đích giữa các TT để tìm đường đi

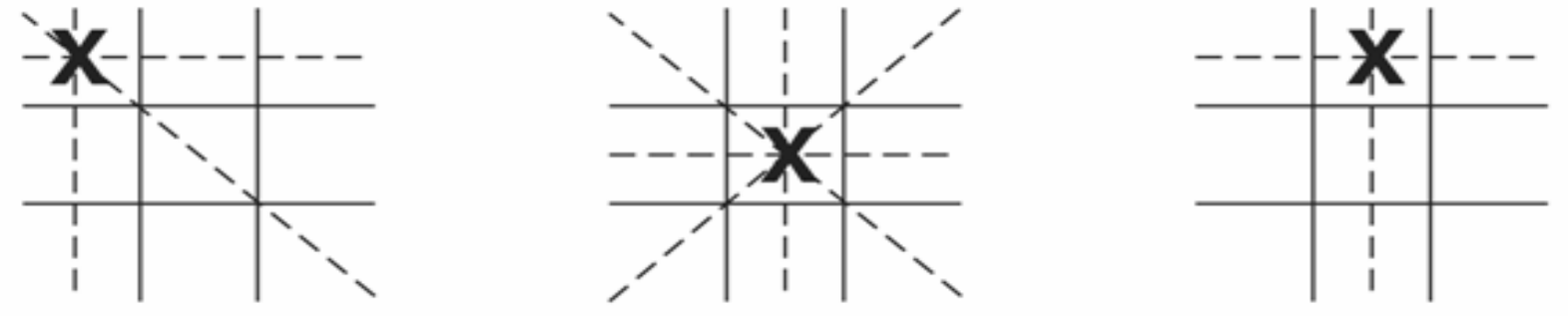
Tìm kiếm có thông tin heuristic (tt)

- Tìm kiếm có thông tin hỗ trợ heuristic chính là hàm heuristic
- 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)*
  - *TK tốt nhất (best-first search), A* search, Greedy,...*

Ví dụ phép đo Heuristics

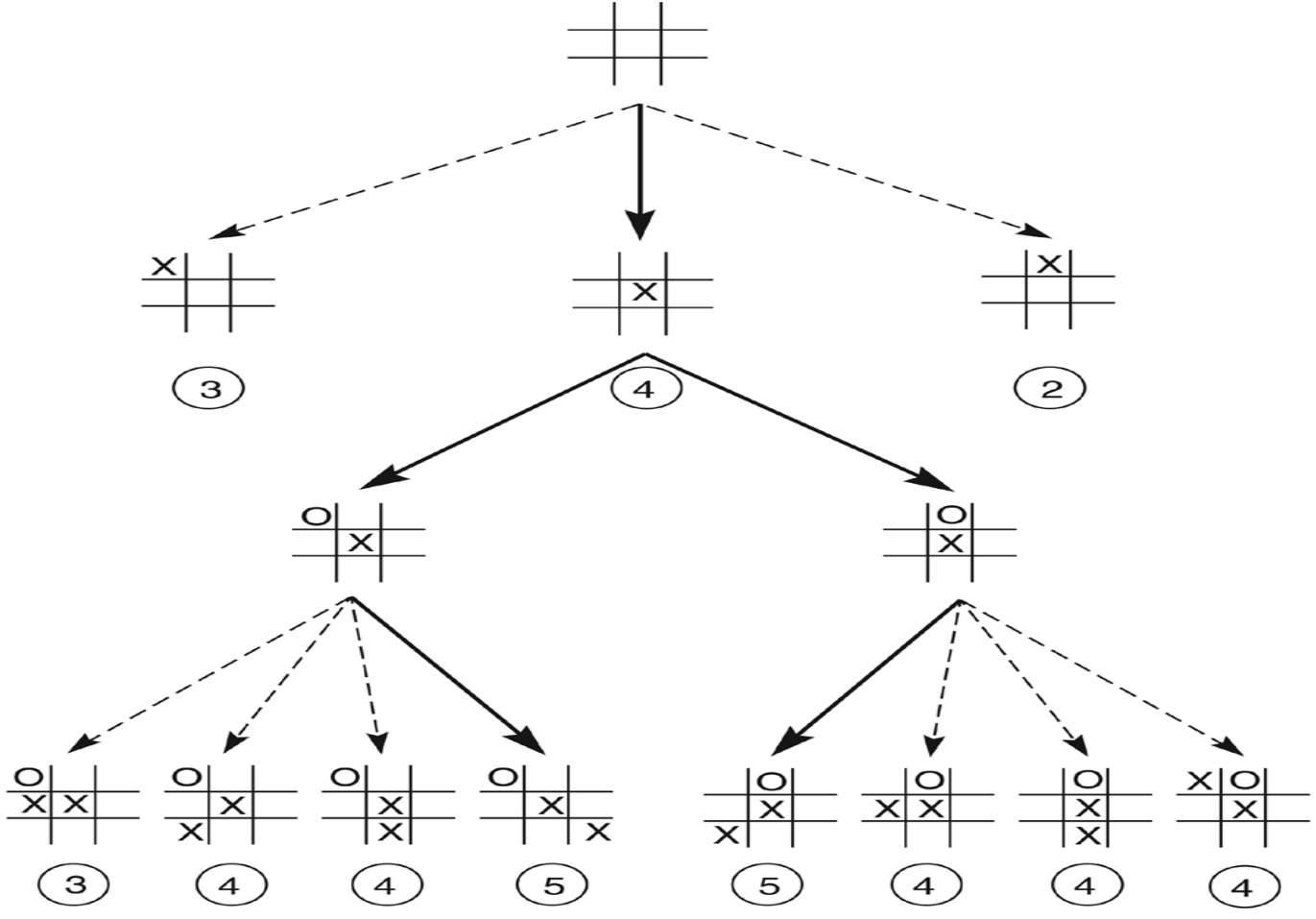


Ví dụ phép đo Heuristics (tt)



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

- Chọn một trạng thái tốt hơn trạng thái hiện hành để mở rộng. Nếu không, thuật toán 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

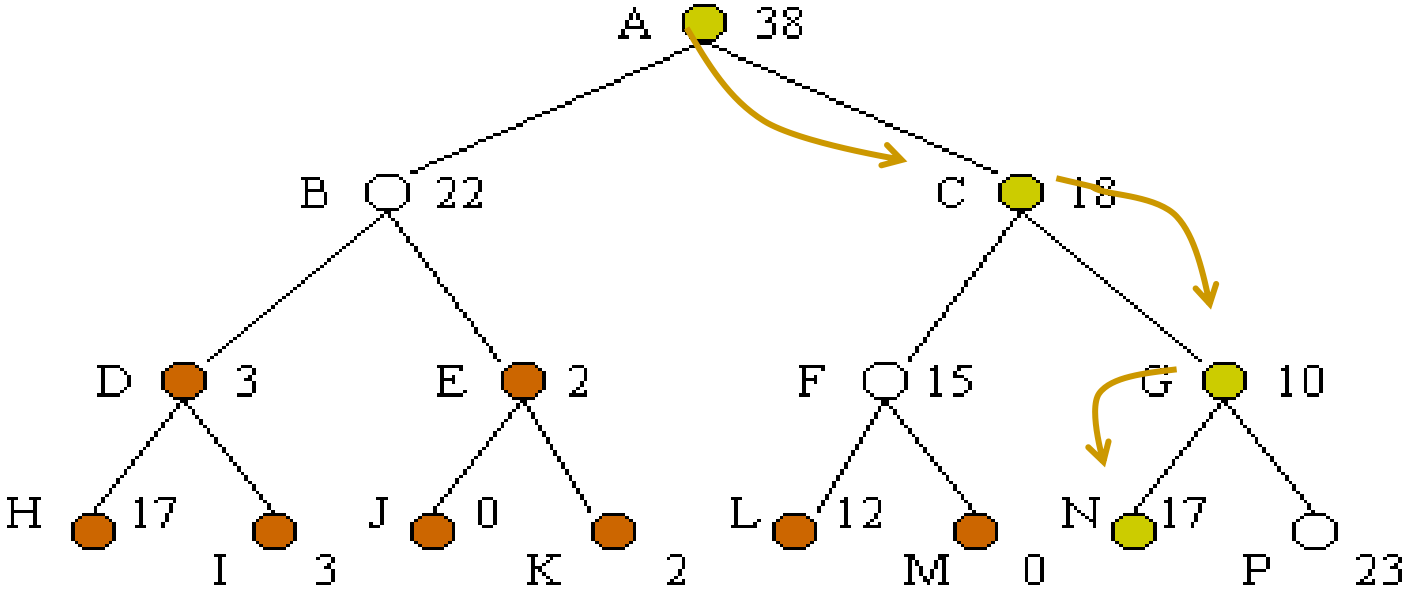
Tìm kiếm leo đồi (tt)

- Giải thuật
  - Xét trạng thái bắt đầu
    - Nếu là đích ⇒ dừng
    - Ngược lại: thiết lập TT bắt 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 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)

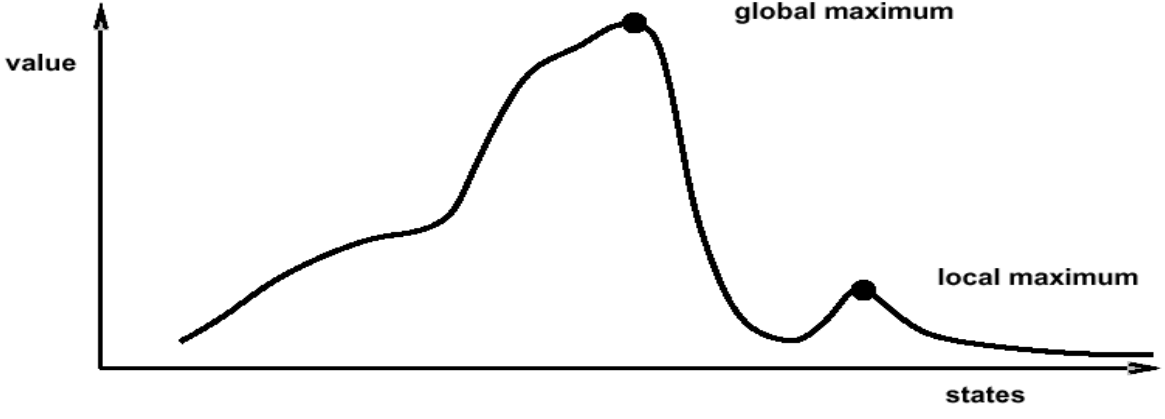
```
function HILL-CLIMBING( problem) returns a state that is a local maximum
    inputs: problem, a problem
    local variables: current, a node
                     neighbor, a node

    current ← MAKE-NODE(INITIAL-STATE[problem])
    loop do
        neighbor ← a highest-valued successor of current
        if VALUE[neighbor] ≤ VALUE[current] then return STATE[current]
        current ← neighbor
```

Tìm kiếm leo đồi (tt)



Tìm kiếm leo đồi (tt)

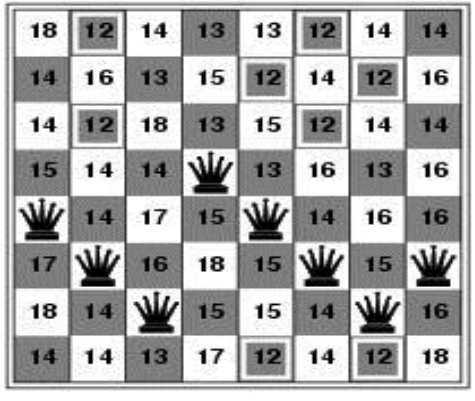
- Hiệu quả nếu có được hàm (H) đánh giá tốt
- Giới hạn
  - Có khuynh hướng bị sa lầy ở những cực đại cục bộ
    - Lời giải tìm được không tối ưu
    - Không tìm được lời giải mặc dù có tồn tại lời giải
  - Có thể gặp vòng lặp vô hạn do không lưu giữ thông tin về các trạng thái đã duyệt

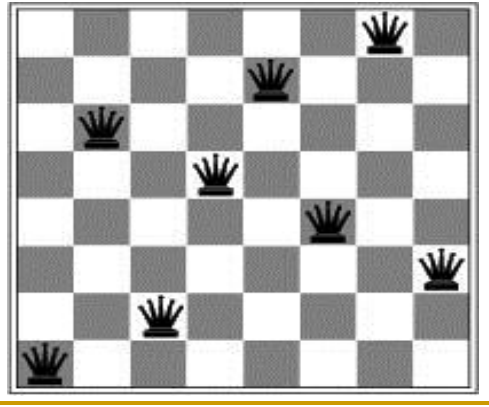


Tìm kiếm leo đồi (tt)

- 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

Main Program

```
Function best_first_search
    open := [Start]
    closed := [ ]
    while open ≠ [ ]
        remove leftmost state from open, call it X
        if X is a goal then return SUCCESS
        else visit(X)
        put X on closed
        sort open by heuristic merit (best leftmost)
    return FAIL
End function
```

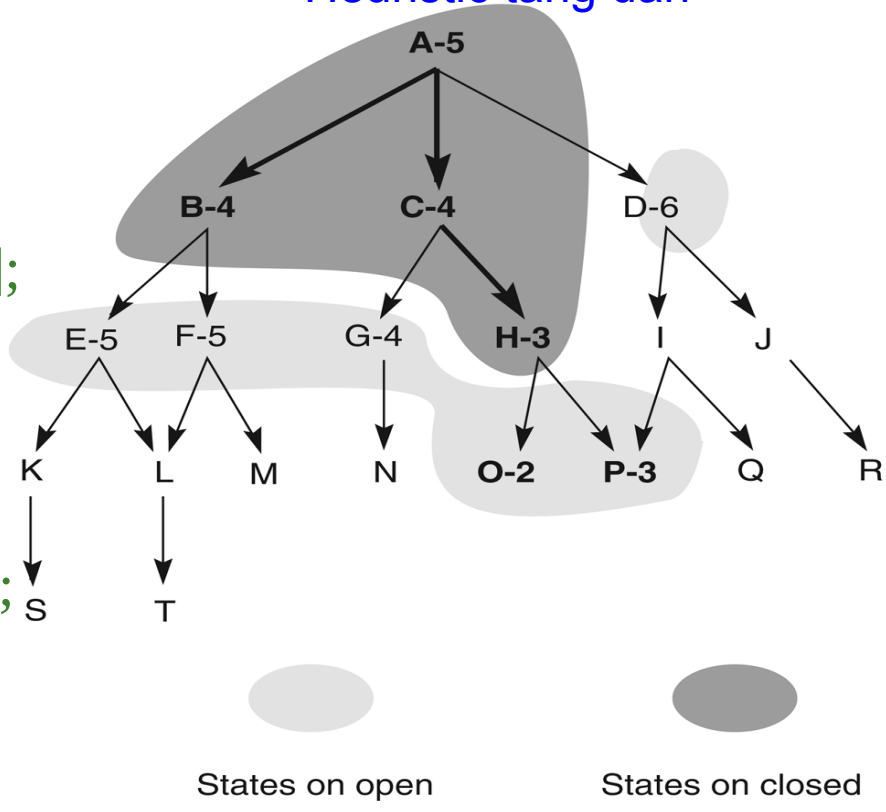
Process children of X

```
Subroutine visit(X)
    generate children of X
    for each child of X
        case  % 2 cases
            the child is not on open or closed: child has never been seen before
                assign the child a heuristic value
                add the child to open
            the child is already on open: child is waiting
                if the child was reached by a shorter path
                then give the state on open the shorter path
    End Subroutine
```

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



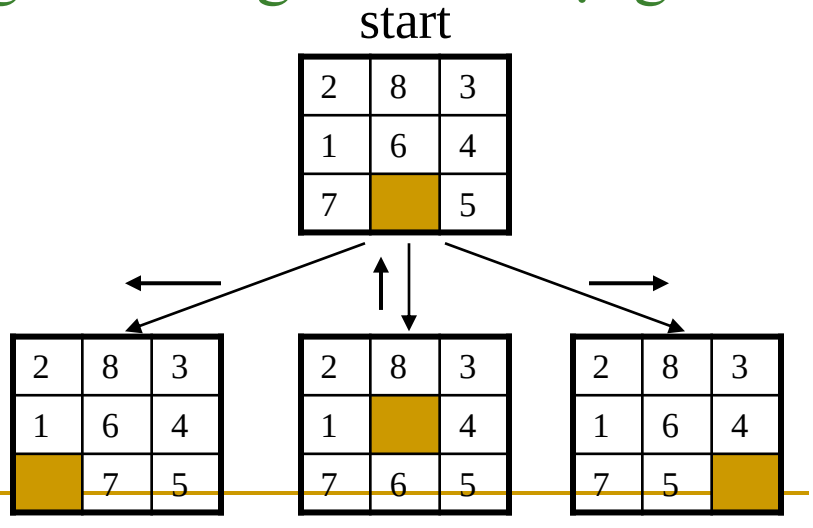
Cài đặt hàm đánh giá (EF)

- 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.

1	2	3
8		4
7	6	5

goal

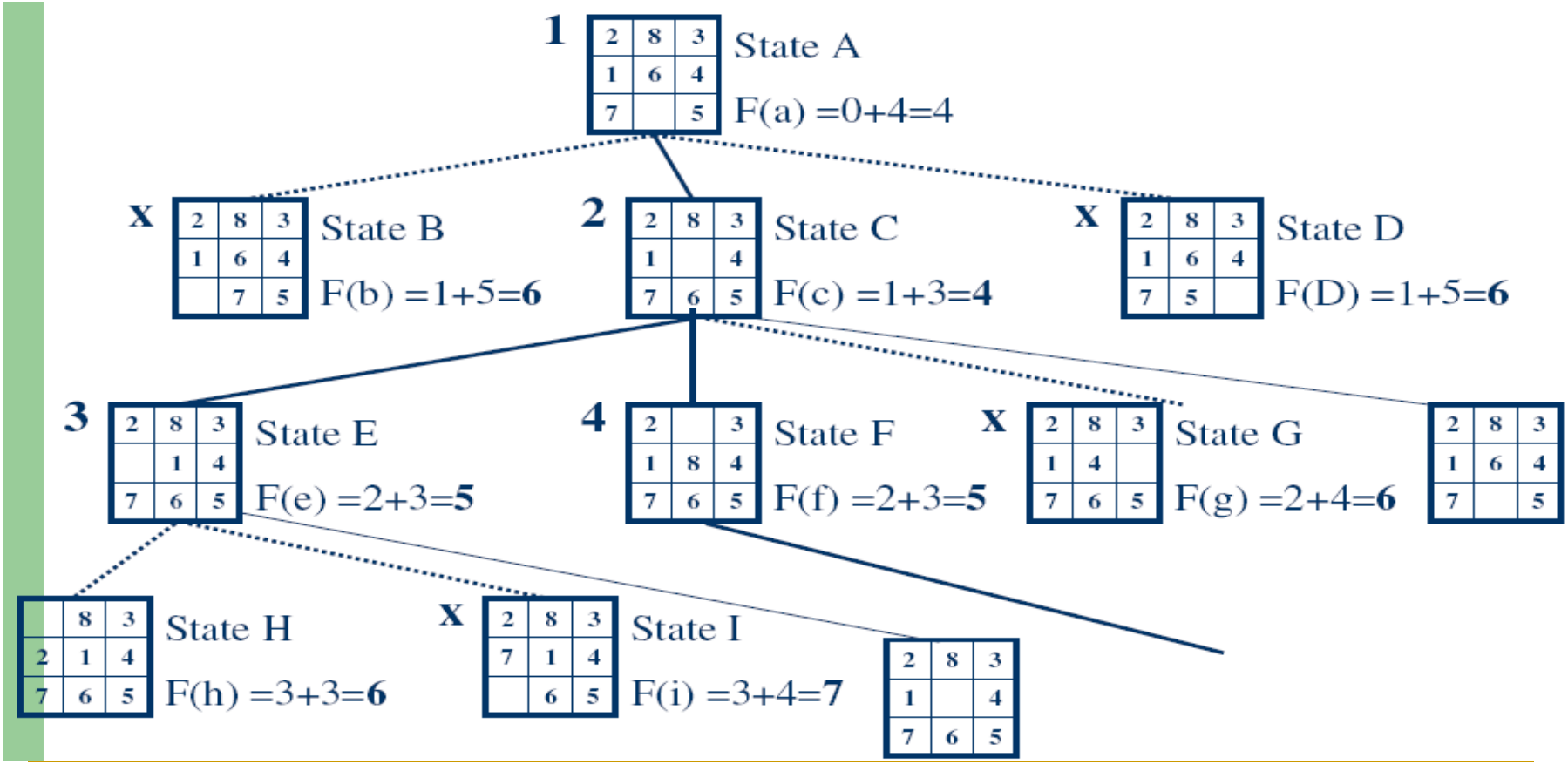
g(n) = 0



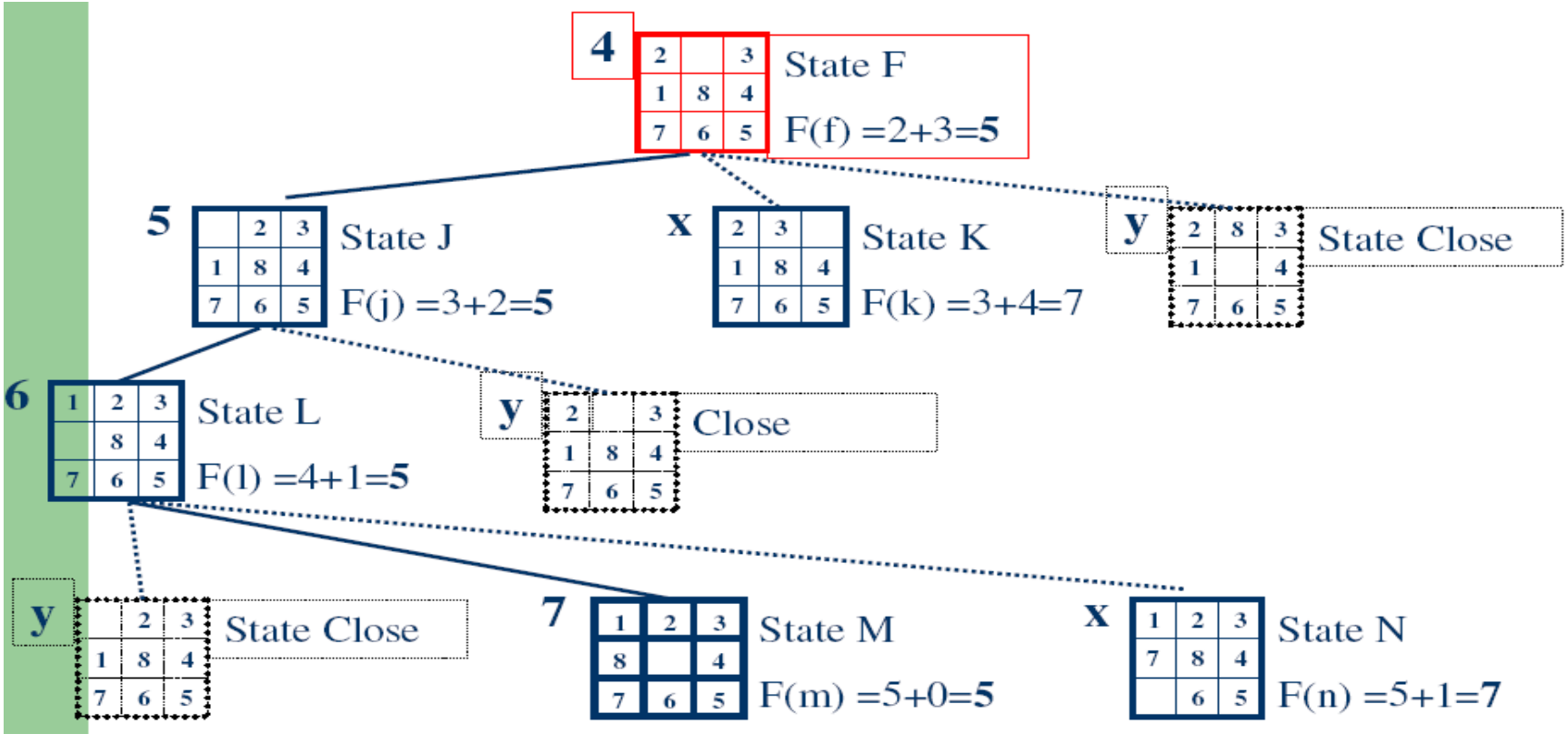
h(n): số lượng các vị trí còn sai; g(n) = 1

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

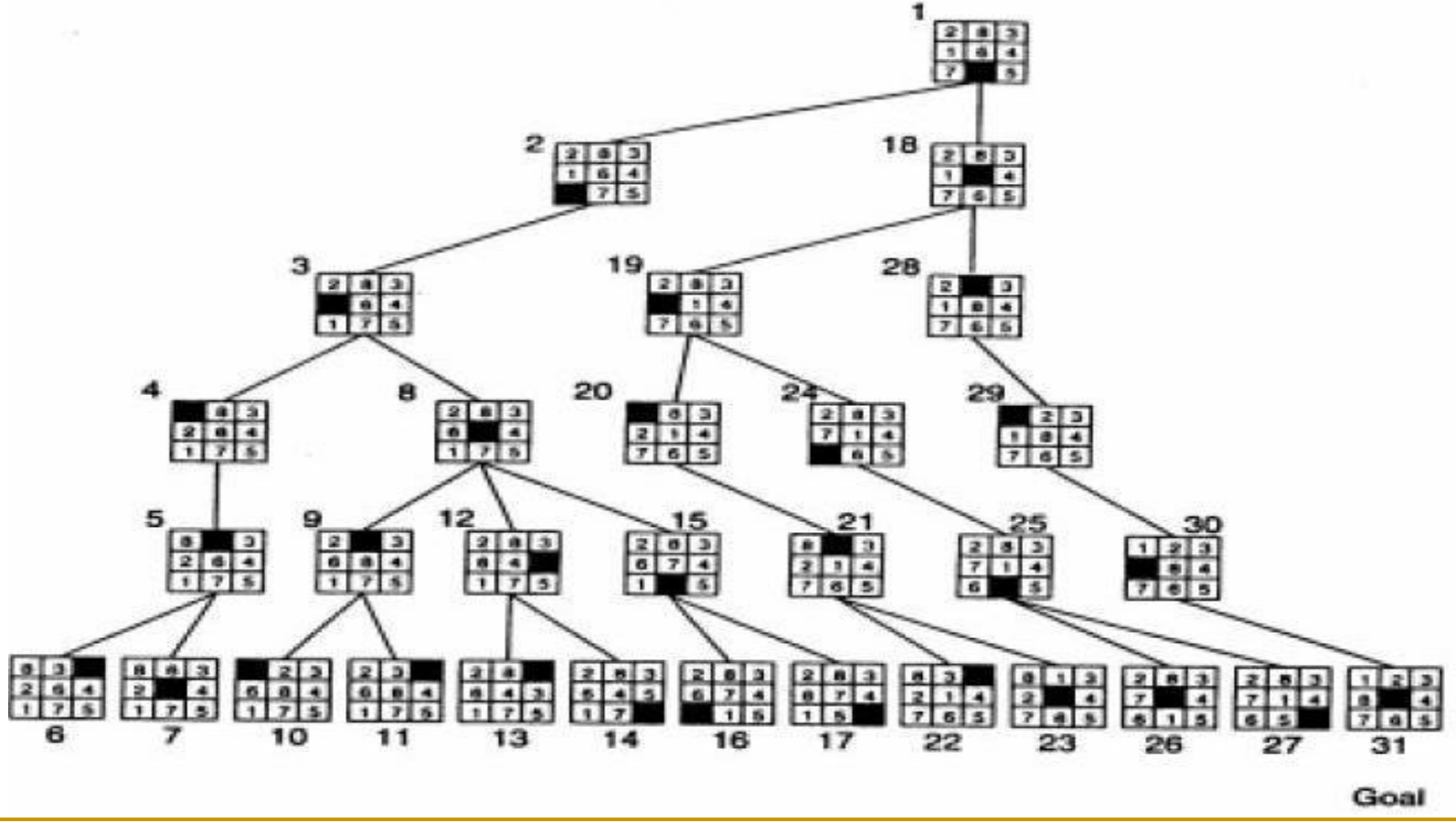
Ví dụ


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

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

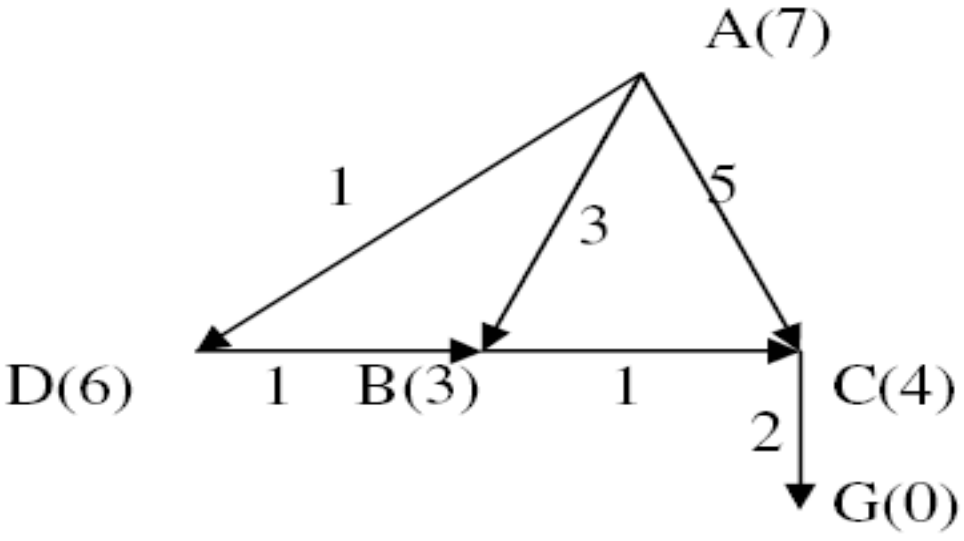
Thông tin mỗi nút

- Mỗi trạng thái n 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(0,7,7,-)	
2	B(3,3,6,A), D(1,6,7,A), C(5,4,9,A)	A(0,7,7,-)
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

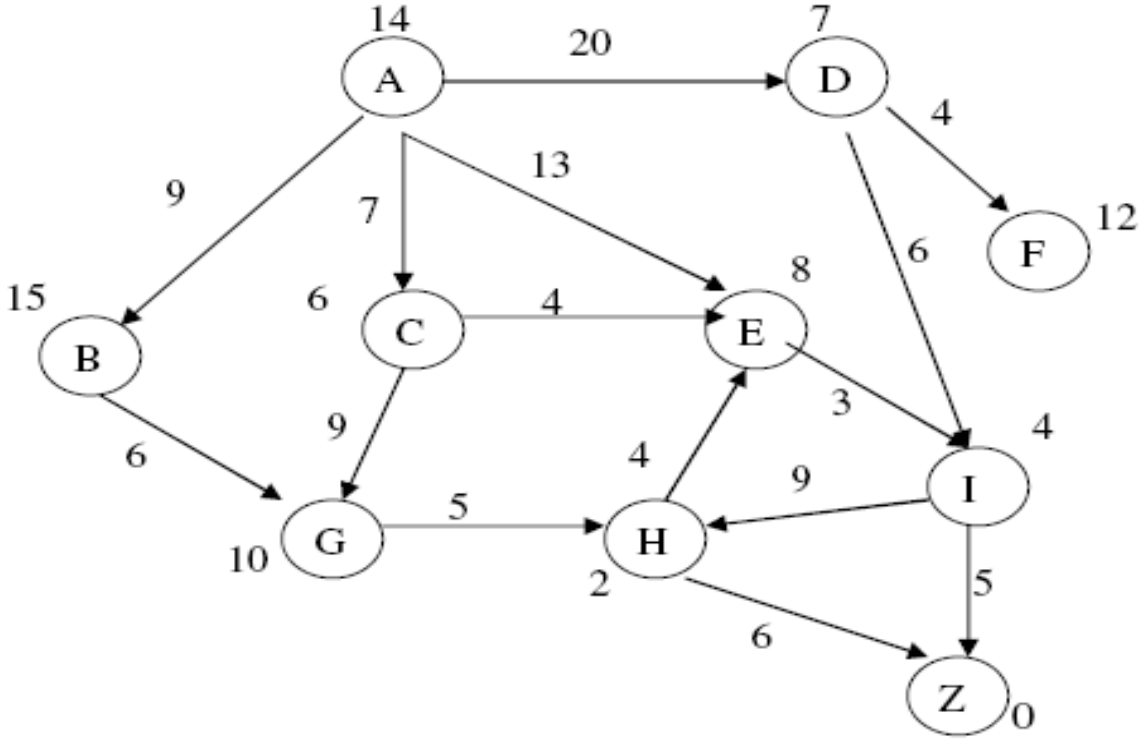
Chi tiết các bước

- 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 nhưng đường đi mới qua D đến B ngắn hơn nên B phải được 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 **then**
                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') **then**
                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') **then**
                If g(m)<g(m') **then** f(m)=g(m)+h(m); cha(m)=n;
                                Đưa m vào open; loại m' khỏi closed;
        Sắp xếp open để t.thái tốt nhất nằm bên trái;

Ví dụ về A*



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

Các link minh họa tìm kiếm

- 8-Puzzle với GT TK A*
  - http://www.cs.rmit.edu.au/AI-Search/Product (RMIT)
- Map Search với GT TK leo đồi, tốt nhất, A*
  - http://www.ai.mit.edu/courses/6.034f/searchpair.html

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

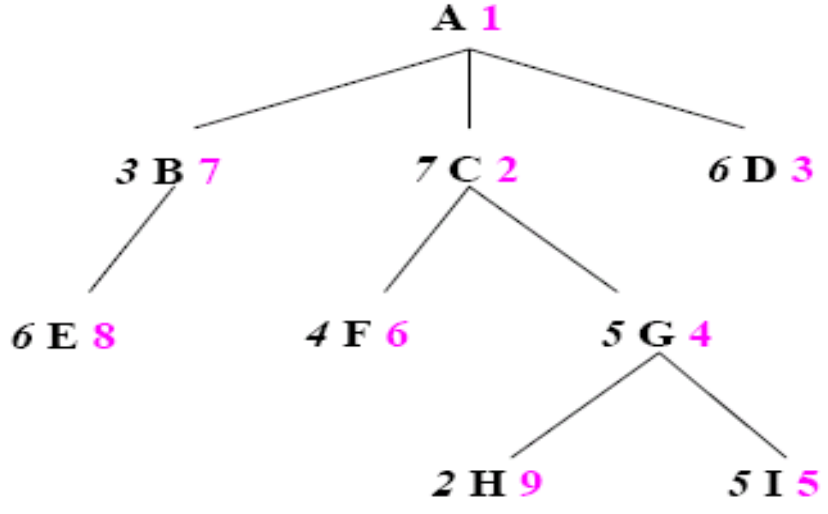
b) Theo giải thuật tìm kiếm rộng

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 t



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*