# Table of Contents

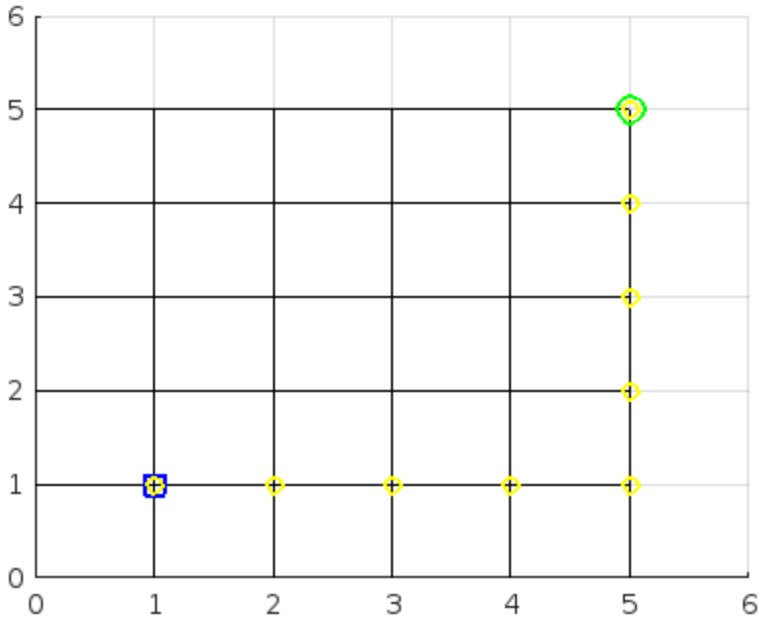


```
clc;
clear;

% Set up the environment
gridSize = 5; % 5x5 grid
goal = [5, 5]; % Goal position at (5,5)

% Initial position of the agent (source)
agentPos = [1, 1]; % Starting at the top-left corner

% Directions for movement (up, down, left, right)
directions = [0, 1; 0, -1; 1, 0; -1, 0]; % Right, Left, Down, Up

% Set up the grid for visualization
figure;
axis([0 gridSize+1 0 gridSize+1]); % Set axis limits
hold on;
grid on;

% Plot the grid
for i = 1:gridSize
    plot([i, i], [0, gridSize], 'k'); % Vertical lines
    plot([0, gridSize], [i, i], 'k'); % Horizontal lines
end

% Plot the goal position
plot(goal(2), goal(1), 'go', 'MarkerSize', 10, 'LineWidth', 2); % Green
circle for goal

% Plot the agent's initial position (blue square)
plot(agentPos(2), agentPos(1), 'bs', 'MarkerSize', 10, 'LineWidth', 2);

% BFS function to find the shortest path from start to goal
function path = bfs(gridSize, start, goal, directions)
    % Initialize visited array and parent array
    visited = false(gridSize);
    parent = NaN(gridSize, gridSize, 2); % To track the path (store row and
column as a 2-element vector)
    queue = start;
    visited(start(1), start(2)) = true;

    while ~isempty(queue)
        current = queue(1, :); % Get the first element
        queue(1, :) = []; % Remove it from the queue

        % If we reached the goal, reconstruct the path
        if isequal(current, goal)
```

```
            path = reconstructPath(parent, start, goal);
            return;
        end

        % Explore neighbors (right, left, down, up)
        for i = 1:size(directions, 1)
            neighbor = current + directions(i, :);

            % Check if the neighbor is valid (inside grid)
            if isValid(neighbor, gridSize, visited)
                queue = [queue; neighbor]; % Add the neighbor to the queue
                visited(neighbor(1), neighbor(2)) = true;
                parent(neighbor(1), neighbor(2), :) = current; % Store the
parent (row, column)
            end
        end
    end
    path = []; % No path found
end

% Check if the position is within grid and not visited
function valid = isValid(pos, gridSize, visited)
    valid = (pos(1) > 0 && pos(1) <= gridSize && pos(2) > 0 && pos(2) <=
gridSize ...
             && ~visited(pos(1), pos(2)));
end

% Reconstruct the path from the parent array
function path = reconstructPath(parent, start, goal)
    path = goal;
    current = goal;
    while ~isequal(current, start)
        % Get the parent coordinates from the parent array
        current = squeeze(parent(current(1), current(2), :))';
        path = [current; path]; % Add to the path
    end
end

% Find the path using BFS
path = bfs(gridSize, agentPos, goal, directions);

% --- Visualization: Show the path ---
if ~isempty(path)
    for i = 1:size(path, 1)
        dbstop in BFS at 88
        plot(path(i, 2), path(i, 1), 'yo', 'MarkerSize', 6, 'LineWidth', 2);
% Yellow circles for path
        pause(0.5); % Pause to visualize the movement
    end
    disp('Path found!');
else
    disp('No path found!');
end
```

```
% Pause to visualize the process
pause(1); % Wait for 1 second before exiting
```

*Path found!*




*Published with MATLAB® R2024b*