

CÁC PHƯƠNG PHÁP TÌM KIẾM LỜI GIẢI

Nguyễn Thị Hải Bình

Email: nth.binh@hutech.edu.vn

Nội dung chính của bài học

1. Giới thiệu về giải quyết vấn đề bằng tìm kiếm
2. Không gian trạng thái
3. Chiến lược tìm kiếm mù
4. Chiến lược tìm kiếm kinh nghiệm
5. Chiến lược tìm kiếm tối ưu
6. Một số bài toán



Ví dụ: 8-puzzle (Bài toán 8 số)

1	2	3
7		6
5	4	8

Start state

1	2	3
4	5	6
7	8	

Goal state

1	2	3
7		6
5	4	8

Start state

1	2	3
7	4	6
5		8

1	2	3
7	4	6
	5	8

1	2	3
	4	6
7	5	8

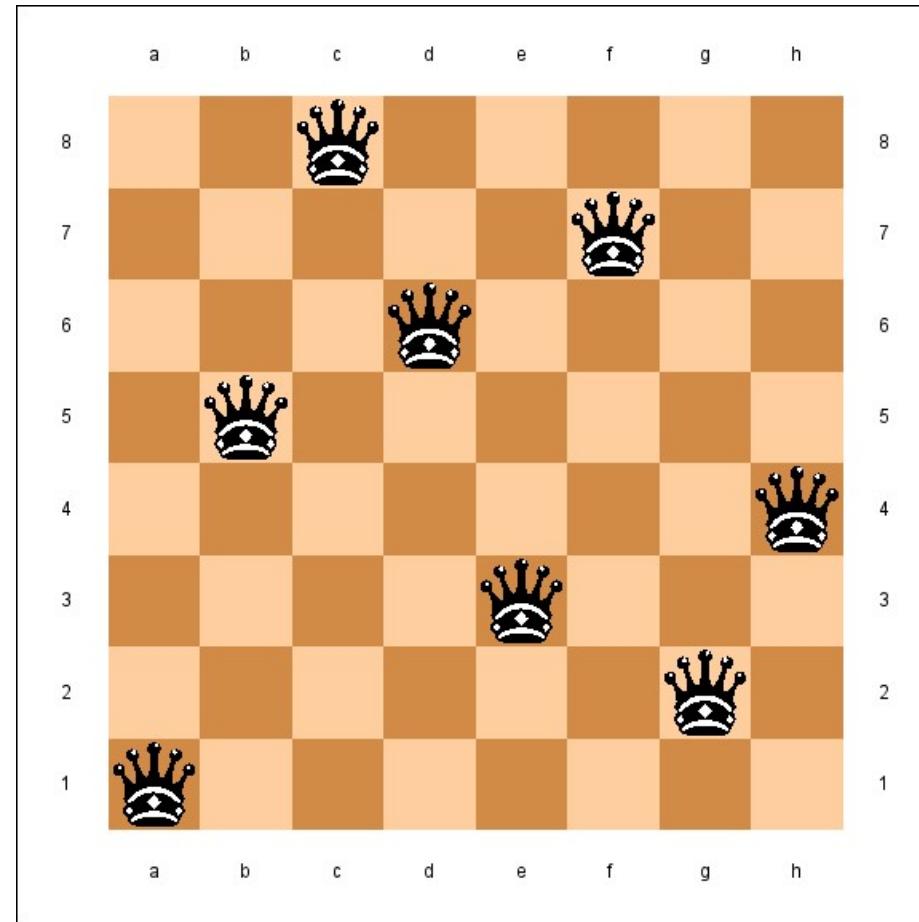
1	2	3
4		6
7	5	8

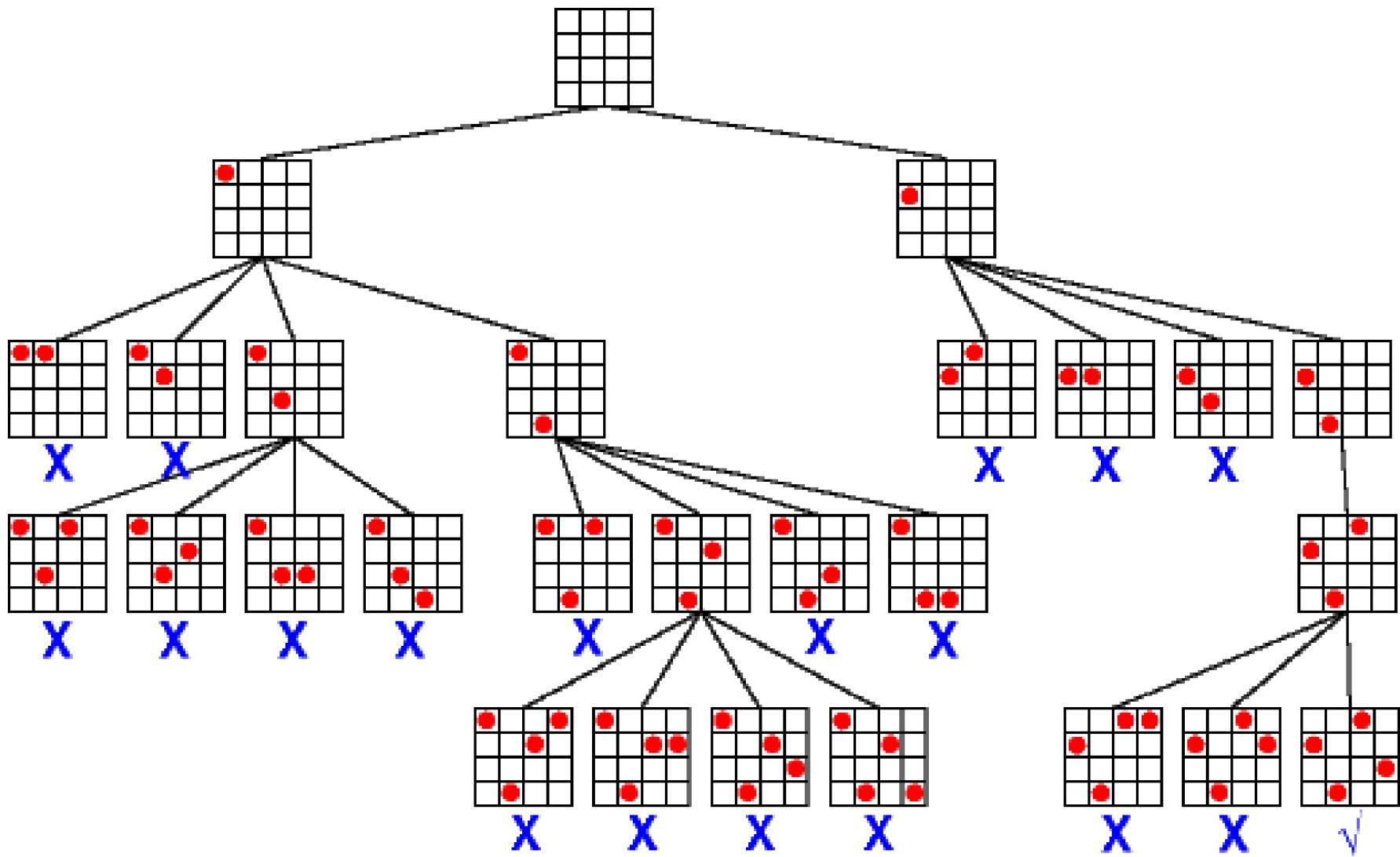
1	2	3
4	5	6
7		8

1	2	3
4	5	6
7	8	

Goal state

Ví dụ: bài toán 8 quân hậu (8 queen problem)





Ví dụ



- Route finding
- Travelling salesman problem
- VLSI layout (**Very-large-scale integration**)
- Robot navigation
- Web searching

Giải quyết vấn đề bằng tìm kiếm



- Các chiến lược hay giải thuật tìm kiếm được sử dụng để giải các bài toán trong AI.
- **Các bước để giải quyết vấn đề bằng tìm kiếm**
 - Bước 1. Xây dựng không gian trạng thái
 - Bước 2. Áp dụng các giải thuật tìm kiếm để tìm lời giải

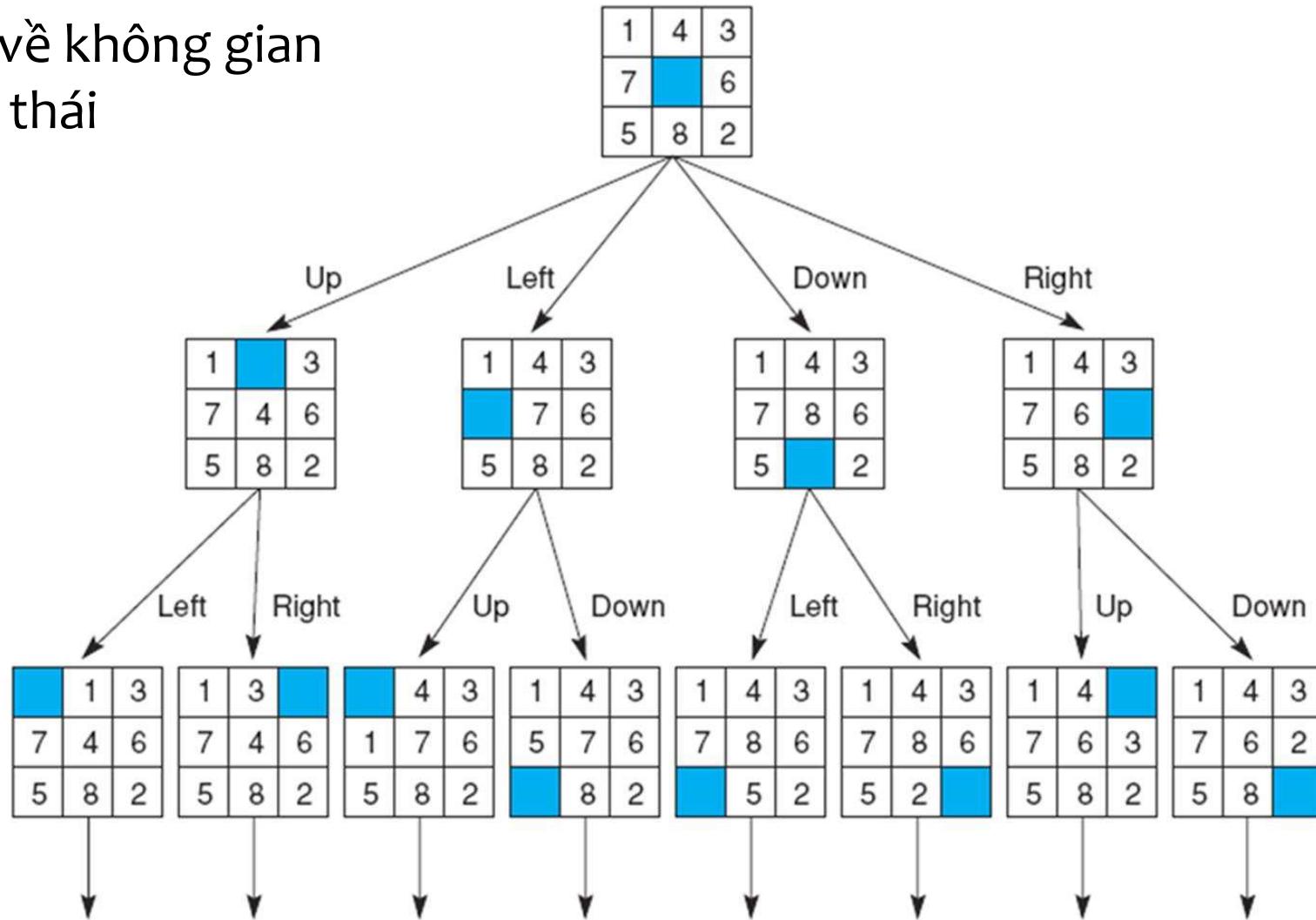
KHÔNG GIAN TRẠNG THÁI

Khái niệm không gian tìm kiếm



- Xác định không gian tìm kiếm là bước đầu để giải quyết một vấn đề bằng phương pháp tìm kiếm.
- Không gian trạng thái của một bài toán là tập các trạng thái có thể có của bài toán.
- Các trạng thái đó đạt được bằng cách thực hiện chuỗi các hành động xuất phát từ trạng thái ban đầu.

Ví dụ về không gian trạng thái



Biểu diễn vấn đề trong không gian trạng thái



Các yếu tố cần xác định

- Cách biểu diễn trạng thái
- Trạng thái ban đầu
- Tập hợp các toán tử:
 - Các **hành động** hoặc **phép biến đổi** cho phép **chuyển** một trạng thái sang trạng thái khác.
- Không gian trạng thái U của vấn đề:
 - Tập hợp tất cả các trạng thái có thể đạt tới từ trạng thái ban đầu bằng cách áp dụng một dãy các toán tử
- Tập hợp T **các trạng thái kết thúc** (trạng thái đích)
 - T là tập con của U.
 - Có vấn đề T chỉ có một phần tử (có duy nhất **một trạng thái đích**), có vấn đề có **nhiều trạng thái đích** và ta không thể xác định trước các trạng thái đích.
 - Trong phần lớn vấn đề, chỉ có thể **mô tả** trạng thái đích là trạng thái thỏa mãn một số điều kiện nào đó.

Ví dụ: xây dựng KGTT cho bài toán 8-puzzle



- Định nghĩa trạng thái:
 - Trạng thái được biểu diễn bởi vị trí của các ô (bao gồm cả ô trống)
- Tập hợp toán tử:
 - Toán tử = Sự dịch chuyển của ô trống
 - Bao gồm: Left, Right, Up, and Down
- Trạng thái ban đầu:
 - Xác định bởi đề bài
- Trạng thái kết thúc:
 - Xác định bởi đề bài

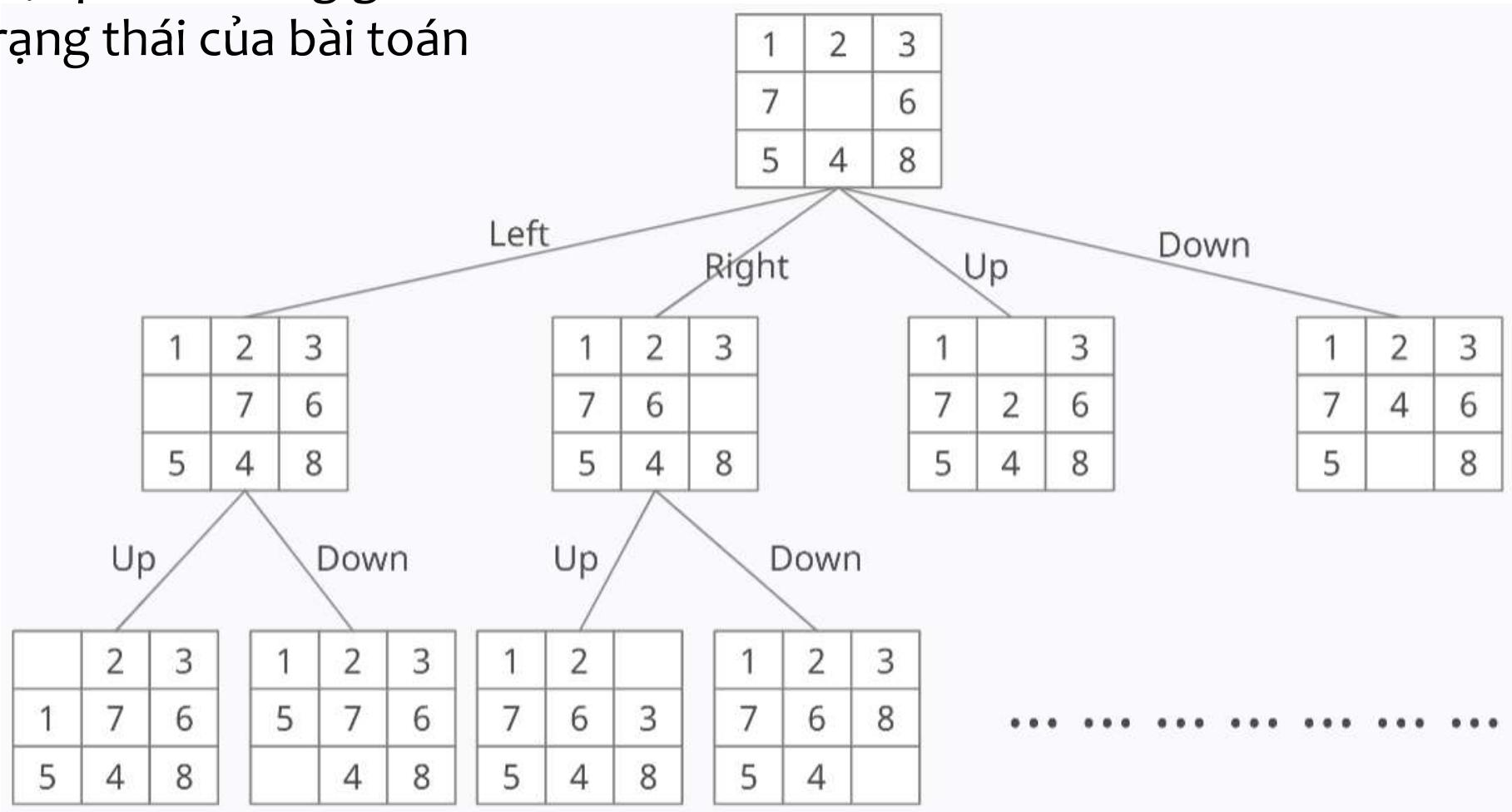
1	2	3
7		6
5	4	8

Start state

1	2	3
4	5	6
7	8	

Goal state

Một phần không gian trạng thái của bài toán





Ví dụ: xây dựng KGTT cho bài toán 8-Queen

- Định nghĩa trạng thái = vị trí của các quân hậu trên bàn cờ.
- Xác định tập hợp toán tử
- Trạng thái ban đầu : Bàn cờ trống
- Trạng thái kết thúc : Cờ tam quân hậu
trên bàn cờ và tam quân hậu này
không tấn công lẫn nhau.

vị trí

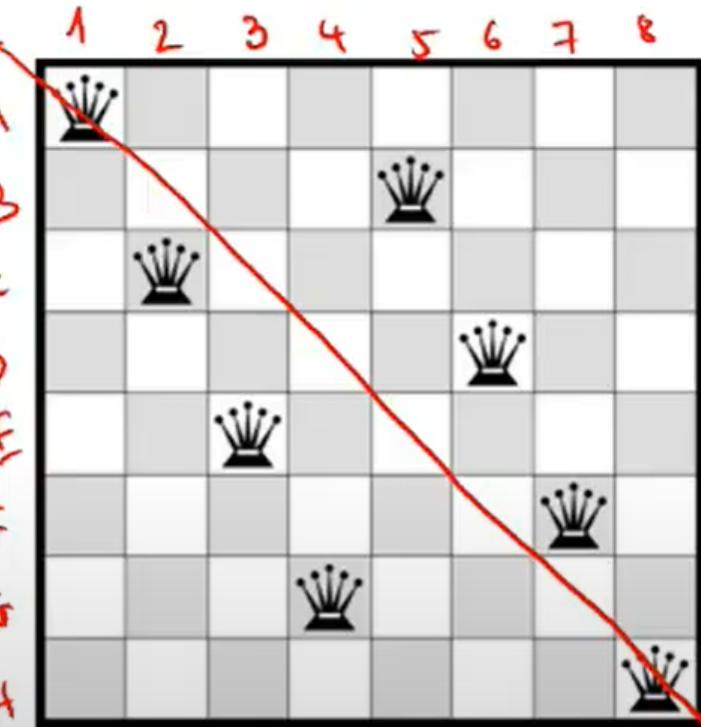


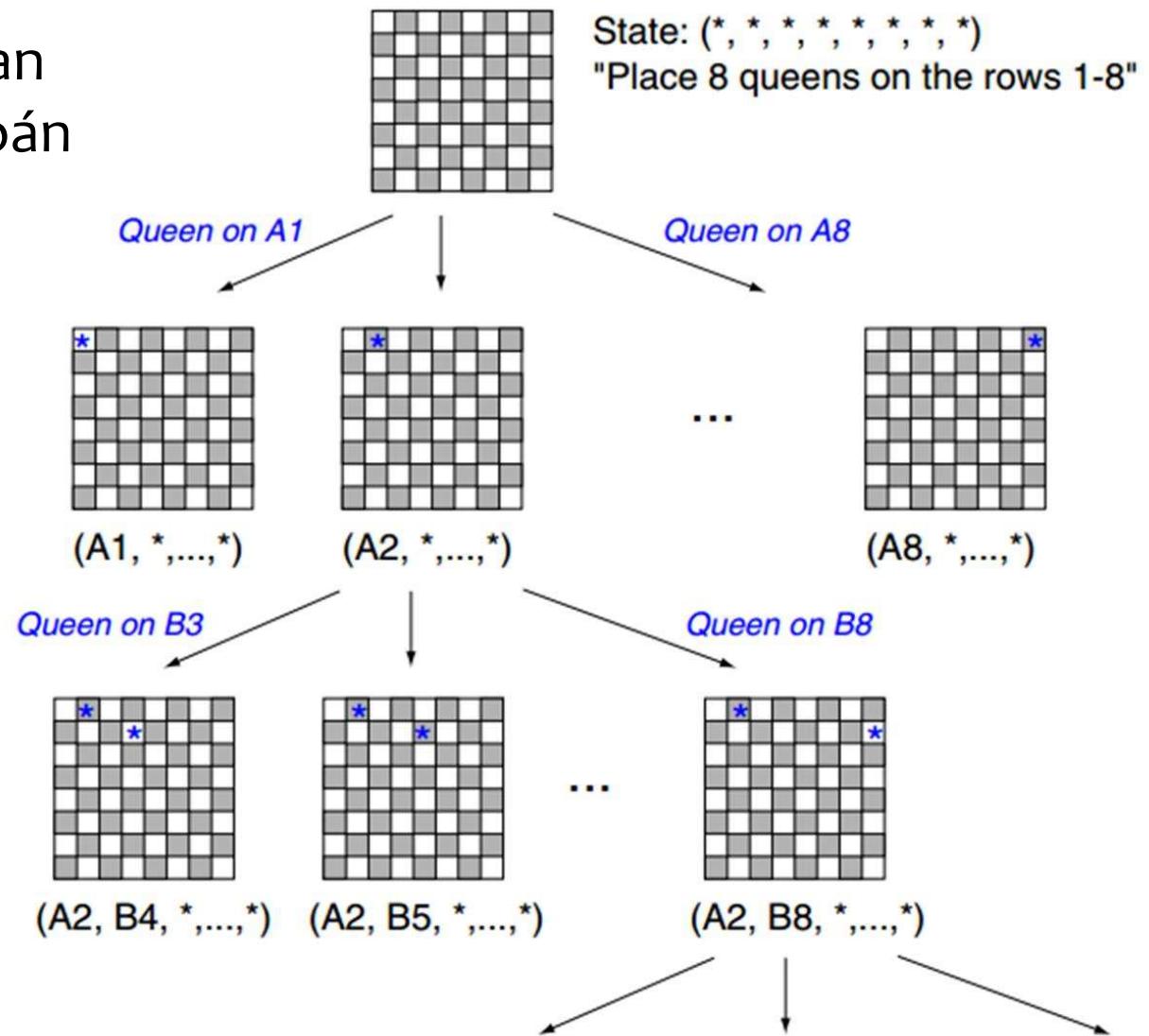
Figure 1: Almost a solution of the 8-queens problem

Ví dụ: xây dựng KGTT cho bài toán 8-Queen



- Định nghĩa trạng thái
 - Sự sắp xếp của các quân hậu trên bàn cờ (số lượng quân hậu từ 0 đến 8)
- Xác định tập hợp toán tử
 - Toán tử = Thêm một quân hậu vào vị trí trống trên bàn cờ
- Trạng thái ban đầu
 - Bàn cờ trống
- Trạng thái kết thúc
 - Có 8 quân hậu trên bàn cờ, các quân hậu không tấn công nhau

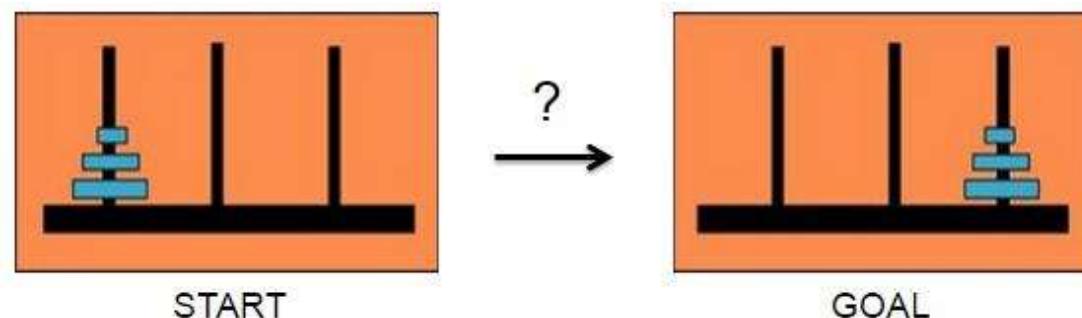
Một phần không gian trạng thái của bài toán



Ví dụ: KGTT cho bài toán tháp Hà Nội



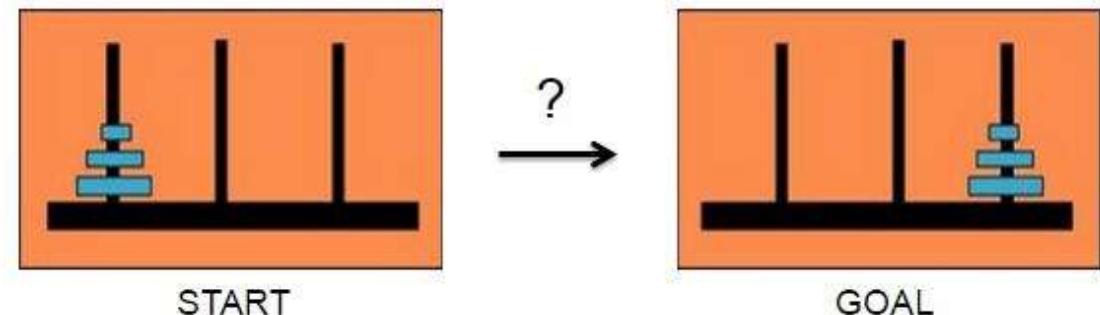
- Định nghĩa trạng thái
- Xác định tập hợp toán tử
- Trạng thái ban đầu
- Trạng thái kết thúc

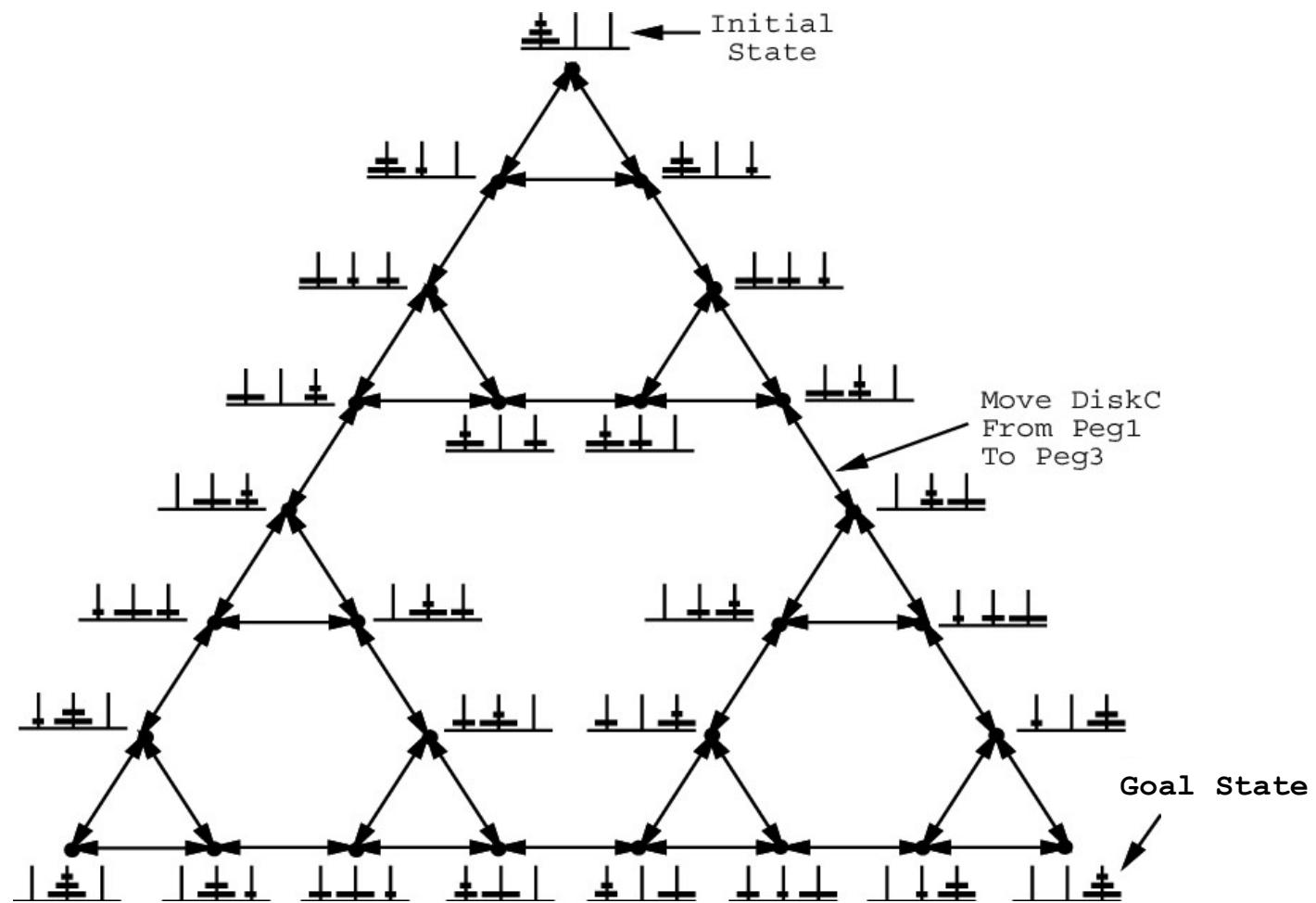


Ví dụ: KGTT cho bài toán tháp Hà Nội



- Định nghĩa trạng thái
 - Vị trí của mỗi đĩa trên các cọc.
- Xác định tập hợp toán tử
 - Toán tử = Dịch chuyển một đĩa sang một cọc khác (không vi phạm quy tắc dịch chuyển)
- Trạng thái ban đầu
 - Ba đĩa ở cọc đầu tiên
- Trạng thái kết thúc
 - Ba đĩa ở cọc cuối cùng

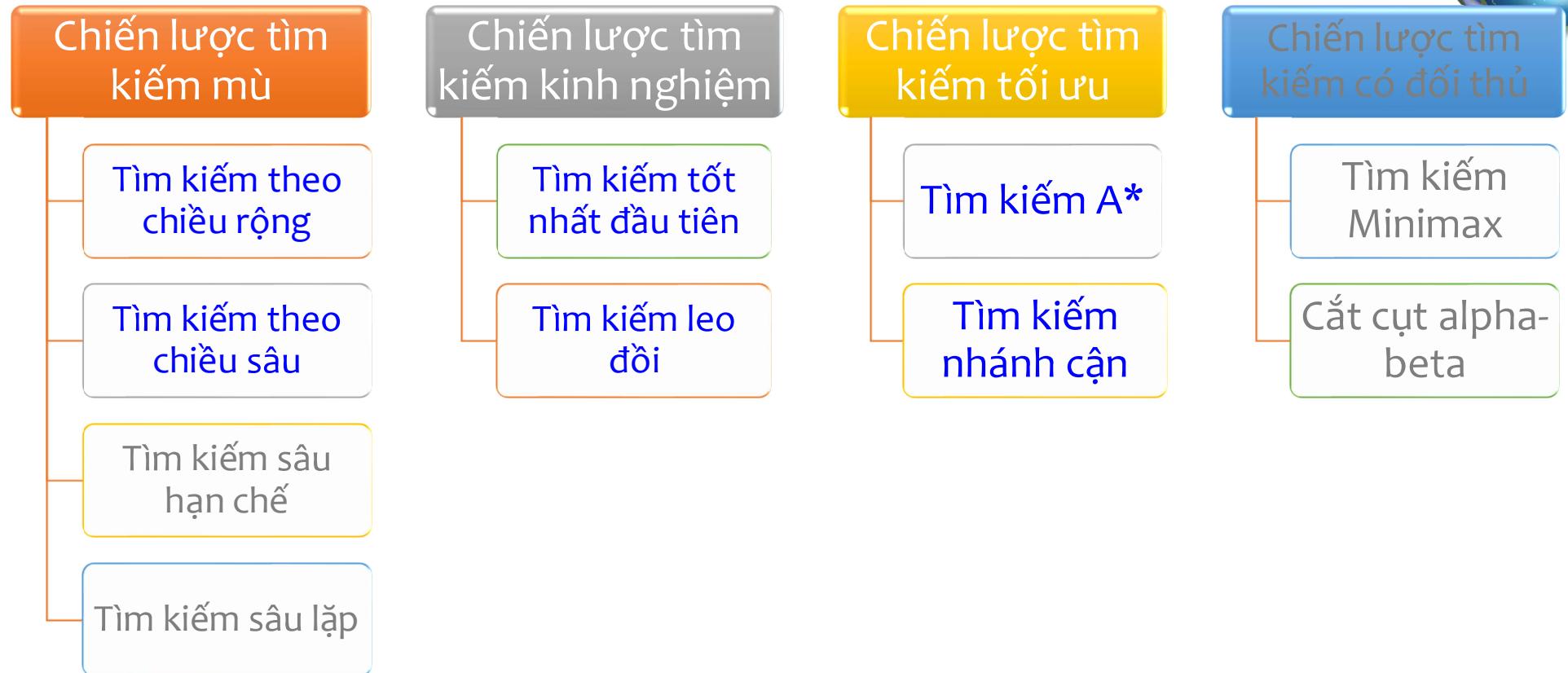




Knoblock, Craig. (2000). Abstracting the Tower of Hanoi. Working Notes of AAAI-90 Workshop on Automatic Generation of Approximations and Abstractions. 20

CÁC THUẬT TOÁN TÌM KIẾM

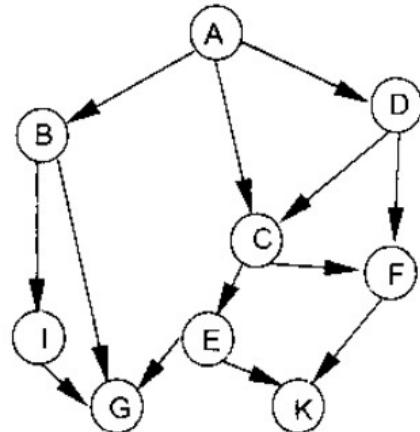
Các thuật toán tìm kiếm



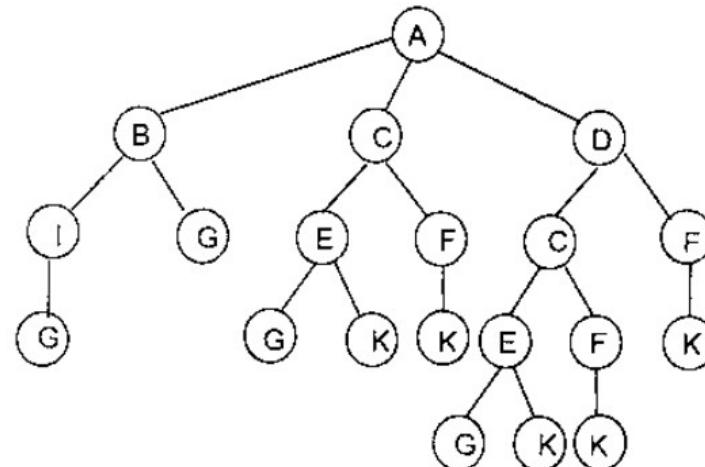
Cây tìm kiếm



- Quá trình tìm kiếm = xây dựng cây tìm kiếm
- Cây tìm kiếm
 - Đỉnh: các trạng thái
 - Gốc: trạng thái ban đầu



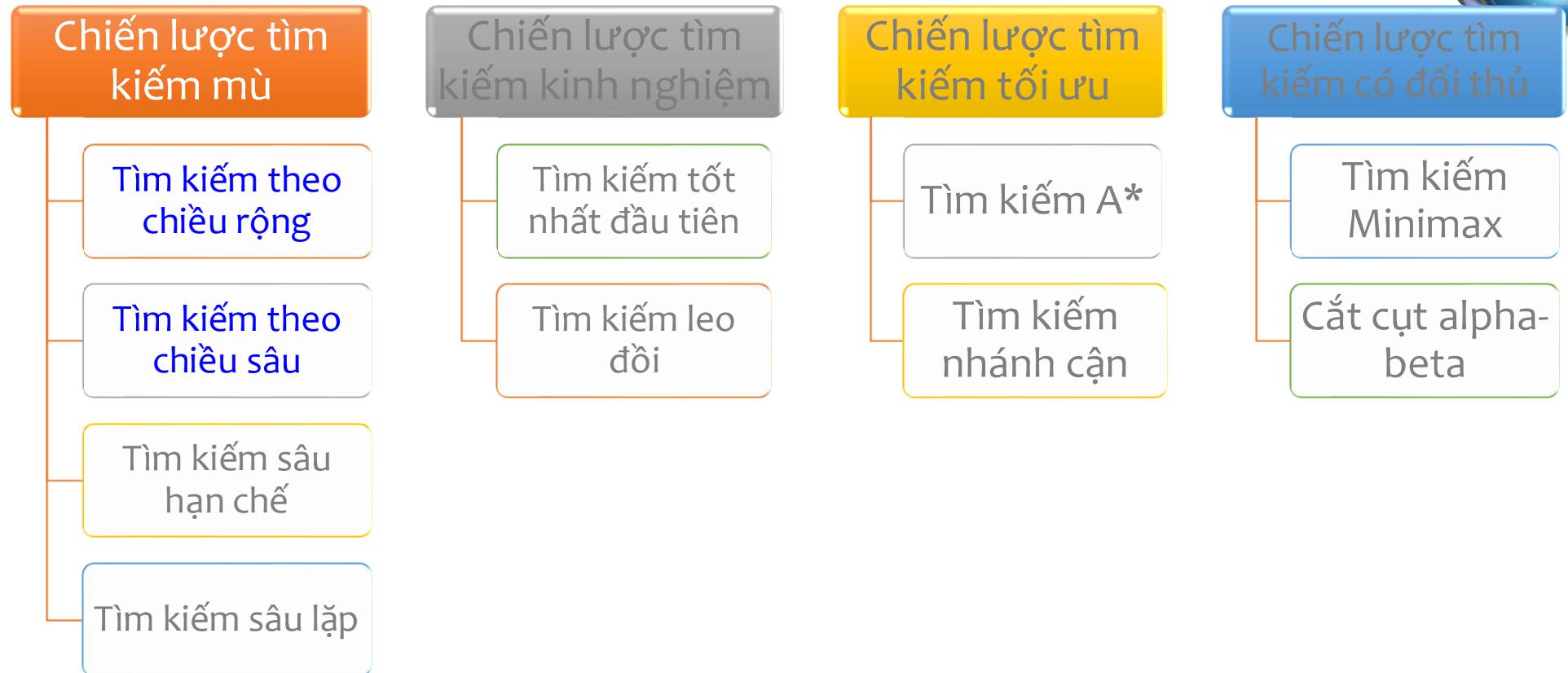
a) Đồ thị không gian trạng thái



b) Cây tìm kiếm tương ứng

CHIẾN LƯỢC TÌM KIẾM MÙ

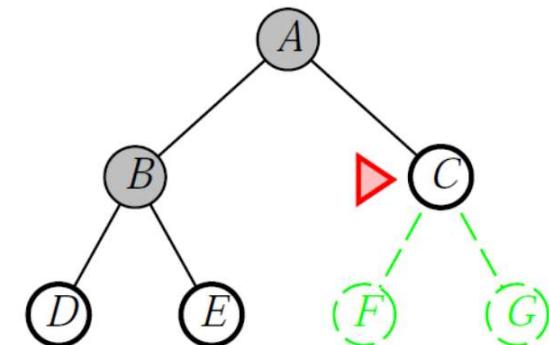
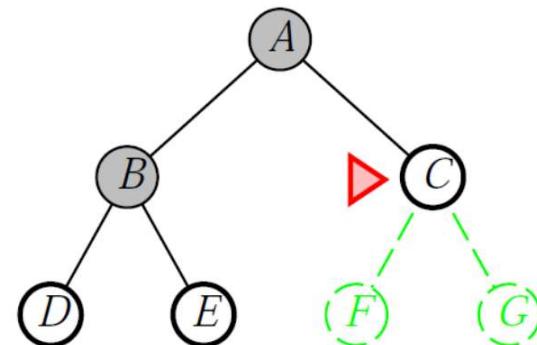
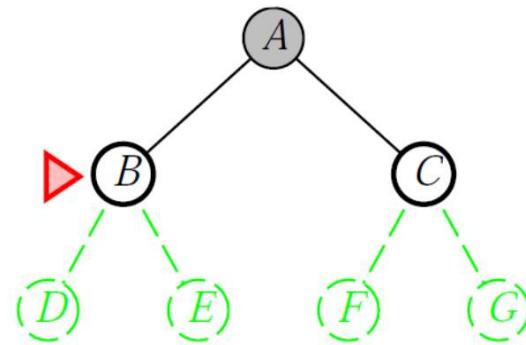
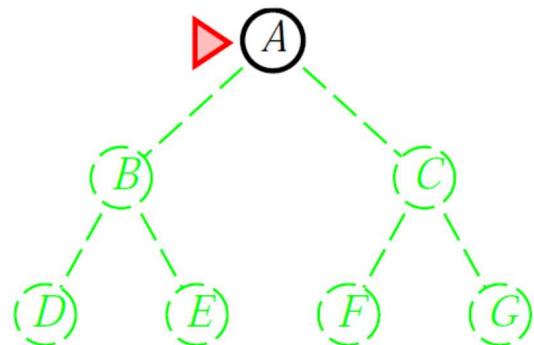
Các thuật toán tìm kiếm



Tìm kiếm theo chiều rộng (Breath First Search – BFS)



- Các trạng thái được phát triển theo thứ tự mà chúng được sinh ra



Thuật toán BFS



Procedure Breadth_First_Search

begin

1. Khởi tạo danh sách L chỉ chứa trạng thái ban đầu

2. **loop do**

 2.1 **if** L rỗng **then**

 {thông báo tìm kiếm thất bại; stop};

 2.2 Loại trạng thái u ở đầu danh sách L;

 2.3 **if** u là trạng thái kết thúc **then**

 {thông báo tìm kiếm thành công; stop};

 2.4 **for** mỗi trạng thái v kề u **do**

 {đặt v vào cuối danh sách L;

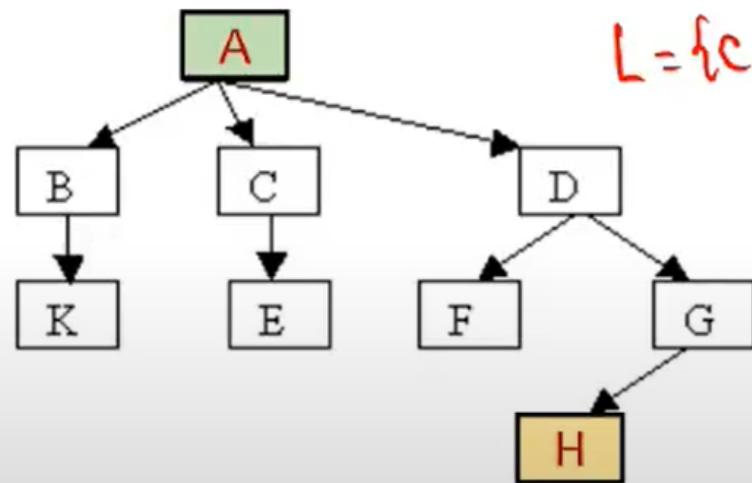
 father(v)=u};

end;

BFS - Ví dụ 1

Trạng thái bắt đầu: A

Trạng thái kết thúc: H



kết quả: A → D → G → H.

Mô tả quá trình tìm kiếm
• Khởi tạo: L = {A} Đoạn bao nhiêu trạng thái?

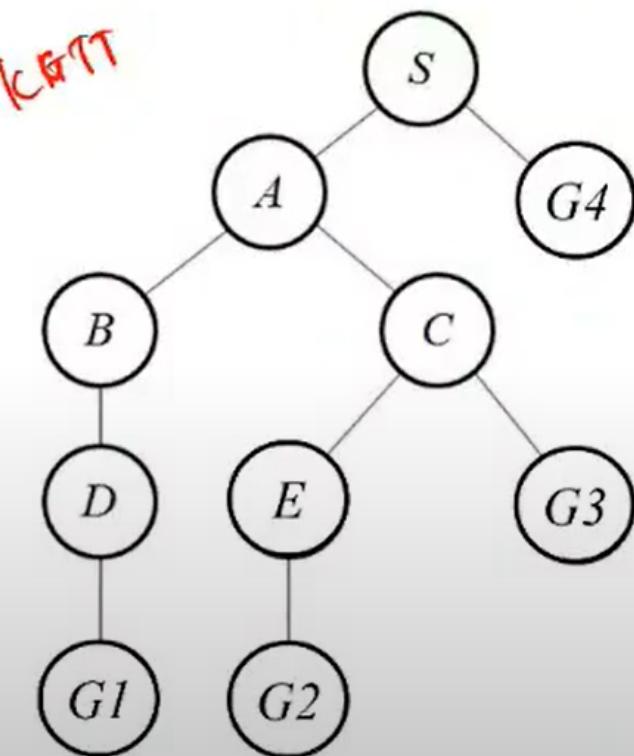
Phát triển trạng thái	Trạng thái kề	Danh sách L
A	B, C, D	{B, C, D}
B	K	{C, D, K}
C	E	{D, K, E}
D	F, G	{K, E, F, G}
K		{E, F, G}
E		{F, G}
F		{G}
G	H	{H}
H		

TÌM KIẾM - DỪNG.

BFS - Ví dụ 2

Trạng thái bắt đầu: S

Trạng thái kết thúc: {G₁, G₂, G₃, G₄}



Mô tả quá trình tìm kiếm

- Khởi tạo: L = {S}



Phát triển trạng thái	Trạng thái kề	Danh sách L
S	A, G ₄	{A, G ₄ }
A	B, C	{G ₄ , B, C}
G ₄	Trao - Đổi (*)	{B, C}
		Cây tìm kiếm.
KL : S → G ₄ .		

Tính chất

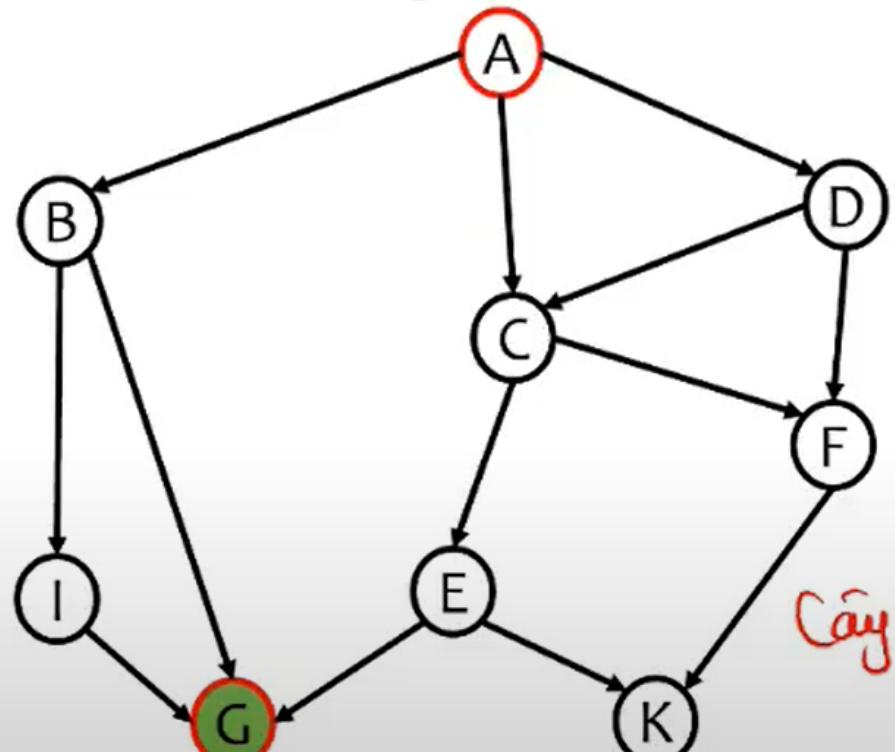


- Luôn luôn tìm ra đường đi từ trạng thái đầu tới trạng thái đích, nếu đường đi tồn tại
- Đường đi tìm được là đường đi ngắn nhất (if cost = 1 per step)
- Độ phức tạp của thuật toán?
 - Đánh giá dựa theo các yếu tố sau
 - Nhân tố nhánh b – số lượng tối đa trạng thái kề được sinh ra khi phát triển một trạng thái
 - Độ dài d – độ dài của đường đi từ trạng thái đầu tới đích
 - Độ phức tạp thời gian (time complexity) và độ phức tạp không gian (memory complexity)

$$1 + b + b^2 + \dots + b^d = O(b^d)$$

Trạng thái lắp

Trạng thái khởi đầu



Trạng thái kết thúc

$$L = \{ A \}$$

$$KL: A \rightarrow B \rightarrow G.$$



Phát triển trạng thái	Trạng thái kề	Danh sách L
A	B, C, D	{B, C, D}
B	I, G	{C, D, I, G}
C	E, F	{D, I, G, E, F}
D	I	{I, G, E, F}
I	G	{G, E, F}
G		
	TTKT - Đã k	{E, F}

Cây tìm kiếm :

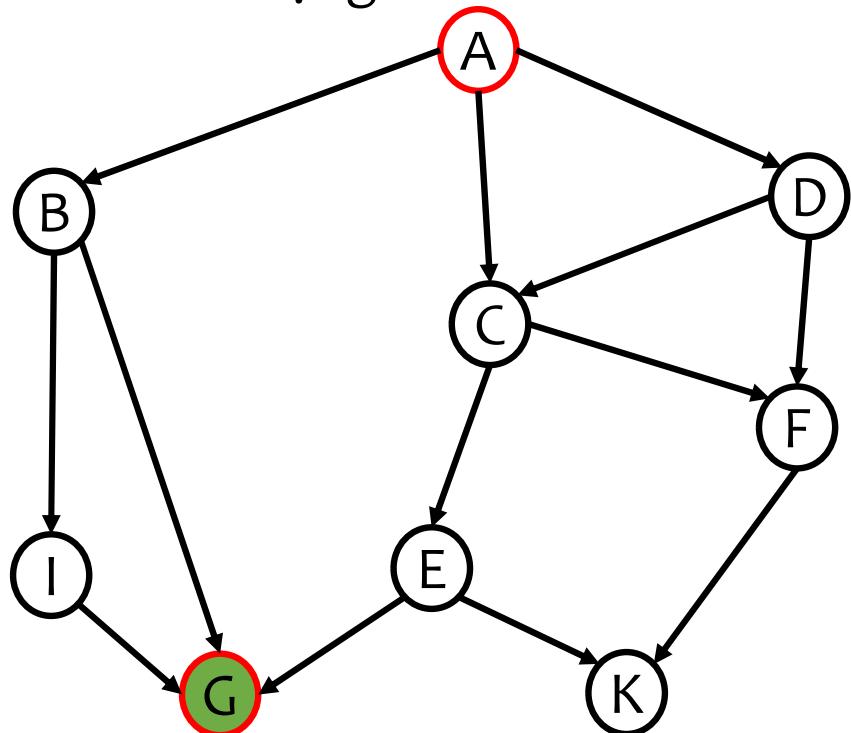
8 trạng thái

34



Trạng thái lặp

Trạng thái khởi đầu



Trạng thái kết thúc

- Nếu cây tìm kiếm chứa **nhiều đỉnh ứng với cùng một trạng thái**, các trạng thái này được gọi là **trạng thái lặp**.
- Trong đồ thị biểu diễn **không gian trạng thái**, các trạng thái lặp ứng với các **đỉnh có nhiều đường đi dẫn tới nó từ trạng thái ban đầu**.



Loại bỏ trạng thái lặp

• Vấn đề của trạng thái lặp

- Trạng thái lặp gây lãng phí thời gian
- Nếu đồ thị có chu trình quá trình tìm kiếm sẽ không dừng
- Vì vậy, trong quá trình tìm kiếm cần tránh sinh ra các trạng thái đã phát triển

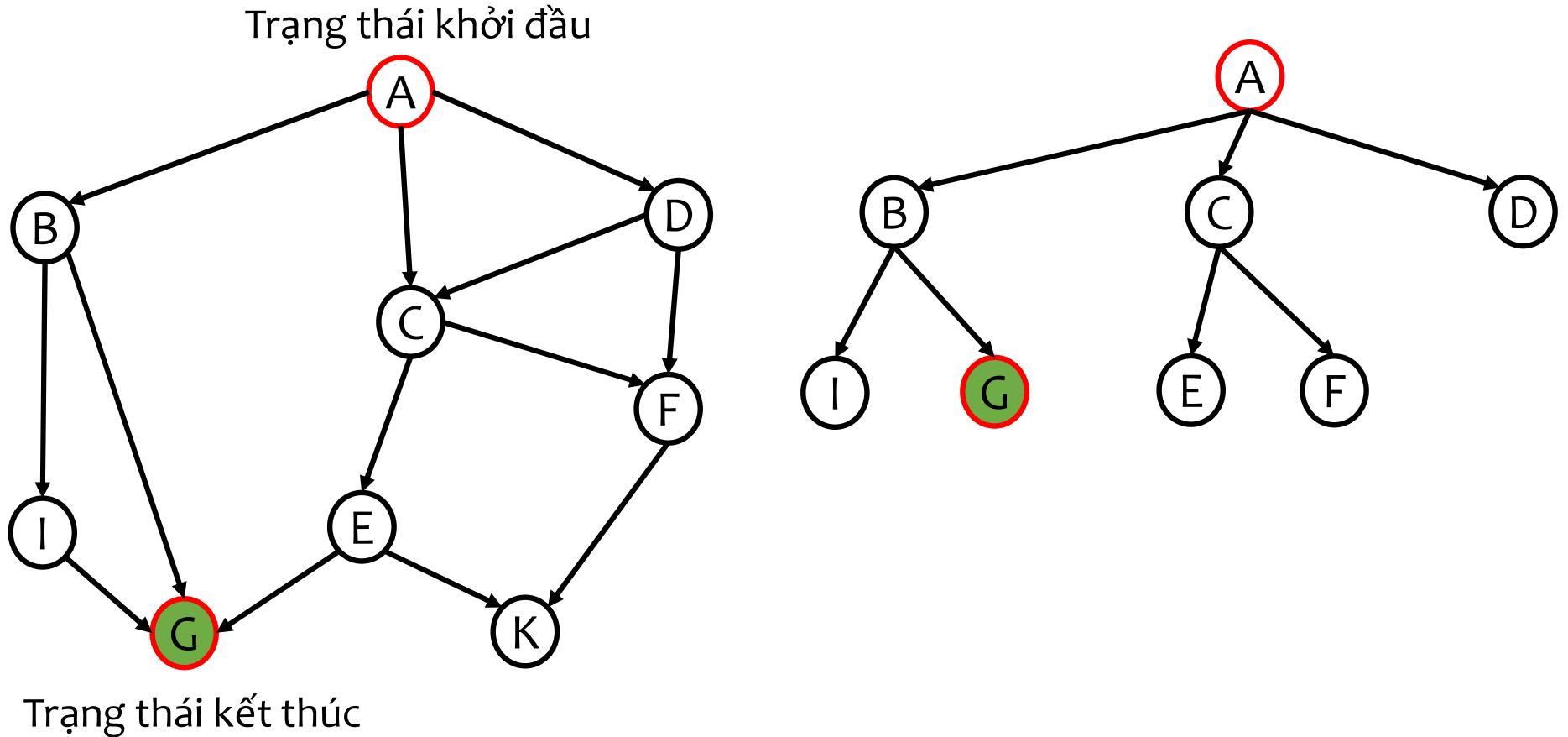
• Giải pháp

- Khi phát triển đỉnh u, không sinh ra các đỉnh trùng với cha u
- Khi phát triển đỉnh u, không sinh ra các đỉnh trùng với một đỉnh nào đó nằm trên đường đi dẫn tới u
- Khi phát triển một đỉnh, không sinh ra các đỉnh đã được sinh ra

• Cài đặt

- Lưu tất cả các trạng thái đã sinh ra vào danh sách Q
- Trạng thái v được thêm vào danh sách L nếu v không nằm trong Q

Cây tìm kiếm tương ứng được sinh ra nếu loại bỏ trạng thái lặp.





BFS – Ví dụ 3: bài toán 8-puzzle

- **Định nghĩa trạng thái:**

- Trạng thái được biểu diễn bởi vị trí của các ô (bao gồm cả ô trắng)

- **Tập hợp toán tử:**

- Toán tử = Sự dịch chuyển của ô trắng
- Bao gồm: Up, Left, Right, Down

- **Trạng thái bắt đầu:**

1	2	5
3	4	
6	7	8

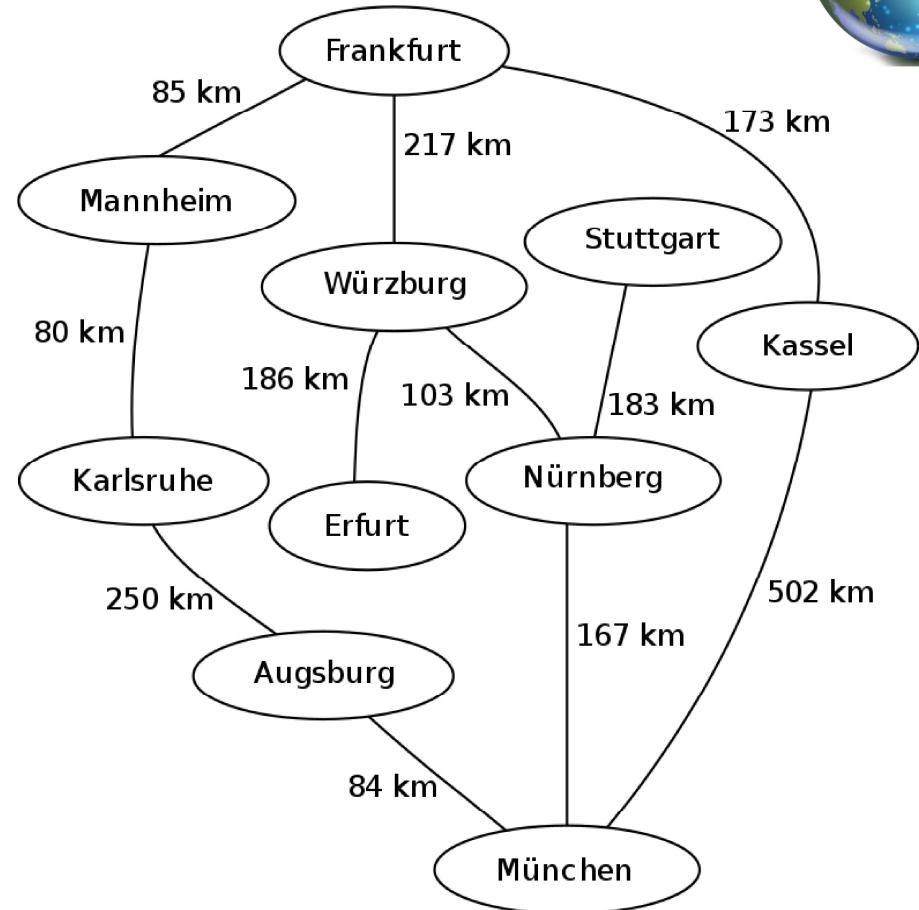
- **Trạng thái kết thúc:**

	1	2
3	4	5
6	7	8

BFS – Ví dụ 4: Tìm đường đi trên bản đồ



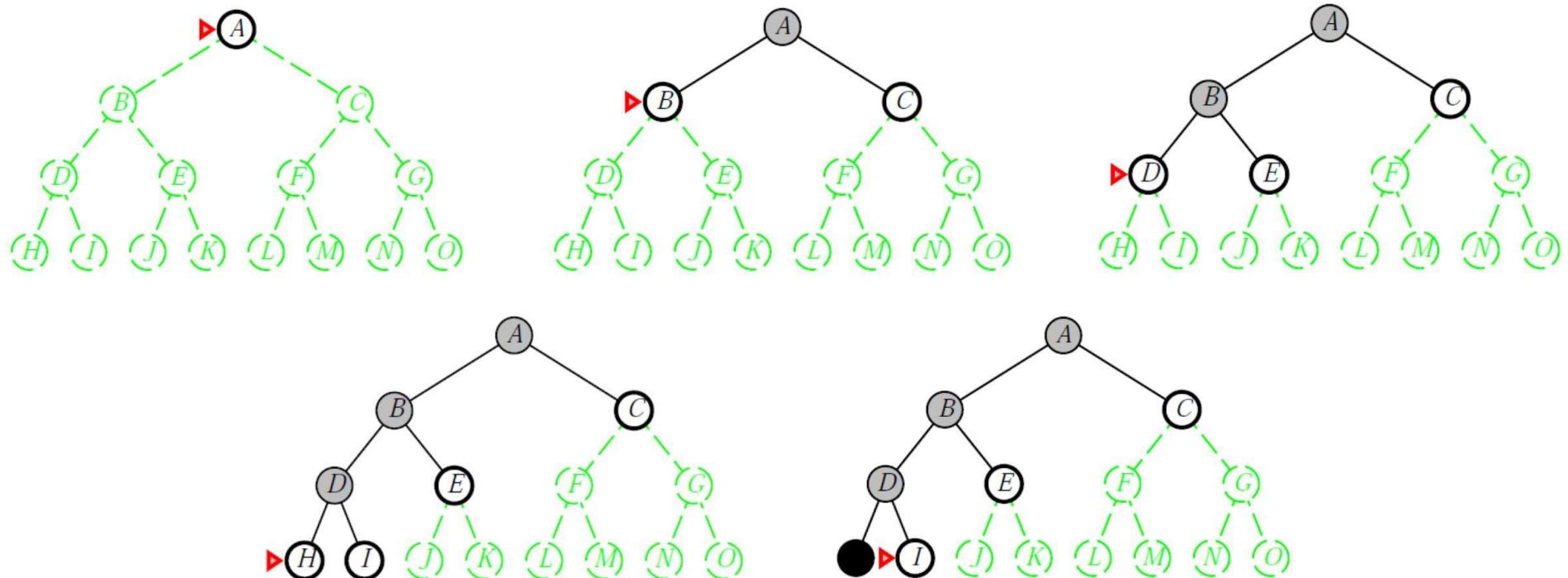
- Cho bản đồ (giản lược) của Southern Germany.
- Hãy tìm đường đi từ Frankfurt tới München.





Tìm kiếm theo chiều sâu (Depth First Search – DFS)

- Trạng thái được chọn để phát triển là trạng thái được sinh ra sau cùng trong các trạng thái chờ phát triển khác



Thuật toán DFS



Procedure Depth_First_Search

begin

1. Khởi tạo danh sách L chỉ chứa trạng thái ban đầu

2. **loop do**

 2.1 **if** L rỗng **then**

 {thông báo tìm kiếm thất bại; stop};

 2.2 Loại trạng thái u ở đầu danh sách L;

 2.3 **if** u là trạng thái kết thúc **then**

 {thông báo tìm kiếm thành công; stop};

 2.4 **for** mỗi trạng thái v kề u **do**

 {đặt v vào **đầu** danh sách L;

 father(v)=u};

end;

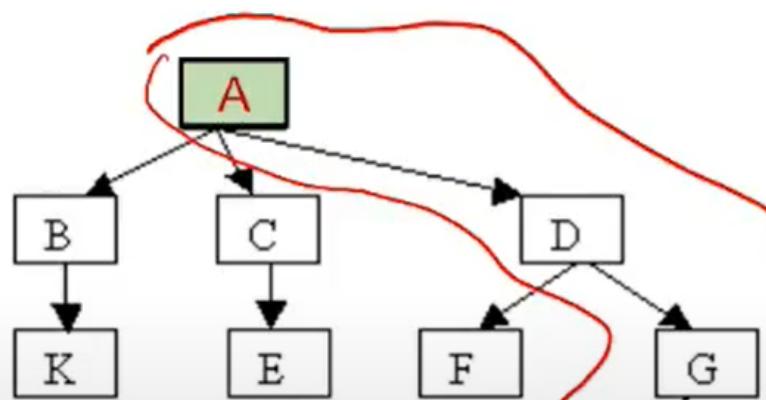
DFS - Ví dụ 1

Mô tả quá trình tìm kiếm

- Khởi tạo: $L = \{A\}$

Trạng thái bắt đầu: A

Trạng thái kết thúc: H

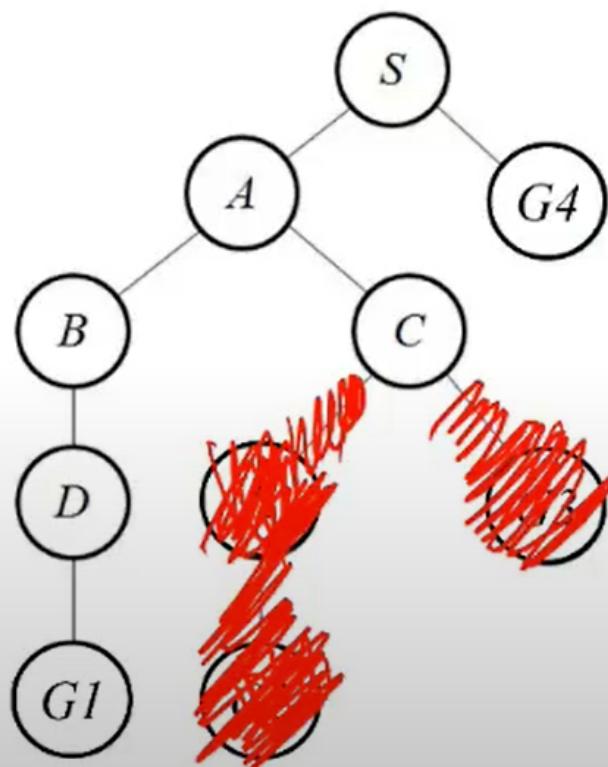


Phát triển trạng thái	Trạng thái kề	Danh sách L
A	B, C, D	B, C, D
B	K	K, C, D
K		C, D
C	E	E, D
E		D
D	F, G	F, G
F		G
G	H	H
H	TTKT – Dừng	

DFS - Ví dụ 2

Trạng thái bắt đầu: S

Trạng thái kết thúc: {G₁, G₂, G₃, G₄}



Mô tả quá trình tìm kiếm

- Khởi tạo: L = {S}

Phát triển trạng thái	Trạng thái kề	Danh sách L
S	A, G ₄	{A, G ₄ }
A	B, C	{B, C, G ₄ }
B	D	{D, C, G ₄ }
D	G ₁ .	{G ₁ , C, G ₄ }
G ₁ .	TTKT - Dừng	{C, G ₄ }

KL: S → A → B → D → G₁.

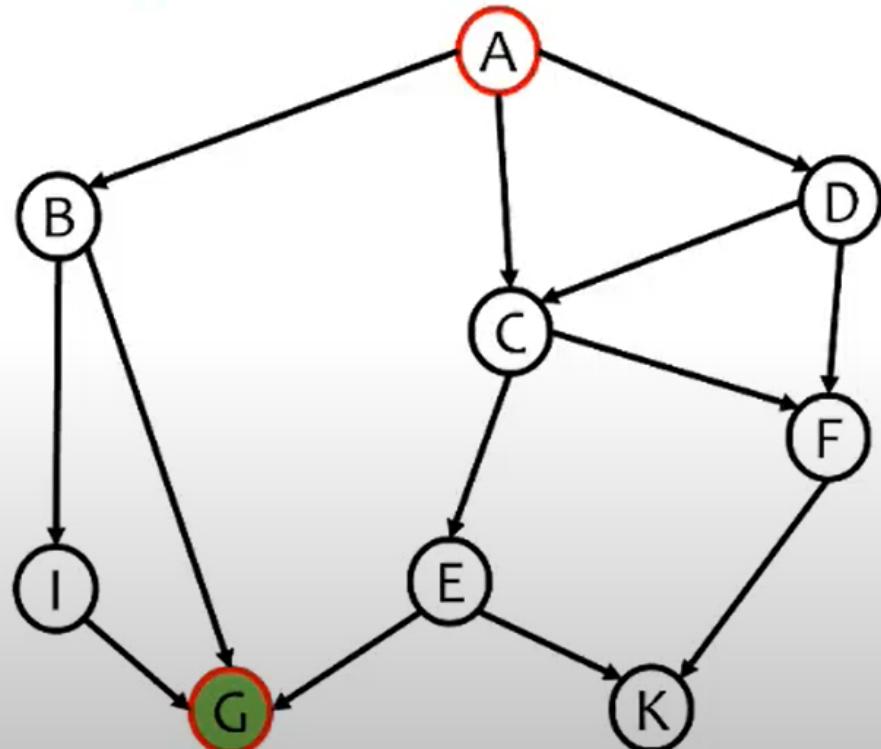
DFS - Ví dụ 3

$$L = \{A\}^{\bullet}$$



Trạng thái bắt đầu: § A

Trạng thái kết thúc: G



Phát triển trạng thái	Trạng thái kè	Danh sách L
A	B, C, D	{B, C, D}
B	I, G.	{I, G, C, D}
I		{G, C, D}
G	TTKT - Đóng	{C, D}.

```

graph TD
    A((A)) --> B((B))
    A --> C((C))
    B --> I((I))
    B --> G((G))
    C --- D((D))
  
```

Cây tìm kiếm tương ứng được sinh ra nếu loại bỏ trạng thái lặp.

DFS – Ví dụ 4: bài toán 8-puzzle



- **Định nghĩa trạng thái:**

- Trạng thái được biểu diễn bởi vị trí của các ô (bao gồm cả ô trắng)

- **Tập hợp toán tử:**

- Toán tử = Sự dịch chuyển của ô trắng
- Bao gồm: Up, Down, Left, Right

- **Trạng thái bắt đầu:**

1	2	5
3	4	
6	7	8

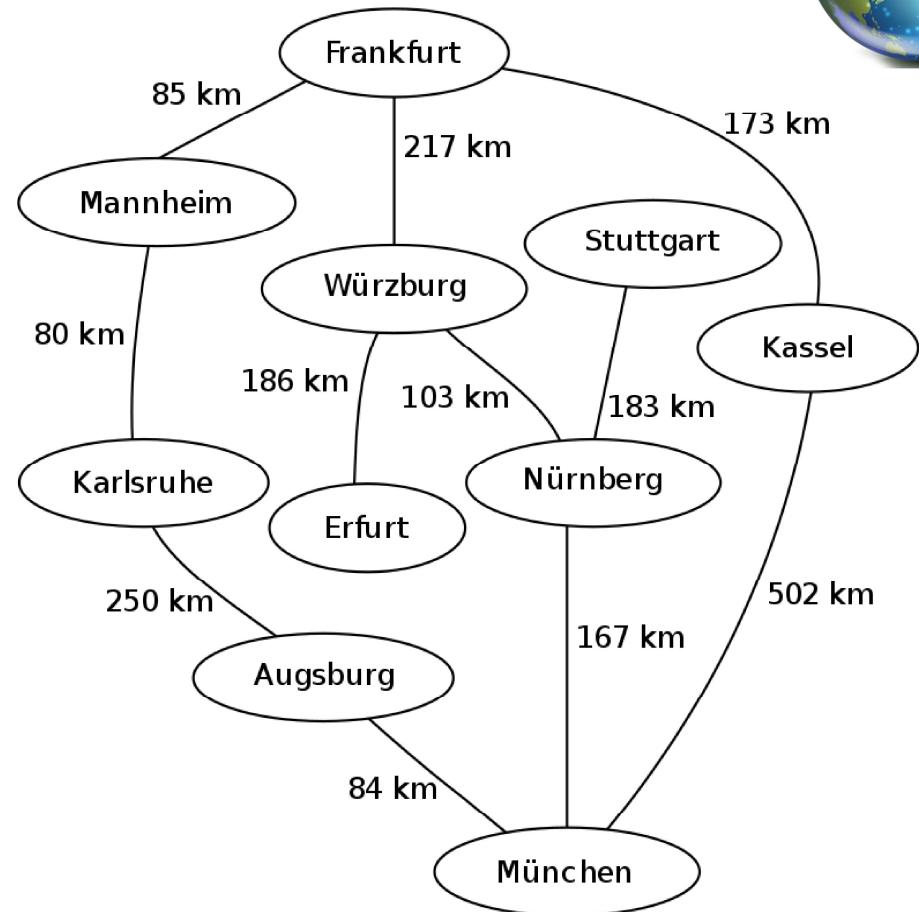
- **Trạng thái kết thúc:**

	1	2
3	4	5
6	7	8

DFS – Ví dụ 5: Tìm đường đi trên bản đồ



- Cho bản đồ (giản lược) của Southern Germany.
- Hãy tìm đường đi từ Frankfurt tới München.



Cây tìm kiếm tương ứng sinh ra khi sử dụng thuật toán DFS

Số trạng thái cần phát triển = ?

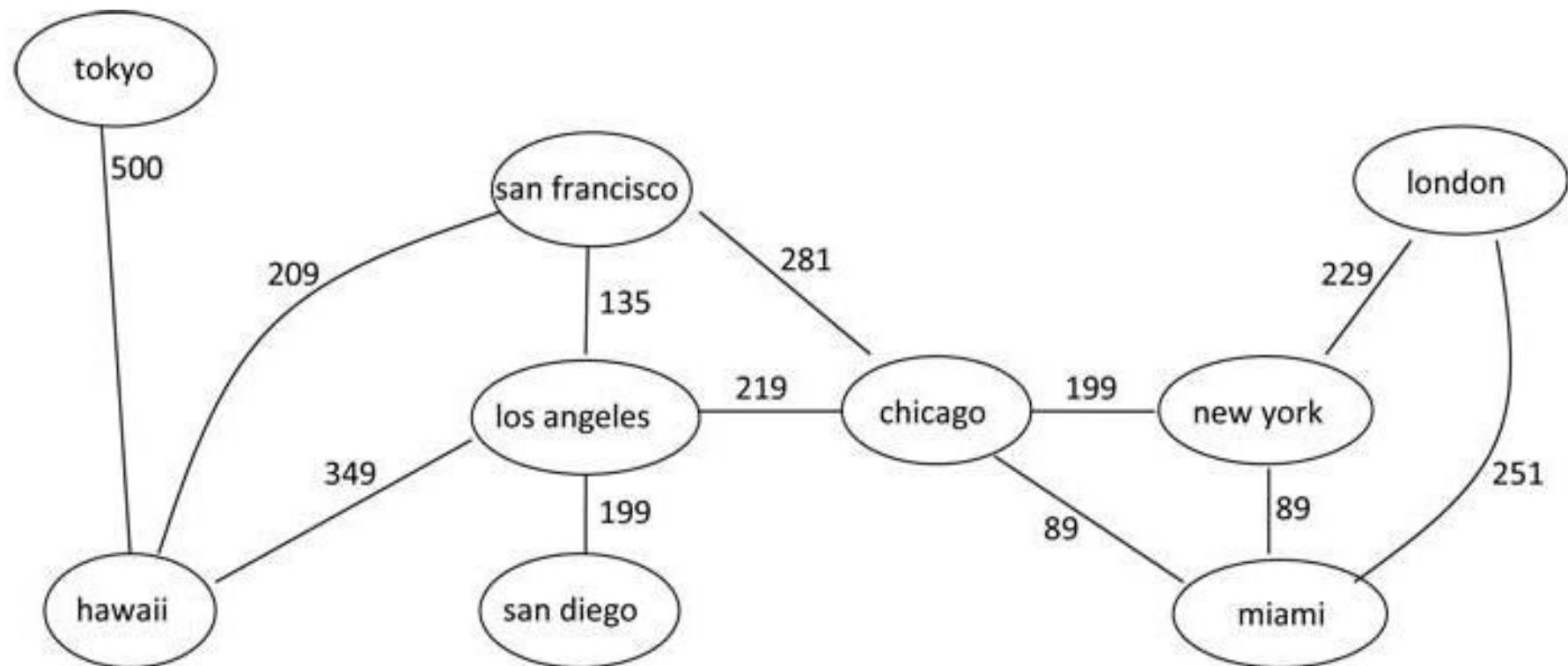
Số trạng thái được sinh ra = ?

Đường đi tìm được = ?

BÀI TẬP TÌM KIẾM MỤ

Tìm đường đi từ Tokyo tới London

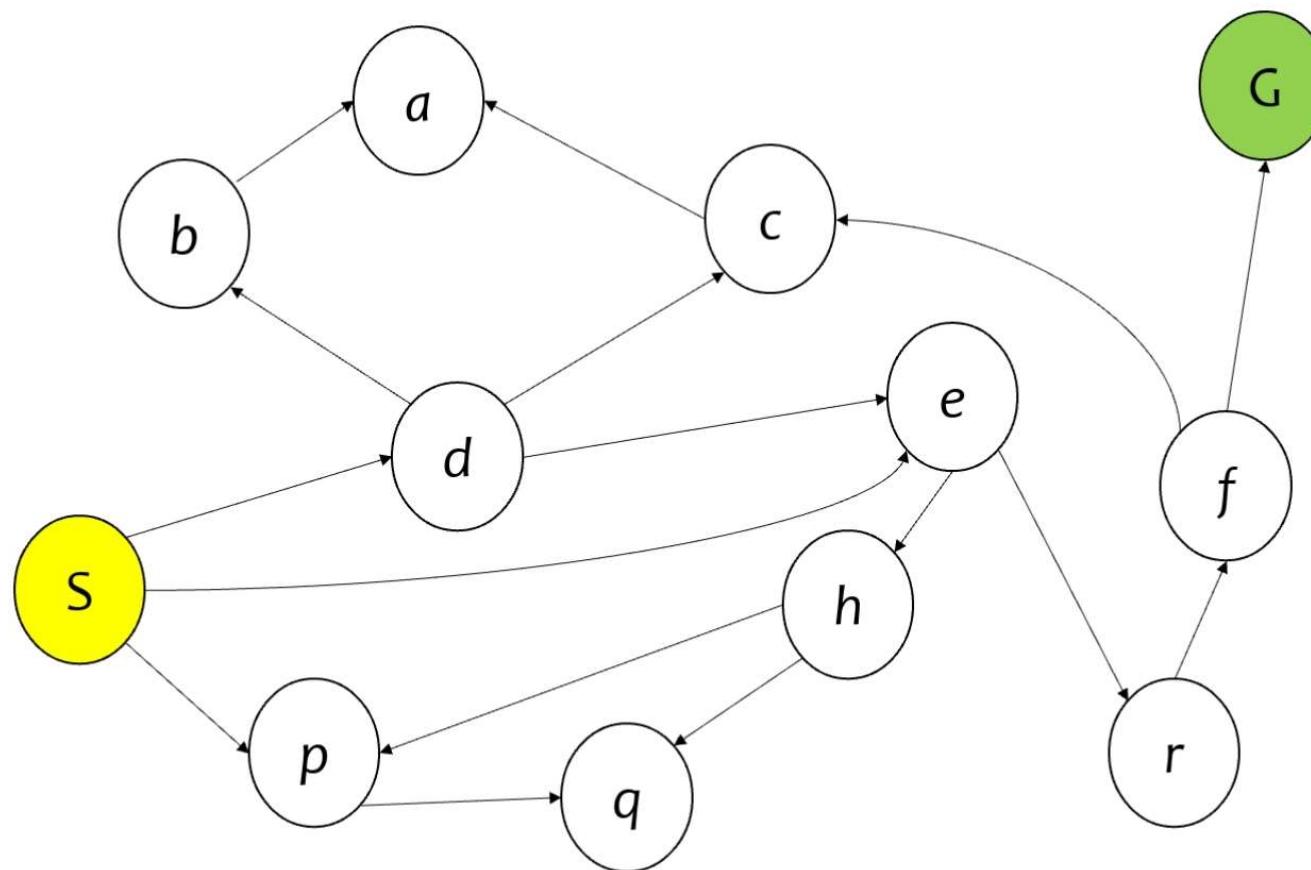
Tìm đường đi từ Hawaii tới New York



Phát triển trạng thái	trạng thái kè	Danh sách L
S	D,E,P	D,E,P
D	B,C,E	B,C,E,D,E,P
B	A	A,C,E,D,E,P
A		C,E,D,E,P
C		E,D,E,P
E	R,H	R,H,D,E,P
R	F	F,H,D,E,P
F	G	G,H,D,E,P
G	DỪNG	DỪNG

S->D->B->A->C->E->R->F->G

Tìm đường đi từ S đến G



Bài toán tìm đường để robot đi từ A3 sang E2. Giả sử robot chỉ có thể đi sang trái, sang phải, hoặc đi xuống.

1. Xác định không gian trạng thái

2. Áp dụng BFS hoặc DFS để tìm đường đi

Xác định ✓

Xác TT sẵn duy
vẽ cây tìm kiếm

Xác định thứ tự
phát triển trạng thái

Khuôn .

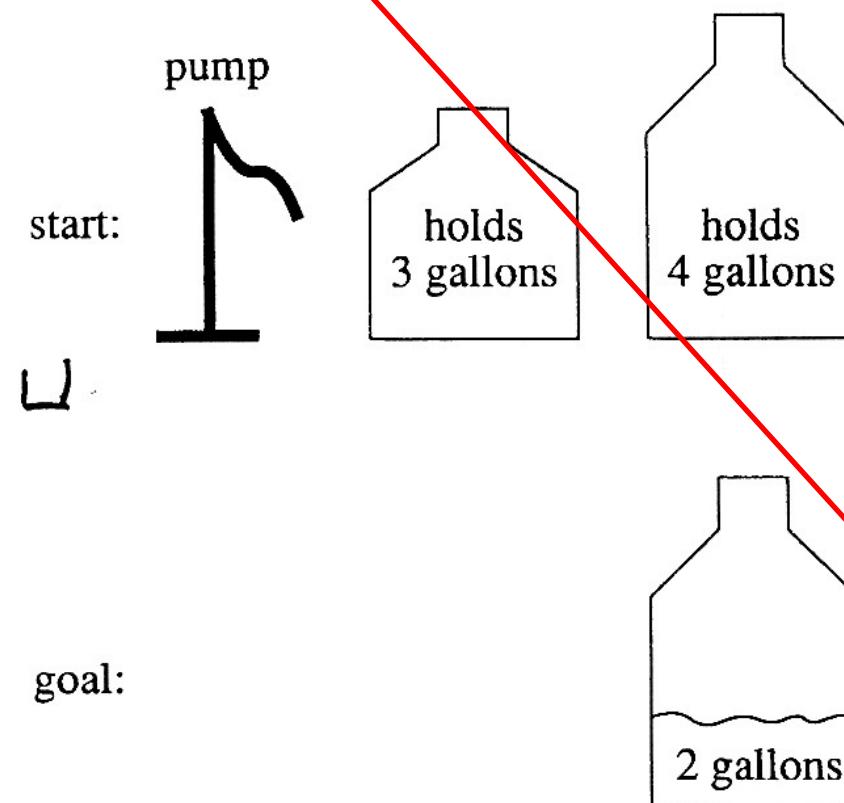


Nhom

1 bài

- Dinh nghia TT.
- Toan tu.
- TT bat dau
- TT ket thuc

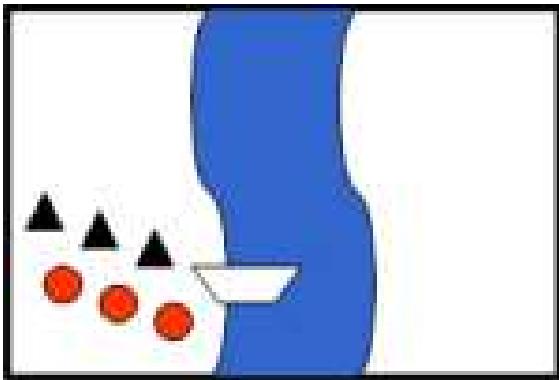
Water jug problem



Missionaries and Cannibals problem

On one bank of a river are three missionaries and three cannibals. There is one boat available that can hold up to two people and that they would like to use to cross the river. If the cannibals ever outnumber the missionaries on either of the river's banks, the missionaries will get eaten.

How can the boat be used to safely carry all the missionaries and cannibals across the river?

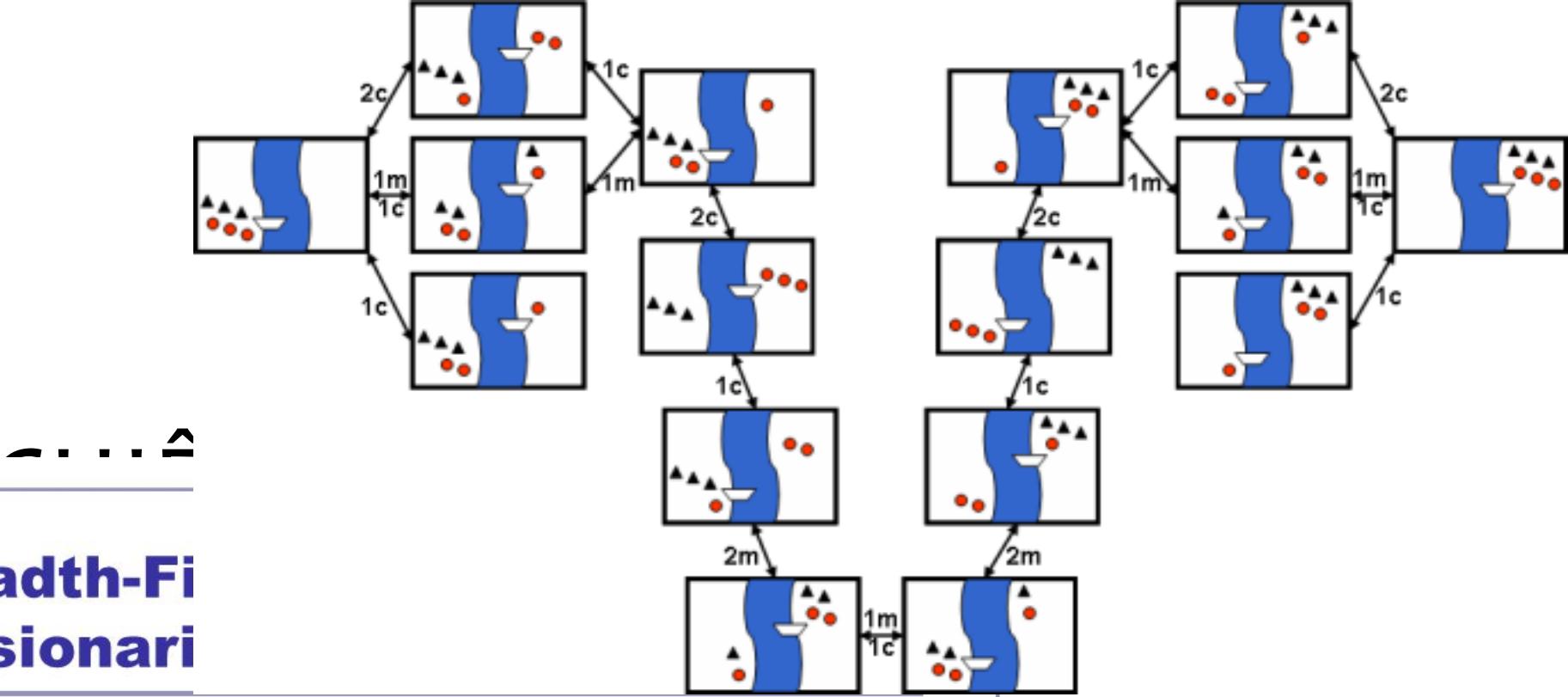


Start state

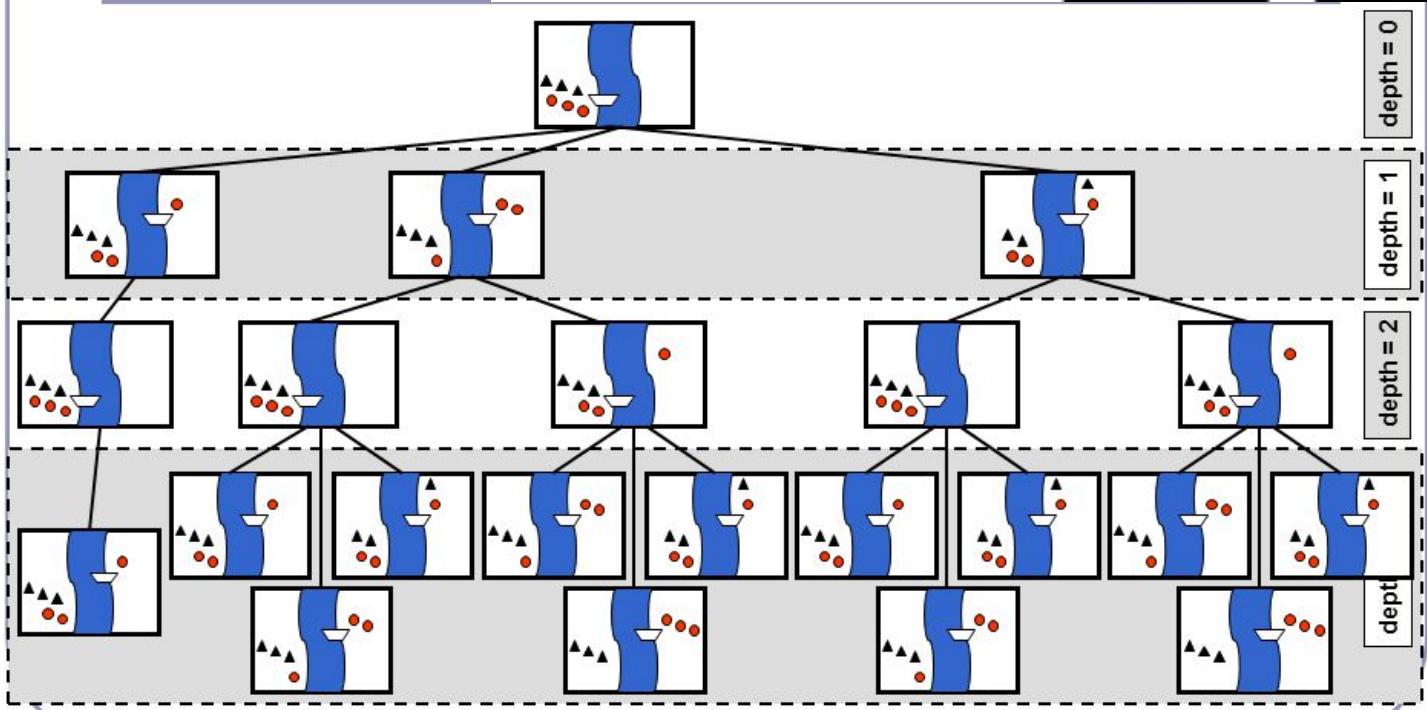
Black triangles represent missionaries.
Red circles represent cannibals.



Goal state

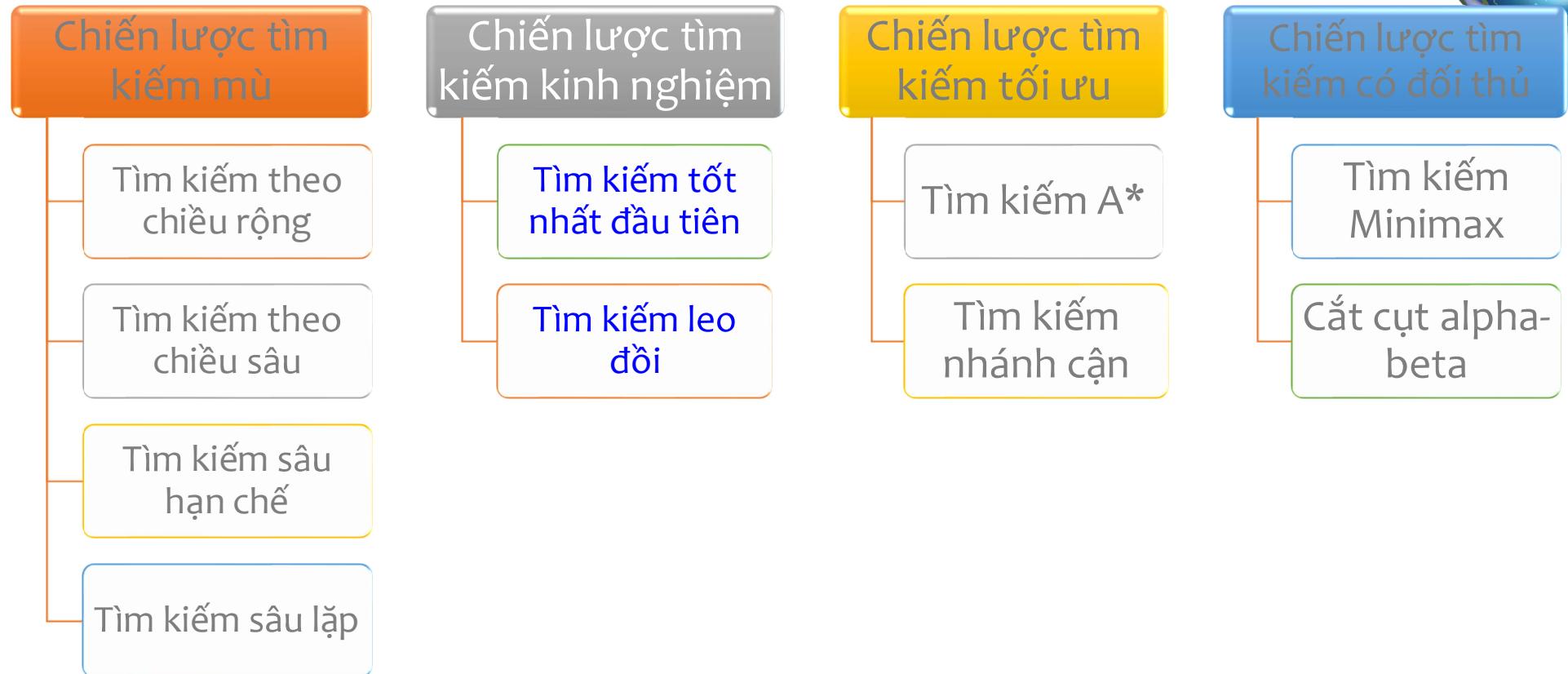


Breadth-Fi Missionari



63

Các thuật toán tìm kiếm



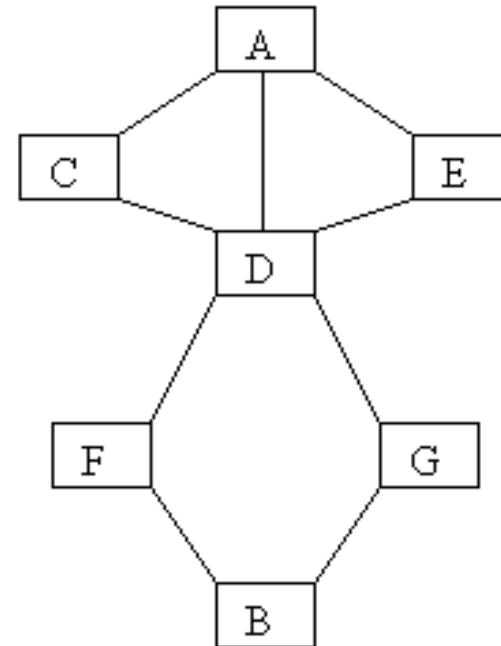
Tìm kiếm kinh nghiệm



- Ví dụ: tìm đường đi từ A đến B

- Biết đường chim bay từ C đến B **gần hơn** đường chim bay từ D đến B và từ E đến B

- Người đi sẽ chọn ???



Hàm đánh giá (heuristic function)

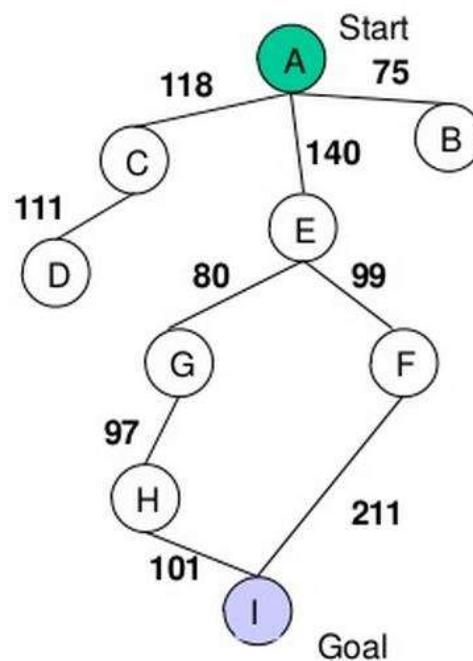


- Hàm đánh giá $h(n)$
 - Là khoảng cách ước lượng để đi từ trạng thái n tới đích.
 - $h(n) = 0 \rightarrow ???$
- Xây dựng hàm đánh giá?



Ví dụ về hàm đánh giá – bài toán tìm đường đi

- Sử dụng độ dài theo đường chim bay từ một thành phố đến thành phố đích làm hàm đánh giá



State	Heuristic: $h(n)$
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	98
I	0

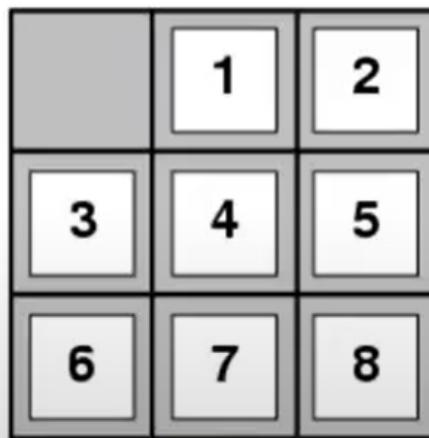
$h(n)$ = straight-line distance heuristic

Ví dụ về hàm đánh giá – Bài toán 8-puzzle



- $h_1(n)$ = number of misplaced tiles (not include the blank)
- $h_2(n)$ = total Manhattan distance (the sum of the distances of the tiles from their goal positions)

$$d_M = a + b = |x_1 - x_2| + |y_1 - y_2|$$



$$a = |x_1 - x_2|$$

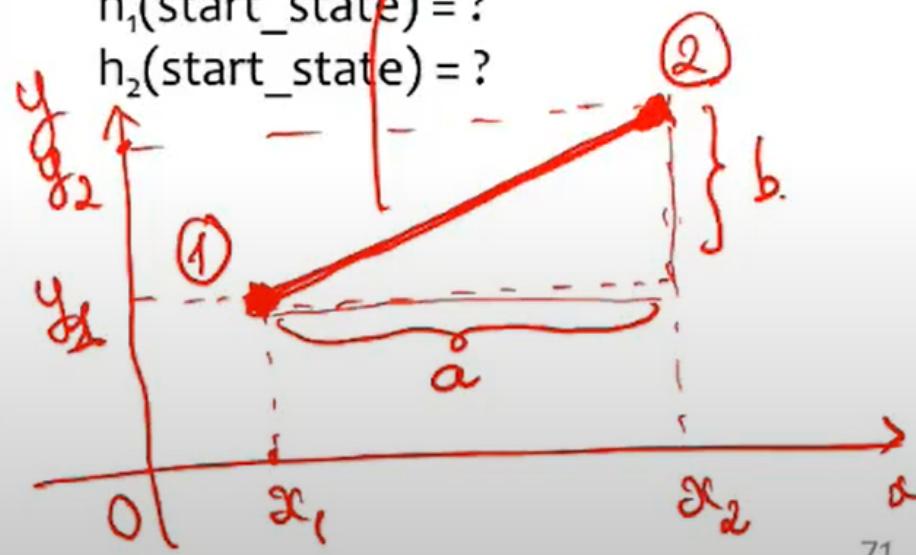
$$b = |y_1 - y_2|$$

Khi Euclidean:

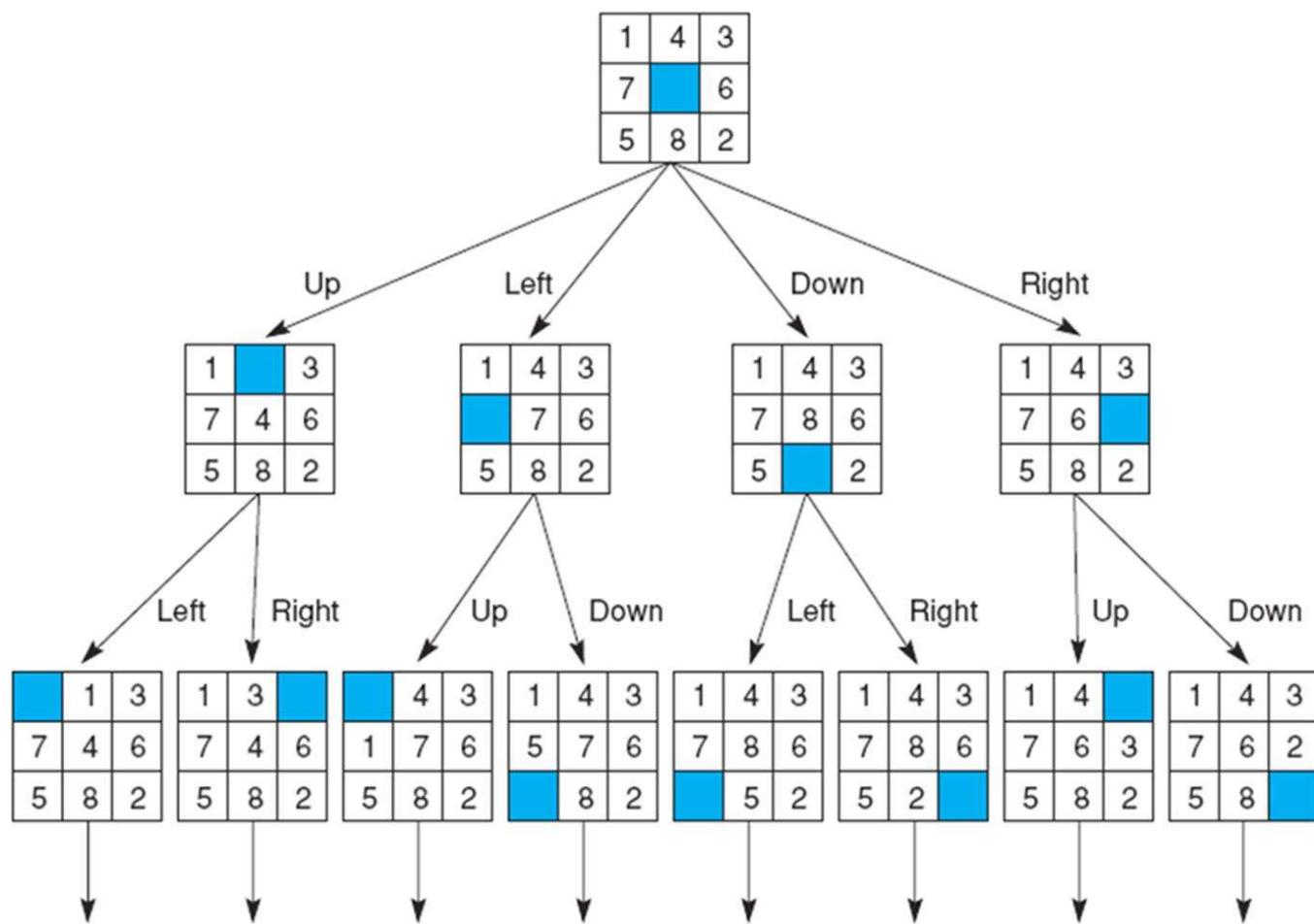
$$d_E = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

$$h_1(\text{start_state}) = ?$$

$$h_2(\text{start_state}) = ?$$



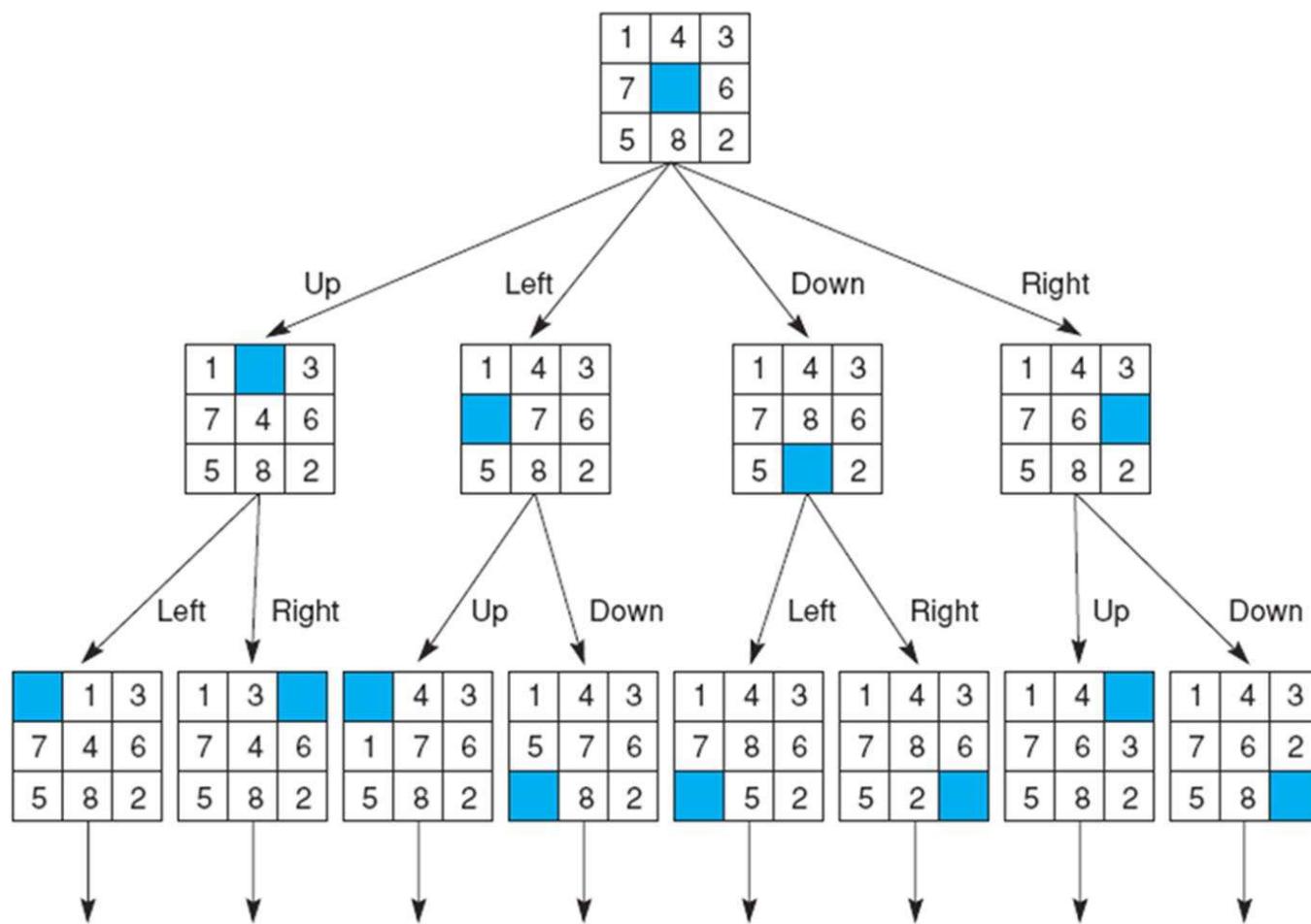
Tính giá trị của hàm đánh giá h_1 tại mỗi trạng thái



1	2	3
4	5	6
7	8	

Goal state

Tính giá trị của hàm đánh giá h_2 tại mỗi trạng thái



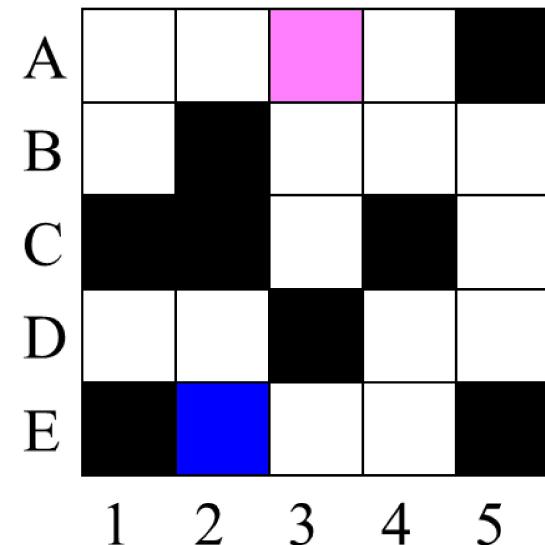
1	2	3
4	5	6
7	8	

Goal state

Ví dụ về hàm đánh giá – Bài toán tìm đường đi cho robot



- **Problem:** To get from square **A3** to square **E2**, one step at a time, avoiding obstacles (black squares).
- **Operators:** (in order)
 - **go_left(n)**
 - **go_down(n)**
 - **go_right(n)**
 - each operator costs 1.
- **Heuristic = ?** → **Manhattan distance**



Vẽ cây tìm kiếm (tới độ sâu 3) và tính giá trị của hàm đánh giá tại mỗi trạng thái.

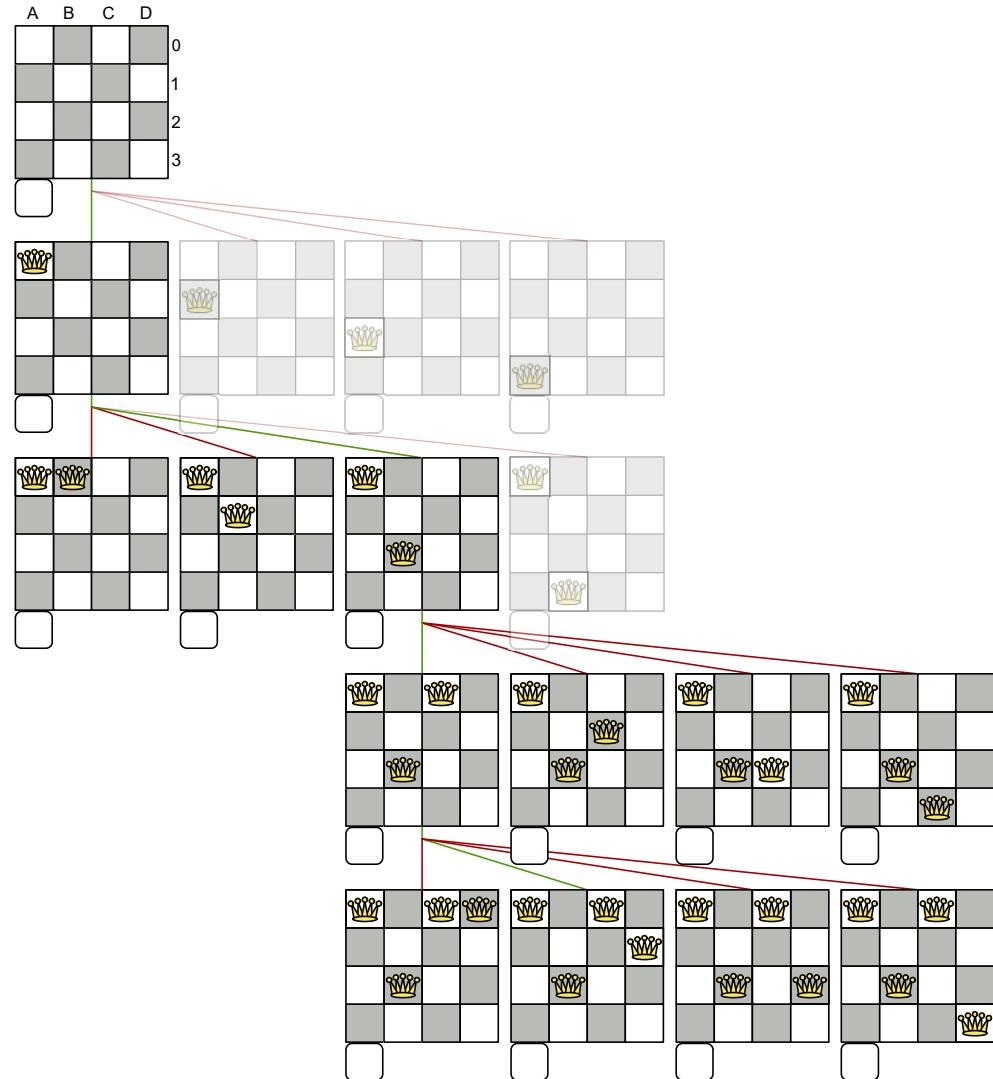


Ví dụ về hàm đánh giá – Bài toán n-queens

- Heuristic function = number of pairs of queens that are attacking each other, either directly or indirectly

18	12	14	13	13	12	14	14
14	16	13	15	12	14	12	16
14	12	18	13	15	12	14	14
15	14	14	15	13	16	13	16
15	14	17	15	14	16	16	16
17	15	16	18	15	17	15	16
18	14	15	15	15	14	16	16
14	14	13	17	12	14	12	18

4-queens: Tính giá trị của hàm đánh giá cho các trạng thái sau:



Nhận xét về hàm đánh giá



- Có thể có nhiều cách xây dựng hàm đánh giá.
- Các hàm đánh giá có thể không phải hàm tối ưu.
- Cách chọn hàm đánh giá quyết định rất nhiều kết quả của các kỹ thuật tìm kiếm kinh nghiệm.

Tìm kiếm tốt nhất đầu tiên (Best First Search)



- Tìm kiếm theo chiều rộng được hướng dẫn bởi hàm đánh giá
- Đỉnh được chọn để phát triển là đỉnh tốt nhất được xác định bởi hàm đánh giá (đỉnh có giá trị hàm đánh giá nhỏ nhất), đỉnh này có thể ở mức (độ sâu) hiện tại hoặc mức trên.

Procedure TimKiemTotNhatDauTien

begin

1. Khởi tạo danh sách L chỉ chứa trạng thái ban đầu
2. loop do

 2.1. if L rỗng then

 {thông báo tìm kiếm thất bại; stop};

 2.2. Loại trạng thái u ở đầu danh sách L;

 2.3. if u là trạng thái kết thúc then

 {thông báo tìm kiếm thành công; stop};

 2.4. for mỗi trạng thái v kề u do

 Xen v vào danh sách L sao cho L được sắp theo thứ tự
 tăng dần của hàm đánh giá;

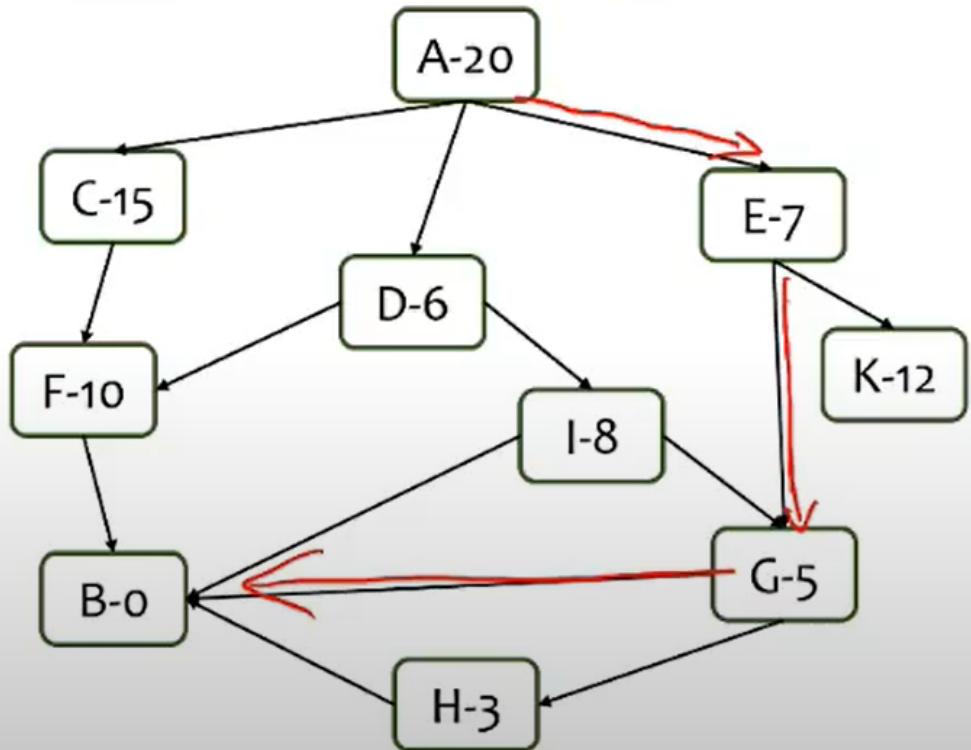
end;

Best First Search – Ví dụ 1



10:22 → 10:40

- Trạng thái bắt đầu: A
- Trạng thái kết thúc = B

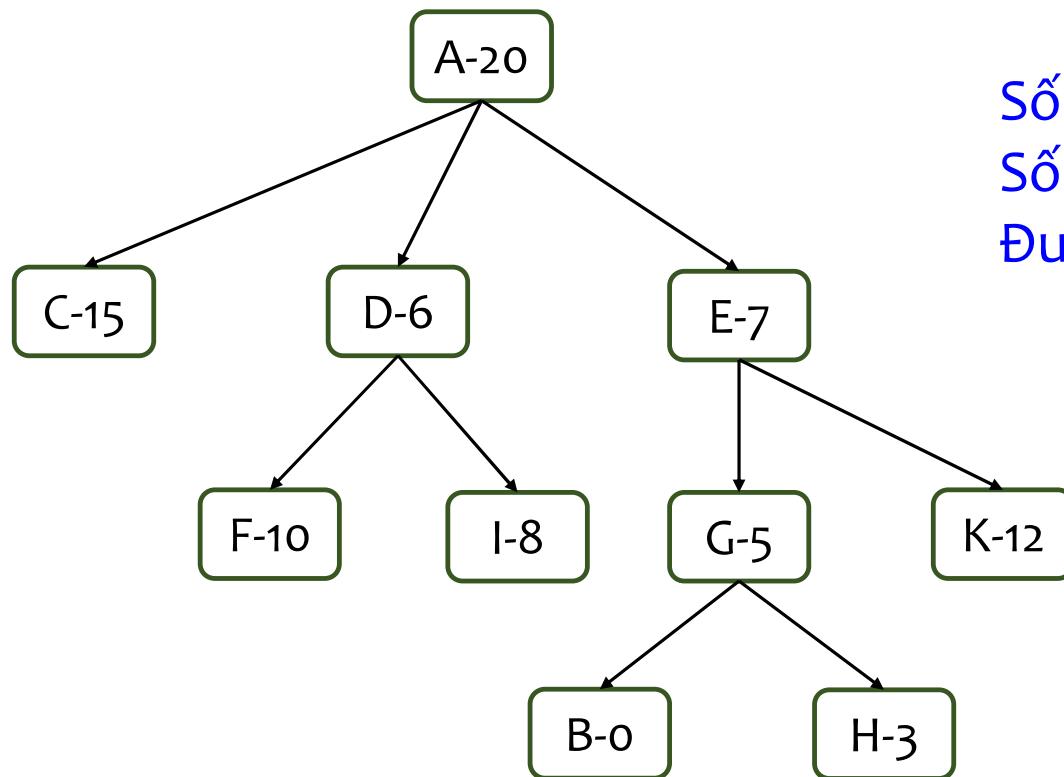


Mô tả quá trình tìm kiếm

- Khởi tạo: L = {A}

Phát triển trạng thái	Trạng thái kề	Danh sách L
A	C, D, E	D, E, C
D	I, F	E, I, F, C
E	G, K	G, I, F, K, C
G	B, H	B, H, E, F, K, C
B	TTRQ - Dừng	

Cây tìm kiếm tương ứng sinh ra khi sử dụng thuật toán Best First Search

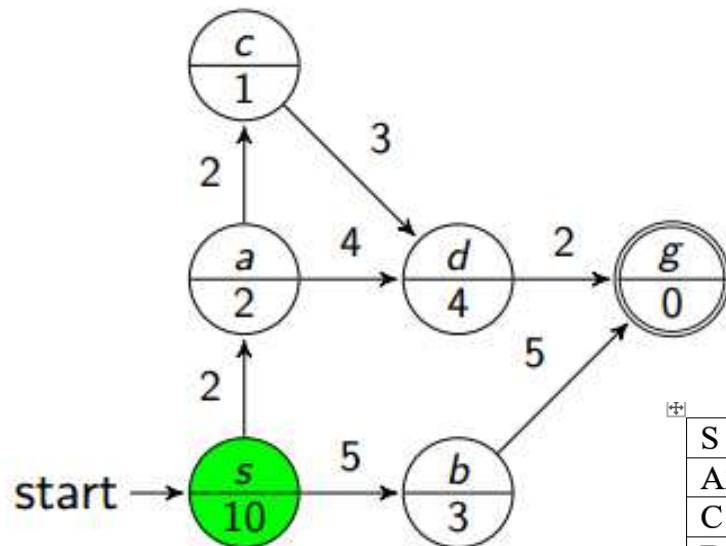


Số trạng thái cần phát triển = ?
Số trạng thái được sinh ra = ?
Đường đi tìm được = ?



Best First Search – Ví dụ 2

- Trạng thái bắt đầu: s
- Trạng thái kết thúc = ?



Mô tả quá trình tìm kiếm

- Khởi tạo: $L = \{s\}$

Phát triển trạng thái	Trạng thái kề	Danh sách L
S	A2 B3	A2 B3
A2	C1 D4	C1 B3 D4
C1		B3 D4
B3	G0	G0 D4
G0	DÙNG	D4
S->B->G		

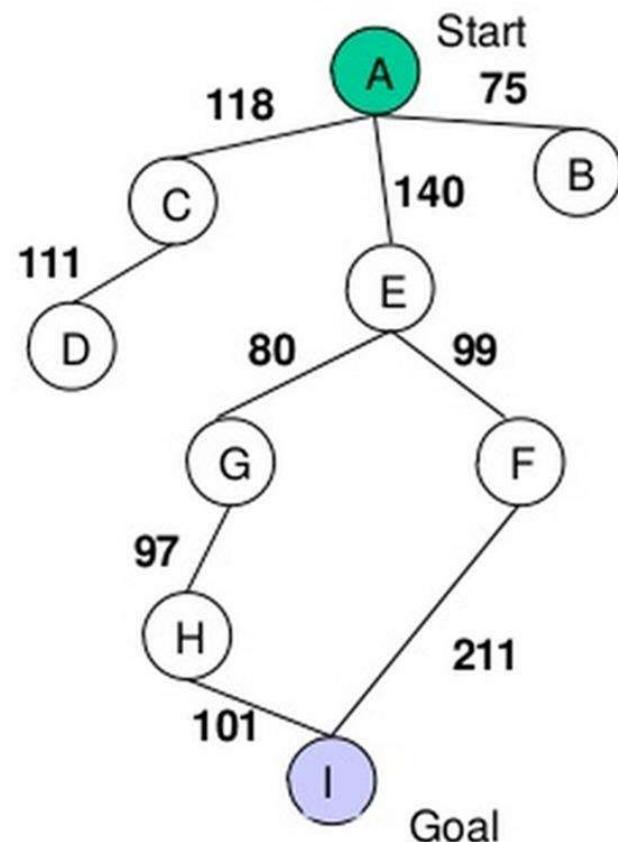
Cây tìm kiếm tương ứng sinh ra khi sử dụng thuật toán Best First Search

Số trạng thái cần phát triển = ?

Số trạng thái được sinh ra = ?

Đường đi tìm được = ?

Best First Search – Ví dụ 3

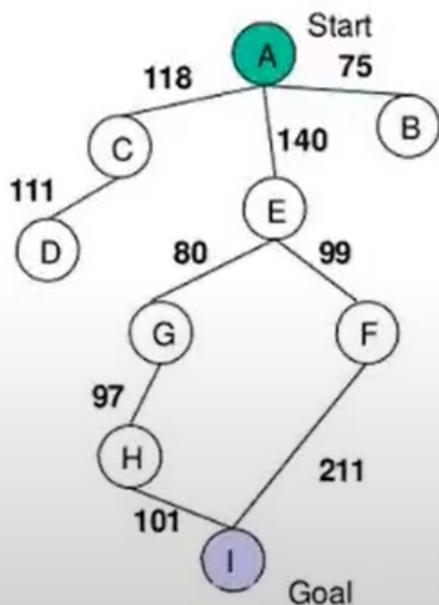


State	Heuristic: $h(n)$
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	98
I	0

$h(n)$ = straight-line distance heuristic

Best First Search – Ví dụ 3

- Trạng thái bắt đầu: A
- Trạng thái kết thúc = I



State	Heuristic: $h(n)$
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	98
I	0

Deep (PT) & TT
Độ : 4

BFS Mô tả quá trình tìm kiếm

- Khởi tạo: $L = \{A\}$

Phát triển trạng thái	Trạng thái kề	Danh sách L
A	B, C, E	{B, C, E}
B.	-	{C, E}
C	D	{E, D}
E.	F, G	{D, F, G}
D	-	{F, G}
F.	I	{G, I}
G	H	{I, H}
I.	TTICT - Dừng	

KL: A \rightarrow E \rightarrow F \rightarrow I .

Cây tìm kiếm tương ứng sinh ra khi sử dụng thuật toán Best First Search

Số trạng thái cần phát triển = ?

Số trạng thái được sinh ra = ?

Đường đi tìm được = ?

Best First Search – Ví dụ 4: bài toán 8-puzzle



- **Định nghĩa trạng thái:**

- Trạng thái được biểu diễn bởi vị trí của các ô (bao gồm cả ô trống)

- **Tập hợp toán tử:**

- Toán tử = Sự dịch chuyển của ô trống
- Bao gồm: Up, Left, Right, Down

- **Hàm đánh giá = Manhattan distance**

- **Trạng thái bắt đầu:**

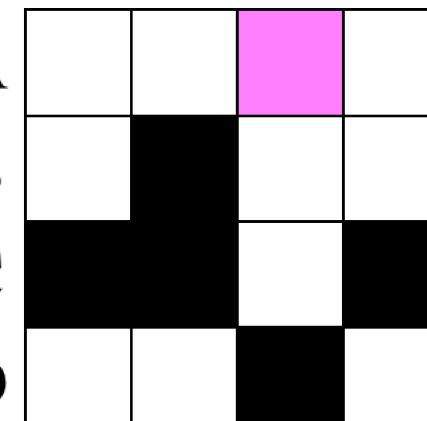
1	2	5
3	4	
6	7	8

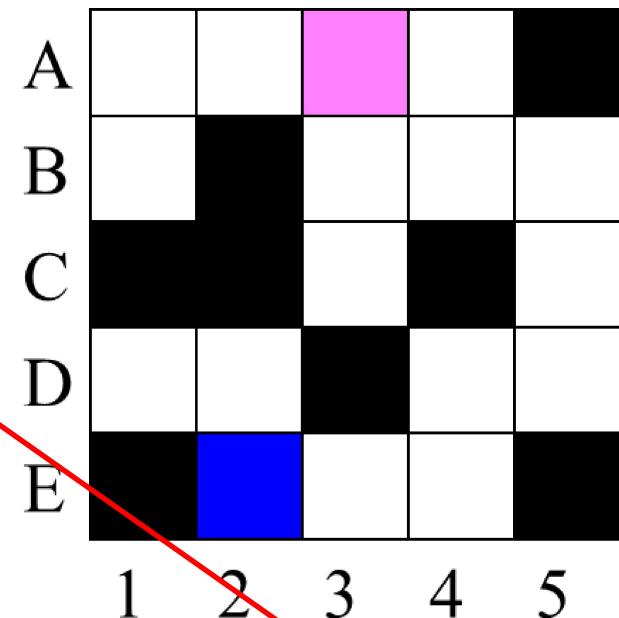
- **Trạng thái kết thúc:**

	1	2
3	4	5
6	7	8

~~Best First Search – Ví dụ 5: tìm đường đi cho robot~~



- **Problem:** To get from square **A3** to square **E2**, one step at a time, avoiding obstacles (black squares).
 - **Operators:** (in order)
 - go_left(n)
 - go_down(n)
 - go_right(n)
 - each operator costs 1.
 - **Heuristic function = Manhattan distance**



Cây tìm kiếm tương ứng sinh ra

Số trạng thái cần phát triển = ?

Số trạng thái được sinh ra = ?

Chuỗi các toán tử cần thực hiện để đi từ trạng thái bắt đầu tới trạng thái kết thúc = ?

Tìm kiếm leo đồi (Hill Climbing Search)



- Tìm kiếm theo chiều sâu được hướng dẫn bởi hàm đánh giá.
- Khi phát triển một đỉnh u thì bước tiếp theo ta **chọn** trong số các đỉnh con của u, **đỉnh có nhiều hứa hẹn nhất** để phát triển, đỉnh này được xác định bởi hàm đánh giá.

Procedure TimKiemLeoDoi

begin

1. Khởi tạo danh sách L chỉ chứa trạng thái ban đầu
 2. loop do
 - 2.1 if L rỗng then

{thông báo tìm kiếm thất bại; stop};
 - 2.2 Loại trạng thái u ở đầu danh sách L;
 - 2.3 if u là trạng thái kết thúc then

{thông báo tìm kiếm thành công; stop};
 - 2.4 for mỗi trạng thái v kề u do

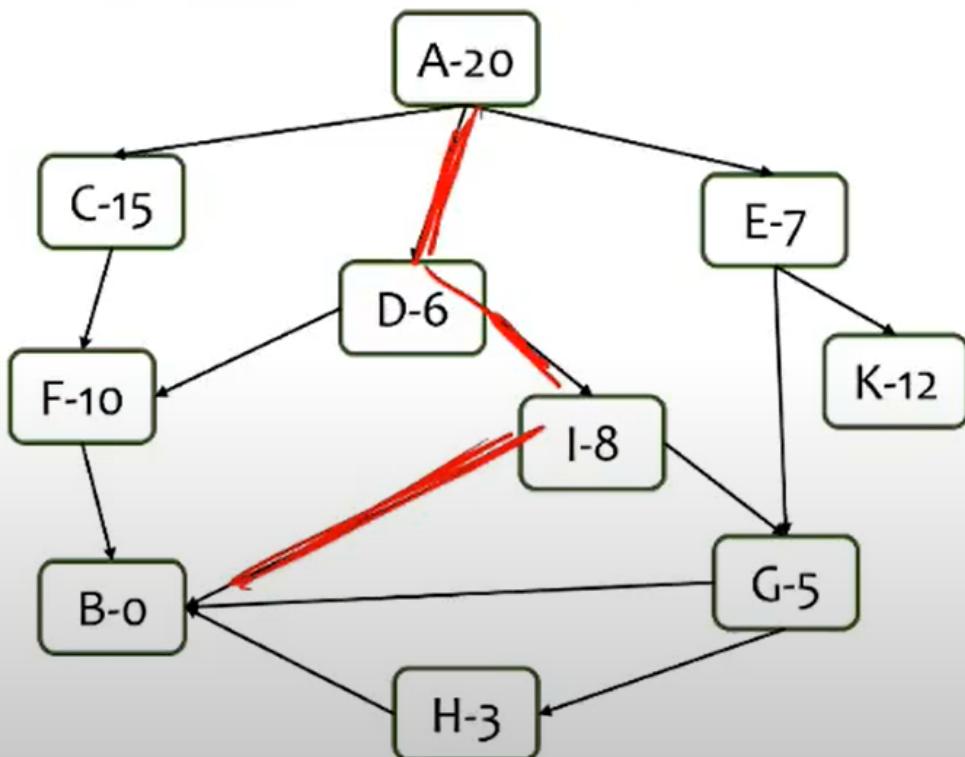
Xen v vào danh sách L1 sao cho L1 được sắp theo
thứ tự tăng dần của hàm đánh giá;
 - 2.5 Chuyển danh sách L1 vào đầu danh sách L
- end;**



Hill Climbing Search – Ví dụ 1

⑤

- Trạng thái bắt đầu: A
- Trạng thái kết thúc = ?



Mô tả quá trình tìm kiếm

- Khởi tạo: $L = \{A\}$

Phát triển trạng thái	Trạng thái kề	Danh sách L
A	D, E, C	D, E, C
D	I, F	I, F, E, C
I	B, G	B, G, I, F, E, C
B	TTC - Dừng	

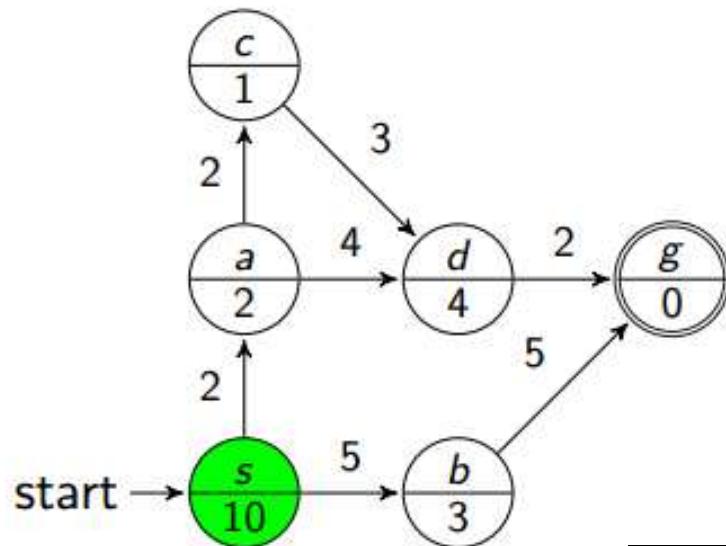
Hill Climbing Search – Ví dụ 2



- Trạng thái bắt đầu: s
 - Trạng thái kết thúc = ?

Mô tả quá trình tìm kiếm

- Khởi tạo: $L = \{s\}$



S	A2 B3	A2 B3
A2	C1 D4	C1 D4 B3
C1		D4 B3
D4	G0	G0 B3
G0	DÙNG	B3
S→A→D→G		

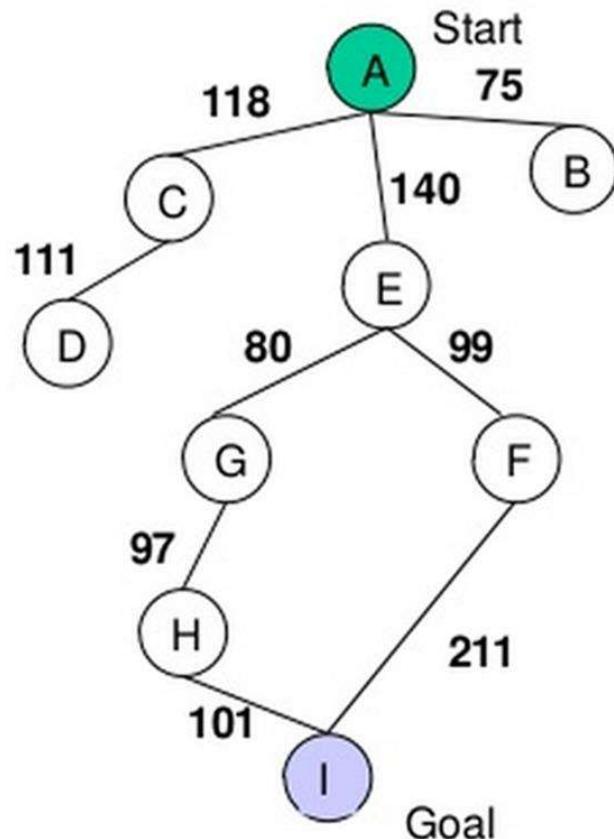
Cây tìm kiếm tương ứng sinh ra

Số trạng thái cần phát triển = ?

Số trạng thái được sinh ra = ?

Đường đi tìm được = ?

Hill Climbing Search – Ví dụ 3



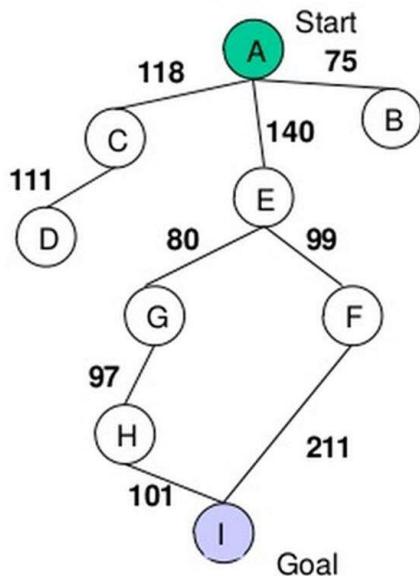
State	Heuristic: $h(n)$
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	98
I	0

$h(n)$ = straight-line distance heuristic

Hill Climbing Search – Ví dụ 3



- Trạng thái bắt đầu: A
 - Trạng thái kết thúc = I



State	Heuristic: $h(n)$
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	98
I	0

Mô tả quá trình tìm kiếm

- Khởi tạo: $L = \{A\}$

Cây tìm kiếm tương ứng sinh ra

Số trạng thái cần phát triển = ?

Số trạng thái được sinh ra = ?

Đường đi tìm được = ?

Hill Climbing Search – Ví dụ 4: bài toán 8-puzzle



- Định nghĩa trạng thái:

- Trạng thái được biểu diễn bởi vị trí của các ô (bao gồm cả ô trắng)

- Tập hợp toán tử:

- Toán tử = Sự dịch chuyển của ô trắng
- Bao gồm: Up, Left, Right, Down

- Hàm đánh giá = Manhattan distance

- Trạng thái bắt đầu:

1	2	3
8	6	
7	5	4

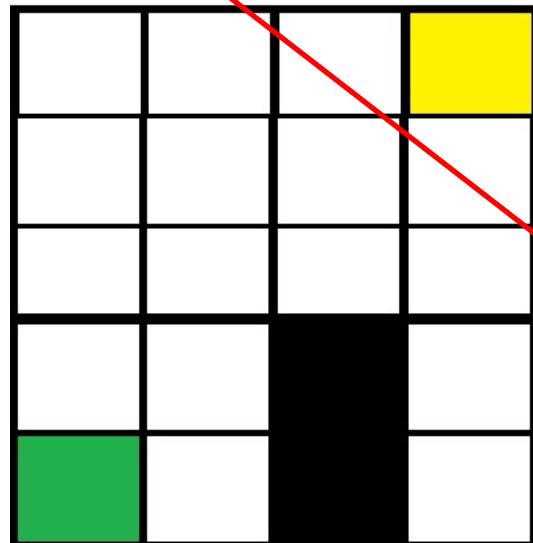
- Trạng thái kết thúc:

1	2	3
8		4
7	6	5

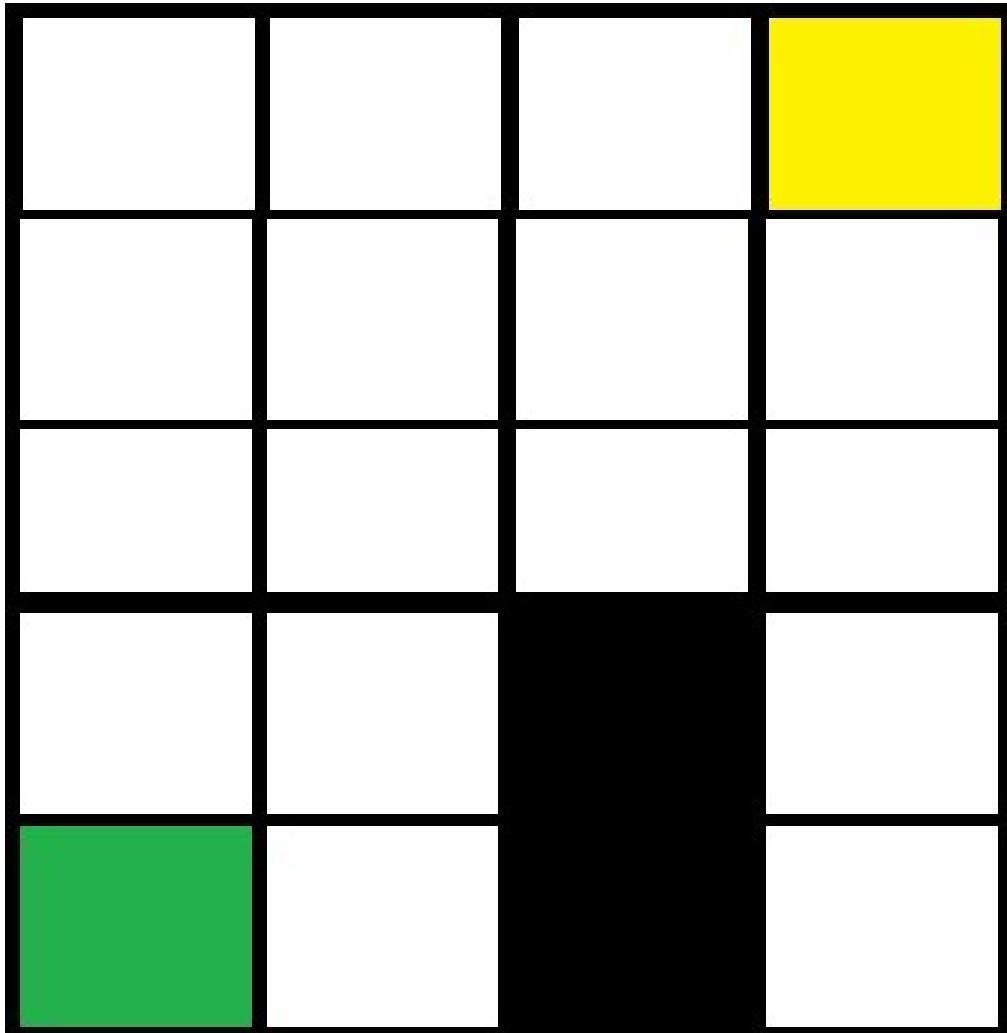
Hill Climbing Search – Ví dụ 5: tìm đường đi cho robot



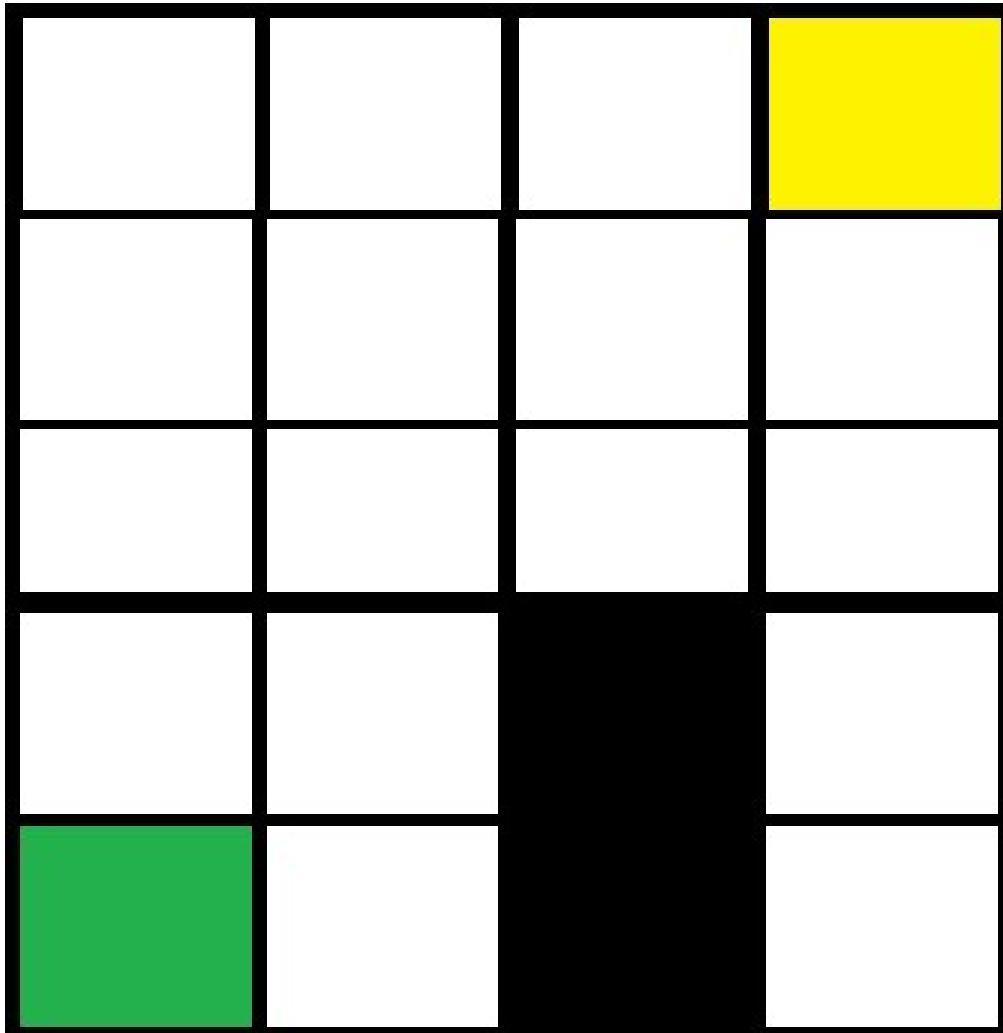
- **Problem:** To get from the green square to the yellow square, one step at a time, avoiding obstacles (black squares).
- **Operators:** The robot can move in 8 directions.
- **Heuristic function = Manhattan distance or Euclidean distance.**



**Heuristic function =
Manhattan distance**

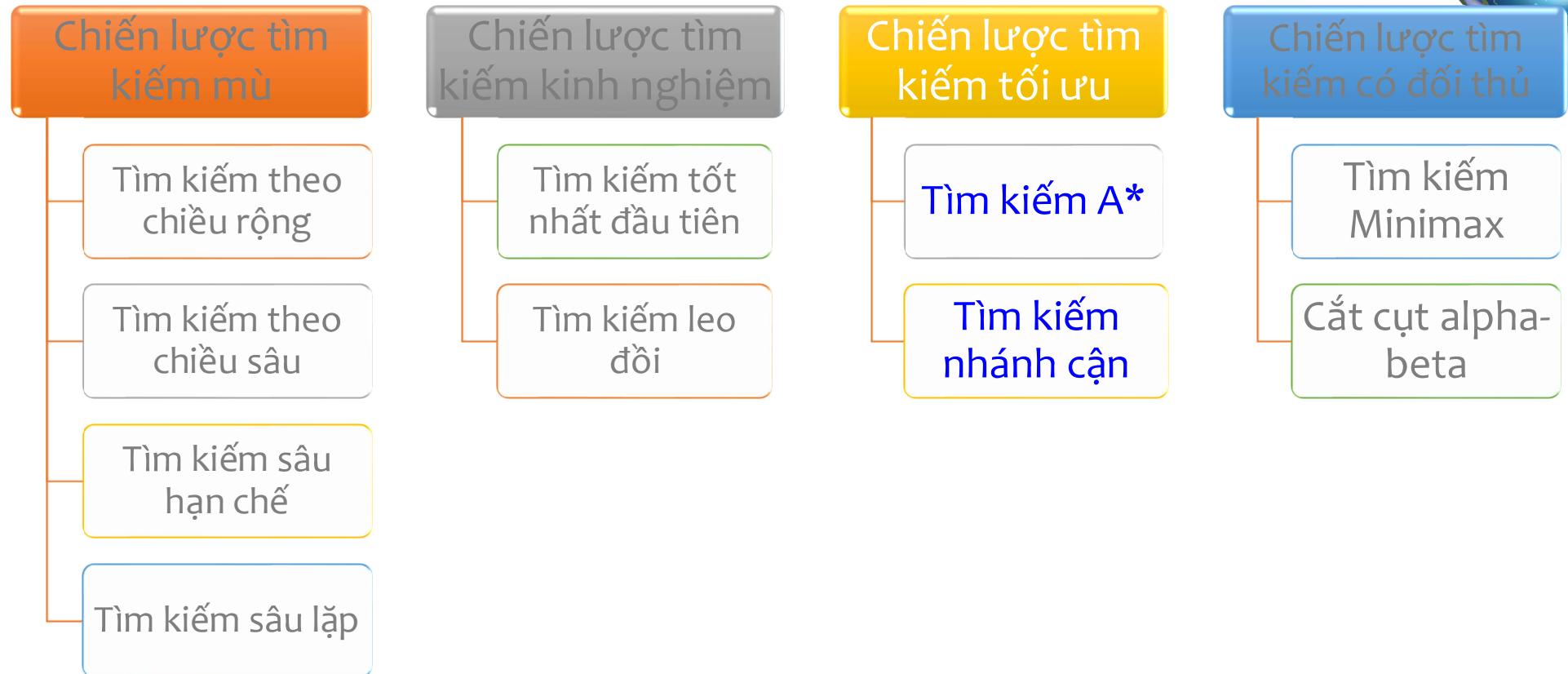


**Heuristic function =
Euclidean distance**



CHIẾN LƯỢC TÌM KIẾM TỐI ƯU

Các thuật toán tìm kiếm



Bài toán tìm đường đi ngắn nhất

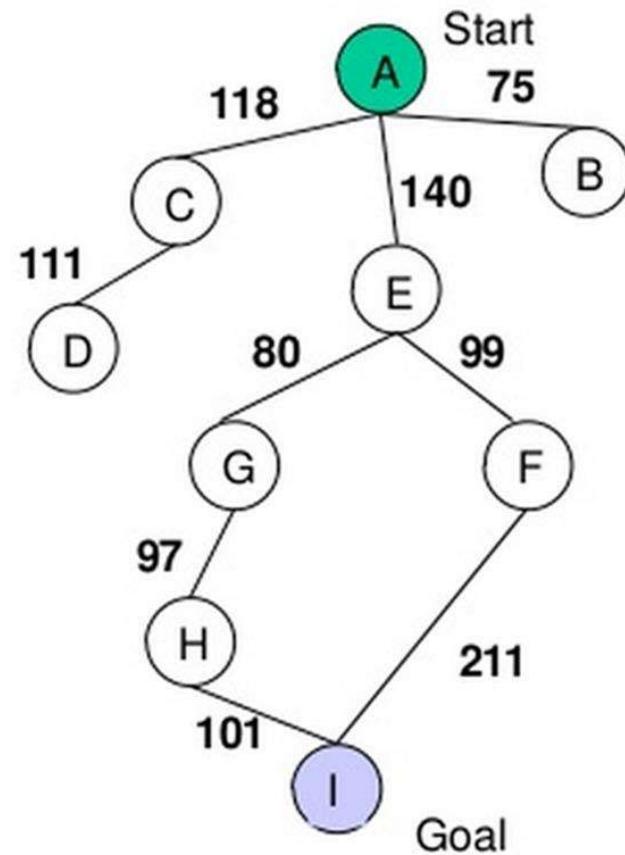


- Cho một không gian trạng thái.
- Giả sử **chi phí** để đưa trạng thái u thành v là $k(u,v) \geq 0$.
- **Tìm đường đi** từ trạng thái đầu đến trạng thái đích, sao cho **tổng chi phí** là **nhỏ nhất**.
- Ví dụ:
 - Bài toán tìm đường trên bản đồ giao thông. Tìm đường đi ngắn nhất từ A đến B

BT tìm đường đi ngắn nhất



- Giải pháp
 - Trước đây?
 - Bây giờ
 - Tìm kiếm A*
 - Tìm kiếm nhánh cận



kết cách với lượng tử $\leftarrow f(B) = g(B) + h(B)$

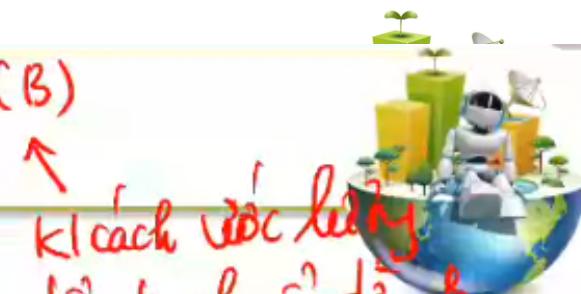
~~trạng thái bắt đầu tới trạng thái đích~~ \leftarrow tổng kết cách để di chuyển từ trạng thái bắt đầu

Tìm kiếm tốt nhất đầu tiên với hàm đánh giá $f(u)$

Hàm đánh giá $f(u) = g(u) + h(u)$ tới trạng B.

$g(u)$ = độ dài đường đi ngắn nhất từ trạng thái bắt đầu đến u

$h(u)$ = độ dài đường đi ngắn nhất từ u đến đích



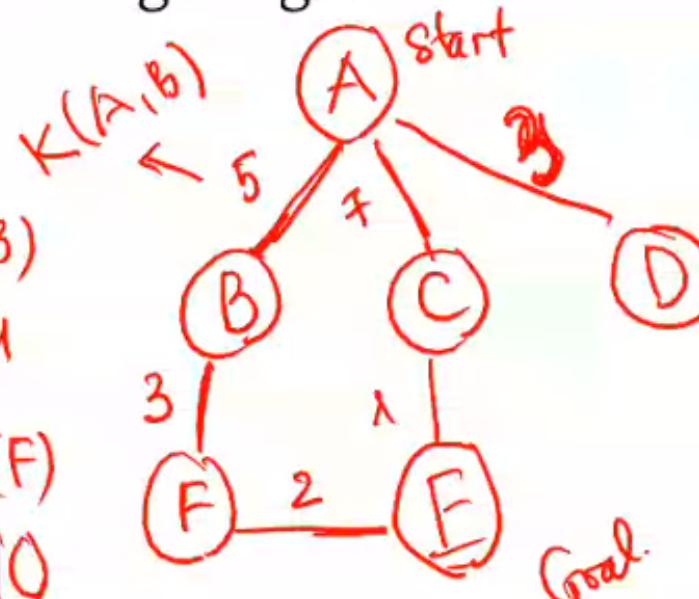
kết cách với lượng tử để di chuyển từ B

tới đích

$$u = B:$$

$$\begin{aligned}f(B) &= g(B) + h(B) \\&= 5 + 6 = 11\end{aligned}$$

$$\begin{aligned}f(F) &= g(F) + h(F) \\&= 8 + 2 = 10\end{aligned}$$



Trạng thái	$h(u)$ - kinh nghiệm
A	10
B	6
C	3
D	9
E	0

$h(F) = 2$.

Procedure TimKiemA*

begin

1. Khởi tạo danh sách L chỉ chứa trạng thái ban đầu

2. loop do

 2.1 if L rỗng then

 thông báo tìm kiếm thất bại;
 stop;

 2.2 Loại trạng thái u ở đầu danh sách L;

 2.3 if u là trạng thái kết thúc then

 thông báo tìm kiếm thành công;
 stop;

 2.4 for mỗi trạng thái v kề u do

$g(v) = g(u) + k(u,v);$

$f(v) = g(v) + h(v);$

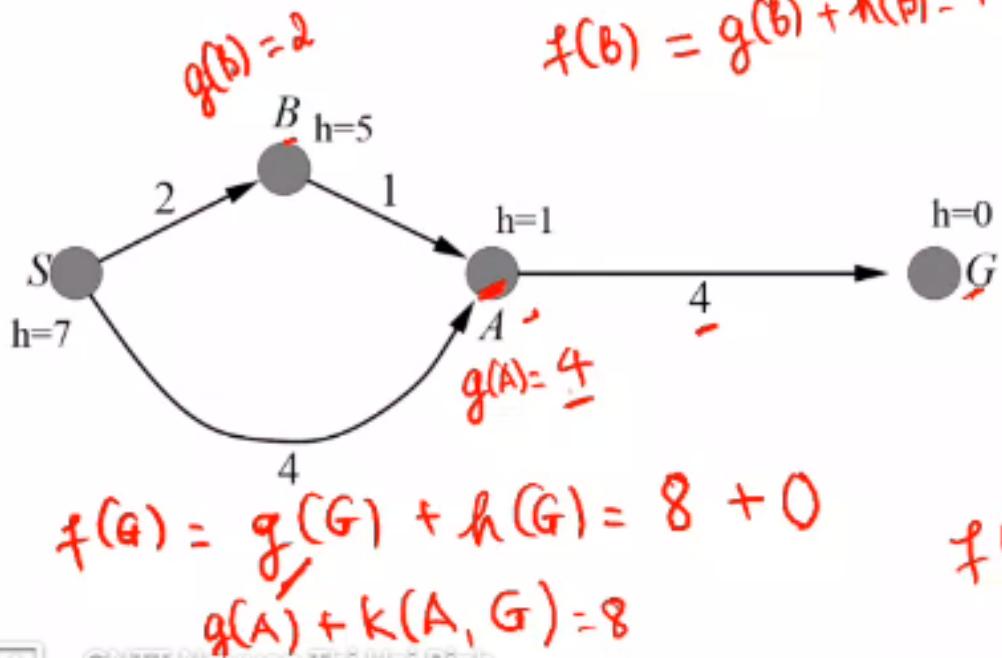
 Xen v vào danh sách L sao cho L được sắp theo thứ tự tăng dần của f;

end;

A* – Ví dụ 1

$S \rightarrow B \rightarrow A \rightarrow G$

- Trạng thái bắt đầu: S
- Trạng thái kết thúc = ?G.



Mô tả quá trình tìm kiếm

- Khởi tạo: L = {S}

Phát triển trạng thái	Trạng thái kề	Danh sách L
S	A_5 , B_7	A_5^S, B_7
A	G_8	B_7, G_8^A
B	A_4	A_4, G_8
A	G_7	G_7, G_8
(G)	TTKT - Duy-	

$$f(G) = g(G) + h(G) = 0 + 0 = 0$$

$$g(A) + h(A, G)$$

$$f(A) = g(A) + h(A) = 3 + 1 = 4$$

$$g(B) + h(A, B)$$



Cây tìm kiếm tương ứng sinh ra

Số trạng thái cần phát triển = ?

Số trạng thái được sinh ra = ?

Đường đi tìm được = ?

A* - Ví dụ 2

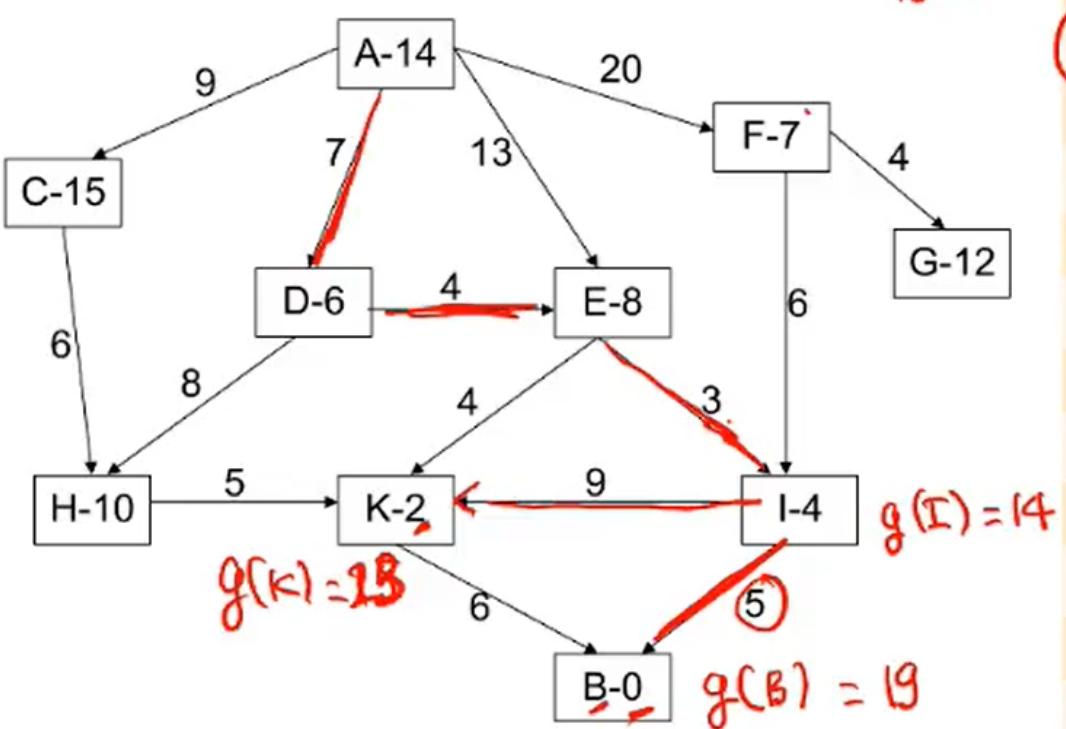
Best First

Nhóm

Mô tả quá trình tìm kiếm

- Khởi tạo: $L = \{A\}$

- Trạng thái bắt đầu: A
- Trạng thái kết thúc = ? B.



Phát triển trạng thái	Trạng thái kề	Danh sách L
→ A	C_{47}, D_{43}, E_{41} F_{57}	D, E, C, F
→ D	H_{55}, E_{49}	$E_{49}^D, E_{41}^A, C, H, F$
→ E	K_{23}, I_{18}	K, I, E_{41}^A, C, H, F
→ K	B_{21}	I, E, B, C, H, F
→ I	B_{19}, K_{23} TTCCT - Đã y	B_{19}, E, B, C, H, F

KL: $A \rightarrow D \rightarrow E \rightarrow I \rightarrow B$.

Cây tìm kiếm tương ứng sinh ra

Số trạng thái cần phát triển = ?

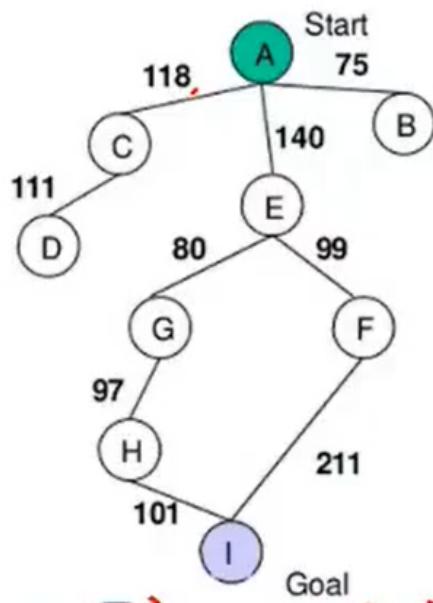
Số trạng thái được sinh ra = ?

Đường đi tìm được = ?

A* – Ví dụ 3

10:03 → 10:13.

- Trạng thái bắt đầu: A
 - Trạng thái kết thúc = ?



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

State	Heuristic: $h(n)$
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	98
I	0

Mô tả quá trình tìm kiếm

- Khởi tạo: $L = \{A\}$

Phát triển trạng thái	Trạng thái kè	Danh sách L
A	B ₄₄₉ , C ₄₄₇ , E ₃₉₃	E, C, B
E	G ₄₄₃ , F ₄₁₇	G, F, C, B
G	H ₄₄₅ ,	H, F, C, B
H	I ₄₁₈	F, I, C, B
F	I ₄₅₀ ,	I ₄₁₈ ^H , C, B, I ₄₅₀
I	TTKT - Duy.	
$A \rightarrow E \rightarrow G \rightarrow H \rightarrow I$.		418

Cây tìm kiếm tương ứng sinh ra

Số trạng thái cần phát triển = ?

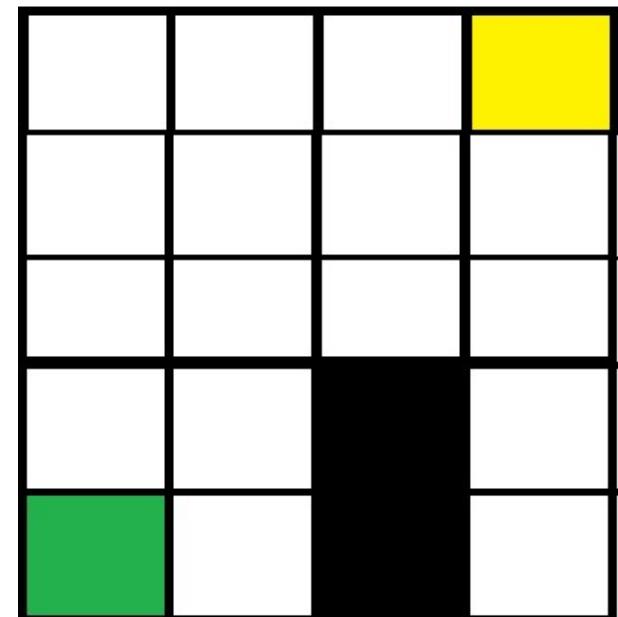
Số trạng thái được sinh ra = ?

Đường đi tìm được = ?

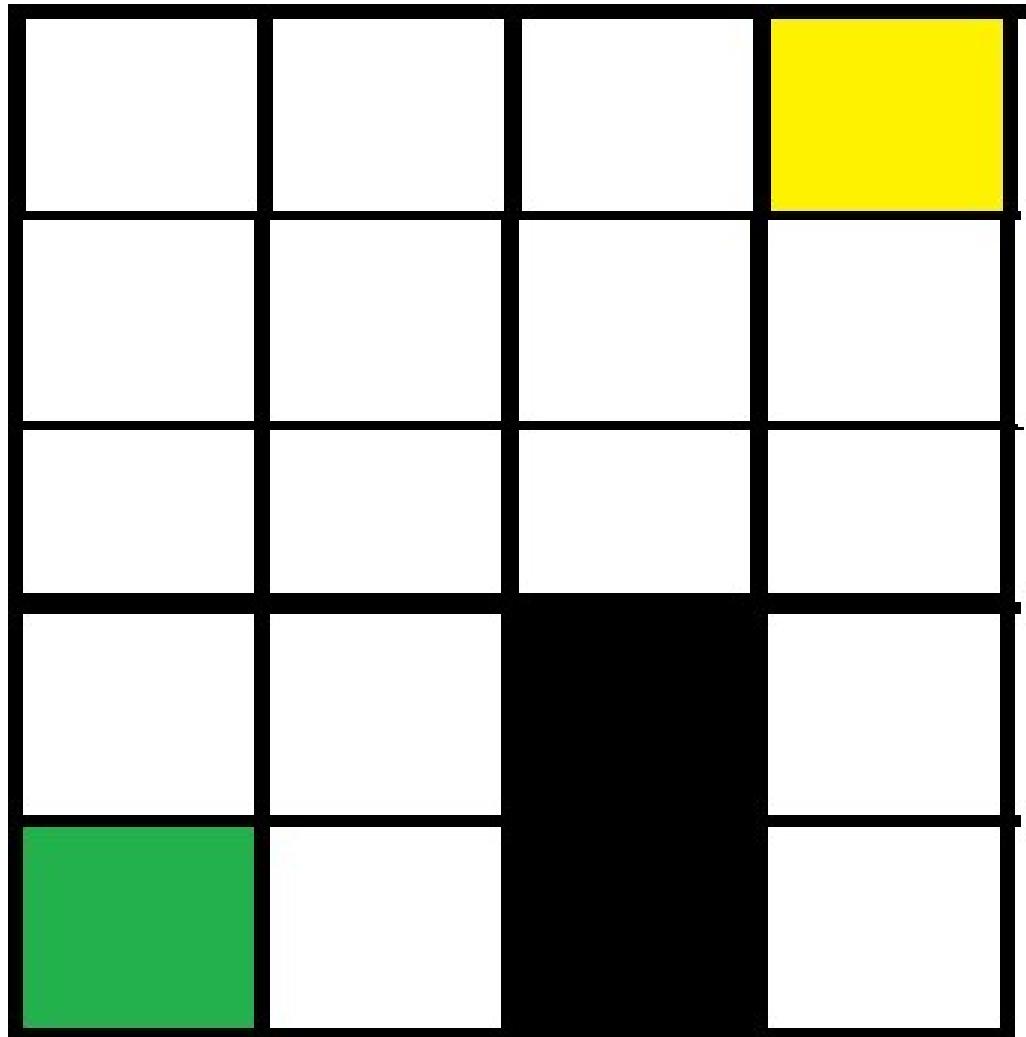
A* – Ví dụ 4: tìm đường đi cho robot



- **Problem:** To get from the green cell to the yellow cell, one step at a time, avoiding obstacles (black squares).
- **Operators:** The robot can move in 8 directions.
- **$g(n)$ = the cost that has been accrued in reaching the cell.**
 - Cost to move from cell A to cell B = $\text{Euclidean_distance}(A, B)$
- **$h(n) = \text{Manhattan_distance}(n, goal_cell)$**
- **$f(n) = ?$**



$g(n) = \text{Euclidean_distance}(\text{start_state}, n)$
 $h(n) = \text{Manhattan_distance}(n, \text{goal_state})$
 $f(n) = g(n) + h(n)$



Tìm kiếm nhánh và cận



- Sử dụng kỹ thuật tìm kiếm leo đồi với hàm đánh giá $f(u)$.
- Hàm đánh giá $f(u) = g(u) + h(u)$
 - $g(u)$ = độ dài đường đi ngắn nhất từ trạng thái bắt đầu đến u
 - $h(u)$ = độ dài đường đi ngắn nhất từ u đến đích

Procedure TimKiemNhanhVaCan

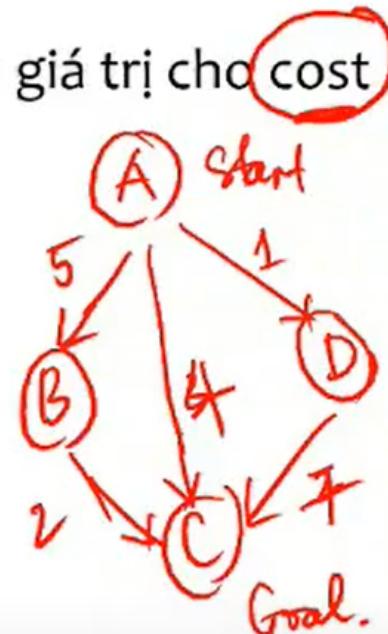
begin

1. Khởi tạo danh sách L chỉ chứa trạng thái ban đầu; gán giá trị cho cost
2. loop do
 - 2.1 if L rỗng then stop
 - 2.2 Loại trạng thái u ở đầu danh sách L;
 - 2.3 if u là trạng thái kết thúc then
 - if $g(u) \leq cost$ then {cost = $g(u)$; quay lại 2.1}
 - 2.4. if $f(u) > cost$ then Quay lại 2.1
 - 2.5 for mỗi trạng thái v kề u do
 - { $g(v) = g(u) + k(u,v)$; $f(v) = g(v) + h(v)$;
Xen v vào danh sách L1 sao cho L1 được sắp theo thứ tự tăng dần
của f};
 - 2.6 Chuyển danh sách L1 vào đầu danh sách L

end;

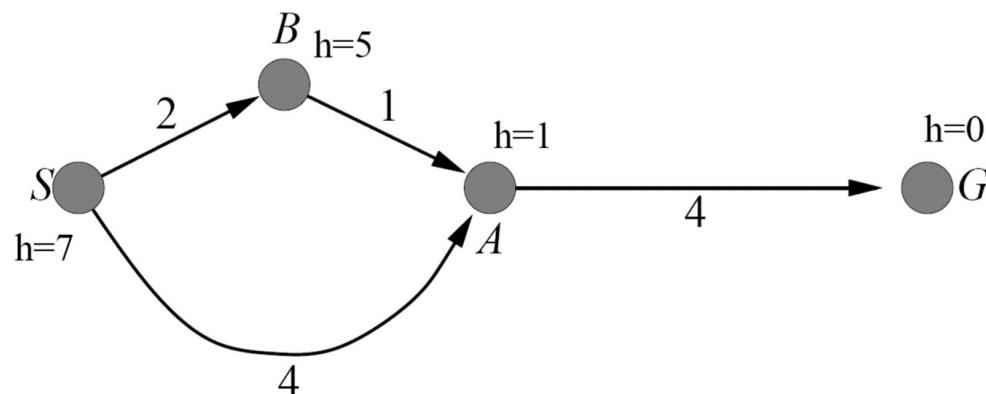
~~A → D → C : cost = 7~~

$A \rightarrow B \rightarrow C : cost = 7$
 $A \rightarrow C : cost = 4$



Nhánh và cận – Ví dụ 1

- Trạng thái bắt đầu: S
 - Trạng thái kết thúc = ?



Mô tả quá trình tìm kiếm

- Khởi tạo: $L = \{S\}$

Phát triển trạng thái	Trạng thái	Danh sách Cost
S7	A5,B7	A5,B7
A5	A4,G8	A4,G8,B7
A4	G7	G7,G8,B7
G7		G8,B7
G8		B7
B7		cost = 7

Cây tìm kiếm tương ứng sinh ra

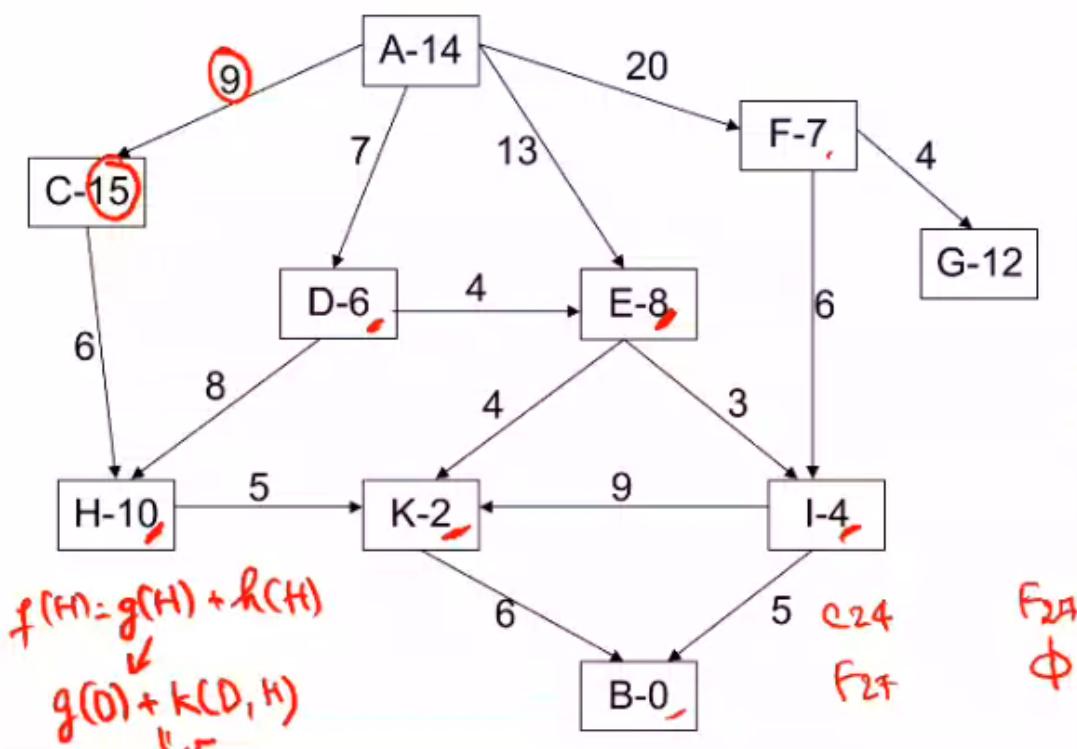
Số trạng thái cần phát triển = ?

Số trạng thái được sinh ra = ?

Đường đi tìm được = ?

Nhánh và cận – Ví dụ 2

- Trạng thái bắt đầu: A
- Trạng thái kết thúc = ?



Mô tả quá trình tìm kiếm

- Khởi tạo: $L = \{A\}$

$cost = +\infty$

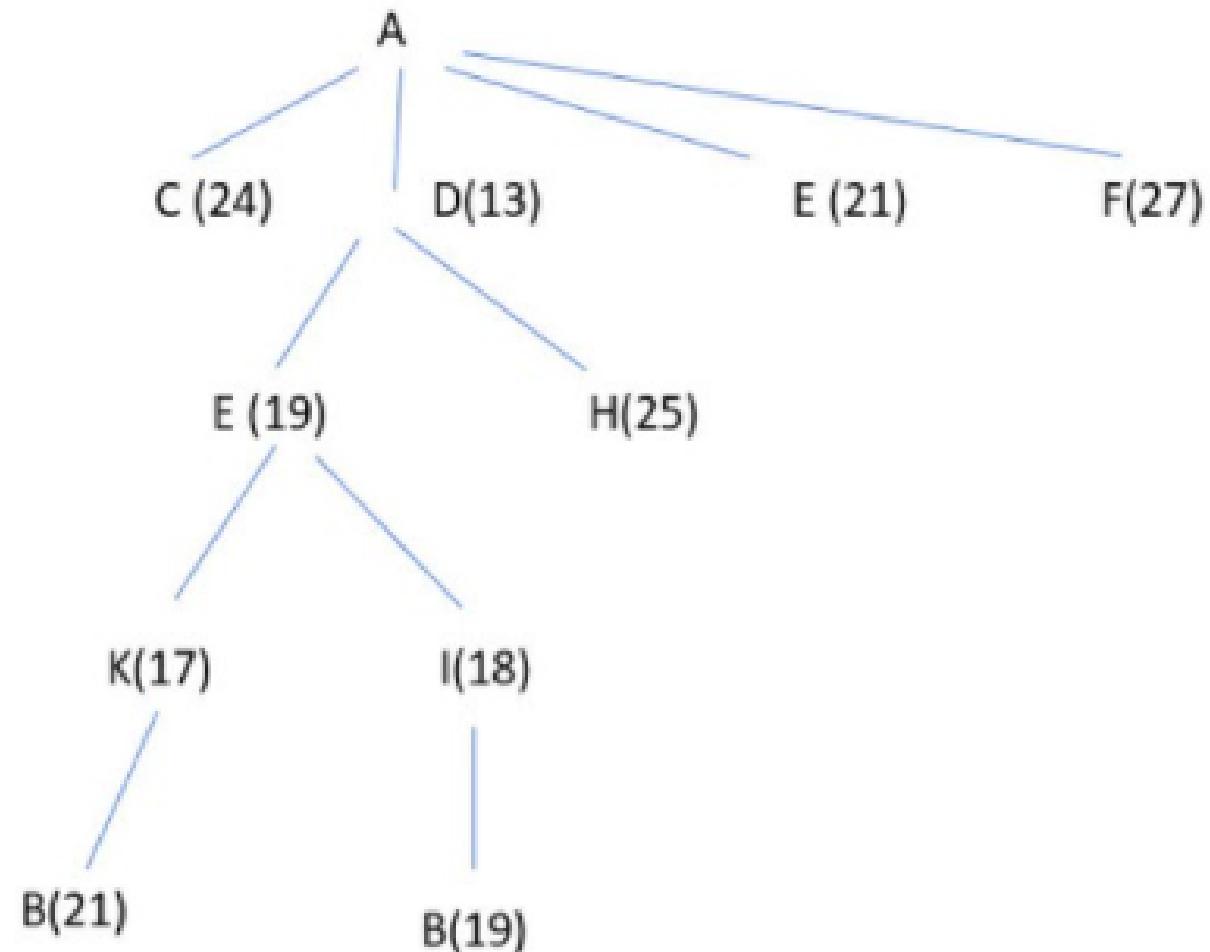
Phát triển trạng thái	Trạng thái kè	Danh sách L
A	$C_{24}^g, D_B^7, E_{21}^{13}, F_{27}^{20}$	$D_{13}, E_{21}, C_{24}, F_{27}$
D	$H_{25}^{25}, E_{19}^{11}, K_{17}^{15}, I_{18}^{14}$	$E_{19}, H_{25}, E_{21}, C_{24}, G_7$
E ₁₉	K_{17}^{15}, I_{18}^{14}	$K_{17}, I_{18}, H_{25}, E_{21}, C_{24}, F_{27}$
K ₁₇	B_{21}^{21}	$B_{21}, I_{18}, H_{25}, E_{21}, C_{24}, F_{27}$
B ₂₁	TTKT $\rightarrow cost = 21$	I_{18}, H_{25}, \dots
I ₁₈	K_{25}^{23}, B_{19}^{19}	$B_{19}, K_{25}, H_{25}, \dots$
B ₁₉	TTKT $\rightarrow cost = 19$	K_{25}, H_{25}, \dots
K ₂₅		$H_{25}, E_{21}, C_{24}, F_{27}$
E ₂₁		C_{24}, F_{27}

Cây tìm kiếm tương ứng sinh ra

Số trạng thái cần phát triển = ?

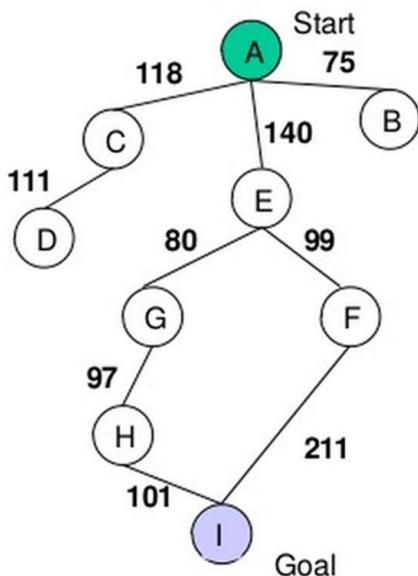
Số trạng thái được sinh ra = ?

Đường đi tìm được = ?



Nhánh và cận – Ví dụ 3

- Trạng thái bắt đầu: A
- Trạng thái kết thúc = ?



State	Heuristic: $h(n)$
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	98
I	0

Mô tả quá trình tìm kiếm

- Khởi tạo: $L = \{A\}$

Phát triển trạng thái	Trạng thái kề	Danh sách L
A	$C_{447}, E_{393}, B_{449}$	E, C, B
E	G_{413}, F_{417}	G, F, C, B
G	H_{415}	H, F, C, B
H	I_{418}	I, F, C, B
I	TTKT \rightarrow cost: 418	F, C, B
F	I_{450}	I, C, B
I		C, B
C		B
B		\emptyset - Dừng

$A \rightarrow E \rightarrow G \rightarrow H \rightarrow I$.

Cây tìm kiếm tương ứng :

Số trạng thái cần phát triển

Số trạng thái được sinh ra :

Đường đi tìm được = ?

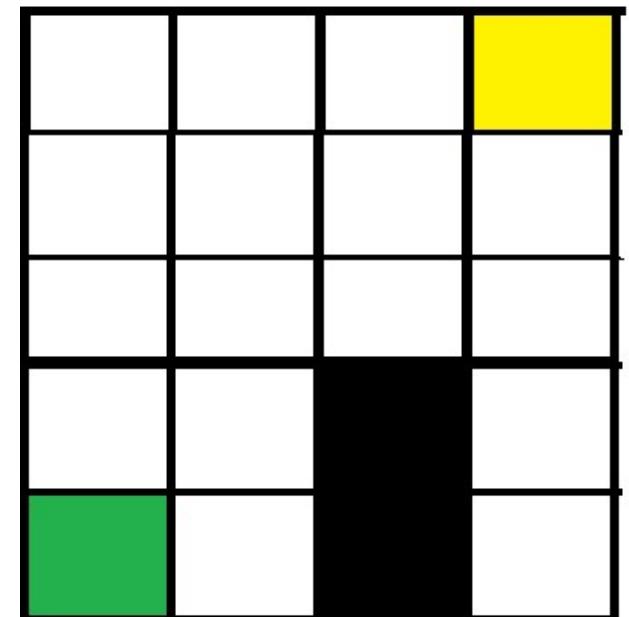
Phát triển trạng thái	Trạng thái kè	L
A366	C447,B449,E393	E393,C447,B449
E393	G413,F417	G413,F417,C447,B449
G413	H415	H415,F417,C447,B449
H415	I418	I418, F417,C447,B449
I418	TTKT->cost=418	F417,C447,B449
F417	I450	I450,C447,B449
I450	TTKT ->cost = 418	C447,B449

C447	D473	B449
B449	Rỗng	
	<input checked="" type="checkbox"/>	<input type="checkbox"/>
đường đi:	Khoảng cách từ A->I	
A->E->G->H->I	cost = 418	

Nhánh và cận – Ví dụ 4: tìm đường đi cho robot



- **Problem:** To get from the green cell to the yellow cell, one step at a time, avoiding obstacles (black squares).
- **Operators:** The robot can move in 8 directions.
- **$g(n)$ = the cost that has been accrued in reaching the cell.**
 - Cost to move from cell A to cell B = $\text{Euclidean_distance}(A, B)$
- **$h(n) = \text{Manhattan_distance}(n, goal_cell)$**
- **$f(n) = ?$**



A*

trivium

$$g(n) = \text{Euclidean_distance}(\text{start_state}, n)$$

$$h(n) = \text{Manhattan_distance}(n, \text{goal_state})$$

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

$$L = \{E_1\}$$

$$u = E_1 \rightarrow L = \{D_2, D_1, E_2\}$$

$$u = D_2 \rightarrow L = \{C_3, C_2, \cancel{D_1}\}$$

Bear ~~way~~
E2 ~~7~~, C1 ~~8~~
~~E2 7, C1 8~~

$$u = C_3 \rightarrow L = \{B_4, B_3, C_4, C_2, E_2, B_2, D_4, \dots, C_1\}$$

$$u = B_4 \rightarrow L = \{A_4, \dots\}$$

1	2	3	4
H = 1 shark car	Lap	G = 5.6 H = 1 F = 6.6	G = 5.2 H = 0 F = 5.2
G = 4.2 H = 3 F = 7.2	G = 3.8 H = 2 F = 5.8	G = 4.2 H = 1 F = 5.2	G = 3.8 H = 2 F = 5.8
G = 2.8 H = 5 F = 7.8	G = 2.4 H = 4 F = 6.4	G = 3.8 H = 3 F = 5.8	G = 3.8 H = 2 F = 5.8
G = 1 H = 6 F = 7	G = 1.4 H = 5 F = 6.4	G = 2.4 H = 6 F = 7.2	
		G = 1 H = 6 F = 7	G = 2.4 H = 6 F = 8.4

Một số bài toán



- Bài toán người bán hàng (TSP – Travelling Salesman Problem)
- Bài toán phân công công việc

Tự học
Tham khảo mục 3.3 trong giáo trình