Tên : Nguyễn Quý Văn

Lớp s22-61TH1

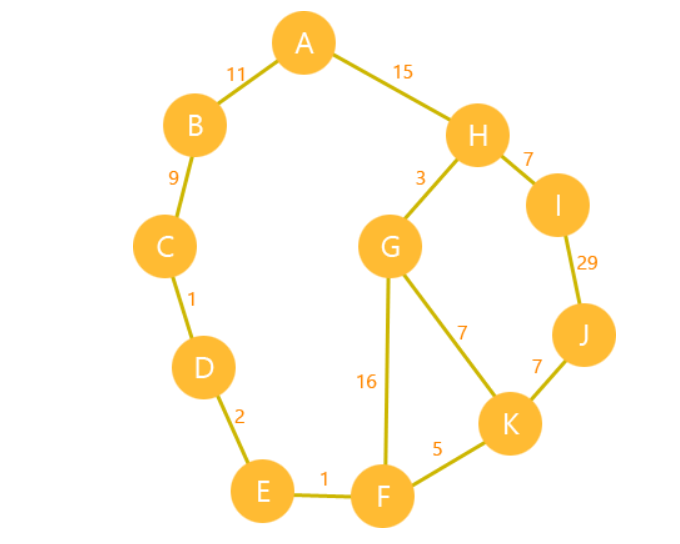
Ngày: 30/03/2022

Số thứ tự 28

Mssv : 1951065673

Môn Trí Tuệ Nhân tạo

Map



Heuristic

A=60 / B=53 / C=36 / D=35 / E=35 / F=19 / G=16 / H=38 / I=23 / J=0 / K=7

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Lần lặp | Đỉnh | Đỉnh kề | Open | Close |
| 0 |  |  | A60 |  |
| 1 | A | B, H | B64, H53 | A |
| 2 | H | G, I, A | B64, G34, I45 | A, H |
| 3 | G | H, K, F | B64, I45, K32, F53 | A, H, G |
| 4 | K | G, F, J | B64, J32, F49, I45 | A, H, G |
| 5 | K (dừng) |  |  |  |

**Code Astar**

import queue #import thư viện

from mymap import myMap, heuristic #import file mymap

from modules import showResultWithAttr, showStep, aweSomeSort #import file modules

def AStar(start, goal):

q = queue.deque() # khởi tạo thuật toán

q.append(start) # enqueue

previous = [] # khởi tạo previous kiểu dictionary

for city in myMap.keys():

previous.append((city, {'from': None, 'total\_cost': heuristic[city]}))

previous = dict(previous)

counter = 0

while 1:

counter += 1

showStep(counter, q, previous) # Show các bước thực hiện thuật toán

curCity = q.popleft() # enqueue (Đây cũng là phần tử nhỏ nhất vì đã sắp xếp queue tăng dần)

if curCity == goal: #nếu curCity tới điểm goal

print('Success!') #xuất ra hoàn thành

showResultWithAttr(previous, start, goal)

return True # Tìm thấy goal

curCityTotalCost = previous[curCity]['total\_cost'] - heuristic[curCity] # Chi phí để đi từ start đến curCity

if curCity != goal:

for city in myMap[curCity].keys(): # Các thành phố có thể đi đến được từ curCity

cityTotalCost = previous[city]['total\_cost'] # Chi phí được tính ở bước trước đó

totalCost = myMap[curCity][city]['cost'] + curCityTotalCost + heuristic[city] # Chi phí để đi từ start -> curCity -> city

# Nếu chưa đi qua thành phố này hoặc chi phi đi từ start -> curCity -> city tốt hơn chi phí trước đó

if previous[city]['from'] == None or totalCost < cityTotalCost :

if q.count(city) != 0: # Nếu đã có city này trong hàng đợi thì xóa nó đi

q.remove(city)

q.append(city) # Thêm vào hàng đợi

previous[city]['from'] = curCity # Lưu lại dấu vết

previous[city]['total\_cost'] = totalCost # Cập nhật lại chi phí mới

if len(q) == 0:

print('Fail!')#xuất ra thất bại

return False # Không thể đi đến goal từ start

q = queue.deque(aweSomeSort(q, previous)) # Sắp xếp lại hàng đợi tăng dần theo total\_cost

start = "A"

goal = "K"

AStar(start, goal)

**File mymap.py**

myMap = {

"A": {

"B": { "cost": 11 },

"H": { "cost": 15 }

},

"B": {

"A": { "cost": 11 },

"C": { "cost": 9 }

},

"C": {

"B": { "cost": 9 },

"D": { "cost": 1 }

},

"D": {

"C": { "cost": 1 },

"E": { "cost": 2 }

},

"E": {

"D": { "cost": 2 },

"F": { "cost": 1 }

},

"F": {

"E": { "cost": 1 },

"K": { "cost": 5 },

"G": { "cost": 16 }

},

"G": {

"F": { "cost": 16 },

"K": { "cost": 7 },

"H": { "cost": 3 }

},

"K": {

"G": { "cost": 7 },

"F": { "cost": 5 },

"J": { "cost": 7 }

},

"H": {

"A": { "cost": 15 },

"G": { "cost": 3 },

"I": { "cost": 7 }

},

"I": {

"H": { "cost": 7 },

"J": { "cost": 29 }

},

"J": {

"I": { "cost": 29 },

"K": { "cost": 7 }

},

}

heuristic = {

"A": 60,

"B": 53,

"C": 36,

"D": 35,

"E": 35,

"F": 19,

"G": 16,

"H": 38,

"I": 23,

"J": 0,

"K": 7

}

File module

def showResult(previous, start, goal):

curCity = goal # Truy vết từ điểm đích về điểm bắt đầu

print('Result:', curCity, end=' ')

while curCity != start:

curCity = previous[curCity]

print('->', curCity, end=' ')

def showResultWithAttr(previous, start, goal, attr='total\_cost'):

curCity = goal # Truy vết từ điểm đích về điểm bắt đầu

print('Result:', (curCity, previous[curCity][attr]), end=' ')

while curCity != start:

curCity = previous[curCity]['from']

print('->', (curCity, previous[curCity][attr]), end=' ')

def showStep(counter, q, previous, attr='total\_cost'):

print('%d. {' % counter, end=' ')

i = 0

lenQ = len(q)

for v in q:

i += 1

if (i < lenQ):

print((v, previous[v][attr], previous[v]['from']), end=', ')

else:

print((v, previous[v][attr], previous[v]['from']), end=' ')

print('}')

def aweSomeSort(array, previous, sortBy='total\_cost'): # QuickSort (python version)

less = []

equal = []

greater = []

if len(array) > 1:

pivot = previous[array[0]][sortBy]

for city in array:

cost = previous[city][sortBy]

if cost < pivot:

less.append(city)

if cost == pivot:

equal.append(city)

if cost > pivot:

greater.append(city)

return aweSomeSort(less, previous, sortBy) + equal + aweSomeSort(greater, previous, sortBy) # toán tử nối mảng

else:

return array

result:

