Trí tuệ nhân tạo

Tuần 2

Giảng viên: Trần Đức Minh

Nội dung trình bày

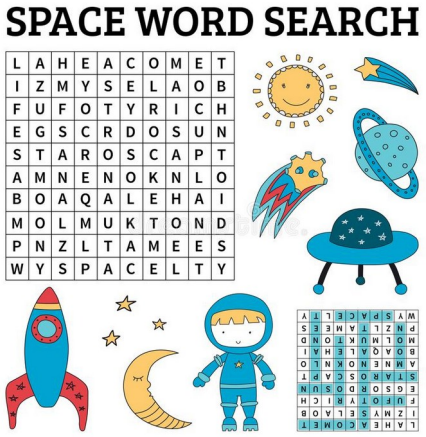
- Đặt vấn đề
- Không gian trạng thái
- Chiến lược tìm kiếm theo bề rộng
- Chiến lược tìm kiếm theo chiều sâu

Đặt vấn đề

- Cái chúng ta muốn
  - Xây dựng được một hệ thống **tự giải quyết**, **tự tìm lời giải, tự tìm kết quả** của một vấn đề nào đó.
- Vậy cái chúng ta cần là
  - Một mô tả (mô hình hóa) vấn đề cần giải quyết.
  - Các thuật toán, chiến lược tìm kiếm được sử dụng để giải quyết vấn đề.

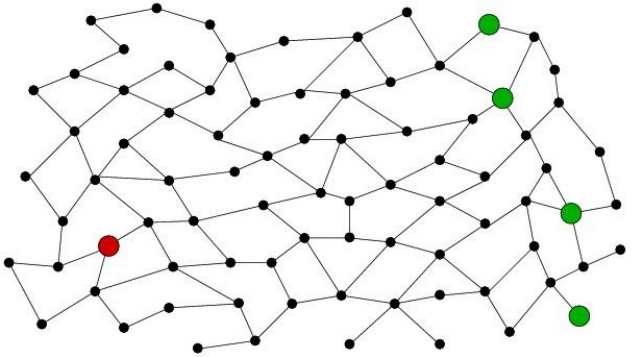
Đặt vấn đề

- Để giải quyết một vấn đề nào đó bằng tìm kiếm, việc đầu tiên ta cần phải làm là xác định **không gian tìm kiếm**.
  - Tức đó là nơi chúng ta cần tìm kiếm lời giải ?
- **Không gian tìm kiếm** bao gồm tất cả các đối tượng mà ta cần quan tâm tìm kiếm.



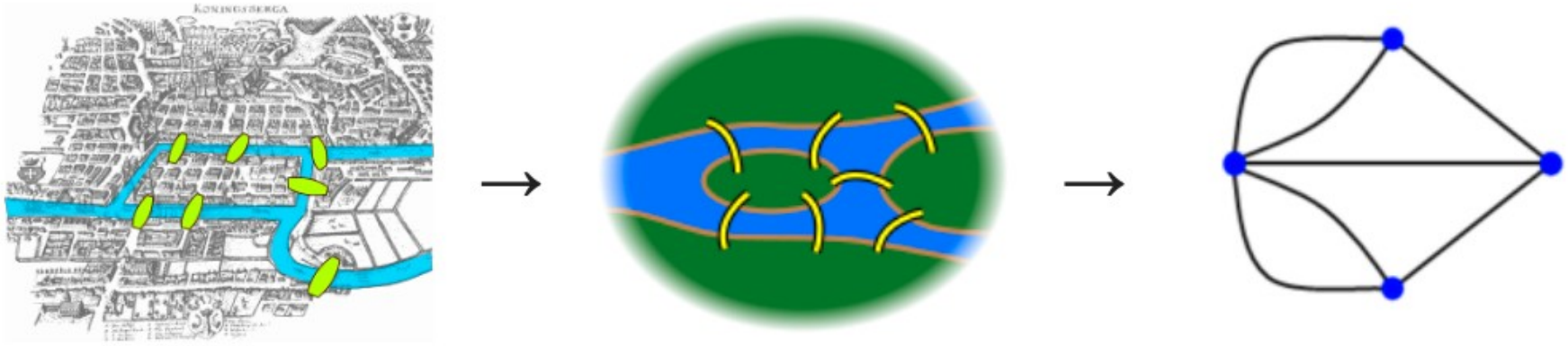
Đặt vấn đề

- Khi ta coi mỗi đối tượng trên không gian tìm kiếm là một **trạng thái** của bài toán và các trạng thái này **có thể có mối liên kết với nhau** (thao tác chuyển trạng thái) thì ta nhận được một **không gian trạng thái** (state space).
- Vậy không gian trạng thái cũng chứa các đối tượng mà ta cần quan tâm tìm kiếm.



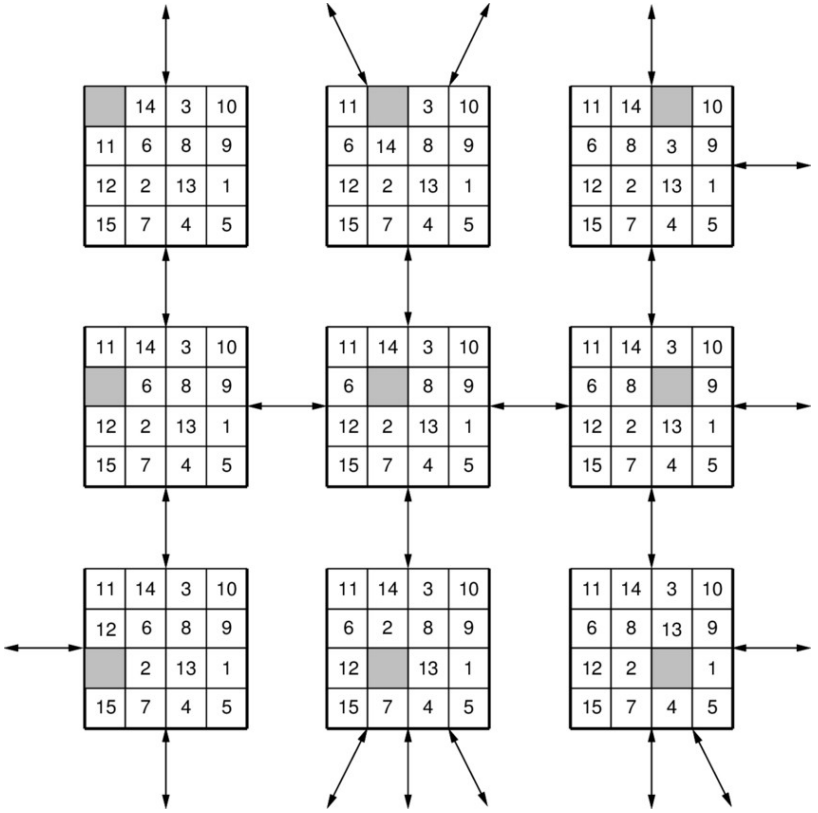
Không gian trạng thái

- Không gian trạng thái của bài toán thường được **biểu diễn như một đồ thị**.
- Ví dụ: Bài toán 7 cây cầu Königsberg



Không gian trạng thái

- Ví dụ: Một phần không gian trạng thái của bài toán Ta canh với 15 ô chữ số

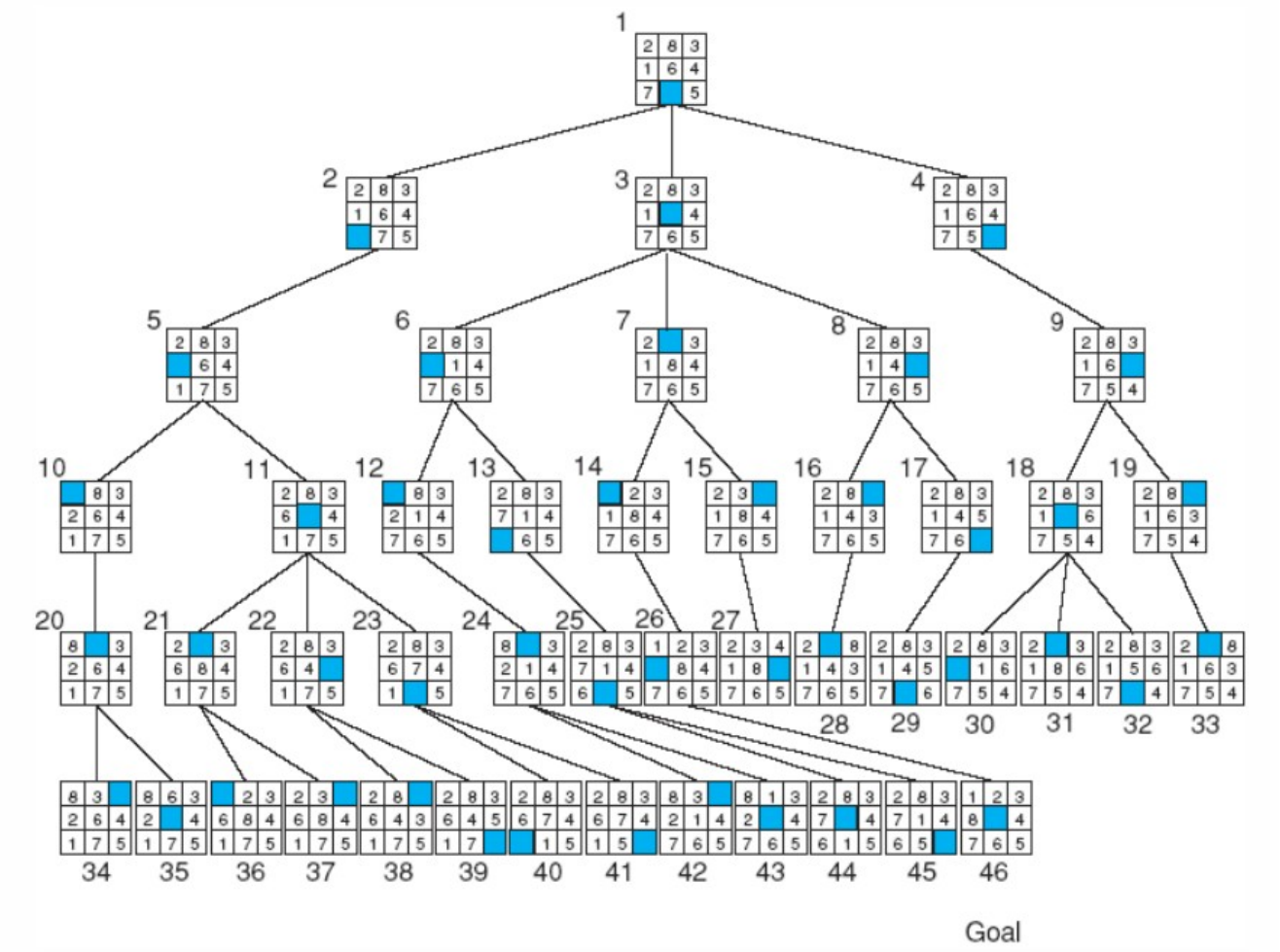


Không gian trạng thái

- Một không gian trạng thái được biểu diễn bằng một nhóm gồm bốn yếu tố [N, A, S, GD], trong đó:
  - N (node) là tập hợp các nút hay các trạng thái của đồ thị.
  - A (arc) là tập các cung (hay các liên kết) giữa các nút.
  - S (Start) chứa (các) trạng thái ban đầu của bài toán ($S \subset N$ và $S \neq \varnothing$).
  - GD (Goal Description) chứa (các) trạng thái đích của bài toán ($GD \subset N$ và $GD \neq \varnothing$). Các trạng thái trong GD được mô tả theo một trong hai đặc tính:
    - Đặc tính có thể đo lường được các trạng thái gặp trong quá trình tìm kiếm. Ví dụ: Trạng thái cuối của trò chơi Ta Canh
    - Đặc tính của đường đi được hình thành trong quá trình tìm kiếm. Ví dụ: Tìm được đường đi ngắn nhất trong một dãy nút của đồ thị.
  - Đường đi của lời giải (solution path) là đường đi qua đồ thị này từ một nút trong S đến một nút trong GD.

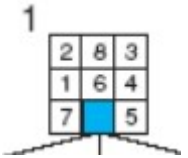
Không gian trạng thái

- Ví dụ 1: Bài toán Ta canh với 8 ô chữ số

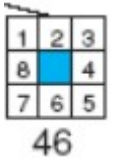


Không gian trạng thái

- Ví dụ 1: Bài toán Ta canh với 8 ô chữ số
  - **N** là tập hợp tất cả các vị thế (trạng thái) của bài toán.
  - **A** là tập hợp các bước chuyển từ vị thế nọ sang vị thế kia của bài toán.
  - **S** chứa trạng thái ban đầu



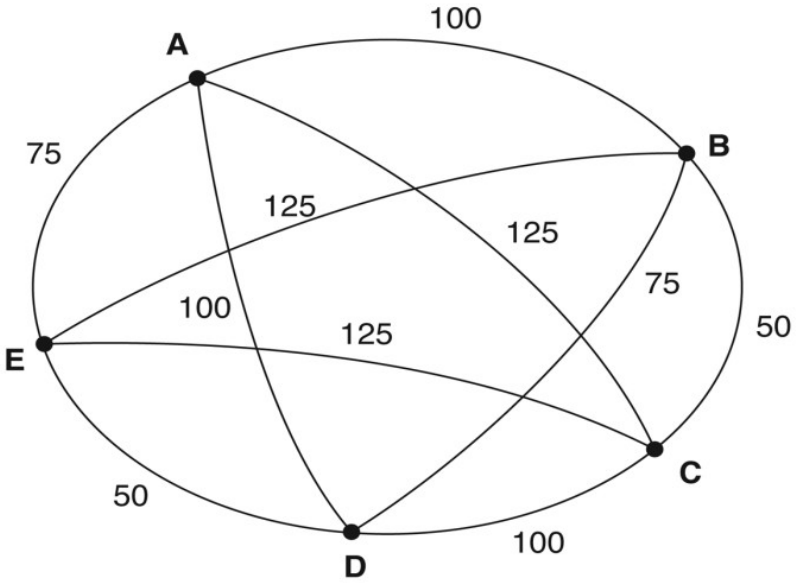
  - **GD** chứa trạng thái đích



  - **Đường đi của lời giải**: 1 - 3 - 7 - 14 - 26 - 46

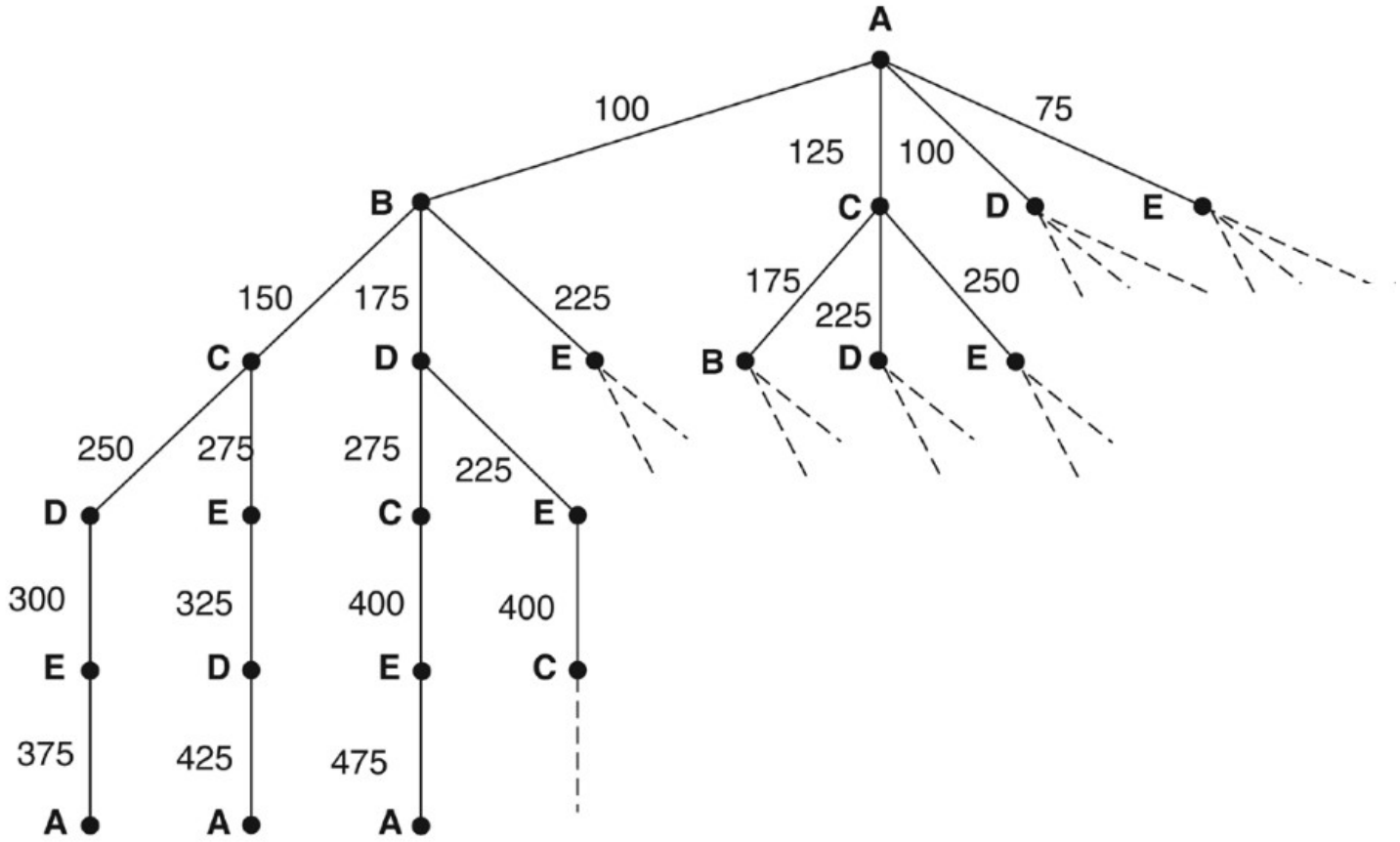
Không gian trạng thái

- Ví dụ 2: Bài toán người đưa thư
  - Một người đưa thư xuất phát từ bưu điện, ông ta phải đến một số điểm đến để phát thư rồi quay trở về điểm xuất phát, hỏi người đó phải đi như thế nào để khoảng cách đường đi là ngắn nhất.



Không gian trạng thái

- Ví dụ 2: Bài toán người đưa thư



Không gian trạng thái

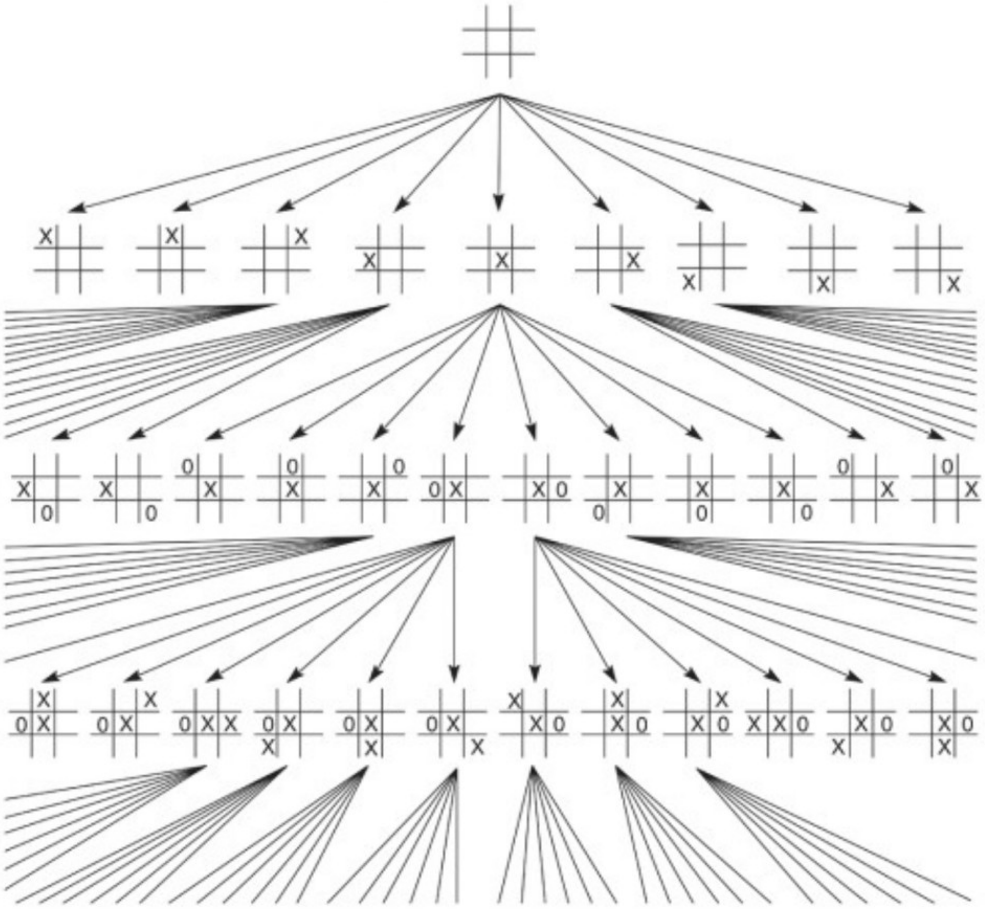
- Ví dụ 2: Bài toán người đưa thư
  - **N** là tập hợp tất cả các điểm đến của bài toán.
  - **Arc** là tập hợp các bước chuyển từ điểm đến nọ sang điểm đến kia.
    - Mỗi bước (cung) có một trọng số là khoảng cách giữa 2 điểm.
  - **S** chứa điểm xuất phát là điểm A.
  - **GD** chứa điểm quay về cũng là điểm A.

Không gian trạng thái

- Ví dụ 2: Bài toán người đưa thư
  - Mỗi đường đi xuất phát từ điểm A rồi quay trở về điểm A đều có một giá trị **cost** là tổng khoảng cách người đưa thư phải đi.
    - cost(ABCDEA) = 375
    - cost(ABCEDA) = 425
    - cost(ABDCEA) = 475
  - **Đường đi của lời giải** là đường đi có giá trị cost nhỏ nhất.

Không gian trạng thái

- Ví dụ 3: Bài toán trò chơi Tic-tac-toe



Không gian trạng thái

- Ví dụ 3: Bài toán trò chơi Tic-tac-toe
  - **N** là tập hợp tất cả các vị thế của trò chơi.
  - **A** là tập hợp các bước chuyển từ vị thế nọ sang vị thế kia của bài toán.
  - **S** chứa trạng thái ban đầu 
  - **GD** chứa các trạng thái đích là các trạng thái xác định một quân cờ nào đó chiến thắng.
  - **Đường đi**: Lựa chọn vị thế tốt nhất tiếp theo đối với bên đang chuẩn bị đi nước cờ kế tiếp.

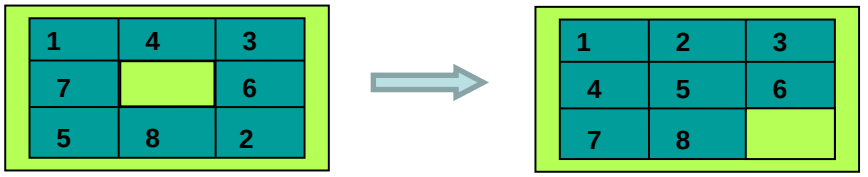
Không gian trạng thái

- Một số lưu ý về không gian trạng thái
  - **Vấn đề cần giải quyết** được chia thành một tập hợp các bước chuyển từ các trạng thái bắt đầu đến các trạng thái kết thúc.
  - Không gian trạng thái được sử dụng để **giải quyết vấn đề** thông qua việc **tìm kiếm trên không gian trạng thái**.

Biểu diễn vấn đề trong không gian trạng thái

- Muốn biểu diễn một vấn đề trong không gian trạng thái, ta cần xác định các yếu tố sau:
  - Trạng thái ban đầu.
  - Trạng thái kết thúc.
  - Một tập hợp các toán tử. Trong đó mỗi toán tử mô tả một hành động hoặc một phép biến đổi có thể đưa một trạng thái tới một trạng thái khác.
  - Chi phí cho mỗi bước chuyển trạng thái.
- **Tập hợp tất cả các trạng thái có thể đạt tới từ trạng thái ban đầu** bằng cách áp dụng một dãy toán tử, **lập thành không gian trạng thái của vấn đề**.

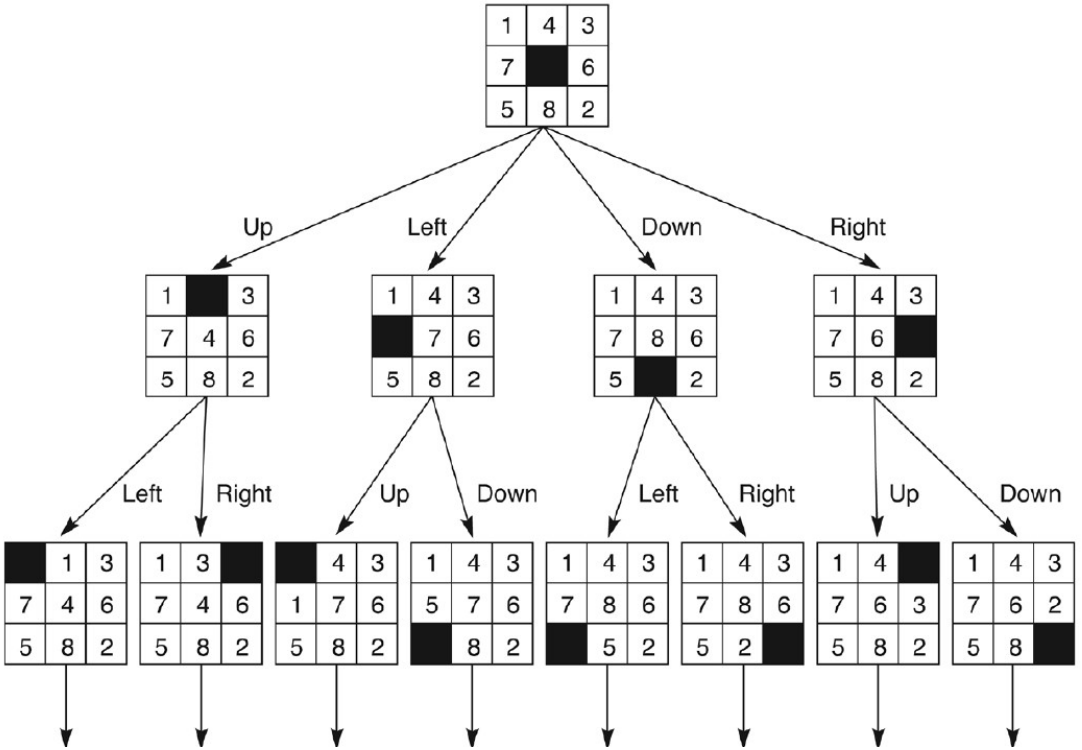
Biểu diễn vấn đề trong không gian trạng thái



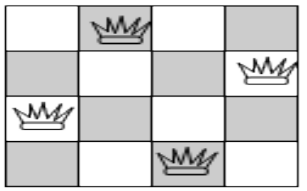
- Ví dụ 1: Trò chơi Ta Canh
  - Trạng thái bắt đầu là trạng thái ở bên trái
  - Trạng thái kết thúc là trạng thái ở bên phải
  - Tương ứng với các quy tắc dịch chuyển quân ta có:
    - Di chuyển vào ô trống lên phía trên (Up)
    - Di chuyển vào ô trống xuống dưới (Down)
    - Di chuyển vào ô trống sang bên phải (Right)
    - Di chuyển vào ô trống sang bên trái (Left)
  - Chi phí cho mỗi bước chuyển là 0.

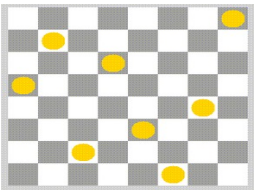
Biểu diễn vấn đề trong không gian trạng thái

- Ví dụ 1: Trò chơi Ta Canh
  - Một phần không gian trạng thái của trò chơi



Biểu diễn vấn đề trong không gian trạng thái





- Ví dụ 2: Bài toán n con hậu
  - Trạng thái bắt đầu là trạng thái không có con hậu nào trên bàn cờ
  - Trạng thái kết thúc là trạng thái có đầy đủ các con hậu trên bàn cờ và không con nào ăn được con nào
  - Các hành động chuyển trạng thái:
    - Đặt con hậu trên bàn cờ sao cho con hậu mới đặt lên không bị ăn bởi các con hậu đã có trên bàn cờ.
  - Chi phí cho mỗi bước chuyển là 0.

Chiến lược tìm kiếm trên đồ thị không gian trạng thái

- Để giải quyết vấn đề bằng tìm kiếm trên không gian trạng thái, ta cần xác định trước các vấn đề sau:
  - Trạng thái ban đầu
  - Các toán tử dùng để chuyển trạng thái
  - Các trạng thái kết thúc
- Từ một trạng thái bất kỳ ta sử dụng các toán tử để sinh ra các trạng thái kế tiếp.
- Các trạng thái kế tiếp sẽ được đánh giá xem có nằm trong tập trạng thái kết thúc hay không.
- Vậy, thông thường khi ta muốn giải quyết vấn đề thông qua không gian trạng thái thì việc tìm lời giải của vấn đề được quy về việc
  - Tìm đường đi từ trạng thái ban đầu tới một trạng thái kết thúc.
  - Hoặc tìm toán tử chuyển trạng thái tốt nhất tiếp theo.

Phương pháp đánh giá chiến lược tìm kiếm trên đồ thị không gian trạng thái

- Các tiêu chuẩn đánh giá chiến lược
  - **Tính đầy đủ**: Lời giải của bài toán có tìm được không nếu bài toán tồn tại lời giải?
  - **Độ phức tạp về thời gian**
  - **Độ phức tạp về không gian lưu trữ**: Kích cỡ của bộ nhớ cần cho giải thuật ? (phụ thuộc vào Cấu trúc dữ liệu lưu trữ)
  - **Tính tối ưu**: Có tìm ra lời giải có chi phí tối ưu ?
- Độ phức tạp về thời gian và không gian lưu trữ có thể được đo bằng
  - **b**: Độ phân nhánh của cây
  - **d**: Độ sâu của lời giải ngắn nhất
  - **m**: Độ sâu tối đa của không gian trạng thái (có thể vô hạn).

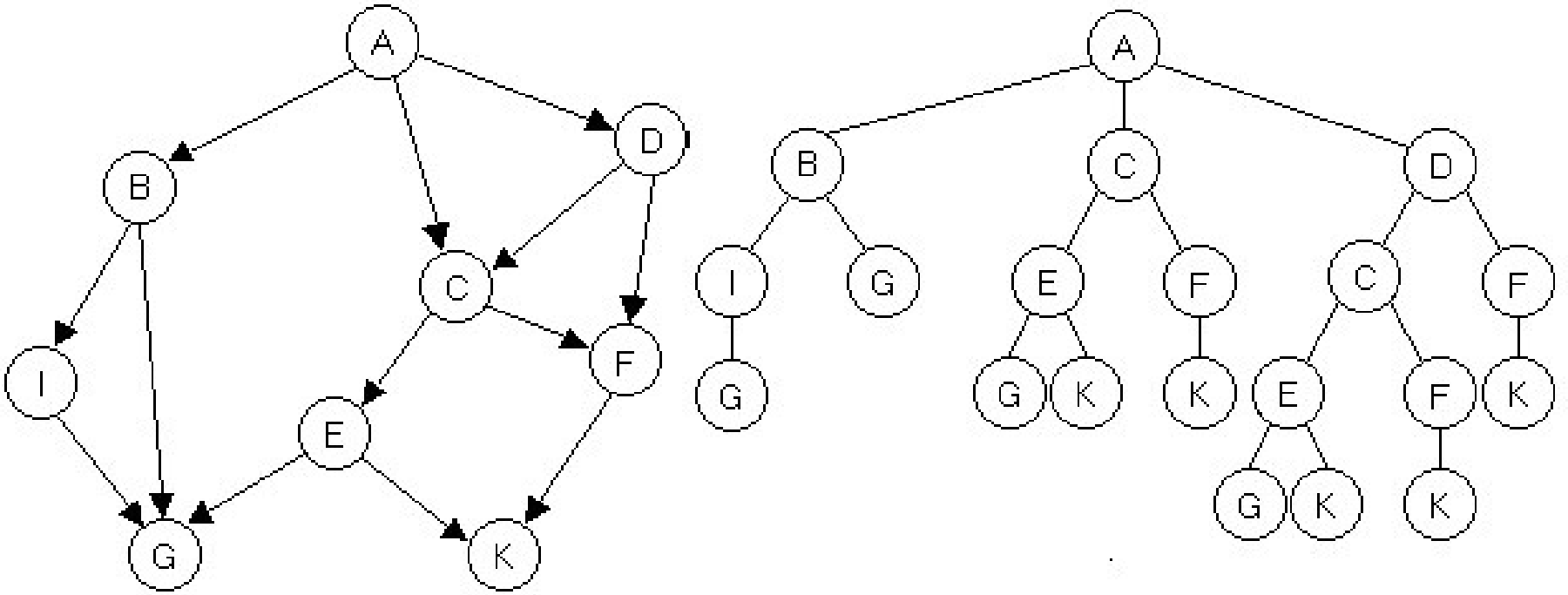
Phân loại chiến lược tìm kiếm

- **Chiến lược tìm kiếm mù**
  - Không có sự hướng dẫn nào cho sự tìm kiếm.
  - Chỉ phát triển các trạng thái từ trạng thái ban đầu cho đến khi gặp trạng thái đích.
- **Chiến lược tìm kiếm kinh nghiệm (heuristic)**
  - Dựa vào kinh nghiệm, trực giác để đánh giá trạng thái nhằm hướng dẫn cho công việc tìm kiếm.
  - Trong quá trình phát triển trạng thái, ta sẽ chọn trạng thái được đánh giá là tốt nhất để phát triển.

Cây tìm kiếm

- Cây tìm kiếm là cây mà các đỉnh của nó thể hiện lời giải tiềm năng của bài toán.
  - Gốc của cây tìm kiếm tương ứng với trạng thái ban đầu.
  - Các mức tiếp theo của cây là các đỉnh kề với các đỉnh ở mức trước.
  - Thông thường cây tìm kiếm được mở rộng đến trạng thái đích thì dừng.
- Vậy quá trình tìm kiếm tương tự như quá trình xây dựng cây tìm kiếm.
- Mỗi chiến lược tìm kiếm trong không gian trạng thái tương ứng với một phương pháp xây dựng cây tìm kiếm.

Cây tìm kiếm

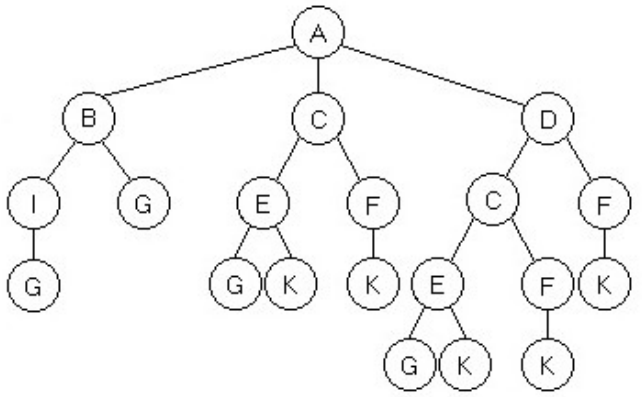


a) Đồ thị không gian trạng thái

b) Cây tìm kiếm tương ứng

Trạng thái lặp

- Cây tìm kiếm có thể chứa nhiều trạng thái lặp lại, điều này làm cho thuật toán tìm kiếm sẽ phí tổn thời gian để phát triển lại các trạng thái mà ta đã gặp và phát triển.
- Việc không xử lý tốt các trạng thái lặp lại có thể dẫn đến bùng nổ tổ hợp đối với độ phức tạp thời gian và không gian.

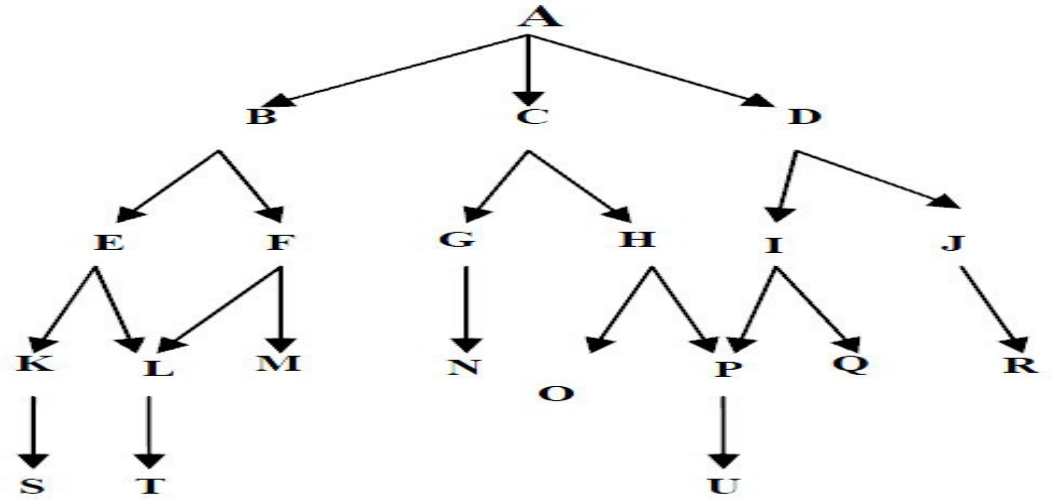


Trạng thái lặp

- Ta áp dụng một trong các phương pháp sau để tránh trạng thái lặp:
  - Khi phát triển đỉnh X, không sinh ra các đỉnh trùng với cha của X.
  - Khi phát triển đỉnh X, không sinh ra các đỉnh trùng với một đỉnh nào đó nằm trên đường đi dẫn tới X.
  - Không sinh ra các đỉnh mà nó đã được sinh ra, tức là chỉ sinh ra các đỉnh mới.

Chiến lược tìm kiếm theo bề rộng

- **Chiến lược tìm kiếm theo bề rộng** (Breadth-First Search) sẽ khảo sát không gian theo từng mức. Chỉ đến khi trong một mức cho trước không còn một trạng thái nào để khảo sát thì thuật toán mới chuyển sang mức tiếp theo.



- Ví dụ:
  - Các trạng thái được xem xét như sau: A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U

Thuật toán BFS

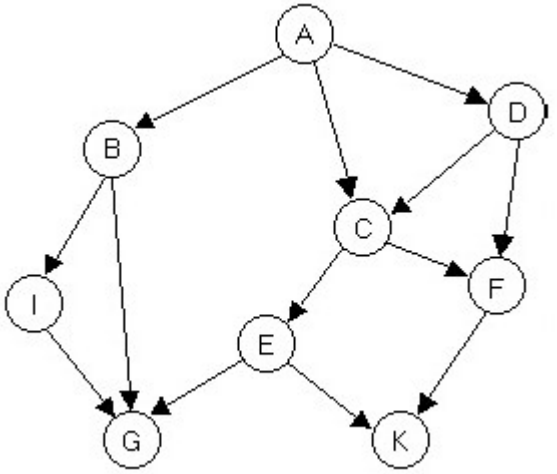
- Ta sử dụng hàng đợi **open** (có cấu trúc FIFO) để lưu giữ các trạng thái được sinh ra nhưng chưa được khảo sát.
- Hàm **father(X)** dùng để lưu lại cha của đỉnh X trên đường đi.
  - Ví dụ: Cú pháp **father(X)** ← **Y** tức là đỉnh cha của đỉnh X là đỉnh Y
- Xác định trạng thái bắt đầu **S**
- Xác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

Thuật toán BFS

- **BREADTH-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 từ hàng đợi open đưa vào biến trạng thái **X**, sau đó loại bỏ trạng thái đó khỏi hàng đợi.
    - 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 hàng đợi **open**
        - **father(**$\mathbf{Y_i}$**)** $\leftarrow$ **X**
  - **end loop**

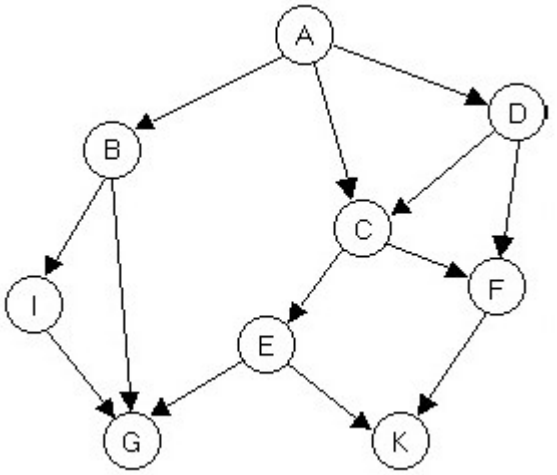
Thuật toán BFS

- Ví dụ 1: Sử dụng thuật toán BFS để duyệt đồ thị bên dưới. Biết rằng
  - A là trạng thái bắt đầu
  - D là trạng thái kết thúc
  - Đầu mũi tên chỉ trạng thái được sinh bởi trạng thái ở đuôi mũi tên.



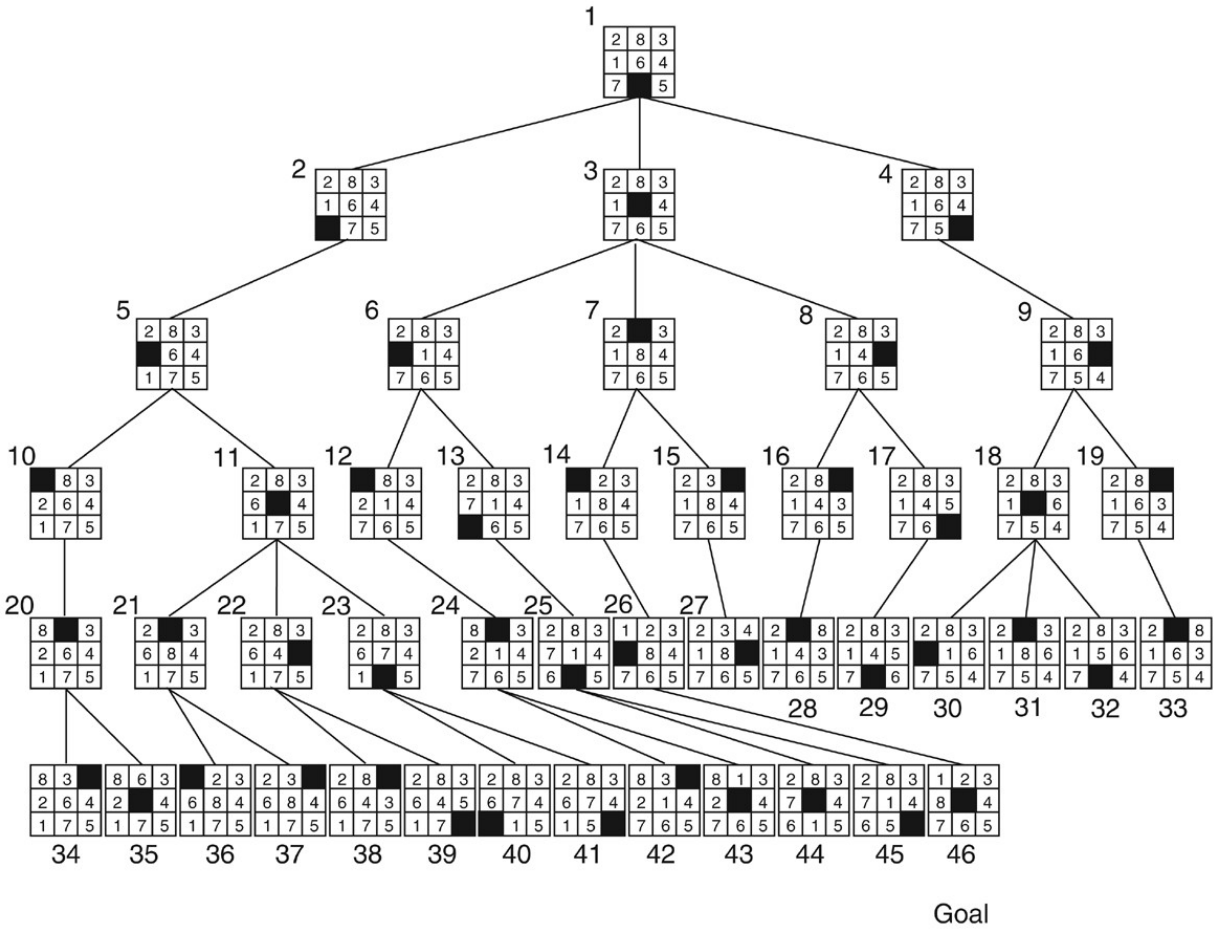
Thuật toán BFS

- open=[A];
- X=A; open=[B,C,D]; father[B,C,D] = A;
- X=B; open=[C,D,I,G]; father[I,G] = B;
- X=C; open=[D,I,G,E,F]; father[E,F] = C;
- X = D; => Tìm kiếm thành công



Thuật toán BFS

- Ví dụ 2: Bài toán Ta canh với thuật toán BFS

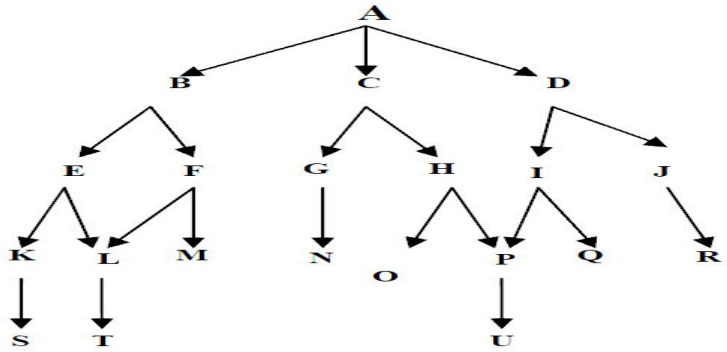


Thuật toán BFS

- **Nhận xét**
  - Nếu tồn tại đường đi từ trạng thái đầu tới trạng thái kết thúc thì thuật toán luôn thực hiện thành công.
  - Không gian lưu trữ hết sức tốn kém là vấn đề lớn nhất đối với phương pháp tìm kiếm theo chiều rộng.
- **Đánh giá**
  - Tính đầy đủ: Lời giải bài toán luôn tìm được nếu b (độ phân nhánh của cây) là hữu hạn.
  - Độ phức tạp về thời gian:
    - $1+b+b^2+...+b^d = O(b^d)$
  - Độ phức tạp về không gian lưu trữ: O(bd)
  - Tính tối ưu: Thuật toán BFS luôn tìm ra lời giải với ít trạng thái trung gian nhất (đường đi từ trạng thái bắt đầu đến trạng thái kết thúc).

Chiến lược tìm kiếm theo chiều sâu

- **Chiến lược tìm kiếm theo chiều sâu** (Depth-First Search) là khi một trạng thái được xem xét, tất cả các con của nó được xét đến rồi đến các thế hệ sau của các con đó được xem xét ưu tiên trước bất kỳ một trạng thái anh em nào của nó. Tìm kiếm theo chiều sâu sẽ tiến sâu hơn vào trong không gian tìm kiếm khi nào còn có thể. Chỉ khi nào không tìm được các con cháu xa hơn của trạng thái thì mới xem xét đến các trạng thái anh em của nó.



- Ví dụ:
  - Các trạng thái được xem xét như sau: A, B, E, K, S, L, T, F, M, C, G, N, H, O, P, U, D, I, Q, J, R

Thuật toán DFS

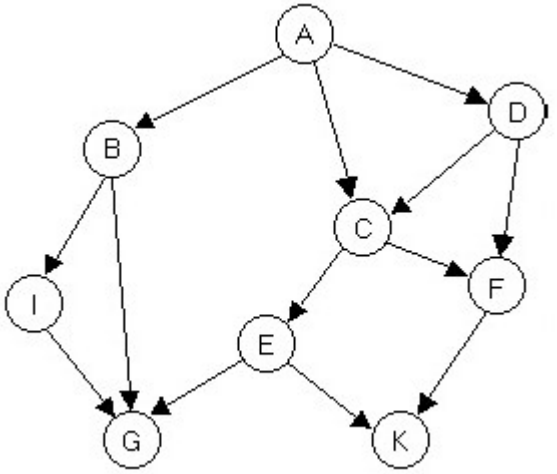
- Ta sử dụng stack **open** (có cấu trúc LIFO) để lưu giữ các trạng thái được sinh ra nhưng chưa được khảo sát.
- Hàm **father(X)** dùng để lưu lại cha của đỉnh X trên đường đi.
  - Ví dụ: Cú pháp **father(X)** ← **Y** tức là đỉnh cha của đỉnh X là đỉnh Y
- Xác định trạng thái bắt đầu **S**
- Xác định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

Thuật toán DFS

- **DEPTH-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 từ stack **open** đưa vào biến trạng thái **X**, sau đó loại bỏ trạng thái đó khỏi stack.
    - 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 stack **open**
        - **father(**$\mathbf{Y_i}$**)** $\leftarrow$ **X**
  - **end loop**

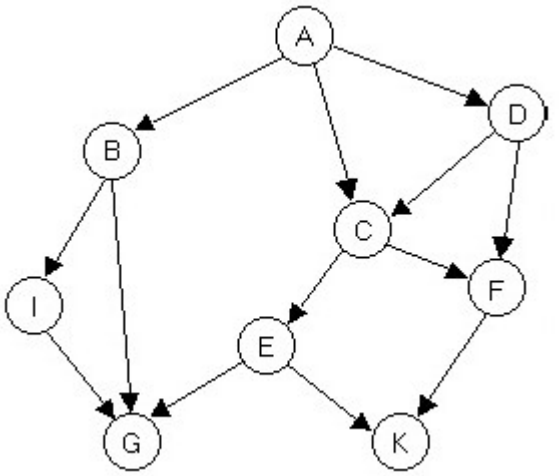
Thuật toán DFS

- Ví dụ 1: Sử dụng thuật toán DFS để duyệt đồ thị bên dưới. Biết rằng
  - A là trạng thái bắt đầu
  - C là trạng thái kết thúc
  - Đầu mũi tên chỉ trạng thái được sinh bởi trạng thái ở đuôi mũi tên.



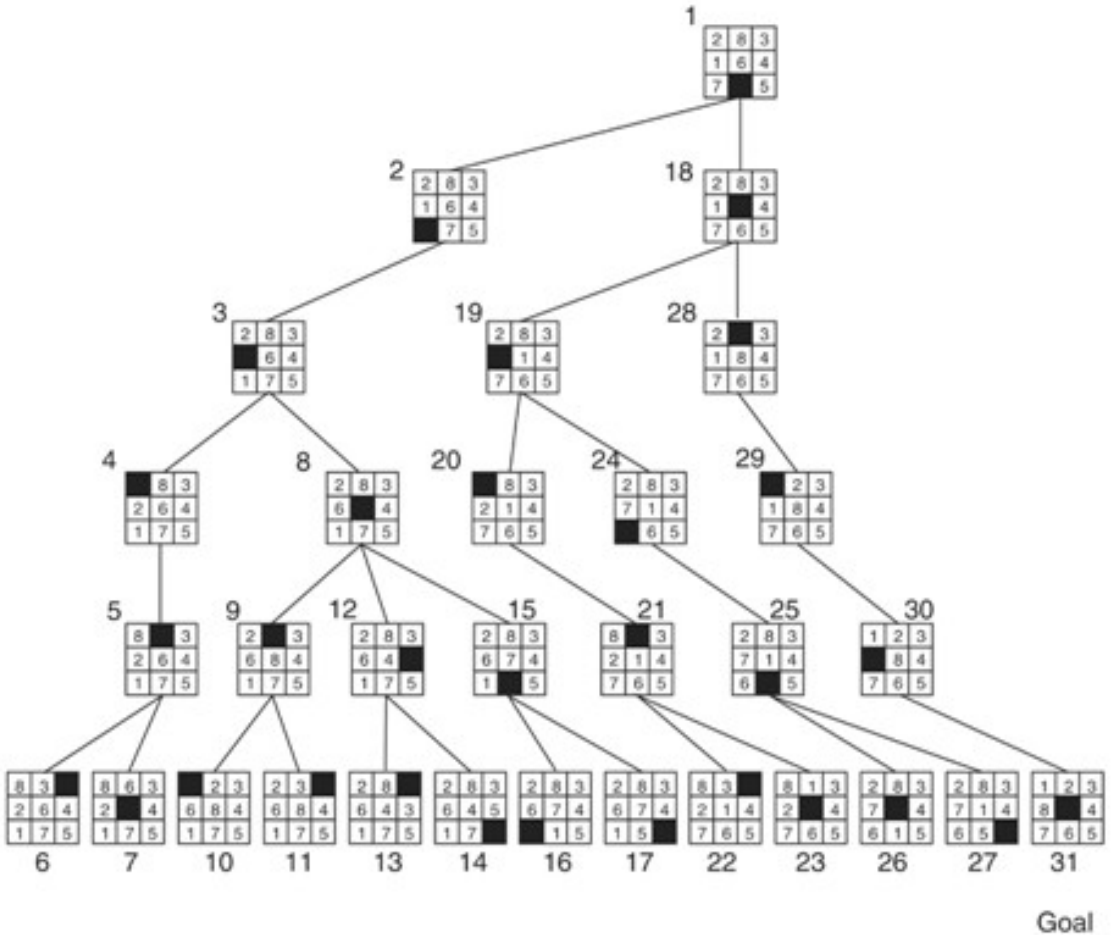
Thuật toán DFS

- open=[A];
- X=A; open=[B,C,D]; father[B,C,D] = A;
- X=B; open=[I,G,C,D]; father[I,G] = B;
- X=I; open=[G^I,G^B,C,D]; father[G^I] = I;
- X=G^I; open=[G^B,C,D];
- X=G^B; open=[C,D];
- X = C; => Tìm kiếm thành công



Thuật toán DFS

- Ví dụ 2: Bài toán Ta canh với thuật toán DFS



Thuật toán DFS

- **Nhận xét**
  - Nếu tồn tại đường đi từ trạng thái bắt đầu tới trạng thái kết thúc thì thuật toán luôn thực hiện thành công đối với không gian tìm kiếm theo chiều sâu là hữu hạn.
    - Lý do, trong trường hợp không gian trạng thái vô hạn thì thuật toán có thể luôn đi sâu xuống mà không đi đúng nhánh chứa trạng thái kết thúc.
- **Đánh giá**
  - Tính đầy đủ: Lời giải bài toán chưa chắc đã tìm được trong trường hợp không gian bài toán là vô hạn.
  - Độ phức tạp về thời gian: $O(b^m)$
  - 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.

Hết Tuần 2

Cảm ơn các bạn đã chú ý lắng nghe !!!