

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

# **Cơ sở trí tuệ nhân tạo**

Tìm kiếm mù

**Nguyễn Ngọc Đức**

**2024**

# Nội dung



**1** Breadth-first Search

**2** Depth-first Search

**3** Uniform-cost Search

# Tìm kiếm mù

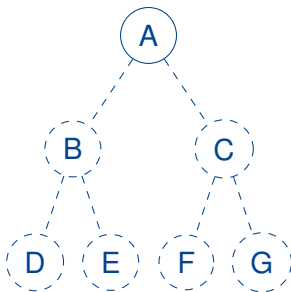


- Chiến lược tìm kiếm không dựa trên thông tin nào khác ngoài định nghĩa bài toán
- Các thuật toán tìm kiếm mù chỉ có khả năng sinh successor và phân biệt trạng thái đích
- Mỗi chiến lược tìm kiếm là một thể hiện (đồ thị/cây) của bài toán tìm kiếm tổng quát

# Breadth-first Search

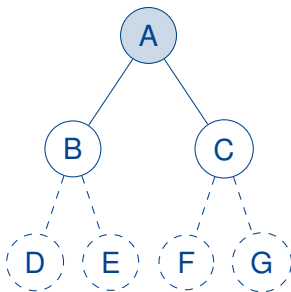
# Breadth-first Search

- Breadth-first Search (tìm kiếm theo chiều rộng - BFS) là chiến lược tìm kiếm đơn giản.
- Tất cả các nút ở một độ sâu nhất định phải được mở trước khi mở các nút ở độ sâu tiếp theo



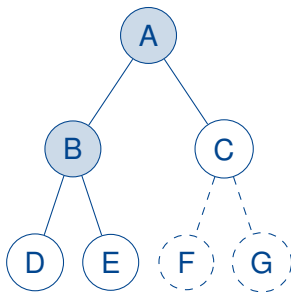
# Breadth-first Search

- Breadth-first Search (tìm kiếm theo chiều rộng - BFS) là chiến lược tìm kiếm đơn giản.
- Tất cả các nút ở một độ sâu nhất định phải được mở trước khi mở các nút ở độ sâu tiếp theo



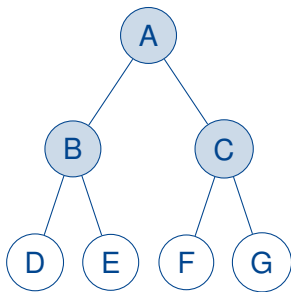
# Breadth-first Search

- Breadth-first Search (tìm kiếm theo chiều rộng - BFS) là chiến lược tìm kiếm đơn giản.
- Tất cả các nút ở một độ sâu nhất định phải được mở trước khi mở các nút ở độ sâu tiếp theo



# Breadth-first Search

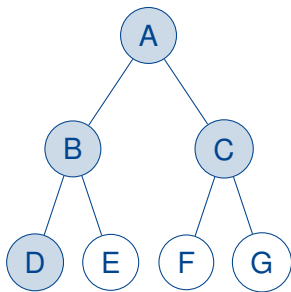
- Breadth-first Search (tìm kiếm theo chiều rộng - BFS) là chiến lược tìm kiếm đơn giản.
- Tất cả các nút ở một độ sâu nhất định phải được mở trước khi mở các nút ở độ sâu tiếp theo





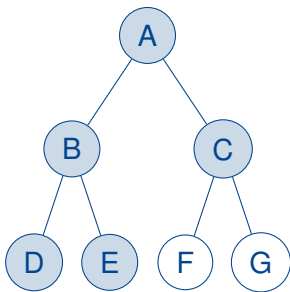
# Breadth-first Search

- Breadth-first Search (tìm kiếm theo chiều rộng - BFS) là chiến lược tìm kiếm đơn giản.
- Tất cả các nút ở một độ sâu nhất định phải được mở trước khi mở các nút ở độ sâu tiếp theo



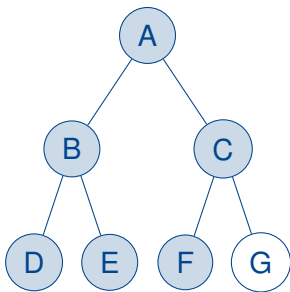
# Breadth-first Search

- Breadth-first Search (tìm kiếm theo chiều rộng - BFS) là chiến lược tìm kiếm đơn giản.
- Tất cả các nút ở một độ sâu nhất định phải được mở trước khi mở các nút ở độ sâu tiếp theo



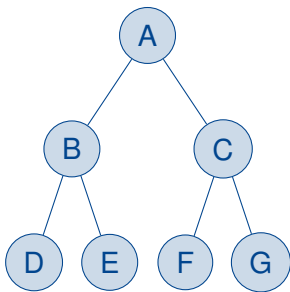
# Breadth-first Search

- Breadth-first Search (tìm kiếm theo chiều rộng - BFS) là chiến lược tìm kiếm đơn giản.
- Tất cả các nút ở một độ sâu nhất định phải được mở trước khi mở các nút ở độ sâu tiếp theo



# Breadth-first Search

- Breadth-first Search (tìm kiếm theo chiều rộng - BFS) là chiến lược tìm kiếm đơn giản.
- Tất cả các nút ở một độ sâu nhất định phải được mở trước khi mở các nút ở độ sâu tiếp theo

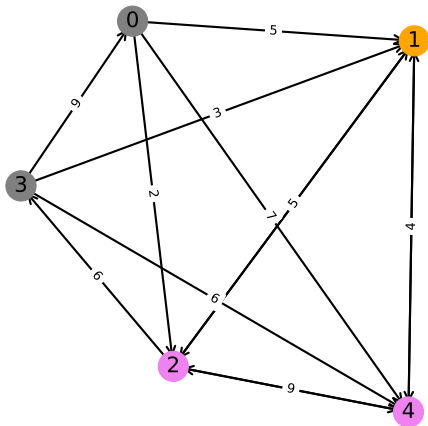


# Ví dụ BFS

Đỉnh bắt đầu 1

Đỉnh đích 2

0	5	2	0	7
0	0	2	0	5
0	5	0	6	2
9	3	0	0	6
0	4	9	0	0

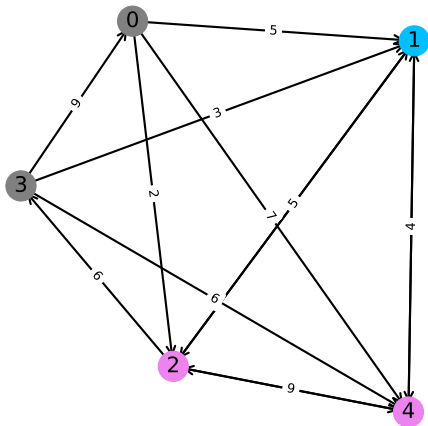


# Ví dụ BFS

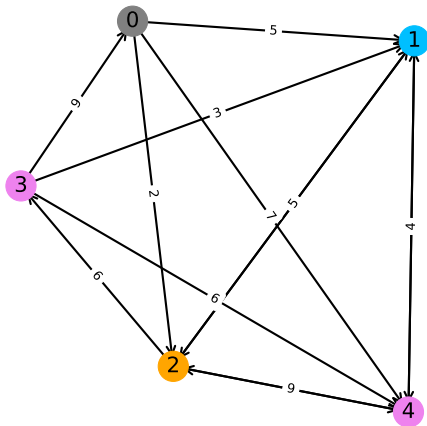
Đỉnh bắt đầu 1

Đỉnh đích 2

0	5	2	0	7
0	0	2	0	5
0	5	0	6	2
9	3	0	0	6
0	4	9	0	0



## Đỉnh đích 2

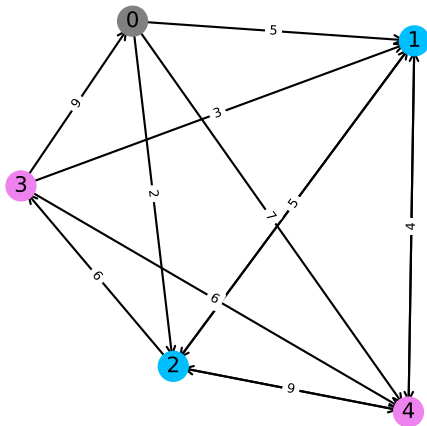
$$\begin{bmatrix} 0 & 5 & 2 & 0 & 7 \\ 0 & 0 & 2 & 0 & 5 \\ 0 & 5 & 0 & 6 & 2 \\ 9 & 3 & 0 & 0 & 6 \\ 0 & 4 & 9 & 0 & 0 \end{bmatrix}$$


# Ví dụ BFS

Đỉnh bắt đầu 1

Đỉnh đích 2

0	5	2	0	7
0	0	2	0	5
0	5	0	6	2
9	3	0	0	6
0	4	9	0	0



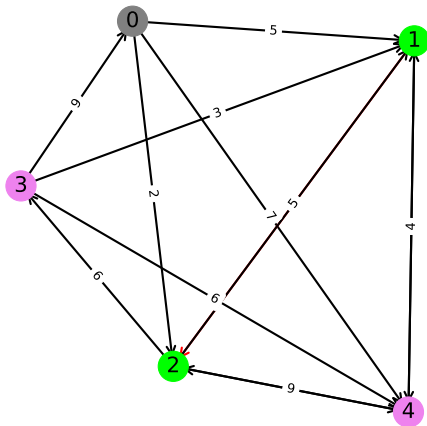


# Ví dụ BFS

Đỉnh bắt đầu 1

Đỉnh đích 2

0	5	2	0	7
0	0	2	0	5
0	5	0	6	2
9	3	0	0	6
0	4	9	0	0



# Breadth-first Search

- **Time:**  $1 + b + b^2 + b^3 + \dots + b^d = O(b^d)$
- **Space:**  $O(b^{d-1})$  cho tập mở và  $O(b^d)$  cho biên
- **Complete:** có (nếu  $b$  hữu hạn)
- **Optimal:** chi phí di chuyển như nhau

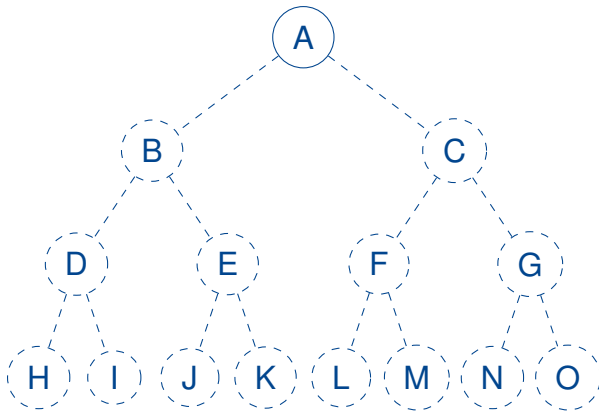
Depth	Nodes	Time	Memory
2	110	.11 ms	107 KB
4	11110	11 ms	10.6 MB
6	$10^6$	1.1 s	1 GB
8	$10^8$	2 mins	103 GB
10	$10^{10}$	3 hrs	10 TB
12	$10^{12}$	13 days	1 PB
14	$10^{14}$	3.5 yrs	99 PB
16	$10^{16}$	350 yrs	10 EB

Giả sử  $b = 10$ , tốc độ giải 1m nodes/s, lưu trữ 1000 bytes 1 node

# Depth-first Search

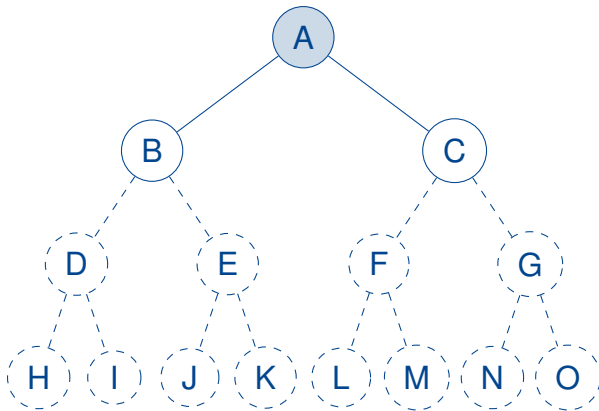
# Depth-first Search

- Depth-first Search - DFS (tìm kiếm theo chiều sâu) mở các nút sâu nhất trước



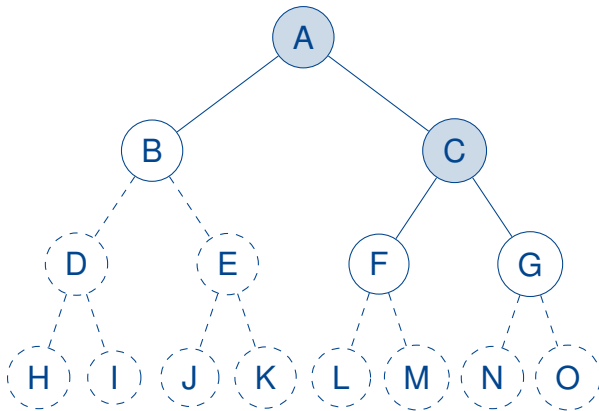
# Depth-first Search

- Depth-first Search - DFS (tìm kiếm theo chiều sâu) mở các nút sâu nhất trước



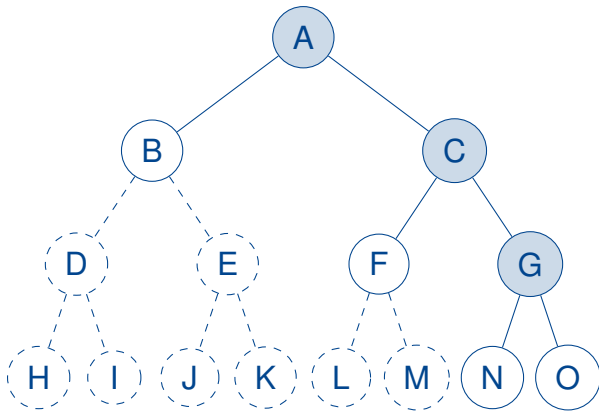
# Depth-first Search

- Depth-first Search - DFS (tìm kiếm theo chiều sâu) mở các nút sâu nhất trước



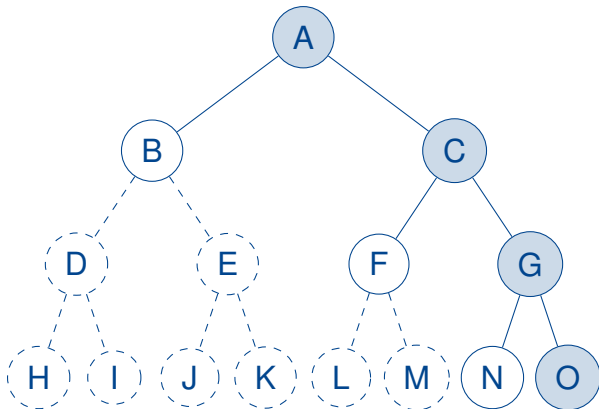
# Depth-first Search

- Depth-first Search - DFS (tìm kiếm theo chiều sâu) mở các nút sâu nhất trước



# Depth-first Search

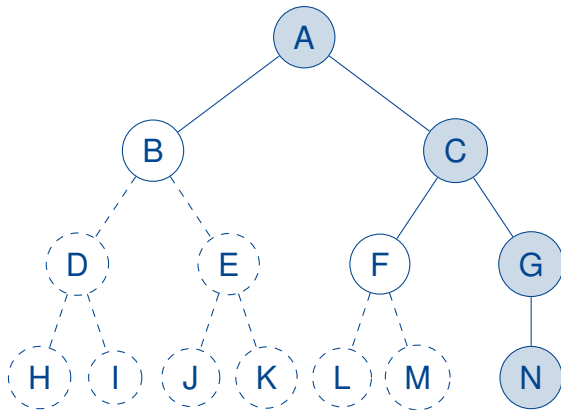
- Depth-first Search - DFS (tìm kiếm theo chiều sâu) mở các nút sâu nhất trước





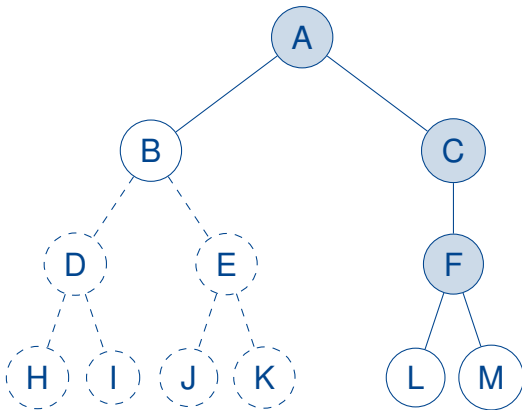
# Depth-first Search

- Depth-first Search - DFS (tìm kiếm theo chiều sâu) mở các nút sâu nhất trước



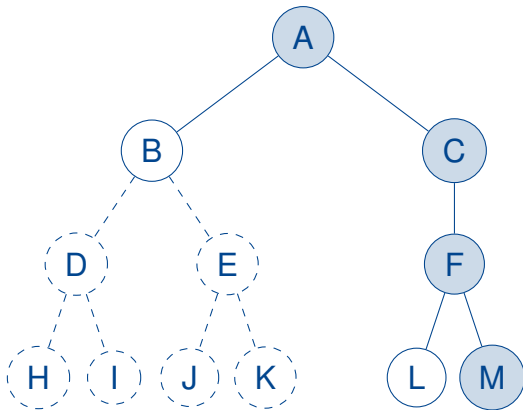
# Depth-first Search

- Depth-first Search - DFS (tìm kiếm theo chiều sâu) mở các nút sâu nhất trước



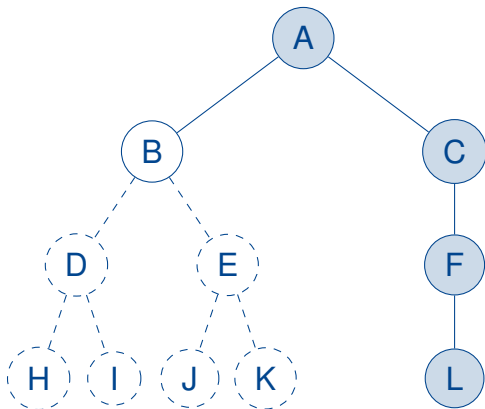
# Depth-first Search

- Depth-first Search - DFS (tìm kiếm theo chiều sâu) mở các nút sâu nhất trước



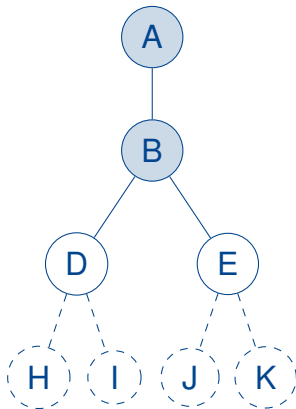
# Depth-first Search

- Depth-first Search - DFS (tìm kiếm theo chiều sâu) mở các nút sâu nhất trước



# Depth-first Search

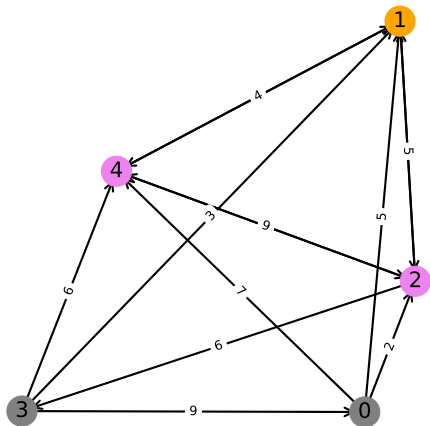
- Depth-first Search - DFS (tìm kiếm theo chiều sâu) mở các nút sâu nhất trước



# Ví dụ DFS

Đỉnh bắt đầu 1

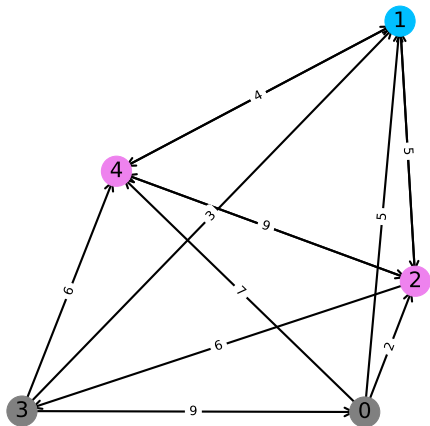
Đỉnh đích 2

$$\begin{bmatrix} 0 & 5 & 2 & 0 & 7 \\ 0 & 0 & 2 & 0 & 5 \\ 0 & 5 & 0 & 6 & 2 \\ 9 & 3 & 0 & 0 & 6 \\ 0 & 4 & 9 & 0 & 0 \end{bmatrix}$$


# Ví dụ DFS

Đỉnh bắt đầu 1

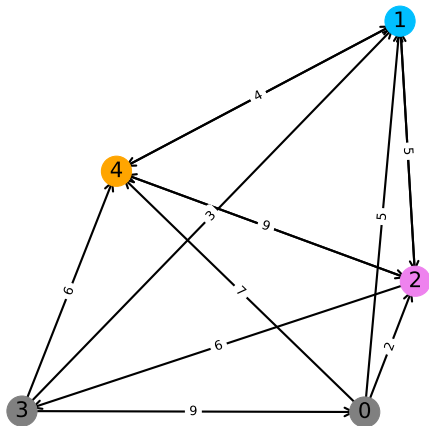
Đỉnh đích 2

$$\begin{bmatrix} 0 & 5 & 2 & 0 & 7 \\ 0 & 0 & 2 & 0 & 5 \\ 0 & 5 & 0 & 6 & 2 \\ 9 & 3 & 0 & 0 & 6 \\ 0 & 4 & 9 & 0 & 0 \end{bmatrix}$$


# Ví dụ DFS

Đỉnh bắt đầu 1

Đỉnh đích 2

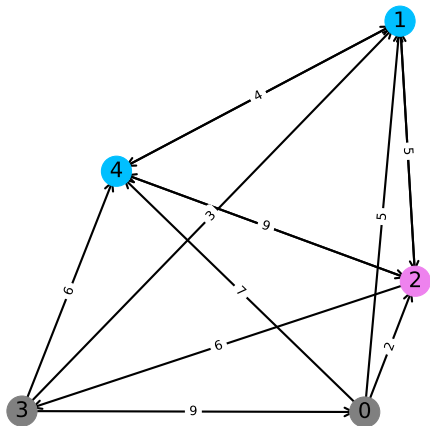
$$\begin{bmatrix} 0 & 5 & 2 & 0 & 7 \\ 0 & 0 & 2 & 0 & 5 \\ 0 & 5 & 0 & 6 & 2 \\ 9 & 3 & 0 & 0 & 6 \\ 0 & 4 & 9 & 0 & 0 \end{bmatrix}$$




# Ví dụ DFS

Đỉnh bắt đầu 1

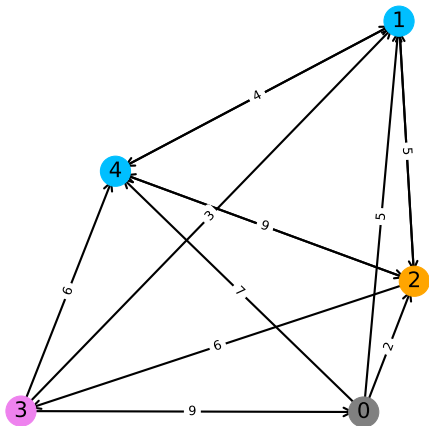
Đỉnh đích 2

$$\begin{bmatrix} 0 & 5 & 2 & 0 & 7 \\ 0 & 0 & 2 & 0 & 5 \\ 0 & 5 & 0 & 6 & 2 \\ 9 & 3 & 0 & 0 & 6 \\ 0 & 4 & 9 & 0 & 0 \end{bmatrix}$$


# Ví dụ DFS

Đỉnh bắt đầu 1

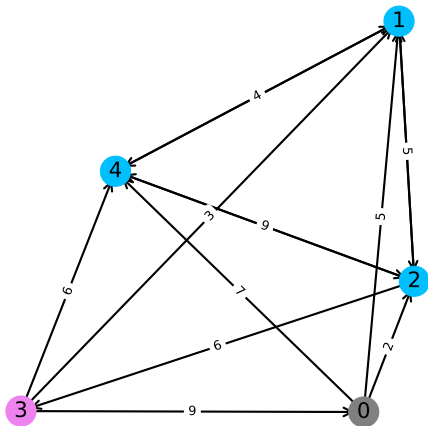
Đỉnh đích 2

$$\begin{bmatrix} 0 & 5 & 2 & 0 & 7 \\ 0 & 0 & 2 & 0 & 5 \\ 0 & 5 & 0 & 6 & 2 \\ 9 & 3 & 0 & 0 & 6 \\ 0 & 4 & 9 & 0 & 0 \end{bmatrix}$$


# Ví dụ DFS

Đỉnh bắt đầu 1

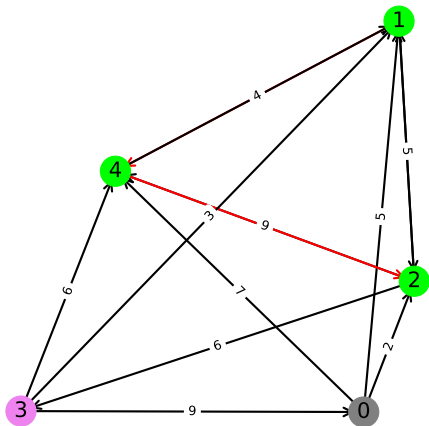
Đỉnh đích 2

$$\begin{bmatrix} 0 & 5 & 2 & 0 & 7 \\ 0 & 0 & 2 & 0 & 5 \\ 0 & 5 & 0 & 6 & 2 \\ 9 & 3 & 0 & 0 & 6 \\ 0 & 4 & 9 & 0 & 0 \end{bmatrix}$$


# Ví dụ DFS

Đỉnh bắt đầu 1

Đỉnh đích 2

$$\begin{bmatrix} 0 & 5 & 2 & 0 & 7 \\ 0 & 0 & 2 & 0 & 5 \\ 0 & 5 & 0 & 6 & 2 \\ 9 & 3 & 0 & 0 & 6 \\ 0 & 4 & 9 & 0 & 0 \end{bmatrix}$$


# Depth-first Search

- **Time:**  $O(b^m)$
- **Space:**  $O(bm)$  kích thước không gian tuyến tính!
- **Complete:** có (nếu không gian hữu hạn)
- **Optimal:** lời giải "phải nhất".

# Depth-first Search

- **Time:**  $O(b^m)$ 
  - Tệ nếu  $m$  lớn hơn nhiều so với  $b$
- **Space:**  $O(bm)$  kích thước không gian tuyến tính!
- **Complete:** có (nếu không gian hữu hạn)
- **Optimal:** lời giải "phải nhất".

# Uniform-cost Search

# Uniform-cost Search I

**Uniform-cost search (UCS)** mở nút  $n$  với **chi phí đường đi thấp nhất**  $g(n)$

- Tương tự với thuật toán Dijkstra

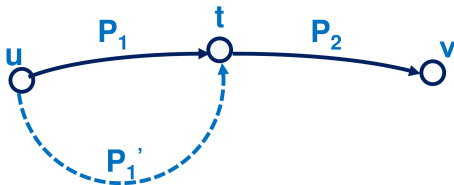
## Nguyên lý Bellman

Gọi  $P$  là đường đi ngắn nhất từ đỉnh  $u$  đến đỉnh  $v$  và  $t \in P$ . Giả sử  $P = P_1 \oplus P_2$  với  $P_1$  là đường đi con của  $P$  từ  $u$  đến  $t$  và  $P_2$  là đường đi con của  $P$  từ  $t$  đến  $v$  khi đó  $P_1$  cũng là đường đi ngắn nhất từ  $u$  đến  $t$



# Uniform-cost Search II

**Chứng minh.** Giả sử tồn tại  $P'_1$  là đường đi ngắn hơn  $P_1$  ta có



$$g(P'_1) < g(P_1) \Rightarrow g(P'_1 \oplus P_2) < g(P_1 \oplus P_2) = g(P)$$

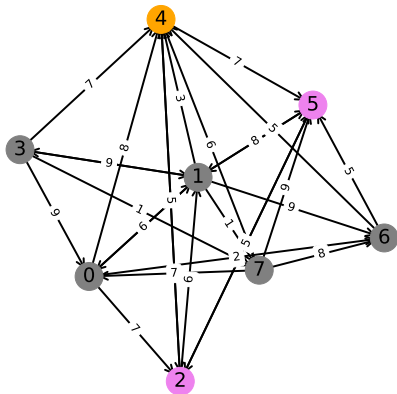
Vô lý vì P là đường đi ngắn nhất từ u đến v

# Ví dụ UCS

Đỉnh bắt đầu 4

Đỉnh đích 3

0	6	7	0	8	0	2	0
6	0	0	4	3	4	9	1
0	6	0	0	4	4	0	0
9	9	0	0	7	0	0	1
0	0	5	0	0	7	0	0
0	8	5	0	0	0	0	0
0	0	0	0	5	5	0	0
7	0	0	0	6	9	8	0



Queue: (5, 2, 4), (7, 5, 4)

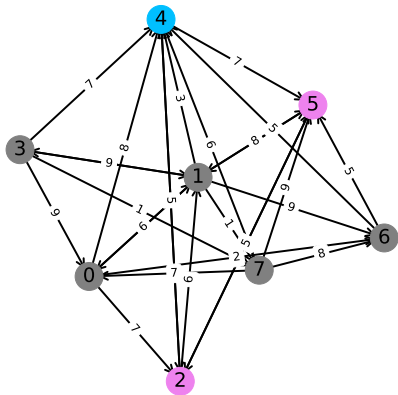
Explored: 4: 4

# Ví dụ UCS

Đỉnh bắt đầu 4

Đỉnh đích 3

0	6	7	0	8	0	2	0
6	0	0	4	3	4	9	1
0	6	0	0	4	4	0	0
9	9	0	0	7	0	0	1
0	0	5	0	0	7	0	0
0	8	5	0	0	0	0	0
0	0	0	0	5	5	0	0
7	0	0	0	6	9	8	0



Queue: (5, 2, 4), (7, 5, 4)

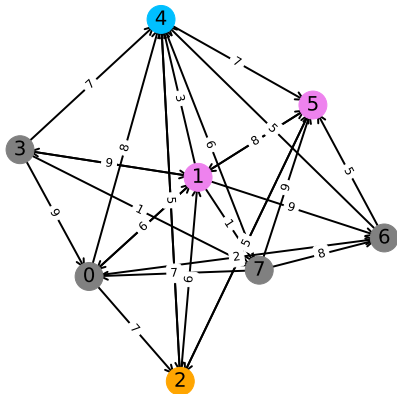
Explored: 4: 4

# Ví dụ UCS

Đỉnh bắt đầu 4

Đỉnh đích 3

0	6	7	0	8	0	2	0
6	0	0	4	3	4	9	1
0	6	0	0	4	4	0	0
9	9	0	0	7	0	0	1
0	0	5	0	0	7	0	0
0	8	5	0	0	0	0	0
0	0	0	0	5	5	0	0
7	0	0	0	6	9	8	0



Queue: (7, 5, 4), (9, 5, 2), (11, 1, 2)

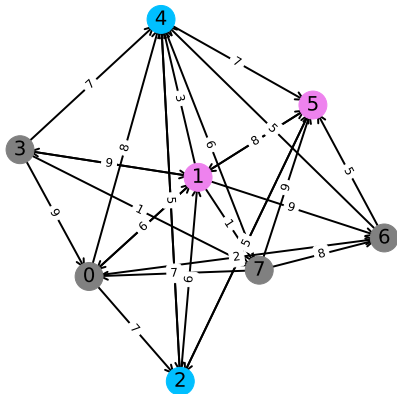
Explored: 4: 4, 2: 4

# Ví dụ UCS

Đỉnh bắt đầu 4

Đỉnh đích 3

0	6	7	0	8	0	2	0
6	0	0	4	3	4	9	1
0	6	0	0	4	4	0	0
9	9	0	0	7	0	0	1
0	0	5	0	0	7	0	0
0	8	5	0	0	0	0	0
0	0	0	0	5	5	0	0
7	0	0	0	6	9	8	0



Queue: (7, 5, 4), (9, 5, 2), (11, 1, 2)

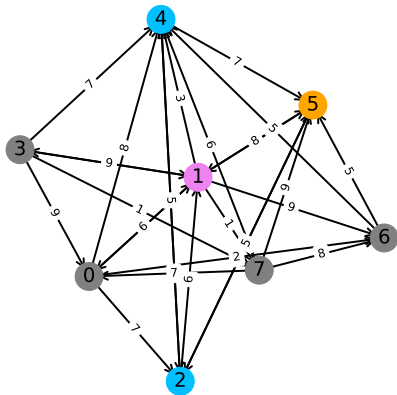
Explored: 4: 4, 2: 4

# Ví dụ UCS

Đỉnh bắt đầu 4

Đỉnh đích 3

0	6	7	0	8	0	2	0
6	0	0	4	3	4	9	1
0	6	0	0	4	4	0	0
9	9	0	0	7	0	0	1
0	0	5	0	0	7	0	0
0	8	5	0	0	0	0	0
0	0	0	0	5	5	0	0
7	0	0	0	6	9	8	0



Queue: (9, 5, 2), (11, 1, 2), (15, 1, 5)

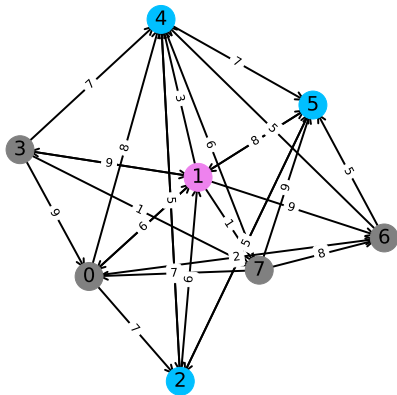
Explored: 4: 4, 2: 4, 5: 4

# Ví dụ UCS

Đỉnh bắt đầu 4

Đỉnh đích 3

0	6	7	0	8	0	2	0
6	0	0	4	3	4	9	1
0	6	0	0	4	4	0	0
9	9	0	0	7	0	0	1
0	0	5	0	0	7	0	0
0	8	5	0	0	0	0	0
0	0	0	0	5	5	0	0
7	0	0	0	6	9	8	0



Queue: (9, 5, 2), (11, 1, 2), (15, 1, 5)

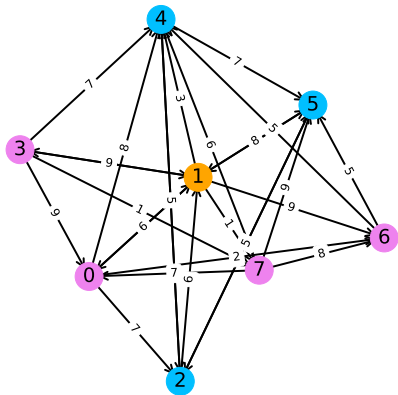
Explored: 4: 4, 2: 4, 5: 4

# Ví dụ UCS

Đỉnh bắt đầu 4

Đỉnh đích 3

0	6	7	0	8	0	2	0
6	0	0	4	3	4	9	1
0	6	0	0	4	4	0	0
9	9	0	0	7	0	0	1
0	0	5	0	0	7	0	0
0	8	5	0	0	0	0	0
0	0	0	0	5	5	0	0
7	0	0	0	6	9	8	0



Queue: (15, 1, 5)

Explored: 4: 4, 2: 4, 5: 4

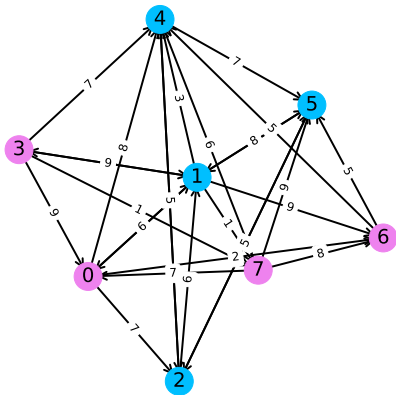


# Ví dụ UCS

Đỉnh bắt đầu 4

Đỉnh đích 3

0	6	7	0	8	0	2	0
6	0	0	4	3	4	9	1
0	6	0	0	4	4	0	0
9	9	0	0	7	0	0	1
0	0	5	0	0	7	0	0
0	8	5	0	0	0	0	0
0	0	0	0	5	5	0	0
7	0	0	0	6	9	8	0



Queue: (15, 1, 5)

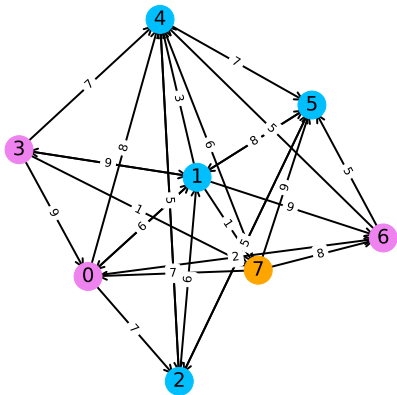
Explored: 4: 4, 2: 4, 5: 4

# Ví dụ UCS

Đỉnh bắt đầu 4

Đỉnh đích 3

0	6	7	0	8	0	2	0
6	0	0	4	3	4	9	1
0	6	0	0	4	4	0	0
9	9	0	0	7	0	0	1
0	0	5	0	0	7	0	0
0	8	5	0	0	0	0	0
0	0	0	0	5	5	0	0
7	0	0	0	6	9	8	0



Queue: (12, 7, 1), (15, 1, 5), (15, 3, 1),  
(17, 0, 1), (20, 6, 1)

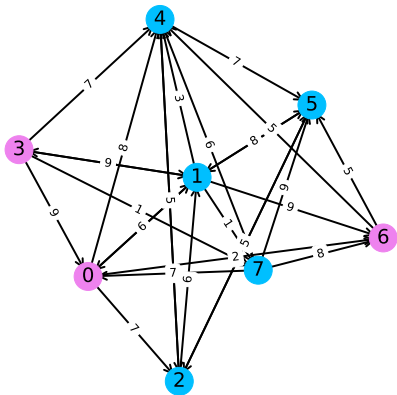
Explored: 4: 4, 2: 4, 5: 4, 1: 2

# Ví dụ UCS

Đỉnh bắt đầu 4

Đỉnh đích 3

0	6	7	0	8	0	2	0
6	0	0	4	3	4	9	1
0	6	0	0	4	4	0	0
9	9	0	0	7	0	0	1
0	0	5	0	0	7	0	0
0	8	5	0	0	0	0	0
0	0	0	0	5	5	0	0
7	0	0	0	6	9	8	0



Queue: (12, 7, 1), (15, 1, 5), (15, 3, 1),  
(17, 0, 1), (20, 6, 1)

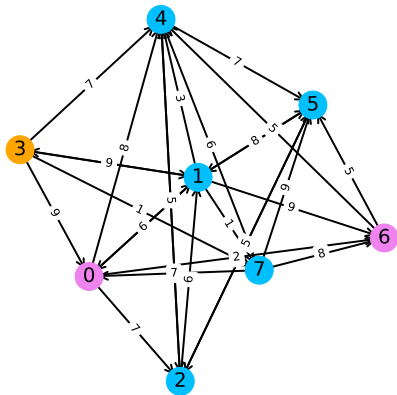
Explored: 4: 4, 2: 4, 5: 4, 1: 2

# Ví dụ UCS

Đỉnh bắt đầu 4

Đỉnh đích 3

0	6	7	0	8	0	2	0
6	0	0	4	3	4	9	1
0	6	0	0	4	4	0	0
9	9	0	0	7	0	0	1
0	0	5	0	0	7	0	0
0	8	5	0	0	0	0	0
0	0	0	0	5	5	0	0
7	0	0	0	6	9	8	0



Queue: (15, 3, 1), (17, 0, 1), (19, 0, 7),  
(20, 6, 1), (20, 6, 7)

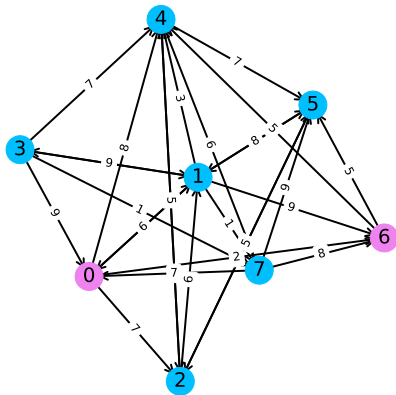
Explored: 4: 4, 2: 4, 5: 4, 1: 2, 7: 1

# Ví dụ UCS

Đỉnh bắt đầu 4

Đỉnh đích 3

0	6	7	0	8	0	2	0
6	0	0	4	3	4	9	1
0	6	0	0	4	4	0	0
9	9	0	0	7	0	0	1
0	0	5	0	0	7	0	0
0	8	5	0	0	0	0	0
0	0	0	0	5	5	0	0
7	0	0	0	6	9	8	0



Queue: (15, 3, 1), (17, 0, 1), (19, 0, 7),  
(20, 6, 1), (20, 6, 7)

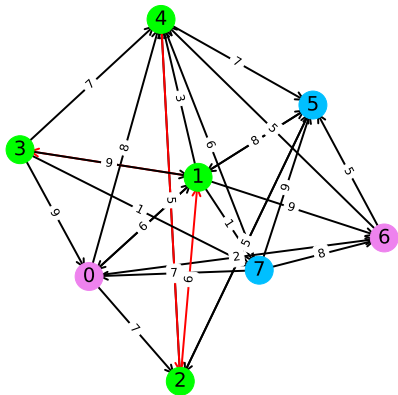
Explored: 4: 4, 2: 4, 5: 4, 1: 2, 7: 1

# Ví dụ UCS

Đỉnh bắt đầu 4

Đỉnh đích 3

0	6	7	0	8	0	2	0
6	0	0	4	3	4	9	1
0	6	0	0	4	4	0	0
9	9	0	0	7	0	0	1
0	0	5	0	0	7	0	0
0	8	5	0	0	0	0	0
0	0	0	0	5	5	0	0
7	0	0	0	6	9	8	0

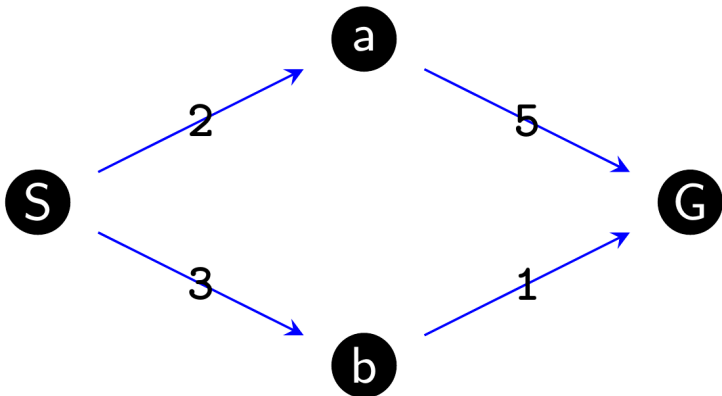


Queue: (19, 0, 7), (20, 6, 1), (20, 6, 7)

Explored: 4: 4, 2: 4, 5: 4, 1: 2, 7: 1, 3: 1

# Uniform-cost Search

Ta nên dừng khi ta enqueue hay dequeue đích?



Hình 1: Tìm đường đi từ S đến G

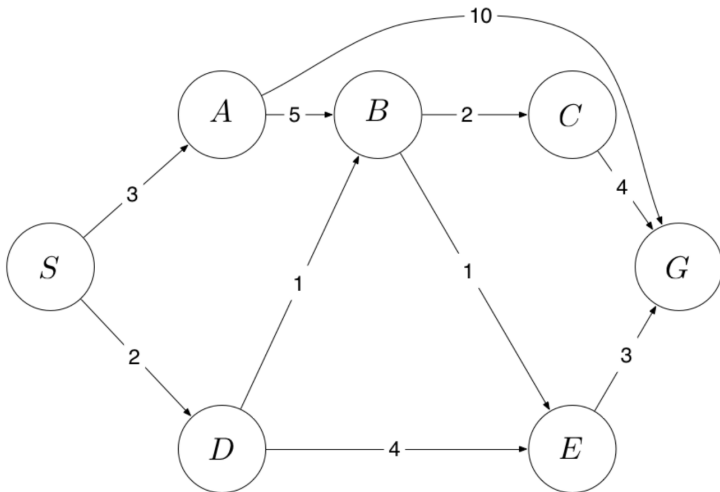
# Đánh giá



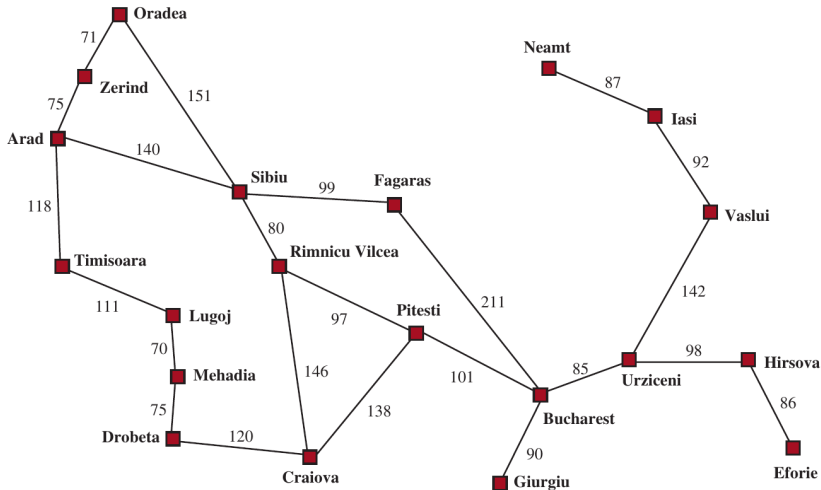
- Với chi phí di chuyển thấp nhất là  $\epsilon$ ,  $C^*$  là chi phí lời giải tối ưu
- **Time:**  $O(b^{1+\lceil C^*/\epsilon \rceil})$
- **Space:**  $O(b^{1+\lceil C^*/\epsilon \rceil})$
- **Complete:** có
- **Optimal:** có



# Bài tập I



# Bài tập II



# Tổng kết



- Bài toán tìm kiếm
- Các thuật toán tìm kiếm mù: BFS, DFS, UCS

# 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.