**CIA-2**

**Question:**

Create a 100x100 grid with obstacles in between 2 random points. Build an MDP based RL agent to optimise both policies and actions at every state. Benchmark DP method with other RL solutions for the same problem.

To solve this grid we use MDP (Markov Decision Process) based RL agent that starts at a random point and aims to reach a randomly chosen target point while optimizing its policy to avoid obstacles and take the shortest path, here obstacles act as barriers, and the agent cannot move through them.

State, s: Each cell in the grid represents a unique state. For a 100×100 grid, we have 10,000 possible states.

Action: The agent can move in four directions: up, down, left, and right.

Probabilities P(s’ / s ,a)**:** As this is a deterministic environment, taking an action from a state leads to a specific next state.

Rewards R(s,a,s′):

* +100 for reaching the goal.
* -1 for each step taken to encourage finding the shortest path.
* -10 for hitting an obstacle.

All the state values should be initialized to 0 , for each state we update the value function based on the bellman optimality equation ,



We repeat this until the changes in values across all states fall below a certain threshold , once the values converge we extract the optimal policy.

**Q-learning:**

This is a model-free, off-policy reinforcement learning algorithm that learns the optimal action-value function Q(s,a) by iteratively updating its estimates. In each step, the agent takes an action in the current state, observes the reward, and updates the Q-value using the highest estimated Q-value from the next state (regardless of the actual action taken). This "off-policy" approach allows Q-learning to converge to the optimal policy by always aiming to maximize long-term rewards, making it highly effective in scenarios where exploration is essential, even if some actions aren't always followed.

**SARSA:**

SARSA (State-Action-Reward-State-Action) is a model-free, on-policy reinforcement learning algorithm that updates its Q-value estimates based on the action taken in the next state. After an action is taken and a reward is observed, SARSA updates the Q-value using the value of the subsequent state-action pair actually chosen by the policy. This "on-policy" approach makes SARSA more conservative than Q-learning, as it aligns updates with the policy the agent is following, which is often helpful in environments with higher risks or noisy rewards, where more caution is beneficial.

Dynamic Programming (DP) is the better choice for this grid-world problem because it can find the best solution when we know all the details about how the environment works. It quickly calculates the optimal path to the goal by using the known rules of movement and rewards. Unlike other methods, DP doesn't need to explore or take risks because it already knows what the best actions are. This makes it stable and reliable, ensuring that the agent will always behave in a predictable way.