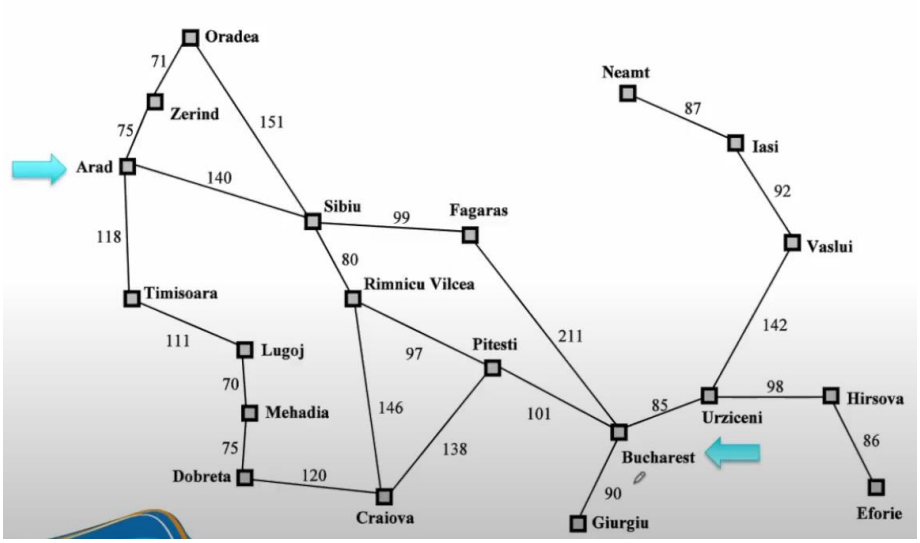
VẤN ĐỀ VỀ TÌM KIẾM

Bài toán Tìm kiếm có năm thành phần:

Q, S, G, succs, cost

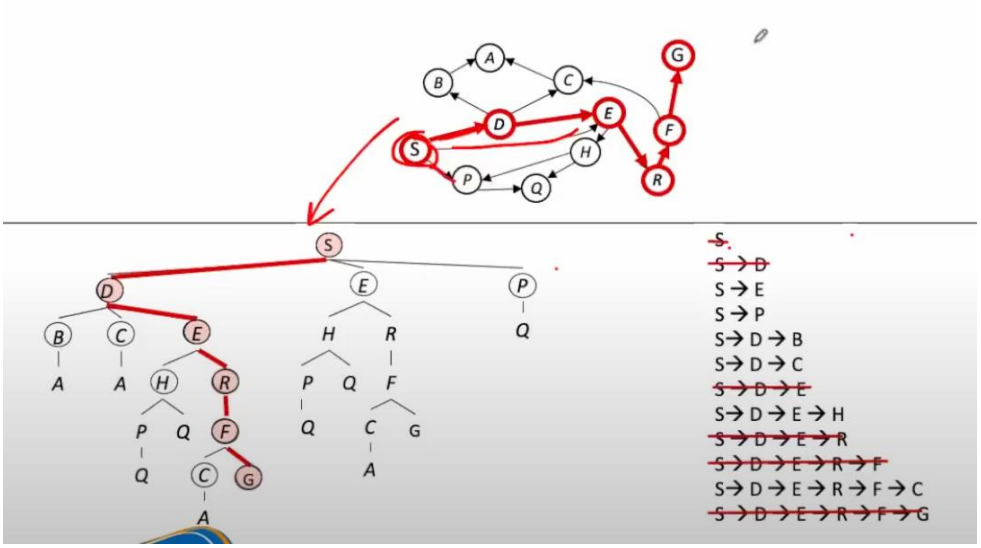
- Q: tập hữu hạn các trạng thái
- $S \subseteq Q$: tập khác rỗng gồm các trạng thái ban đầu
- $G \subseteq Q$: tập khác rỗng gồm các trạng thái đích
- **succs**(*s*): là hàm nhận một trạng thái *s* làm đầu vào và trả về một tập các trạng thái có thể đến từ s trong một bước.
- **cost**(*s, s'*): là hàm nhận hai trạng thái *s* và *s'* làm đầu vào và trả về chi phí di chuyển một bước từ *s* đến *s'*. Hàm chi phí chỉ được định nghĩa khi *s'* là trạng thái con của *s*.



Tìm kiếm bằng cây tìm kiếm

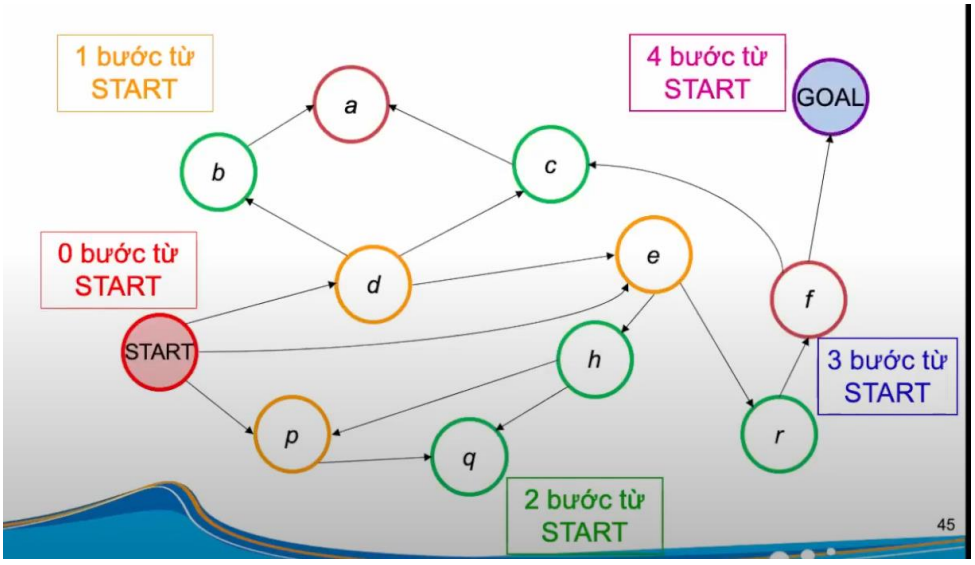
Thực hiện:

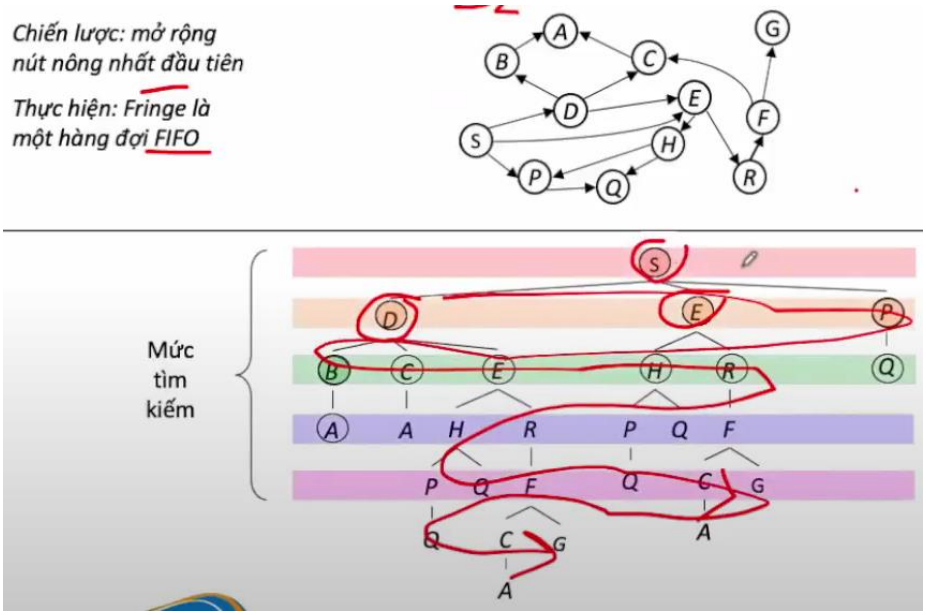
- Mở rộng ra các nút tiềm năng
- Duy trì một bộ nhớ fringe
- Cố gắng mở rộng càng ít nút càng tốt



Tìm kiếm chiều rộng (Breadth-First-Search – BFS)

- Gán nhãn mọi trạng thái có thể đến được từ *S* trong 1 bước nhưng không thể đến được trong ít hơn 1 bước.
- Tiếp đó gán nhãn mọi trạng thái đến được từ S trong 2 bước nhưng không đến được trong ít hơn 2 bước.
- Tiếp đó gán nhãn cho mọi trạng thái đến được từ S trong 3 bước nhưng không đến được trong ít hơn 3 bước
- …cứ thế cho đến khi đi đến trạng thái Goal

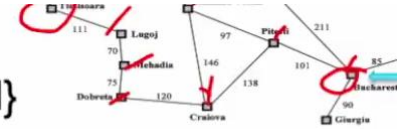




BFS - Một cách khác: Đi lui

- Gán nhãn mọi trạng thái có thể đi đến *G* trong 1 bước nhưng không thể đi đến trong ít hơn 1 bước.
- Gán nhãn mọi trạng thái có thể đi đến *G* trong 2 bước nhưng không thể đi đến trong ít hơn 2 bước.
- … cho đến khi đi đến trạng thái START
- Các nhãn "số bước đến đích" xác định đường đi ngắn nhất. Không cần thêm thông tin lưu trữ.

Bài tập ví dụ 1: *Các thành phố*



- ☐ V_0 = {Arad}
- ☐ V_1 = {Sibiu, Timisoara, Zerind}
- ☐ V_2 = {Faragas, Rimnicu Vilcea, Lugoj, Oradea}
- ☐ V_3 = {**Bucharest**, Pitesti, Craiova, Mehadia, ~~Sibiu~~}

Đã tìm thấy Goal, dừng thuật toán.

Đường đi là: Arad → Sibiu → Faragas → Bucharest

Chi phí: 450

* Sibiu đã có trong V_1 thì không đưa vào V_3 nữa.

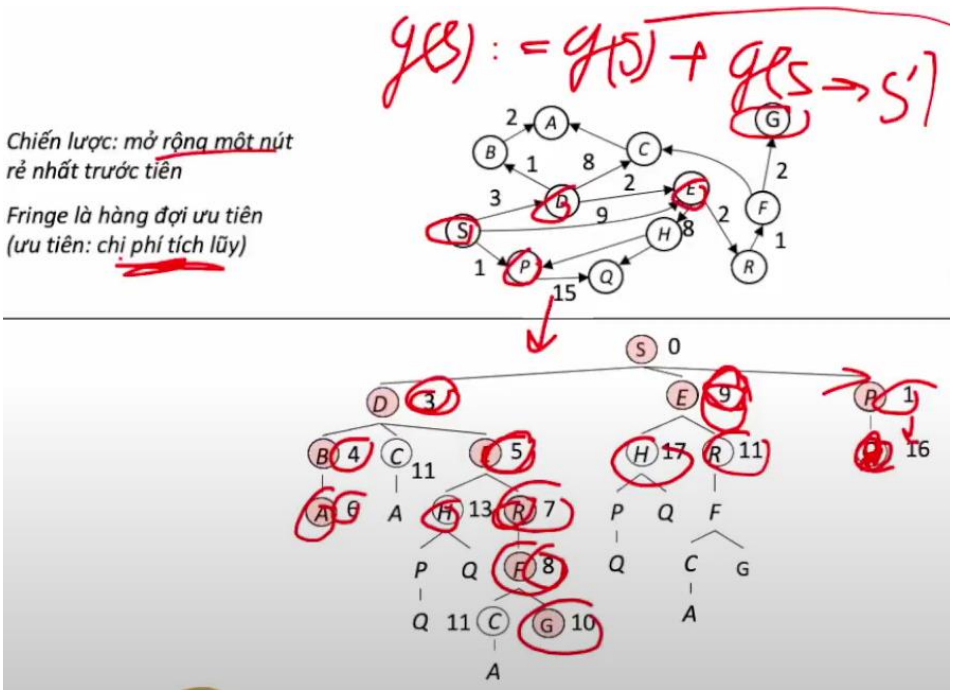
Tìm kiếm chi phí đồng nhất (Uniform-Cost-Search – UCS)

PQ = Tập các trạng thái đã được mở hay đang đợi mở.

Độ ưu tiên của trạng thái s = g(s) = chi phí đến S dùng đường đi cho bởi con trỏ quay lui. = g(s) + g(s->s') (bằng chi phí từ start đi qua nút cha tới nút đó)

Dừng chỉ khi trạng thái đích được lấy ra khỏi hang đợi ưu tiên. Nếu không, ta sẽ bỏ lỡ một đường đi khác ngắn hơn.

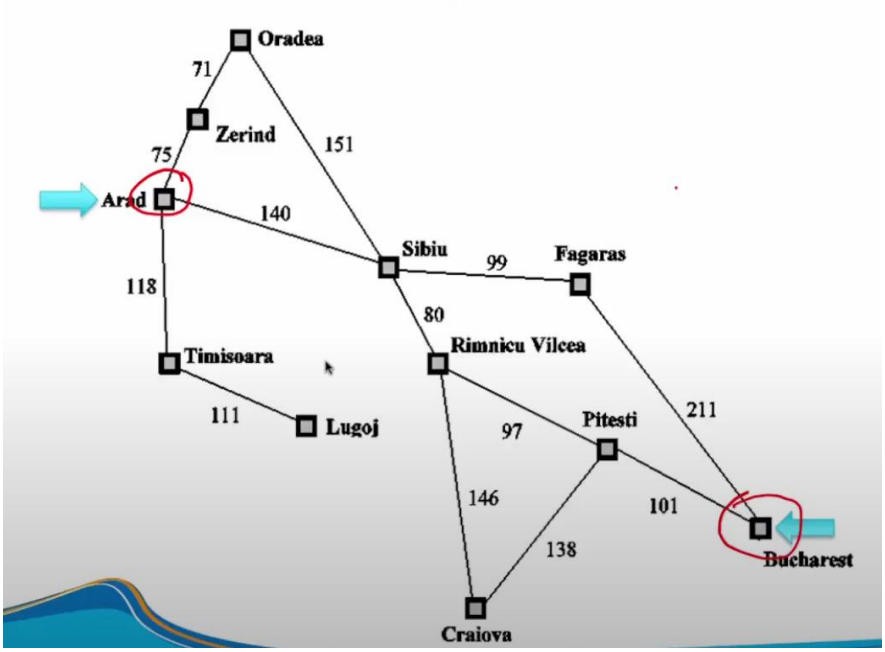
<u>Bài tập ví dụ 1:</u>



Sử dụng 2 stack: Open và Close

Thứ tự các đỉnh mở: S(0) -> R(1) -> D(3) -> B(4) -> E(5) -> A(6) -> R(7) -> F(8) -> E(9) -> G(10)

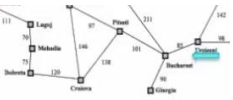
<u>Bài tập ví dụ 2:</u>



- V_0 = {(Arad, 0)}
- V_1 = {(Sibiu, 140), (Timisoara, 118), (Zerind, 75)}
- V_2 = {(Faragas, 239), (Rimnicu Vilcea, 220), (Lugoj, 229), (Oradea, 146)}
- V_3 = {(Bucharest, 450), (Pitesti, 317), (Craiova, 366), ~~(Sibiu, 297)~~}
- V_4 = {(Bucharest, 418), ~~(Pitesti, 504)~~}
- V_5 = { } rỗng, dừng thuật toán.

Đường đi là: Arad → Sibiu → Rimnicu Vilcea → Pitesti → Bucharest. Chi phí: 418

<u>Bài tập ví dụ 3</u>: Các thành phố



- PQ = {(Arad, 0)}
- PQ = {(Zerind, 75), (Timisoara, 118), (Sibiu, 140)}
- PQ = {(Timisoara, 118), (Sibiu, 140), (Oradea, 146)}
- PQ = {(Sibiu, 140), (Oradea, 146), (Lugoj, 229)}
- PQ = {(Oradea, 146), (Rimnicu Vilcea, 220), (Lugoj, 229), (Faragas, 239)}
- PQ = {(Rimnicu Vilcea, 220), (Lugoj, 229), (Faragas, 239)}
- PQ = {(Lugoj, 229), (Faragas, 239), (Pitesti, 317), (Craiova, 366)}
- PQ = {(Faragas, 239), (Mehadia, 299), (Pitesti, 317), (Craiova, 366)}
- PQ = {(Mahadia, 299), (Pitesti, 317), (Craiova, 366), (Bucharest, 450)}
- PQ = {(Pitesti, 317), (Craiova, 366), (Dobreta, 374), (Bucharest, 450)}
- PQ = {(Craiova, 366), (Dobreta, 374), (Bucharest, 418)}
- PQ = {(Dobreta, 374), (Bucharest, 418)}
- PQ = {(Bucharest, 418)}
- PQ = { }

Arad → Sibiu → Rimnucu Vilcea → Pitesti → Bucharest. Chi phí: 418

Tìm kiếm theo chiều sâu (Depth-First-Search – DFS)

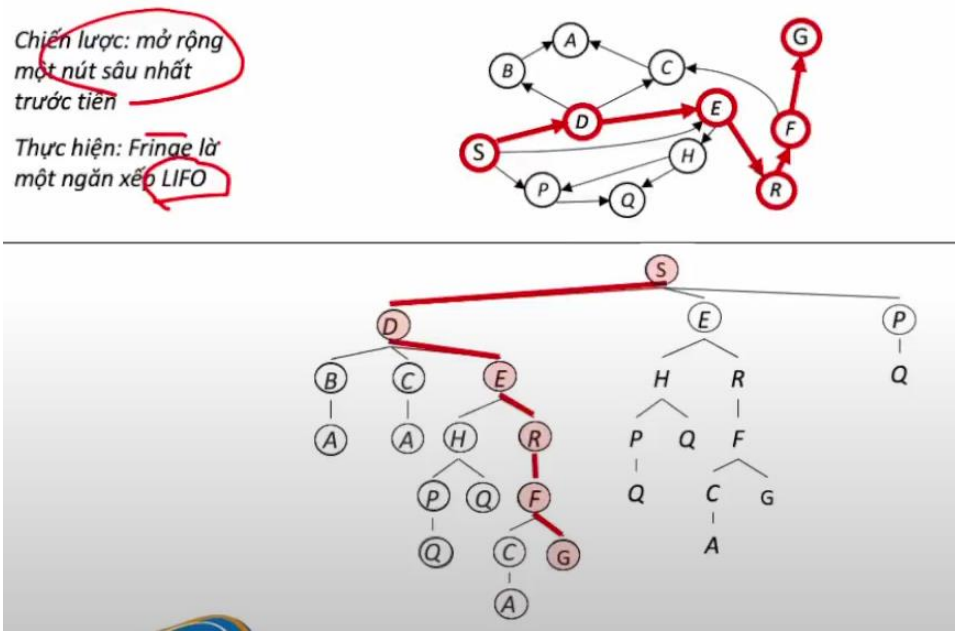
Một lựa chọn khác với BFS. Luôn mở từ nút vừa mới mở nhất, nếu nó có bất kì nút con chưa thử nào. Ngược lại, quay về nút trước đó trên đường đi hiện tại.

Sử dụng một cấu trúc dữ liệu, gọi là **Path**, để biểu diễn đường đi từ START đến trạng thái hiện tại.

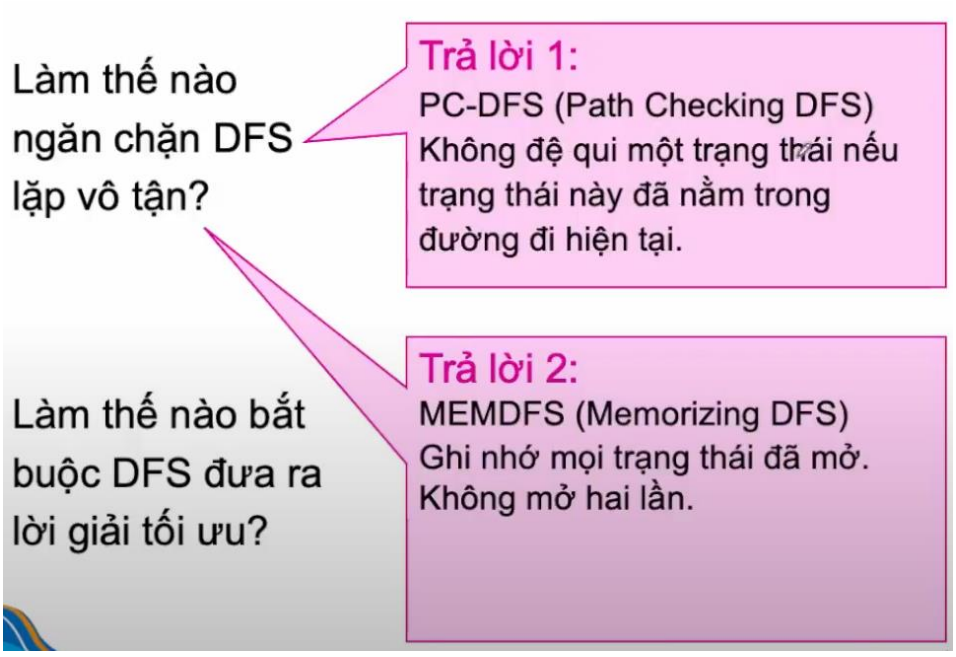
- Ví dụ: Path P = <START, d, e, r>

Với mỗi nút trên đường đi, cần ghi nhớ những nút con nào vẫn còn có thể mở. Ví dụ, tại điểm bên dưới, ta có:

Ví dụ minh họa 1:



Thứ tự các đỉnh mở lần lượt là: S -> D -> B -> A -> C -> A -> E -> H -> P -> Q -> Q -> R -> F -> C -> A -> G



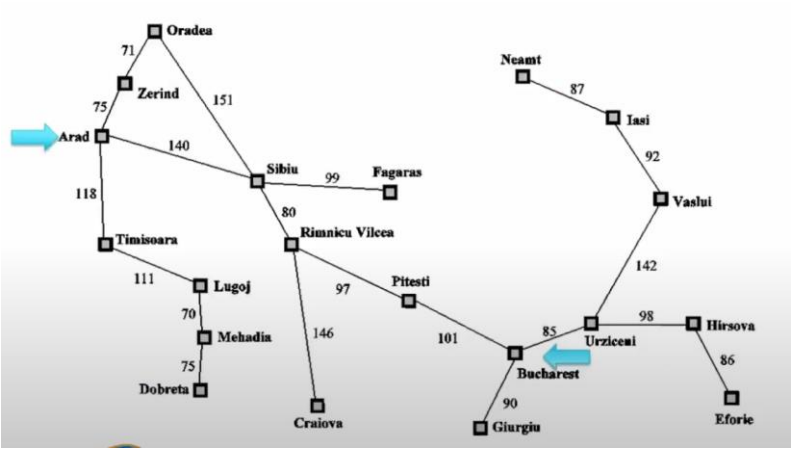
Tránh trường hợp cây có đường đi sâu vô tận bằng cách đặt ra giới hạn độ sâu *l*.

- Không đầy đủ nếu chọn $l < d$
- Có thể không tối ưu nếu chọn $l > d$.

Giới hạn độ sâu phụ thuộc vào tri thức nhận biết về bài toán.

- Ví dụ:
  - Bản đồ Romania có 20 thành phố $\Rightarrow l = 19$
  - Trong thực tế, một thành phố bất kì có thể đến trong vòng 9 bước $\Rightarrow l = 9$ tốt hơn

Ví dụ minh họa 2:



Kết quả thực hiện bằng PC-DFS, chọn đỉnh đi tiếp theo thứ tự bảng chữ cái.

STT	Đường đi	Danh sách mở của nút hiện tại
1	Arad	{Sibiu, Timisoara, Zerind}
2	Arad → Sibiu	{Faragas, Oradea, Riminicu Vilcea}
3	Arad → Sibiu → Faragas	{NULL}
4	Arad → Sibiu	{Oradea, Riminicu Vilcea}
5	Arad → Sibiu → Oradea	{Zerind}
6	Arad → Sibiu → Oradea → Zerind	{NULL} (Arad đang nằm trên đường đi}
7	Arad → Sibiu → Oradea	{NULL}