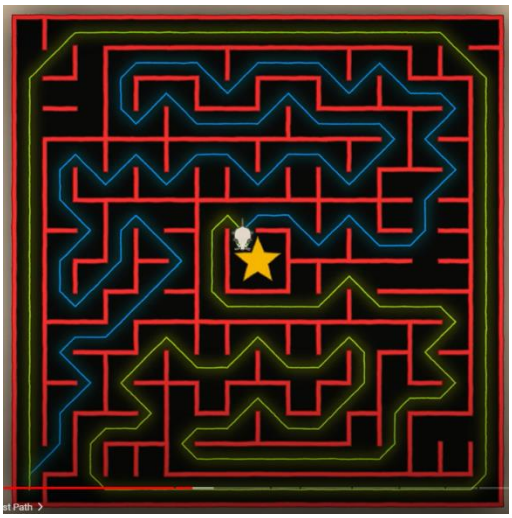
Project Akhir Mata Kuliah Pemrograman Lanjut Kelas C

LABIRIN



Labirin merupakan masalah klasik yang masih tetap menarik untuk dibahas hingga kini. Silakan menonton video tentang kompetisi labirin pada tautan berikut

https://www.youtube.com/watch?v=ZMQbHMgK2rw

Pada proyek akhir ini, kita akan mencoba menyelesaikan masalah labirin tersebut dalam versi yang sederhana.

Dalam video tersebut dijelaskan berbagai tantangan dan solusi yang diajukan oleh berbagai peserta lomba dari berbagai negara.

Berikut ini adalah salah satu contoh solusi algoritma *depth first search* untuk menjadi jalur labirin menggunakan Python.

Maze.py

```
1   '''
2   Author: Mahendra Data - https://github.com/mahendradata
3   References: https://www.geeksforgeeks.org/rat-in-a-maze/
4   '''
5   class Maze:
6
7       MOVE = (
8           (-1, 0), # UP
9           (1, 0), # Down
10          (0, -1), # Left
11          (0, 1), # Right
12      )
13
14      # Public function for solving the maze
15      def solve(self, maze):
16          self.MAZE = maze
17          self.SIZE = (len(maze), len(maze[0]))
18          self.FINISH = (self.SIZE[0]-1, self.SIZE[1]-1)
19
20          self.PATH = [[0 for i in range(self.SIZE[1])] for i in range(self.SIZE[0])]
21          self.PATH[0][0] = 1
22
23          print("Maze")
24          Maze.display(maze)
25
26
27          if self.__run__(0, 0):
28              print(f"Solution")
29              Maze.display(self.PATH)
30          else:
31              print("Solution does not exist")
32
33      # Helper function to display 2D array
34      def display(matrix):
35          for r in matrix:
36              for c in r:
37                  print("." if c else "#", end=" ")
```

```
            print()

    # Private recursive function for solving the maze
    def __run__(self, row, col):
        if (row, col) == self.FINISH:
            return True

        for move_row, move_col in Maze.MOVE:
            new_row = row + move_row
            new_col = col + move_col
            if self.__is_valid__(new_row, new_col):
                self.PATH[new_row][new_col] = 1
                if self.__run__(new_row, new_col):
                    return True
                self.PATH[new_row][new_col] = 0

        return False

    # Private function to check the validity of a move
    def __is_valid__(self, row, col):
        # If out of the maze's border
        if row < 0 or col < 0 or row >= self.SIZE[0] or col >= self.SIZE[1]:
            return False

        # If hit the maze's wall
        if self.MAZE[row][col] == 0:
            return False

        # If the path has been visited
        if self.PATH[row][col] == 1:
            return False

        return True  # If the move is valid


# Driver program to test Maze class
if __name__ == "__main__":
    solver = Maze()
    maze = [[1, 0, 0, 0],
            [1, 1, 0, 1],
            [0, 1, 0, 0],
            [1, 1, 1, 1]]
    solver.solve(maze)
    print()

    maze = [[1, 0, 0, 0],
            [1, 1, 1, 1],
            [0, 1, 0, 0],
            [1, 1, 0, 1]]
    solver.solve(maze)
    print()

    maze = [[1, 0, 1, 1, 1],
            [1, 0, 1, 0, 1],
            [1, 0, 1, 1, 1],
            [1, 1, 1, 0, 1],
            [1, 0, 1, 0, 1]]
    solver.solve(maze)
    print()

    maze = [[1, 1, 1, 1, 1],
            [1, 0, 1, 0, 1],
            [1, 1, 1, 1, 1],
            [1, 0, 1, 0, 1],
            [1, 1, 1, 1, 1]]
    solver.solve(maze)
```

	Contoh *Output*
	Maze . # # # . . # . # . # # . . . . Solution . # # # . . # # # . # # # . . . Maze . # # # . . . . # . # # . . # . Solution does not exist Maze . # . . . . # . # . . # . . . . . . # . . # . # . Solution . # . . . . # . # . . # . # . . . . # . # # # # . Maze . . . . . . # . # . . . . # . # . . . . . . Solution . # . . . . # . # . . # . # . . # . # . . . . # .

Maze.py ini adalah contoh program Python untuk mencari jalur keluar dari labirin dengan algoritma *backtracking* + *depth first search*.

Dalam Maze.py ini, titik mulai (*starting point*) labirin selalu dimulai dari baris 0 dan kolom 0 dengan titik akhir (*finishing point*) adalah baris B-1 dan kolom K-1, yang mana B adalah jumlah baris dan K adalah jumlah kolom dari labirin tersebut. Misalnya, bila B adalah 5 dan K adalah 4, maka titik akhir dari labirin tersebut adalah (4,3) karena indeks *list* pada Python dimulai dari 0.

Pada contoh program Maze.py tersebut, peta labirin disederhanakan ke dalam bentuk Array 2 dimensi dengan nilai 1 atau 0. Nilai 1 menandakan bahwa titik tersebut dapat dilewati, sedangkan titik 0 menandakan titik tersebut tidak dapat dilewati.

Pelajarilah program tersebut lalu lakukan modifikasi program sesuai dengan intrusi-intrusi berikut

Tugas 1

Tugas ke-1 Anda adalah menambahkan fungsi pada program Maze.py untuk menghitung langkah yang diperlukan untuk mencapai titik akhir dari titik awal?

Contoh *output* untuk Tugas 1 adalah sebagai berikut:

```
Maze
. # # #
. . # .
# . # #
. . . .
Solution has 7 steps
. # # #
. . # #
# . # #
# . . .
```

Tugas 2

Tugas ke-2 Anda adalah memodifikasi program **Maze.py** pada Tugas 1 agar dapat digunakan untuk mencari solusi labirin dengan titik awal dan titik akhir yang berbeda?

Contoh *output* untuk Tugas 2 adalah sebagai berikut:

```
Maze start at (1, 1) and finish at (0, 2)
. # . .
. . # .
# . # .
. . . .
Solution has 9 steps
# # . .
# . # .
# . # .
# . . .

Maze start at (1, 1) and finish at (3, 3)
. # # #
. . . .
# . # #
. . # .
Solution does not exist

Maze start at (2, 2) and finish at (0, 0)
. # . . .
. # . # .
. # . . .
. . . # .
. # . # .
Solution has 7 steps
. # # # #
. # # # #
. # . # #
. . . # #
# # # # #
```

Tugas 3

Kelemahan dari Maze.py adalah jalur yang dihasilkan belum tentu yang terpendek. Ubahkan Maze.py agar dapat menghasilkan jalur yang lebih pendek. Tugas ke-3 Anda adalah memodifikasi program Maze.py hasil dari Tugas 2 agar dapat menghasilkan jalur yang lebih baik dari yang dihasilkan oleh Maze.py.

Contoh *output* untuk Tugas 3 adalah sebagai berikut:

```
Maze start at (0, 0) and finish at (4, 4)
. # . . .
. # . # .
. # . . .
. . . # .
. # . # .
Solution has 11 steps
. # # # #
. # # # #
. # . . .
. . . # .
# # # # .
```

Cara Pengerjaan:

1. Project ini dikerjakan secara berkelompok dengan maksimal anggota **2 orang mahasiswa**.
2. Pengumpulan tugas dilakukan melalui **Brone** oleh salah satu mahasiswa anggota tiap kelompok.
3. Batas akhir pengumpulan tugas adalah tanggal **28 November 2023 pukul 12:50 WIB**.
4. Tugas dipresentasikan ke dosen pada tanggal **28 November 2023 pukul 12:50 WIB di kelas**.