Technical Explanation – Frontier Selection Strategy

This document explains the frontier selection strategy implemented in the function `determine_frontier_path()` for the JSIC WinterHack 2026 Candidate Selection Challenge. The goal of this algorithm is to guide the autonomous robot efficiently through an unknown maze
environment while maintaining progress toward the final goal.

1. Overview

Frontier-based exploration identifies boundary cells between known free space and unexplored areas.
The proposed strategy evaluates each detected frontier according to a multi-criteria scoring function
that balances exploration efficiency, goal-directed behaviour, and motion stability.
The frontier with the lowest score is selected as the next navigation target.

2. Scoring Criteria

Each candidate frontier cell is evaluated using four main metrics derived from the simulation state:

1. **Path Length (w_len):** The total distance of the A* path from the robot's current cell to the frontier,
   representing travel cost. Shorter paths are preferred.
2. **Goal Proximity (w_goal):** A reward inversely proportional to the frontier's distance from the final goal.
   Frontiers nearer to the global goal receive higher priority, ensuring goal-directed exploration.
3. **Heading Alignment (w_head):** The angular difference between the robot's current heading and the
   direction toward the frontier. Smaller differences reduce unnecessary turning and improve efficiency.
4. **BFS/Reachability Cost (w_bfs):** The breadth-first-search distance or cell-based proximity metric
   encouraging selection of accessible frontiers within the currently mapped space.

The combined score is computed as:

$$score = w_{len} * path_{len} + w_{head} * heading_{diff} + w_{bfs} * bfs_{dist} - w_{goal} * goal_{bonus}$$

Here, `goal_bonus = 1 / (1 + d_goal)` acts as an attraction term that rewards frontiers closer to the final goal.
The weights dynamically adapt depending on the robot's distance to the goal: when far, exploration is favoured;
when near, goal convergence is prioritised.

3. Decision Logic

1. Detect all reachable frontiers using `detect_frontiers(state)`.
2. For each frontier, attempt to plan a path using the current occupancy grid map via `plan_unknown_world()`.
3. Compute the metrics and total score for reachable candidates.
4. Select the frontier with the minimum score as `state.frontier_goal`.
5. If no frontiers are reachable, fall back to the closest one to the goal or the final goal itself.

4. Advantages

- **Goal-directed:** Always biases exploration toward the maze exit.
- **Efficient:** Prefers reachable, shorter, and better-aligned frontiers.
- **Robust:** Automatically falls back when no frontiers are reachable.
- **Adaptive:** Dynamically adjusts exploration vs. exploitation balance based on goal proximity.

5. Summary

This frontier selection strategy achieves a balance between systematic exploration and efficient goal-seeking.
It ensures stable and intelligent motion planning suitable for real-time SLAM environments, demonstrating
effective autonomous decision-making under partial observability.