Mã sinh viên: N22DCCN142

Tên sinh viên: VŨ ĐÌNH KHOA

Implement BFS, UCS with some following requirements:

- State space as graph

- Graph is represented as adjacency list.

Run BFS and UCS to find the shortest path between Arab and Bucharest: track expanded nodes, frontier and explored list.

from queue import Queue, PriorityQueue

def bfs(graph, start, goal):

"""

Thực hiện thuật toán BFS để tìm đường đi ngắn nhất từ start đến goal.

"""

frontier = Queue()

frontier.put(start)

explored = set()

parent = {start: None}

expanded\_nodes = []

while not frontier.empty():

current = frontier.get()

expanded\_nodes.append(current)

if current == goal:

# Tìm thấy đích, tái tạo đường đi

path = []

while current:

path.append(current)

current = parent[current]

path.reverse()

return path, expanded\_nodes, list(frontier.queue), list(explored)

explored.add(current)

for neighbor in graph[current]:

if neighbor not in explored and neighbor not in frontier.queue:

frontier.put(neighbor)

parent[neighbor] = current

return None, expanded\_nodes, list(frontier.queue), list(explored) # Không tìm thấy đường đi

def ucs(graph, start, goal):

"""

Thực hiện thuật toán UCS để tìm đường đi ngắn nhất từ start đến goal.

"""

frontier = PriorityQueue()

frontier.put((0, start)) # (cost, node)

explored = set()

parent = {start: None}

cost\_so\_far = {start: 0}

expanded\_nodes = []

while not frontier.empty():

current\_cost, current\_node = frontier.get()

expanded\_nodes.append(current\_node)

if current\_node == goal:

# Tìm thấy đích, tái tạo đường đi

path = []

while current\_node:

path.append(current\_node)

current\_node = parent[current\_node]

path.reverse()

return path, expanded\_nodes, [node for \_, node in frontier.queue], list(explored)

explored.add(current\_node)

for neighbor, weight in graph[current\_node]:

new\_cost = current\_cost + weight

if neighbor not in cost\_so\_far or new\_cost < cost\_so\_far[neighbor]:

cost\_so\_far[neighbor] = new\_cost

frontier.put((new\_cost, neighbor))

parent[neighbor] = current\_node

return None, expanded\_nodes, [node for \_, node in frontier.queue], list(explored) # Không tìm thấy đường đi

# Đồ thị dưới dạng danh sách kề (adjacency list)

graph = {

"Arab": [("Sibiu", 140), ("Timisoara", 118), ("Zerind", 75)],

"Sibiu": [("Fagaras", 99), ("Rimnicu Vilcea", 80), ("Arab", 140)],

"Timisoara": [("Lugoj", 111), ("Arab", 118)],

"Zerind": [("Oradea", 71), ("Arab", 75)],

"Fagaras": [("Bucharest", 211), ("Sibiu", 99)],

"Rimnicu Vilcea": [("Pitesti", 97), ("Craiova", 146), ("Sibiu", 80)],

"Lugoj": [("Mehadia", 70), ("Timisoara", 111)],

"Oradea": [("Sibiu", 151), ("Zerind", 71)],

"Pitesti": [("Bucharest", 101), ("Rimnicu Vilcea", 97)],

"Craiova": [("Pitesti", 138), ("Rimnicu Vilcea", 146)],

"Mehadia": [("Drobeta", 75), ("Lugoj", 70)],

"Drobeta": [("Craiova", 120), ("Mehadia", 75)],

"Bucharest": [("Fagaras", 211), ("Pitesti", 101)]

}

# Chạy thuật toán BFS và UCS

path\_bfs, expanded\_bfs, frontier\_bfs, explored\_bfs = bfs(graph, "Arab", "Bucharest")

path\_ucs, expanded\_ucs, frontier\_ucs, explored\_ucs = ucs(graph, "Arab", "Bucharest")

# In kết quả

print("BFS:")

print("Đường đi ngắn nhất:", path\_bfs)

print("Các node đã mở rộng:", expanded\_bfs)

print("Danh sách frontier:", frontier\_bfs)

print("Danh sách explored:", explored\_bfs)

print("\nUCS:")

print("Đường đi ngắn nhất:", path\_ucs)

print("Các node đã mở rộng:", expanded\_ucs)

print("Danh sách frontier:", frontier\_ucs)

print("Danh sách explored:", explored\_ucs)