

Chương 2. BÀI TOÁN VÀ PHƯƠNG PHÁP TÌM KIẾM LỜI GIẢI



TRÍ TUỆ NHÂN TẠO

Artificial Intelligence

Đoàn Vũ Thịnh
Khoa Công nghệ Thông tin
Đại học Nha Trang
Email: thinhdv@ntu.edu.vn

Nha Trang, 06-2023

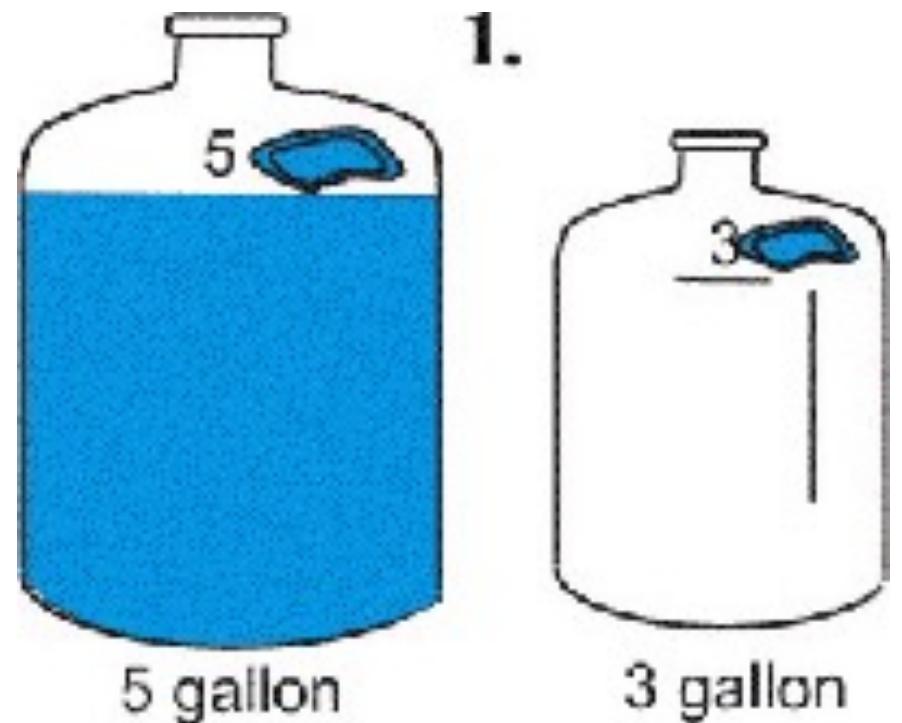
Chương 2. BÀI TOÁN VÀ PHƯƠNG PHÁP TÌM KIẾM LỜI GIẢI

Để thiết kế giải thuật chung giải các bài toán này, chúng ta nên phát biểu bài toán theo dạng 5 thành phần:

- Trạng thái bài toán,
- Trạng thái đầu,
- Trạng thái đích,
- Các phép chuyển trạng thái,
- Lược đồ chi phí các phép chuyển trạng thái (chi phí)

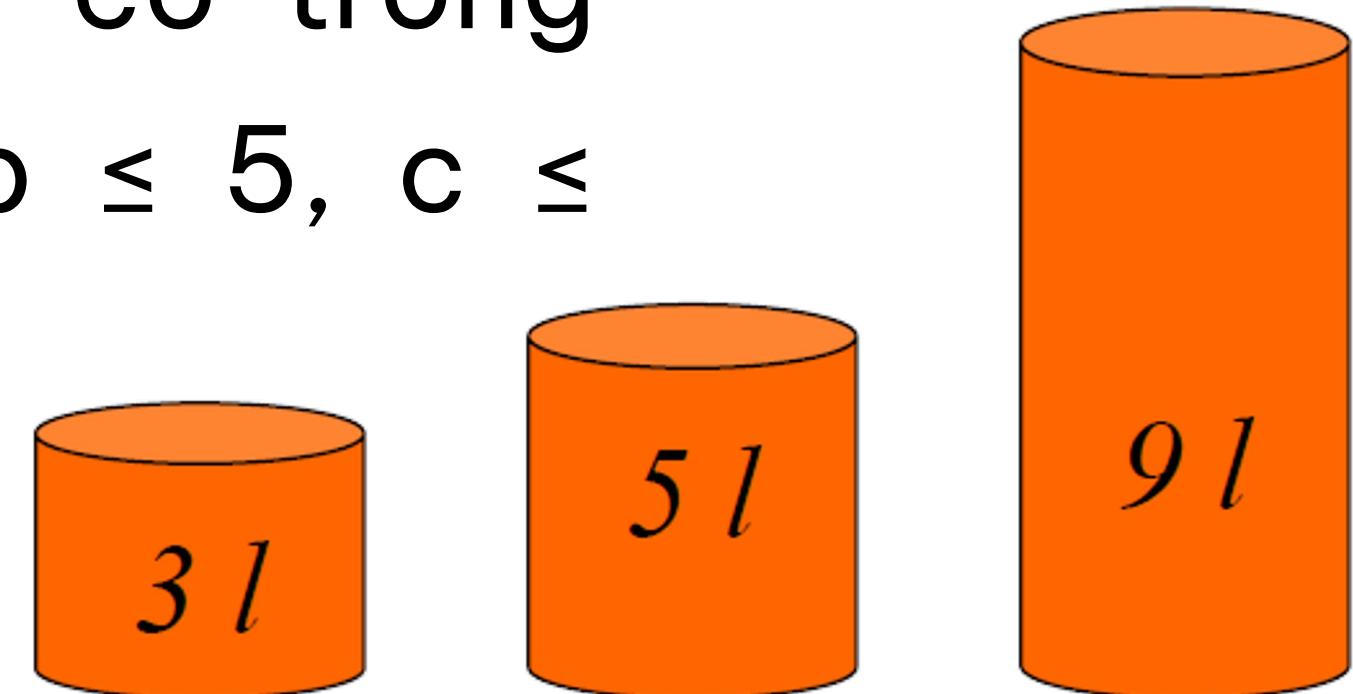
Chương 2. BÀI TOÁN VÀ PHƯƠNG PHÁP TÌM KIẾM LỜI GIẢI

- Bài toán đổ dầu có nguồn gốc từ bài toán “Die Hard 3 Puzzle”, xuất hiện trong bộ phim “Die Hard with a Vengeance” năm 1995. Trong bài toán này, nhân vật chính phải sử dụng hai thùng có dung tích 3 gallon và 5 gallon để đổ được chính xác 4 gallon dầu. Bài toán này đã tạo ra sự quan tâm và thách thức về khả năng tư duy tìm kiếm và giải quyết vấn đề.



Chương 2. BÀI TOÁN VÀ PHƯƠNG PHÁP TÌM KIẾM LỜI GIẢI

- Trạng thái bài toán: Gọi số nước có trong 3 can lần lượt là a, b, c ($a \leq 3$, $b \leq 5$, $c \leq 9$ là trạng thái của bài toán)
- Trạng thái đầu: $(0, 0, 0)$
- Trạng thái đích: $(0, 0, 7)$
- Các phép chuyển trạng thái: từ trạng thái (a,b,c) có thể chuyển sang trạng thái (x,y,z) thông qua các thao tác như làm rỗng 1 can, chuyển từ can này sang can kia đến khi hết nước ở can nguồn hoặc can đích bị đầy.
- Lược đồ chi phí các phép chuyển trạng thái (chi phí): 1

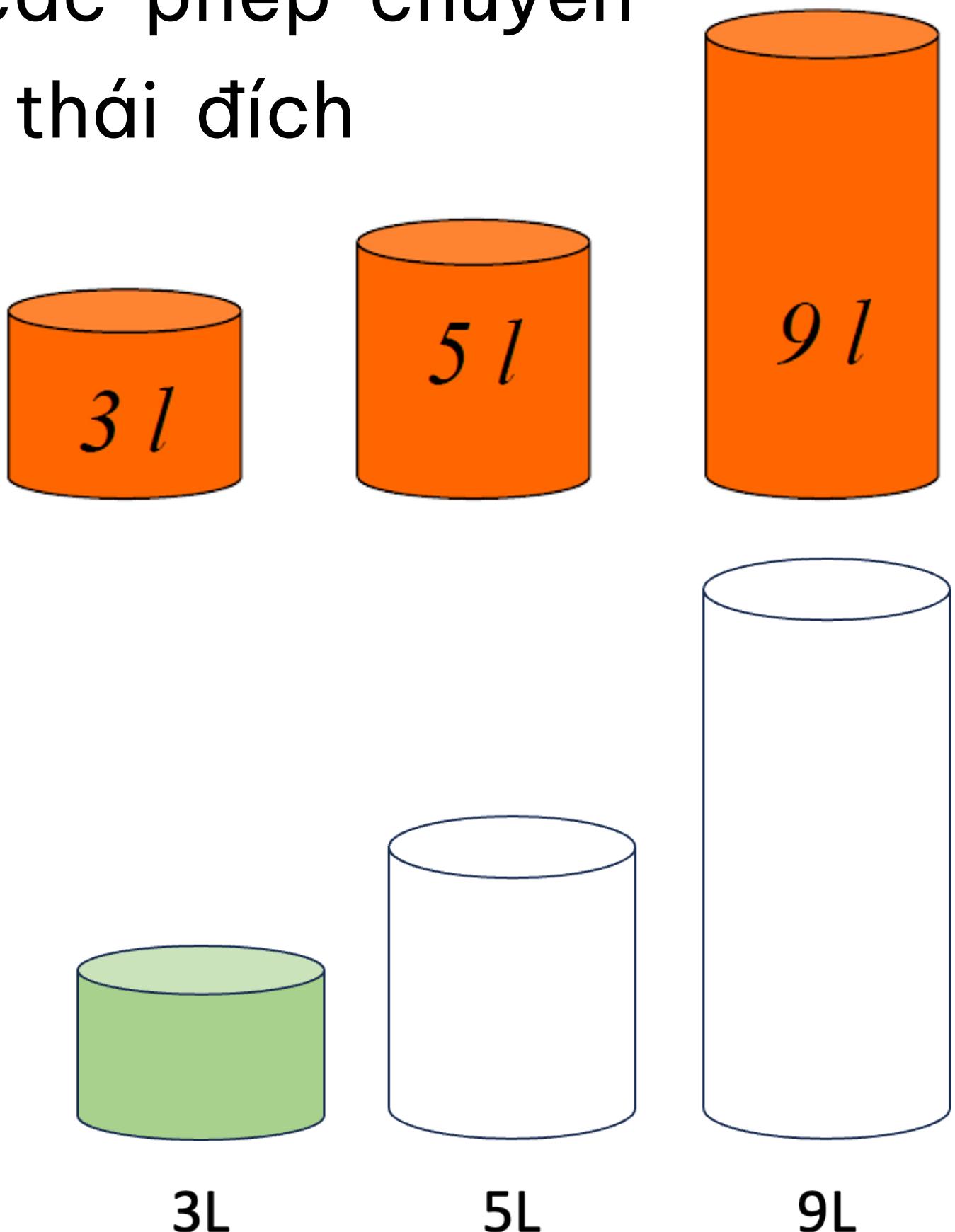


Chương 2. BÀI TOÁN VÀ PHƯƠNG PHÁP TÌM KIẾM LỜI GIẢI

- Một lời giải của bài toán là một dãy các phép chuyển trạng thái từ trạng thái đầu đến trạng thái đích

Bước	a	b	c
0	0	0	0
1	3	0	0
2	0	0	3
3	3	0	3
4	0	0	6
5	3	0	6
6	0	3	6
7	3	3	6
8	1	5	6
9	0	5	7

Chi phí = 9

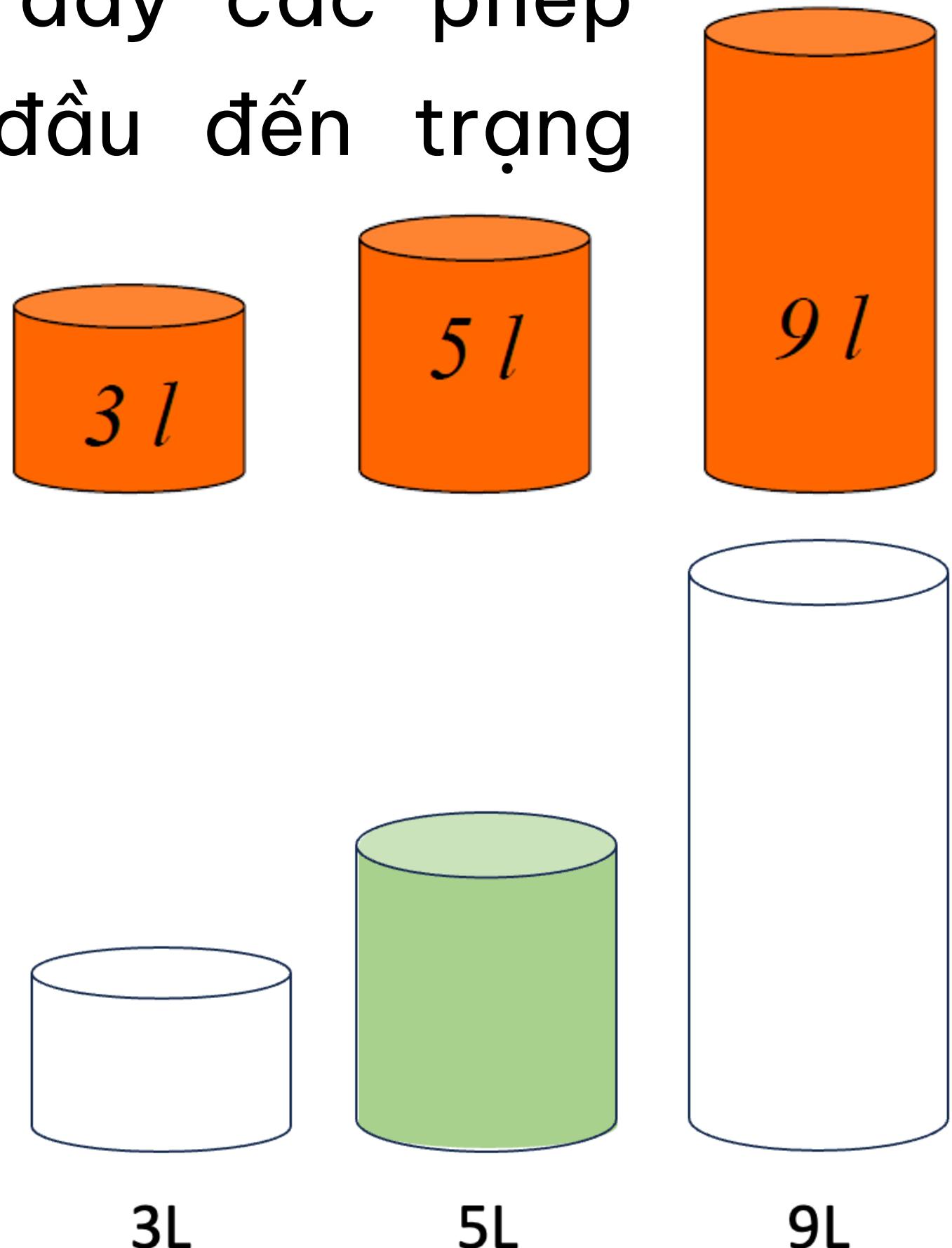


Chương 2. BÀI TOÁN VÀ PHƯƠNG PHÁP TÌM KIẾM LỜI GIẢI

- Một lời giải của bài toán là một dãy các phép chuyển trạng thái từ trạng thái đầu đến trạng thái đích

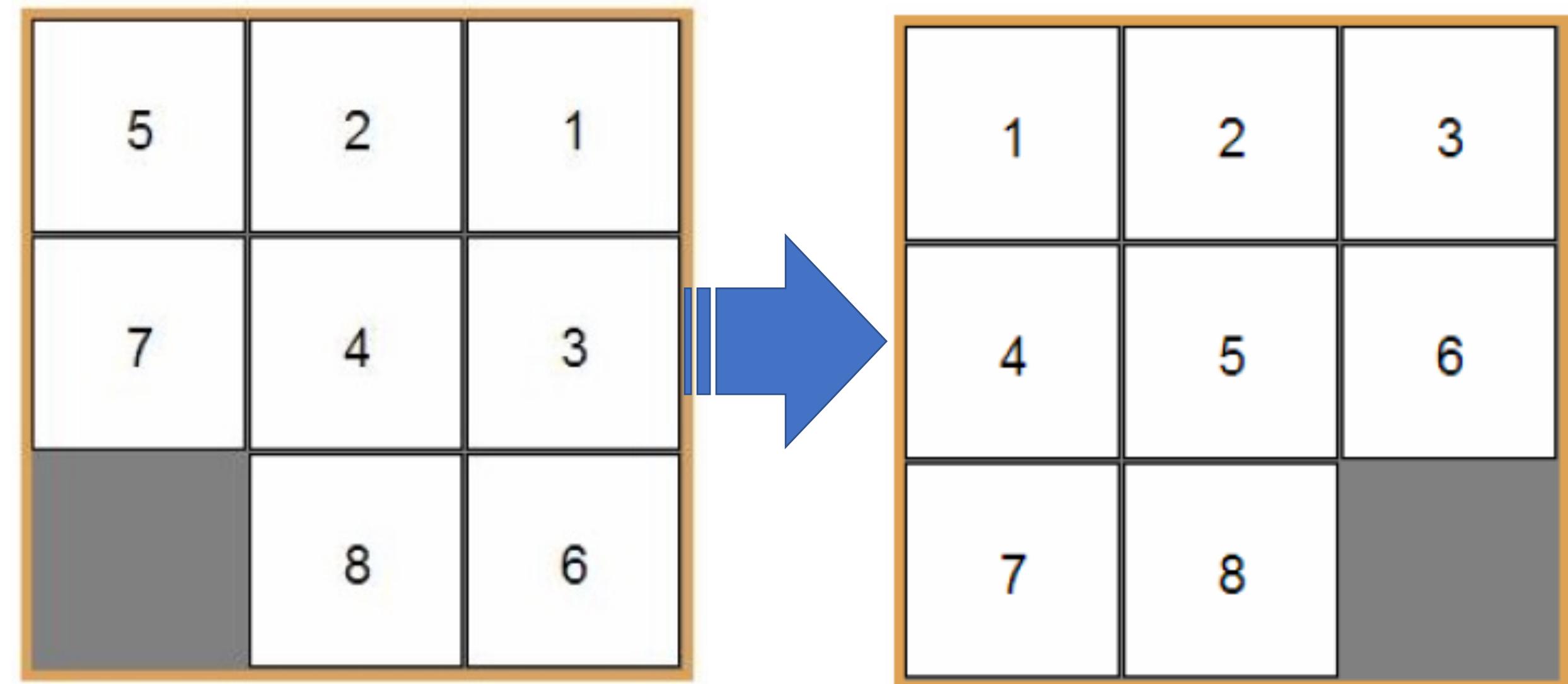
Bước	a	b	c
0	0	0	0
1	0	5	0
2	3	2	0
3	3	0	2
4	3	5	2
5	3	0	7

Chi phí = 5



Chương 2. BÀI TOÁN VÀ PHƯƠNG PHÁP TÌM KIẾM LỜI GIẢI

- Bàn cờ kích thước 3×3 , trên bàn cờ có 8 quân cờ đánh số từ 1 đến 8 và có một ô trống. Có thể chuyển một quân cờ có chung cạnh với ô trống sang ô trống. Tìm dãy các phép chuyển để từ trạng thái ban đầu về trạng thái đích.



Chương 2. BÀI TOÁN VÀ PHƯƠNG PHÁP TÌM KIẾM LỜI GIẢI

- Bài toán di chuyển 8 số trên bàn cờ có thể phát biểu dưới dạng 5 thành phần:
 - Biểu diễn trạng thái: mảng 2 chiều kích thước 3×3 , phần tử của mảng lưu số hiệu quân cờ (từ 0 đến 9, 0 là vị trí trống).
 - Trạng thái đầu (hình bên trái)
 - Trạng thái đích (hình bên phải)
 - Phép chuyển trạng thái: đổi chỗ ô có số hiệu 0 với một trong các ô có cùng cạnh.
 - Chi phí: mỗi phép chuyển có chi phí 1.

The figure shows two 3x3 grids with a light gray background and a dark gray border. The left grid represents the initial state (start) and the right grid represents the goal state (target). Both grids have 9 cells, each containing a number from 0 to 8 or a blank space (0). In the start state, the numbers are arranged as follows:

5	2	1
7	4	3
0	8	6

In the target state, the numbers are arranged as follows:

1	2	3
4	5	6
7	8	0

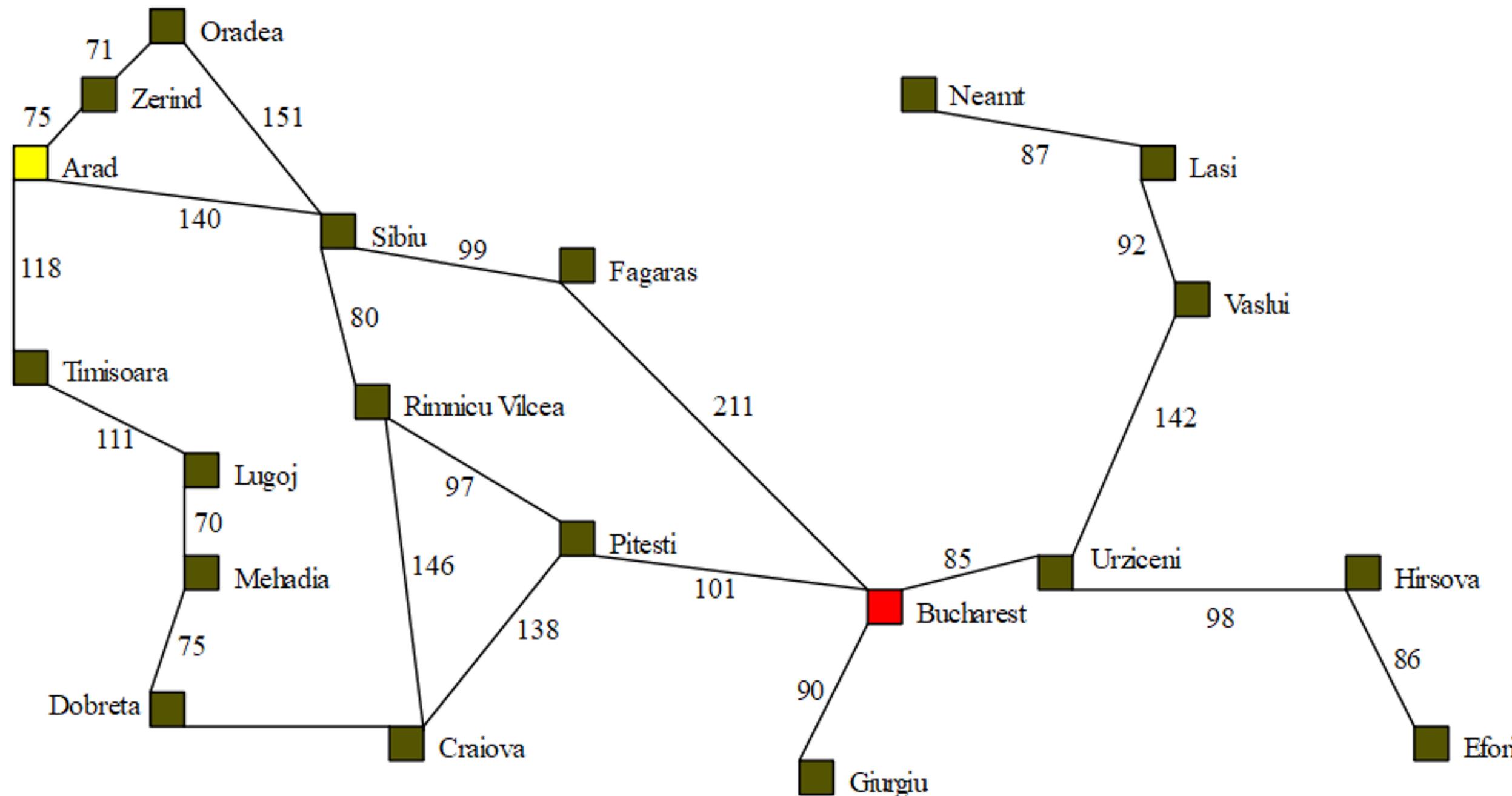
Chương 2. BÀI TOÁN VÀ PHƯƠNG PHÁP TÌM KIẾM LỜI GIẢI

5	2	1	5	2	1	0	2	1	2	0	1	2	4	1	2	4	1
7	4	3	0	4	3	5	4	3	5	4	3	5	0	3	0	5	3
0	8	6	7	8	6	7	8	6	7	8	6	7	8	6	7	8	6

4	1	3	4	1	3	4	1	3	4	1	0	4	0	1	0	4	1
0	2	5	2	0	5	2	5	0	2	5	3	2	5	3	2	5	3
7	8	6	7	8	6	7	8	6	7	8	6	7	8	6	7	8	6

0	1	3	1	0	3	1	2	3	1	2	3	1	2	3	1	2	3
4	2	5	4	2	5	4	0	5	4	5	0	4	5	6	7	8	0
7	8	6	7	8	6	7	8	6	7	8	6	7	8	0	Chi phí: 16		

Chương 2. BÀI TOÁN VÀ PHƯƠNG PHÁP TÌM KIẾM LỜI GIẢI

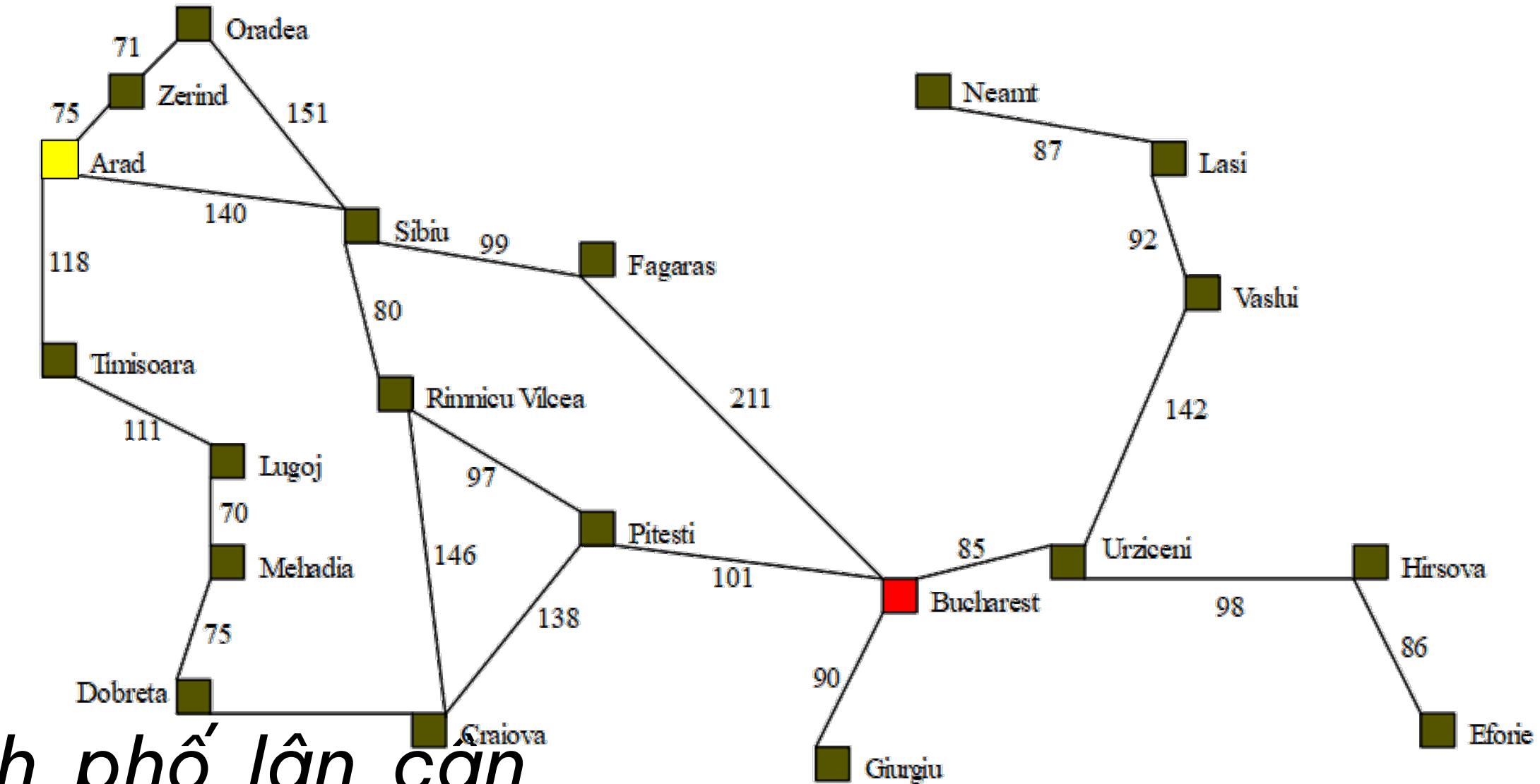


"Artificial Intelligence: A Modern Approach" -
Stuart Russell và Peter Norvig

Một ôtô robot tìm đường từ Arad đến Bucharest. Robot này không có bản đồ đầy đủ, nhưng khi nó đến một thành phố mới, nó có bộ cảm biến đọc được biển chỉ đường đến các thành lân cận.

Chương 2. BÀI TOÁN VÀ PHƯƠNG PHÁP TÌM KIẾM LỜI GIẢI

- Bài toán có thể phát biểu theo 5 thành phần:
- Trạng thái: vị trí của ôtô
- Trạng thái đầu:
 - *Thành phố Arad*
- Trạng thái đích:
 - *Thành phố Bucharest*
- Phép chuyển trạng thái:
 - *từ thành phố sang thành phố lân cận*
- Chi phí:
 - *khoảng cách giữa 2 thành phố*
- Một ví dụ của lời giải: **Arad \Rightarrow Sibiu \Rightarrow Fagaras \Rightarrow Bucharest.**



Chương 2. BÀI TOÁN VÀ PHƯƠNG PHÁP TÌM KIẾM LỜI GIẢI

▪ Giải thuật tổng quát tìm kiếm lời giải

▪ Không gian trạng thái của bài toán

- Mỗi bài toán với 5 thành phần như mô tả ở trên, chúng ta có thể xây dựng được một cấu trúc đồ thị với **các nút là các trạng thái của bài toán**, các **cung là phép chuyển trạng thái**. Không gian trạng thái có thể là vô hạn hoặc hữu hạn. Ví dụ, với bài toán di chuyển 8 số trên bàn cờ, không gian trạng thái có số lượng là 8!
- Lời giải của bài toán là một đường đi trong không gian trạng thái có **điểm đầu là trạng thái đầu và điểm cuối là trạng thái đích**. Nếu không gian trạng thái của bài toán là nhỏ, có thể liệt kê và lưu vừa trong bộ nhớ của máy tính thì việc tìm đường đi trong không gian trạng thái có thể áp dụng các thuật toán tìm đường đi trong lý thuyết đồ thi.

Chương 2. BÀI TOÁN VÀ PHƯƠNG PHÁP TÌM KIẾM LỜI GIẢI

▪ Giải thuật tổng quát tìm kiếm lời giải

Function General_Search(*problem, strategy*) returns a *solution*, or failure

```
cây-tìm-kiếm ← trạng-thái-đầu;  
while (1)  
{  
    if (cây-tìm-kiếm không thể mở rộng được nữa) then return failure  
    nút-lá ← Chọn-1-nút-lá(cây-tìm-kiếm, strategy)  
    if (node-lá là trạng-thái-đích) then return Đường-đi(trạng-thái-đầu, nút-lá)  
    else mở-rộng(cây-tìm-kiếm, các-trạng-thái-kề(nút-lá))  
}
```

- Tìm kiếm theo chiều rộng (nút lá nào xuất hiện trong cây sớm hơn thì được chọn trước để phát triển cây).
- Tìm kiếm theo chiều sâu (ngược lại)

Chương 2. BÀI TOÁN VÀ PHƯƠNG PHÁP TÌM KIẾM LỜI GIẢI

▪ Giải thuật tổng quát tìm kiếm lời giải

▪ Đánh giá giải thuật: b^d

- Tính đầy đủ: có tìm được lời giải của bài toán không?
- Độ phức tạp thời gian: thời gian thực hiện giải thuật?
- Độ phức tạp không gian: Kích cỡ của bộ nhớ cần cho giải thuật? cấu trúc dữ liệu lưu các trạng thái (nút lá) của cây tìm kiếm
- Tính tối ưu: Giải thuật có tìm ra lời giải có chi phí tối ưu (nhỏ nhất hoặc lớn nhất tùy theo ngữ cảnh của bài toán)?

b: số nhánh tối đa của một nút, hay là số phép chuyển trạng thái tối đa của một trạng thái tổng quát.

d: độ sâu của lời giải có chi phí nhỏ nhất

Chương 2. BÀI TOÁN VÀ PHƯƠNG PHÁP TÌM KIẾM LỜI GIẢI

▪ Tìm kiếm theo chiều rộng (Breath First Search)

Begin

```
Open={s};           //s: đỉnh xuất phát
Close=∅;           //Close: tập các đỉnh đã xét
While(Open != ∅){
    n = Retrieve(Open); //n: đỉnh đang xét
    If(n==g) Return TRUE; //g: đỉnh kết thúc
    Open = Open ∪ Γ(n); //Γ(n): các đỉnh có thể đi trực tiếp từ n
    Close = Close ∪ {n}
}
```

End;

Chương 2. BÀI TOÁN VÀ PHƯƠNG PHÁP TÌM KIẾM LỜI GIẢI

- Tìm kiếm theo chiều rộng (Breath First Search)

Begin

Open={s};

Close=∅;

While(Open != ∅){

n = Retrieve(Open);

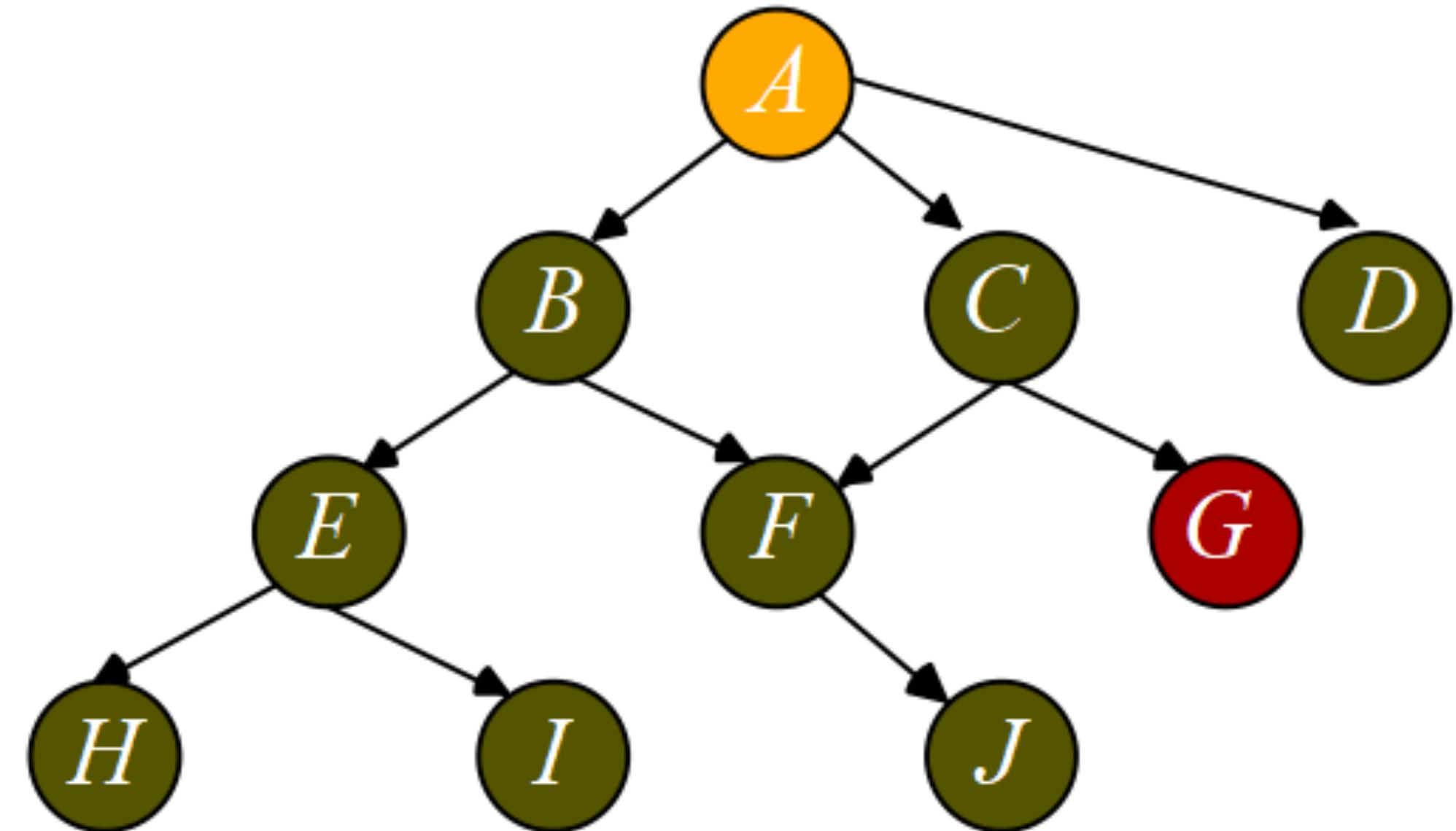
If(n==g) Return TRUE;

Open = Open ∪ Γ(n);

Close = Close ∪ {n}

}

End;



Ví dụ 1

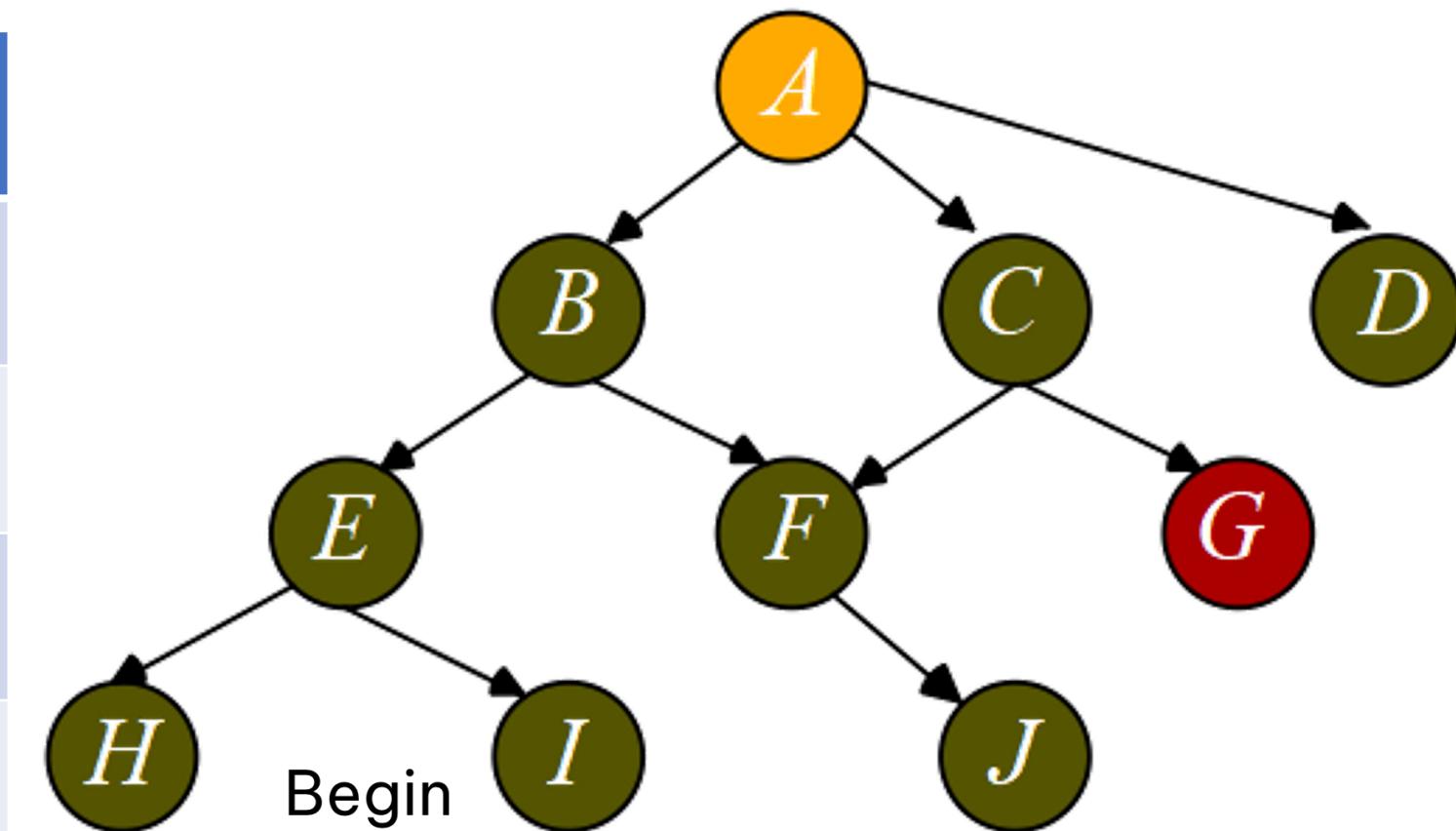
s = A là đỉnh bắt đầu g=G là đỉnh kết thúc

Chương 2. BÀI TOÁN VÀ PHƯƠNG PHÁP TÌM KIẾM LỜI GIẢI

▪ Tìm kiếm theo chiều rộng (Breath First Search)

Bước	n	$\Gamma(n)$	Open	Close
0			{A}	\emptyset
1	A	{B,C,D}	{B,C,D}	{A}
2	B	{E,F}	{C,D,E,F}	{A,B}
3	C	{F,G}	{D,E,F,G}	{A,B,C}
4	D	\emptyset	{E,F,G}	{A,B,C,D}
5	E	{H,I}	{F,G,H,I}	{A,B,C,D,E}
6	F	{J}	{G,H,I,J}	{A,B,C,D,E,F}
7	G	TRUE		

▪ A \Rightarrow C \Rightarrow G



```

Begin
  Open={s};
  Close= $\emptyset$ ;
  While(Open !=  $\emptyset$ ){
    n = Retrieve(Open);
    If(n==g) Return TRUE;
    Open = Open  $\cup$   $\Gamma(n)$ ;
    Close = Close  $\cup$  {n}
  }
End;
  
```

Chương 2. BÀI TOÁN VÀ PHƯƠNG PHÁP TÌM KIẾM LỜI GIẢI

- Tìm kiếm theo chiều rộng (Breath First Search)

Begin

Open={s};

Close=∅;

While(Open != ∅){

n = Retrieve(Open);

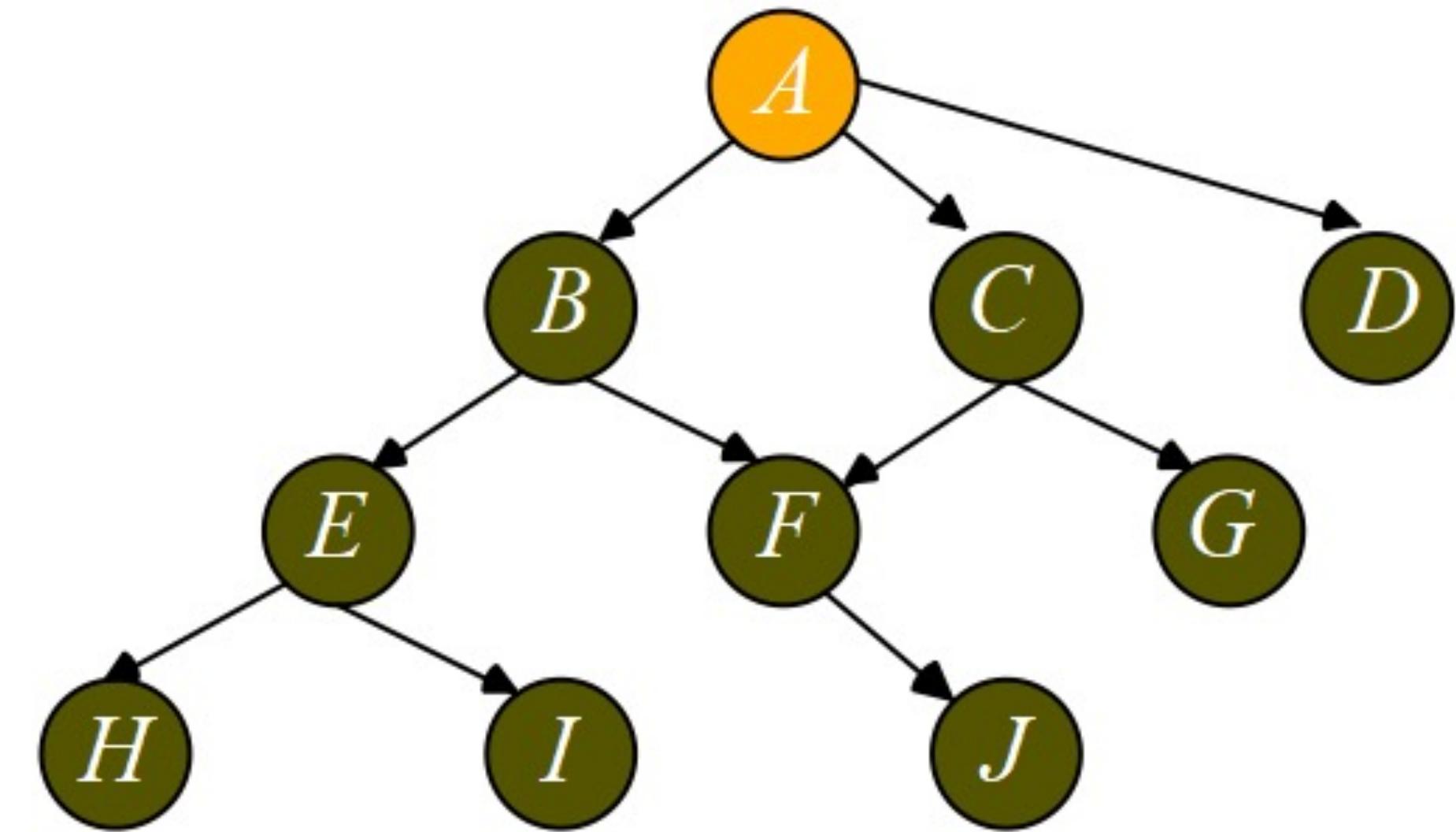
If(n==g) Return TRUE;

Open = Open ∪ Γ(n);

Close = Close ∪ {n}

}

End;



Ví dụ 2

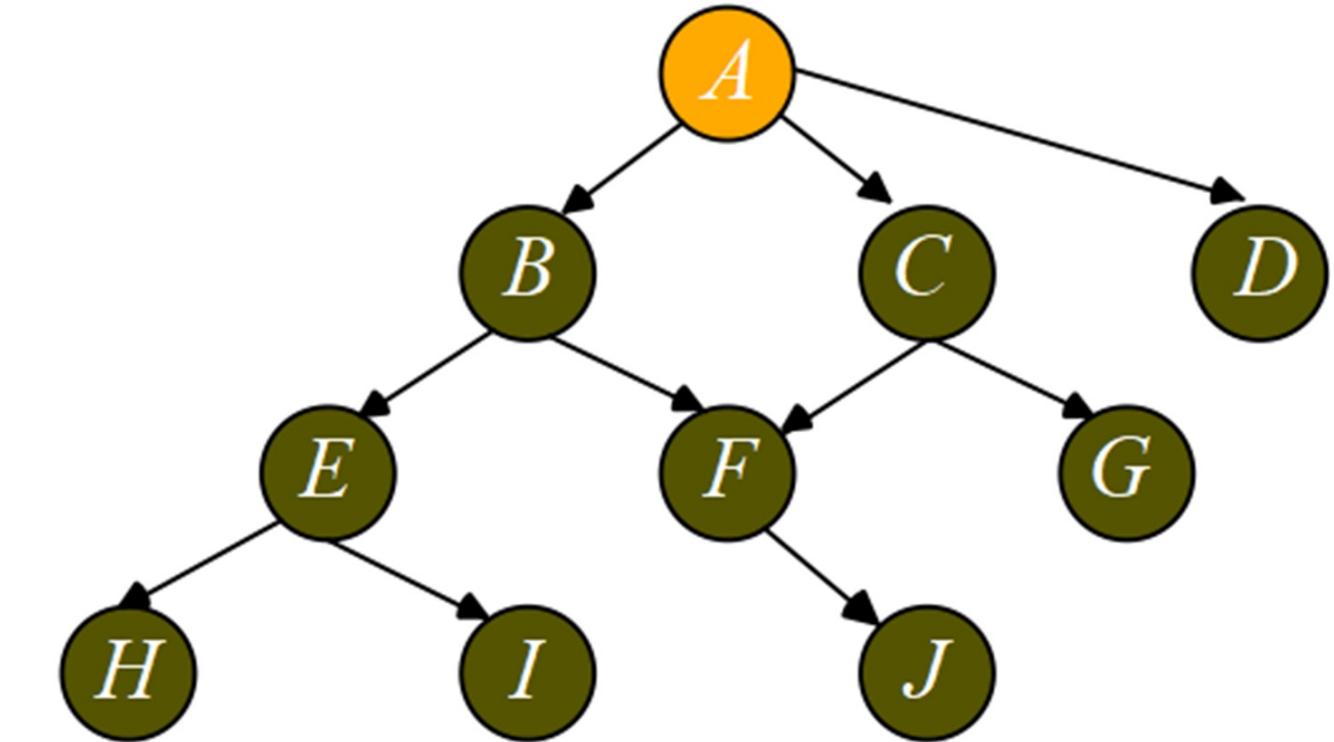
s = A là đỉnh bắt đầu g=U là đỉnh kết thúc

Chương 2. BÀI TOÁN VÀ PHƯƠNG PHÁP TÌM KIẾM LỜI GIẢI

- Tìm kiếm theo chiều rộng (Breath First Search)

Bước	n	$\Gamma(n)$	Open	Close
0			{A}	\emptyset
1	A	{B,C,D}	{B,C,D}	{A}
2	B	{E,F}	{C,D,E,F}	{A,B}
3	C	{F,G}	{D,E,F,G}	{A,B,C}
4	D	\emptyset	{E,F,G}	{A,B,C,D}
5	E	{H,I}	{F,G,H,I}	{A,B,C,D,E}

s = A là đỉnh bắt đầu g=U là đỉnh kết thúc



Begin

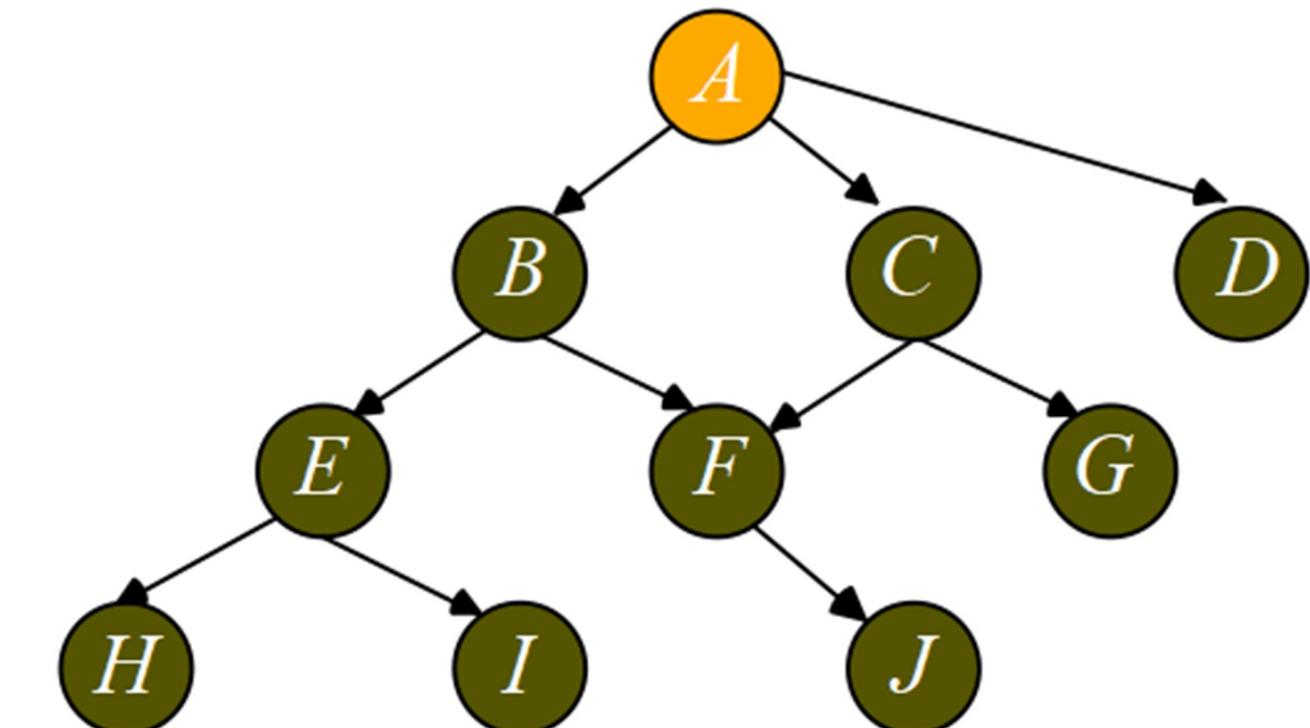
```

Open={s};
Close= $\emptyset$ ;
While(Open !=  $\emptyset$ ){
    n = Retrieve(Open);
    If(n==g) Return TRUE;
    Open = Open  $\cup$   $\Gamma(n)$ ;
    Close = Close  $\cup$  {n}
}
End;
  
```

Chương 2. BÀI TOÁN VÀ PHƯƠNG PHÁP TÌM KIẾM LỜI GIẢI

▪ Tìm kiếm theo chiều rộng (Breath First Search)

Bước	n	$\Gamma(n)$	Open	Close
6	F	{J}	{G,H,I,J}	{A,B,C,D,E,F}
7	G	\emptyset	{H,I,J}	{A,B,C,D,E,F,G}
8	H	\emptyset	{I,J}	{A,B,C,D,E,F,G,H}
9	I	\emptyset	{J}	{A,B,C,D,E,F,G,H,I}
10	J	\emptyset	\emptyset	{A,B,C,D,E,F,G,H,I,J}
11		FALSE		



Begin

```

Open={s};
Close= $\emptyset$ ;
While(Open !=  $\emptyset$ ){
    n = Retrieve(Open);
    If(n==g) Return TRUE;
    Open = Open  $\cup$   $\Gamma(n)$ ;
    Close = Close  $\cup$  {n}
}
End;
  
```

$s = A$ là đỉnh bắt đầu $g=U$ là đỉnh kết thúc

Chương 2. BÀI TOÁN VÀ PHƯƠNG PHÁP TÌM KIẾM LỜI GIẢI

- Tìm kiếm theo chiều rộng (Breath First Search)

Begin

Open={s};

Close=∅;

While(Open != ∅){

n = Retrieve(Open);

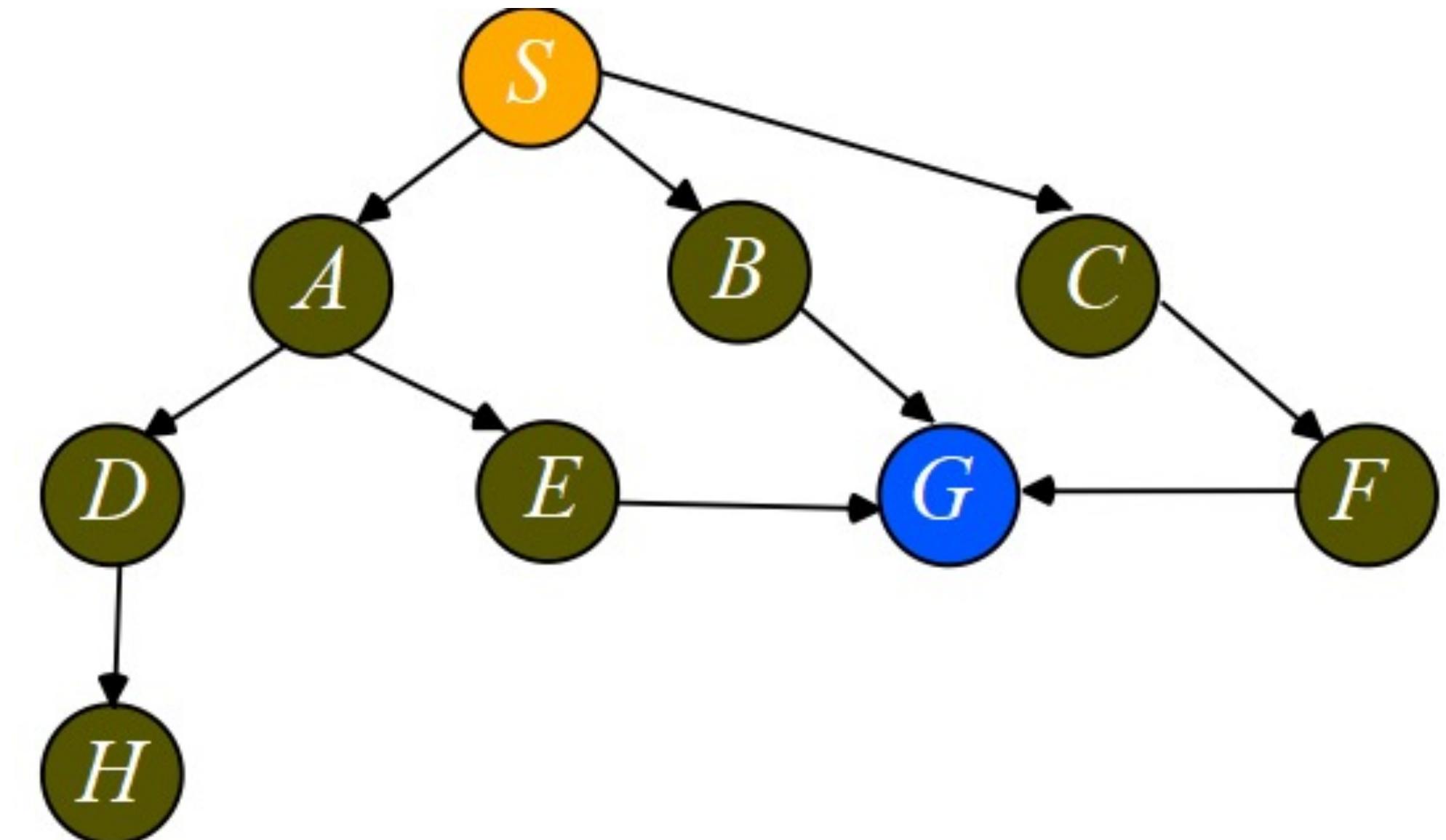
If(n==g) Return TRUE;

Open = Open ∪ Γ(n);

Close = Close ∪ {n}

}

End;



Ví dụ 3

S là đỉnh bắt đầu

G là đỉnh kết thúc

Chương 2. BÀI TOÁN VÀ PHƯƠNG PHÁP TÌM KIẾM LỜI GIẢI

▪ Tìm kiếm theo chiều rộng (Breath First Search)

Begin

Open={s};

Close=∅;

While(Open != ∅){

n = Retrieve(Open);

If(n==g) Return TRUE;

Open = Open \cup $\Gamma(n)$;

Close = Close \cup {n}

}

End;

Nguyên tắc của BFS:

- *Tìm 1 nút biên và các nút kề*
- *Lấy 1 nút đầu của OPEN ra khỏi Queue*
- *Đưa 1 nút vào cuối OPEN*
- *Không đưa nút đã duyệt/hoặc đã có vào Queue*
- *Khi thêm dựa theo thứ tự alphaB*

Chương 2. BÀI TOÁN VÀ PHƯƠNG PHÁP TÌM KIẾM LỜI GIẢI

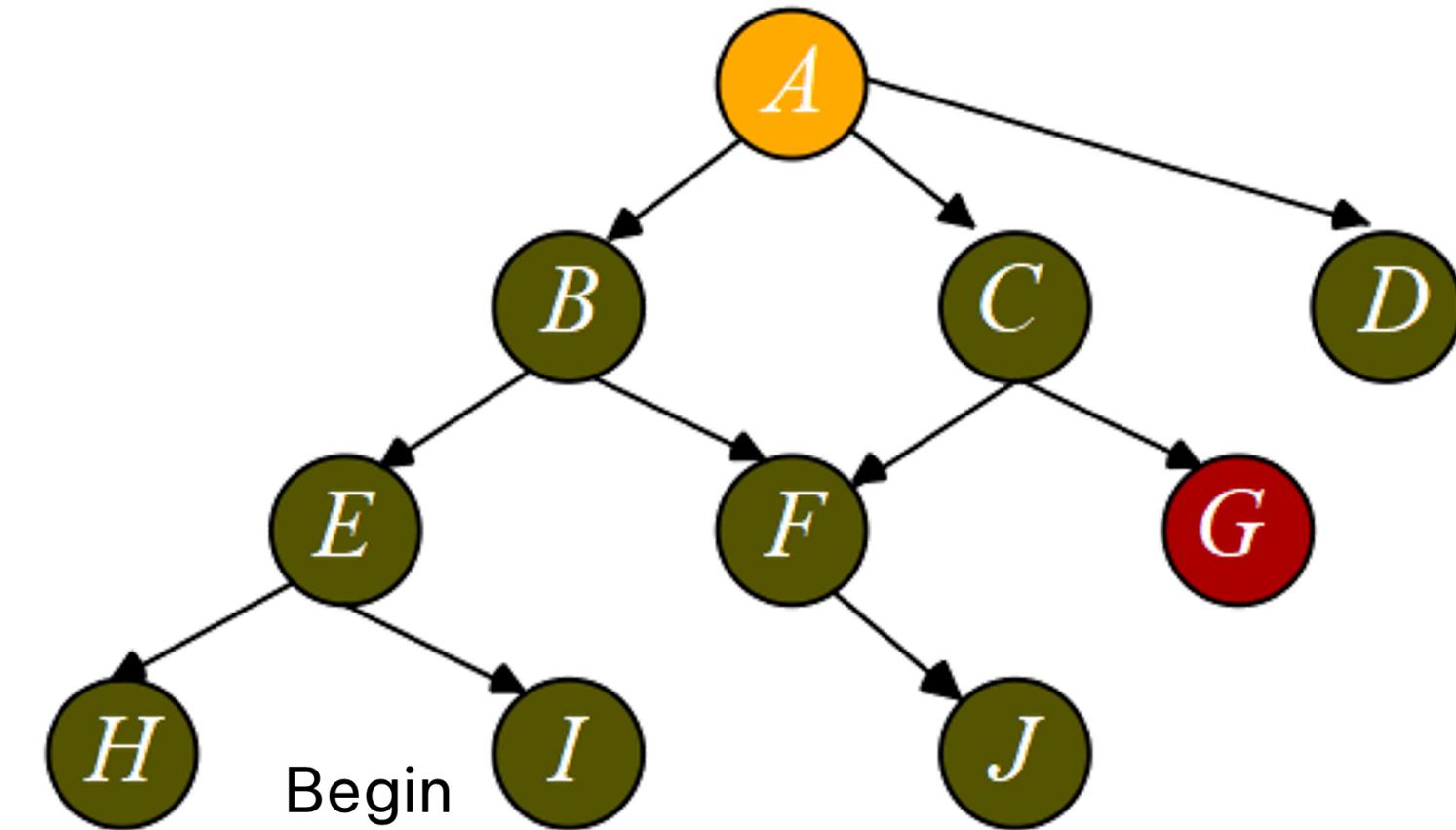
- Tìm kiếm theo chiều rộng (Breath First Search)

										10
0	1	1	1	0	0	0	0	0	0	
1	0	0	0	1	1	0	0	0	0	
1	0	0	0	0	1	1	0	0	0	
1	0	0	0	0	0	0	0	0	0	
0	1	0	0	0	0	0	1	1	0	
0	1	1	0	0	0	0	0	0	0	
0	0	1	0	0	0	0	0	0	0	
0	0	0	0	1	0	0	0	0	0	
0	0	0	0	0	1	0	0	0	0	
A	B	C	D	E	F	G	H	I	J	

Nhập đỉnh bắt đầu: 0

Nhập đỉnh kết thúc: 6

Đường đi từ 0 đến 6: 0 2 6



```

Open={s};
Close=∅;
While(Open != ∅){
    n = Retrieve(Open);
    If(n==g) Return TRUE;
    Open = Open ∪ Γ(n);
    Close = Close ∪ {n}
}
End;
  
```

Chương 2. BÀI TOÁN VÀ PHƯƠNG PHÁP TÌM KIẾM LỜI GIẢI

▪ Tìm kiếm theo chiều rộng (Breath First Search)

Begin

Open={s};

Close=∅;

While(Open != ∅){

n = Retrieve(Open);

If(n==g) Return TRUE;

Open = Open \cup $\Gamma(n)$;

Close = Close \cup {n}

}

End;

```
int dothi[100][100];//tap cac dinh cua do thi
int visited[100];//tap cac dinh da xet: Close
int queue[100];//hang doi chua cac dinh chua xet: Open

int front = 0, rear = 0;
queue[rear++] = start;
// Them dinh vao hang doi

visited[start] = 1;
// Danh dau dinh Start da duoc tham

int parent[n];
parent[start] = -1;
// Khoi tao dinh cha co nut goc la -1
```

Chương 2. BÀI TOÁN VÀ PHƯƠNG PHÁP TÌM KIẾM LỜI GIẢI

▪ Tìm kiếm theo chiều rộng (Breath First Search)

Begin

Open={s};

Close=∅;

While(Open != ∅){

n = Retrieve(Open);

If(n==g) Return TRUE;

Open = Open ∪ Γ(n);

Close = Close ∪ {n}

}

End;

```
while (front < rear) {
    int current = queue[front++];
    // Lay dinh dau tien trong hang doi
    if (current == end) break;
    // Neu tim thay dinh end thi ket thuc

    for (int i = 0; i < n; i++) {
        if (dothi[current][i] == 1 && visited[i] == 0) {
            //Cac dinh ke dinh dang xet va chua duoc duyet
            queue[rear++] = i;
            // Them dinh i vao hang doi
            visited[i] = 1;
            // Danh dau dinh i da duoc duyet
            parent[i] = current;
            // Luu tru dinh cha la dinh hien tai
        }
    }
}
```

Chương 2. BÀI TOÁN VÀ PHƯƠNG PHÁP TÌM KIẾM LỜI GIẢI

▪ Tìm kiếm theo chiều rộng (Breath First Search)

Begin

Open={s};

Close=∅;

While(Open != ∅){

n = Retrieve(Open);

If(n==g) Return TRUE;

Open = Open ∪ Γ(n);

Close = Close ∪ {n}

}

End;

```
if (visited[end] == 0) {
    printf("Khong tim thay duong di tu %d den %d\n",
           start, end);
    return;
}

// In ra duong di bang cach in nguoc DS tu start
int path[n], len = 0;

for (int i = end; i != -1; i = parent[i]) {
    path[len++] = i;
}

printf("Duong di tu %d den %d: ", start, end);
for (int i = len - 1; i >= 0; i--) {
    printf("%d ", path[i]);
}
```

Chương 2. BÀI TOÁN VÀ PHƯƠNG PHÁP TÌM KIẾM LỜI GIẢI

▪ Tìm kiếm theo chiều sâu (Depth First Search)

Begin

Open={s};

Close=∅;

While(Open != ∅){

n = Retrieve(Open);

If(n==g) Return TRUE;

Open = Open ∪ Γ(n);

Close = Close ∪ {n}

}

End;

Nguyên tắc của DFS:

- *Tìm 1 nút biên và các nút kề*
- *Lấy 1 nút đầu của OPEN ra khỏi Queue*
- *Chèn 1 nút vào đầu OPEN*
- *Không đưa nút đã duyệt/hoặc đã có vào Queue*
- *Khi thêm dựa theo thứ tự alphaB*

Chương 2. BÀI TOÁN VÀ PHƯƠNG PHÁP TÌM KIẾM LỜI GIẢI

- Tìm kiếm theo chiều sâu (Depth First Search)

Begin

Open={s};

Close=∅;

While(Open != ∅){

n = Retrieve(Open);

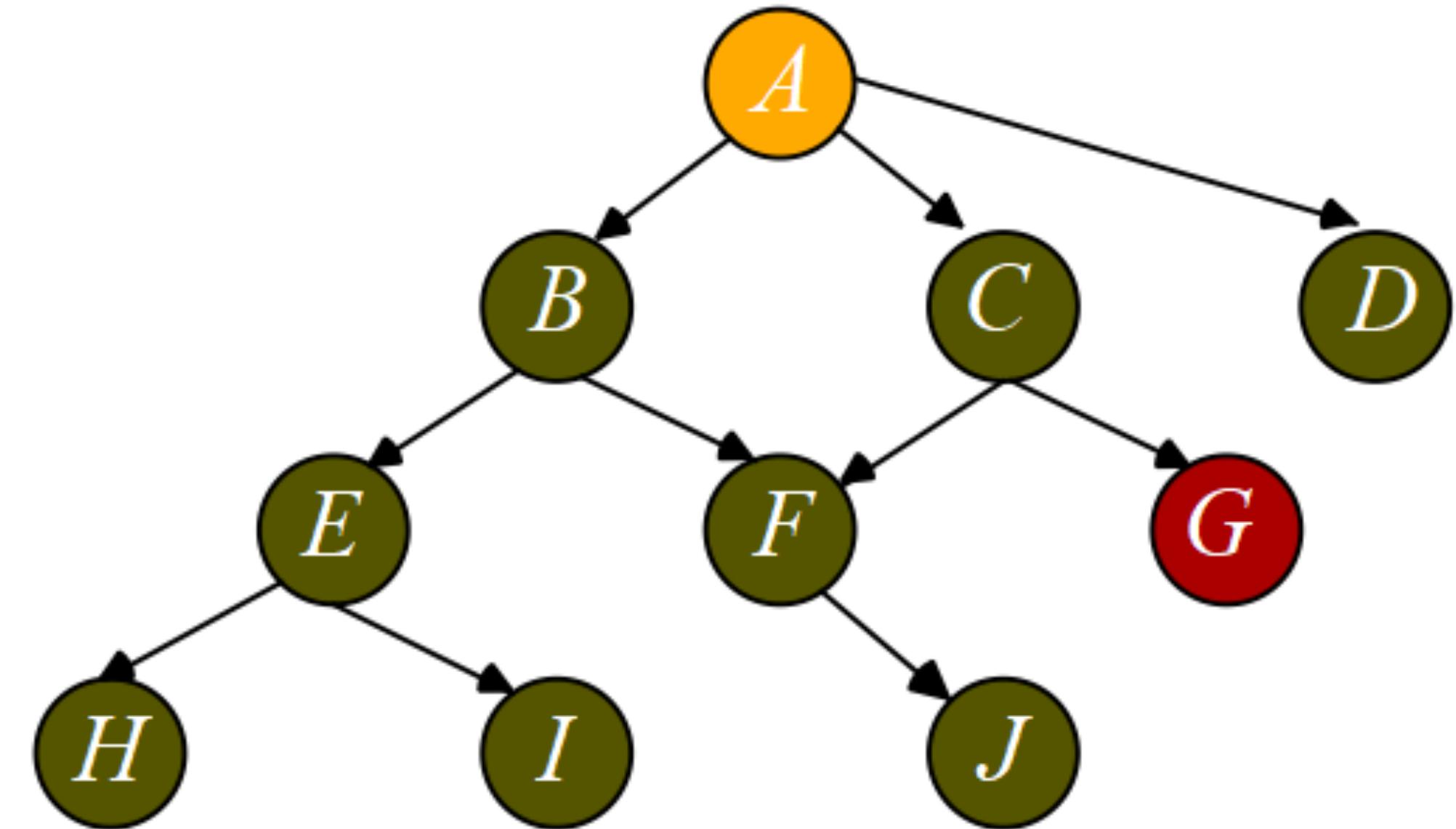
If(n==g) Return TRUE;

Open = Open ∪ Γ(n);

Close = Close ∪ {n}

}

End;



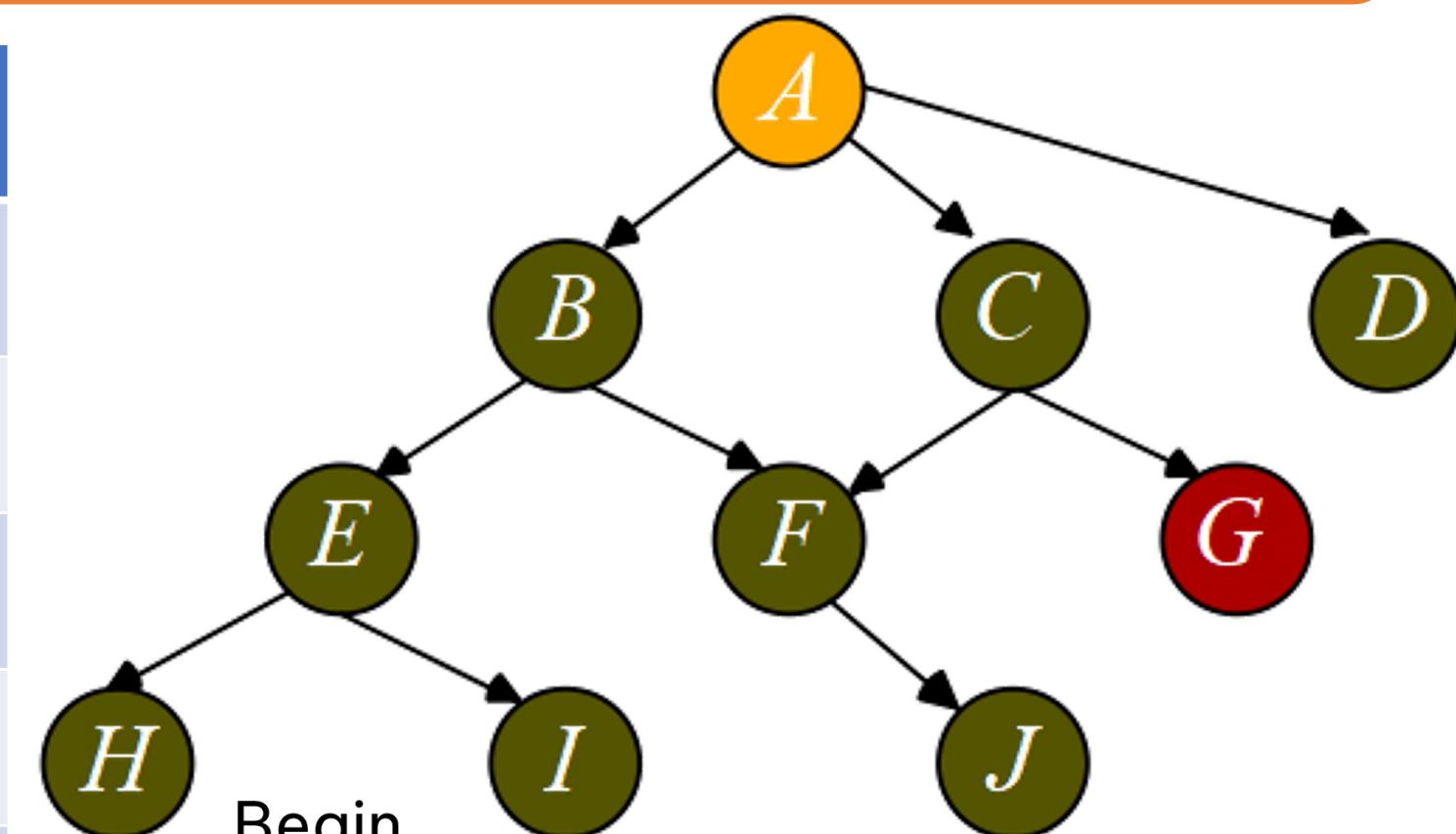
Ví dụ 1

s = A là đỉnh bắt đầu g=G là đỉnh kết thúc

Chương 2. BÀI TOÁN VÀ PHƯƠNG PHÁP TÌM KIẾM LỜI

▪ Tìm kiếm theo chiều sâu (Depth First Search)

Bước	n	$\Gamma(n)$	Open	Close
0			{A}	\emptyset
1	A	{B,C,D}	{B,C,D}	{A}
2	B	{E,F}	{E,F,C,D}	{A,B}
3	E	{H,I}	{H,I,F,C,D}	{A,B,E}
4	H	\emptyset	{I,F,C,D}	{A,B,E,H}
5	I	\emptyset	{F,C,D}	{A,B,E,H,I}
6	F	{J}	{J,C,D}	{A,B,E,H,I,F}
7	J	\emptyset	{C,D}	{A,B,E,H,I,F,J}
8	C	{F,G}	{G,D}	{A,B,E,H,I,F,J,C}
9	G	TRUE	■ A \Rightarrow C \Rightarrow G	



```

Begin
  Open={s};
  Close= $\emptyset$ ;
  While(Open !=  $\emptyset$ ){
    n = Retrieve(Open);
    If(n==g) Return TRUE;
    Open = Open  $\cup$   $\Gamma(n)$ ;
    Close = Close  $\cup$  {n}
  }
End;
  
```

Chương 2. BÀI TOÁN VÀ PHƯƠNG PHÁP TÌM KIẾM LỜI GIẢI

- Tìm kiếm theo chiều sâu (Depth First Search)

Begin

Open={s};

Close=∅;

While(Open != ∅){

n = Retrieve(Open);

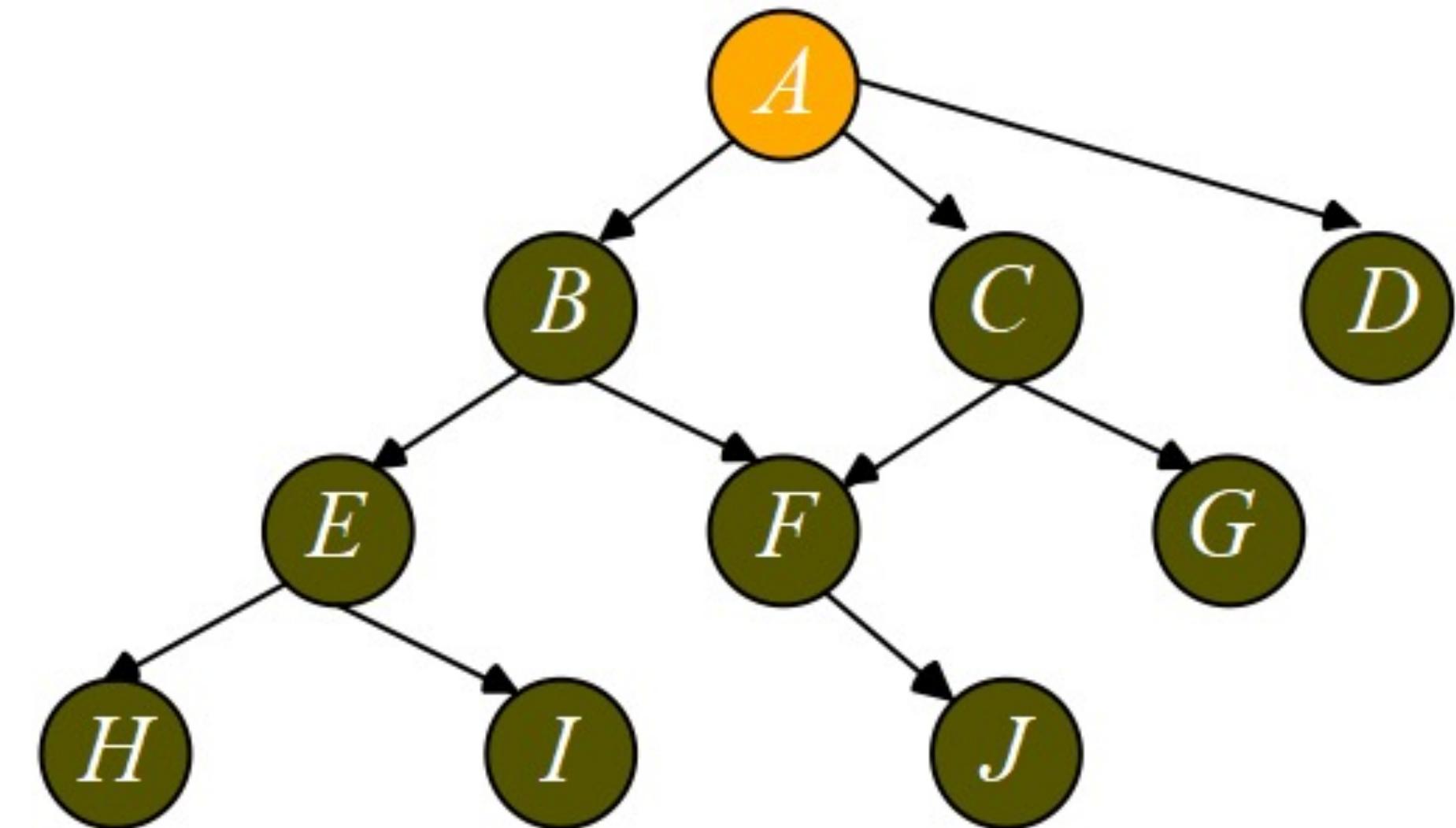
If(n==g) Return TRUE;

Open = Open ∪ Γ(n);

Close = Close ∪ {n}

}

End;



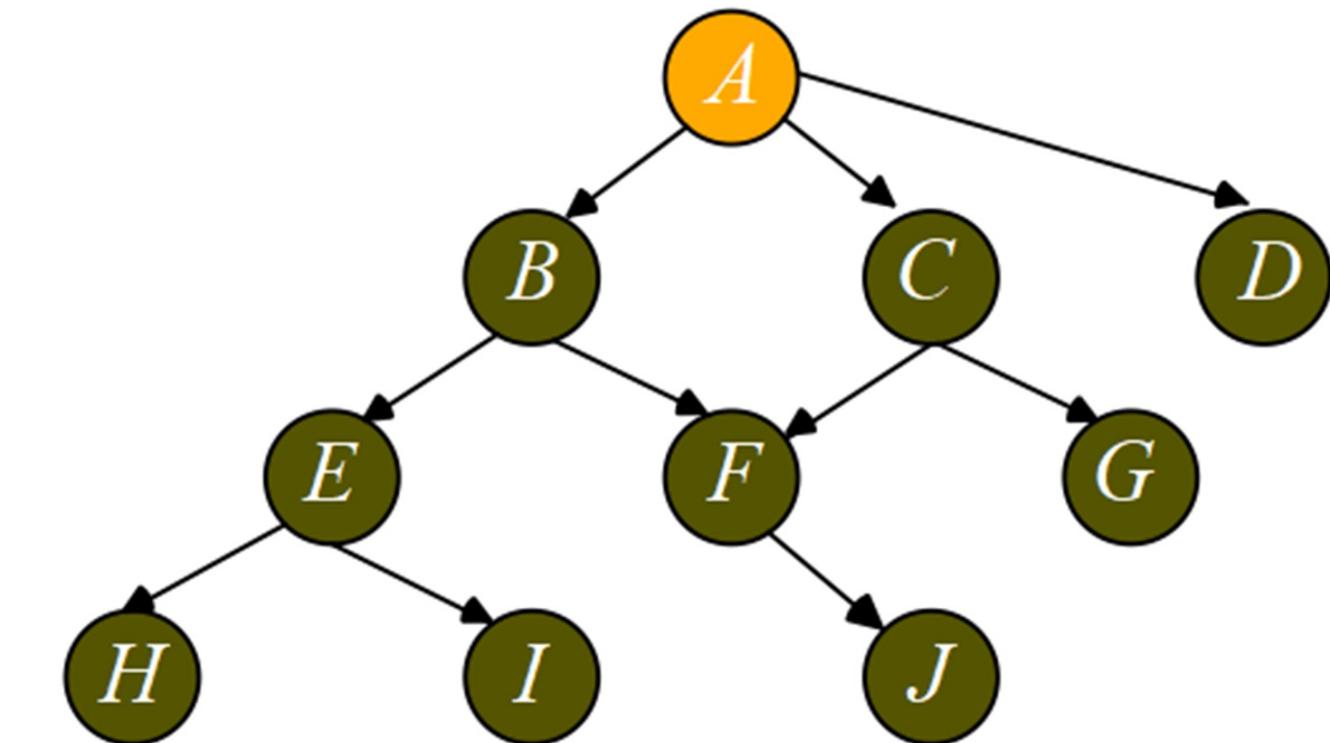
Ví dụ 2

s = A là đỉnh bắt đầu g=U là đỉnh kết thúc

Chương 2. BÀI TOÁN VÀ PHƯƠNG PHÁP TÌM KIẾM LỜI GIẢI

▪ Tìm kiếm theo chiều sâu (Depth First Search)

Bước	n	$\Gamma(n)$	Open	Close
0			{A}	\emptyset
1	A	{B,C,D}	{B,C,D}	{A}
2	B	{E,F}	{E,F,C,D}	{A,B}
3	E	{H,I}	{H,I,F,C,D}	{A,B,E}
4	H	\emptyset	{I,F,C,D}	{A,B,E,H}
5	I	\emptyset	{F,C,D}	{A,B,E,H,I}



```

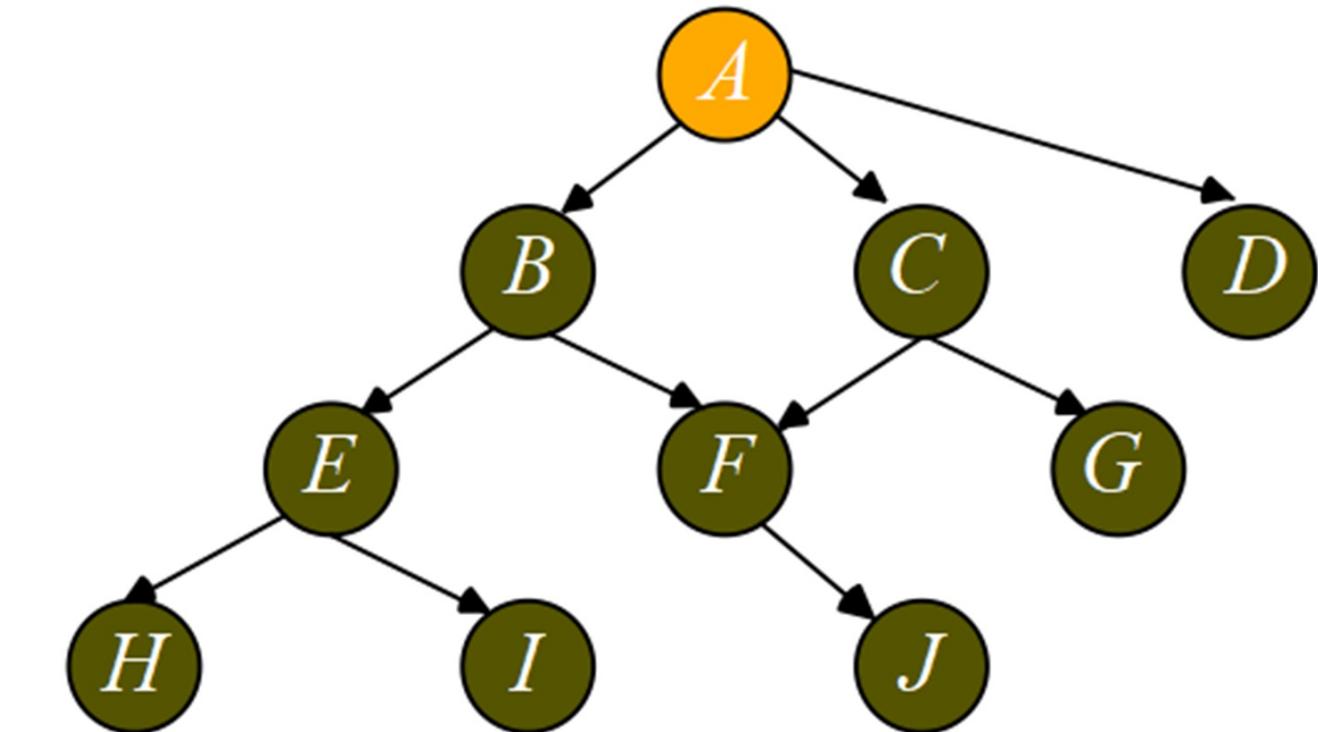
Begin
  Open={s};
  Close= $\emptyset$ ;
  While(Open !=  $\emptyset$ ){
    n = Retrieve(Open);
    If(n==g) Return TRUE;
    Open = Open  $\cup$   $\Gamma(n)$ ;
    Close = Close  $\cup$  {n}
  }
End;
  
```

s = A là đỉnh bắt đầu g=U là đỉnh kết thúc

Chương 2. BÀI TOÁN VÀ PHƯƠNG PHÁP TÌM KIẾM LỜI GIẢI

▪ Tìm kiếm theo chiều sâu (Depth First Search)

Bước	n	$\Gamma(n)$	Open	Close
6	F	{J}	{J,C,D}	{A,B,E,H,I,F}
7	J	\emptyset	{C,D}	{A,B,E,H,I,F,J}
8	C	{F,G}	{G,D}	{A,B,E,H,I,F,J,C}
9	G	\emptyset	{D}	{A,B,E,H,I,F,J,C,G}
10	D	\emptyset	\emptyset	{A,B,E,H,I,F,J,C,G,D}
11	\emptyset	FALSE		



```

Begin
  Open={s};
  Close= $\emptyset$ ;
  While(Open !=  $\emptyset$ ){
    n = Retrieve(Open);
    If(n==g) Return TRUE;
    Open = Open  $\cup$   $\Gamma(n)$ ;
    Close = Close  $\cup$  {n}
  }
End;
  
```

$s = A$ là đỉnh bắt đầu $g=U$ là đỉnh kết thúc

Chương 2. BÀI TOÁN VÀ PHƯƠNG PHÁP TÌM KIẾM LỜI GIẢI

- Tìm kiếm theo chiều sâu (Depth First Search)

Begin

Open={s};

Close=∅;

While(Open != ∅){

n = Retrieve(Open);

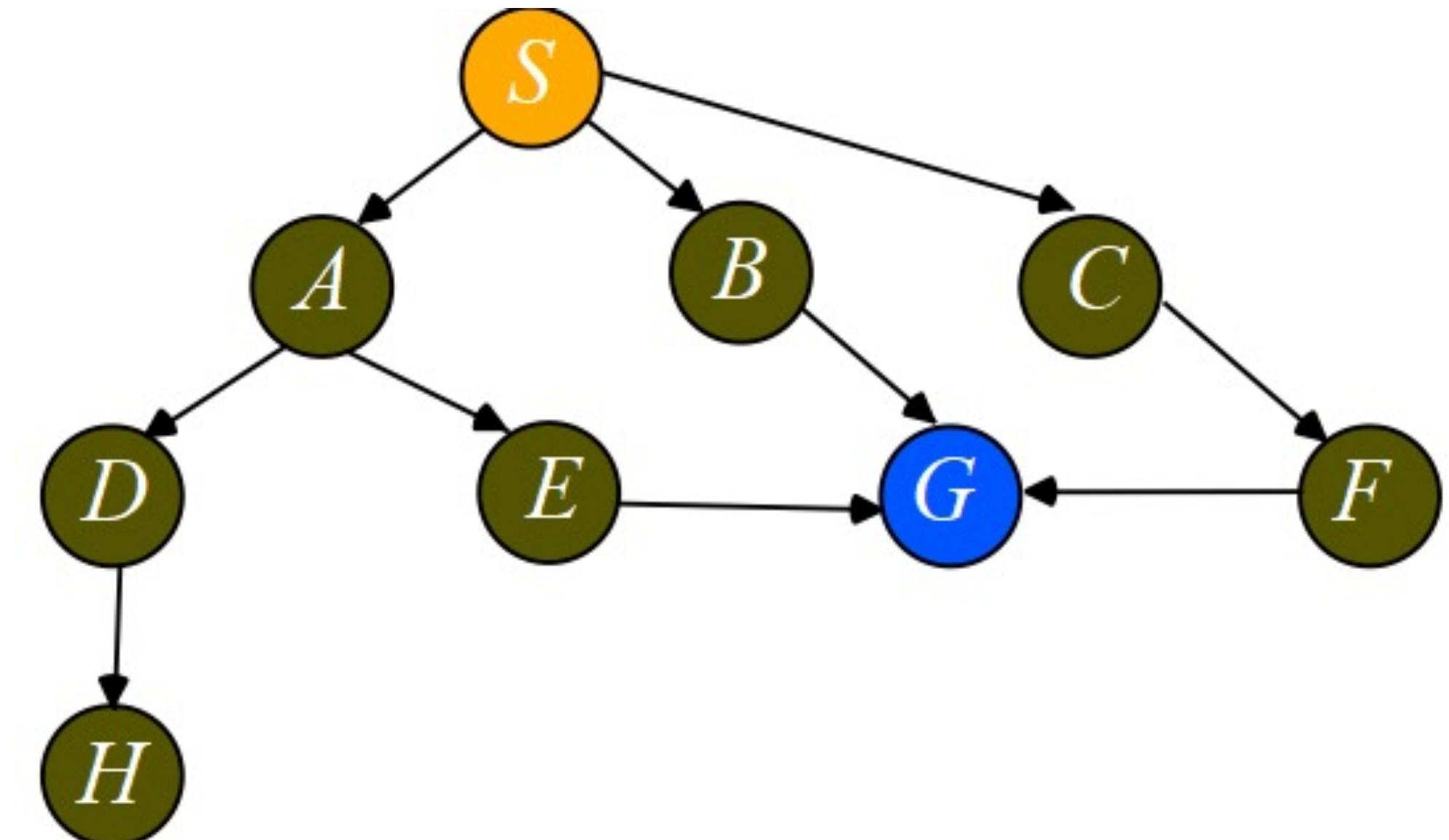
If(n==g) Return TRUE;

Open = Open ∪ Γ(n);

Close = Close ∪ {n}

}

End;



Ví dụ 3

S là đỉnh bắt đầu

G là đỉnh kết thúc

Chương 2. BÀI TOÁN VÀ PHƯƠNG PHÁP TÌM KIẾM LỜI GIẢI

■ BFS

Open = [A]; closed = []

Open = [B,C,D];

closed = [A]

Open = [C,D,E,F];
closed = [B,A]

Open = [D,E,F,G,H]; closed = [C,B,A]

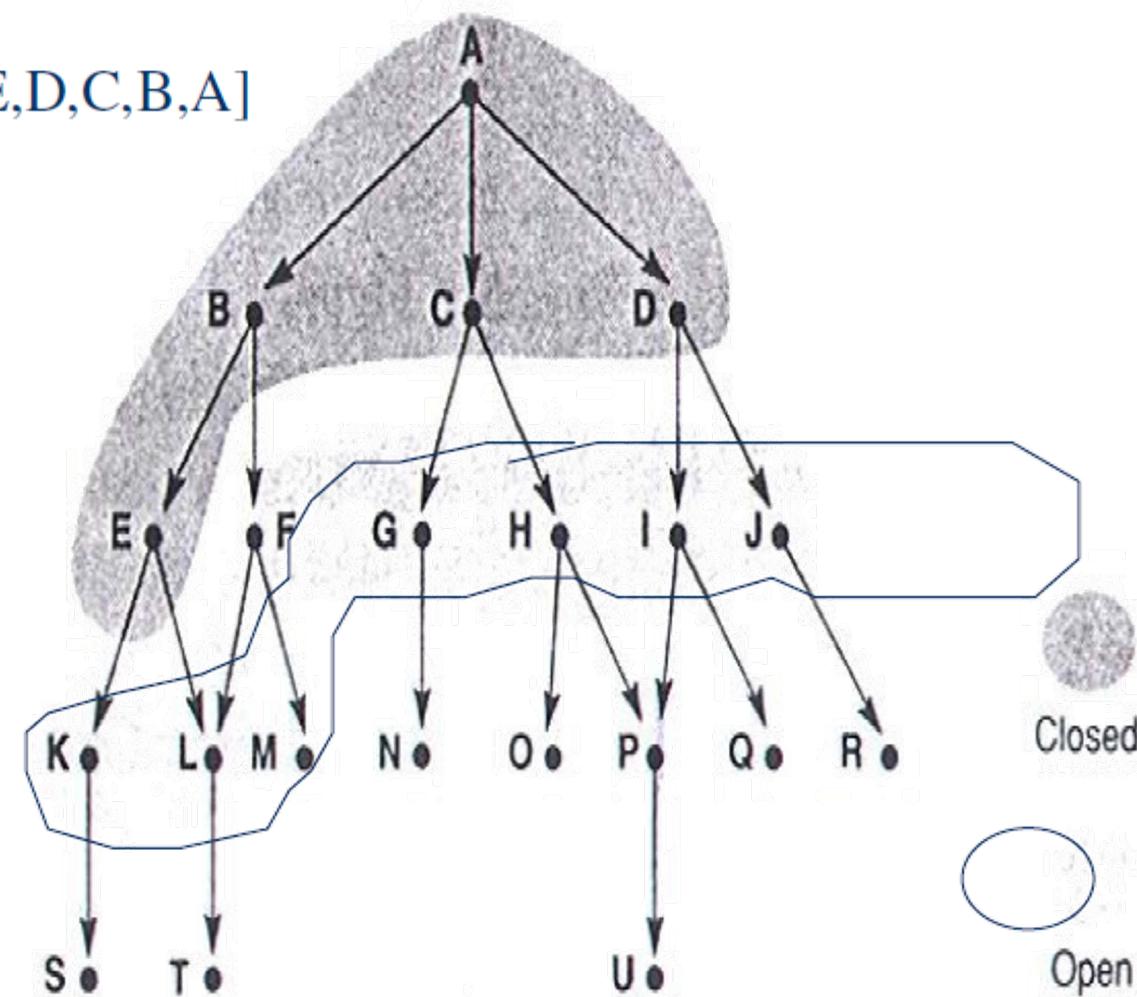
Open = [E,F,G,H,I,J]; closed = [D,C,B,A]

Open = [F,G,H,I,J,K,L];closed = [E,D,C,B,A]

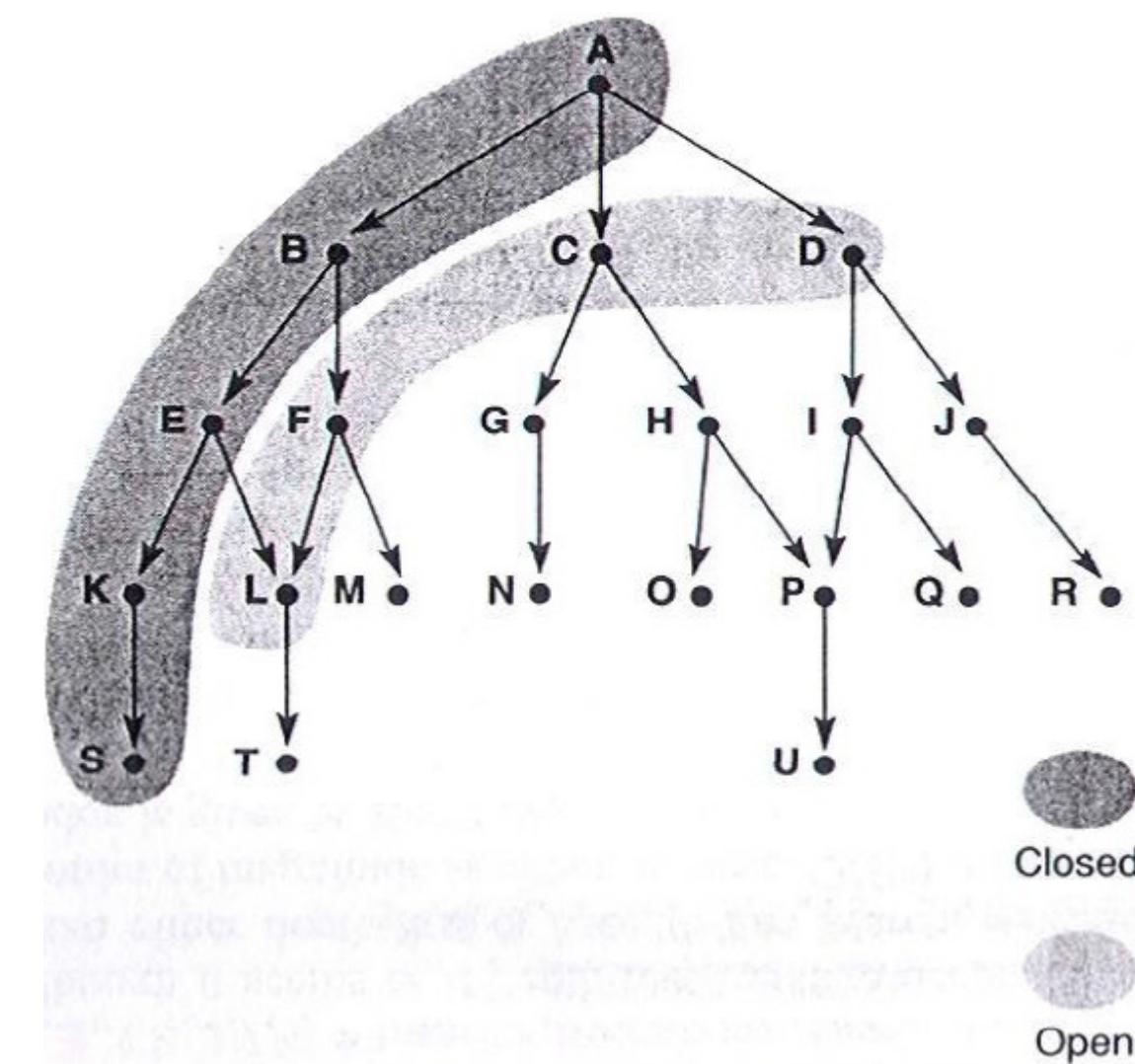
Open = [G,H,I,J,K,L,M];(vì L đã có trong open);

closed = [F,E,D,C,B,A]

...



■ DFS



Open = [A]; closed = []

Open = [B,C,D]; closed = [A]

Open = [E,F,C,D];closed = [B,A]

Open = [K,L,F,C,D];
closed = [E,B,A]

Open = [S,L,F,C,D];
closed = [K,E,B,A]

Open = [L,F,C,D];
closed = [S,K,E,B,A]

Open = [T,F,C,D];
closed = [L,S,K,E,B,A]

Open = [F,C,D];
closed = [T,L,S,K,E,B,A]

...

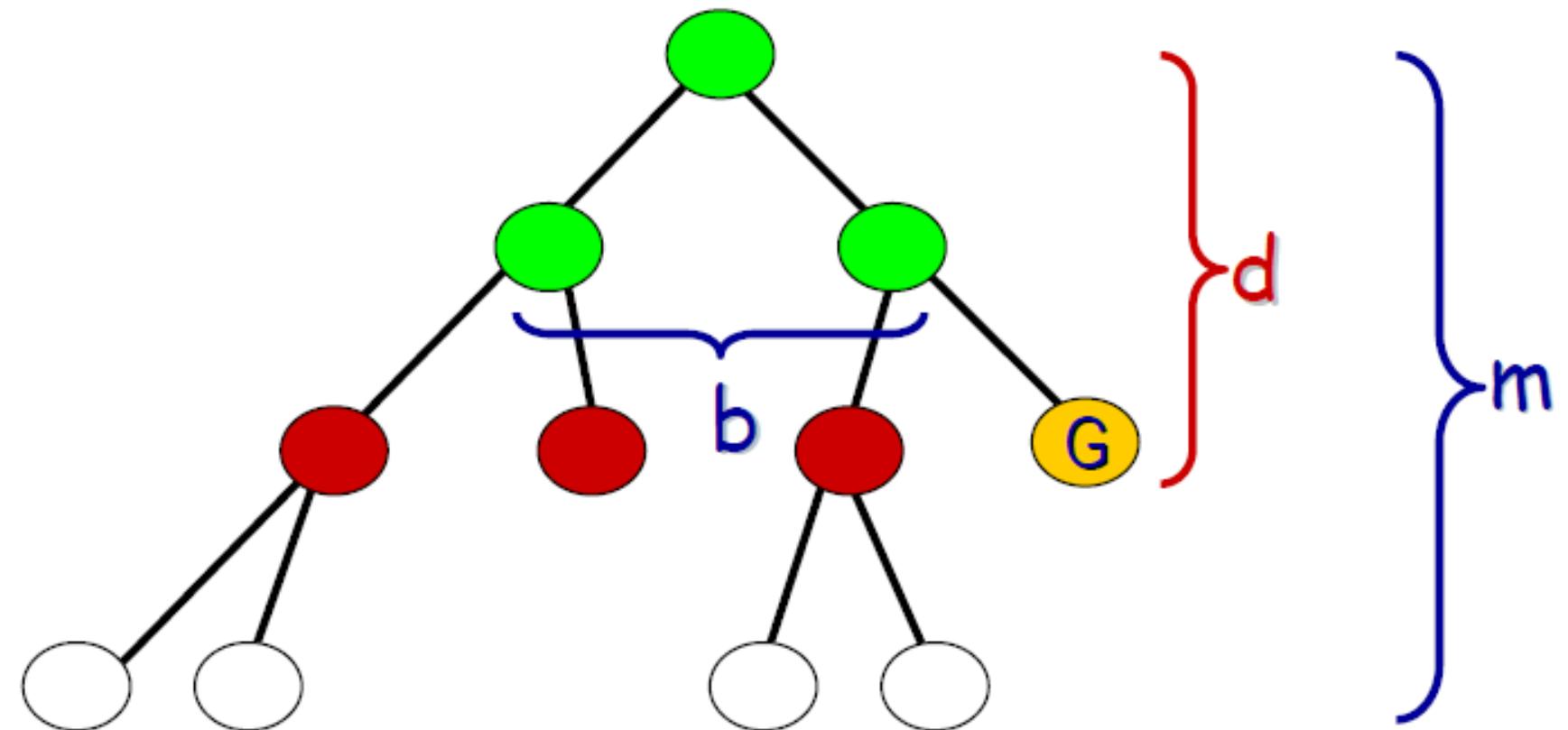
Chương 2. BÀI TOÁN VÀ PHƯƠNG PHÁP TÌM KIẾM LỜI GIẢI

▪ BFS

- OPEN được tổ chức dạng LIFO
- Nghiệm với số cung bé nhất
- Độ phức tạp thời gian: $O(b^d)$
- Độ phức tạp không gian: $O(b^d)$

▪ DFS

- OPEN được tổ chức dạng FIFO
- Thường cho kết quả nhanh hơn
- Độ phức tạp thời gian: $O(b^m)$
- Độ phức tạp không gian: $O(b.m)$



Hàng đợi trong giải thuật BFS chỉ chứa các nút lá (kích thước là b^d)

Chương 2. BÀI TOÁN VÀ PHƯƠNG PHÁP TÌM KIẾM LỜI GIẢI

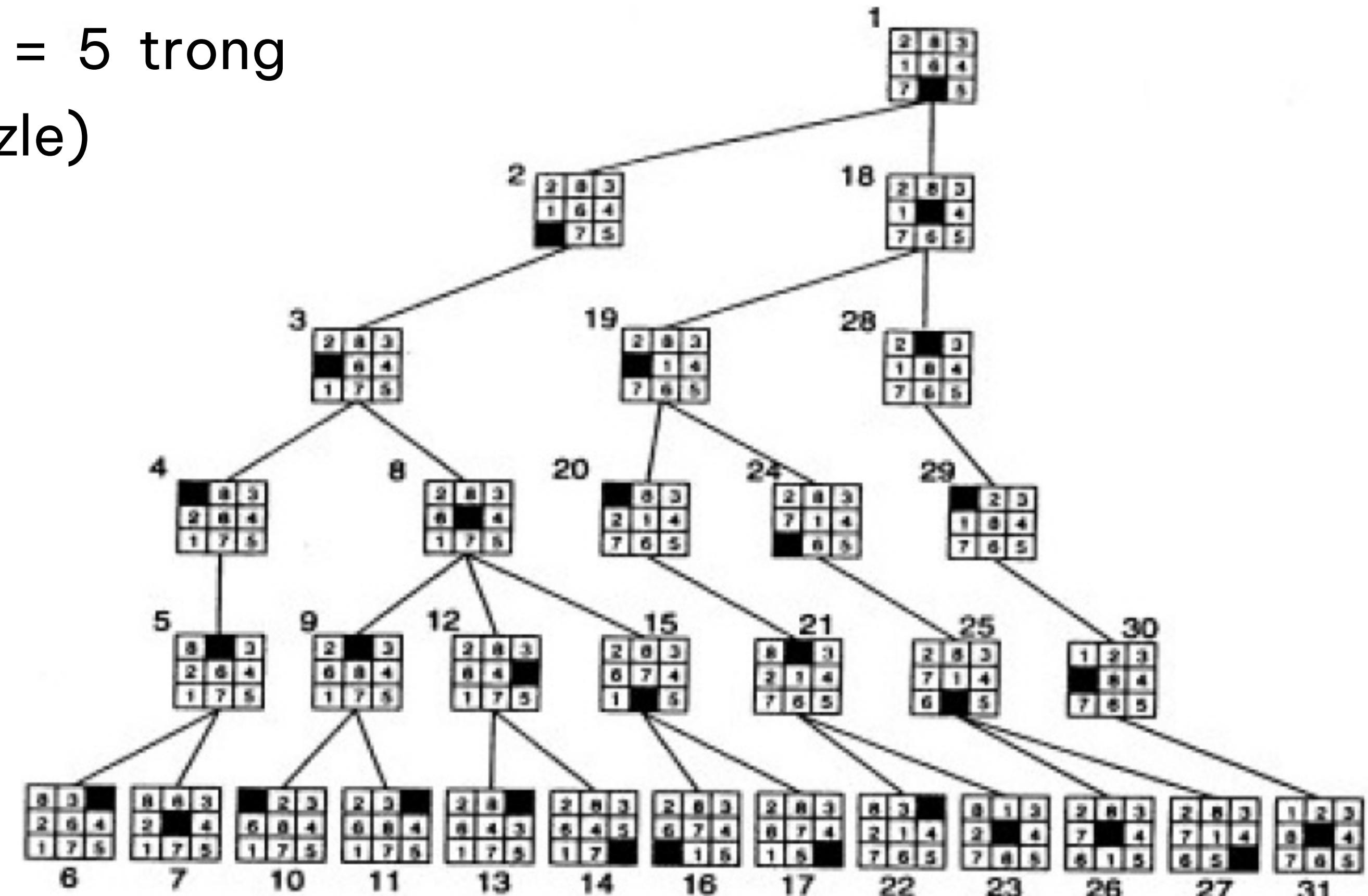
- Tìm kiếm theo chiều sâu có giới hạn (Depth Limited Search)
 - Giải thuật tìm kiếm theo chiều sâu ở trên có ưu điểm là nó có thể sinh ra lời giải nhanh chóng mà không tốn kém bộ nhớ.
 - Tuy nhiên nếu không gian trạng thái của bài toán là vô hạn thì rất có thể nó không tìm được lời giải của bài toán khi hướng tìm kiếm không chứa trạng thái đích.
 - ⇒ Để khắc phục nhược điểm này, chúng ta có thể đặt giới hạn độ sâu trong giải thuật: nếu độ sâu của trạng thái đang xét vượt quá ngưỡng nào đó thì chúng ta không bổ sung các nút kề với trạng thái này nữa mà chuyển sang hướng tìm kiếm khác.

Chương 2. BÀI TOÁN VÀ PHƯƠNG PHÁP TÌM KIẾM LỜI GIẢI

- Tìm kiếm theo chiều sâu có giới hạn (Depth Limited Search)
 - Chiến lược giới hạn
 - Cố định một độ sâu MAX, như các kỳ thủ chơi cờ tính trước một số nước nhất định.
 - Tùy theo cấu hình phần cứng của máy tính.
 - Sử dụng Meta Knowledge trong giới hạn độ sâu
 - Giới hạn độ sâu cũng đồng nghĩa với việc co hẹp không gian trạng thái và nguy cơ dẫn đến mất nghiệm.

Chương 2. BÀI TOÁN VÀ PHƯƠNG PHÁP TÌM KIẾM LỜI GIẢI

- Tìm kiếm theo chiều sâu có giới hạn (Depth Limited Search)
 - DFS có giới hạn = 5 trong trò chơi 8 - puzzle)



HẾT CHƯƠNG 2