Objective:

Develop a pathfinding algorithm to navigate a robot through a grid with obstacles. The goal is to find the shortest path from a starting point to a destination point.

Problem Description:

You are given a 2D grid representing a map of an environment where:

- `0` represents an empty cell where the robot can move.
- `1` represents an obstacle where the robot cannot move.

The robot can move up, down, left, or right (but not diagonally). Given a starting point and a destination point, you need to find the shortest path from the start to the destination. If there is no valid path, the function should return an indication that no path is found.

Input:

- A 2D array `grid` of size `m x n`.
- A pair of integers `(startX, startY)` representing the starting point.
- A pair of integers `(endX, endY)` representing the destination point.

Output:

- A list of tuples representing the path from start to destination (inclusive).
- If no path exists, return an empty list or indicate that no path is found.

Example:

Input:

```
plaintext
Copy code
grid = [
    [0, 0, 1, 0],
    [0, 1, 0, 0],
    [0, 0, 0, 1],
    [0, 1, 0, 0]
]
start = (0, 0)
end = (3, 3)
```

Output:

```
plaintext
Copy code
[(0, 0), (1, 0), (2, 0), (2, 1), (2, 2), (3, 2), (3, 3)]
```

Instructions:

1. Implement the A* or Dijkstra's algorithm to find the shortest path in the grid.
2. Optimize the solution to handle larger grids efficiently.
3. Handle edge cases such as the start or end points being on an obstacle or no possible path.

Bonus:

- Implement a visualization of the grid and the path found (using a simple console output or a graphical library).
- Allow the user to input the grid, start, and end points.

Submission Guidelines:

1. Create a GitHub repository and push your code there. Share the link with us.
2. Include a README file that explains how to set up and run your application, as well as any assumptions you made.
3. Provide instructions on how to run the unit tests.