Chương 2: Chiến lược tìm kiếm mù

Giảng viên: Đoàn Thanh Nghị

Khoa CNTT - ĐH An Giang

Nội dung

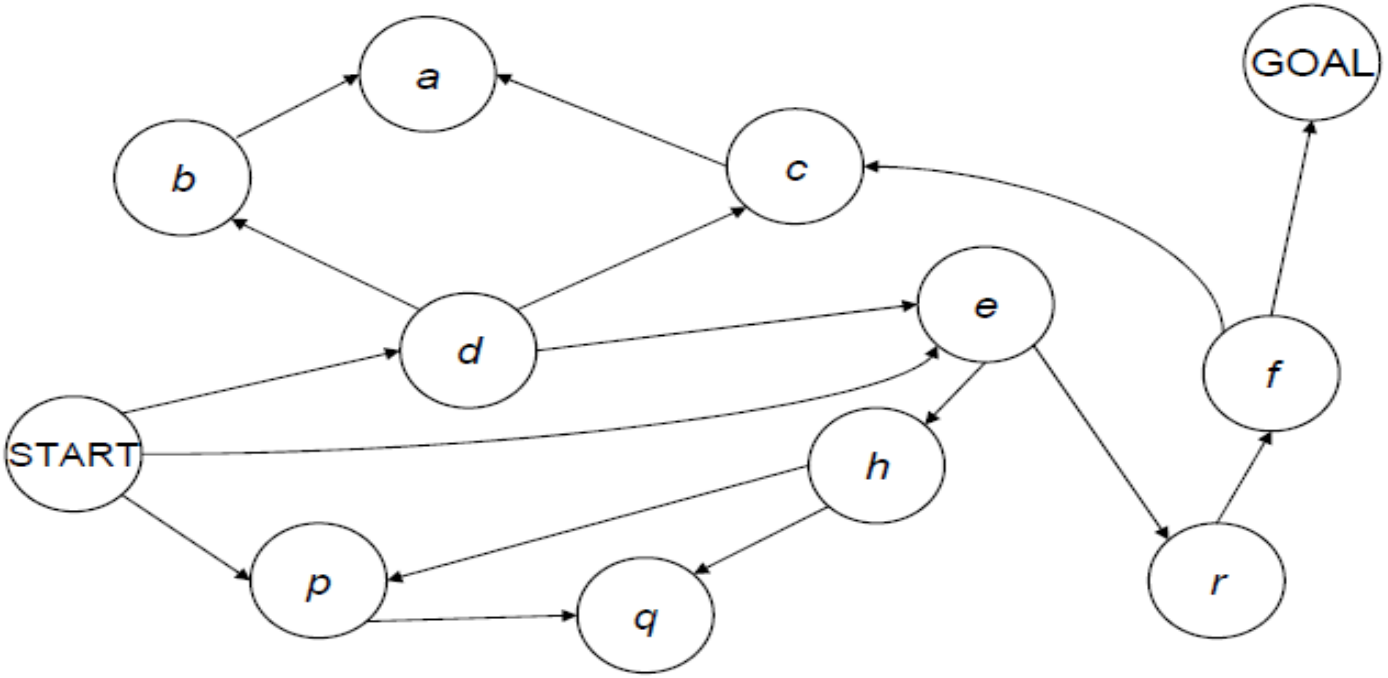
- Bài toán tìm kiếm
  - Biểu diễn bài toán
  - Tìm kiếm
- Các chiến lược điều khiển
- Các đặc trưng của bài toán
- Vấn đề trong thiết kế chương trình tìm kiếm

Ref: http://www.cs.cmu.edu/~awm/tutorials

Mô hình ứng dụng của TTNT

TTNT = Presentation & Search

Bài toán tìm kiếm



Làm sao có thể đi từ S đến G? Và số lần chuyển đổi có thể ít nhất?

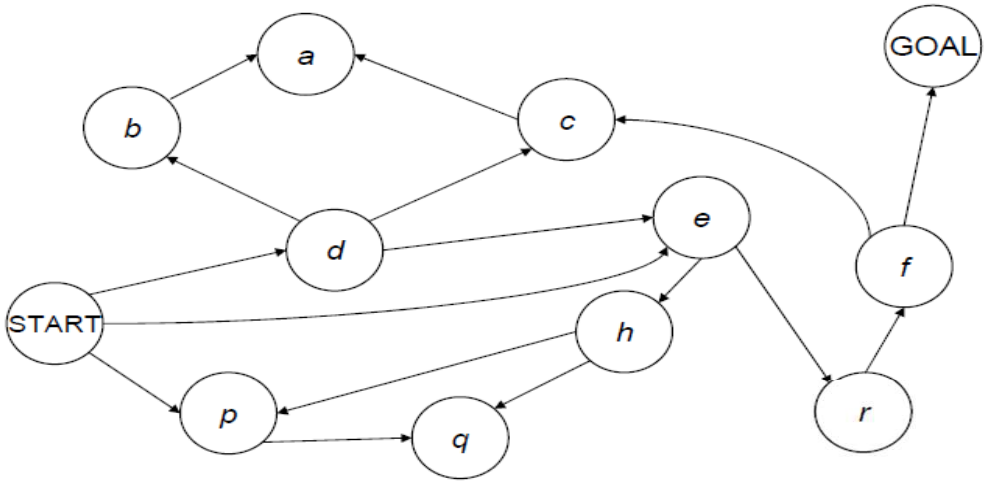
Bài toán tìm kiếm (tt)

- Giải bài toán bằng cách tìm kiếm, gồm:
  - Cấu trúc bài toán: VD. tìm đường đi trên đồ thị
  - Biểu diễn bài toán bằng không gian trạng thái
  - Giải bài toán = Tìm ra một trạng thái/con đường trong không gian trạng thái (trạng thái đầu -> trạng thái đích)

Bài toán tìm kiếm (tt)

- Không gian trạng thái của bài toán tìm kiếm có 5 thành phần: Q, S, G, sucss, cost
  - Q: tập hữu hạn các trạng thái (nút của Graph)
  - S⊆Q: tập hữu hạn khác rỗng các trạng thái bắt đầu
  - G ⊆Q : tập hữu hạn khác rỗng các trạng thái đích
  - **succs**: Q→P(Q) hàm nhận một trạng thái làm đầu vào và trả về kết quả các trạng thái.
  - **cost**: Q, Q→ Số dương là hàm nhận 2 tham số đầu vào là TT s và s'
    - Trả về chi phí từ s đến s'.
    - Hàm chỉ được định nghĩa khi s' là con của s.

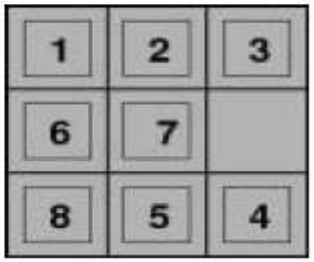
Bài toán tìm kiếm (tt)

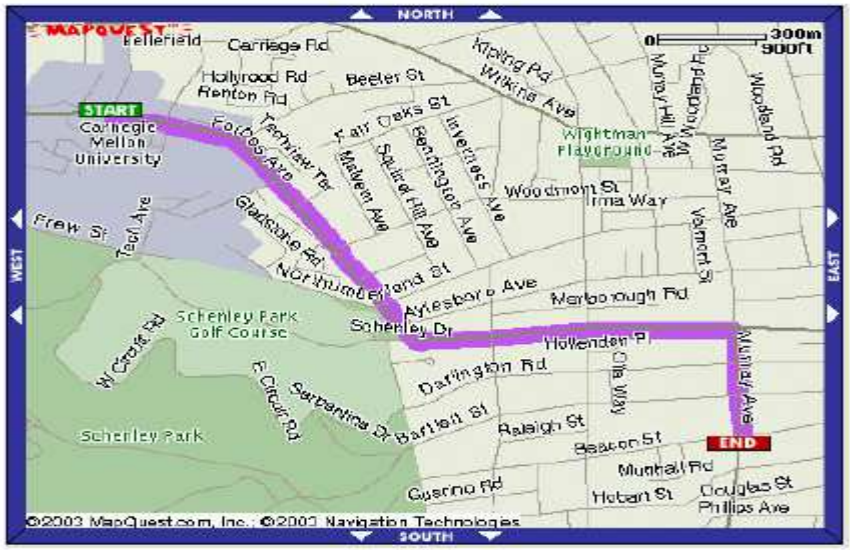


- Q={START, *a* , *b* , *c* , *d* , *e* , *f* , *h* , *p* , *q* , *r*, *GOAL*}
- S={START}
- G={GOAL}
- **succs**(b)={a}, **succs**(e)={h,r},…
- **cost**(s,s')=1, cho tất cả chuyển trạng thái

Tại sao chúng ta quan tâm đến bài toán như vậy?

Các loại bài toán tìm kiếm





Slide 7

Các loại bài toán tìm kiếm (tt)

- Fully observable, deterministic
  - single-belief-state problem
- Non-observable
  - sensorless (conformant) problem
- Partially observable/non-deterministic
  - contingency problem
  - interleave search and execution
- Unknown state space
  - exploration problem
  - execution first

state space vs. database search

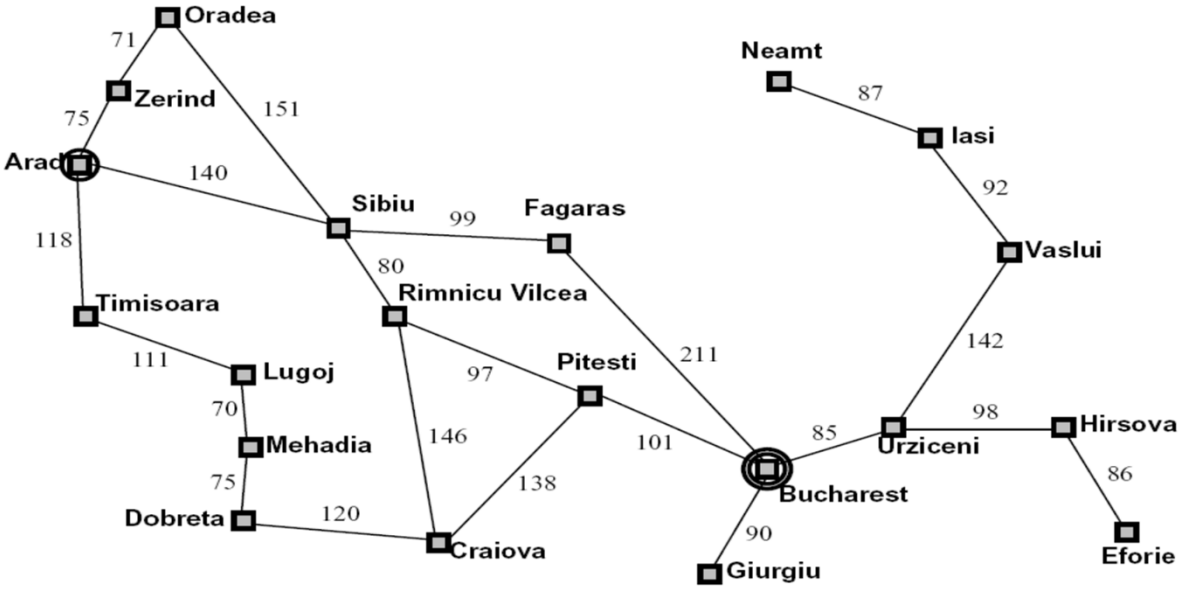
State Space

- Không gian tìm kiếm thường là một đồ thị (graph)
- Mục tiêu tìm kiếm là một path
- Phải lưu trữ toàn bộ không gian trong quá trình tìm kiếm
- Không gian tìm kiếm biến động liên tục trong quá trình tìm kiếm
- Đặc tính của trạng thái / nút chứa nhiều thuộc tính bị thay đổi giá trị trong quá trình tìm kiếm

Database

- Không gian tìm kiếm là một danh sách hay cây
- Tìm kiếm một record/nút
- Phần tử đã duyệt qua là không còn dùng tới
- Không gian tìm kiếm là cố định trong quá trình tìm kiếm
- Thuộc tính của một record/nút là cố định

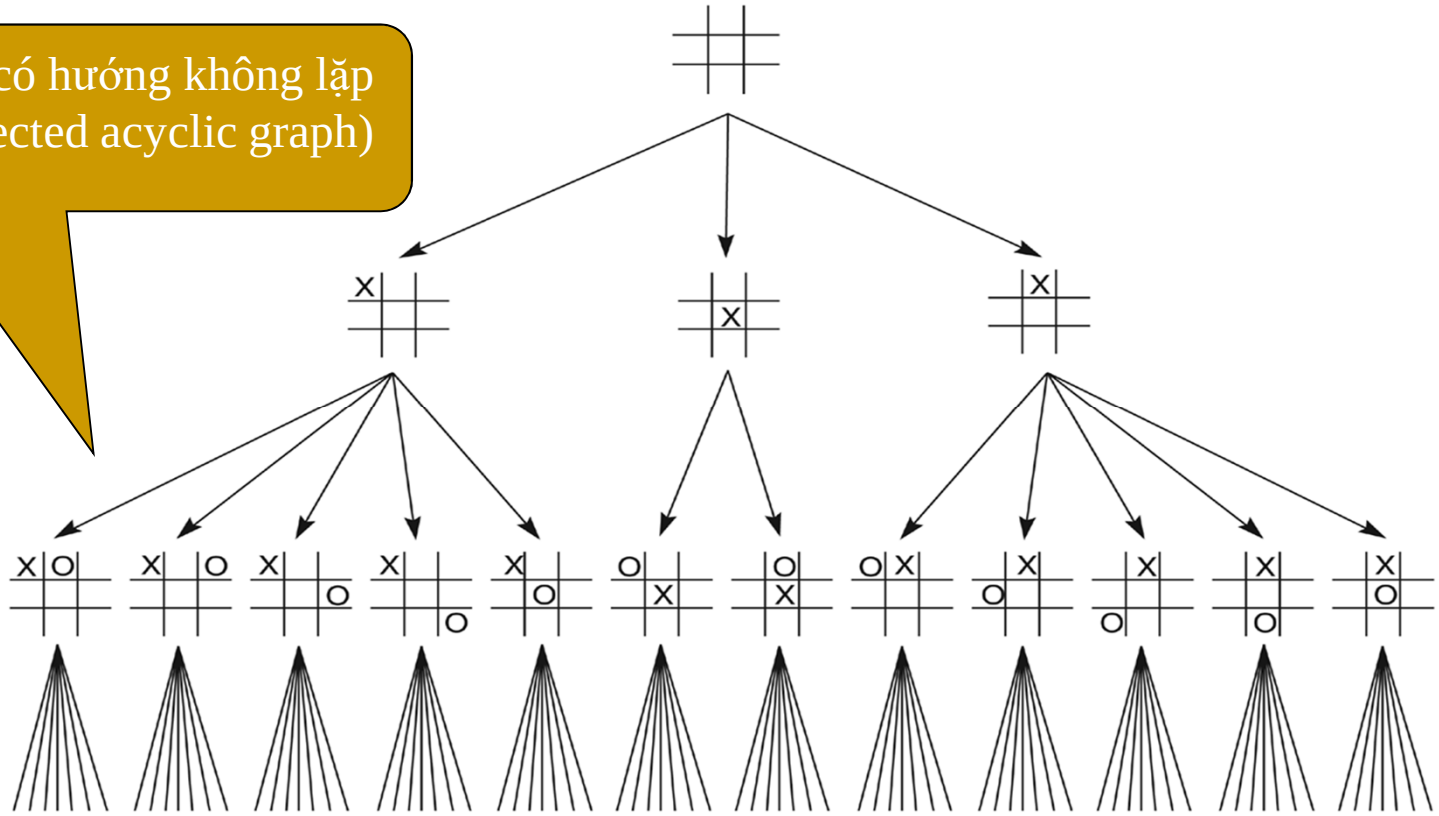
Bài toán Romania



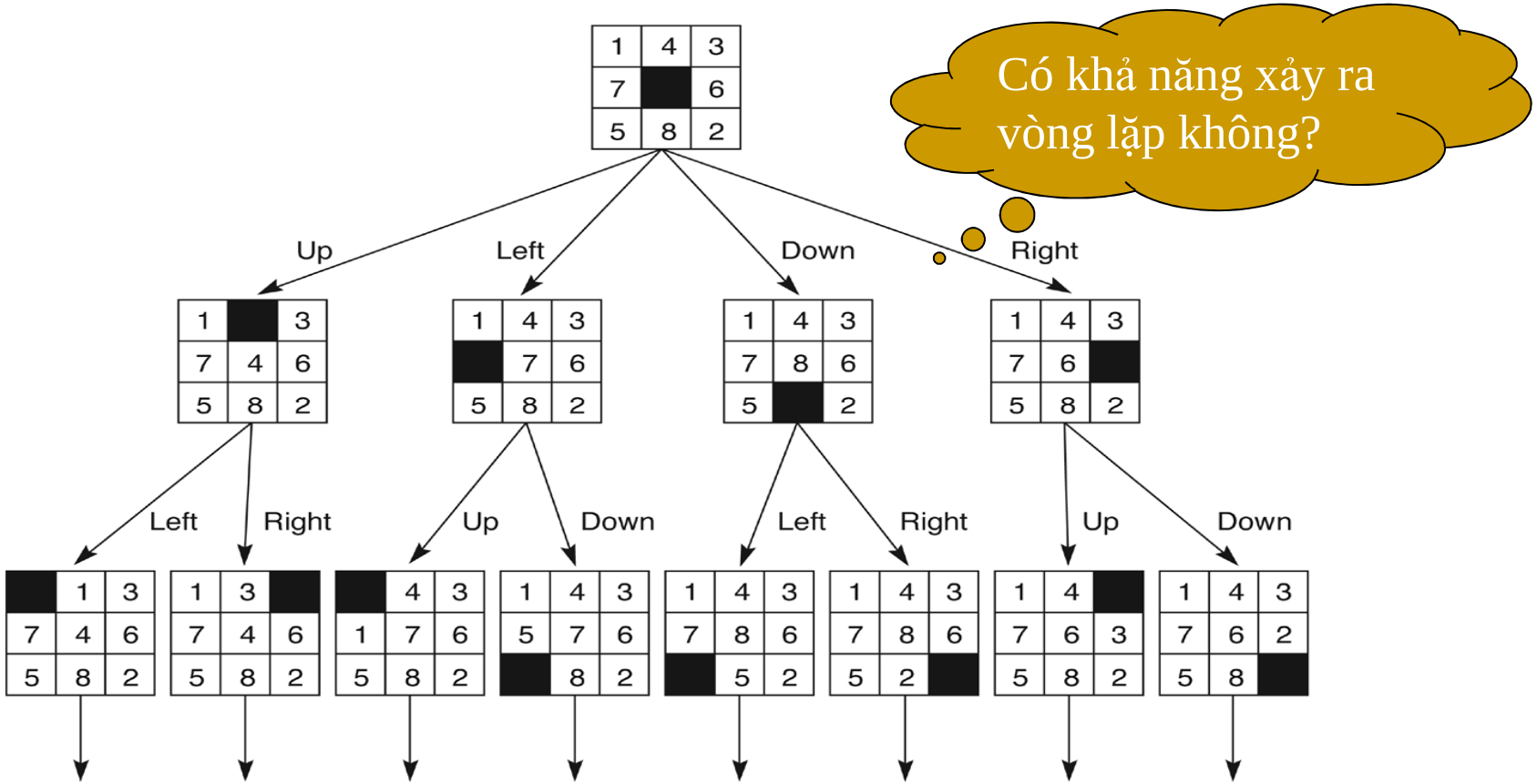
- State space:
  - Cities
- Successor function:
  - Go to adj city with cost = dist
- Start state:
  - Arad
- Goal test:
  - Is state == Bucharest?
- Solution?

Bài toán: Tic tac toe

Đồ thị có hướng không lặp lại (directed acyclic graph)



Bài toán: 8 puzzle



Chiến lược tìm kiếm?

- Khi tìm kiếm lời giải, từ một trạng thái nào đó chưa phải là trạng thái đích, ta dựa theo hàm **succs** sinh ra tập các trạng thái mới (mở rộng)
- Để được lời giải, ta phải liên tục chọn trạng thái mới, mở rộng, kiểm tra cho đến khi tìm được trạng thái đích hoặc không mở rộng được KGTT
- Tập các trạng thái được mở rộng sẽ có nhiều phần tử, việc chọn trạng thái nào để tiếp tục mở rộng được gọi là chiến lược tìm kiếm

Đánh giá một chiến lược?

- Tính đầy đủ: chiến lược phải đảm bảo tìm được lời giải (nếu có lời giải)?
- Tính tối ưu: lời giải có tốt hơn so với một số chiến lược khác hay không?
- Độ phức tạp không gian: cần bao nhiêu đơn vị bộ nhớ để tìm được lời giải?
- Độ phức tạp thời gian: cần bao nhiêu thời gian để tìm được lời giải?

Thông tin mỗi nút ?

- Nội dung trạng thái mà nút hiện hành đang biểu diễn
- Nút cha đã sinh ra nó
- **succs** đã được sử dụng để sinh ra nút hiện hành
- Độ sâu của nút (tính từ nút gốc)
- Giá trị đường đi từ nút gốc đến nút hiện hành

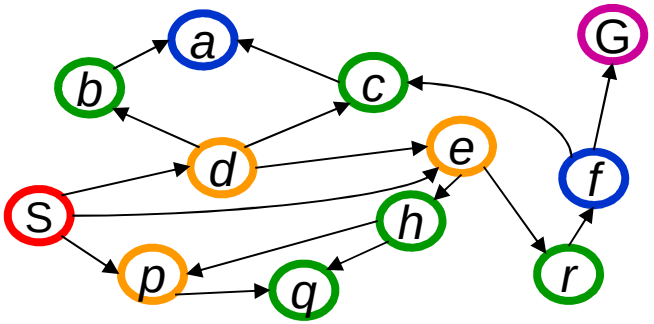
Tìm kiếm mù?

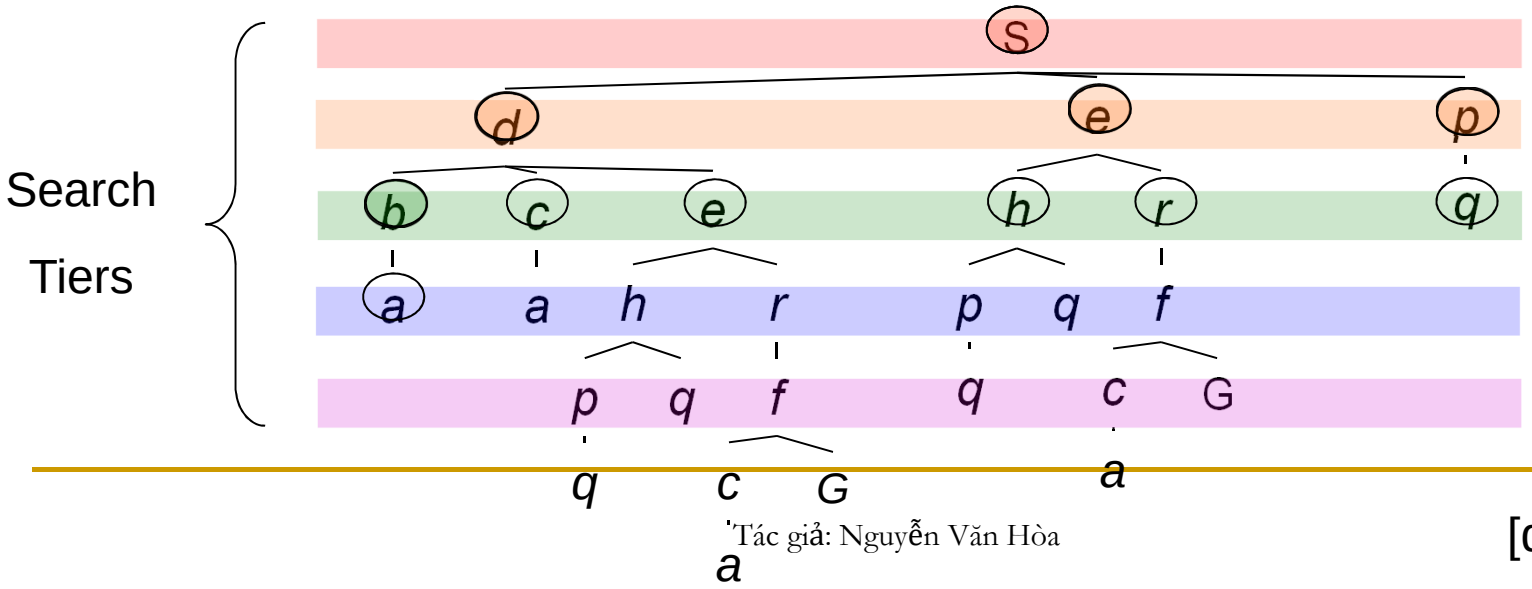
- Trạng thái được chọn để phát triển dựa theo cấu trúc của KGTT mà dùng thông tin hỗ trợ
- Là chiến lược tìm kiếm mù không hiệu quả
- Đây là cơ sở để chúng ta cải tiến và thu được những chiến lược hiệu quả hơn
- Hai giải thuật tìm kiếm mù
  - Tìm kiếm theo chiều rộng (Breadth-First-Search)
  - Tìm kiếm theo chiều sâu (Depth First Search)

Tìm kiếm theo chiều rộng

Strategy: expand shallowest node first

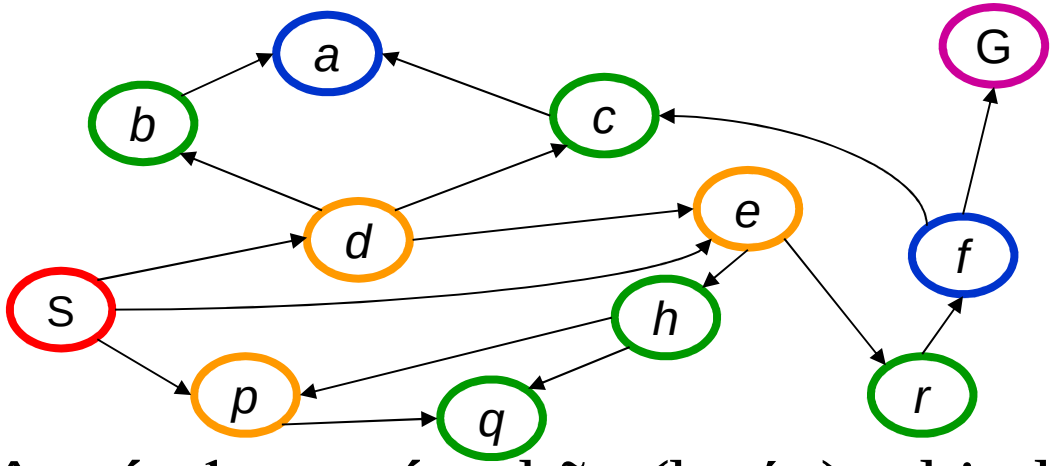
Implementation: Fringe is a FIFO queue





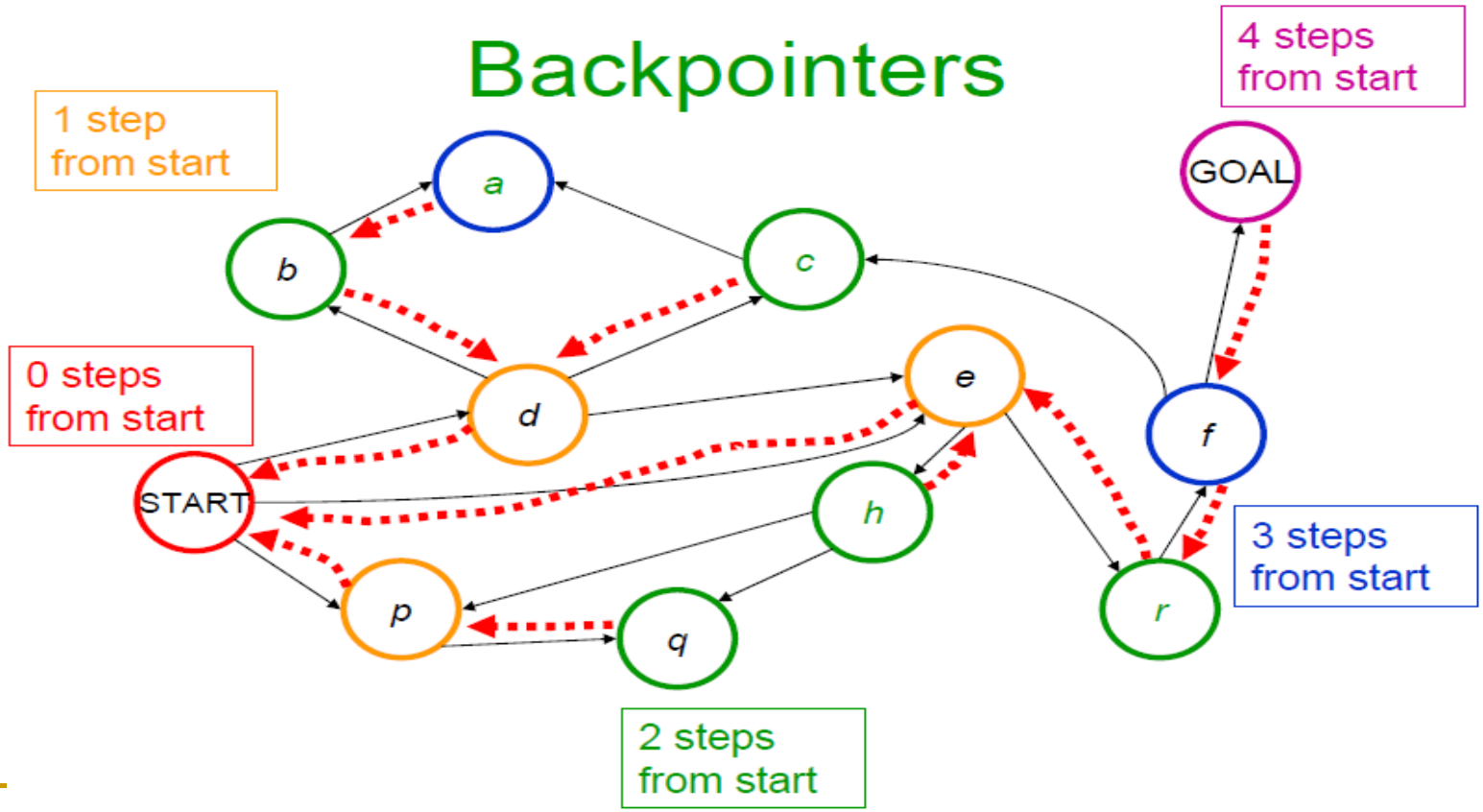
[demo: bfs]

Ghi nhớ đường đi

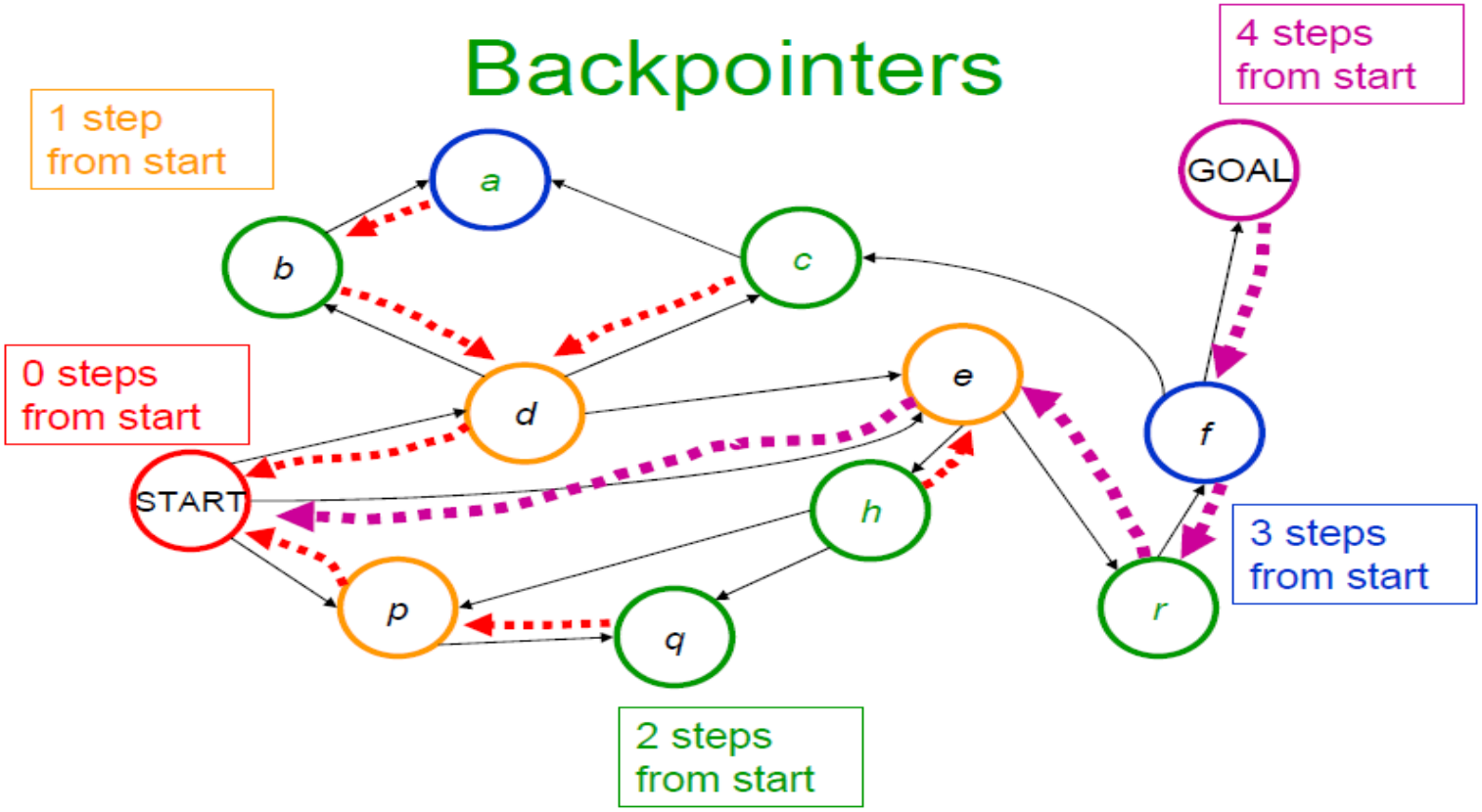


- Khi một nút được gán nhãn (bước), ghi nhận trạng thái trước đó (con trỏ quay lui). Tất cả ghi nhận được dùng để phát sinh lời giải khi đã đến đích
  - Tôi đã đến đích, trước khi tới đích tôi đã ở f, rồi r..
  - Do đó lời giải sẽ là S → e → r → f → G

Con trỏ quay lui - backpointers



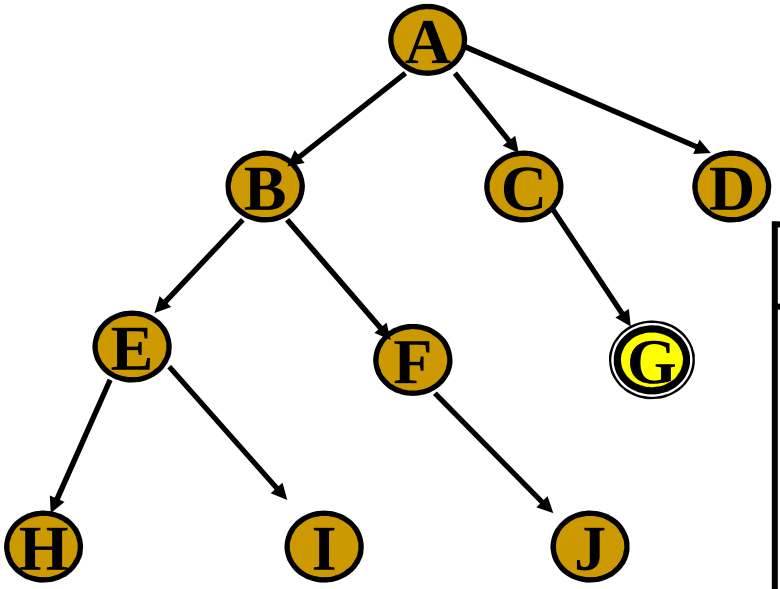
Con trỏ quay lui



Giải thuật -BFS

Open :=[**START**]
Close:= ∅
previous(**START**)=**NULL**
WHILE (Open không chứa GOAL và khác rỗng) **do**
  Lấy TT **s** nằm bên trái nhất trong **Open**
  Đặt **s** vào **Close**;
  for mỗi TT **s'** trong **succs**(**s**)
    IF s' chưa gán nhãn (chưa xét) **then**
      Đặt previous(**s'**) := **s**;
      Đưa **s'** vào bên **PHẢI** nhất của **Open**
IF Open rỗng **return FAILURE**
Else Xây dựng lời giải.
  Định nghĩa S_k = GOAL; Đường đi được tính dựa trên hàm **previous** với S_{k-1} = **previous**(S_k). Cho đến khi S_{k-1} là **START**

Tìm kiếm theo chiều rộng(tt)

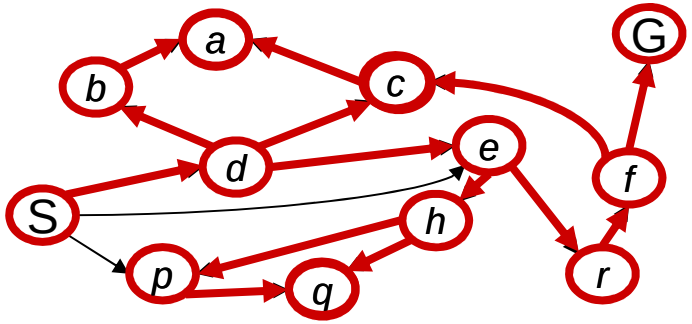


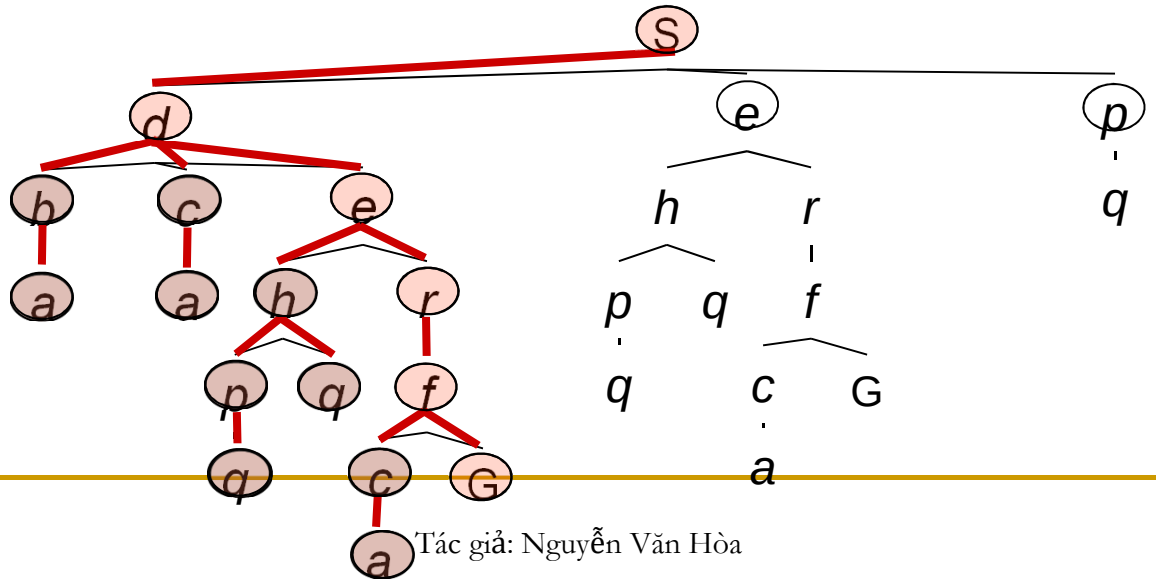
Lần lặp	X	Open	Close
0		[A]	[ ]
1	A	[B C D]	[A]
2	B	[C D E F]	[A B]
3	C	[D E F G]	[A B C]
4	D	[E F G]	[A B C D]
5	E	[F G H I]	[A B C D E]
6	F	[G H I J]	[A B C D E F]
7	**G**	[H I J]	[A B C D E F]

Depth First Search

Strategy: expand deepest node first

Implementation: Frontier is a LIFO stack



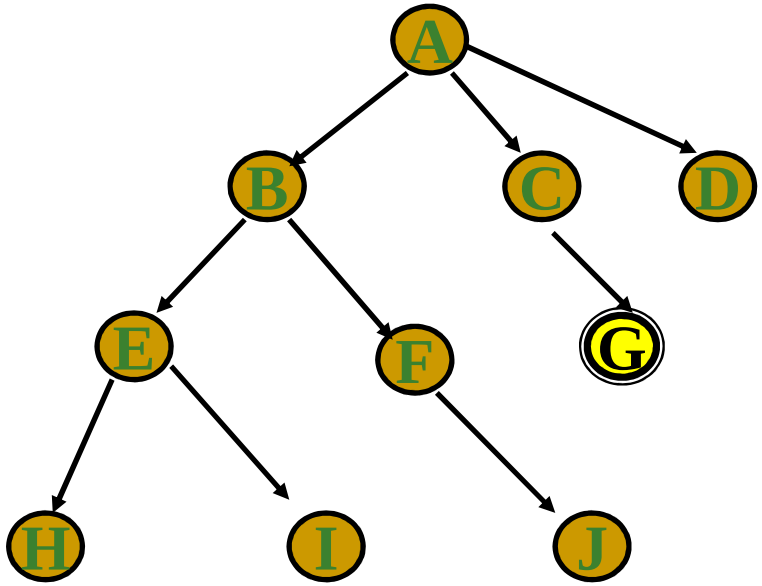


[demo: dfs]

Depth-First-Search (tt)

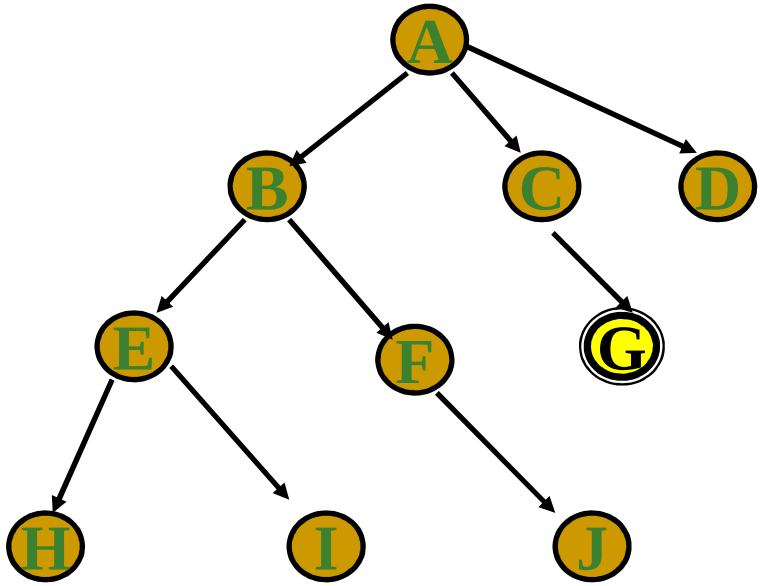
Open :=[**START**]
Close:= ∅
previous(**START**)=**NULL**
WHILE (Open không chứa GOAL và khác rỗng) **do**
  Lấy TT **s** nằm bên trái nhất trong **Open**
  Đặt **s** vào **Close**;
  for mỗi TT **s'** trong **succs**(**s**)
    IF s chưa gán nhãn (chưa xét) **then**
      Đặt **previous**(**s'**) := **s**;
      Đưa **s'** vào bên **TRÁI** nhất của **Open**
IF Open rỗng **return FAILURE**
Else Xây dựng lời giải.
  Định nghĩa S_k = GOAL; Đường đi được tính dựa trên hàm **previous** với S_{k-1} = **previous**(S_k). Cho đến khi S_{k-1} là **START**

Depth-First-Search (tt)



Lần lặp	X	Open	Close
0		[A]	[ ]
1	A	[B C D]	[A]
2	B	[E F C D]	[A B]
3			

Depth-First-Search (tt)



Lần lặp	X	Open	Close
0		[A]	[ ]
1	A	[B C D]	[A]
2	B	[E F C D]	[A B]
3	E	[H I F C D]	[A B E]
4	H	[I F C D]	[A B E H]
5	I	[F C D]	[A B E H I]
6	F	[J C D]	[A B E H I F]
7	J	[C D]	[A B E H I F J]
8	C	[G D]	[A B E H I F J C]
9	**G**		

Breath First vs Depth First

- Breath First: “open” được tổ chức dạng FIFO (Queue)
- Depth First: “open” được tổ chức dạng LIFO (Stack)
- Đặc tính
  - Breath First search hiệu quả khi lời giải nằm gần gốc của cây tìm kiếm, tìm nhiều lời giải
  - Depth First search hiệu quả khi lời giải nằm sâu trong cây tìm kiếm và có một phương án chọn hướng đi chính xác
- Kết quả
  - Breath First search chắc chắn tìm ra kết quả nếu có
  - Depth First có thể sa lầy
- Bùng nổ tổ hợp là khó khăn lớn nhất cho các giải thuật này

Depth first search có giới hạn

- Depth first search có khả năng lặp vô tận do các trạng thái con sinh ra liên tục → độ sâu tăng vô tận
- Khắc phục bằng cách giới hạn độ sâu của giải thuật
- Sâu bao nhiêu thì vừa? → chiến lược giới hạn:
  - Cố định một độ sâu MAX, như các danh thủ chơi cờ tính trước được số nước nhất định
  - Theo cấu hình tài nguyên của máy tính
  - Meta knowledge trong việc định giới hạn độ sâu
- Giới hạn độ sâu ⇒ co hẹp không gian trạng thái ⇒ có thể mất nghiệm hoặc không tìm thấy nghiệm

Các đặc trưng của bài toán

- Một số yếu tố cần phân tích khi chọn kỹ thuật giải bài toán:
  - Khả năng phân rã bài toán
  - Khả năng lờ đi và quay lui
  - Khả năng dự đoán toàn cục
  - Đích là một trạng thái hay con đường (tập các TT)
  - Lượng tri thức cần để giải bài toán
  - Có cần sự can thiệp của con người trong quá trình giải không?

Các đặc trưng của bài toán (tt)

- Khả năng phân rã bài toán
  - Phân rã được: như BT tính tích phân ký hiệu
    - Giải bằng cách
      - Chia nhỏ BT lớn thành các BT con độc lập
      - Giải từng BT nhỏ
      - Kết hợp thành BT lớn
  - Không phân rã được: BT thế giới các khối (??)

Các đặc trưng của bài toán (tt)

- Các bước giải có thể lờ đi hay quay lui
  - Có thể lờ đi : như BT chứng minh định lý
    - Vì: định lý vẫn đúng sau một vài bước áp dụng các luật
  - Có thể quay lui: như BT 8-puzzle
    - Vì: có thể di chuyển theo hướng ngược lại để về TT trước
  - Không thể quay lui: như BT chơi cờ
    - Vì: game over!

Các đặc trưng của bài toán (tt)

- Các bước giải có thể lờ đi hay quay lui
  - Có thể lờ đi
    - Có thể áp dụng chiến lược điều khiển đơn giản không cần quay lui
    - Dể dàng hiện thực
  - Có thể quay lui
    - Chiến lược phức tạp hơn để quay lui được tại những bước lỗi.
    - Có thể dùng Push-Down Stack
  - Không thể quay lui
    - Dùng các chiến lược phức tạp hơn vì mỗi khi ra quyết định thì đó là quyết định cuối cùng
    - Có thể dùng giải pháp Planning

Các đặc trưng của bài toán (tt)

- Khả năng dự đoán của bài toán:
  - Có thể dự đoán được: như BT 8 puzzle

    $\Rightarrow$ có thể đề ra 1 chuỗi các nước đi và tự tin vào kết quả sẽ xãy ra

    $\Rightarrow$ Có thể quay lui được
  - Không thể dự đoán được: như các game có đối kháng
    - Cần theo đuổi nhiều kế hoạch
    - Có chiến lược/đánh giá để chọn kế hoạch tốt

Các đặc trưng của bài toán (tt)

- Lời giải là tuyệt đối hay tương đối
  - Tuyệt đối (best-path): như bài toán TSP
    - Tính toán khó hơn (tổng quát)
    - Cần giải thuật tìm kiếm toàn diện hơn
  - Tương đối (any-path): như bài toán suy luận đời thường (xem sau)
    - Có thể dùng heuristic để giải trong thời gian hợp lý

Các đặc trưng của bài toán (tt)

- Lời giải là trạng thái hay con đường (tập các TT)
  - Trạng thái: như bài toán tìm ra cách hiểu phù hợp cho câu.
  - Ví dụ:
    - "The bank president ate a dish of pasta salad with the fork."
    - Từng từ như: bank, president, … có thể được hiểu theo nhiều cách
    - Một kiểu tìm kiếm nào đó được thực hiện để tìm ra cách hiểu toàn bộ cho câu
  - Con đường
  - Song, điều này cũng tương đối. Vì có thể biểu diễn trạng thái để nó có thể bao gồm thông tin về một phần hay toàn bộ con đường

Các đặc trưng của bài toán (tt)

- Vai trò của tri thức là gì?
  - Cần ít tri thức:
    - Như bài toán: “chơi cờ”
    - Tri thức ~ luật để di chuyển hợp lệ, cơ chế điều khiển, chiến lược điều khiển để tăng tốc tìm kiếm
  - Cần nhiều tri thức
    - Như bài toán: Hiểu câu chuyện trên tạp chí
    - Tri thức: nhiều, cả những cái đã ghi tường minh và cả những cái
    - không được ghi trong chính câu chuyện

Vấn đề trong thiết kế CT tìm kiếm

- Sự tìm kiếm
  - Tìm kiếm ~ duyệt cây, từ TT bắt đầu -> TT đích
  - Cả cây tìm kiếm thường không được xây dựng sẵn
  - Cấu trúc đồ thị thường thay thế cho cây trong biểu diễn KGTT
- Các vấn đề
  - Xác định hướng tìm (forward hay backward reasoning)
  - Cách lựa chọn luật để áp dụng (matching)
  - Cách biểu diễn nút (NODE) của quá trình tìm kiếm
  - Các NODE trong đồ thị có thể được phát sinh và xem xét nhiều lần trong quá trình duyệt $\Rightarrow$ cần loại bỏ những NODE lặp lại $\Rightarrow$ Cần lưu lại các NODE đã xét.

Vấn đề trong thiết kế CT …

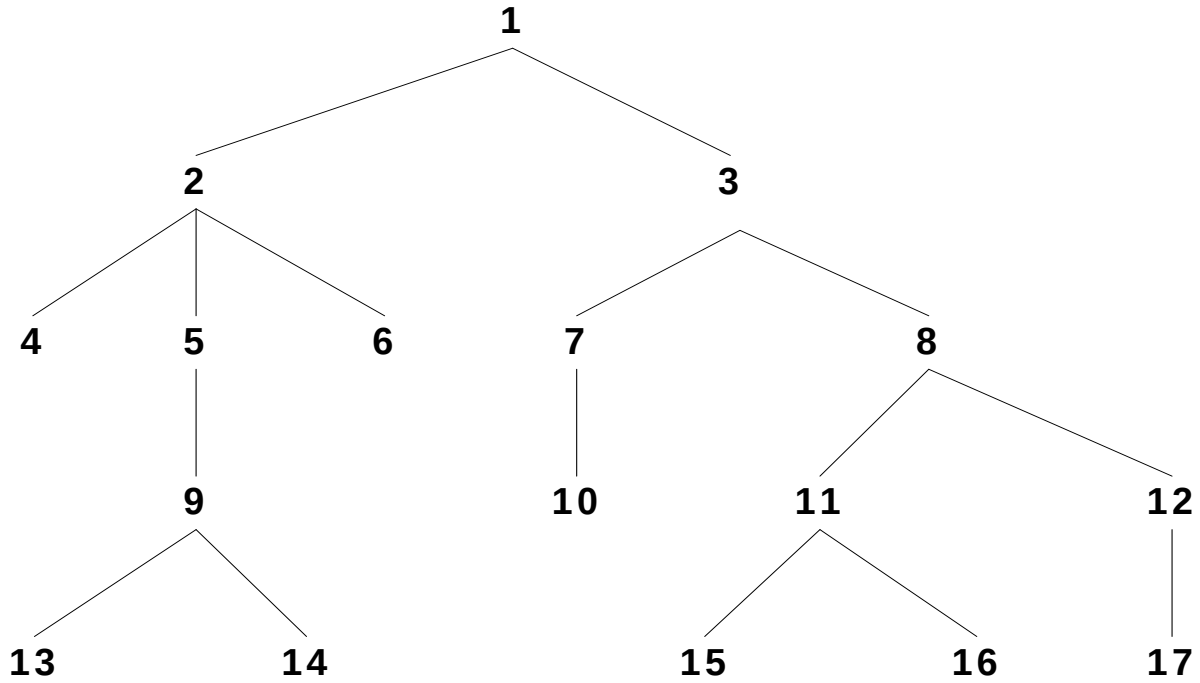
- Giải thuật kiểm tra NODE lặp lại (DFS)
  - Xem xét tập NODE đã tạo ra, để xem NODE mới đã có chưa
  - Nếu chưa thì thêm NODE mới vào đồ thị
  - Nếu đã có:
    - Thiết lập điểm mở rộng kế tiếp là con của NODE đang tồn tại , NODE có thể bỏ đi
    - Nếu GT có lưu giữ con đường tốt nhất hiện có thì cần xem xét xem nó đạt đến NODE mới trên con đường tốt hơn không, nếu vậy thì cập nhật lại con đường tốt nhất

Link minh họa tìm kiếm

- 8-Puzzle với GT TK theo chiều rộng & sâu
  - http://www.cs.rmit.edu.au/AI-Search/Product (RMIT)
- Map Search với GT TK theo chiều rộng & sâu
  - http://www.ai.mit.edu/courses/6.034f/searchpair.html

BÀI TẬP 1

Xét đồ thị trạng thái sau đây, với mỗi chiến lược tìm kiếm bên dưới hãy liệt kê với danh sách thứ tự các nút được duyệt qua:



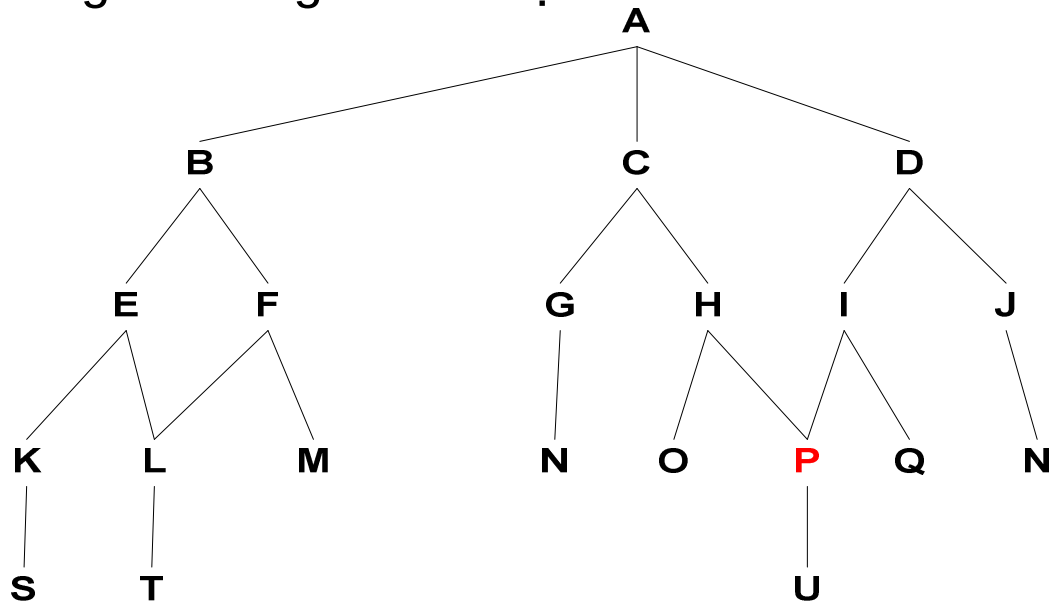
1/ Tìm kiếm rộng (BFS)

2/ Tìm kiếm sâu (DFS)

3/ Tìm kiếm sâu với độ sâu là 3

BÀI TẬP 2

Giả sử P là nút mục tiêu của đồ thị bên dưới. Hãy liệt kê danh sách thứ tự các nút duyệt qua ứng với từng chiến lược tìm kiếm.



1/ Tìm kiếm rộng (BFS)

2/ Tìm kiếm sâu (DFS)

3/ Tìm kiếm sâu với độ sâu là 3