BTTH - NM TTNT - TUAN 2

21280099 - Nguyễn Công Hoài Nam

Ngày 6 tháng 11 năm 2023

I. Cài đặt chương trình

1. Kiểm tra chương trình

Về tổng thể chương trình hoạt động đúng, tuy nhiên còn một lỗi nhỏ

```
# file generate-full-space-tree.py
arg = argparse.ArgumentParser()
arg.add_argument("-d", "--depth", required=False,
                 help="MAximum depth upto which you want to generate Space State Tree")

args = vars(arg.parse_args())

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

Đoạn code trên dùng để tạo và lấy đối số depth tức là độ sâu của cây trạng thái
Ở đây nếu khi ta tạo cây trạng thái mà không truyền đối số depth trong lệnh thì khi này chương trình sẽ lấy giá trị depth mặc định là 20.
Tuy nhiên khi lấy **args = vars(arg.parse_args())** thì depth sẽ là một key của dict args và được gán bằng None (nếu không được truyền depth).
Vậy nên chương trình sẽ hiểu là là lấy dữ liệu thành công thay vì lấy depth mặc định là 20. Khi đó, chương trình sẽ báo lỗi do ép kiểu int cho None

```
[nchn@fedora Tuần_2]$  cd /run/media/nchn/Data/Course\ HK1\ 23-24/Introduce\ to\ AI/Tuần_2-20231103T075512Z-001/Tuần_2 ; /usr/bin/env /bin/python /home/nchn/.vscode/extensions/ms-python.python-2023.20.0/pythonFiles/lib/python/debugpy/adapter/../../debugpy/launcher 56503 -- /run/media/nchn/Data/Course\ HK1\ 23-24/Introduce\ to\ AI/Tuần_2-20231103T075512Z-001/Tuần_2/generate_full_space_tree.py
Traceback (most recent call last):
  File "/run/media/nchn/Data/Course HK1 23-24/Introduce to AI/Tuần_2-20231103T075512Z-001/Tuần_2/generate_full_space_tree.py", line 28, in <module>
    max_depth = int(args.get("depth", 20))
                ^^^^^^^^^^^^^^^^^^^^^^^^^
TypeError: int() argument must be a string, a bytes-like object or a real number, not 'NoneType'
```

Hình 1: Lỗi

Sửa lại code như sau

```
# Cac doan code o tren khong thay doi
# max_depth = int(args.get("depth", 20))
# Sua lai
depth_argument = args.get("depth")
# max_depth = 20 neu depth_argument = None
max_depth = int(depth_argument) if depth_argument is not None else 20
```

Ở file **main.py** cũng có lỗi tương tự

```
arg = argparse.ArgumentParser()
arg.add_argument("-m", "--method", required=False, help="Specify which method to use")
arg.add_argument("-l", "--legend", required=False, help="Specify if you want to display legend on graph")

args = vars(arg.parse_args())

solve_method = args.get("method")
solve_method = solve_args.get("method","bfs")
legend_flag = args.get("legend",False)
```

Chương trình không báo lỗi nhưng đối số **m,l** sẽ là None thay vì **bfs, False**.
Sửa lại code như sau

```
solve_method = args.get("method") if args.get("method") is not None else "bfs"
legend_flag = args.get("legend") if args.get("legend") is not None else False
```

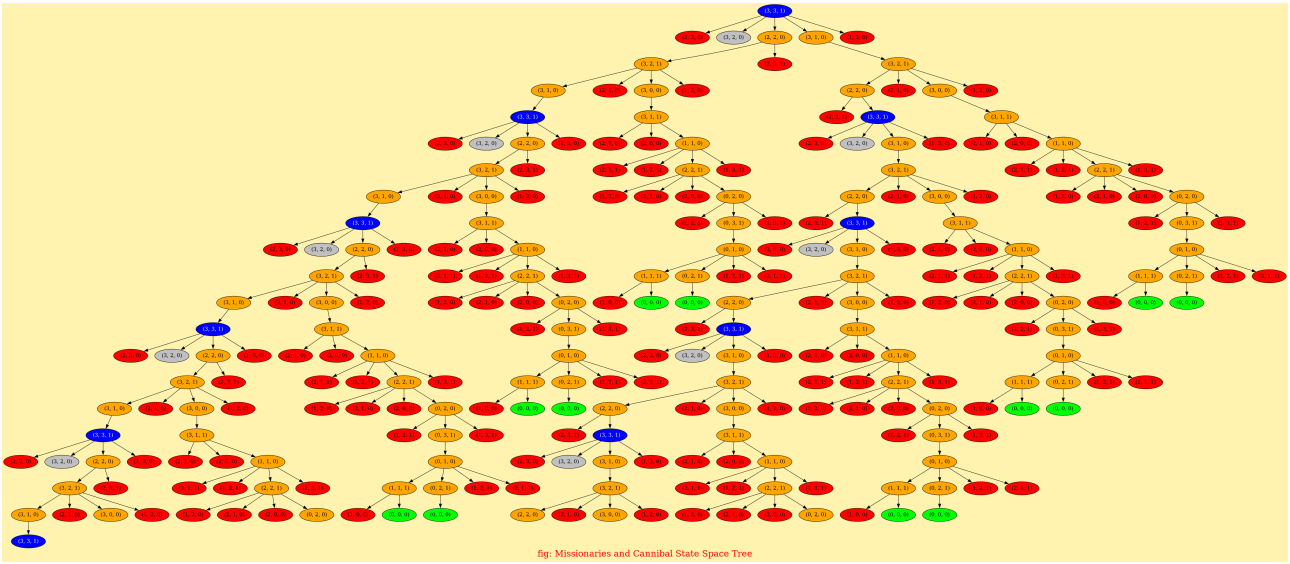
2. Thực thi chương trình

a. Tạo cây trạng thái

Tạo cây trạng thái với depth = 20
python generate_full_space_tree.py -d 20

```
File bfs_legend.png successfully written.
[nchn@192 Tuần_2]$ python generate_full_space_tree.py -d 20
File state_space_20.png successfully written.
```

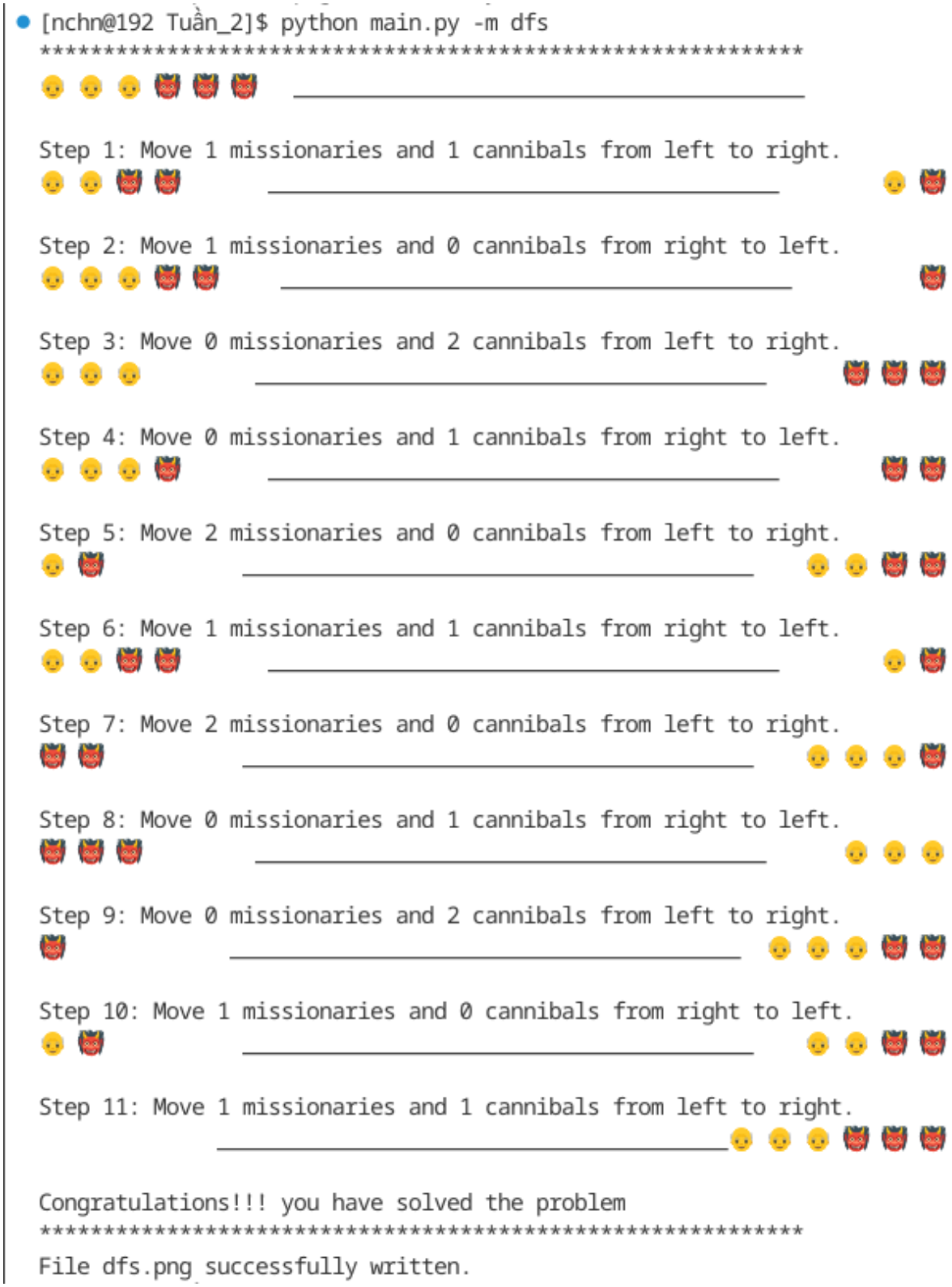
Hình 2: Tạo cây trạng thái - Console



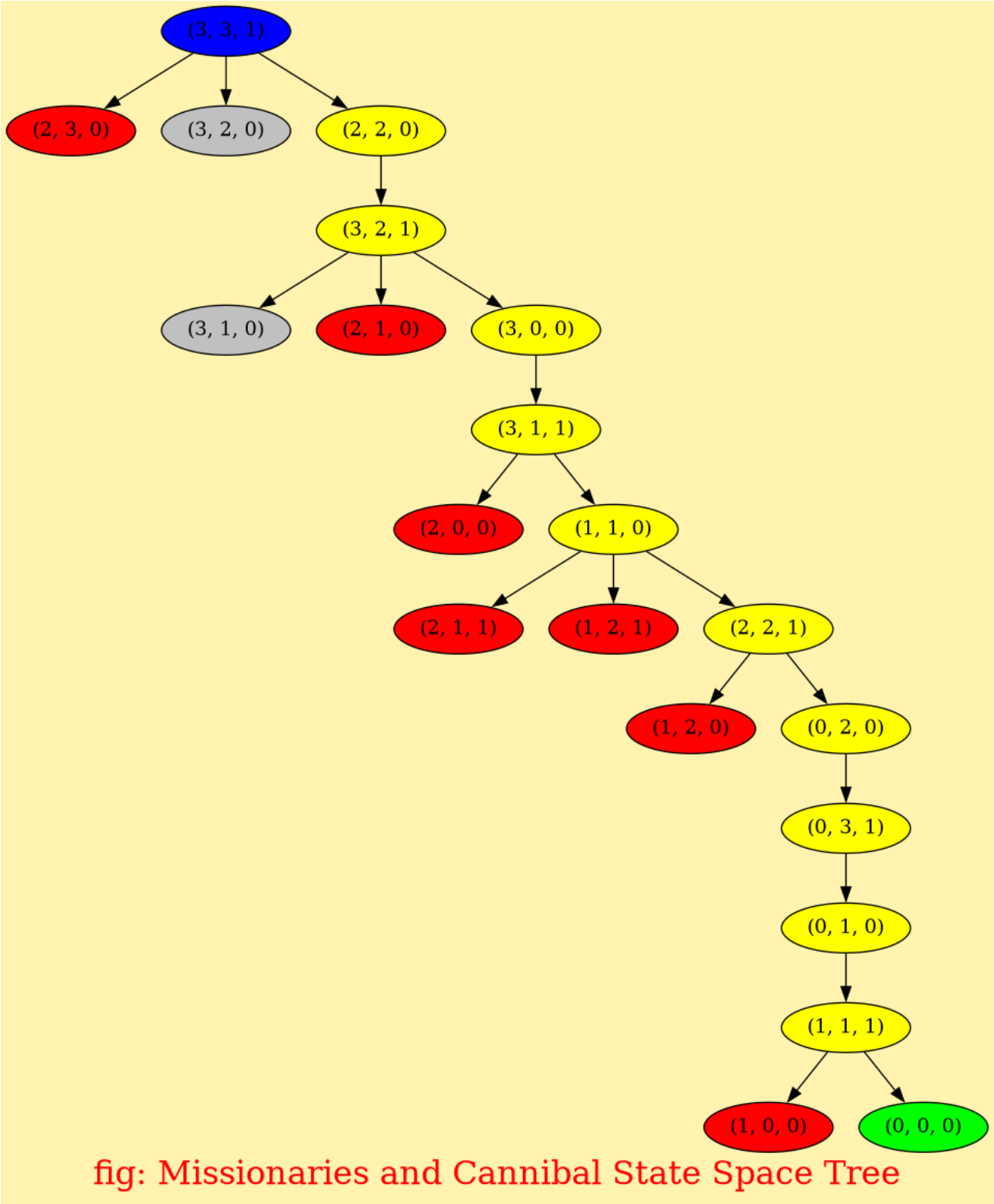
Hình 3: Ảnh cây trạng thái

b. Cây DFS

Chạy DFS
python main.py -m dfs



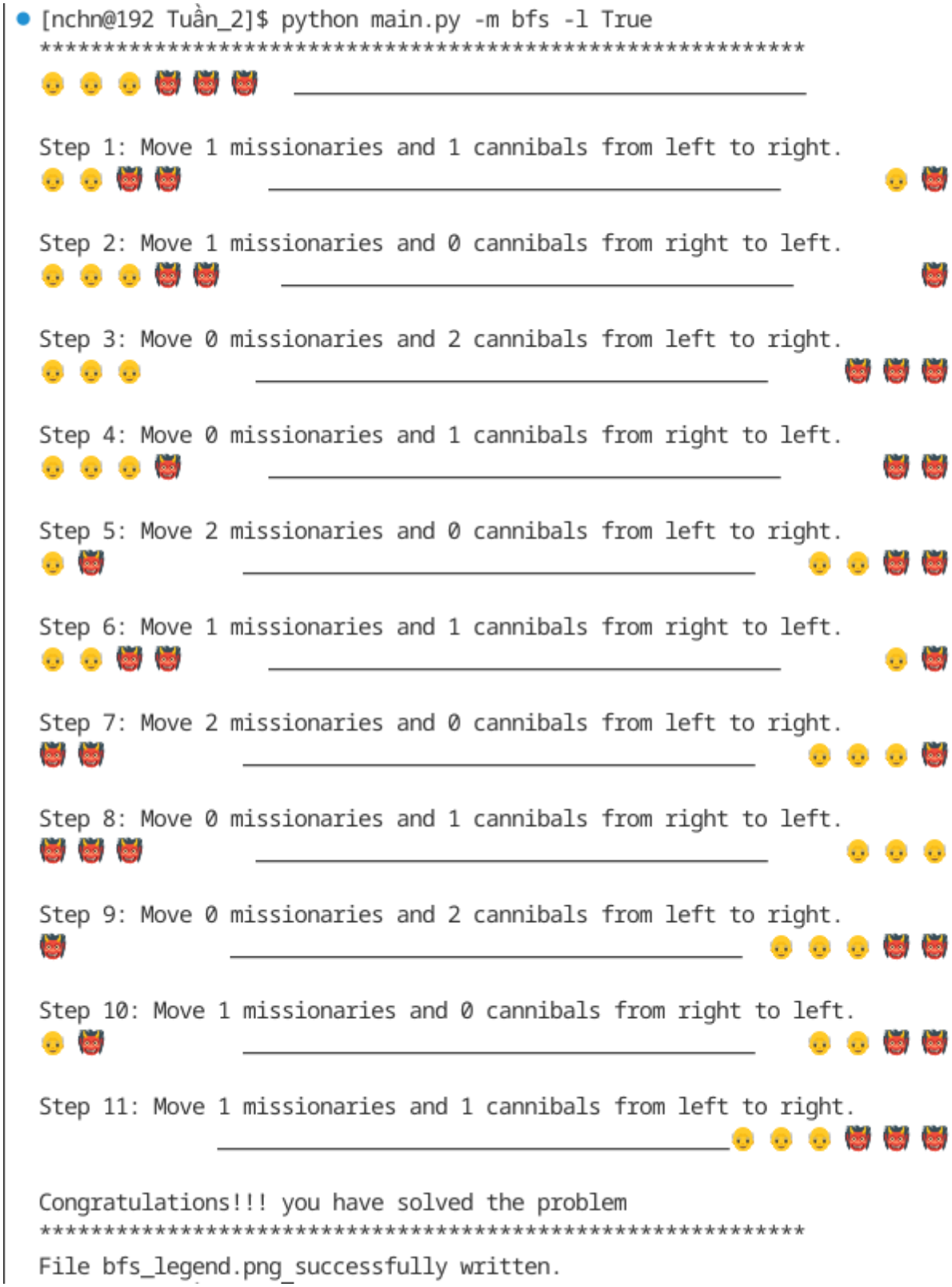
Hình 4: Thuật toán DFS - Console



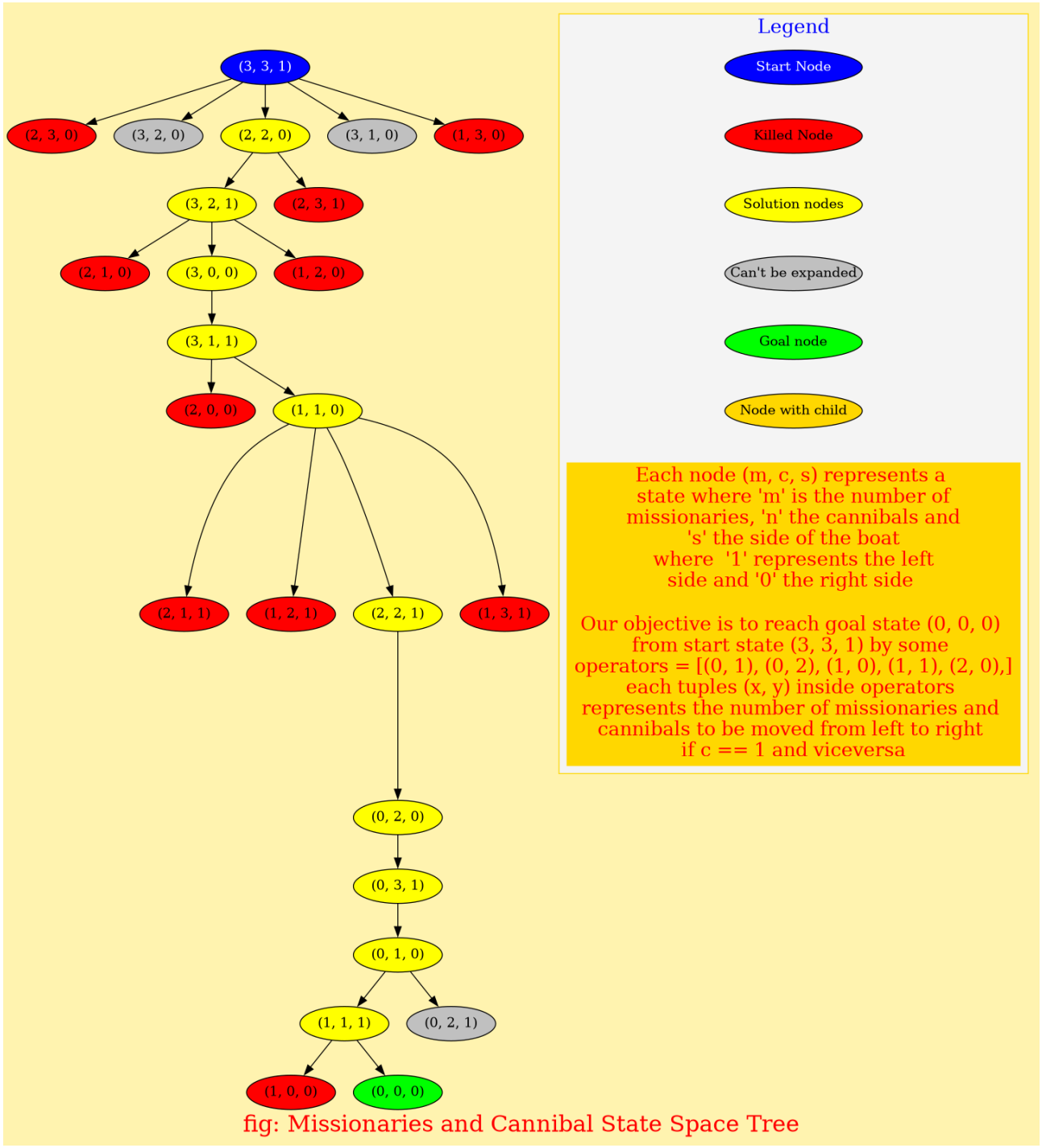
Hình 5: Ảnh xuất thuật toán DFS

c. Cây BFS

Chạy BFS với legend (chú thích)
python main.py -m bfs -l True



Hình 6: Thuật toán DFS - Console



Hình 7: Ảnh xuất thuật toán DFS

II. Giải thích code

1. File generate-full-space-tree.py

Import các thư viện cần thiết

```
from collections import deque
import pydot
import argparse
import os
```

Set path cho Graphviz (thư viện để vẽ cây đồ thị)

```
# Set it to bin folder of graphviz
os.environ["PATH"] += os.pathsep + 'C:\Program Files\Graphviz\bin'
```

Khởi tạo một số biến

```
#So cach di chuyen voi (x,y) la so nguoi va quy tren thuyen
options = [(1, 0), (0, 1), (1, 1), (0, 2), (2, 0)]
#Khoi tao dict Parent luu Node cha cua cac Node
Parent = dict()
#Tao graph voi cac thuoc tinh nhu mau chu, mau nen, kieu graph, co chu, ...
graph = pydot.Dot(graph_type='graph',strict=False, bgcolor="#fff3af",
                  label="fig: Missionaries and Cannibal State Space Tree",
                  fontcolor="red", fontsize="24", overlap="true")
```

Biến i để theo dõi số lượng node trong cây, thiết lập đối số depth (-d) là độ sâu tối đa và mặc định là 20

```
#To track node
i = 0
arg = argparse.ArgumentParser()
arg.add_argument("-d", "--depth", required=False,
                 help="MAximum depth upto which you want to generate Space State Tree")
args = vars(arg.parse_args())
depth_argument = args.get("depth")
max_depth = int(depth_argument) if depth_argument is not None else 20
```

Hàm boolean trả về các di chuyển hợp lệ (số người và quỷ tối đa là 3 và không âm)

```
def is_valid_move(number_missionaries, number_cannnibals):
    """
    Checks if number constraints are satisfied
    """
    return (0 <= number_missionaries <= 3) and (0 <= number_cannnibals <= 3)
```

Hàm xuất ảnh của cây đồ thị có dạng **{file_name}_{max_depth}.png**

file_name input của hàm, mặc định là 'state_space'

max_depth độ sâu của cây đồ thị

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

Hàm vẽ đường đi (cạnh) giữa hai node có 5 tham số đầu vào

number_missionaries: số lượng người

number_cannnibals: số lượng quỷ

side: vị trị của thuyền (1 nếu bờ trái và ngược lại)

depth_level: độ sâu của node

node_num: số thứ tự node

```
def draw_edge(number_missionaries, number_cannnibals, side, depth_level, node_num):
    # Khoi tao u,v la None
    u, v = None, None
    # Neu node cha cua node khong la None (khong phai start node)
    if Parent[(number_missionaries, number_cannnibals, side, depth_level, node_num)] is not None:
        # tao node u va them u vao cay voi cac thong so
        u = pydot.Node(str(Parent[(number_missionaries, number_cannnibals,side,depth_level, node_num)]),
        label=str(Parent[(number_missionaries,number_cannnibals, side, depth_level, node_num)][:3]))
        graph.add_node(u)

        # Tuong tu voi v
        v = pydot.Node(str((number_missionaries, number_cannnibals, side, depth_level, node_num)),
        label=str((number_missionaries, number_cannnibals, side)))
        graph.add_node(v)

        #Ve canh noi u -> v va them vao cay
        edge = pydot.Edge(str(Parent[(number_missionaries, number_cannnibals, side, depth_level, node_num)]),
        str((number_missionaries, number_cannnibals, side, depth_level, node_num) ), dir='forward')
        graph.add_edge(edge)
    # Neu node cha la None thi node do la start node
    else:
        # Tao va them start node vao cay
        v = pydot.Node(str((number_missionaries, number_cannnibals, side, depth_level, node_num)),
        label=str((number_missionaries, number_cannnibals, side)))
        graph.add_node(v)
    # Tra ve hai node u,v
    return u, v
```

Hàm boolean **is_start_state** trả về node có phải là node gốc không (3,3,1)

```
def is_start_state(number_missionaries, number_cannnibals, side):
    return (number_missionaries, number_cannnibals, side) == (3, 3, 1)
```

Hàm boolean **is_goal_state** trả về node có phải là node kết thúc không (0,0,0)

```
def is_goal_state(number_missionaries, number_cannnibals, side):
    return (number_missionaries, number_cannnibals, side) == (0, 0, 0)
```

Hàm boolean **number_of_cannibals_exceeds** trả về có bên bờ nào số người ít hơn số quỷ không (người sẽ bị ăn thịt)

```

def number_of_cannibals_exceeds(number_missionaries, number_cannnibals):
    number_missionaries_right = 3 - number_missionaries
    number_cannnibals_right = 3 - number_cannnibals
    return (number_missionaries > 0 and number_cannnibals > number_missionaries) \
    or (number_missionaries_right > 0 and number_cannnibals_right > number_missionaries_right)
```

Tổng hàm tạo cây hoàn chỉnh

```
def generate():
    # Khoi tao bien dem i, hang doi q, so node node_num
    global i
    q = deque()
    node_num = 0
    # Them node goc vao hang doi q
    q.append((3, 3, 1, 0, node_num))
    # Dat node cha cua node goc la None
    Parent[(3, 3, 1, 0, node_num)] = None

    # Vong lap duyet qua cac trang thai cua hang doi q
    while q:

        #Lay node dau hang doi ra
        number_missionaries, number_cannnibals, side, depth_level, node_num = q.popleft()

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

        # Tao va ve duong di (canh) giua hai node
        # O vong lap dau tien thi tao node goc
        u, v = draw_edge(number_missionaries, number_cannnibals, side, depth_level, node_num)
        # Neu la node goc thi to mau xanh duong, chu trang
        if is_start_state(number_missionaries, number_cannnibals, side):
            v.set_style("filled")
            v.set_fillcolor("blue")
            v.set_fontcolor("white")
        # Neu la node ket thuc thi to mau xanh la
        elif is_goal_state(number_missionaries, number_cannnibals, side):
            v.set_style("filled")
            v.set_fillcolor("green")
            # Tao cay hoan chinh ne chuyen sang vong lap ke tiep
            continue
            # Return neu chi tim duong di toi node ket thuc (bfs,dfs)
            # return True

        # Neu so quy nhieu hon so nguoi thi to mau do, chuyen sang vong lap ke tiep
        elif number_of_cannibals_exceeds(number_missionaries, number_cannnibals):
            v.set_style("filled")
            v.set_fillcolor("red")
            continue
        # Truong hop con lai thi to mau cam
        else:
            v.set_style("filled")
            v.set_fillcolor("orange")
        # Neu do sau (depth) hien tai bang do sau toi da thi ket thuc (hoan thanh)
        if depth_level == max_depth:
            return True
        # Sua lai gia tri cua side de tinh toan buoc di chuyen (-1 neu side = 1 va nguoc lai)
        op = -1 if side == 1 else 1

        # Bien boolean kiem tra xem co the mo rong node hay khong
        can_be_expanded = False

        # i = node_num
        # Duyet tung cach di chuyen hop le trong options
        for x, y in options:
```

```
            # Tinh toan cac node ke
            # Vi du (3,3,1) = (3 + -1*1,3+-1*0,0) = (2,3,0)
            next_m, next_c, next_s = number_missionaries + op * x, number_cannnibals + op * y, int(not side)

            # Neu node cha la None (node goc) hoac node mo rong khac node cha
            # (Neu giong thi node mo rong bi trung voi node cha)
            if Parent[(number_missionaries, number_cannnibals, side, depth_level, node_num)] is None
            or(next_m, next_c, next_s) != Parent[(number_missionaries, number_cannnibals, side, depth_level,
            node_num)][:3]:
                # Kiem tra xem node mo rong co hop le khong
                if is_valid_move(next_m, next_c):
                    # Them node vao hang doi va Parent (tap node cha)
                    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_cannnibals, side, depth_level, node_num)
        # Neu khong mo rong duoc thi to node mau xam
        if not can_be_expanded:
            v.set_style("filled")
            v.set_fillcolor("gray")
    # Ket thuc thoat vong lap
    return False
```

Dòng lệnh dưới để chạy thuật toán bằng dòng lệnh trong command line

```
if __name__ == "__main__":
    # Neu tao cay hoan chinh thanh cong thi xuat ra anh bang ham write_image()
    if generate():
        write_image()
```

2. File solve.py

Import thư viện và đặt path cho Graphviz

```
import os
import emoji
import pydot
import random
from collections import deque

# Set it to bin folder of graphviz
os.environ["PATH"] += os.pathsep + 'C:\Program Files\Graphviz\bin'
```

Khởi tạo 3 ditc() lưu trữ đường đi

Parent: lưu node cha của các node (m,c,s).

Move: lưu cách di chuyển. (x là số người, y số quỷ, side là bờ)

node_list: lưu Node tạo bởi Pydot được sử dụng để tô màu đường đi tìm được.

```
Parent, Move, node_list = dict(), dict(), dict()
```

Khởi tạo lớp đối tượng Solution (các phương thức sẽ tình bày ở phía dưới).

```
class Solution():
    def __init__(self):

    def is_valid_move(self, number_missionaries, number_cannnibals):

    def is_goal_state(self, number_missionaries, number_cannnibals, side):

    def is_start_state(self, number_missionaries, number_cannnibals, side):

    def number_of_cannibals_exceeds(self, number_missionaries, number_cannnibals):

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

    def solve(self, solve_method="dfs"):

    def draw_legend(self):

    def draw(self, *, number_missionaries_left, number_cannnibals_left, number_missionaries_right,
     number_cannnibals_right):

    def show_solution(self):

    def draw_edge(self, number_missionaries, number_cannnibals, side, depth_level):
```

```
23
24      def bfs(self):
25
26      def dfs(self, number_missionaries, number_cannnibals, side, depth_level):
```

Phương thức khởi tạo

```
def __init__(self):
    # Start state (3M, 3C, Left)
    # Goal State (0M, 0C, Right)
    # Each state gives the number of missionaries and cannibals on the left side

    # Khoi tao trang thai ban dau va ket thuc
    self.start_state = (3, 3, 1)
    self.goal_state = (0, 0, 0)

    # So cach di chuyen co the thuc hien
    self.options = [(1, 0), (0, 1), (1, 1), (0, 2), (2, 0)]

    # Vi tri cua thuyen dau (trai hay phai)
    self.boat_side = ["right", "left"]

    # Do thi pydot dung de ve duong di
    self.graph = pydot.Dot(graph_type='graph', bgcolor="#fff3af",
                           label="fig: Missionaries and Cannibal State Space Tree",fontcolor="red", fontsize="24")
    # Dict luu node da duyet
    self.visited = {}
    # Boolean the hien bai toan duoc giai quyet hay chua (ban dau False)
    self.solved = False
```

Phương thức boolean **is_valid_move** kiểm tra bước đi hợp lệ

```
def is_valid_move(self, number_missionaries, number_cannnibals):
    """
    Checks if number constraints are satisfied
    """
    return (0 <= number_missionaries <= 3) and (0 <= number_cannnibals <= 3)
```

Phương thức boolean **is_goal_state** kiểm tra node có phải node kết thúc (goal) không

```
def is_goal_state(self, number_missionaries, number_cannnibals, side):
    return (number_missionaries, number_cannnibals, side) == self.goal_state
```

Phương thức boolean **is_start_state** kiểm tra node có phải node gốc (start) không

```
def is_start_state(self, number_missionaries, number_cannnibals, side):
    return (number_missionaries, number_cannnibals, side) == self.start_state
```

Phương thức boolean **number_of_cannibals_exceeds** kiểm tra xem số quỷ có nhiều hơn số người không

```
def number_of_cannibals_exceeds(self, number_missionaries, number_cannnibals):
    number_missionaries_right = 3 - number_missionaries
    number_cannnibals_right = 3 - number_cannnibals
    return (number_missionaries > 0 and number_cannnibals > number_missionaries) \
           or (number_missionaries_right > 0 and number_cannnibals_right > number_missionaries_right)
```

Phương thức **write_image** xuất hình ảnh giải pháp tìm được

```
def write_image(self, file_name="state_space.png"):
    try:
        self.graph.write_png(file_name)
    except Exception as e:
        print("Error while writing file", e)
    print(f"File {file_name} successfully written.")
```

Phương thức **solve** lựa chọn phương pháp tìm đường đi (BFS hay DFS). Mặc định là DFS

```
def solve(self, solve_method="dfs"):
    self.visited = dict()
    # Thiet lap cac thuoc tinh cho node goc la None
    Parent[self.start_state] = None
    Move[self.start_state] = None
    node_list[self.start_state] = None

    # Tra ve phuong phap lua chon, DFS hay BFS
    # *self.start_state = (m,c,s)
    return self.dfs(*self.start_state, 0) if solve_method == "dfs" else self.bfs()
```

Phướng thức **draw_legend** để vẽ chú thích (optional)

```
def draw_legend(self):
    """
        Utility method to draw legend on graph if  legend flag is ON
    """
    # Tao doi tuong kieu Cluster trong do thi de chua chu thich
    graphlegend = pydot.Cluster(graph_name="legend", label="Legend", fontsize="20", color="gold",
                                fontcolor="blue", style="filled", fillcolor="#f4f4f4")

    # Tao cac node voi mau sac tuong ung va them vao do thi

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

    # Node killed (nguoi bi quy an thit), mau do
    node2 = pydot.Node("2", style="filled", fillcolor="red", label="Killed Node", fontcolor="black", width="2",
     fixedsize="true")
    graphlegend.add_node(node2)

    # Node Solution (duong di tim duoc)
    node3 = pydot.Node("3", style="filled", fillcolor="yellow", label="Solution nodes", width="2", fixedsize="true")
    graphlegend.add_node(node3)

    # Node khong mo rong duoc, mau xam
    node4 = pydot.Node("4", style="filled", fillcolor="gray", label="Can't be expanded",  width="2", fixedsize="true
    ")
    graphlegend.add_node(node4)

    # Node ket thuc (goal), mau xanh la
    node5 = pydot.Node("5", style="filled", fillcolor="green", label="Goal node", width="2", fixedsize="true")
    graphlegend.add_node(node5)
    # Node co node con, mau gold
    node7 = pydot.Node("7", style="filled", fillcolor="gold", label="Node with child", width="2", fixedsize="true")
    graphlegend.add_node(node7)

    # Chu thich
    description = "Each node (m, c, s) represents a \nstate where 'm' is the number of\n missionaries, 'n' the
     cannibals and \n's' 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 \noperators = [(0, 1), (0, 2), (1, 0), (1, 1), (2, 0),]\n"\
           "each tuples (x, y) inside operators \nrepresents the number of missionaries and \ncannibals to be moved
      from left to right \nif c == 1 and viceversa"
    # Khung ghi chu thich, mau gold
    node6 = pydot.Node("6", style="filled", fillcolor="gold", label= description, shape="plaintext", fontsize="20",
     fontcolor="red")
    graphlegend.add_node(node6)

    # Them graph chu thich vao do thi chinh
    self.graph.add_subgraph(graphlegend)

    # Them canh an (vo hinh) giua cac node voi nhau
    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"))
```

Phương thức **draw** sử dụng thư viện emoji để vẽ các di chuyển của bài toán lên console

```
def draw(self, *, number_missionaries_left, number_cannnibals_left, number_missionaries_right,
     number_cannnibals_right):
        """
            Draw state on console using emojis
        """
        # Chuyen so nguoi va quy thanh emoji (so luong * emoji)
        left_m = emoji.emojize(f":old_man: " * number_missionaries_left)
        left_c = emoji.emojize(f":ogre: " * number_cannnibals_left)
        right_m = emoji.emojize(f":old_man: " * number_missionaries_right)
        right_c = emoji.emojize(f":ogre: " * number_cannnibals_right)

        # In ra man hinh, can chinh cho deu nhau.
        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("")
```

Phương thức **show_solution** hiển thị giải pháp cho bài toán

```
def show_solution(self):
    # Tu node ket thuc (node goal) lan nguoc ve node goc (node start)
    # Node goal
```

```
    state = self.goal_state
    # Cac list luu duong di va node
    path, steps, nodes = [] ,[], []
    # Lap cho den khi state la None (node goc)
    while state is not None:
        # Them node hien tai vao paths, steps, nodes
        path.append(state)
        steps.append(Move[state])
        nodes.append(node_list[state])

        # Luu node cha cho node
        state = Parent[state]

    #Dao nguoc list steps, nodes (duong di can hien thi)
    steps, nodes = steps[::-1], nodes[::-1]

    # Trang thai ban dau va ket thuc
    number_missionaries_left, number_cannnibals_left = 3, 3
    number_missionaries_right, number_cannnibals_right = 0, 0

    # In trang thai dau (node goc)
    print("*" * 60)
    self.draw(number_missionaries_left=number_missionaries_left, number_cannnibals_left=number_cannnibals_left,
              number_missionaries_right=number_missionaries_right, number_cannnibals_right=number_cannnibals_right)

    # Lap lan luot cac buoc trong steps va nodes
    for i, ((number_missionaries, number_cannnibals, side), node) in enumerate(zip(steps[1:], nodes[1:])):

        # Neu node khong phai node goc thi to mau vang
        if node.get_label() != str(self.start_state):
            node.set_style("filled")
            node.set_fillcolor("yellow")
        # In ra cach di chuyen
        print(f"Step {i + 1}: Move {number_missionaries} missionaries and {number_cannnibals} cannibals from {self.
     boat_side[side]} to {self.boat_side[int(not side)]}.")

        # Xac dinh huong di chuyen cua thuyen
        op = -1 if side == 1 else 1

        # Cap nhat so luong nguoi va quy
        number_missionaries_left = number_missionaries_left + op * number_missionaries
        number_cannnibals_left = number_cannnibals_left + op * number_cannnibals

        number_missionaries_right = number_missionaries_right - op * number_missionaries
        number_cannnibals_right = number_cannnibals_right - op * number_cannnibals

        # Ve cac trang thai sau moi lan di chuyen (so quy, nguoi ben trai va phai)
        self.draw(number_missionaries_left=number_missionaries_left, number_cannnibals_left=number_cannnibals_left,
                  number_missionaries_right=number_missionaries_right, number_cannnibals_right=
     number_cannnibals_right)

    # In ra man hinh thuc hien thanh cong
    print("Congratulations!!! you have solved the problem")
    print("*" * 60)
```

Phương thức **draw_edge** dùng để vẽ cạnh giữa hai node (tương tự trong file generate-full-space-tree.py)

```
def draw_edge(self, number_missionaries, number_cannnibals, side, depth_level):
    u, v = None, None
    # Kiem tra neu node cha khong la None (khong la node goc)
    if Parent[(number_missionaries, number_cannnibals, side)] is not None:
        # Tao node u va them vao do thi
        # depth - 1 vi la node truoc
        u = pydot.Node(str(Parent[(number_missionaries, number_cannnibals, side)] + (depth_level - 1, )),
                       label=str(Parent[((number_missionaries, number_cannnibals, side))]))
        self.graph.add_node(u)
        # Tuong tu voi v, v la node sau
        v = pydot.Node(str((number_missionaries, number_cannnibals, side, depth_level)),
                       label=str((number_missionaries, number_cannnibals, side)))
        self.graph.add_node(v)
        # Ve canh noi hai node va them vao do thi
        edge = pydot.Edge(str(Parent[(number_missionaries, number_cannnibals, side)] + (depth_level - 1, )),
                          str((number_missionaries, number_cannnibals, side, depth_level) ), dir='forward')
        self.graph.add_edge(edge)
    # Voi node cha la Nonde (TH node goc)
    else:
        # For start node
        # Tao node v (node goc) va them vao do thi
        v = pydot.Node(str((number_missionaries, number_cannnibals, side, depth_level)),
                       label=str((number_missionaries, number_cannnibals, side)))
        self.graph.add_node(v)
    # Tra ve node u,v
```

```
    return u, v
```

Thuật toán **bfs**

```
def bfs(self):
    # Khoi tao hang doi q
    q = deque()
    # Them node goc vao hang doi va danh dau da tham trong tap visited
    q.append(self.start_state + (0, ))
    self.visited[self.start_state] = True

    # Lap cac phan tu trong hang doi
    while q:
        Lay ra phan tu dau hang doi
        number_missionaries, number_cannnibals, side, depth_level = q.popleft()
        # Draw Edge from u -> v
        # Where u = Parent[v]
        # and v = (number_missionaries, number_cannnibals, side, depth_level)

        Ve canh tu u -> v
        u, v = self.draw_edge(number_missionaries, number_cannnibals, side, depth_level)

        # Neu node la node goc thi to mau xanh duong
        if self.is_start_state(number_missionaries, number_cannnibals, side):
            v.set_style("filled")
            v.set_fillcolor("blue")
            v.set_fontcolor("white")
        # Neu la node ket thuc (node goal) to mau xanh la
        elif self.is_goal_state(number_missionaries, number_cannnibals, side):
            v.set_style("filled")
            v.set_fillcolor("green")
            return True
        # Neu la node killed (nguoi bi an thit) to mau do
        elif self.number_of_cannibals_exceeds(number_missionaries, number_cannnibals):
            v.set_style("filled")
            v.set_fillcolor("red")
            continue
        # Neu khong phai cac TH tren thi to mau cam
        else:
            v.set_style("filled")
            v.set_fillcolor("orange")
        # Xac dinh huong di de tinh toan buoc di chuyen
        op = -1 if side == 1 else 1

        can_be_expanded = False

        # Voi moi cach di chuyen trong options
        for x, y in self.options:
            # Tinh buoc di chuyen ke tiep
            next_m, next_c, next_s = number_missionaries + op * x, number_cannnibals + op * y, int(not side)
            # Kiem tra xem node da duoc tham chua
            if (next_m, next_c, next_s) not in self.visited:
                # Kiem tra xem buoc di chuyen co hop le hay khong
                if self.is_valid_move(next_m, next_c):
                    can_be_expanded = True
                    # Danh dau da tham, them vao hang doi
                    self.visited[(next_m, next_c, next_s)] = True
                    q.append((next_m, next_c, next_s, depth_level + 1))

                    # Them node cha cho node vua duoc tham
                    Parent[(next_m, next_c, next_s)] = (number_missionaries, number_cannnibals, side)
                    # Luu lai duong di vao Move va node_list
                    Move[(next_m, next_c, next_s)] = (x, y, side)
                    node_list[(next_m, next_c, next_s)] = v
        # Neu khong the mo rong, to mau xam
        if not can_be_expanded:
            v.set_style("filled")
            v.set_fillcolor("gray")
        # Ket thuc lap, ket thuc chuong trinh
        return False
```

Thuật toán **dfs**, sử dụng đệ quy

```
def dfs(self, number_missionaries, number_cannnibals, side, depth_level):
    # Them node vao Visited su dung de quy
    self.visited[(number_missionaries, number_cannnibals, side)] = True

    # Draw Edge from u -> v
    # Where u = Parent[v]

    # Ve canh tu node u -> node v
    u, v = self.draw_edge(number_missionaries, number_cannnibals, side, depth_level)
```

```
    # Neu node la node goc thi to mau xanh duong
    if self.is_start_state(number_missionaries, number_cannnibals, side):
        v.set_style("filled")
        v.set_fillcolor("blue")
    # Neu la node ket thuc (node goal) to mau xanh la
    elif self.is_goal_state(number_missionaries, number_cannnibals, side):
        v.set_style("filled")
        v.set_fillcolor("green")
        return True
    # Neu la node killed (nguoi bi an thit) to mau do
    elif self.number_of_cannibals_exceeds(number_missionaries, number_cannnibals):
        v.set_style("filled")
        v.set_fillcolor("red")
        return False
    # Neu khong phai cac TH tren thi to mau cam
    else:
        v.set_style("filled")
        v.set_fillcolor("orange")
    # Bien boolean the hien thuat toan duoc giai chua
    # Mac dinh False
    solution_found = False
    # Xac dinh huong di
    operation = -1 if side == 1 else 1

    can_be_expanded = False

    # Voi moi cach di chuyen trong options
    for x, y in self.options:
        # Tinh toan cach di chuyen ke tiep
        next_m, next_c, next_s = number_missionaries + operation * x, number_cannnibals + operation * y, int(not
     side)

        # Neu node chua duoc tham
        if (next_m, next_c, next_s) not in self.visited:
            # Kiem tra xem node co hop le khong
            if self.is_valid_move(next_m, next_c):
                can_be_expanded = True
                # Luu node cha cho node vua tham
                Parent[(next_m, next_c, next_s)] = (number_missionaries, number_cannnibals, side)
                # Luu lai duong di vao Move va node_list
                Move[(next_m, next_c, next_s)] = (x, y, side)
                node_list[(next_m, next_c, next_s)] = v

                # De quy lai ham de tim duong di
                # Neu tim duoc tra ve True va ket thuc de quy
                # Neu khong tiep tuc de quy
                solution_found = (solution_found or self.dfs(next_m, next_c, next_s, depth_level + 1))

                if solution_found:
                    return True
    # Neu node khong mo rong duoc, to mau xam
    if not can_be_expanded:
        v.set_style("filled")
        v.set_fillcolor("gray")

    # Luu lai co giai phap hay khong
    self.solved = solution_found
    return solution_found
```

3. File main.py

```
# Import thu vien
from solve import Solution
import argparse
import itertools
# Thiet lap doi so
arg = argparse.ArgumentParser()
# -m la method bfs hay dfs
arg.add_argument("-m", "--method", required=False, help="Specify which method to use")
# -l la co hien thi chu thich hay khong
arg.add_argument("-l", "--legend", required=False, help="Specify if you want to display legend on graph")

args = vars(arg.parse_args())

solve_method = solve_args.get("method") if args.get("method") is not None else "bfs"
legend_flag = args.get("legend") if args.get("legend") is not None else False



def main():
```

```
    #Khoi tao doi tuong Solution
    s = Solution()

    Kiem tra xem thuat toan co tim ra giai phap khong
    if(s.solve(solve_method)):
        # Neu tim ra
        # Hien thi giai phap len console
        s.show_solution()
        # Ten file dau ra
        output_file_name = f"{solve_method}"
        # Ve chu thich neu co yeu cau
        # Draw legend if legend_flag is set
        if legend_flag:
            if legend_flag[0].upper() == 'T' :
                output_file_name += "_legend.png"
                s.draw_legend()
            else:
                output_file_name += ".png"
        else:
             output_file_name += ".png"

        # Xuat hinh anh voi ten dat o truoc
        s.write_image(output_file_name)
    else:
        #Neu khong tim thay giai phap thong bao
        raise Exception("No solution found")

# Kiem tra de chay chuong trinh bang command line
if __name__ == "__main__":
    main()
```