Question 7: Uninformed Search in a Grid Environment

In this task, you will build an **AI Pathfinder** that visualizes how different "blind" search algorithms explore a map. You will implement six fundamental search strategies to navigate a grid from a **Start Point (S)** to a **Target Point (T)** while avoiding static walls.

The goal is not just to find the path, but to **visualize the process**. Your GUI must show exactly how the algorithm "thinks"—which nodes it checks first, which ones it ignores, and the final path it chooses.

1. Algorithms to Implement

You must implement **ALL** of the following Uninformed Search algorithms.

1. **Breadth-First Search (BFS)**
2. **Depth-First Search (DFS)**
3. **Uniform-Cost Search (UCS)**
4. **Depth-Limited Search (DLS)**
5. **Iterative Deepening DFS (IDDFS)**
6. **Bidirectional Search**

2. Strict Movement Order

When expanding nodes (adding neighbors to your queue/stack), you must follow this specific **Clockwise** order including only the **Main Diagonal**:

1. **Up**
2. **Right**
3. **Bottom**
4. **Bottom-Right** (Diagonal)
5. **Left**
6. **Top-Left** (Diagonal)

Note: Do not check Top-Right or Bottom-Left diagonals.

4. Mandatory GUI & Visualization

A simple console output is **not acceptable**. You must implement a Graphical User Interface (GUI) using a library of your choice (e.g., Matplotlib, Pygame, Tkinter, etc.).

The GUI must visualize the search step-by-step:

1. **Frontier Nodes:** Visually distinguish nodes currently in the queue/stack waiting to be explored.
2. **Explored Nodes:** Visually mark nodes that have already been visited.
3. **Final Path:** Highlight the final successful route from Start to Target.
4. **Dynamic Updates:** The GUI should update in real-time (or with a slight delay) to show the "flow" of the search algorithm.

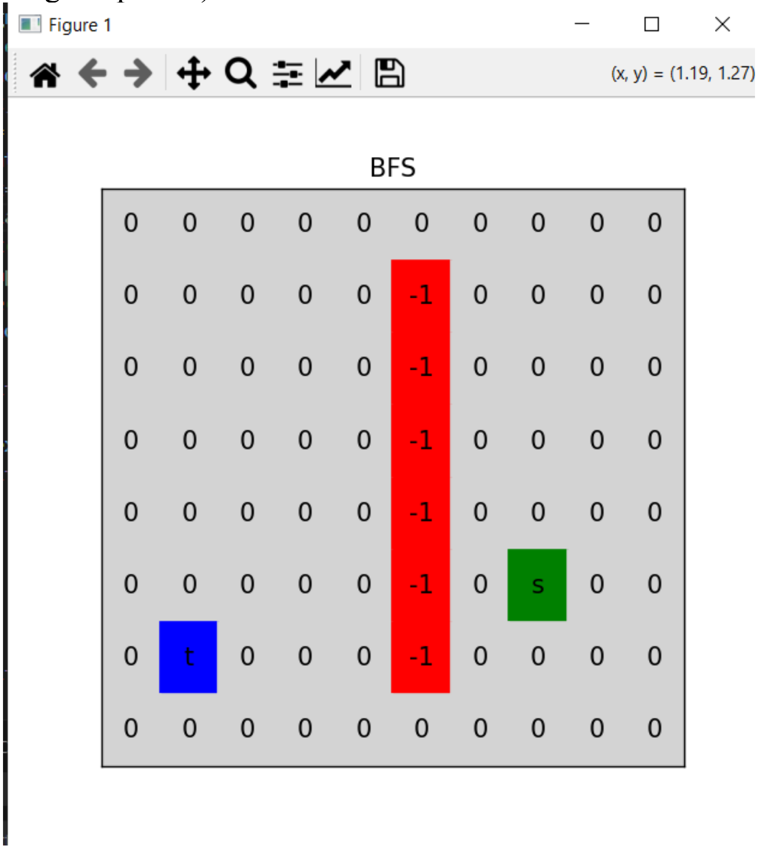
Visualization Behaviour:

- The GUI should not just jump to the result. It must animate the search **step-by-step** (e.g., with a small time delay between steps) so the user can watch the algorithm "flood" the grid.

○

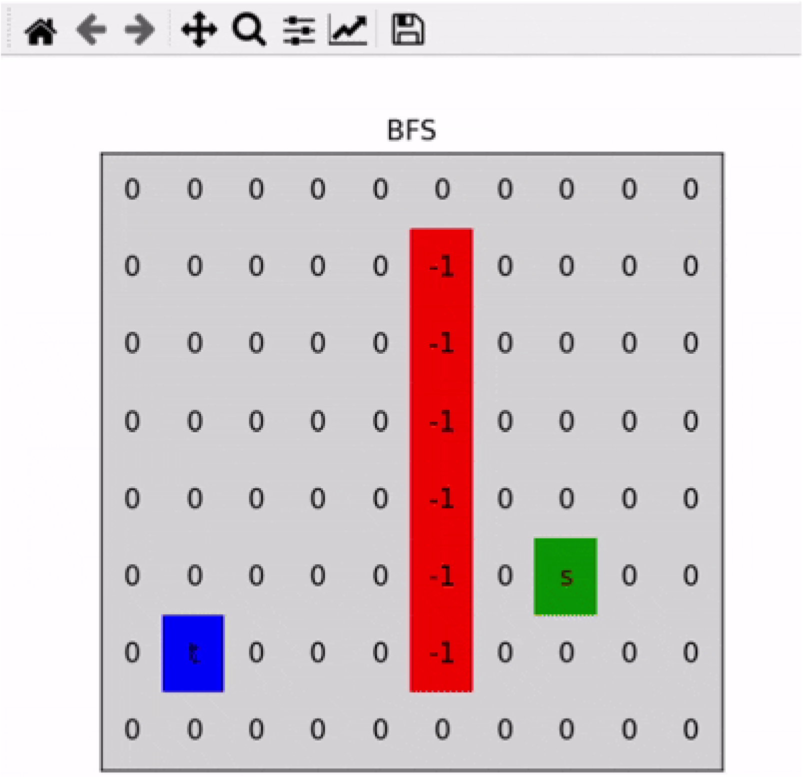
Example

Below is an example grid before and after applying the BFS algorithm:
(this GUI was made using matplotlib)



0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	-1	0	0	0	0
0	0	0	0	0	-1	0	0	0	0
0	0	0	0	0	-1	0	0	0	0
0	0	0	0	0	-1	0	0	0	0
0	0	0	0	0	-1	0	s	0	0
0	t	0	0	0	-1	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Below is a GIF attached for better understanding (click to play).



0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	-1	0	0	0	0
0	0	0	0	0	-1	0	0	0	0
0	0	0	0	0	-1	0	0	0	0
0	0	0	0	0	-1	0	0	0	0
0	0	0	0	0	-1	0	s	0	0
0	t	0	0	0	-1	0	0	0	0
0	0	0	0	0	0	0	0	0	0