```
from collections import deque

# Maze
# 0 = free path, 1 = wall
maze = [
    [1, 0, 1, 0, 0],
    [1, 0, 0, 0, 0],
    [0, 0, 0, 1, 0],
    [1, 0, 0, 0, 1],
    [0, 0, 1, 0, 0]
]
start = (0, 0)  # Start point(row,col)
goal = (0, 4)  # Goal(row,col)

# Directions: up,down,left,right
direction = [(-1, 0), (1, 0), (0, -1), (0, 1)]


def bfs(start, goal):
    queue = deque([start])
    visited = set([start])
    parent = {start: None}
    while queue:
        r, c = queue.popleft()
        if (r, c) == goal:
            break
        for dr, dc in direction:
            nr, nc = r + dr, c + dc
            if 0 <= nr < len(maze) and 0 <= nc < len(maze[0]):
                if maze[nr][nc] == 0 and (nr, nc) not in visited:
                    queue.append((nr, nc))
                    visited.add((nr, nc))
                    parent[(nr, nc)] = (r, c)

    # Build the path
    path = []
    node = goal
    while node is not None:
        path.append(node)
        node = parent.get(node)
    path.reverse()
```

```
    # If path exist, retun it
    if path[0] == start:
        return path
    else:
        return None


path = bfs(start, goal)
if path:
    print("Path found: ", path)
else:
    print("Path not found!")
```