3.1 DFS

```python
import matplotlib.pyplot as plt
import sys
from matplotlib.animation import FuncAnimation
import numpy as np

def find_the_path(lst, now):
    total_path = [now]
    while now in lst:
        now = lst[now]
        total_path.append(now)
    return total_path[::-1]

def dfs(maze):
    rows = len(maze)
    cols = len(maze[0]) if rows > 0 else 0
    start = (0, 0)
    end = (rows - 1, cols - 1)
    max_depth = 0

    iterations = []
    final_path = None

    while True:
        visited = set()
        lst = {}
        stack = [(start, 0)]
        found = False

        # 记录本次迭代的访问顺序
        current_visited_order = []

        while stack:
            now, depth = stack.pop()
            current_visited_order.append(now)

            if now == end:
                final_path = find_the_path(lst, end)
                found = True
                break

            if depth < max_depth:
                visited.add(now)
                for dx, dy in [(-1, 0), (1, 0), (0, - 1), (0, 1)]:
                    x = now[0] + dx
                    y = now[1] + dy
                    nxt = (x, y)
                    if 0 <= x < rows and 0 <= y < cols and maze[x][y] == 0 and nxt not in visite
                        lst[nxt] = now
                        stack.append((nxt, depth + 1))

        iterations.append({
            "visited_order": current_visited_order,
            "found": found,
            "path": final_path[:] if found else None
        })

        if found:
            break

        max_depth += 1

    return iterations

def visualize_maze_with_path(maze, iterations):
    fig, ax = plt.subplots(figsize=(len(maze[0]), len(maze)))
    ax.imshow(maze, cmap='Greys', interpolation='nearest')

    # 设置坐标轴
    ax.set_xticks(range(len(maze[0])))
    ax.set_yticks(range(len(maze)))
    ax.set_xticks([x - 0.5 for x in range(1, len(maze[0]))], minor=True)
    ax.set_yticks([y - 0.5 for y in range(1, len(maze))], minor=True)
    ax.grid(which="minor", color="black", linestyle='-', linewidth=2)

    # 初始化绘图元素
    scatter = ax.scatter([], [], s=10, color='blue', alpha=0.5)
    line, = ax.plot([], [], marker='o', markersize=8, color='red', linewidth=3)

    # 计算总帧数
    total_frames = sum(len(iter["visited_order"]) for iter in iterations)
    total_frames += len(iterations[-1]["path"])
    def update(frame):
        # 确定当前属于哪个阶段
        cum_frames = 0
        current_stage = 0
        path_stage = False

        for i, iter in enumerate(iterations):
            if frame < cum_frames + len(iter["visited_order"]):
                current_stage = i
                break
            cum_frames += len(iter["visited_order"])

        if frame >= total_frames - len(iterations[-1]["path"]):
            path_stage = True
            path_frame = frame - (total_frames - len(iterations[-1]["path"]))

        if not path_stage:
            current_iter = iterations[current_stage]
            frames_in_stage = frame - cum_frames
            visited_x, visited_y = zip(*current_iter["visited_order"][:frames_in_stage+1])
            scatter.set_offsets(np.column_stack([visited_y, visited_x]))
        else:
            final_iter = iterations[-1]
            all_visited = set()
            for iter in iterations:
                all_visited.update(iter["visited_order"])
            visited_x, visited_y = zip(*all_visited)
            scatter.set_offsets(np.column_stack([visited_y, visited_x]))

            if path_frame < len(final_iter["path"]):
                path_x, path_y = zip(*final_iter["path"][:path_frame+1])
                line.set_data(path_y, path_x)

        return scatter, line

    ani = FuncAnimation(fig, update, frames=total_frames, interval=500, blit=True, repeat=False
    plt.show()

input = sys.stdin.read().split()
idx = 0
n = int(input[idx])
idx += 1
```

```
m = int(input[idx])
idx += 1

maze = []
for _ in range(n):
    row = list(map(int, input[idx:idx+m]))
    maze.append(row)
    idx += m

iterations = dfs(maze)
print(len(iterations[-1]['path']) - 1)

visualize_maze_with_path(maze, iterations)
```

3.2 BFS

```python
from collections import deque
import matplotlib.pyplot as plt
import sys
from matplotlib.animation import FuncAnimation
import numpy as np
# 回溯路径
def find_the_path(lst, now):
    total_path = [now]
    while now in lst:
        now = lst[now]
        total_path.append(now)
    return total_path[::-1]



def bfs(maze):
    rows = len(maze)
    cols = len(maze[0]) if rows > 0 else 0
    start = (0, 0)
    end = (rows - 1, cols - 1)

    queue = deque()
    queue.append(start)
    visited = set()
    visited.add(start)
    visited_order = []
    lst = {}

    while queue:
        now = queue.popleft()
        visited_order.append(now)
        if now == end:
            return find_the_path(lst, now), visited_order

        for dx, dy in [(-1, 0), (1, 0), (0, -1), (0, 1)]:
            x = now[0] + dx
            y = now[1] + dy
            nxt = (x, y)

            if 0 <= x < rows and 0 <= y < cols and maze[x][y] == 0:
                if nxt not in visited:
```

```
                lst[nxt] = now
                visited.add(nxt)
                queue.append(nxt)

def visualize_maze_with_path(maze, path, visited_order):
    fig, ax = plt.subplots(figsize=(len(maze[0]), len(maze))) # 设置图形大小
    ax.imshow(maze, cmap='Greys', interpolation='nearest')  # 使用灰度色图，并关闭插值

    # 设置坐标轴刻度和边框
    ax.set_xticks(range(len(maze[0])))
    ax.set_yticks(range(len(maze)))
    ax.set_xticks([x - 0.5 for x in range(1, len(maze[0]))], minor=True)
    ax.set_yticks([y - 0.5 for y in range(1, len(maze))], minor=True)
    ax.grid(which="minor", color="black", linestyle='-', linewidth=2)

    # 初始化空的散点图和线图
    scatter = ax.scatter([], [], s=10, color='blue', alpha=0.5)
    line, = ax.plot([], [], marker='o', markersize=8, color='red', linewidth=3)

    # 动画更新函数
    def update(frame):
        # 显示已访问的节点
        if frame < len(visited_order):
            visited_x, visited_y = zip(*visited_order[:frame+1])
            scatter.set_offsets(np.column_stack([visited_y, visited_x]))

        # 显示路径
        if frame >= len(visited_order):
            path_frame = frame - len(visited_order)
            if path_frame < len(path):
                path_x, path_y = zip(*path[:path_frame+1])
                line.set_data(path_y, path_x)

        return scatter, line

    # 计算总帧数（访问过程+路径绘制）
    total_frames = len(visited_order) + len(path)

    # 创建动画
    ani = FuncAnimation(fig, update, frames=total_frames, interval=500, blit=True, repeat=False
    plt.show()
```

```python
input = sys.stdin.read().split()
idx = 0
n = int(input[idx])
idx +=1
m = int(input[idx])
idx +=1

maze = []
for _ in range(n):
    row = list(map(int, input[idx:idx+m]))
    maze.append(row)
    idx += m
path, visited_order = bfs(maze)
print(len(path) - 1)
visualize_maze_with_path(maze, path, visited_order)
```

3.3 Dijkstra

```python
import heapq
import matplotlib.pyplot as plt
import sys
from matplotlib.animation import FuncAnimation
import numpy as np

# 回溯路径
def find_the_path(lst, now):
    total_path = [now]
    while now in lst:
        now = lst[now]
        total_path.append(now)
    return total_path[::-1]


def dijkstra(maze):
    rows = len(maze)
    cols = len(maze[0])
    start = (0, 0)
    end = (rows - 1, cols - 1)

    heap = []
    heapq.heappush(heap, (0, start))

    g_score = {start: 0}
    lst = {}
    visited = set()
    visited_order = []
    while heap:
        now_g, now = heapq.heappop(heap)
        if now in visited:
            continue
        visited.add(now)
        visited_order.append(now)
        if now == end:
            return find_the_path(lst, now), visited_order

        for dx, dy in [(-1, 0), (1, 0), (0, -1), (0, 1)]:
            x = now[0] + dx
            y = now[1] + dy
            nxt = (x, y)

            if 0 <= x < rows and 0 <= y < cols and maze[x][y] == 0:
                new_g = now_g + 1
                if nxt not in g_score or new_g < g_score.get(nxt, float('inf')):
                    lst[nxt] = now
                    g_score[nxt] = new_g
                    heapq.heappush(heap, (new_g, nxt))

def visualize_maze_with_path(maze, path, visited_order):
    fig, ax = plt.subplots(figsize=(len(maze[0]), len(maze))) # 设置图形大小
    ax.imshow(maze, cmap='Greys', interpolation='nearest')  # 使用灰度色图，并关闭插值

    # 设置坐标轴刻度和边框
    ax.set_xticks(range(len(maze[0])))
    ax.set_yticks(range(len(maze)))
    ax.set_xticks([x - 0.5 for x in range(1, len(maze[0]))], minor=True)
    ax.set_yticks([y - 0.5 for y in range(1, len(maze))], minor=True)
    ax.grid(which="minor", color="black", linestyle='-', linewidth=2)

    # 初始化空的散点图和线图
    scatter = ax.scatter([], [], s=10, color='blue', alpha=0.5)
    line, = ax.plot([], [], marker='o', markersize=8, color='red', linewidth=3)

    # 动画更新函数
    def update(frame):
        # 显示已访问的节点
        if frame < len(visited_order):
            visited_x, visited_y = zip(*visited_order[:frame+1])
            scatter.set_offsets(np.column_stack([visited_y, visited_x]))

        # 显示路径
        if frame >= len(visited_order):
            path_frame = frame - len(visited_order)
            if path_frame < len(path):
                path_x, path_y = zip(*path[:path_frame+1])
                line.set_data(path_y, path_x)

        return scatter, line

    # 计算总帧数（访问过程+路径绘制）
    total_frames = len(visited_order) + len(path)
    # 创建动画
    ani = FuncAnimation(fig, update, frames=total_frames, interval=500, blit=True, repeat=False
    plt.show()


input = sys.stdin.read().split()
idx = 0
n = int(input[idx])
idx +=1
m = int(input[idx])
idx +=1

maze = []
for _ in range(n):
    row = list(map(int, input[idx:idx+m]))
    maze.append(row)
    idx += m
path, visited_order = dijkstra(maze)
print(len(path) - 1)
visualize_maze_with_path(maze, path, visited_order)
```

3.4 A star

```
import heapq
import matplotlib.pyplot as plt
import sys
from matplotlib.animation import FuncAnimation
import numpy as np

# 启发式函数
def distance(a, b):
    return abs(a[0] - b[0]) + abs(a[1] - b[1])

# 回溯路径
def find_the_path(lst, now):
    total_path = [now]
    while now in lst:
        now = lst[now]
        total_path.append(now)
    return total_path[::-1]

def A_star(maze):
    rows = len(maze)
    cols = len(maze[0])
    start = (0, 0)
    end = (rows - 1, cols - 1)

    heap = []
    heapq.heappush(heap, (0, 0, start))

    g_score = {start: 0}
    f_score = {start: distance(start, end)}

    lst = {}
    visited = set()
    visited_order = []

    while heap:
        now_f, now_g, now = heapq.heappop(heap)
        visited.add(now)
        visited_order.append(now)

        if now == end:
            return find_the_path(lst, now), visited_order

        for dx, dy in [(-1, 0), (1, 0), (0, -1), (0, 1)]:
            x = now[0] + dx
            y = now[1] + dy
            nxt = (x, y)

            if 0 <= x < rows and 0 <= y < cols and maze[x][y] == 0:
                new_g = now_g + 1
                if nxt not in g_score or new_g < g_score.get(nxt, float('inf')):
                    lst[nxt] = now
                    g_score[nxt] = new_g
                    f_score[nxt] = new_g + distance(nxt, end)
                    heapq.heappush(heap, (f_score[nxt], new_g, nxt))

def visualize_maze_with_path(maze, path, visited_order):
    fig, ax = plt.subplots(figsize=(len(maze[0]), len(maze))) # 设置图形大小
    ax.imshow(maze, cmap='Greys', interpolation='nearest')  # 使用灰度色图，并关闭插值

    # 设置坐标轴刻度和边框
    ax.set_xticks(range(len(maze[0])))
    ax.set_yticks(range(len(maze)))
    ax.set_xticks([x - 0.5 for x in range(1, len(maze[0]))], minor=True)
    ax.set_yticks([y - 0.5 for y in range(1, len(maze))], minor=True)
    ax.grid(which="minor", color="black", linestyle='-', linewidth=2)

    # 初始化空的散点图和线图
    scatter = ax.scatter([], [], s=10, color='blue', alpha=0.5)
    line, = ax.plot([], [], marker='o', markersize=8, color='red', linewidth=3)

    # 动画更新函数
    def update(frame):
        # 显示已访问的节点
        if frame < len(visited_order):
            visited_x, visited_y = zip(*visited_order[:frame+1])
            scatter.set_offsets(np.column_stack([visited_y, visited_x]))

        # 显示路径
        if frame >= len(visited_order):
            path_frame = frame - len(visited_order)
            if path_frame < len(path):
                path_x, path_y = zip(*path[:path_frame+1])
                line.set_data(path_y, path_x)
        return scatter, line

    # 计算总帧数（访问过程+路径绘制）
    total_frames = len(visited_order) + len(path)

    # 创建动画
    ani = FuncAnimation(fig, update, frames=total_frames, interval=500, blit=True, repeat=False
    plt.show()

input = sys.stdin.read().split()
idx = 0
n = int(input[idx])
idx +=1
m = int(input[idx])
idx +=1

maze = []
for _ in range(n):
    row = list(map(int, input[idx:idx+m]))
    maze.append(row)
    idx += m
path, visited_order = A_star(maze)
print(len(path) - 1)
visualize_maze_with_path(maze, path, visited_order)
```