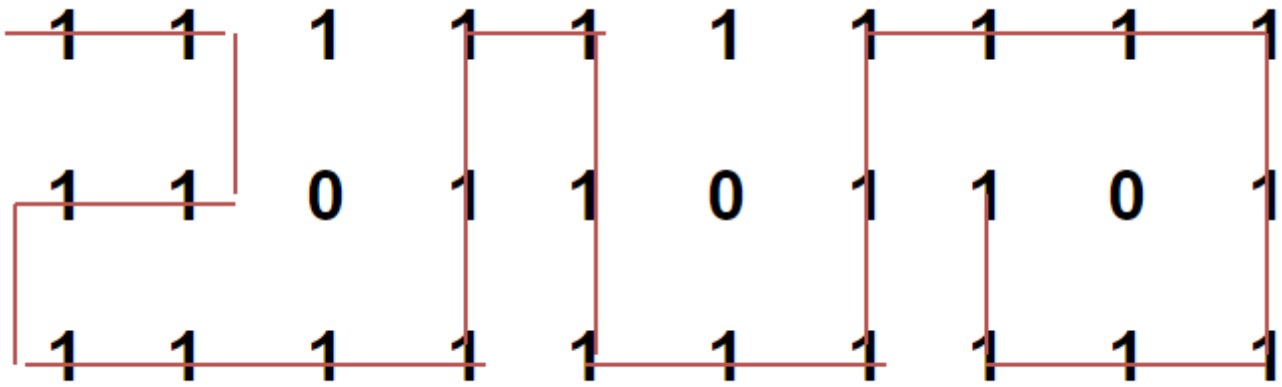
# Longest Possible Route in a Matrix with Hurdles

Given an M x N matrix, with a few hurdles arbitrarily placed, calculate the length of the longest possible route possible from source to a destination within the matrix. We are allowed to move to only adjacent cells which are not hurdles. The route cannot contain any diagonal moves and a location once visited in a particular path cannot be visited again.
For example, the longest path with no hurdles from source to destination is highlighted below.




```
def findLongestRoute(matrix, destination):
    di, dj = destination
    n, m = len(matrix), len(matrix[0])
    visited = [[False] * m for i in range(n)]
    return findLongestRouteUtil(matrix, di, dj, 0, 0, n, m, 0, visited)


def findLongestRouteUtil(matrix, di, dj, i, j, n, m, count, visited):
    if i < 0 or j < 0 or i >= n or j >= m or matrix[i][j] == 0 or
visited[i][j] is True:
        return 0
    if i == di and j == dj:
        return 0
    visited[i][j] = True
    up = findLongestRouteUtil(matrix, di, dj, i - 1, j, n, m, count + 1,
visited)
    right = findLongestRouteUtil(matrix, di, dj, i, j + 1, n, m, count +
1, visited)
    down = findLongestRouteUtil(matrix, di, dj, i + 1, j, n, m, count + 1,
visited)
    left = findLongestRouteUtil(matrix, di, dj, i, j - 1, n, m, count + 1,
visited)
```

```
    visited[i][j] = False
    return 1 + max(max(max(up, right), down), left)


matrix = [[1, 1, 1],
          [0, 1, 1],
          [1, 1, 0]]
print(findLongestRoute(matrix, [2, 2]))
```