NHẬP MÔN TRÍ TUỆ NHÂN TẠO

Tên : Trần Kim Khanh

Mssv:21110318

BÀI TẬP THỰC HÀNH TUẦN 2

Đọc hiểu code đã cho và trình bày lại chi tiết hơn đoạn code cho sẵn dùng để làm gì:

Mô tả Bài Toán

- **Tình huống**: Có 3 người truyền giáo và 3 con quỷ ở bờ trái của một con sông, và họ cần di chuyển qua bờ phải mà không để ai bị ăn thịt.
- **Điều kiện**:

  Số người truyền giáo phải luôn lớn hơn hoặc bằng số con quỷ ở cùng một bờ sông. Nếu số quỷ nhiều hơn số người truyền giáo ở một bờ thì những con quỷ sẽ ăn thịt người truyền giáo tại bờ đó.

  Có một chiếc thuyền, và chiếc thuyền này có thể chở tối đa 2 người (có thể là người hoặc quỷ).

- **Mục tiêu**: Tìm cách để đưa tất cả người truyền giáo và con quỷ sang bờ bên phải an toàn mà không vi phạm điều kiện an toàn.

  Biểu Diễn Không Gian Trạng Thái

  Mỗi trạng thái có thể được biểu diễn bằng một bộ ba số (a, b, k):

  - **a** là số người truyền giáo ở bờ trái.
  - **b** là số con quỷ ở bờ trái.
  - **k** là vị trí của thuyền:

    k = 1 nếu thuyền ở bờ trái.

    k = 0 nếu thuyền ở bờ phải.

  Trạng Thái Bắt Đầu

- Trạng thái ban đầu là (3, 3, 1), có nghĩa là cả 3 người truyền giáo, 3 con quỷ và thuyền đều ở bờ trái.

Trạng Thái Kết Thúc

- Trạng thái kết thúc là (0, 0, 0), tức là tất cả mọi người đã được đưa sang bờ phải an toàn

Các hàm kiểm tra trạng thái:

genarate_full_space_tree :

Tạo đố tượng đồ thị :

graph = pydot.Dot(graph_type='graph',

strict=False,

bgcolor="#fff3af",

label="fig: Missionaries and Cannibal State Space Tree",

fontcolor="red",

fontsize="24",

overlap="true")

Thiết lập đối số dòng lệnh :

```
# Thiết lập argparse để nhận đối số từ dòng lệnh
arg_parser = argparse.ArgumentParser()
arg_parser.add_argument("--depth", required=False, help="Maximum depth up to which you want to generate the State Spa
args = vars(arg_parser.parse_args())

# Lấy độ sâu tối đa từ đối số
# python generate_full_space_tree.py --depth 5 ví dụ
max_depth = int(args.get("depth", 20))
```

max của depth chỉ tới 20.

```
def is_valid_move(number_missionaries, number_cannibals):
    """Kiểm tra xem các ràng buộc về số lượng có được thỏa mãn không."""
    return (0 <= number_missionaries <= 3) and (0 <= number_cannibals <= 3)

def write_image(file_name="state_space"):
    """Ghi hình ảnh đồ thị vào tệp PNG."""
    try:
        graph.write_png(f"{file_name}_{max_depth}.png")
        print(f"File {file_name}_{max_depth}.png successfully written.")
    except Exception as e:
        print("Error while writing file:", e)
```

is_valid_move: Kiểm tra xem số lượng người truyền giáo và thổ dân có nằm trong giới hạn (0 đến 3) hay không.

write_image: Ghi hình ảnh của đồ thị ra tệp PNG.

Vẽ các cạnh trong đồ thị:

```
def draw_edge(number_missionaries, number_cannibals, side, depth_level, node_num):
    u, v = None, None

    # Kiểm tra xem có nút cha không
    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)

        # Tạo nút cho nút hiện tại
        v = pydot.Node(str((number_missionaries, number_cannibals, side, depth_level, node_num)),
                   label=str((number_missionaries, number_cannibals, side)))
        graph.add_node(v)
        #! kiểm  tra thay Node thành
        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
```

Các hàm trạng thái:

is_start_state: Kiểm tra trạng thái bắt đầu của bài toán.

is_goal_state: Kiểm tra trạng thái mục tiêu.

number_of_cannibals_exceeds: Kiểm tra xem số lượng thổ dân có vượt quá số lượng người truyền giáo không, đảm bảo an toàn cho người truyền giáo.

def generate(): Đây là hàm chính để giai bài toán tìm kiếm theo chiều rộng. Hàm trả về true nếu đi đến độ sâu nhất của cây, ngược lại trả về false

lênh chạy để vẽ ra hình chi tiết –depth [tham số độ sâu] , tối đa trong bài là 20

--depth 9 :

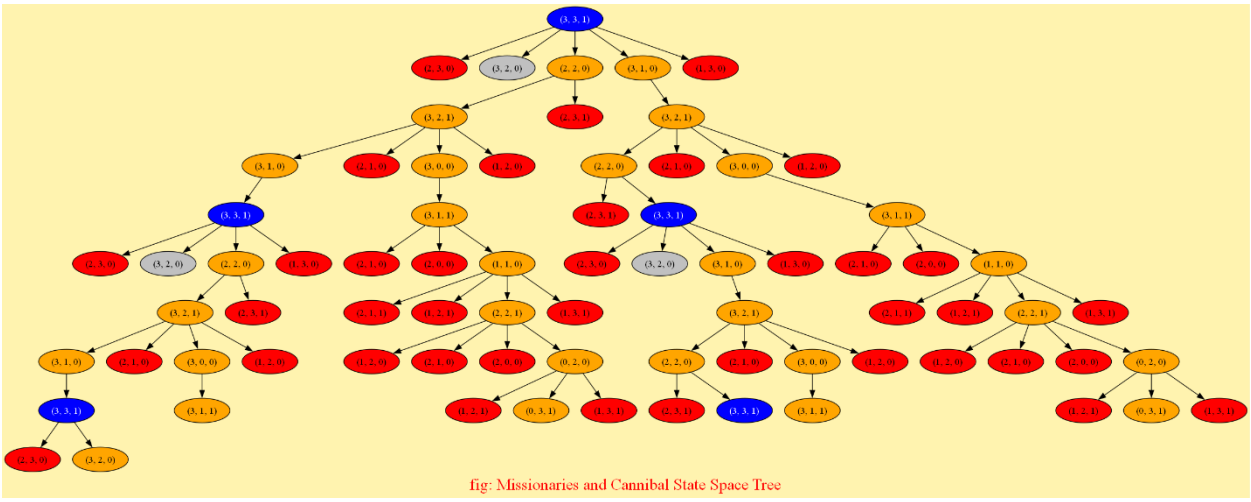
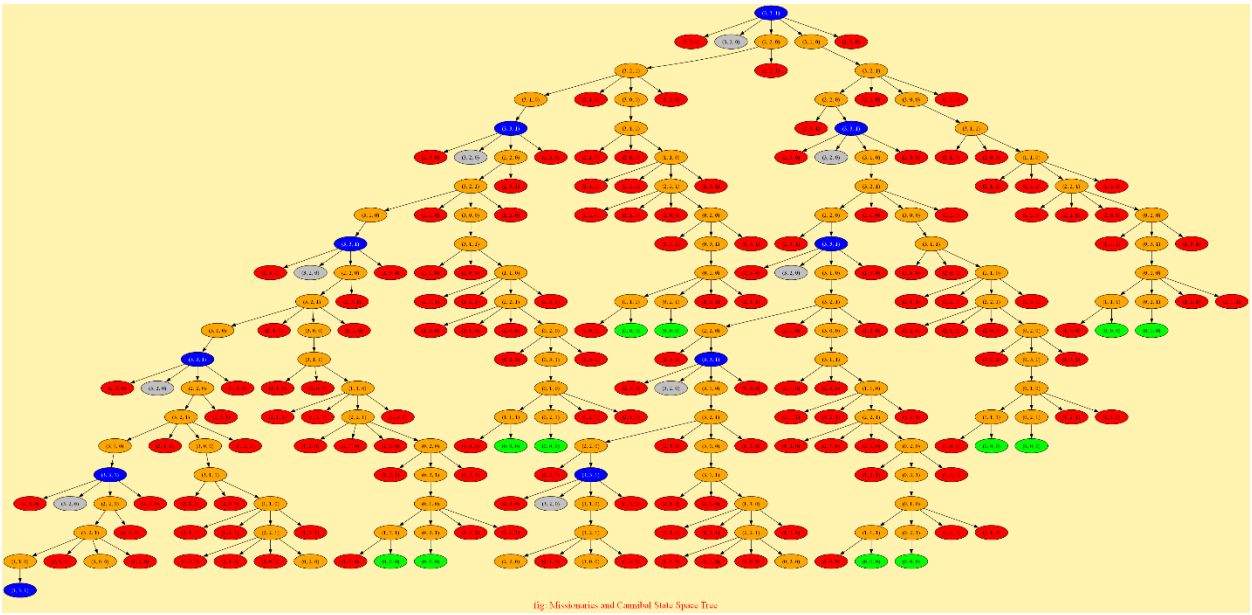


fig: Missionaries and Cannibal State Space Tree

depth : 20



File solve.py

self.start_state = (3, 3, 1) trạng thái ban đầu

self.goal_state = (0, 0, 0) trạng thái kết thúc

self.options = [(1, 0), (0, 1), (1, 1), (0, 2), (2, 0)] chuyển trạng thái của bài toán

self.boat_side = ["right", "left"] bờ bên phải và bở trên trái

self.visited: Từ điển để theo dõi các trạng thái đã thăm.

self.solved: Biến boolean để theo dõi trạng thái đã được giải quyết hay chưa.

is_valid_move(self, number_missionaries, number_cannibals): Kiểm tra xem các giới hạn về số lượng người truyền giáo và quỷ có được thoả mãn hay không. Hàm trả về True nếu số lượng nằm trong khoảng từ 0 đến 3.

is_start_state(self, number_missionaries, number_cannibals, side):

Kiểm tra xem trạng thái hiện tại có phải là trạng thái bắt đầu hay không. Hàm trả về True nếu trạng thái hiện tại khớp với self.start_state.

is_goal_state(self, number_missionaries, number_cannibals, side):

Kiểm tra xem trạng thái hiện tại có phải là trạng thái mục tiêu hay không. Hàm trả về True nếu trạng thái hiện tại khớp với self.goal_state.

number_of_cannibals_exceeds(self, number_missionaries, number_cannibals): kiểm tra số lượng quỷ và truyền giáo có bị ăn hết không

write_image(self, file_name="state_space.png"):

Ghi đồ thị trạng thái vào một tệp hình ảnh PNG. Nếu có lỗi xảy ra trong quá trình ghi, hàm sẽ in ra thông báo lỗi.

solve(self, solve_method="bfs"):

Hàm giải quyết bài toán bằng cách gọi thuật toán duyệt theo chiều sâu (DFS) hoặc theo chiều rộng (BFS), tuỳ thuộc vào tham số solve_method.

+Nếu solve_method là "dfs", hàm sẽ gọi self.dfs(...).

+Nếu không, hàm sẽ gọi self.bfs().

draw_legend(self)

Vẽ một chú giải cho đồ thị để giải thích các loại nút khác nhau, bao gồm nút bắt đầu, nút mục tiêu, nút không hợp lệ, và nút giải pháp

draw(self, *, number_missionaries_left, number_cannibals_left, number_missionaries_right, number_cannibals_right):

Vẽ trạng thái hiện tại của bài toán trên console bằng cách sử dụng emoji để biểu diễn người truyền giáo và thịt. Nó sẽ in ra số lượng người truyền giáo và thịt ở cả hai bên của thuyền.

show_solution(self):

Hiển thị các bước giải quyết bài toán từ trạng thái mục tiêu đến trạng thái bắt đầu. Nó sử dụng thông tin về các bước đi và các nút trong quá trình giải quyết để in ra từng bước di chuyển

draw_edge(self, number_missionaries, number_cannibals, side, depth_level):

Vẽ các cạnh giữa các nút trong đồ thị trạng thái. Nó tạo ra các nút cho trạng thái hiện tại và trạng thái cha, và thêm cạnh giữa chúng.

bfs(self)

Thuật toán tìm kiếm theo chiều rộng để tìm ra giải pháp cho bài toán

Main.py:

-m hoặc --method: Tham số để chỉ định phương pháp giải (mặc định là "bfs" nếu không chỉ định).

-l hoặc --legend: Tham số để chỉ định có hiển thị chú giải trên đồ thị hay không.

Giải bài toán DFS:

Lệnh chạy: python main.py -m dfs

```
Step 1: Move 1 missionaries and 1 cannibalsfrom left to right
:old man: :old man: ogre: ogre: ____________________________________:old man: ogre:

Step 2: Move 1 missionaries and 0 cannibalsfrom right to left
:old man: :old man: :old man: ogre: ogre: ____________________________________      ogre:

Step 3: Move 0 missionaries and 2 cannibalsfrom left to right
:old man: :old man: :old man: ____________________________________ogre: ogre: ogre:

Step 4: Move 0 missionaries and 1 cannibalsfrom right to left
:old man: :old man: :old man: ogre: ____________________________________ogre: ogre:

Step 5: Move 2 missionaries and 0 cannibalsfrom left to right
:old man: ogre: ____________________________________:old man: :old man: ogre: ogre:

Step 6: Move 1 missionaries and 1 cannibalsfrom right to left
:old man: :old man: ogre: ogre: ____________________________________:old man: ogre:

Step 7: Move 2 missionaries and 0 cannibalsfrom left to right
ogre: ogre:   ____________________________________:old man: :old man: :old man: ogre:

Step 8: Move 0 missionaries and 1 cannibalsfrom right to left
ogre: ogre: ogre: ____________________________________:old man: :old man: :old man:

Step 9: Move 0 missionaries and 2 cannibalsfrom left to right
ogre:         ____________________________________:old man: :old man: :old man: ogre: ogre:

Step 10: Move 1 missionaries and 0 cannibalsfrom right to left
:old man: ogre: ____________________________________:old man: :old man: ogre: ogre:

Step 11: Move 1 missionaries and 1 cannibalsfrom left to right
              ____________________________________:old man: :old man: :old man: ogre: ogre: ogre:
```

ảnh của thuật toán theo cây

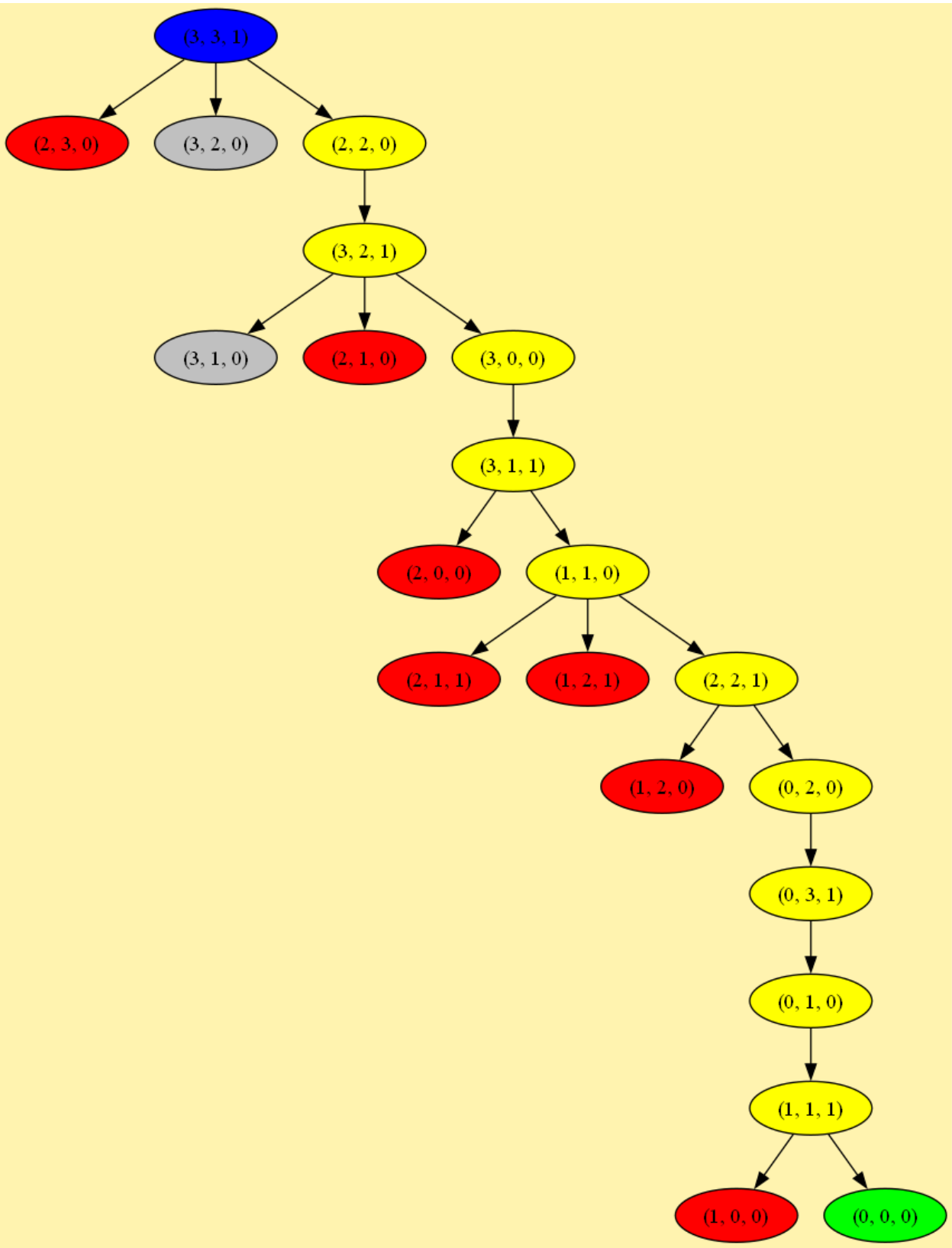


fig: Missionaries and Cannibal State Space Tree

python main.py -m dfs -l True lệnh chạy DFS co chú thích

```
Step 1: Move 1 missionaries and 1 cannibalsfrom left to right
:old man: :old man: ogre: ogre: _____________________________________:old man: ogre:

Step 2: Move 1 missionaries and 0 cannibalsfrom right to left
:old man: :old man: :old man: ogre: ogre: _____________________________________      ogre:

Step 3: Move 0 missionaries and 2 cannibalsfrom left to right
:old man: :old man: :old man: _____________________________________ogre: ogre: ogre:

Step 4: Move 0 missionaries and 1 cannibalsfrom right to left
:old man: :old man: :old man: ogre: _____________________________________ogre: ogre:

Step 5: Move 2 missionaries and 0 cannibalsfrom left to right
:old man: ogre: _____________________________________:old man: :old man: ogre: ogre:

Step 6: Move 1 missionaries and 1 cannibalsfrom right to left
:old man: :old man: ogre: ogre: _____________________________________:old man: ogre:

Step 7: Move 2 missionaries and 0 cannibalsfrom left to right
ogre: ogre:   _____________________________________:old man: :old man: :old man: ogre:

Step 8: Move 0 missionaries and 1 cannibalsfrom right to left
ogre: ogre: ogre: _____________________________________:old man: :old man: :old man:

Step 9: Move 0 missionaries and 2 cannibalsfrom left to right
ogre:         _____________________________________:old man: :old man: :old man: ogre: ogre:

Step 10: Move 1 missionaries and 0 cannibalsfrom right to left
:old man: ogre: _____________________________________:old man: :old man: ogre: ogre:

Step 11: Move 1 missionaries and 1 cannibalsfrom left to right
             _____________________________________:old man: :old man: :old man: ogre: ogre: ogre:
```

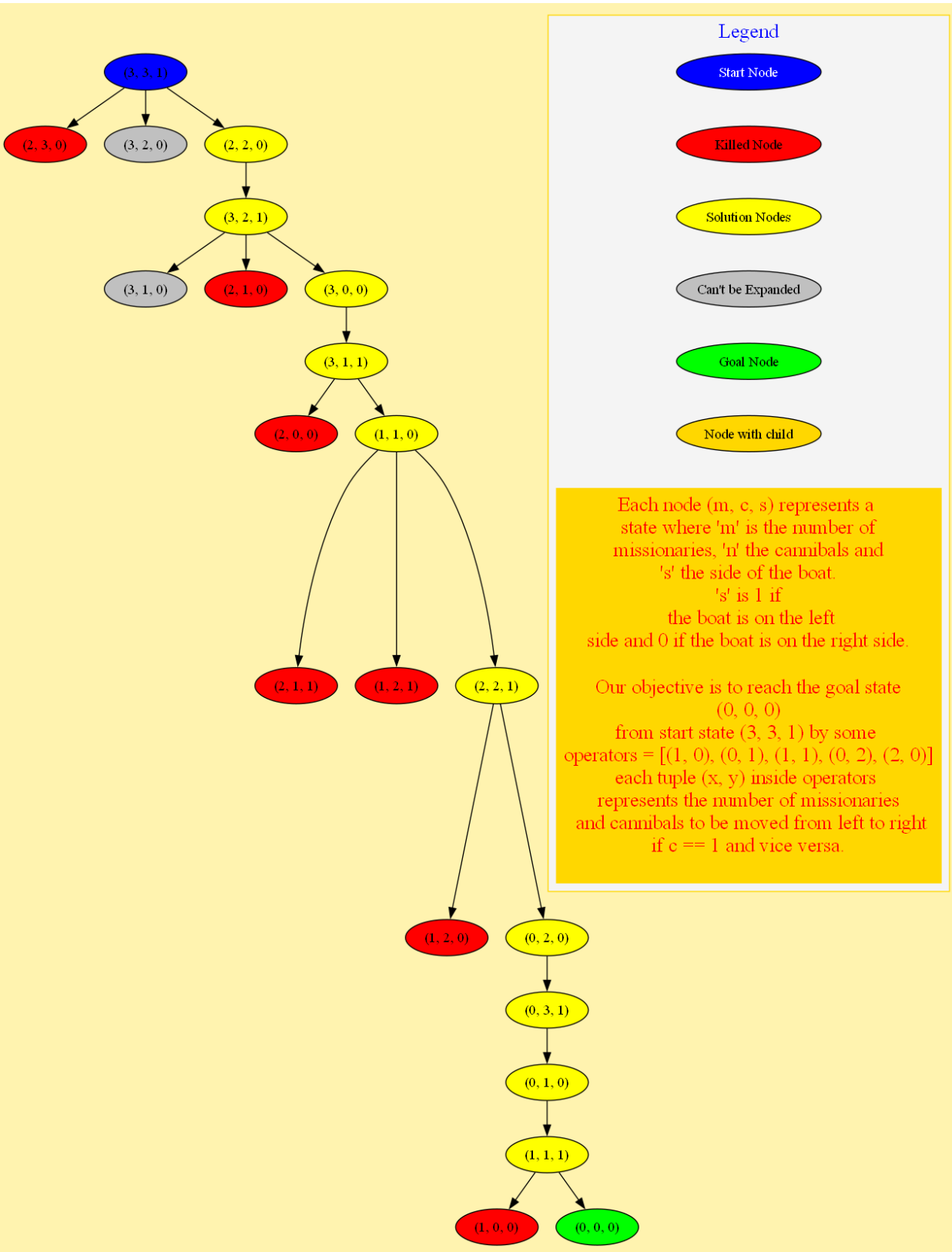
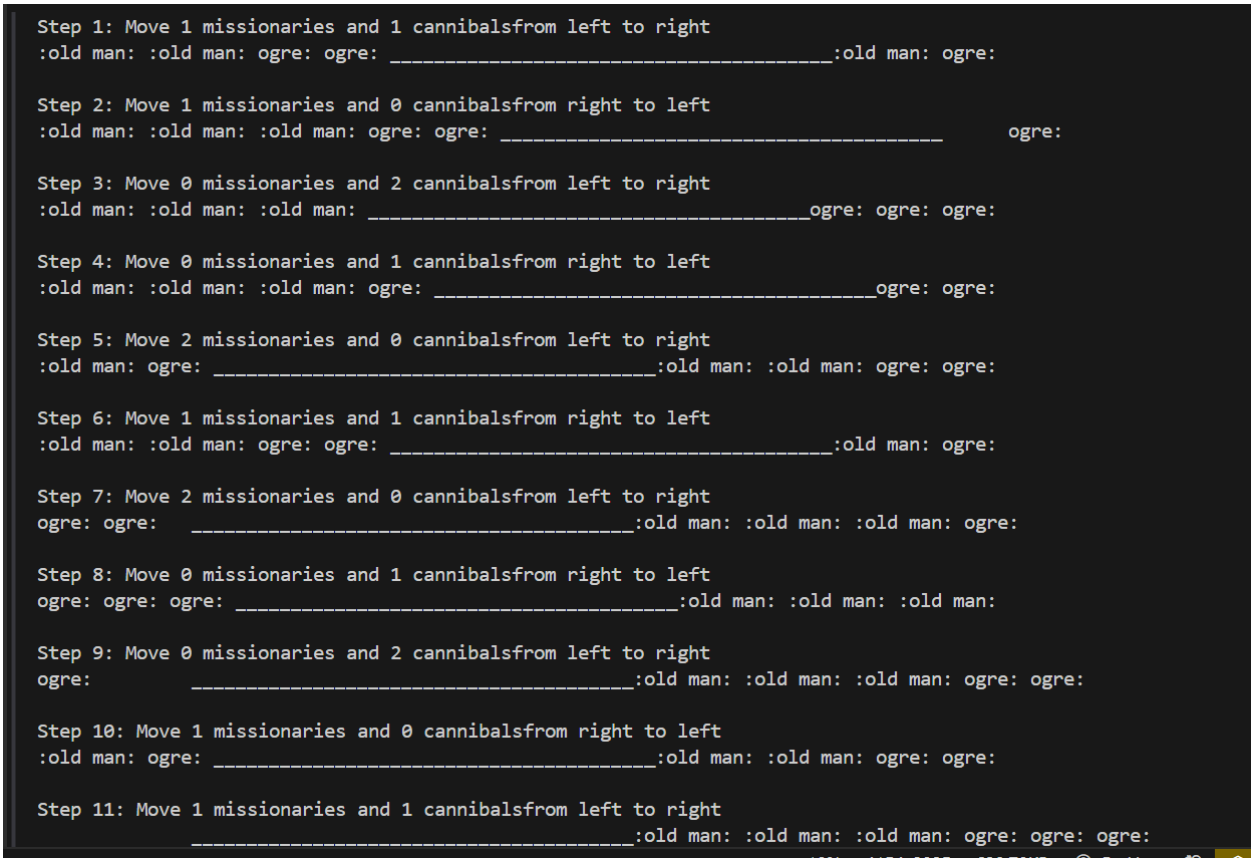


fig: Missionaries and Cannibal State Space Tree

Giải thuật BFS : python main.py -m bfs



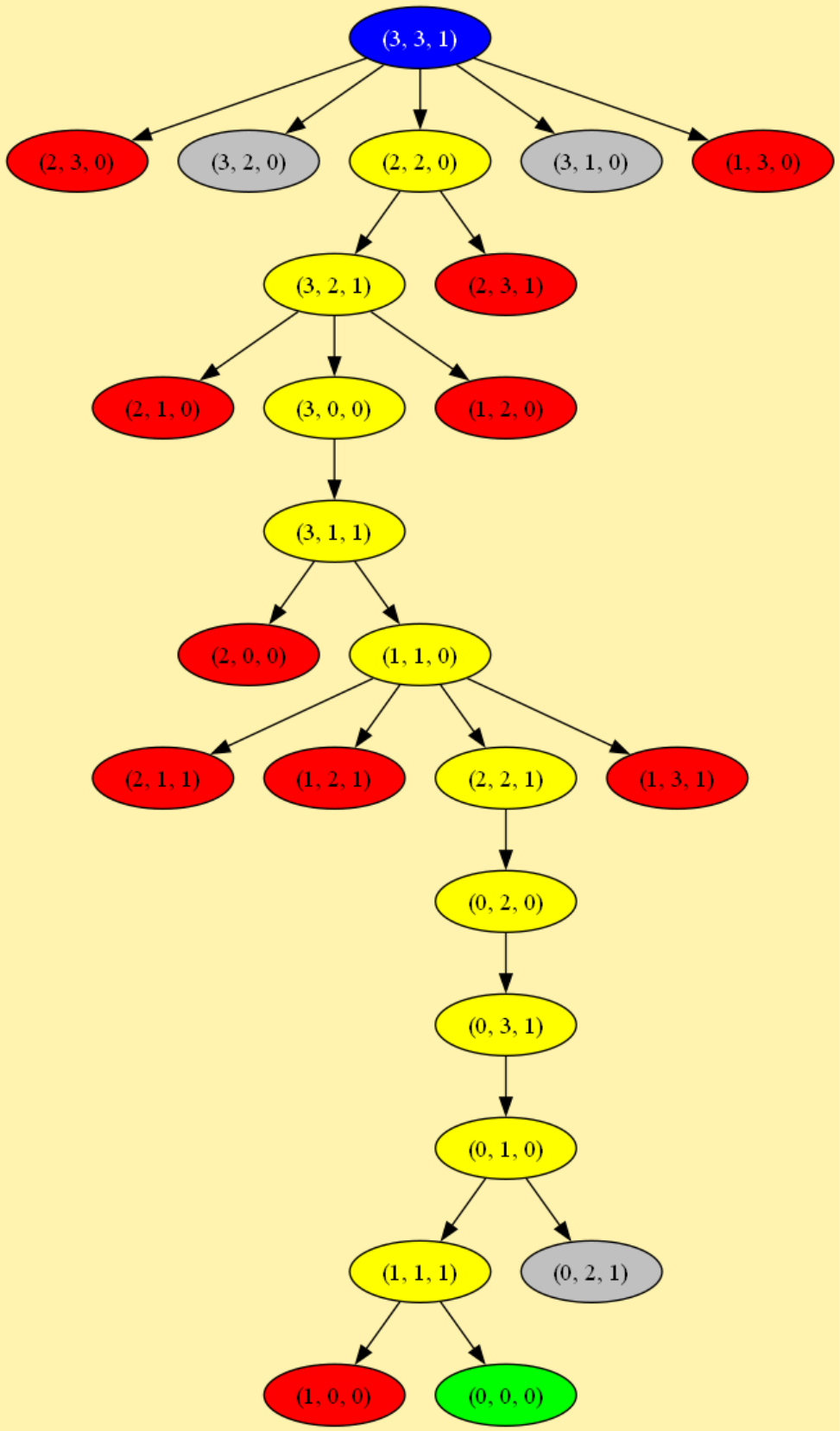
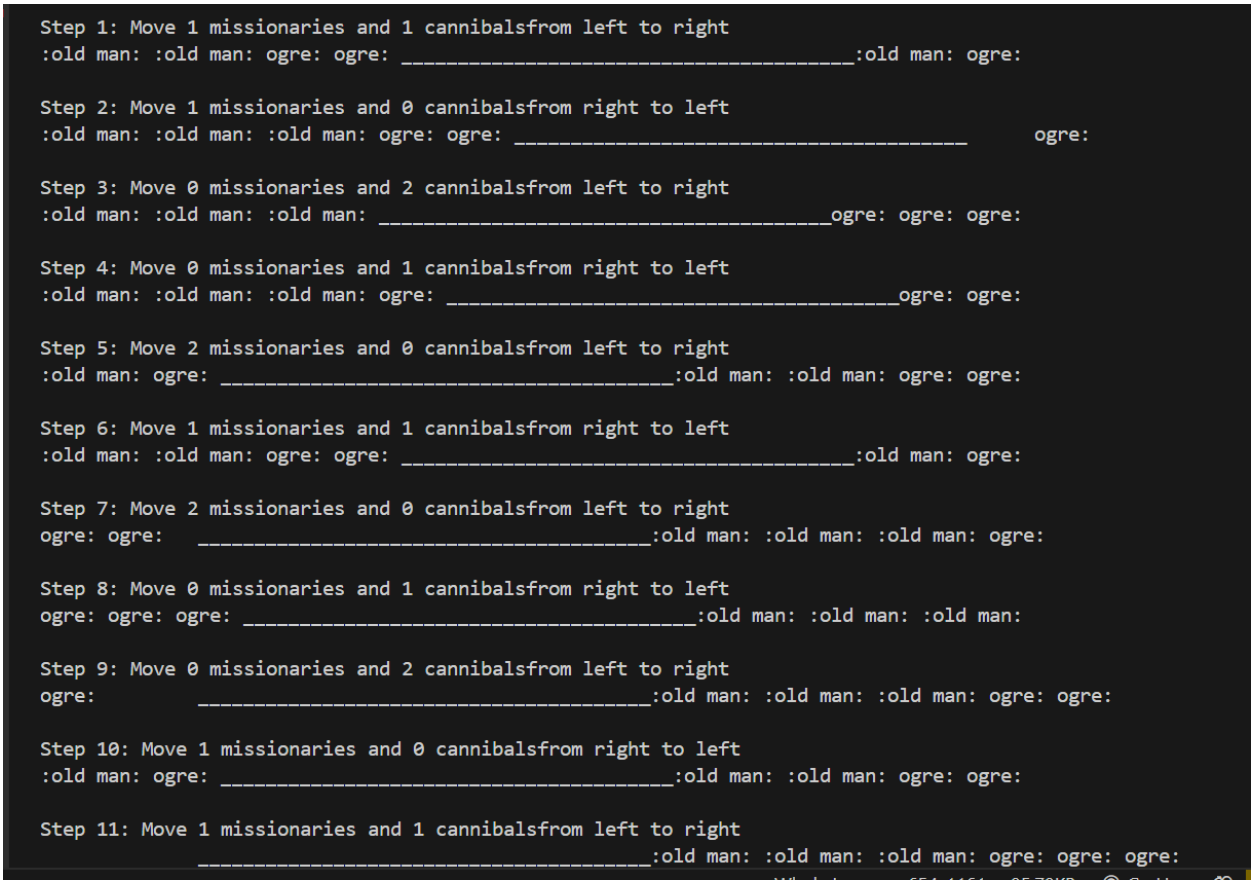


fig: Missionaries and Cannibal State Space Tree

BFS với legend: python main.py -m bfs -l True



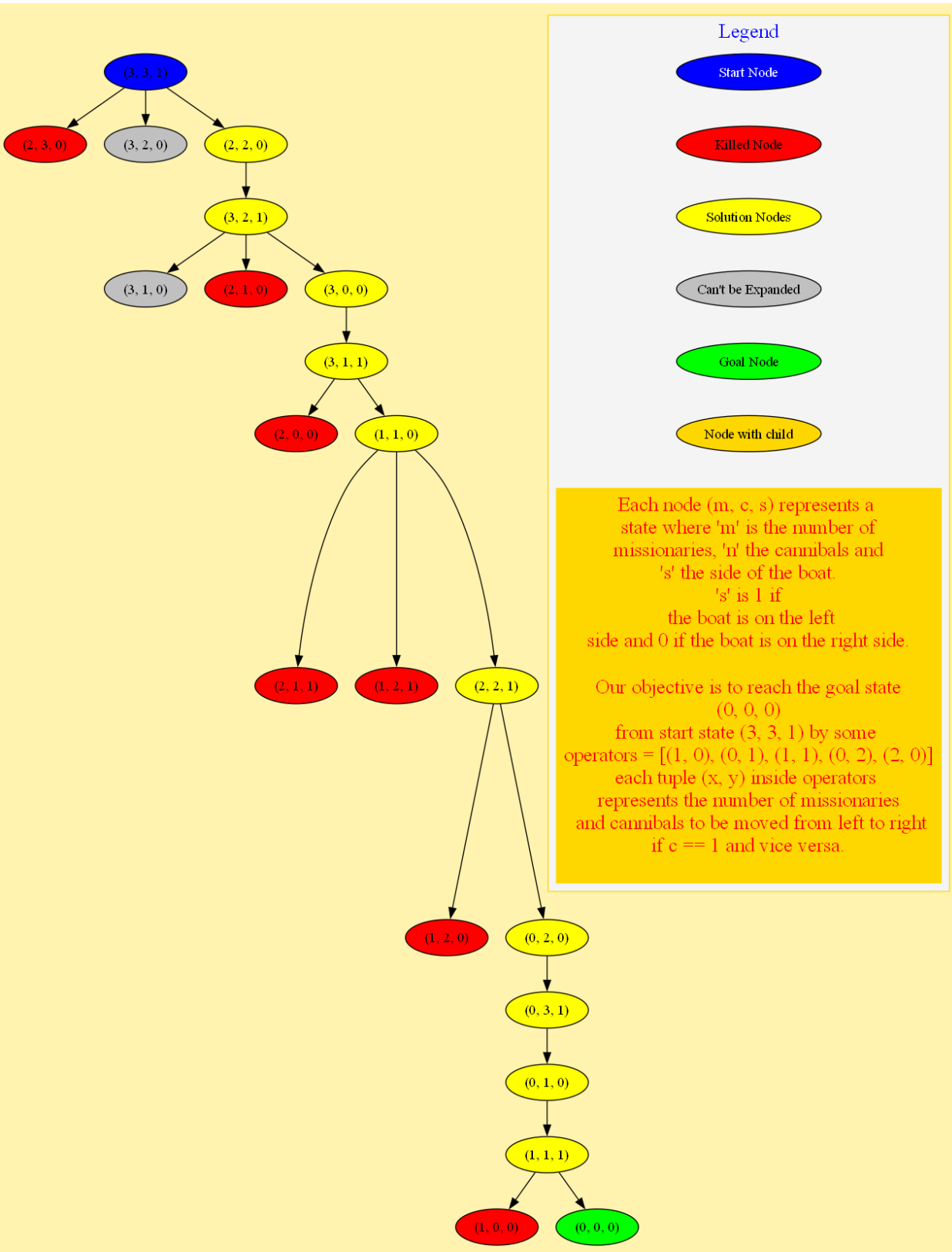


fig: Missionaries and Cannibal State Space Tree

Nhận xét : trong quá trình thử nghiệm với hai thuật toán tìm kiếm là Depth-First Search (DFS) và Breadth-First Search (BFS), cả hai thuật toán đều tìm ra được nghiệm cho bài toán. Tuy nhiên, thuật toán DFS đã hoàn thành việc tìm kiếm nhanh hơn so với thuật toán BFS

Thuật toán DFS: Tìm kiếm theo chiều sâu, nghĩa là nó mở rộng các nút (nodes) ở các cấp độ thấp nhất trước. Điều này cho phép thuật toán tập trung vào việc khám phá một nhánh cụ thể trong cây tìm kiếm cho đến khi nó đạt được một nút cuối cùng (goal node) hoặc không còn nút nào để mở rộng. Nhờ vậy, DFS thường tìm được nghiệm sớm hơn, đặc biệt trong những bài toán có nhiều nhánh nhưng ít nút sâu hơn.

Thuật toán BFS: Ngược lại, BFS tìm kiếm theo chiều rộng, mở rộng các nút ở các cấp độ thấp nhất theo thứ tự từ trái sang phải. Điều này dẫn đến việc BFS phải kiểm tra nhiều nút hơn ở các cấp độ cao trước khi tiếp cận những nút ở các cấp độ thấp hơn. Kết quả là, BFS thường mất nhiều thời gian hơn để tìm ra nghiệm, nhất là khi cây tìm kiếm có cấu trúc rộng rãi.