**Anjana R – 21110251 Reinforcement Learning CIA-02**

1. **PROBLEM STATEMENT**

We are tasked with building a 100x100 grid environment where an agent must find an optimal path from a starting point to a target location, with obstacles scattered between them. Our objective is to formulate the environment as an MDP and use reinforcement learning to train the agent to navigate effectively. Finally, we will benchmark the RL agent against a DP method to evaluate its performance.

1. **PROBLEM STATEMENT**

**Background:** In many real-world applications, autonomous systems must navigate complex environments with limited knowledge of obstacles and possible paths. Examples include self-driving cars, delivery drones, and robotic navigation in warehouses. Our project simulates these types of environments where an agent learns optimal navigation policies through trial and error, adapting to dynamically changing paths.

**Use case Scenario:** Imagine an autonomous drone on a mission in an unknown terrain filled with potential hazards. The drone's task is to deliver supplies from a starting base to a rescue site, both randomly chosen within the grid. Using sensors, it detects obstacles but has no prior knowledge of the environment layout. This scenario allows us to create a dynamic and engaging environment to test the RL-based pathfinding agent.

1. **PROBLEM CONSTRAINTS AND ASSUMPTIONS**

* **Grid Size:** A 100x100 grid is used to represent the environment, with each cell representing a position that the drone can occupy.
* **Obstacles:** Obstacles are randomly placed and static, representing hazards or impassable terrain that the agent must avoid.
* **Start and Target Points:** Two random points are chosen as the start and goal locations.
* **Actions:** The drone has four possible actions: move up, down, left, or right.
* **Reward Structure:**
* Goal Reward: +100 for reaching the target.
* Obstacle Penalty: -10 for hitting an obstacle.
* Movement Cost: -1 for each step to encourage faster routes.
* Discount Factor: A discount factor (γ) of 0.9 is used, prioritizing immediate rewards but still valuing long-term gains.
* **Exploration vs. Exploitation:** To ensure the agent explores the grid effectively, we use an ε-greedy policy that balances exploration and exploitation.

1. **ALGORITHM DESIGN:** The algorithm is designed as an MDP problem solved with Q-learning, a popular model-free RL technique. Q-learning allows the agent to learn the optimal action-value function to maximize cumulative reward from any given state.

**Algorithm Steps:**

* Initialize the Q-Table: A 3D matrix of shape (100, 100, 4) is created, where each cell corresponds to a Q-value for each action at each grid position.
* Define Reward Function: Set rewards for goal, obstacles, and movement.
* Exploration Policy: Use an ε-greedy policy where the agent occasionally takes random actions to explore the grid.
* Q-Value Update: For each action taken, the Q-value is updated using the Bellman equation:



Termination: The learning loop terminates when a certain number of episodes is completed, or when convergence is observed.

1. **BENCHMARKING WITH DYNAMIC PROGRAMMING:**

To benchmark, we can implement a DP method such as Value Iteration. The performance of the RL algorithm can then be compared based on the time to convergence and the path optimality.