**Assignment Report: Dynamic Goal-Based Agent for Warehouse Logistics Optimization**

**Introduction**

This report provides a solution for Question 1 of the given assignment, which involves representing a warehouse as an N×M matrix. The matrix includes randomly placed packages, drop-off points, and obstacles. The environment is then displayed to visualize the initial configuration.

**Problem Statement**

A robotic agent operates within an N×M grid environment representing a warehouse. The agent starts at a predefined location and must deliver packages to their respective drop-off points while avoiding obstacles. The matrix representation must:

* Clearly depict packages, drop-off locations, and obstacles.
* Ensure random placement of elements with no overlaps.
* Be visually displayed for validation.

**Implementation**

**1. Environment Setup**

The warehouse is represented using a 2D matrix where:

* 0 represents empty spaces.
* 1 represents package locations.
* 2 represents drop-off locations.
* 3 represents obstacles.

**a) Generating the Matrix**

A function generateMatrix(row, col) initializes a zero matrix of dimensions N×M.

**b) Random Placement of Elements**

The function generateRandomIndices(row, col, n) selects n random positions within the matrix for packages, drop-off locations, and obstacles, ensuring no overlapping positions.

**c) Placing Symbols**

The function placeSymbols(s, indices, warehouse) assigns values (1, 2, or 3) to the randomly chosen locations within the matrix.

**d) Displaying the Matrix**

Finally, the function displayEnvironment(warehouse) prints the generated matrix, showing the warehouse layout.

**2. Code Implementation**

The EnvironmentSetup class is responsible for generating and displaying the warehouse matrix. The following steps outline the execution:

1. Define the warehouse dimensions N=4, M=6.
2. Randomly generate indices for:
   * Packages (num\_package=6)
   * Drop locations (num\_package=6)
   * Obstacles (num\_obstacle=10)
3. Place the corresponding symbols in the matrix.
4. Display the warehouse configuration.

The implementation successfully generates and displays the warehouse matrix with randomly placed packages, drop locations, and obstacles. This representation will serve as the foundation for further goal-based agent implementation in subsequent questions.

**Assumptions:**

1. The agent is not aware of the obstacles beforehand, when moving if any obstacle is found and there is no alternative path, then penalty will incur and will be adjusted in the final score
2. The Packages are delivered on their specific drop location

**Outputs:**

The Warehouse with Package, Drop & obstacles

A screenshot of a computer

AI-generated content may be incorrect.

The final score, path cost

A screen shot of a computer screen

AI-generated content may be incorrect.

**QB) Report on Common Meetup Search Problem**

**Assumptions:** I used the geojson dataset of taluka level granularity, I choose “Haryana”, “Punjab” & “Chandigarh” states taluka’s. Friend 1 I’ve assumed is in city “Sirsa” (Haryana) and Friend 2 is in city “Ludhiana” (Punjab).

* Used GeoPandas library to read the Geojson dataset file and perform operations on the dataset
* Heuristic function considered is Straight line distance between city 1 & city 2 with corresponding to all the relative nodes.

**Q1.**  Have plotted all the map using the matplotlib library and used Networkx library for creating the graph plotting.

A map of a country

AI-generated content may be incorrect.

**Q2.** Solved the optimal meetup place using GBFS and A\* search algorithm

**a. Greedy Best First Search (GBFS)**

* Expands the node with the lowest heuristic value h(i,j)h(i, j).
* Does not consider movement cost, leading to suboptimal paths in some cases.
* May result in higher node exploration and backtracking.
* GBFS using SLD as heuristicsA black screen with white text

  AI-generated content may be incorrect.
* GBFS using road distance as heuristics valueA black screen with white text

  AI-generated content may be incorrect.

**b. A\* Search Algorithm**

* Expands the node with the lowest f(n)=g(n)+h(n)f(n) = g(n) + h(n).
* Guarantees an optimal solution.
* Balances heuristic estimation with actual movement cost.

**3. Variation in Heuristic Function**

The heuristic function is changed from straight-line distance d(i,j)d(i, j) to:

1. **Used Road Distance** (real-world road distance between cities)

**Comparison of Heuristics:**

* **Straight-line Distance:** Fast but unrealistic, may not reflect actual travel conditions.
* **Road Distance:** More accurate but increases computational cost.
* **Train Distance:** Limited to certain routes, useful for real-world scenarios.

**Search Cost Analysis with New Heuristics:**

**Conclusion:** A\* search provides an optimal and efficient way to find the common meetup point. Greedy Best First Search may lead to suboptimal paths due to its focus on heuristic values. Realistic heuristics like road and train distances improve search accuracy but increase computational effort.