

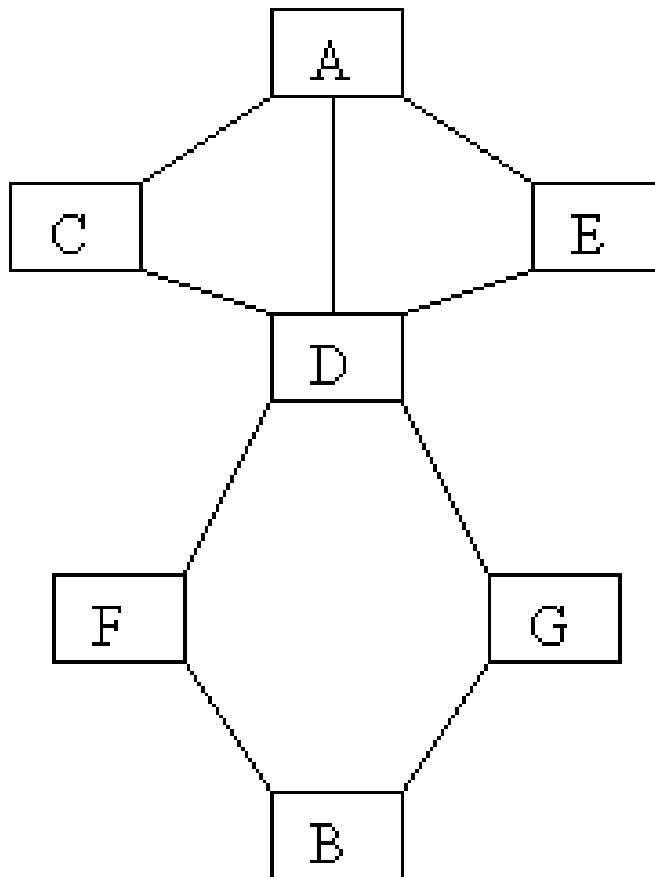
# Giải quyết vấn đề bằng Tìm kiếm

TS. Nguyễn Quốc Tuấn  
Bộ môn Mạng và Các HTTT

# Nội dung

- Biểu diễn vấn đề trong không gian trạng thái
- Các chiến lược tìm kiếm

## Ví dụ 1: Xét 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. Toán tử biến đổi trạng thái này thành trạng thái khác

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

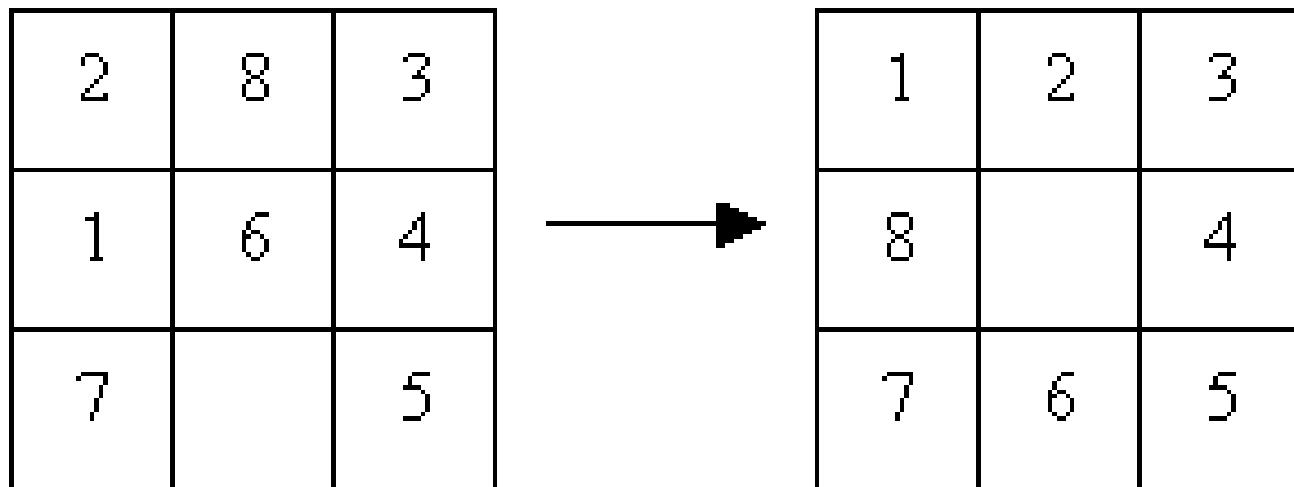
# Biểu diễn một vấn đề trong không gian trạng thái

Cần xác định các yếu tố :

- Tập không gian trạng thái:  $U$
- Trạng thái đầu : ký hiệu là  $u_o$  ( $u_o \in U$ )
- Tập trạng thái kết thúc : ký hiệu  $T$ ,
  - $T$  có thể có một trạng thái hoặc nhiều trạng thái
  - $T \subset U$
- 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  $R$ 
  - $R : U \rightarrow U$
- 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 KGTT cho các vấn đề

- Ví dụ: Bài toán 8 số



Luật chơi: Chỉ được dịch các số vào ô trống

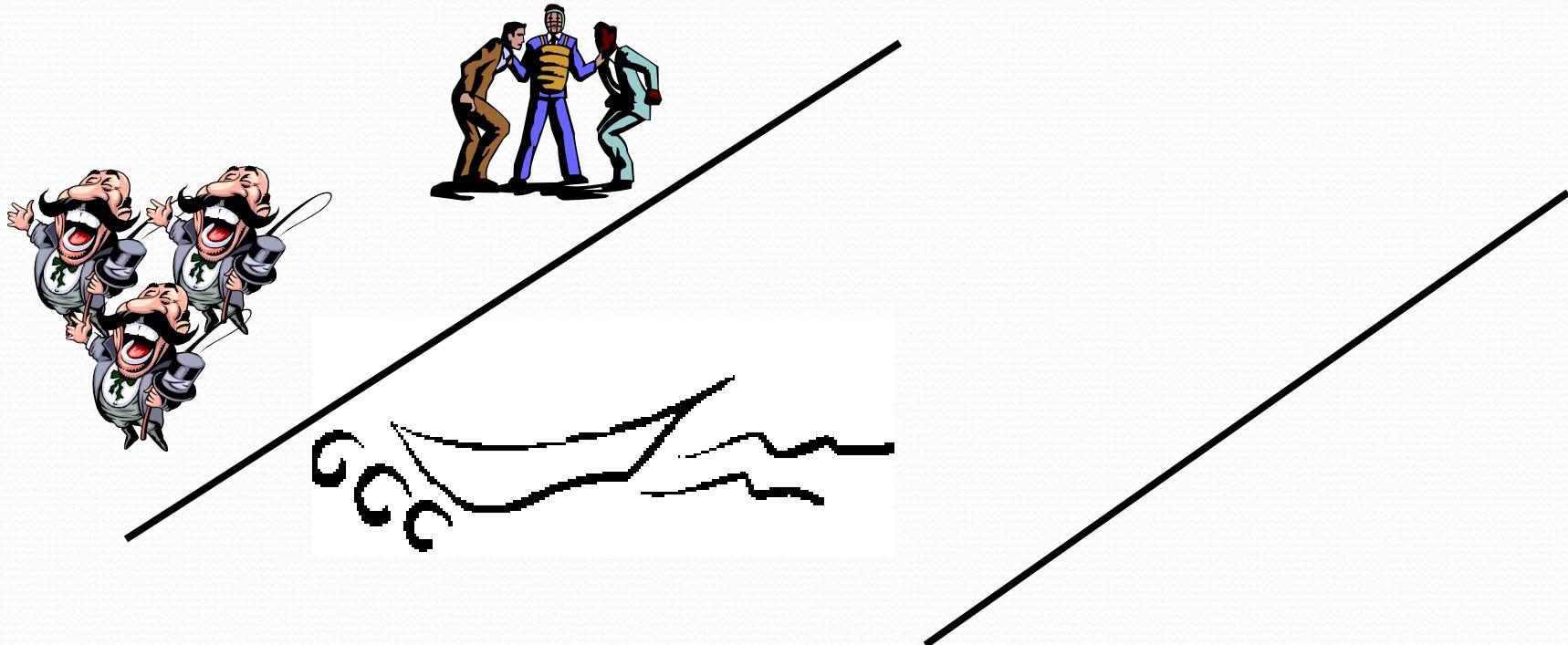
Yêu cầu: Tìm ra một dãy các dịch chuyển để biến đổi trạng thái đầu thành trạng thái kết thúc

# **Không gian trạng thái**

- TTĐ:
- TTKT:
- 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

- Dùng một bộ 3 số  $(a,b,k)$  để biểu diễn các trạng thái
  - $a$  - số triệu phú,  $(3\ 3\ 1)$
  - $b$  - số kẻ cướp ở bờ tả,  $(2\ 2\ 0)$
  - $k=1$  thuyền đang ở bờ tả, ngược lại  $k=0$ ,  $(3\ 2\ 1)$
- Không gian trạng thái được xác định như sau
  - TTĐ: $(3,3,1)$ ,  $(3\ 0\ 0)$
  - TTKT: $(0,0,0)$ ,  $(3\ 1\ 1)$
  - Tập các toán tử bao gồm các toán tử:
    - Thuyền chở 1 kẻ cướp,  $(1\ 1\ 0)$
    - Thuyền chở 1 triệu phú,  $(2\ 2\ 1)$
    - Thuyền chở 1 triệu phú, 1 kẻ cướp,  $(0\ 2\ 0)$
    - Thuyền chở 2 kẻ cướp,  $(0\ 3\ 1)$
    - Thuyền chở 2 triệu phú,  $(0\ 1\ 0)$
    - $(0\ 2\ 1)$
    - $(0\ 0\ 0)$

## 2. Các chiến lược tìm kiếm

- Cho  $u$  là một trạng thái, nếu áp dụng toán tử  $R$  ta thu được  $v$  thì  $v$  được gọi là trạng thái kế 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 kế 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

## Một số chiến lược tìm kiếm:

- Chiến lược tìm kiếm mù: Không có sự hướng dẫn nào cho sự tìm kiếm, ta chỉ phát triển trạng thái đầu cho tới khi gặp trạng thái đích, có 2 kỹ thuật cho chiến lược này đó là tìm kiếm rộng và tìm kiếm sâu.
- Chiến lược tìm kiếm kinh nghiệm: Trong rất nhiều vấn đề ta có thể dựa vào hiểu biết của chúng ta về vấn đề đó để đánh giá các trạng thái. Trong quá trình phát triển các trạng thái ta sẽ chọn 1 trong số các trạng thái chờ phát triển một trạng thái được đánh giá tốt nhất để phát triển,.. quá trình tiếp tục cho tới khi gặp trạng thái đích.

## 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,  $\text{father}(v)=u$  nếu  $u$  là cha của  $v$

## **Procedure TimKiemRong //Breath first search**

begin

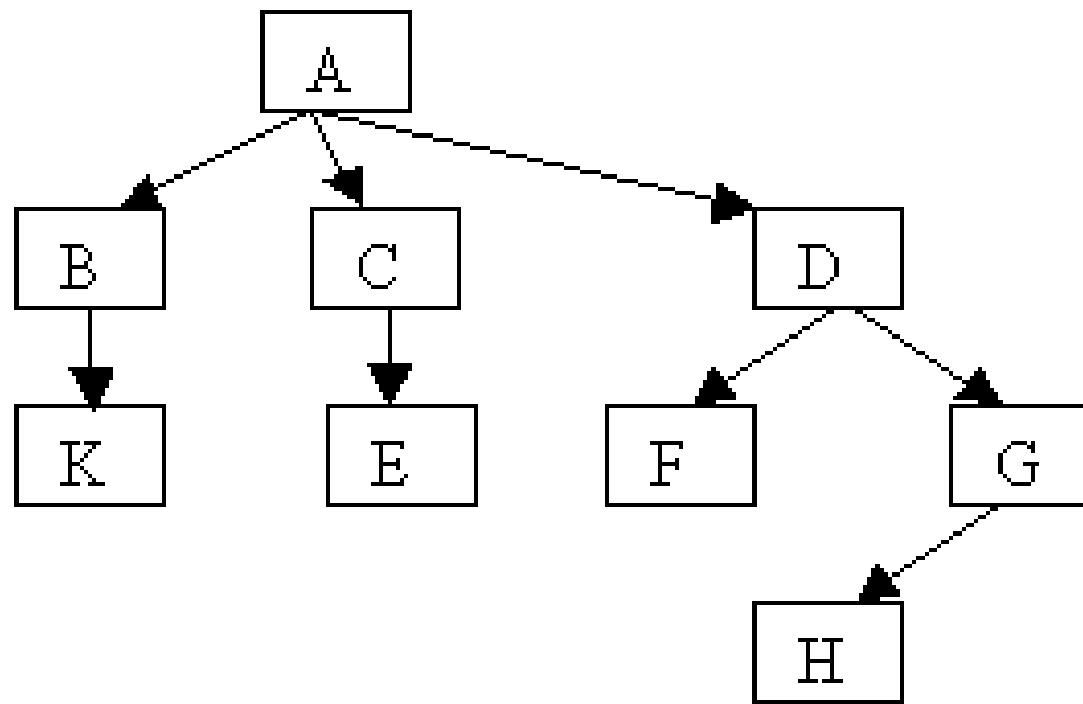
1. Khởi tạo danh sách L chỉ chứa trạng thái ban đầu
2. loop do
  - 2.1 if L rỗng then  
    {thông báo tìm kiếm thất bại; stop};
  - 2.2 Loại trạng thái u ở đầu danh sách L;
  - 2.3 if u là trạng thái kết thúc then  
    {thông báo tìm kiếm thành công; stop};
  - 2.4 for mỗi trạng thái v kề u do  
 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

end;

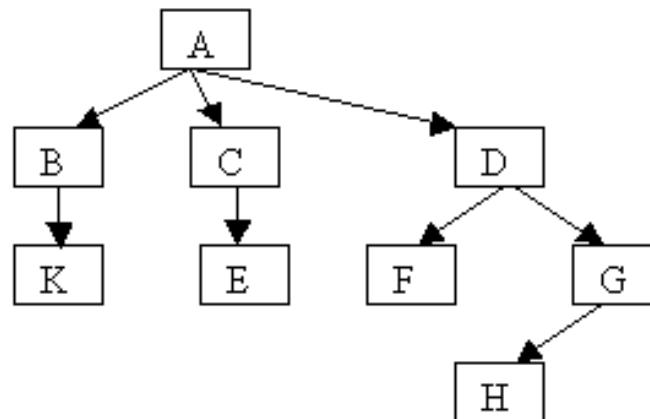
Ví dụ: Cho không gian trạng thái

TTĐ: A

TTKT:H



## Mô tả quá trình tìm kiếm



Phát triển TT	Trạng thái kề	Danh sách L
		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 //Depth first search

begin

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

2. loop do

2.1 if L rỗng then

{thông báo tìm kiếm thất bại; stop};

2.2 Loại trạng thái u ở đầu danh sách L;

2.3 if u là trạng thái kết thúc then

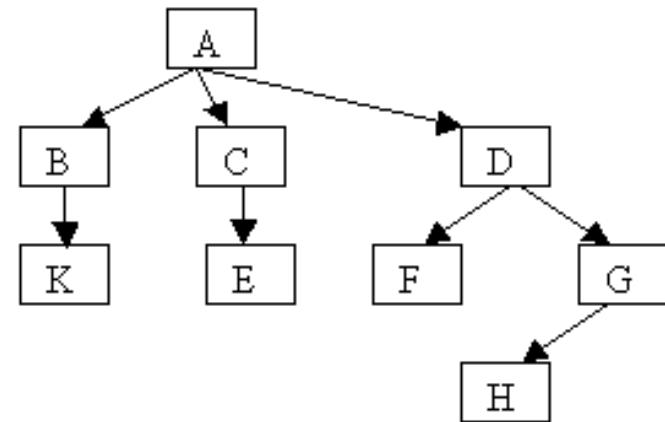
{thông báo tìm kiếm thành công; stop};

2.4 for mỗi trạng thái v kề u do

{đặt v vào đầu danh sách L; father(v)=u};

end;

Xét ví dụ trên, kết quả của quá trình tìm bảng sau

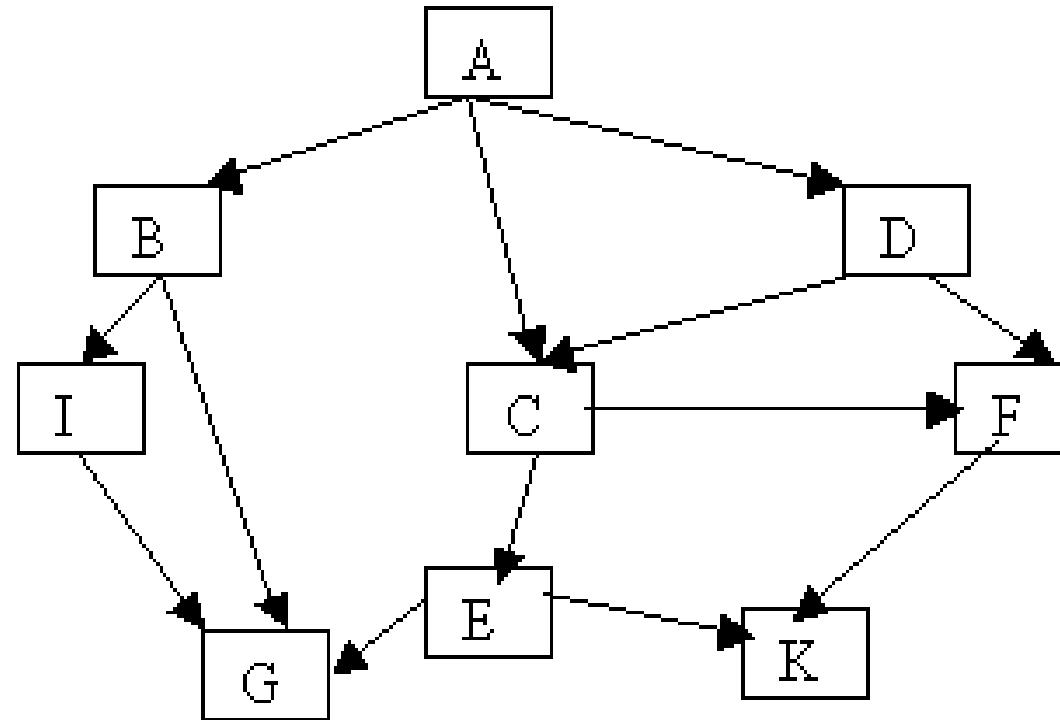


Phát triển TT	Trạng thái kề	Danh sách L
		A
A	B,C,D	B,C,D
D	F,G	B,C,F,G
G	H	B,C,F,H
H	TTKT-DUNG	

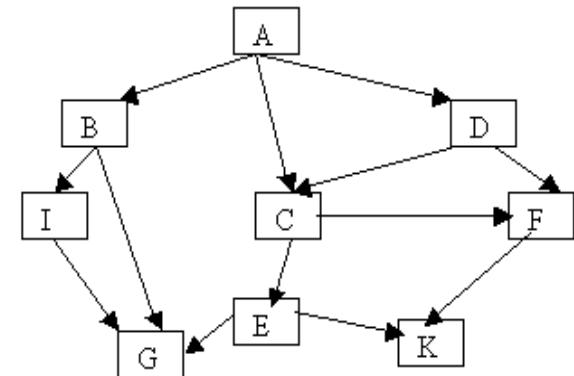
## 2.3 Trạng thái lắp và loại bỏ trạng thái lắp

Xét đồ thị không gian trạng thái

TTĐ: A, TTKT:G



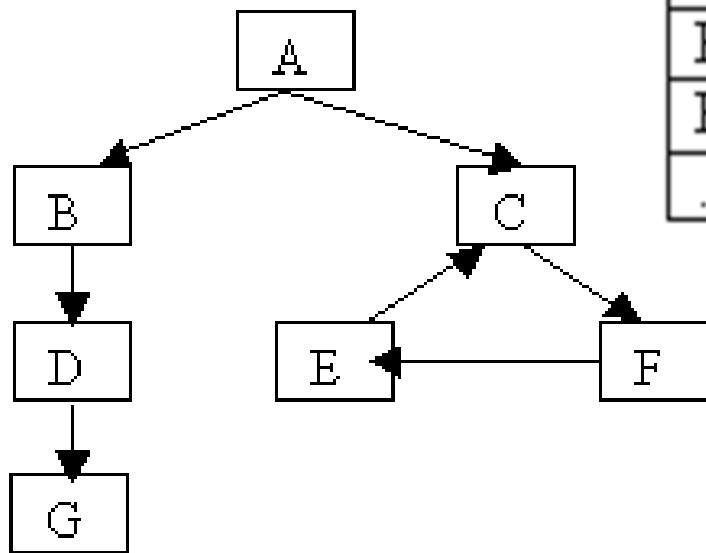
# Quá trình tìm kiếm thể hiện theo bảng



Phát triển TT	Trạng thái kề	Danh sách Q	Danh sách L
			A
A	B,C,D	A,B,C,D	B,C,D
D	C,F	A,B,C,D,F	B,C,F
F	K	A,B,C,D,F,K	B,C,K
K		A,B,C,D,F,K	B,C
C	E,F	A,B,C,D,F,K,E	B,E
E	G,K	A,B,C,D,F,K,E,G	B,G
G	TTKT/DUNG		

## 2.4 Tìm kiếm sâu lắp

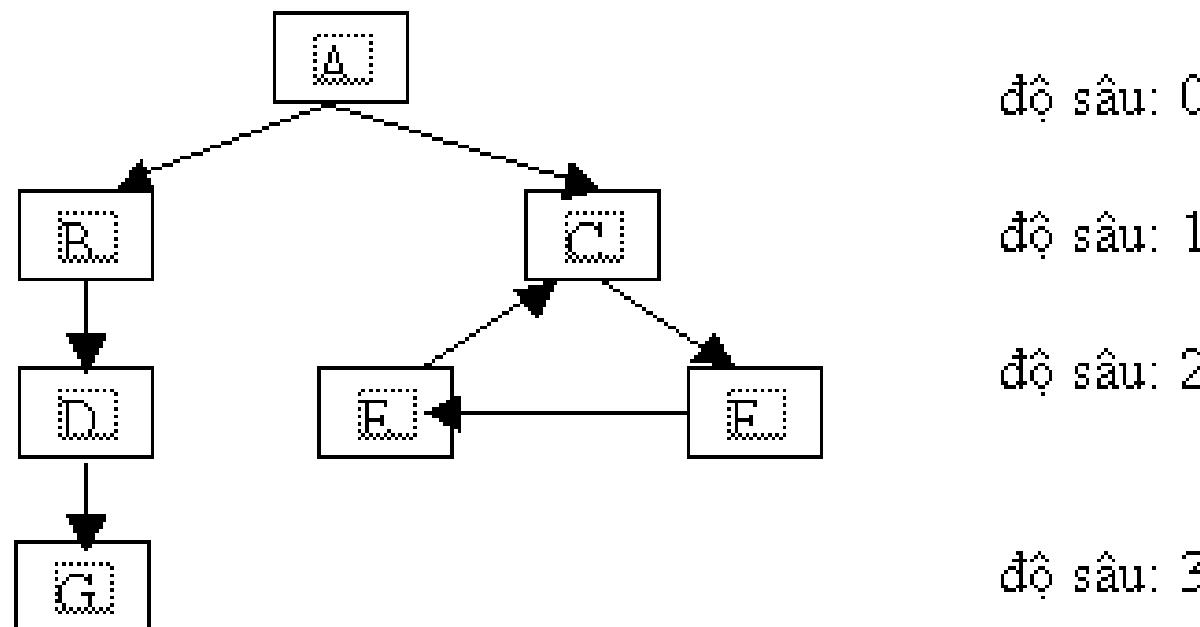
Nếu cây tìm kiếm chứa nhánh vô hạn, khi tìm kiếm theo độ sâu ta có thể bị mắc kẹt ở nhánh đó.



Phát triển TT	Trạng thái kè	Danh sách L
		A
A	B,C	B,C
C	F	B,F
F	E	B,E
E	C	B,C
...	...	...

Để khắc phục ta chỉ tìm kiếm đến độ sâu 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 - ở đây ta xây dựng 2 thủ tục



## Procedure TKSHC(d)

Begin

1. Khởi tạo danh sách L chỉ chứa TT ban đầu  $u_o$ ;  $\text{depth}(u_o)=0$ ;
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  
          if  $\text{depth}(v) \leq d$  then  
            {Đặt  $v$  vào đầu danh sách L;  $\text{depth}(v) = \text{depth}(u) + 1$ }

end;

## Procedure TKSL

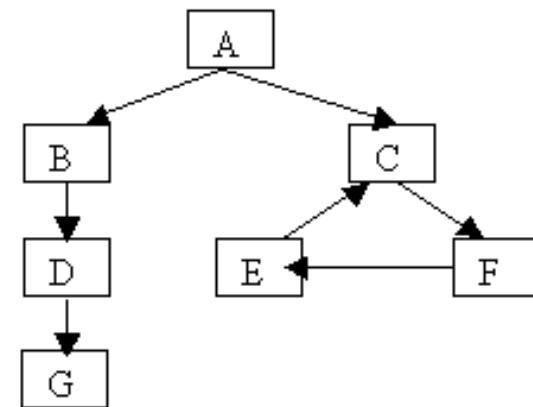
Begin

For d=0 to max do

{ TKSHC(d);  
if thành công then exit}

End;

# Tìm kiếm sâu hạn chế với $d=3$



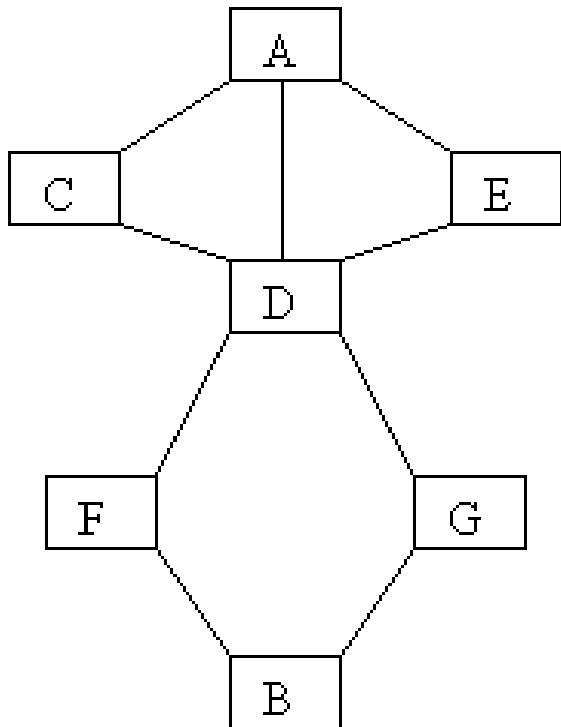
Phát triển TT	Trạng thái kè	Danh sách L
		A(0)
A(0)	B(1),C(1)	B(1),C(1)
C(1)	F(2)	B(1),F(2)
F(2)	E(3)	B(1),E(3)
E(3)	C(4)	B(1)
B(1)	D(2)	D(2)
D(2)	G(3)	G(3)
G(3)	TTKT - dừng	

# Các chiến lược tìm kiếm kinh nghiệm

## Hàm đánh giá và tìm kiếm kinh nghiệm

- Trong nhiều vấn đề ta có thể sử dụng kinh nghiệm, tri thức của chúng ta trong lĩnh vực bài toán để đánh giá các trạng thái của bài toán.
- Thông qua việc đánh giá các trạng thái ta có thể xác định mức độ lợi thế của các trạng thái trong quá trình tìm kiếm.

## Ví dụ: Tìm đường đi từ A đến B



Giả sử người đi đang đứng ở A và phải cân nhắc xem đi đến C, D hay E

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

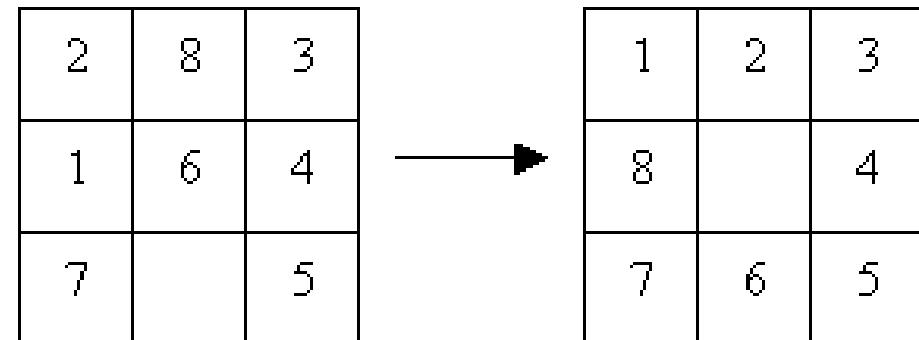
Người đi sẽ chọn ???

Câu trả lời: theo tự nhiên người đi sẽ chọn cách đi sang C

- Kinh nghiệm ở của con người đã được sử dụng
- Kinh nghiệm sai có thể dẫn ta đi chệch hướng, do đó tìm kiếm kém hiệu quả
- Trong bài toán tìm kiếm, kinh nghiệm được thể hiện qua việc xây dựng hàm đánh giá, tùy thuộc vào từng bài toán mà hàm đánh giá được xây dựng khác nhau.

- Ví dụ 1: Trong bài toán tìm đường đi trên bản đồ giao thông, ta có thể lấy độ dài đường chim bay từ một nút đến nút đích làm giá trị hàm đánh giá.

- Ví dụ 2: Bài toán 8 số



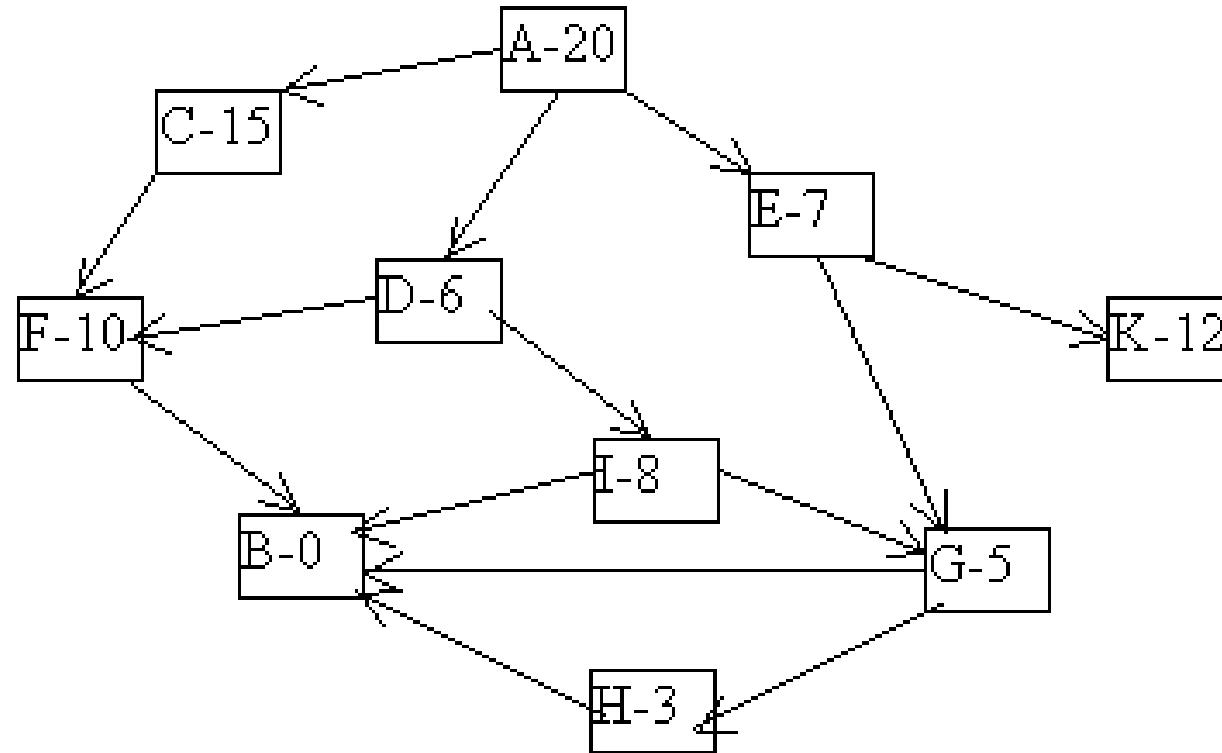
- Hàm  $h_1$ : Với mỗi trạng thái  $u$  thì  $h_1(u)$  là số quân không nằm đúng vị trí của nó trong trạng thái đích, ví dụ  $h_1(u_o)=4$
- Hàm  $h_2$ :  $h_2(u)$  là tổng khoảng cách giữa vị trí của các quân trong trạng thái  $u$  và vị trí của nó trong trạng thái đích. Ở đây khoảng cách được hiểu là số ít nhất các dịch chuyển theo hàng hoặc cột để đưa một quân tới vị trí của nó trong trạng thái đích, ví dụ  $h_2(u_o) = 1+1+1+2 = 5$ , vì quân 1, 2, 6 cần ít nhất 1 dịch chuyển, quân 8 cần ít nhất 2 dịch chuyển

- Có hai chiến lược tìm kiếm kinh nghiệm:
  - Tìm kiếm tốt nhất đầu tiên (Best-First Search)  
= Tìm kiếm theo bề rộng + Hàm đánh giá
  - Tìm kiếm leo đồi (Hill-Climbing Search)  
= Tìm kiếm theo độ sâu + Hàm đánh giá

# Tìm kiếm tốt nhất đầu tiên

- Là tìm kiếm theo bề rộng được hướng dẫn bởi hàm đánh giá
- Điểm khác biệt:
  - Trong tìm kiếm theo bề rộng ta lần lượt phát triển các đỉnh ở mức hiện tại để sinh ra các đỉnh ở mức tiếp theo
  - Trong tìm kiếm tốt nhất đầu tiên, ta chọn đỉnh để phát triển là đỉnh tốt nhất được xác định bởi hàm đánh giá (đỉnh có giá trị hàm đánh giá nhỏ nhất), đỉnh này có thể ở mức hiện tại hoặc mức trên

Ví dụ: Cho đồ thị không gian trạng thái với TTĐ A, TTKTB, giá trị đánh giá trạng thái ghi tại mỗi nút



## Procedure TimKiemTotNhatDauTien

begin

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

2.1 if L rỗng then

{thông báo tìm kiếm thất bại; stop};

2.2 Loại trạng thái u ở đầu danh sách L;

2.3 if u là trạng thái kết thúc then

{thông báo tìm kiếm thành công; stop};

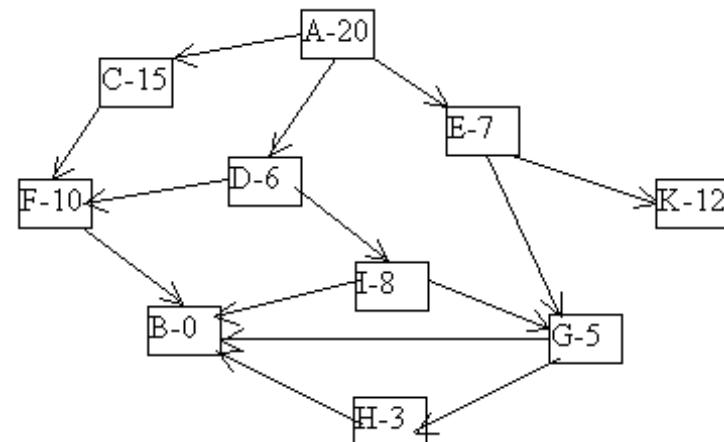
2.4 for mỗi trạng thái v kề u do

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

end;

# Quá trình tìm kiếm bằng kỹ thuật tìm kiếm tốt nhất đầu tiên

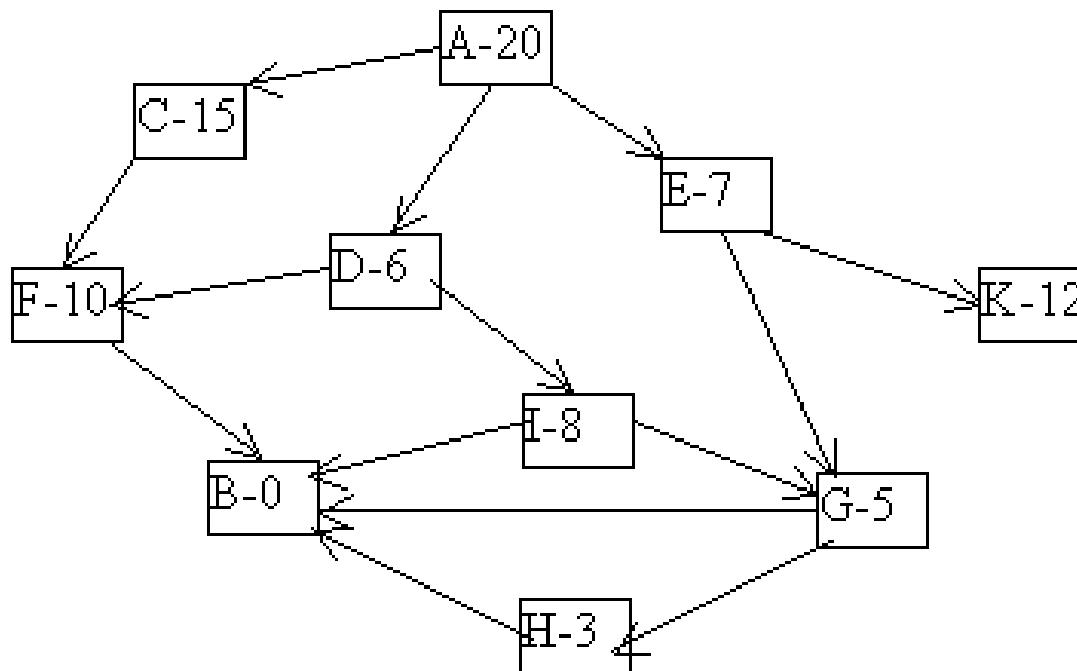
Phát triển TT	Trạng thái kề	Danh sách L
		A20
A20	C15,D6,E7	D6,E7,C15
D6	F10,I8	E7,I8,F10,C15
E7	G5,K12	G5,I8,F10,K12,C15
G5	B0,H3	B0,H3,I8,F10,K12,C15
B0	TTKT-DUNG	



# Tìm kiếm leo đồi

- Tìm kiếm leo đồi là tìm kiếm theo độ sâu được hướng dẫn bởi hàm đánh giá
- Khác với tìm kiếm theo độ sâu là khi ta phát triển một đỉnh  $u$  thì bước tiếp theo ta chọn trong số các đỉnh con của  $u$ , đỉnh có nhiều hứa hẹn nhất để phát triển, đỉnh này được xác định bởi hàm đánh giá.
- Về kỹ thuật ta sử dụng một danh sách  $L$  lưu các trạng thái chờ phát triển, sử dụng danh sách  $L_1$  để lưu tạm thời các trạng thái kề của  $u$  khi phát triển  $u$ . Danh sách  $L_1$  được sắp theo thứ tự tăng dần của hàm đánh giá rồi được chuyển vào danh sách  $L$  sao cho trạng thái tốt nhất kề  $u$  đứng ở đầu danh sách  $L$

Cho đồ thị không gian trạng thái, TTĐ: A TTKT: B



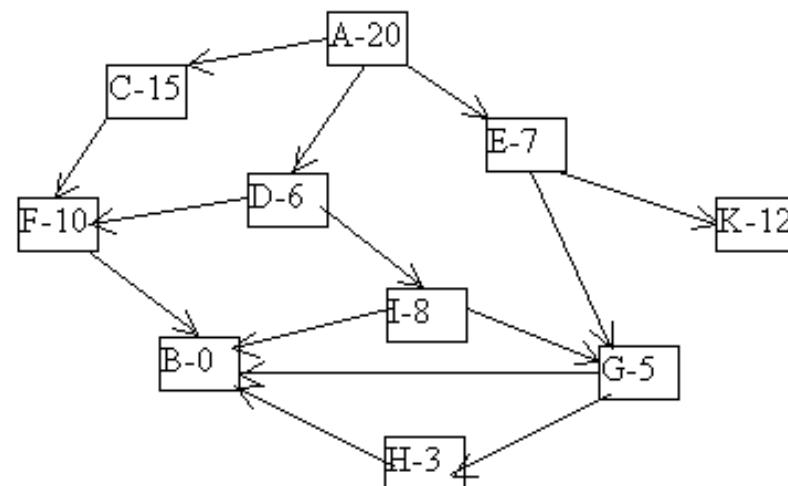
## Procedure TimKiemLeoDoi

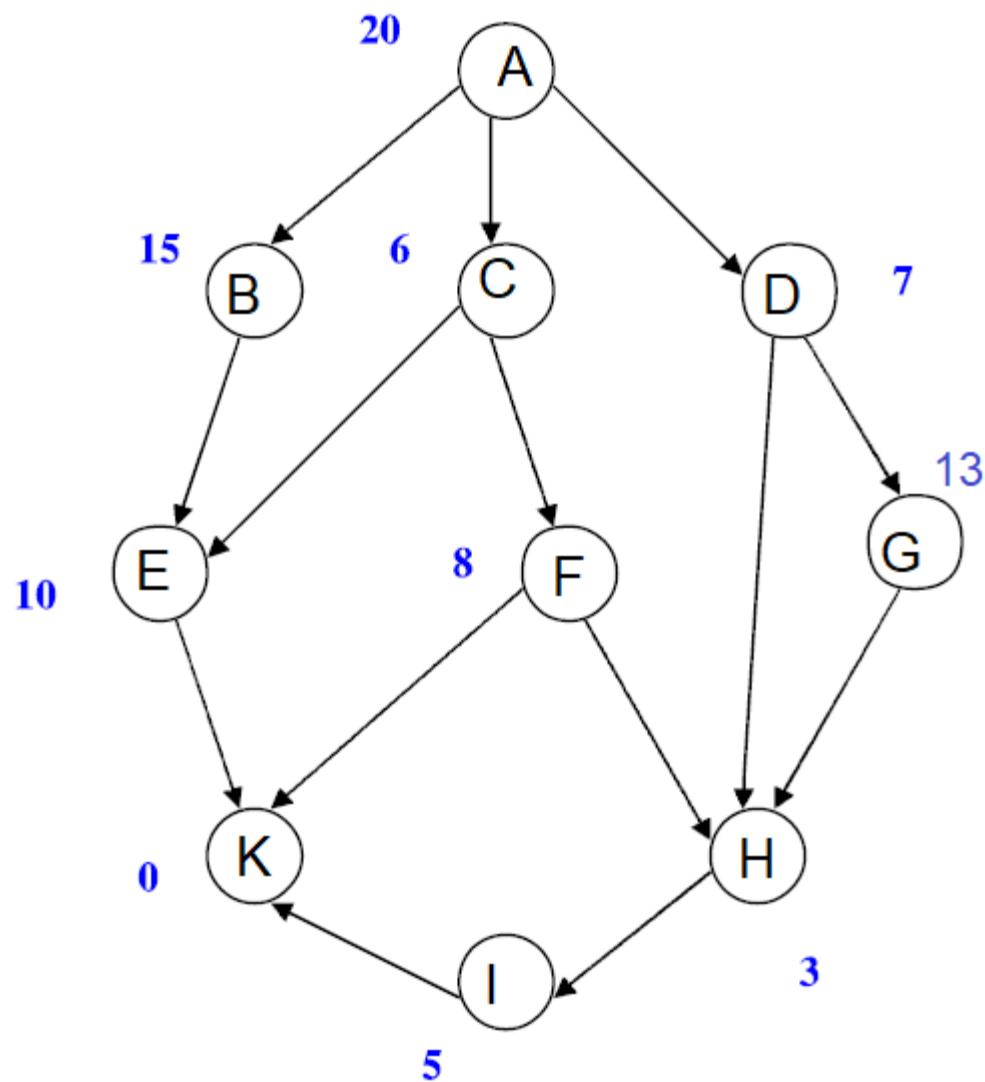
begin

1. Khởi tạo danh sách L chỉ chứa trạng thái ban đầu
2. loop do
  - 2.1 if L rỗng then  
    {thông báo tìm kiếm thất bại; stop};
  - 2.2 Loại trạng thái u ở đầu danh sách L;
  - 2.3 if u là trạng thái kết thúc then  
    {thông báo tìm kiếm thành công; stop};
  - 2.4 for mỗi trạng thái v kề u do  
        Xen v vào danh sách L1 sao cho L1 được sắp theo  
        thứ tự tăng dần của hàm đánh giá;
  - 2.5 Chuyển danh sách L1 vào đầu danh sách L
- end;

# Quá trình tìm kiếm theo PP leo đồi

Phát triển TT	Trạng thái kề	Danh sách L1	Danh sách L
			A20
A20	C15,D6,E7	D6,E7,C15	D6,E7,C15
D6	F10,I8	I8,F10	I8,F10,E7,C15
I8	B0,G5	B0,G5	B0,G5,F10,E7,C15
B0	TTKT-DỨNG		





2	8	3
1	6	4
7		5



1	2	3
8		4
7	6	5