BÁO CÁO THỰC HÀNH AI LAB2

Họ và Tên: Lư Xuân Dương
MSSV: 22280015

GENERATE_FULL_SPACE_TREE.PY

```
options = [(1, 0), (0, 1), (1, 1), (0, 2), (2,0)]
Parent = dict()
graph= pydot.Dot(graph_type='graph', strict=False, bgcolor="#fff3af",
                 label= "fig:Missionaries and Cannibal State Space Tree",
                 fontcolor="red", fontsize= "24", overlap= "true")
```

 # Mục đích để khởi tạo đồ thị với kích thước 24, màu chữ đỏ ....

```
i=0
arg= argparse.ArgumentParser()
arg.add_argument("-d", "--depth", required=False,
                 help="Maximun depth upto which you want to generate Space State Tree")
```

 # Thêm tham số -d hoặc –depth (nhưng có không thêm do không bắt buộc)

```
args = vars(arg.parse_args())
max_depth= int(args.get("depth", 20))
```

 # Nếu không thêm depth thì mặc định depth =20

```
def draw_edge(number_missionaries, number_cannibals, side, depth_level, node_num):
    u, v = None, None
    if Parent[(number_missionaries, number_cannibals, side, depth_level, node_num)] is not None:
        u = pydot.Node(str(Parent[(number_missionaries, number_cannibals, side, depth_level, node_num)]),
                       label= str(Parent[(number_missionaries, number_cannibals, side, depth_level, node_num)][:3]))
        graph.add_node(u)
        v = pydot.Node(str((number_missionaries, number_cannibals, side, depth_level, node_num)),
                       label= str((number_missionaries, number_cannibals, side)))
        graph.add_node(v)

        edge = pydot.Edge(str(Parent[(number_missionaries, number_cannibals, side, depth_level, node_num)]),
                          str((number_missionaries, number_cannibals, side, depth_level, node_num) ), dir= 'forward')
        graph.add_edge(edge)
    else:
        #For start node
        v = pydot.Node(str((number_missionaries, number_cannibals, side, depth_level, node_num)),
                       label= str((number_missionaries, number_cannibals, side)))
        graph.add_node(v)
    return u,v
```

Mục đích:

- Nếu tồn tại nút cha của nút hiện tại (tức là nút này không phải nút gốc), hàm sẽ tạo cạnh (edge) kết nối từ nút cha (u) tới nút con (v).
- Mỗi trạng thái của bài toán, bao gồm số lượng người truyền giáo, kẻ ăn thịt và vị trí của họ (bờ sông bên trái hoặc bên phải), sẽ được biểu diễn dưới dạng một **nút (node)** trong đồ thị.

```
def number_of_cannibals_exceeds(number_missionries, number_cannibals):
    number_missionries_right = 3 - number_missionries
    number_cannibals_right = 3- number_cannibals
    return (number_missionries > 0 and number_cannibals > number_missionries)\
or (number_missionries_right > 0 and number_cannibals_right> number_missionries_right)
```

Mục đích: Hàm number_of_cannibals_exceeds kiểm tra xem tại một trạng thái bất kỳ của bài toán "Người truyền giáo và Kẻ ăn thịt", số lượng kẻ ăn thịt có nhiều hơn số lượng người truyền giáo hay không. Đây là một điều kiện để xác định trạng thái không hợp lệ so với yêu cầu bài toán. Nhưng trường hợp dẫn đến False(Hay người truyền giáo bị ăn thịt) thì thuật toán sẽ loại bỏ nó , ngăn chặn việc người truyền giáo vị ăn thịt, từ đó chỉ tập trung vào các trường hợp có khả năng dẫn đến lời giải.

```
def generate():
    global i
    q= deque()
    node_num = 0
    q.append((3, 3, 1, 0, node_num))

    Parent [(3, 3, 1, 0, node_num)] = None

    while q:

        number_missionaries, number_cannibals, side, depth_level, node_num = q.popleft()
        #print ( number_missionaries, number_cannibals  )
        #Draw Edge from u -> v
        #Where u = Parent[v]
        # and v= (number_missionaries, number_cannibals, side, depth_level)
        u, v = draw_edge(number_missionaries, number_cannibals, side, depth_level, node_num)

        if is_start_state(number_missionaries, number_cannibals, side):
            v.set_style("filled")
            v.set_fillcolor("blue")
            v.set_fontcolor("white")
        elif is_goal_state(number_missionaries, number_cannibals, side):
            v.set_style("filled")
            v.set_fillcolor("green")
            continue
            # return True

        elif number_of_cannibals_exceeds(number_missionaries, number_cannibals):
            v.set_style("filled")
            v.set_fillcolor("red")
            continue
        else:
            v.set_style("filled")
            v.set_fillcolor("orange")

        if depth_level == max_depth:
            return True
```

```
    op = -1 if side ==1 else 1
    can_be_expanded = False
    # i = node_num
    for x, y in options:
        next_m, next_c, next_s = number_missionaries + op * x, number_cannibals + op*y, int(not side)

        if Parent[(number_missionaries, number_cannibals, side, depth_level, node_num)] is None or(next_m, next_c, next_s)\
        != Parent[(number_missionaries, number_cannibals, side, depth_level, node_num)][:3]:
            if is_valid_move(next_m, next_c):
                can_be_expanded = True
                i += 1
                q.append((next_m, next_c, next_s, depth_level+1, i))
                #Keep track of parent
                Parent[(next_m, next_c, next_s, depth_level+1, i)] =\
                (number_missionaries, number_cannibals, side, depth_level, node_num)
    if not can_be_expanded:
        v.set_style("filled")
        v.set_fillcolor("gray")
return False
```

Mục đích:

- Hàm này thực hiện việc tìm kiếm và xây dựng cây trạng thái của bài toán, đồng thời kiểm tra xem có thể đạt được trạng thái đích (là trạng thái hợp lệ, không có người truyền giáo nào bị ăn thịt) hay không
- **Khởi tạo hàng đợi (queue)**:

  + q = deque(): Sử dụng một hàng đợi để triển khai BFS.

  + q.append((3, 3, 1, 0, node_num)): Thêm trạng thái ban đầu (3 người truyền giáo và 3 kẻ ăn thịt ở bờ bên trái, thuyền ở bờ bên trái, cấp độ sâu 0) vào hàng đợi.

    + Parent[(3, 3, 1, 0, node_num)] = None: Gán trạng thái ban đầu không có nút cha (Parent của nó là None).

- u, v = draw_edge(...): Gọi hàm draw_edge để vẽ cạnh từ nút cha (u) tới nút hiện tại (v). Nút v sau đó sẽ được tạo ra với các màu khác nhau dựa trên trạng thái của nó:

  + **Xanh lam (blue)** cho trạng thái bắt đầu.
  + **Xanh lá (green)** cho trạng thái mục tiêu (thành công).
  + **Đỏ (red)** cho trạng thái không hợp lệ.
  + **Cam (orange)** cho trạng thái hợp lệ khác.
  +**Xám (gray)** cho nút không thể mở rộng.

⇨ Hàm trên khá giống với thuật toán BFS

SOLVE.PY

```
def solve(self, solve_method="dfs"):
    self.visited = dict()
    Parent[self.start_state] = None
    Move[self.start_state] = None
    node_list[self.start_state] = None

    return self.dfs(*self.start_state, 0) if solve_method == "dfs" else self.bfs()
```

Dùng để lựa chọn thuật toán DFS hay BFS tùy vào input của người dùng. Và khởi tạo các biến liên quan để phục vụ cho 2 thuật toán BFS và DFS. Khi khởi tạo thì các biến ở **start_node** nên các biến đó sẽ là None.

- **node5** (Nút đích - "Goal Node"): Màu xanh lá.

```
def draw_legend(self):
    """
    Utility method to draw legend on graph if legend flag is ON
    """
    graphlegend = pydot.Cluster(graph_name="Legend", label="Legend", fontsize="20", color="gold",
                                fontcolor="blue", style="filled", fillcolor="#f4f4f4")

    node1 = pydot.Node("1", style="filled", fillcolor="blue", label="Start Node", fontcolor="white", width="2", fixedsize="true")
    graphlegend.add_node(node1)

    node2 = pydot.Node("2", style="filled", fillcolor="red", label="Killed Node", fontcolor="black", width="2", fixedsize="true")
    graphlegend.add_node(node2)

    node3 = pydot.Node("3", style="filled", fillcolor="yellow", label="Solution Node", width="2", fixedsize="true")
    graphlegend.add_node(node3)

    node4 = pydot.Node("4", style="filled", fillcolor="gray", label="Can't be expanded", width="2", fixedsize="true")
    graphlegend.add_node(node4)

    node5 = pydot.Node("5", style="filled", fillcolor="green", label="Goal Node", width="2", fixedsize="true")
    graphlegend.add_node(node5)

    node7 = pydot.Node("7", style="filled", fillcolor="gold", label="Node with child", width="2", fixedsize="true")
    graphlegend.add_node(node7)


    description = "Each node (m, c, s) represents a \nstate where 'm' is the number of\n missionaries,\ 'c' is the cannibals \
    and \n's' is the side of the boat\n"
    " Where '1' represents the left \nside and '0' the right side \n\nOur objective is to reach goal state (0, 0, 0)\
    \nfrom start state (3, 3, 1) by some \noperations = [(1, 0), (0, 1), (2, 0), (0, 2), (1, 1)]\n"\
        "Each tuple (x, y) inside operators \nrepresents the number of missionaries and\
    \ncannibals to be moved from left to right \nif s == 1 and vice versa."


    node6 = pydot.Node("6", style="filled", fillcolor="gold", label=description, shape="plaintext", fontsize=20, fontcolor="red")
    graphlegend.add_node(node6)

    self.graph.add_subgraph(graphlegend)

    self.graph.add_edge(pydot.Edge(node1, node2, style="invis"))
    self.graph.add_edge(pydot.Edge(node2, node3, style="invis"))
    self.graph.add_edge(pydot.Edge(node3, node4, style="invis"))
    self.graph.add_edge(pydot.Edge(node4, node5, style="invis"))
    self.graph.add_edge(pydot.Edge(node5, node7, style="invis"))
    self.graph.add_edge(pydot.Edge(node7, node6, style="invis"))
```

Dùng để mô tả các trạng thái của 2 bên bờ sông ứng với màu sắc của mỗi node

node1 (Nút bắt đầu - "Start Node"): Màu xanh , chữ màu trắng.
node2 (Nút bị loại - "Killed Node"): Màu đỏ, chữ màu đen.
node3 (Nút giải pháp - "Solution Node"): Màu vàng.
node4 (Không thể mở rộng - "Can't be expanded"): Màu xám.

node5 (Nút đích - "Goal Node"): Màu xanh lá.
node7 (Nút có con - "Node with child"): Màu vàng.

node6 là một nút mô tả các ký hiệu trên đồ thị.

self.graph.add_subgraph(graphlegend): Thêm cụm chú giải vào đồ thị .

Các cạnh vô hình (style="invis") được thêm giữa các nút để sắp xếp các nút trong chú giải theo thứ tự mong muốn.

```
def draw(self, *, number_missionaries_left, number_cannibals_left, number_missionaries_right, number_cannibals_right):
    """
        raw state on console using emojis
    """
    left_m = emoji.emojize(f"old_man: " * number_missionaries_left)
    left_c = emoji.emojize(f"ogre: " * number_cannibals_left)
    right_m = emoji.emojize(f":old_man: " * number_missionaries_right)
    right_c = emoji.emojize(f":ogre: " * number_cannibals_right)

    print('{}{}{}{}{}'.format(left_m, left_c + " " * (14 - len(left_m) - len(left_c)),\
        "_" * 40, " " * (12 - len(right_m) - len(right_c)) + right_m, right_c))
    print("")
```

left_m: Tạo chuỗi emoji "old man" (biểu thị người truyền giáo) ở bờ trái, dựa trên số lượng number_missionaries_left.

left_c: Tạo chuỗi emoji "ogre" (biểu thị kẻ ăn thịt) ở bờ trái, dựa trên số lượng number_cannibals_left.

right_m: Tạo chuỗi emoji "old man" ở bờ phải, dựa trên number_missionaries_right.

right_c: Tạo chuỗi emoji "ogre" ở bờ phải, dựa trên number_cannibals_rig

Hàm draw giúp hiển thị trực quan trạng thái của bài toán trên bảng điều khiển, làm cho trò chơi dễ theo dõi hơn bằng các emoji tượng trưng.

```
def show_solution(self):
    # Recursively start from Goal State
    # And find parent until start state is reached

    state = self.goal_state
    path, steps, nodes = [], [], []

    while state is not None:
        path.append(state)
        steps.append(Move[state])
        nodes.append(node_list[state])

        state = Parent[state]

    steps, nodes = steps[::-1], nodes[::-1]
    number_missionaries_left, number_cannibals_left = 3, 3
    number_missionaries_right, number_cannibals_right = 0, 0

    print("*" * 60)
    self.draw(number_missionaries_left=number_missionaries_left, number_cannibals_left=number_cannibals_left,
              number_missionaries_right=number_missionaries_right, number_cannibals_right=number_cannibals_right)
```

- Duyệt ngược từ trạng thái đích (goal_state) đến trạng thái bắt đầu (start_state) bằng cách truy ngược Parent[state].
- Với mỗi state, thêm vào path, steps, và nodes.
- Duyệt tiếp tục cho đến khi state là None (tức là đã đến trạng thái bắt đầu).

Hàm show_solution có nhiệm vụ hiển thị toàn bộ lộ trình giải quyết bài toán "Người truyền giáo và Kẻ ăn thịt" từ trạng thái đích (goal state) đến trạng thái bắt đầu (start state). Hàm này truy ngược từ trạng thái đích, tìm ra cha của mỗi trạng thái cho đến khi đạt đến trạng thái bắt đầu và hiển thị các bước chuyển đổi ở từng giai đoạn.

```
def draw_edge(self, number_missionaries, number_cannibals, side, depth_level):
    u, v = None, None
    if Parent[(number_missionaries, number_cannibals, side)] is not None:
        u = pydot.Node(str(Parent[(number_missionaries, number_cannibals, side)] + (depth_level - 1,)),
                       label=str(Parent[((number_missionaries, number_cannibals, side))]))
        self.graph.add_node(u)


        v = pydot.Node(str((number_missionaries, number_cannibals, side, depth_level)),
                       label=str((number_missionaries, number_cannibals, side)))
        self.graph.add_node(v)

        edge = pydot.Edge(str(Parent[(number_missionaries, number_cannibals, side)] + (depth_level - 1,)),
                          str((number_missionaries, number_cannibals, side, depth_level)), dir="forward")
        self.graph.add_edge(edge)
    else:
        # For start node
        v = pydot.Node(str((number_missionaries, number_cannibals, side, depth_level)),
                       label=str((number_missionaries, number_cannibals, side)))
        self.graph.add_node(v)
    return u, v
```

Hàm draw_edge có nhiệm vụ vẽ các cạnh (edge) trong đồ thị đại diện cho các trạng thái của bài toán "Người truyền giáo và Kẻ ăn thịt", kết nối các trạng thái con với trạng thái cha của chúng để biểu diễn các bước di chuyển giữa các trạng thái.

```
def bfs(self):
    q = deque()
    q.append(self.start_state + (0, ))
    self.visited[self.start_state] = True

    while q:
        number_missionaries, number_cannibals, side, depth_level = q.popleft()

        # Draw Edge from u -> v
        # Where u = Parent[v]
        # and v = (number_missionaries, number_cannibals, side, depth_level)
        u, v = self.draw_edge(number_missionaries, number_cannibals, side, depth_level)

        if self.is_start_state(number_missionaries, number_cannibals, side):
            v.set_style("filled")
            v.set_fillcolor("blue")
            v.set_fontcolor("white")
        elif self.is_goal_state(number_missionaries, number_cannibals, side):
            v.set_style("filled")
            v.set_fillcolor("green")
            return True
        elif self.number_of_cannibals_exceeds(number_missionaries, number_cannibals):
            v.set_style("filled")
            v.set_fillcolor("red")
            continue
        else:
            v.set_style("filled")
            v.set_fillcolor("orange")
```

```
    for x, y in self.options:
        next_m = number_missionaries + op * x
        next_c = number_cannibals + op * y
        next_s = int(not side)

        if (next_m, next_c, next_s) not in self.visited:
            if self.is_valid_move(next_m, next_c):
                can_be_expanded = True
                self.visited[(next_m, next_c, next_s)] = True
                q.append((next_m, next_c, next_s, depth_level + 1))

                # Keep track of parent and corresponding move
                Parent[(next_m, next_c, next_s)] = (number_missionaries, number_cannibals, side)
                Move[(next_m, next_c, next_s)] = (x, y, side)
                node_list[(next_m, next_c, next_s)] = v

    if not can_be_expanded:
        v.set_style("filled")
        v.set_fillcolor("gray")

return False
```

- Tạo một hàng đợi q để lưu trữ trạng thái đang xét.
- Trạng thái bắt đầu (self.start_state) được đưa vào hàng đợi, với độ sâu ban đầu là 0 ((0,)).
- Đánh dấu trạng thái bắt đầu là đã được thăm (self.visited[self.start_state] = True).
- Trong khi hàng đợi không rỗng, lấy phần tử đầu tiên ra (trạng thái hiện tại `number_missionaries`, `number_cannibals`, `side`, `depth_level`).
- Tạo và vẽ cạnh nối từ nút cha `u` đến nút con `v` với thông tin trạng thái (`self.draw_edge()`).
- Nếu là trạng thái bắt đầu (self.is_start_state()), tô màu xanh dương.
- Nếu là trạng thái đích (self.is_goal_state()), tô màu xanh lá và trả về True (kết thúc tìm kiếm thành công).
- Nếu số lượng kẻ ăn thịt người vượt quá số lượng người truyền giáo (self.number_of_cannibals_exceeds()), tô màu đỏ và bỏ qua trạng thái này.
- Duyệt qua tất cả các lựa chọn di chuyển (self.options) để tính toán trạng thái kế tiếp (next_m, next_c, next_s).
- Kiểm tra nếu trạng thái kế tiếp chưa được thăm và là hợp lệ (self.is_valid_move()):
- Đánh dấu trạng thái này là đã thăm.
- Thêm vào hàng đợi với độ sâu tăng thêm 1.
- Lưu thông tin về cha của trạng thái và bước di chuyển tương ứng (Parent và Move).
- Thêm nút vào danh sách các nút (node_list).
- Nếu trạng thái hiện tại không thể mở rộng, tô màu xám để đánh dấu nó là trạng thái ngõ cụt.

```
def dfs(self, number_missionaries, number_cannnibals, side, depth_level):
    self.visited[(number_missionaries, number_cannnibals, side)] = True

    # Draw Edge (from u -> v)
    # Where u = Parent[v]
    u, v = self.draw_edge(number_missionaries, number_cannnibals, side, depth_level)


    if self.is_start_state(number_missionaries, number_cannnibals, side):
        v.set_style("filled")
        v.set_fillcolor("blue")
    elif self.is_goal_state(number_missionaries, number_cannnibals, side):
        v.set_style("filled")
        v.set_fillcolor("green")
        return True
    elif self.number_of_cannibals_exceeds(number_missionaries, number_cannnibals):
        v.set_style("filled")
        v.set_fillcolor("red")
        return False
    else:
        v.set_style("filled")
        v.set_fillcolor("orange")

    solution_found = False
    operation = -1 if side == 1 else 1

    can_be_expanded = False
```

```
    for x, y in self.options:
        next_m, next_c, next_s = number_missionaries + operation * x, number_cannnibals + operation * y, int(not side)

        if (next_m, next_c, next_s) not in self.visited:
            if self.is_valid_move(next_m, next_c):
                can_be_expanded = True
                # Keep track of Parent state and corresponding move
                Parent[(next_m, next_c, next_s)] = (number_missionaries, number_cannnibals, side)
                Move[(next_m, next_c, next_s)] = (x, y, side)
                node_list[(next_m, next_c, next_s)] = v

                solution_found = (solution_found or self.dfs(next_m, next_c, next_s, depth_level + 1))

                if solution_found:
                    return True

    if not can_be_expanded:
        v.set_style("filled")
        v.set_fillcolor("gray")

    self.solved = solution_found
    return solution_found
```

- self.visited[(number_missionaries, number_cannibals, side)] = True: Đánh dấu trạng thái hiện tại là đã thăm để tránh lặp lại trong quá trình tìm kiếm.

- Nếu là **trạng thái bắt đầu** (self.is_start_state()), tô màu xanh dương (v.set_fillcolor("blue")).

- Nếu là **trạng thái đích** (self.is_goal_state()), tô màu xanh lá (v.set_fillcolor("green")) và trả về True (kết thúc thành công).

- Nếu **số kẻ ăn thịt người vượt quá số người truyền giáo** (self.number_of_cannibals_exceeds()), tô màu đỏ (v.set_fillcolor("red")) và trả về False.

- Nếu không thuộc các trường hợp trên, tô màu cam (v.set_fillcolor("orange"))
-