TRÍ TUỆ NHÂN TẠO

Bài 4: Tìm kiếm mù

Nội dung

1. Khái niệm tìm kiếm mù
2. Thuật toán
3. Các biến thể
    1. Tìm kiếm theo chiều rộng (BFS)
    2. Tìm kiếm theo chi phí đồng nhất (UCS)
    3. Tìm kiếm theo chiều sâu (DFS)
    4. Tìm kiếm giới hạn chiều sâu (DLS)
    5. Tìm kiếm sâu dần (IDS)
    6. Tìm kiếm hai chiều (BS)
4. Bài tập và câu hỏi

Phần 1

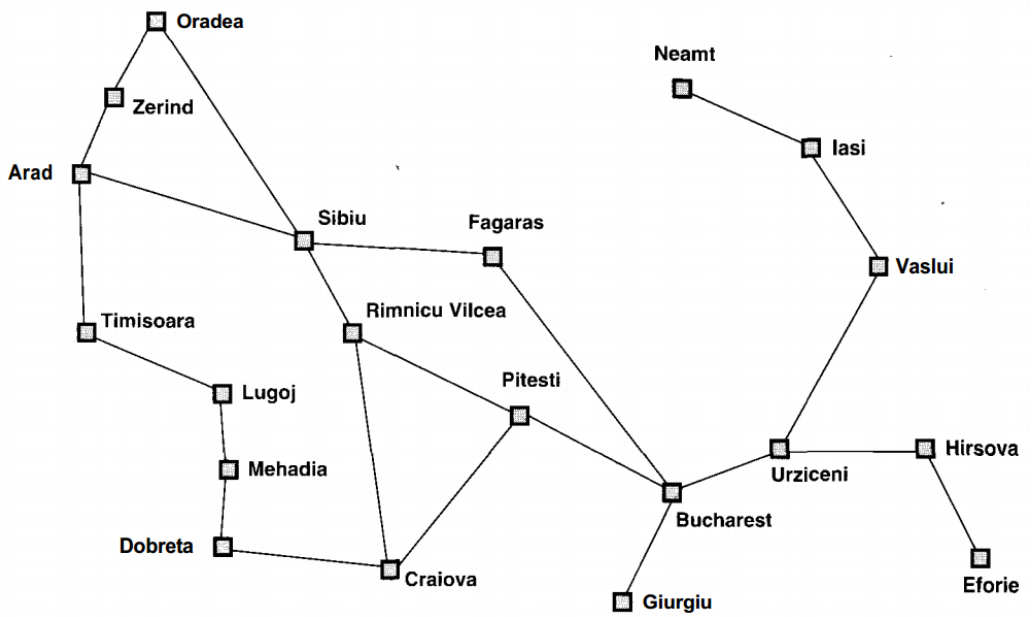
Khái niệm tìm kiếm mù

Nhắc lại quan điểm "AI là tìm kiếm"

- Hình trạng / Trạng thái (state)
- Bước chuyển (path/operator)
- Chi phí bước chuyển (path cost)
- Hình trạng đích (goal states - GS)
- Hình trạng xuất phát (start state - SS)
- Lời giải = Các bước chuyển từ SS đến GS
- Tìm lời giải = Tìm đường đi
  - Tìm càng nhanh thì càng thông minh?

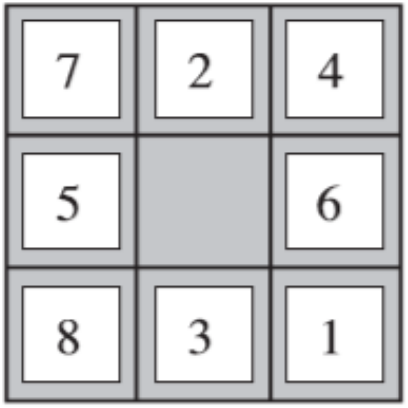
Bài toán tìm đường đi

- **Hình trạng** là gì?
- **Bước chuyển**?
- **Chi phí** bước chuyển?
- Hình trạng **đích**?
- Hình trạng **xuất phát**?
- Kích thước không gian?

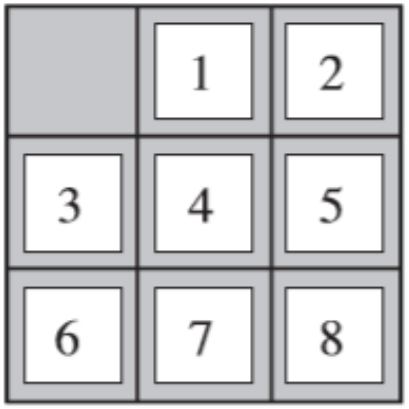


Bài toán 8-mảnh

- **Hình trạng** là gì?
- **Bước chuyển**?
- **Chi phí** bước chuyển?
- Hình trạng **đích**?
- Hình trạng **xuất phát**?
- Kích thước không gian?



Start State



Goal State

Bài toán "Nhóm người sang sông"

- Có 4 người A, B, C, D đang đứng ở bên bờ sông và muốn sang bên bờ kia
- Có 1 chiếc phao cho 2 người (1 người dùng vẫn được)
- Muốn qua sông nhất thiết phải dùng phao
- Thời gian qua sông của mỗi người là khác nhau. Nếu 2 người cùng dùng phao thì tính theo thời gian của người bơi chậm hơn
- A bơi qua sông mất 1 phút, B mất 2 phút, C mất 5 phút, D mất 10 phút
- Nhóm cần ít nhất bao nhiêu phút để qua sông?

Khái niệm tìm kiếm mù

- Xuất phát từ hình trạng ban đầu (START) và tìm các bước chuyển để đến (một) hình trạng đích (GOAL)
- Thông tin duy nhất là chi phí của từng bước chuyển, **không có thông tin bổ sung**
  - Chính vì không có thông tin bổ sung, nên ta không có định hướng cho việc tìm kiếm, dẫn đến hệ quả là ta tìm không theo trật tự nào cả (như người mù)
- Bản chất: Xuất phát từ START, lần lượt **DUYỆT** qua các hình trạng liên quan cho đến khi gặp GOAL

Phần 2

Thuật toán

Thuật toán

```
function SEARCH(START) return solution/failure {
  S = {START}
  loop {
     if S is EMPTY then return failure
     node = REMOVE-ONE(S)
     if node is GOAL then return SOLUTION(node)
     S = S + EXPAND(node)
  }
}
```

S: ***Tập các hình trạng đang được xem xét***

REMOVE-ONE(S): ***Lấy một phần tử ra khỏi tập S***

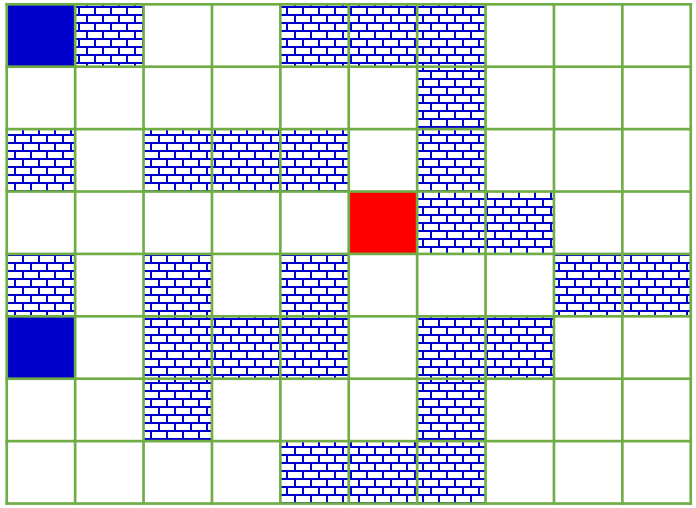
EXPAND(node): ***Tập hình trạng liên quan đến node***

Các vấn đề cần quan tâm

- Cách hoạt động của hàm REMOVE-ONE
- Cách thực hiện của hàm EXPAND
- Cấu trúc dữ liệu của S
- Lưu trữ thông tin như thế nào để có thể dò lại đường đi từ START đến GOAL
  - Làm thế nào trả về SOLUTION phù hợp?
- Đánh giá các kết quả tìm kiếm được
  - Độ phức tạp thời gian
  - Độ phức tạp không gian
  - Thuật toán có tìm được kết quả (nếu có) hay không?
  - Thuật toán có tìm ra được kết quả tốt ưu (tốt) hay không?

Ví dụ

- Di chuyển đến được các ô chung cạnh, không đi vào các ô là "tường" (có đánh dấu màu xanh)
- Yêu cầu: đi từ ô xanh đến ô Đỏ
- Thứ tự bổ sung vào S: Trên – Dưới – Trái – Phải
- Xét 2 trường hợp S dùng Stack và Queue
- Hãy chỉ ra thứ tự các ô nằm trong S



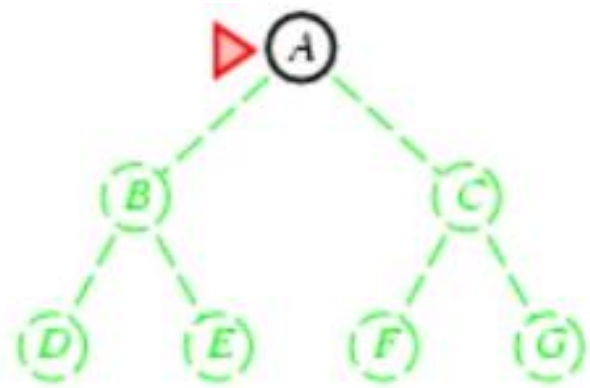
Phần 3

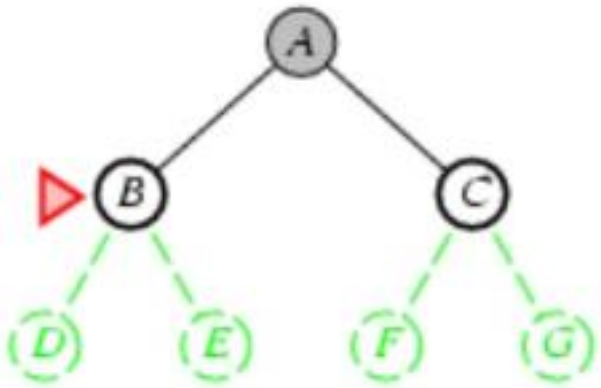
Các biến thể

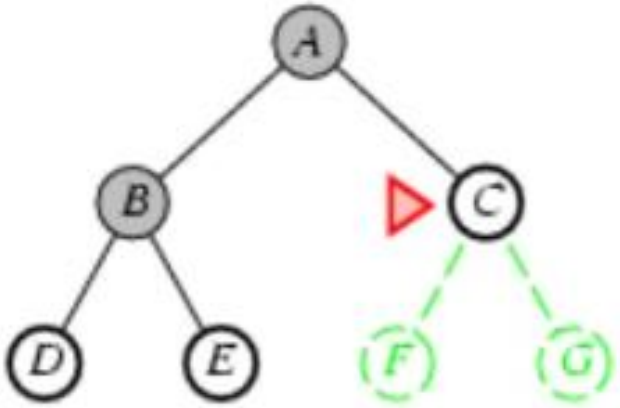
3.1 Tìm kiếm theo chiều rộng (BFS)

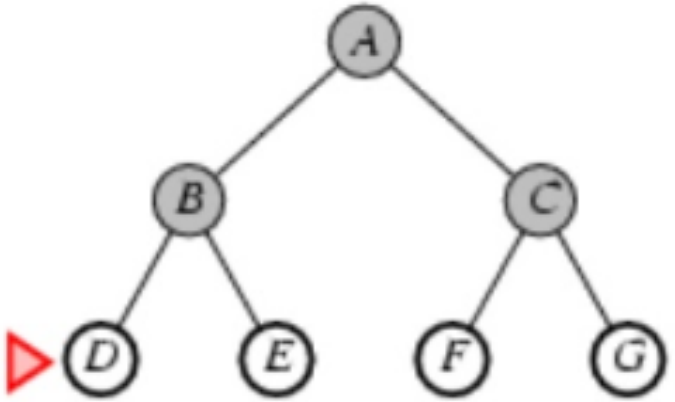
- Tên tiếng Anh: Breadth-First Search
- S sử dụng cấu trúc lưu trữ kiểu QUEUE
- Hàm REMOVE-ONE lấy phần tử ở đầu QUEUE
- Hàm EXPAND đẩy các phần tử mới vào cuối QUEUE
- Đặc trưng:
  - Bùng nổ về số hình trạng nằm trong QUEUE
  - Tốt cho các bài toán chi phí đều
  - Là tiền đề cơ bản cho các thuật toán hiệu quả

3.1 Tìm kiếm theo chiều rộng (BFS)



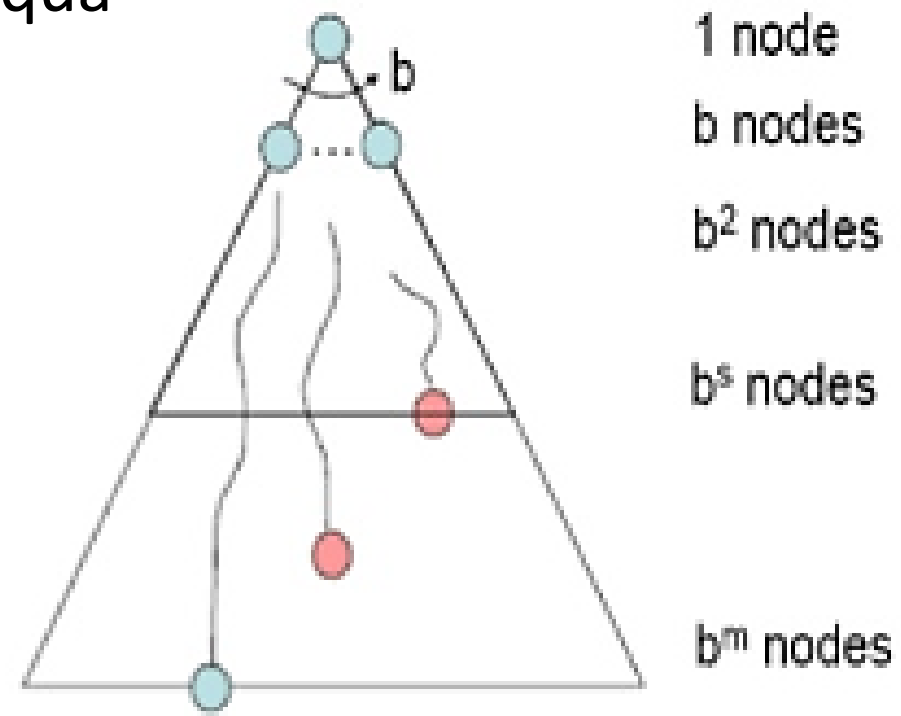






3.1 Tìm kiếm theo chiều rộng (BFS)

- Độ phức tạp thời gian: $1 + b + b^2 + \ldots + b^d \sim O(b^{d+1})$
- Độ phức tạp không gian: lưu trữ mọi node ~ $O(b^{d+1})$
- Thuật toán có tìm được kết quả (nếu có) hay không? Có
- Thuật toán có tìm được kết quả tốt ưu hay không? Có



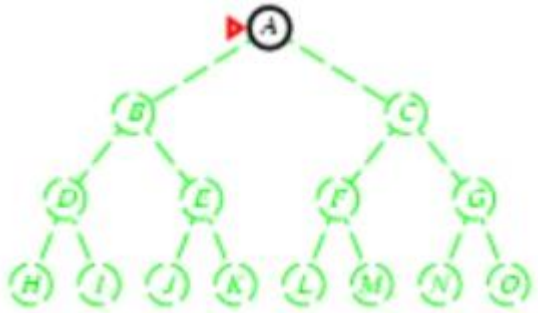
3.2 Tìm kiếm theo chi phí đồng nhất (UCS)

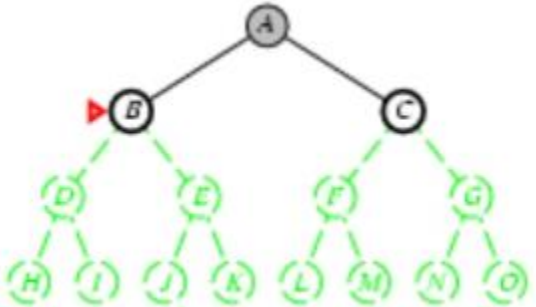
- Tên tiếng Anh: Uniform Cost Search (nhiều sách dịch là tìm kiếm theo chi phí tối thiểu hoặc chi phí đều)
- UCS là biến thể của BFS: thay vì chọn hình trạng bất kỳ để phát triển, UCS chọn hình trạng có chi phí thấp nhất
  - Nếu chi phí cho mỗi bước chuyển đều bằng nhau thì UCS ~ BFS
- Độ phức tạp thời gian: phụ thuộc vào số lượng hình trạng có chi phí thấp hơn chi phí tối thiểu $C^* \sim O(b^{[C^*/\varepsilon]})$
- Độ phức tạp không gian: phụ thuộc vào các hình trạng có chi phí thấp hơn chi phí tối thiểu $C^* \sim O(b^{[C^*/\varepsilon]})$
- Thuật toán có tìm được kết quả (nếu có) hay không? Có*
- Thuật toán có tìm được kết quả tốt ưu hay không? Có*
- Cẩn thận nếu chi phí bước chuyển tối thiểu ε là số âm

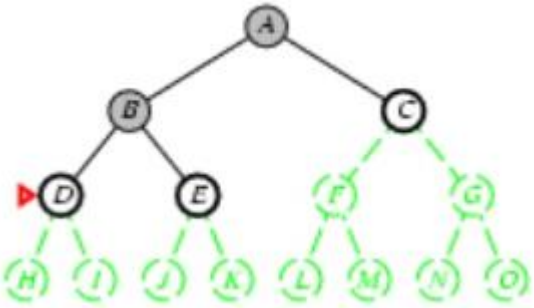
3.3 Tìm kiếm theo chiều sâu (DFS)

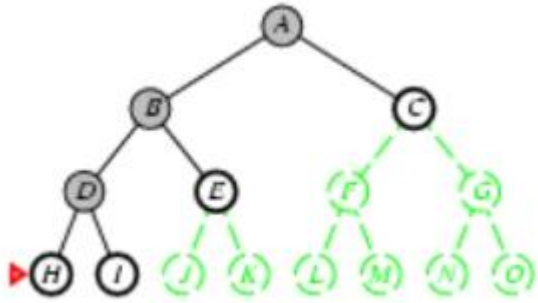
- Tên tiếng Anh: Depth-First Search
- S sử dụng cấu trúc lưu trữ kiểu STACK
- Hàm REMOVE-ONE lấy phần tử ở cuối STACK
- Hàm EXPAND đẩy các phần tử mới vào cuối STACK
- Đặc trưng:
  - Rủi ro về thời gian tìm kiếm (nhầm đường)
  - Ít gặp vấn đề về chi phí bộ nhớ
  - Phù hợp với không gian tìm kiếm dạng đồ thị thưa (số bước chuyển từ một hình trạng thường không nhiều)

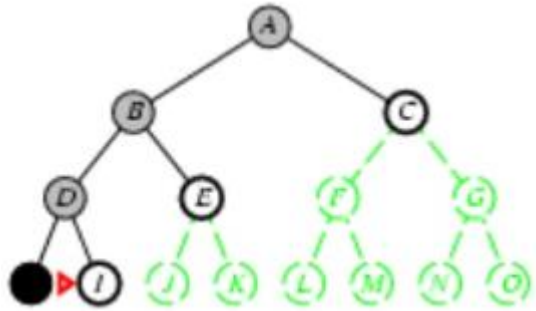
3.3 Tìm kiếm theo chiều sâu (DFS)

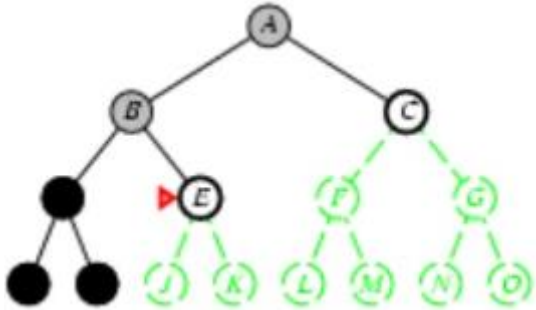












3.3 Tìm kiếm theo chiều sâu (DFS)

- Độ phức tạp thời gian: $O(b^m)$
- Độ phức tạp không gian: lưu trữ node theo đường đi chiều sâu ~ O(mb)
- Thuật toán có tìm được kết quả (nếu có) hay không? Không hoàn toàn
  - Nếu không gian trạng thái có độ sâu vô hạn hoặc chứa các chu trình (trạng thái lặp lại)
- Thuật toán có tìm được kết quả tốt ưu hay không? Không

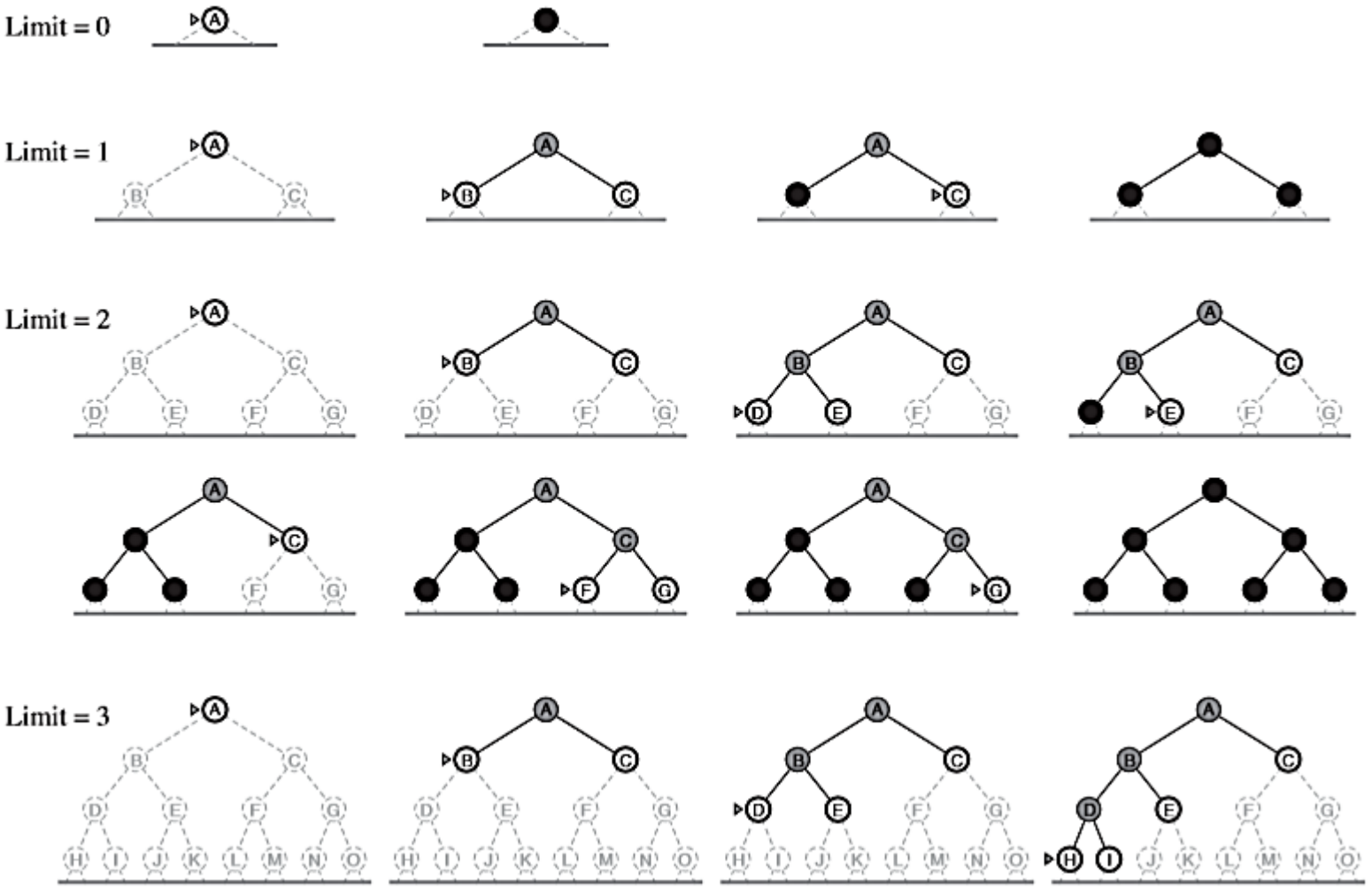
3.4 Tìm kiếm giới hạn chiều sâu (DLS)

- Tên tiếng Anh: Depth-Limited Search
- Là một biến thể của tìm kiếm theo chiều sâu
- Giới hạn chiều sâu xác định, không đi quá sâu
  - Chẳng hạn giới hạn không tìm quá độ sâu 10
  - Hoặc giới hạn chi phí tìm kiếm không quá 15000
- Vấn đề:
  - Lý do nào để chọn các giới hạn này?
  - Không tồn tại nghiệm trong các giới hạn đó thì sao?

3.5 Tìm kiếm sâu dần (IDS)

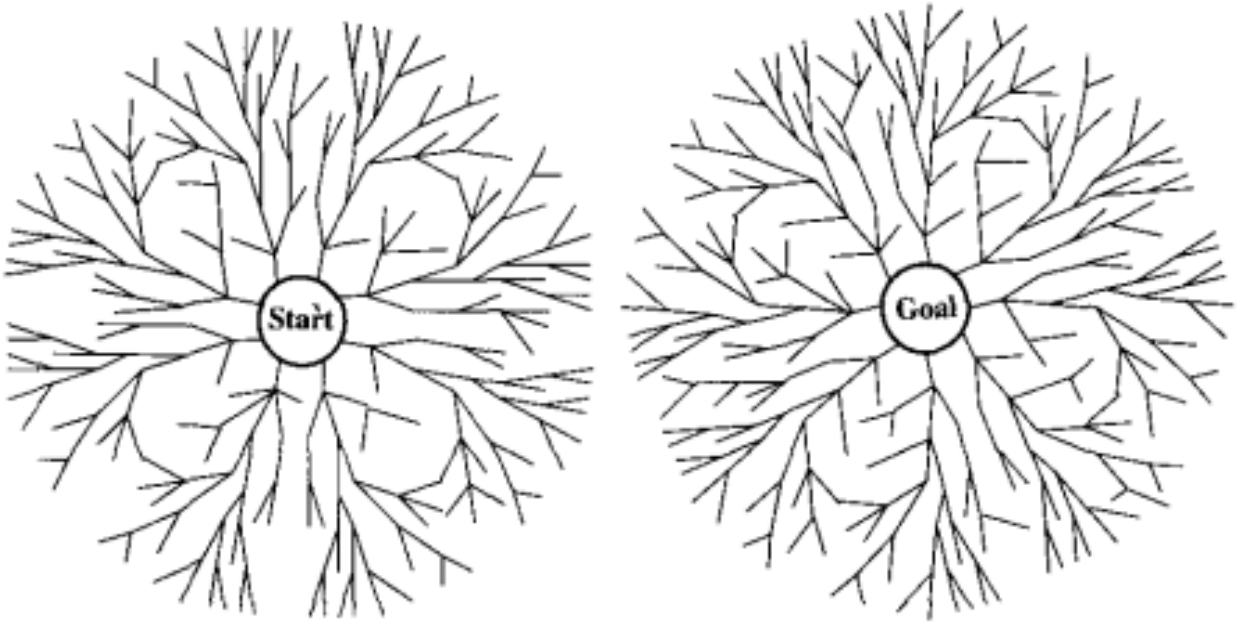
- Tên tiếng Anh: Iterative-Deepening Search
- Là một biến thể khác của tìm kiếm theo chiều sâu và tìm kiếm giới hạn chiều sâu
- Gồm 2 bước cơ bản:
  1. Tìm kiếm giới hạn chiều sâu
  2. Nếu tìm không tìm được kết quả thì tăng chiều sâu và tìm lại
- Đặc trưng:
  - Giới hạn tốt cho các bài toán tối ưu
  - Rủi ro nếu phải lặp lại quá trình tìm kiếm
  - Không phù hợp với những chi phí âm

3.5 Tìm kiếm sâu dần (IDS)



3.6 Tìm kiếm hai chiều (BS)

- Tên tiếng Anh: Bidirectional Search
- Tìm kiếm đồng thời từ 2 phía: từ điểm xuất phát đi và từ điểm dừng ngược lại
- Ưu điểm: hạn chế được sự bùng nổ tổ hợp theo chiều sâu (vì chiều sâu trung bình giảm còn một nửa)



Phần 4

Bài tập và câu hỏi

Câu hỏi

Bài toán 4 màu: bản đồ các quốc gia (vẽ trên mặt phẳng) luôn có thể tô mà không quá 4 màu

- Tô màu riêng từng quốc gia, mỗi quốc gia chỉ một màu
- Không có hai quốc gia chung đường biên giới có cùng màu

Giả sử cần viết một chương trình tìm ra cách tô màu, hãy:

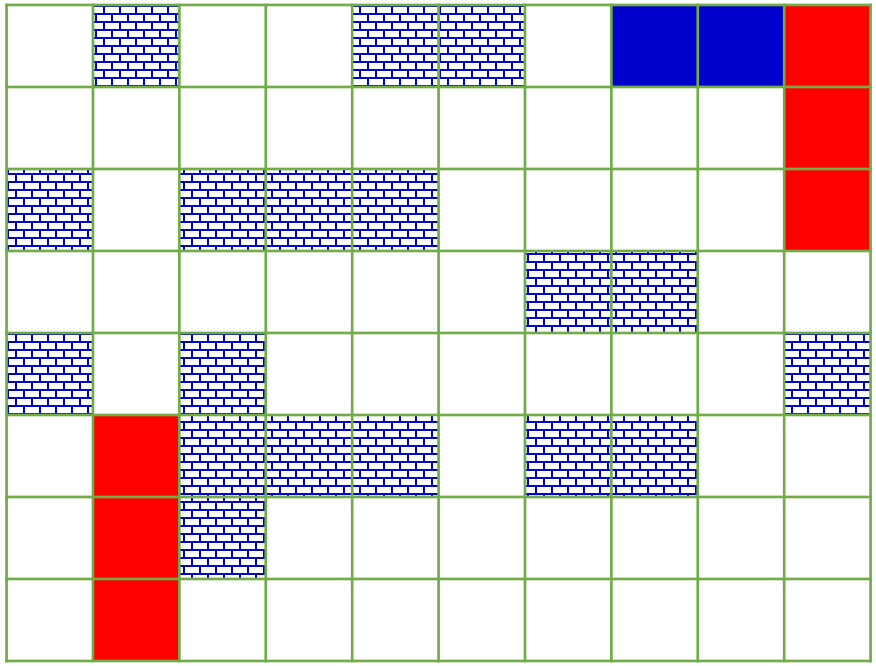
- Khái niệm hình trạng trong bài toán này
- Không gian trạng thái của bài toán
- Các trạng thái ban đầu
- Các trạng thái kết thúc

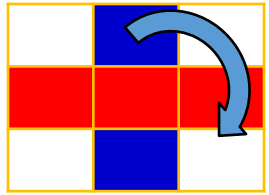
Có thể áp dụng những thuật toán nào vừa được đề cập để viết chương trình?

Bài toán dịch chuyển đồ vật

Bài toán di chuyển đồ vật trong căn phòng: một thiết bị di chuyển đồ vật trong phòng bằng cách đẩy nó theo 4 hướng (**lên**, **xuống**, **trái**, **phải**) hoặc **xoay** đồ vật 90°.

Xây dựng một thuật toán chỉ ra phương án dịch chuyển đồ vật từ vị trí ban đầu đến một vị trí cho trước.





Lĩnh vực áp dụng: lập trình di chuyển của robot