



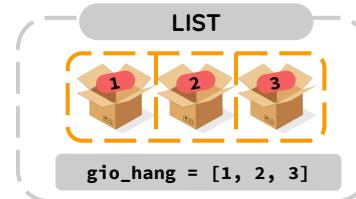
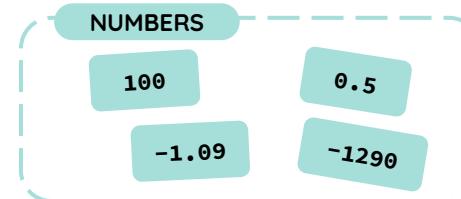
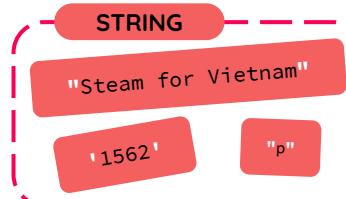
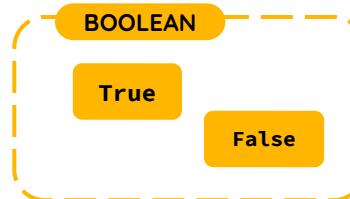
# CS 201 - Huấn luyện kỹ năng **Python Cơ Bản 5**



Đây là một sản phẩm trí tuệ có bản quyền thuộc về STEAM for Vietnam. Các bên chỉ được sử dụng với mục đích học tập, nghiên cứu, và không được quyền sử dụng sản phẩm này nhằm mục đích thu lợi nhuận dù trực tiếp hay gián tiếp.

# ÔN LẠI BÀI CŨ

## CÁC KIỂU DỮ LIỆU



### PHÉP TOÁN LOGIC

and  
or  
not

### THAO TÁC CHUỖI

+

str()

len()

### CÁC TOÁN TỬ

+

-

\*

/

%

//

\*\*

==

!=

>

<

>=

<=

### THAO TÁC DANH SÁCH

append()  
len()

range()  
gio\_hang[0]



# ÔN LẠI BÀI CŨ

```
if điều_kiện_a:  
    # Công việc A  
elif điều_kiện_b:  
    # Công việc B  
else:  
    # Công việc C
```

```
while điều_kiện_lặp:  
    # Công việc lặp lại
```

```
for phần_tử in danh_sách:  
    # Công việc lặp lại
```

```
def tên_hàm(các_tham_số):  
    # Xử lý trong hàm  
    a = 1  
    # Giá trị trả về của hàm(nếu có)  
    return a
```

```
b = tên_hàm(các_tham_số)
```

```
class LớpĐốiTượng:  
    def __init__(self, các_tham_số):  
        # Các giá trị khởi tạo cho đối tượng  
  
    def tên_phương_thức(self, các_tham_số):  
        # Xử lý trong hành động  
        được_nhấn = False  
        # Giá trị trả về của hành động (nếu có)  
        return được_nhấn
```

```
đối_tượng = LớpĐốiTượng(các_tham_số)  
đối_tượng.tên_phương_thức(các_tham_số)
```



# ÔN LẠI BÀI CŨ



The diagram shows a code snippet in a light gray box:

```
from vex import Bumper, Ports
bumper_front = Bumper(Ports.PORT3)
bumper_front.pressing()
```

- Gói (package)**: Points to the word `vex`.
- Lớp đối tượng (class)**: Points to the class name `Bumper`.
- Đối tượng cụ thể (object)**: Points to the variable `bumper_front`.
- Tham số (arguments)**: Points to the argument `PORT3`.
- Phương thức thực hiện hành động của đối tượng (method)**: Points to the method name `pressing`.



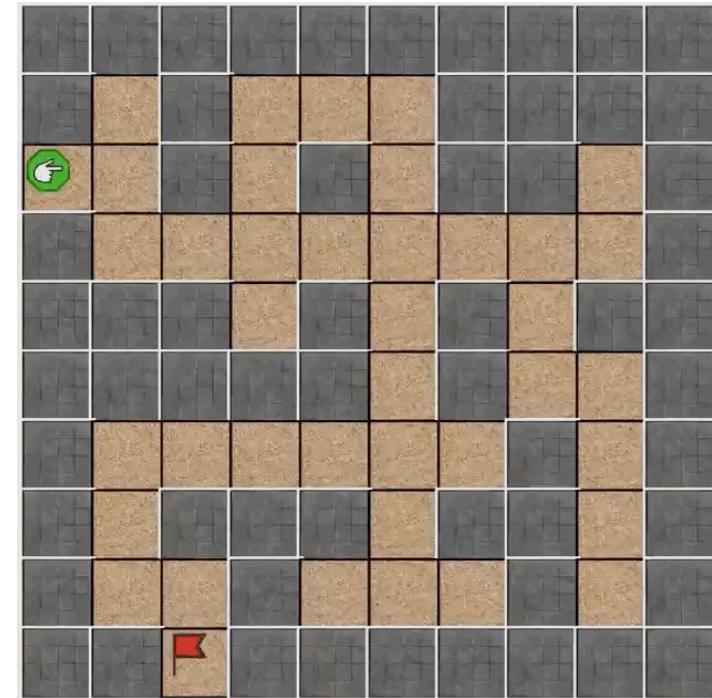
# ÔN LẠI BÀI CŨ

- Scope (phạm vi) trong Python
  - Biến toàn cục
  - Biến cục bộ
- FSM (Finite state machine) (Trạng thái máy hữu hạn)
  - Tìm hiểu về FSM
  - Ứng dụng của FSM trong lập trình robotics

# Mê cung (Maze)

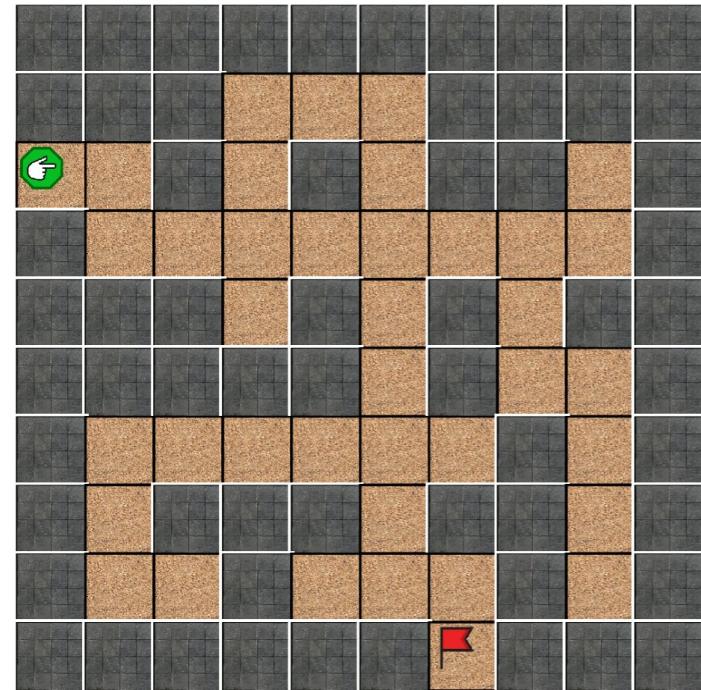
Chúng ta sẽ có đường đi từ **lối vào** đến **lối ra**

Sau đó, thêm các lối đi khác để gây rối cho người giải mê cung



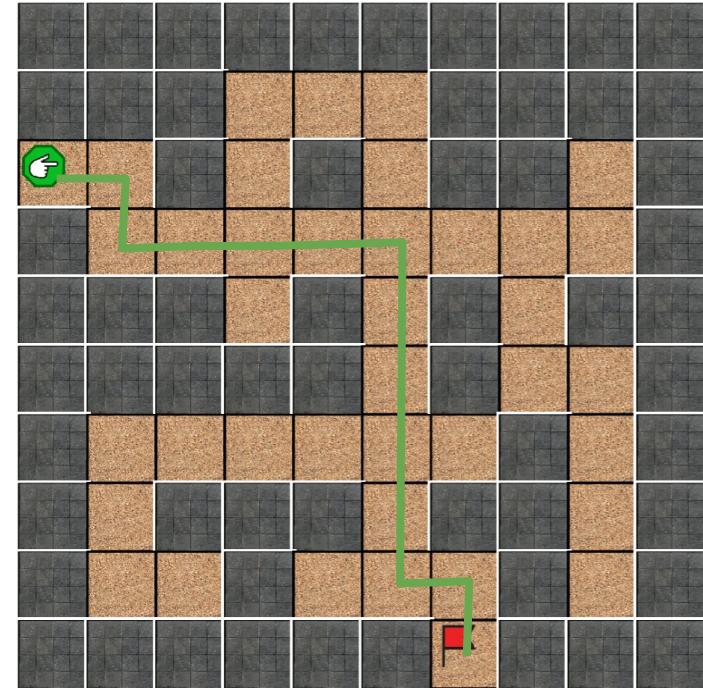
# Mê cung 2D

Đây là một ví dụ về mê cung 2D

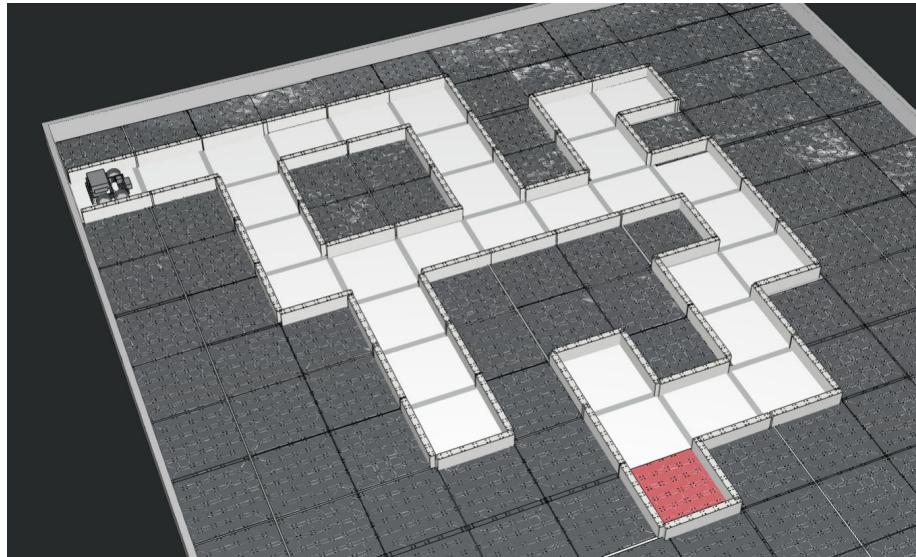


# Mê cung 2D

Còn đây là đường đi để vượt qua  
mê cung ấy



# Robot trong mê cung



[https://bit.ly/S4V\\_CS201\\_Maze\\_DFS](https://bit.ly/S4V_CS201_Maze_DFS)

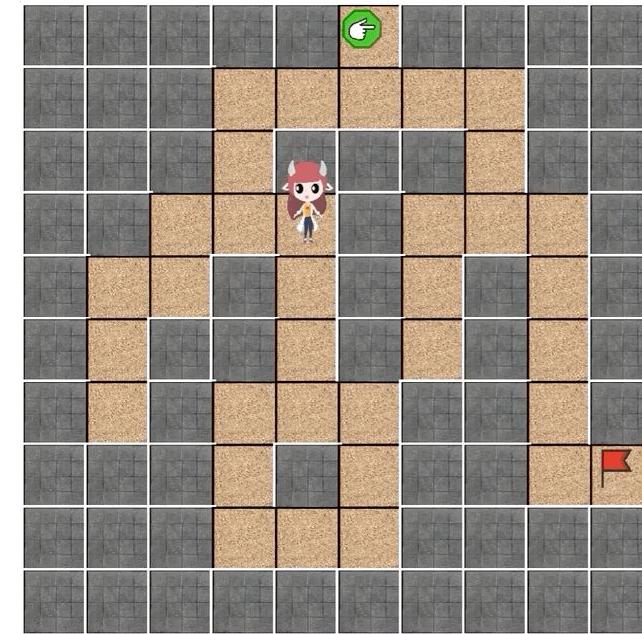


# Mê cung

Nếu tớ cứ đi lung tung mãi  
thì chắc cũng tìm được  
đường ra nhỉ?



Không hiệu quả

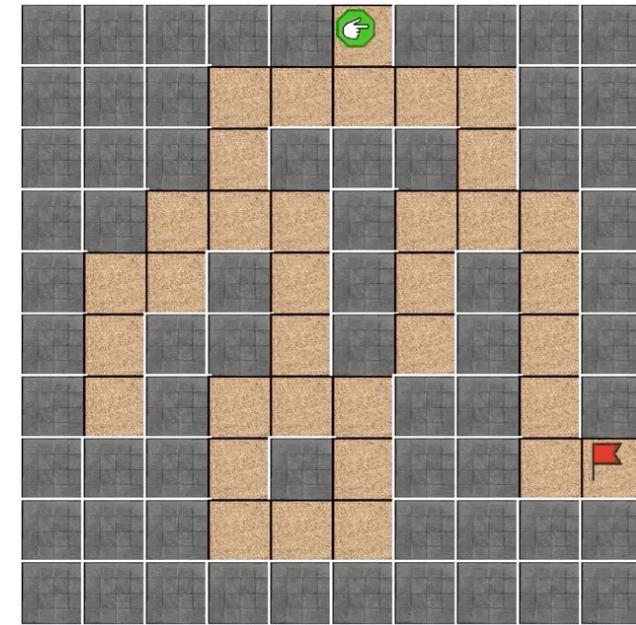


# Mê cung

Tớ sẽ đánh dấu những nơi nào mình đã đi qua và cố gắng không đi vào đó nữa!

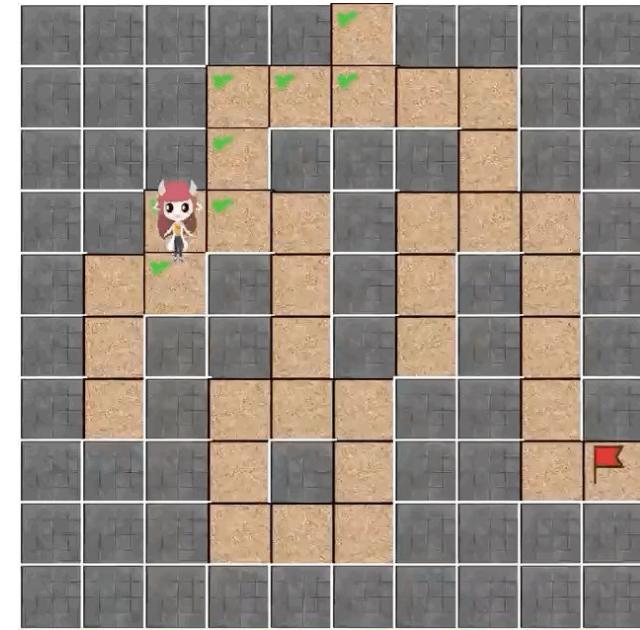


Không hiệu quả



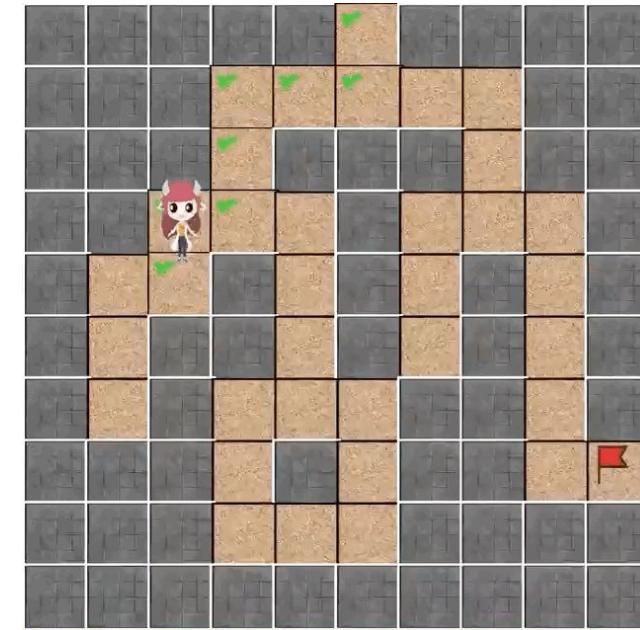
# Bí kíp “Quay ngược thời gian”

Nếu xung quanh là đường cùt hoặc đường đã đi thì tớ cần quay lại theo những bước mà mình đã đi để tìm lối rẽ khác!



# Bí kíp “Quay ngược thời gian”

Nhưng nhiều bước như vậy  
thì sao nhớ hết? Tớ không  
phải là “Siêu trí tuệ”!



# TÌM THEO CHIỀU SÂU

(Depth-First Search)

# Tìm theo chiều sâu (Depth-First Search)

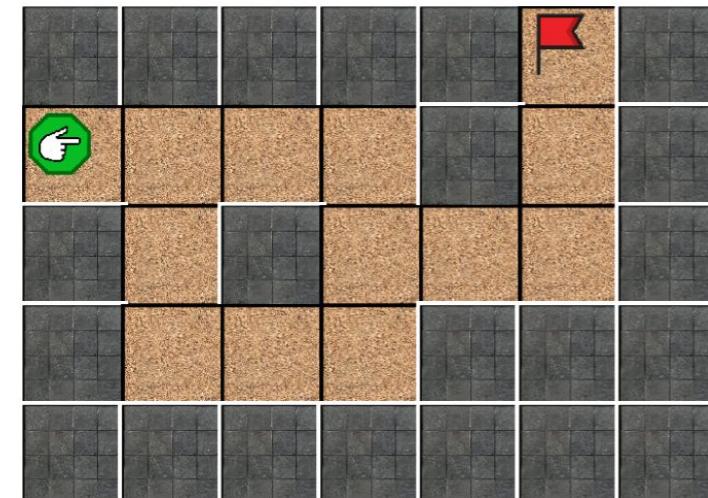
Chuẩn bị

Tập giấy

Bút chì

Viên phấn

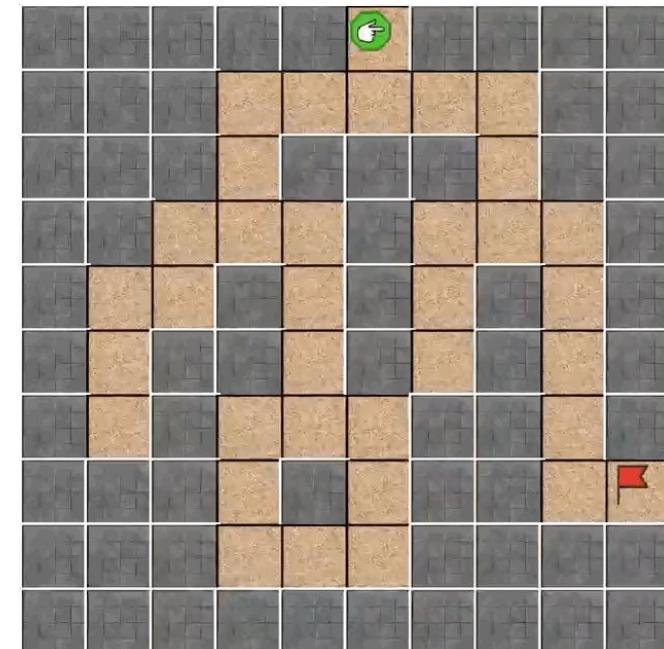
Hộp đựng giấy



# Tìm theo chiều sâu (Depth-First Search)

## Ý tưởng

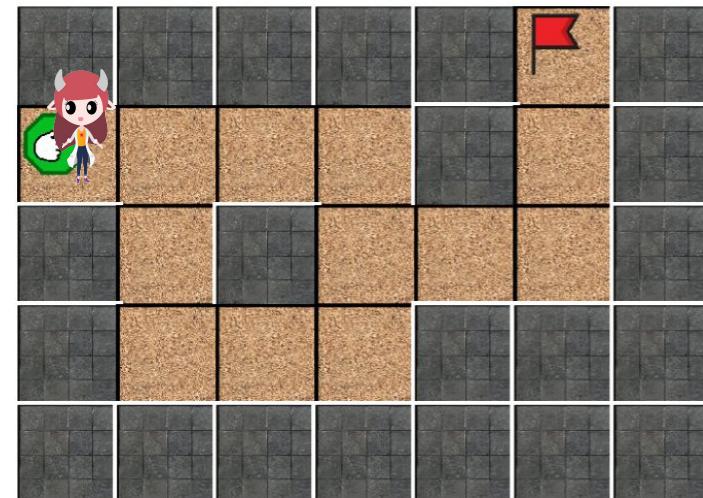
Chúng ta sẽ tìm theo **một nhánh** đến khi tìm được đường ra. Nếu hết đường thì **quay lại** đến khi thấy **nhánh mới**.



# Tìm theo chiều sâu (Depth-First Search)

## Thuật toán

**Bước 1** Đến điểm xuất phát

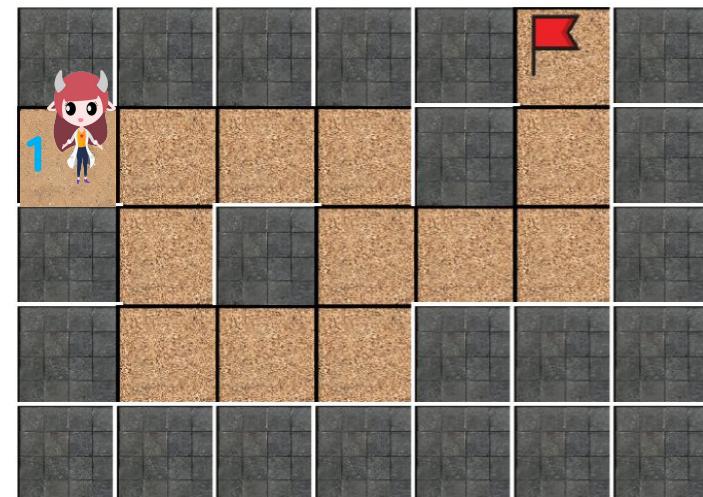


# Tìm theo chiều sâu (Depth-First Search)

## Thuật toán

**Bước 1** Đến điểm xuất phát

**Bước 2** Dùng phấn đánh số vị trí hiện tại  
(Không dùng các số trùng nhau)



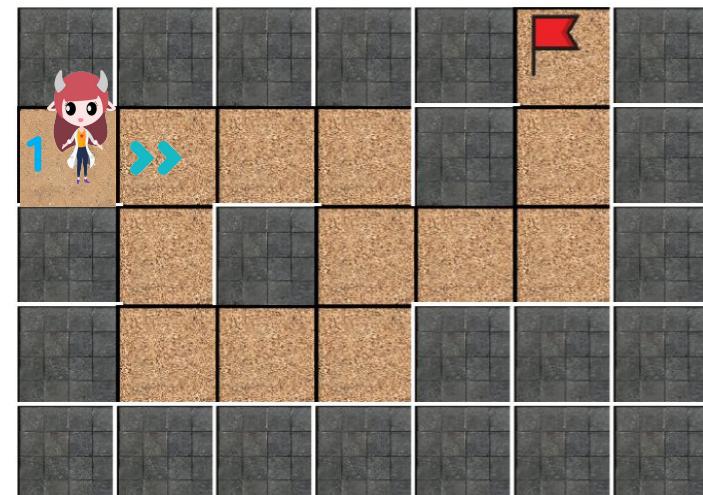
# Tìm theo chiều sâu (Depth-First Search)

## Thuật toán

**Bước 1** Đến điểm xuất phát

**Bước 2** Dùng phấn đánh số vị trí hiện tại  
*(Không dùng các số trùng nhau)*

**Bước 3** Nhìn xung quanh theo thứ tự: **Trên - dưới - trái - phải** xem có đường nào chưa đánh dấu không. **Nếu có**, sang **bước 4**.



# Tìm theo chiều sâu (Depth-First Search)

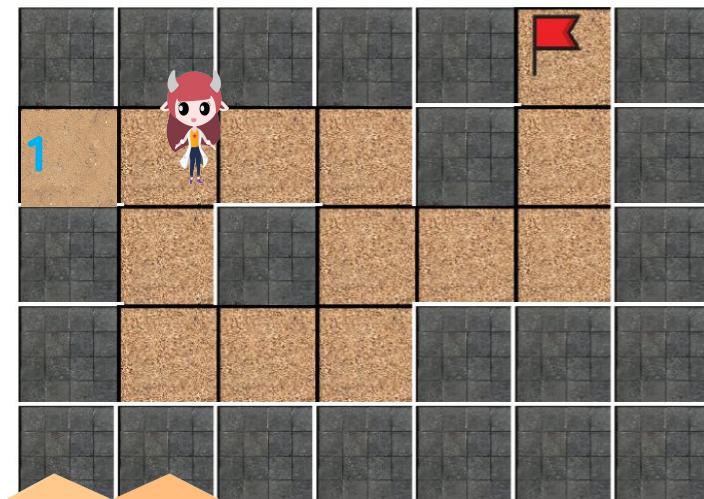
## Thuật toán

**Bước 1** Đến điểm xuất phát

**Bước 2** Dùng phấn đánh số vị trí hiện tại  
*(Không dùng các số trùng nhau)*

Nhìn xung quanh theo thứ tự: **Trên - dưới - trái - phải** xem có đường nào chưa đánh dấu không. **Nếu có**, sang **bước 4**.

**Bước 4** Đi sang ô tiếp theo, ghi lại số của ô cũ rồi bỏ vào hộp. Quay lại **bước 2**



# Tìm theo chiều sâu (Depth-First Search)

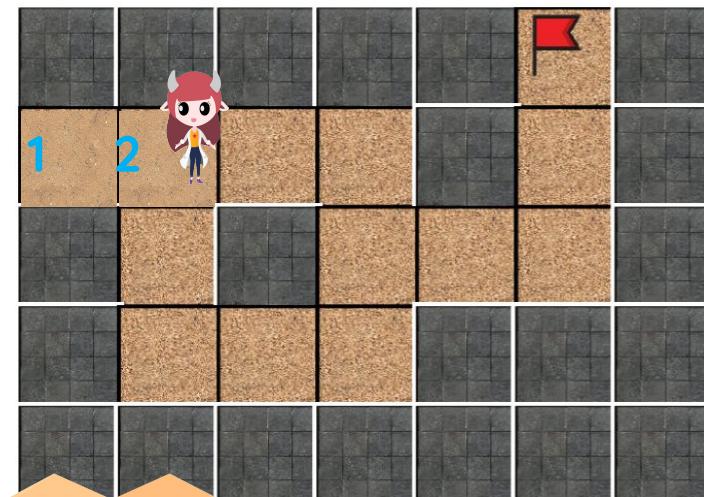
## Thuật toán

**Bước 1** Đến điểm xuất phát

**Bước 2** Dùng phấn đánh số vị trí hiện tại  
*(Không dùng các số trùng nhau)*

Nhìn xung quanh theo thứ tự: **Trên - dưới - trái - phải** xem có đường nào chưa đánh dấu không. **Nếu có**, sang **bước 4**.

**Bước 3** Đi sang ô tiếp theo, ghi lại số của ô cũ rồi bỏ vào hộp. Quay lại **bước 2**



# Tìm theo chiều sâu (Depth-First Search)

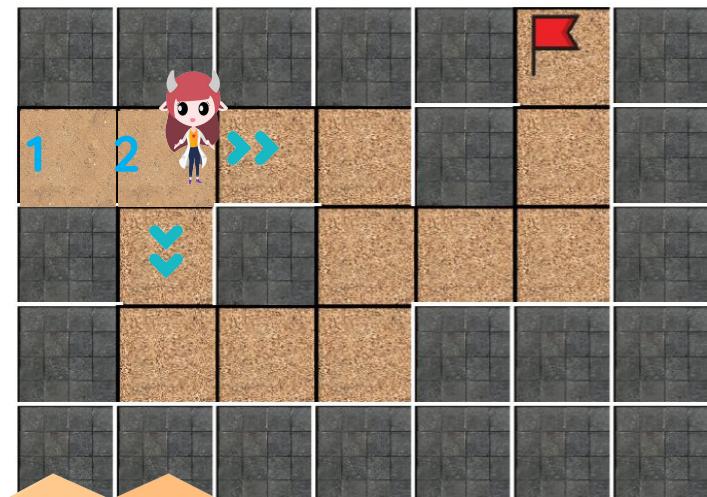
## Thuật toán

**Bước 1** Đến điểm xuất phát

**Bước 2** Dùng phấn đánh số vị trí hiện tại  
*(Không dùng các số trùng nhau)*

**Bước 3** Nhìn xung quanh theo thứ tự: **Trên - dưới - trái - phải** xem có đường nào chưa đánh dấu không. **Nếu có**, sang **bước 4**.

**Bước 4** Đi sang ô tiếp theo, ghi lại số của ô cũ rồi bỏ vào hộp. Quay lại **bước 2**



# Tìm theo chiều sâu (Depth-First Search)

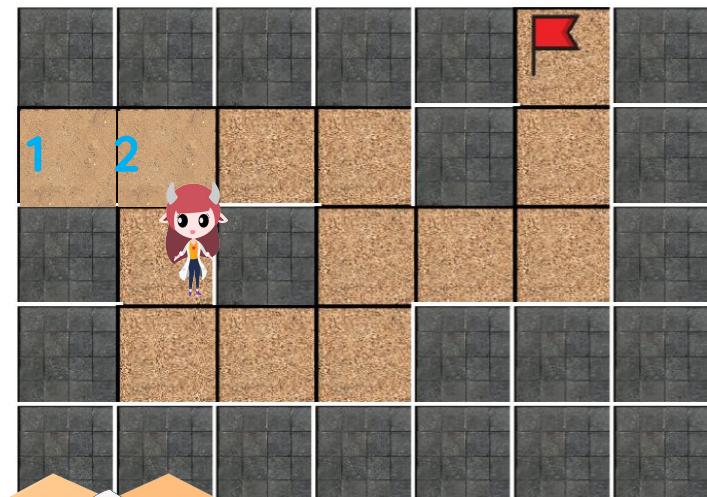
## Thuật toán

**Bước 1** Đến điểm xuất phát

**Bước 2** Dùng phấn đánh số vị trí hiện tại  
*(Không dùng các số trùng nhau)*

Nhìn xung quanh theo thứ tự: **Trên - dưới - trái - phải** xem có đường nào chưa đánh dấu không. **Nếu có**, sang **bước 4**.

**Bước 4** Đi sang ô tiếp theo, ghi lại số của ô cũ rồi bỏ vào hộp. Quay lại **bước 2**



# Tìm theo chiều sâu (Depth-First Search)

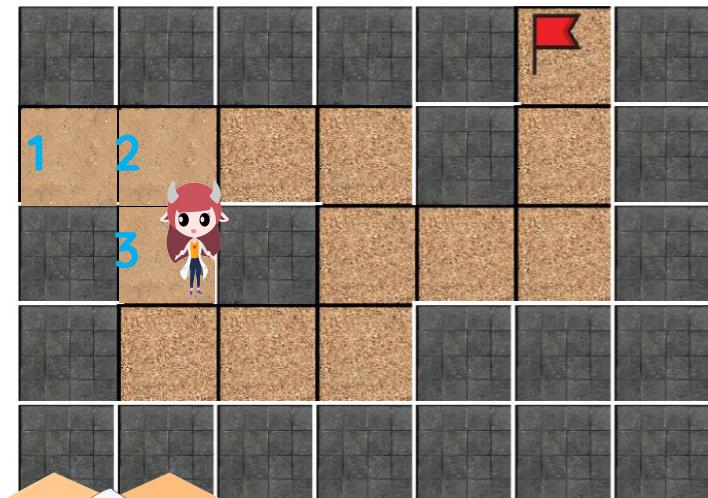
## Thuật toán

**Bước 1** Đến điểm xuất phát

**Bước 2** Dùng phấn đánh số vị trí hiện tại  
*(Không dùng các số trùng nhau)*

**Bước 3** Nhìn xung quanh theo thứ tự: **Trên - dưới - trái - phải** xem có đường nào chưa đánh dấu không. **Nếu có**, sang **bước 4**.

**Bước 4** Đi sang ô tiếp theo, ghi lại số của ô cũ rồi bỏ vào hộp. Quay lại **bước 2**



2

# Tìm theo chiều sâu (Depth-First Search)

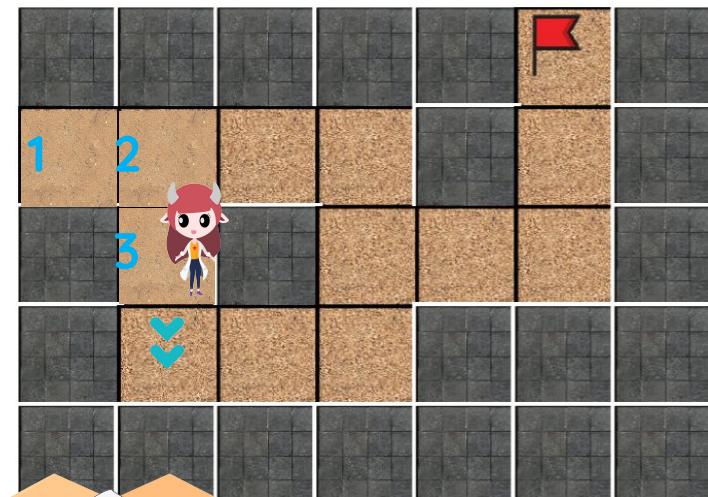
## Thuật toán

**Bước 1** Đến điểm xuất phát

**Bước 2** Dùng phấn đánh số vị trí hiện tại  
*(Không dùng các số trùng nhau)*

**Bước 3** Nhìn xung quanh theo thứ tự: **Trên - dưới - trái - phải** xem có đường nào chưa đánh dấu không. **Nếu có**, sang **bước 4**.

**Bước 4** Đi sang ô tiếp theo, ghi lại số của ô cũ rồi bỏ vào hộp. Quay lại **bước 2**



# Tìm theo chiều sâu (Depth-First Search)

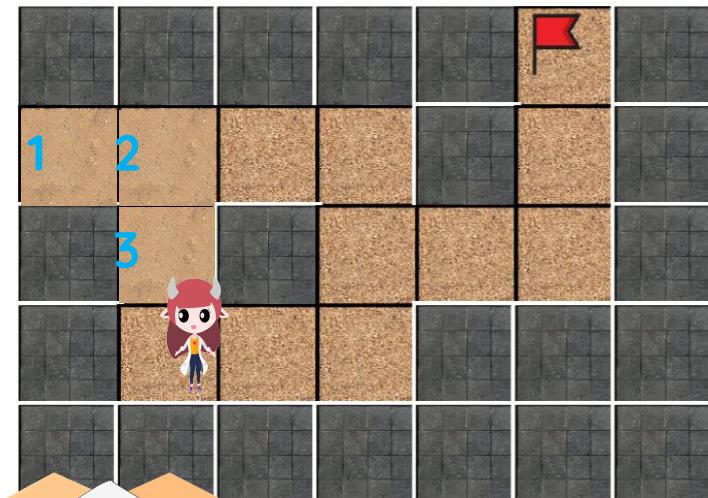
## Thuật toán

**Bước 1** Đến điểm xuất phát

**Bước 2** Dùng phấn đánh số vị trí hiện tại  
*(Không dùng các số trùng nhau)*

**Bước 3** Nhìn xung quanh theo thứ tự: **Trên - dưới - trái - phải** xem có đường nào chưa đánh dấu không. **Nếu có**, sang **bước 4**.

**Bước 4** Đi sang ô tiếp theo, ghi lại số của ô cũ rồi bỏ vào hộp. Quay lại **bước 2**



3



# Tìm theo chiều sâu (Depth-First Search)

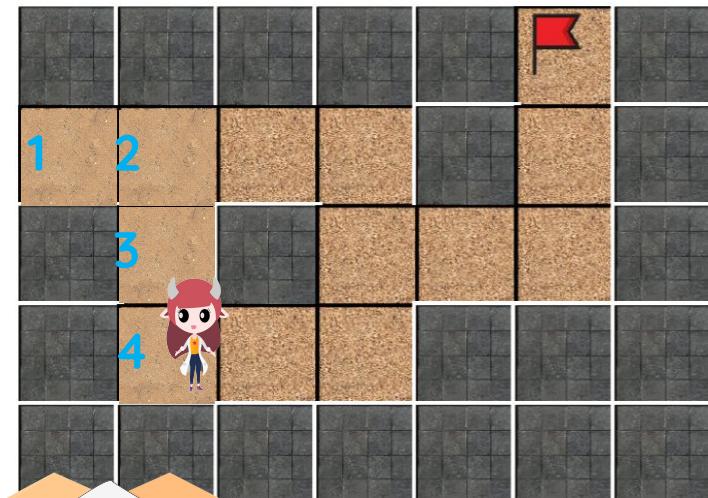
## Thuật toán

**Bước 1** Đến điểm xuất phát

**Bước 2** Dùng phấn đánh số vị trí hiện tại  
(Không dùng các số trùng nhau)

**Bước 3** Nhìn xung quanh theo thứ tự: **Trên - dưới - trái - phải** xem có đường nào chưa đánh dấu không. **Nếu có**, sang **bước 4**.

**Bước 4** Đi sang ô tiếp theo, ghi lại số của ô cũ rồi bỏ vào hộp. Quay lại **bước 2**



3



# Tìm theo chiều sâu (Depth-First Search)

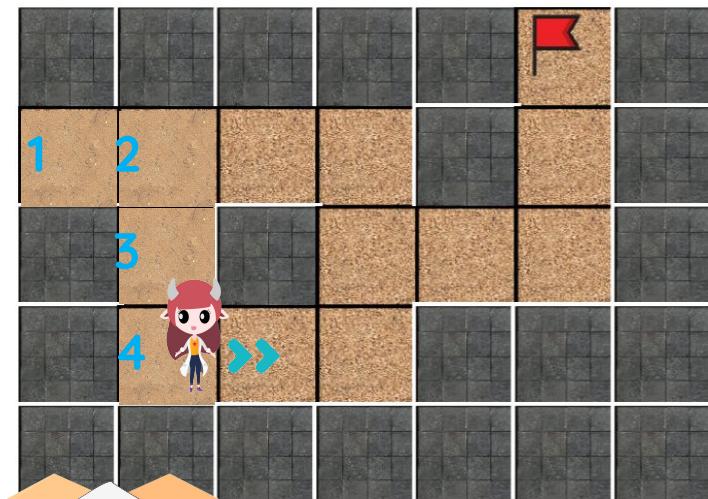
## Thuật toán

**Bước 1** Đến điểm xuất phát

**Bước 2** Dùng phấn đánh số vị trí hiện tại  
*(Không dùng các số trùng nhau)*

**Bước 3** Nhìn xung quanh theo thứ tự: **Trên - dưới - trái - phải** xem có đường nào chưa đánh dấu không. **Nếu có**, sang **bước 4**.

**Bước 4** Đi sang ô tiếp theo, ghi lại số của ô cũ rồi bỏ vào hộp. Quay lại **bước 2**



3



# Tìm theo chiều sâu (Depth-First Search)

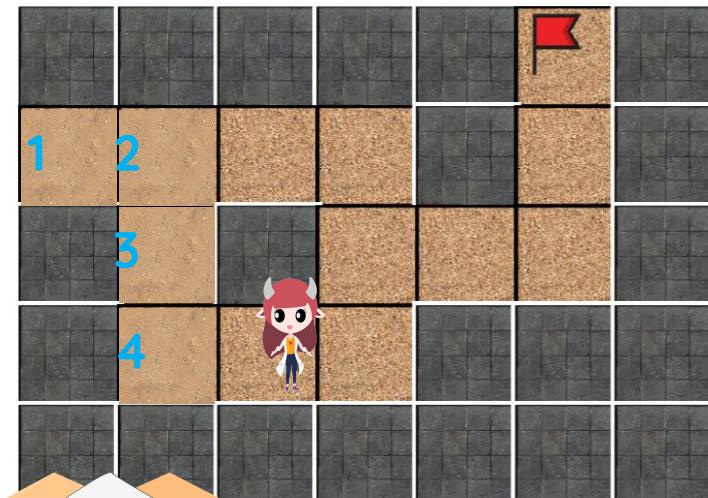
## Thuật toán

**Bước 1** Đến điểm xuất phát

**Bước 2** Dùng phấn đánh số vị trí hiện tại  
*(Không dùng các số trùng nhau)*

**Bước 3** Nhìn xung quanh theo thứ tự: **Trên - dưới - trái - phải** xem có đường nào chưa đánh dấu không. **Nếu có**, sang **bước 4**.

**Bước 4** Đi sang ô tiếp theo, ghi lại số của ô cũ rồi bỏ vào hộp. Quay lại **bước 2**



# Tìm theo chiều sâu (Depth-First Search)

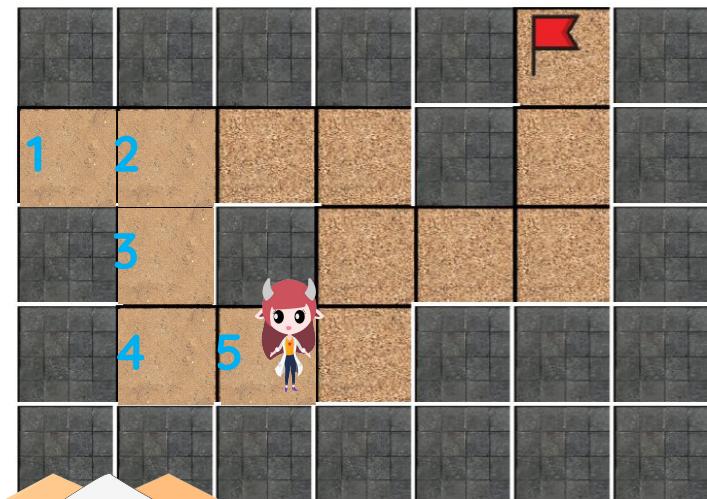
## Thuật toán

**Bước 1** Đến điểm xuất phát

**Bước 2** Dùng phấn đánh số vị trí hiện tại  
(Không dùng các số trùng nhau)

**Bước 3** Nhìn xung quanh theo thứ tự: **Trên - dưới - trái - phải** xem có đường nào chưa đánh dấu không. **Nếu có**, sang **bước 4**.

**Bước 4** Đi sang ô tiếp theo, ghi lại số của ô cũ rồi bỏ vào hộp. Quay lại **bước 2**



# Tìm theo chiều sâu (Depth-First Search)

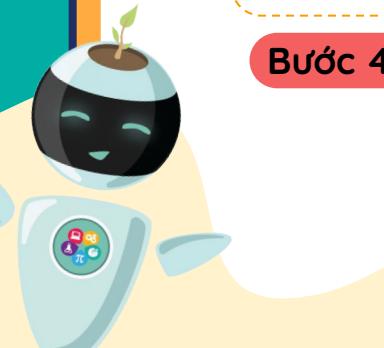
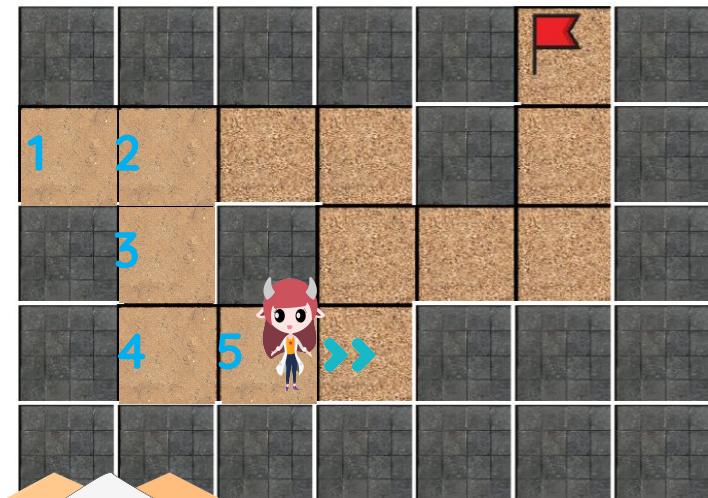
## Thuật toán

**Bước 1** Đến điểm xuất phát

**Bước 2** Dùng phấn đánh số vị trí hiện tại  
*(Không dùng các số trùng nhau)*

**Bước 3** Nhìn xung quanh theo thứ tự: **Trên - dưới - trái - phải** xem có đường nào chưa đánh dấu không. **Nếu có**, sang **bước 4**.

**Bước 4** Đi sang ô tiếp theo, ghi lại số của ô cũ rồi bỏ vào hộp. Quay lại **bước 2**



# Tìm theo chiều sâu (Depth-First Search)

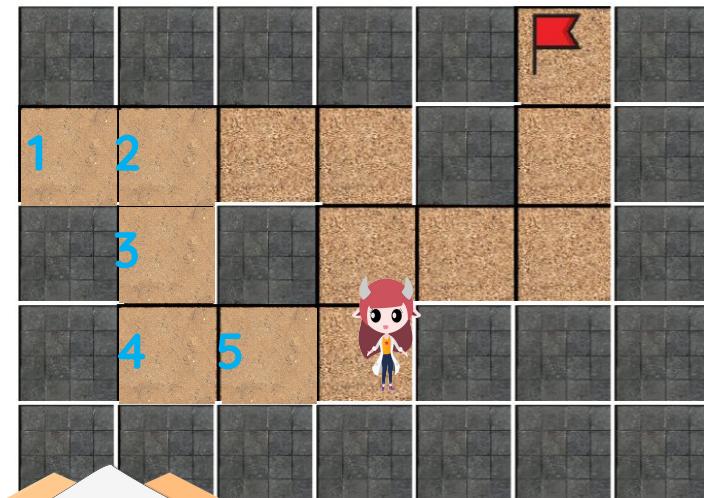
## Thuật toán

**Bước 1** Đến điểm xuất phát

**Bước 2** Dùng phấn đánh số vị trí hiện tại  
*(Không dùng các số trùng nhau)*

**Bước 3** Nhìn xung quanh theo thứ tự: **Trên - dưới - trái - phải** xem có đường nào chưa đánh dấu không. **Nếu có**, sang **bước 4**.

**Bước 4** Đi sang ô tiếp theo, ghi lại số của ô cũ rồi bỏ vào hộp. Quay lại **bước 2**



5



# Tìm theo chiều sâu (Depth-First Search)

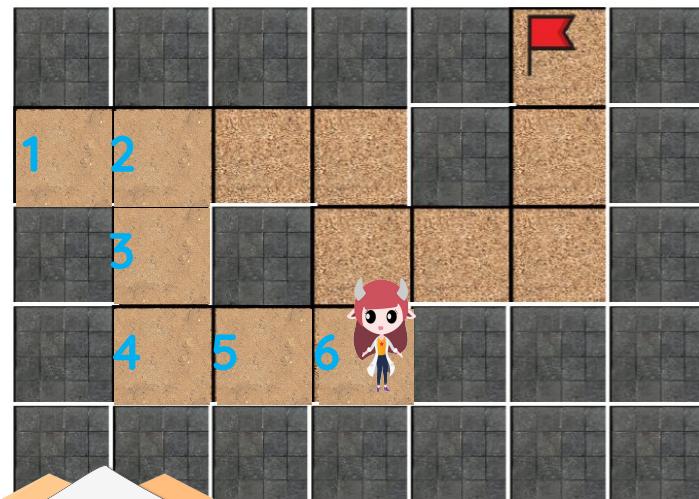
# Thuật toán

- Bước 1** Đến điểm xuất phát

**Bước 2** Dùng phấn đánh số vị trí hiện tại  
**(Không dùng các số trùng nhau)**

**Bước 3** Nhìn xung quanh theo thứ tự: **Trên - dưới - trái - phải** xem có đường nào chưa đánh dấu không. **Nếu có**, sang **bước 4**.

**Bước 4** Di sang ô tiếp theo, ghi lại số của ô cũ rồi bỏ vào hộp. Quay lại **bước 2**



# Tìm theo chiều sâu (Depth-First Search)

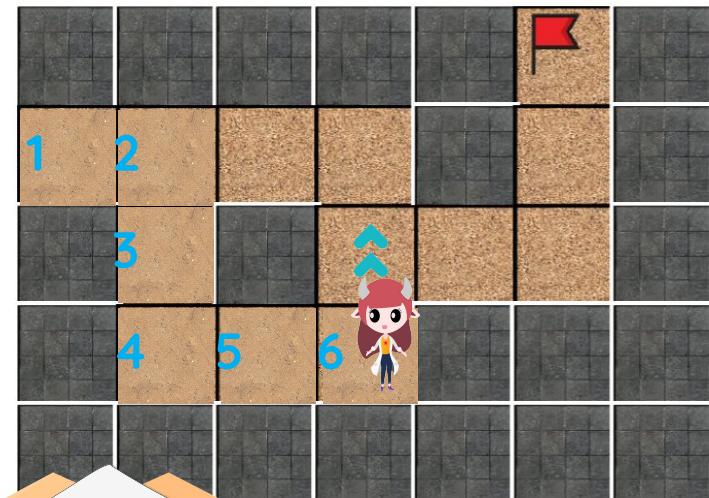
## Thuật toán

**Bước 1** Đến điểm xuất phát

**Bước 2** Dùng phấn đánh số vị trí hiện tại  
*(Không dùng các số trùng nhau)*

**Bước 3** Nhìn xung quanh theo thứ tự: **Trên - dưới - trái - phải** xem có đường nào chưa đánh dấu không. **Nếu có**, sang **bước 4**.

**Bước 4** Đi sang ô tiếp theo, ghi lại số của ô cũ rồi bỏ vào hộp. Quay lại **bước 2**



5



# Tìm theo chiều sâu (Depth-First Search)

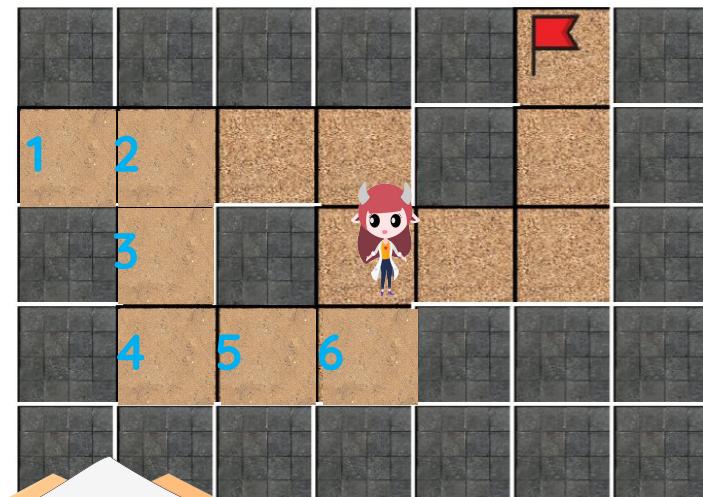
# Thuật toán

- Bước 1** Đến điểm xuất phát

**Bước 2** Dùng phấn đánh số vị trí hiện tại  
**(Không dùng các số trùng nhau)**

**Bước 3** Nhìn xung quanh theo thứ tự: **Trên - dưới - trái - phải** xem có đường nào chưa đánh dấu không. **Nếu có**, sang **bước 4**.

**Bước 4** Di sang ô tiếp theo, ghi lại số của ô cũ rồi bỏ vào hộp. Quay lại **bước 2**



# Tìm theo chiều sâu (Depth-First Search)

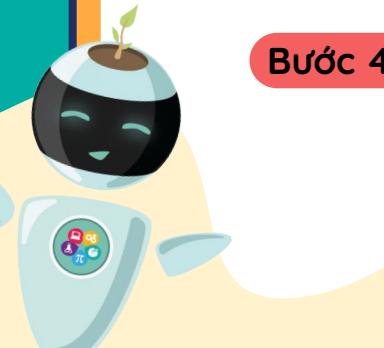
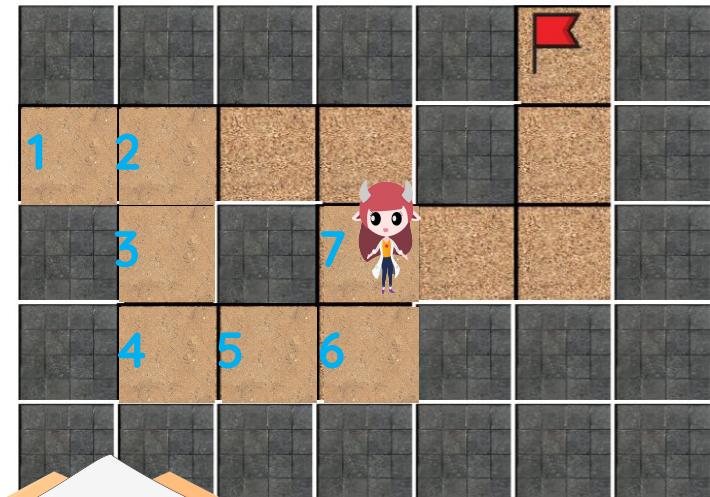
## Thuật toán

**Bước 1** Đến điểm xuất phát

**Bước 2** Dùng phấn đánh số vị trí hiện tại  
(Không dùng các số trùng nhau)

**Bước 3** Nhìn xung quanh theo thứ tự: **Trên - dưới - trái - phải** xem có đường nào chưa đánh dấu không. **Nếu có**, sang **bước 4**.

**Bước 4** Đi sang ô tiếp theo, ghi lại số của ô cũ rồi bỏ vào hộp. Quay lại **bước 2**



# Tìm theo chiều sâu (Depth-First Search)

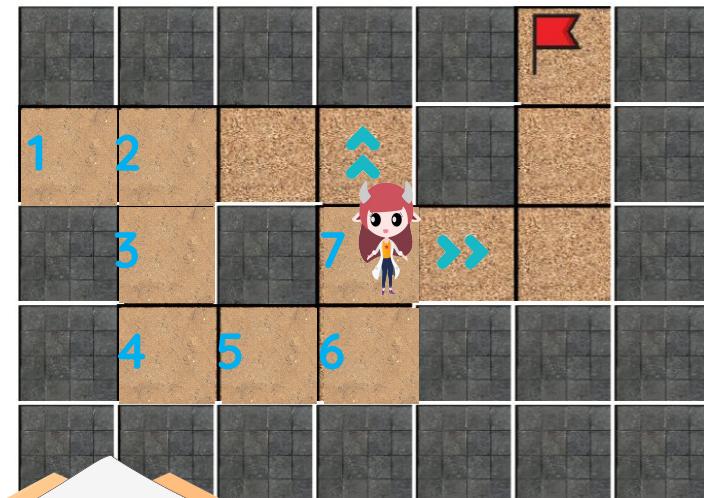
## Thuật toán

**Bước 1** Đến điểm xuất phát

**Bước 2** Dùng phấn đánh số vị trí hiện tại  
*(Không dùng các số trùng nhau)*

**Bước 3** Nhìn xung quanh theo thứ tự: **Trên - dưới - trái - phải** xem có đường nào chưa đánh dấu không. **Nếu có**, sang **bước 4**.

**Bước 4** Đi sang ô tiếp theo, ghi lại số của ô cũ rồi bỏ vào hộp. Quay lại **bước 2**



# Tìm theo chiều sâu (Depth-First Search)

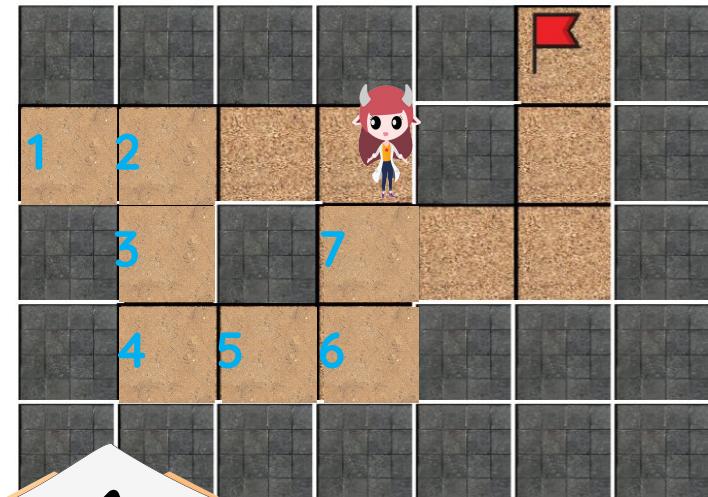
## Thuật toán

**Bước 1** Đến điểm xuất phát

**Bước 2** Dùng phấn đánh số vị trí hiện tại  
*(Không dùng các số trùng nhau)*

**Bước 3** Nhìn xung quanh theo thứ tự: **Trên - dưới - trái - phải** xem có đường nào chưa đánh dấu không. **Nếu có**, sang **bước 4**.

**Bước 4** Đi sang ô tiếp theo, ghi lại số của ô cũ rồi bỏ vào hộp. Quay lại **bước 2**



# Tìm theo chiều sâu (Depth-First Search)

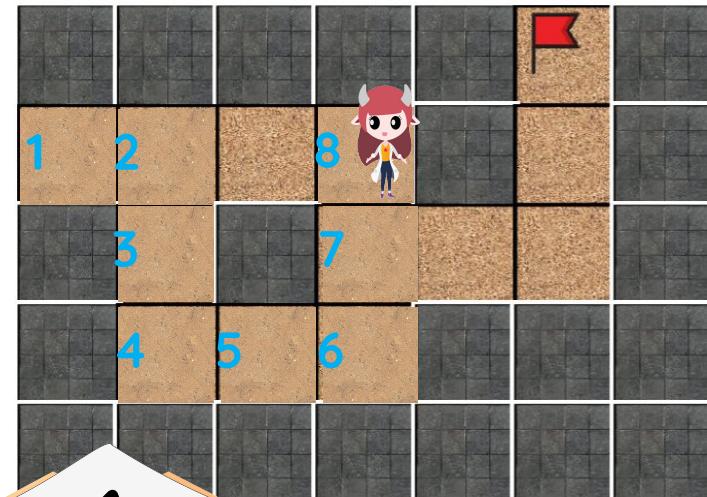
## Thuật toán

**Bước 1** Đến điểm xuất phát

**Bước 2** Dùng phấn đánh số vị trí hiện tại  
(Không dùng các số trùng nhau)

**Bước 3** Nhìn xung quanh theo thứ tự: **Trên - dưới - trái - phải** xem có đường nào chưa đánh dấu không. **Nếu có**, sang **bước 4**.

**Bước 4** Đi sang ô tiếp theo, ghi lại số của ô cũ rồi bỏ vào hộp. Quay lại **bước 2**



# Tìm theo chiều sâu (Depth-First Search)

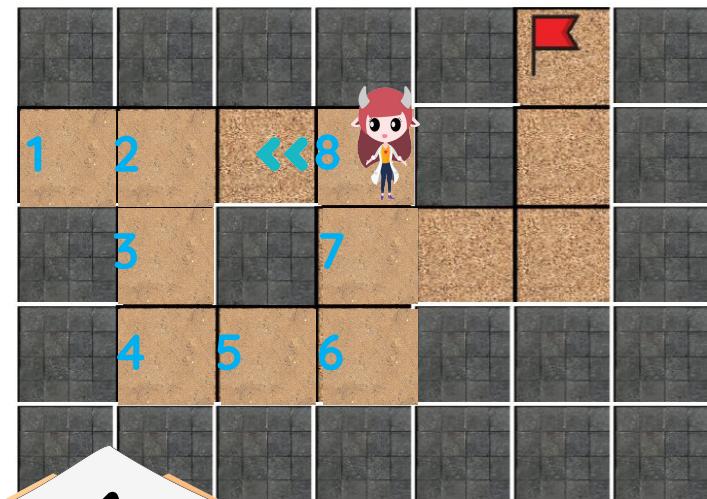
## Thuật toán

**Bước 1** Đến điểm xuất phát

**Bước 2** Dùng phấn đánh số vị trí hiện tại  
*(Không dùng các số trùng nhau)*

**Bước 3** Nhìn xung quanh theo thứ tự: **Trên - dưới - trái - phải** xem có đường nào chưa đánh dấu không. **Nếu có**, sang **bước 4**.

**Bước 4** Đi sang ô tiếp theo, ghi lại số của ô cũ rồi bỏ vào hộp. Quay lại **bước 2**



# Tìm theo chiều sâu (Depth-First Search)

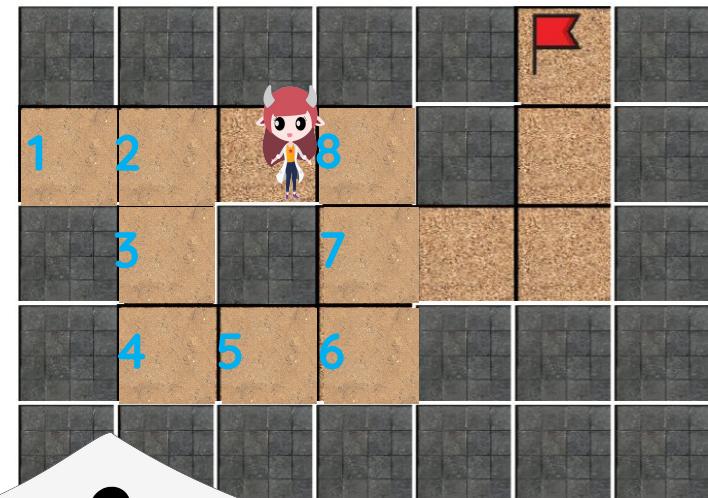
## Thuật toán

**Bước 1** Đến điểm xuất phát

**Bước 2** Dùng phấn đánh số vị trí hiện tại  
*(Không dùng các số trùng nhau)*

**Bước 3** Nhìn xung quanh theo thứ tự: **Trên - dưới - trái - phải** xem có đường nào chưa đánh dấu không. **Nếu có**, sang **bước 4**.

**Bước 4** Đi sang ô tiếp theo, ghi lại số của ô cũ rồi bỏ vào hộp. Quay lại **bước 2**



# Tìm theo chiều sâu (Depth-First Search)

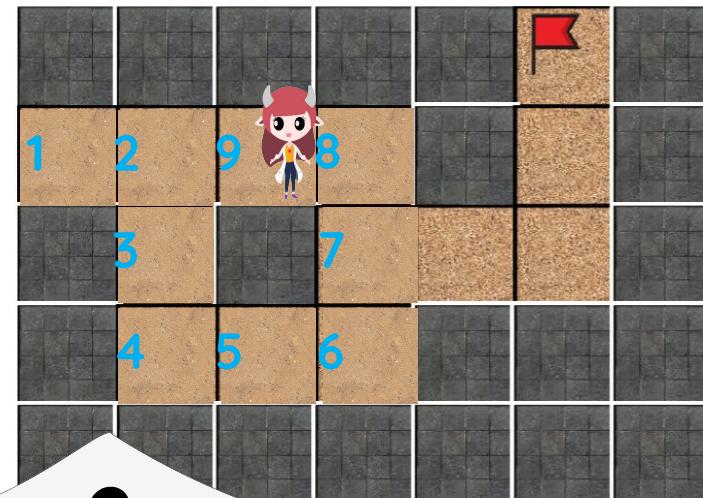
## Thuật toán

**Bước 1** Đến điểm xuất phát

**Bước 2** Dùng phấn đánh số vị trí hiện tại  
(Không dùng các số trùng nhau)

Nhìn xung quanh theo thứ tự: **Trên - dưới - trái - phải** xem có đường nào chưa đánh dấu không. **Nếu có**, sang **bước 4**.

**Bước 3** Đi sang ô tiếp theo, ghi lại số của ô cũ rồi bỏ vào hộp. Quay lại **bước 2**



# Tìm theo chiều sâu (Depth-First Search)

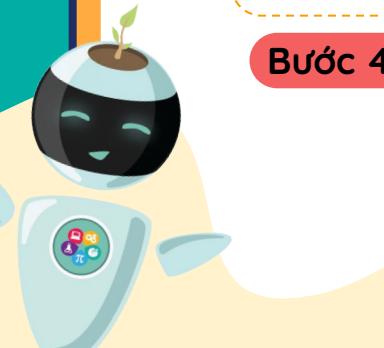
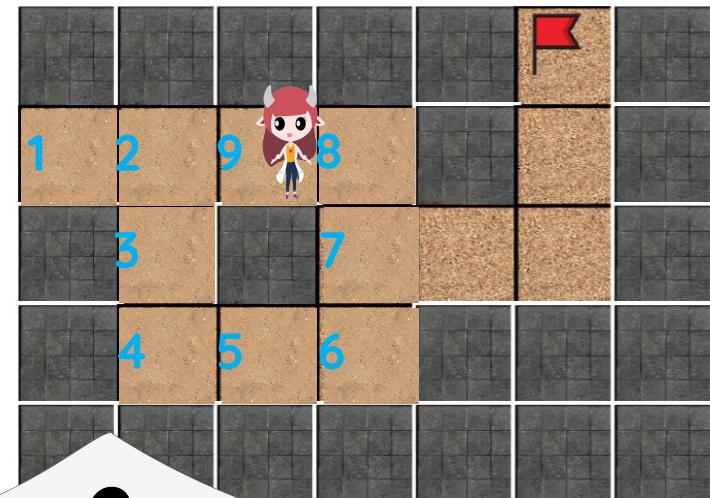
## Thuật toán

**Bước 1** Đến điểm xuất phát

**Bước 2** Dùng phấn đánh số vị trí hiện tại  
*(Không dùng các số trùng nhau)*

**Bước 3** Nhìn xung quanh theo thứ tự: **Trên - dưới - trái - phải** xem có đường nào chưa đánh dấu không. **Nếu có**, sang **bước 4**.

**Bước 4** Đi sang ô tiếp theo, ghi lại số của ô cũ rồi bỏ vào hộp. Quay lại **bước 2**



# Tìm theo chiều sâu (Depth-First Search)

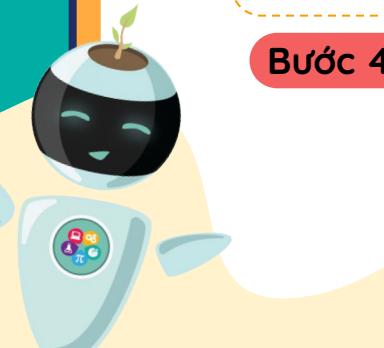
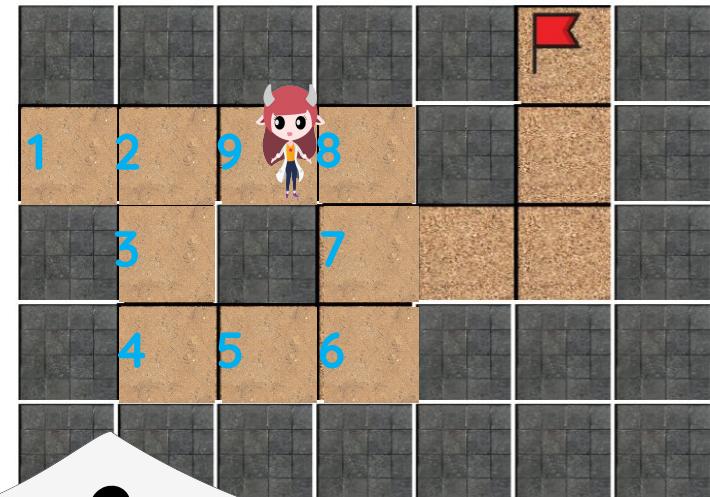
## Thuật toán

**Bước 1** Đến điểm xuất phát

**Bước 2** Dùng phấn đánh số vị trí hiện tại  
*(Không dùng các số trùng nhau)*

**Bước 3** Nhìn xung quanh theo thứ tự: **Trên - dưới - trái - phải** xem có đường nào chưa đánh dấu không. **Nếu có**, sang **bước 4**. **Nếu không**, sang **bước 5**.

**Bước 4** Đi sang ô tiếp theo, ghi lại số của ô cũ rồi bỏ vào hộp. Quay lại **bước 2**



# Tìm theo chiều sâu (Depth-First Search)

## Thuật toán

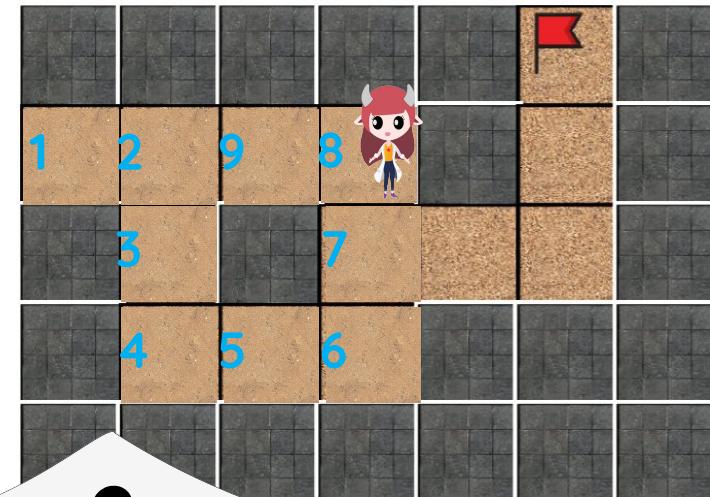
**Bước 1** Đến điểm xuất phát

**Bước 2** Dùng phấn đánh số vị trí hiện tại  
(Không dùng các số trùng nhau)

**Bước 3** Nhìn xung quanh theo thứ tự: **Trên - dưới - trái - phải** xem có đường nào chưa đánh dấu không. **Nếu có**, sang **bước 4**. **Nếu không**, sang **bước 5**.

**Bước 4** Đi sang ô tiếp theo, ghi lại số của ô cũ rồi bỏ vào hộp. Quay lại **bước 2**

**Bước 5** Lấy 1 tờ giấy trong hộp, quay lại vị trí trong tờ giấy. Quay lại **bước 3**.



# Tìm theo chiều sâu (Depth-First Search)

## Thuật toán

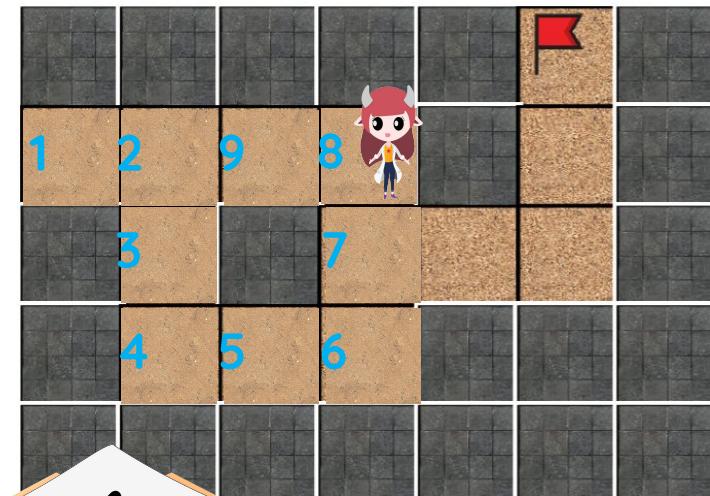
**Bước 1** Đến điểm xuất phát

**Bước 2** Dùng phấn đánh số vị trí hiện tại  
*(Không dùng các số trùng nhau)*

**Bước 3** Nhìn xung quanh theo thứ tự: **Trên - dưới - trái - phải** xem có đường nào chưa đánh dấu không. **Nếu có**, sang **bước 4**. **Nếu không**, sang **bước 5**.

**Bước 4** Đi sang ô tiếp theo, ghi lại số của ô cũ rồi bỏ vào hộp. Quay lại **bước 2**

**Bước 5** Lấy 1 tờ giấy trong hộp, quay lại vị trí trong tờ giấy. Quay lại **bước 3**.



# Tìm theo chiều sâu (Depth-First Search)

## Thuật toán

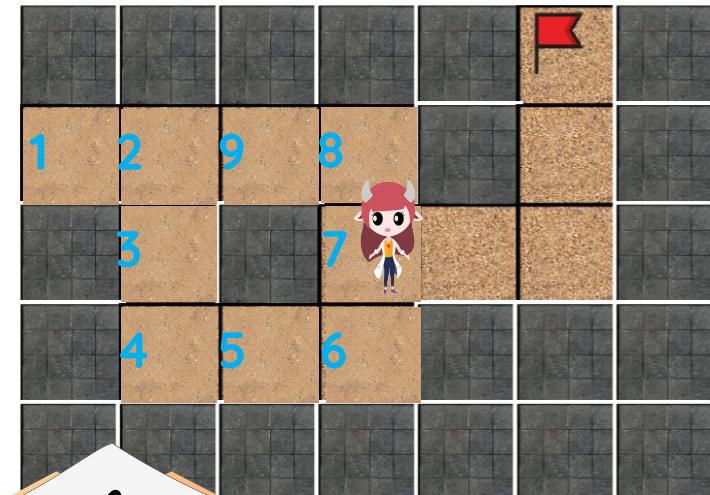
**Bước 1** Đến điểm xuất phát

**Bước 2** Dùng phấn đánh số vị trí hiện tại  
(Không dùng các số trùng nhau)

**Bước 3** Nhìn xung quanh theo thứ tự: **Trên - dưới - trái - phải** xem có đường nào chưa đánh dấu không. **Nếu có**, sang **bước 4**. **Nếu không**, sang **bước 5**.

**Bước 4** Đi sang ô tiếp theo, ghi lại số của ô cũ rồi bỏ vào hộp. Quay lại **bước 2**

**Bước 5** Lấy 1 tờ giấy trong hộp, quay lại vị trí trong tờ giấy. Quay lại **bước 3**.



# Tìm theo chiều sâu (Depth-First Search)

## Thuật toán

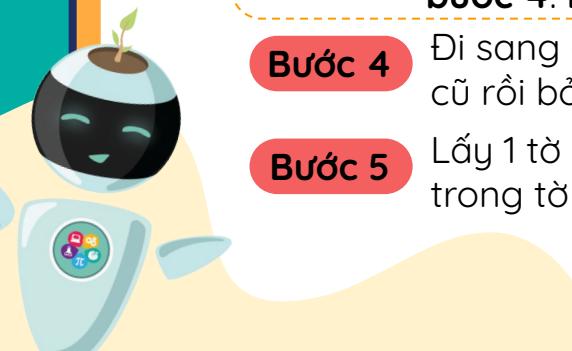
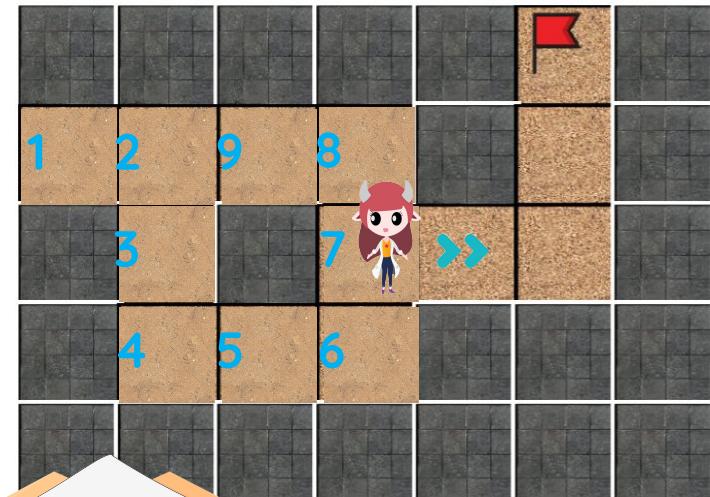
**Bước 1** Đến điểm xuất phát

**Bước 2** Dùng phấn đánh số vị trí hiện tại  
(Không dùng các số trùng nhau)

**Bước 3** Nhìn xung quanh theo thứ tự: **Trên - dưới - trái - phải** xem có đường nào chưa đánh dấu không. **Nếu có**, sang **bước 4**. **Nếu không**, sang **bước 5**.

**Bước 4** Đi sang ô tiếp theo, ghi lại số của ô cũ rồi bỏ vào hộp. Quay lại **bước 2**

**Bước 5** Lấy 1 tờ giấy trong hộp, quay lại vị trí trong tờ giấy. Quay lại **bước 3**.



# Tìm theo chiều sâu (Depth-First Search)

## Thuật toán

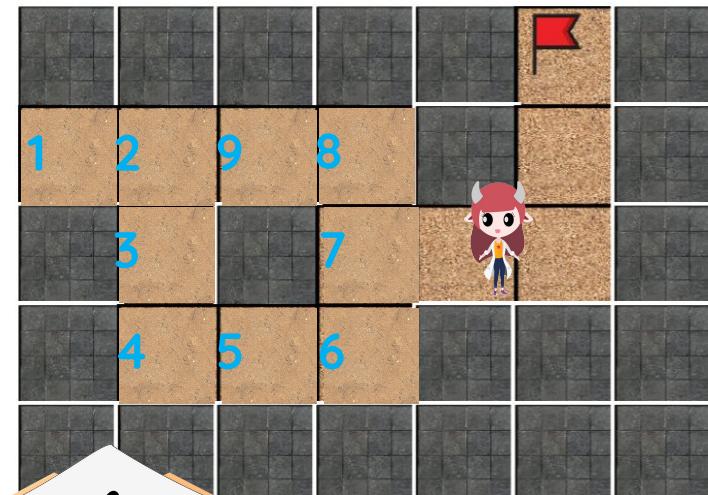
**Bước 1** Đến điểm xuất phát

**Bước 2** Dùng phấn đánh số vị trí hiện tại  
*(Không dùng các số trùng nhau)*

**Bước 3** Nhìn xung quanh theo thứ tự: **Trên - dưới - trái - phải** xem có đường nào chưa đánh dấu không. **Nếu có**, sang **bước 4**. **Nếu không**, sang **bước 5**.

**Bước 4** Đi sang ô tiếp theo, ghi lại số của ô cũ rồi bỏ vào hộp. Quay lại **bước 2**

**Bước 5** Lấy 1 tờ giấy trong hộp, quay lại vị trí trong tờ giấy. Quay lại **bước 3**.



# Tìm theo chiều sâu (Depth-First Search)

## Thuật toán

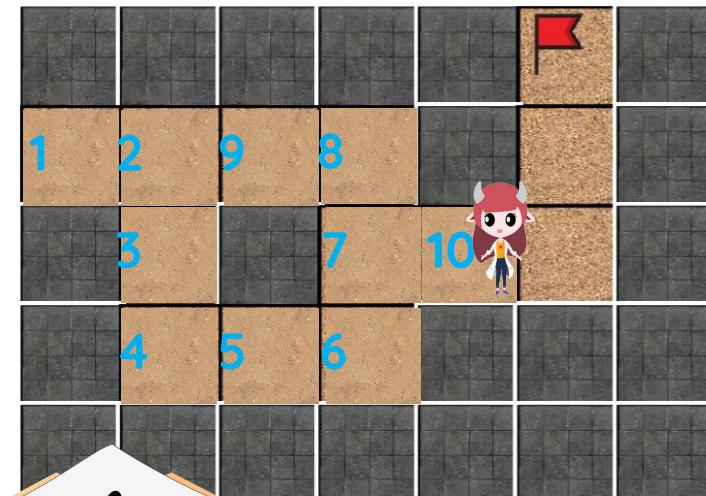
**Bước 1** Đến điểm xuất phát

**Bước 2** Dùng phấn đánh số vị trí hiện tại  
(Không dùng các số trùng nhau)

**Bước 3** Nhìn xung quanh theo thứ tự: **Trên - dưới - trái - phải** xem có đường nào chưa đánh dấu không. **Nếu có**, sang **bước 4**. **Nếu không**, sang **bước 5**.

**Bước 4** Đi sang ô tiếp theo, ghi lại số của ô cũ rồi bỏ vào hộp. Quay lại **bước 2**

**Bước 5** Lấy 1 tờ giấy trong hộp, quay lại vị trí trong tờ giấy. Quay lại **bước 3**.



# Tìm theo chiều sâu (Depth-First Search)

## Thuật toán

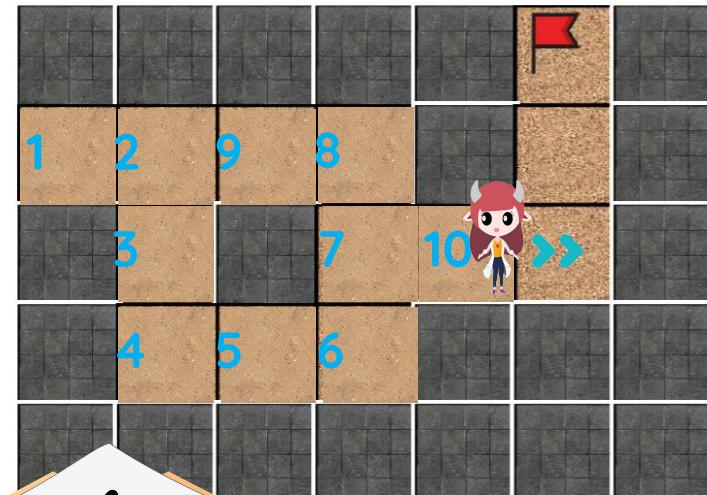
**Bước 1** Đến điểm xuất phát

**Bước 2** Dùng phấn đánh số vị trí hiện tại  
*(Không dùng các số trùng nhau)*

**Bước 3** Nhìn xung quanh theo thứ tự: **Trên - dưới - trái - phải** xem có đường nào chưa đánh dấu không. **Nếu có**, sang **bước 4**. **Nếu không**, sang **bước 5**.

**Bước 4** Đi sang ô tiếp theo, ghi lại số của ô cũ rồi bỏ vào hộp. Quay lại **bước 2**

**Bước 5** Lấy 1 tờ giấy trong hộp, quay lại vị trí trong tờ giấy. Quay lại **bước 3**.



# Tìm theo chiều sâu (Depth-First Search)

## Thuật toán

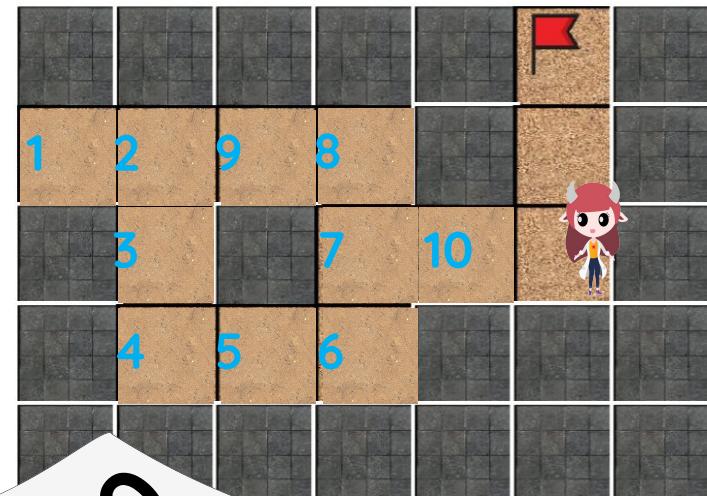
**Bước 1** Đến điểm xuất phát

**Bước 2** Dùng phấn đánh số vị trí hiện tại  
*(Không dùng các số trùng nhau)*

**Bước 3** Nhìn xung quanh theo thứ tự: **Trên - dưới - trái - phải** xem có đường nào chưa đánh dấu không. **Nếu có**, sang **bước 4**. **Nếu không**, sang **bước 5**.

**Bước 4** Đi sang ô tiếp theo, ghi lại số của ô cũ rồi bỏ vào hộp. Quay lại **bước 2**

**Bước 5** Lấy 1 tờ giấy trong hộp, quay lại vị trí trong tờ giấy. Quay lại **bước 3**.



# Tìm theo chiều sâu (Depth-First Search)

## Thuật toán

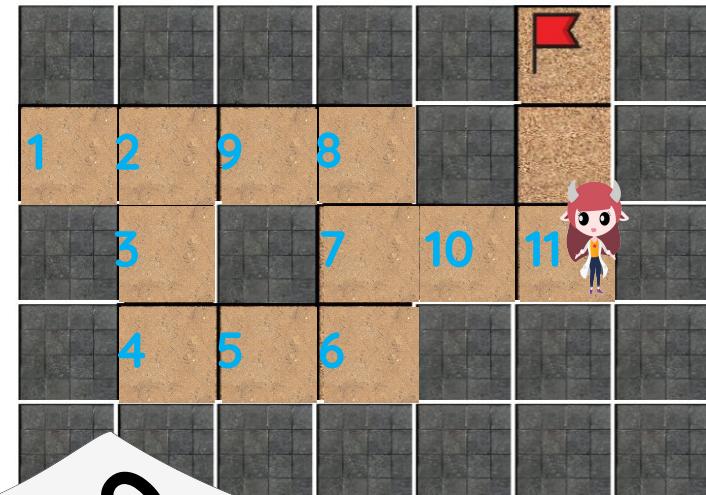
**Bước 1** Đến điểm xuất phát

**Bước 2** Dùng phấn đánh số vị trí hiện tại  
(Không dùng các số trùng nhau)

**Bước 3** Nhìn xung quanh theo thứ tự: **Trên - dưới - trái - phải** xem có đường nào chưa đánh dấu không. **Nếu có**, sang **bước 4**. **Nếu không**, sang **bước 5**.

**Bước 4** Đi sang ô tiếp theo, ghi lại số của ô cũ rồi bỏ vào hộp. Quay lại **bước 2**

**Bước 5** Lấy 1 tờ giấy trong hộp, quay lại vị trí trong tờ giấy. Quay lại **bước 3**.



# Tìm theo chiều sâu (Depth-First Search)

## Thuật toán

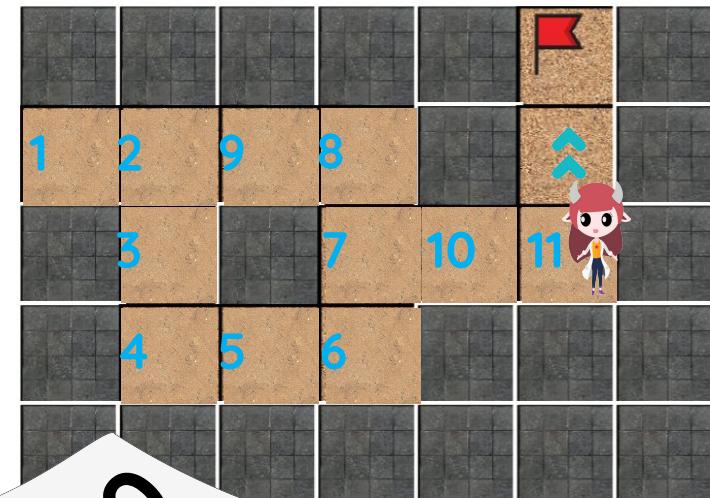
**Bước 1** Đến điểm xuất phát

**Bước 2** Dùng phấn đánh số vị trí hiện tại  
*(Không dùng các số trùng nhau)*

**Bước 3** Nhìn xung quanh theo thứ tự: **Trên - dưới - trái - phải** xem có đường nào chưa đánh dấu không. **Nếu có**, sang **bước 4**. **Nếu không**, sang **bước 5**.

**Bước 4** Đi sang ô tiếp theo, ghi lại số của ô cũ rồi bỏ vào hộp. Quay lại **bước 2**

**Bước 5** Lấy 1 tờ giấy trong hộp, quay lại vị trí trong tờ giấy. Quay lại **bước 3**.



# Tìm theo chiều sâu (Depth-First Search)

## Thuật toán

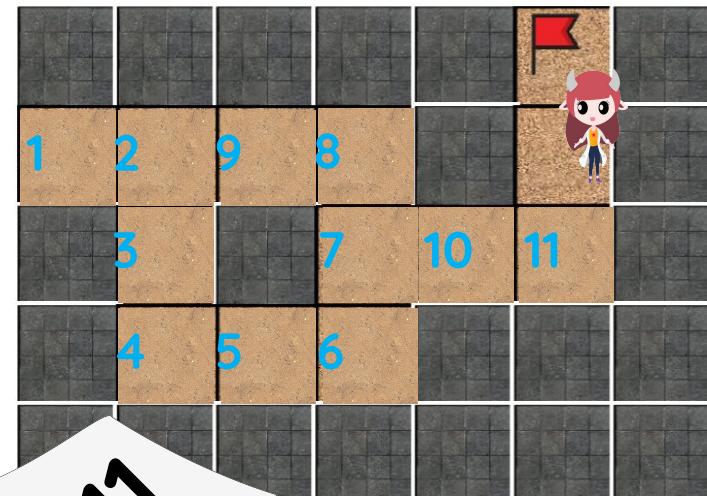
**Bước 1** Đến điểm xuất phát

**Bước 2** Dùng phấn đánh số vị trí hiện tại  
*(Không dùng các số trùng nhau)*

**Bước 3** Nhìn xung quanh theo thứ tự: **Trên - dưới - trái - phải** xem có đường nào chưa đánh dấu không. **Nếu có**, sang **bước 4**. **Nếu không**, sang **bước 5**.

**Bước 4** Đi sang ô tiếp theo, ghi lại số của ô cũ rồi bỏ vào hộp. Quay lại **bước 2**

**Bước 5** Lấy 1 tờ giấy trong hộp, quay lại vị trí trong tờ giấy. Quay lại **bước 3**.



# Tìm theo chiều sâu (Depth-First Search)

## Thuật toán

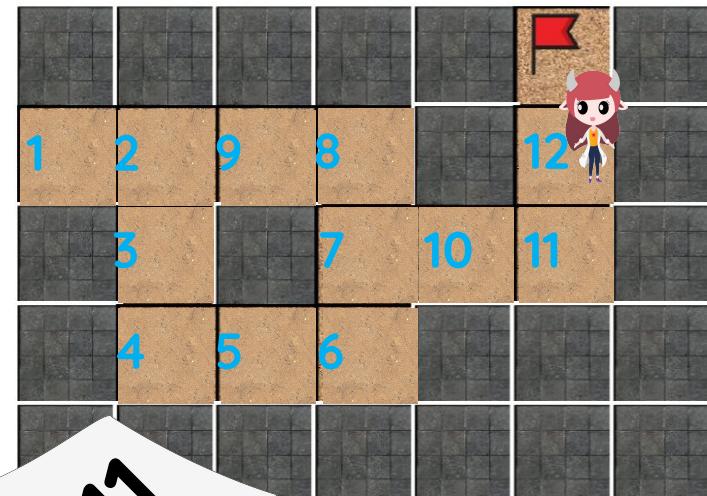
**Bước 1** Đến điểm xuất phát

**Bước 2** Dùng phấn đánh số vị trí hiện tại  
(Không dùng các số trùng nhau)

**Bước 3** Nhìn xung quanh theo thứ tự: **Trên - dưới - trái - phải** xem có đường nào chưa đánh dấu không. **Nếu có**, sang **bước 4**. **Nếu không**, sang **bước 5**.

**Bước 4** Đi sang ô tiếp theo, ghi lại số của ô cũ rồi bỏ vào hộp. Quay lại **bước 2**

**Bước 5** Lấy 1 tờ giấy trong hộp, quay lại vị trí trong tờ giấy. Quay lại **bước 3**.



# Tìm theo chiều sâu (Depth-First Search)

## Thuật toán

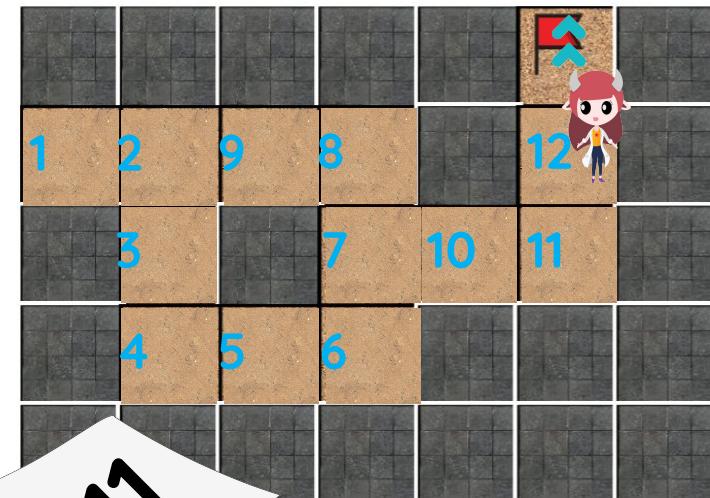
**Bước 1** Đến điểm xuất phát

**Bước 2** Dùng phấn đánh số vị trí hiện tại  
*(Không dùng các số trùng nhau)*

**Bước 3** Nhìn xung quanh theo thứ tự: **Trên - dưới - trái - phải** xem có đường nào chưa đánh dấu không. **Nếu có**, sang **bước 4**. **Nếu không**, sang **bước 5**.

**Bước 4** Đi sang ô tiếp theo, ghi lại số của ô cũ rồi bỏ vào hộp. Quay lại **bước 2**

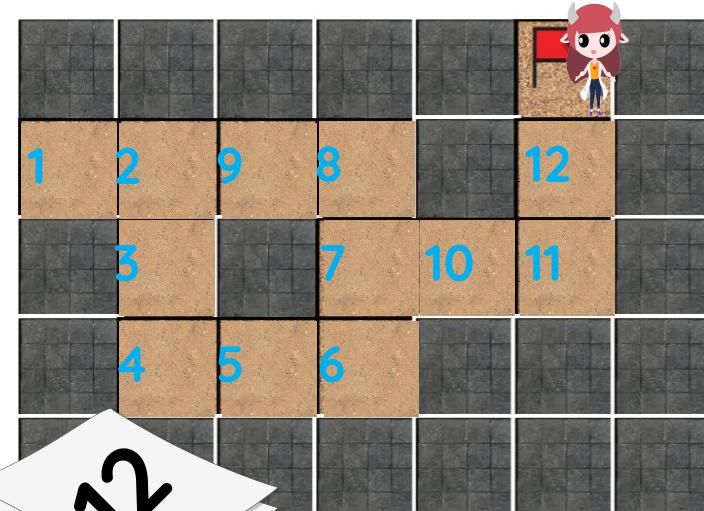
**Bước 5** Lấy 1 tờ giấy trong hộp, quay lại vị trí trong tờ giấy. Quay lại **bước 3**.



# Tìm theo chiều sâu (Depth-First Search)

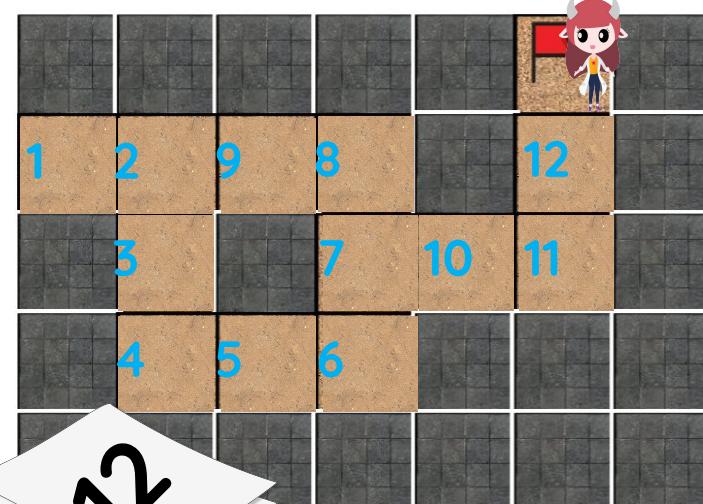
## Thuật toán

- Bước 1** Đến điểm xuất phát
- Bước 2** Dùng phấn đánh số vị trí hiện tại  
*(Không dùng các số trùng nhau)*
- Bước 3** Nhìn xung quanh theo thứ tự: **Trên - dưới - trái - phải** xem có đường nào chưa đánh dấu không. **Nếu có**, sang **bước 4**. **Nếu không**, sang **bước 5**.
- Bước 4** Đi sang ô tiếp theo, ghi lại số của ô cũ rồi bỏ vào hộp. Quay lại **bước 2**
- Bước 5** Lấy 1 tờ giấy trong hộp, quay lại vị trí trong tờ giấy. Quay lại **bước 3**.



# Tìm theo chiều sâu (Depth-First Search)

Tada! Vậy chúng ta  
đã đến đích rồi!





## Quiz #1

**Đâu KHÔNG phải là yêu cầu bắt buộc để thực hiện tìm kiếm theo chiều sâu?**

- a. Không được phép đánh số trùng nhau giữa hai ô
- b. Khi đi sang ô tiếp theo, bỏ tờ giấy của ô cũ vào hộp
- c. Luôn luôn phải đi theo thứ tự trên - dưới - trái - phải
- d. Khi gặp đường cùt, lấy một tờ giấy ở trong hộp ra





## Quiz #1

Đâu KHÔNG phải là yêu cầu bắt buộc để thực hiện tìm kiếm theo chiều sâu?

- a. Không được phép đánh số trùng nhau giữa hai ô
- b. Khi đi sang ô tiếp theo, bỏ tờ giấy của ô cũ vào hộp
- c. Luôn luôn phải đi theo thứ tự trên - dưới - trái - phải**
- d. Khi gặp đường cùt, lấy một tờ giấy ở trong hộp ra



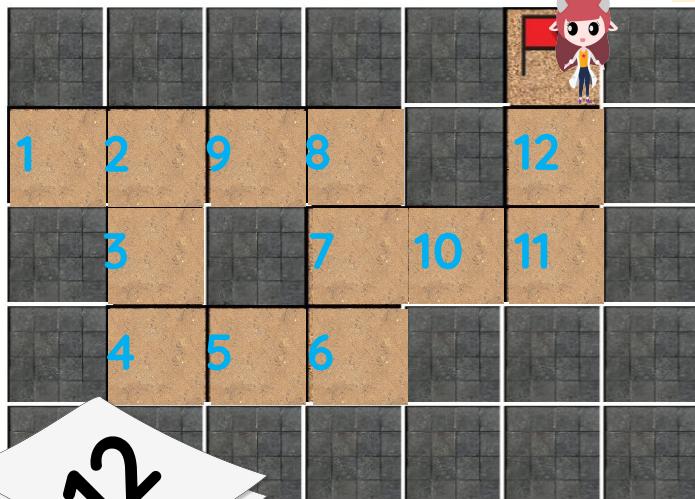


# Tìm theo chiều sâu (Depth-First Search)



Nhưng làm thế nào để  
đi về lại?

Hãy dùng chiếc hộp chứa các  
mẫu giấy ghi lại đường đi!

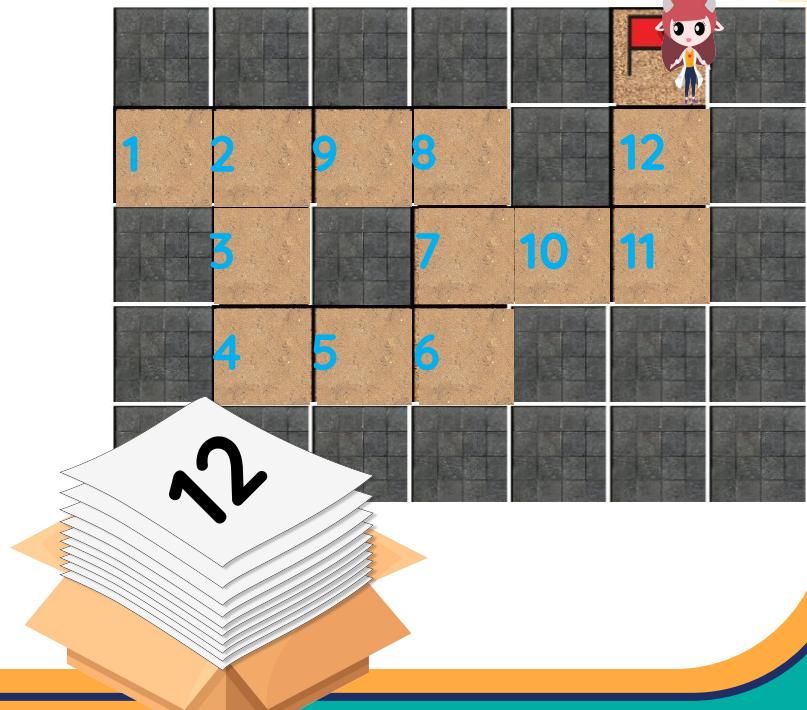




# Tìm theo chiều sâu (Depth-First Search)



Ahh! Chỉ cần lấy ra **lần lượt** các tờ giấy và đi theo thì ta có thể trở về rồi!

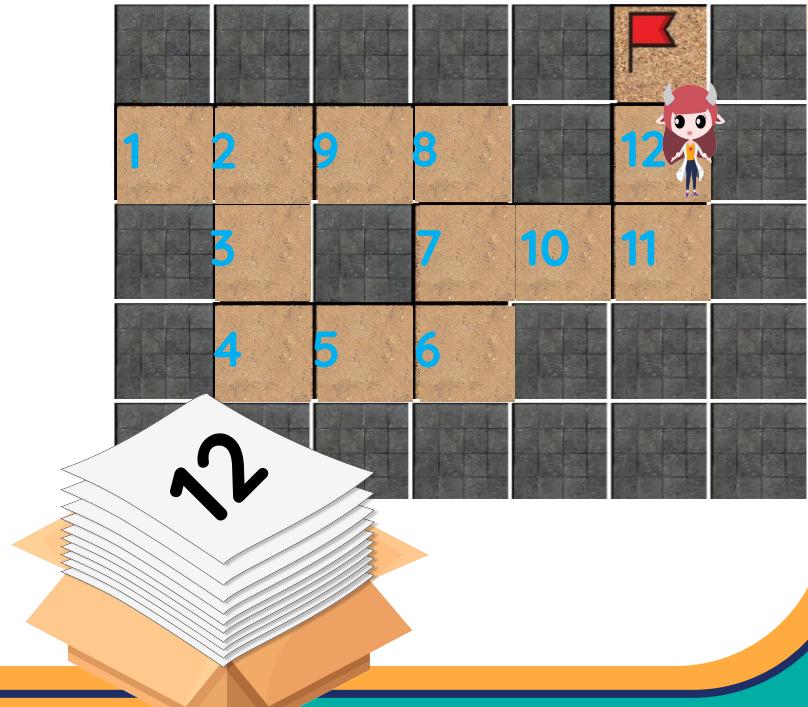




# Tìm theo chiều sâu (Depth-First Search)



Ahh! Chỉ cần lấy ra **lần lượt** các tờ giấy và đi theo thì ta có thể trở về rồi!

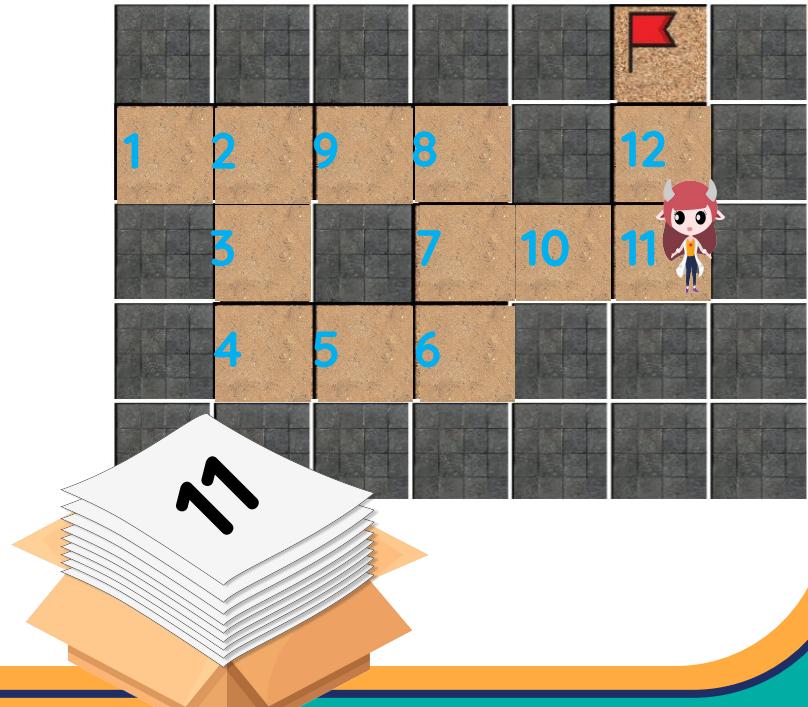




# Tìm theo chiều sâu (Depth-First Search)



Ahh! Chỉ cần lấy ra **lần lượt** các tờ giấy và đi theo thì ta có thể trở về rồi!

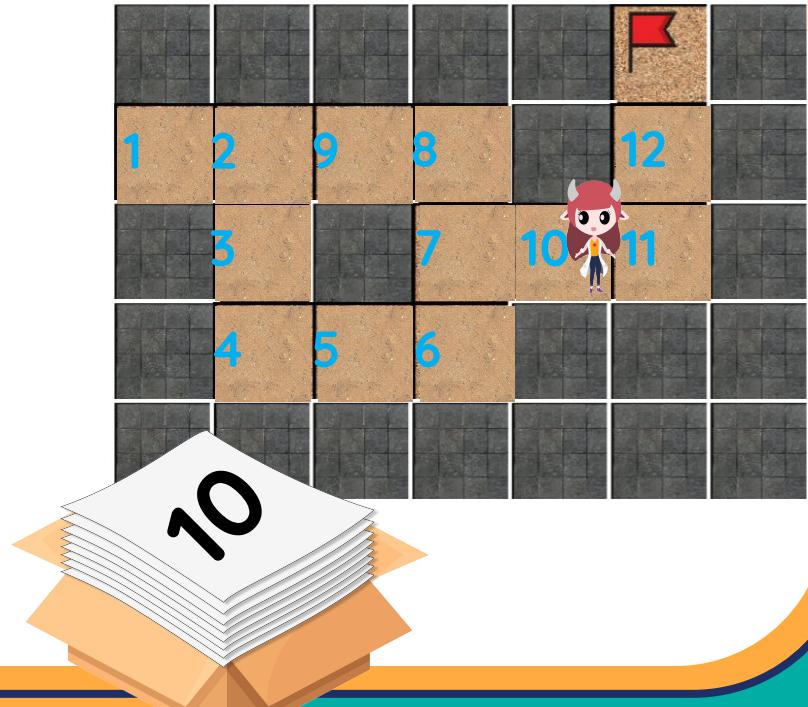




# Tìm theo chiều sâu (Depth-First Search)



Ahh! Chỉ cần lấy ra **lần lượt** các tờ giấy và đi theo thì ta có thể trở về rồi!

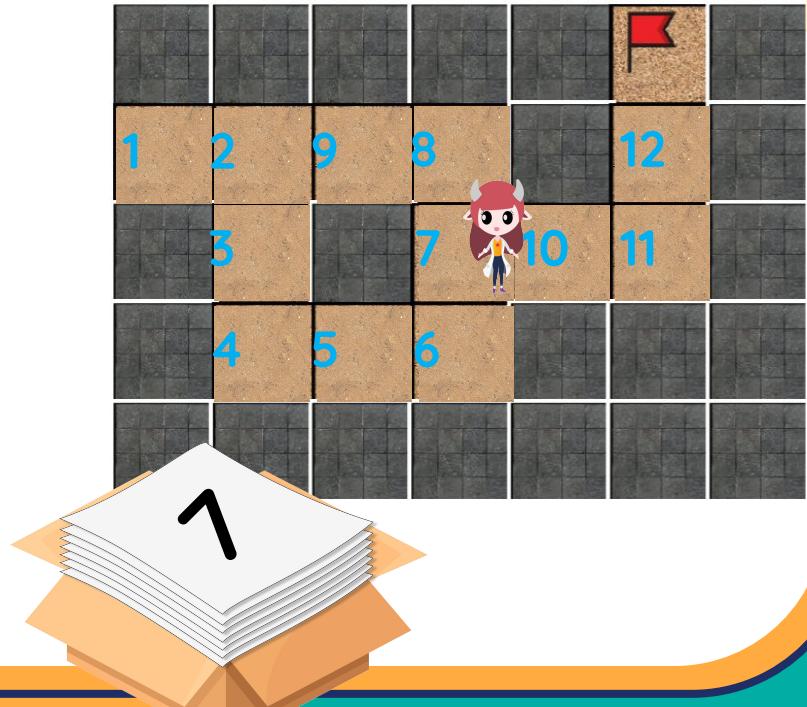




# Tìm theo chiều sâu (Depth-First Search)



Ahh! Chỉ cần lấy ra **lần lượt** các tờ giấy và đi theo thì ta có thể trở về rồi!

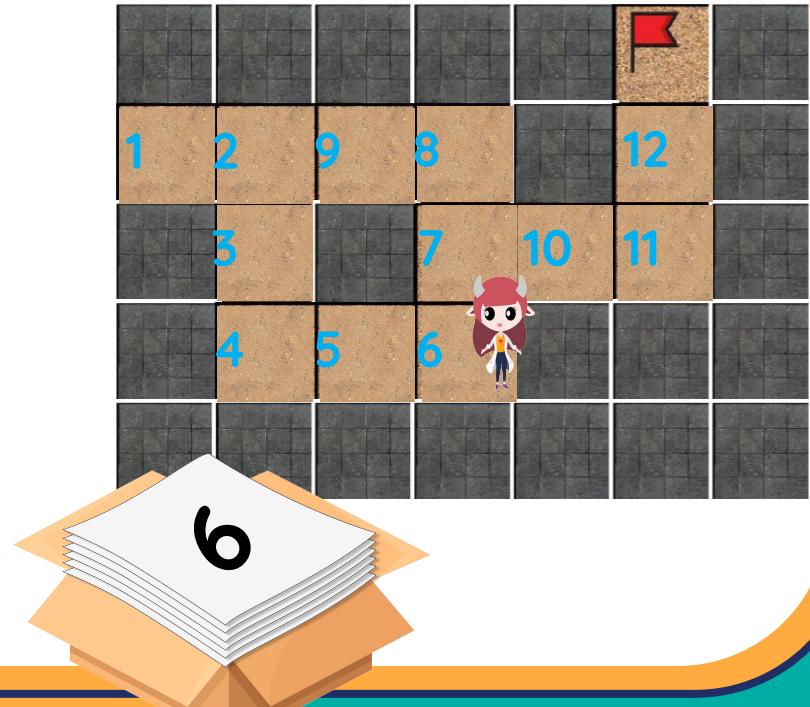




# Tìm theo chiều sâu (Depth-First Search)



Ahh! Chỉ cần lấy ra **lần lượt** các tờ giấy và đi theo thì ta có thể trở về rồi!

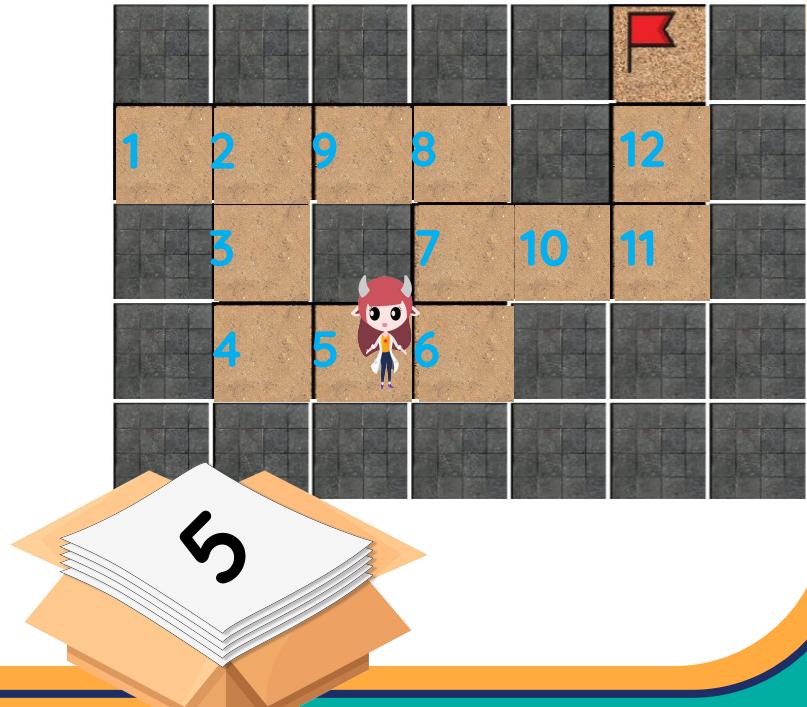




# Tìm theo chiều sâu (Depth-First Search)



Ahh! Chỉ cần lấy ra **lần lượt** các tờ giấy và đi theo thì ta có thể trở về rồi!

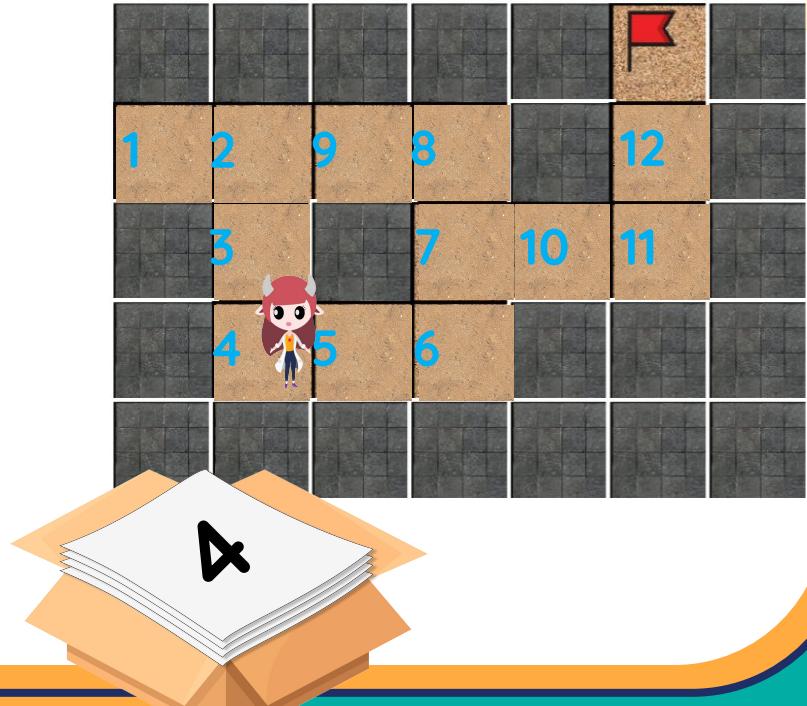




# Tìm theo chiều sâu (Depth-First Search)



Ahh! Chỉ cần lấy ra **lần lượt** các tờ giấy và đi theo thì ta có thể trở về rồi!

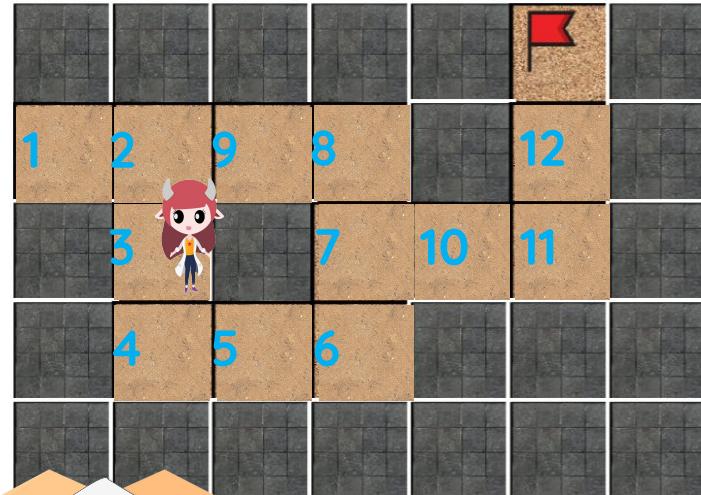




# Tìm theo chiều sâu (Depth-First Search)



Ahh! Chỉ cần lấy ra **lần lượt** các tờ giấy và đi theo thì ta có thể trở về rồi!

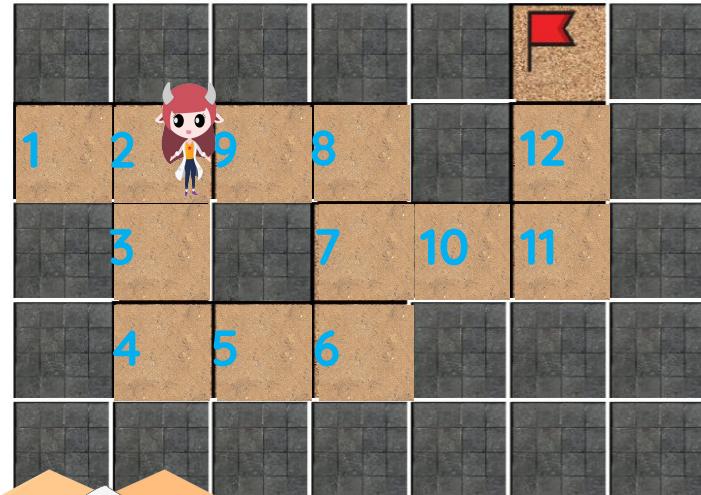




# Tìm theo chiều sâu (Depth-First Search)



Ahh! Chỉ cần lấy ra **lần lượt** các tờ giấy và đi theo thì ta có thể trở về rồi!

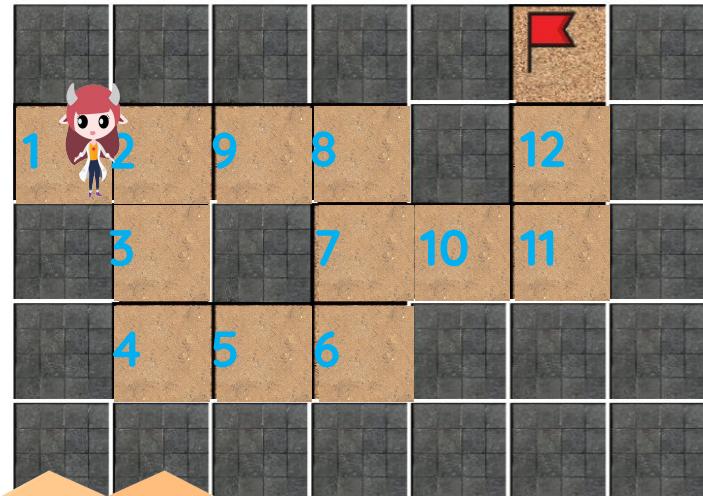




# Tìm theo chiều sâu (Depth-First Search)



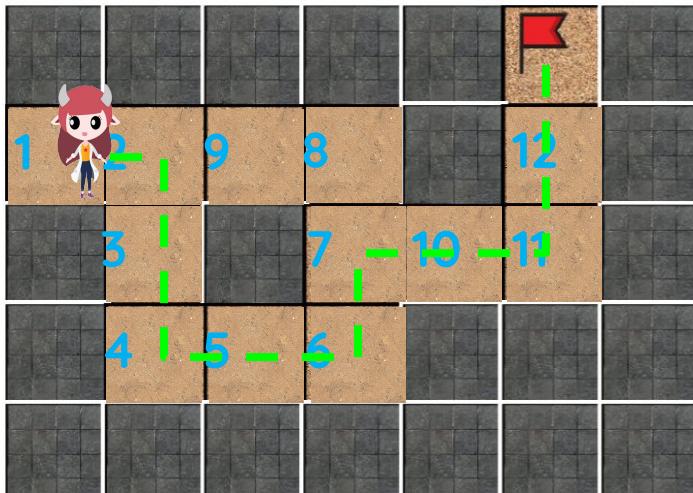
Vậy là tớ đã tìm được  
đường ra!





# Tìm theo chiều sâu (Depth-First Search)

Cách tìm lối đi này được gọi là  
**Tìm kiếm theo chiều sâu**  
(Depth-First Search)



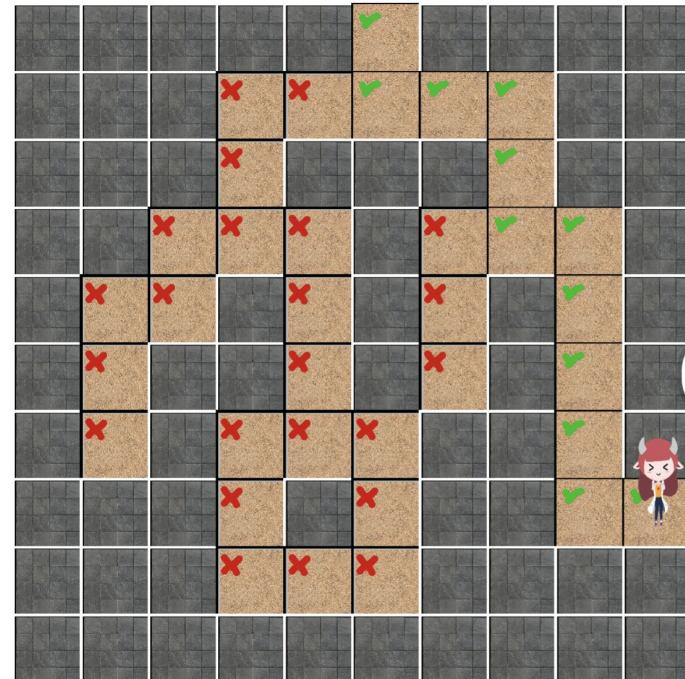


# Tìm theo chiều sâu (Depth-First Search)

## Hiệu quả?

Nếu thực hiện đúng thuật toán tìm đường theo chiều sâu (DFS), ta sẽ luôn tìm được lối ra trong mê cung (nếu mê cung đó có ít nhất một lối ra).

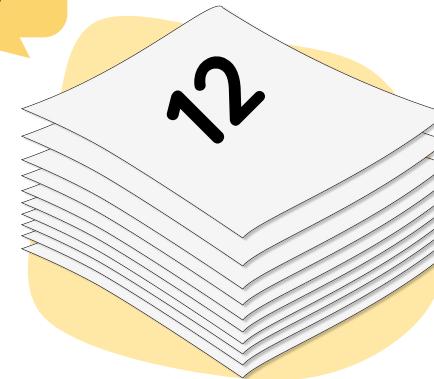
Đây là một trong những cách hiệu quả cho **con người** để tìm đường đi trong mê cung và đồng thời lưu lại lời giải.



# NGĂN XẾP (STACK)

Xếp giấy trong hộp mà ta dùng để lưu đường đi được gọi là **ngăn xếp (stack)**.  
**Ngăn xếp** là một cấu trúc dữ liệu.

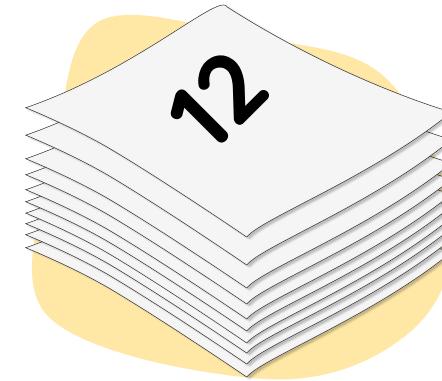
Hi! Tui là  
**stack** nè.



# NGĂN XẾP (STACK)

## Nguyên tắc

Tờ giấy nào đưa **vào sau** thì sẽ được lấy **ra trước**  
**(Last In First Out - LIFO)**



# NGĂN XẾP (STACK)

Thao tác **thêm** một tờ giấy vào **ngăn xếp (stack)** được gọi là **push**

Thao tác **lấy** một tờ giấy ra khỏi **ngăn xếp (stack)** được gọi là **pop**



# NGĂN XẾP (STACK)

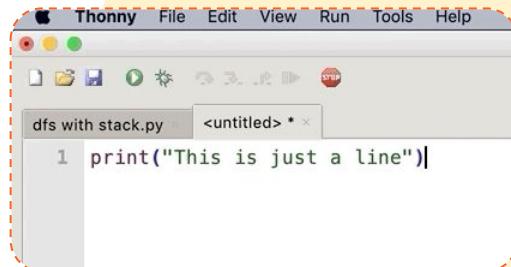
Trong thực tế, chúng ta có ví dụ gì về **ngăn xếp (stack)**



# NGĂN XẾP (STACK)

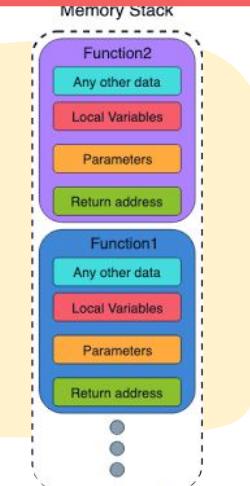
Trong máy tính, chúng ta **ngăn xếp (stack)** để làm gì?

Chức năng Undo



```
Thonny File Edit View Run Tools Help
dfs with stack.py <untitled> * x
1 print("This is just a line")|
```

Quản lý bộ nhớ





# NGĂN XẾP (STACK)

Chúng ta hãy dùng **stack** để **đảo ngược** một câu nhé!

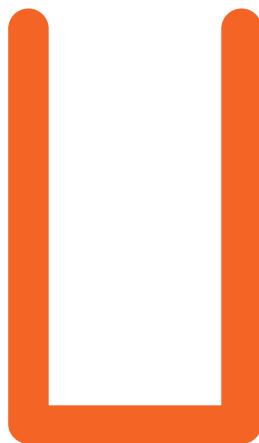
Đạo	Trọng	Sư	Tôn
-----	-------	----	-----



# NGĂN XẾP (STACK)

Đầu tiên, cho lần lượt đưa (**push**) các từ vào **stack**

Đạo	Trọng	Sư	Tôn
-----	-------	----	-----

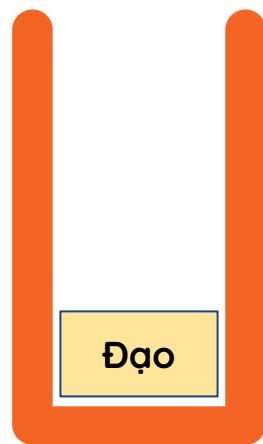




# NGĂN XẾP (STACK)

Đầu tiên, cho lần lượt đưa (**push**) các từ vào **stack**

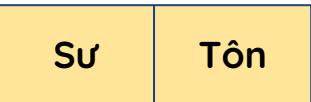
Trọng	Sư	Tôn
-------	----	-----





# NGĂN XẾP (STACK)

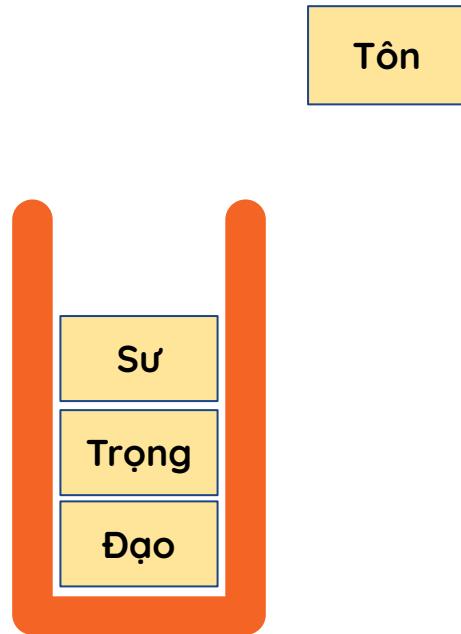
Đầu tiên, cho lần lượt đưa (**push**) các từ vào **stack**





# NGĂN XẾP (STACK)

Đầu tiên, cho lần lượt đưa (**push**) các từ vào **stack**





# NGĂN XẾP (STACK)

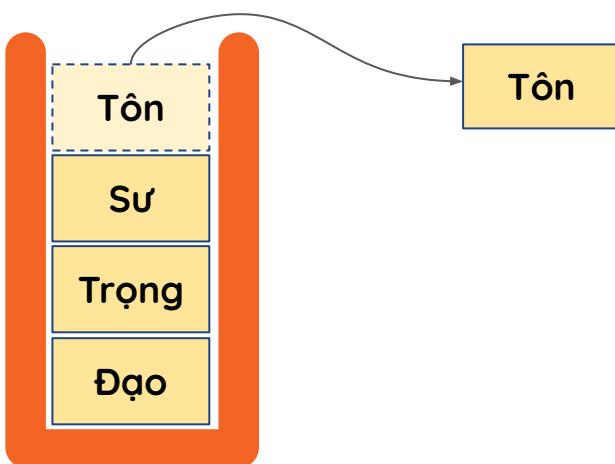
Đầu tiên, cho lần lượt đưa (**push**) các từ vào **stack**





# NGĂN XẾP (STACK)

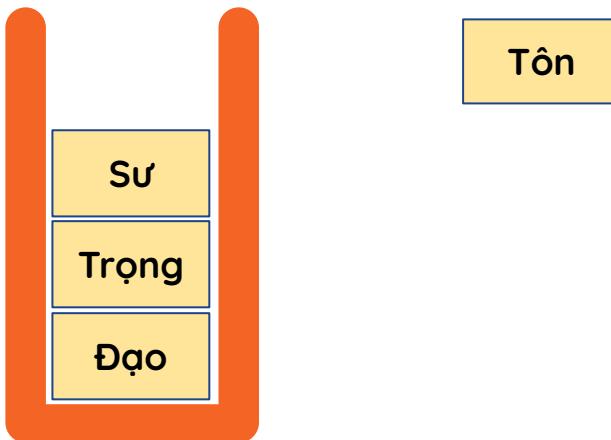
Sau đó, lần lượt lấy (**pop**) các từ ra khỏi **stack**





# NGĂN XẾP (STACK)

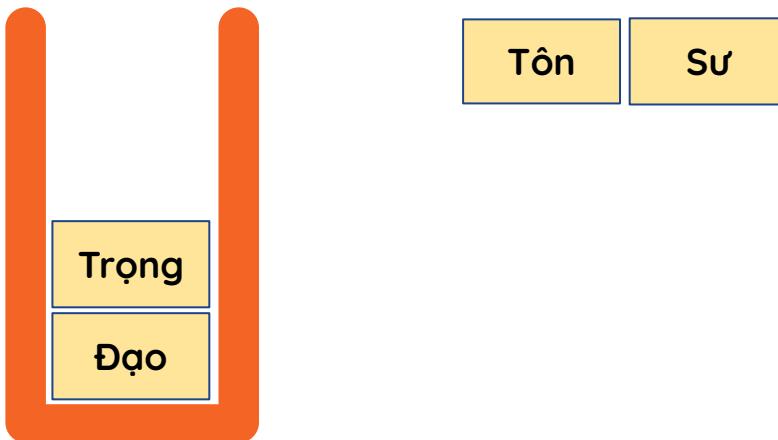
Sau đó, lần lượt lấy (**pop**) các từ ra khỏi **stack**





# NGĂN XẾP (STACK)

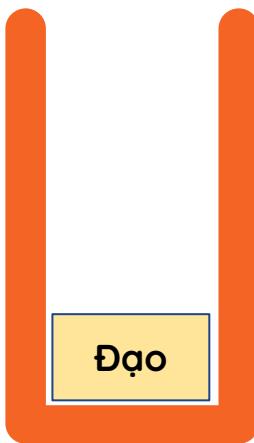
Sau đó, lần lượt lấy (**pop**) các từ ra khỏi **stack**





# NGĂN XẾP (STACK)

Sau đó, lần lượt lấy (**pop**) các từ ra khỏi **stack**

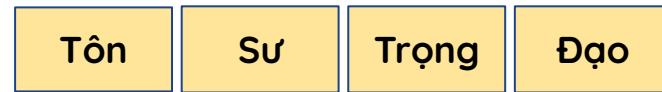
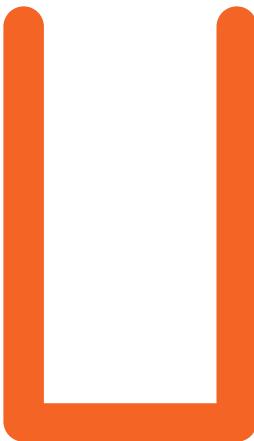


Tôn Sư Trọng



# NGĂN XẾP (STACK)

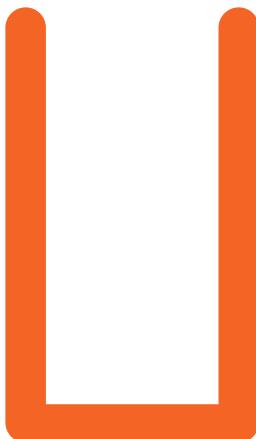
Sau đó, lần lượt lấy (**pop**) các từ ra khỏi **stack**





# NGĂN XẾP (STACK)

Sau đó, lần lượt lấy (**pop**) các từ ra khỏi **stack**



Tôn      Sư      Trọng      Đạo

[https://bit.ly/S4V\\_CS201\\_Stack](https://bit.ly/S4V_CS201_Stack)



## Quiz #2

**Đâu không phải là ứng dụng của stack**

- a. Tìm kiếm theo chiều sâu
- b. Chức năng Undo (quay lại) của các phần mềm
- c. Đảo ngược danh sách
- d. Xếp hàng mua vé xem phim

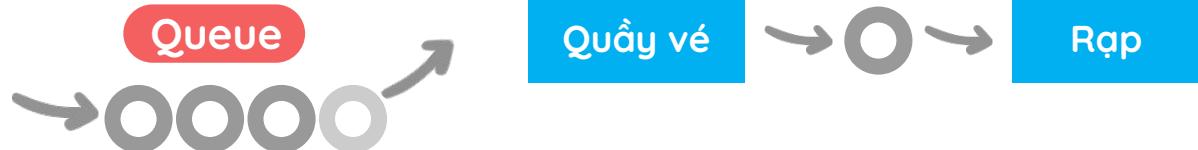




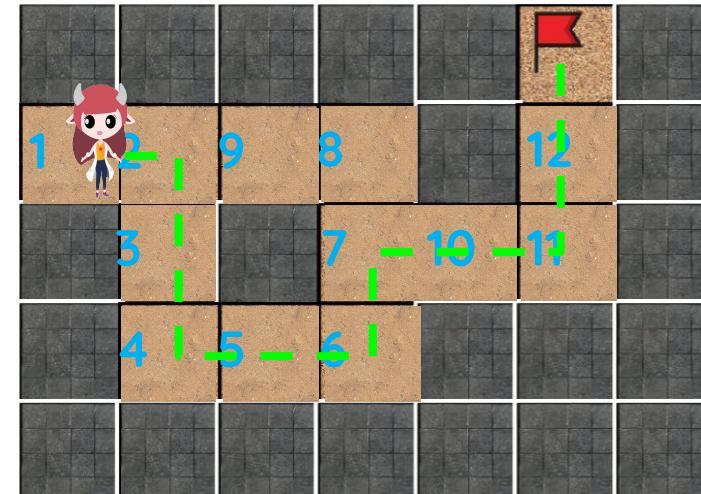
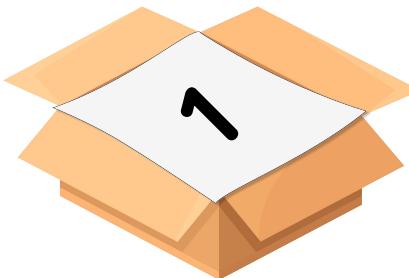
## Quiz #2

Đâu không phải là ứng dụng của stack

- a. Tìm kiếm theo chiều sâu
- b. Chức năng Undo (quay lại) của các phần mềm
- c. Đảo ngược danh sách
- d. Xếp hàng mua vé xem phim**



# Ngăn xếp - Tìm theo chiều sâu



# NGHỈ GIẢI LAO

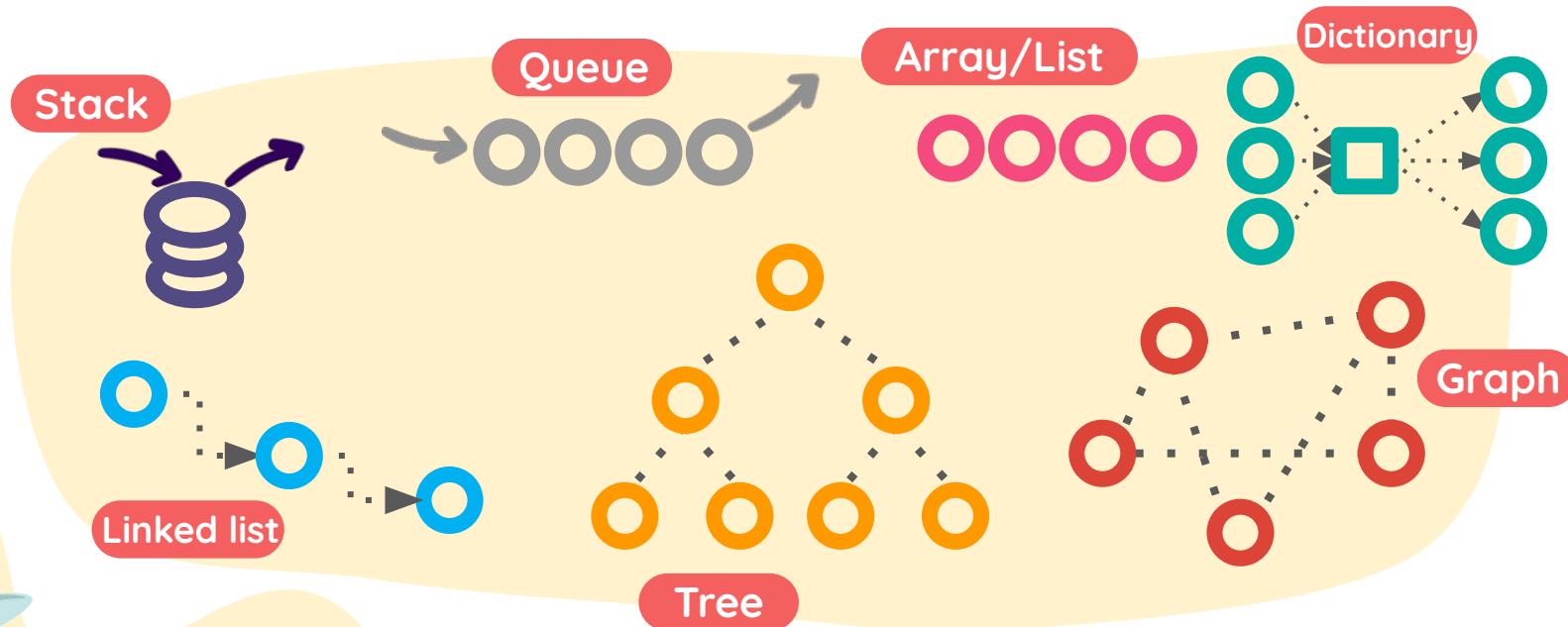


Z Z

Học sinh quay lại sau 10 phút nghỉ giải lao

# CẤU TRÚC DỮ LIỆU (DATA STRUCTURE)

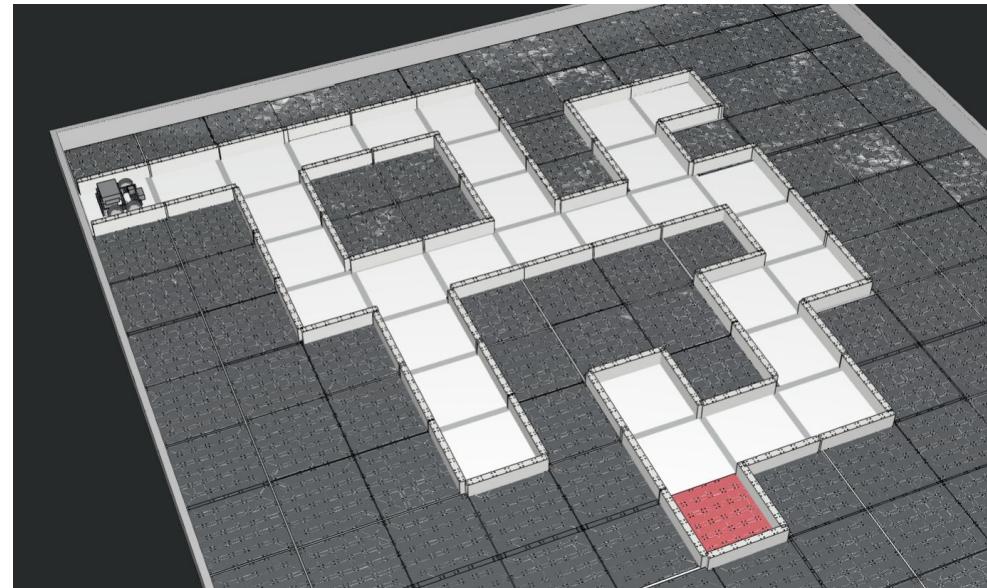
Chúng ta có rất **nhiều** loại **cấu trúc dữ liệu**



# Biểu diễn DFS bằng Python (Phần nâng cao)

# Biểu diễn DFS bằng Python

Làm sao để sử dụng DFS  
trong Python và giúp robot  
tìm đường?



# Biểu diễn DFS bằng Python

Cùng chia nhỏ vấn đề ra để giải quyết nhé!

- 0 Mã hóa bản đồ mê cung sang **số** 
- 1 Áp dụng thuật toán DFS để tìm đường
- 2 Ra lệnh cho robot di chuyển theo đường đã tìm được



# Biểu diễn DFS bằng Python

- 0 Mã hóa bản đồ mê cung sang **số**

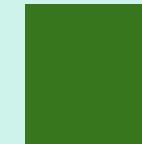
Một mê cung gồm



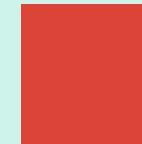
Tường



Lối đi



Điểm xuất phát



Điểm đích

0

1

2

3

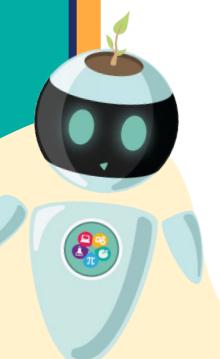
# Biểu diễn DFS bằng Python

0 Mã hóa bản đồ mê cung sang **số**

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[2, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0]
[0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0]
[0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0]
[0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0]
[0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0]
[0, 0, 0, 1, 0, 0, 0, 1, 1, 0]
[0, 0, 0, 1, 0, 1, 0, 1, 0, 0]
[0, 0, 0, 0, 0, 1, 1, 1, 0, 0]
[0, 0, 0, 0, 0, 3, 0, 0, 0, 0]



1 hàng = 1 danh sách



# Biểu diễn DFS bằng Python

0 Mã hóa bản đồ mê cung sang **số**

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[2, 1, 1, 1, 1, 1, 0, 0, 0, 0]
[0, 0, 1, 0, 0, 1, 0, 1, 1, 0]
[0, 0, 1, 0, 0, 1, 0, 1, 0, 0]
[0, 0, 1, 1, 1, 1, 1, 1, 1, 0]
[0, 0, 0, 1, 0, 0, 0, 0, 1, 0]
[0, 0, 0, 1, 0, 0, 0, 1, 1, 0]
[0, 0, 0, 1, 0, 1, 0, 1, 0, 0]
[0, 0, 0, 0, 0, 1, 1, 1, 0, 0]
[0, 0, 0, 0, 0, 3, 0, 0, 0, 0]



```
maze_array = [
    [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
    [ 2, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0 ],
    [ 0, 0, 1, 0, 0, 1, 0, 1, 1, 0 ],
    [ 0, 0, 1, 0, 0, 1, 0, 1, 0, 0 ],
    [ 0, 0, 1, 1, 1, 1, 1, 1, 1, 0 ],
    [ 0, 0, 0, 1, 0, 0, 0, 0, 1, 0 ],
    [ 0, 0, 0, 1, 0, 0, 0, 1, 1, 0 ],
    [ 0, 0, 0, 1, 0, 0, 0, 0, 1, 0 ],
    [ 0, 0, 0, 1, 0, 1, 0, 1, 0, 0 ],
    [ 0, 0, 0, 0, 0, 1, 1, 1, 0, 0 ],
    [ 0, 0, 0, 0, 0, 3, 0, 0, 0, 0 ]
]
```

Danh sách hai chiều hay còn gọi là ma trận

# Biểu diễn DFS bằng Python

Cùng chia nhỏ vấn đề ra để giải quyết nhé!

- 0 Mã hóa bản đồ mê cung sang **số**
- 1 Áp dụng thuật toán DFS để tìm đường ←
- 2 Ra lệnh cho robot di chuyển theo đường đã tìm được



# Biểu diễn DFS bằng Python

## 1 Áp dụng thuật toán DFS

**Bước 1** Đến điểm xuất phát

**Bước 2** Dùng phấn đánh số vị trí hiện tại  
*(Không dùng các số trùng nhau)*

Nhìn xung quanh theo thứ tự: **Trên - dưới - trái - phải** xem có đường nào chưa đánh dấu không. **Nếu có**, sang **bước 4**. **Nếu không**, sang **bước 5**.

**Bước 4** Đi sang ô tiếp theo, ghi lại số của ô cũ rồi bỏ vào hộp. Quay lại **bước 2**

**Bước 5** Lấy 1 tờ giấy trong hộp, quay lại vị trí trong tờ giấy. Quay lại **bước 3**.

# Biểu diễn DFS bằng Python

Cùng chia nhỏ vấn đề ra để giải quyết nhé!

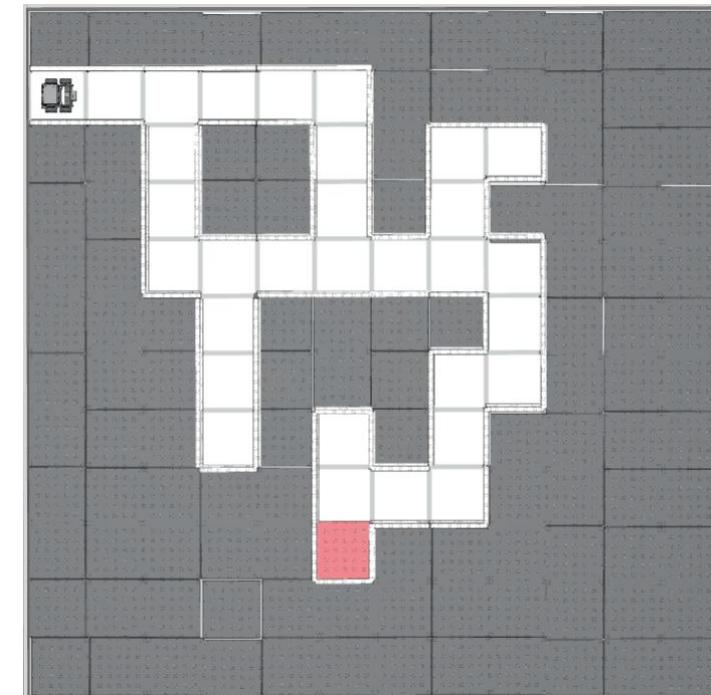
- 0 Mã hóa bản đồ mê cung sang **số**
  - 1 Áp dụng thuật toán DFS để tìm đường
  - 2 Ra lệnh cho robot di chuyển theo đường đã tìm được



# Biểu diễn DFS bằng Python

Đây là chương trình biểu diễn thuật toán  
**tìm kiếm theo chiều sâu (DFS)**

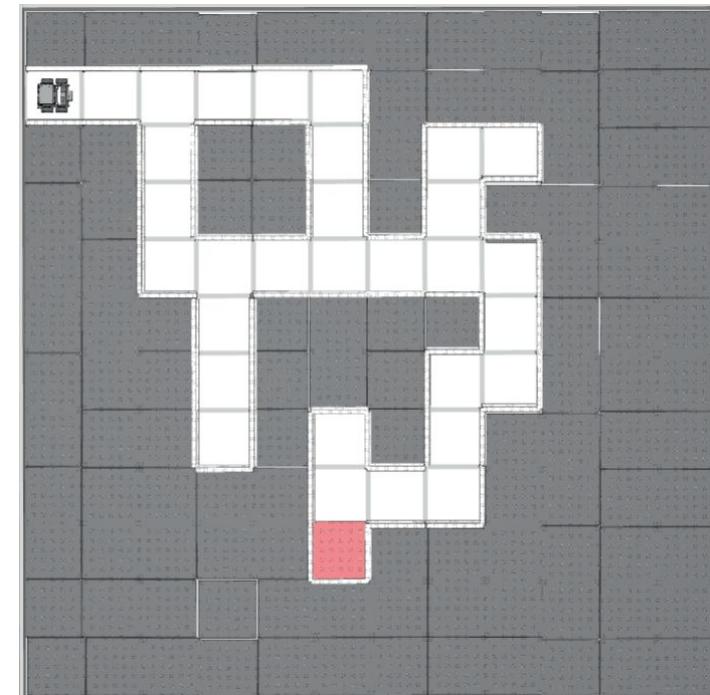
[https://bit.ly/S4V\\_CS201\\_Maze\\_DFS](https://bit.ly/S4V_CS201_Maze_DFS)



# Biểu diễn DFS bằng Python

Đây chỉ là chương trình đơn giản, chúng ta **phải biết trước được bản đồ** của mê cung.

Phiên bản nâng cao hơn còn có thể giúp robot tự tìm đường trong mê cung mà *không cần bản đồ!* Hẹn ở lớp chính thức!



# TỔNG KẾT

# KHÓA HỌC PYTHON CHO ROBOTICS

- Các kiểu dữ liệu cơ bản: chuỗi, số (integer, float), boolean
- Các kiểu dữ liệu nâng cao: danh sách, ngăn xếp, v.v...
- Các toán tử toán học, các toán tử boolean/logic
- Điều kiện if/elif/else
- Vòng lặp while, for
- Lập trình hướng đối tượng (Object-oriented Programming)
- Đọc hiểu/sử dụng VEX API trên virtual platform Robot Mesh
- Phạm vi (local/global - cục bộ/toàn cục)
- Máy trạng thái hữu hạn - Finite State Machine
- Tìm kiếm theo chiều sâu (Depth-first Search)



# LỊCH BUỔI GIAO LƯU

Chủ Nhật tuần sau (25/07/2021)

Giờ học: 7:30 sáng giờ VN





Hẹn gặp lại!