情報処理 III

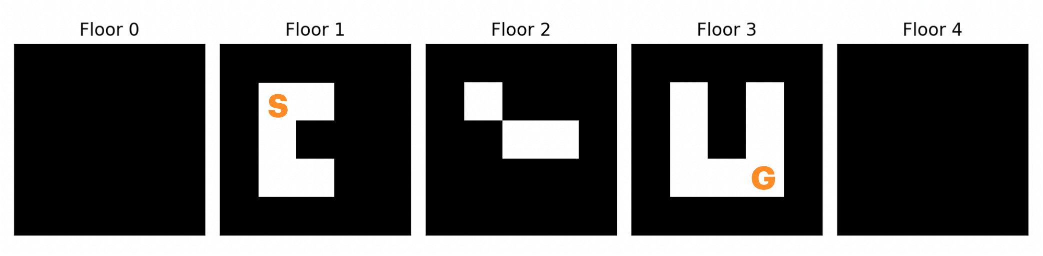
後期 第2回課題

4D38　宮里 孝希

2024/10/04

1.

階の構造



プログラム

```
import numpy as np
import matplotlib.pyplot as plt

# 3次元迷路の定義（0階と4階を壁にする）
maze = [
    [
        [2, 2, 2, 2, 2],
        [2, 2, 2, 2, 2],
        [2, 2, 2, 2, 2],
        [2, 2, 2, 2, 2],
        [2, 2, 2, 2, 2],
    ],  # 0階（全て壁）
    [
        [2, 2, 2, 2, 2],
        [2, 0, 0, 2, 2],
        [2, 0, 2, 2, 2],
        [2, 0, 0, 2, 2],
        [2, 2, 2, 2, 2],
    ],  # 1階
    [
        [2, 2, 2, 2, 2],
        [2, 0, 2, 2, 2],
        [2, 2, 0, 0, 2],
        [2, 2, 2, 2, 2],
        [2, 2, 2, 2, 2],
    ],  # 2階
    [
        [2, 2, 2, 2, 2],
        [2, 0, 2, 0, 2],
        [2, 0, 2, 0, 2],
        [2, 0, 0, 0, 2],
        [2, 2, 2, 2, 2],
    ],  # 3階
    [
        [2, 2, 2, 2, 2],
        [2, 2, 2, 2, 2],
        [2, 2, 2, 2, 2],
        [2, 2, 2, 2, 2],
        [2, 2, 2, 2, 2],
    ],  # 4階（全て壁）
]


# 経路を記録するためのリスト
route_floor = [0 for _ in range(100)]
route_row = [0 for _ in range(100)]
route_col = [0 for _ in range(100)]

# スタックポインタと成功フラグ
stack_pointer = 0
found_exit = 0

# スタート位置とゴール位置
start_floor, start_row, start_col = 1, 1, 1
end_floor, end_row, end_col = 3, 3, 3


def visit(floor, row, col):
    global found_exit, result, stack_pointer
    maze[floor][row][col] = 1  # 現在の位置を訪問済みにする
    route_floor[stack_pointer] = floor
    route_row[stack_pointer] = row
    route_col[stack_pointer] = col

    stack_pointer += 1

    # ゴールに到達した場合
    if floor == end_floor and row == end_row and col ==
end_col:
        for k in range(stack_pointer):
            result += "({:d},{:d},{:d})".format(
                route_floor[k], route_row[k], route_col[k]
            )
        found_exit = 1

    # 右方向に移動
    if found_exit != 1 and maze[floor][row][col + 1] == 0:
        visit(floor, row, col + 1)
    # 下方向に移動
    if found_exit != 1 and maze[floor][row + 1][col] == 0:
        visit(floor, row + 1, col)
    # 左方向に移動
    if found_exit != 1 and maze[floor][row][col - 1] == 0:
        visit(floor, row, col - 1)
    # 上方向に移動
    if found_exit != 1 and maze[floor][row - 1][col] == 0:
        visit(floor, row - 1, col)
    # 下の階に移動
    if found_exit != 1 and floor > 0 and maze[floor - 1][row]
[col] == 0:
        visit(floor - 1, row, col)
    # 上の階に移動
    if found_exit != 1 and floor < len(maze) - 1 and maze[floor
+ 1][row][col] == 0:
        visit(floor + 1, row, col)

    stack_pointer -= 1

    return found_exit

def plot_maze(maze):
    floors = len(maze)
    fig, axes = plt.subplots(1, floors, figsize=(10, 3))

    for f in range(floors):
        ax = axes[f]
        # 壁を黒、通路を白に設定し、行列を転置して描画
        color_map = np.where(np.array(maze[f]) == 2, 0, 1)

        ax.imshow(color_map, cmap="gray", origin="upper")  #
originは"upper"にして座標系に合わせる
        ax.set_title(f"Floor {f}")
        ax.set_xticks([])
        ax.set_yticks([])

    plt.tight_layout()
    plt.show()

plot_maze(maze) # For debug

print("3次元迷路の探索")
result = ""

if visit(start_floor, start_row, start_col):
    print(result)
else:
    print("出口が見つかりません")
```

実行結果

```
(1,1,1)(2,1,1)(3,1,1)(3,2,1)(3,3,1)(3,3,2)(3,3,3)
```