**Design Defense Document: Treasure Hunt AI Agent**

**Introduction** The Treasure Hunt game requires an AI-controlled pirate to find the treasure before the human player using a deep Q-learning algorithm. This document outlines the problem, solution approach, and implementation details for the intelligent agent that navigates the maze environment efficiently.

### ****Human vs. Machine Problem-Solving****

A human would approach the problem by visually analyzing the maze, identifying possible paths, avoiding obstacles, and planning a route to the goal. Humans rely on reasoning and memory to recall previous failed attempts and adjust their strategy accordingly.

The AI agent, however, learns through reinforcement. It initially explores different paths randomly, collecting information about rewards and penalties. Over time, using deep Q-learning, the agent refines its pathfinding by associating actions with the highest future rewards, mimicking a learned behavior without explicit planning.

The major difference between human and AI problem-solving is adaptability and experience. A human can generalize from past experiences and use intuition, whereas the AI must explicitly learn from trial and error. However, once trained, the AI can outperform humans in speed and precision when solving similar mazes.

### ****Purpose of the Intelligent Agent in Pathfinding****

The goal of the pirate agent is to navigate the maze while optimizing for the best possible path to the treasure. The AI is designed to efficiently make decisions using reinforcement learning principles.

* **Exploration vs. Exploitation:** Exploration involves trying unknown paths to gather more information, while exploitation uses learned knowledge to choose the best-known path. A balance is required for optimal performance.
* **Balancing Strategy:** Initially, high exploration ensures broad learning, while gradual reduction in exploration (epsilon decay) allows the agent to commit to the best-discovered strategies.
* **Reinforcement Learning’s Role:** The agent uses previous experiences to update Q-values, ensuring it improves its decision-making over time.

The AI continuously refines its strategy by using a Q-learning model, which updates the agent’s understanding of which paths lead to success and which should be avoided. Over multiple iterations, the agent learns to avoid obstacles and select the fastest route to the treasure.

### ****Implementation of Deep Q-Learning****

Deep Q-learning enables the pirate agent to optimize its movement over time by storing previous experiences and improving decision-making based on trial and error.

#### ****1. State Representation:****

The maze is represented as an **8x8 matrix** where each position is a state. The agent perceives the environment through state transitions, moving from one position to another.

#### ****2. Action Space:****

The pirate can move in four directions: **up, down, left, and right**. The agent must decide which action to take based on the predicted future rewards.

#### ****3. Rewards and Penalties:****

* **+1** for reaching the treasure.
* **-0.75** for hitting an obstacle.
* **-0.8** for moving out of bounds.
* **-0.04** for each valid move (to discourage unnecessary wandering).

#### ****4. Neural Network Implementation:****

* A **multi-layer neural network** is used to approximate Q-values.
* The model consists of **two hidden layers** with ReLU activation and an output layer with linear activation.
* The **Adam optimizer** is used for adjusting weights based on learning from previous episodes.

#### ****5. Experience Replay:****

* The AI stores past experiences (state, action, reward, next state) in memory.
* During training, a **random sample of experiences** is used to prevent overfitting and improve generalization.
* This allows the model to learn from past mistakes efficiently.

#### ****6. Training Strategy:****

* The model is trained over **1,000 episodes**, with **epsilon decay** ensuring a smooth transition from exploration to exploitation.
* The training loop updates Q-values based on observed rewards and future predictions.
* After sufficient training, the AI consistently finds the treasure with minimal steps.

### ****Evaluation of AI Performance and Optimization Techniques****

The AI’s performance is measured by its ability to reach the treasure in the shortest number of steps and with the least number of penalties. The efficiency of training is determined by:

* The **rate of convergence** (how quickly the agent improves).
* The **win rate** (how often the pirate finds the treasure).
* The **average number of steps** taken to complete the maze.

To optimize performance, several **fine-tuning techniques** can be applied:

1. **Adjusting Hyperparameters:**

* Increasing the number of neurons in hidden layers may improve learning.
* Modifying the learning rate can impact the speed of convergence.

1. **Modifying Reward Function:**

* More granular rewards (e.g., increasing penalties for unnecessary moves) can enhance decision-making.

1. **Using Double Q-Learning:**

* Helps to reduce overestimation errors in Q-values, leading to better training stability.

### ****Conclusion****

The deep Q-learning algorithm effectively trains the pirate agent to find the treasure with an optimized strategy. By leveraging reinforcement learning principles, the agent can make increasingly intelligent decisions over time. The balance between exploration and exploitation ensures that the agent generalizes well across different maze configurations.

The combination of **experience replay, epsilon-greedy strategies, and a deep neural network** enables the pirate to navigate efficiently and reach the treasure with a high success rate. Future improvements could include implementing **advanced reinforcement learning techniques** like **Double DQN** or **Prioritized Experience Replay** to further refine the agent’s decision-making capabilities.

This project demonstrates how **AI can be used for complex decision-making tasks** and serves as a fundamental introduction to **deep reinforcement learning** in a game environment.