CHƯƠNG 2: CÁC PHƯƠNG PHÁP TÌM KIẾM KINH NGHIỆM (HEURISTIC SEARCH)

2.1. Hàm đánh giá và tìm kiếm kinh nghiệm

2.2. Tìm kiếm tốt nhất đầu tiên (best first search)

2.3. Tìm kiếm leo đồi (hill climbing search)

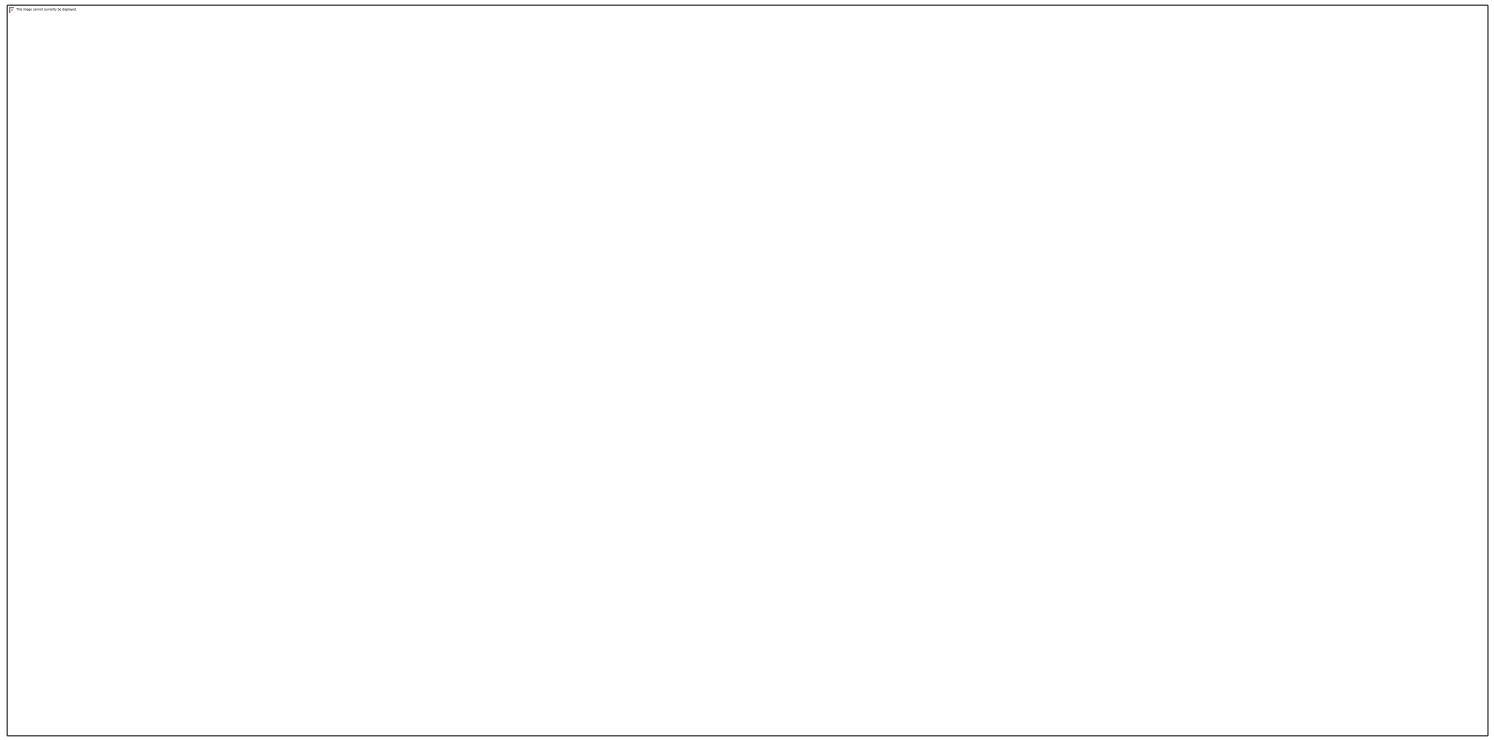
2.4. Tìm kiếm beam (beam search)

2.1. HÀM ĐÁNH GIÁ VÀ TÌM KIẾM KINH NGHIỆM

Nguyên nhân:

- Các chiến lược tìm kiếm mù kém hiệu quả và không thể áp dụng được trong nhiều trường hợp.
- Sử dụng thông tin của trạng thái kết hợp với nhận xét dựa theo kinh nghiệm để cải tiến là quan điểm chung của các chiến lược tìm kiếm kinh nghiệm.

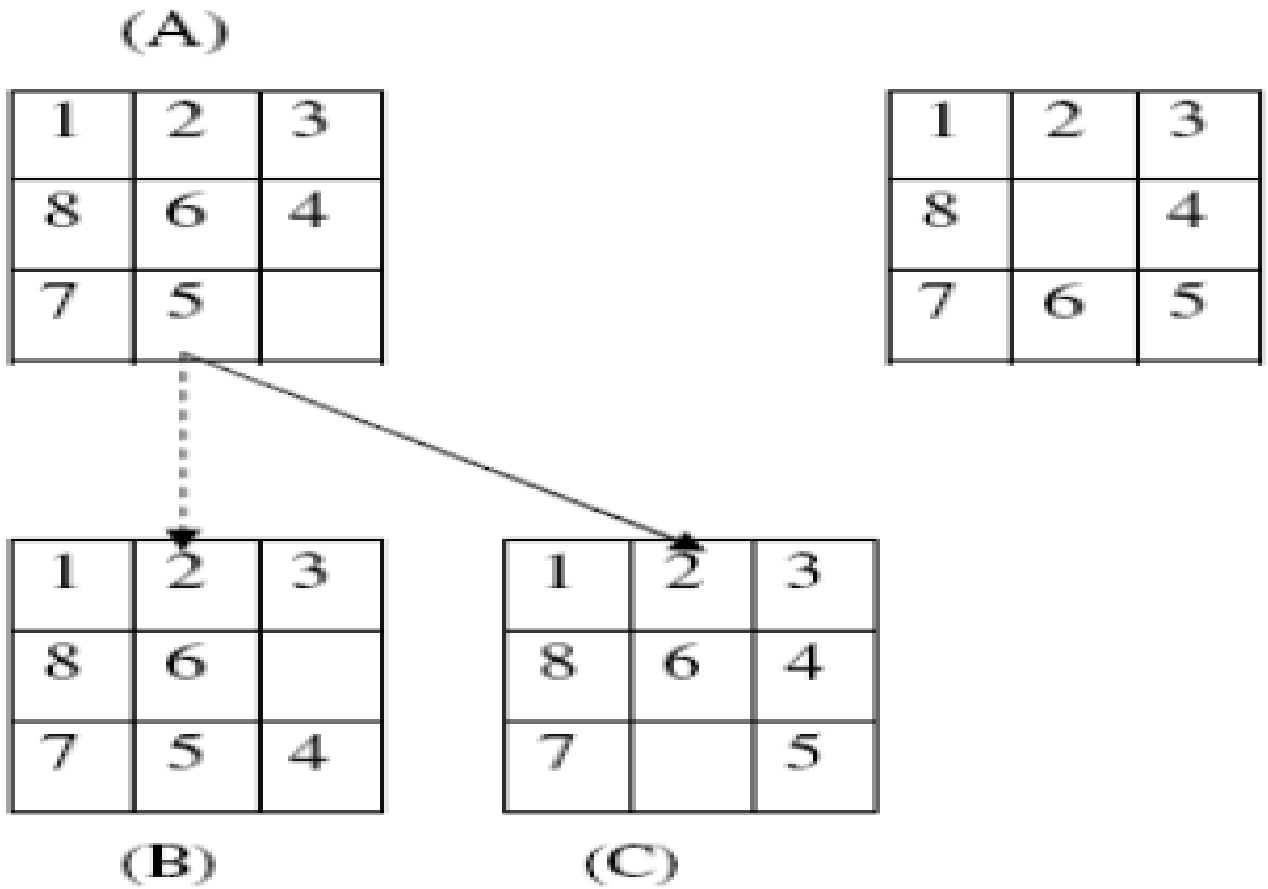
1. HÀM ĐÁNH GIÁ

- 

1. HÀM ĐÁNH GIÁ (KHÁI NIỆM)

 Hàm đánh giá là một hàm ước lượng khả năng về đích của mỗi trạng thái.
 Chỉ là ước lượng vì nói chung chúng ta không thể tính toán chính xác khả năng này, nếu tính được nghĩa là đã tìm được lời giải!
 Tuy nhiên, nhờ kinh nghiệm trong nhiều bài toán cụ thể, có thể ước lượng được g.trị này.

1. HÀM ĐÁNH GIÁ (VÍ DỤ)



HÌNH THỨC HÓA

$$h : X \rightarrow R^{+}$$

- Trong đó *X* là KGTTcủa bài toán.
- Dễ thấy, nếu u là trạng thái đích thì h(u)=0. Mỗi u ta có một giá trị ước lượng là h(u)
- h(B)=3; h(C)=1
- Hàm đánh giá còn được gọi là hàm heuristic. Giá trị hàm càng nhỏ thì khả năng về đích của trạng thái càng lớn

Tính h(D), h(E)

1	5	3
8		4
7	6	2

(D)

1	4	3
8		2
7	6	5

(E)

CÁC BƯỚC XÂY DỰNG CHIẾN LƯỢC TÌM KIẾM

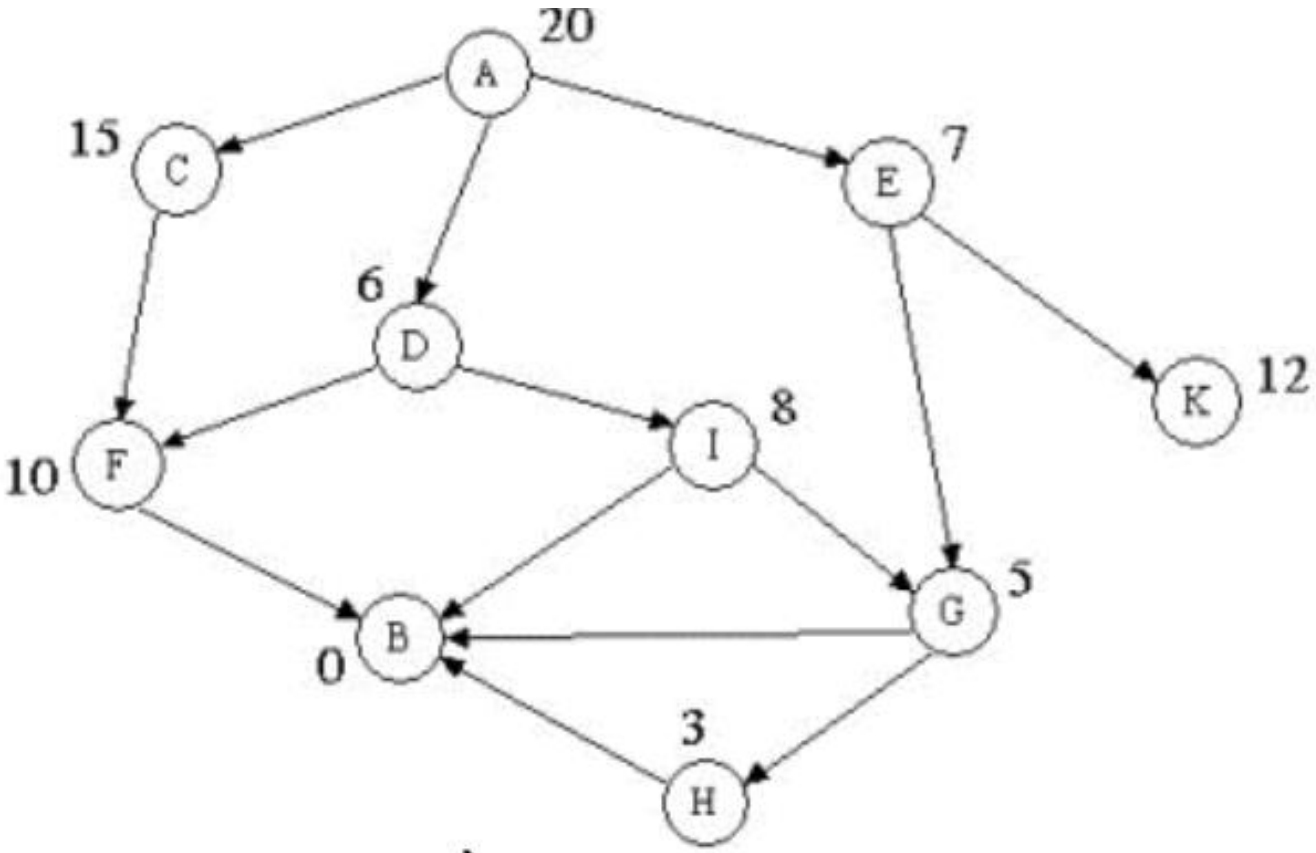
+ Biểu diễn bài toán bằng một KGTT thích hợp
+ Xây dựng hàm đánh giá
+ Thiết kế chiến lược chọn trạng thái

.

II. CÁC CHIẾN LƯỢC TÌM KIẾM KINH NGHIỆM.

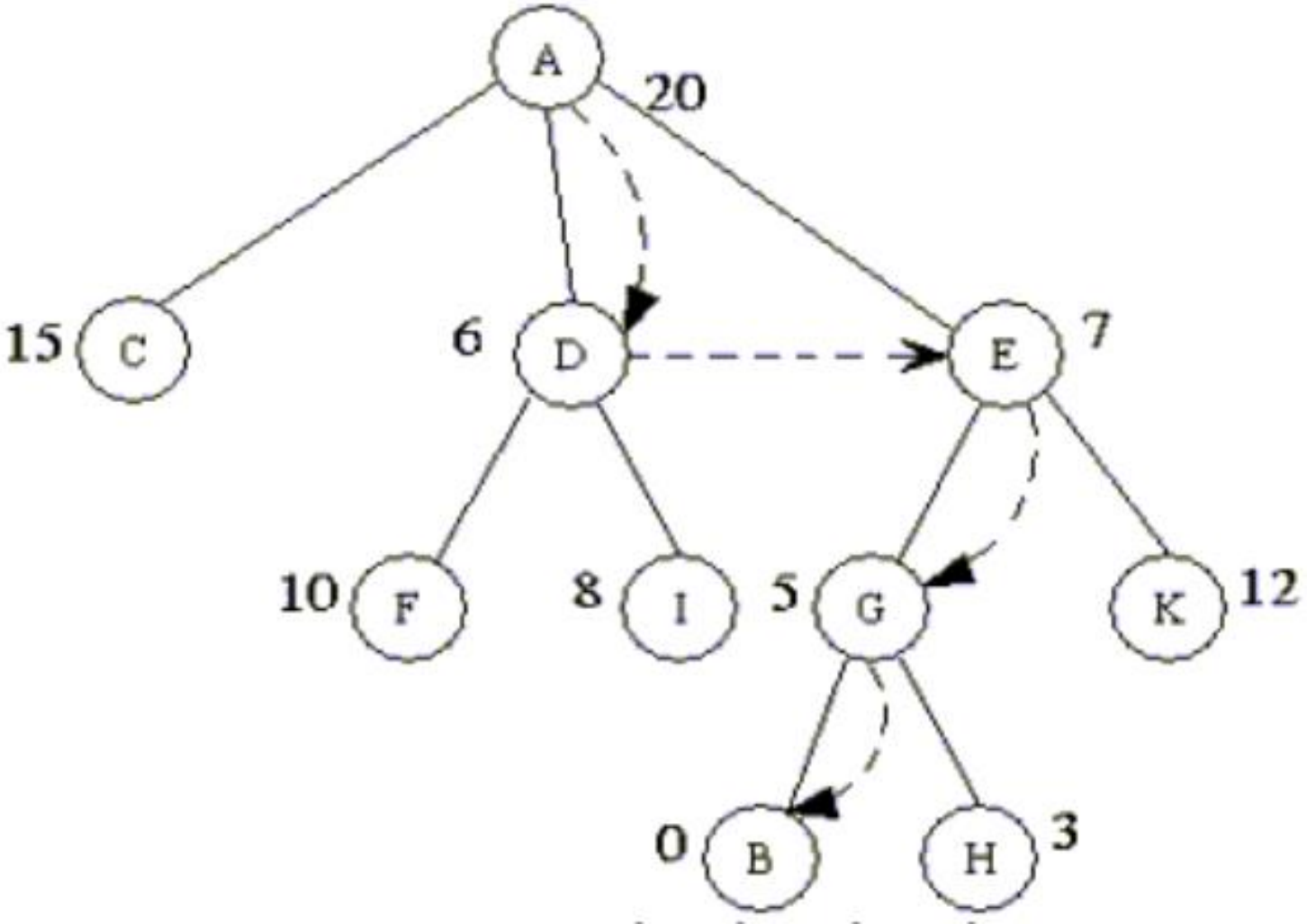
- Tìm kiếm tốt nhất đầu tiên **Best First Search** = Tìm kiếm theo bề rộng + Hàm đánh giá
- Tìm kiếm leo đồi Hill Climbing Search = tìm kiếm theo độ sâu+ hàm đánh giá

2.2. TÌM KIẾM TỐT NHẤT ĐẦU TIÊN **BEST FIRST SEARCH**



Hình 2.2 Đồ thị không gian trạng thái

2.2. TÌM KIẾM TỐT NHẤT ĐẦU TIÊN **BEST FIRST SEARCH**



2.2. TÌM KIẾM TỐT NHẤT ĐẦU TIÊN **BEST FIRST SEARCH**

Ý tưởng:

Tìm kiếm tốt nhất đầu tiên = Tìm kiếm theo chiều rộng + Hàm đánh giá.

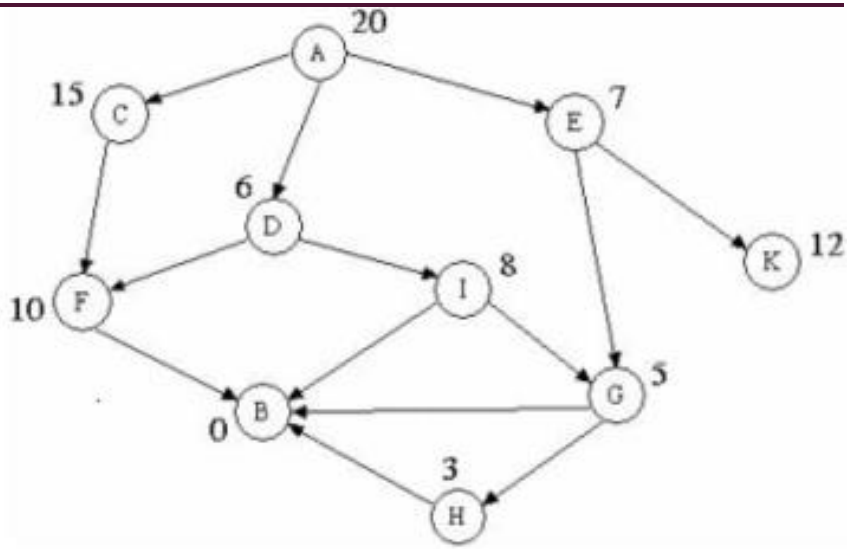
Ý nghĩa: khác với phương pháp tìm kiếm theo chiều rộng, các node không được phát triển lần lượt mà được lựa chọn dựa trên hàm đánh giá (tốt nhất), đỉnh này có thể ở mức hiện tại hoặc ở mức trên.

Cài đặt: Dùng hàng đợi ưu tiên. Sắp xếp các node trong hàng theo thứ tự giảm dần của hàm đánh giá.

Một số dạng mở rộng:

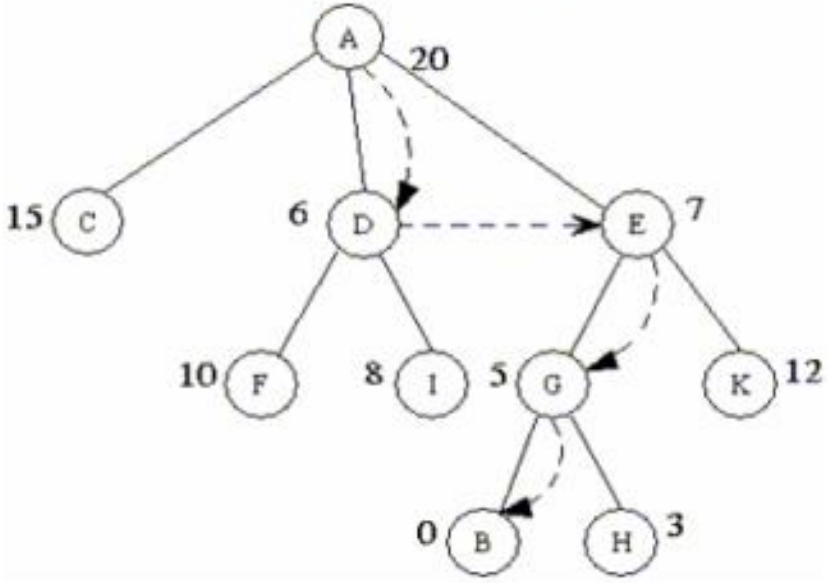
- Tìm kiếm tham lam tốt nhất đầu tiên (Greedy best-first search).
- Tìm kiếm A* (A* search).

VÍ DỤ



Xét trạng thái ban đầu là A;
Trạng thái kết thúc là B
Thực hiện:
A được xét → C, D, E
Chọn D vì h(D)=6 min, sinh ra F, L
Trong số các đỉnh chưa xét: C, E, F, L chọn E vì h(E)= 7
Sinh ra G và K
Chọn G để phát triển vì h(G)=5 min, sinh ra B và H
B là trạng thái kết thúc.

Xét không gian trạng thái sau



2.2. TÌM KIẾM TỐT NHẤT ĐẦU TIÊN **BEST FIRST SEARCH**- CÀI ĐẶT

Procedure Best_First_Search;

Begin

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

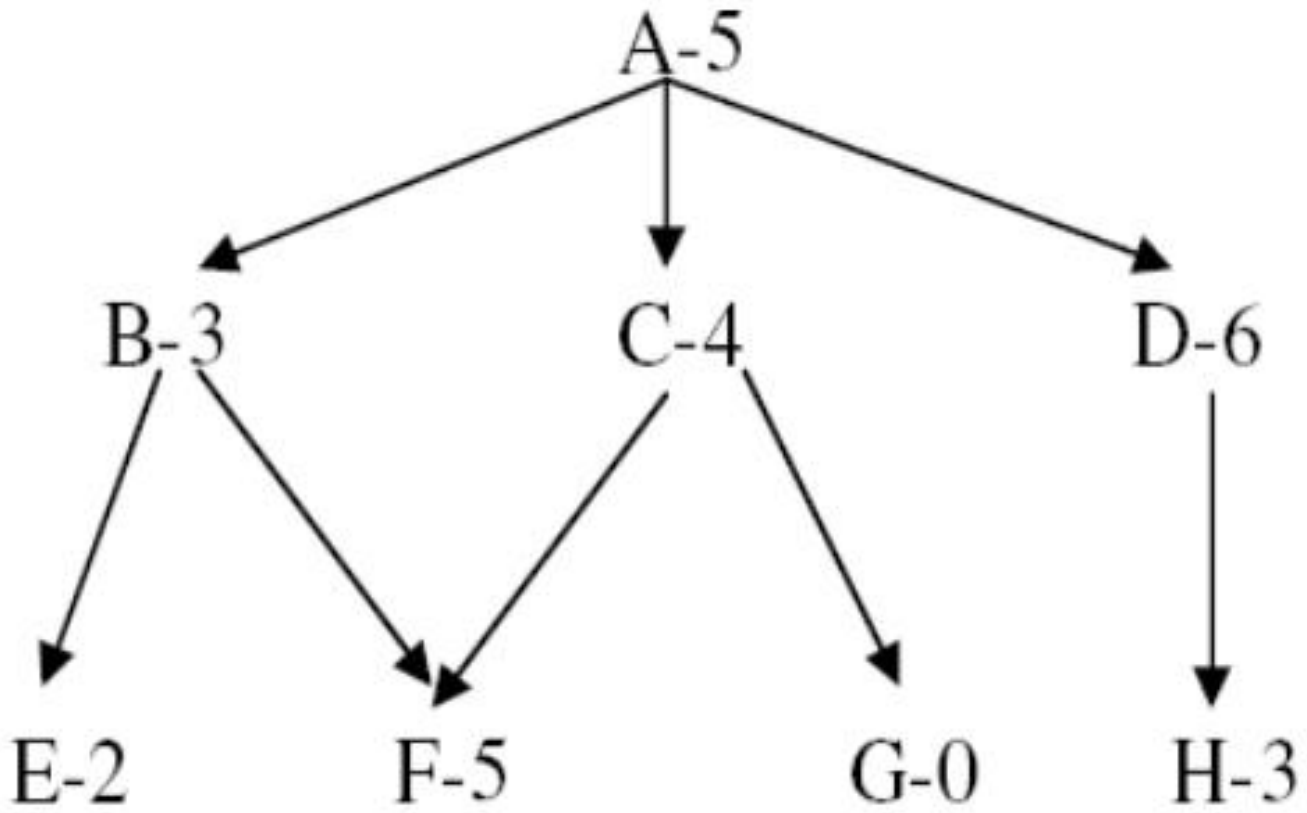
2. **Loop do**

    1. **If** L rỗng **then** { thông báo thất bại; **stop**; }
    2. Loại trạng thái u ở đầu danh sách L;
    3. **If** u là trạng thái kết thúc **then** { thông báo thành công; stop; }
    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;

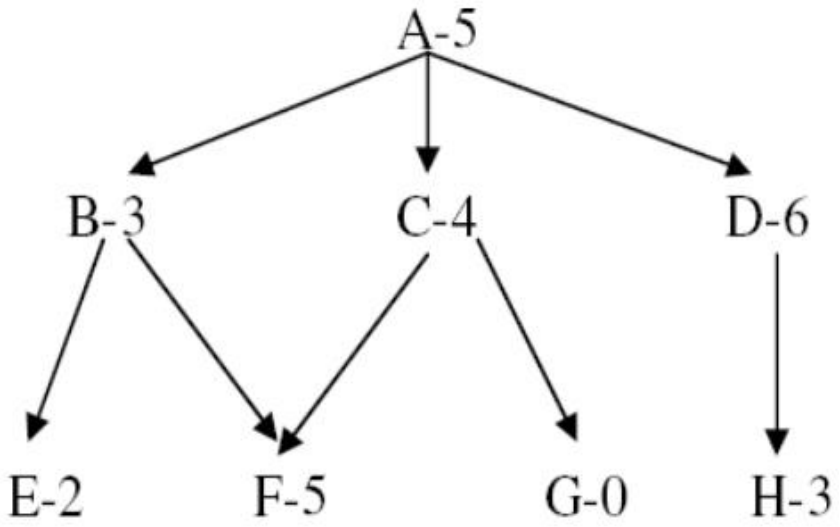
3. **End**;

2.2. TÌM KIẾM TỐT NHẤT ĐẦU TIÊN **BEST FIRST SEARCH**- VÍ DỤ



2.2. TÌM KIẾM TỐT NHẤT ĐẦU TIÊN **BEST FIRST SEARCH**- VÍ DỤ

Bước 1	Open	Close (bổ sung mỗi thời điểm)
1	A-5	
2	B-3, C-4, D-6	A-5
3	E-2, C-4, F-5,D-6	B-3
4	C-4, F-5,D-6	E-2
5	G-0 ,F-5,D-6	C-4



2.3. TÌM KIẾM LEO ĐỒI (HILL CLIMBING SEARCH)

- **Ý tưởng:**

  Tìm kiếm leo đồi = tìm kiếm theo chiều sâu + hàm đánh giá

- **Ý nghĩa:**
  - Phương pháp này được thực hiện nhờ hàm đánh giá.
  - Khác với phương pháp tìm kiếm theo chiều sâu, khi phát triển đỉnh **u**, chọn trong số các đỉnh con của u, đỉnh nào có nhiều hứa hẹn nhất thì phát triển.

- **Cài đặt:**
  - Sử dụng ngăn xếp có ưu tiên.

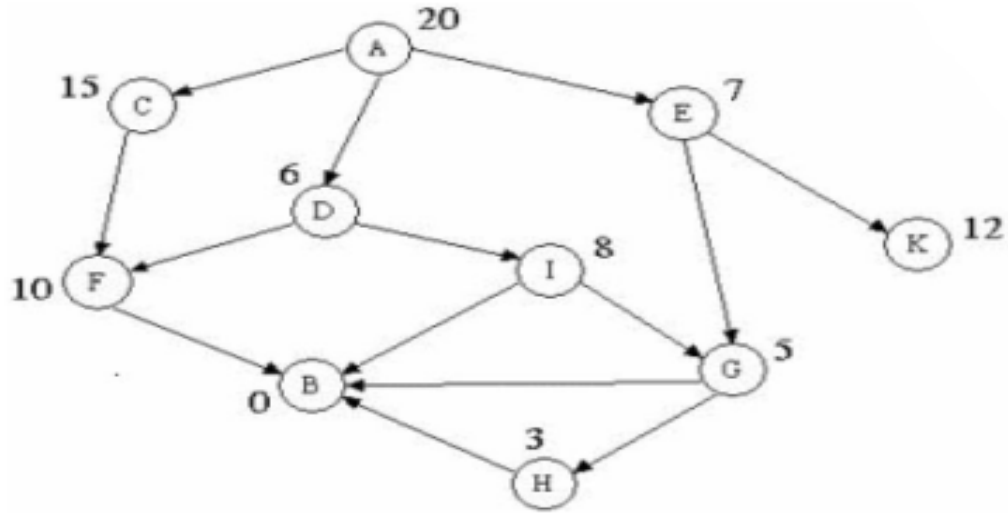
Ví dụ về Hill-climbing search

Đầu vào:

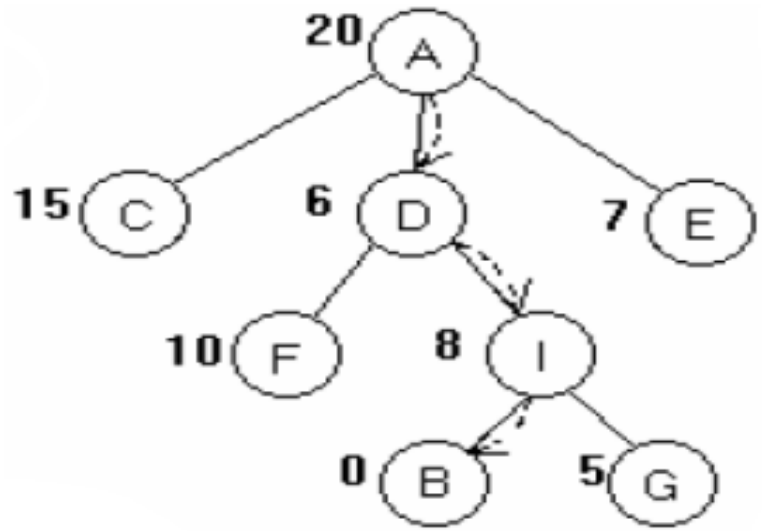
- Trạng thái đầu là A,
- Trạng thái kết thúc là B.

Thực hiện:

- A được xét → C, D, E.
- Chọn D, vì h(D) = 6 (min), sinh ra F,I.
- Trong số các đỉnh con của D, chọn I, vì h(I) = 8.
- Với I được chọn, sinh ra B và G.
- B là trạng thái kết thúc.



Xét không gian trạng thái sau



2.3. TÌM KIẾM LEO ĐỒI (HILL CLIMBING SEARCH)

Procedure Hill-Climbing_search;

Begin

1. Khởi tạo ngăn xếp S chỉ chứa trạng thái đầu;
2. **Loop do**
   - 2.1 **If** S rỗng **then** {thông báo thất bại; stop};
   - 2.2 Lấy trạng thái u ở đầu ngăn xếp S;
   - 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** đặt v vào danh sách L;
   - 2.5 Sắp xếp L theo thứ tự tăng dần của hàm đánh giá sao cho trạng thái tốt nhất ở đầu danh sách L;
   - 2.6 Chuyển danh sách Lvào ngăn xếp S;

End;

2.6. TÌM KIẾM CHÙM (BEAM SEARCH)

- **Ý tưởng:**

  Tìm kiếm theo chiều rộng + k node để phát triển + hàm đánh giá

- **Ý nghĩa:**
  - Tìm kiếm beam giống tìm kiếm theo chiều rộng,
  - Tuy nhiên trong tìm kiếm beam, hạn chế **k** đỉnh tốt nhất để phát triển.
  - Như vậy, số đỉnh cần phát triển ở mức **d** là k^d.

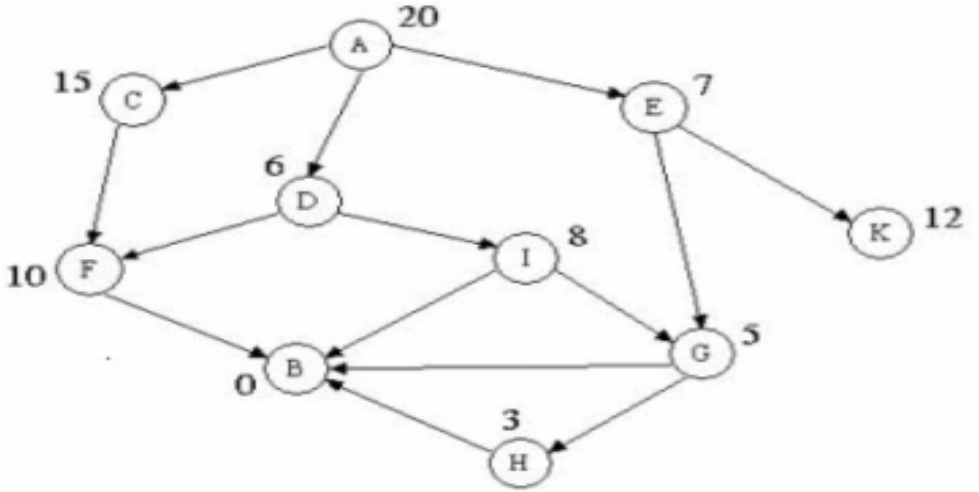
2.6. TÌM KIẾM CHÙM (BEAM SEARCH)

Đầu vào:

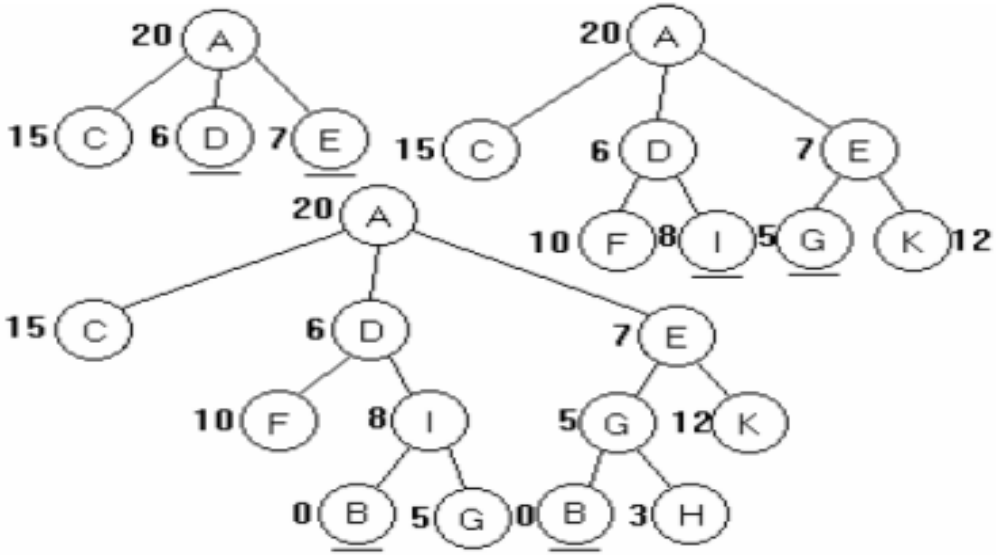
- Trạng thái đầu là A,
- Trạng thái kết thúc là B.

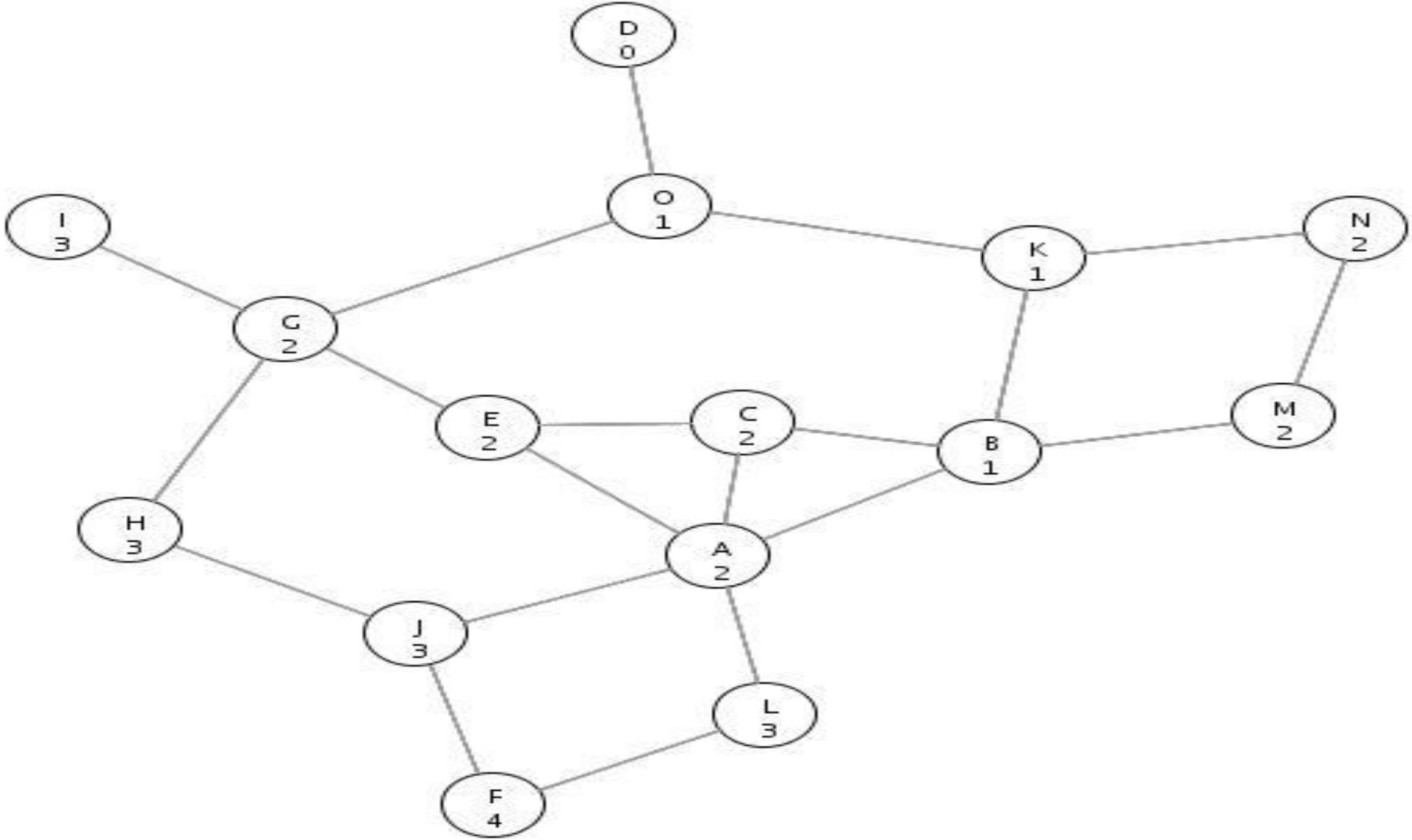
Thực hiện:

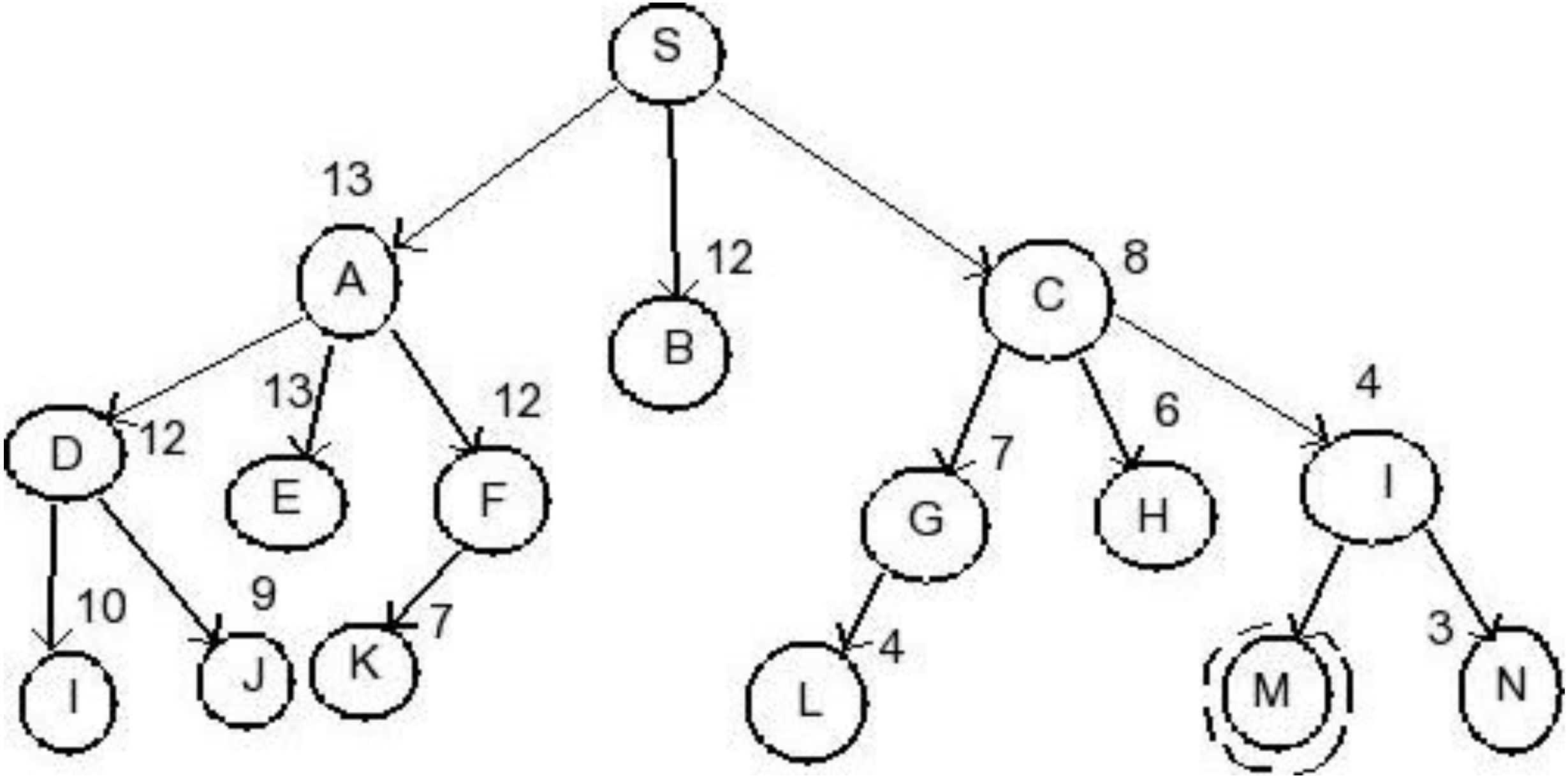
- Ví dụ lấy k = 2.
- A được xét → C, D, E.
- Chọn D và E để phát triển, với D sinh ra F và I, với E sinh ra G và K.
- Chọn I và G để phát triển, sinh ra B, G, B, H
- Chọn được B (qua I) và B (qua G).
- B là trạng thái kết thúc.



Xét không gian trạng thái sau




D
0
O
1
I
3
K
1
N
2
G
2
E
2
C
2
B
1
M
2
H
3
A
2
J
3
L
3
F
4


S
13
A
12
B
8
C
13
12
D
12
E
F
6
4
7
G
H
I
10
9
7
4
3
I
J
K
L
M
N