

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 != ∅){  //Open: các đỉnh có thể xét ở bước kế tiếp
    n = Retrieve(Open); //n: đỉnh đang xét
    Close = Close ∪ {n} //gán vào tập CLOSE
    If(n==g) Return TRUE; //g: đỉnh kết thúc
    If(n!=g && n∉{OPEN, CLOSE})//n là đỉnh chưa thuộc về OPEN/CLOSE
        Open = Open ∪ Γ(n); //Γ(n): các đỉnh có thể đi trực tiếp từ 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);

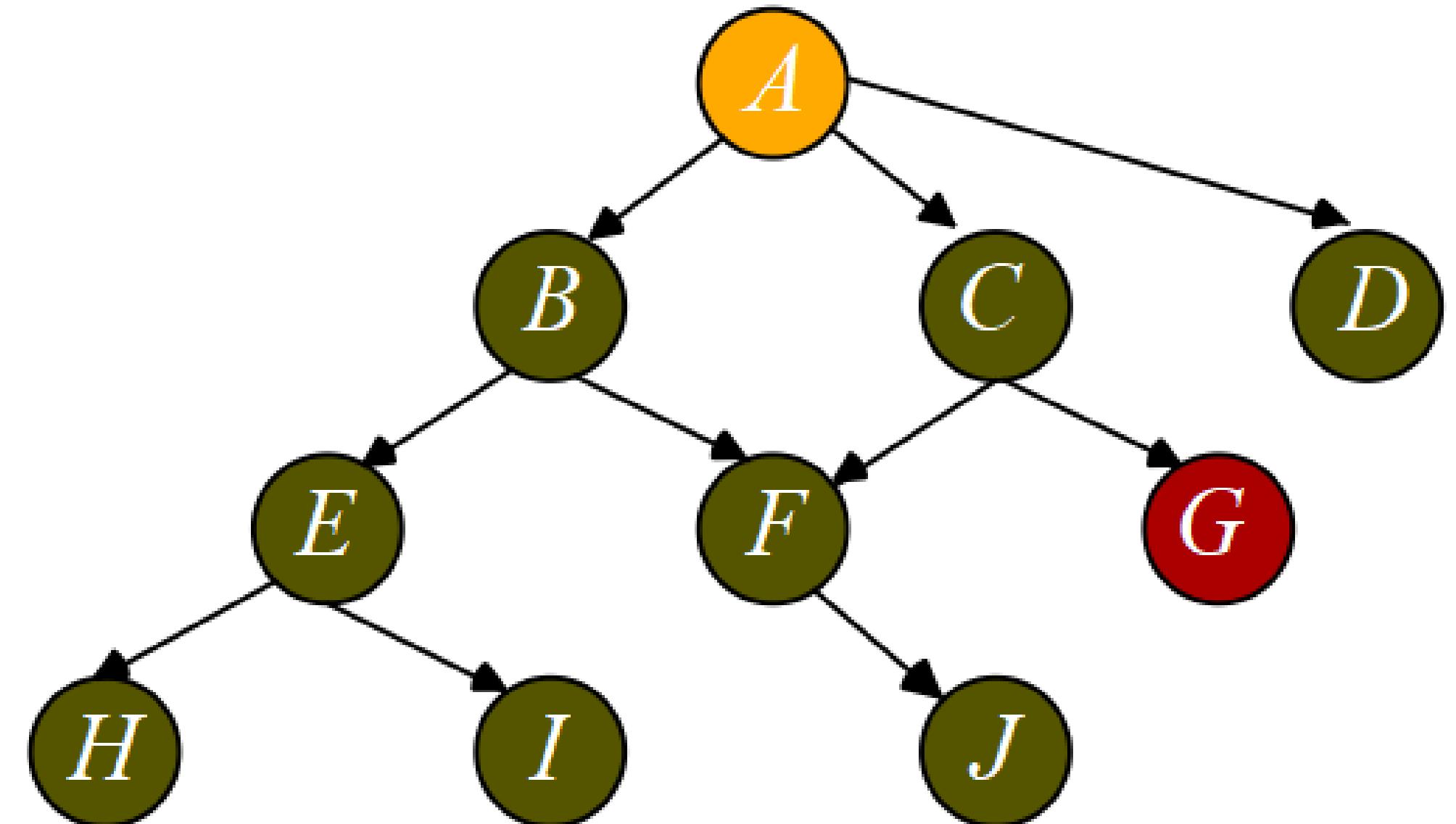
 If(n==g) Return TRUE;

 Open = Open ∪ Γ(n);

 Close = Close ∪ {n}

}

End;



Ví dụ 1

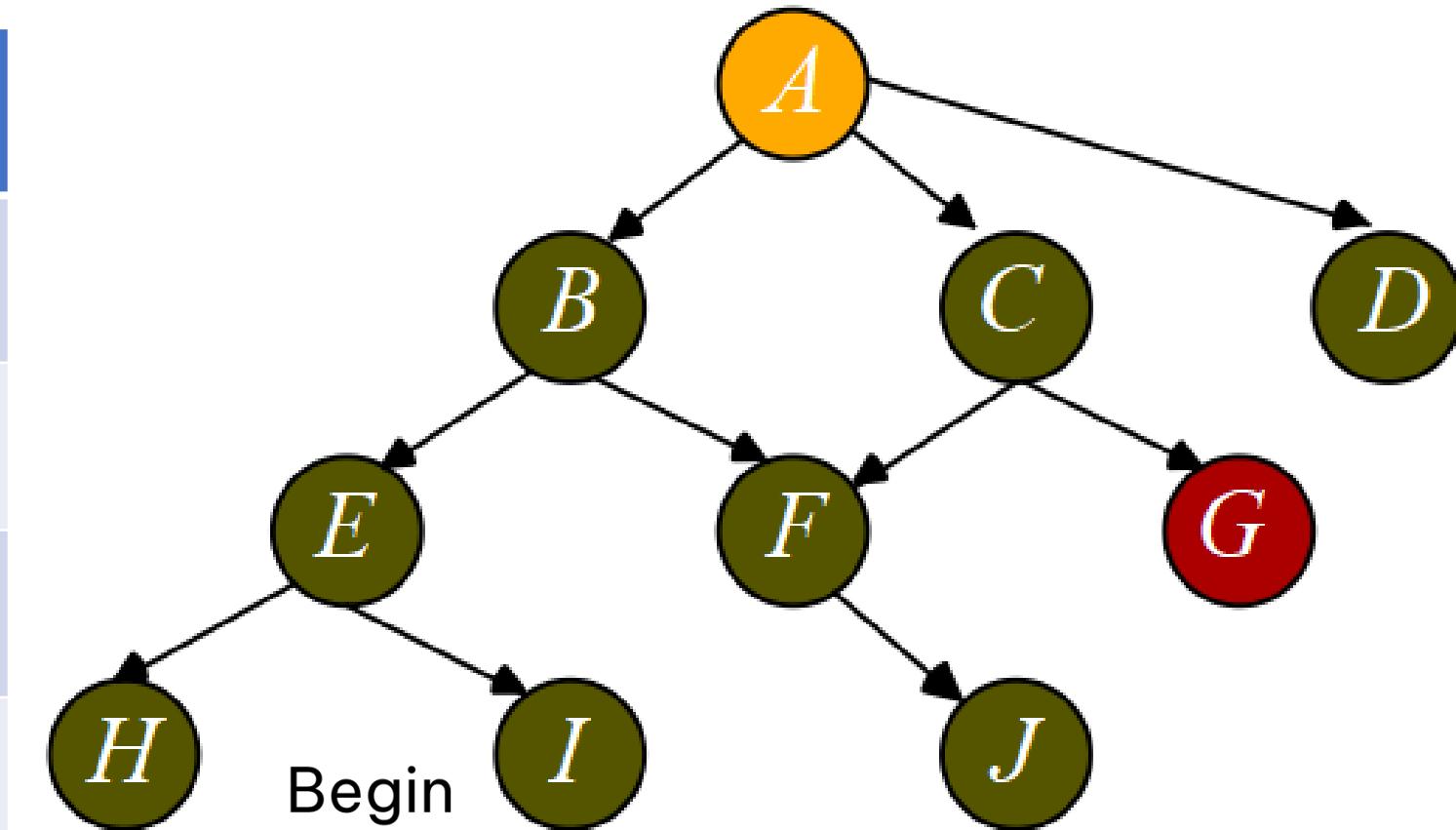
s = A là đỉnh bắt đầu g=G là đỉnh kết thúc

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);

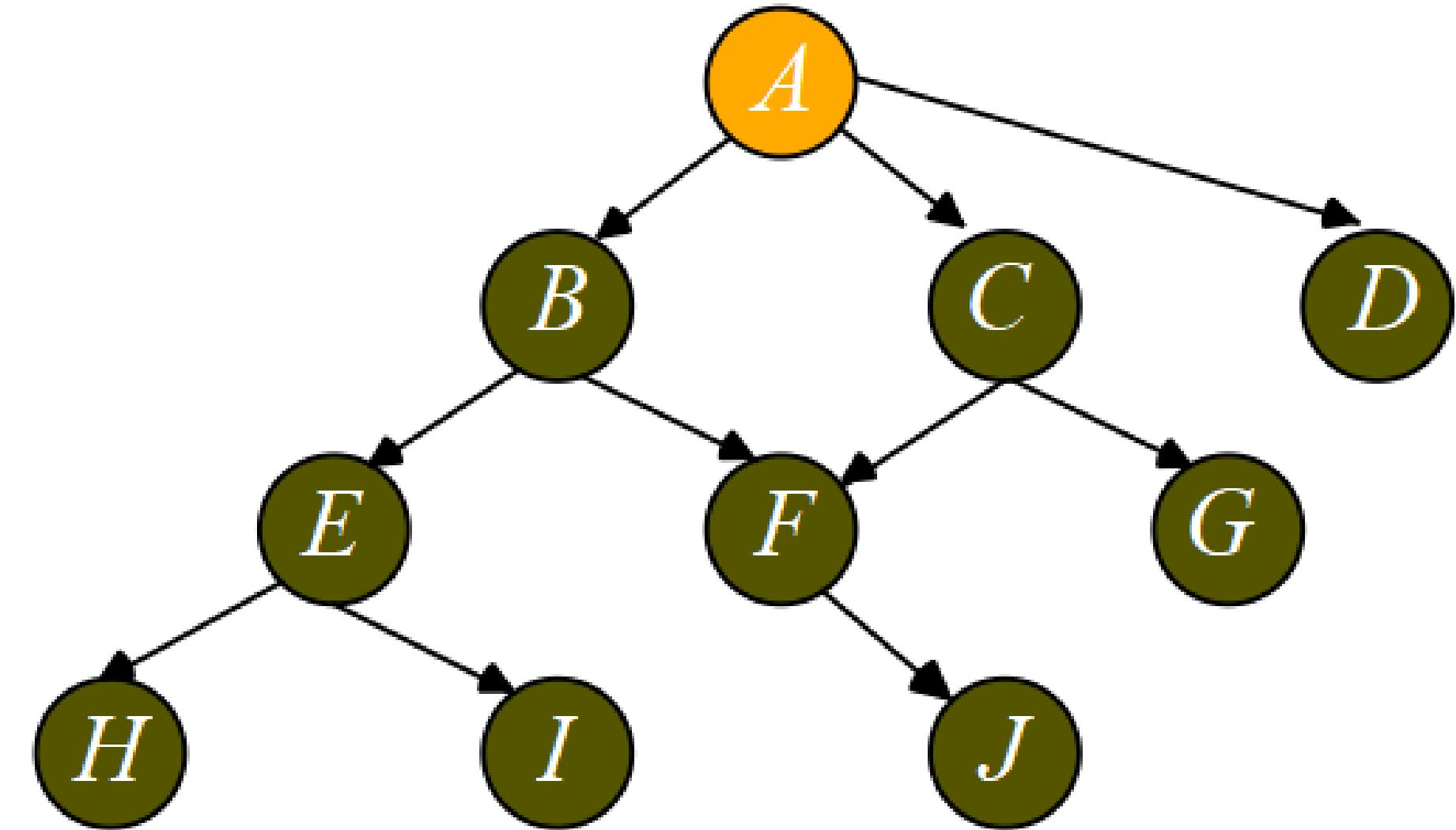
If(n==g) Return TRUE;

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

Close = Close \cup {n}

}

End;



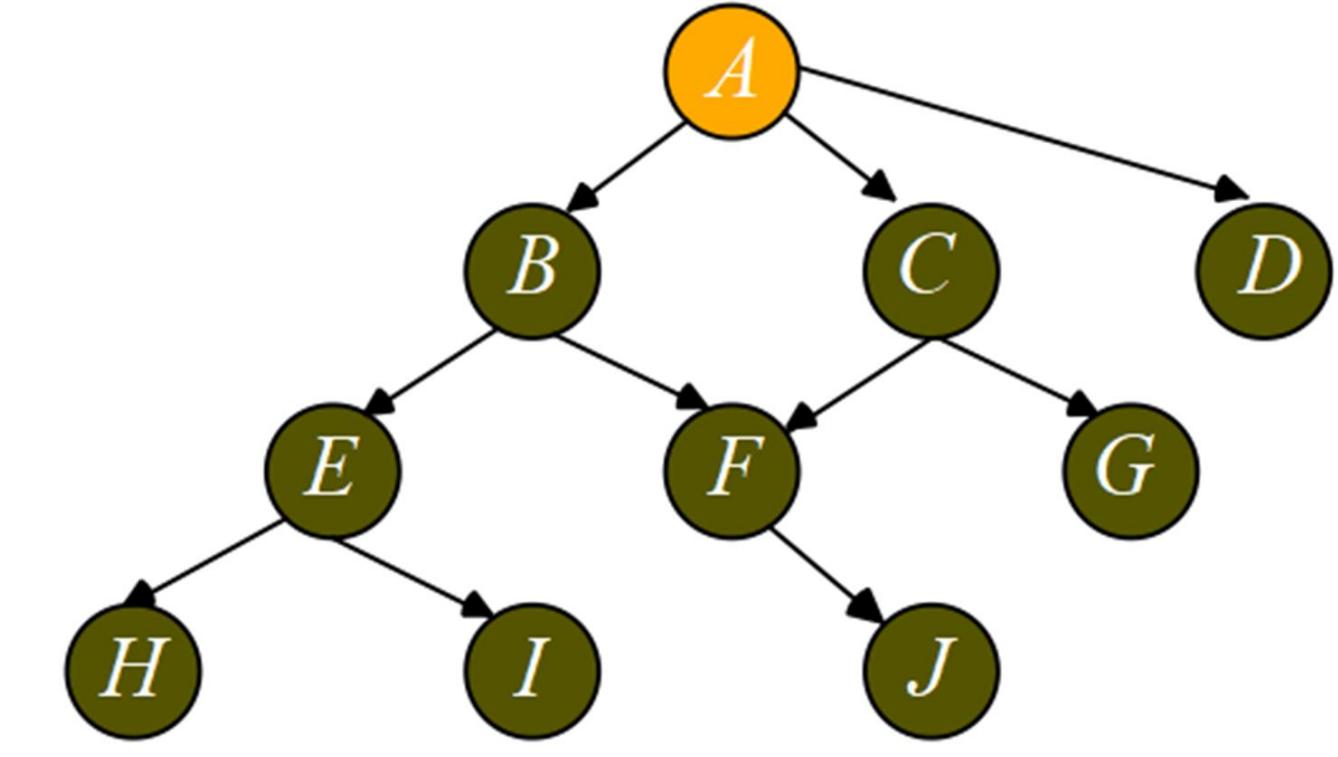
Ví dụ 2

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}



Begin

```

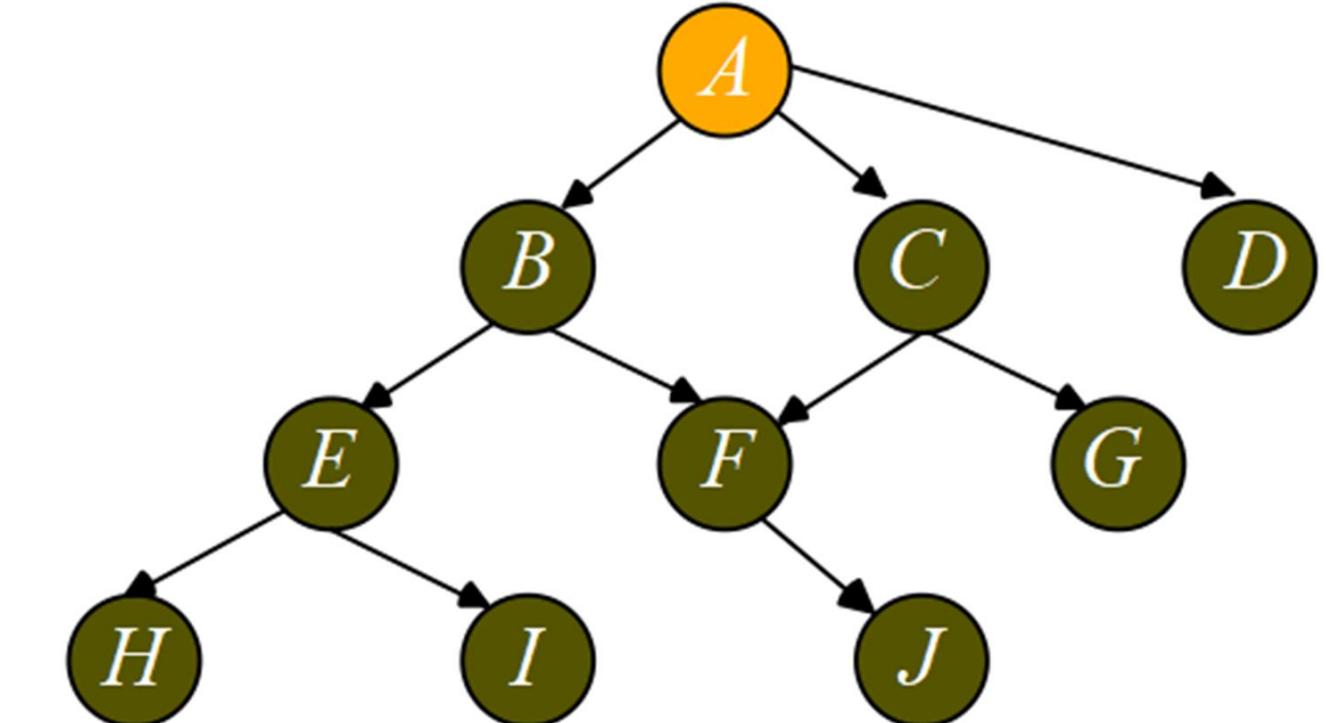
Open={s};
Close= $\emptyset$ ;
While(Open !=  $\emptyset$ ){
    n = Retrieve(Open);
    If(n==g) Return TRUE;
    Open = Open  $\cup$   $\Gamma(n)$ ;
    Close = Close  $\cup$  {n}
}
End;
  
```

$s = A$ 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)

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);

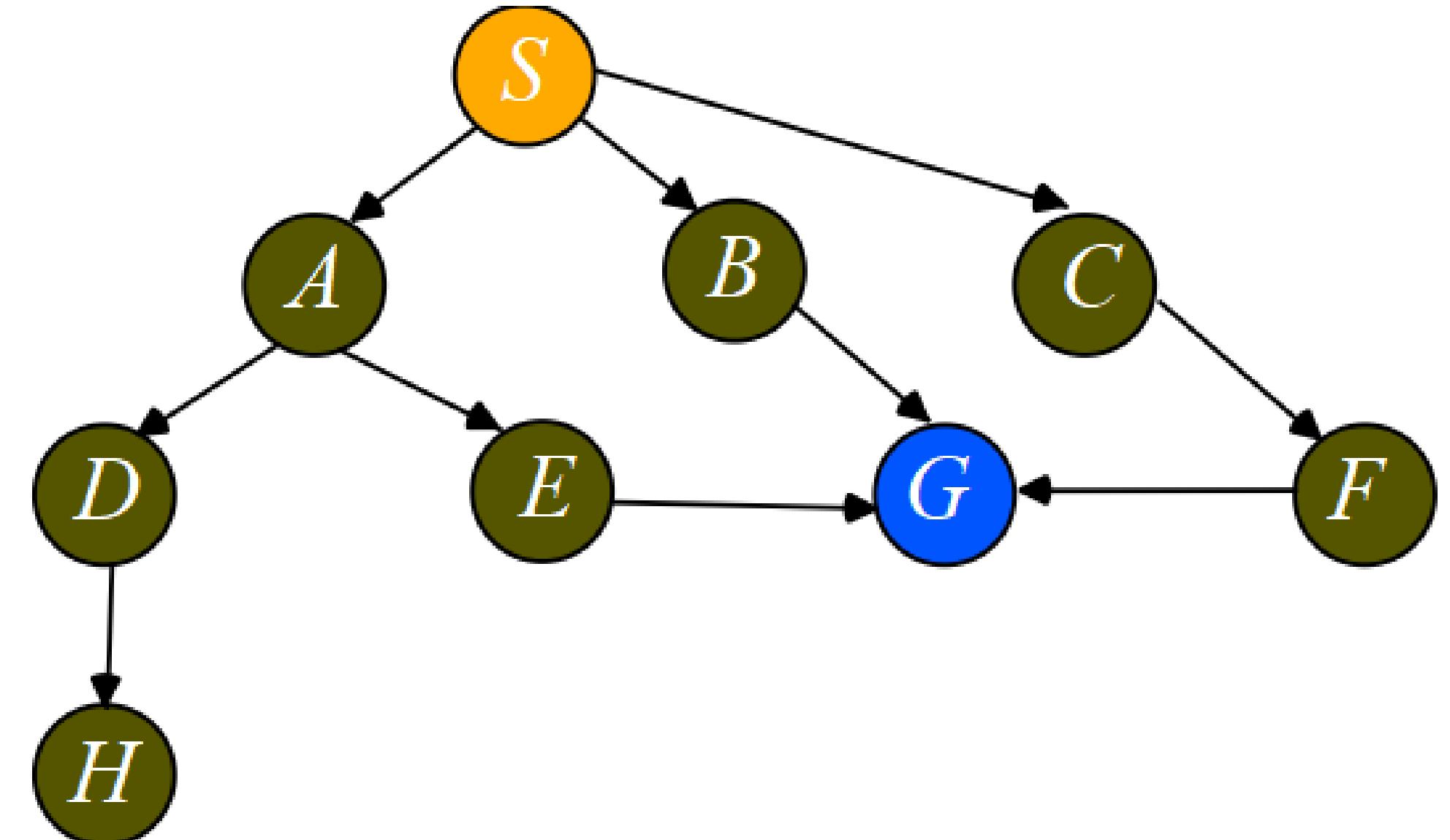
 If(n==g) Return TRUE;

 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 ∪ Γ(n);  
    Close = Close ∪ {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 1 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0

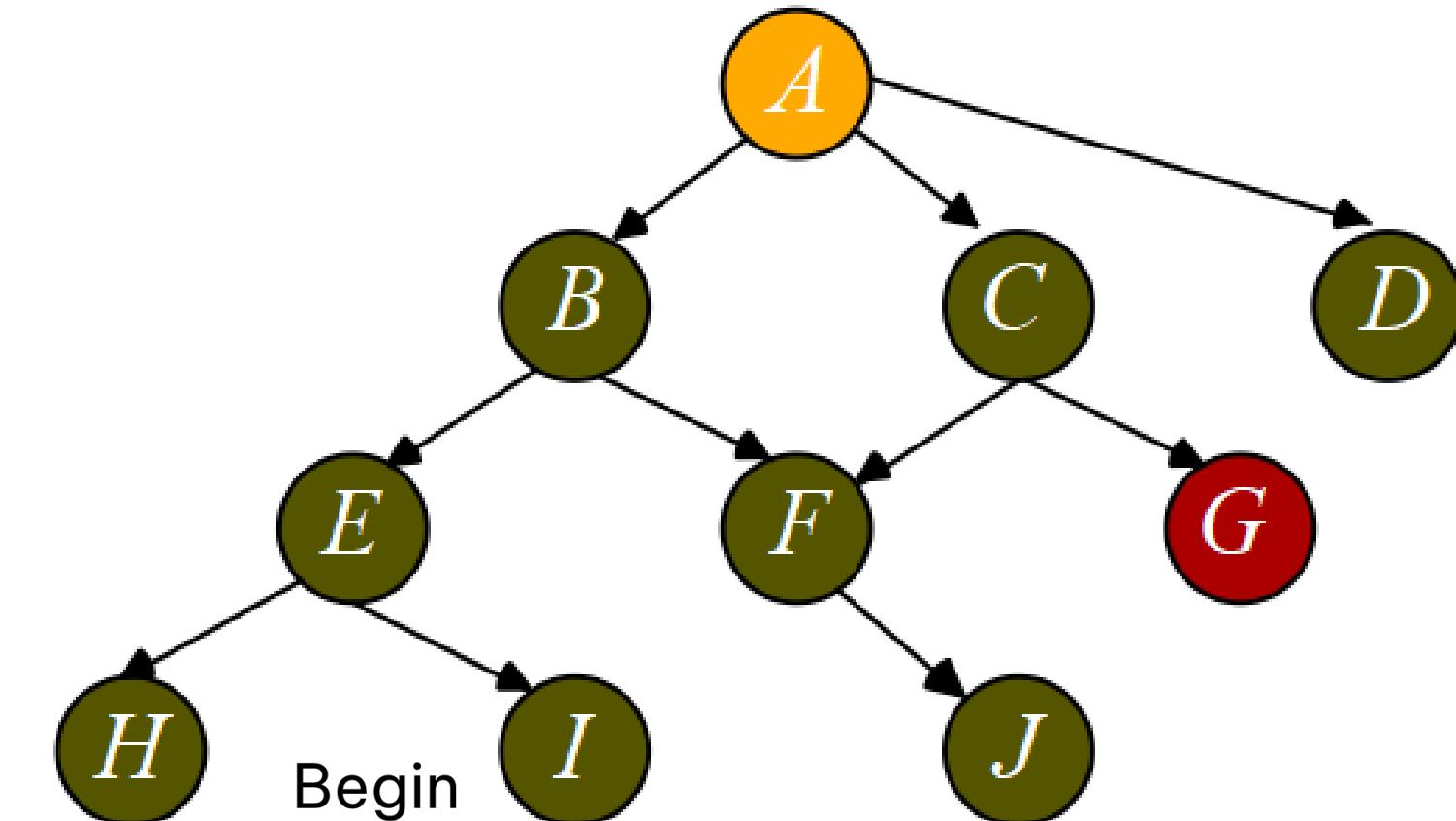
A | B | C | D | E | F | G | H | I | J

Nhap dinh bat dau: 0

Nhap dinh ket thuc: 6

Duong di tu 0 den 6: 0 → 2 → 6

A
B
C
D
E
F
G
H
I
J



Begin

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 \cup $\Gamma(n)$;

Close = Close \cup {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 \cup $\Gamma(n)$;

Close = Close \cup {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};           //s: đỉnh xuất phát
Close=∅;           //Close: tập các đỉnh đã xét
While(Open != ∅){  //Open: các đỉnh có thể xét ở bước kế tiếp
    n = Retrieve(Open); //n: đỉnh đang xét
    Close = Close ∪ {n} //gán vào tập CLOSE
    If(n==g) Return TRUE; //g: đỉnh kết thúc
    If(n!=g && n∉{OPEN, CLOSE})//n là đỉnh chưa thuộc về OPEN/CLOSE
        Open = Γ(n) ∪ Open; //Γ(n): các đỉnh có thể đi trực tiếp từ n
    }
}
// chèn vào đầu OPEN
```

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);

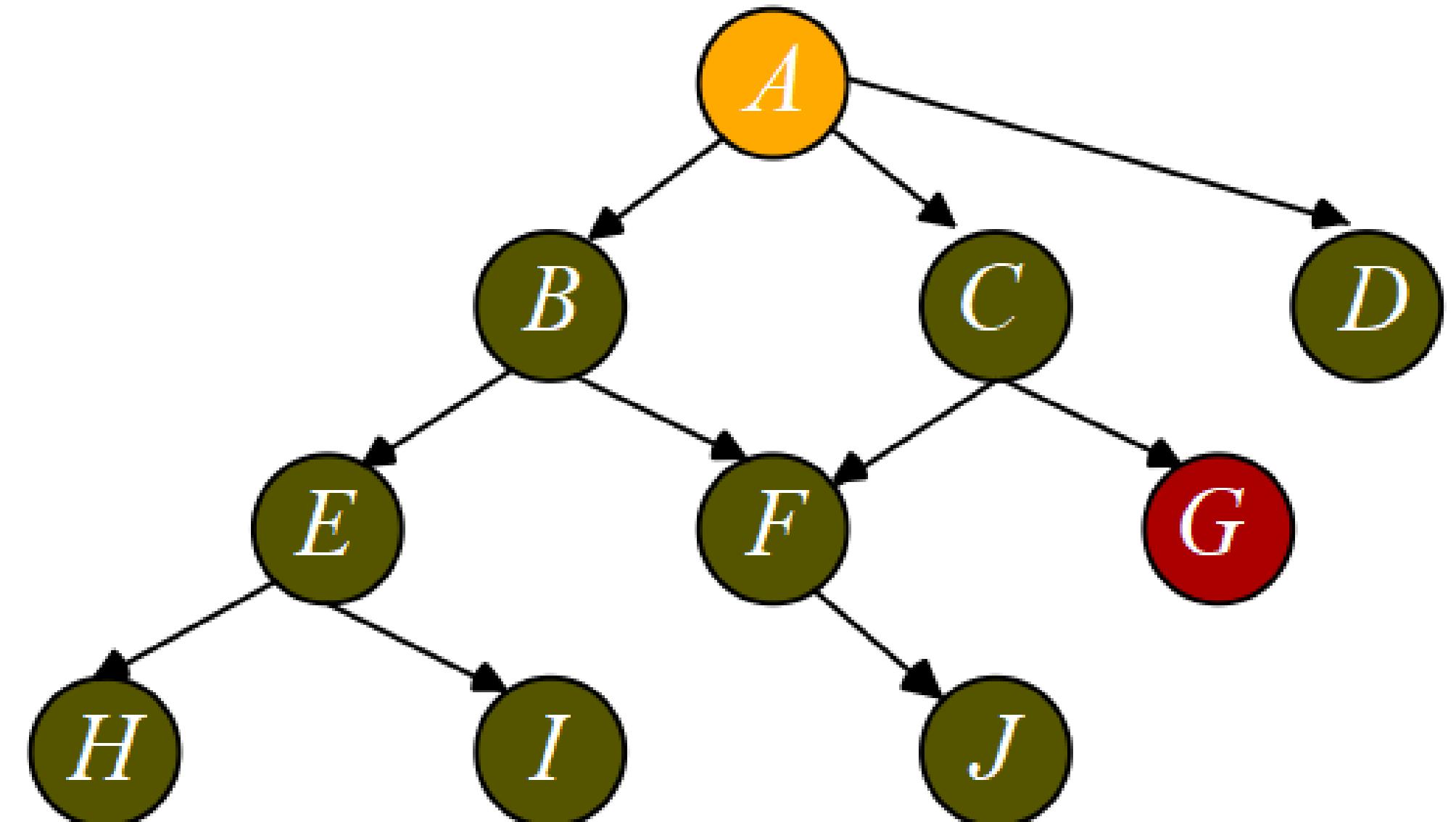
 If(n==g) Return TRUE;

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

 Close = Close \cup {n}

}

End;



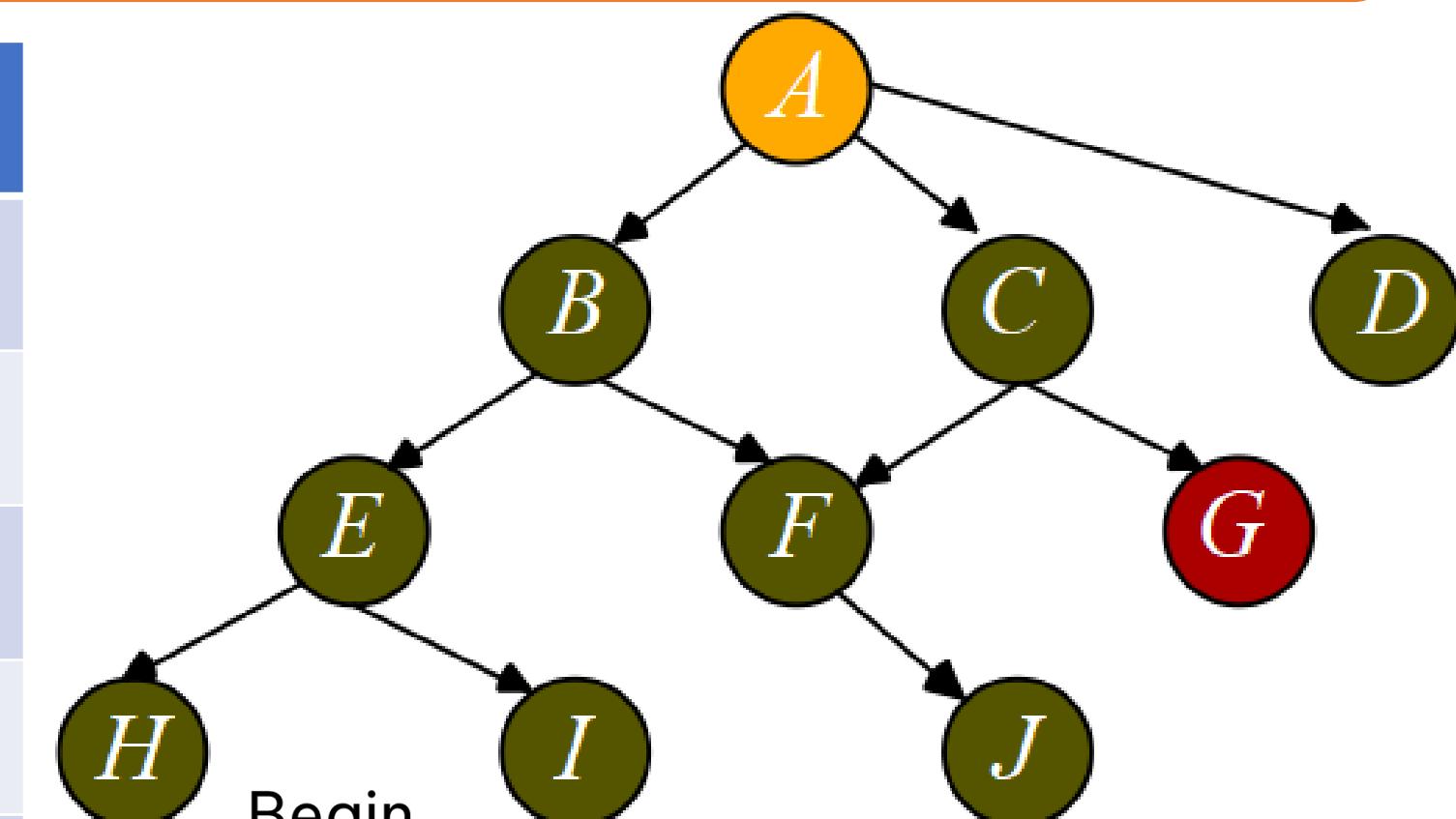
Ví dụ 1

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 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);

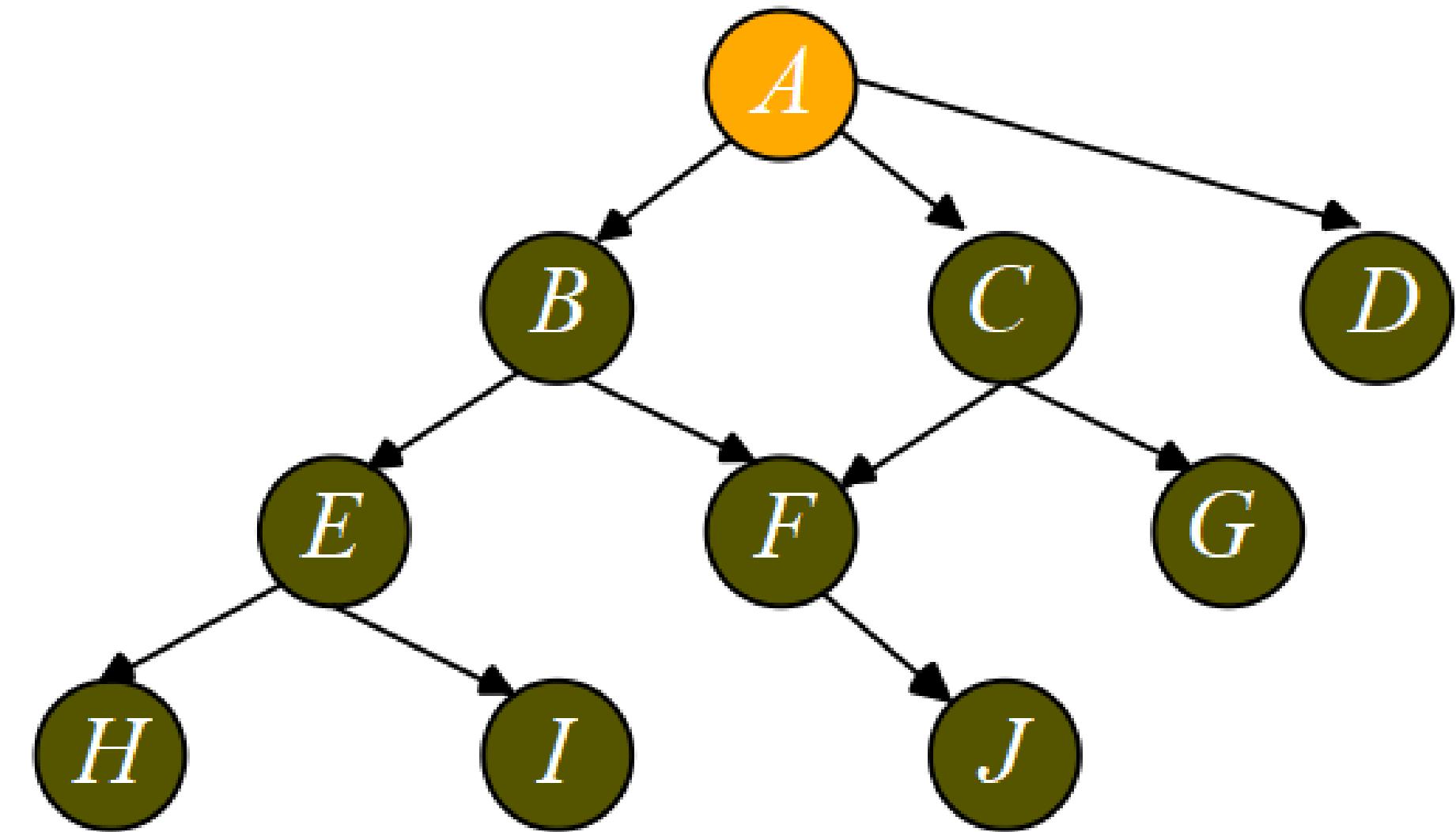
If(n==g) Return TRUE;

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

Close = Close \cup {n}

}

End;



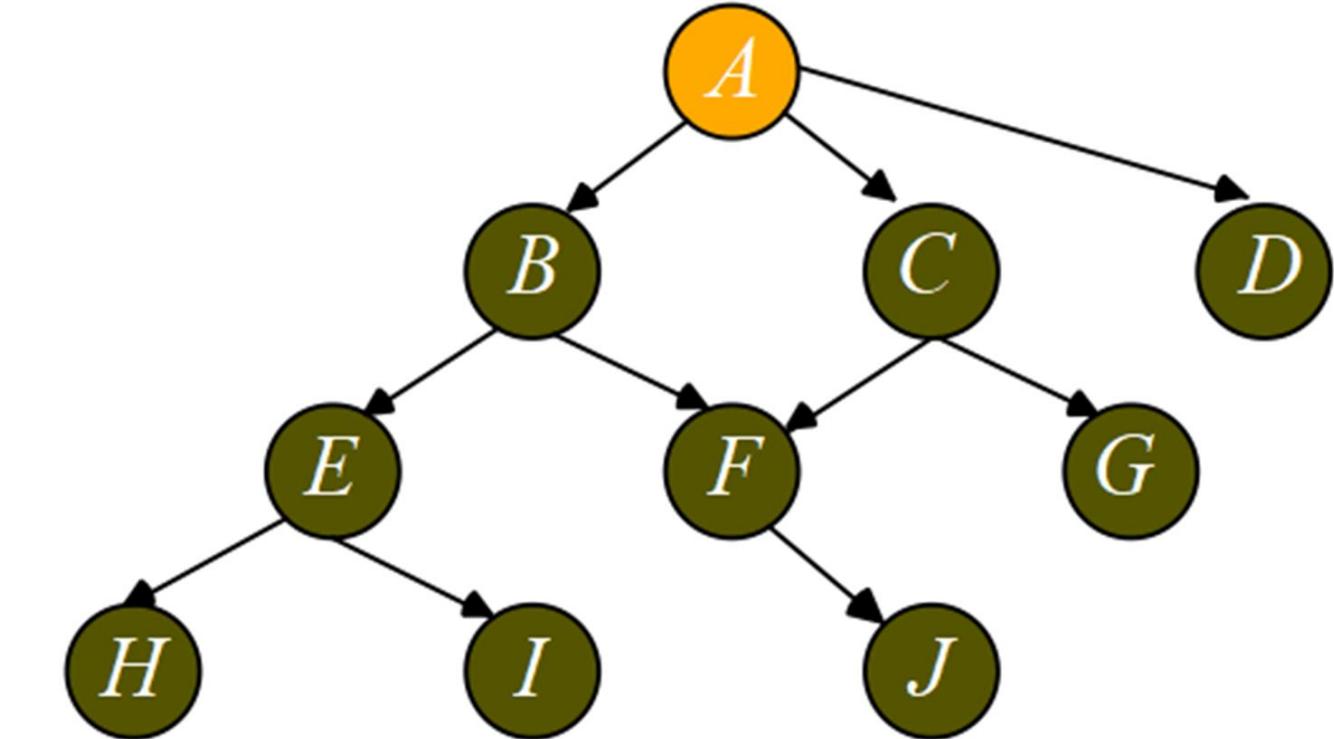
Ví dụ 2

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}

}

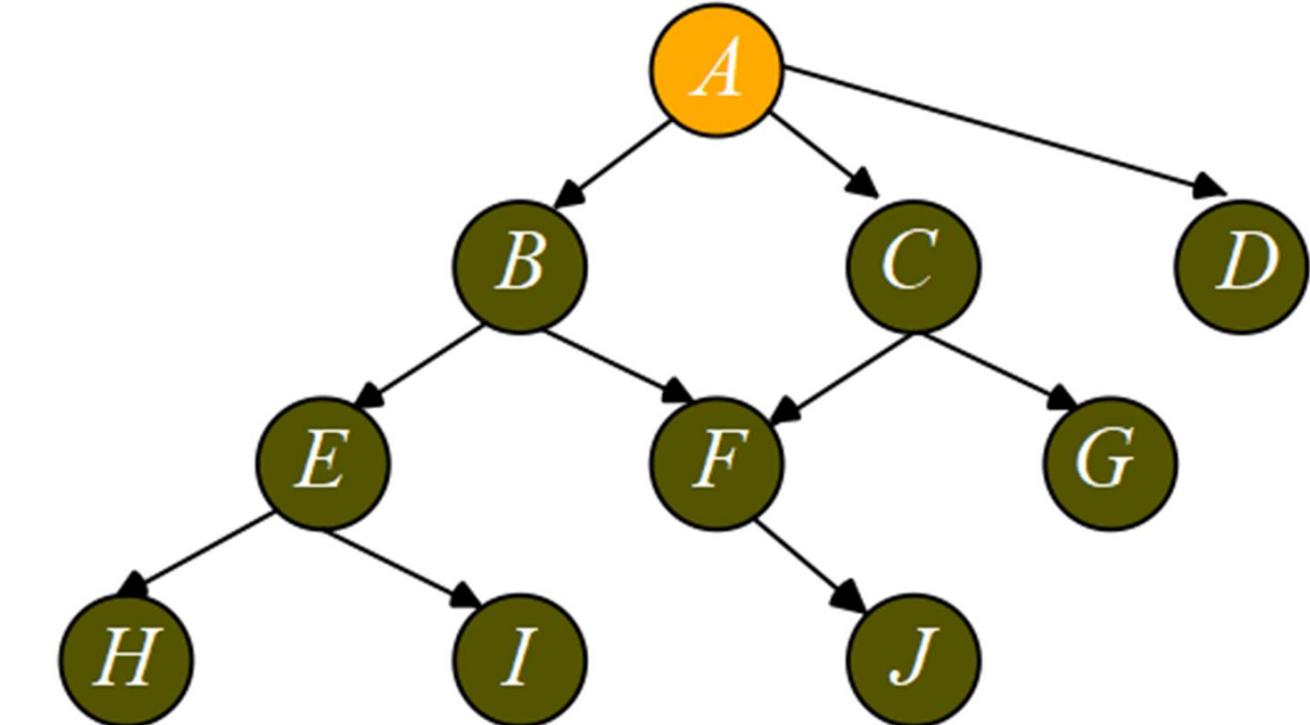
End;

$s = A$ 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);

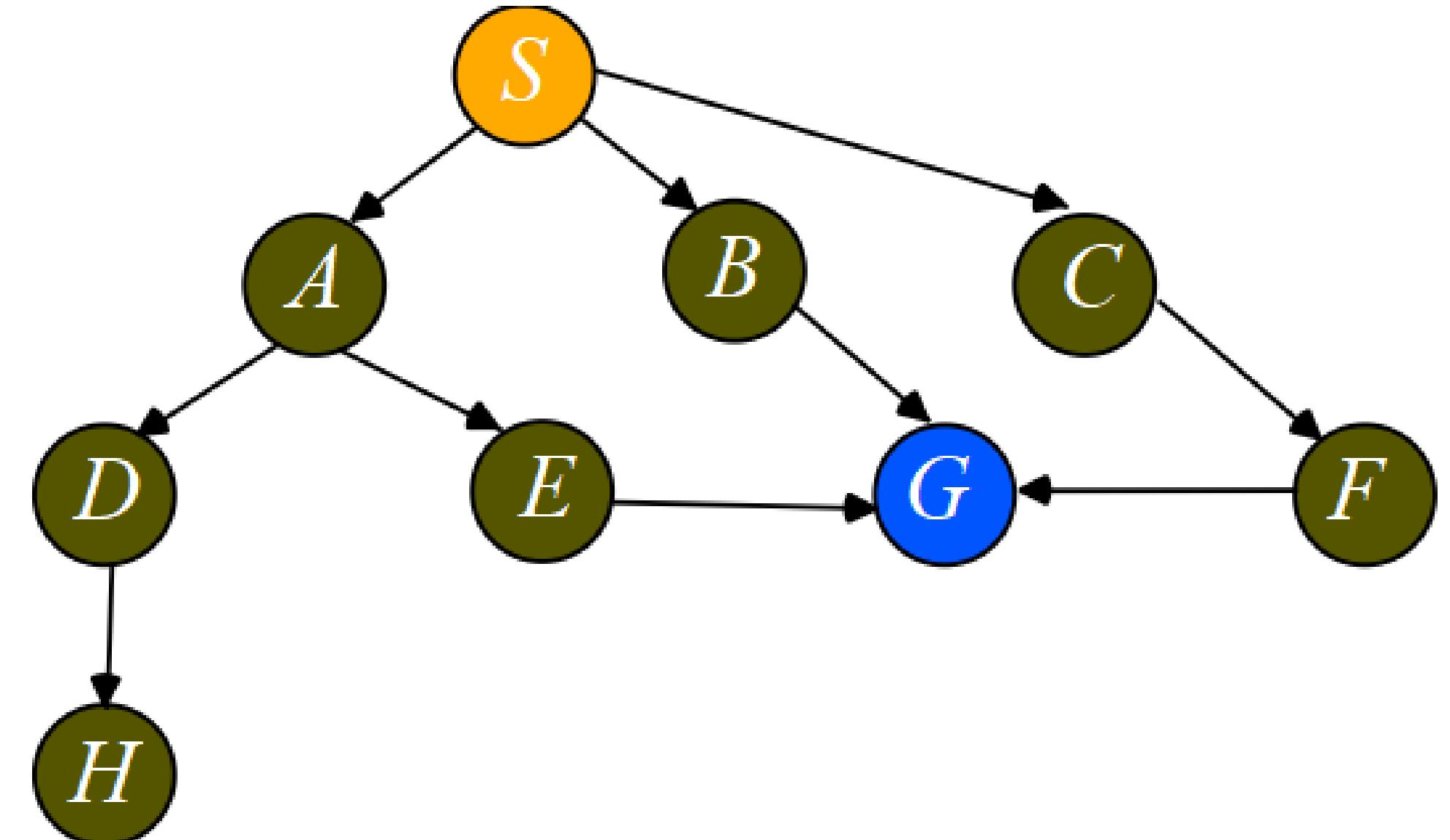
If(n==g) Return TRUE;

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]

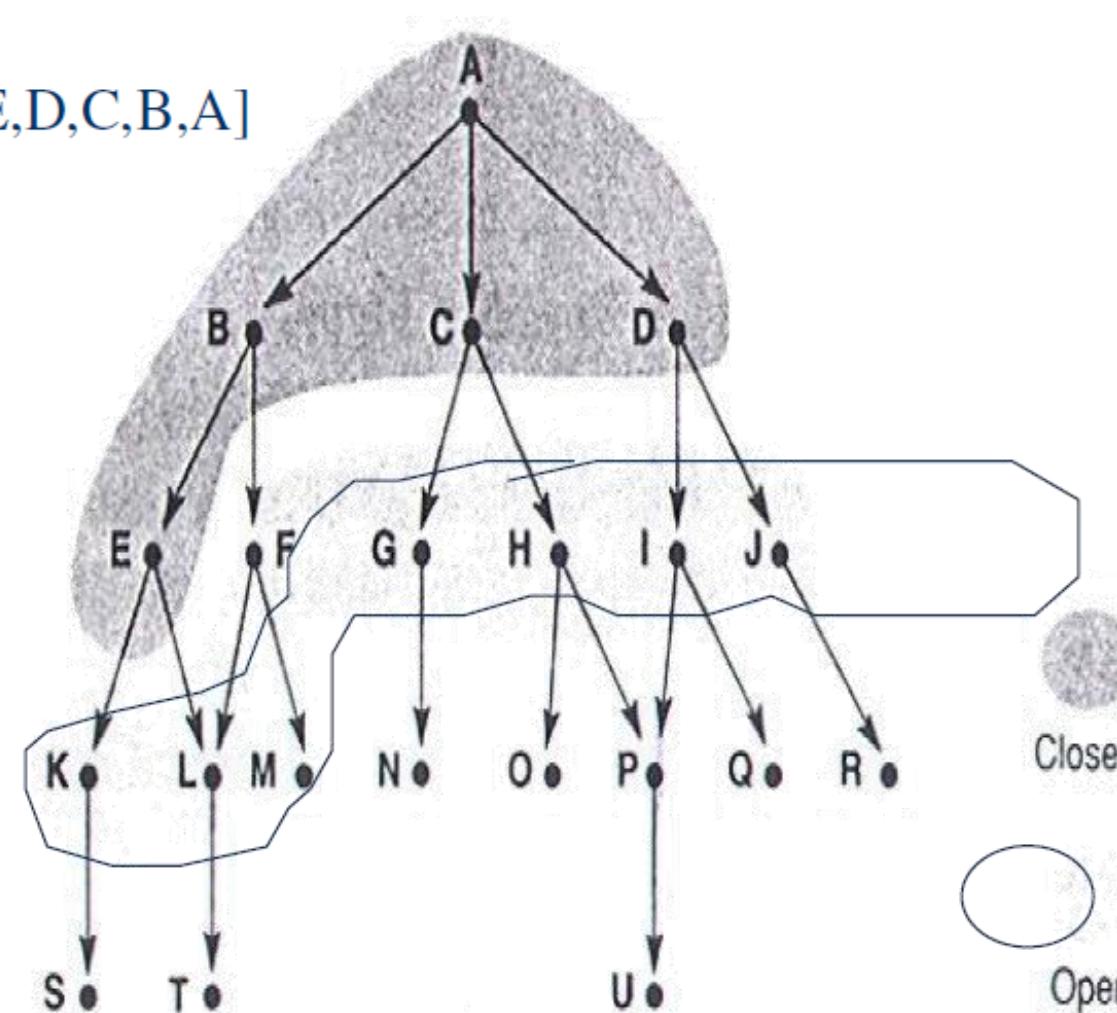
Open = [E,F,G,H,I,J]; closed = [D,C,B,A]

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

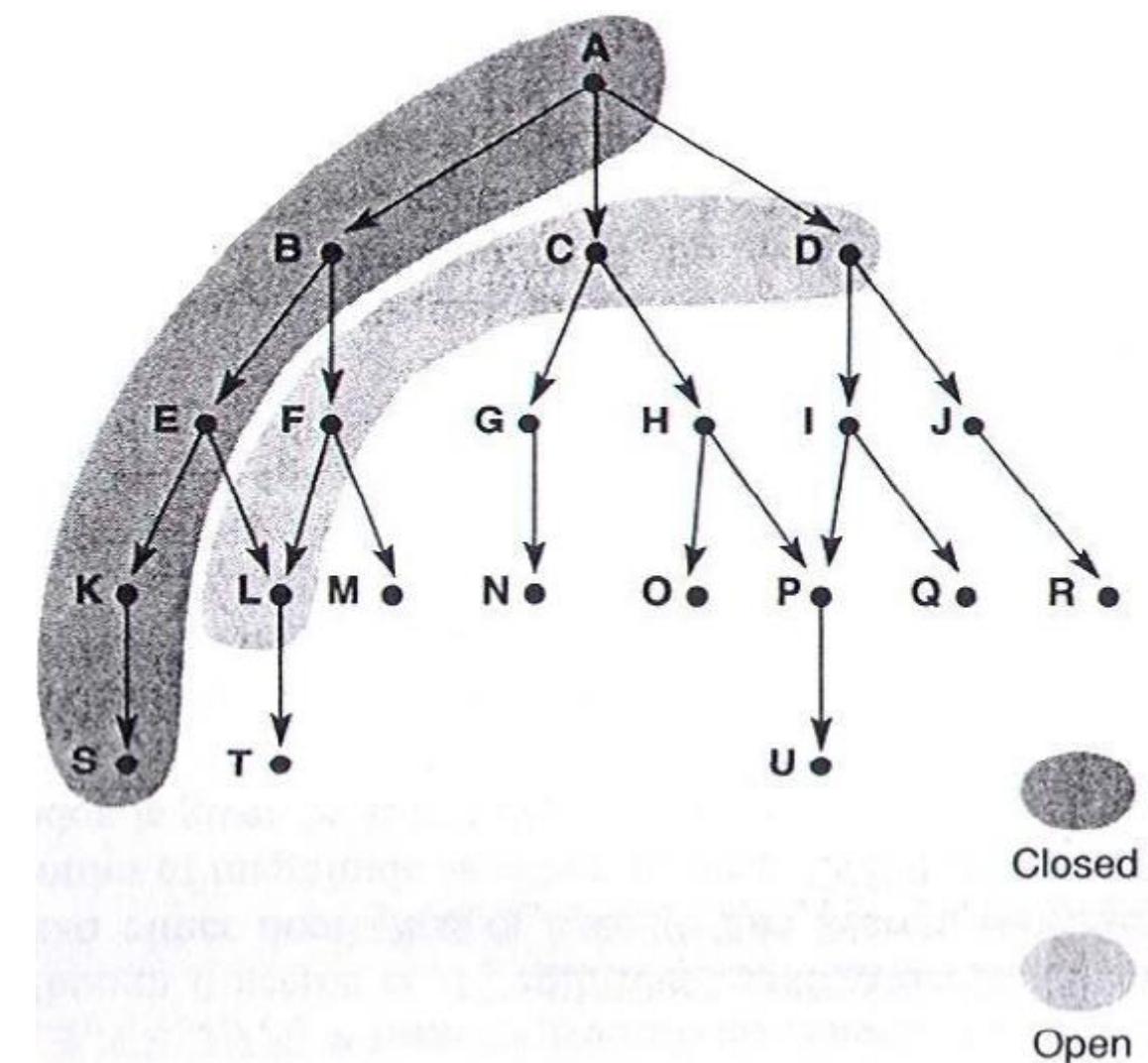
Open = [G,H,I,J,K,L,M];(vì L đã có trong open);

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]

...

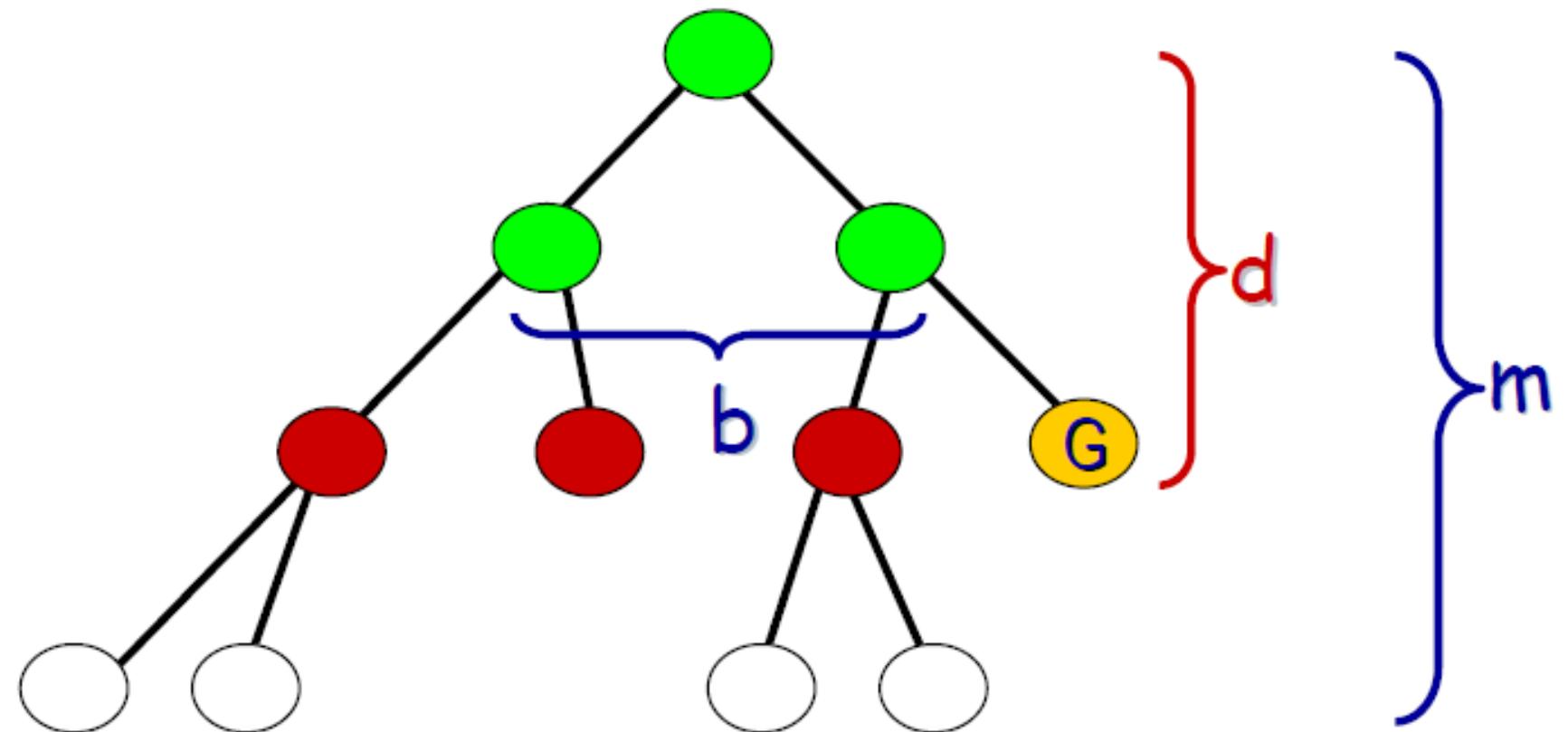
Chương 2. BÀI TOÁN VÀ PHƯƠNG PHÁP TÌM KIẾM LỜI GIẢI

■ 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: $O(b.m)$



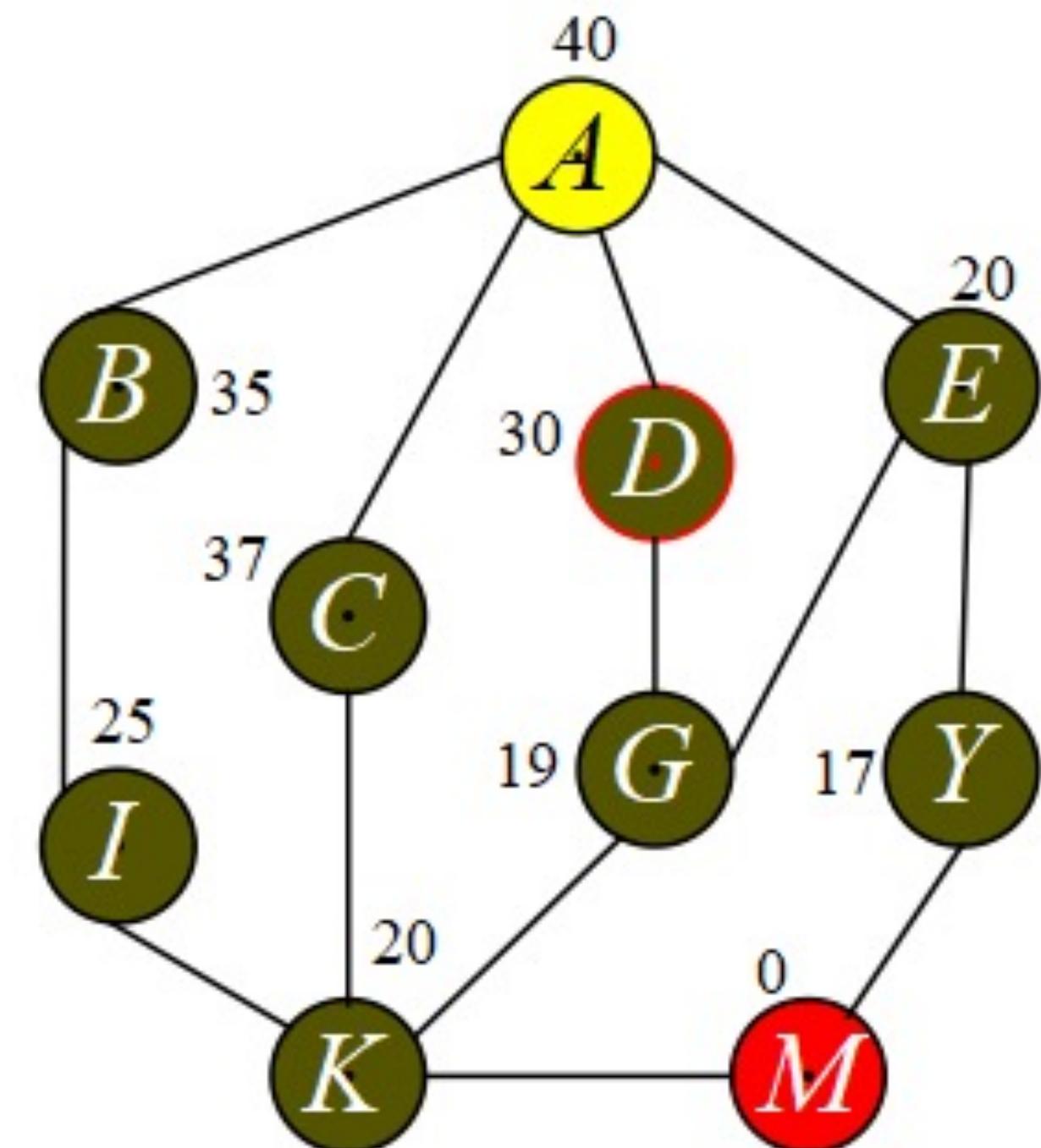
Hàng đợi trong giải thuật BFS chỉ chứa các nút lá (kích thước là b^d)

Chương 3. HEURISTIC

- Tìm kiếm leo đồi (Hill Climbing Search)
- Là giải thuật tối ưu của tìm kiếm cục bộ (Stefanie Gunther, 1983) nhằm tìm kiếm trạng thái tốt hơn có sử dụng hàm lượng giá h' .
- Bước 1. Khởi tạo ngăn xếp OPEN = start
- Bước 2. Loop:
 - if ($\text{OPEN} = \emptyset$): thất bại
 - $n = \text{phần tử đầu danh sách OPEN}$
 - if ($n = \text{goal}$): thành công (END)
 - for ($m: \text{mỗi đỉnh kề } \notin \{\text{OPEN} \cup \text{CLOSE}\}$) $\rightarrow \Gamma(n) // n$ chưa xét đến
 - Sắp xếp danh sách $\Gamma(n)$ theo thứ tự tăng dần của hàm đánh giá h'
 - Chèn $\Gamma(n)$ vào đầu danh sách OPEN

Chương 3. HEURISTIC

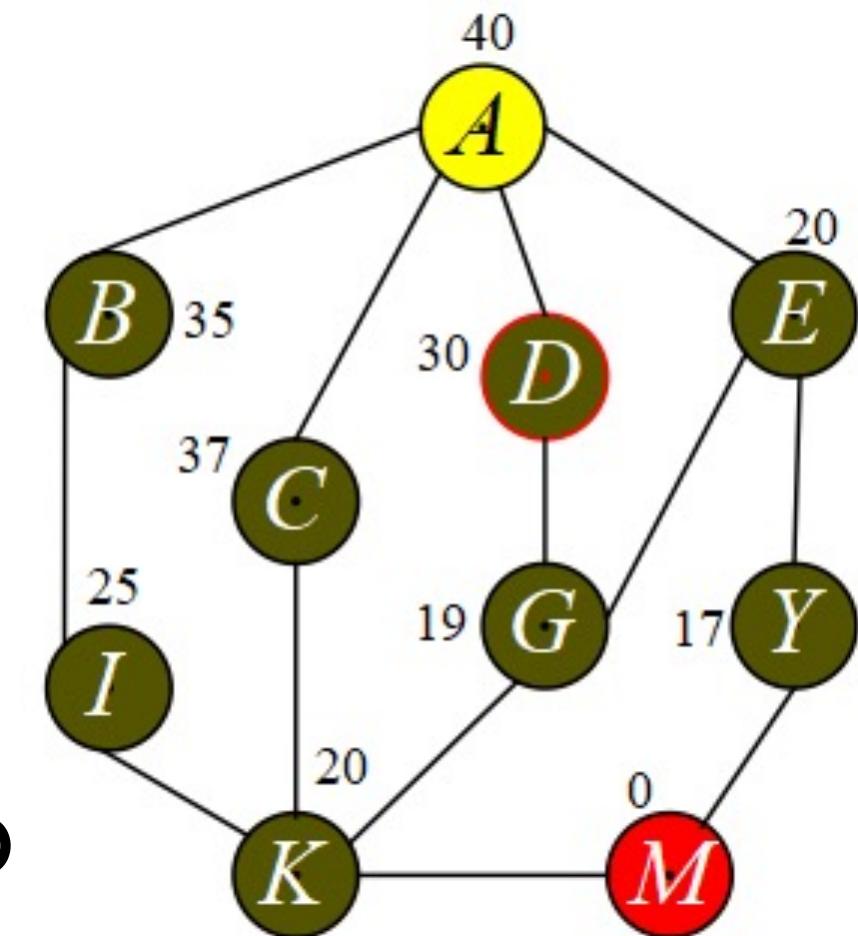
- Tìm kiếm leo đồi (Hill Climbing Search)
- Trình bày thuật toán leo đồi dốc đứng. Áp dụng thuật toán để tìm đường đi ngắn nhất từ đỉnh A đến M trên đồ thị, với các ước lượng heuristic của các trạng thái so với trạng thái đích được cho trong hình.



Chương 3. HEURISTIC

▪ Tìm kiếm leo đồi (Hill Climbing Search)

Bước	n	$\Gamma(n)$	OPEN	CLOSE
0			A	\emptyset
1	A	B(35), C(37), D(30), E(20)	E(20), D(30), B(35), C(37)	A
2	E	G(19), Y(17)	Y(17), G(19), D(30), B(35), C(37)	E
3	Y	M(0)	M(0), G(19), D(30), B(35), C(37)	Y
4	M	TRUE		

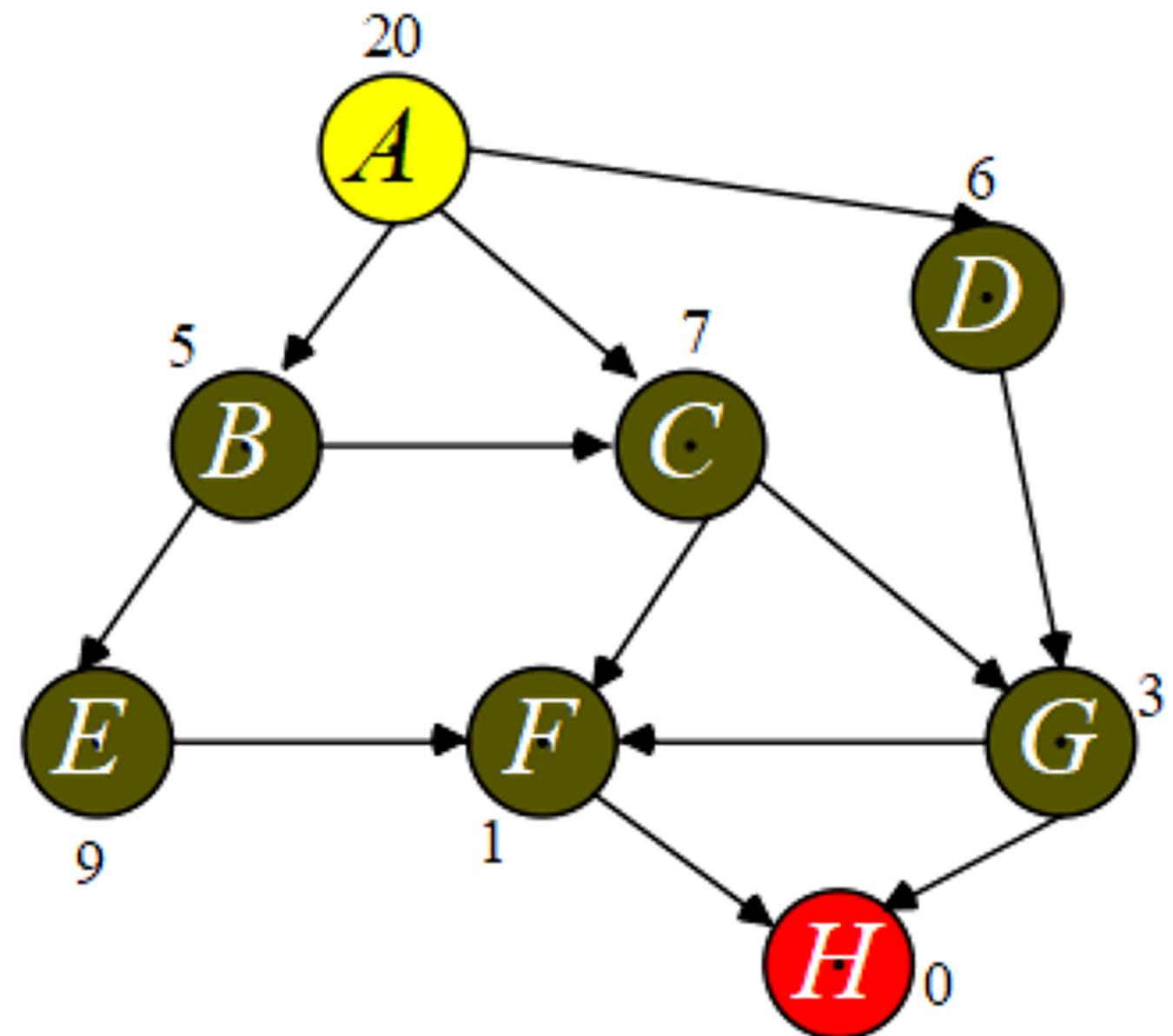


▪ A \Rightarrow E \Rightarrow Y \Rightarrow M

- Giải thuật có khuynh hướng sa lầy ở cực đại cục bộ
- Lời giải tìm được không tối ưu
- Không tìm được lời giải dù rằng có lời giải
- Có thể rơi vào vòng lặp vô hạn vì không lưu giữ trạng thái của các đỉnh đã xét.

Chương 3. HEURISTIC

- Tìm kiếm leo đồi (Hill Climbing Search)
- Trình bày thuật toán leo đồi dốc đứng. Áp dụng thuật toán để tìm đường đi ngắn nhất từ đỉnh A đến M trên đồ thị, với các ước lượng heuristic của các trạng thái so với trạng thái đích được cho trong hình.



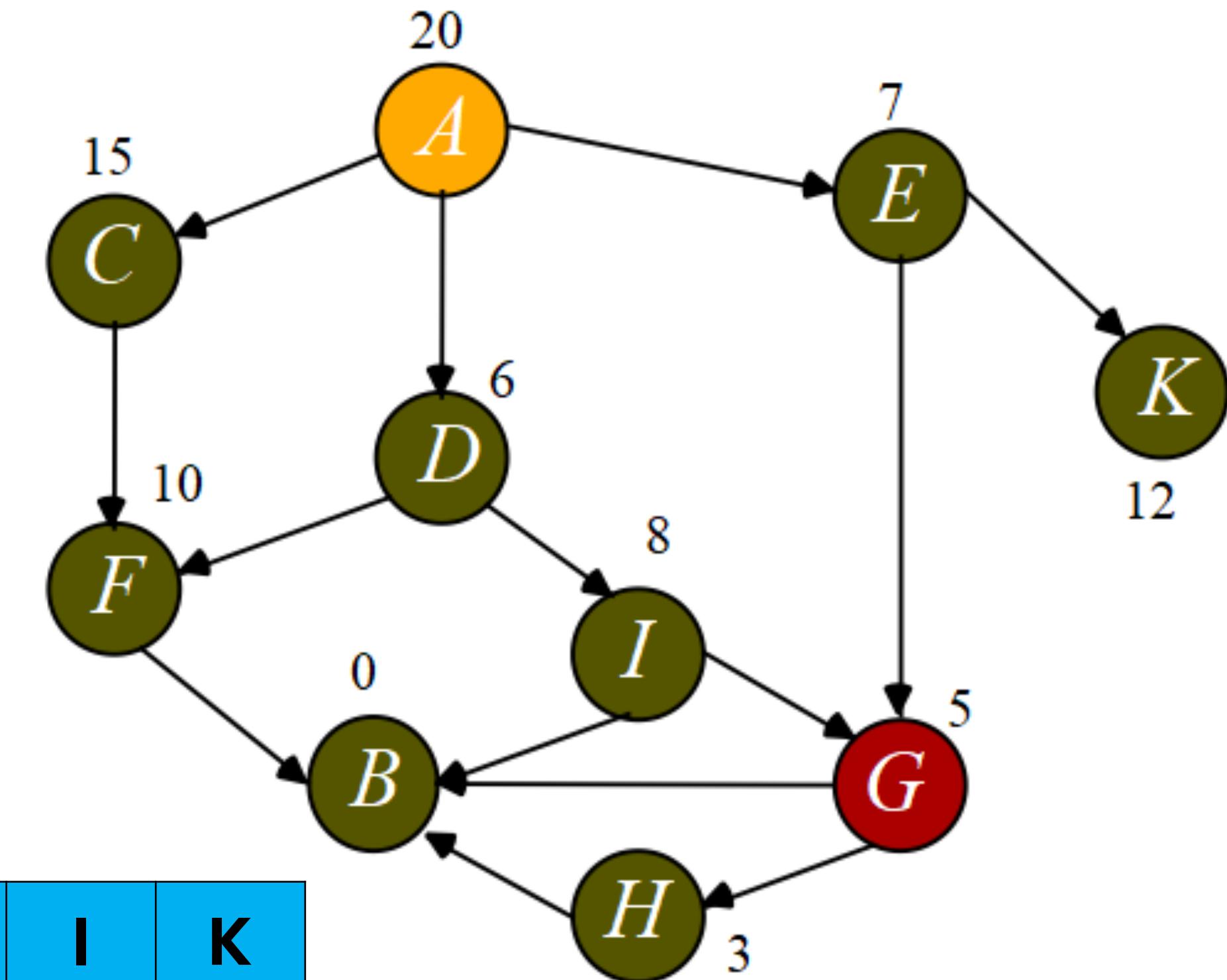
Chương 4. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

- Giải thuật tìm kiếm tốt nhất đầu tiên (Best First Search)
 - 1. Đặt OPEN chứa trạng thái đầu
 - 2. Lặp cho đến khi tìm được trạng thái đích hoặc không còn nút nào trong OPEN, thực hiện:
 - 2.1. Chọn trạng thái tốt nhất (T_{max}) trong OPEN (Xóa T_{max} ra khỏi danh sách OPEN)
 - 2.2. Nếu T_{max} là trạng thái đích thì Kết thúc
 - 2.3. Ngược lại, tạo ra các trạng thái kế tiếp T_k có thể có từ trạng thái T_{max} . Đối với mỗi T_k thực hiện:
 - Tính $f(T_k)$; Thêm T_k vào OPEN($T_k \notin \{OPEN, CLOSE\}$)

Chương 4. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

- Giải thuật tìm kiếm tốt nhất đầu tiên (Best First Search)
- Trình bày thuật toán Best-First Search (BFS). Áp dụng BFS để tìm đường đi ngắn nhất từ đỉnh A đến G trên đồ thị với các ước lượng heuristic của các trạng thái so với trạng thái đích được liệt kê:

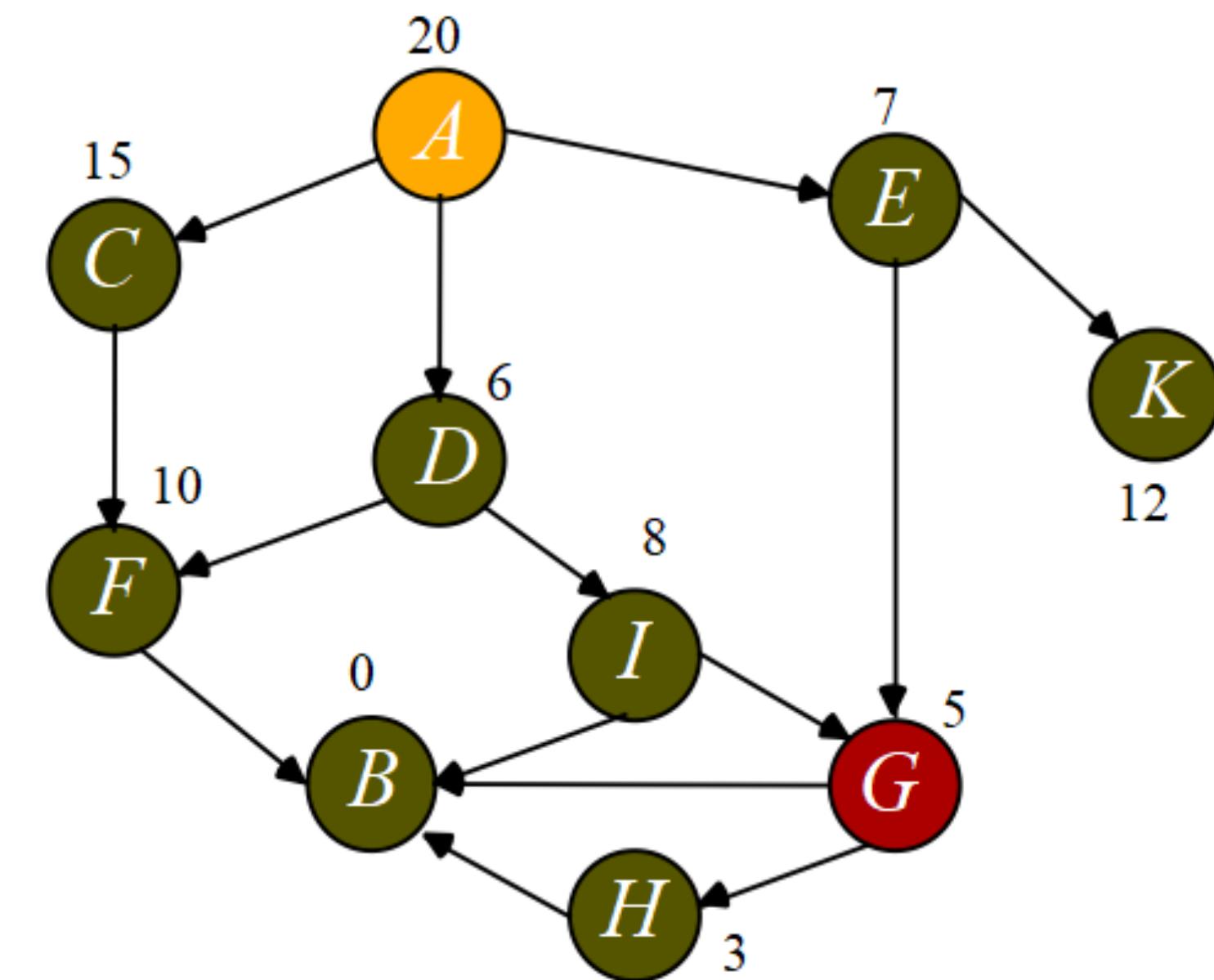
A	B	C	D	E	F	G	H	I	K
20	0	15	6	7	10	5	3	8	12



Chương 4. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

- Giải thuật tìm kiếm tốt nhất đầu tiên (Best First Search)

Bước	n	$\Gamma(n)$	Open	Close	$f_{SORT\uparrow}$
0			{A}	\emptyset	
1	A	{C,D,E}	{D,E,C}	{A}	$f(C, D, E)$
2	D	{F,I}	{E,I,F,C}	{A,D}	$f(C, E, F, I)$
3	E	{K,G}	{G,I,F,K,C}	{A,D,E}	$f(C, F, I, K, G)$
4	G	TRUE			

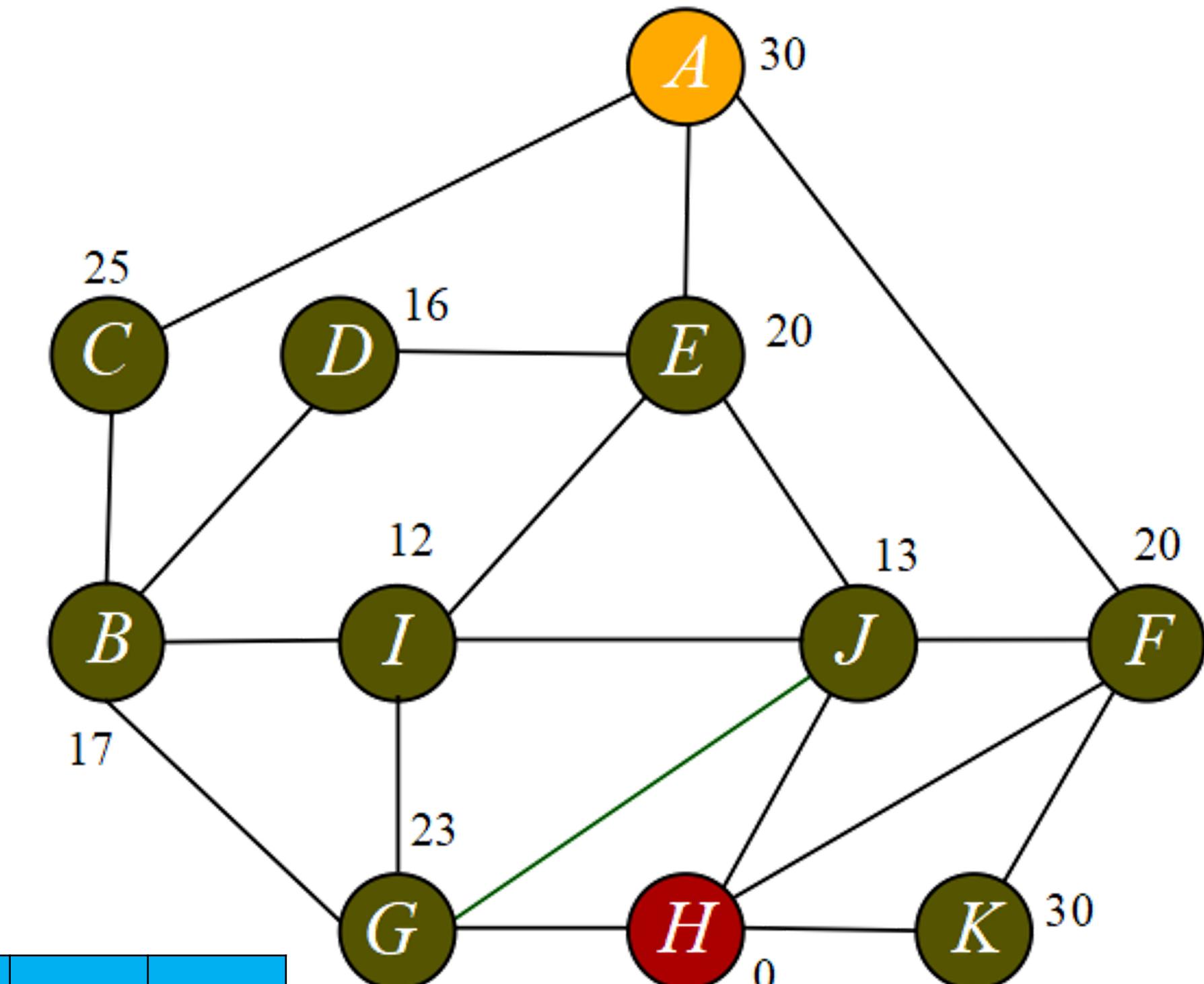


- $A \Rightarrow E \Rightarrow G$

Chương 4. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

- Giải thuật tìm kiếm tốt nhất đầu tiên (Best First Search)
- Trình bày thuật toán Best-First Search (BFS). Áp dụng BFS để tìm đường đi ngắn nhất từ đỉnh **A** đến **H** trên đồ thị với các ước lượng heuristic của các trạng thái so với trạng thái đích được liệt kê:

A	B	C	D	E	F	G	H	I	J	K
30	17	25	16	20	20	23	0	12	13	30



Chương 4. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

- Giải thuật AT (Algorithm for Tree)
- Thuật giải BFS khá đơn giản. Tuy vậy, trên thực tế cũng như tìm kiếm theo chiều rộng và chiều sâu, hiếm khi ta dùng BFS một cách trực tiếp. Thông thường, người ta thường dùng các phiên bản của BFS là AT, AKT và A*
- Thông thường, trong các phương án tìm kiếm BFS. Độ tốt f của một trạng thái được tính dựa vào 2 giá trị **g** và **h'**.
 - h' : ước lượng chi phí từ trạng thái hiện hành – trạng thái đích
 - g : chiều dài đoạn đường đã đi từ trạng thái ban đầu đến trạng thái hiện tại (g là chi phí thực sự chứ không phải ước lượng)

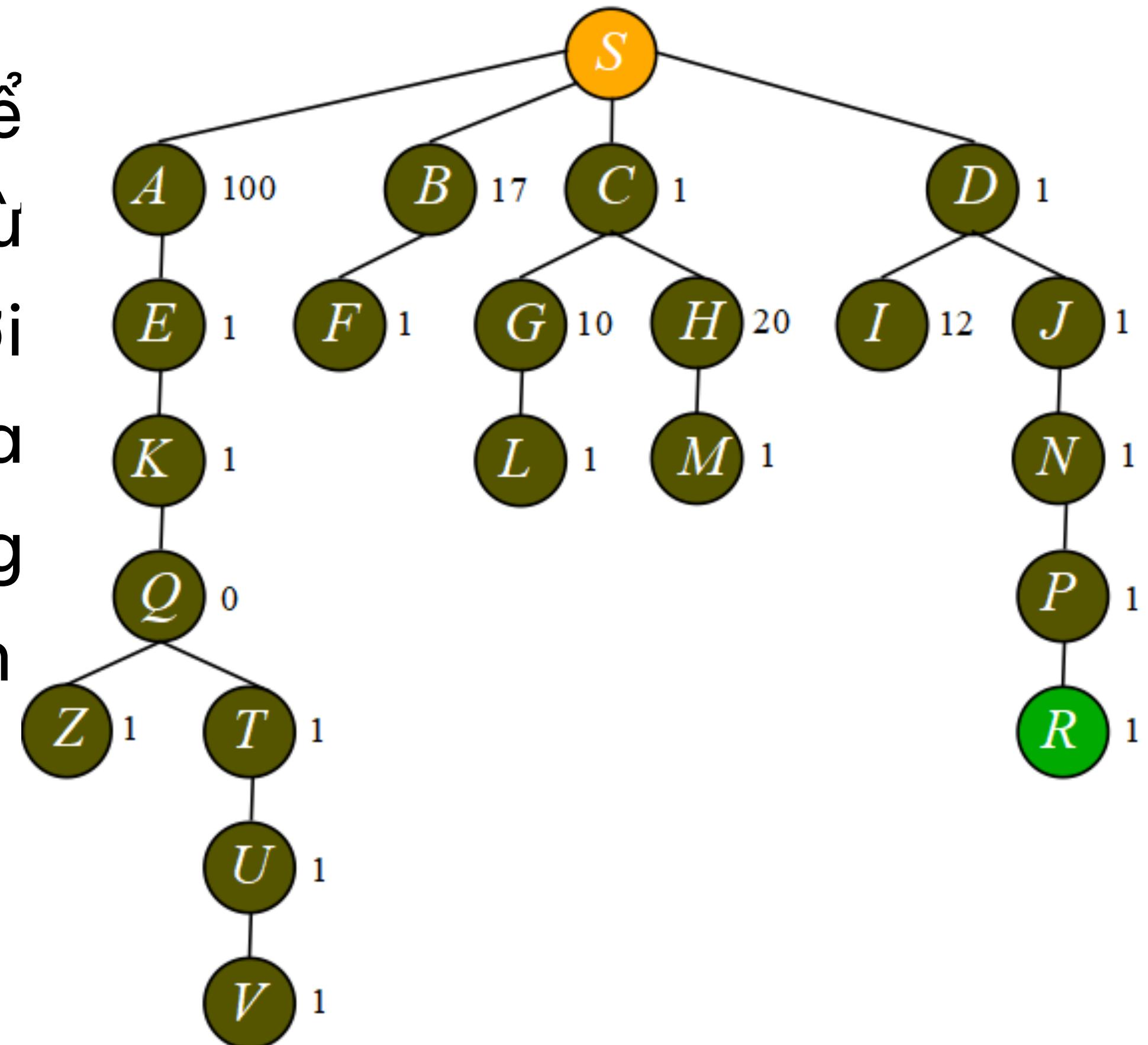
Chương 4. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

- Giải thuật AT (Algorithm for Tree)
 - Bước 1: Chọn $s = \text{đỉnh xuất phát}; g(s)=0$
 - Bước 2: Chọn 1 đỉnh từ tập OPEN: $=n$, có $g(n) = \min$
 - 2.1. Nếu $n \equiv \text{đích}$: đường đi từ $s \Rightarrow n$ là tối ưu
 - 2.2. Nếu $\text{OPEN} = \emptyset$: không tìm thấy đường đi
 - 2.3. Nếu $\exists \square$ nhiều hơn 1 đỉnh n có $g(n) = \min$:
 - Chọn ngẫu nhiên một đỉnh $\rightarrow n$
 - Bước 3: Đưa đỉnh đã duyệt vào CLOSE, tạo $\Gamma(n)$: các đỉnh mà n trỏ tới
 - for mỗi nút con m của $\Gamma(n)$:
 - *if* ($m \notin \text{OPEN} \& m \notin \text{CLOSE}$):
 - $g(m) := g(n) + \text{cost}(n, m); // \text{cost}(n, m) \equiv h(m)$
 - $\text{OPEN} := \text{OPEN} \cup \{m\}$
 - Bước 4: Quay lại bước 2

Chương 4. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

▪ Giải thuật AT (Algorithm for Tree)

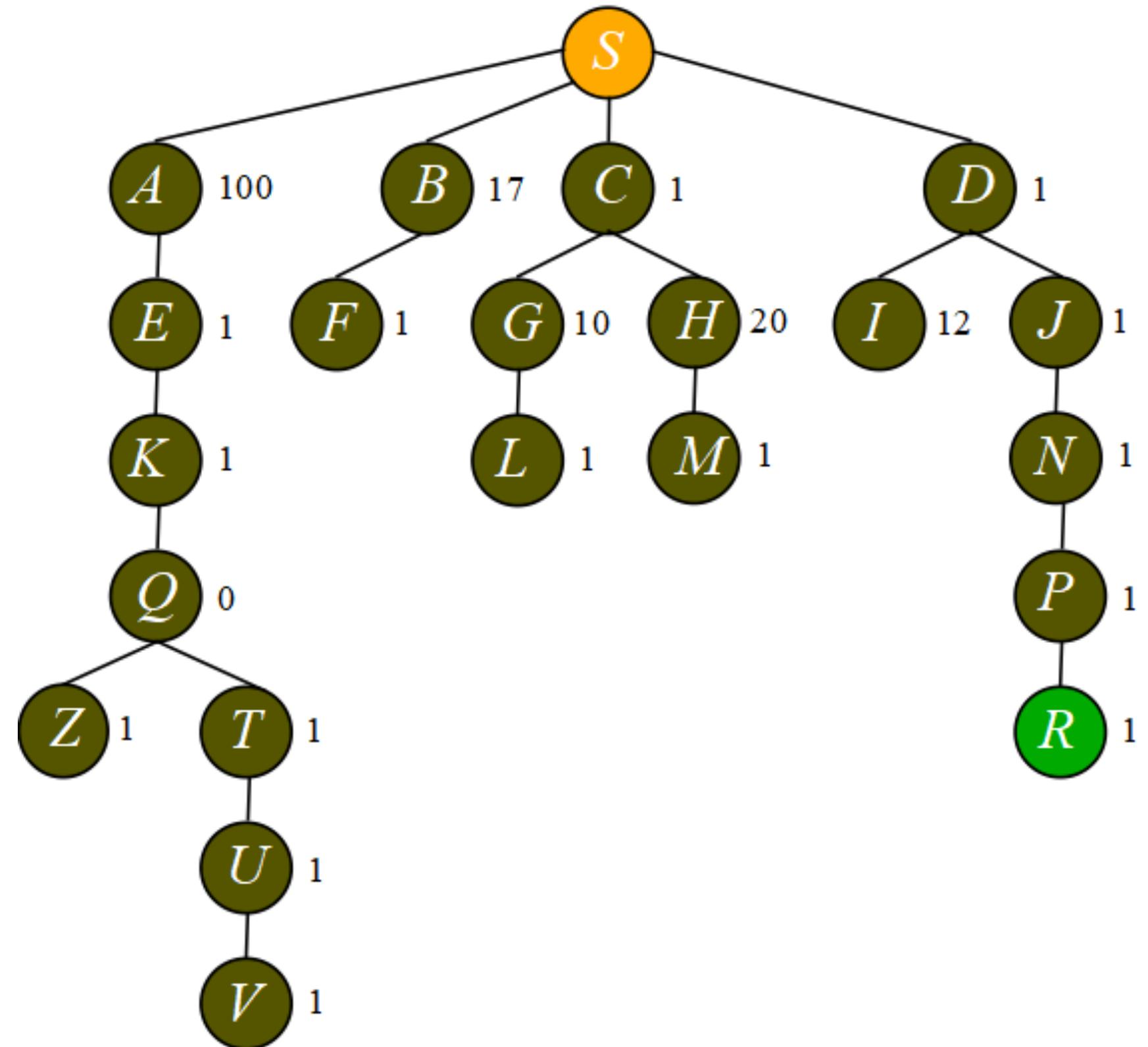
- Trình bày thuật toán AT để tìm đường đi ngắn nhất từ đỉnh **S** đến **R** trên đồ thị với các ước lượng heuristic của các trạng thái so với trạng thái đích được liệt kê như hình



Chương 4. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

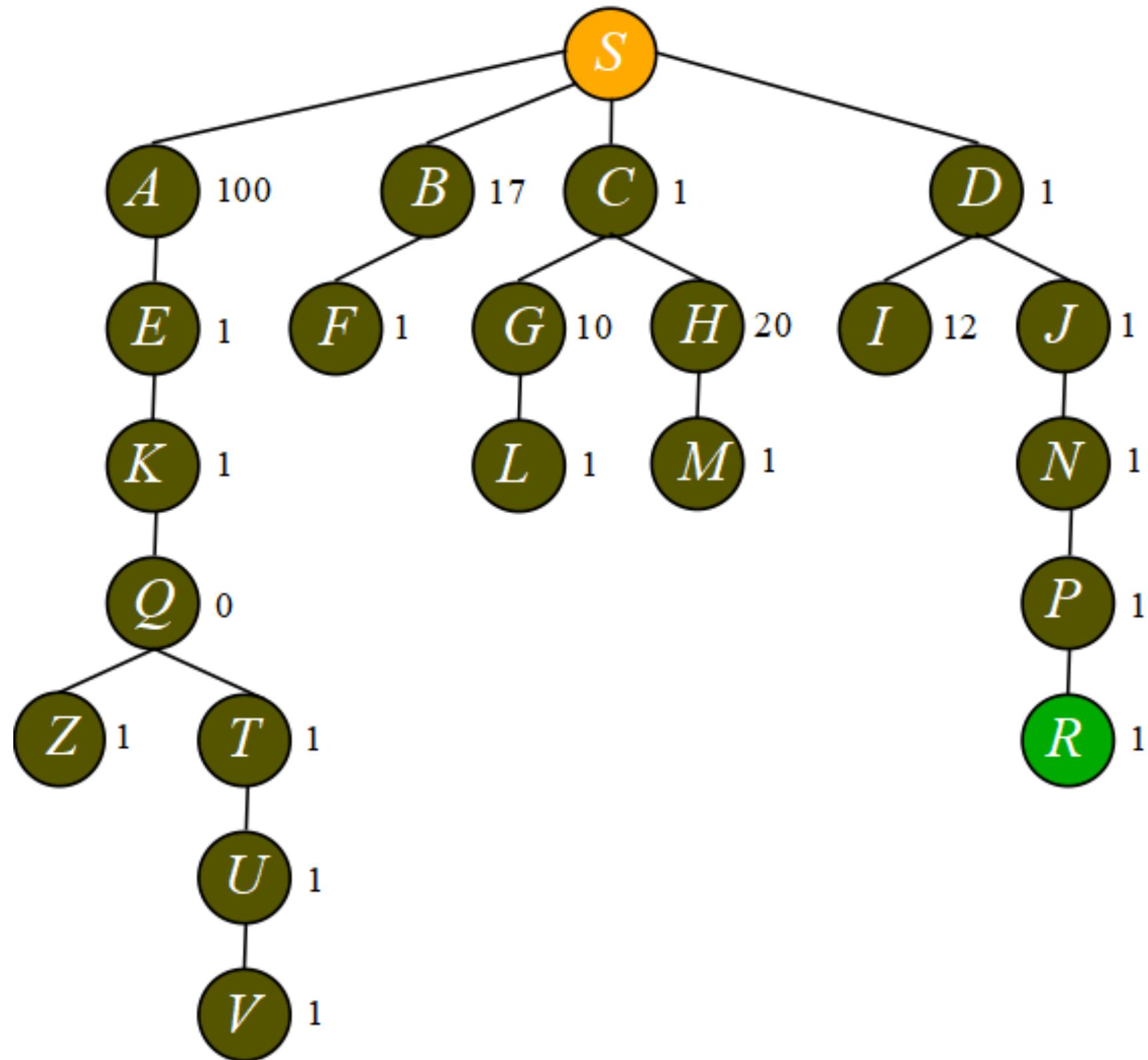
▪ Giải thuật AT (Algorithm for Tree)

- Bước 1: OPEN(S); $g(S)=0$
- Bước 2: $n=S$, $\Gamma(n)=\{A,B,C,D\}$, CLOSE={S}
 - $g(A \notin \text{OPEN})=g(S) + g(S \rightarrow A) // g(S)+g(A)$
 $= 0+100 = 100$
 - $g(B \notin \text{OPEN})=g(S) + g(S \rightarrow B) = 0+17 = 17$
 - $g(C \notin \text{OPEN})=g(S) + g(S \rightarrow C) = 0+1 = 1$
 - $g(D \notin \text{OPEN})=g(S) + g(S \rightarrow D) = 0+1 = 1$
 - $g(\min) = g(C)$
 - $\text{OPEN}=\{C,D,B,A\}$



Chương 4. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

▪ Giải thuật AT (Algorithm for Tree)

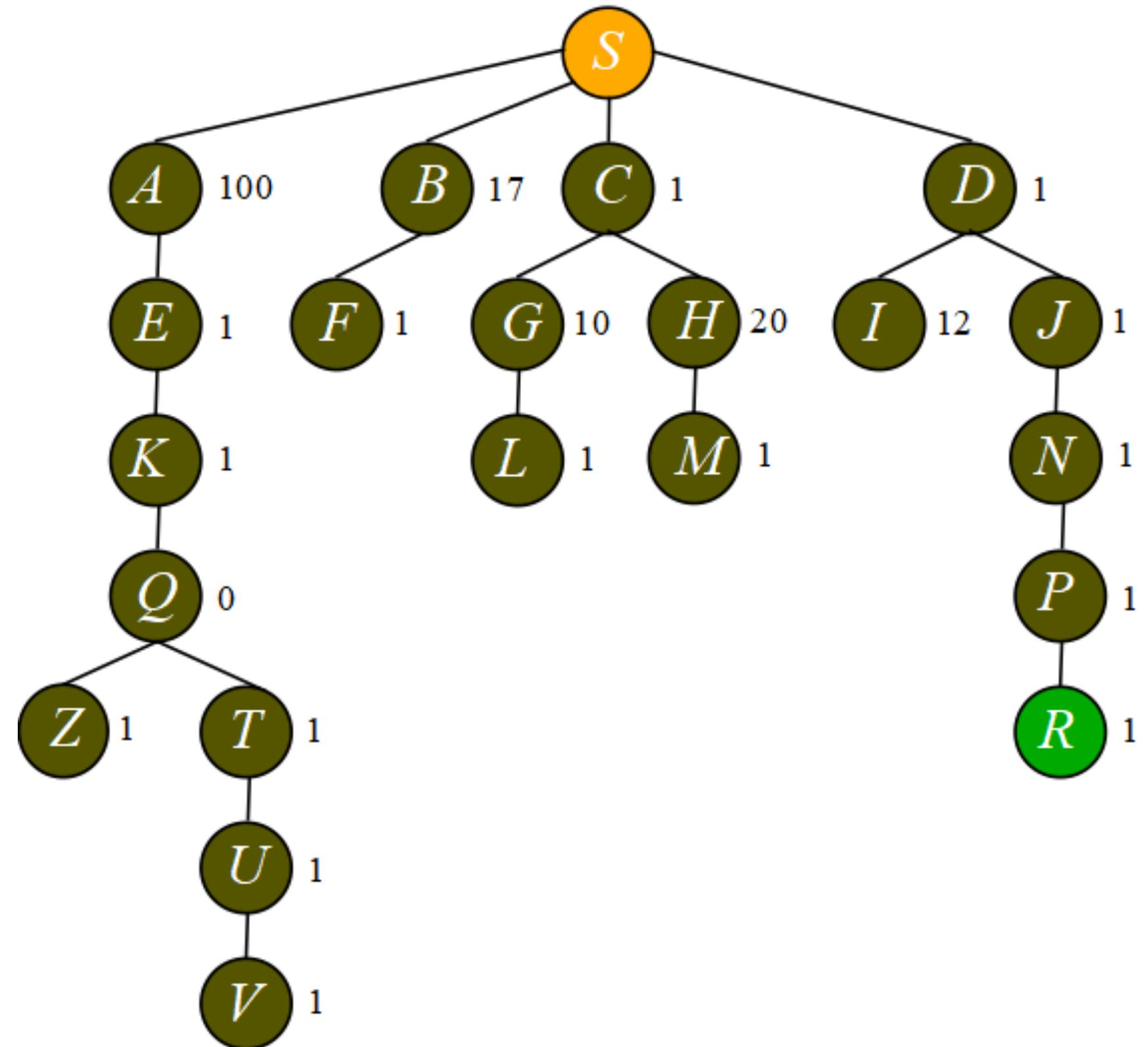


- Bước 3: $n=C$, $\Gamma(n)=\{G,H\}$, CLOSE= $\{S,C\}$
 - $g(A) = 100$
 - $g(B) = 17$
 - $g(D) = 1$
 - $g(G \notin \text{OPEN}) = g(C) + g(C \rightarrow G)$ $= 1+10 = 11$
 - $g(H \notin \text{OPEN}) = g(C) + g(C \rightarrow H)$ $= 1+20 = 21$
 - $g(\min) = g(D)$
 - $\text{OPEN}=\{D,G,B,H,A\}$

Chương 4. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

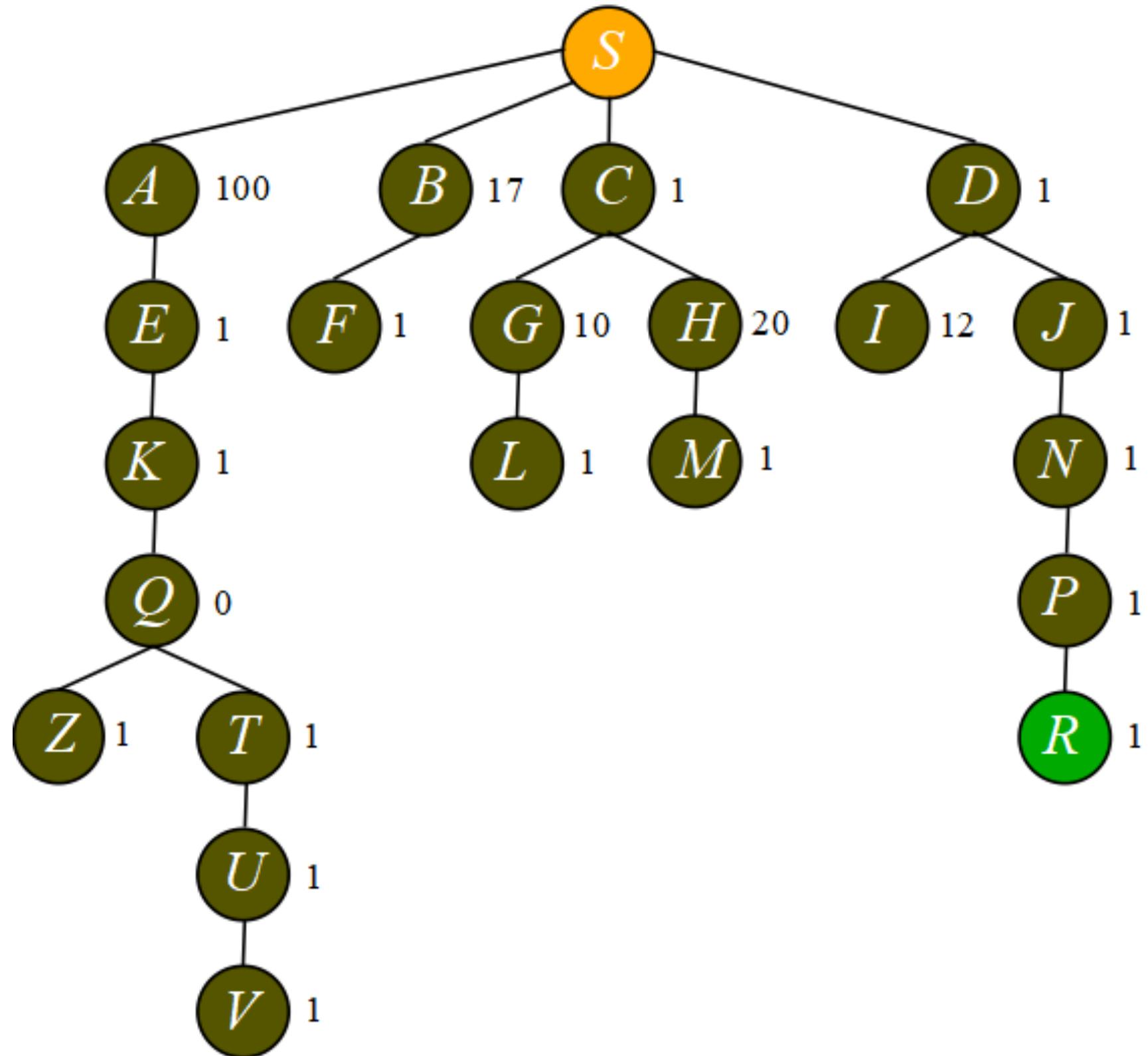
▪ Giải thuật AT (Algorithm for Tree)

- Bước 4: $n=D$, $\Gamma(n)=\{I, J\}$, CLOSE= $\{S, C, D\}$
 - $g(A)= 100$
 - $g(B)= 17$
 - $g(G)= 11$
 - $g(H)= 21$
 - $g(I \notin OPEN) = g(D) + g(D \rightarrow I) = 1 + 12 = 13$
 - $g(J \notin OPEN) = g(D) + g(D \rightarrow J) = 1 + 1 = 2$
 - $g(\min) = g(J)$
 - $OPEN=\{J, G, I, B, H, A\}$



Chương 4. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

▪ Giải thuật AT (Algorithm for Tree)



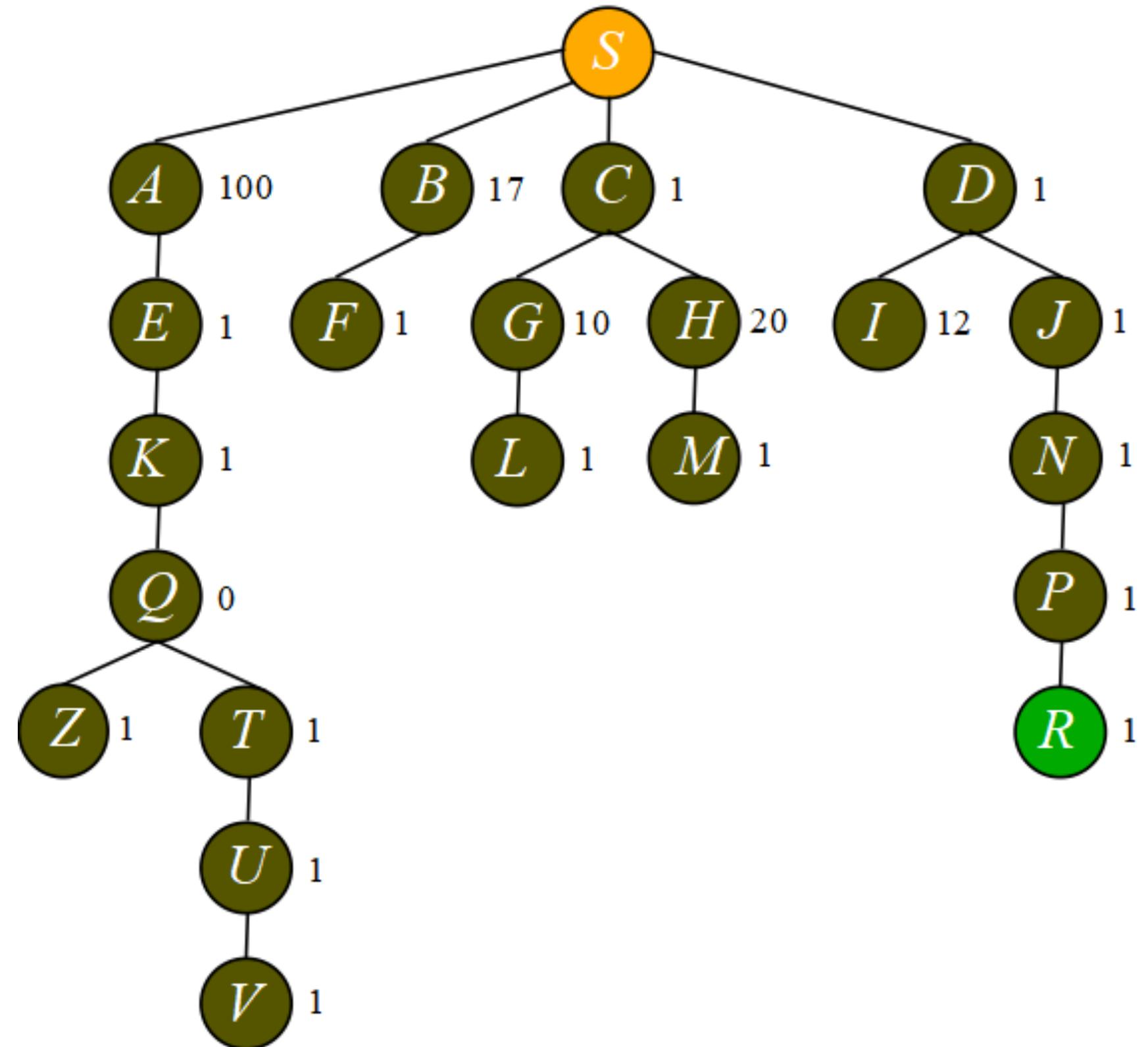
- Bước 5: $n=J$, $\Gamma(n)=\{N\}$, CLOSE= $\{S,C,D,J\}$
 - $g(A)=100$
 - $g(B)=17$
 - $g(G)=11$
 - $g(H)=21$
 - $g(I)=13$
 - $g(N \notin \text{OPEN})=g(J) + g(J \rightarrow N) = 2+1 = 3$
 - $g(\min) = g(N)$
 - $\text{OPEN}=\{N,G,I,B,H,A\}$

Chương 4. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

▪ Giải thuật AT (Algorithm for Tree)

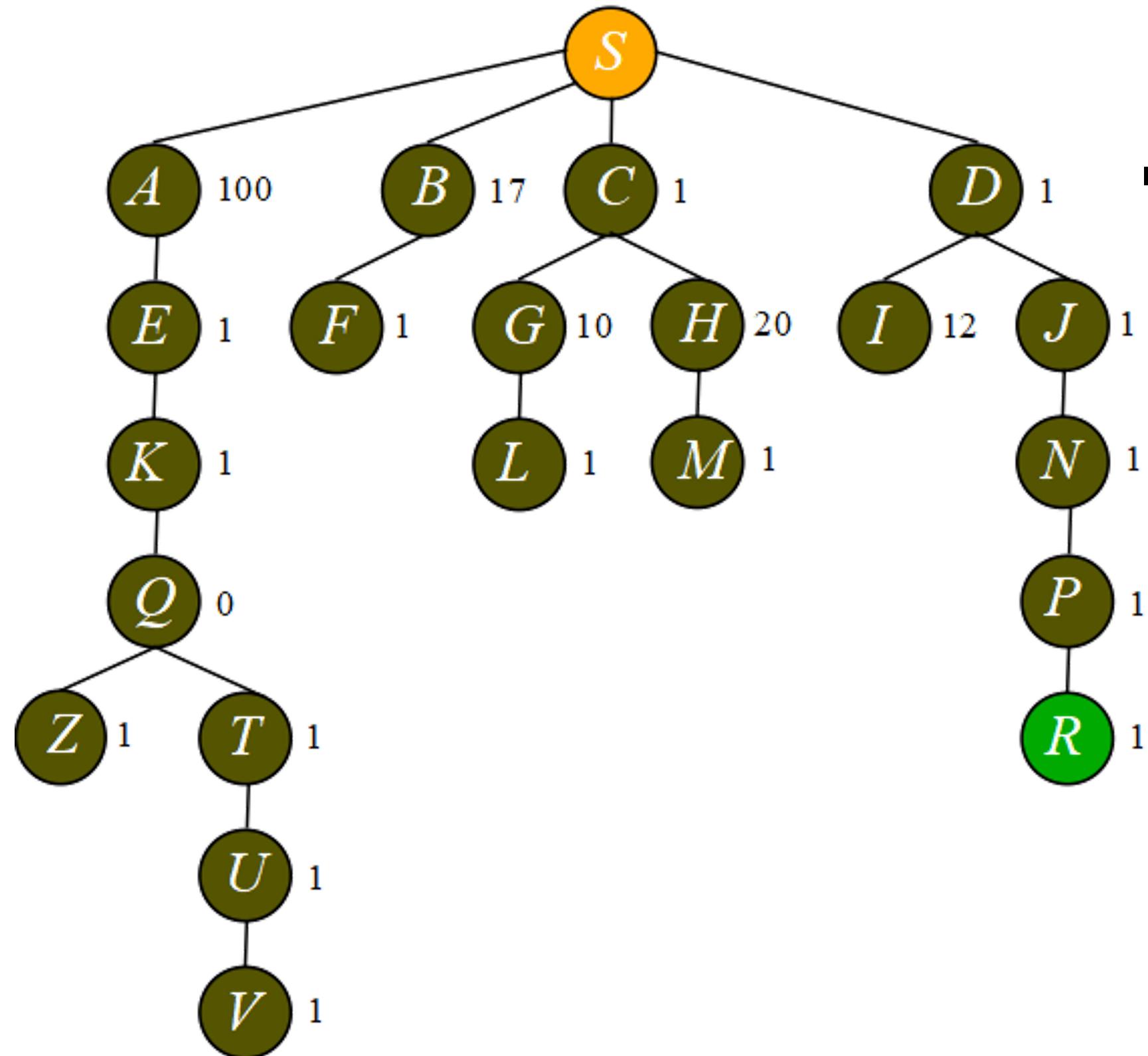
- Bước 6: $n=N$, $\Gamma(n)=\{P\}$, CLOSE= $\{S,C,D,J,N\}$

- $g(A)= 100$
- $g(B)= 17$
- $g(G)= 11$
- $g(H)= 21$
- $g(I)= 13$
- $g(P \notin OPEN) = g(N) + g(N \rightarrow P) = 3 + 1 = 4$
- $g(\min) = g(P)$
- $OPEN=\{P,G,I,B,H,A\}$



Chương 4. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

▪ Giải thuật AT (Algorithm for Tree)



- Bước 7: $n=P$, $\Gamma(n)=\{R\}$, CLOSE= $\{S,C,D,J,N,P\}$
 - $g(A)=100$
 - $g(B)=17$
 - $g(G)=11$
 - $g(H)=21$
 - $g(I)=13$
 - $g(R \notin \text{OPEN})=g(P) + g(P \rightarrow R) = 4+1 = 5$
 - $g(\min) = g(R)$
 - $\text{OPEN}=\{R,G,I,B,H,A\}$
- Bước 8: $n=R$ TRUE

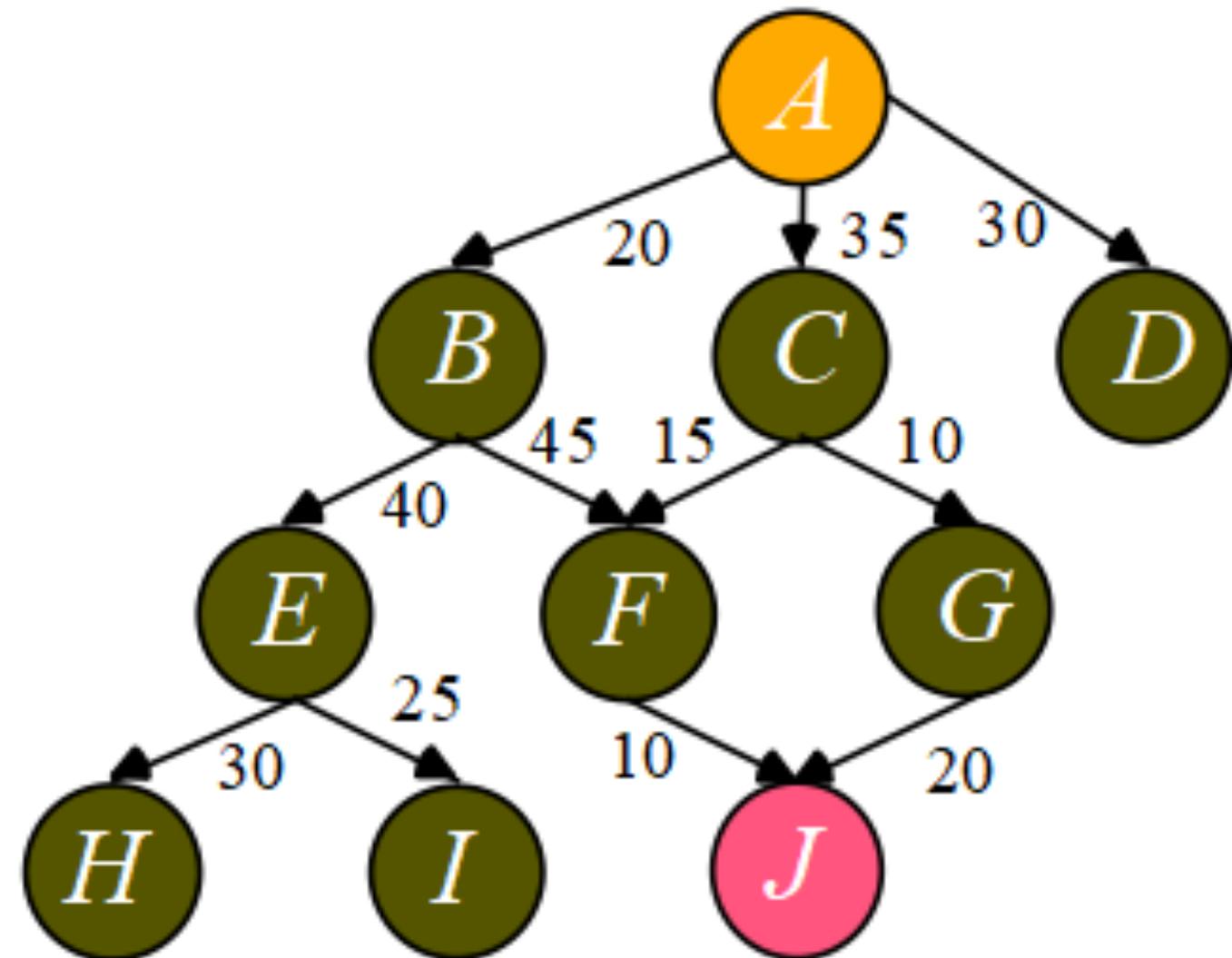
Chương 4. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

- Giải thuật Cost Minimization Search (CMS)
- Bước 1: Chọn $s = \text{đỉnh xuất phát}; g(s)=0$
- Bước 2: Chọn 1 đỉnh từ tập OPEN: $=n$, có $g(n) = \min$
 - 2.1. Nếu $n \equiv \text{đích}$: đường đi từ $s \Rightarrow n$ là tối ưu
 - 2.2. Nếu $\text{OPEN} = \emptyset$: không tìm thấy đường đi
 - 2.3. Nếu $\exists \square$ nhiều hơn 1 đỉnh n có $g(n) = \min$:
 - Kiểm tra ($\exists \square \equiv \text{đích}$): dừng
 - Ngược lại, chọn ngẫu nhiên một đỉnh: $=n$
- Bước 3: Đưa đỉnh đã duyệt vào CLOSE, tạo $\Gamma(n)$: các đỉnh mà n trỏ tới
 - for mỗi nút con m của $\Gamma(n)$:
 - $g(m) := g(n) + cost(n, m)$
 - $OPEN := OPEN \cup \{m\}$
 - So sánh $g(m)$ với $g_{\text{NEW}}(m)$ và cập nhật
- Bước 4: Quay lại bước 2

Chương 4. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

- Giải thuật Cost Minimization Search (CMS)

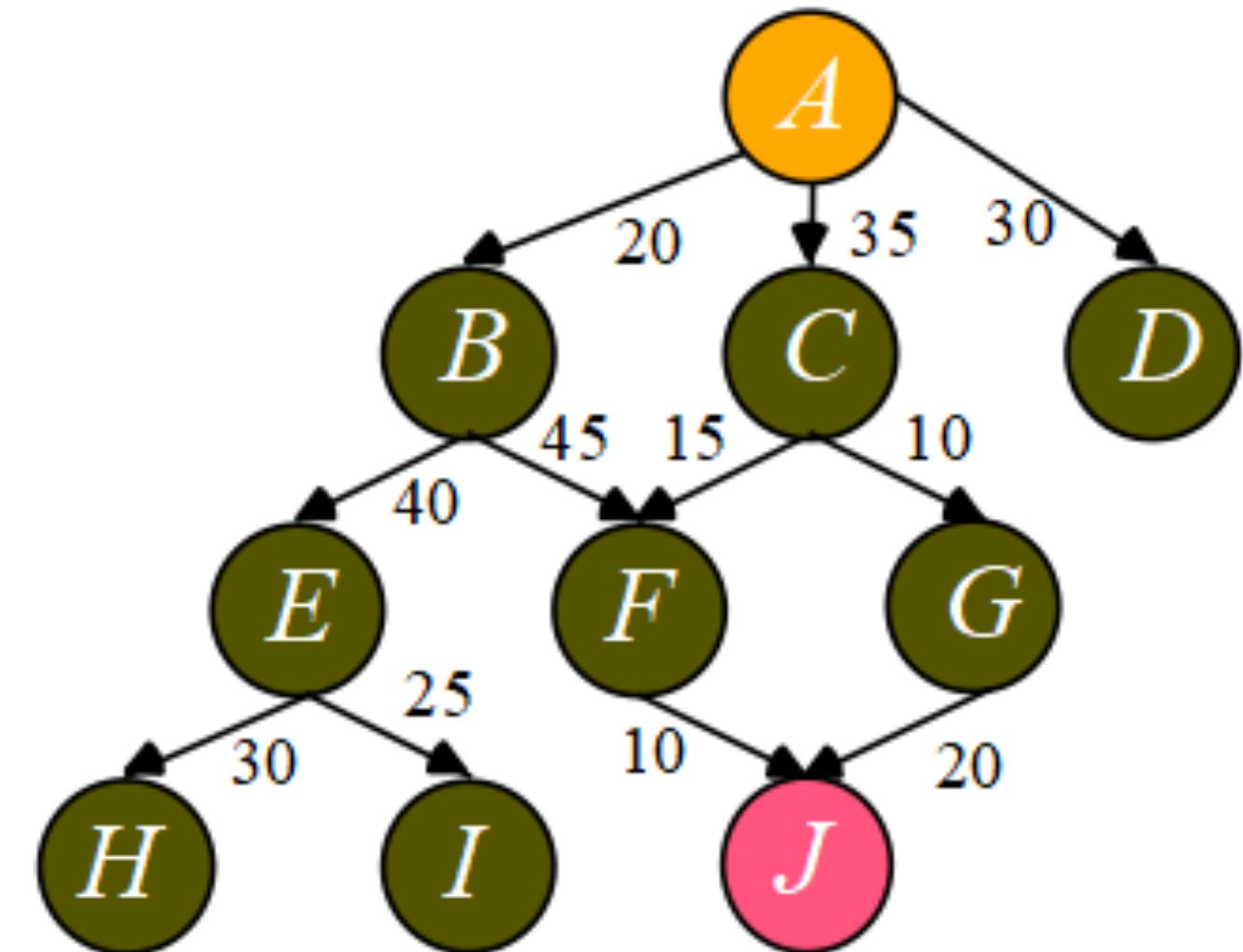
- Trình bày thuật toán CMS để tìm đường đi ngắn nhất từ đỉnh A đến J trên đồ thị với các ước lượng heuristic của các trạng thái so với trạng thái đích được liệt kê như hình



Chương 4. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

▪ Giải thuật Cost Minimization Search (CMS)

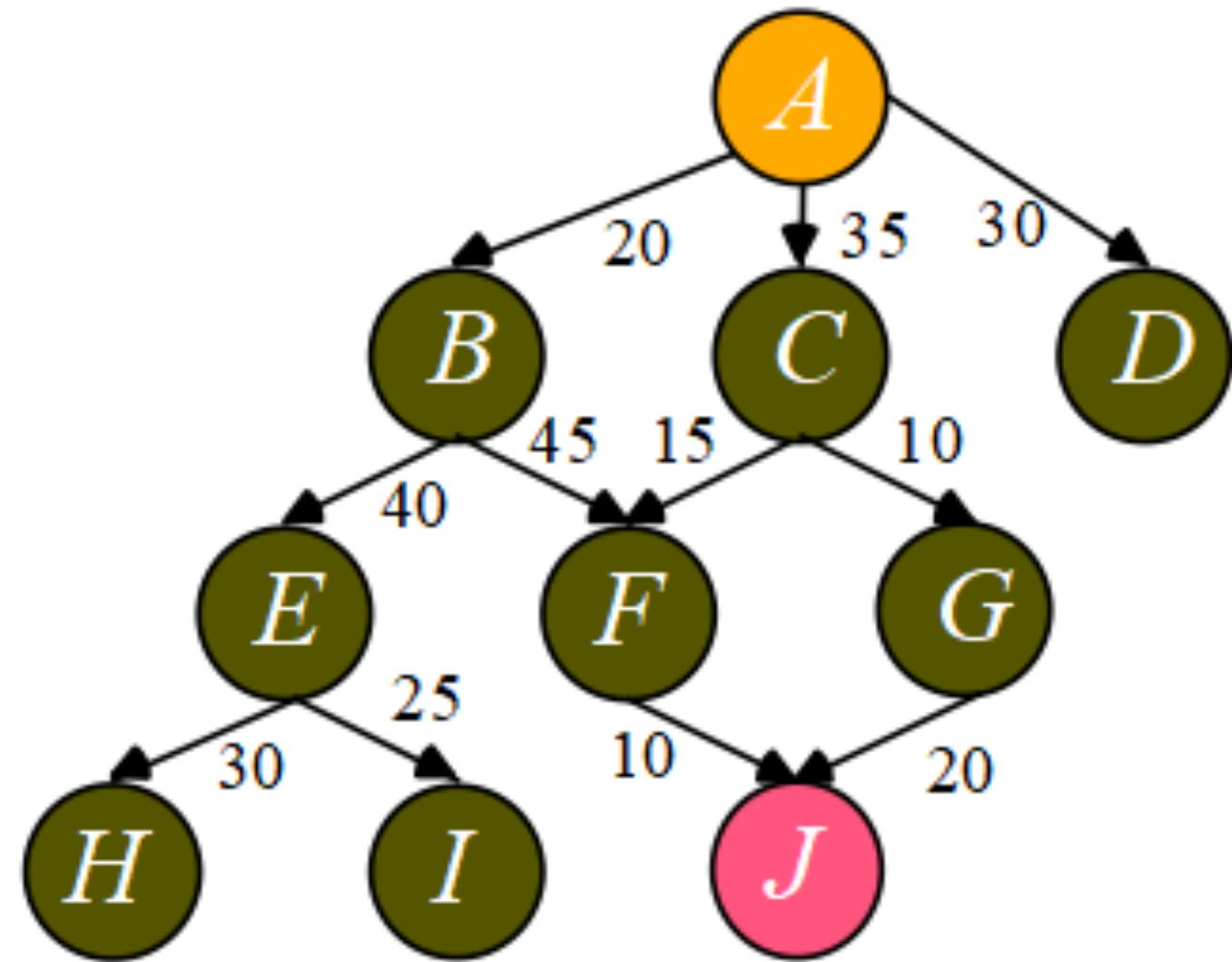
- Bước 1: OPEN(A); $g(A)=0$
- Bước 2: $n=A$, $\Gamma(n)=\{B,C,D\}$, CLOSE={A}
 - $g(B) = g(A) + g(A \rightarrow B) = 0+20 = 20$; Father(B) = A
 - $g(C) = g(A) + g(A \rightarrow C) = 0+35= 35$; Father(C) = A
 - $g(D) = g(A) + g(A \rightarrow D) = 0+30 = 30$; Father(D) = A
 - $g(\min) = g(B)$
 - OPEN={B,D,C}



Chương 4. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

▪ Giải thuật Cost Minimization Search (CMS)

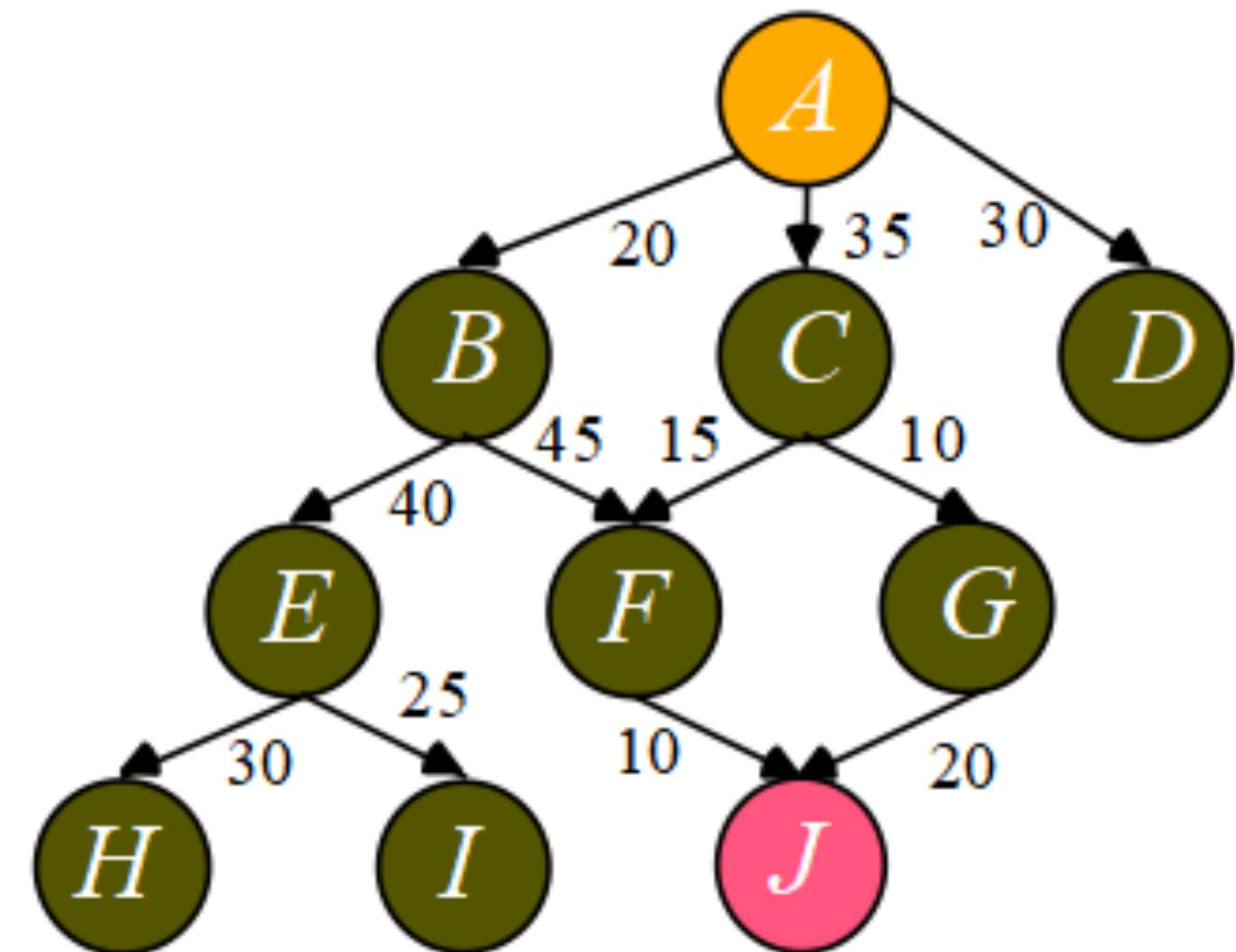
- Bước 3: $n=B$, $\Gamma(n)=\{E,F\}$, CLOSE= $\{A,B\}$
 - $g(C) = 35$
 - $g(D) = 30$
 - $g(E) = g(B) + g(B \rightarrow E) = 20 + 40 = 60$; Father(E) = B
 - $g(F) = g(B) + g(B \rightarrow F) = 20 + 45 = 65$; Father(F) = B
 - $g(\min) = g(D)$
 - OPEN= $\{D,C,E,F\}$



Chương 4. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

▪ Giải thuật Cost Minimization Search (CMS)

- Bước 4: $n=D$, $\Gamma(n)=\emptyset$, CLOSE={A,B,D}
 - $g(C) = 35$
 - $g(E) = 60$
 - $g(F) = 65$
 - OPEN={C,E,F}
- Bước 5: $n=C$, $\Gamma(n)=\{F,G\}$, CLOSE={A,B,D,C}
 - $g(E) = 60$
 - $g(F) = 65$
 - $g(F) = g(C) + g(C \rightarrow F) = 35 + 10 = 50$
 - $g(F) = \min(50, 65) = 50$; Father(F) = C
 - $g(G) = g(C) + g(C \rightarrow G) = 35 + 10 = 45$; Father(G) = C
 - $g(\min) = g(G)$
 - OPEN={G,F,E}



Chương 4. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

▪ Giải thuật Cost Minimization Search (CMS)

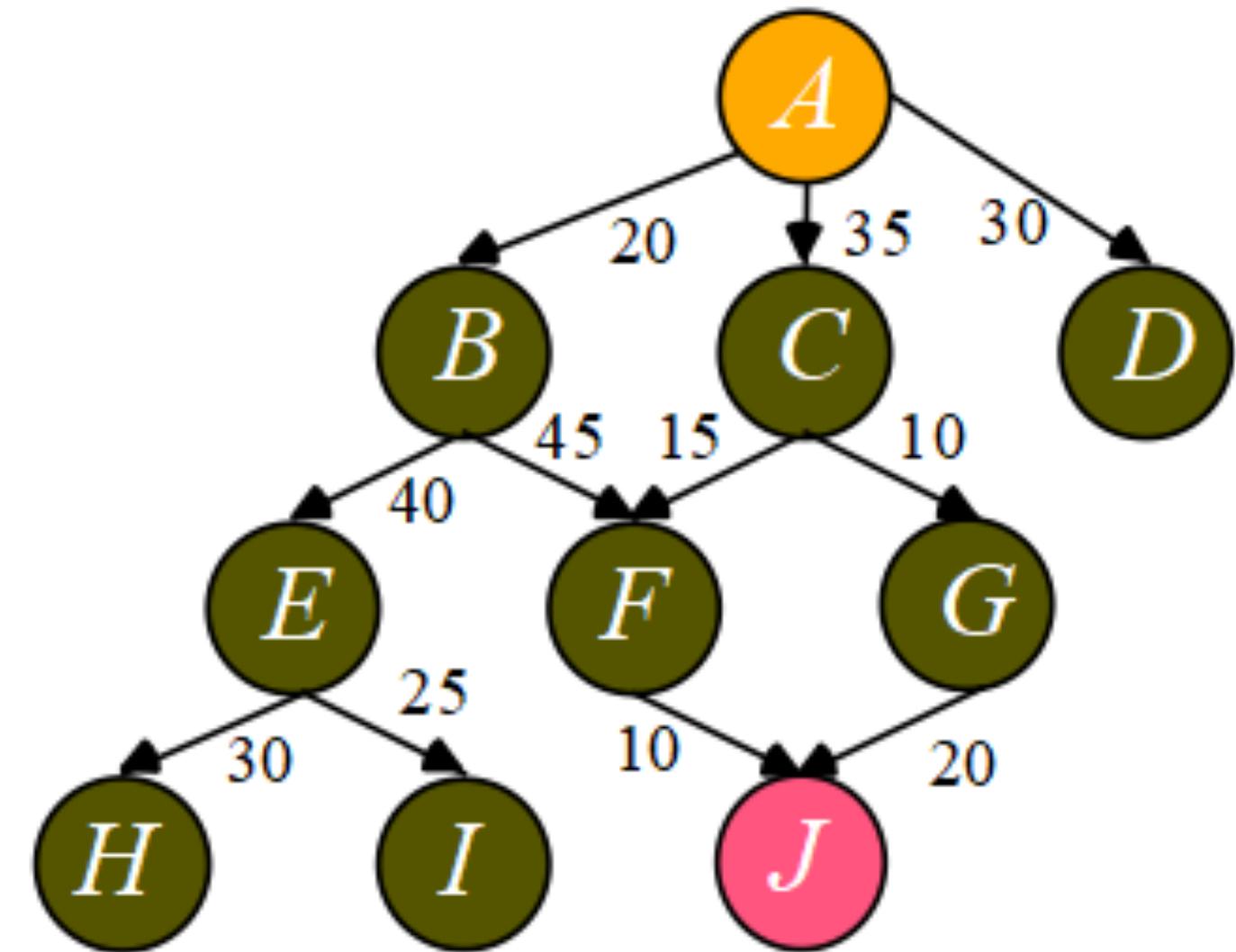
- Bước 6: $n=G$, $\Gamma(n)=\{J\}$, CLOSE={A,B,D,C,G}

- $g(E) = 60$
 - $g(F) = 65$
 - $g(J) = g(G) + g(G \rightarrow J) = 45 + 20 = 65$; Father(J) = G
 - OPEN={F,E,J}

- Bước 7: $n=F$, $\Gamma(n)=\{J\}$, CLOSE={A,B,D,C,G,F}

- $g(E) = 60$
 - $g(J) = 65$
 - $g(J) = g(G) + g(G \rightarrow J) = 45 + 20 = 65$
 - $g(J) = \min(65, 65) = 65$; Father(J) = F
 - OPEN={E,J}

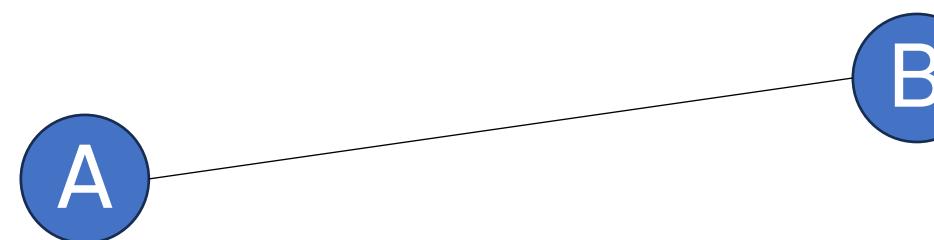
- Bước 8: $n=J$, TRUE



▪ $A \Rightarrow C \Rightarrow F \Rightarrow J$

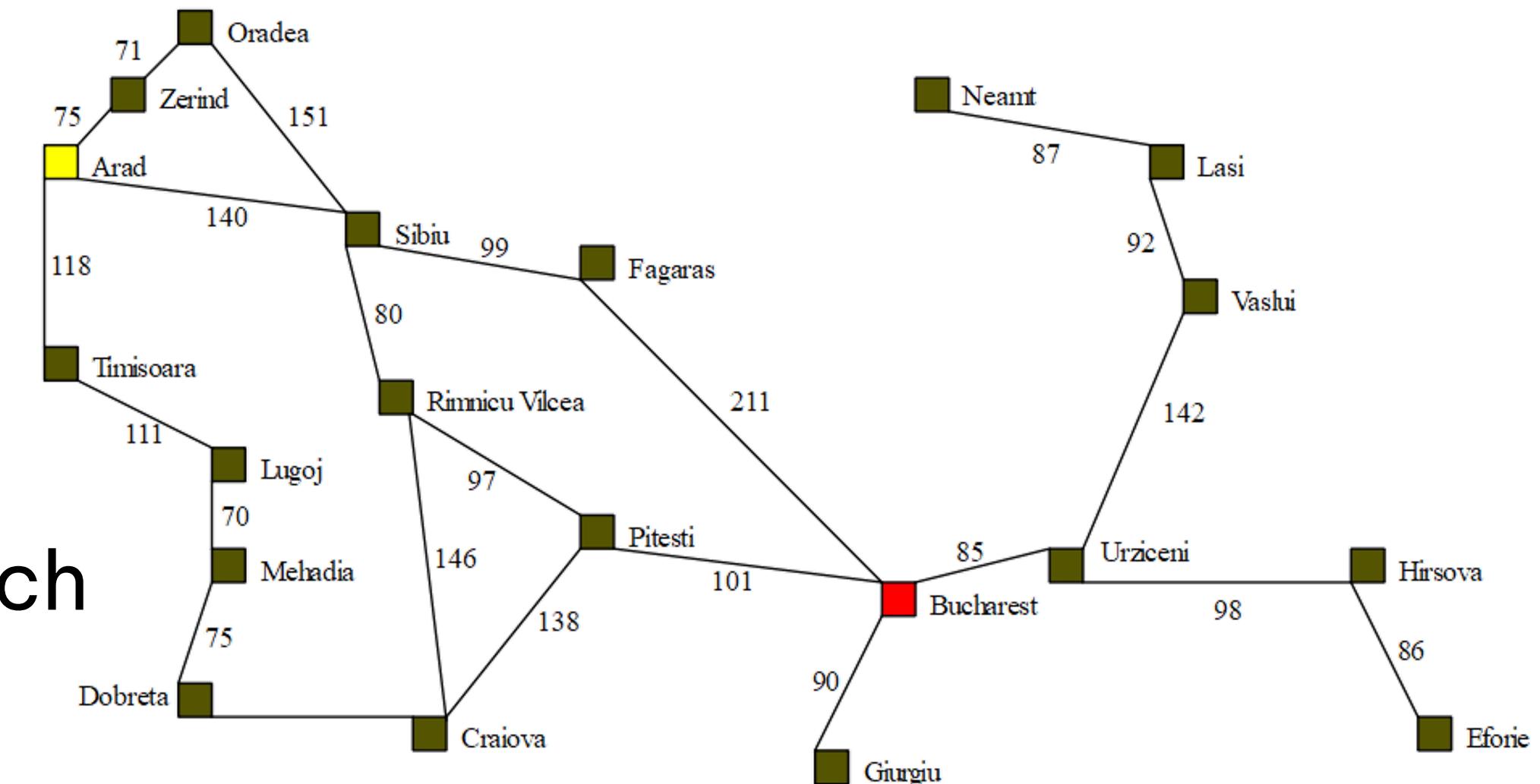
Chương 4. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

- Algorithm for Knowledgeable Tree Search (AKT)
- Thuật giải AKT là mở rộng của giải thuật A^T bằng cách sử dụng thêm thông tin ước lượng h' (khoảng cách Manhattan).
- Độ tốt của hàm $f = g + h'$



$$h' = |X_A - X_B| + |Y_A - Y_B|$$

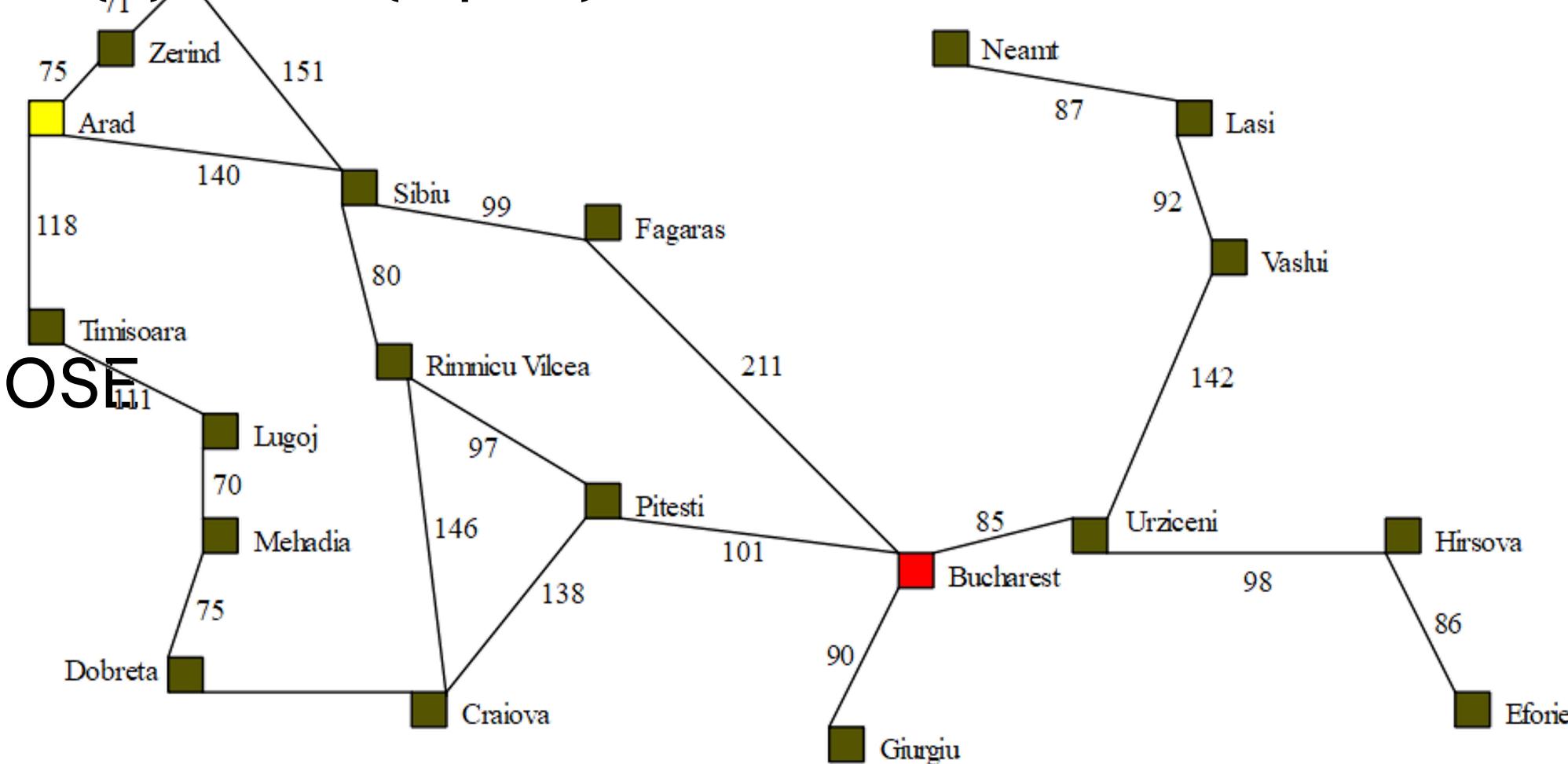
- h' còn được gọi là khoảng cách Euclid



Chương 4. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

▪ Algorithm for Knowledgeable Tree Search (AKT)

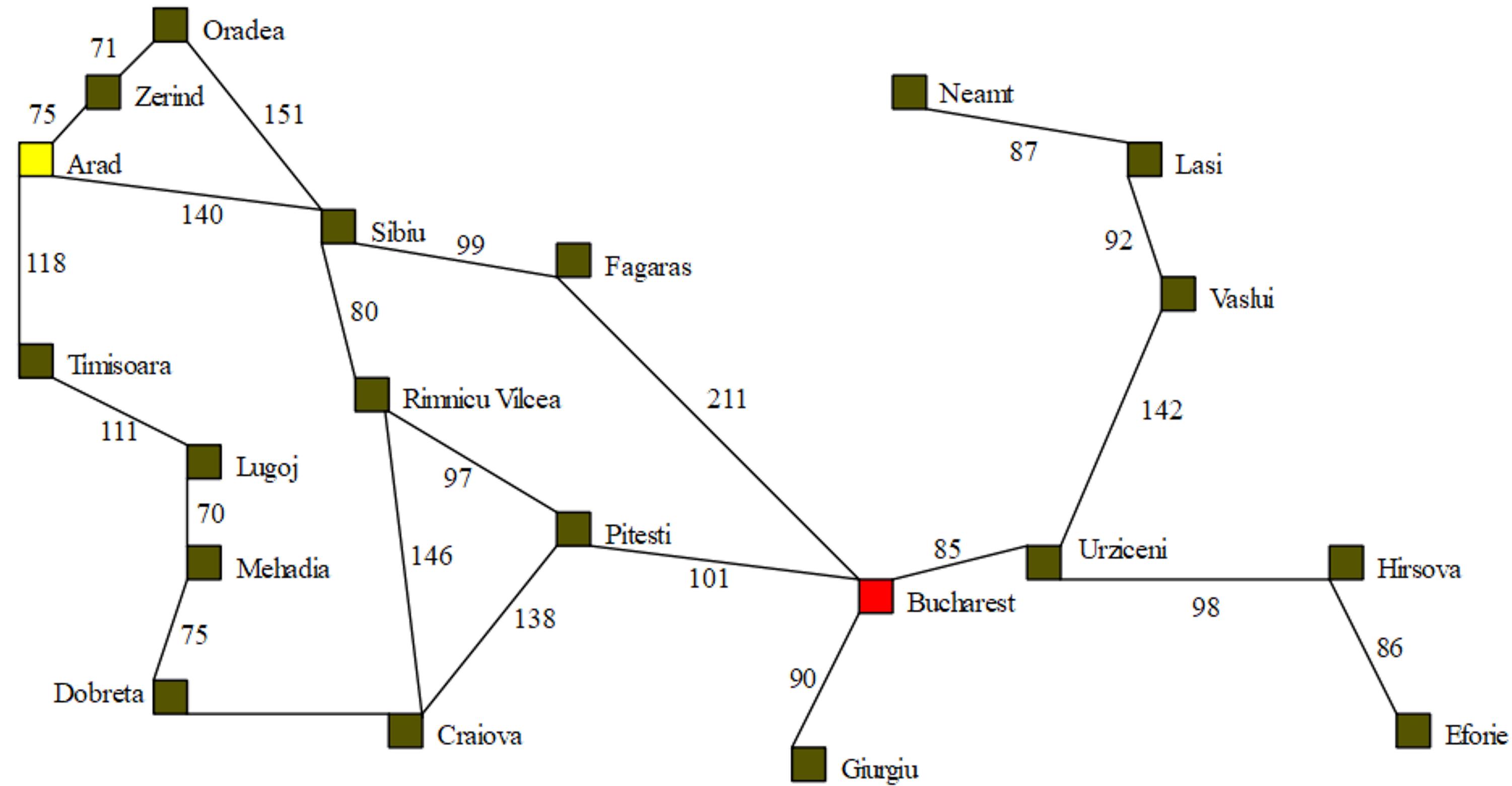
- Open:= {Start};
 - While (Open $\neq \emptyset$)
 - n= Retrieve(Open); // Chọn n sao cho $f(n) = \min(\text{Open})$
 - If (n = Goal): Return True
 - Else
 - Tạo $\Gamma(n)$;
 - for mỗi nút con m của $\Gamma(n)$ $\notin \text{CLOSE}$
 - $g(m) = g(n) + \text{Cost}(n,m)$
 - $f(m) = g(m) + h'(m)$
 - OPEN = OPEN $\cup \{m\}$
 - CLOSE = CLOSE $\cup \{n\}$
 - Return False;



Chương 4. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

- Algorithm for Knowledgeable Tree Search (AKT)
- Tìm đường đi ngắn nhất từ Arad → Bucharest:

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



Chương 4. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

▪ Algorithm for Knowledgeable Tree Search (AKT)

▪ Tìm đường đi ngắn nhất từ Arad → Bucharest:

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

- Start = Arad, $\Gamma(n) = \{\text{Timisoara, Sibiu, Zerind}\}$,
- CLOSE = \emptyset
 - $g(\text{Timisoara}) = g(\text{Arad}) + \text{cost}(\text{Arad} \rightarrow \text{Timisoara}) = 0 + 118 = 118$
 - $f(\text{Timisoara}) = g(\text{Timisoara}) + h'(\text{Timisoara}) = 118 + 329 = 447$
 - $g(\text{Sibiu}) = g(\text{Arad}) + \text{cost}(\text{Arad} \rightarrow \text{Sibiu}) = 0 + 140 = 140$
 - $f(\text{Sibiu}) = g(\text{Sibiu}) + h'(\text{Sibiu}) = 140 + 253 = 393$
 - $g(\text{Zerind}) = g(\text{Arad}) + \text{cost}(\text{Arad} \rightarrow \text{Zerind}) = 0 + 75 = 75$
 - $f(\text{Zerind}) = g(\text{Zerind}) + h'(\text{Zerind}) = 75 + 374 = 449$
- OPEN= {Sibiu – 393, Timisoara – 447, Zerind-449}
 - $f_{\min} = f(\text{Sibiu}),$
 - Father(Sibiu) = Arad
 - Father(Timisoara) = Arad
 - Father(Zerind) = Arad

Chương 4. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

▪ Algorithm for Knowledgeable Tree Search (AKT)

▪ Tìm đường đi ngắn nhất từ Arad → Bucharest:

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

- $n = \text{Sibiu}$, $\Gamma(n) = \{\text{Arad}, \text{Fagaras}, \text{Oradea}, \text{Rimnicu Vilcea}\}$,
- CLOSE = {Arad, Sibiu}
 - $g(\text{Fagaras}) = g(\text{Sibiu}) + \text{cost}(\text{Sibiu} \rightarrow \text{Fagaras}) = 140 + 99 = 239$
 - $f(\text{Fagaras}) = g(\text{Fagaras}) + h'(\text{Fagaras}) = 239 + 178 = 417$
 - $g(\text{Oradea}) = g(\text{Sibiu}) + \text{cost}(\text{Sibiu} \rightarrow \text{Oradea}) = 140 + 151 = 291$
 - $f(\text{Oradea}) = g(\text{Oradea}) + h'(\text{Oradea}) = 291 + 380 = 671$
 - $g(\text{Rimnicu Vilcea}) = g(\text{Sibiu}) + \text{cost}(\text{Sibiu} \rightarrow \text{Rim.Vilcea}) = 140 + 80 = 220$
 - $f(\text{Rimnicu Vilcea}) = g(\text{Fagaras}) + h'(\text{Rimnicu Vilcea}) = 220 + 193 = 413$
- OPEN= {Rimnicu Vilcea - 413, Fagaras - 417, Timisoara - 447, Zerind-449}
 - $f_{\min} = f(\text{Rimnicu Vilcea})$,
 - Father(Fagaras) = Sibiu
 - Father(Oradea) = Sibiu
 - Father(Rimnicu Vilcea) = Sibiu

Chương 4. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

- Algorithm for Knowledgeable Tree Search (AKT)
- Tìm đường đi ngắn nhất từ Arad → Bucharest:

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

- $n = \text{Rimnicu Vilcea}$, $\Gamma(n) = \{\text{Sibiu, Craiova, Pitesti}\}$,
- CLOSE = {Arad, Sibiu, Rimnicu Vilcea}
 - $g(\text{Craiova}) = g(\text{R.Vilcea}) + \text{cost}(\text{R.Vilcea} \rightarrow \text{Craiova}) = 220 + 146 = 366$
 - $f(\text{Craiova}) = g(\text{Craiova}) + h'(\text{Craiova}) = 366 + 160 = 526$
 - $g(\text{Pitesti}) = g(\text{R.Vilcea}) + \text{cost}(\text{R.Vilcea} \rightarrow \text{Pitesti}) = 220 + 97 = 317$
 - $f(\text{Pitesti}) = g(\text{Pitesti}) + h'(\text{Pitesti}) = 317 + 98 = 415$
- OPEN= {Pitesti - 415, Fagaras - 417, Timisoara - 447, Zerind-449, Craiova - 526}
 - $f_{\min} = f(\text{Pitesti})$,
 - Father(Craiova) = Rimnicu Vilcea
 - Father(Pitesti) = Rimnicu Vilcea

Chương 4. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

▪ Algorithm for Knowledgeable Tree Search (AKT)

▪ Tìm đường đi ngắn nhất từ Arad → Bucharest:

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

- $n = \text{Pitesti}$, $\Gamma(n) = \{\text{Rimnicu Vilcea}, \text{Bucharest}, \text{Craiova}\}$,
- CLOSE = {Arad, Sibiu, Rimnicu Vilcea, Pitesti}
 - $g(\text{Bucharest}) = g(\text{Pitesti}) + \text{cost}(\text{Pitesti} \rightarrow \text{Bucharest}) = 317 + 101 = 418$
 - $f(\text{Bucharest}) = g(\text{Bucharest}) + h'(\text{Bucharest}) = 418 + 0 = 418$
 - $g_{\text{new}}(\text{Craiova}) = g(\text{Pitesti}) + \text{cost}(\text{Pitesti} \rightarrow \text{Craiova}) = 317 + 138 = 455$
 - $f_{\text{new}}(\text{Craiova}) = g(\text{Craiova}) + h'(\text{Craiova}) = 455 + 160 = 615$
- OPEN= {Fagaras - 417, Bucharest - 418, Timisoara - 447, Zerind-449, **Craiova - 526 < f_{new} 615**}
 - $f_{\text{min}} = f(\text{Fagaras})$,
 - **Father(Fagaras) = Sibiu (từ slide 28)**
 - **Father(Craiova) = Rimnicu Vilcea (vẫn giữ nguyên)**

Chương 4. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

▪ Algorithm for Knowledgeable Tree Search (AKT)

▪ Tìm đường đi ngắn nhất từ Arad → Bucharest:

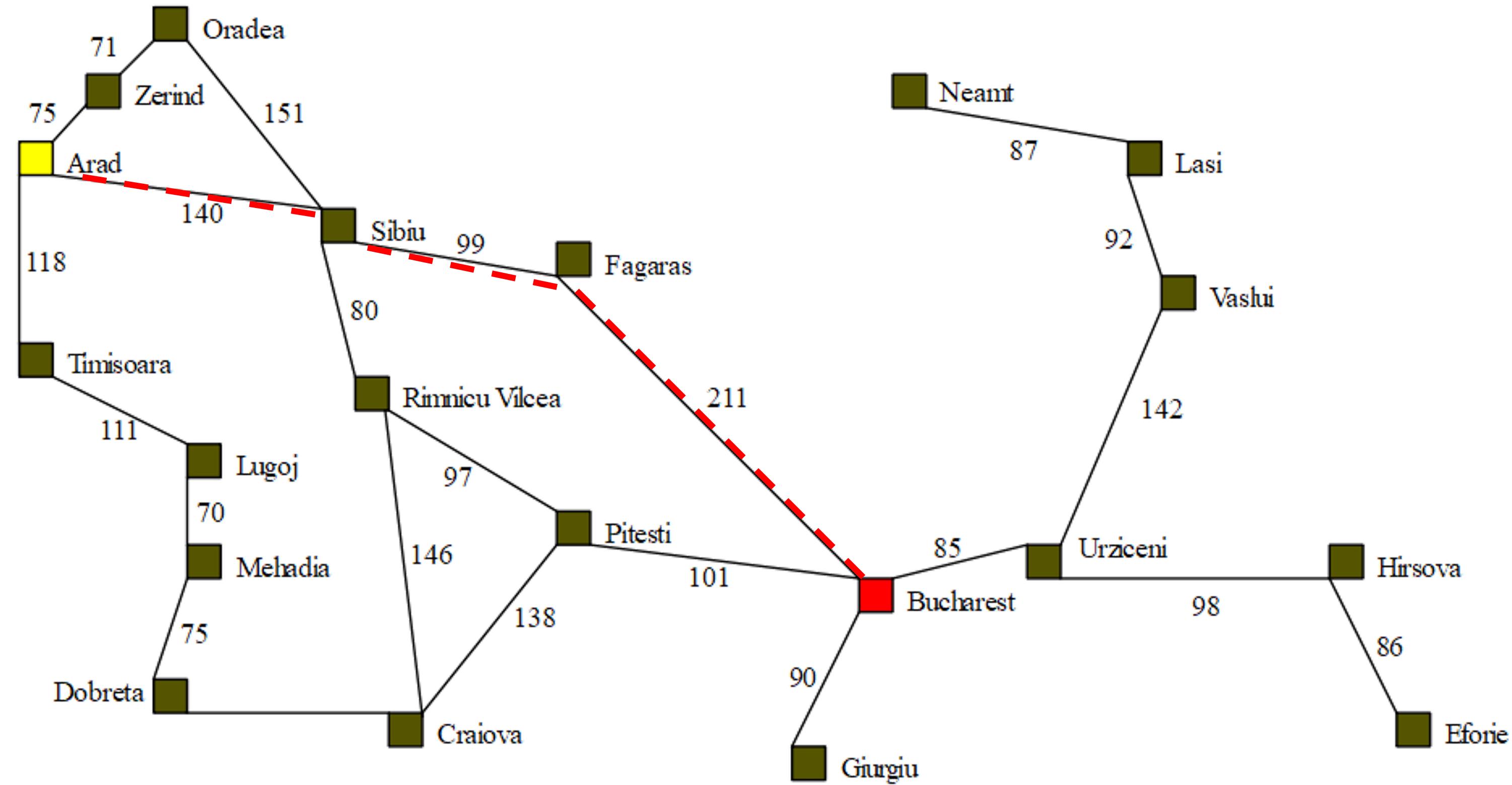
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Lasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

- $n = \text{Fagaras}$, $\Gamma(n) = \{\text{Sibiu}, \text{Bucharest}\}$,
- $\text{CLOSE} = \{\text{Arad}, \text{Sibiu}, \text{Rimnicu Vilcea}, \text{Pitesti}, \text{Fagaras}\}$
 - $g_{\text{new}}(\text{Bucharest}) = g(\text{Fagaras}) + \text{cost}(\text{Fagaras} \rightarrow \text{Bucharest}) = 417 + 0 = 417$
 - $f_{\text{new}}(\text{Bucharest}) = g_{\text{new}}(\text{Bucharest}) + h'(\text{Bucharest}) = 417 + 0 = 417$
- $\text{OPEN} = \{\text{Bucharest}_{\text{new}} - 417, \text{Timisoara} - 447, \text{Zerind}-449, \text{Craiova} - 526 < f_{\text{new}} 615\}$
 - $f_{\text{min}} = f(\text{Bucharest})$,
 - **Father(Bucharest) = Fagaras**
- $n = \text{Bucharest}$, TRUE

Chương 4. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

- Algorithm for Knowledgeable Tree Search (AKT)
- Tìm đường đi ngắn nhất từ Arad → Bucharest:

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



Chương 4. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

- A star(A*)
- Thuật giải A* tránh việc xét các nhánh tìm kiếm đã xác định (cho đến thời điểm hiện tại) có chi phí cao.
- Sử dụng hàm đánh giá $f(u) = g(u) + h(u)$
 - $g(u)$: chi phí của đường đi từ nút gốc \square nút hiện tại
 - $h(u)$: ước lượng heuristic = khoảng cách Manhattan
 - $f(u)$: chi phí tổng thể ước lượng của đường đi qua nút hiện tại (u) \rightarrow đích
- A* là phiên bản đặc biệt của A^{KT} áp dụng cho trường hợp đồ thị
- A* mở rộng A^{KT} bằng cách bổ sung cách giải quyết trường hợp khi mở một nút mà nút này đã có sẵn trong OPEN hoặc CLOSE.

Chương 4. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

- A star(A*)
- OPEN: tập chứa các trạng thái đã được sinh ra nhưng chưa xét đến
 - Các phần tử được sắp xếp theo độ tốt (giá trị của hàm f)
 - Phần tử có độ ưu tiên cao nhất = phần tử tốt nhất
- CLOSE: Danh sách các trạng thái đã xét
- Hàm REMOVE(): Lấy phần tử ở đầu danh sách OPEN
- Hàm INSERT(): Chèn phần tử vào OPEN theo thứ tự ưu tiên
- Hàm FATHER(x): Trạng thái cha của X

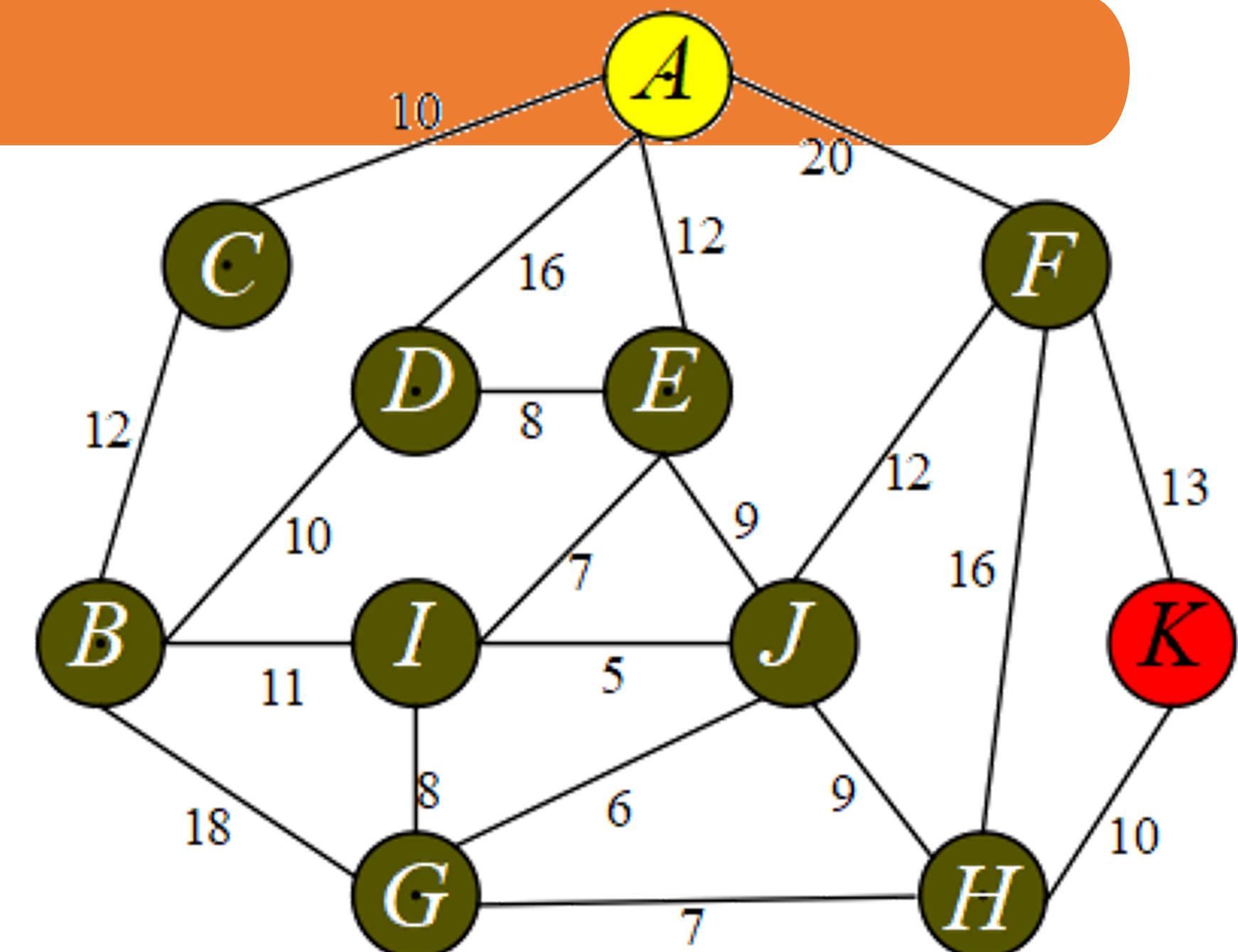
```
function A_Star_Search(){  
    Input: start, goal;  
    Output: Đường đi từ start → goal hoặc thất bại  
}
```

Chương 4. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

- A star(A*)
- OPEN = start; CLOSE = \emptyset ; $g(\text{start}) = 0$; $f(\text{start}) = h(\text{start})$
- Loop:
 - if ($\text{OPEN} = \emptyset$): Thất bại
 - $n = \text{REMOVE}(\text{OPEN})$ //phần tử đầu ds OPEN ($f(n) == \min$)
 - if ($n \equiv \text{goal}$): Tìm thấy đường đi
 - Ngược lại:
 - Đưa $\{n\}$ vào danh sách CLOSE
 - for (mỗi trạng thái m là con của $\{n\}$):
 - Tính $g(m) = g(n) + \text{cost}(n,m)$
 - Tính $f(m) = g(m) + h(m)$
 - if ($\exists! t \in \text{OPEN} \equiv m$): **(cond 1)**
 - elseif ($\exists! t \in \text{CLOSE} \equiv m$): **(cond 2)**
 - else $\text{INSERT}(n, \text{OPEN}) + \text{Sắp xếp lại OPEN} // m \notin \{\text{OPEN}, \text{CLOSE}\}$
 - if ($\exists! t \in \text{OPEN} \equiv m$):
 - if ($f(m)_{\text{new}} < f(t)_{\text{old}}$):
 - $g(t)_{\text{old}} = g(m)_{\text{new}}$
 - $f(t)_{\text{old}} = f(m)_{\text{new}}$
 - FATHER(t) = n
 - elseif ($\exists! t \in \text{CLOSE} \equiv m$):
 - if ($f(m)_{\text{new}} < f(t)_{\text{old}}$):
 - $g(t)_{\text{old}} = g(m)_{\text{new}}$
 - $f(t)_{\text{old}} = f(m)_{\text{new}}$
 - FATHER(t) = n

Chương 4. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

- A star(A*)
- Áp dụng giải thuật A* để tìm đường đi ngắn nhất $A \rightarrow K$, với các ước lượng heuristic của các trạng thái so với trạng thái đích được cho ở bảng sau:

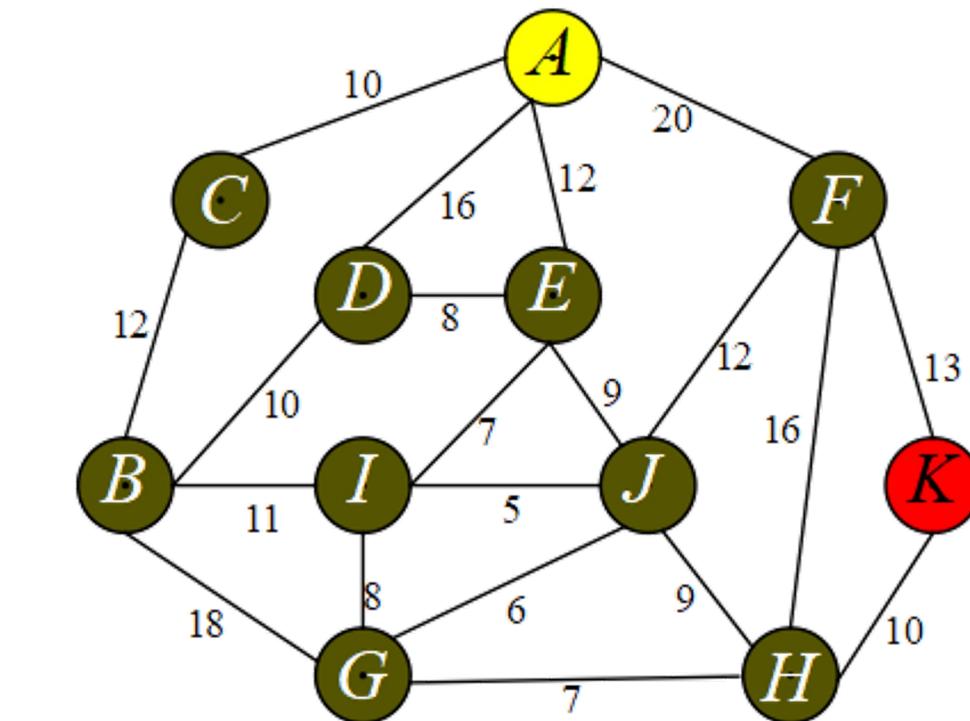


A	B	C	D	E	F	G	H	I	J	K
33	35	43	36	28	13	17	10	24	19	0

Chương 4. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

- A star(A*)

- Start = {A}, goal = {K}, OPEN = {A}, CLOSE = \emptyset
- $g(A) = 0, f(A) = h(A) = 33$
- Bước 1. n=A, CLOSE={A}
 - $g(C) = g(A)+\text{cost}(A,C) = 0+10 = 10$
 - $f(C) = g(C)+h(C) = 10+43 = 53$
 - $g(D) = g(A)+\text{cost}(A,D) = 0+16 = 16$
 - $f(D) = g(D)+h(D) = 16+36 = 49$
 - $g(E) = g(A)+\text{cost}(A,E) = 0+12 = 12$
 - $f(E) = g(E)+h(E) = 12+28 = 40$
 - $g(F) = g(A)+\text{cost}(A,F) = 0+20 = 20$
 - $f(F) = g(F)+h(F) = 20+13 = 33$
 - $\{C,D,E,F\} \notin \text{OPEN} \text{ và } \{C,D,E,F\} \notin \text{CLOSE}$
 - OPEN = {F(33), E(40), D(49), C(53)}

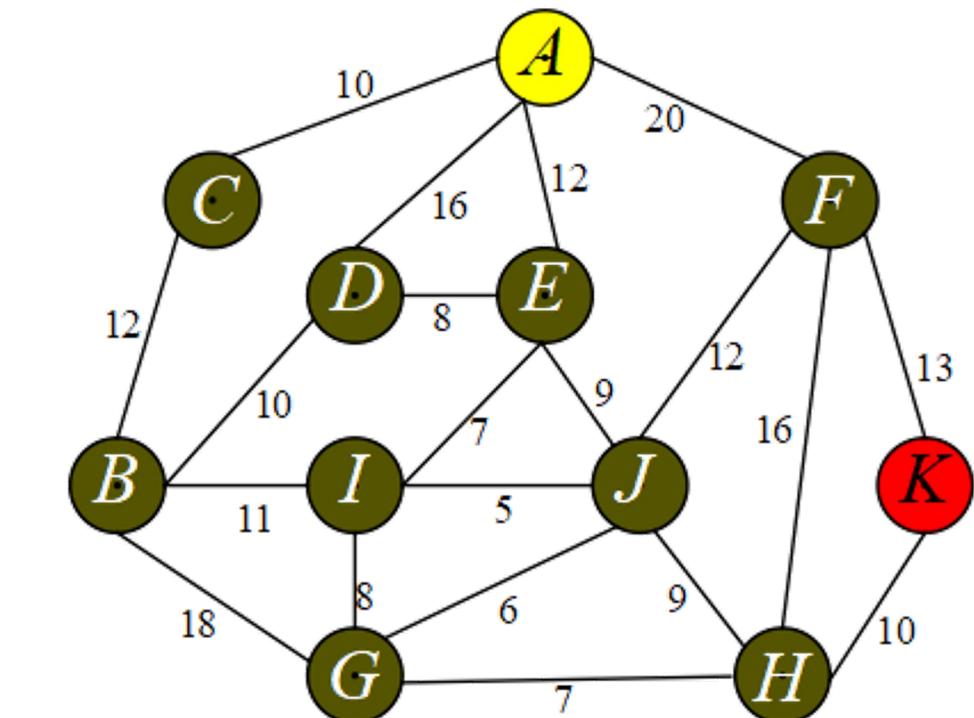


A	B	C	D	E	F	G	H	I	J	K
33	35	43	36	28	13	17	10	24	19	0

- Father(F, E, D, C) = A

Chương 4. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

- A star(A*)
- Bước 2. $n=F$, $CLOSE=\{A,F\}$, $OPEN = \{E,D,C\}$
 - $g_{new}(A) = g(F)+cost(F,A) = 20+20 = 40$
 - $f_{new}(A) = g(A)+h(A) = 40+33 = 73$
 - $t=A \in CLOSE, f_{new}(A) < f(t) (73 < 55)$: FALSE
 - $g(A)=0, f(A)=33$
 - $g(J) = g(F)+cost(F,J) = 20+12 = 32$
 - $f(J) = g(J)+h(J) = 32+19 = 51$
 - $g(H) = g(F)+cost(F,H) = 20+16 = 36$
 - $f(H) = g(H)+h(H) = 36+10 = 46$
 - $g(K) = g(F)+cost(F,K) = 20+13 = 33$
 - $f(K) = g(K)+h(K) = 33+0 = 33$
 - $\{H,J,K\} \notin OPEN$ và $\{H,J,K\} \notin CLOSE$
 - $OPEN = \{K(33),E(40), H(46), D(49), J(51), C(53)\}$



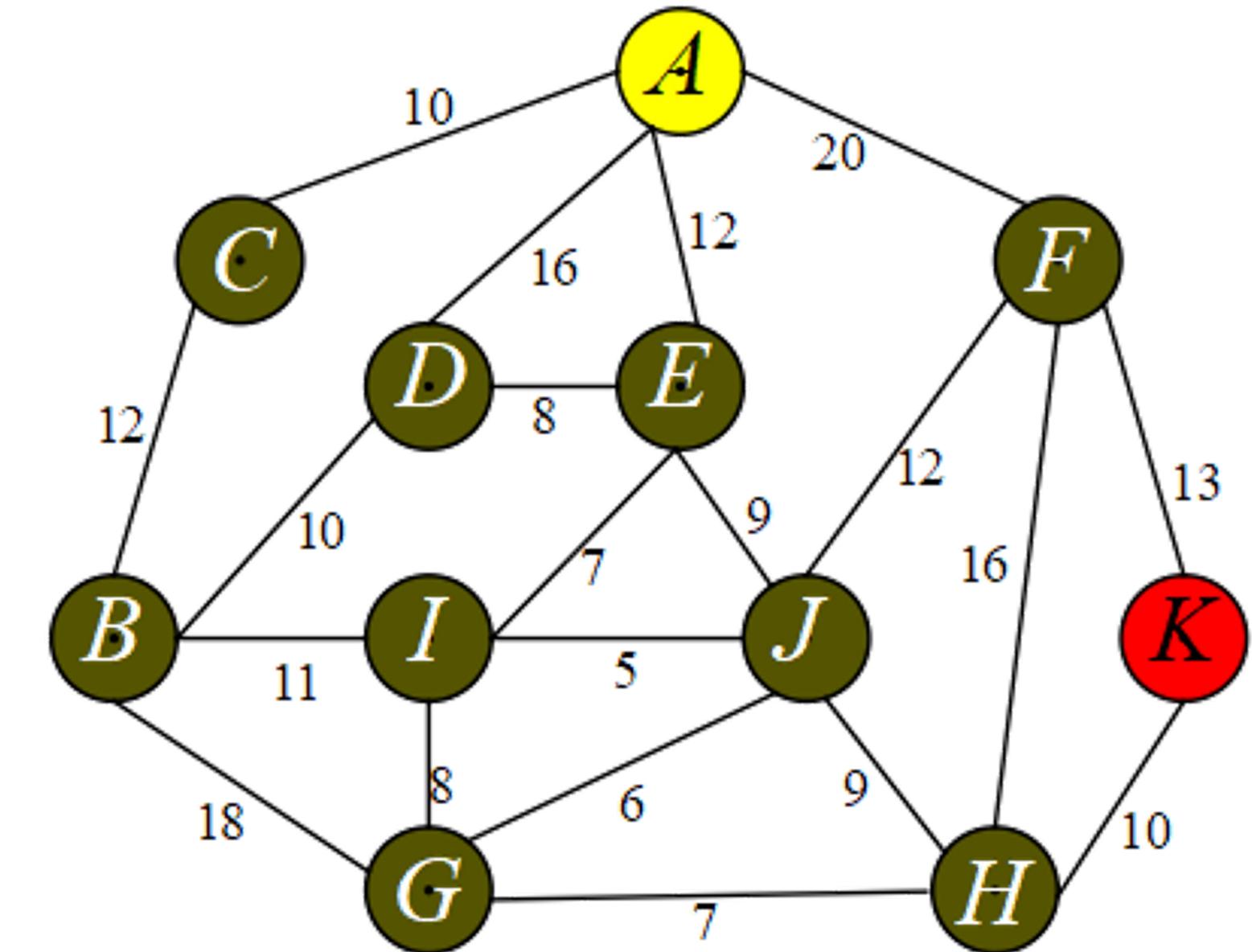
A	B	C	D	E	F	G	H	I	J	K
33	35	43	36	28	13	17	10	24	19	0

- Father(J, H, K) = F

Chương 4. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

- A star(A*)
- Bước 3. n=K, TRUE
- Father(J, H, K) = F
- Father(F, E, D, C) = A

▪ $A \Rightarrow F \Rightarrow K$

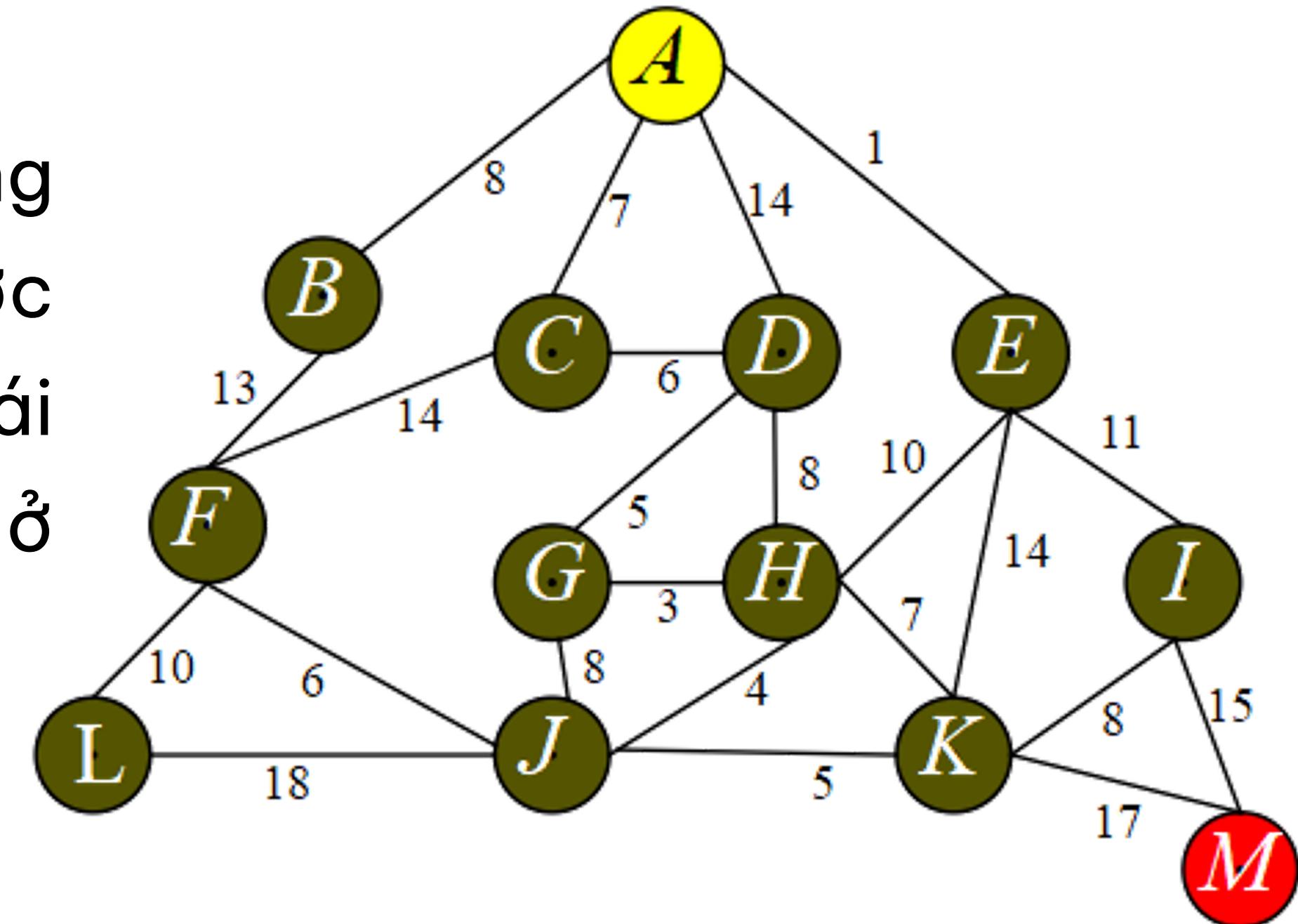


A	B	C	D	E	F	G	H	I	J	K
33	35	43	36	28	13	17	10	24	19	0

Chương 4. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

- A star(A*)

- Áp dụng giải thuật A* để tìm đường đi ngắn nhất $A \rightarrow M$, với các ước lượng heuristic của các trạng thái so với trạng thái đích được cho ở bảng sau:

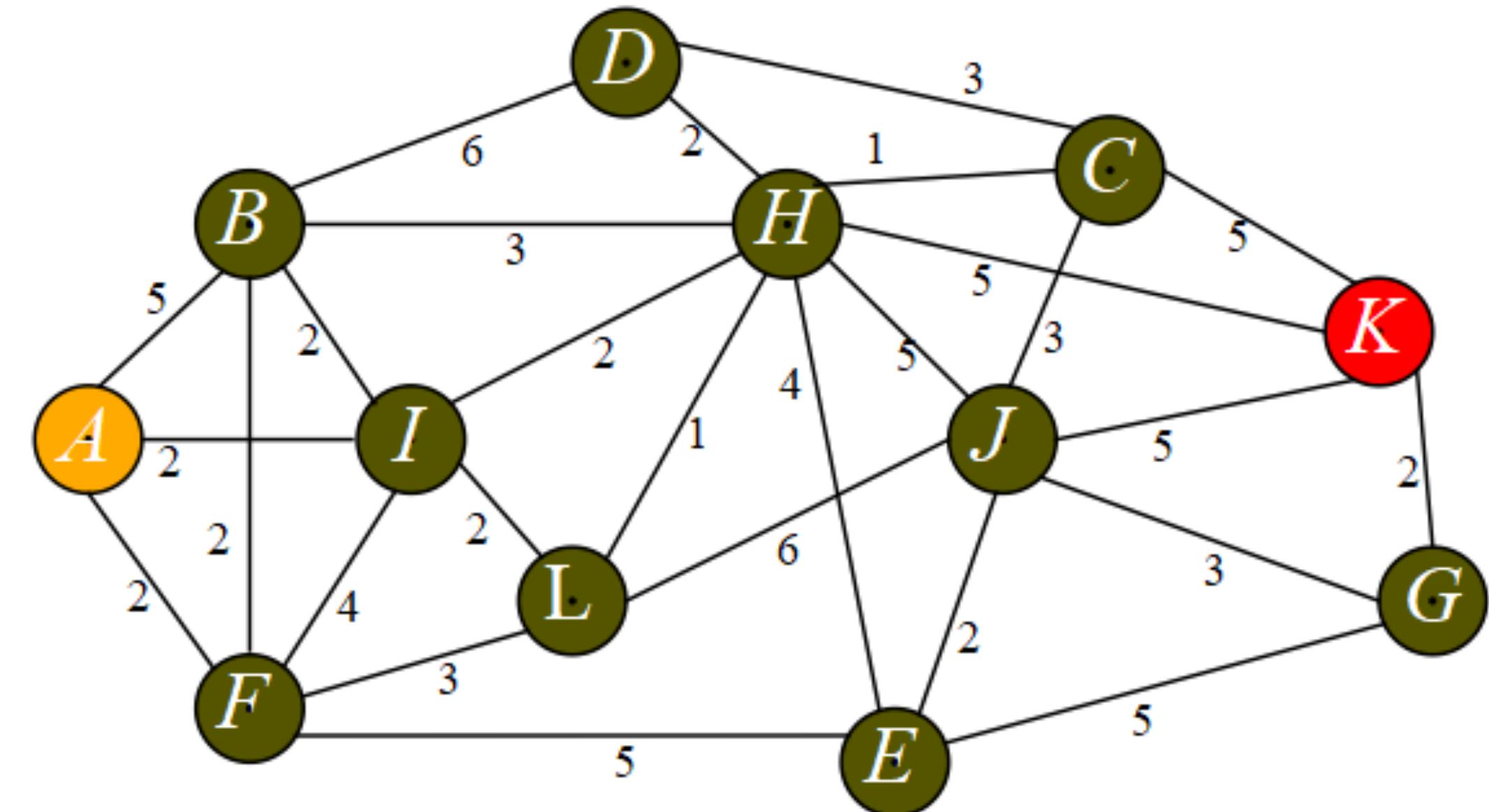


A	B	C	D	E	F	G	H	I	J	K	L	M
27	34	33	32	26	28	27	24	15	22	17	15	0

Chương 4. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

- A star(A*)

- Áp dụng giải thuật A* để tìm đường đi ngắn nhất A → K, với các ước lượng heuristic của các trạng thái so với trạng thái đích được cho ở bảng sau:



A	B	C	D	E	F	G	H	I	J	K	L
9	8	5	7	7	9	2	5	7	5	0	6

Chương 4. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

- Nhánh - cận (Branch-and-Bound)
- Thuật toán tìm kiếm leo đồi kết hợp với hàm đánh giá $f(u)$. Tại mỗi bước, khi phát triển trạng thái u , chọn trạng thái con v tốt nhất ($f(v)$ nhỏ nhất) của u để phát triển ở bước sau.
- Quá trình tiếp tục như vậy cho đến khi gặp trạng thái w là đích, hoặc w không có đỉnh kề, hoặc w có $f(w)$ lớn hơn độ dài đường đi tối ưu tạm thời. Trong các trường hợp này, chúng ta không phát triển đỉnh w nữa, tức là cắt bỏ những nhánh xuất phát từ w , và quay lên cha của w để tiếp tục đi xuống trạng thái tốt nhất trong số những trạng thái còn lại chưa được phát triển

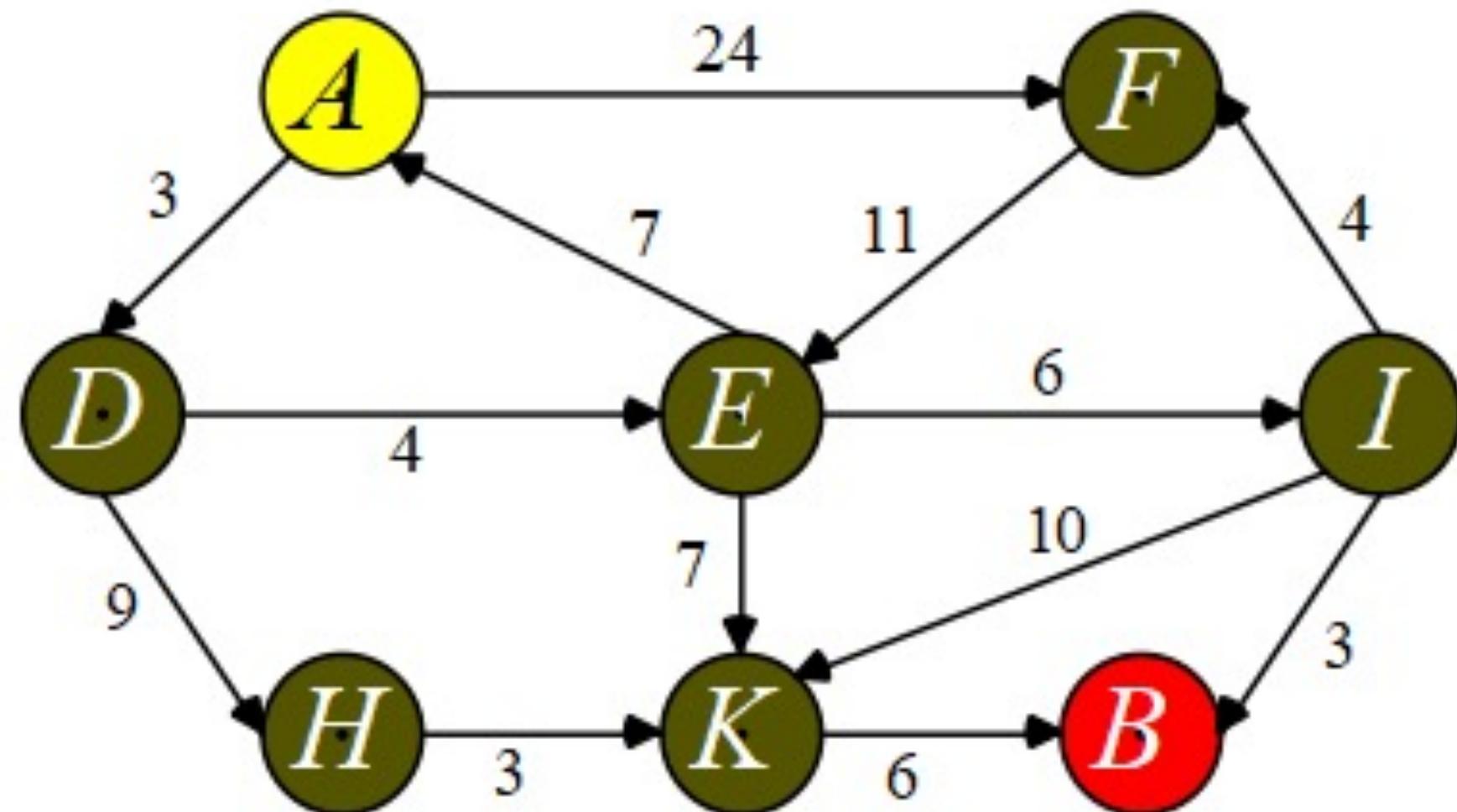
Chương 4. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

- Nhánh – cận (Branch-and-Bound)
- Bước 1. OPEN = {start}, min = $+\infty$
- Bước 2. Loop:
 - 2.1. if(OPEN $\equiv \emptyset$): STOP
 - 2.2. n:= lấy phần tử đầu danh sách OPEN
 - 2.3. if(n \equiv goal):
 - if ($f(n) < min$): { $min=f(n)$; quay lại 2.1 //cập nhật lại min}}
 - 2.4. if ($f(n) > min$): quay lại 2.1 //cắt bỏ nhánh con
 - else for ($\forall m \notin CLOSE$: đỉnh kề n):
 - $g(m) = g(n) + cost(m,n)$
 - $f(m) = g(m)+h(m)$
 - Đưa m vào danh sách $\Gamma(n)$
 - 2.5. Sắp xếp $\Gamma(n)$ theo thứ tự tăng dần của f
 - 2.6. Chèn $\Gamma(n)$ vào đầu OPEN

Chương 4. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

- Nhánh - cận (Branch-and-Bound)

- Áp dụng thuật toán nhánh cận để tìm đường đi ngắn nhất từ đỉnh A → B, với các ước lượng heuristic của các trạng thái so với trạng thái đích được cho ở Bảng.



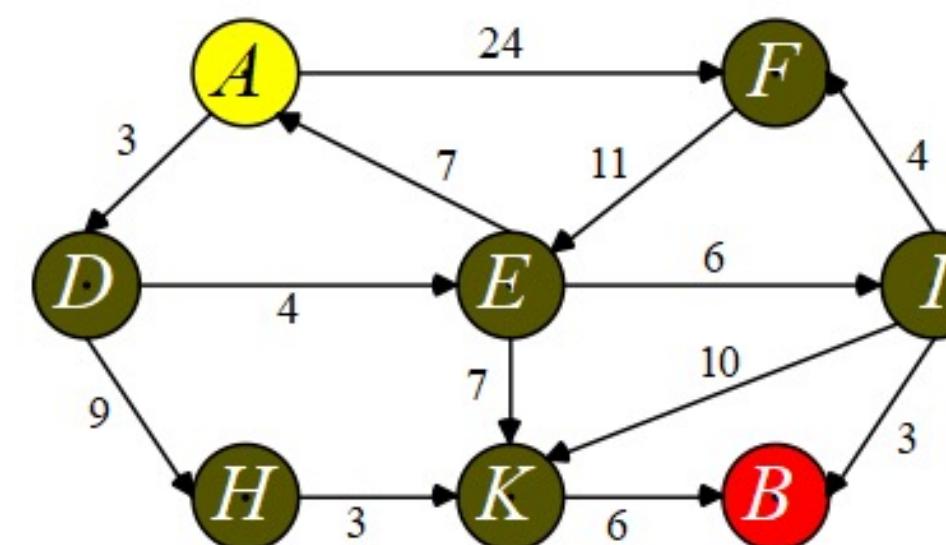
A	B	D	E	F	H	I	K
13	0	2	8	9	7	6	4

Chương 4. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

- Nhánh - cận (Branch-and-Bound)

Bước	n	m	$g(m) = g(n) + \text{cost}(n,m)$	$f(m) = g(m) + h(m)$	$\Gamma(n) \uparrow$	OPEN	min
0			$g(A) = 0$	$f(A) = h(A) = 13$	\emptyset	A	$+\infty$
1	A $f(A) < \text{min}$	D F	$D: g(D) = g(A) + \text{cost}(A,D)$ $= 0 + 3 = 3$ $F: g(F) = g(A) + \text{cost}(A,F)$ $= 0 + 24 = 24$	$f(D) = g(D) + h(D)$ $= 3 + 2 = 5$ $f(F) = g(F) + h(F)$ $= 24 + 9 = 33$	$D^5 \rightarrow F^{33}$	D^5, F^{33}	$+\infty$

- Father(D,F) = A



VET								
A	B	D	E	F	H	I	K	
1	-	-	-	-	-	-	-	-

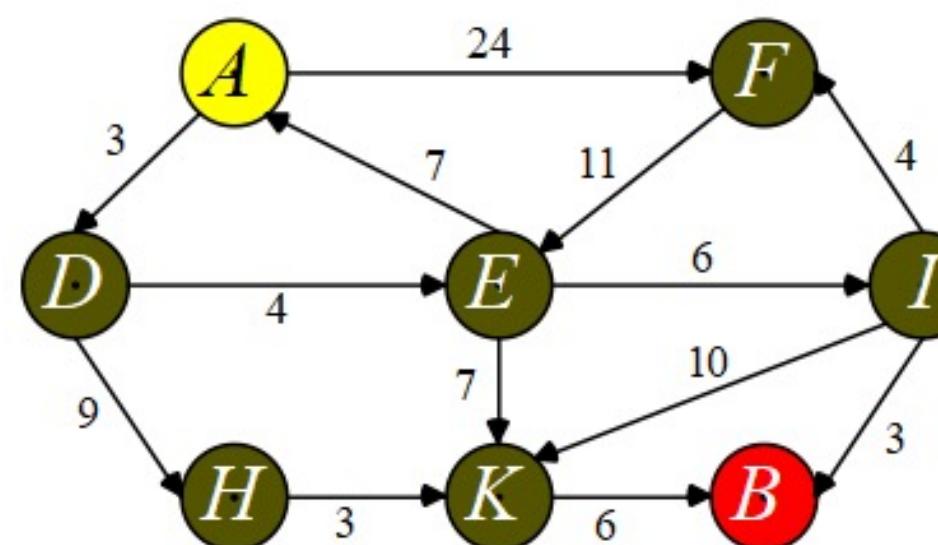
HÀM LƯỢNG GIÁ (H)								
A	B	D	E	F	H	I	K	
13	0	2	8	9	7	6	4	

Chương 4. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

- Nhánh - cận (Branch-and-Bound)

Bước	n	m	$g(m) = g(n) + \text{cost}(n,m)$	$f(m) = g(m) + h(m)$	$\Gamma(n) \uparrow$	OPEN	min
2	D f(D)<min	E H	$\begin{aligned} g(E) &= g(D) + \text{cost}(D,E) \\ &= 3 + 4 = 7 \end{aligned}$	$\begin{aligned} f(E) &= g(E) + h(E) \\ &= 7 + 8 = 15 \end{aligned}$	$E^{15} \rightarrow H^{19}$	E^{15}, H^{19}, F^{33}	$+\infty$

- Father(D,F) = A
- Father(E,H) = D



VET							
A	B	D	E	F	H	I	K
1	-	2	-	-	-	-	-

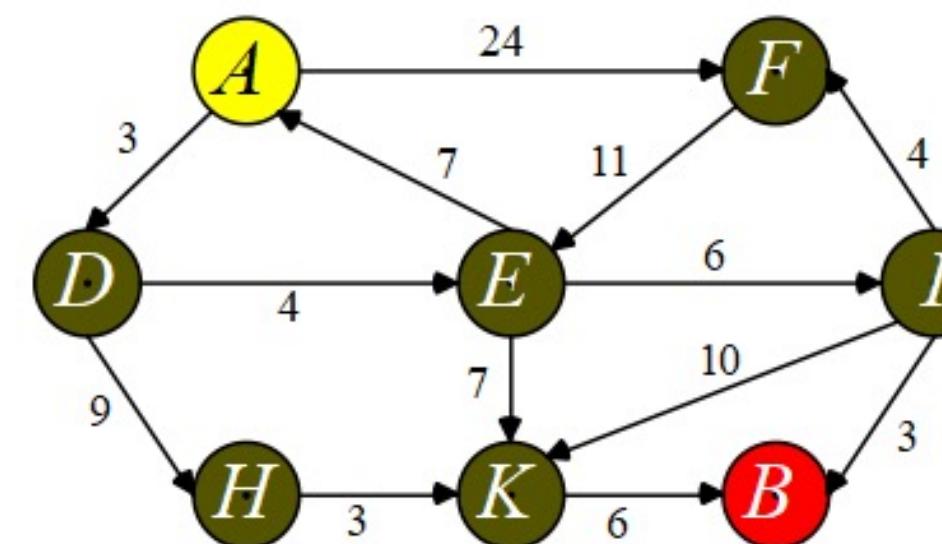
HÀM LƯỢNG GIÁ (H)							
A	B	D	E	F	H	I	K
13	0	2	8	9	7	6	4

Chương 4. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

Nhánh - cận (Branch-and-Bound)

Bước	n	m	$g(m) = g(n) + \text{cost}(n,m)$	$f(m) = g(m) + h(m)$	$\Gamma(n) \uparrow$	OPEN	min
3 E $f(E) < \text{min}$	I		$g(I) = g(E) + \text{cost}(E,I)$ $= 7 + 6 = 13$	$f(I) = g(I) + h(I)$ $= 13 + 6 = 19$	$K^{18} \rightarrow$ $I^{19} \rightarrow$ A^{26}	$K^{18}, I^{19}, A^{27},$ H^{19}, F^{33}	$+ \infty$
	K		$g(K) = g(E) + \text{cost}(E,K)$ $= 7 + 7 = 14$	$f(K) = g(K) + h(K)$ $= 14 + 4 = 18$			
	A		$g(A) = g(E) + \text{cost}(E,A)$ $= 7 + 7 = 14$	$f(A) = g(A) + h(A)$ $= 14 + 13 = 27$			

- Father(D,F) = A
- Father(E,H) = D
- Father(I,K,A) = E



VET							
A	B	D	E	F	H	I	K
1	-	2	3	-	-	-	-

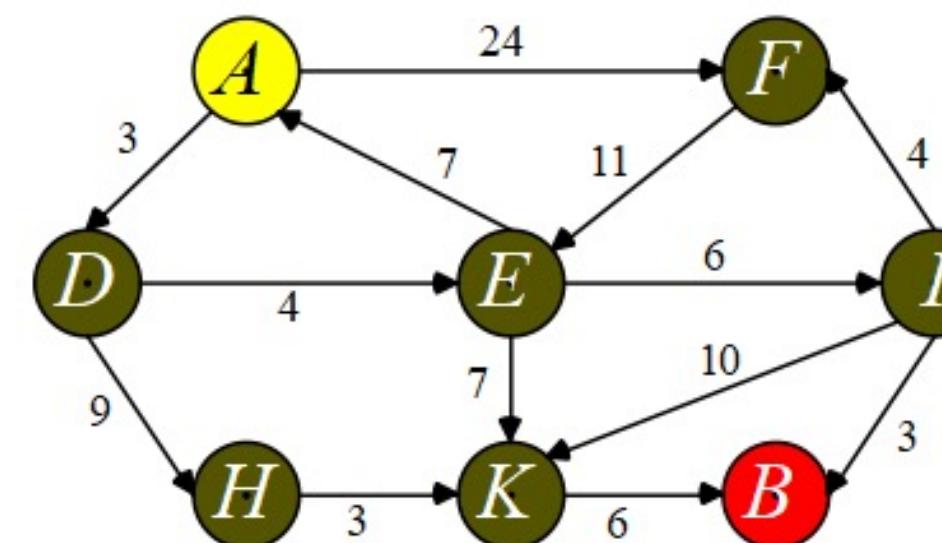
HÀM LƯỢNG GIÁ (H)							
A	B	D	E	F	H	I	K
13	0	2	8	9	7	6	4

Chương 4. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

▪ Nhánh - cận (Branch-and-Bound)

Bước	n	m	$g(m)=g(n) + \text{cost}(n,m)$	$f(m)=g(m) + h(m)$	$\Gamma(n) \uparrow$	OPEN	min
4	K $f(K) < \text{min}$	B	$g(B) = g(K) + \text{cost}(K,B)$ $= 14 + 6 = 20$	$f(B) = g(B) + h(B)$ $= 20 + 0 = 20$	B^{20}	$B^{20}, I^{19}, A^{26},$ H^{19}, F^{33}	$+\infty$

- Father(D,F) = A
- Father(E,H) = D
- Father(I,K,A) = E



VET							
A	B	D	E	F	H	I	K
1	-	2	3	-	-	-	4

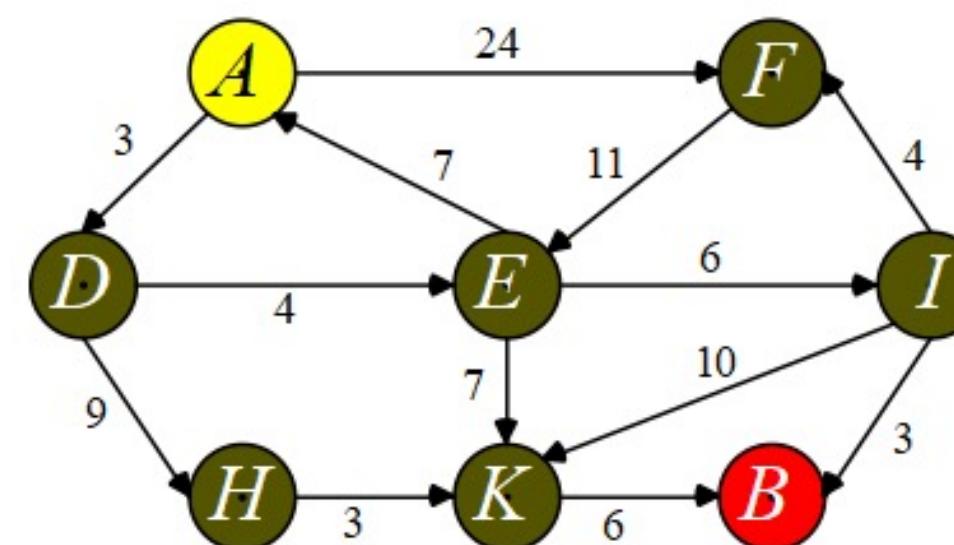
HÀM LƯỢNG GIÁ (H)							
A	B	D	E	F	H	I	K
13	0	2	8	9	7	6	4

Chương 4. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

Nhánh - cận (Branch-and-Bound)

Bước	n	m	$g(m) = g(n) + \text{cost}(n,m)$	$f(m) = g(m) + h(m)$	$\Gamma(n) \uparrow$	OPEN	min
5	B	B		$f(B) = 20 < \text{min}$		$I^{19}, A^{19}, H^{19}, F^{33}$	20
6 $f(I) < \text{min}$	I	B	$g(B) = g(I) + \text{cost}(I,B)$ = $13 + 3 = 16$	$f(B) = g(B) + h(B)$ = $16 + 0 = 16$	$B^{16} \rightarrow$ $K^{27} \rightarrow$ F^{36}	$B^{16}, K^{27}, F^{36},$ A^{19}, H^{19}, F^{33}	20
		F	$g(F) = g(I) + \text{cost}(I,F)$ = $13 + 4 = 17$	$f(F) = g(F) + h(F)$ = $17 + 19 = 36$			
		K	$g(K) = g(I) + \text{cost}(I,K)$ = $13 + 10 = 23$	$f(K) = g(K) + h(K)$ = $23 + 4 = 27$			

- Father(D,F) = A
- Father(E,H) = D
- Father(I,K,A) = E



VET							
A	B	D	E	F	H	I	K
1	-	2	3	-	-	-	4

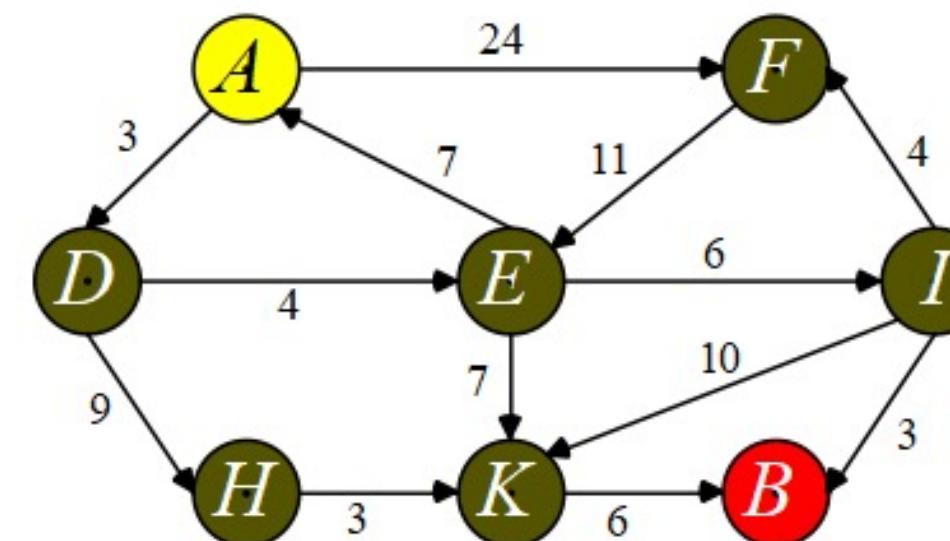
HÀM LƯỢNG GIÁ (H)							
A	B	D	E	F	H	I	K
13	0	2	8	9	7	6	4

Chương 4. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

▪ Nhánh - cận (Branch-and-Bound)

Bước	N	m	$g(m) = g(n) + \text{cost}(n,m)$	$f(m) = g(m) + h(m)$	$\Gamma(n) \uparrow$	OPEN	min
7	B	B		$f(B) = 16 < \text{min}$		$K^{27}, F^{36}, A^{19}, H^{19}, F^{33}$	16
8	K		quay lại 2.1 (cắt bỏ nhánh con)	$f(K) = 27 > \text{min}$		$F^{36}, A^{19}, H^{19}, F^{33}$	16

- Father(D,F) = A
- Father(E,H) = D
- Father(I,K,A) = E



VET							
A	B	D	E	F	H	I	K
1	-	2	3	-	-	-	4

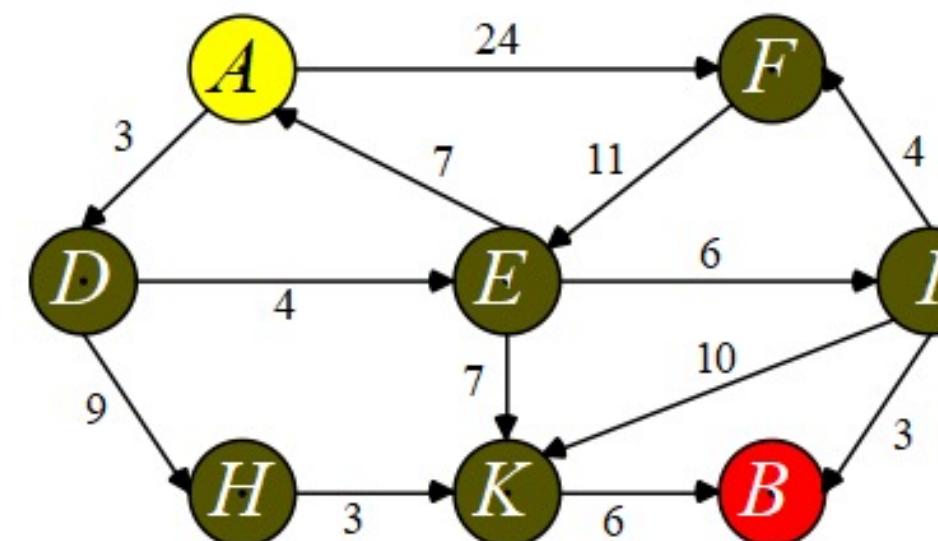
HÀM LƯỢNG GIÁ (H)							
A	B	D	E	F	H	I	K
13	0	2	8	9	7	6	4

Chương 4. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

▪ Nhánh - cận (Branch-and-Bound)

Bước	N	m	$g(m) = g(n) + \text{cost}(n,m)$	$f(m) = g(m) + h(m)$	$\Gamma(n) \uparrow$	OPEN	min
9	F		quay lại 2.1 (cắt bỏ nhánh con)	$f(F) = 36 > min$		A^{19}, H^{19}, F^{33}	16
10	A		quay lại 2.1 (cắt bỏ nhánh con)	$f(A) = 19 > min$		A^{19}, H^{19}, F^{33}	16

- Father(D,F) = A
- Father(E,H) = D
- Father(I,K,A) = E



VET							
A	B	D	E	F	H	I	K
1	-	2	3	-	-	-	4

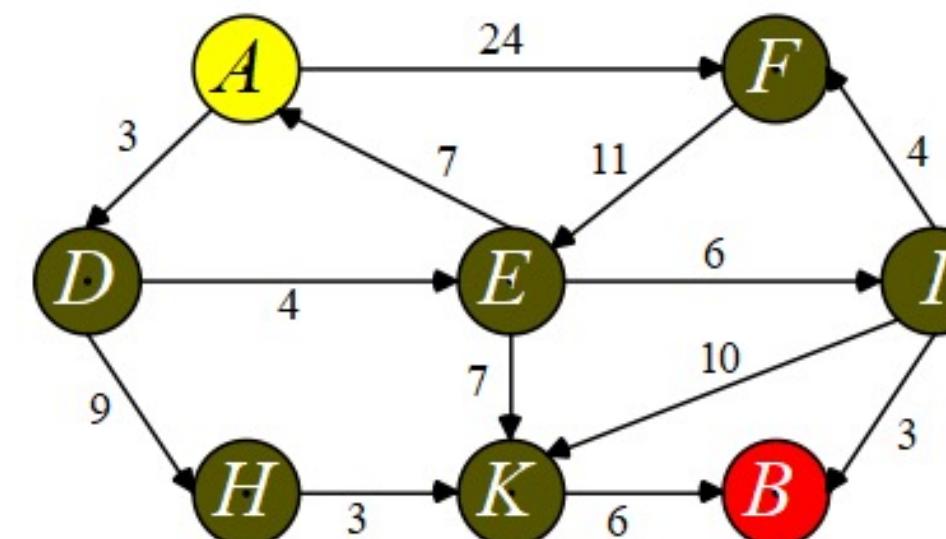
HÀM LƯỢNG GIÁ (H)							
A	B	D	E	F	H	I	K
13	0	2	8	9	7	6	4

Chương 4. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

▪ Nhánh - cận (Branch-and-Bound)

Bước	N	m	$g(m) = g(n) + \text{cost}(n,m)$	$f(m) = g(m) + h(m)$	$\Gamma(n) \uparrow$	OPEN	min
11	H		quay lại 2.1 (cắt bỏ nhánh con)	$f(H) = 19 > min$		F^{33}	16
12	F		quay lại 2.1 (cắt bỏ nhánh con)	$f(F) = 33 > min$		\emptyset	16
13	TRUE						

- Father(D,F) = A
- Father(E,H) = D
- Father(I,K,A) = E



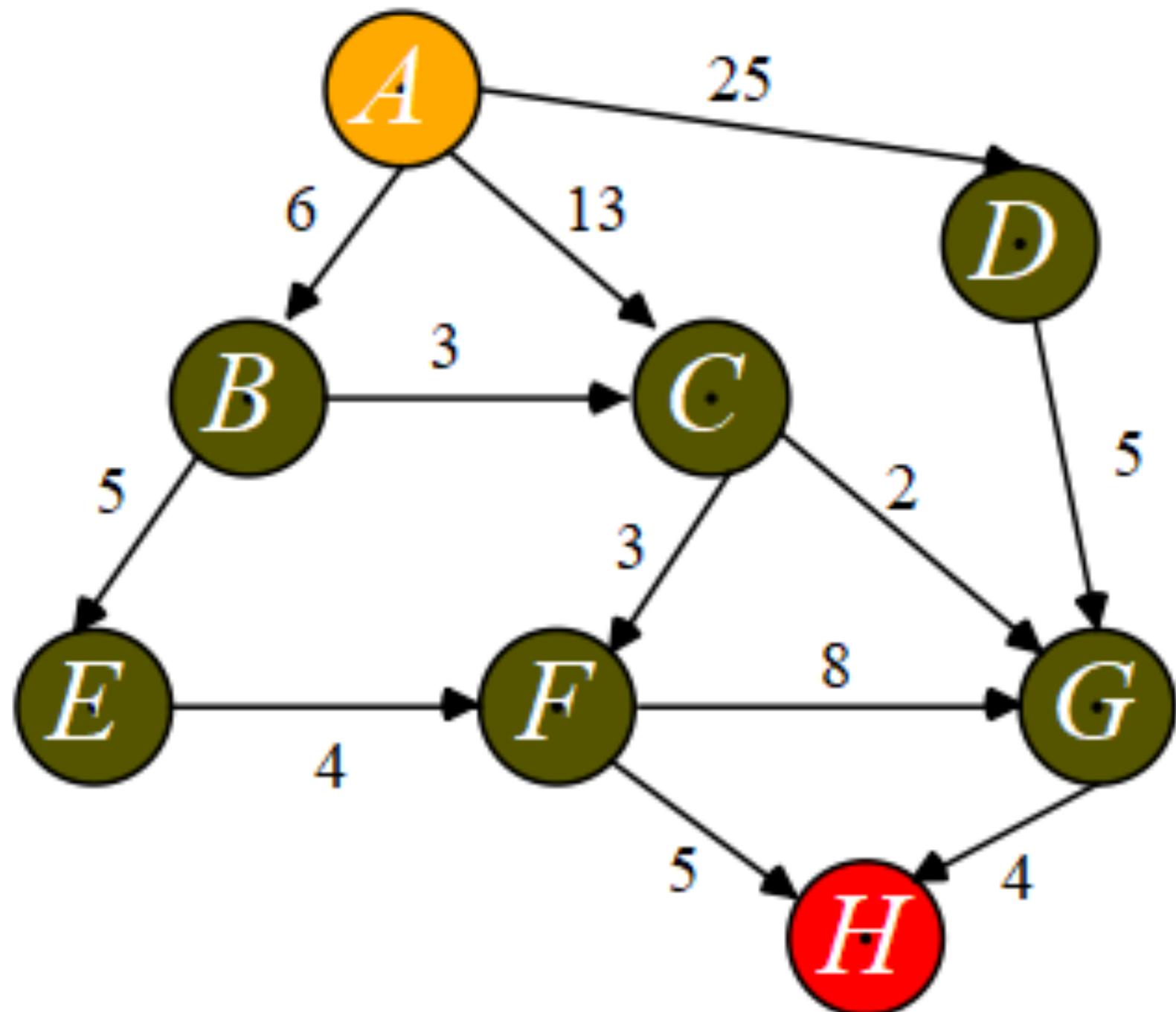
VET							
A	B	D	E	F	H	I	K
1	-	2	3	-	-	-	4

HÀM LƯỢNG GIÁ (H)							
A	B	D	E	F	H	I	K
13	0	2	8	9	7	6	4

Chương 4. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

- Nhánh - cận (Branch-and-Bound)

- Áp dụng thuật toán nhánh cận để tìm đường đi ngắn nhất từ đỉnh A → I, với các ước lượng heuristic của các trạng thái so với trạng thái đích được cho ở Bảng.



A	B	C	D	E	F	G	H
20	5	7	6	9	1	3	0

Chương 5. GIẢI THUẬT TÌM KIẾM LỜI GIẢI CHO TRÒ CHƠI

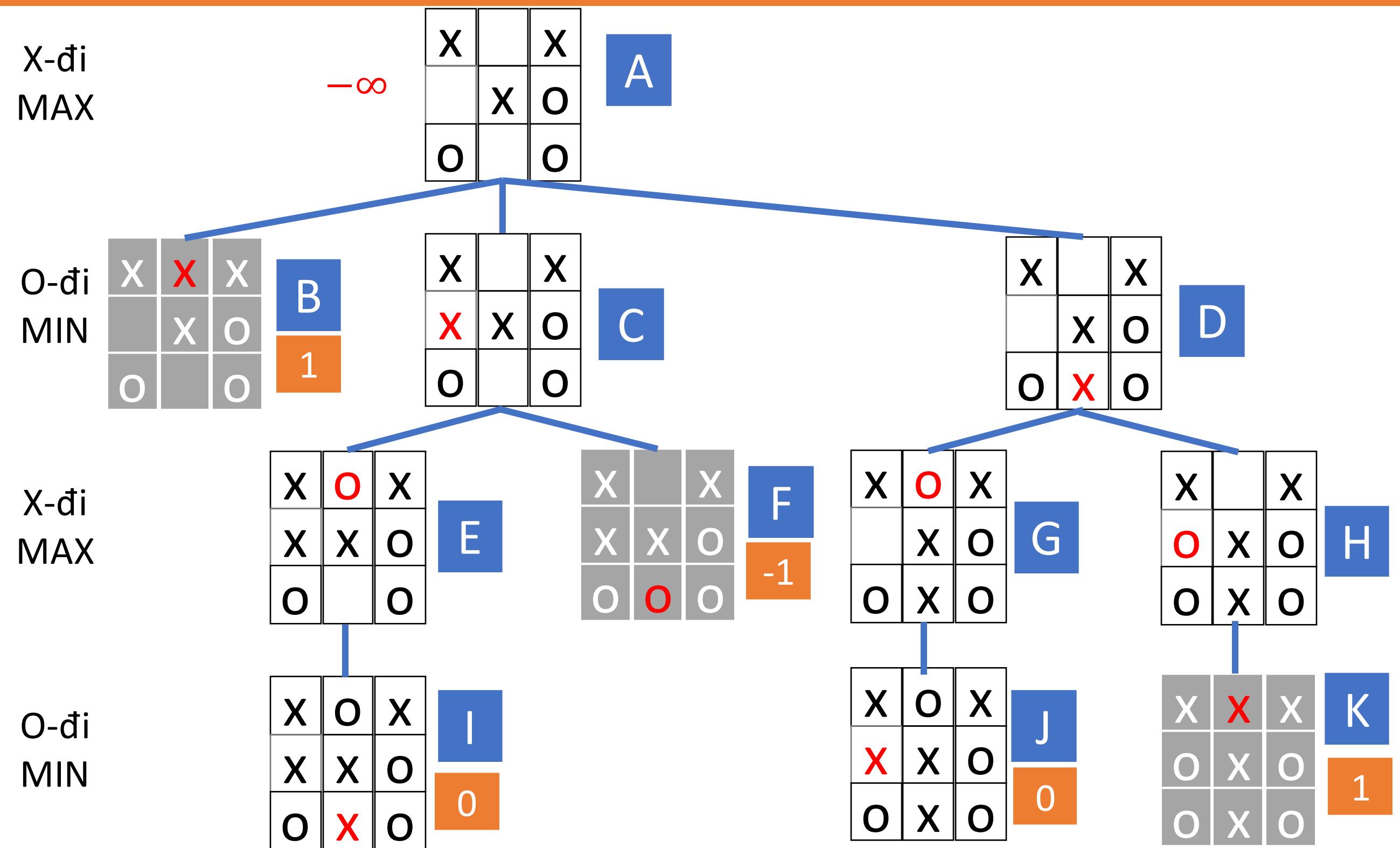
Trò chơi 2 người đối kháng

Kỹ thuật vét cạn (MIN-MAX)

- Giá trị nút MAX là giá trị lớn nhất của các nút con
- Giá trị nút MIN là giá trị nhỏ nhất của các nút con
- Bài toán tìm MAX hoặc MIN của dãy số (giải thuật vét cạn)
- Có 2 cách giải:
 - Cách 1: MAX = MIN = A0, duyệt A1-An tìm MAX, MIN
 - Cách 2: MAX = $-\infty$ và MIN = $+\infty$
- Áp dụng cách 2 cho bài toán: Mỗi nút sẽ có một giá trị tạm: Nút MAX có giá trị tạm = $-\infty$ và nút MIN có giá trị tạm $+\infty$
- Khi tìm thấy giá trị của một nút con thì tính lại giá trị tạm của nút cha: giá trị tạm mới của nút MAX = LN(giá trị tạm cũ, giá trị nút con) VÀ giá trị tạm mới của nút MIN = NN(giá trị tạm cũ, giá trị của con)

Chương 5. GIẢI THUẬT TÌM KIẾM LỜI GIẢI CHO TRÒ CHƠI

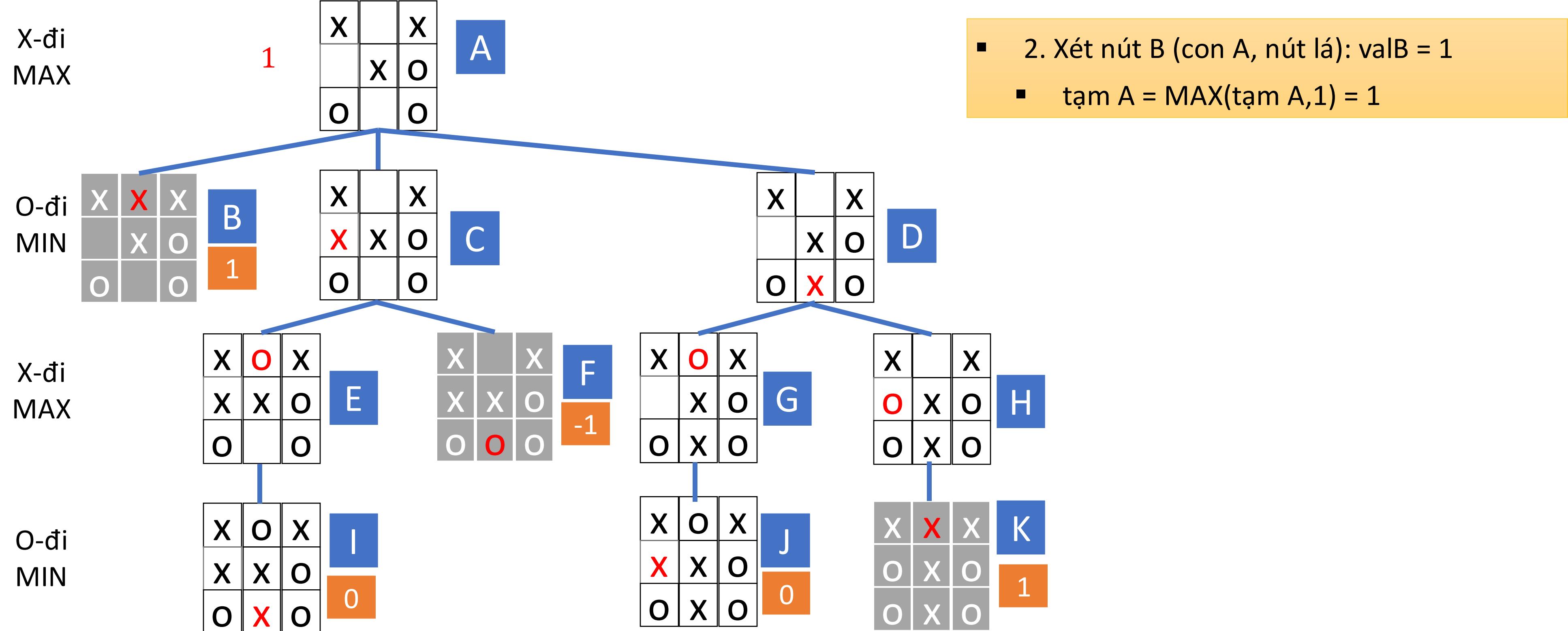
Kỹ thuật vét cạn (MIN-MAX)



- 1. Xét nút A: tạm $A = -\infty$ (vì A là MAX)

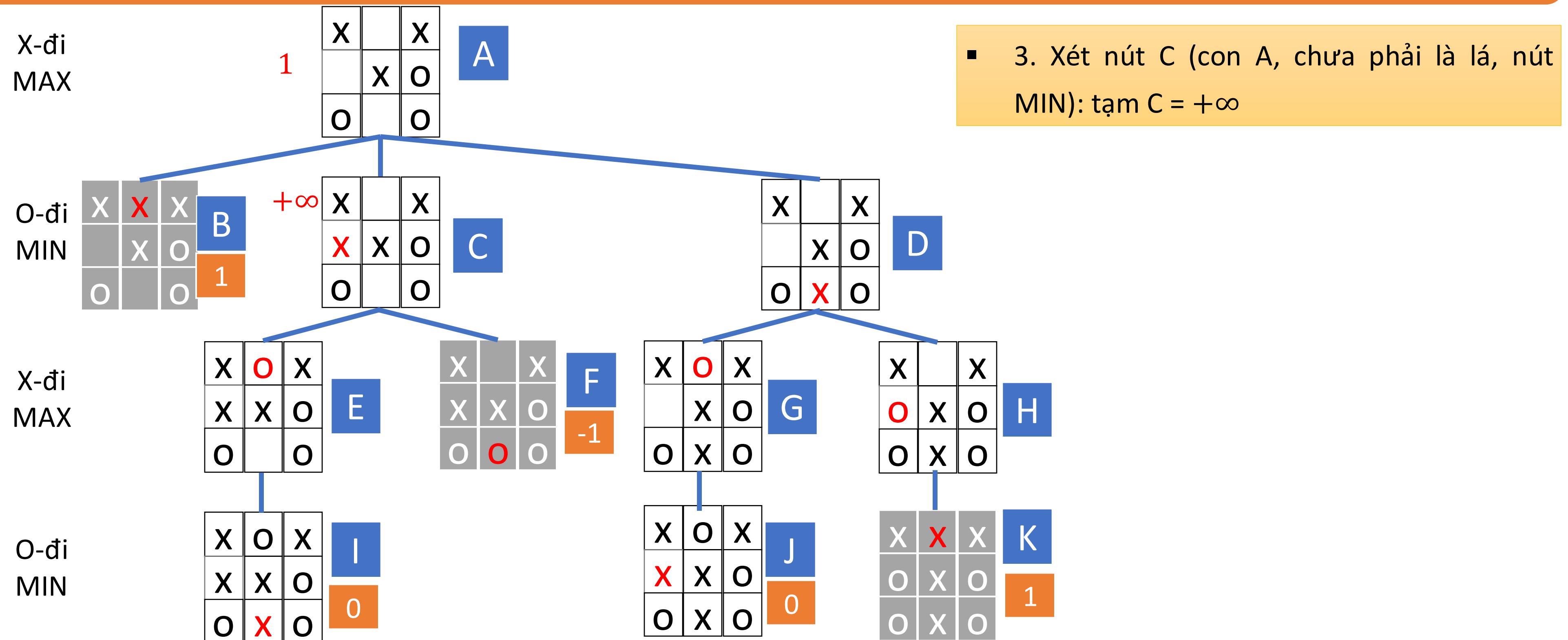
Chương 5. GIẢI THUẬT TÌM KIẾM LỜI GIẢI CHO TRÒ CHƠI

Kỹ thuật vét cạn (MIN-MAX)



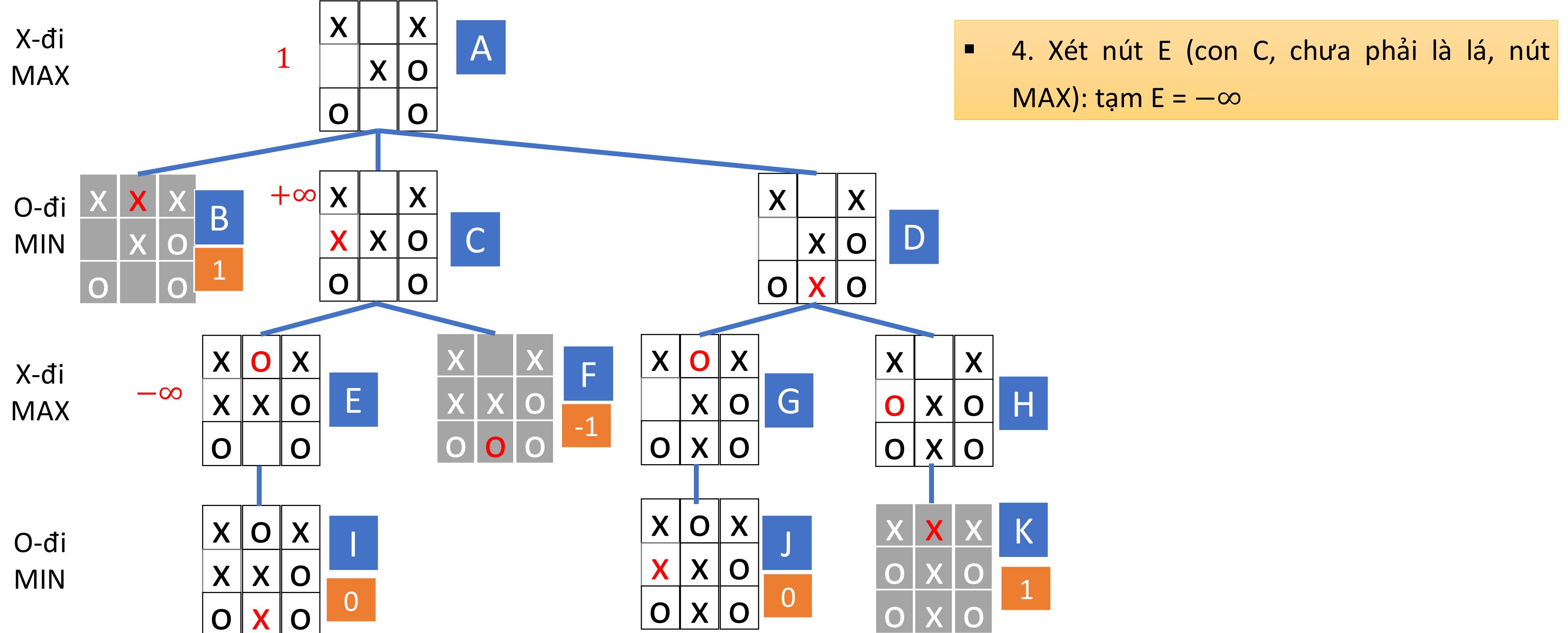
Chương 5. GIẢI THUẬT TÌM KIẾM LỜI GIẢI CHO TRÒ CHƠI

Kỹ thuật vét cạn (MIN-MAX)



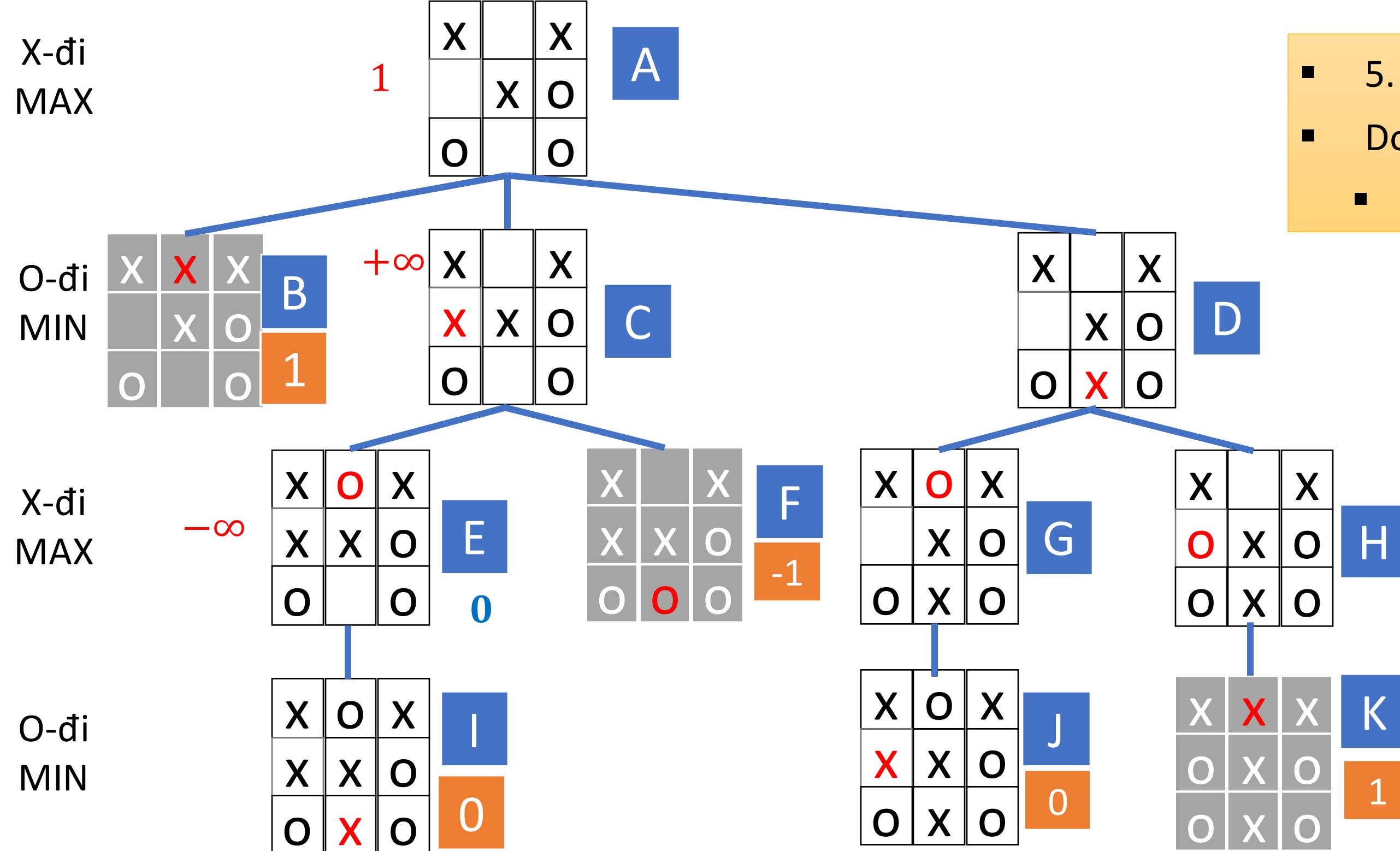
Chương 5. GIẢI THUẬT TÌM KIẾM LỜI GIẢI CHO TRÒ CHƠI

Kỹ thuật vét cạn (MIN-MAX)



Chương 5. GIẢI THUẬT TÌM KIẾM LỜI GIẢI CHO TRÒ CHƠI

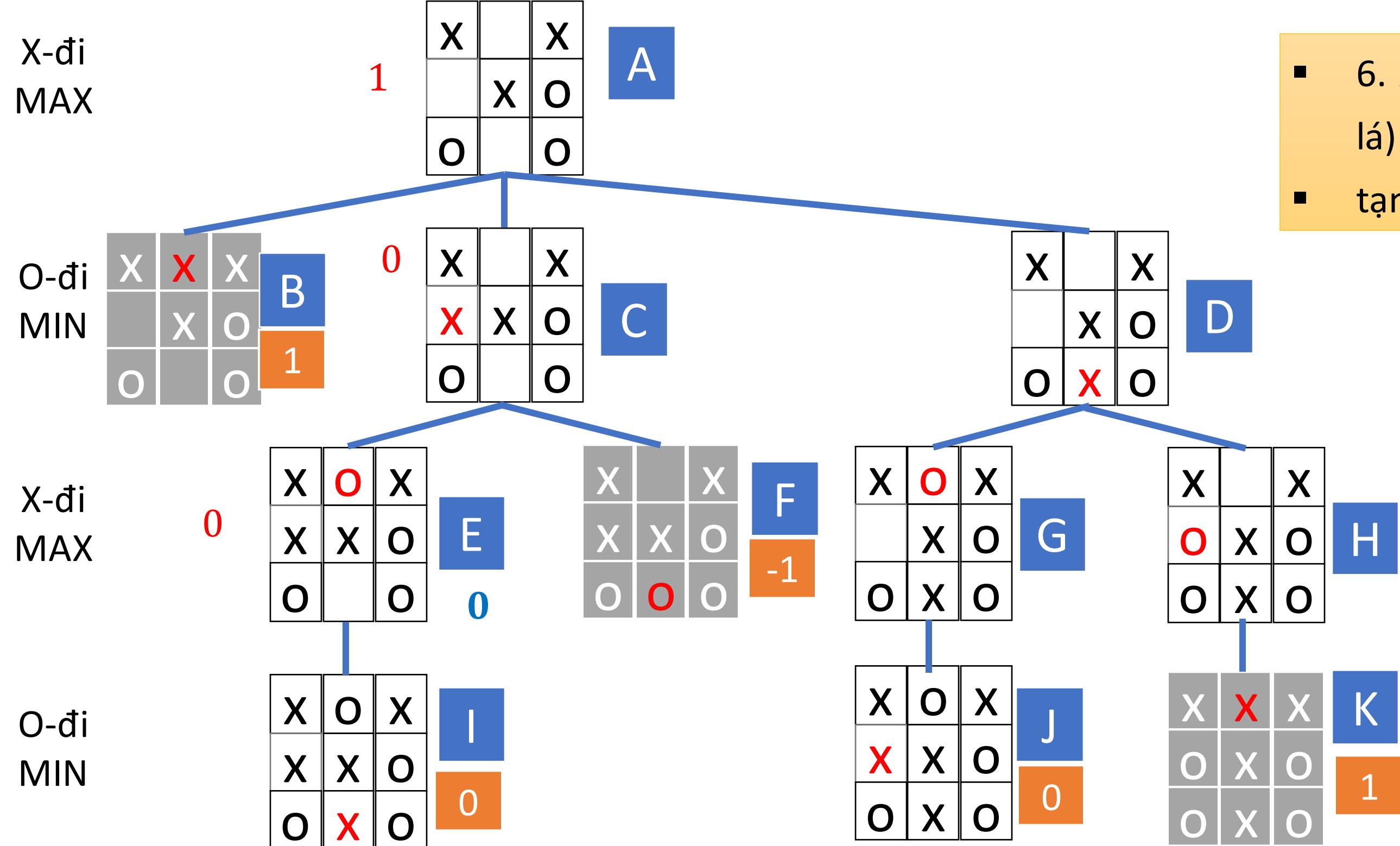
Kỹ thuật vét cạn (MIN-MAX)



- 5. Xét nút I (nút lá, con E): 0
- Do E có 1 nút con (I): xét xong
- valE = max(tạm E, 0) = 0

Chương 5. GIẢI THUẬT TÌM KIẾM LỜI GIẢI CHO TRÒ CHƠI

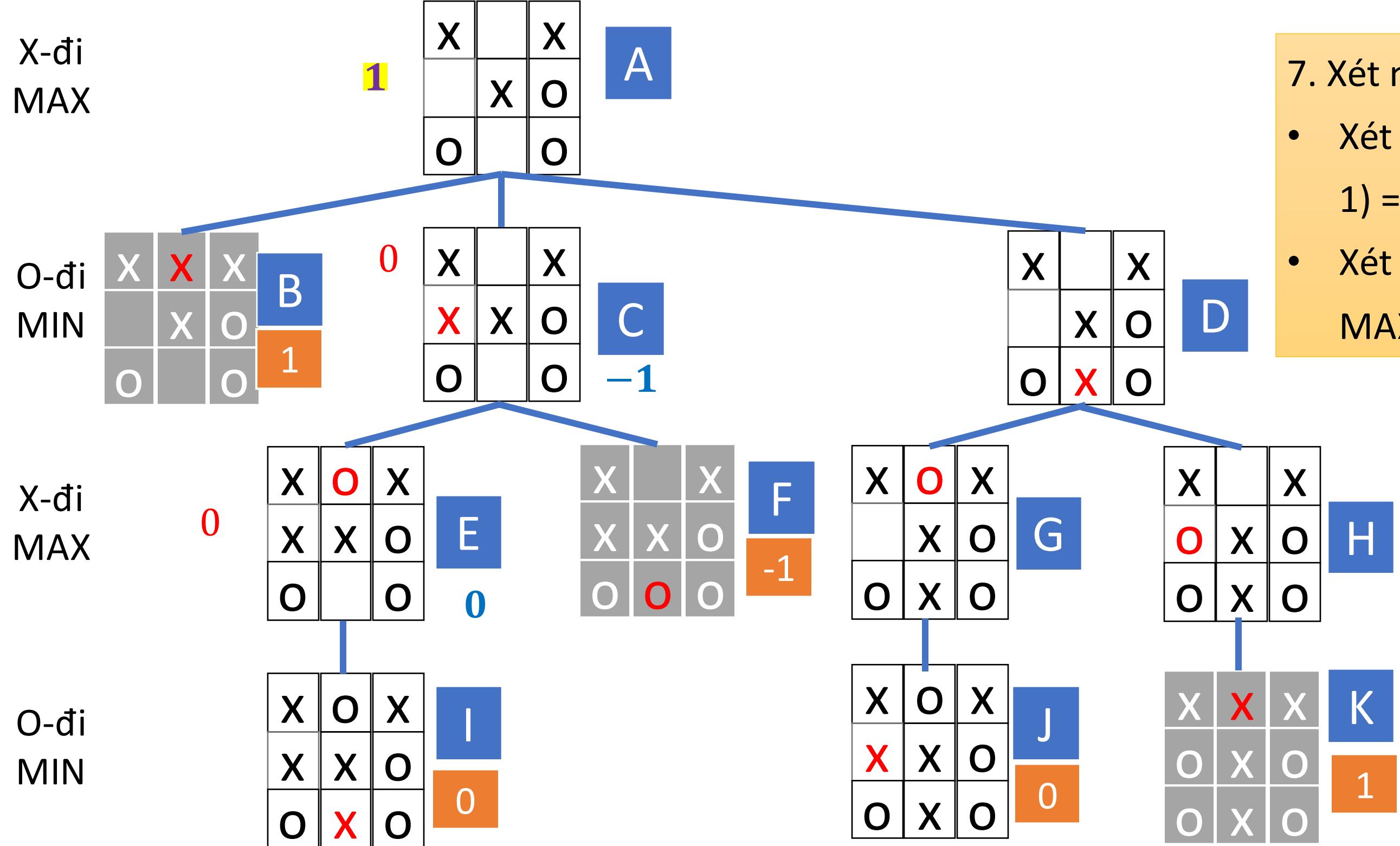
Kỹ thuật vét cạn (MIN-MAX)



- 6. Xét nút C (nút MIN, E con C, chưa là nút lá):
- tạm C = MIN(tạm C, 0) = 0

Chương 5. GIẢI THUẬT TÌM KIẾM LỜI GIẢI CHO TRÒ CHƠI

Kỹ thuật vét cạn (MIN-MAX)

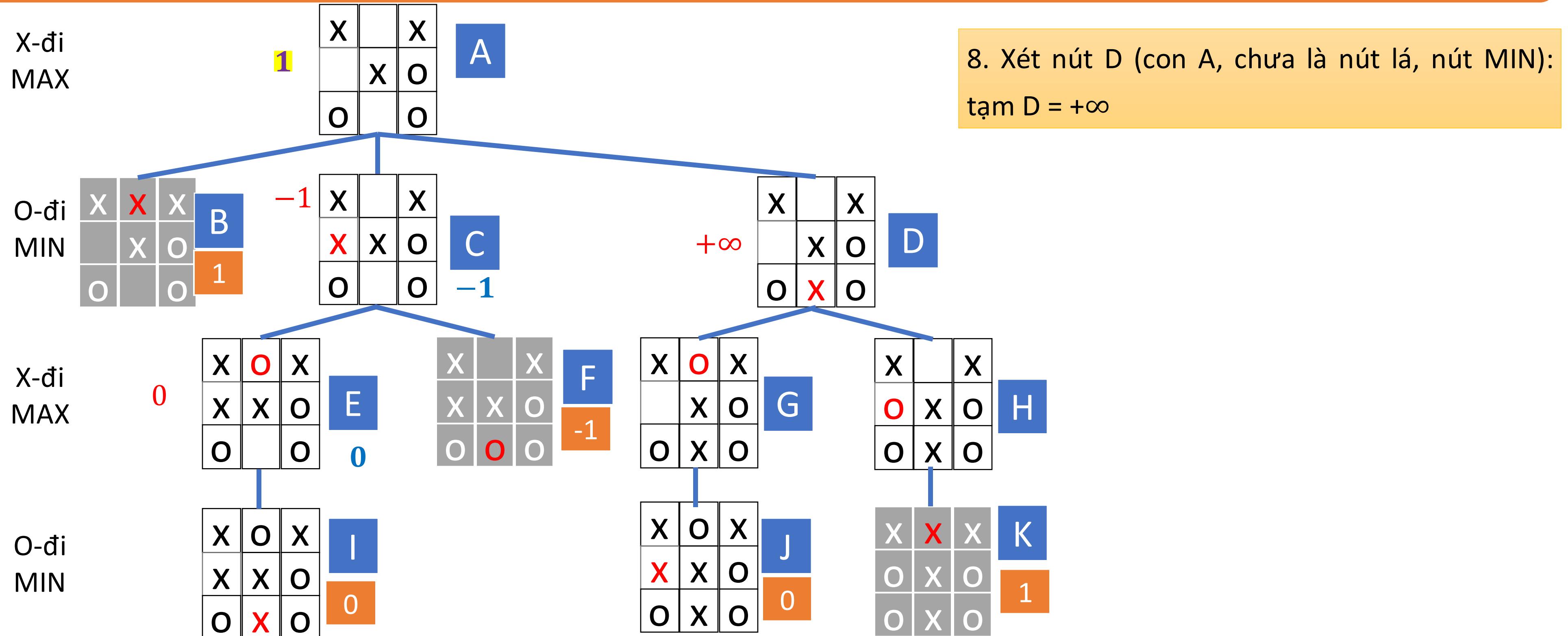


7. Xét nút F (nút lá, F con C): -1

- Xét nút cha của F=C, có giá trị = MIN(tạm C, -1) = -1
- Xét nút cha của C=A, có giá trị tạm A = MAX(tạm A, -1) = 1

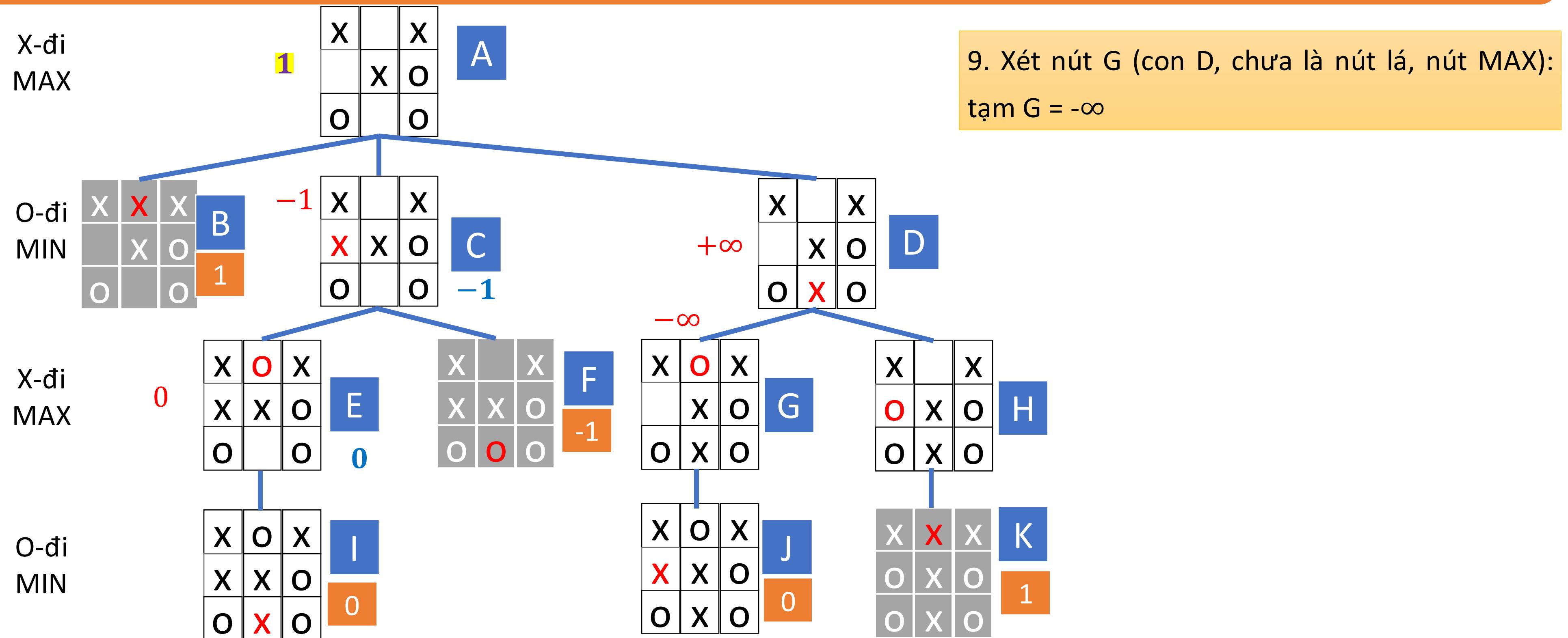
Chương 5. GIẢI THUẬT TÌM KIẾM LỜI GIẢI CHO TRÒ CHƠI

Trò chơi 2 người đối kháng



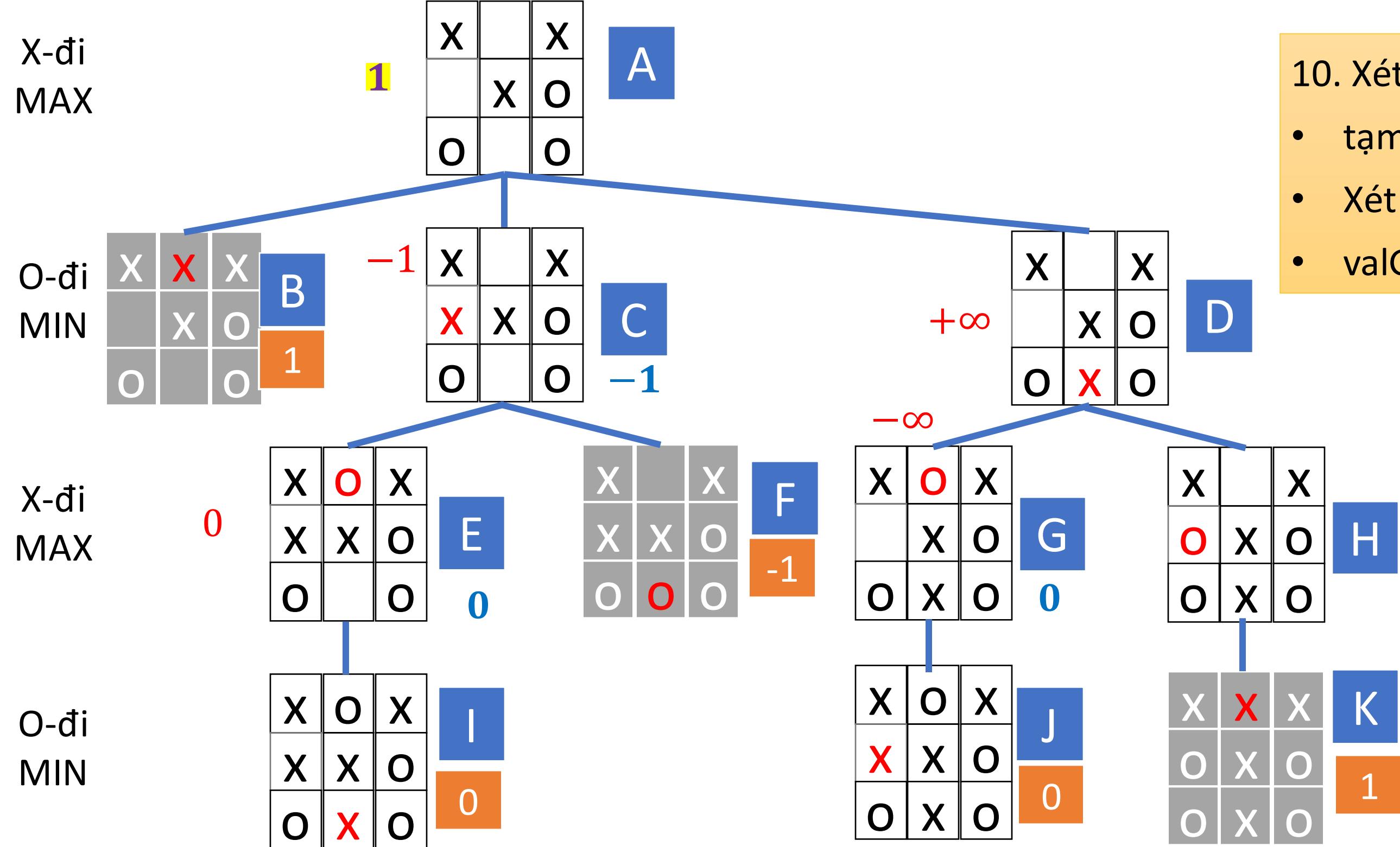
Chương 5. GIẢI THUẬT TÌM KIẾM LỜI GIẢI CHO TRÒ CHƠI

Kỹ thuật vét cạn (MIN-MAX)



Chương 5. GIẢI THUẬT TÌM KIẾM LỜI GIẢI CHO TRÒ CHƠI

Kỹ thuật vét cạn (MIN-MAX)

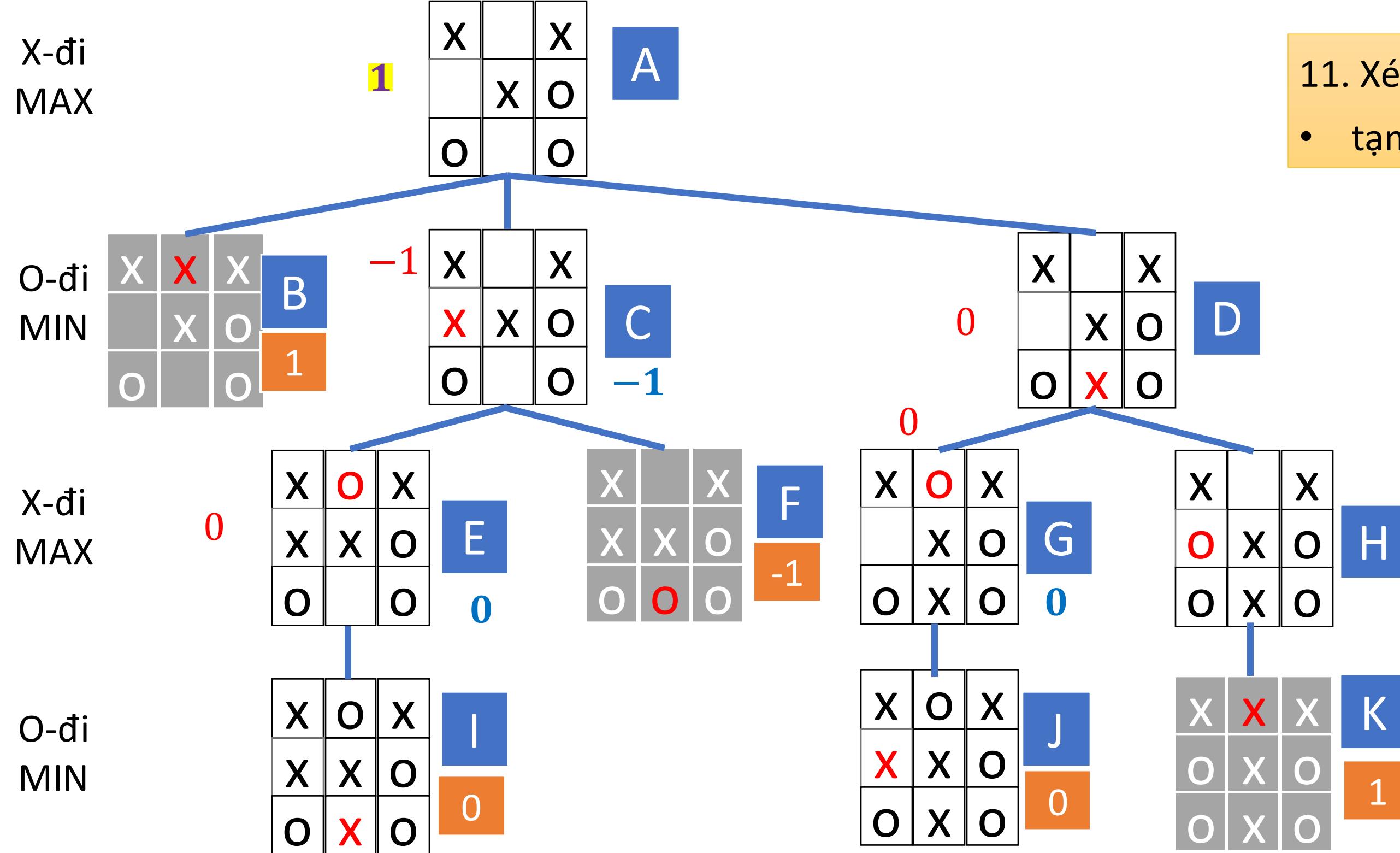


10. Xét nút J (con G, nút lá): 0

- tạmG=MAX(tạm G, 0) = 0
- Xét nút G cha J: xét xong (hết con)
- valG = tạm G

Chương 5. GIẢI THUẬT TÌM KIẾM LỜI GIẢI CHO TRÒ CHƠI

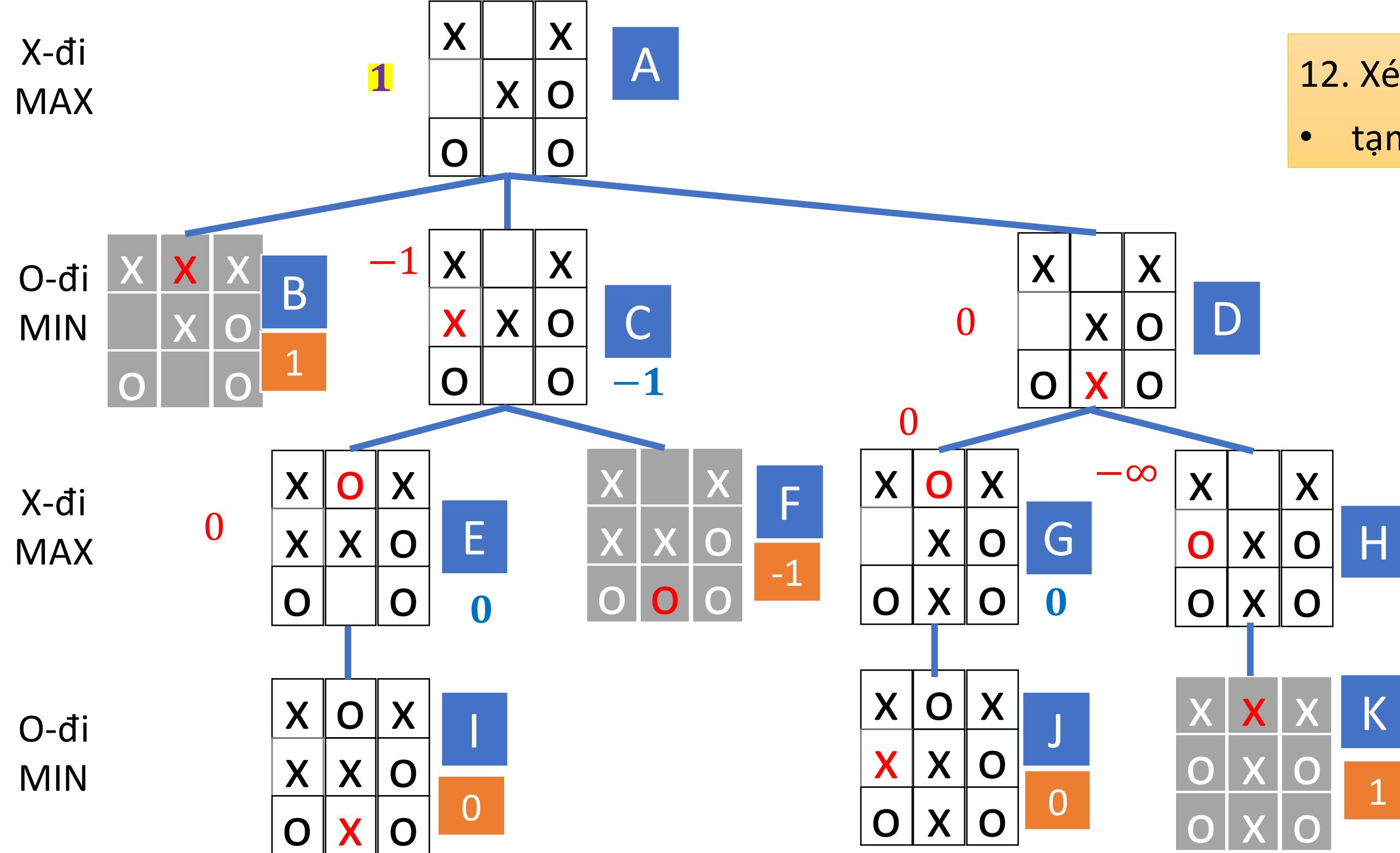
Kỹ thuật vét cạn (MIN-MAX)



11. Xét nút D (chưa là nút lá, nút MIN):
• tạmD=MIN(tạm D, 0) = 0

Chương 5. GIẢI THUẬT TÌM KIẾM LỜI GIẢI CHO TRÒ CHƠI

Kỹ thuật vét cạn (MIN-MAX)

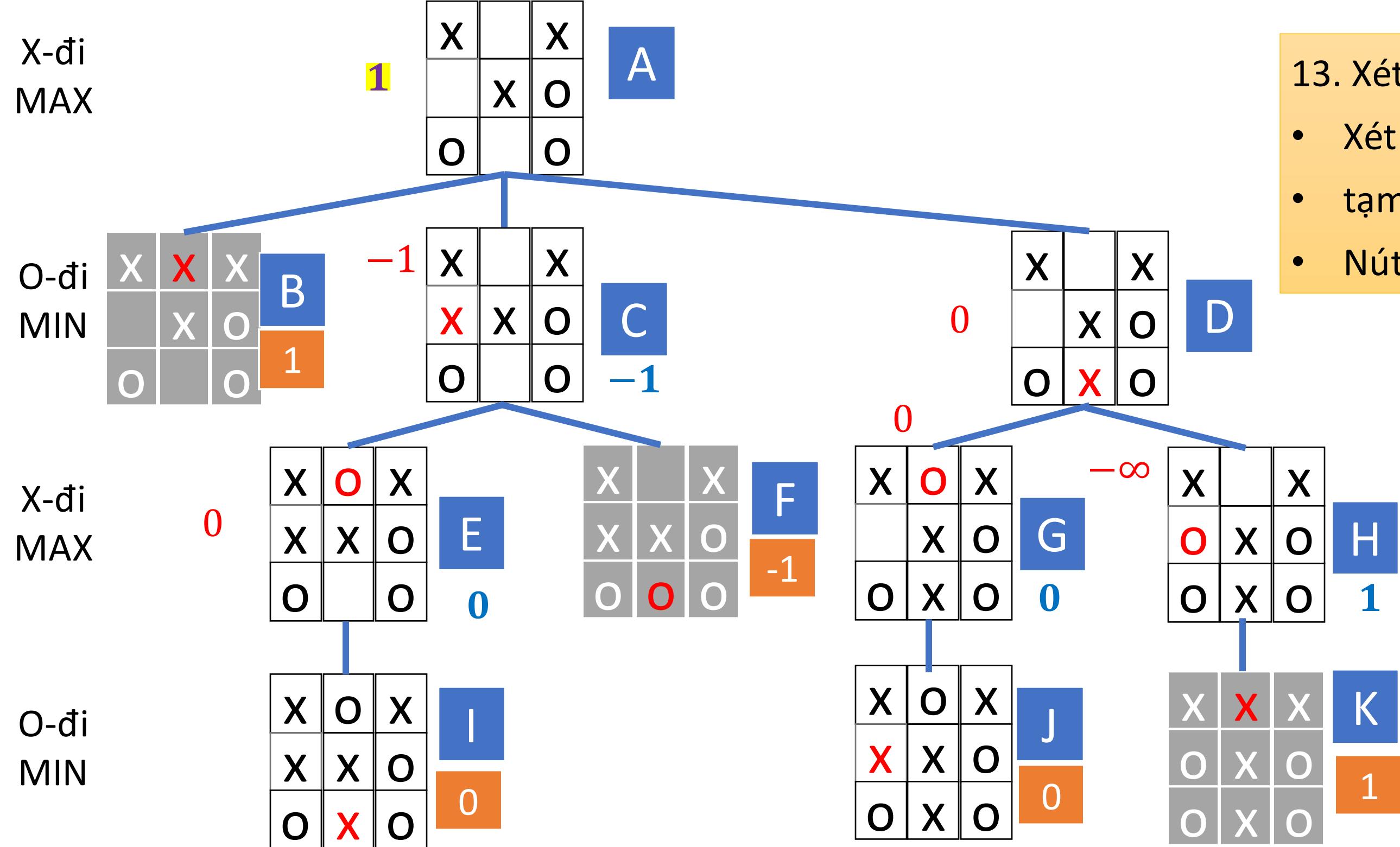


12. Xét nút H (chưa là nút lá, nút MAX):

- tạmH=-∞

Chương 5. GIẢI THUẬT TÌM KIẾM LỜI GIẢI CHO TRÒ CHƠI

Kỹ thuật vét cạn (MIN-MAX)

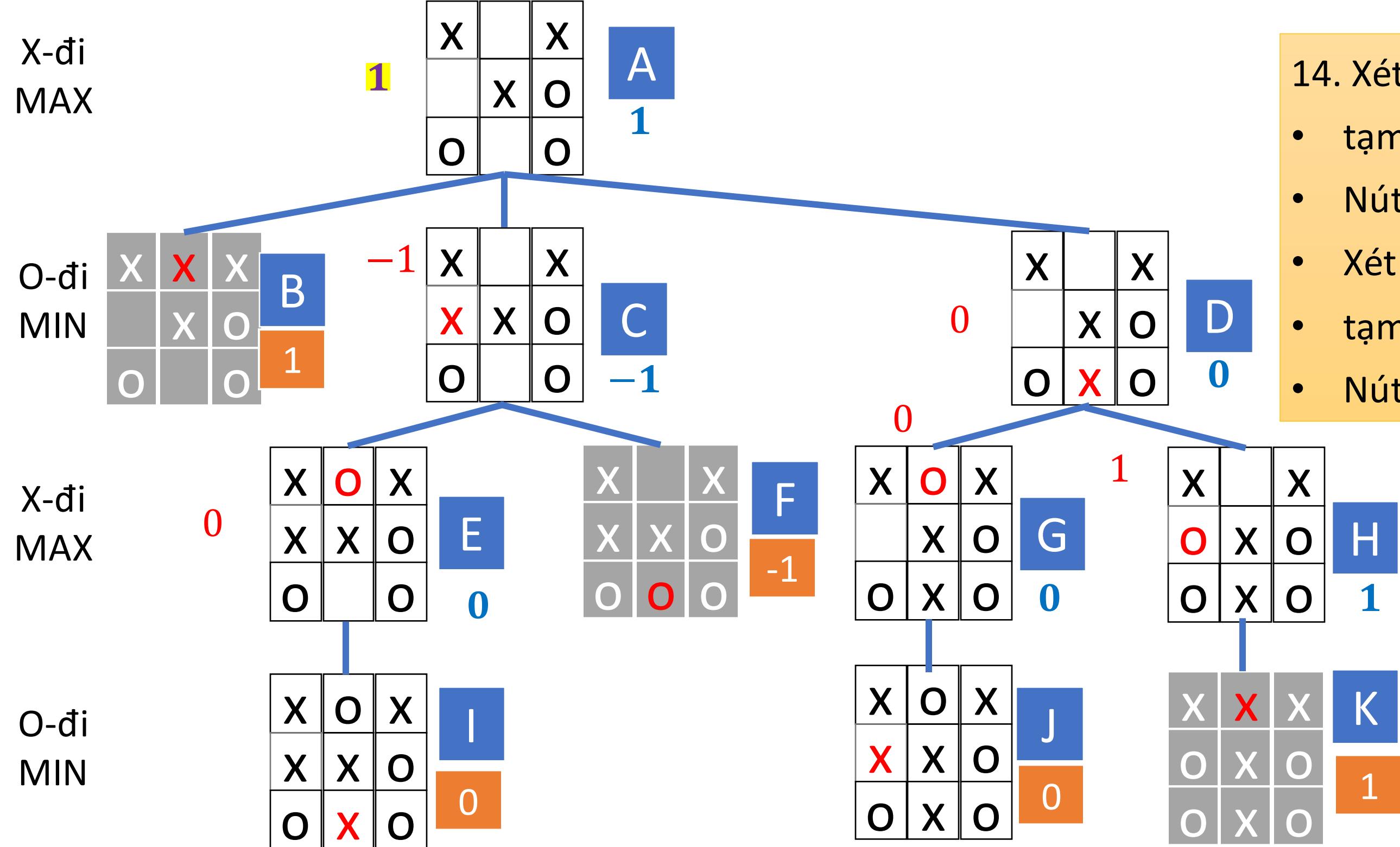


13. Xét nút K (nút lá): 1

- Xét nút H cha K:
- tạm H = MAX(tạm H, 1)=1
- Nút H hết con: val H = tạm H = 1

Chương 5. GIẢI THUẬT TÌM KIẾM LỜI GIẢI CHO TRÒ CHƠI

Kỹ thuật vét cạn (MIN-MAX)

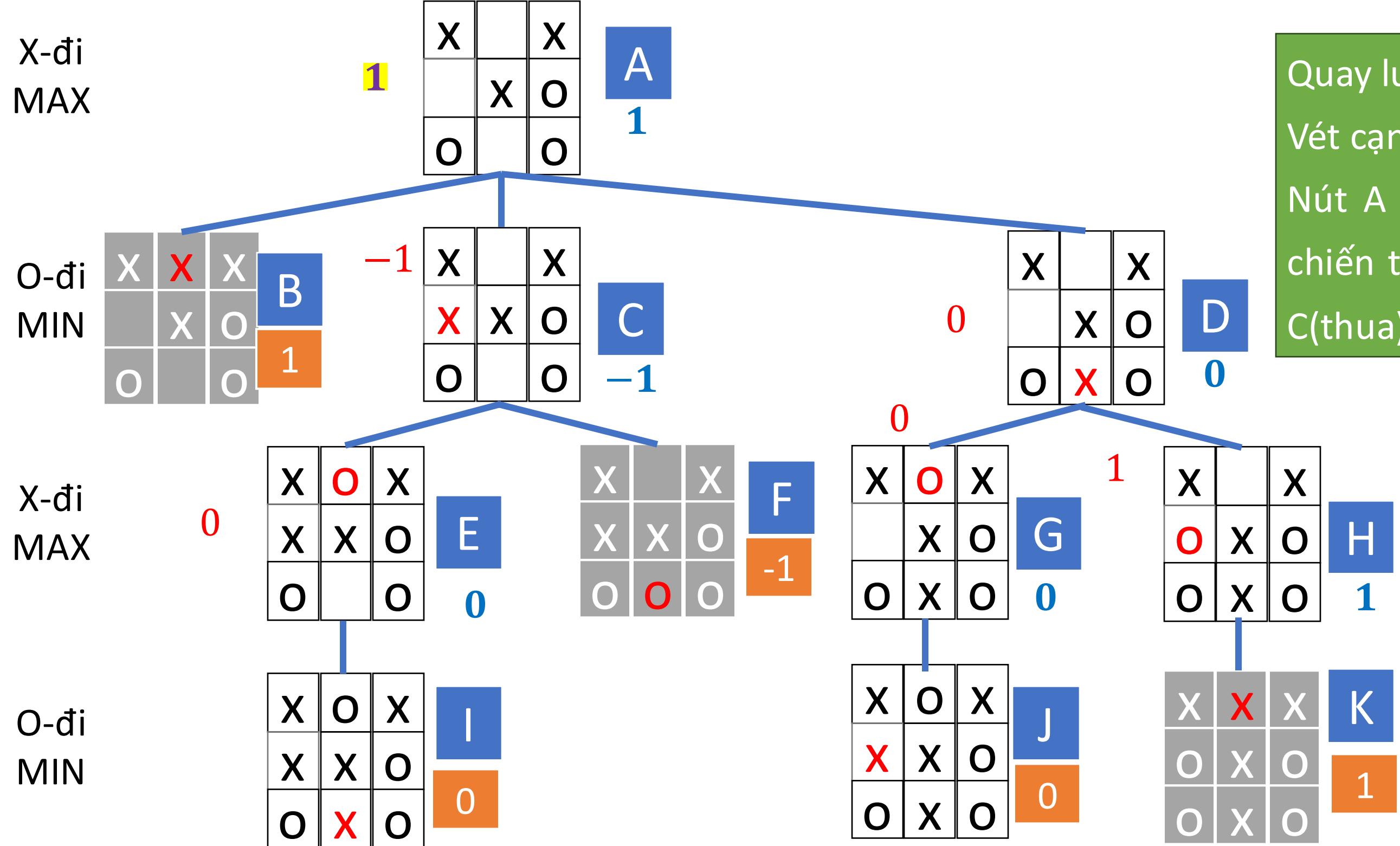


14. Xét nút D (chưa là lá, nút MIN):

- tạm D = MIN(tạm D, 1)=0
- Nút D hết con: val D = tạm D = 0
- Xét nút A (chưa là lá, nút MAX):
- tạm A = MAX(tạm A, 1) = 1
- Nút A hết con: val A = tạm A

Chương 5. GIẢI THUẬT TÌM KIẾM LỜI GIẢI CHO TRÒ CHƠI

Kỹ thuật vét cạn (MIN-MAX)



Quay lui: Xét nút lá → xét nút cha

Vét cạn: Xét hết tất cả các nút

Nút A ban đầu có giá trị là 1 (hướng đi để X chiến thắng). Đường đi theo B (thắng), D(hòa), C(thua)

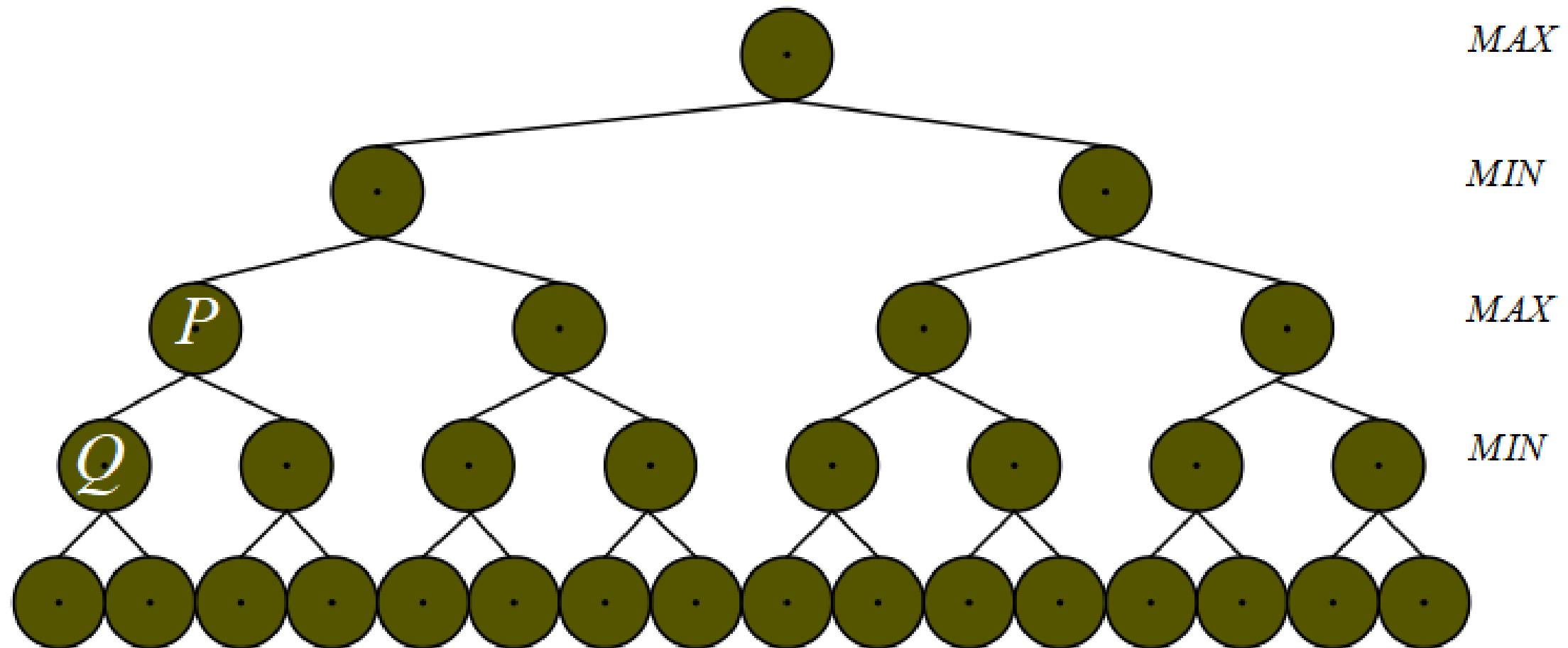
Từ kết quả này ta nhận thấy việc xét tất cả các nút là không cần thiết.

Có cách nào để hạn chế các nút/nhánh

Chương 5. GIẢI THUẬT TÌM KIẾM LỜI GIẢI CHO TRÒ CHƠI

Xém tỉa $\alpha - \beta$ (Alpha – Beta pruning)

- P là nút đang xét và đã có 1 số nút con đã xét và một số con chưa xét.
- Q là nút đang xét ($Q \subset P$)
- Gọi V_P, V_Q : giá trị tạm P,Q
- Nếu P là MAX thì Q là MIN (MIN –MAX đan xen nhau)



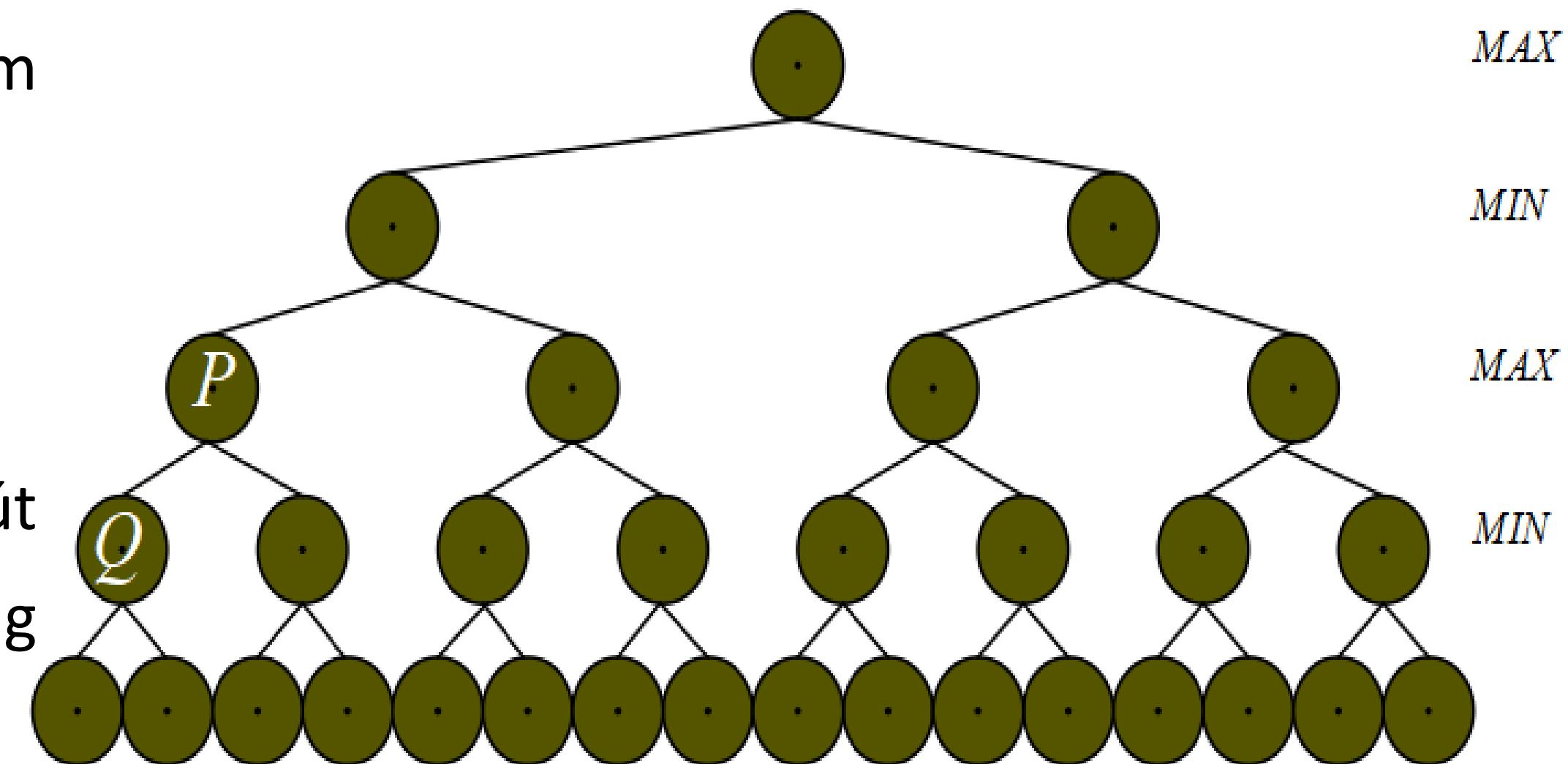
- Nếu $V_P \geq V_Q$: cắt tỉa các con chưa xét của Q (cắt tỉa α)
- Nếu P là nút MIN thì Q sẽ là nút MAX: $V_P \leq V_Q$: cắt tỉa các con chưa xét của Q (cắt tỉa β)

Chương 5. GIẢI THUẬT TÌM KIẾM LỜI GIẢI CHO TRÒ CHƠI

Xém tỉa $\alpha - \beta$ (Alpha – Beta pruning)

- Nếu $VP \geq VQ$: cắt tỉa các con chưa xét của Q (cắt tỉa α) – vì sao?

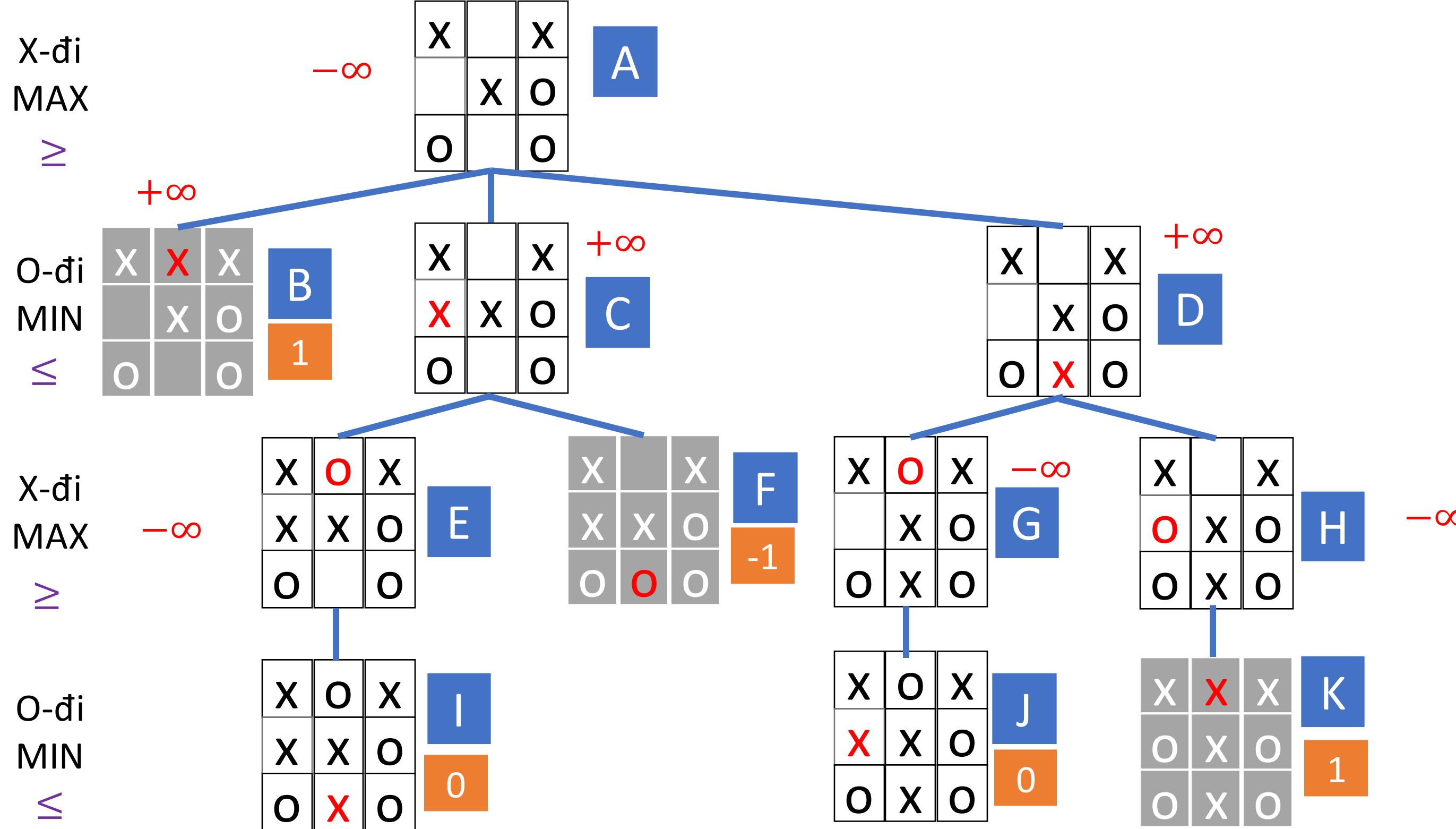
- Nếu xét hết tất cả các con của Q, ta sẽ tìm được giá trị của Q là V
- Vì Q là nút MIN: $V \leq VQi \leq VP$ $_{(*)}$
- Vì P là nút MAX(V_p, V)
- Từ (*) ta nhận thấy có xét tất cả các nút con của Q cũng không làm ảnh hưởng đến giá trị của nút P



- Nếu tất cả các nút con của Q đã xét hoặc bị cắt thì giá trị tạm của Q (V_Q) sẽ là giá trị của nút Q

Chương 5. GIẢI THUẬT TÌM KIẾM LỜI GIẢI CHO TRÒ CHƠI

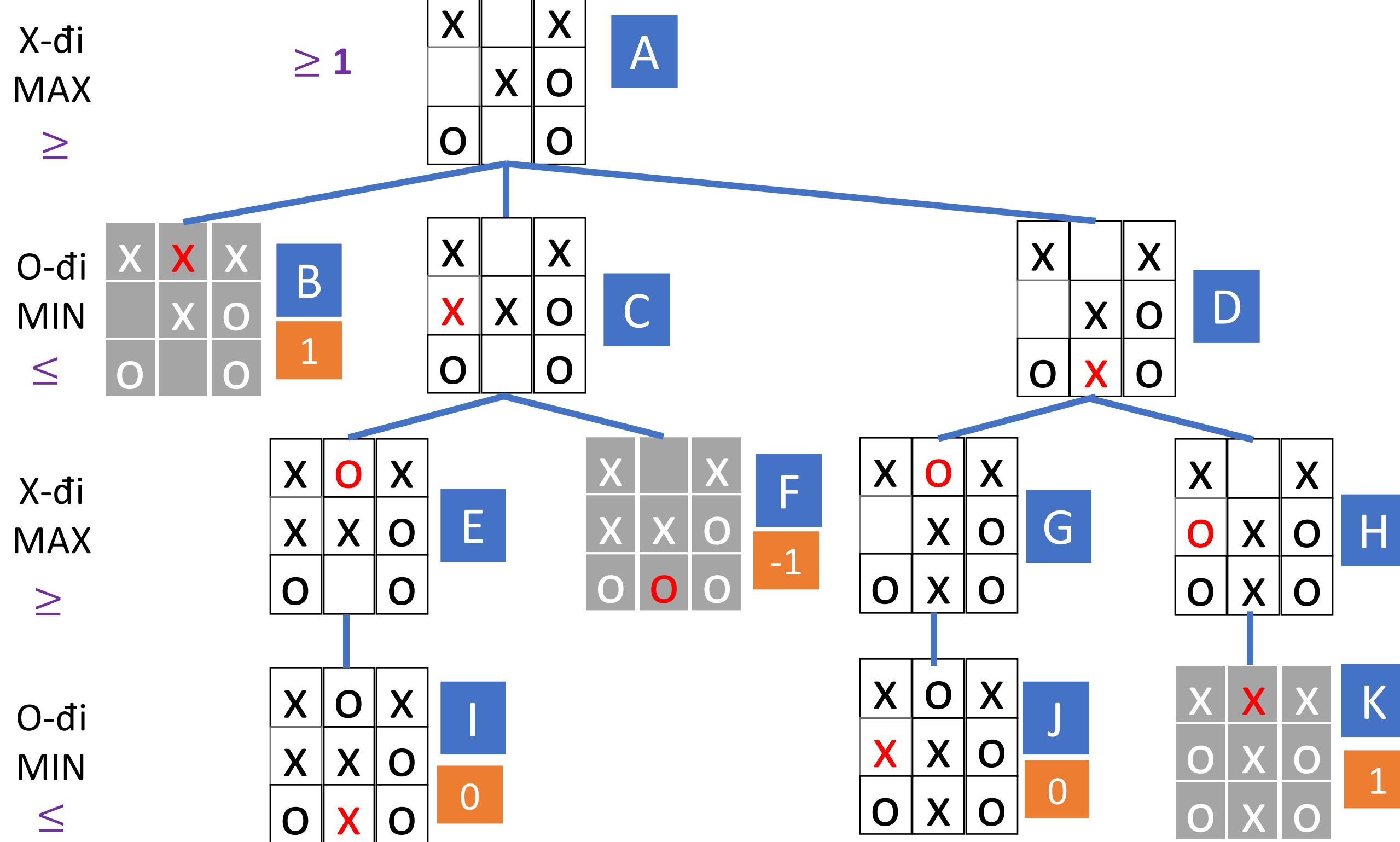
Xém tia $\alpha - \beta$ (Alpha – Beta pruning)



▪ Ghi chú: MAX \geq và MIN \leq

Chương 5. GIẢI THUẬT TÌM KIẾM LỜI GIẢI CHO TRÒ CHƠI

Xém tǐa $\alpha - \beta$ (Alpha – Beta pruning)

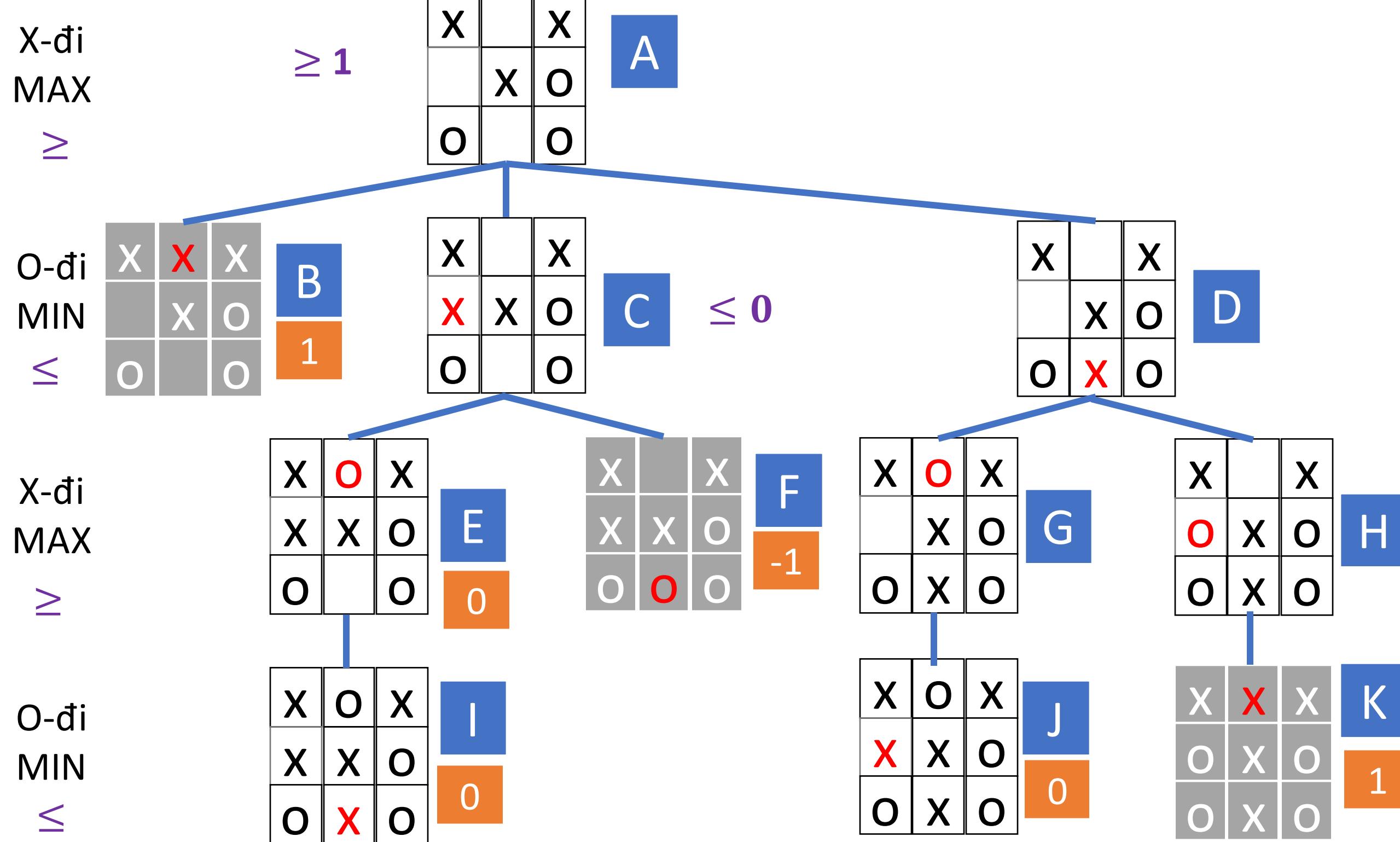


- Xét nút A có nút con B là nút lá = 1
nên tạm $A \geq 1$

- Ghi chú: $\text{MAX} \geq$ và $\text{MIN} \leq$

Chương 5. GIẢI THUẬT TÌM KIẾM LỜI GIẢI CHO TRÒ CHƠI

Xém tǐa $\alpha - \beta$ (Alpha – Beta pruning)

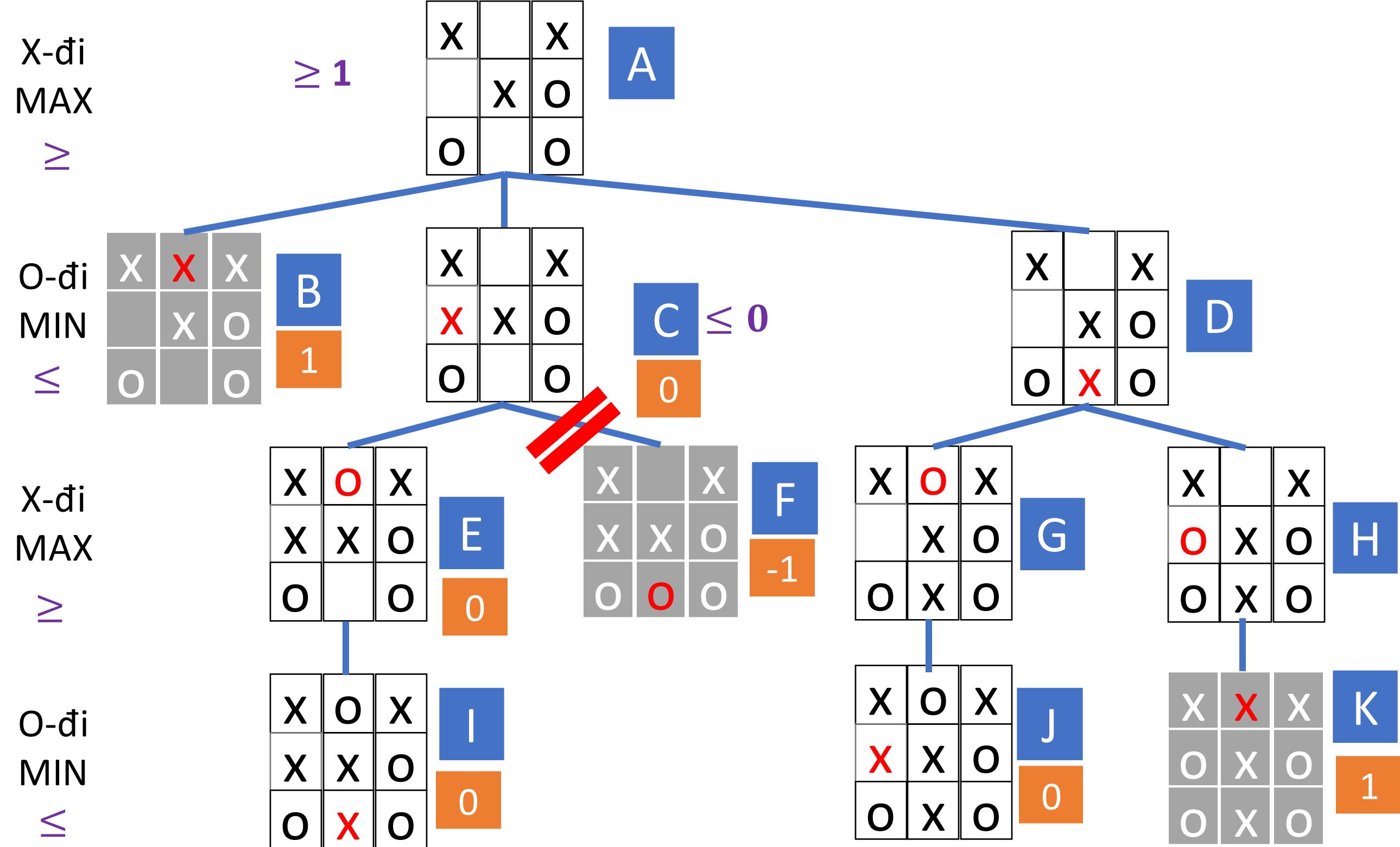


- Xét nút C có nút con E chưa là nút lá = 1 nên
- Xét nút I là con E có giá trị là 0 nên $\text{valE} \geq 0$
- **Nên tạm $C \leq 0$**

- Ghi chú: MAX \geq và MIN \leq

Chương 5. GIẢI THUẬT TÌM KIẾM LỜI GIẢI CHO TRÒ CHƠI

Xém tǐa $\alpha - \beta$ (Alpha – Beta pruning)

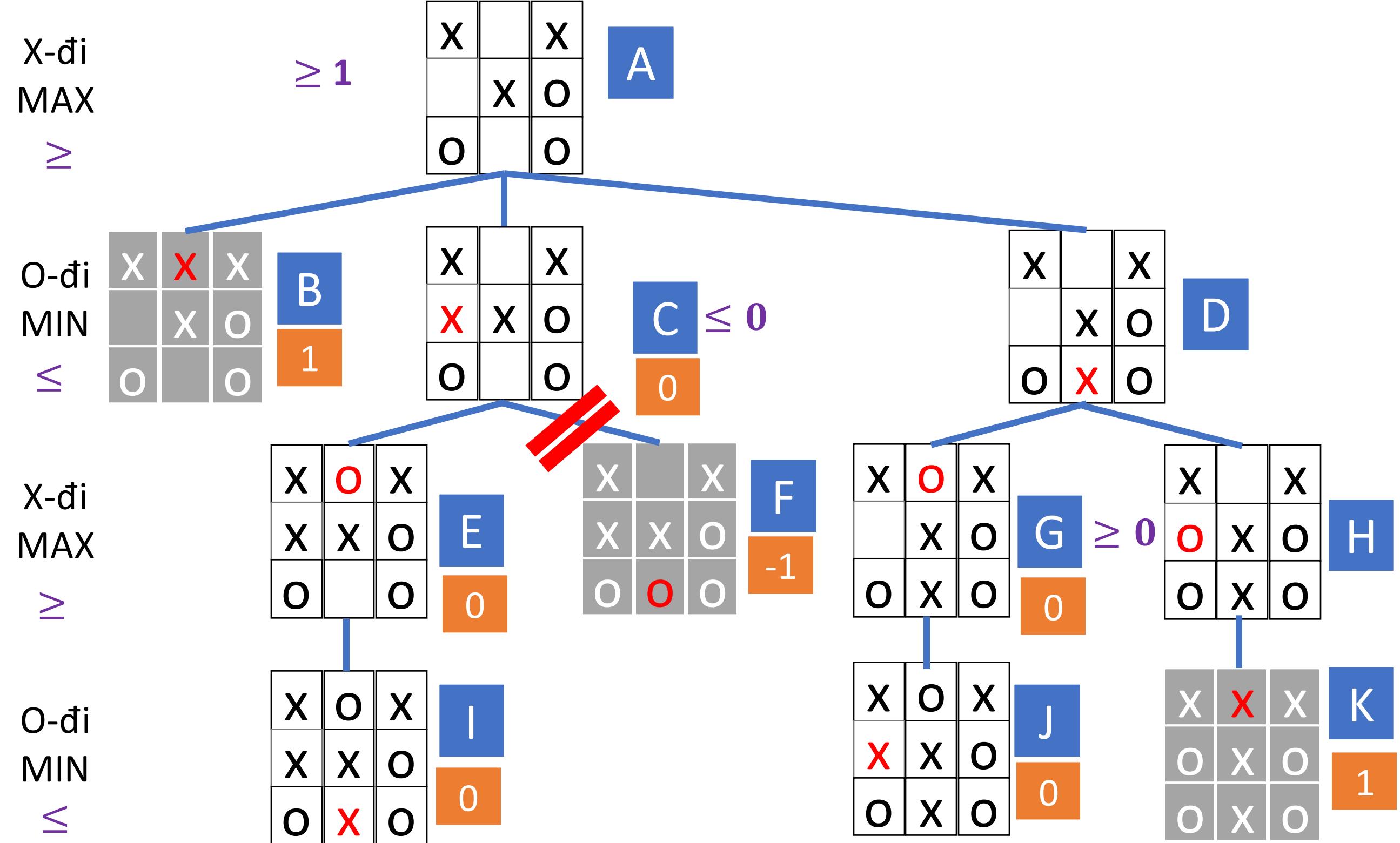


- A là nút MAX ($>=1$), C là MIN (≤ 0). Một số $X \leq 0$ và $X \geq 1$ thì không tồn tại X
- Nên nút còn lại của C (nút F) không cần xét
- Xén tǐa alpha
- Nút C sẽ nhận giá trị valC=0

- Ghi chú: MAX \geq và MIN \leq

Chương 5. GIẢI THUẬT TÌM KIẾM LỜI GIẢI CHO TRÒ CHƠI

Xém tǐa $\alpha - \beta$ (Alpha – Beta pruning)

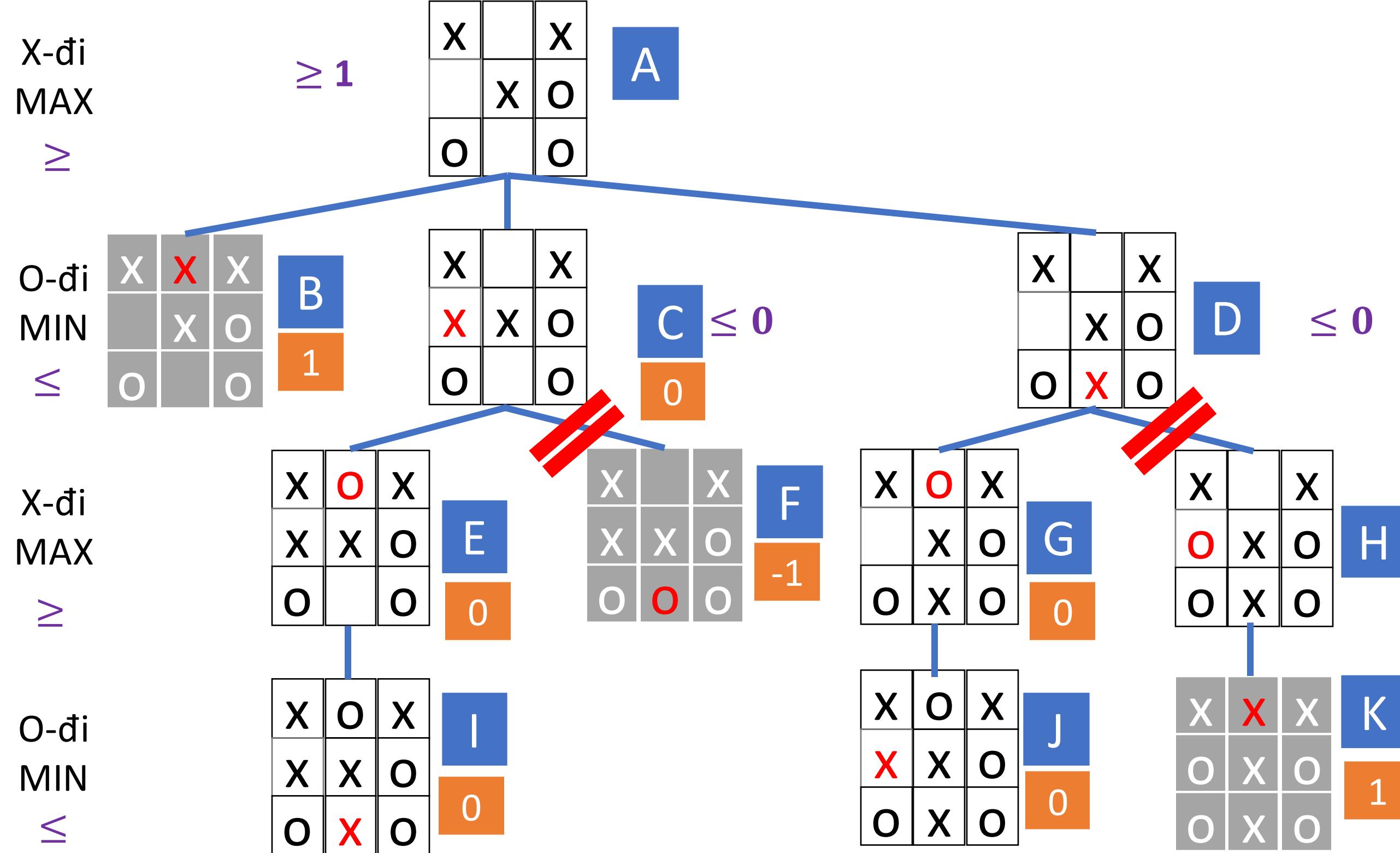


- Xét nút D có 2 con G và H chưa có giá trị nên
- Xét nút G, có con là nút J có giá trị là 0 nên tạm $G \geq 0$
- Vì nút G có 1 con đã xét nên $valG=0$

- Ghi chú: $MAX \geq$ và $MIN \leq$

Chương 5. GIẢI THUẬT TÌM KIẾM LỜI GIẢI CHO TRÒ CHƠI

Xém tǐa $\alpha - \beta$ (Alpha – Beta pruning)

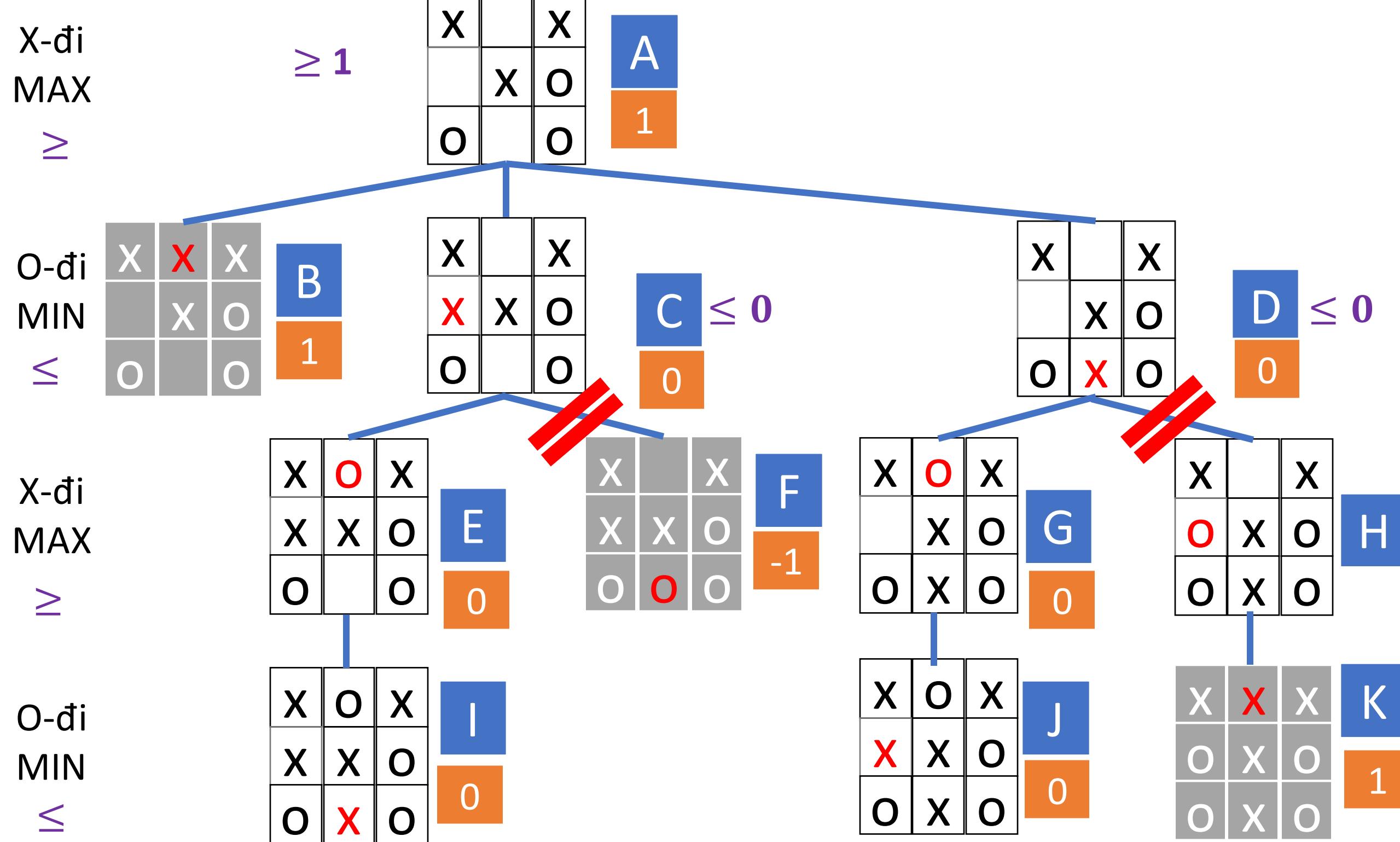


- D là nút MIN có tạm $D \leq 0$
- Nút A là nút MAX ≥ 1
- D là nút MIN ≤ 0
- $X \leq 0$ và $X \geq 1$ thì $\exists! X$
- Cắt tǐa nhánh con của D là H

- Ghi chú: MAX \geq và MIN \leq

Chương 5. GIẢI THUẬT TÌM KIẾM LỜI GIẢI CHO TRÒ CHƠI

Xém tǐa $\alpha - \beta$ (Alpha – Beta pruning)



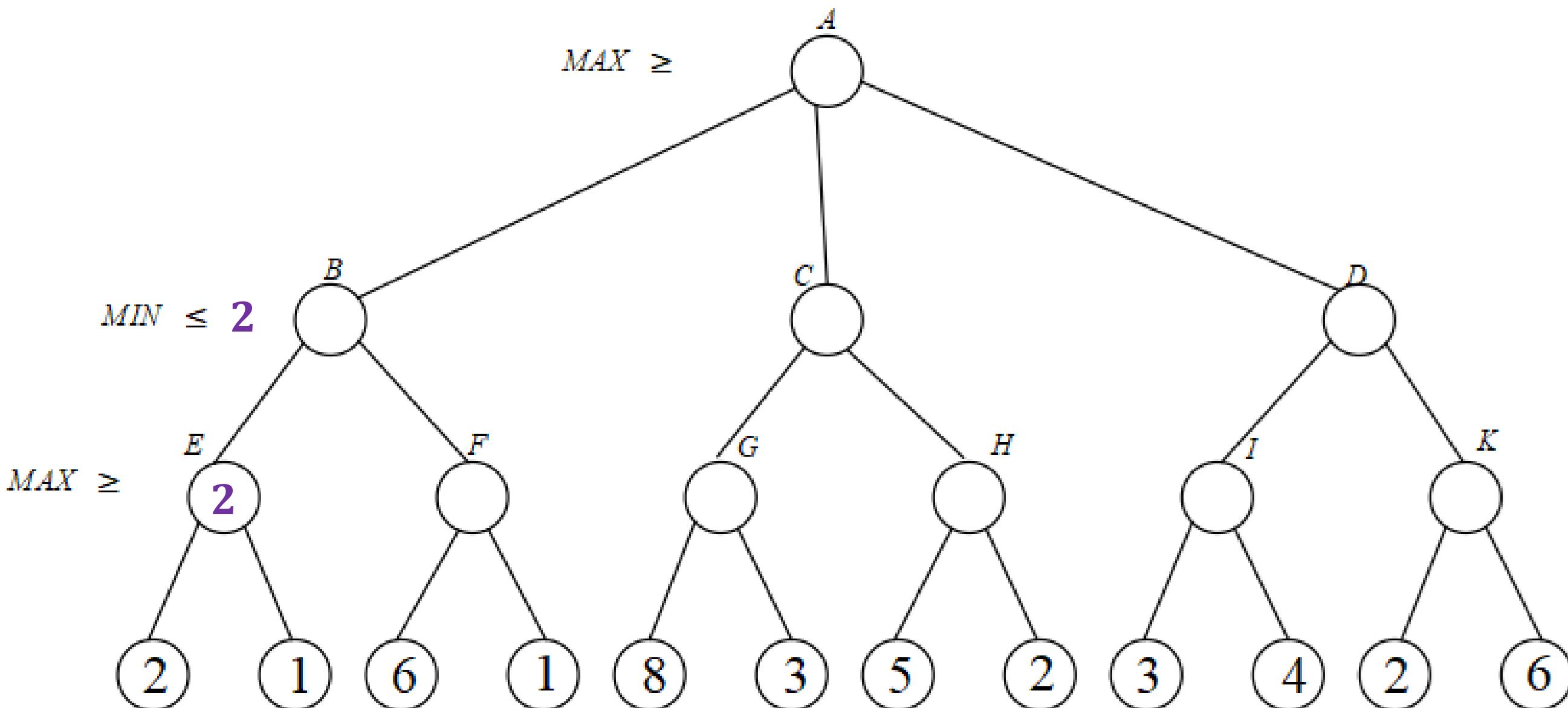
- Nút D có các nhánh đã xét nên valD= 0
- Các nút con B,C,D của D đã xét. MAX(tamA, valD) =1 nên valA=1

- Kết quả nút A so với PP vét cạn là không đổi nhưng số trường hợp phải xét ít hơn nhiều so với PP trước.

- Ghi chú: MAX \geq và MIN \leq

Chương 5. GIẢI THUẬT TÌM KIẾM LỜI GIẢI CHO TRÒ CHƠI

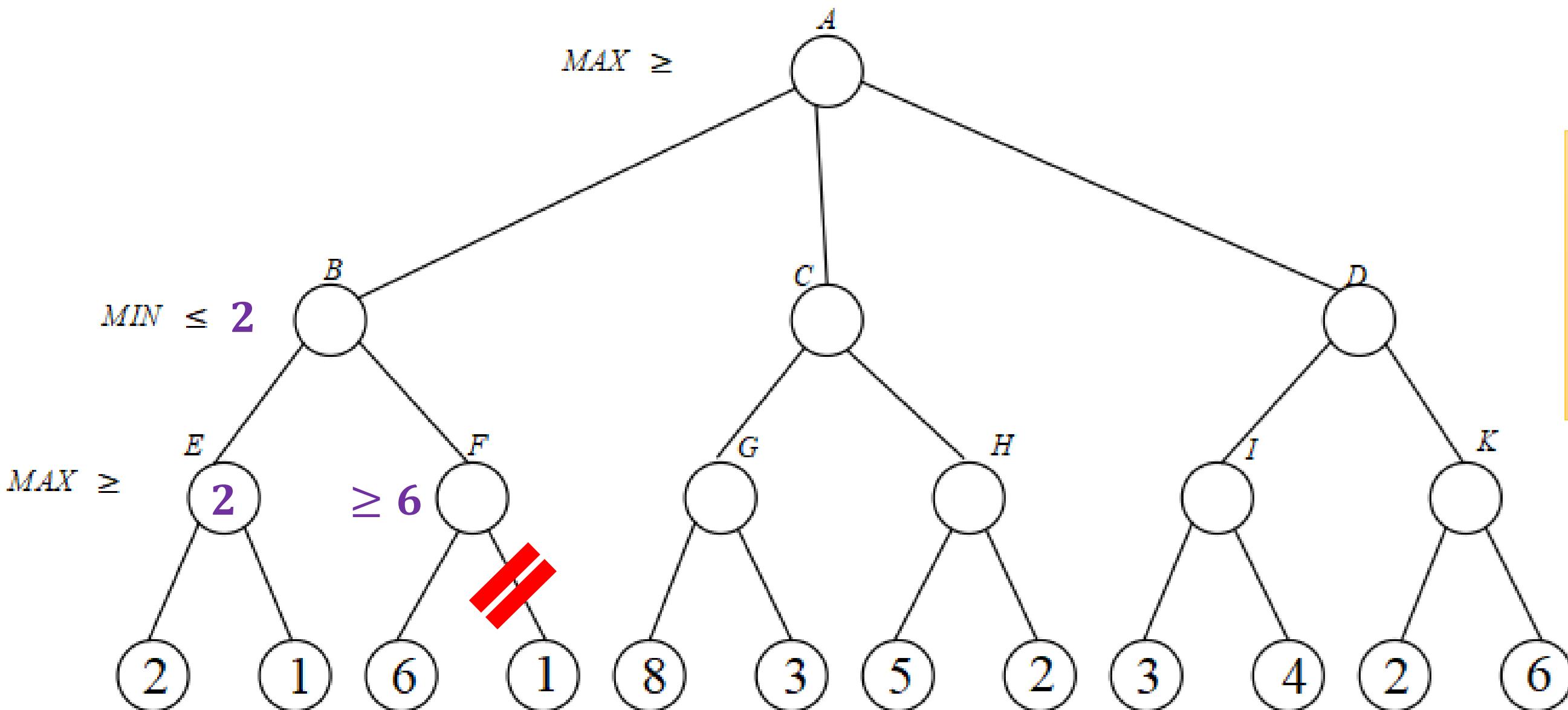
Xém tǐa $\alpha - \beta$ (Alpha – Beta pruning)



- Xét nút E có 2 nút con đã biết nên $valE = MAX(2,1)=2$
- Nút B có tạm B: ≤ 2

Chương 5. GIẢI THUẬT TÌM KIẾM LỜI GIẢI CHO TRÒ CHƠI

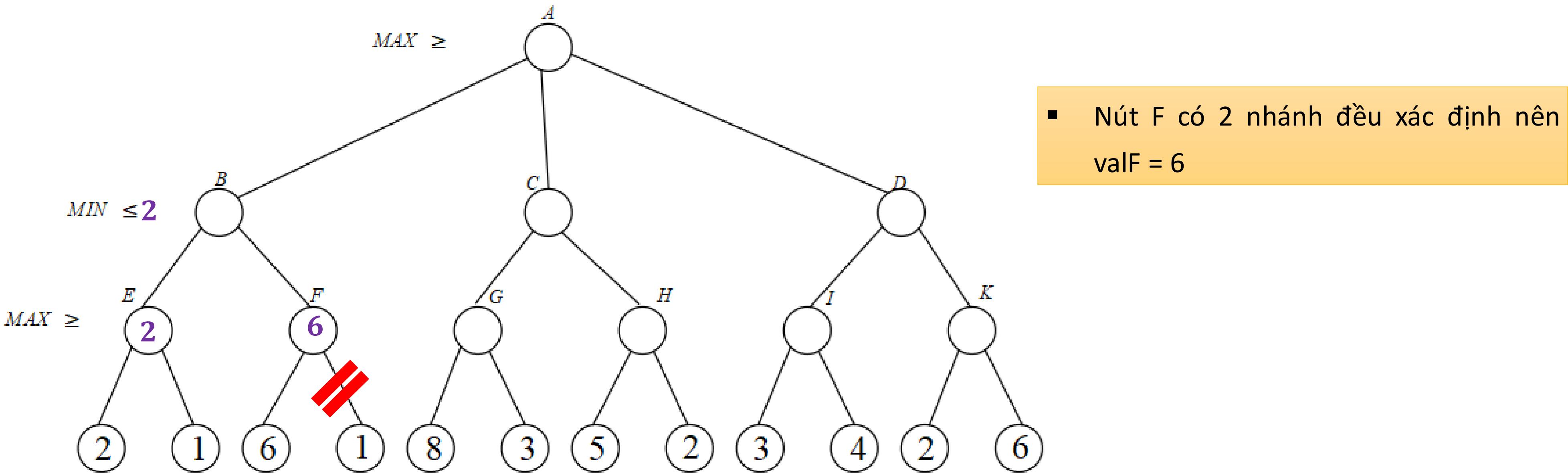
Xém tǐa $\alpha - \beta$ (Alpha – Beta pruning)



- Xét nút F có nút con trái =6 nên tạm $F >= 6$
- $\nexists! X (X \geq 6 \text{ và } X \leq 2)$
- *cắt tỉa nhánh con của F bên phải*

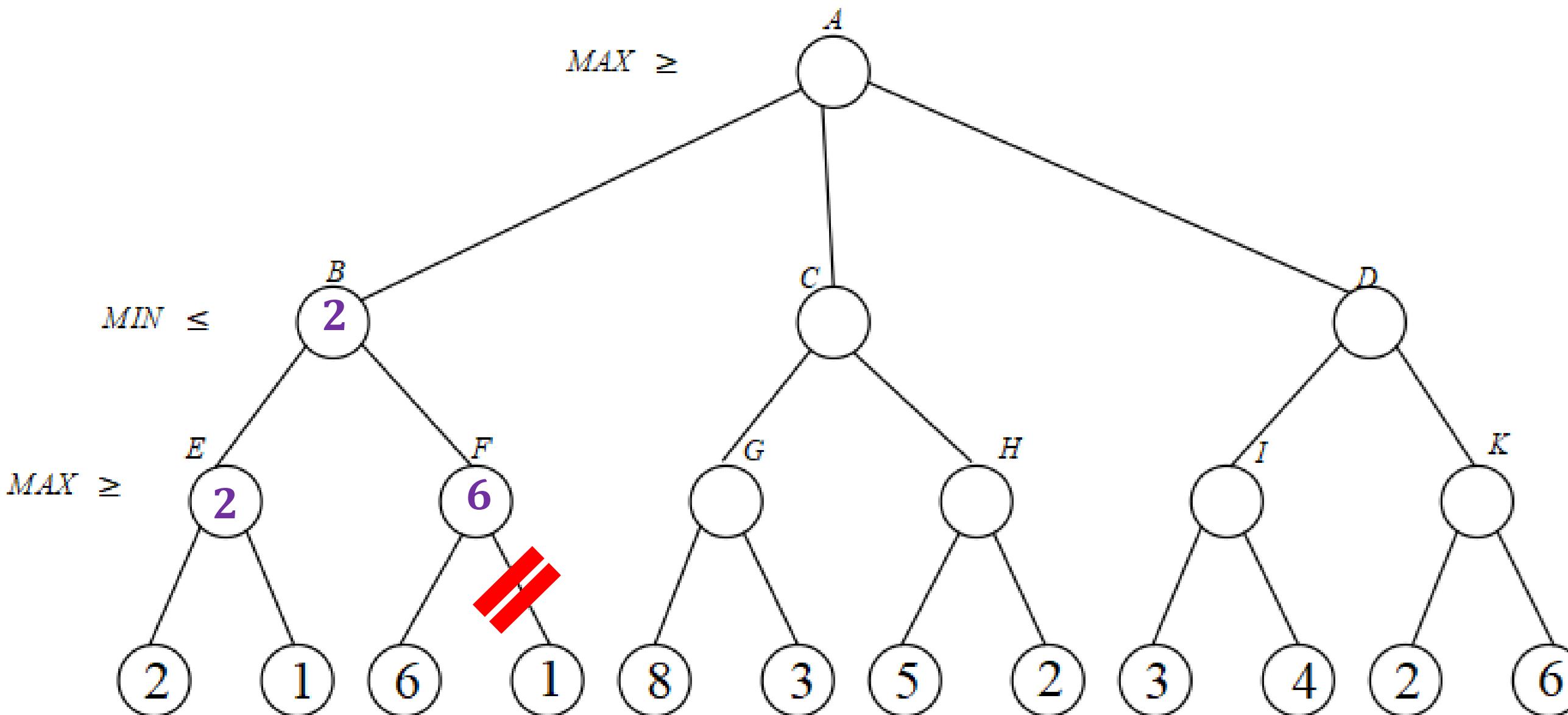
Chương 5. GIẢI THUẬT TÌM KIẾM LỜI GIẢI CHO TRÒ CHƠI

Xém tǐa $\alpha - \beta$ (Alpha – Beta pruning)



Chương 5. GIẢI THUẬT TÌM KIẾM LỜI GIẢI CHO TRÒ CHƠI

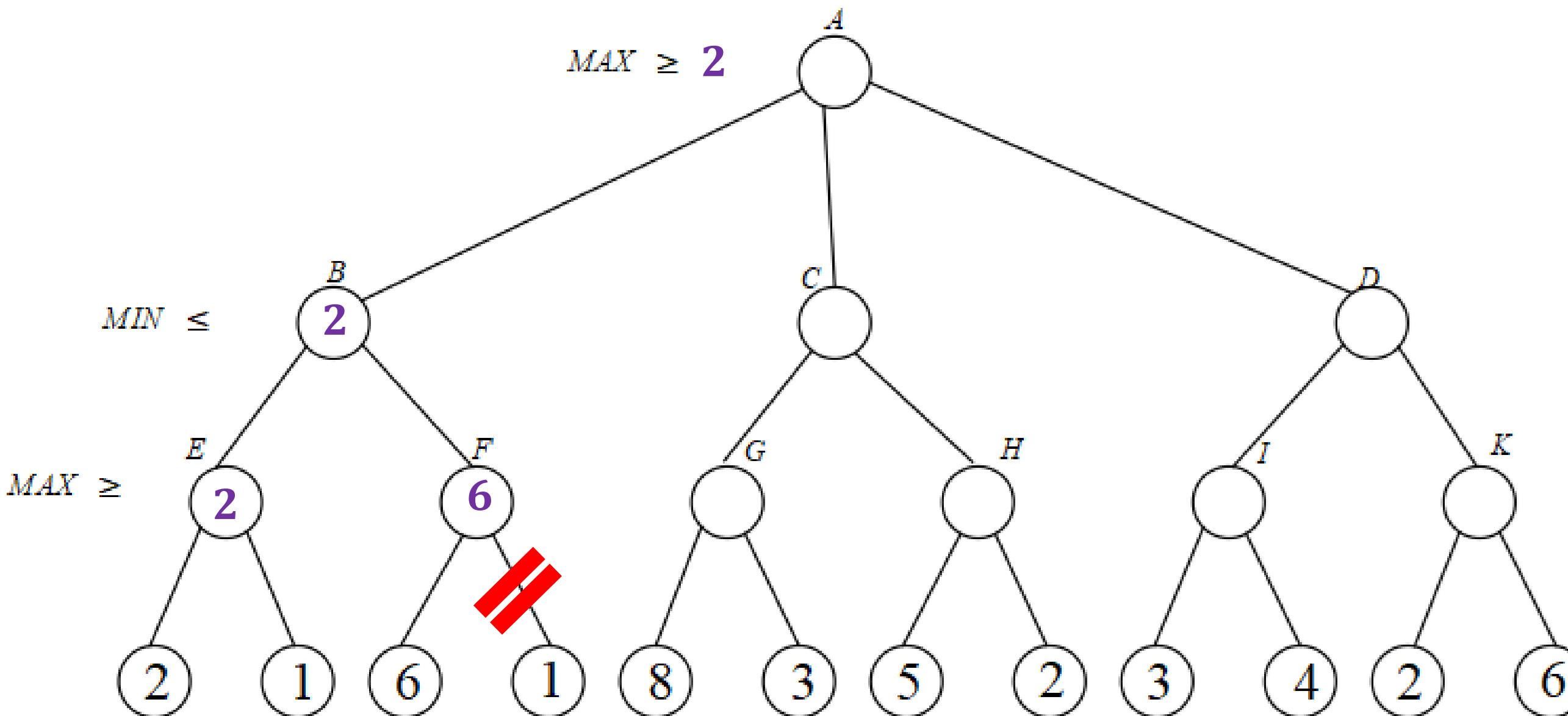
Xém tǐa $\alpha - \beta$ (Alpha – Beta pruning)



- Nút F có 2 nhánh đều xác định nên $valF = 6$
- Nút B có 2 nhánh đều đã xét nên $valB = 2$

Chương 5. GIẢI THUẬT TÌM KIẾM LỜI GIẢI CHO TRÒ CHƠI

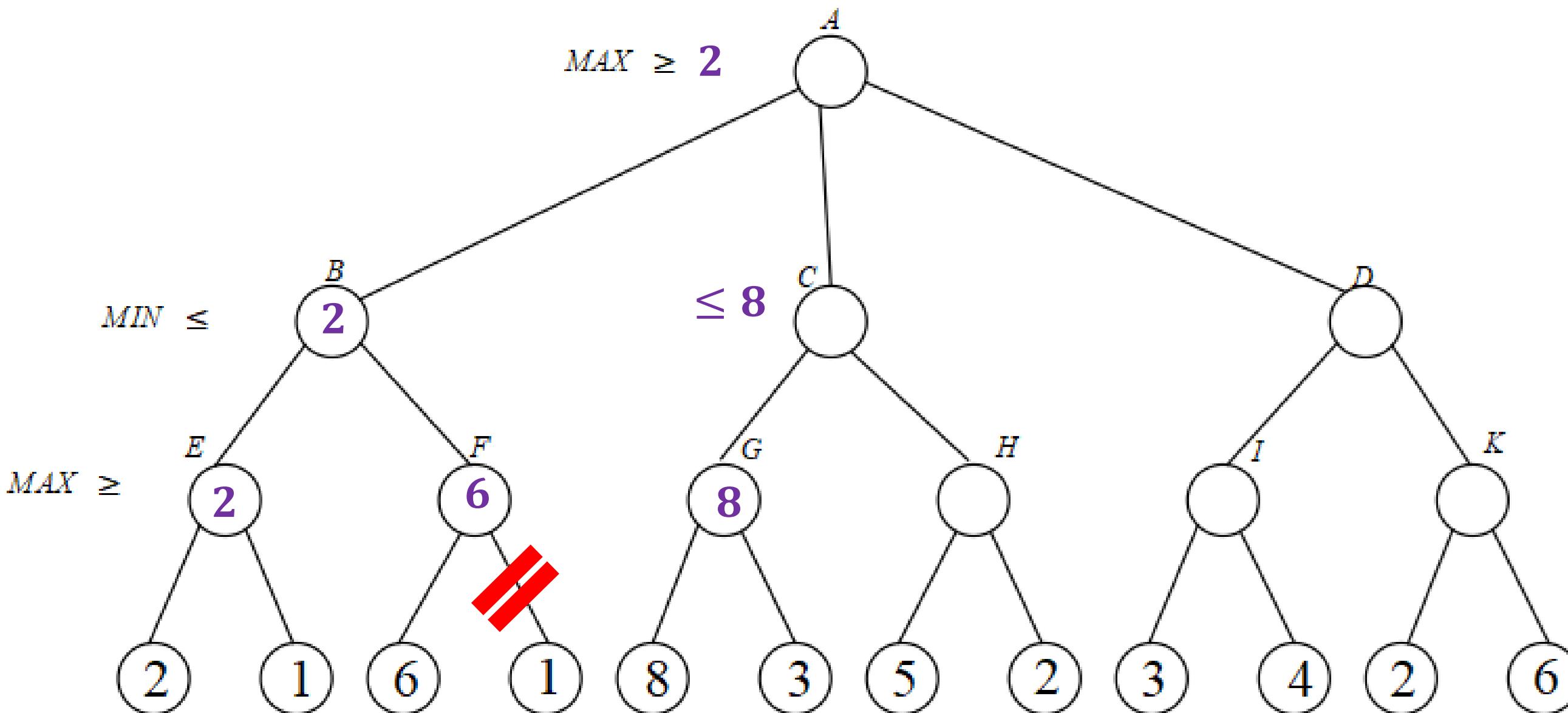
Xém tǐa $\alpha - \beta$ (Alpha – Beta pruning)



- Nút F có 2 nhánh đều xác định nên $valF = 6$
- Nút B có 2 nhánh đều đã xét nên $valB = 2$
- Nút A có tạm $A \geq 2$

Chương 5. GIẢI THUẬT TÌM KIẾM LỜI GIẢI CHO TRÒ CHƠI

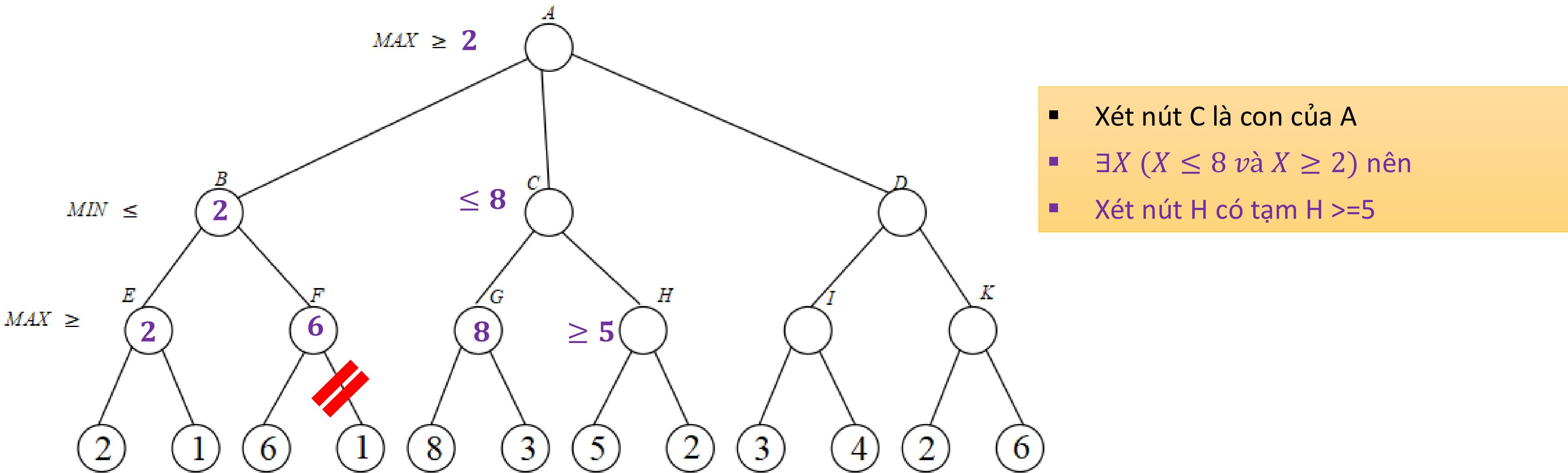
Xém tǐa $\alpha - \beta$ (Alpha – Beta pruning)



- Xét nút C có 2 con G và H đều chưa có giá trị
- Xét nút G có $valG = MAX(8,3)$
- Tạm $C \leq 8$

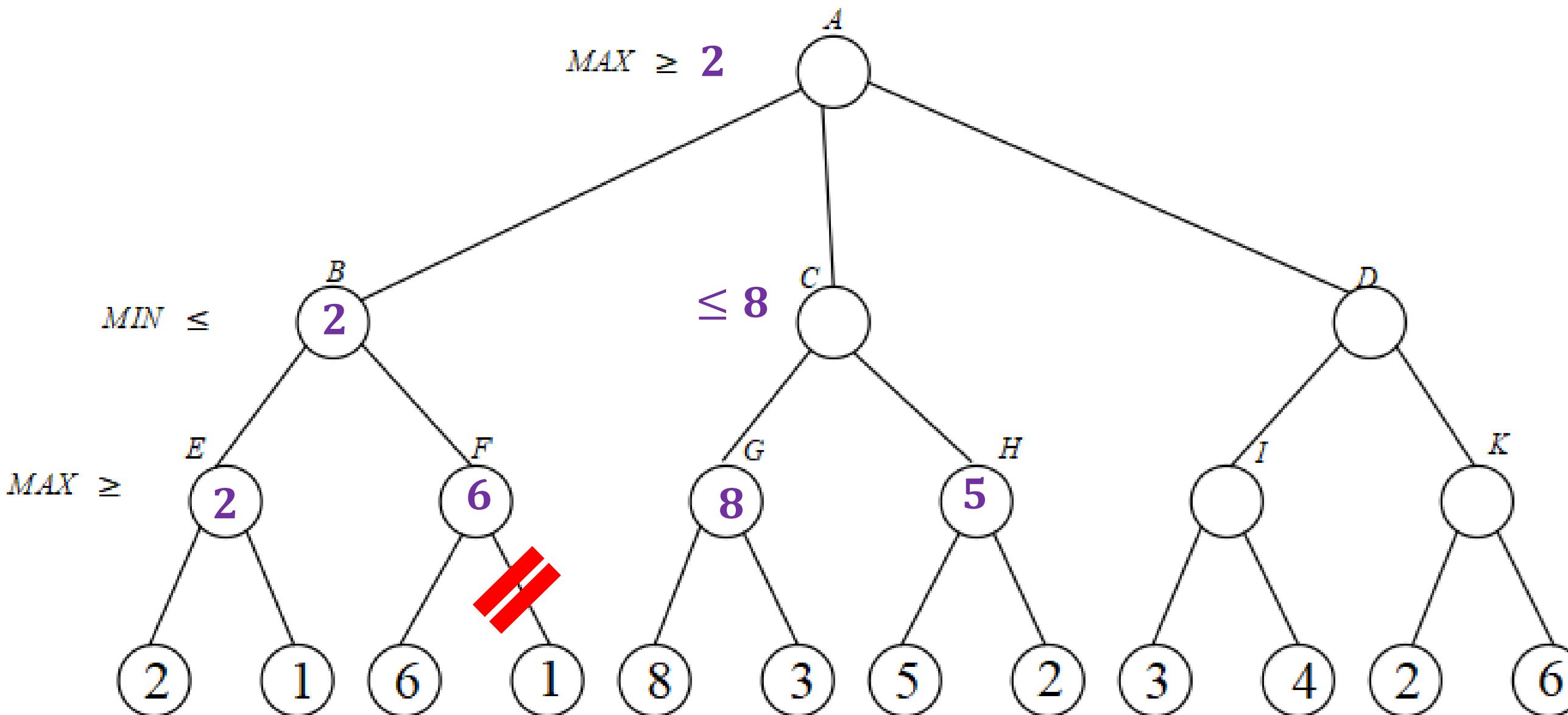
Chương 5. GIẢI THUẬT TÌM KIẾM LỜI GIẢI CHO TRÒ CHƠI

Xém tǐa $\alpha - \beta$ (Alpha – Beta pruning)



Chương 5. GIẢI THUẬT TÌM KIẾM LỜI GIẢI CHO TRÒ CHƠI

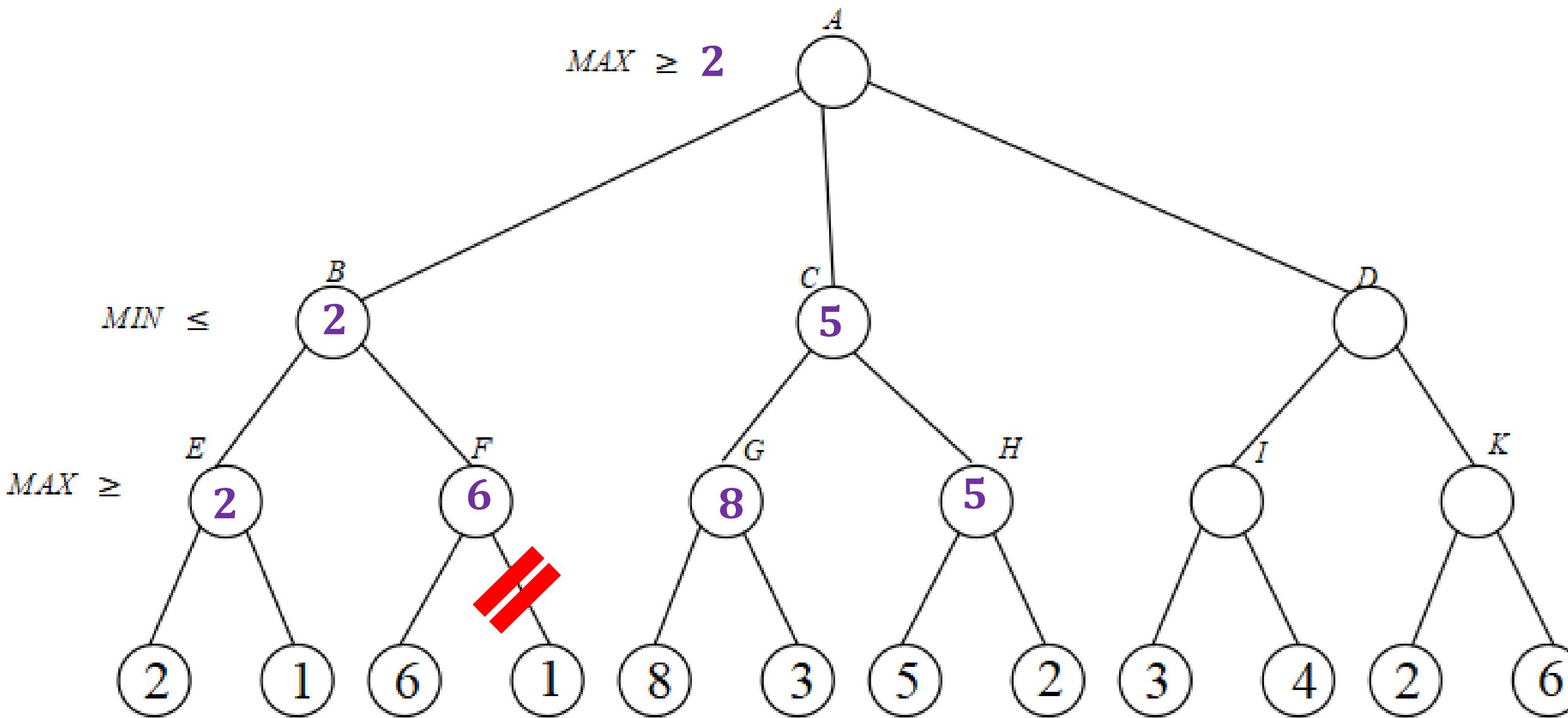
Xém tǐa $\alpha - \beta$ (Alpha – Beta pruning)



- Xét nút H là con của C
- $\exists X (X \leq 8 \text{ và } X \geq 5)$ nên
- Xét nút con bên phải H và tính lại $valH = MAX(5,2) = 5$

Chương 5. GIẢI THUẬT TÌM KIẾM LỜI GIẢI CHO TRÒ CHƠI

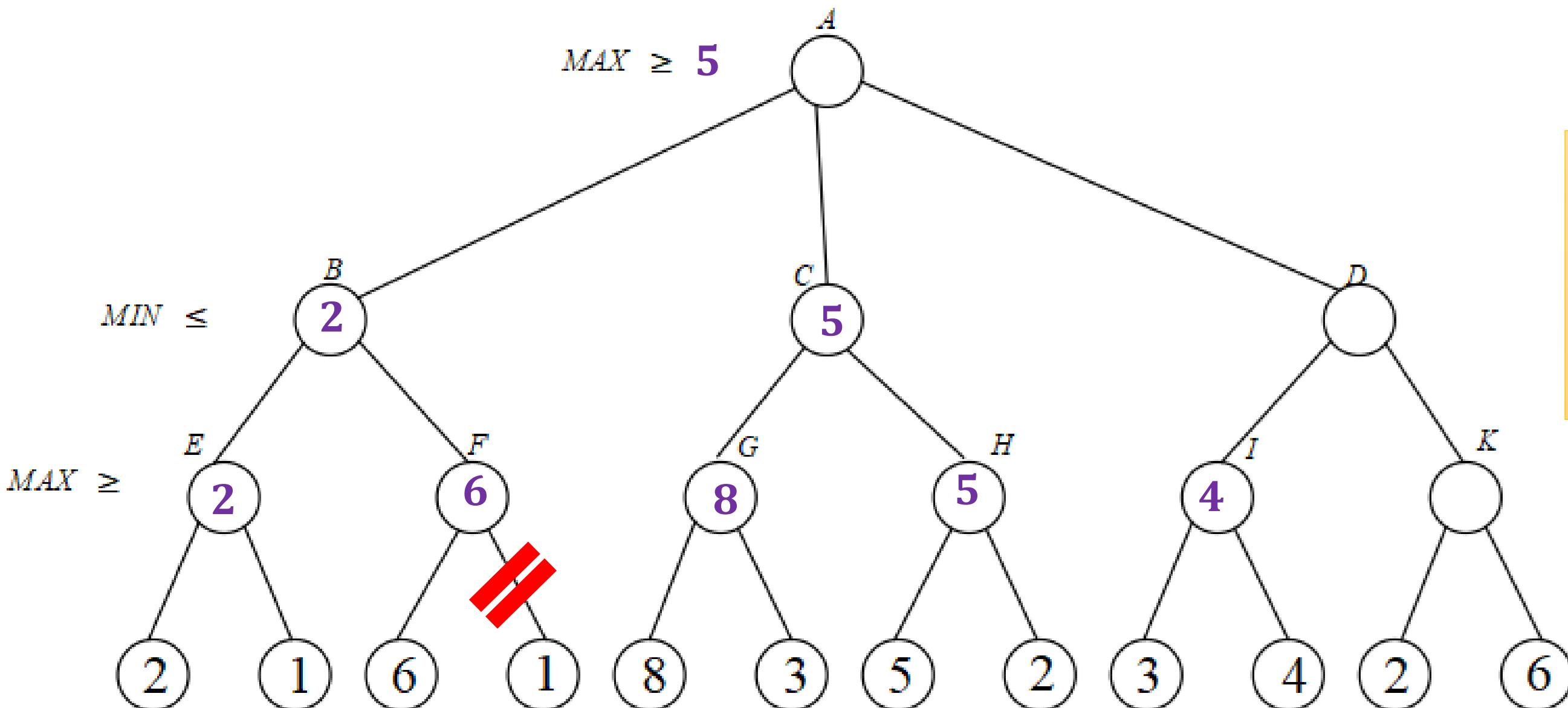
Xém tǐa $\alpha - \beta$ (Alpha – Beta pruning)



- Xét nút C có 2 nút con đã biết
- $\text{valC} = \text{MIN}(\text{tạm C}, 5) = 5$
- $\text{valA} = \text{MAX}(\text{tạm A}, 5) = 5$

Chương 5. GIẢI THUẬT TÌM KIẾM LỜI GIẢI CHO TRÒ CHƠI

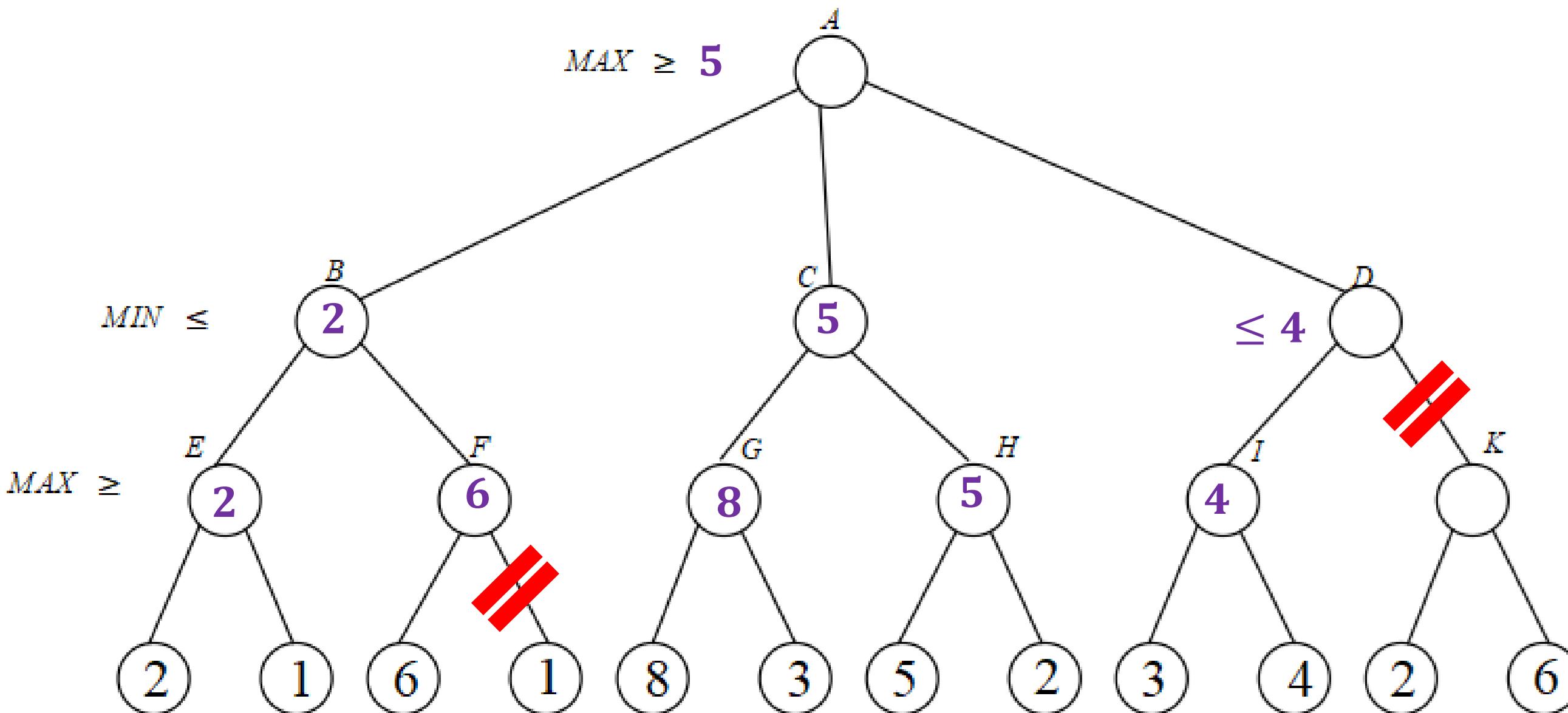
Xém tǐa $\alpha - \beta$ (Alpha – Beta pruning)



- Xét nút D là con của A chưa có giá trị và có 2 con I,E chưa có giá trị.
- Xét nút I: $I = MAX(3,4)$

Chương 5. GIẢI THUẬT TÌM KIẾM LỜI GIẢI CHO TRÒ CHƠI

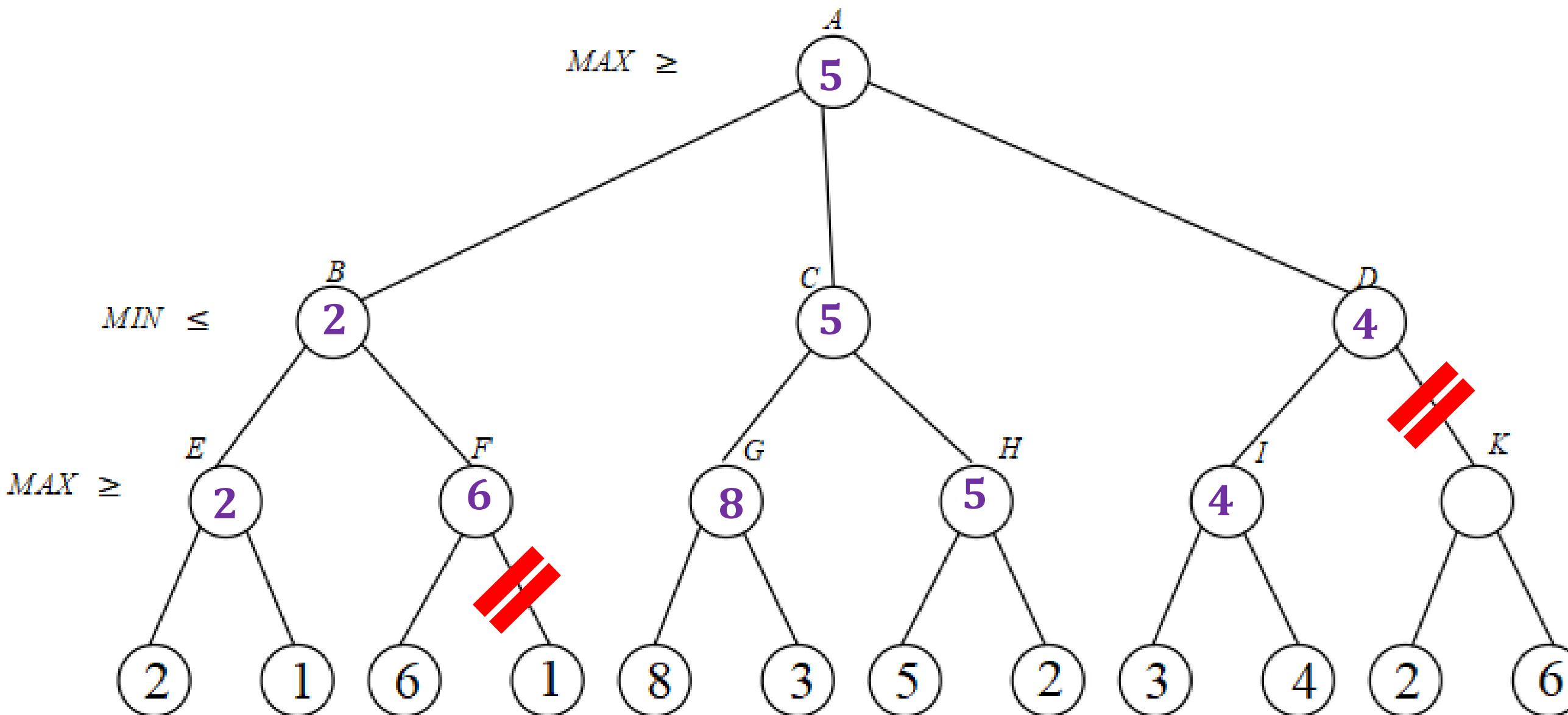
Xém tǐa $\alpha - \beta$ (Alpha – Beta pruning)



- Xét nút D có con là I: tạm $D \leq 4$
- Xét nút A có con là D
- $\nexists X (X \leq 4 \text{ và } X \geq 5)$ nên xén các con của D (nhánh K)

Chương 5. GIẢI THUẬT TÌM KIẾM LỜI GIẢI CHO TRÒ CHƠI

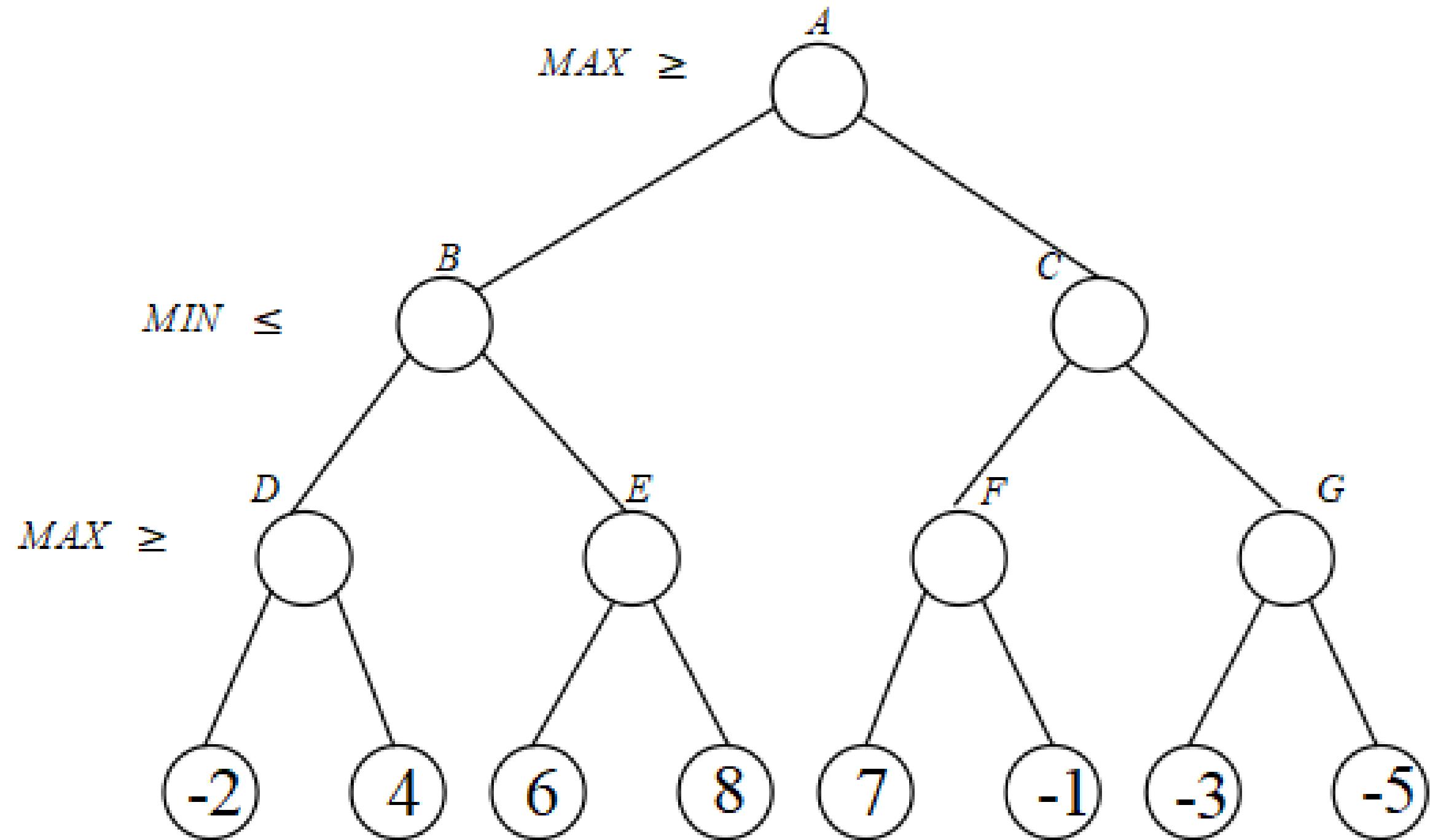
Xém tǐa $\alpha - \beta$ (Alpha – Beta pruning)



- Xét nút A có 3 con đã biết
- $\text{valA} = \text{MAX}(\text{tạm A}, 5) = 5$

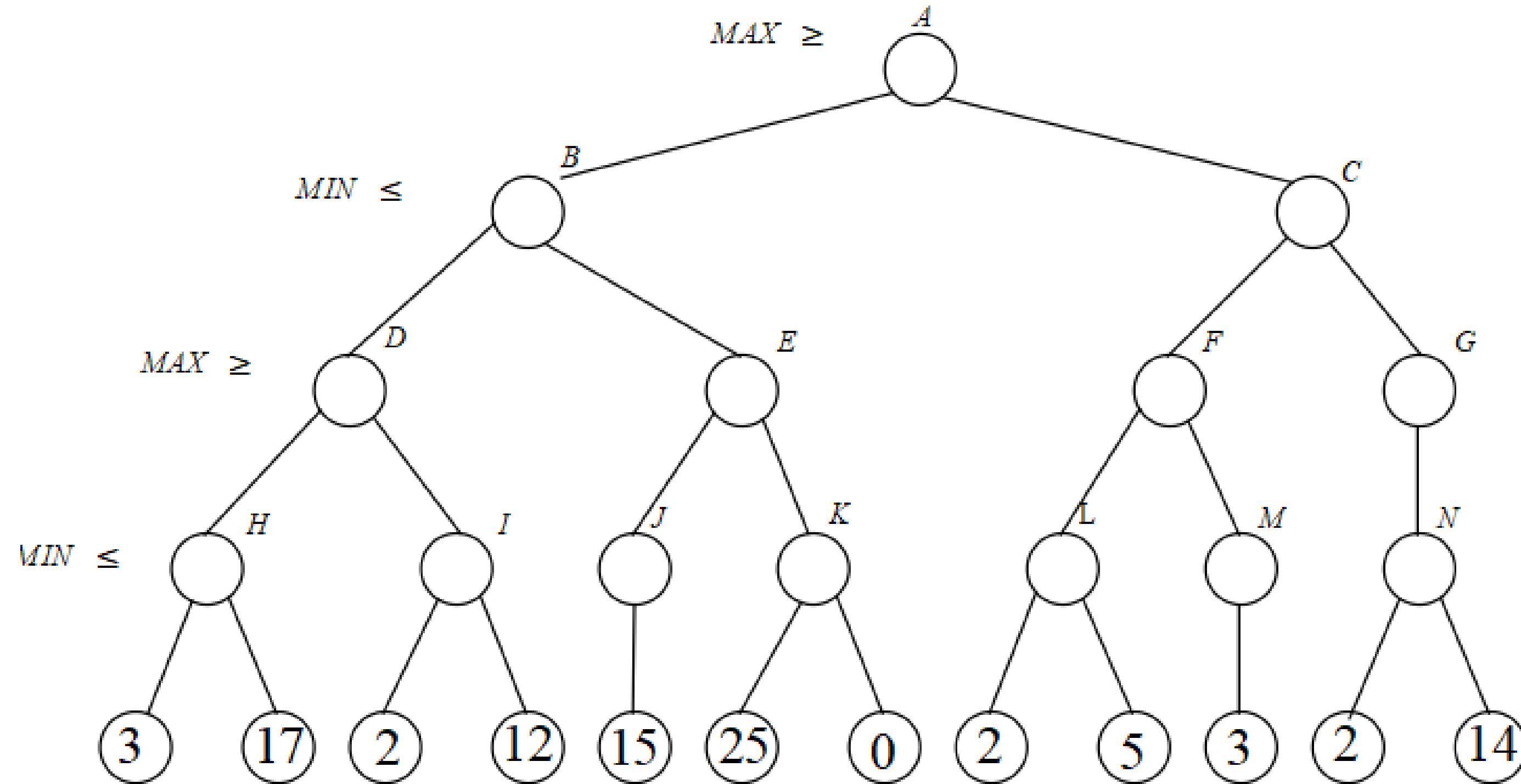
Chương 5. GIẢI THUẬT TÌM KIẾM LỜI GIẢI CHO TRÒ CHƠI

Xém tia $\alpha - \beta$ (Alpha – Beta pruning)



Chương 5. GIẢI THUẬT TÌM KIẾM LỜI GIẢI CHO TRÒ CHƠI

Xém tǐa $\alpha - \beta$ (Alpha – Beta pruning)



Chương 5. GIẢI THUẬT TÌM KIẾM LỜI GIẢI CHO TRÒ CHƠI

Xém tia $\alpha - \beta$ (Alpha – Beta pruning)

