3.1 DFS

```
import matplotlib.pyplot as plt
import sys

# 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 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

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

        while stack:
            now, depth = stack.pop()
            if now == 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))
        if found:
            path = find_the_path(lst, end)
            return path, visited.union({now for now, _ in stack})
        max_depth += 1

def visualize_maze_with_path(maze, path, visited=None):
    plt.figure(figsize=(len(maze[0]), len(maze))) # 设置图形大小
    plt.imshow(maze, cmap='Greys', interpolation='nearest') # 使用灰度色图，并关闭插值

    # 绘制所有访问过的格子
    if visited is not None:
        visited_x, visited_y = zip(*visited)
        plt.scatter(visited_y, visited_x, s=10, color='blue', alpha=0.5)

    # 绘制路径
    if path:
        path_x, path_y = zip(*path)
        plt.plot(path_y, path_x, marker='o', markersize=8, color='red', linewidth=3)

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

    plt.axis('on') # 显示坐标轴
    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 = dfs(maze)
print(len(path) - 1)
visualize_maze_with_path(maze, path, visited)
```

3.2 BFS

```
from collections import deque
import matplotlib.pyplot as plt
import sys

# 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)
    lst = {}

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

        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=None):
    plt.figure(figsize=(len(maze[0]), len(maze))) # 设置图形大小
    plt.imshow(maze, cmap='Greys', interpolation='nearest') # 使用灰度色图，并关闭插值

    # 绘制所有访问过的格子
    if visited is not None:
        visited_x, visited_y = zip(*visited)
        plt.scatter(visited_y, visited_x, s=10, color='blue', alpha=0.5)

    # 绘制路径
    if path:
        path_x, path_y = zip(*path)
        plt.plot(path_y, path_x, marker='o', markersize=8, color='red', linewidth=3)

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

    plt.axis('on') # 显示坐标轴
    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 = bfs(maze)
print(len(path) - 1)
visualize_maze_with_path(maze, path, visited)
```

3.3 Dijkstra

```
import heapq
import matplotlib.pyplot as plt
import sys

# 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()

    while heap:
        now_g, now = heapq.heappop(heap)
        if now in visited:
            continue
        visited.add(now)

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

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

```
            # 检查边界和障碍
            if 0 <= x < rows and 0 <= y < cols and maze[x][y] == 0:
                new_g = now_g + 1
                # 更新路径
                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=None):
    plt.figure(figsize=(len(maze[0]), len(maze))) # 设置图形大小
    plt.imshow(maze, cmap='Greys', interpolation='nearest') # 使用灰度色图，并关闭插值

    # 绘制所有访问过的格子
    if visited is not None:
        visited_x, visited_y = zip(*visited)
        plt.scatter(visited_y, visited_x, s=10, color='blue', alpha=0.5)

    # 绘制路径
    if path:
        path_x, path_y = zip(*path)
        plt.plot(path_y, path_x, marker='o', markersize=8, color='red', linewidth=3)

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

    plt.axis('on') # 显示坐标轴
    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 = dijkstra(maze)
print(len(path) - 1)
visualize_maze_with_path(maze, path, visited)
```

3.4 A star

```
import heapq
import matplotlib.pyplot as plt
import sys

# 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):
    # 行数
    rows = len(maze)
    # 列数
    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)}
    # 记录父节点
    lst = {}
    # 记录所有访问过的节点
    visited = set()

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

        if now == end:
```

```
            return find_the_path(lst, now), visited

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

            # 检查边界和障碍
            if 0 <= x < rows and 0 <= y < cols and maze[x][y] == 0:
                new_g = now_g + 1
                # 更新路径
                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=None):
    plt.figure(figsize=(len(maze[0]), len(maze))) # 设置图形大小
    plt.imshow(maze, cmap='Greys', interpolation='nearest') # 使用灰度色图，并关闭插值

    # 绘制所有访问过的格子
    if visited is not None:
        visited_x, visited_y = zip(*visited)
        plt.scatter(visited_y, visited_x, s=10, color='blue', alpha=0.5)

    # 绘制路径
    if path:
        path_x, path_y = zip(*path)
        plt.plot(path_y, path_x, marker='o', markersize=8, color='red', linewidth=3)

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

    plt.axis('on') # 显示坐标轴
    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 = A_star(maze)
print(len(path) - 1)
visualize_maze_with_path(maze, path, visited)
```