Stark Industries Pvt. Ltd.

PROBLEM STATEMENT – PS1: BLIND FLIGHT – STARK LOGISTICS RESCUE

1. Background

A localized EMP blast has partially crippled Stark Industries' autonomous logistics network.

Visual navigation, path planning, and energy-aware routing subsystems are offline.

Shipments carrying sensitive Stark tech are stranded across three very different environments:

- A heavily shielded Stark R&D lab
- A dense forest perimeter
- A wide-open desert testing range

The high-level control centre is still operational, but it can only transmit pre-computed path

commands, not continuous joystick control. You are being brought in as an external systems engineer to rebuild the perception + planning stack using the limited sensor data that is still available.

This problem statement defines PS1, which will run as a 15-day Kaggle-based hackathon with a live leaderboard.

2. Objective

Given:

- A top-down rendered map image of size 20 × 20 tiles
- A corresponding "velocity boost" field for the same grid

you must:

1. Infer a valid route from the START tile to the GOAL tile while avoiding walls.
2. Exploit the boost field to minimise travel cost rather than just geometric distance.
3. Output the route as a sequence of moves in the format: l, r, u, d (left, right, up, down).

Your final goal is to design a system that can reliably:

- Decode the map image into tile classes.
- Understand terrain-specific movement cost.
- Use the velocity boosts to produce a near-optimal path cost.

3. Map Representation

Each map is a 20 × 20 discrete grid. Every cell belongs to exactly one of the following classes:

Class 0 – Walkable Path

- Normal floor / traversable terrain.
- Base cost depends on terrain type (lab / forest / desert).

Class 1 – Wall

- Impassable obstacle.
- The agent cannot enter these cells under any circumstance.

Class 2 – Hazard

- Traversable but expensive.
- Represents things like sticky liquid, quicksand, puddles, etc.
- Higher base cost than normal walkable path.

Class 3 – START

- Starting position of the Stark shipment vehicle / drone.
- Exactly one per map.

Class 4 – GOAL

- Destination tile (the secure vault / extraction pad / safe depot).
- Exactly one per map.

The underlying class grid for each test map is NOT provided to participants. It exists only in the hidden private files for evaluation.

4. Terrains

There are three distinct terrain themes. Each has its own look and its own base movement costs.

4.1 Stark Lab

• Visual: metal tiles, conduits, arc-reactor leaks, vaults, drones.

• Interpretation: tightly constrained industrial interior with dangerous "plasma" pits and sticky spills.

• Base costs:

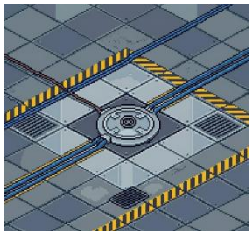
– Class 0 (floor): 1.0

– Class 2 (hazard): 3.0

– Class 3 (start): 1.0

– Class 4 (goal): 2.0

– Class 1 (wall): inf



floor

hazard

goal

Start

wall(i)

wall(ii)

4. 2 Forest Perimeter

• Visual: soil, grass, trees, puddles, roots, foliage.

• Interpretation: natural terrain with uneven cost due to vegetation and waterlogging.

• Base costs:

– Class 0 (ground): 1.5

– Class 2 (hazard): 2.8

– Class 3 (start): 1.5

– Class 4 (goal): 2.5

ground

hazard

goal

Start

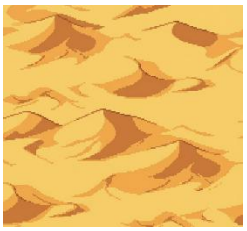
wall(i)

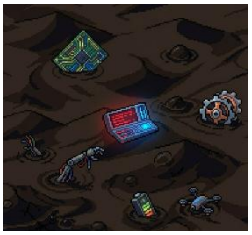
4.3 Desert Test Range

- Visual: sand, rocks, cactus / obstacles, dunes, quicksand patches.

- Interpretation: relatively open, but with a few very punishing hazard tiles.

- Base costs:
  - Class 0 (sand): 1.2
  - Class 2 (hazard): 3.7
  - Class 3 (start): 1.2
  - Class 4 (goal): 2.2


sand


hazard


goal


Start


wall(i)


wall(ii)

NOTE: The exact visuals in the sprite packs are for you to learn from. You are expected to train models that can recognise these patterns and generalise to unseen maps.

5. Velocity Boost Field

For every test image, a corresponding JSON file is provided in test/velocities/ with the same base name as the image (e.g., 0001.json for 0001.png).

This JSON contains:

- "image_id" : string
- "grid_size": integer (always 20)
- "boost" : 20 × 20 matrix of floats in the open interval (−0.99, 0.99)

Interpretation:

- Positive boost → locally favourable tile (like riding a wind stream or conveyor).
- Negative boost → locally unfavourable tile (like moving against a current).
- Boost values are terrain-agnostic; they are an extra layer on top of the terrain-dependent base costs.

Movement cost model (this is what the leaderboard uses):

base_cost(i, j) = BASE_COST[terrain][class(i, j)]

step_cost(i, j) = base_cost(i, j) − boost(i, j)

You pay step_cost(i, j) when you MOVE INTO cell (i, j).

Total path cost = sum of step_cost along every move from START to GOAL.

The map guarantees:

- All walkable classes have base_cost ≥ 1.0
- boost(i, j) ∈ (−0.99, 0.99)

→ So every step_cost is strictly positive and there are no negative-cost cycles.

6. Data Provided

During the hackathon you will get a small training set plus the full test set for evaluation:

6.1 Training Set

Location (relative to Kaggle notebook root):

- train/images/*.png – 20 × 20 tile maps (rendered images).
- train/labels/*.json – contains the true class grid "grid" for each map.
- train/velocities/*.json – contains the boost field for each training map.

Notes:

- Training size is intentionally limited (≈20 maps) due to Mark 85 shutting down mid-stream.
- You are strongly encouraged to perform data augmentation, synthetic map generation, and/or self-supervised techniques to improve performance.

6.2 Test Set

- test/images/*.png – maps for leaderboard evaluation.
- test/velocities/*.json – boost fields for these maps.

7. Required Outputs and Submission Format

For every test image, your system must output a path as a sequence of moves using the characters:

- 'l' → move left (column − 1)
- 'r' → move right (column + 1)
- 'u' → move up (row − 1)
- 'd' → move down (row + 1)

Moves must be valid 4-connected steps on the grid.

Submission file (CSV):

- Filename (recommended): submission.csv
- Columns:

  image_id,path

  0001,rruulldd...

  0002,ddrruul...

  ...

Constraints:

- At most one row per image_id.
- 'path' may be empty, but that will be treated as an invalid path (very low score).

Submission / sanity checks

- Run the provided sample_code once end-to-end:
  - It will generate submission_baseline.csv from a very naive model.
  - Upload it to Kaggle to confirm that your environment and format are correct.
- After that, replace the perception and planner modules with your improved versions.

10. Constraints and Notes

- The hackathon lasts 15 days.
- The Kaggle competition is strictly leaderboard-based during the event.
- Maximum 5 submissions per day.
- GPU usage on Kaggle is strongly recommended for training.

11. Final Deliverables

At the end of the hackathon, each participant must submit:

A. Kaggle Submission

- Your best-performing CSV file on the final test set.
- This decides your leaderboard position.

B. Code Repository (Git)

- Public or private Git repository link (GitHub / GitLab / etc.).
- Must contain:
  - Training and inference code.
  - Instructions to reproduce your results.
  - Clear description of dependencies and hardware requirements.

C. Short Technical Report (maximum 3–4 pages)

Mandatory structure:

NAME:

ENROLMENT NUMBER:

BRANCH:

YEAR:

<BODY>

ALL THE BEST!