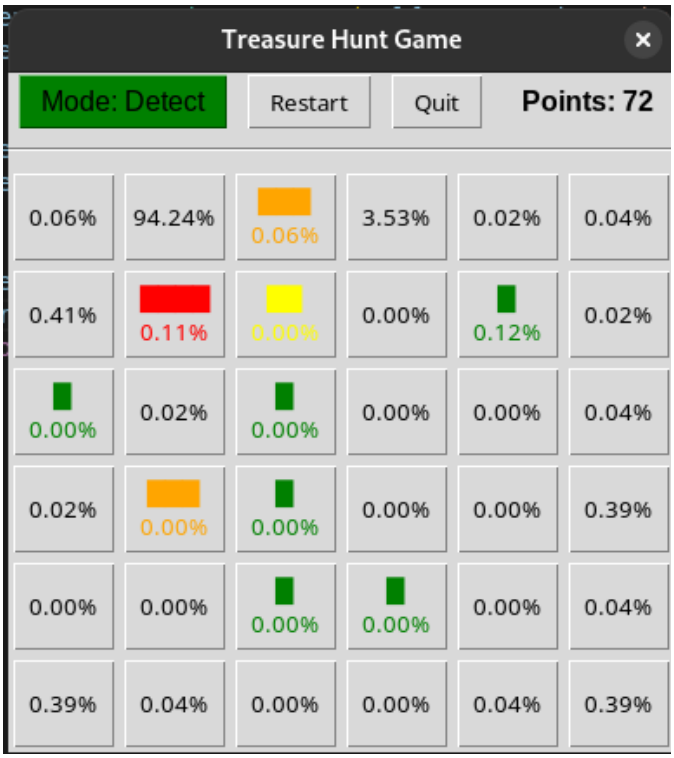
Project Handout: Treasure Hunt Game
INF13262M

aanjos@uevora.pt

Project Overview

The objective of this project is to design and implement an interactive game that uses probabilistic detection mechanisms on an $m \times n$ grid. Players locate a hidden treasure by strategically using detection moves, which incur a point penalty. Once confident, the player makes a single attempt to dig for the treasure. Points are deducted for each detection, and the goal is to find the treasure while conserving as many points as possible.

The game can be implemented using either a **graphical interface** (mouse clicks) or a **text-based interface** (keyboard commands). Here is a possible graphical interface (GUI):



Each time a player detects a cell, the game should display the signal strength emitted by the sensor, update the probabilities for the treasure's location across all cells, and deduct points from the player's score. The player can then decide to either detect another cell or (toggle the mode to) dig for the treasure if they feel confident about its location.

You are free to design the game interface (whether graphical or text-based) as you will, as long as it follows the rules and gameplay described in this document.

Note that, in the example above, the signal strength (colored bar) reported by the detector isn't always the most reliable indicator of the treasure's location. Instead, the probability displayed on each cell provides the best guidance for finding the treasure.

Game Rules and Mechanics

1. **Game Modes**:
   - **Detect Mode**: Players may "detect" the treasure by probing cells within the grid. Each detection reduces the player's score.
   - **Dig Mode**: Players have a single chance to "dig". If they dig in the correct cell, they win; otherwise, the game ends with a loss.
2. **Scoring System**:
   - The game begins with a score of 100. The presented score should always be integer.
   - Each detection decreases the score proportionally to the number of cells, such that detecting every cell would reduce the score to 0.
   - The objective is to minimize detections before making a successful dig.
3. **Detector Mechanics**: The detector provides a "signal strength" indicating the proximity to the treasure. The manufacturer has provided the following tables, showing the probability of each signal strength at different distances d between the detected cell and the treasure.

$d = 0$ (same cell as treasure):

Signal Strength	Probability
++++	80%
+++	10%
++	7%
+	3%

$d = 1$ (one cell away from treasure):

Signal Strength	Probability
++++	8%
+++	80%
++	8%
+	4%

$d = 2$ (two cells away from treasure):

Signal Strength	Probability
++++	4%
+++	8%
++	80%
+	8%

$d \geq 3$ (three or more cells away):

Signal Strength	Probability
++++	3%
+++	7%
++	10%
+	80%

Project Components and Evaluation Criteria

1. **Source Code** (20% of final grade)
   - Implement the game in Python.
   - Ensure the game can be played in either graphical (e.g., using `tkinter`) or text-based mode.
   - Code should be clear, documented, and follow game rules. Don't use cryptic variable and function names, and include comments where necessary.
   - Grid size should be parametrizable (either through user input or in the code).
2. **Report** (30% of final grade, max 4 pages in total)
   - **Project Summary**: Briefly explain the game's objectives and rules.
   - **Design & Architecture**: Describe the Bayesian network model used, how Conditional Probability Tables are derived from detector specifications, and how the probability of the treasure being at each cell is computed. This is the most important section of the report.
   - **Challenges and Reflection**: Describe challenges encountered, solutions, limitations of your project. Provide self-assessment (0–20), with rationale, for the report and code, individually. If working in group, specify each member's contribution (e.g., Bob 35%; Alice 65%).
3. **Oral Defense** (50% of final grade)

- **Technical Explanation**: Explain key code components and detection logic.
- **Q&A**: Answer questions about code, probabilities' computation, and design choices.

Scoring Breakdown

- **Source Code (20%)**
  - Full marks for complete, functional code.
  - Partial credit for incomplete functionality.
- **Report (30%)**
  - Full marks for clarity and completeness.
  - Partial marks for underdeveloped sections.
  - **Important:** A 25% penalty applies to the report if the project is not implemented in code.
- **Oral Defense (50%)**
  - Full marks for clear presentation and in-depth explanations.
  - Partial credit for incomplete explanations or weak responses.

  Please keep in mind that if you are working in a group, the grade awarded will be for the group as a whole, not for individual members. Be thoughtful in selecting your group members, as the performance of each member will **impact the entire group's grade**.

Final Remarks

While the game concept is straightforward, constructing the Conditional Probability Tables (CPTs) for each detector location variable based on the manufacturer's specifications can be tricky. To simplify the process, start by drawing a small 4×4 grid and the corresponding Bayesian network on paper. Manually work out the CPTs for each detector location, simulate a few detection scenarios, and calculate the probability of each cell containing the treasure after each detection. Doing this by hand will make the coding process significantly easier. In other words, **understand the probabilistic reasoning before coding**.

Although implementing a graphical interface is optional, it can greatly enhance the player experience. If you decide to include a GUI, we recommend using Python's `tkinter` library. Keep in mind, the focus of the project is not on the GUI itself, but it surely makes it easier see the game in action.

In your report, clearly describe the Bayesian network mechanics conceptually and explain how they connect to your code. Including diagrams, formulas, and examples will make your explanations clearer.

We will be evaluating your understanding of the probabilistic reasoning behind the game, your approach to modeling it, and your implementation in code.

Groups are limited to 2 students. If you choose to work in a group, please email the instructor with the names of the members by December 30, 2024. No groups will be accepted after this date.

Academic Honesty

All work should be original. Code or text from outside sources must be cited. Plagiarism or cheating will result in disciplinary action as per university policy.

Submission Guidelines

- **Deadline**: January 10, 2025, 23:59 on Moodle. Late submissions will not be accepted.
- **Files**: Submit a `.zip` containing a `.py` file and a PDF report.
- **Oral Defense Dates**: January 13–14, 2025 (schedule to be confirmed).