# **EX 3.3 Deep-limited search**

Implement DLS algorithm and run it to find the shortest path between Arab and Bucharest (track node, limit variables).

1. Source code:

# Đánh số các đỉnh bằng số tự nhiên từ 0 đến n - 1.

name = ["Oradea", "Zerind", "Arad", "Timisoara", "Sibiu",

"Lugoj", "Rimnicu Vilcea", "Fagaras", "Pitesti", "Mehadia",

"Dobreta", "Craiova", "Bucharest", "Giurgiu", "Urziceni",

"Hirsova", "Eforie", "Vaslui", "Lasi", "Neamt"]

def load\_data():

n = 20

a = [[[-1, -1] for i in range(4)] for j in range(n)]

# Tạo danh sách kè của các đỉnh.

a[0][0] = [1, 71]

a[0][1] = [4, 151]

a[1][0] = [0, 71]

a[1][1] = [2, 75]

a[2][0] = [1, 75]

a[2][1] = [3, 118]

a[2][2] = [4, 140]

a[3][0] = [2, 118]

a[3][1] = [5, 111]

a[4][0] = [0, 151]

a[4][1] = [2, 140]

a[4][2] = [7, 99]

a[4][3] = [6, 80]

a[5][0] = [3, 111]

a[5][1] = [9, 70]

a[6][0] = [4, 80]

a[6][1] = [8, 97]

a[6][2] = [11, 146]

a[7][0] = [4, 99]

a[7][1] = [12, 211]

a[8][0] = [6, 97]

a[8][1] = [11, 138]

a[8][2] = [12, 101]

a[9][0] = [5, 70]

a[9][1] = [10, 75]

a[10][0] = [9, 75]

a[10][1] = [11, 120]

a[11][0] = [6, 146]

a[11][1] = [8, 138]

a[11][2] = [10, 120]

a[12][0] = [7, 211]

a[12][1] = [8, 101]

a[12][2] = [13, 90]

a[12][3] = [14, 85]

a[13][0] = [12, 90]

a[14][0] = [12, 85]

a[14][1] = [15, 98]

a[14][2] = [17, 142]

a[15][0] = [14, 98]

a[15][1] = [16, 86]

a[16][0] = [15, 86]

a[17][0] = [14, 142]

a[17][1] = [18, 92]

a[18][0] = [17, 92]

a[18][1] = [19, 87]

a[19][0] = [18, 87]

start = 2

goal = 12

return n, a, start, goal

def depth\_limited\_search(a, current, goal, cost, min\_cost, depth, path, min\_path, visited\_states):

# Nếu độ sâu này không thể --> Thoát.

if depth < 0:

return False

# Nếu đã duyệt qua và có đường đi tốt hơn --> Thoát.

if current in visited\_states:

if visited\_states[current] < cost:

return False

# Gán cost vào visited\_states.

visited\_states[current] = cost

# Thêm điểm hiện tại vào path.

path.append(current)

# Nếu đã đến điểm cuối --> Kiểm tra xem có đường đi tốt hơn không.

if current == goal:

if cost < min\_cost[0]:

# Nếu tốt hơn, cập nhật lại giá trị.

min\_cost[0] = cost

min\_path[:] = path[:]

path.pop() # Backtrack trước khi return True để tìm đương tối ưu hơn.

# Nếu tìm thấy đường đi tốt hơn --> Thoát.

return True

found = False # Giá trị sẽ return.

# Duyệt qua các điểm kè của điểm hiện tại.

for neighbor, \_ in a[current]:

# Nếu tồn tại đường đi và không phải là điểm đã đi qua.

if neighbor != -1 and neighbor not in path:

# Thực hiện để tìm đường đi tốt nhất.

if depth\_limited\_search(a, neighbor, goal, cost + 1, min\_cost, depth - 1, path, min\_path, visited\_states):

# Nếu tìm thấy đường đi tốt hơn --> Thoát.

found = True

# Backtrack.

path.pop()

# Return giá trị tìm được.

return found

def Iterative\_Deepening\_DepthFirstSearch(n, a, start, goal):

limit = 0

visited\_states = {}

while (limit < n):

min\_path = []

min\_cost = [float('inf')]

if depth\_limited\_search(a, start, goal, 1, min\_cost, limit, [], min\_path, visited\_states):

return min\_path, min\_cost[0]

limit += 1

return [], float('inf')

def main():

n, a, start, goal = load\_data()

min\_path, min\_cost = Iterative\_Deepening\_DepthFirstSearch(n, a, start, goal)

print("Cost: ", min\_cost, "(Số đỉnh đi qua)")

print("Path: ", end = "")

for value in min\_path:

print(name[value], end = " ")

if \_\_name\_\_ == "\_\_main\_\_":

main()

1. DLS:

🔹 Lần chạy limit = 0, start = 2(Arad), goal = 12(Bucharest)

→ Cost: Số đỉnh đi qua.

| current | depth | cost | min\_cost | path min\_path | visited\_states[current] |
| --- | --- | --- | --- | --- | --- |
| Arad | 0 | 1 | inf | path: Arad  min\_path: [] | 1 |

🔹 Lần chạy limit = 1, start = 2(Arad), goal = 12(Bucharest)

→ Cost: Số đỉnh đi qua.

| current | depth | cost | min\_cost | path min\_path | visited\_states[current] |
| --- | --- | --- | --- | --- | --- |
| Arad | 1 | 1 | inf | path: Arad  min\_path: [] | 1 |
| Zerind | 0 | 2 | inf | path: Arad, Zerind  min\_path: [] | 2 |
| Timisoara | 0 | 2 | inf | path: Arad, Timisoara  min\_path: [] | 2 |
| Sibiu | 0 | 2 | inf | path: Arad, Sibiu  min\_path: [] | 2 |

🔹 Lần chạy limit = 2, start = 2(Arad), goal = 12(Bucharest)

→ Cost: Số đỉnh đi qua.

| current | depth | cost | min\_cost | path min\_path | visited\_states[current] |
| --- | --- | --- | --- | --- | --- |
| Arad | 2 | 1 | inf | path: Arad  min\_path: [] | 1 |
| Zerind | 1 | 2 | inf | path: Arad, Zerind  min\_path: [] | 2 |
| Oradea | 0 | 3 | inf | path: Arad, Zerind, Oradea  min\_path: [] | 3 |
| Timisoara | 1 | 2 | inf | path: Arad, Timisoara  min\_path: [] | 2 |
| Lugoj | 0 | 3 | inf | path: Arad, Timisoara, Lugoj  min\_path: [] | 3 |
| Sibiu | 1 | 2 | inf | path: Arad, Sibiu,  min\_path: [] | 2 |
| Oradea | 0 | 3 | inf | path: Arad, Sibiu, Oradea  min\_path: [] | 3 |
| Fagaras | 0 | 3 | inf | path: Arad, Sibiu, Fagaras  min\_path: [] | 3 |
| Rimnicu Vilcea | 0 | 3 | inf | path: Arad, Sibiu, Rimnicu Vilcea  min\_path: [] | 3 |

🔹 Lần chạy limit = 3, start = 2(Arad), goal = 12(Bucharest)

→ Cost: Số đỉnh đi qua.

| current | depth | cost | min\_cost | path min\_path | visited\_states[current] |
| --- | --- | --- | --- | --- | --- |
| Arad | 3 | 1 | inf | path: Arad  min\_path: [] | 1 |
| Zerind | 2 | 2 | inf | path: Arad, Zerind  min\_path: [] | 2 |
| Oradea | 1 | 3 | inf | path: Arad, Zerind, Oradea  min\_path: [] | 3 |
| Sibiu | 0 | 4 | inf | path: Arad, Zerind, Oradea (Đã dừng)  min\_path: [] | 2 < 4 → Dừng. |
| Timisoara | 2 | 2 | inf | path: Arad, Timisoara  min\_path: [] | 2 |
| Lugoj | 1 | 3 | inf | path: Arad, Timisoara, Lugoj  min\_path: [] | 3 |
| Mehadia | 0 | 4 | inf | path: Arad, Timisoara, Lugoj, Mehadia  min\_path: [] | 4 |
| Sibiu | 2 | 2 | inf | path: Arad, Sibiu,  min\_path: [] | 2 |
| Oradea | 1 | 3 | inf | path: Arad, Sibiu, Oradea  min\_path: [] | 3 |
| Zerind | 0 | 4 | inf | path: Arad, Sibiu, Oradea (Đã dừng)  min\_path: [] | 2 < 4 → Dừng |
| Fagaras | 1 | 3 | inf | path: Arad, Sibiu, Fagaras  min\_path: [] | 3 |
| Bucharest | 0 | 4 | 4 | path: Arad, Sibiu, Fagaras, Bucharest  min\_path: Arad, Sibiu, Fagaras, Bucharest | 4 |
| Rimnicu Vilcea | 1 | 3 | 4 | path: Arad, Sibiu, Rimnicu Vilcea  min\_path: Arad, Sibiu, Fagaras, Bucharest | 3 |
| Pitesti | 0 | 4 | 4 | path: Arad, Sibiu, Rimnicu Vilcea, Pitesti  min\_path: Arad, Sibiu, Fagaras, Bucharest | 4 |
| Craiova | 0 | 4 | 4 | path: Arad, Sibiu, Rimnicu Vilcea, Craiova  min\_path: Arad, Sibiu, Fagaras, Bucharest | 4 |

⇒ Cost = 4 (Số đỉnh đi qua)

Path: Arad → Sibiu → Fagaras → Bucharest