



**Name:** Insha Nadeem

**Roll Number:** 220985

**Program:** BSAI-IV-A

**Submitted To:** Sir Tahir Akram

**Date:** 23 March 2024

**Course:** Artificial Intelligence

**Assignment:** 01

# Maze Game

Modify maze lab assignment for BFS, DFS, Hill Climbing, and greedy search algorithms.

## BFS Approach :

```
from pyamaze import maze, agent, textLabel
from queue import Queue

def bfs(m):
    start = (m.rows, m.cols)
    queue = Queue()
    queue.put(start)
    visited = {start: None}

    while not queue.empty():
        currCell = queue.get()
        if currCell == (1, 1):
            break
        for d in 'ESNW':
            if m.maze_map[currCell][d] == True:
                if d == 'E':
                    childCell = (currCell[0], currCell[1] + 1)
                elif d == 'W':
                    childCell = (currCell[0], currCell[1] - 1)
                elif d == 'N':
                    childCell = (currCell[0] - 1, currCell[1])
                elif d == 'S':
                    childCell = (currCell[0] + 1, currCell[1])

                if childCell not in visited:
                    visited[childCell] = currCell
                    queue.put(childCell)

    fwdPath = {}
    cell = (1, 1)
    while cell != start:
        fwdPath[visited[cell]] = cell
        cell = visited[cell]
```

```
    return fwdPath

if __name__ == '__main__':
    m = maze(10, 10)
    m.CreateMaze()
    path = bfs(m)

    a = agent(m, footprints=True)
    m.tracePath({a: path})
    l = textLabel(m, 'BFS Path Length', len(path) + 1)

    m.run()
```

## **DFS Approach :**

```
from pyamaze import maze, agent, textLabel

def dfs(m):
    start = (m.rows, m.cols)
    stack = [start]
    visited = {start: None}

    while stack:
        currCell = stack.pop()
        if currCell == (1, 1):
            break
        for d in 'ESNW':
            if m.maze_map[currCell][d] == True:
                if d == 'E':
                    childCell = (currCell[0], currCell[1] + 1)
                elif d == 'W':
                    childCell = (currCell[0], currCell[1] - 1)
                elif d == 'N':
                    childCell = (currCell[0] - 1, currCell[1])
                elif d == 'S':
                    childCell = (currCell[0] + 1, currCell[1])

                if childCell not in visited:
                    visited[childCell] = currCell
                    stack.append(childCell)

    fwdPath = {}
    cell = (1, 1)
    while cell != start:
        fwdPath[visited[cell]] = cell
        cell = visited[cell]

    return fwdPath

if __name__ == '__main__':
    m = maze(10, 10)
    m.CreateMaze()
    path = dfs(m)

    a = agent(m, footprints=True)
    m.tracePath({a: path})
```

```
l = textLabel(m, 'DFS Path Length', len(path) + 1)
```

```
m.run()
```

## **Hill Climbing Approach :**

```
from pyamaze import maze, agent, textLabel

def h(cell1, cell2):
    x1, y1 = cell1
    x2, y2 = cell2
    return abs(x1 - x2) + abs(y1 - y2)

def hill_climbing(m):
    start = (m.rows, m.cols)
    currCell = start
    path = []

    while currCell != (1, 1):
        path.append(currCell)
        neighbors = []
        for d in 'ESNW':
            if m.maze_map[currCell][d] == True:
                if d == 'E':
                    childCell = (currCell[0], currCell[1] + 1)
                elif d == 'W':
                    childCell = (currCell[0], currCell[1] - 1)
                elif d == 'N':
                    childCell = (currCell[0] - 1, currCell[1])
                elif d == 'S':
                    childCell = (currCell[0] + 1, currCell[1])

                neighbors.append((childCell, h(childCell, (1, 1))))

        neighbors.sort(key=lambda x: x[1])
        nextCell = neighbors[0][0]
        if nextCell == currCell:
            break
        currCell = nextCell

    return path

if __name__ == '__main__':
    m = maze(10, 10)
    m.CreateMaze()
    path = hill_climbing(m)
```

```
a = agent(m, footprints=True)
m.tracePath({a: path})
l = textLabel(m, 'Hill Climbing Path Length', len(path) + 1)

m.run()
```

## Greedy Search Approach :

```
from pyamaze import maze, agent, textLabel
from queue import PriorityQueue

def h(cell1, cell2):
    x1, y1 = cell1
    x2, y2 = cell2
    return abs(x1 - x2) + abs(y1 - y2)

def greedy_search(m):
    start = (m.rows, m.cols)
    currCell = start
    path = []
    open_set = PriorityQueue()
    open_set.put((h(start, (1, 1)), currCell))

    while not open_set.empty():
        _, currCell = open_set.get()
        if currCell == (1, 1):
            break
        path.append(currCell)
        for d in 'ESNW':
            if m.maze_map[currCell][d] == True:
                if d == 'E':
                    childCell = (currCell[0], currCell[1] + 1)
                elif d == 'W':
                    childCell = (currCell[0], currCell[1] - 1)
                elif d == 'N':
                    childCell = (currCell[0] - 1, currCell[1])
                elif d == 'S':
                    childCell = (currCell[0] + 1, currCell[1])

                open_set.put((h(childCell, (1, 1)), childCell))

    return path

if __name__ == '__main__':
    m = maze(10, 10)
    m.CreateMaze()
    path = greedy_search(m)

    a = agent(m, footprints=True)
```



```
m.tracePath({a: path})  
l = textLabel(m, 'Greedy Best-First Search Path Length', len(path) + 1)  
  
m.run()
```

## Output:

