EXPERIMENT - 10

Aim: Solving a Markov Decision Process (MDP) using Value Iteration

1. Learning Objectives

• Define and understand MDP components — States (S), Actions (A), Transition Model (T), Reward Function (R), and Discount Factor (γ).
• Translate the GridWorld problem into a formal MDP structure.
• Implement Value Iteration and apply the Bellman Optimality Equation.
• Extract the optimal policy from the converged value function.
• Visualize and analyze the impact of γ and living penalty on agent behavior.

2. Code Snippets

value_iteration()

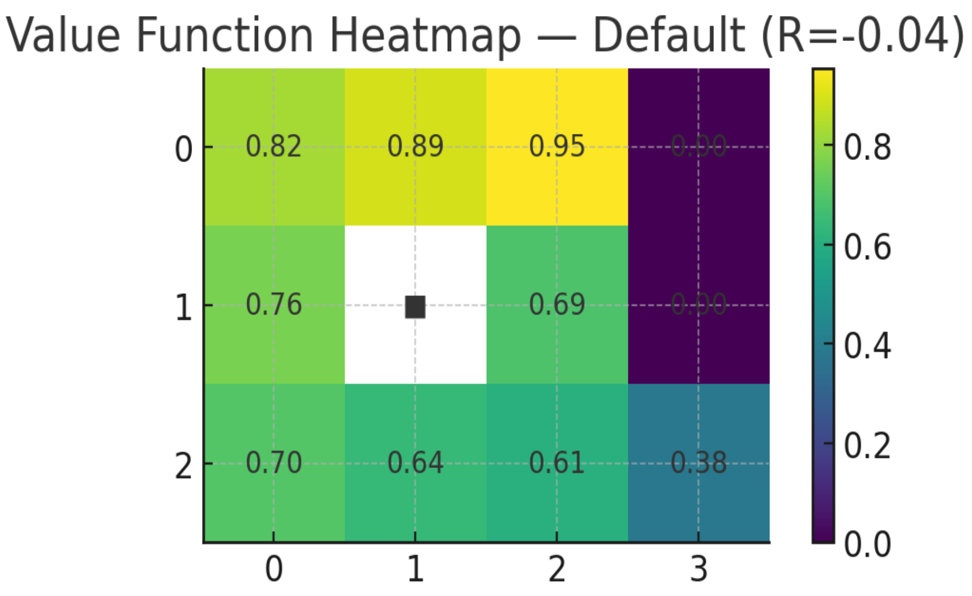
```
def value_iteration(gamma=0.99, theta=1e-4, living_penalty=-0.04): R = make_rewards(living_penalty) states
= get_all_states() V = {s: 0.0 for s in states} # Terminal states value set to 0 by convention for planning
with terminal rewards baked in R V[GOAL] = 0.0 V[PIT] = 0.0 def q_value(s, a): return sum(p * (R[ns] +
gamma * V[ns]) for p, ns in get_next_states(s, a)) while True: delta = 0.0 for s in states: if s in
TERMINALS: continue v = V[s] V[s] = max(q_value(s, a) for a in ACTIONS) delta = max(delta, abs(v - V[s]))
if delta < theta: break return V, R
```

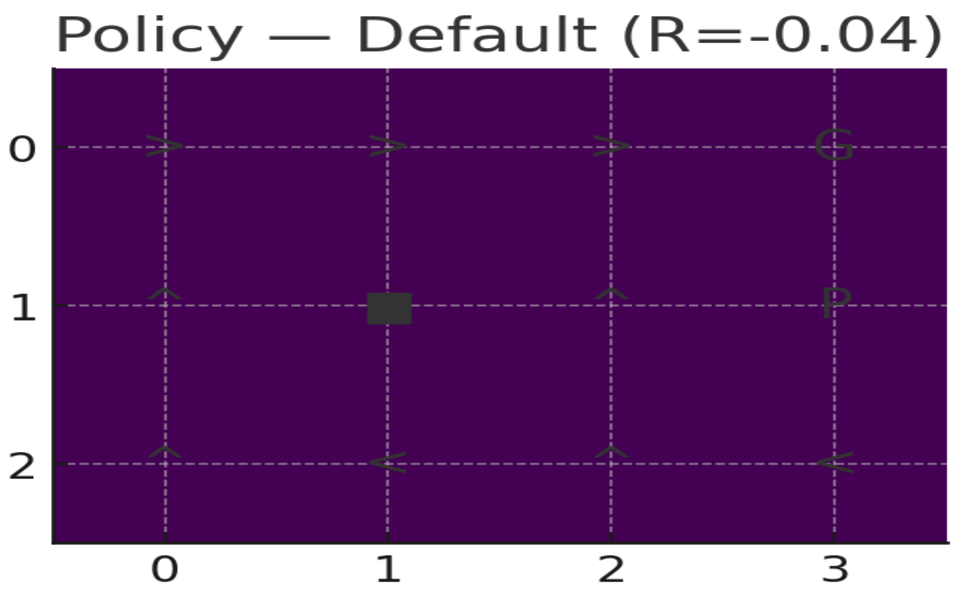
extract_policy()

```
def extract_policy(V, gamma=0.99, living_penalty=-0.04): R = make_rewards(living_penalty) policy = {}
states = get_all_states() def q_value(s, a): return sum(p * (R[ns] + gamma * V[ns]) for p, ns in
get_next_states(s, a)) for s in states: if s in TERMINALS: policy[s] = None else: qs = [(a, q_value(s, a))
for a in ACTIONS] best_a = max(qs, key=lambda x: x[1])[0] policy[s] = best_a return policy
```

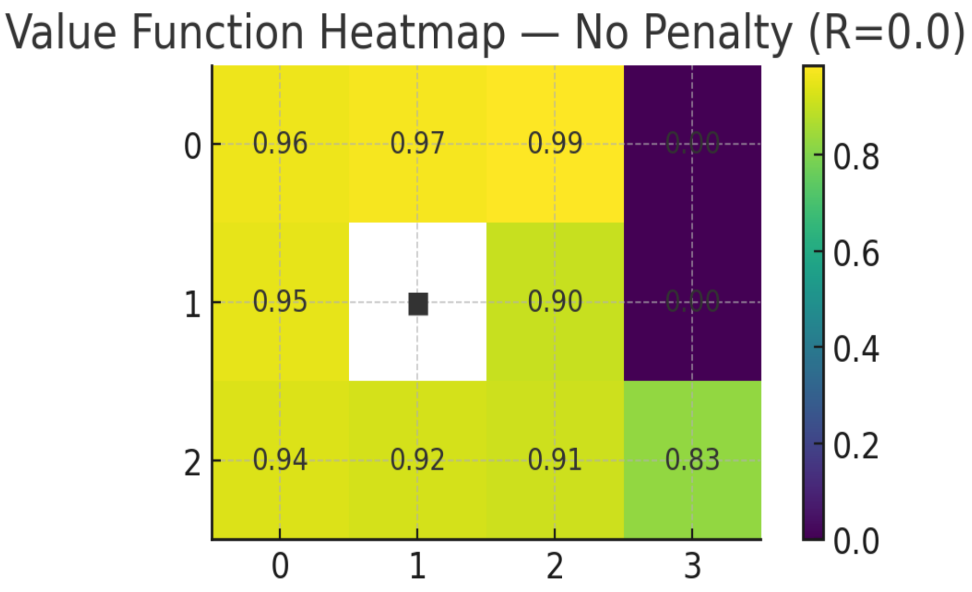
3. Results and Visualizations

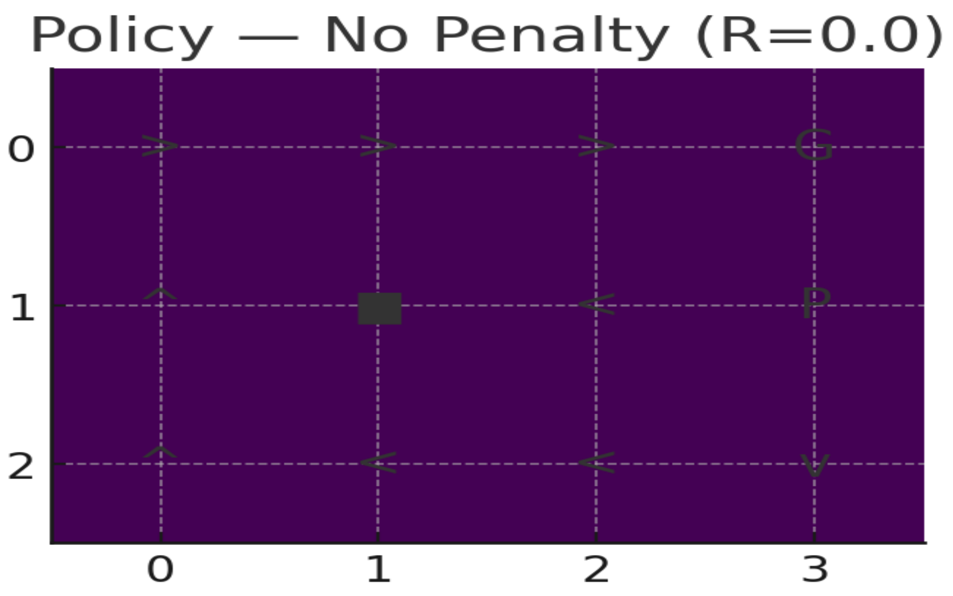
(a) Default Case: Living Penalty R = -0.04



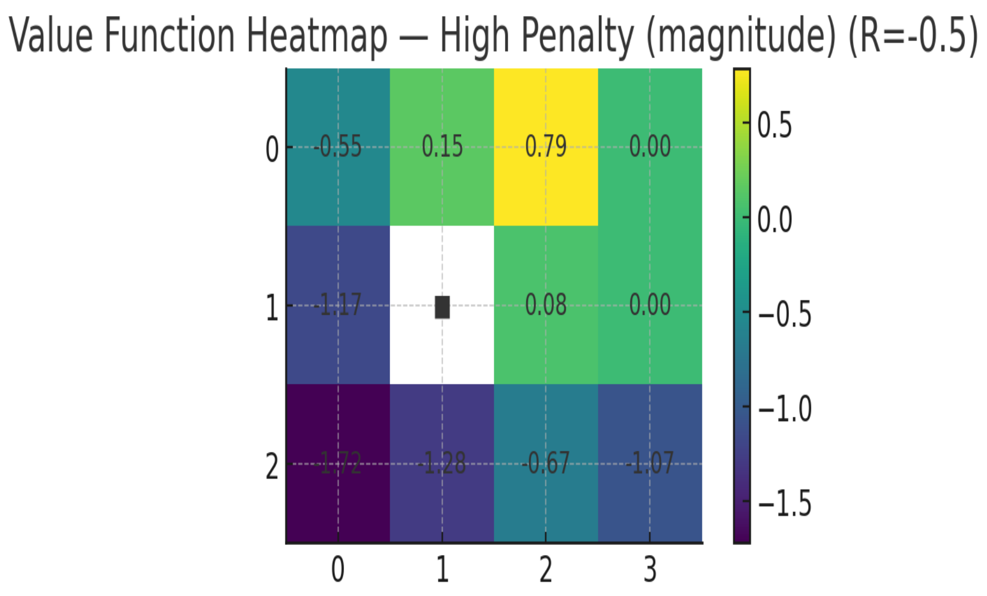


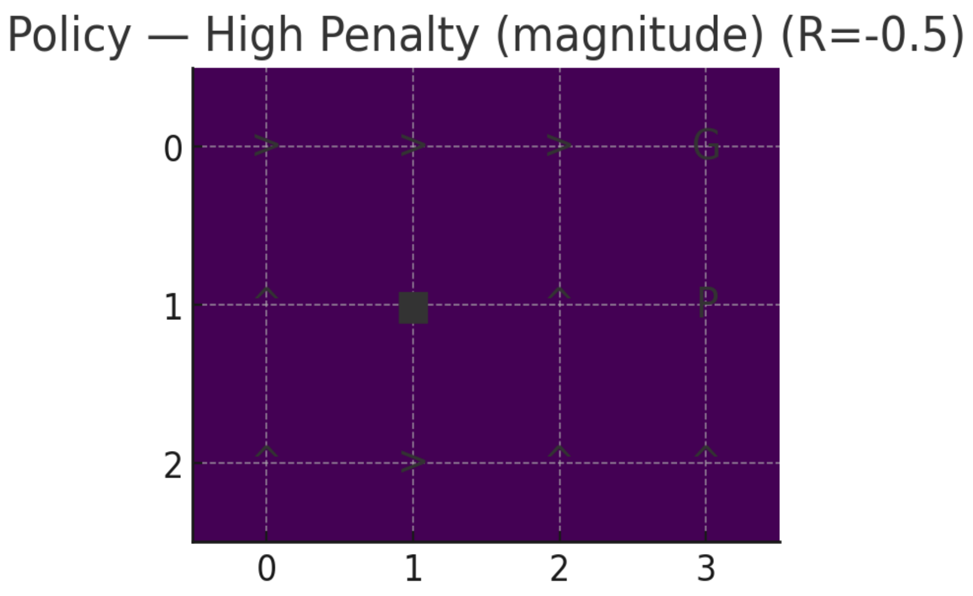
(b) No Penalty Case: R = 0.0





(c) High Penalty Case: R = -0.5





4. Analysis and Discussion

Q1. Does the policy make sense? Does it correctly avoid the pit and find the goal?
Yes. In the default case (R = -0.04), the policy learned through Value Iteration guides the agent toward the goal at (0,3) while avoiding the pit at (1,3) and the wall at (1,1). The resulting value function increases smoothly toward the goal, confirming correct convergence.

Q2. When the living penalty R(s) = 0.0, does the policy change? Why or why not?
When the living penalty is removed, the agent no longer incurs a cost for taking steps. As a result, the agent may be less motivated to minimize the number of moves. The policy often remains similar to the default but with less urgency to reach the goal.

Q3. When the living penalty R(s) = -0.5 (a high penalty), what happens to the policy and why?
With a high negative penalty, every step is costly. The agent becomes highly focused on reaching the goal in the fewest moves. The overall values drop due to the large per-step penalty, but the path becomes more direct and aggressive.

5. Conclusion

This experiment demonstrates solving a Markov Decision Process using the Value Iteration algorithm. By repeatedly applying the Bellman Optimality Equation, the algorithm converges to the optimal state values and policy. Changes in the living penalty directly influence the agent’s preference for shorter or safer paths, highlighting how reward structure affects optimal decision-making.