4107056006 資工三 游庭瑋

A* 演算法報告

1. 目的：

本作業目的在於利啟發式演算法來達成尋找從起點到終點的最短路徑。有別於 Dijkstra 演算法，A* 並沒有先將所有路徑資訊進行考量，而是由現有資訊與推測資訊去找出最好的選擇，並查詢有沒有因為這個選擇造成曾經選過得路徑也有更好的選擇

2. 實做部份：

這次我選用 python 作為實做的語言。首先先定義每個座標點的類別，property 應有位置、F、G、H 數值及父節點，如下圖：

```python
class Node:
    POS = (0, 0)
    F = 999
    G = 999
    H = 999
    FROM = None
```

接下來定義 constructor 及重載各類操作符，方便之後對節點的操作

```python
def __init__(self, x, y, from_=None):
    self.POS = (x, y)
    self.FROM = from_

def __eq__(self, other):
    return self.POS == other.POS

def __ne__(self, other):
    return self.POS[0] != other.POS[0] or self.POS[1] != other.POS[1]

def __gt__(self, other):
    return self.F > other.F

def __ge__(self, other):
    return self.F >= other.F

def __lt__(self, other):
    return self.F < other.F

def __le__(self, other):
    return self.F <= other.F
```

再來是演算法本身：

```
44    def a_star(start_pos, end_pos, map_: list):
45        open_list = [start_pos]
46        close_list = []
47        """start search"""
48        start_pos.G = 0
49        start_pos.H = _distance(start_pos, end_pos)
50        start_pos.F = start_pos.H
```

我們首先將起始點放入 open_list，代表其尚未被搜尋，並建造一個空的 close_list，並將起始點的 G 設為 0，表示與起始點距離為 0；H 設為起點到終點的歐拉距離；F 即為兩者之和。

```
    while len(open_list) != 0:
        x = lowest_f_in(open_list)
        if x == end_pos:
            print('Found path!')
            return construct_route(end_pos)
        open_list.remove(x)
        close_list.append(x)
        for y in neighbor_of(x, map_):
            if y in close_list:
                continue
            guess_g = x.G + _real_distance(x, y)
            if y not in open_list:
                guess_better = True
            elif guess_g < y.G:
                guess_better = True
            else:
                guess_better = False
            if guess_better:
                # print(f'from {x.POS} to {y.POS}')
                y.FROM = x
                y.G = guess_g
                y.H = _distance(y, end_pos)
                y.F = y.G + y.H
                open_list.append(y)
            if y == end_pos:
                end_pos = y
    raise PathNotFoundError()
```

a. 接下來要先找到 open_list 裡 F 最小的節點 x，檢查看是不是終點。將其移出 open_list，放入 close_list 代表尋找過。

b. 接著將它周圍的九宮格的每一個點開始測量，如果已經在 close_list，及已經搜尋過則跳過。否則將其 G 假定為從 x 到其的距離（斜角 1.4，直線 1）加上 x 原本的 G 值。

c. 接下來檢查它有沒有在 open_list 裡，如果沒有則代表它是完全未接觸過的點，可以優先考慮；否則若新的路徑比較短也可以優先考慮，不然就不考慮。接著更新它的數值。如果它已經是終點了也要紀錄下來。重複以上 a. b. c. 直到 open_list 是空的，即沒有可以考慮的點了。

main 函數：

```
def main():
    map_str = '''
        S 0 0 0 0 0
        0 0 0 0 # #
        0 0 0 0 0 0
        # # # # 0 0
        # 0 0 0 0 #
        0 0 E 0 0 #
    '''
    exam_map, start_pos, end_pos = maps_to_array(map_str)
    print(start_pos.POS)
    print(end_pos.POS)
    '''
    exam_map = [
        [0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 1, 1],
        [0, 0, 0, 0, 0, 0],
        [1, 1, 1, 1, 0, 0],
        [1, 0, 0, 0, 0, 1],
        [0, 0, 0, 0, 0, 1]
    ]
    '''
```

```
    print('Map looks like this:')
    for row in exam_map:
        for item in row:
            print(item, end=' ')
        print()
    print()
    route = a_star(start_pos, end_pos, exam_map)
    for row in range(len(exam_map)):
        for col in range(len(exam_map[row])):
            if Node(row, col) in route:
                print('*', end=' ')
            else:
                print(exam_map[row][col], end=' ')
        print()
```

即簡單驅動 astar 演算法及顯示結果

輔助將文字轉換為使用格式的副程式

```
def maps_to_array(s: str):
    map_ = []
    lines = s.strip().split('\n')
    line_count = 0
    char_count = 0
    start_pos = None
    end_pos = None
    for line in lines:
        line_arr = []
        for char in line:
            if char == 'S':
                start_pos = StartNode(line_count, char_count)
                line_arr.append(0)
                char_count += 1
            elif char == 'E':
                end_pos = Node(line_count, char_count)
                char_count += 1
                line_arr.append(0)
                char_count += 1
            elif char == '0':
                line_arr.append(0)
                char_count += 1
            elif char == '#':
                line_arr.append(1)
                char_count += 1
            else:
                pass
        map_.append(line_arr)
        char_count = 0
        line_count += 1
        map_.append(line_arr)
        char_count = 0
        line_count += 1
    if start_pos is None or end_pos is None:
        raise StringConvertToArrayFailedException()
    return map_, start_pos, end_pos
```

3. 執行結果：
輸入：

```
map_str = '''
    S 0 0 0 0 0
    0 0 0 0 # #
    0 0 0 0 0 0
    # # # # 0 0
    # 0 0 0 0 #
    0 0 E 0 0 #
'''
```

```
C:\Users\edwar\AppData\Local\Programs\Python\Python38\python.exe D:/Grade3/AI/astar/main.py
Map looks like this:
0 0 0 0 0 0
0 0 0 0 1 1
0 0 0 0 0 0
1 1 1 1 0 0
1 0 0 0 0 1
0 0 0 0 0 1

Found path!
* 0 0 0 0 0
0 * 0 0 1 1
0 0 * * 0 0
1 1 1 1 * 0
1 0 0 * 0 1
0 0 * 0 0 1
```

若以原地圖來看則是：