Project : "490. The Maze" - LC - Depth-First Traversal

NAME: FEVEN BELAY ARAYA
ID: 20027

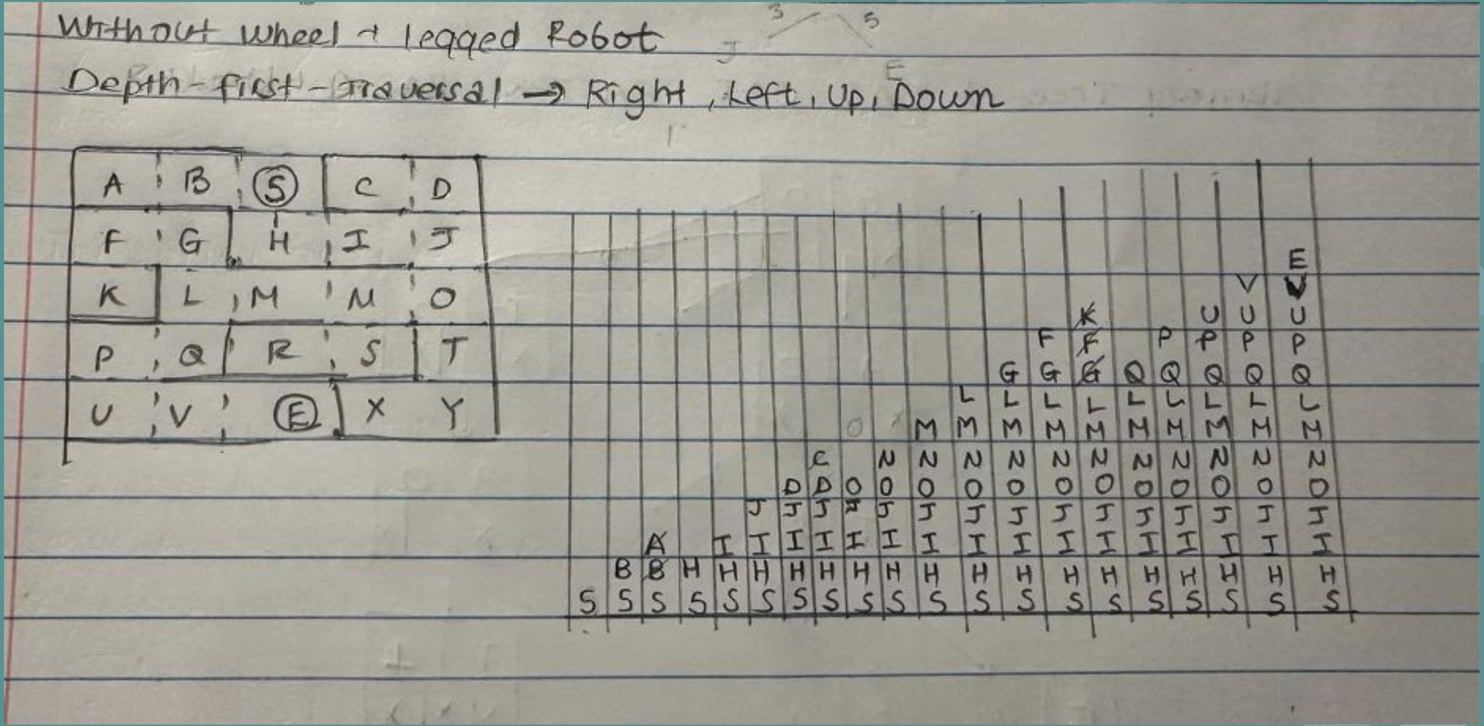
TABLE OF CONTENT

1. INTRODUCTION

- DFS stands for Depth-First Search, which is a fundamental algorithm used in graph theory.
- It explores as far as possible along each branch before backtracking.
- This means that in a graph, it starts at a selected node (the root if you have a tree, or any arbitrary node in a graph).
- It explores as far down as possible along each branch before backtracking.

2. DESIGN

2.1. Without Wheel- Legged Robot



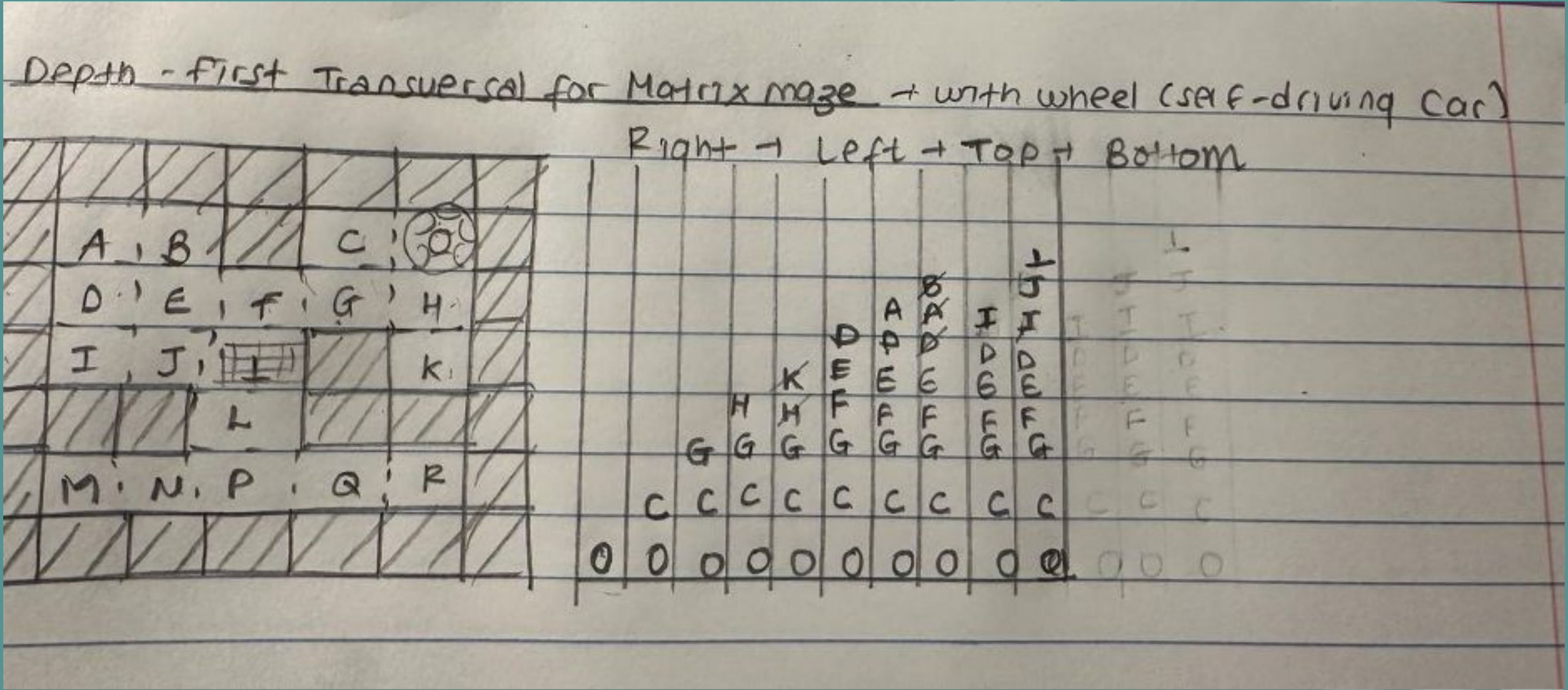
Without wheel + legged Robot

Depth - First - Traversal → Right , Left , Up , Down

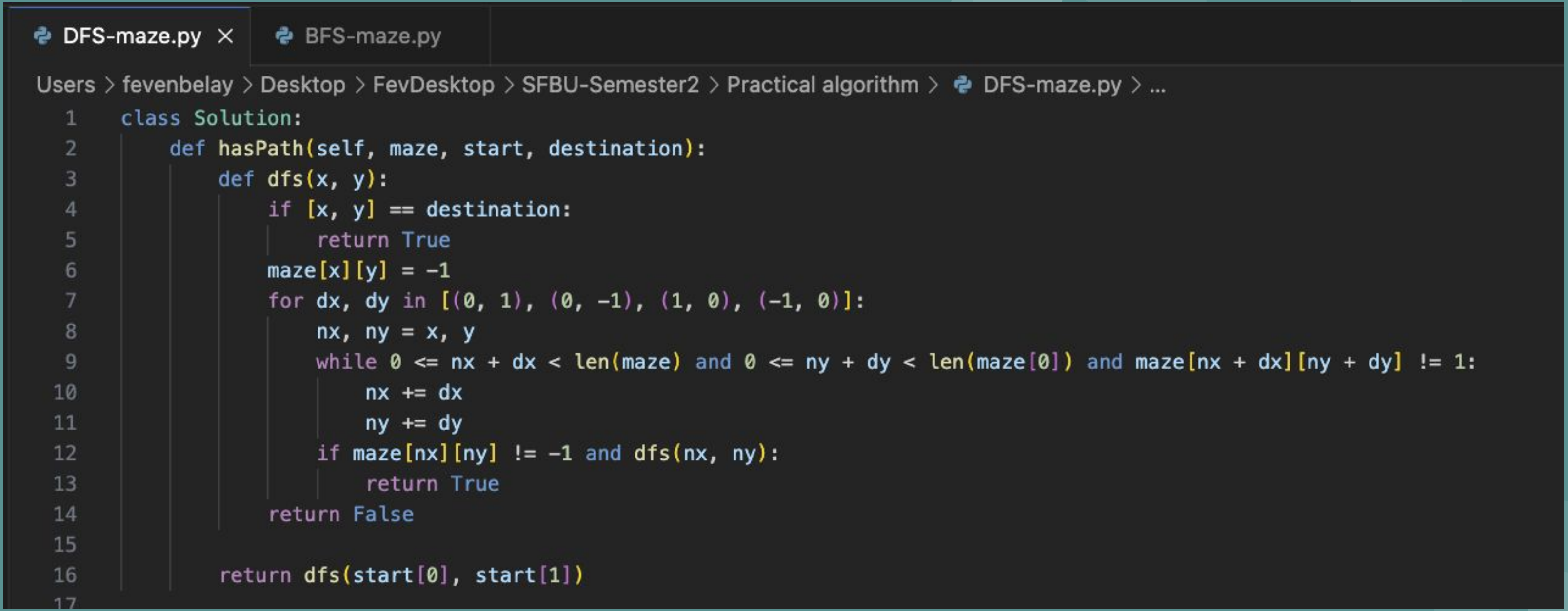
A	B	S	C	D
F	G	H	I	J
K	L	M	N	O
P	Q	R	S	T
U	V	E	X	Y

																			E
																		V	V
														~~K~~			U	U	U
													F	~~F~~		P	~~P~~	P	P
												G	G	~~G~~	Q	Q	Q	Q	Q
											L	L	L	L	L	L	L	L	L
										M	M	M	M	M	M	M	M	M	M
							C		N	N	N	N	N	N	N	N	N	N	N
						D	D	O	O	O	O	O	O	O	O	O	O	O	O
					J	J	J	~~J~~	J	J	J	J	J	J	J	J	J	J	J
		~~A~~		I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
	B	~~B~~	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S

2.2.With Wheel- Self Driving Car


Depth-First Transversal for Matrix maze + with wheel (self-driving car)
Right → Left → Top → Bottom
A B C
D E F G H
I J I K
L
M N P Q R

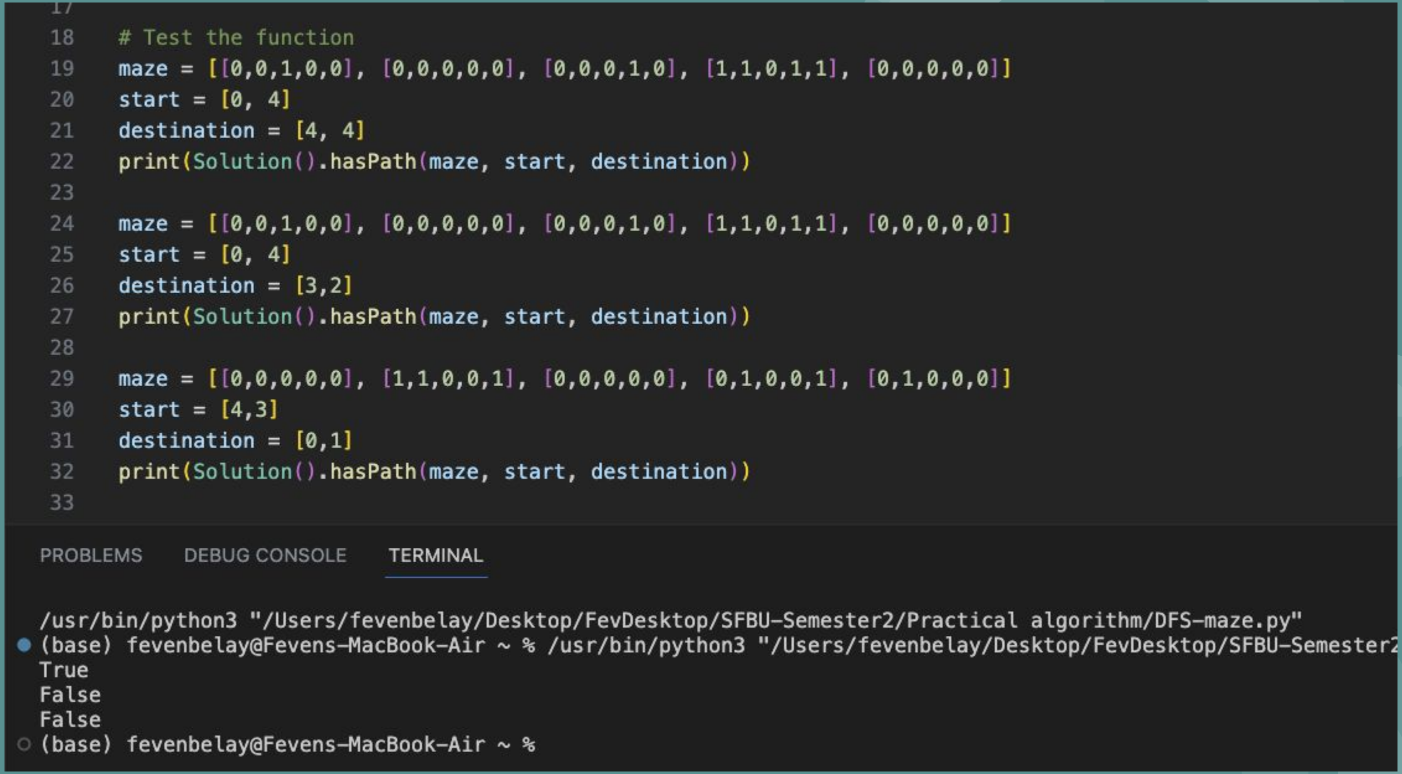
3. IMPLEMENTATION


DFS-maze.py
BFS-maze.py
Users > fevenbelay > Desktop > FevDesktop > SFBU-Semester2 > Practical algorithm > DFS-maze.py > ...

```python
class Solution:
    def hasPath(self, maze, start, destination):
        def dfs(x, y):
            if [x, y] == destination:
                return True
            maze[x][y] = -1
            for dx, dy in [(0, 1), (0, -1), (1, 0), (-1, 0)]:
                nx, ny = x, y
                while 0 <= nx + dx < len(maze) and 0 <= ny + dy < len(maze[0]) and maze[nx + dx][ny + dy] != 1:
                    nx += dx
                    ny += dy
                if maze[nx][ny] != -1 and dfs(nx, ny):
                    return True
            return False

        return dfs(start[0], start[1])
```

4. TEST



```python
# Test the function
maze = [[0,0,1,0,0], [0,0,0,0,0], [0,0,0,1,0], [1,1,0,1,1], [0,0,0,0,0]]
start = [0, 4]
destination = [4, 4]
print(Solution().hasPath(maze, start, destination))

maze = [[0,0,1,0,0], [0,0,0,0,0], [0,0,0,1,0], [1,1,0,1,1], [0,0,0,0,0]]
start = [0, 4]
destination = [3,2]
print(Solution().hasPath(maze, start, destination))

maze = [[0,0,0,0,0], [1,1,0,0,1], [0,0,0,0,0], [0,1,0,0,1], [0,1,0,0,0]]
start = [4,3]
destination = [0,1]
print(Solution().hasPath(maze, start, destination))
```

PROBLEMS DEBUG CONSOLE TERMINAL

```
/usr/bin/python3 "/Users/fevenbelay/Desktop/FevDesktop/SFBU-Semester2/Practical algorithm/DFS-maze.py"
(base) fevenbelay@Fevens-MacBook-Air ~ % /usr/bin/python3 "/Users/fevenbelay/Desktop/FevDesktop/SFBU-Semester2
True
False
False
(base) fevenbelay@Fevens-MacBook-Air ~ %
```

5. ENHANCEMENT IDEAS

1. **Iterative Deepening DFS (IDDFS):**
- Combines the space efficiency of DFS with the level-by-level exploration of Breadth-First Search (BFS).
- Useful for situations with a large or infinite state space where the depth of the solution is unknown.
2. **Parallelization:**
- Implement DFS in a parallel computing environment to explore different branches simultaneously.
- Requires careful management of shared resources and synchronization to prevent conflicts.
3. **Memory Optimization:**
- Optimize the stack usage in recursive implementations or explore alternatives to traditional stack-based approaches to reduce memory overhead.

6. CONCLUSION

- Depth-First Search is a versatile and fundamental algorithm in computer science, particularly in the domain of graph theory.
- While it is powerful in its basic form, the enhancements listed above can significantly improve its performance, efficiency, and applicability to a broad range of problems.
- By incorporating these improvements, DFS can be adapted to meet the specific needs of various applications, from solving complex puzzles to optimizing network traffic, demonstrating its continued relevance and utility in the field.