Sai Jain 1BM18CS040

AI - LAB TEST - 1

Page No.
Date :

Q4 The algoritem accepts the initial maze, the source and the destination cells are provided and if it is the ending cell, it returns True, if it is a wall or an alreadey visited cell and the destination is not achieved, it returns FALSE. The neighbouring cells are explored recursively and if nothing is found at the end, it returns false, so implement back track to explore paths, we will again start with cell : Source

Algoritem :

Initialize the source and destination in maze with walls.

```
def euclid dist :
    dist = mat. sqrt (m-1- x[0])**2 + (m-1)-x[1]**2
    return dist

def. find Shortest Path :
    min dist : 9999 :
    for x in next Path
        if euclid Dist (x, n, m) < min Dist :
            min Dist : euclid Dist (x, n, m)
        next = x
    return next
```

```
def findPath (n, m):
    path.append ([0,0])
    current = [0,0]
while (current != [n-1, m-1]):
    nextPath = []
    for x in neighbours:
        a = []
    a.apped (current[0] + x[0])
    a.append (current[1] + x[1])

    if (next Path exist)
        current = find Shortest Path (next Path n, m)

    else
        closed Path.apped (current)
    if path:
        closedPath.append (current)
    else:
        print ("No Path").

def start()
    n = int (input ("Enter the no. of rows"))
    m = int (input ("Enter the maze cols"))
    [][] = int(input ("Enter maze structure"))
```