```python
# Robot Path Planning with A* – Project 3
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import heapq

def get_neighbors(pos, rows, cols):
    r, c = pos
    for dr, dc in [(-1,0),(1,0),(0,-1),(0,1)]:
        nr, nc = r+dr, c+dc
        if 0 <= nr < rows and 0 <= nc < cols:
            yield nr, nc

def heuristic(a, b):
    return abs(a[0]-b[0]) + abs(a[1]-b[1])

def a_star(grid, start, goal):
    rows, cols = grid.shape
    open_set = []
    heapq.heappush(open_set, (0, start))
    came_from = {}
    g = {start: 0}
    while open_set:
        _, current = heapq.heappop(open_set)
        if current == goal:
            path = []
            while current in came_from:
                path.append(current)
                current = came_from[current]
            path.append(start)
            return path[::-1]
        for nb in get_neighbors(current, rows, cols):
            if grid[nb] == 1:
                continue
            tentative_g = g[current] + 1
            if nb not in g or tentative_g < g[nb]:
                g[nb] = tentative_g
                f = tentative_g + heuristic(nb, goal)
                heapq.heappush(open_set, (f, nb))
                came_from[nb] = current
    return None

def visualize(grid, path, start, goal):
    rows, cols = grid.shape
    plt.imshow(grid, cmap="Greys", origin="lower")
    if path:
        pr = [p[0] for p in path]
        pc = [p[1] for p in path]
        plt.scatter(pc, pr, s=30, c="red")
    plt.scatter(start[1], start[0], c="green", s=80)
    plt.scatter(goal[1], goal[0], c="blue", s=80)
    plt.xticks(range(cols))
    plt.yticks(range(rows))
    plt.grid(True)
    plt.show()

def read_obstacles(grid):
    rows, cols = grid.shape
    n = int(input("Number of obstacles: "))
    for _ in range(n):
        r, c = map(int, input("Obstacle (row col, 0-based): ").split())
        if 0 <= r < rows and 0 <= c < cols:
            grid[r, c] = 1

def read_point(prompt, rows, cols):
    while True:
        r, c = map(int, input(prompt).split())
        if 0 <= r < rows and 0 <= c < cols:
            return (r, c)
        print("Out of range.")

def main():
    while True:
        rows, cols = map(int, input("Grid size (rows cols): ").split())
        grid = np.zeros((rows, cols), dtype=int)
        read_obstacles(grid)
        print("Grid:")
        print(pd.DataFrame(grid))
        start = read_point("Start (row col): ", rows, cols)
        goal = read_point("Goal (row col): ", rows, cols)
        if grid[start] == 1 or grid[goal] == 1:
            print("Start or goal on obstacle.")
        else:
            path = a_star(grid, start, goal)
            if path is None:
                print("No valid path found.")
            else:
                print("Path:", path)
                visualize(grid, path, start, goal)
        again = input("Try again? (y/n): ").strip().lower()
        if again != "y":
            break

if __name__ == "__main__":
    main()
```