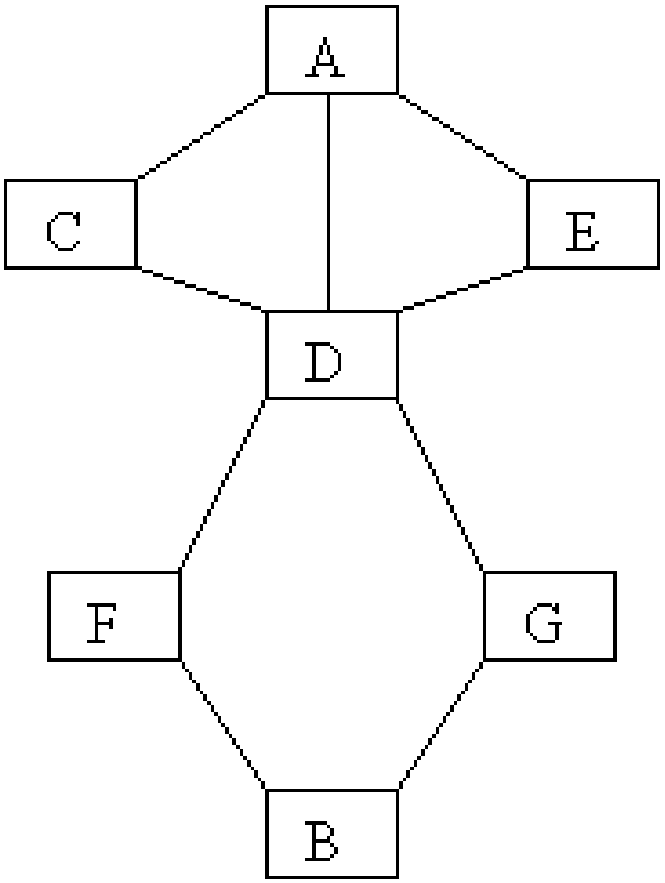
Phần 1
Giải quyết vấn đề bằng tìm kiếm

Chương 1: Các chiến lược tìm kiếm mù

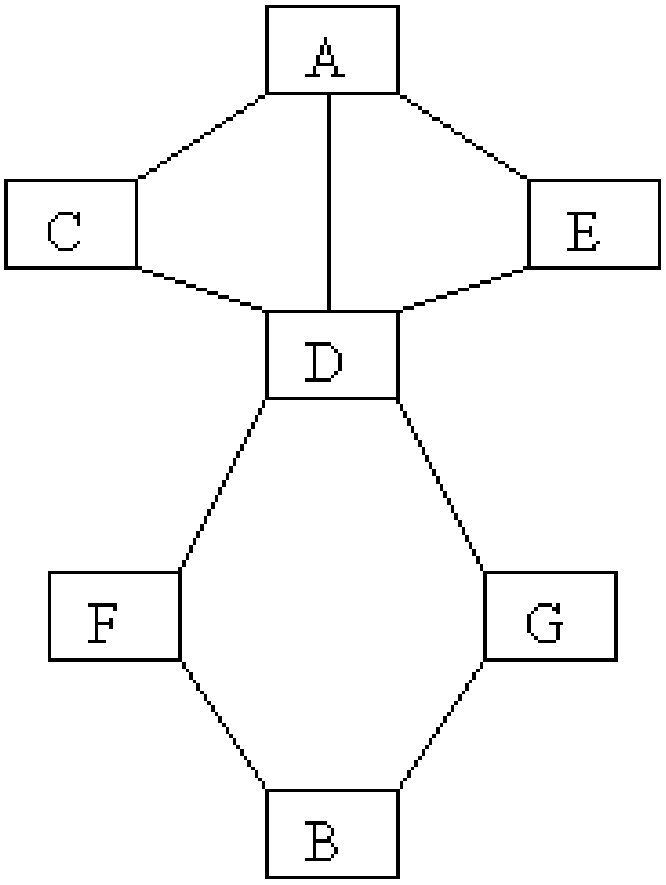
1. Biểu diễn vấn đề trong không gian trạng thái

Ví dụ về tìm kiếm đường đi trên bản đồ

+ Để giải quyết vấn đề bằng tìm kiếm, điều quan trọng là phải xác định được không gian tìm kiếm
+ Không gian này là tất cả các đối tượng mà ta cần quan tâm tìm kiếm, nó có thể liên tục hoặc rời rạc



Ví dụ 1: Bài toán tìm đường đi trên bản đồ giao thông



+Ta quan niệm các nút A, B,.. là các trạng thái

+A trạng thái đầu, B trạng thái kết thúc

+Khi đang ở một nút ta có thể đi sang một nút khác. Các con đường nối 2 nút được biểu diễn bởi một toán tử

+Bài toán tìm đường đi lúc này trở thành tìm một dãy các toán tử để đưa trạng thái đầu thành trạng thái kết thúc

Ví dụ 2: Xét trò chơi cờ vua

+ Ta quan niệm mỗi cách bố trí các quân cờ trên bàn cờ cho ta một trạng thái.

+ Trạng thái ban đầu ứng với sự sắp xếp các quân cờ lúc bắt đầu cuộc chơi.

+ Mỗi nước đi hợp lệ là một toán tử - nó biến đổi một tình huống trên bàn cờ thành một tình huống khác

+ Bài toán chơi cờ rõ ràng là tìm 1 dãy các toán tử (nước đi) để đưa trạng thái ban đầu về trạng thái kết thúc

Các yếu tố cơ bản biểu diễn một vấn đề trong không gian trạng thái

+ Trạng thái đầu, ký hiệu là u
+ Tập trạng thái kết thúc, ký hiệu T, T có thể có một trạng thái hoặc nhiều trạng thái – VD1,2
+ Tập các toán tử, tập hợp tất cả các trạng thái có thể đạt tới từ trạng thái ban đầu bằng cách áp dụng một dãy các toán tử sẽ cho ta một không gian trạng thái, ký hiệu là R
+ Trong trường hợp không gian trạng thái là hữu hạn ta có thể biểu diễn bằng một đồ thị có hướng

Ví dụ về xây dựng không gian trạng thái cho các vấn đề

Ví dụ1: Bài toán 8 số

(1)

2	8	3
1	6	4
7		5

1	2	3
8		4
7	6	5

(2)

Luật chơi: Chỉ được dịch các số vào ô trống

Yêu cầu: Tìm ra một dãy các dịch chuyển để biến đổi **(1)** thành **(2)**

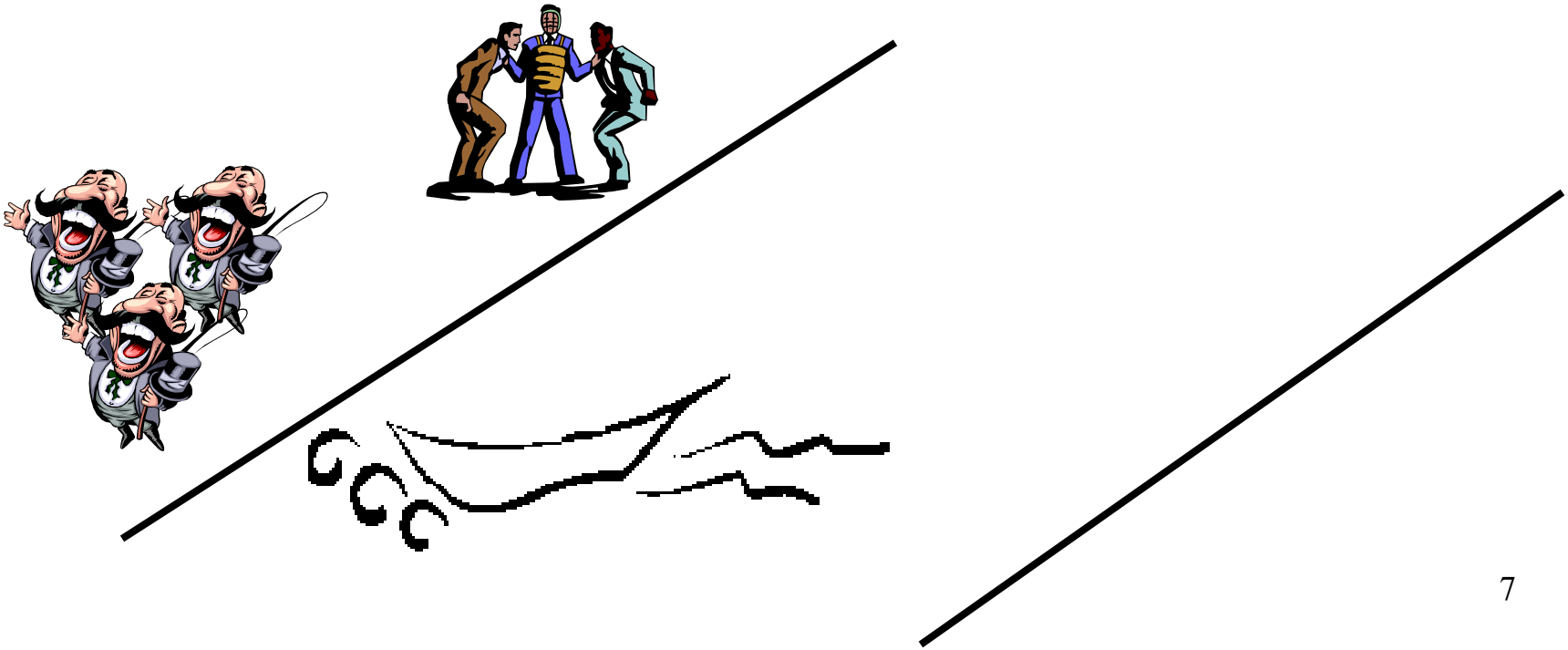
KGTT TTĐ: (1)
TTKT:(2)
Tập các toán tử: Up, Down, Left, Right

Ví dụ 2: Xét bài toán triệu phú và kẻ cướp

+ Có 3 triệu phú và 3 tên cướp và một chiếc thuyền ở bên bờ tả của một con sông

+ Thuyền chỉ chở được một hoặc hai người.

+ Hãy tìm cách đưa 3 triệu phú sang sông sao cho ở mỗi bờ sông số kẻ cướp không được lớn hơn số triệu phú



Dùng một bộ 3 số (a,b,k) để biểu diễn các trạng thái

a - số triệu phú, b - số kẻ cướp ở bờ tả

k=1 thuyền đang ở bờ tả, ngược lại k=0

Không gian trạng thái được xác định như sau:

TTĐ:(3,3,1)

TTKT:(0,0,0)

Tập các toán tử bao gồm các toán tử:

+Thuyền chở 1 kẻ cướp

+Thuyền chở 1 triệu phú

+Thuyền chở 1 triệu phú, 1 kẻ cướp

+Thuyền chở 2 kẻ cướp

+Thuyền chở 2 triệu phú

2. Các chiến lược tìm kiếm mù

- Cho u là một trạng thái, nếu áp dụng toán tử R ta thu được v thì v được gọi là trạng thái kề của u bởi toán tử R hoặc v được sinh từ u bởi R
- Quá trình áp dụng các toán tử để sinh ra các trạng thái kề của u được gọi là quá trình phát triển u
- Thông thường để giải quyết bài toán người ta thường xuất phát từ trạng thái đầu, phát triển nó cho đến khi gặp trạng thái đích
- Các chiến lược tìm kiếm mù:

  + Tìm kiếm theo bề rộng

  + Tìm kiếm theo chiều sâu

2.1 Chiến lược tìm kiếm theo bề rộng:

- Tại mỗi bước ta chọn trạng thái để phát triển là trạng thái được sinh ra trước các trạng thái chờ phát triển khác
- Sử dụng danh sách L để lưu các trạng thái được sinh ra và chờ được phát triển
- Mục tiêu của tìm kiếm là tìm đường đi từ trạng thái đầu đến trạng thái đích nên ta cần lưu vết của đường đi, ta có thể dùng hàm father để lưu lại cha của mỗI đỉnh trên đường đi, father(v) = u nếu u là cha của v

Procedure TimKiemRong

begin

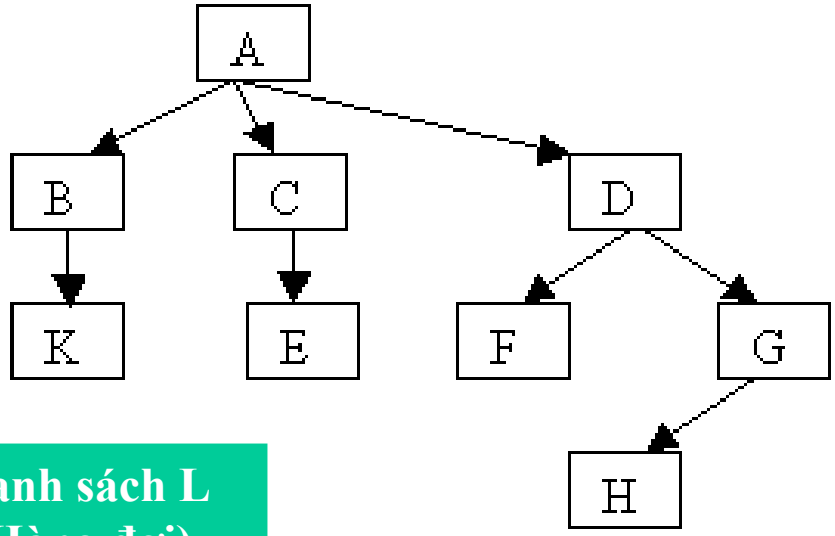
1. Khởi tạo danh sách L chỉ chứa trạng thái ban đầu
2. loop do
   - 2.1 if L rỗng then
     {thông báo tìm kiếm thất bại; stop};
   - 2.2 Loại trạng thái u ở đầu danh sách L;
   - 2.3 if u là trạng thái kết thúc then
     {thông báo tìm kiếm thành công; stop};
   - 2.4 for mỗi trạng thái v kề u do
     {đặt v vào cuối danh sách L; father(v)=u};

end;

Ví dụ: Cho không gian trạng thái

TTĐ:A

TTKT:H



Mô tả quá trình tìm kiếm

Phát triển trạng thái u	Trạng thái kề v	Danh sách L (Hàng đợi)
		A
A	B, C, D	B, C, D
B	K	C, D, K
C	E	D, K, E
D	F, G	K, E, F, G
K		E, F, G
E		F, G
F		G
G	H	H
H (TTKT)	DỪNG	

Nhận xét

- Trong TKR trạng thái nào được sinh ra trước sẽ được phát triển trước, do đó *danh sách L được xử lý như hàng đợi*
- Nếu bài toán có nghiệm (tồn tại đường đi từ trạng thái đầu tới trạng thái đích), thì thuật toán sẽ tìm ra nghiệm, đồng thời đường đi tìm được là đường đi ngắn nhất, trong trường hợp bài toán vô nghiệm và không gian trạng thái hữu hạn, thuật toán sẽ dừng và thông báo vô nghiệm
- Độ phức tạp thuật toán: Sinh viên tự đánh giá

2.2 Chiến lược tìm kiếm theo độ sâu

+ Tại mỗi bước ta chọn trạng thái để phát triển là trạng thái được sinh ra sau cùng trong các trạng thái chờ phát triển khác.

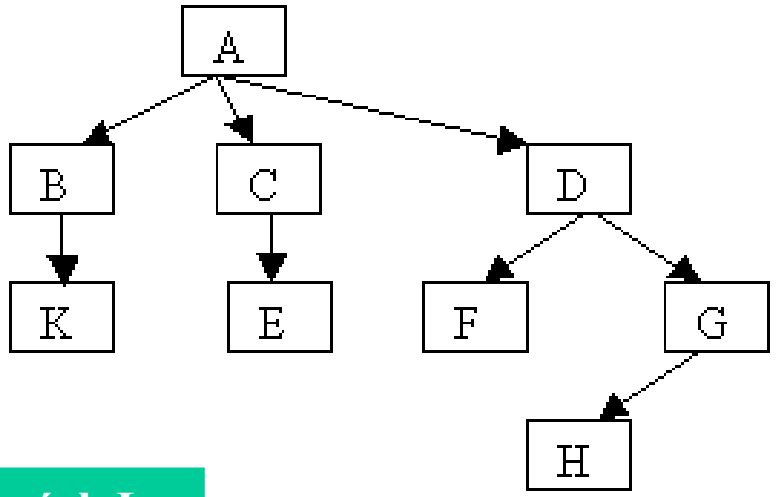
+ Thuật toán tìm kiếm sâu sẽ tương tự như tìm kiếm rộng, chỉ có điều *danh sách L được xây dựng như một ngăn xếp*

```
Procedure TimKiemSau
begin
1.  Khởi tạo danh sách L chỉ chứa trạng thái ban đầu
2.  loop do
    2.1  if L rỗng then
     {thông báo tìm kiếm thất bại; stop};
    2.2  Loại trạng thái u ở đầu danh sách L;
    2.3  if u là trạng thái kết thúc then
                {thông báo tìm kiếm thành công; stop};
    2.4  for mỗi trạng thái v kề u do
                {đặt v vào đầu danh sách L; father(v)=u};
end;
```

Ví dụ: Cho không gian trạng thái

TTĐ:A

TTKT:H



Mô tả quá trình tìm kiếm

Phát triển trạng thái u	Trạng thái kề v	Danh sách L (Ngăn xếp)
		A
A	B, C, D	D, C, B
D	F, G	G, F, C, B
G	H	H, F, C, B
H (TTKT)	DỪNG	

Nhận xét

Thuật toán tìm kiếm sâu không phải lúc nào cũng tìm ra nghiệm

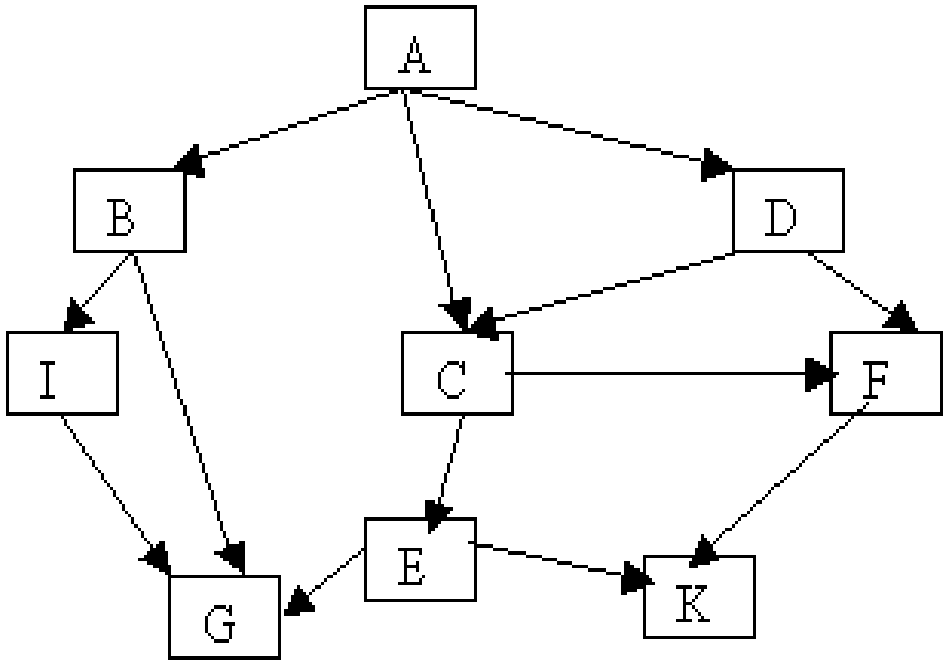
+ Nếu bài toán có nghiệm và không gian trạng thái hữu hạn thì nó luôn tìm ra nghiệm

+ Tuy nhiên trong trường hợp không gian trạng thái là vô hạn thì nó có thể không tìm ra nghiệm, lý do là nó luôn đi xuống theo độ sâu, và nếu nó đi theo một nhánh vô hạn mà nghiệm không nằm trên nhánh đó thì thuật toán sẽ không dừng.

2.3 Trạng thái lặp và loại bỏ trạng thái lặp

Xét đồ thị không gian trạng thái

TTĐ: A, TTKT:G



+ Trong đồ thị này ta thấy có những đỉnh có nhiều đường đi dẫn tới nó từ trạng thái ban đầu.Ví dụ C, E, F – Các trạng thái này được gọi là các trạng thái lặp

+ Quá trình tìm kiếm sẽ lãng phí rất nhiều thời gian để phát triển những trạng thái đã gặp và đã phát triển, đặc biệt khi đồ thị có chu trình quá trình tìm kiếm sẽ không dừng

+ Vì vậy trong quá trình tìm kiếm tránh sinh ra các trạng thái mà ta đã phát triển

Giải pháp

(1) Khi phát triển đỉnh u, không sinh ra các đỉnh trùng với cha u

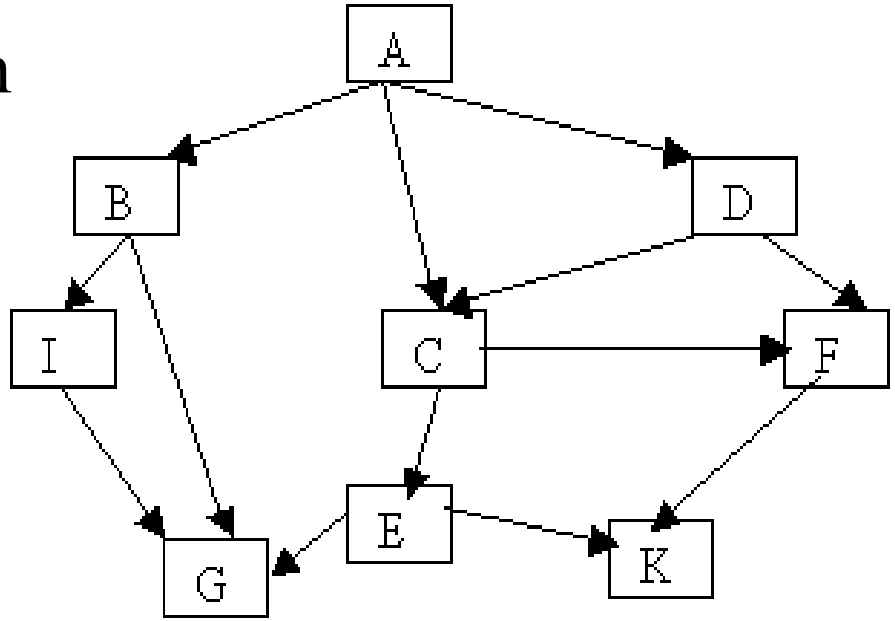
(2) Khi phát triển đỉnh u, không sinh ra các đỉnh trùng với một đỉnh nào đó nằm trên đường đi dẫn tới u

(3) Không sinh ra các đỉnh mà nó đã được sinh ra, tức là chỉ sinh ra các đỉnh mới

+ Hai giải pháp đầu khá đơn giản tuy nhiên không tránh hết được các trạng thái lặp.

+ Để thực hiện giải pháp thứ 3 ta sẽ lưu các trạng thái đã sinh ra vào danh sách Q và các trạng thái chờ phát triển vào danh sách L và đương nhiên trạng thái v lần đầu được sinh ra nếu nó không nằm trong Q

Ví dụ: Xét đồ thị không gian trạng thái
TTĐ: A, TTKT: G



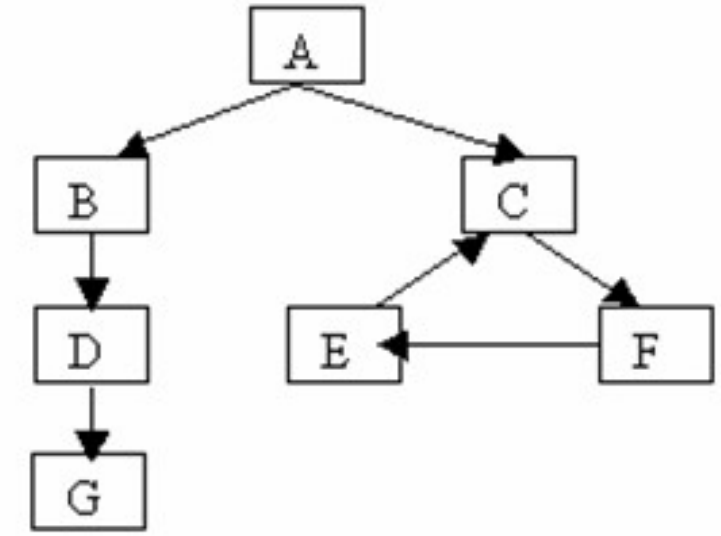
Mô tả quá trình tìm kiếm

Phát triển u	Trạng thái kề v	Danh sách L	Danh sách Q
		A	
A	B, C, D	D, C, B	A, B, C, D
D	C, F	F, C, B	A, B, C, D, F
F	K	K, C, B	A, B, C, D, F, K
K		C, B	A, B, C, D, **F**, K
C	E, **F**	E, B	A, B, C, D, F, **K**, E
E	G, **K**	G, B	A, B, C, D, F, **K**, E, G
G (TTKT)	DỪNG		

2.4 Tìm kiếm sâu lặp

Nếu cây tìm kiếm chứa nhánh vô hạn, khi tìm kiếm theo độ sâu ta có thể bị mắc kẹt ở nhánh đó.

Ví dụ: Cho đồ thị không gian tìm kiếm, với TTĐ: A, TTKT: G

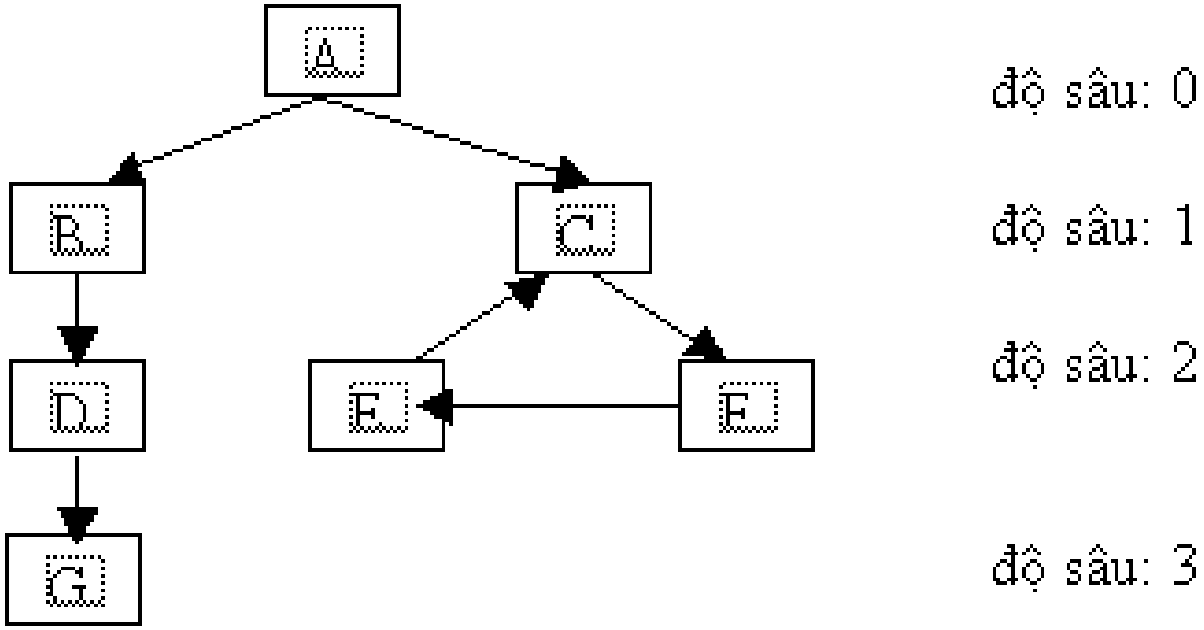


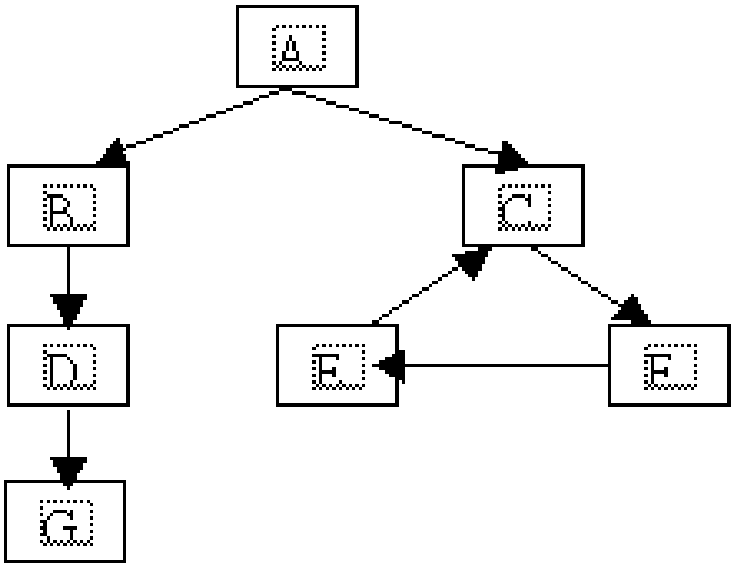
Phát triển trạng thái u	Trạng thái kề v	Danh sách L (Ngăn xếp)
		A
A	B, C	C, B
C	F	F, B
F	E	E, B
E	C	C, B
……	…….	…….

+ Để khắc phục ta chỉ tìm kiếm đến độ sâu hạn chế d nào đó, nếu không tìm được ta lại tăng độ sâu lên d+1 và lại tìm kiếm theo độ sâu này.

+ Quá trình trên được lặp với d=1..max.

+ Khi chọn max đủ lớn ta sẽ tìm được nghiệm





độ sâu: 0

độ sâu: 1

độ sâu: 2

độ sâu: 3

Phát triển trạng thái u	Trạng thái kề v	Danh sách L (Ngăn xếp)
		A(0)
A(0)	B(1),C(1)	C(1),B(1)
C(1)	F(2)	F(2),B(1)
F(2)	E(3)	E(3),B(1)
E(3)	C(4) loại vì **4>d**	B(1)
B(1)	D(2)	D(2)
D(2)	G(3)	G(3)
G(3) TTKT	DỪNG	

Ví dụ: Tìm kiếm sâu hạn chế với d=3

Thủ tục tìm kiếm sâu hạn chế

Procedure TKSHC(d)
Begin
1. Khởi tạo danh sách L chỉ chứa TT ban đầu u_0; depth(u_0)=0;
2. Loop do
2.1 if L rỗng then {Thông báo thất bại; stop}
2.2 Loại trạng thái u ở đầu danh sách L
2.3 if u là trạng thái kết thúc then
{Thông báo thành công; stop}
2.4 for mỗi trạng thái v kề u do
{depth(v)=depth(u)+1
if depth(v)<=d then
Đặt v vào đầu danh sách L;
}
end;

Thủ tục tìm kiếm sâu lặp

```
Procedure TKSL
Begin
For d=0 to max do
      { TKSHC(d);
        if thành công then exit}
End;
```

BÀI TẬP

+ Xây dựng một đồ thị không gian trạng thái có độ sâu >3, không có chu trình. Sử dụng phương pháp tìm kiếm sâu và tìm kiếm rộng để xác định đường đi từ trạng thái đầu tới trạng thái kết thúc

+ Xây dựng một đồ thị không gian trạng thái có độ sâu >3, có chu trình. Sử dụng phương pháp tìm kiếm sâu hạn chế để xác định đường đi từ trạng thái đầu tới trạng thái kết thúc

BÀI TẬP

+ Xây dựng bảng mô tả quá trình tìm kiếm lời giải cho bài toán kẻ cướp và triệu phú theo phương pháp tìm kiếm rộng và phương pháp tìm kiếm sâu

+ Xây dựng bảng mô tả quá trình tìm kiếm lời giải cho bài toán 8 số theo phương pháp tìm kiếm rộng và phương pháp tìm kiếm sâu

Mô tả quá trình tìm kiếm lời giải cho bài toán kẻ cướp-triệu phú

u	Trạng thái kề v	Danh sách L (Hàng đợi)
		(3,3,1)
(3,3,1)	(3,2,0),(2,3,0),(2,2,0),(3,1,0),(1,3,0)	(3,2,0),(2,2,0),(3,1,0)
(3,2,0)	(3,3,1)	(2,2,0),(3,1,0)
(2,2,0)	(3,2,1),(2,3,1),(3,3,1)	(3,1,0), (3,2,1)
(3,1,0)	(3,2,1)	(3,2,1)
(3,2,1)	(3,1,0),(2,2,0),(2,1,0),(3,0,0),(1,2,0)	(2,2,0),(3,0,0)
(2,2,0)	(2,3,1),(3,2,1),(3,3,1)	(3,0,0)
(3,0,0)	(3,1,1),(3,2,1)	(3,1,1)
(3,1,1)	(3,0,0),(2,1,0),(2,0,0),(1,1,0)	(1,1,0)
(1,1,0)	(1,2,1),(2,1,1),(2,2,1),(1,3,1),(3,1,1)	(2,2,1)
(2,2,1)	(2,1,0),(1,2,0),(1,1,0),(2,0,0),(0,2,0)	(0,2,0)
(0,2,0)	(0,3,1),(1,2,1),(1,3,1),(2,2,1)	(0,3,1)
(0,3,1)	(0,2,0),(0,1,0)	(0,1,0)
(0,1,0)	(0,2,1),(1,1,1),(1,2,1),(0,3,1),(2,1,1)	(0,2,1),(1,1,1)
(0,2,1)	(0,1,0),(0,0,0)	(1,1,1),(0,0,0)
(1,1,1)	(1,0,0),(0,1,0),(0,0,0)	(0,0,0)
(0,0,0)	**Trạng thái kết thúc – Dừng**	