ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

Cơ sở trí tuệ nhân tạo

Bài toán tìm kiếm

Nguyễn Ngọc Đức

2024

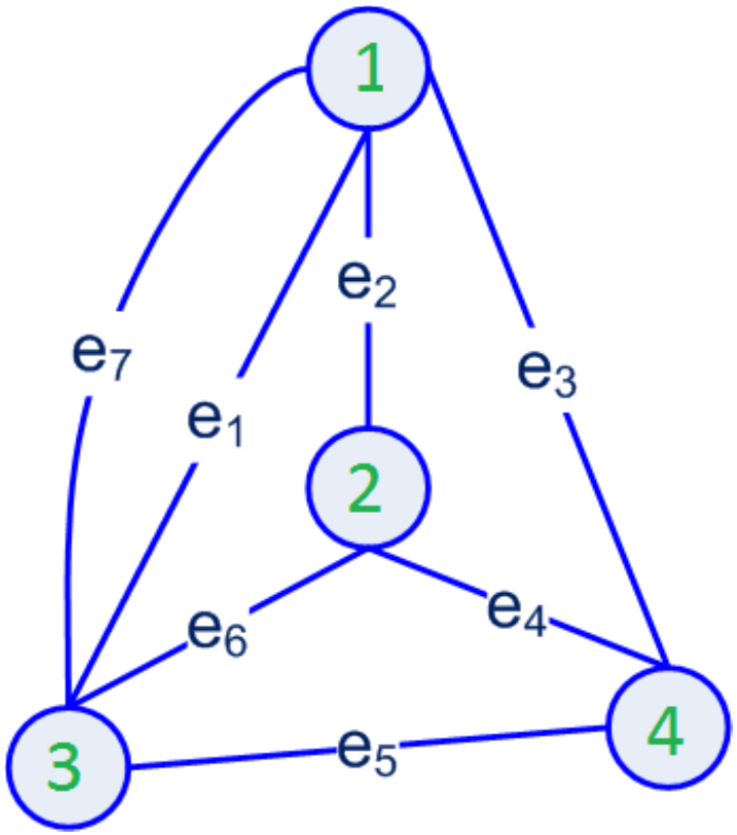
Nội dung

Một số khái niệm

Đồ thị vô hướng

Định nghĩa. Một đồ thị vô hướng $G = (V, E)$ được định nghĩa bởi

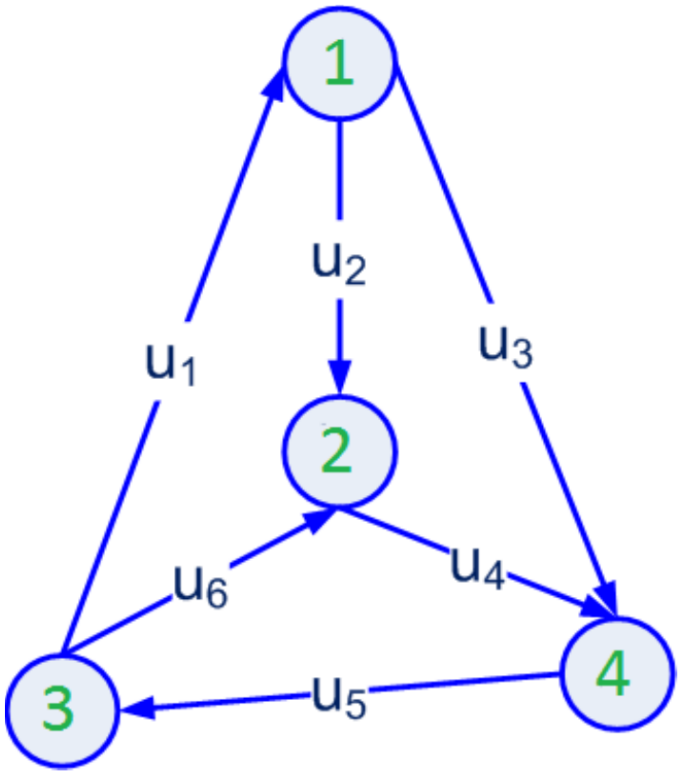
- Tập hợp $V \neq \varnothing$ được gọi là tập đỉnh và $n = |V|$ gọi là cấp của đồ thị;
- Tập hợp E là tập các cạnh của đồ thị; Mỗi cạnh $e \in E$ được liên kết với một cặp đỉnh $\{i, j\}$, không phân biệt thứ tự.



Đồ thị có hướng

Định nghĩa. Một đồ thị có hướng $G = (V, U)$ được định nghĩa bởi

- Tập hợp $V \neq \varnothing$ được gọi là tập đỉnh và $n = |V|$ gọi là cấp của đồ thị;
- Tập hợp U là tập các cạnh của đồ thị; Mỗi cạnh $u \in U$ được liên kết với một cặp đỉnh $(i, j) \in V^2$ ký hiệu $u = (i, j)$ hoặc $u = ij$.



Đường đi

Định nghĩa. Cho $G = (V, E)$ là đồ thị có hướng và hai đỉnh u và v. Khi đó:

- **Đường đi** từ u đến v: **chuỗi cạnh** (có hướng) từ đỉnh u đến đỉnh v
- Đường đi không có cạnh nào xuất hiện quá một lần gọi là **đường đi đơn**
- Đường đi không có đỉnh nào xuất hiện quá một lần gọi là **đường đi sơ cấp**
- Đường đi được gọi là **chu trình** nếu nó bắt đầu và kết thúc tại cùng một đỉnh

Liên thông

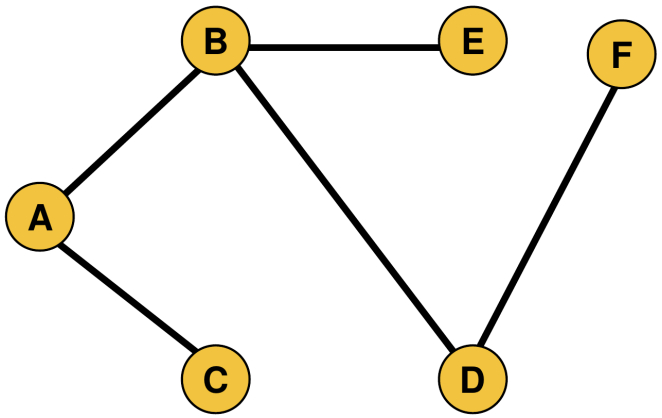
Định nghĩa. Cho $G = (V, E)$ là đồ thị vô hướng trên V ta định nghĩa quan hệ tương đương như sau:

$$u \sim v \Leftrightarrow u = v \text{ hay có một đường đi từ u đến v}$$

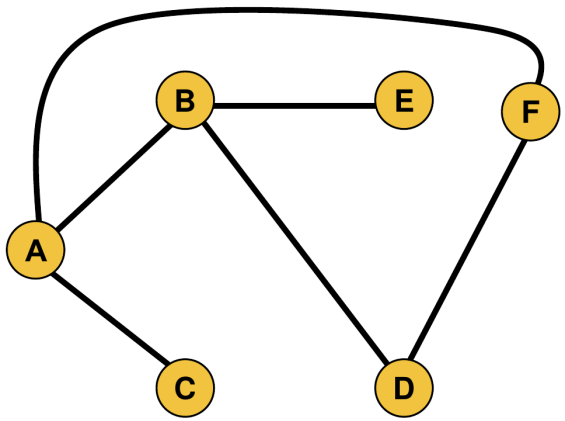
- Nếu $u \sim v$ thì ta nói 2 đỉnh u và v **liên thông** với nhau
- Mỗi lớp tương đương được gọi là **một thành phần liên thông** của G
- Nếu G chỉ có một thành phần liên thông thì G gọi là **liên thông**

Cây

Định nghĩa. Cây là một đồ thị **vô hướng, liên thông** và không có **chu trình**.



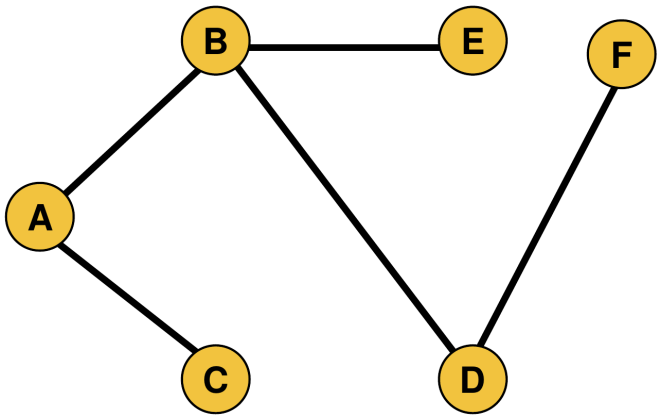
Hình 1: G_1



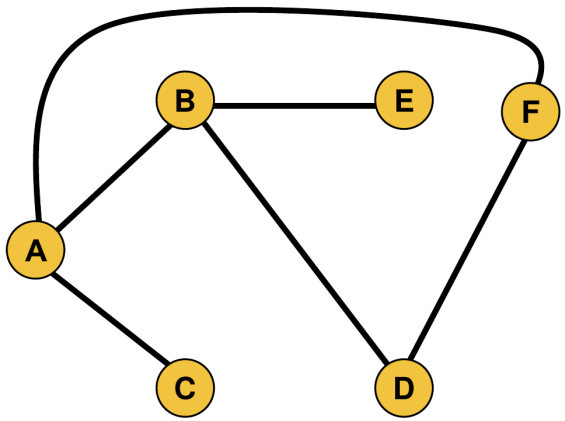
Hình 2: G_2

Cây

Định nghĩa. Cây là một đồ thị **vô hướng, liên thông** và không có **chu trình**.



Hình 1: G_1



Hình 2: G_2

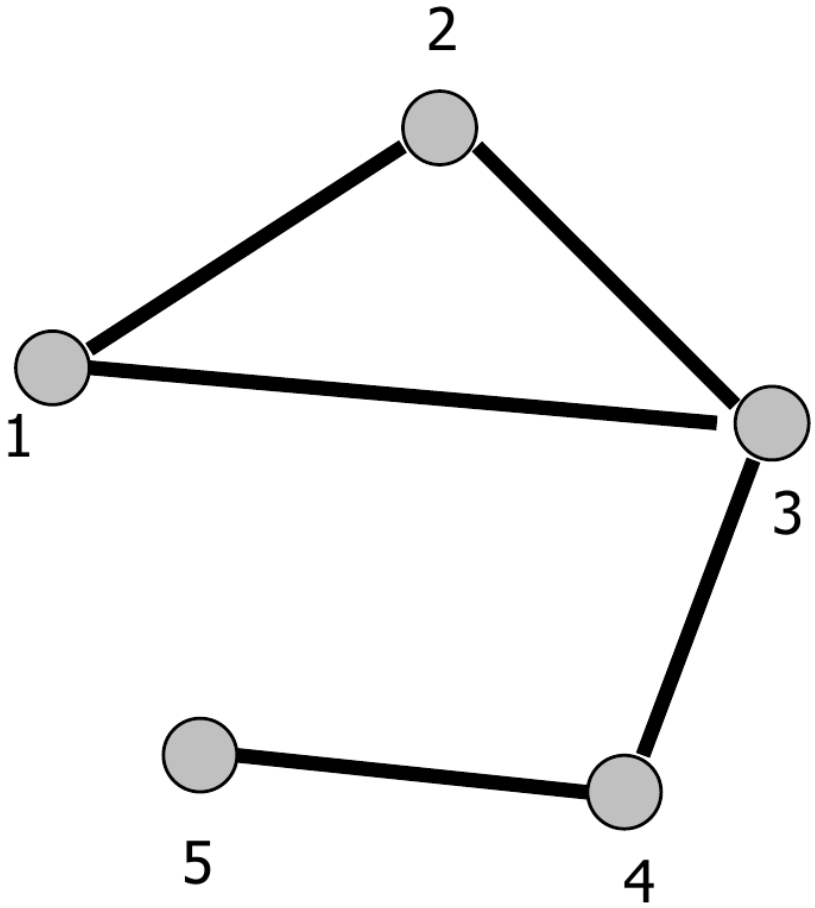
G_1 là cây

Biểu diễn đồ thị

Ma trận kề

- Ma trận đối xứng cho đồ thị vô hướng

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$



Biểu diễn đồ thị

Danh sách kề

- Lưu danh sách các nút lân cận của mỗi nút

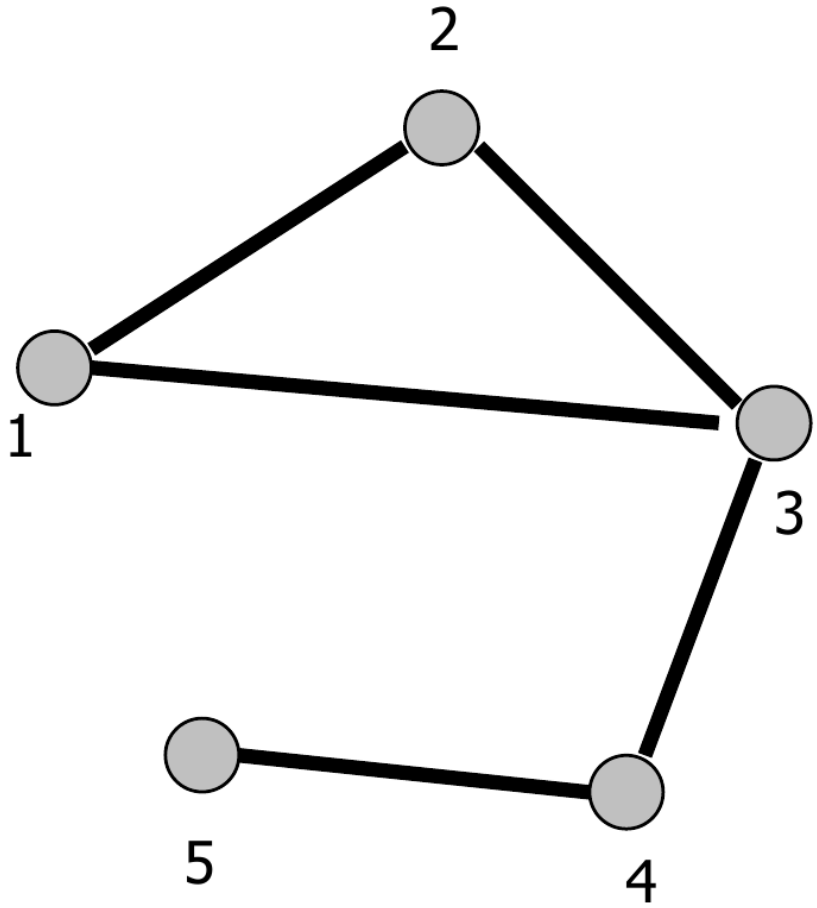
$$1 : [2, 3]$$

$$2 : [1, 3]$$

$$3 : [1, 2, 4]$$

$$4 : [3, 5]$$

$$5 : [4]$$



Biểu diễn đồ thị

Danh sách cạnh

- Lưu danh sách tất cả các cạnh (có hướng) trong đồ thị

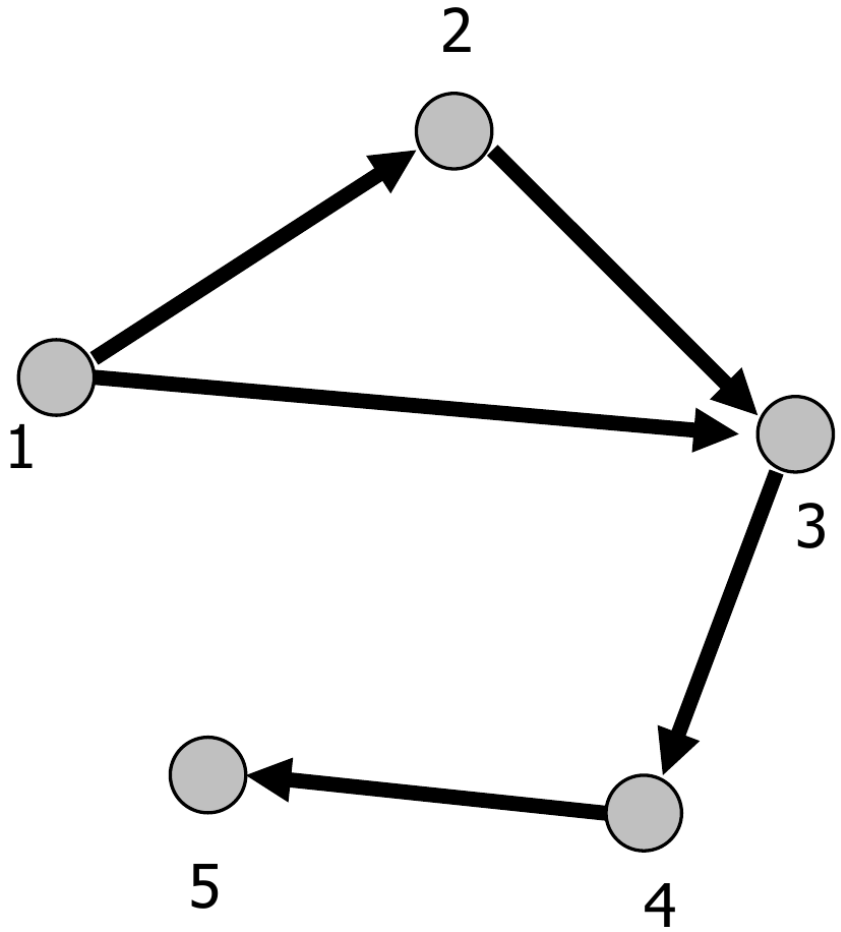
$$(1, 2)$$

$$(2, 3)$$

$$(1, 3)$$

$$(3, 4)$$

$$(4, 5)$$



Biểu diễn đồ thị

Ví dụ. Vẽ đồ thị có ma trận kề sau:

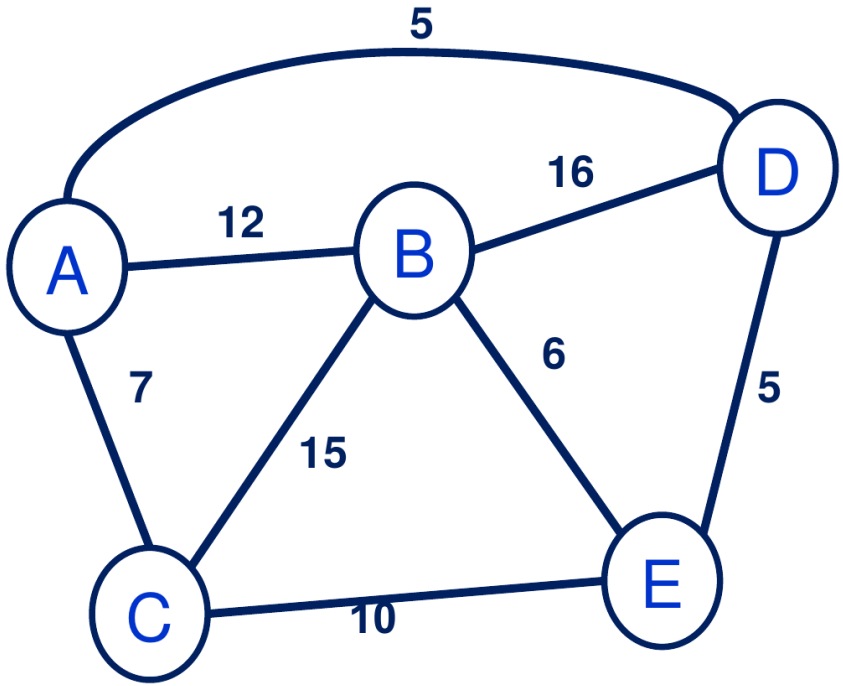
$$\begin{bmatrix} 0 & 12 & 7 & 5 & 0 \\ 12 & 0 & 15 & 16 & 6 \\ 7 & 15 & 0 & 0 & 10 \\ 5 & 16 & 0 & 0 & 5 \\ 0 & 6 & 10 & 5 & 0 \end{bmatrix}$$

Biểu diễn đồ thị

Ví dụ. Vẽ đồ thị có ma trận kề sau:

$$\begin{bmatrix} 0 & 12 & 7 & 5 & 0 \\ 12 & 0 & 15 & 16 & 6 \\ 7 & 15 & 0 & 0 & 10 \\ 5 & 16 & 0 & 0 & 5 \\ 0 & 6 & 10 & 5 & 0 \end{bmatrix}$$

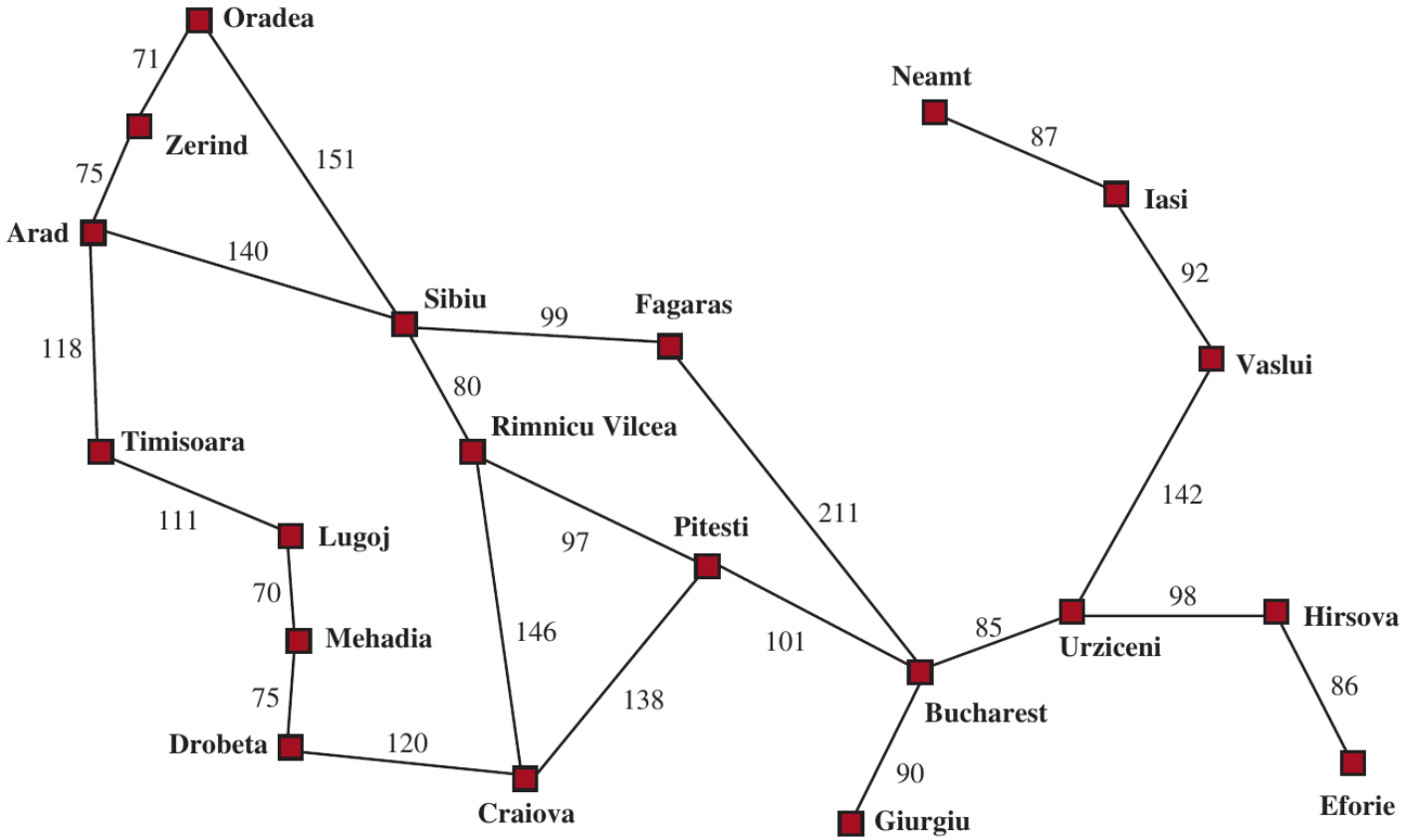
Đáp án.



Bài toán tìm kiếm

Ví dụ: Chuyến du lịch ở Romania

- Bạn đang ở Arad
- Ngày mai bạn cần phải dự đám cưới ở Bucharest



Định nghĩa bài toán tìm kiếm I

Một **bài toán tìm kiếm** có thể được định nghĩa bằng **5** thành phần:

1. **Trạng thái bắt đầu(Initial state)**
   - Ví dụ: trạng thái bắt đầu có thể được mô tả $In(Arad)$
2. Mô tả các **hành động (action)** có thể thực hiện
   - Ví dụ: $Action(In(Arad)) = \{Go(Sibiu), Go(Timisoara), Go(Zerind)\}$
3. **Mô hình di chuyển (transition model):** mô tả kết quả của các hành động
   - Ví dụ: $Result(In(Arad)), Go(Zerind) \rightarrow In(Zerind)$
   - Thuật ngữ **successor** tương ứng với các trạng thái có thể di chuyển được với một hành động duy nhất

Định nghĩa bài toán tìm kiếm II

- Trạng thái bắt đầu, hành động và mô hình di chuyển định nghĩa **không gian trạng thái (state space)** của bài toán
- Không gian trạng thái hình thành nên một **đồ thị** có hướng với đỉnh là các trạng thái và cạnh là các hành động
- Một **đường đi** trong không gian trạng thái là một chuỗi các trạng thái được kết nối bằng một chuỗi các hành động

4. **Kiểm tra đích (goal test):** xác định một trạng thái có là trạng thái đích
   - Ví dụ: $In(Bucharest)$

Định nghĩa bài toán tìm kiếm III

5. Một hàm **chi phí đường đi (path cost)** gán chi phí với giá trị số cho mỗi đường đi
   - **Chi phí đường đi** khi thực hiện hành động a từ trạng thái s để đến trạng thái s' ký hiệu $c(s, a, s')$

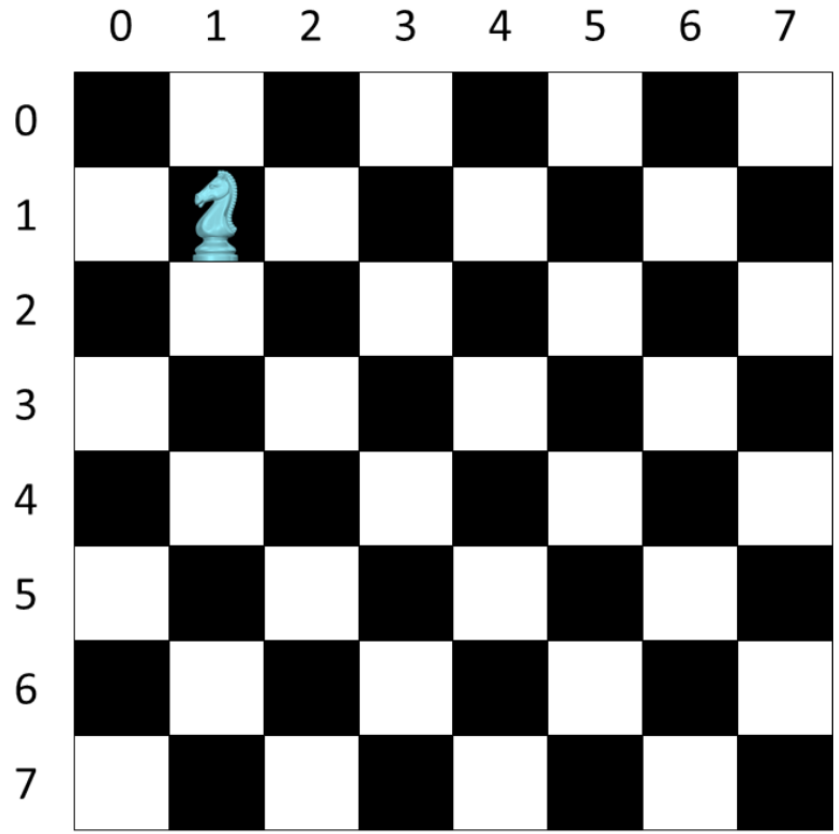
Một lời giải (solution) là một **chuỗi hành động** di chuyển từ trạng thái bắt đầu cho đến trạng thái đích

- Một **lời giải tối ưu** có chi phí đường đi thấp nhất trong số tất cả lời giải

Bài toán mã đi tuần

Phát biểu bài toán

Xác định các bước di chuyển để quân mã có thể đi hết bàn cờ mà không lặp lại bất kỳ ô nào



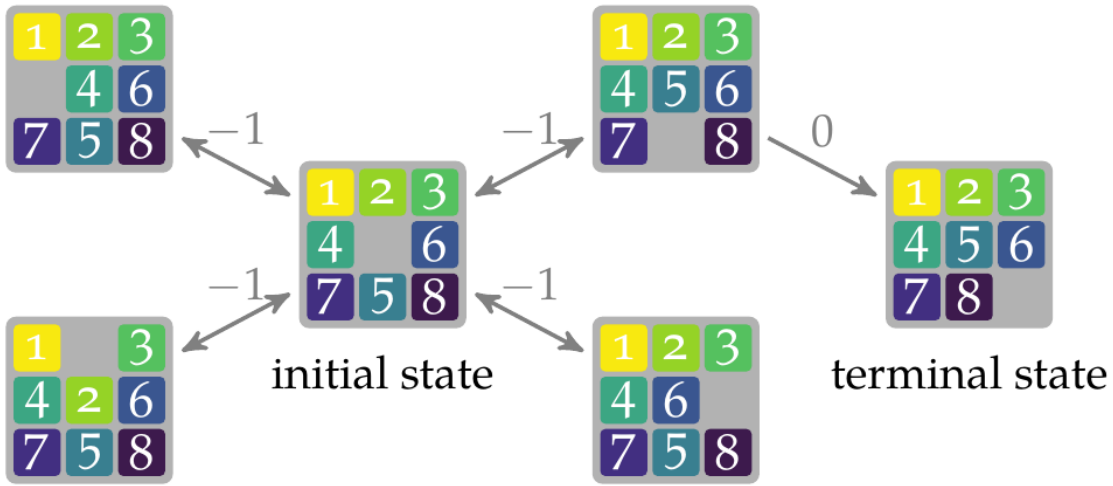
Bài toán mã đi tuần

- **Không gian trạng thái** là **đồ thị** với đỉnh là các trạng thái của bàn cờ (vị trí của quân mã và các ô đã di chuyển)
- Giữa 2 đỉnh có **cạnh liên kết** nếu như 2 trạng thái tương ứng của bàn cờ có thể chuyển đổi đến nhau bằng cách thực hiện **hành động** di chuyển quân mã

Bài toán 8-puzzle

Phát biểu bài toán

Trò chơi 8-puzzle là một hình vuông gồm 3x3 ô. Trong đó có 8 ô có số và 1 ô trống. Mỗi ô cạnh ô trống có thể di chuyển sang thế ô trống. Yêu cầu là làm sao di chuyển các ô để chuyển hình vuông từ trạng thái bắt đầu (initial state) đến trạng thái đích (terminal state)



Tìm kiếm lời giải

Giải quyết bài toán bằng phương pháp tìm kiếm I

- **Tìm kiếm:** quy trình tìm chuỗi hành động để tới được trạng thái đích
- Một thuật toán tìm kiếm nhận một bài toán là **input** trả về kết quả là **lời giải** dưới dạng chuỗi hành động.
- **Thực thi**: khi đã tìm thấy lời giải, thực hiện các hành động
  - Khi thực hiện lời giải, không nhận tín hiệu trả về từ các hành động
  - Hệ thống **open-loop**

Giải quyết bài toán bằng phương pháp tìm kiếm II

```
function SIMPLE-PROBLEM-SOLVING-AGENT(percept) returns an action
persistent: seq, an action sequence, initially empty
            state, some description of the current world state
            goal, a goal, initially null
            problem, a problem formulation
  state ← UPDATE-STATE(state, percept)
  if seq is empty then
    goal ← FORMULATE-GOAL(state)
    problem ← FORMULATE-PROBLEM(state, goal)
    seq ← SEARCH(problem)
    if seq = failure then return a null action
  action ← FIRST(seq)
  seq ← REST(seq)
return action
```

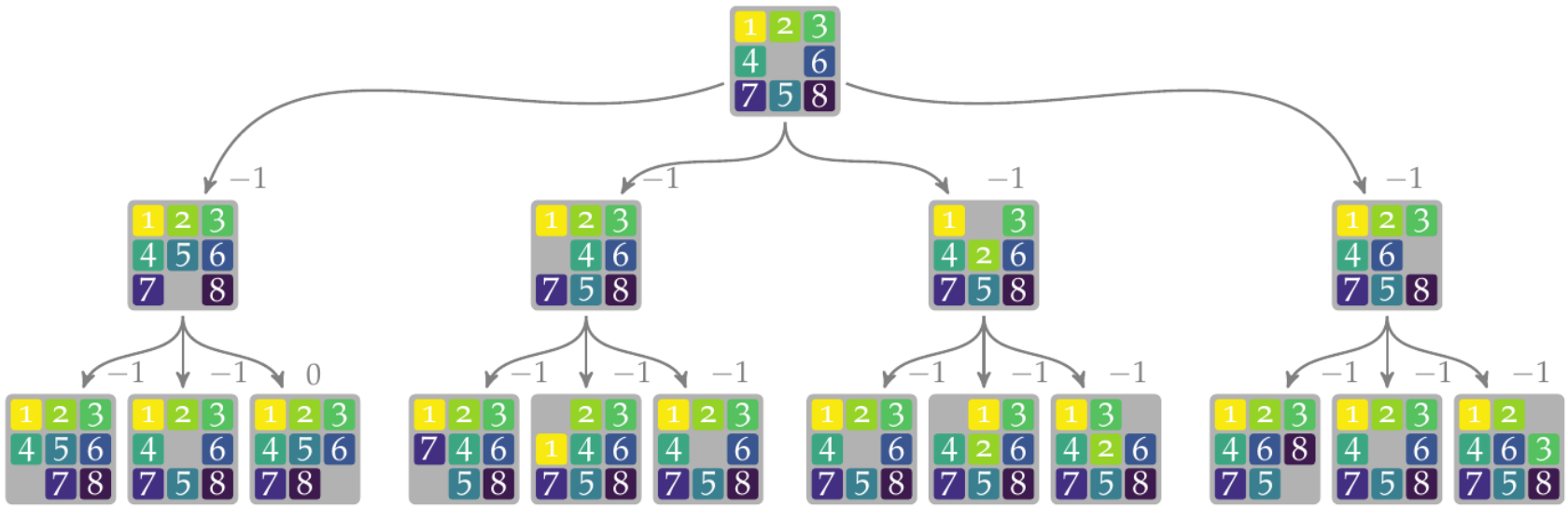
Cây tìm kiếm I

Một lời giải là một chuỗi hành động, các **thuật toán tìm kiếm** thực hiện **chiến lược tìm kiếm** bằng cách xem xét nhiều chuỗi hành động khả thi

- **Cây tìm kiếm:** chuỗi hành động xuất phát từ đỉnh bắt đầu
  - **Nhánh** là các hành động và các **nút** tương ứng với các trạng thái trong không gian tìm kiếm.
  - **Nút gốc** tương ứng với trạng thái bắt đầu
  - Thực hiện hành động bằng cách **mở** trạng thái hiện tại (**nút cha**), sinh tập các trạng thái mới (**các nút con**)
  - **Biên (Frontier):** tập các nút lá có thể mở rộng tại một thời điểm

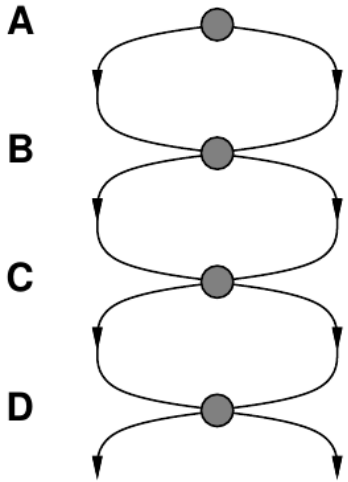
Cây tìm kiếm II

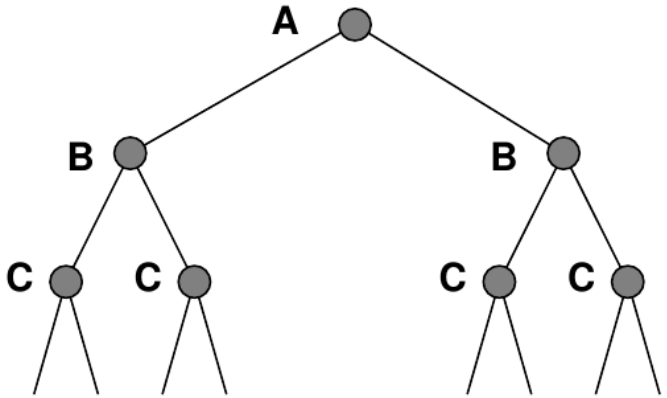
- **Gốc** là trạng thái bắt đầu
- **Một nút trên cây** không chỉ là một trạng thái mà còn là **một kế hoạch**
- **Lưu ý:** một trạng thái có thể xuất hiện nhiều lần trong cây



Đường đi dư thừa

- **Đường đi dư thừa** là không thể tránh khỏi, tồn tại khi nhiều hơn một đường đi giữa các trạng thái





- Di chuyển trên các đường đi dư thừa sẽ làm quy trình tìm kiếm mất kiểm soát
  - Điều này vẫn đúng ngay cả với các thuật toán có khả năng tránh các vòng lặp vô tận

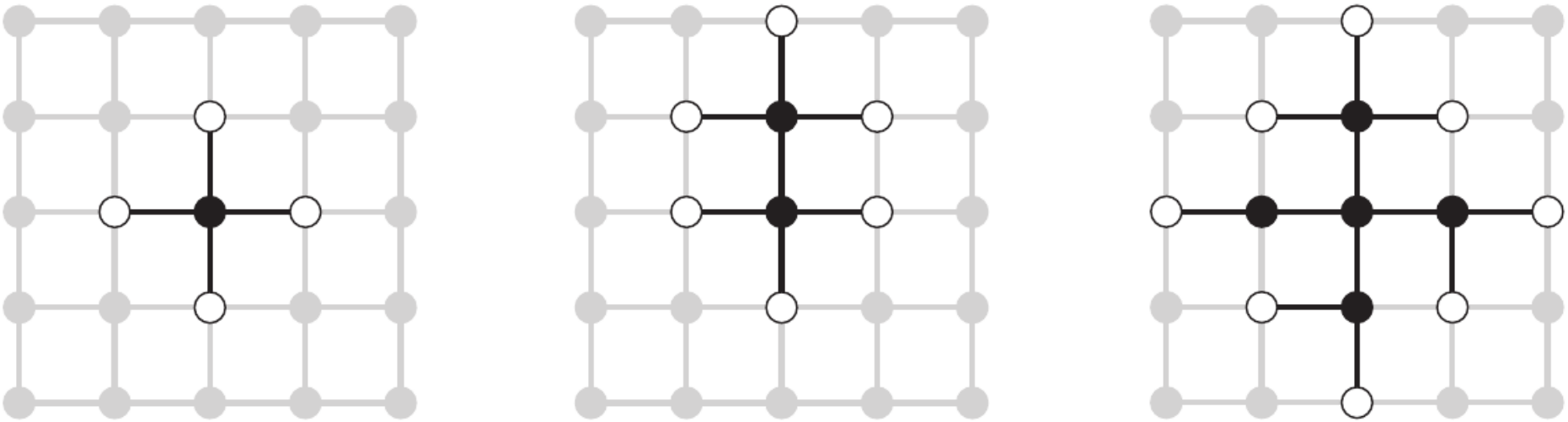
Thuật toán tìm kiếm I

"Those who do not learn from history are doomed to repeat it"

Tạm dịch: Sai lầm sẽ lặp lại với những người không chịu học hỏi

- Các nút đã được mở (khám phá) phải được **ghi nhớ** lại
- **Tính chất:** biên (frontier) **phân tách** không gian trạng thái thành 2 vùng: đã khám phá và chưa khám phá.
  - Mỗi đường đi từ trạng thái bắt đầu đến một trạng thái chưa khám phá cần phải di chuyển qua một trạng thái trong biên

Thuật toán tìm kiếm II



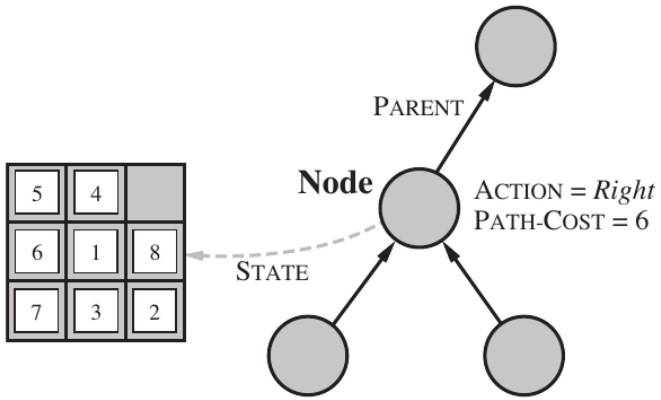
Hình 3: Biên (nút trắng) luôn tách vùng đã thăm (nút đen) với các vùng chưa thăm (nút xám)

Kiến trúc dữ liệu I

Thuật toán tìm kiếm cần phải có một kiến trúc dữ liệu để lưu vết quá trình xây dựng cây tìm kiếm. Với mỗi nút n trên cây tìm kiếm, ta cần phải lưu giữ

1. **Trạng thái (state):** trạng thái tương ứng với nút n
2. **Cha (parent):** nút cha của nút n
3. **Hành động (action):** hành động thực hiện với $parent(n)$ để di chuyển đến n
4. **Chi phí (path-cost):** chi phí di chuyển từ trạng thái bắt đầu đến n, thường được ký hiệu $g(n)$

Kiến trúc dữ liệu II




```
function CHILD-NODE(problem, parent, action) returns a node
  return a node with
    STATE ← problem.RESULT(parent.STATE, action),
    PARENT ← parent,
    ACTION ← action,
    PATH-COST ← parent.PATH-COST +
                problem.STEP-COST(parent.STATE, action)
```

Đánh giá thuật toán I

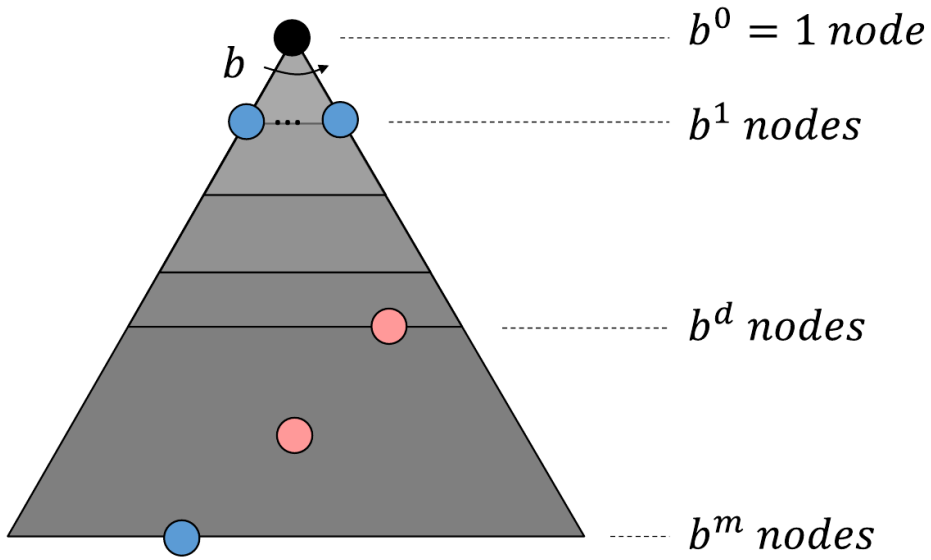
Ta có thể đánh giá hiệu suất của một thuật toán dựa trên 4 tiêu chí:

1. **Độ hoàn thiện (completeness):** Thuật toán có đảm bảo tìm được lời giải?
2. **Tính tối ưu (optimality):** Chiến lược tìm kiếm có tìm được lời giải tốt nhất
3. **Độ phức tạp về mặt thời gian (time complexity):** Thời gian tìm kiếm
4. **Độ phức tạp về mặt không gian (space complexity):** Bộ nhớ cần để thực hiện tìm kiếm

Đánh giá thuật toán II

Độ phức tạp về mặt thời gian và không gian có thể đánh giá thông qua:

- b: số nhánh tối đa của cây
- d: **chiều sâu** thấp nhất của lời giải
- m: **chiều sâu tối đa** của không gian tìm kiếm (có khả năng ∞)



Tổng kết

Tổng kết

- Khái niệm: đồ thị (vô hướng, có hướng), **cây**
- Biểu diễn đồ thị: ma trận kề, danh sách kề, danh sách cạnh
- Bài toán tìm kiếm: **không gian trạng thái**, **lời giải**
- Thuật toán tìm kiếm: **chiến lược tìm kiếm**, **cây tìm kiếm**, **lưu vết**
- Đánh giá thuật toán: hoàn thiện, tối ưu, chi phí không (thời) gian

Tài liệu tham khảo

[1] Bùi Tiến Lên, Bộ môn Khoa học máy tính

Bài giảng môn Cơ sở trí tuệ nhân tạo

[2] Michael Negnevitsky

Artificial Intelligence: A Guide to Intelligent Systems (3rd Edition)

[3] Russell, S. and Norvig, P. (2021).

Artificial intelligence: a modern approach.