Trí tuệ nhân tạo

Tuần 4

Giảng viên: Trần Đức Minh

Nội dung trình bày

- Tổng quan về tìm kiếm kinh nghiệm
- Hàm đánh giá
- Các giai đoạn cơ bản trong vấn đề tìm kiếm heuristic
- Chiến lược tìm kiếm tốt nhất - đầu tiên
- Chiến lược tìm kiếm tham lam tốt nhất - đầu tiên
- Chiến lược tìm kiếm leo đồi
- Chiến lược tìm kiếm beam

Tổng quan

- Các kỹ thuật liên quan đến tìm kiếm mù thường rất kém hiệu quả và trong nhiều trường hợp không thể áp dụng được.
- Để cải thiện việc tìm kiếm ta sẽ sử dụng các phương pháp **tìm kiếm kinh nghiệm** (tìm kiếm heuristic)
  - Đây là các phương pháp có sử dụng **hàm đánh giá** để hướng dẫn sự tìm kiếm.

Hàm đánh giá

- Trong tìm kiếm kinh nghiệm, tương ứng với mỗi trạng thái **u** ta xác định một giá trị **h(u)** thông qua ánh xạ **h**.
  - Giá trị **h(u)** này là một con số được dùng để đánh giá trạng thái u là **"tốt" ở mức độ nào**. Cụ thể, giá trị này được dùng để **đánh giá "sự gần đích"** của trạng thái **u**.
  - Ánh xạ **h** được gọi là hàm đánh giá.

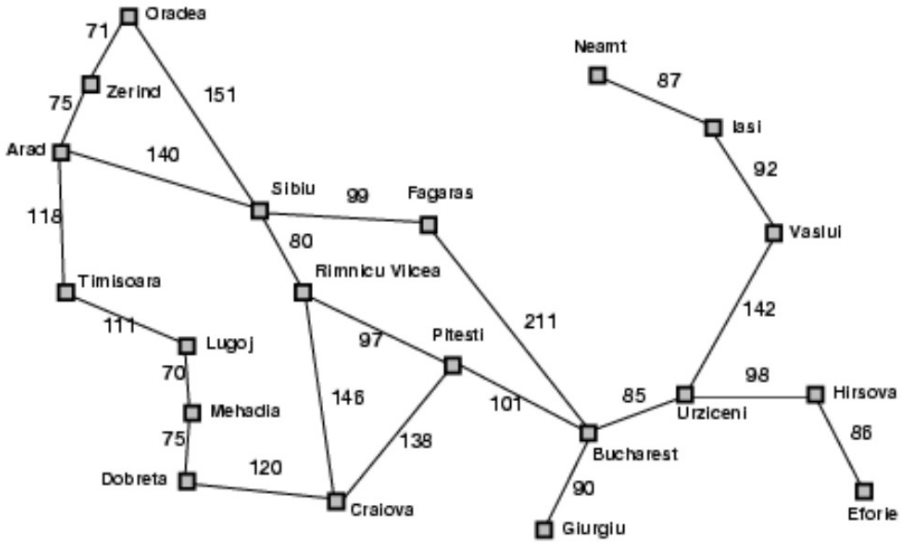
$$\mathbf{h : u \rightarrow h(u)}$$

Hàm đánh giá

- Trong tìm kiếm kinh nghiệm, hàm đánh giá đóng một vai trò rất quan trọng.
- Hàm đánh giá được xây dựng đúng bản chất vấn đề thì việc tìm kiếm mới có hiệu quả.
- Nếu việc xây dựng hàm đánh giá không chính xác, có thể dẫn đến việc tìm kiếm bị chệch hướng và do đó kém hiệu quả.
- Việc xây dựng hàm đánh giá phụ thuộc vào
  - Bài toán cần giải quyết
  - Kinh nghiệm của chuyên gia liên quan đến bài toán cần giải quyết.

Hàm đánh giá

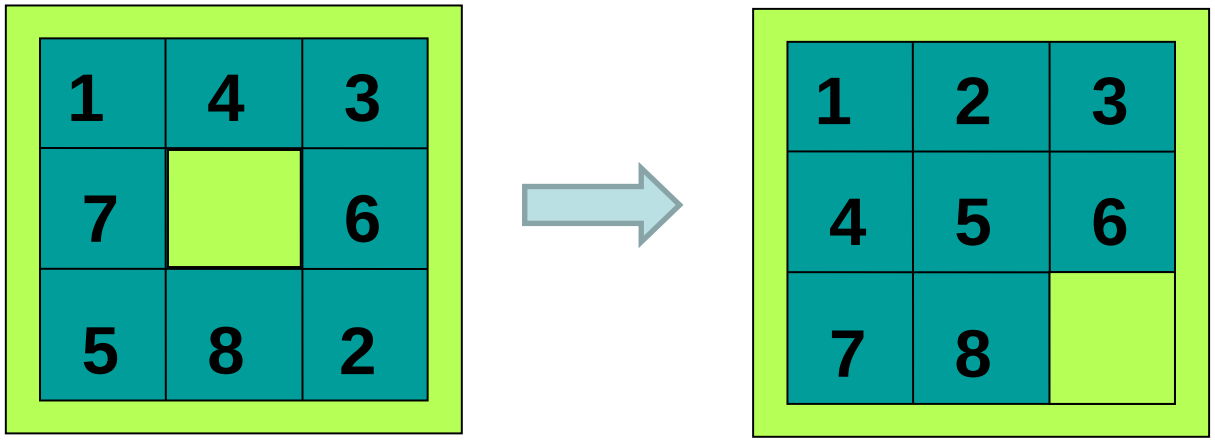
- Ví dụ 1: Trong bài toán tìm kiếm đường đi trên bản đồ giao thông, ta có thể lấy **độ dài đường chim bay** từ một thành phố tới thành phố đích đến làm giá trị của hàm đánh giá.



Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Hàm đánh giá

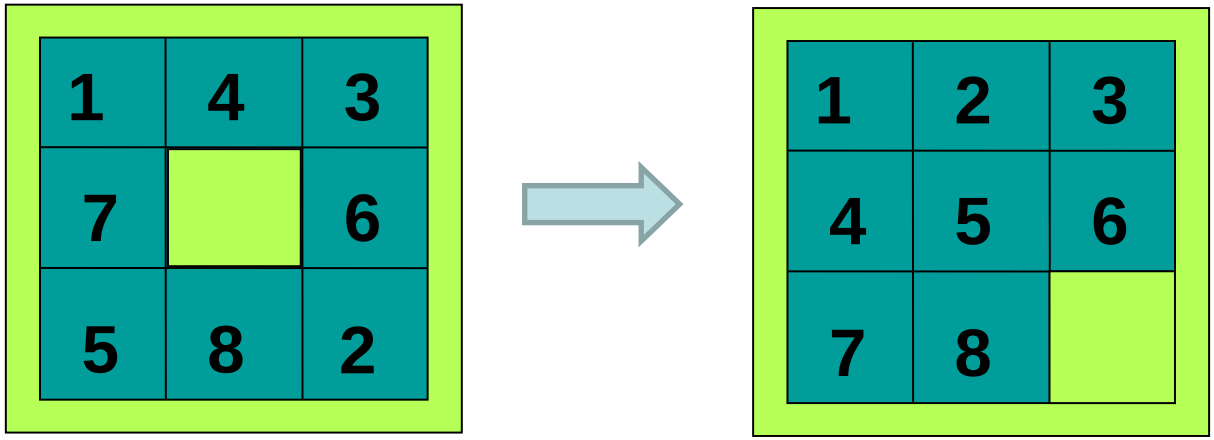
- Ví dụ 2: Trong bài toán Ta Canh, **hàm đánh giá h_1** đối với **trạng thái u** là số ô không nằm đúng vị trí của nó trong trạng thái đích



$h_1(u) = 4$

Hàm đánh giá

- Ví dụ 3: Trong bài toán Ta Canh, **hàm đánh giá h_2** đối với **trạng thái u** là tổng khoảng cách giữa vị trí của các ô trong u và vị trí của nó trong trạng thái đích



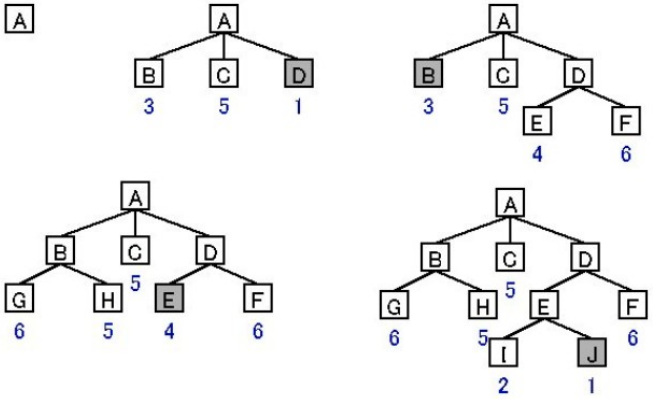
$h_2(u) = 3 + 2 + 2 + 1 = 8$

Các giai đoạn cơ bản trong vấn đề tìm kiếm heuristic

- Các giai đoạn
  - ***Giai đoạn 1***: Tìm cách biểu diễn thích hợp để mô tả các trạng thái và các toán tử của vấn đề.
  - ***Giai đoạn 2***: Xây dựng hàm đánh giá trạng thái.
  - ***Giai đoạn 3***: Thiết kế chiến lược chọn trạng thái để phát triển ở mỗi bước
- Ví dụ: Trong bài toán Ta Canh
  - ***Giai đoạn 1***:
    - Biểu diễn trạng thái của bài toán bằng một **ma trận n x n**, trong đó có **n^2-1 các ô số** và **1 ô trống**.
    - Các toán tử chuyển trạng thái **Up**, **Down**, **Left**, **Right** dùng để chỉ hướng di chuyển của ô trống.
  - ***Giai đoạn 2***: Sử dụng 1 trong 2 hàm đánh giá **h_1** hoặc **h_2**, hoặc sử dụng cả 2 hàm đánh giá.
  - ***Giai đoạn 3***: Lựa chọn chiến lược tìm kiếm heuristic.

Chiến lược tìm kiếm tốt nhất - đầu tiên

- **Chiến lược tìm kiếm tốt nhất - đầu tiên** (Best-first search) là chiến lược **tìm kiếm theo bề rộng** và được hướng dẫn bởi **hàm đánh giá**.
  - Cụ thể hơn, chiến lược này chọn trạng thái để phát triển là trạng thái tốt nhất được xác định bởi hàm đánh giá, trạng thái này có thể nằm ở mức hiện tại hoặc ở mức trên.



Thuật toán tìm kiếm tốt nhất - đầu tiên

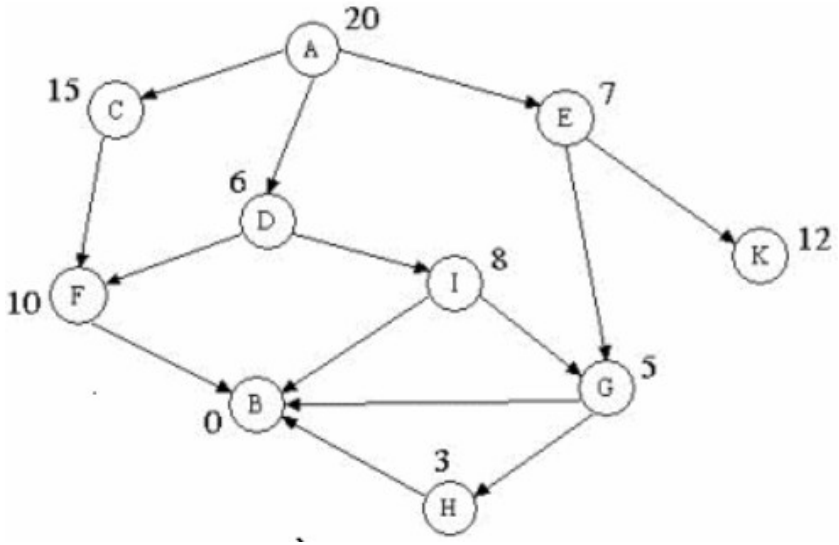
- Ta sử dụng danh sách **open** để lưu giữ các trạng thái được sinh ra nhưng chưa được khảo sát.
- Hàm **father(X)** dùng để lưu lại cha của đỉnh X trên đường đi.
  - Ví dụ: Cú pháp **father(X)** ← **Y** tức là đỉnh cha của đỉnh X là đỉnh Y
- Xác định trạng thái bắt đầu **S**
- Xác định tập các trạng thái kết thúc **GD**
- Xác định các toán tử chuyển trạng thái

Thuật toán tìm kiếm tốt nhất - đầu tiên

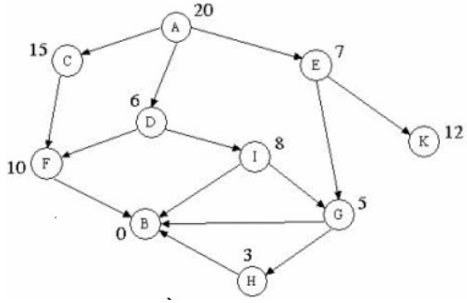
- **BEST-FIRST-SEARCH**
  - Đưa trạng thái bắt đầu **S** vào **open**;
  - **loop do**
    - Nếu **open =** $\varnothing$ thì thông báo Không tìm thấy kết quả. **Kết thúc thuật toán**.
    - Lấy 1 trạng thái ở đầu danh sách open rồi đưa vào biến trạng thái **X**, sau đó loại bỏ trạng thái đó khỏi danh sách.
    - Nếu **X** $\in$ **GD** thì thông báo tìm kiếm thành công. **Kết thúc thuật toán**.
    - Nếu **X** $\notin$ **GD**
      - Với mỗi trạng thái $\mathbf{Y_i}$ được sinh ra bởi trạng thái **X** thông qua các toán tử
        - Đưa trạng thái $\mathbf{Y_i}$ vào danh sách **open** rồi sắp xếp các trạng thái trong danh sách theo hàm đánh giá từ tốt nhất đến xấu nhất
        - **father(Y_i)** $\leftarrow$ **X**
  - **end loop**

Thuật toán tìm kiếm tốt nhất - đầu tiên

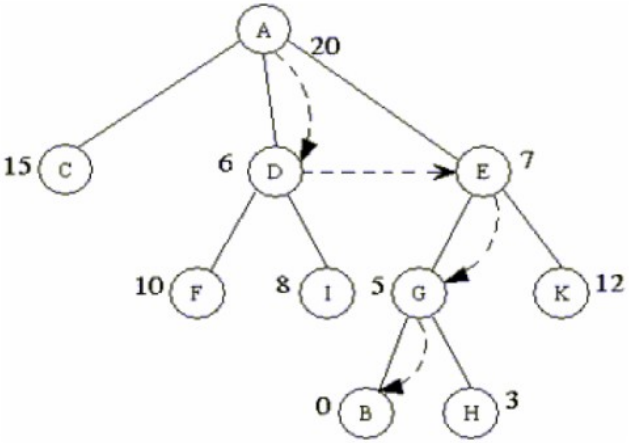
- Ví dụ: Sử dụng thuật toán Best-First Search để duyệt đồ thị bên dưới. Biết rằng
  - A là trạng thái bắt đầu
  - B là trạng thái kết thúc
  - Đầu mũi tên chỉ trạng thái được sinh bởi trạng thái ở đuôi mũi tên.
  - Con số ghi phía trên trạng thái là giá trị của hàm đánh giá.
  - Giả sử trong ví dụ này ta coi hàm đánh giá có giá trị càng nhỏ càng tốt.



Thuật toán tìm kiếm tốt nhất - đầu tiên



- open=[A^{20}];
- X=A^{20}; open=[D^{6},E^{7},C^{15}]; father[D^{6},E^{7},C^{15}]=A^{20};
- X=D^{6}; open=[E^{7},I^{8},F^{10},C^{15}]; father[I^{8},F^{10}]=D^{6};
- X=E^{7}; open=[G^{5},I^{8},F^{10},K^{12},C^{15}]; father[G^{5},K^{12}]=E^{7};
- X=G^{5}; open=[B^{0},H^{3},I^{8},F^{10},K^{12},C^{15}]; father[B^{0},H^{3}]=G^{5};
- X=B^{0}; => Tìm kiếm thành công

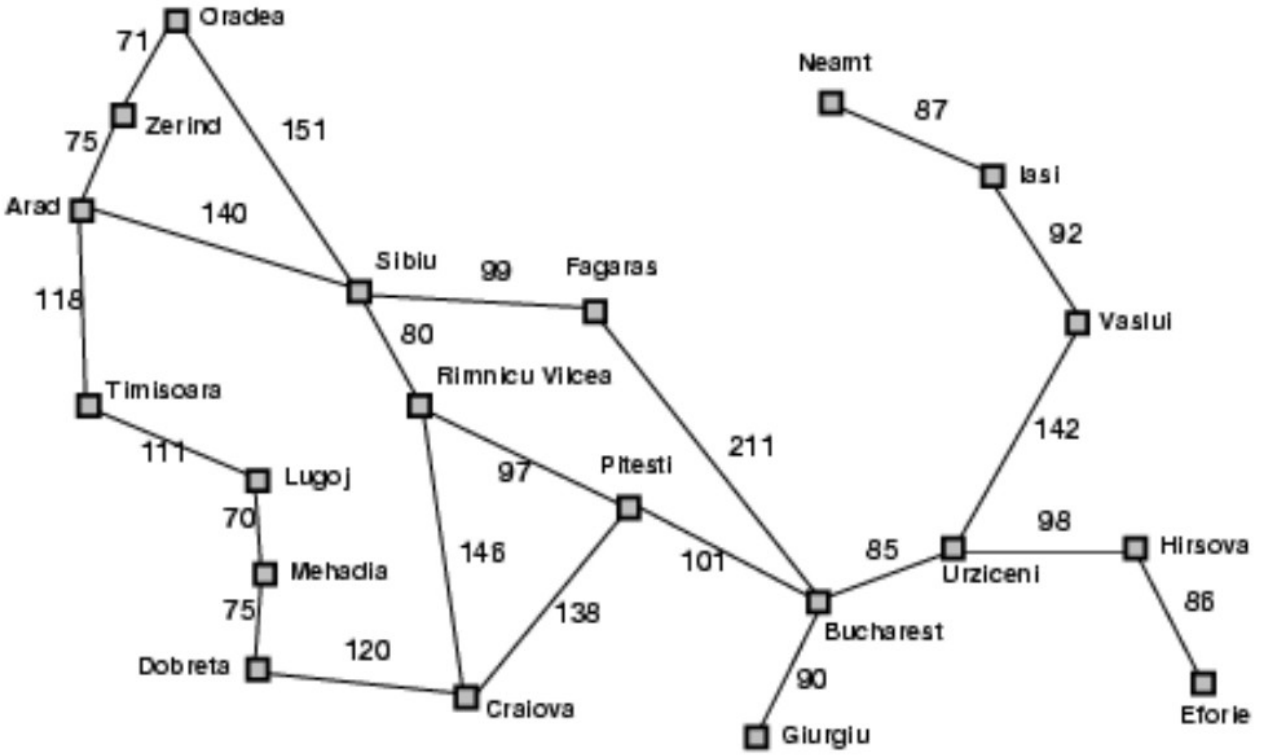


Chiến lược tìm kiếm tham lam tốt nhất - đầu tiên

- **Chiến lược tìm kiếm tham lam tốt nhất - đầu tiên** (Greedy best-first search) là chiến lược tìm kiếm tốt nhất - đầu tiên với **hàm đánh giá** được sử dụng để **đánh giá chi phí đi từ trạng thái hiện tại đến trạng thái đích**.
  - Chiến lược này phát triển các nút được đánh giá gần với nút đích nhất, đó là những nút có ước lượng bởi hàm đánh giá tốt nhất.

Thuật toán tìm kiếm tham lam tốt nhất - đầu tiên

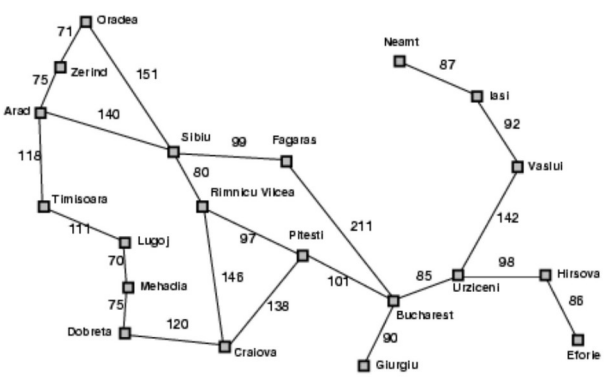
- Ví dụ: Tìm đường đi từ thành phố Arad đến thành phố Bucharest



Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Thuật toán tìm kiếm tham lam tốt nhất - đầu tiên

- Ví dụ: Tìm đường đi từ thành phố Arad đến thành phố Bucharest
  - Thành phố Arad là trạng thái bắt đầu
  - Thành phố Bucharest là trạng thái kết thúc
  - Các con số nằm phía bên phải hình vẽ là khoảng cách theo đường chim bay từ các thành phố đến Bucharest. Ở ví dụ này ta coi chúng là giá trị của hàm đánh giá.

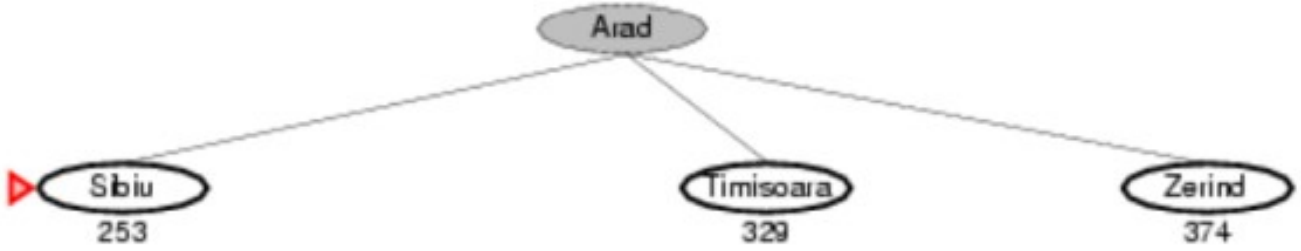


Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vashui	199
Zerind	374

Thuật toán tìm kiếm tham lam tốt nhất - đầu tiên

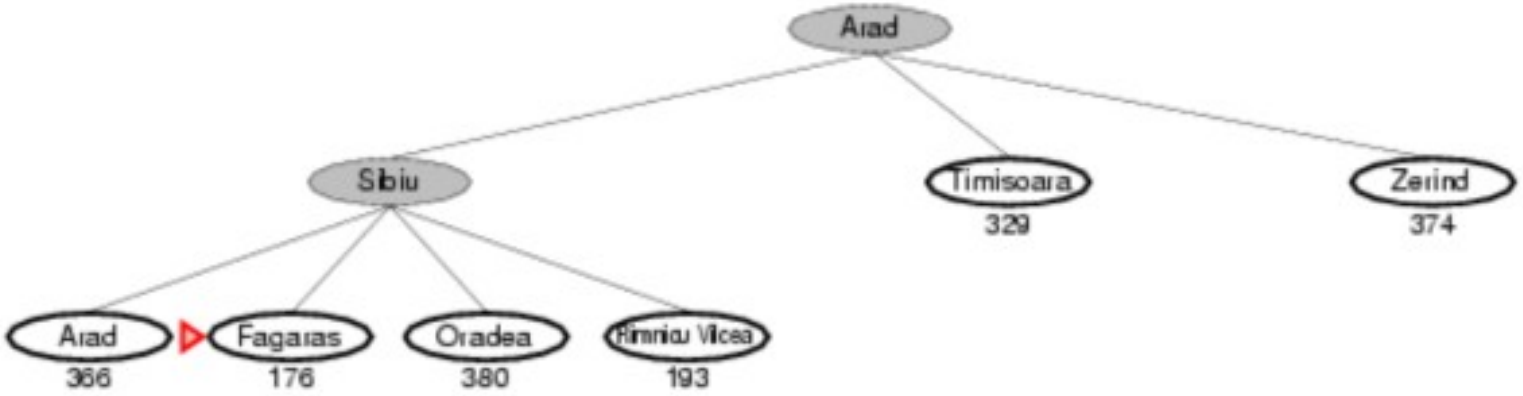
- open=[Arad^{366}];

- X=Arad^{366};
  open=[Sibiu^{253},Timisoara^{329},Zerind^{374}];
  father[Sibiu^{253},Timisoara^{329},Zerind^{374}]=Arad^{366};



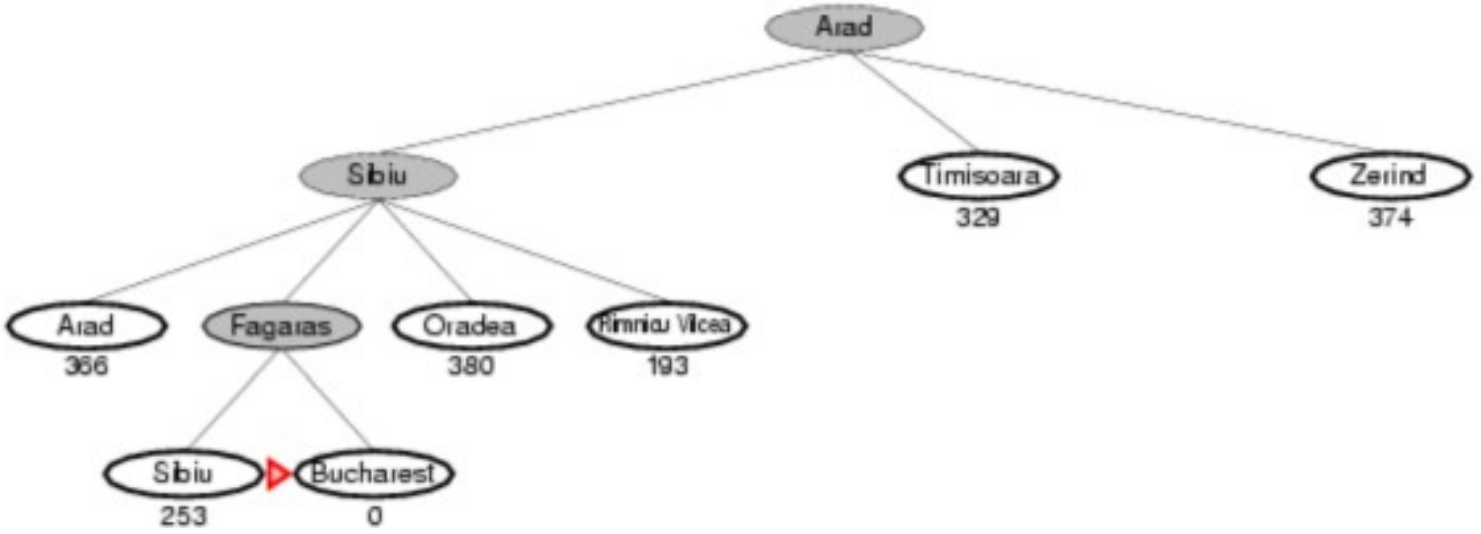
Thuật toán tìm kiếm tham lam tốt nhất - đầu tiên

- X=Sibiu[253]; open=[Fagaras[176],Rimnisu Vilcea[193],Timisoara[329],Zerind[374]]; father[Fagaras[176],Rimnisu Vilcea[193]]=Sibiu[253];



Thuật toán tìm kiếm tham lam tốt nhất - đầu tiên

- X=Fagaras^{176}; open=[Bucharest^{0},Rimnisu Vilcea^{193},Timisoara^{329},Zerind^{374}]; father[Bucharest^{0}]=Fagaras^{176};

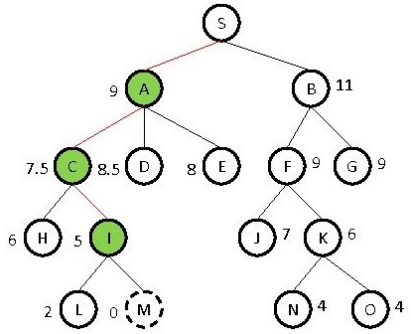


- X=Bucharest^{0}; => Tìm kiếm thành công

Thuật toán tìm kiếm tham lam tốt nhất - đầu tiên

- **Đánh giá**
  - Tính đầy đủ: Lời giải bài toán chưa chắc đã tìm được vì có thể vào vòng lặp luẩn quẩn.
  - Độ phức tạp về thời gian: $O(b^m)$
    - Tuy nhiên nếu có hàm đánh giá tốt thì thời gian này sẽ giảm đi khá nhiều.
  - Độ phức tạp về không gian lưu trữ: $O(b^m)$
  - Tính tối ưu: Không cho lời giải tối ưu do chưa chắc đã chọn được trạng thái kết thúc gần nhất với trạng thái bắt đầu.

Chiến lược tìm kiếm leo đồi



- **Chiến lược tìm kiếm leo đồi** (Hill-climbing search) là chiến lược **tìm kiếm theo chiều sâu** và được hướng dẫn bởi **hàm đánh giá**.
  - Cụ thể hơn, trong chiến lược này, khi phát triển một đỉnh u, thì bước tiếp theo ta chọn trong các trạng thái được sinh ra bởi u, trạng thái nào được coi là tốt nhất thì trạng thái đó sẽ được lựa chọn để phát triển tiếp.
    - Việc đánh giá các trạng thái được thực hiện thông qua hàm đánh giá.

Thuật toán tìm kiếm leo đồi

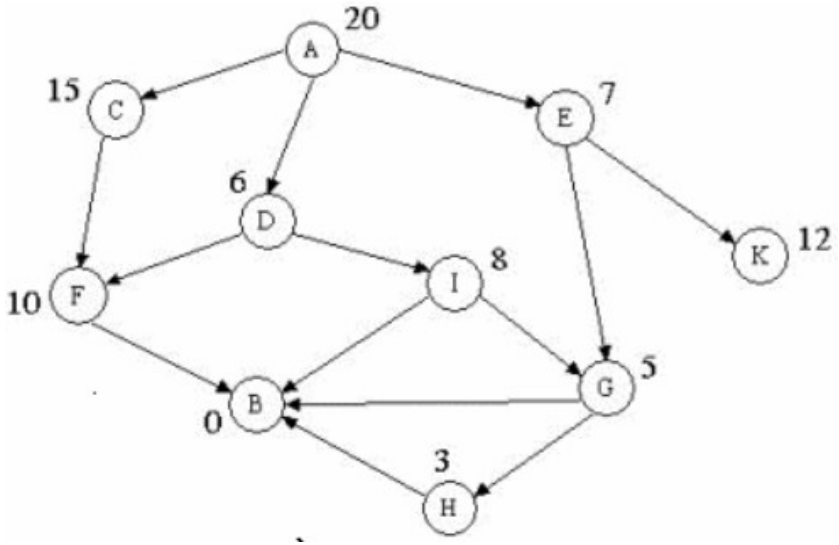
- Ta sử dụng danh sách **open** để lưu giữ các trạng thái được sinh ra nhưng chưa được khảo sát.
- Danh sách **temp** được dùng để lưu giữ các trạng thái được sinh ra bởi một trạng thái nào đó thông qua các toán tử.
- Hàm **father(X)** dùng để lưu lại cha của đỉnh X trên đường đi.
  - Ví dụ: Cú pháp **father(X)** ← **Y** tức là đỉnh cha của đỉnh X là đỉnh Y
- Xác định trạng thái bắt đầu **S**
- Xác định tập các trạng thái kết thúc **GD**
- Xác định các toán tử chuyển trạng thái

Thuật toán tìm kiếm leo đồi

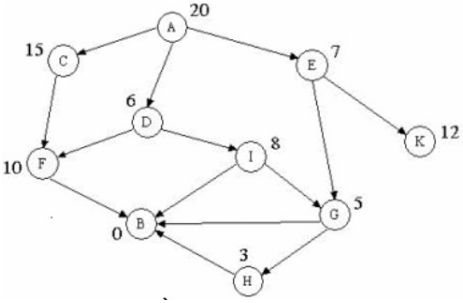
- **HILL-CLIMBING-SEARCH**
  - Đưa trạng thái bắt đầu **S** vào **open**;
  - **loop do**
    - Nếu **open =** $\varnothing$ thì thông báo Không tìm thấy kết quả. **Kết thúc thuật toán**.
    - Lấy 1 trạng thái ở đầu danh sách open rồi đưa vào biến trạng thái **X**, sau đó loại bỏ trạng thái đó khỏi danh sách.
    - Nếu **X** $\in$ **GD** thì thông báo tìm kiếm thành công. **Kết thúc thuật toán**.
    - Nếu **X** $\notin$ **GD**
      - Với mỗi trạng thái $\mathbf{Y_i}$ được sinh ra bởi trạng thái **X** thông qua các toán tử
        - Đưa trạng thái $\mathbf{Y_i}$ vào danh sách **temp** rồi sắp xếp các trạng thái trong danh sách **temp** theo hàm đánh giá từ tốt nhất đến xấu nhất
        - Đưa toàn bộ danh sách **temp** vào đầu danh sách **open**
        - **father(**$\mathbf{Y_i}$**)** $\leftarrow$ **X**
  - **end loop**

Thuật toán tìm kiếm leo đồi

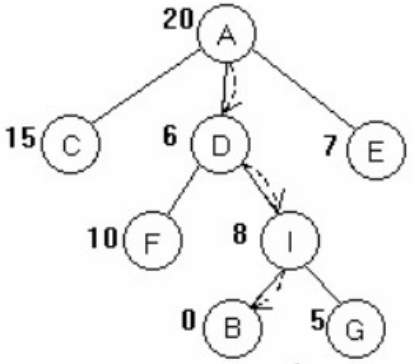
- Ví dụ: Sử dụng thuật toán Hill-Climbing Search để duyệt đồ thị bên dưới. Biết rằng
  - A là trạng thái bắt đầu
  - B là trạng thái kết thúc
  - Đầu mũi tên chỉ trạng thái được sinh bởi trạng thái ở đuôi mũi tên.
  - Con số ghi phía trên trạng thái là giá trị của hàm đánh giá.
  - Giả sử trong ví dụ này ta coi hàm đánh giá có giá trị càng nhỏ càng tốt.



Thuật toán tìm kiếm leo đồi

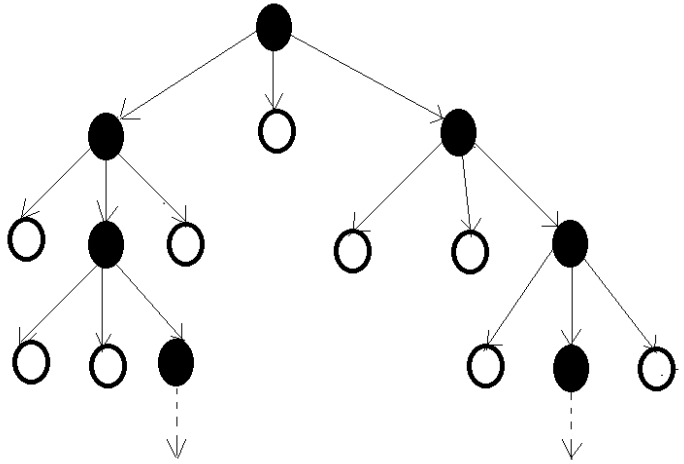


- open=[A^{20}];
- X=A^{20}; temp=[D^{6},E^{7},C^{15}]; open=[D^{6},E^{7},C^{15}]; father[D^{6},E^{7},C^{15}]=A^{20};
- X=D^{6}; temp=[I^{8},F^{10}]; open=[I^{8},F^{10},E^{7},C^{15}]; father[I^{8},F^{10}]=D^{6};
- X=I^{8}; temp=[B^{0},G^{5}]; open=[B^{0},G^{5},F^{10},E^{7},C^{15}]; father[B^{0},G^{5}]=I^{8};
- X=B^{0}; => Tìm kiếm thành công



Chiến lược tìm kiếm beam

- **Chiến lược tìm kiếm beam** (Beam search) là chiến lược **tìm kiếm theo bề rộng** và chỉ **phát triển k đỉnh tốt nhất** ở mỗi mức (các đỉnh tốt nhất này được xác định bởi **hàm đánh giá**).
  - Như vậy, ở bất kỳ mức nào trong tìm kiếm beam cũng chỉ có k đỉnh được phát triển.



Thuật toán tìm kiếm beam

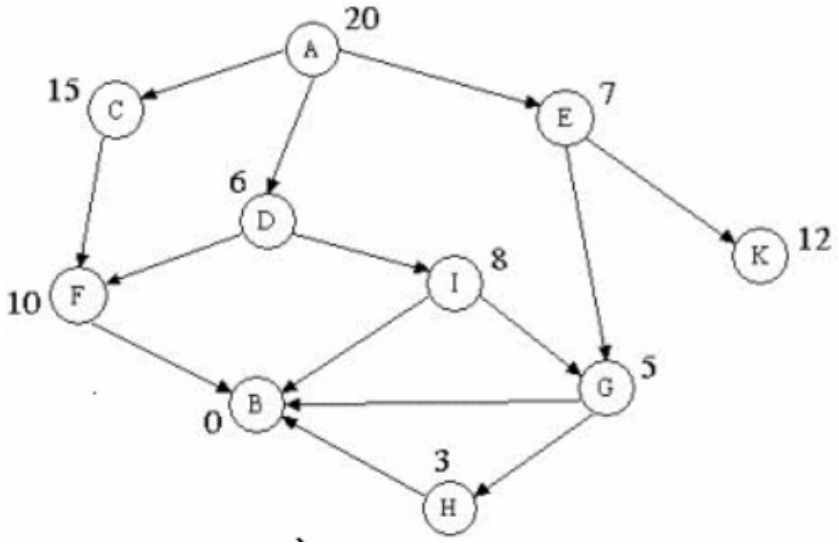
- Ta sử dụng danh sách **open** để lưu giữ các trạng thái được sinh ra nhưng chưa được khảo sát.
- Hàm **father(X)** dùng để lưu lại cha của đỉnh X trên đường đi.
  - Ví dụ: Cú pháp **father(X)** ← **Y** tức là đỉnh cha của đỉnh X là đỉnh Y
- Xác định trạng thái bắt đầu **S**
- Xác định tập các trạng thái kết thúc **GD**
- Xác định các toán tử chuyển trạng thái

Thuật toán tìm kiếm beam

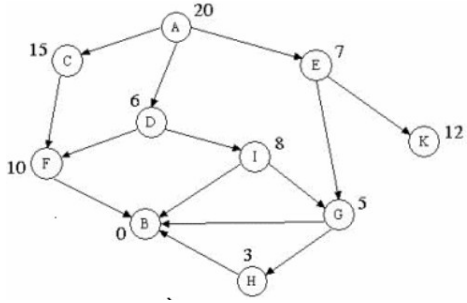
- **BEAM-SEARCH(k)**
  - Đưa trạng thái bắt đầu **S** vào **open**;
  - **loop do**
    - Nếu **open =** $\varnothing$ thì thông báo Không tìm thấy kết quả. **Kết thúc thuật toán**.
    - Lấy **min[k, size(open)]** trạng thái ở đầu danh sách open rồi đưa vào các biến trạng thái $\mathbf{X_k}$, sau đó loại bỏ tất cả các trạng thái bên trong danh sách open để danh sách trở thành rỗng.
    - Với mỗi trạng thái X_k
      - Nếu $\mathbf{X_k} \in$ **GD** thì thông báo tìm kiếm thành công. **Kết thúc thuật toán**.
      - Nếu $\mathbf{X_k} \notin$ **GD**
        - Với tất cả các trạng thái $\mathbf{Y_i}$ được sinh ra bởi các trạng thái $\mathbf{X_k}$ thông qua các toán tử
          - Đưa các trạng thái $\mathbf{Y_i}$ vào danh sách **open** rồi sắp xếp các trạng thái trong danh sách **open** theo hàm đánh giá từ tốt nhất đến xấu nhất
          - Xác định **father(**$\mathbf{Y_i}$**)** $\leftarrow$ $\mathbf{X_k}$ tương ứng
  - **end loop**

Thuật toán tìm kiếm beam

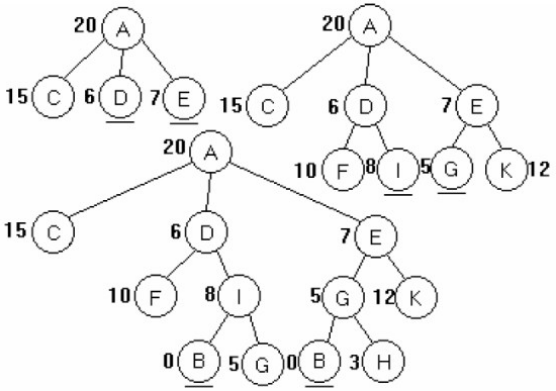
- Ví dụ: Sử dụng thuật toán Beam Search để duyệt đồ thị bên dưới. Biết rằng
  - A là trạng thái bắt đầu
  - B là trạng thái kết thúc
  - Đầu mũi tên chỉ trạng thái được sinh bởi trạng thái ở đuôi mũi tên.
  - Con số ghi phía trên trạng thái là giá trị của hàm đánh giá.
  - Giả sử trong ví dụ này ta coi hàm đánh giá có giá trị càng nhỏ càng tốt.
  - Chỉ phát triển 2 đỉnh ở mỗi mức (k = 2)



Thuật toán tìm kiếm beam



- open=[A^{20}];
- X_k=[A^{20}]; open=[D^6,E^7,C^{15}]; father[D^6,E^7,C^{15}]=A^{20};
- X_k=[$D^{6,}E^7$]; open=[G^5,I^8,F^{10},K^{12}]; father[G^5,K^{12}]=E^7; father[I^8,F^{10}]=D^6;
- X_k=[G^5,I^8]; open=[$B^0{}_I$,$B^0{}_G$,H^3,G^5]; father[$B^0{}_I$,G^5]=I^8; father[$B^0{}_G$,H^3]=G^5;
- X_1=$B^0{}_I$; => Tìm kiếm thành công



Hết Tuần 4

Cảm ơn các bạn đã chú ý lắng nghe !!!