Code:

import heapq

# 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 a, n, start, goal

def a\_star(adj\_matrix, num\_cities, heuristics, start, goal):

frontier = []

heapq.heappush(frontier, (0, start))

came\_from = {start: None}

cost\_so\_far = {start: 0}

while frontier:

# print(frontier)

\_, current = heapq.heappop(frontier)

if current == goal:

break

for neighbor, cost in adj\_matrix[current]:

if (neighbor != -1):

# print(f"Updating {name[neighbor]}: came from {name[current]}", end = ' ')

new\_cost = cost\_so\_far[current] + cost

# print(new\_cost, end = ' ')

# if (neighbor in cost\_so\_far):

# print(cost\_so\_far[neighbor], end = ' ')

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

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

priority = new\_cost + heuristics[name[neighbor]]

# print(priority)

heapq.heappush(frontier, (priority, neighbor))

came\_from[neighbor] = current

# else:

# print()

return came\_from, cost\_so\_far

def create\_heuristics():

return {

'Oradea': 380, 'Zerind': 374, 'Arad': 366, 'Timisoara': 329,

'Sibiu': 253, 'Lugoj': 244, 'Rimnicu Vilcea': 193, 'Fagaras': 176,

'Pitesti': 100, 'Mehadia': 241, 'Drobeta': 242, 'Craiova': 160,

'Bucharest': 0, 'Giurgiu': 77, 'Urziceni': 80, 'Hirsova': 151,

'Eforie': 161, 'Vaslui': 199, 'Iasi': 226, 'Neamt': 234

}

def main():

adj\_matrix, num\_cities, start, goal = load\_data()

# for i in range(num\_cities):

# print(name[i], ": ", end = ' ')

# for j in adj\_matrix[i]:

# if j[0] != -1:

# print( j, end = ' ')

# print()

heuristics = create\_heuristics()

came\_from, cost\_so\_far = a\_star(adj\_matrix, num\_cities, heuristics, start, goal)

print("start: ", name[start])

print("goal : ", name[goal])

print("Route: ", end = ' ')

print(name[start], end ='')

start = goal

route = []

while came\_from[goal] is not None:

route.append(goal)

goal = came\_from[goal]

for value in route[::-1]:

print(' -> ', name[value], end = ' ')

print()

print("Cost : ", cost\_so\_far[start])

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

main()

# 526

Chạy Code:

Priority Queue (Binary Heap): [(0, 2)]

Choose min: 0 2

Updating Zerind: came from Arad 75 449

Updating Timisoara: came from Arad 118 447

Updating Sibiu: came from Arad 140 393

Priority Queue (Binary Heap): [(393, 4), (449, 1), (447, 3)]

Choose min: 393 4

Updating Oradea: came from Sibiu 291 671

Updating Arad: came from Sibiu 280 0

Updating Fagaras: came from Sibiu 239 415

Updating Rimnicu Vilcea: came from Sibiu 220 413

Priority Queue (Binary Heap): [(413, 6), (415, 7), (671, 0), (449, 1), (447, 3)]

Choose min: 413 6

Updating Sibiu: came from Rimnicu Vilcea 300 140

Updating Pitesti: came from Rimnicu Vilcea 317 417

Updating Craiova: came from Rimnicu Vilcea 366 526

Priority Queue (Binary Heap): [(415, 7), (417, 8), (526, 11), (449, 1), (447, 3), (671, 0)]

Choose min: 415 7

Updating Sibiu: came from Fagaras 338 140

Updating Bucharest: came from Fagaras 450 450

Priority Queue (Binary Heap): [(417, 8), (447, 3), (450, 12), (449, 1), (671, 0), (526, 11)]

Choose min: 417 8

Updating Rimnicu Vilcea: came from Pitesti 414 220

Updating Craiova: came from Pitesti 455 366

Updating Bucharest: came from Pitesti 418 450 418

Priority Queue (Binary Heap): [(418, 12), (449, 1), (447, 3), (526, 11), (671, 0), (450, 12)]

Choose min: 418 12 (goal)

start: Arad

goal : Bucharest

Route: Arad -> Sibiu -> Rimnicu Vilcea -> Pitesti -> Bucharest

Cost : 418