HỆ THỐNG TÌM KIẾM (LÊN KẾ HOẠCH)

Trần Trung Kiên

ttkien@fit.hcmus.edu.vn

Tổng thể

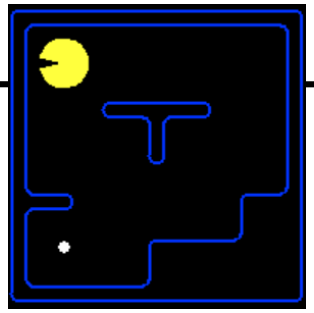
- Hệ thống lên kế hoạch (planning agent)
- Bài toán tìm kiếm (bài toán lên kế hoạch)
- Giải quyết bài toán tìm kiếm
  - Thuật toán Depth-First Search
  - Thuật toán Breadth-First Search
  - Thuật toán Uniform Cost Search

Hệ thống lên kế hoạch (planning agent)

Xem demo

- Pacman muốn ăn hết thức ăn với số bước đi ít nhất
- Đầu tiên, Pacman **lên kế hoạch** trong đầu: xem xét các kế hoạch khác nhau và chọn ra kế hoạch tốt nhất
  - Trong đầu Pacman cần có mô hình về "thế giới": "thế giới" sẽ thay đổi như thế nào khi Pacman thực hiện các hành động
- Sau đó, Pacman thực thi kế hoạch này

Bài toán tìm kiếm



Tìm đường đi đến góc trái dưới

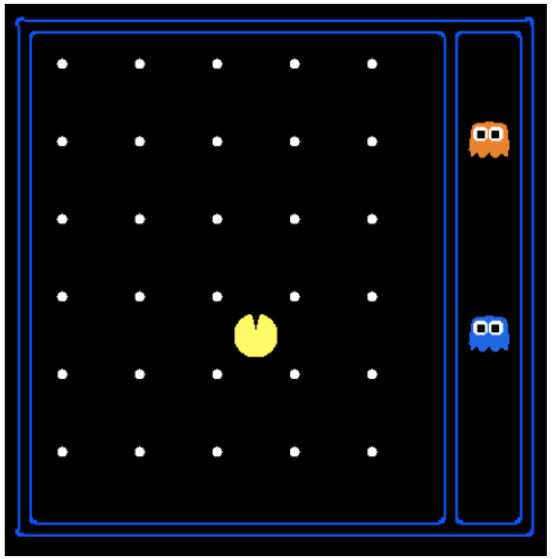
- Một **bài toán tìm kiếm** bao gồm:
  - Không gian trạng thái
    - Xét vd ở bên …
  - Hàm "successor"
    - Cho biết từ một trạng thái sẽ có thể đi đến những trạng thái kế nào (với hành động nào và chi phí bao nhiêu)
    - Xét vd ở bên …
  - Trạng thái bắt đầu, và hàm kiểm tra trạng thái đích
    - Xét vd ở bên …
- Bài toán tìm kiếm là mô hình về thế giới
- Một **lời giải** là một kế hoạch gồm một chuỗi các hành động mà sẽ chuyển từ trạng thái bắt đầu sang trạng thái đích

Không gian trạng thái

- **Trạng thái thế giới thực** bao gồm mọi chi tiết của môi trường (vd, vị trí của Pacman, hướng quay mặt của Pacman, vị trí các điểm thức ăn, vị trí các con ma, …)
- **Trạng thái trong bài toán tìm kiếm** chỉ gồm các thông tin cần thiết cho bài toán tìm kiếm
  - Bài toán tìm đường đi cho Pacman để đến một vị trí nào đó
    - Tập trạng thái = {vị trí Pacman}
  - Bài toán tìm đường đi cho Pacman để ăn hết thức ăn
    - Tập trạng thái = {(vị trí Pacman, mảng bool cho biết tình trạng các điểm thức ăn)}

Không gian trạng thái

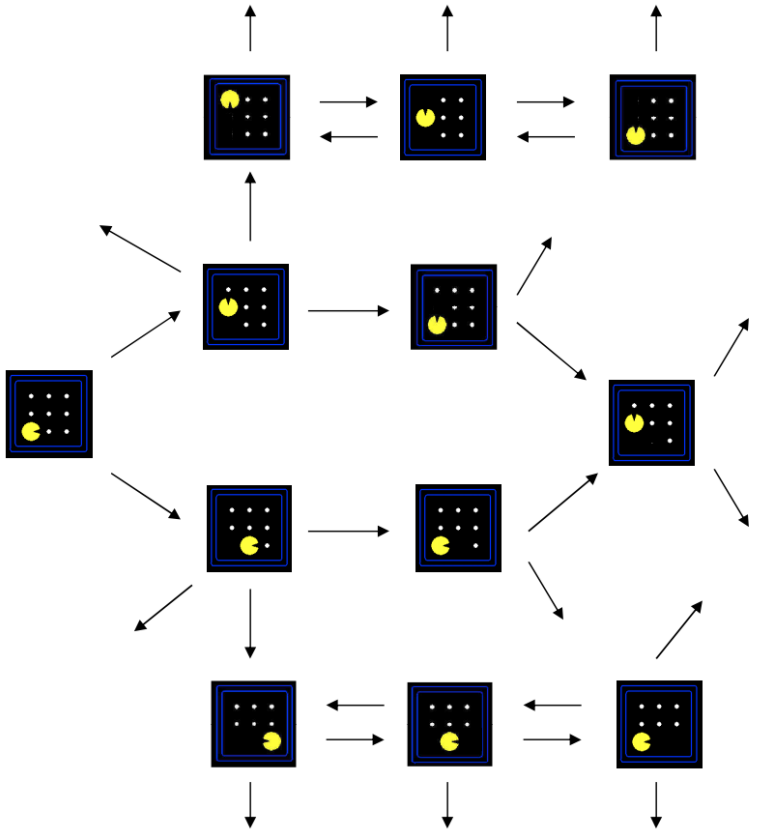
- Cho:
  - Số lượng vị trí của Pacman: 120
  - Số lượng thức ăn: 30
  - Số lượng vị trí của ma: 12
  - Số lượng hướng quay mặt của Pacman: 4
- Hỏi:
  - Số lượng trạng thái thế giới thực?
  - Số lượng trạng thái cho bài toán đến một vị trí?
  - Số lượng trạng thái cho bài toán ăn hết thức ăn?



Nguồn ảnh: UC Berkely. CS188 Intro to AI

Đồ thị (không gian) trạng thái

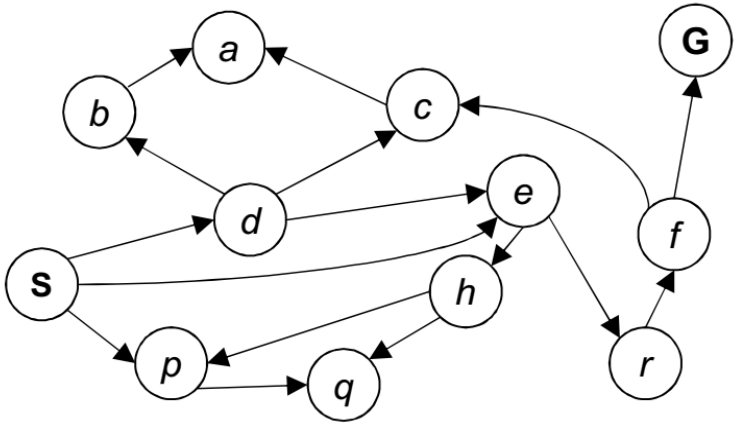
- Là một cách biểu diễn của bài toán tìm kiếm
  - Mỗi đỉnh ứng với một trạng thái
  - Các cạnh đi ra từ một đỉnh ứng với hàm successor tại đỉnh đó
- Trong đồ thị trạng thái, mỗi trạng thái chỉ xuất hiện một lần
- Ta sẽ không xây dựng cả đồ thị này vì thường sẽ rất lớn, nhưng đồ thị này sẽ giúp ích cho việc hình dung



Nguồn ảnh: UC Berkely. CS188 Intro to AI

Đồ thị (không gian) trạng thái

- Là một cách biểu diễn của bài toán tìm kiếm
  - Mỗi đỉnh ứng với một trạng thái
  - Các cạnh đi ra từ một đỉnh ứng với hàm successor tại đỉnh đó
- Trong đồ thị trạng thái, mỗi trạng thái chỉ xuất hiện một lần
- Ta sẽ không xây dựng cả đồ thị này vì thường sẽ rất lớn, nhưng đồ thị này sẽ giúp ích cho việc hình dung

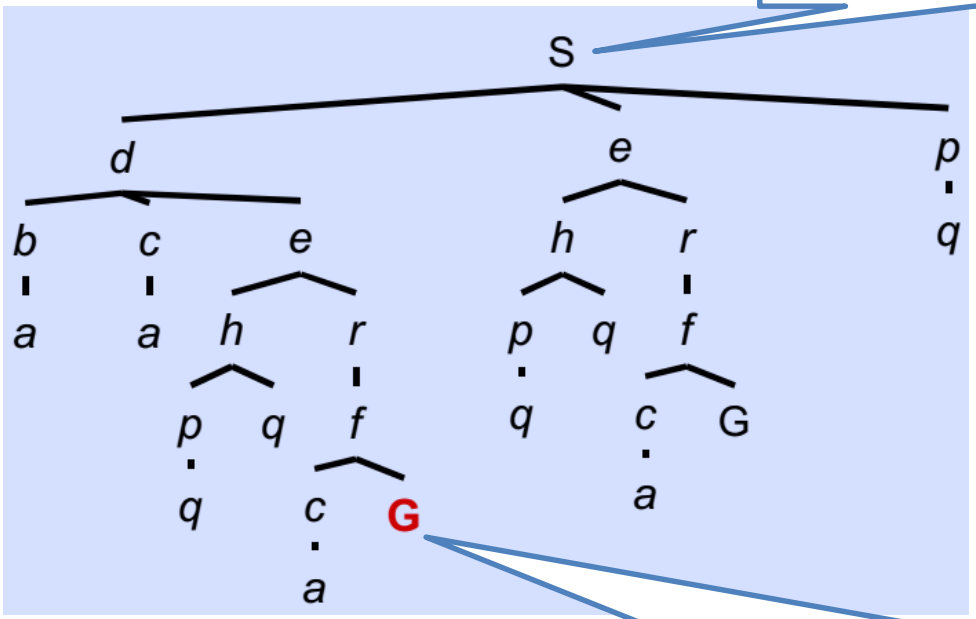


Khi dạy, để giúp hình dung, thường sẽ dùng một bài toán tìm kiếm nhỏ mà có thể vẽ được đồ thị trạng thái

Nguồn ảnh: UC Berkely. CS188 Intro to AI

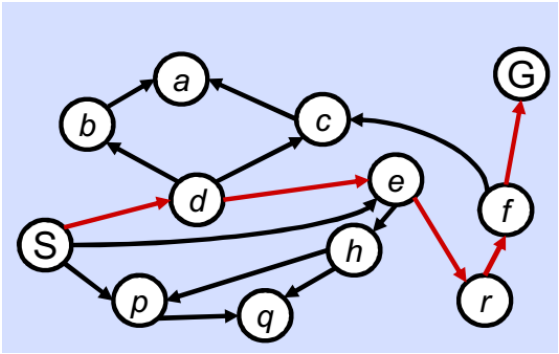
Cây tìm kiếm

Đỉnh gốc là trạng thái bắt đầu (hiện tại)



Các tương lai có thể xảy ra

Đồ thị trạng thái tương ứng



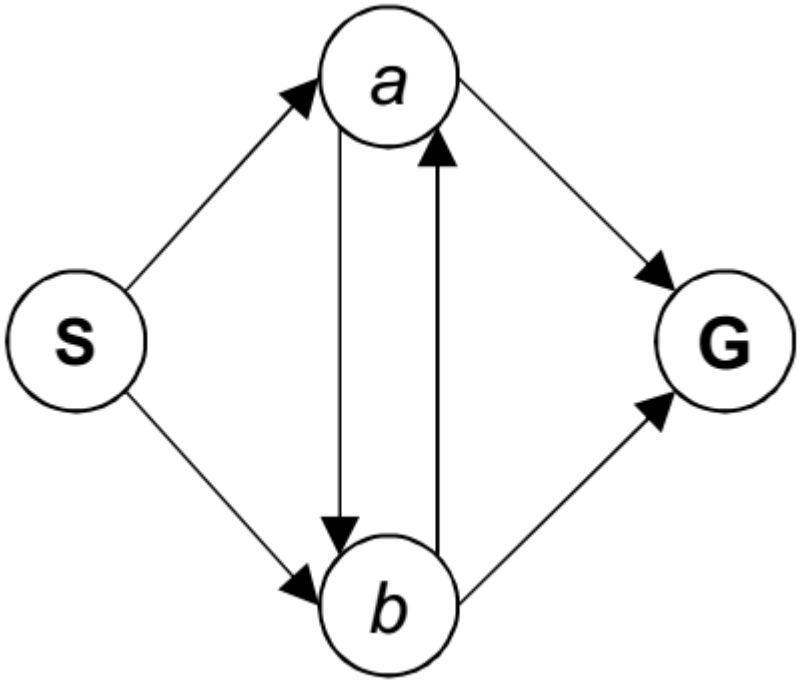
Một đỉnh của cây không chỉ là một trạng thái mà là **một kế hoạch**: một đường đi từ S đến trạng thái này

Do đó, trong cây, một trạng thái có thể xuất hiện nhiều lần

Nguồn ảnh: UC Berkely. CS188 Intro to AI

Quiz: cây tìm kiếm vs đồ thị trạng thái

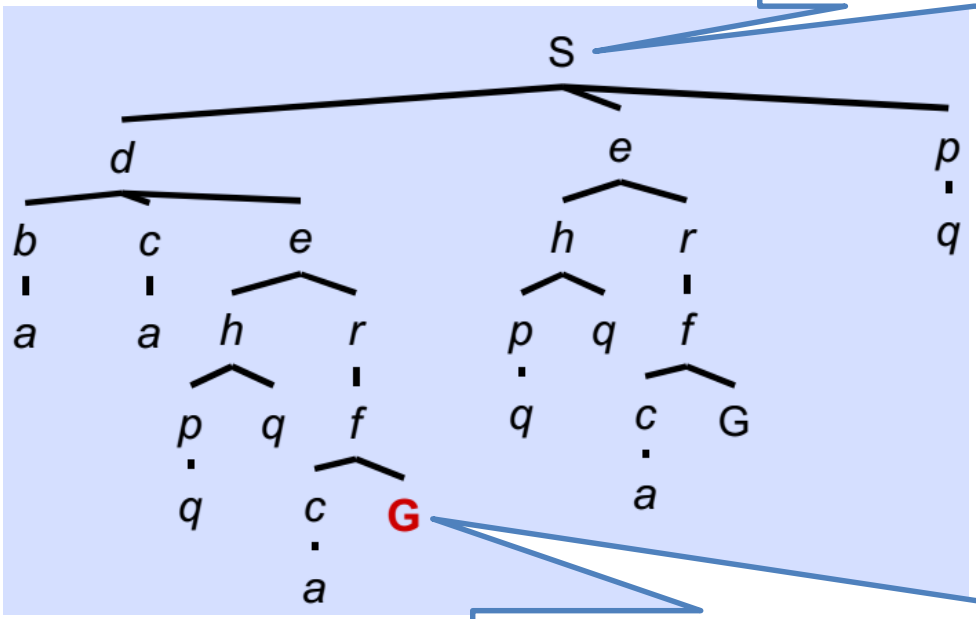
Cây tìm kiếm (bắt đầu từ S) tương ứng với đồ thị ở dưới có bao nhiêu đỉnh?



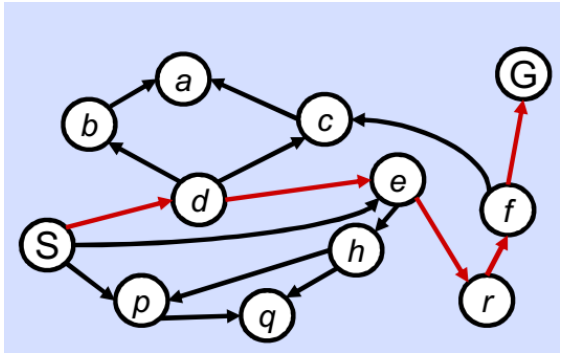
Nguồn ảnh: UC Berkely. CS188 Intro to AI

Cây tìm kiếm

Đỉnh gốc là trạng thái bắt đầu (hiện tại)



Đồ thị trạng thái tương ứng



Một đỉnh của cây không chỉ là một trạng thái mà là **một kế hoạch**: một đường đi từ S đến trạng thái này

Do đó, trong cây, một trạng thái có thể xuất hiện nhiều lần

Ta cũng không xây cả cây tìm kiếm này vì kích thước còn có thể lớn hơn đồ thị trạng thái

Các thuật toán tìm kiếm sẽ chỉ xây một phần của cây này, bắt đầu từ trạng thái bắt đầu S cho đến khi tìm được trạng thái đích G

Tổng thể

- Hệ thống lên kế hoạch (planning agent)
- Bài toán tìm kiếm (bài toán lên kế hoạch)
- Giải quyết bài toán tìm kiếm
  - Thuật toán Depth-First Search
  - Thuật toán Breadth-First Search
  - Thuật toán Uniform-Cost Search

Các thuật toán tìm kiếm

Ý tưởng: từ trạng thái bắt đầu, ta sẽ tiến hành nhiều vòng lặp; ở mỗi vòng lặp, ta sẽ phát triển dần cây tìm kiếm theo một chiến lược nào đó; làm cho đến khi tìm được trạng thái đích

Cụ thể hơn:

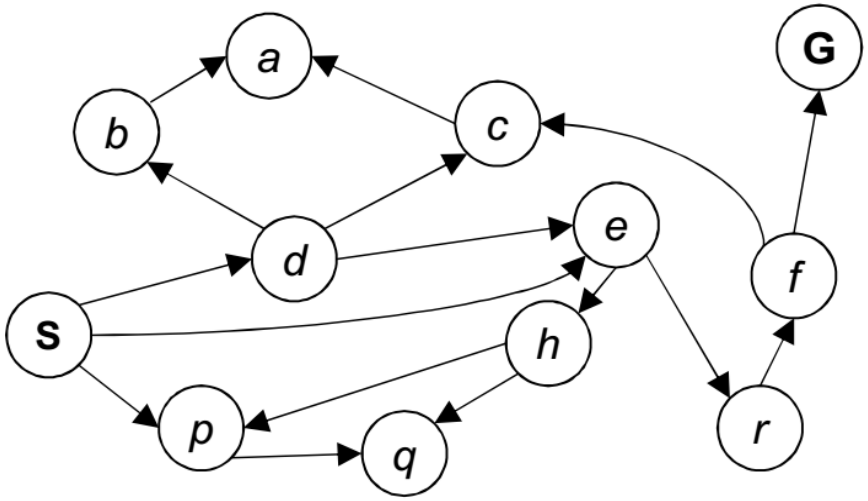
- Trong quá trình tìm kiếm, ta sẽ có biến fringe lưu danh sách các kế hoạch tiềm năng (ứng với các node lá của cây đang xây dựng).
- Ở mỗi vòng lặp, ta sẽ lấy ra từ fringe một kế hoạch theo một chiến lược nào đó và mở rộng ra (dựa vào hàm successor), rồi thêm các kế hoạch mới này vào fringe (ứng với việc phát triển cây)
- Cứ thế cho đến khi kế hoạch được lấy ra đến được trạng thái đích, hoặc trong fringe không còn gì nữa (không tìm được lời giải)

Thuật toán Depth-First Search (DFS)

Nguồn ảnh: UC Berkely. CS188 Intro to AI

Thuật toán DFS

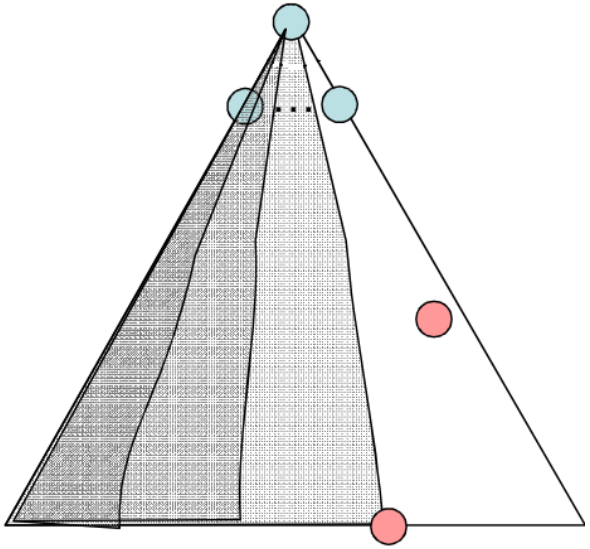
- Chiến lược chọn kế hoạch ở fringe để mở rộng: chọn kế hoạch **"sâu" nhất**
- Có thể dùng stack để cài đặt fringe
- Chạy DFS với đồ thị ở dưới: xem trên bảng …



Nguồn ảnh: UC Berkely. CS188 Intro to AI

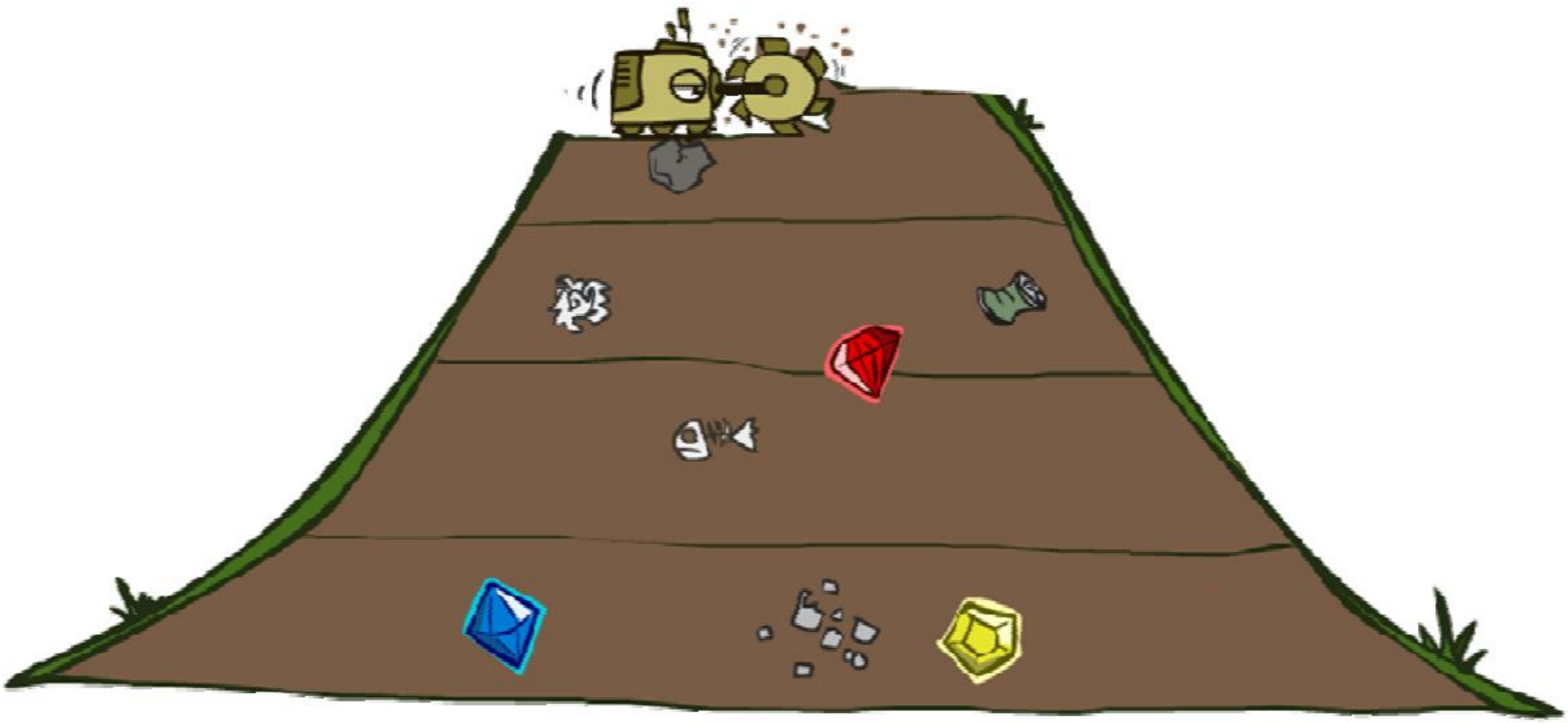
Nhận xét về DFS

- DFS duyệt cây theo hướng từ trái qua phải (hoặc từ phải qua trái tùy theo cách cài đặt)
- DFS có đảm bảo tìm được lời giải (nếu tồn tại)?
  - Nếu trong đồ thị có vòng lặp thì DFS sẽ có thể bị đào sâu mãi mãi; ta sẽ khắc phục vấn đề này ở bài sau
- DFS có đảm bảo tìm được lời giải tối ưu (nếu tồn tại)?
  - Không, DFS tìm được lời giải ở phía trái nhất của cây (nếu không gặp vấn đề bị đào sâu mãi mãi)



Nguồn ảnh: UC Berkely. CS188 Intro to AI

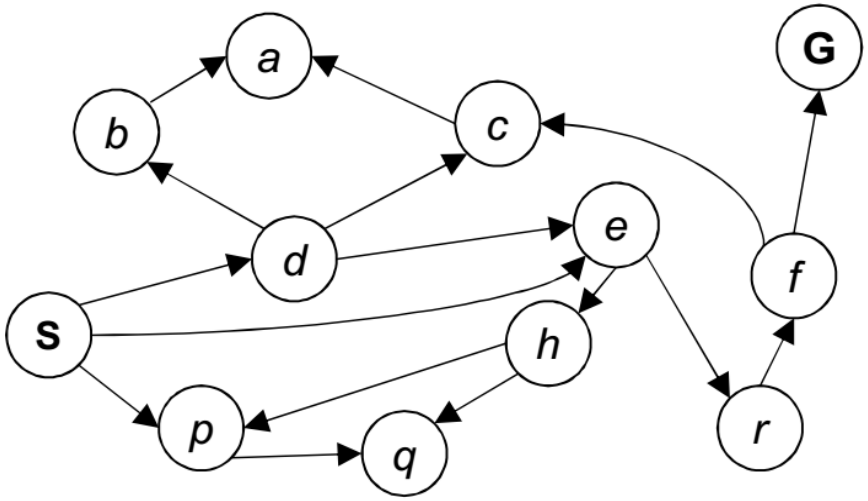
Thuật toán Breadth-First Search (BFS)



Nguồn ảnh: UC Berkely. CS188 Intro to AI

Thuật toán BFS

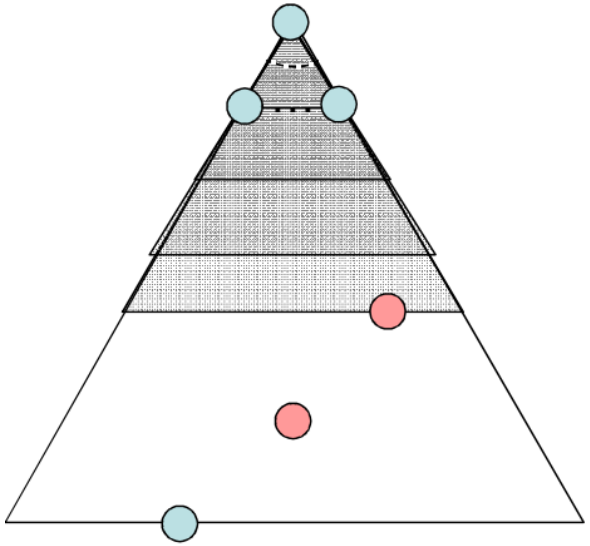
- Chiến lược chọn kế hoạch ở fringe để mở rộng: chọn kế hoạch **"nông" nhất**
- Có thể dùng queue để cài đặt fringe
- Chạy BFS với đồ thị ở dưới: xem trên bảng …



Nguồn ảnh: UC Berkely. CS188 Intro to AI

Nhận xét về BFS

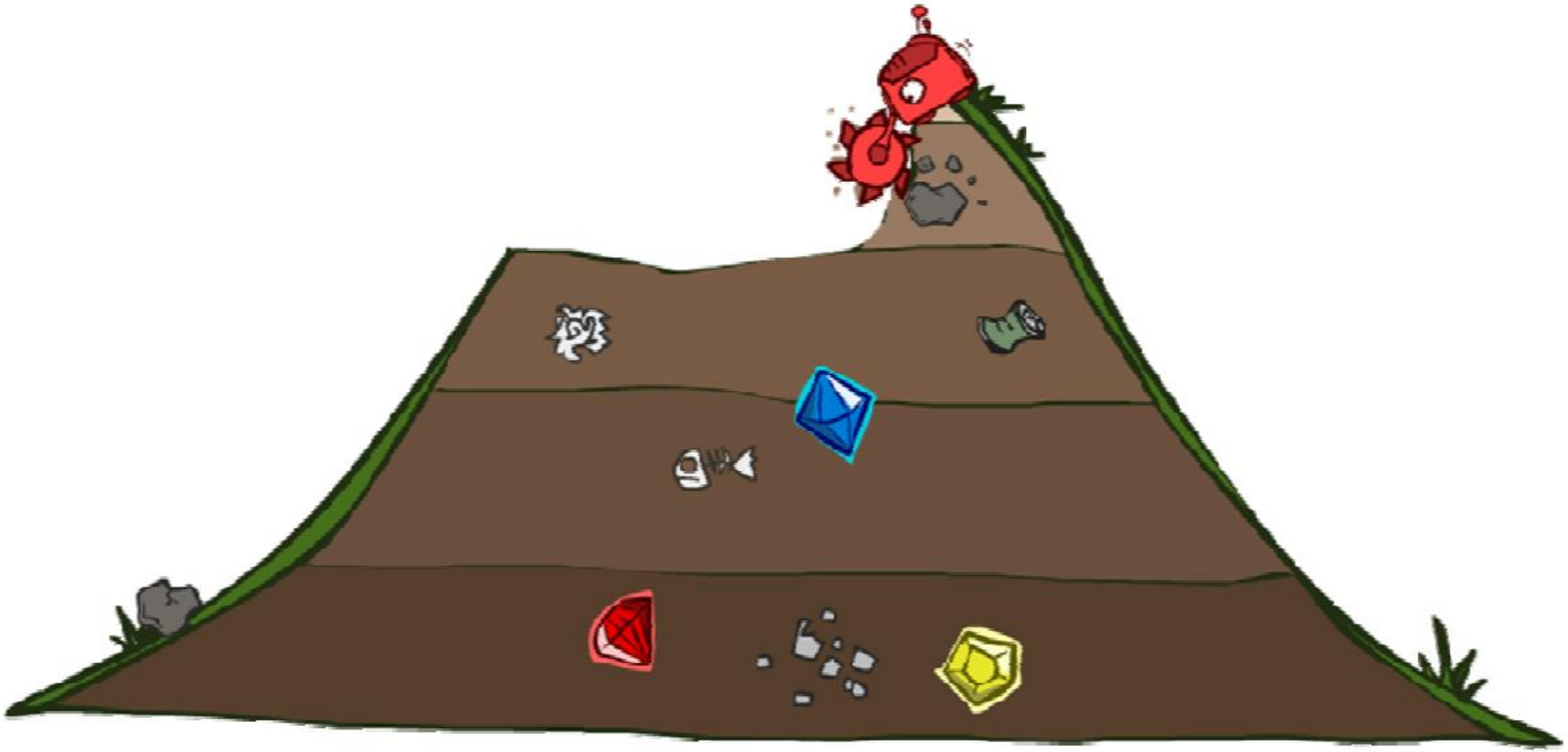
- BFS duyệt cây theo hướng từ trên xuống dưới
- BFS có đảm bảo tìm được lời giải (nếu tồn tại)?
  - Có
- BFS có đảm bảo tìm được lời giải tối ưu (nếu tồn tại)?
  - BFS tìm được lời giải "nông" nhất; lời giải "nông" nhất là lời giải tối ưu khi tất cả các cạnh đều có chi phí bằng nhau



Nguồn ảnh: UC Berkely. CS188 Intro to AI

Quiz: DFS vs BFS

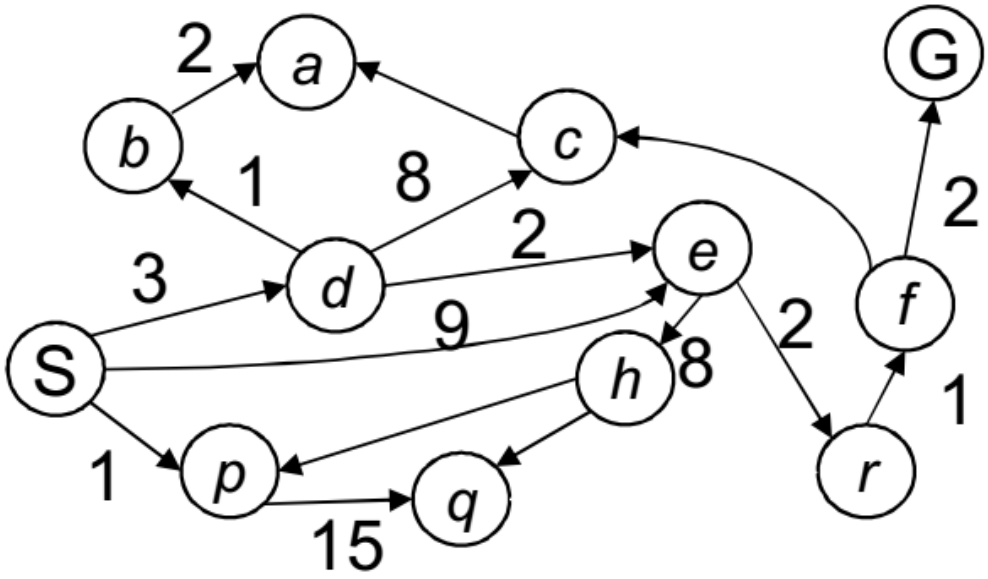
Thuật toán Uniform Cost Search (UCS)



Nguồn ảnh: UC Berkely. CS188 Intro to AI

Thuật toán UCS

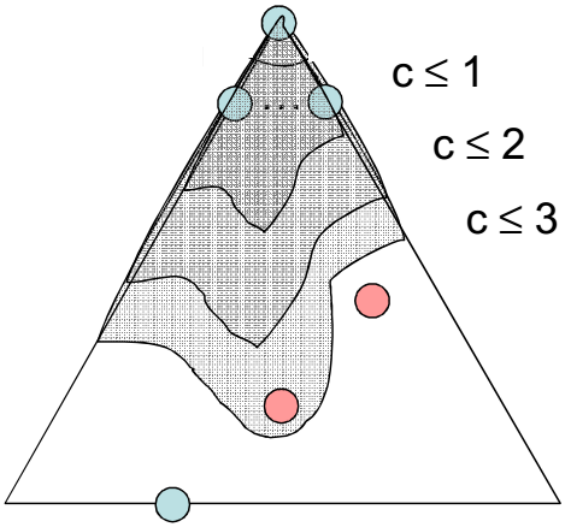
- Chiến lược chọn kế hoạch ở fringe để mở rộng: chọn kế hoạch **có chi phí nhỏ nhất**
- Có thể dùng priority queue để cài đặt fringe
- Chạy UCS với đồ thị ở dưới: xem trên bảng …



Nguồn ảnh: UC Berkely. CS188 Intro to AI

Nhận xét về UCS

- UCS duyệt cây tương tự như BFS, nhưng UCS mở rộng theo chí phí tăng dần (thay vì mở rộng theo chiều sâu tăng dần như ở BFS)
- UCS có đảm bảo tìm được lời giải (nếu tồn tại)?
  - Có
- UCS có đảm bảo tìm được lời giải tối ưu (nếu tồn tại)?
  - Có



Nguồn ảnh: UC Berkely. CS188 Intro to AI

Quiz: DFS vs BFS vs UCS