Lab01: Sử dụng Python để giải bài toán đong nước (2-jar / water jug problem) sử dụng phương pháp tìm kiếm theo BFS (Breadth-First Search).

Yêu cầu: SV tự viết code, không nên tham khảo ChatGPT hay DeepSeek

Bài toán đong nước, còn được gọi là bài toán bình nước hay Water Jug Problem, là một bài toán đố logic trong đó người giải phải tìm cách đong được một lượng nước chính xác nào đó bằng cách sử dụng các bình (hoặc ca, jar) có dung tích khác nhau mà không có bất kỳ dụng cụ đo lường nào khác. Cụ thể trong bài toán này có 2 bình nước dung tích

1. Phát biểu bài toán 2 bình nước:

Cho hai bình (bình A và bình B) với dung tích lần lượt là 7 lít và 5 lít. Một vòi nước để lấy nước (có thể đổ đầy bình bất kỳ lúc nào). Một bồn chứa để đổ nước đi (có thể làm trống bình bất kỳ lúc nào). Một lượng nước mục tiêu 6 lít cần đong được tại bình A. Hỏi: Làm thế nào để đong được chính xác 6 lít nước vào bình A thông qua một chuỗi các thao tác với hai bình này?

Các thao tác (chuyển trạng thái):

- Đổ đầy một bình từ vòi nước.
- Làm trống một bình ra bồn chứa.
- Đổ nước từ bình này sang bình kia. Thao tác này dừng lại khi bình thứ nhất hết nước hoặc bình thứ hai đầy.

2. Phân tích:

- Sử dụng 2 biến x và y để biểu diễn mức nước của 2 bình A và B. Một trạng thái bất kỳ của hai bình nước chính là 1 tuple, *state = (x, y)*. Trạng thái ban đầu *state = (0, 0)*

- Gọi openStates là biến chứa các nút chưa xét trong không gian trạng thái của bài toán. Mỗi lần duyệt các nút mới, các nút này lần lượt chèn vào trong openStates, sau đó lấy tưng nút ra để xét. Vậy, openStates là một hàng đợi. Để có biến hàng đợi, sử dụng lệnh: *from collections import deque*

- Với mỗi trạng thái state =(x,y), ta có thể phát sinh ra 4 trường hợp (tương ứng với các thao tác chuyển trạng thái): (7,y), (x,5), (0,y), (x,0). Để chèn 1 nút trong không gian trạng thái vào hàng đợi openStates, dùng hàm append. Ví dụ: chèn nút (7, y) vào openStates ta dùng lệnh *openStates.append((7,y))*. Nếu muốn chuyển nước từ bình A sang bình B, ta đo lượng (pourAmount) còn trống bên bình B sau đó lấy lượng nước trong bình A trừ pourAmount (không được âm).

- Hàm get_next_states(state) sẽ trả về danh sách các con của nút hiện hành (nút đầu vào của hàm)

```
def get_next_states(state):
    x, y = state
    states = [] # các trạng thái con của trạng thái state

    # 1. Đổ đầy bình 1 hoặc bình 2
```

```
    ....
    # 2. Làm rỗng bình 1 hoặc bình 2
      ....
    # 3. Chuyển nước từ bình 1 sang bình 2
      ....
    # 4. Chuyển nước từ bình 2 sang bình 1
      ....
    return states
```

- Xây dựng hàm bfsSearch() thực hiện việc tìm kiếm trong không gian trạng thái.

```
def bfsSearch(): # Hàm tìm kiếm
    start = (0, 0)  #ban đầu cả hai bình đều trống
    openStates = deque([(start, [])])  # khởi động 1 hàng đợi có 1 nút đầu tiên
    closedStates = set([start]) # các nút đã xét trong không gian trạng thái
    TARGET = 6 # bình A đạt 6 lit

    while openStates: # trong khi openStates khác rỗng
        (x, y), path = openStates.popleft() # lấy 1 phần tử khỏi hàng đợi (bao gồm cả đường dẫn)

        # Kiểm tra mục tiêu
        if x == TARGET:
            return path + [(x, y)]

        for next_state in get_next_states((x, y)):
            if next_state not in closedStates:
                closedStates.add(next_state)
                openStates.append((next_state, path + [(x, y)]))

    return None
```

Gọi hàm bfsSearch() & in kết quả

```
solution = bfsSearch()

# In kết quả
if solution:
    print("Các bước chuyển trạng thái để đạt được 6 lít trong bình 7 lít:")
    for step in solution:
        print(step)
else:
    print("Không tìm thấy lời giải")
```

3. Hãy cập nhật lại code trên để in ra số lượng nút trạng thái đã từng đi qua.

4. Hãy thực hiện lại code trên sử dụng depth first search (DFS). Tính số lượng nút đã từng “ghé thăm”