NHẬP MÔN TRÍ TUỆ NHÂN TẠO

CHƯƠNG 2

GIẢI QUYẾT VẤN ĐỀ BẰNG TÌM KIẾM

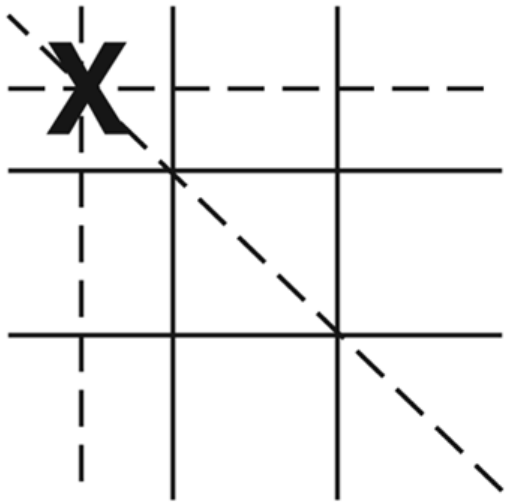
Nội dung

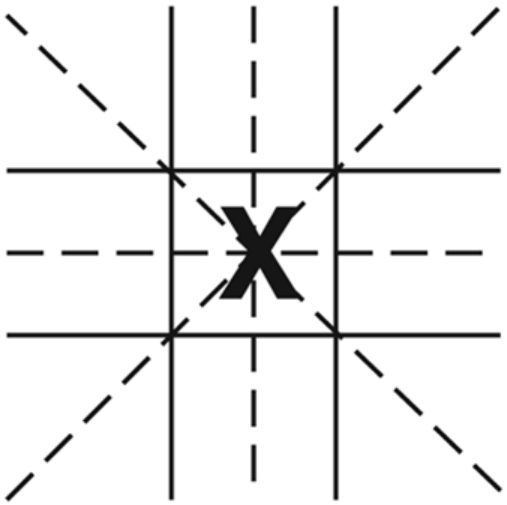
- Biểu diễn bài toán trong KGTT
- Tìm kiếm mù (uninformed search)
- Tìm kiếm heuristic (informed search)
- Cây trò chơi, cắt tỉa alpha –beta
- Bài toán thoả mãn ràng buộc

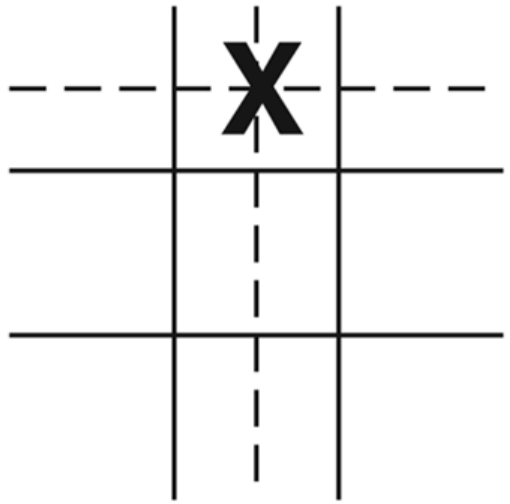
TÌM KIẾM DỰA TRÊN KINH NGHIỆM (INFORMED/ HEURISTIC SEARCH)

Tìm kiếm có thông tin

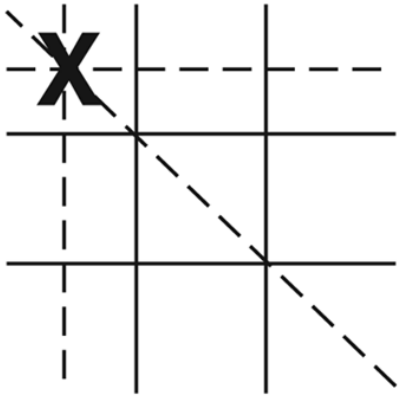
- **Phép đo Heuristic**
- **Tìm kiếm tốt nhất đầu tiên (Best-First-Search - BFS)**
- **Tìm kiếm háu ăn (Greedy Best-First Search)**
- Tìm kiếm A*
- Tìm kiếm leo đồi



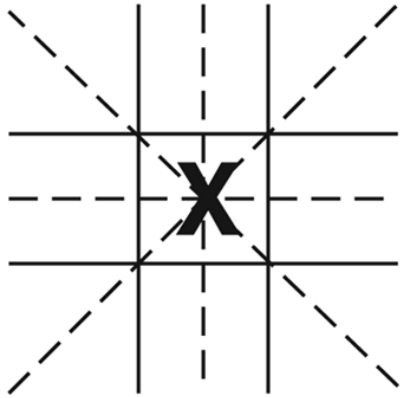




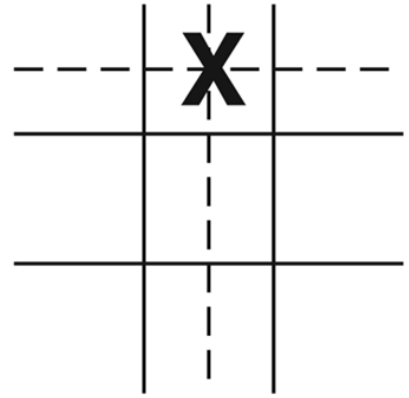
Phép đo heuristic (2)



Three wins through
a corner square



Four wins through
the center square



Two wins through
a side square

Heuristic "*Số đường thắng nhiều nhất*" áp dụng cho các nút con đầu tiên trong tic-tac-toe.

Heuristic

- Để giải quyết các bài toán lớn, phải cung cấp những kiến thức đặc trưng ở từng lĩnh vực để nâng cao hiệu quả của việc tìm kiếm.

- Trong TK KGTT, heuristic là các luật dùng để chọn những **nhánh/ lựa chọn nào có nhiều khả năng nhất** dẫn đến một giải pháp chấp nhận được.

Heuristic

Sử dụng Heuristic trong trường hợp nào?

- Vấn đề có thể có giải pháp chính xác, nhưng **chi phí tính toán** để tìm ra nó không cho phép.

  Ví dụ: cờ vua,...
- Vấn đề có thể **không có giải pháp chính xác** vì sự không rõ ràng trong diễn đạt vấn đề hoặc trong các dữ liệu có sẵn.

  Ví dụ: chẩn đoán y khoa,...

Heuristic

- Thuật toán Heuristic gồm hai phần:
  1. ***Phép đo Heuristic:*** *thể hiện qua hàm đánh giá heuristic, dùng để đánh giá các đặc điểm của một trạng thái trong KGTT.*
  2. ***Giải thuật tìm kiếm heuristic:***
     - Tìm kiếm tốt nhất đầu tiên (best-first search)
       - *Tìm kiếm háu ăn (Greedy best-first search)*
       - *Giải thuật A**
     - Tìm kiếm leo đồi

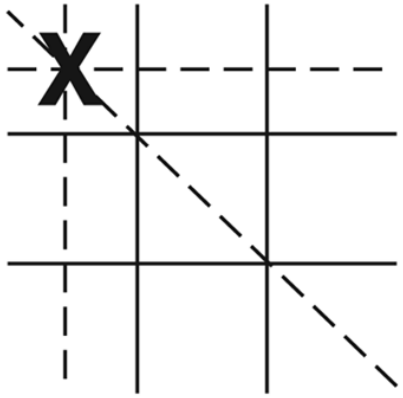
Heuristic

- **Phép đo Heuristic:**
  - *Ước lượng chi phí tối ưu giữa hai hoặc nhiều giải pháp*
  - *Không quá tốn kém để tính toán*
  - *Được xác định cụ thể trong từng bài toán (Ví dụ: đối với bài toán TSP, đánh giá khoảng cách giữa các thành phố sao cho chi phí là thấp nhất)*
- **Hàm đánh giá Heuristic** tại trạng thái n là f(n):

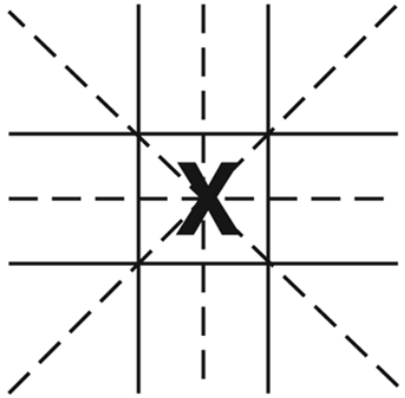
$$f(n) = g(n) + h(n)$$

  - *g(n) = khoảng cách thực sự từ* ***n đến trạng thái bắt đầu***
  - *h(n) = ước lượng heuristic cho khoảng cách từ trạng thái* ***n đến trạng thái đích***

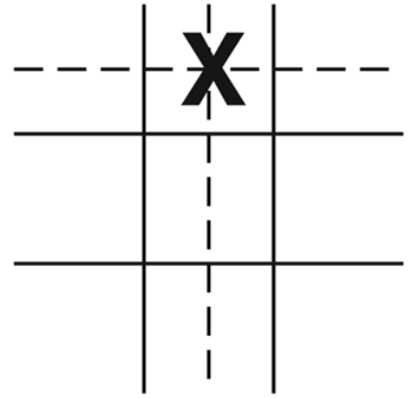
Phép đo heuristic (2)



Three wins through
a corner square



Four wins through
the center square



Two wins through
a side square

Heuristic "*Số đường thắng nhiều nhất*" áp dụng cho các nút con đầu tiên trong tic-tac-toe.

Cài đặt hàm đánh giá Heuristic

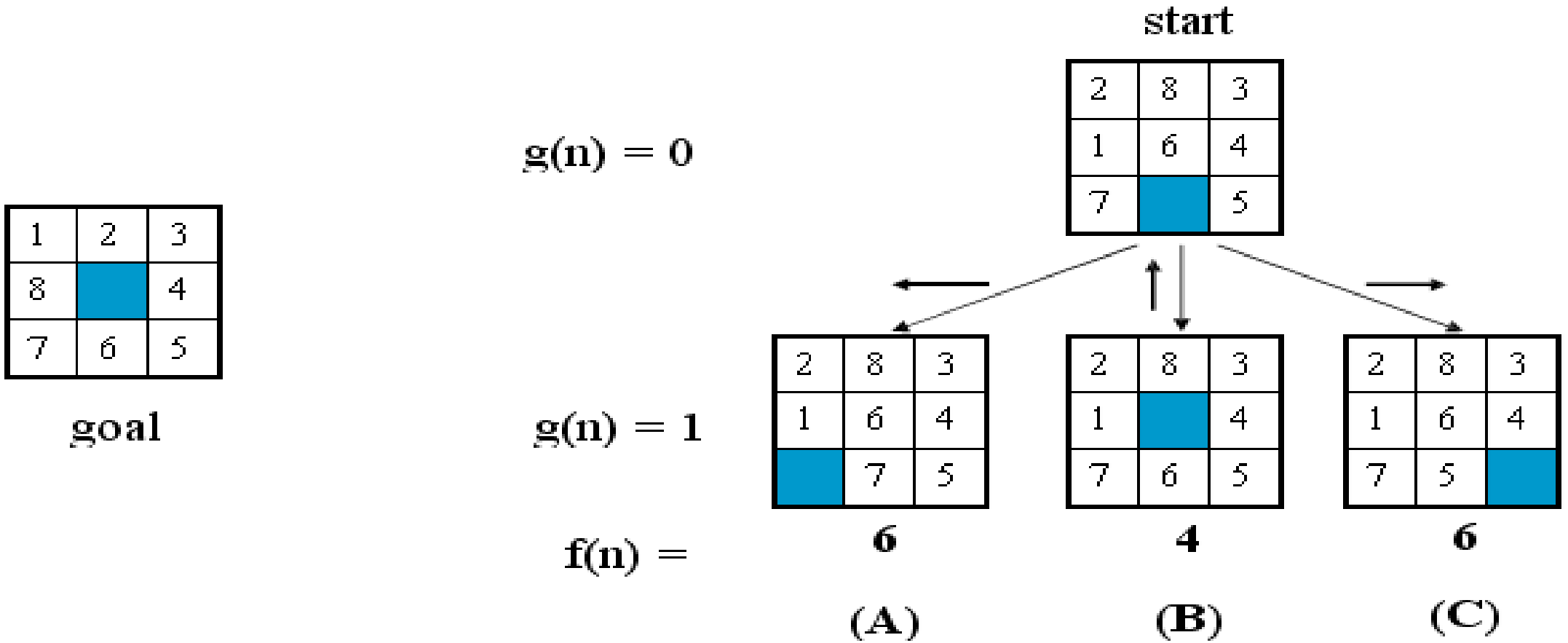
- Xét trò chơi 8-puzzle. Hàm đánh giá Heuristic tại trạng thái n là f(n):

$$\mathbf{f(n) = g(n) + h(n)}$$

  - *g(n) = khoảng cách thực sự từ* ***n đến trạng thái bắt đầu***
  - *h(n) = ước lượng heuristic cho khoảng cách từ trạng thái* ***n đến trạng thái đích***

Cài đặt hàm đánh giá Heuristic

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



Cài đặt hàm đánh giá Heuristic

- *Việc ước lượng heuristic cho khoảng cách từ trạng thái n đến trạng thái đích như thế nào?*
- *Giả sử có các định nghĩa sau:*
  - $h_1(n)$ = số vị trí sai khác của trạng thái n so với goal
  - $h_2(n)$ = khoảng cách dịch chuyển (←,→,↑,↓) ngắn nhất để dịch chuyển các ô chữ nằm sai vị trí về vị trí đúng (khoảng cách Manhattan)

=> $h_1(n) = ???$

=> $h_2(n) = ???$

START

GOAL

Cài đặt hàm đánh giá Heuristic

- *Việc ước lượng heuristic cho khoảng cách từ trạng thái n đến trạng thái đích như thế nào?*
- *Giả sử có các định nghĩa sau:*
  - $h_1(n)$ = số vị trí sai khác của trạng thái n so với goal
  - $h_2(n)$ = khoảng cách dịch chuyển (←,→,↑,↓) ngắn nhất để dịch chuyển các ô chữ nằm sai vị trí về vị trí đúng (khoảng cách Manhattan)

=> $\boldsymbol{h_1(n) = 6}$

=> $\boldsymbol{h_2(n) = 8}$

START

GOAL

Tìm kiếm tốt nhất đầu tiên (Best-first-search)

- Ý tưởng: sử dụng hàm đánh giá trong quá trình phát triển cây tìm kiếm, hàm đánh giá ước lượng độ gần của mỗi đỉnh trạng thái với trạng thái đích, chỉ triển khai cây tìm kiếm theo nhánh có triển vọng đi đến đích nhanh nhất.
- *Tìm kiếm tốt nhất đầu tiên* (best-first search) là tiếp cận tổng quát của tìm kiếm với thông tin bổ sung.
- Việc chọn nút để triển khai dựa trên một *hàm lượng giá* (evaluation function): f(n)

Tìm kiếm tốt nhất đầu tiên (Best-first-search)

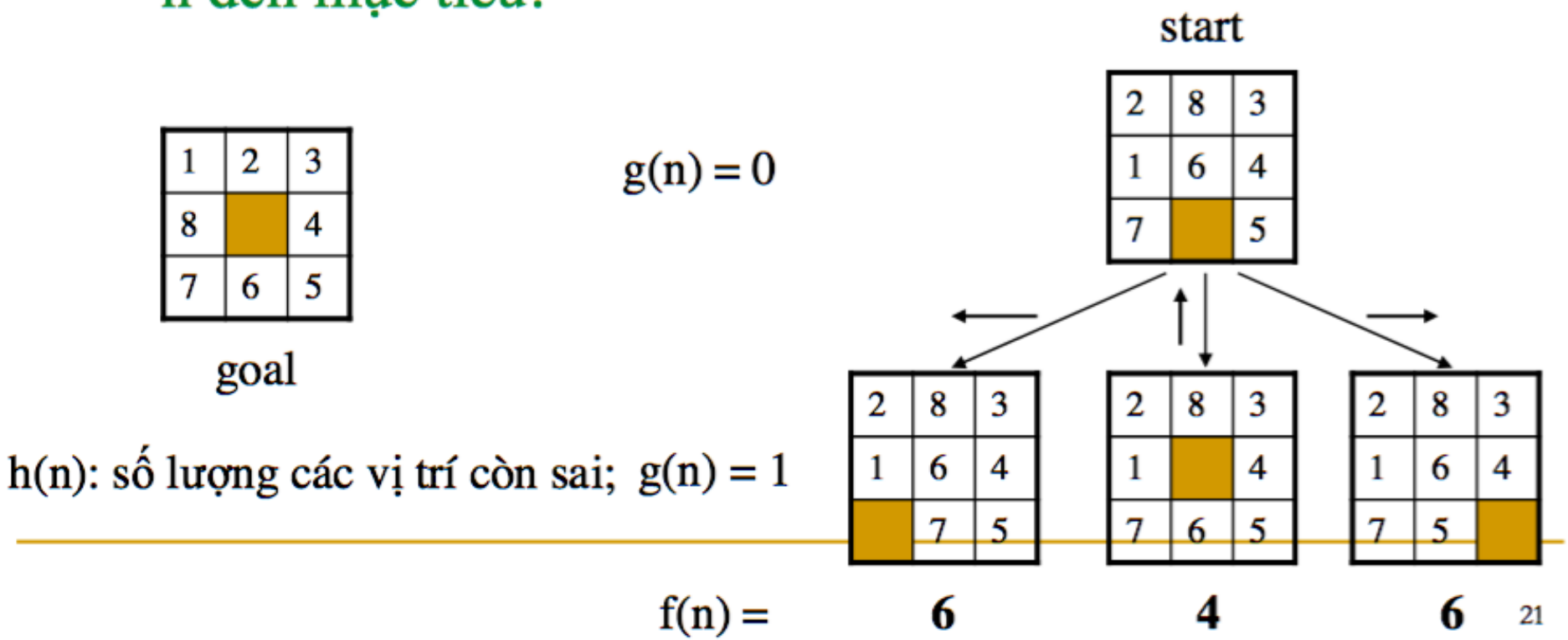
- Hàm lượng giá (evaluation function): f(n) được xây dựng dựa trên việc ước lượng chi phí và nút có giá ước lượng nhỏ nhất được ưu tiên triển khai trước. Sự lựa chọn hàm sẽ quyết định chiến lược tìm kiếm.
- Hàm h(n) là *chi phí ước lượng* của đường đi tốt nhất từ trạng thái của nút đến trạng thái mục tiêu.
  - *f(n) = g(n) +h(n)*
  - *g(n): chi phí thực tế đi từ gốc đến n*
  - *h(n): chi phí ước lượng đi từ n đến nút mục tiêu*
- Hàm heuristic là cách thường dùng nhất để sử dụng kiến thức bổ sung trong quá trình tìm kiếm: h(n) là một hàm không âm bất kỳ và *phụ thuộc vào vấn đề đang giải quyết*. Nếu là nút mục tiêu thì h(n) = 0.

Tìm kiếm tốt nhất đầu tiên (Best-first-search)

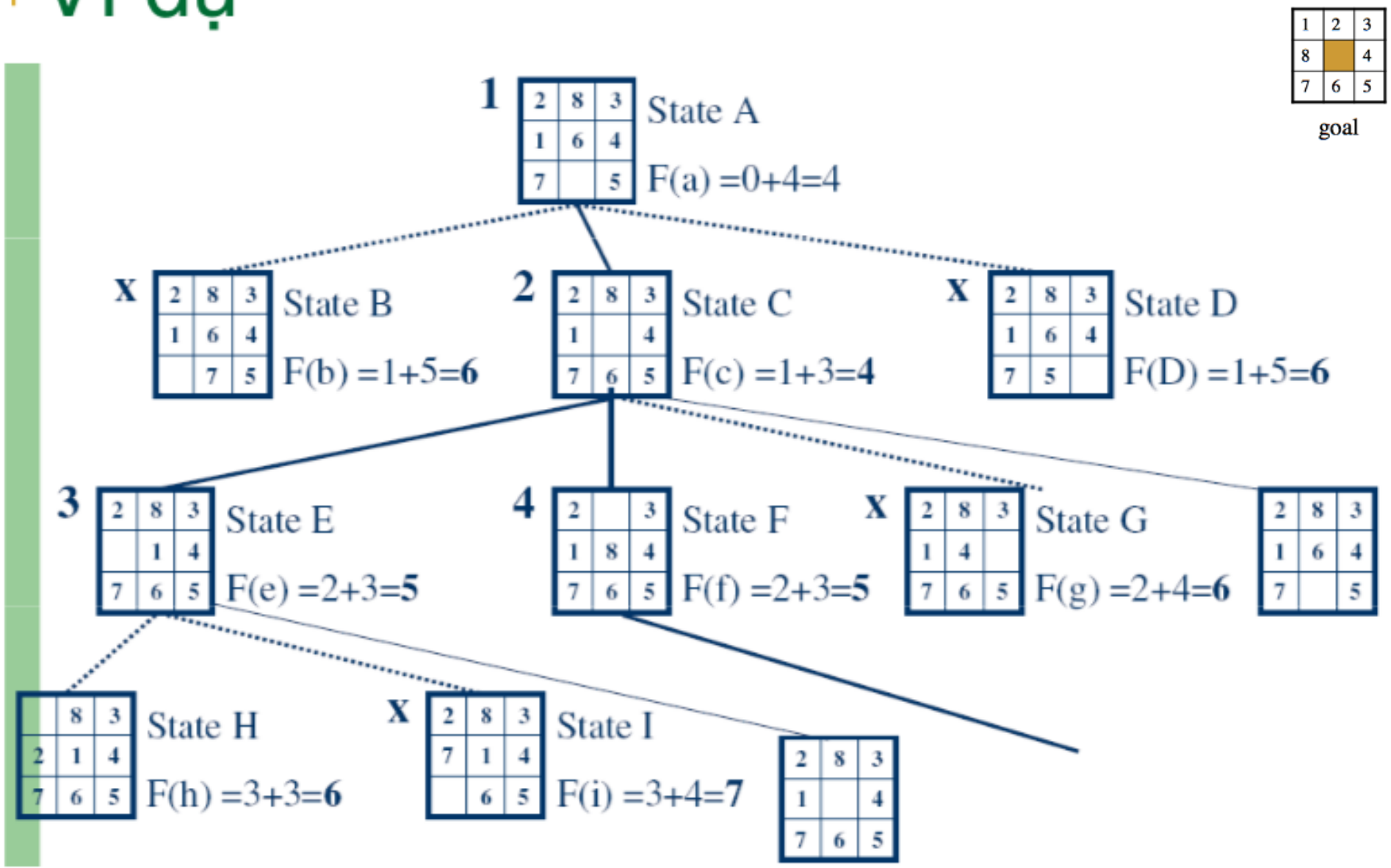
- **Dùng mảng có sắp xếp theo hàm đánh giá**
- Tương tự DFS và BFS, best-first search dùng các danh sách để lưu trữ các trạng thái:
  - *OPEN: lưu các trạng thái sắp được kiểm tra*
  - *CLOSED: lưu các trạng thái đã duyệt qua*
- **Các trạng thái trong OPEN list sẽ được sắp xếp theo thứ tự dựa trên 1 hàm Heuristic nào đó (đặt thứ tự ưu tiên cho các giá trị gần trạng thái đích)**
- Do đó, mỗi lần lặp sẽ xem xét trạng thái tiềm năng nhất trong OPEN list

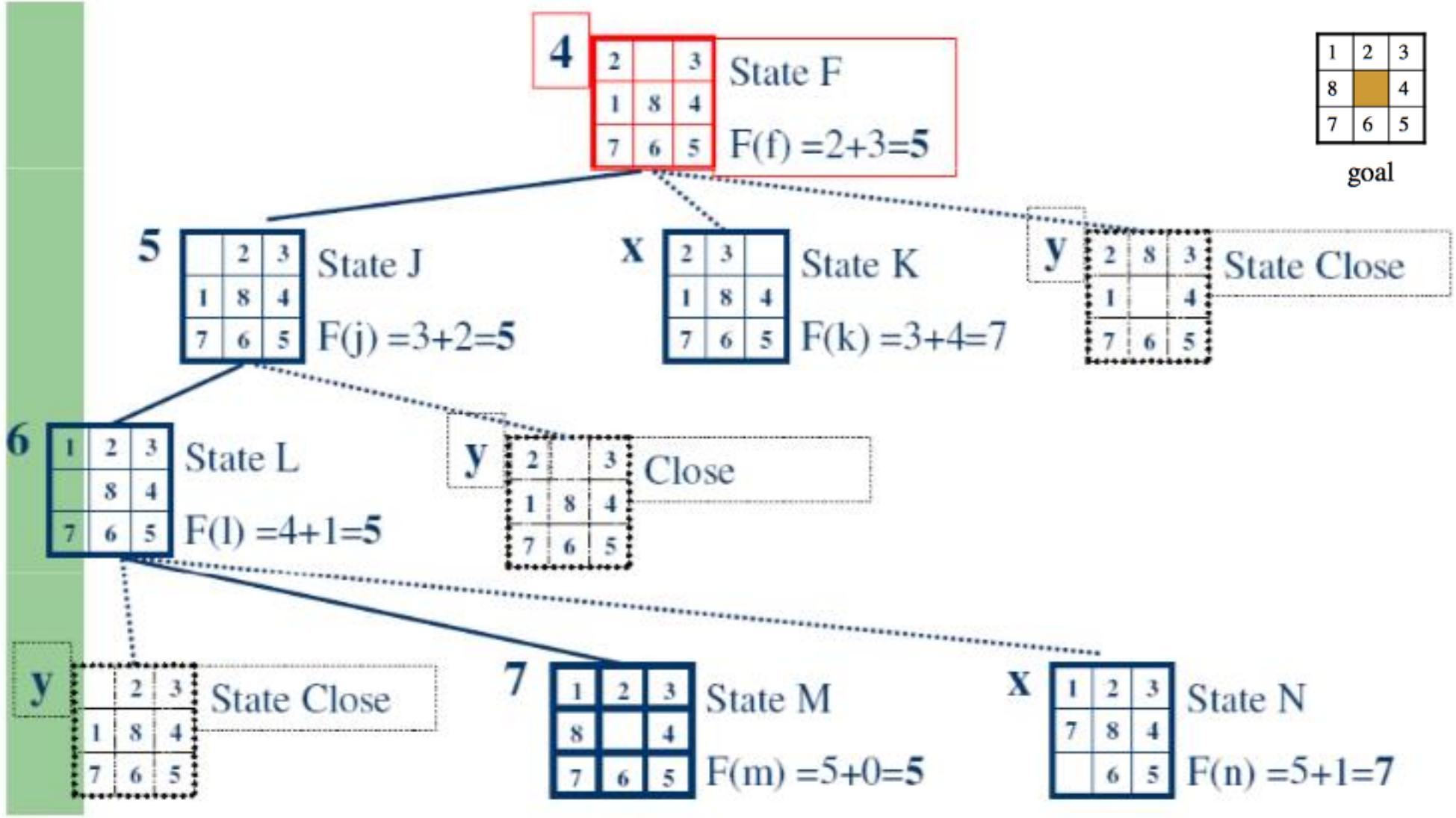
Sử dụng **tìm kiếm tốt nhất đầu tiên** để tìm trạng thái **goal**

- 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.



Ví dụ


goal
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


4
State F
F(f) =2+3=5
goal
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 minh họa

Cho bài toán sau:

Bài toán gồm một bảng 2×2 với các ô chữ được hiển thị như hình và một ô trống. Ở trạng thái bắt đầu (Initial State), các ô được sắp đặt như hình bên dưới, và nhiệm vụ của người giải là tìm cách đưa chúng về đúng thứ tự như minh họa dưới (Goal State).

- Với việc định nghĩa hàm Heuristic h(n) như sau:
"Số vị trí sai khác của trạng thái n so với trạng thái Goal"
- Chi phí mỗi lần di chuyển ô trống (lên, xuống, trái, phải) là 1
Hãy sử dụng giải thuật Best-First-Search để tìm đường đi lời giải cho bài toán.

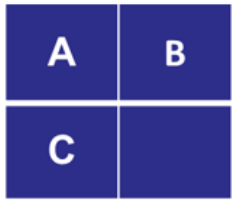
Initialization

C	A
B	

(1)

OPEN (1)

CLOSE

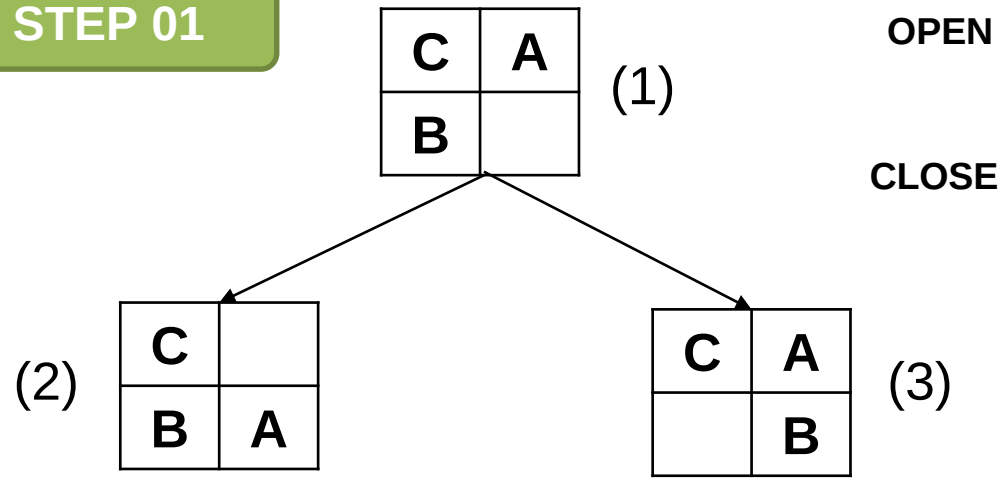


Goal State

h(n) = "Số vị trí sai khác của trạng thái n so với trạng thái Goal"

Thứ tự: Up, Down, Left, Right

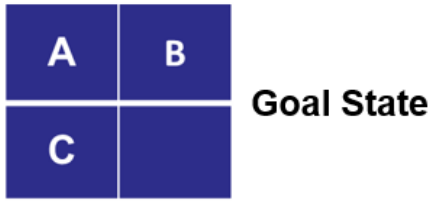
STEP 01



f(n) = h(n) + g(n)
= 4 + 1 = 5

f(n) = h(n) + g(n)
= 4 + 1 = 5

OPEN	(2 (f = 5)) (3 (f = 5))
CLOSE	(1)

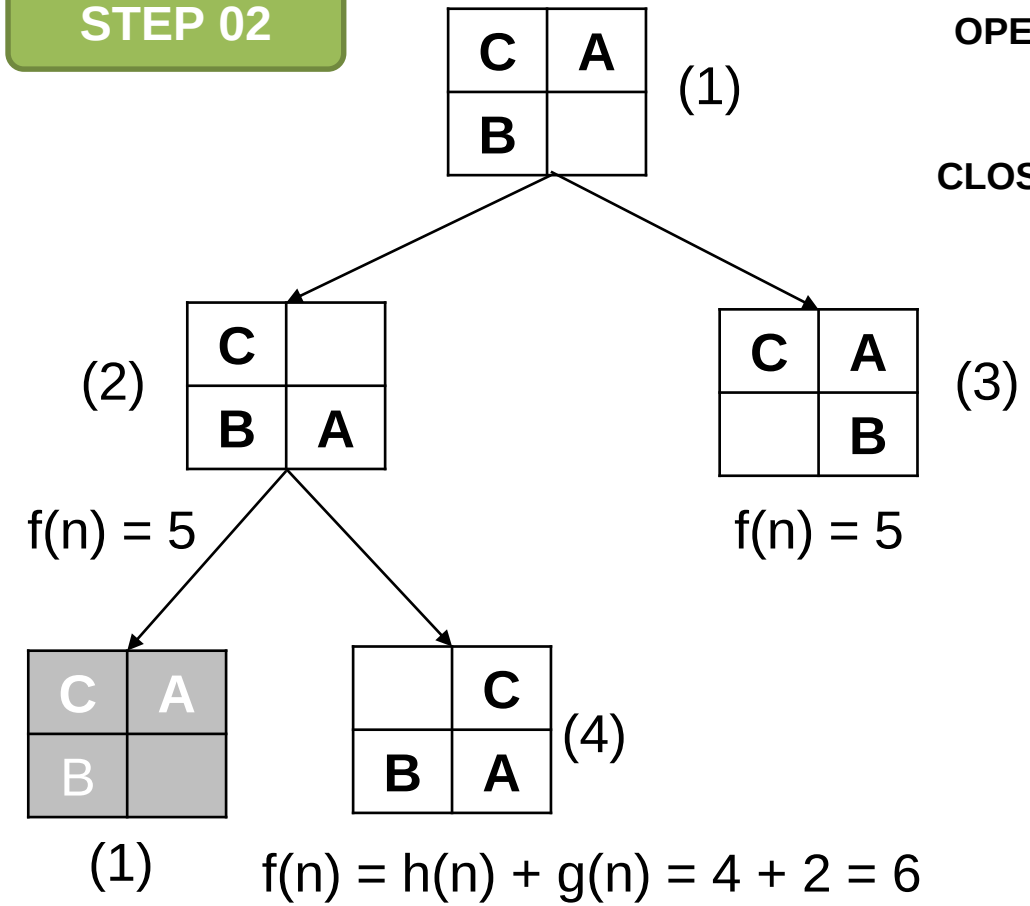


Goal State

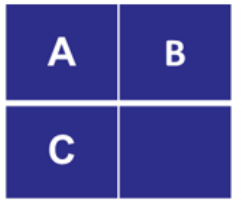
h(n) = "Số vị trí sai khác của trạng thái n so với trạng thái Goal"

Thứ tự: Up, Down, Left, Right

STEP 02



OPEN	(3 (f = 5)) (4 (f = 6))
CLOSE	(1) (2)

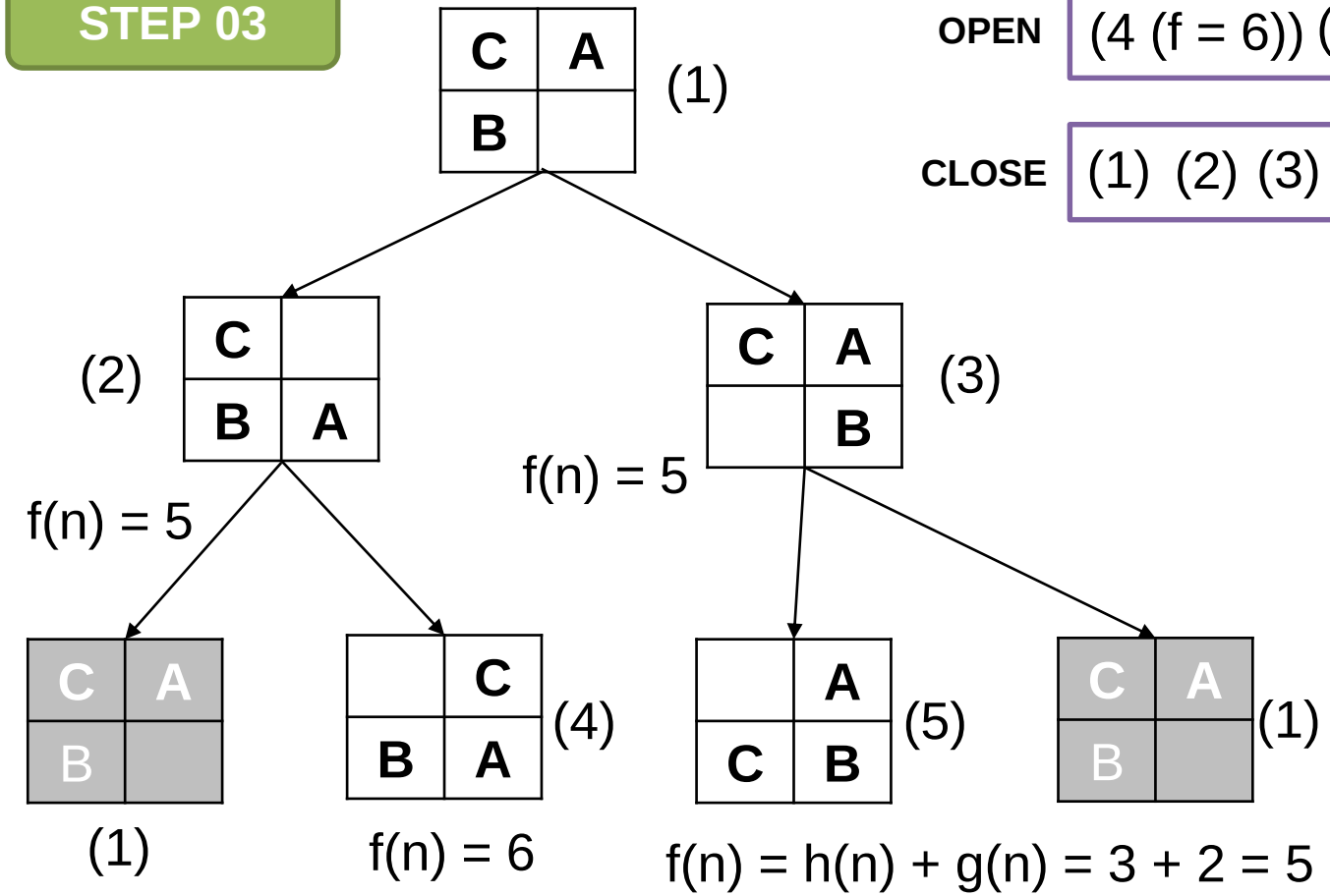


Goal State

h(n) = "Số vị trí sai khác của trạng thái n so với trạng thái Goal"

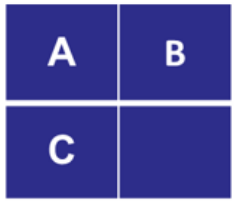
Thứ tự: Up, Down, Left, Right

STEP 03



OPEN | (4 (f = 6)) (5 (f = 5))

CLOSE | (1) (2) (3)

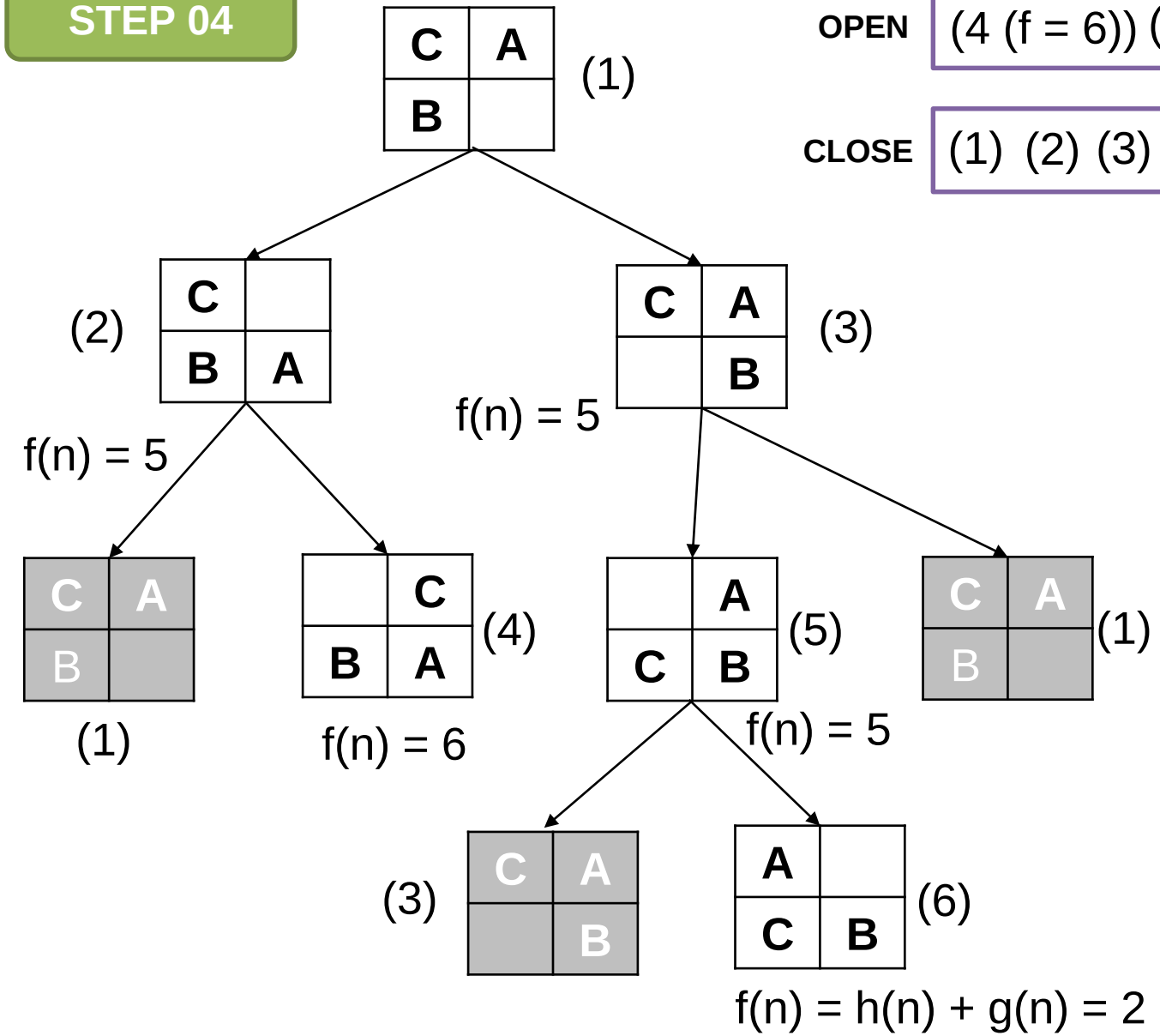


Goal State

h(n) = "Số vị trí sai khác của trạng thái n so với trạng thái Goal"

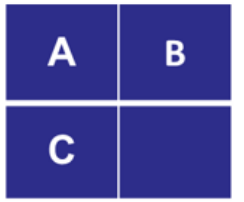
Thứ tự: Up, Down, Left, Right

STEP 04



f(n) = h(n) + g(n) = 2 + 3 = 5

OPEN	(4 (f = 6)) (6 (f = 5))
CLOSE	(1) (2) (3) (5)

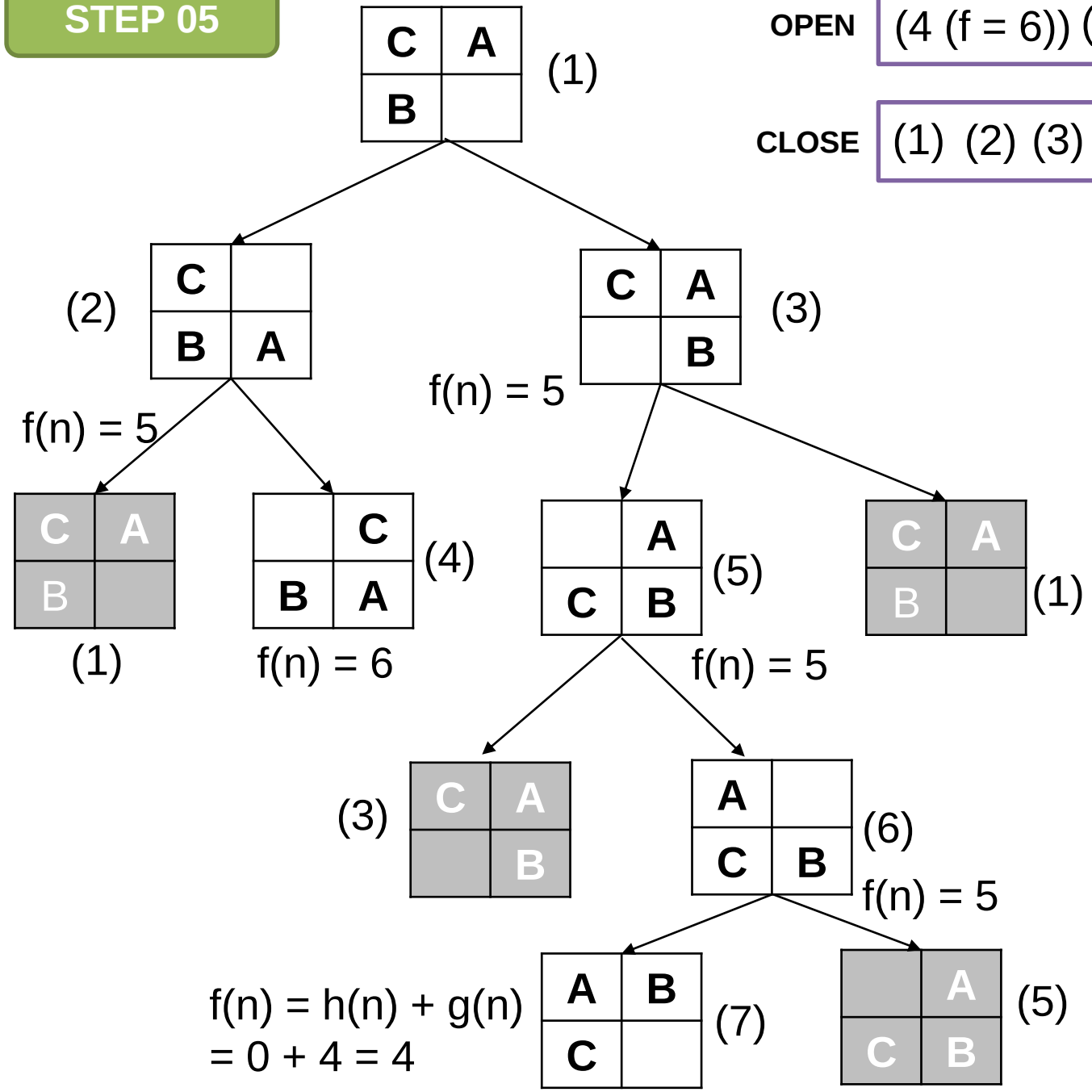


Goal State

h(n) = "Số vị trí sai khác của trạng thái n so với trạng thái Goal"

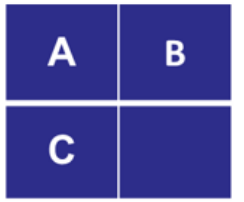
Thứ tự: Up, Down, Left, Right

STEP 05



OPEN (4 (f = 6)) (7 (f = 4))

CLOSE (1) (2) (3) (5) (6)



Goal State

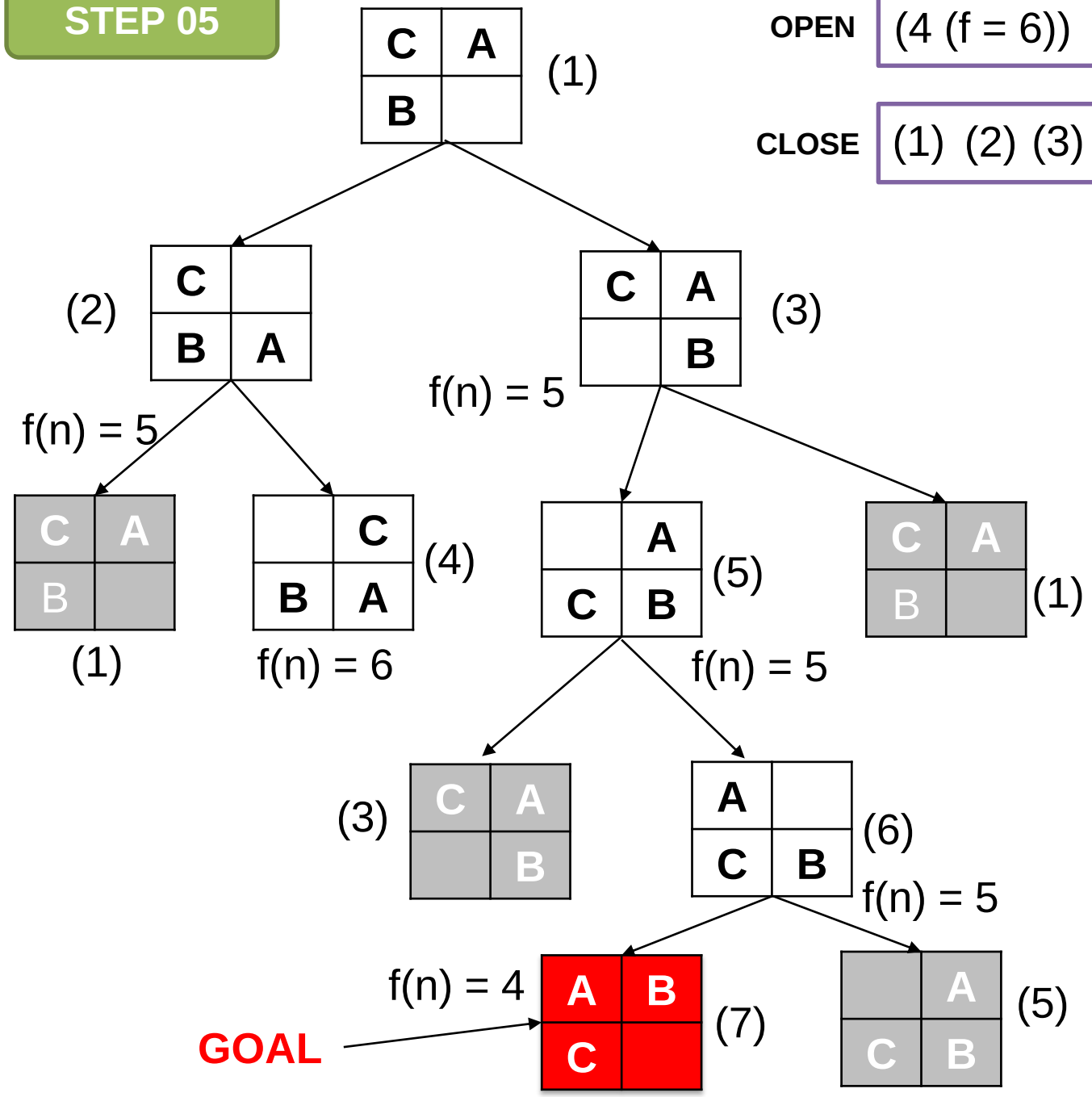
h(n) = "Số vị trí sai khác của trạng thái n so với trạng thái Goal"

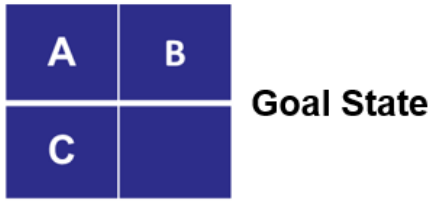
Thứ tự: Up, Down, Left, Right

STEP 05

OPEN (4 (f = 6))

CLOSE (1) (2) (3) (5) (6) (7)





h(n) = "Số vị trí sai khác của trạng thái n so với trạng thái Goal"

Thứ tự: Up, Down, Left, Right

Tìm kiếm háu ăn (Greedy Best-first Search)

- *Tìm kiếm háu ăn* (Greedy Best-first Search) hay còn gọi là *tìm kiếm chỉ sử dụng Heuristic* (Pure Heuristic Search) cố gắng triển khai nút "gần" với mục tiêu nhất.
- Vì thế, nó chỉ dùng hàm heuristic h(n) để lượng giá các nút, tức là

$$\mathbf{f(n) = h(n)}$$

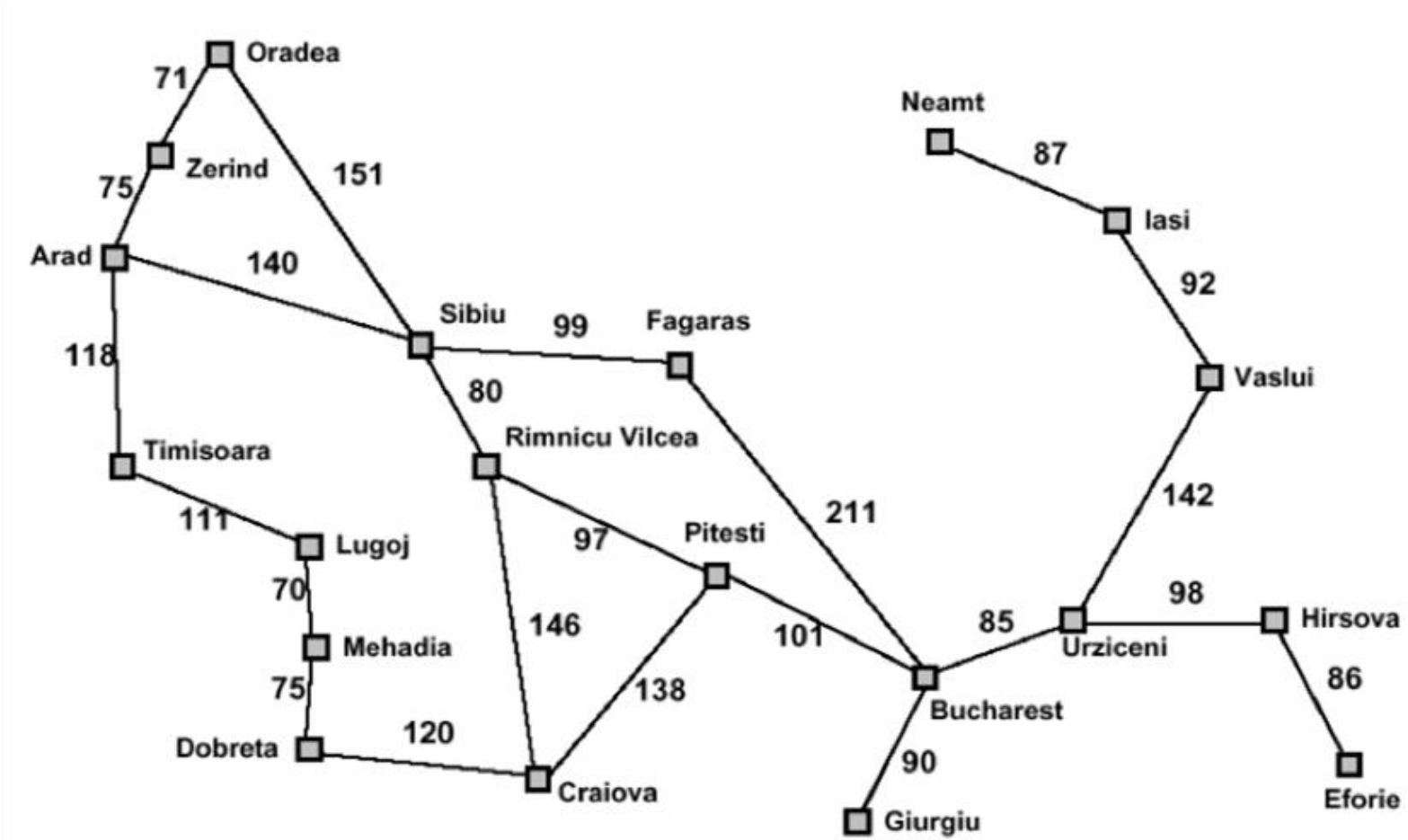
Tìm kiếm háu ăn (Greedy Best-first Search)

Bài toán tìm đường:

- Thành phố xuất phát: Arad
- Thành phố đích: Bucharest
- Các cạnh biểu diễn đường nối trực tiếp giữa hai thành phố, các con số ghi trên các cạnh là chi phí đi giữa hai thành phố.
- Cột bên phải là khoảng cách Euclide từ các thành phố đến thành phố đích Bucharest. **(h(n))**

Tìm kiếm háu ăn (Greedy Best-first Search)

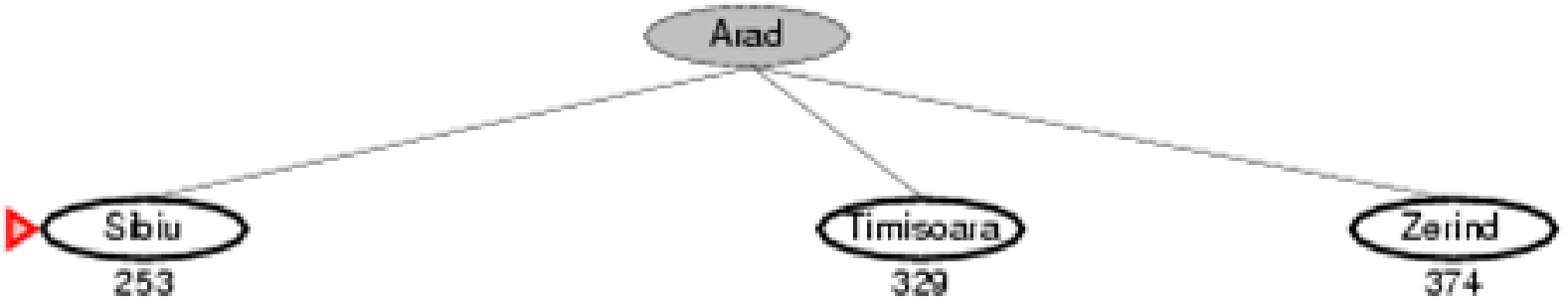
- Initial State = **Arad**
- Goal State = **Bucharest**



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

Tìm kiếm háu ăn

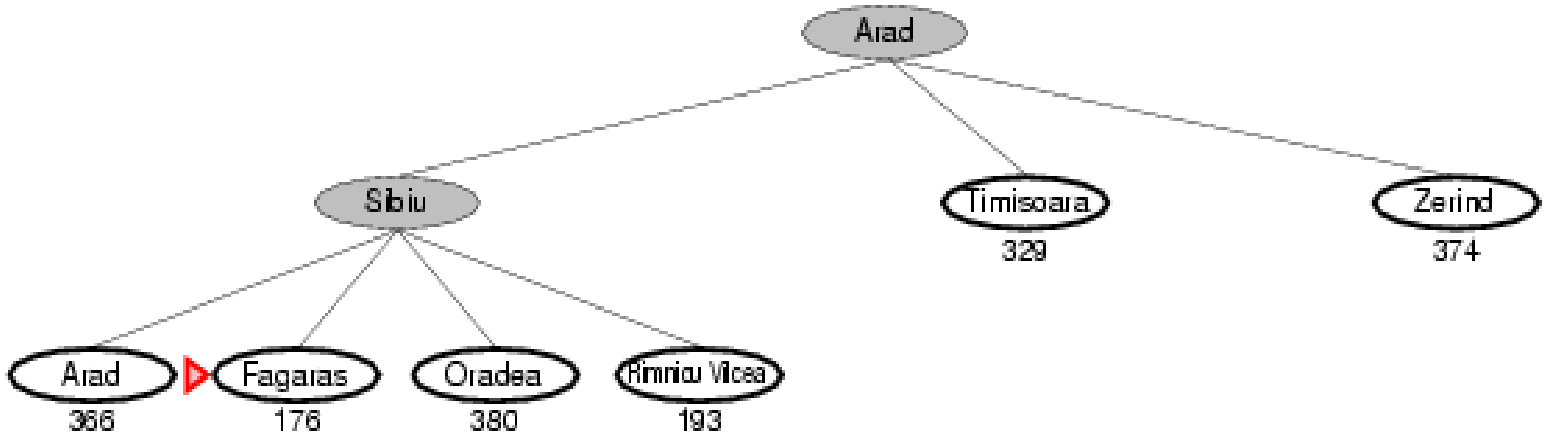
- Initial State = **Arad**
- Goal State = **Bucharest**



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

Tìm kiếm háu ăn

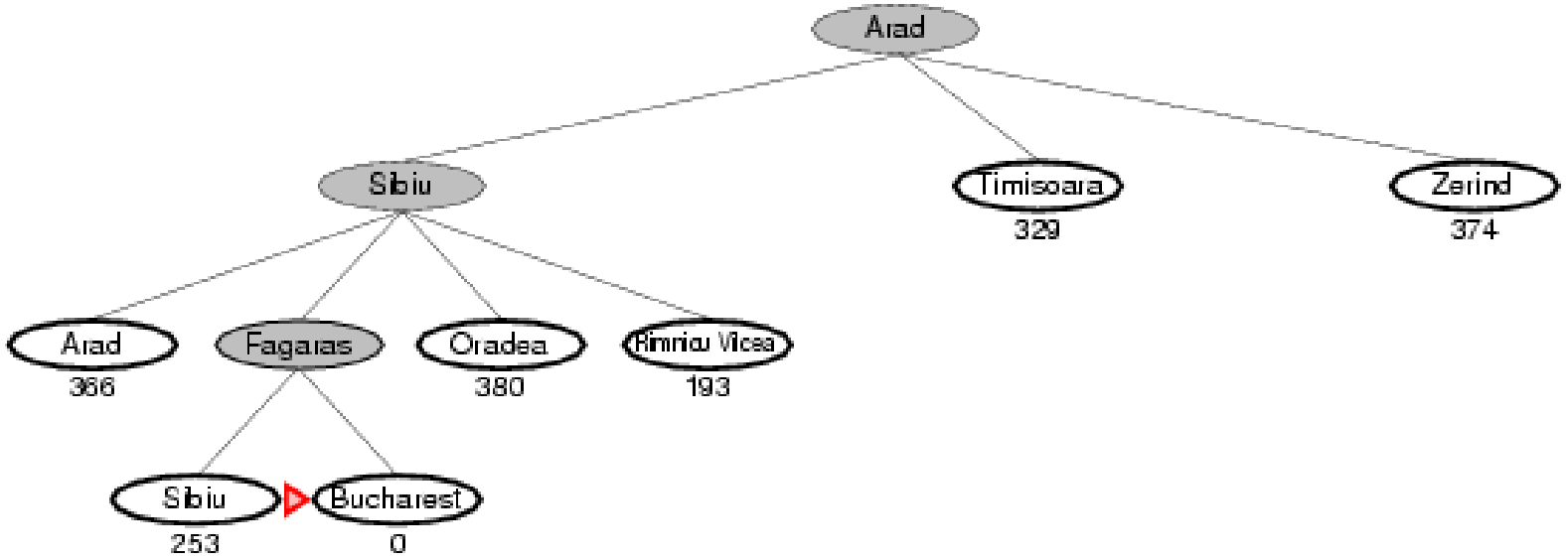
- Initial State = **Arad**
- Goal State = **Bucharest**



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

Tìm kiếm háu ăn

- Initial State = **Arad**
- Goal State = **Bucharest**



Straight–line distance to Bucharest

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


The End