CS5011: A2 - Logic - Tornado Sweeper

Assignment: A2 - Assignment 2

Deadline: 1st November 2019

Weighting: 25% of module mark

Please note that MMS is the definitive source for deadline and credit details. You are expected to have read and understood all the information in this specification and any accompanying documents at least a week before the deadline. You must contact the lecturer regarding any queries well in advance of the deadline.

---

1 Objective

This practical aims to implement an agent that is able to play and solve a logic game.

2 Competencies

- Design and implement a logical agent that is able to perceive certain facts about the world it occupies, and act accordingly
- Use of logical reasoning to solve a puzzle

3 Practical Requirements

3.1 Introduction

The Tornado sweeper game is inspired by the well-known Mineswepeer computer game[1]. A Tornado sweeper world is a grid in the shape of a rhombus with N cells per side, where each cell is of hexagonal shape. T tornadoes are scattered among the cells. The rules of the Tornado sweeper game are similar to those of Minesweeper, played on a hexagonal board[2], but the Tornado worlds are a small subset of those. A logical agent playing the Tornado Sweeper game aims to uncover all cells on the board but those containing a Tornado. If the agent probes a cell that contains a Tornado, the game is over. Otherwise, a clue appears on the cell indicating the number of Tornadoes in the 6 adjacent neighbours of the probed cell. In this practical, your aim is to develop logic-based strategies for an agent to solve the tornado sweeper game.

3.2 The task

Each Tornado world is represented as a Java array of chars, where each char represents an hexagonal cell with the following code:

- `'t'` means that the cell contains a Tornado

[1] `http://www.minesweeper.info/wiki/Main_Page`

[2] Examples can be tried out at Michael Gottlieb's website `https://mzrg.com/js/hexmine/jsmine.html`

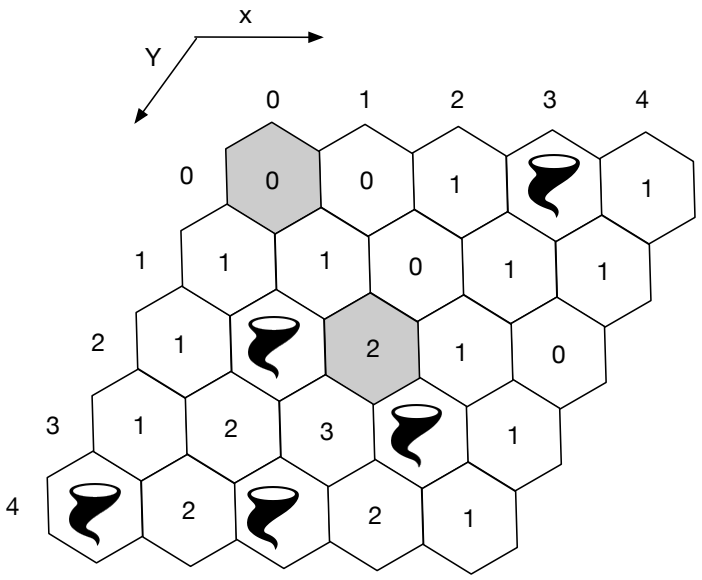
- `'0'–'6'` is the number of Tornadoes in the 6 adjacent neighbours

Each hexagonal cell is represented with its coordinates [x,y].
The Tornado sweeper game can be played at 3 levels with the following characteristics:

- Small worlds: in which N=5, T=5 indicating a board of 5x5 with 5 Tornadoes.
- Medium worlds: in which N=7, T=10 indicating a board of 7x7 with 10 Tornadoes.
- Large worlds: in which N=11, T=28 indicating a board of 11x11 with 28 Tornadoes.

In Figure 1 an example of 5x5 a Tornado world with 5 tornadoes is shown.

Figure 1: Example Tornado World



The rules of the game are as follows:

1. The total number of Tornadoes is known by the agent but should not be used to determine whether the game is over.
2. At the initial step the cells are all covered and the agent has no other information besides the size of the world.
3. Two starting clues are given to the agent: the agent can safely probe the cell at the top left-hand corner, and the cell in the centre of the board at the start of the game (e.g. [0,0] and [2,2] in Figure 1).
4. At each step the agent probes a covered cell, and receives information about the content of the cell. The cell will then be marked as probed/uncovered.
5. If the probed cell is a Tornado, the game is over.
6. If the cell contains a value 0 meaning that no adjacent cells contain Tornadoes, all the adjacent cells will also be probed.
7. If the content of the cell has a value > 0, this is a clue about the number of Tornadoes existing in the 6 adjacent cells. The agent should make inferences about its view of the world and decide which cell should be probed next.

8. If all but T cells are probed without a game over, the agent wins the game.

9. The agent may use flags to signal the presence of a Tornado in covered cells.

There are a number of possible reasoning strategies that the agent can employ, the agent goal is to employ one that limits the amount of random probing needed to solve the game, in order to increase the chances of winning. For more details on the strategies, please refer to the lecture slides.

3.3 Starter Code

For this practical, two starter classes are provided `World.java`, and `Board.java`. The class `World.java` provides the Tornado worlds to be used, 33 in total, defined as Enum Type[3]. The Tornado worlds are named according to different levels of difficulty, *SMALL, MEDIUM, LARGE*. Three additional worlds are included: *TEST0*, the board shown in Figure 1; *TEST1* and *TEST2*, the boards discussed in the lectures. The `Board.java` contains code that showcases how to set the board to be used, and how to print it. Note that the indexes shown on the board in Figure 1 correspond to the indexes of the two dimensional char array given in the starter code. You are free to use this starter code as-is or you may modify it, however, please ensure that you do not change the worlds to be tested (you may include additional ones), and that the printed boards are formatted in a similar way as the printing method shown in `Board.java`.

3.4 Basic Tornado Sweeper Agent

A basic Tornado Sweeper agent is one that can play the Tornado sweeper game by randomly moving on the board. You are required to write a program that takes any of the Tornado worlds provided and permits an agent to play the game by probing the cells. The agent should hold a knowledge base indicating the cells that are yet to be probed, and the information about the cells already probed. The agent's view of the world should be clearly distinct from the board representing the world. In addition, the agent should be able to randomly probe a cell, receive and process the information contained in that cell. We call this approach RPX. The agent should also be able to establish whether the game is over. At each step, a statement should be printed out indicating the action of the agent or the state of the game; e.g. "probe x y" for uncovering a cell in [x,y] coordinates, "flag x y" for marking the presence of a Tornado in [x,y], "tornado-in x y" for uncovering a tornado, "game won" or "game lost", etc …. In addition, the agent's view of the board at each step should be printed.

3.5 Intermediate Tornado Sweeper Agent

An intermediate Tornado Sweeper agent is one that is able to flag potential tornadoes to avoid a game over taking into consideration the knowledge acquired so far. In addition to the random probing, this agent uses a logical strategy, or a combination of strategies, to play the Tornado sweeper game. Two strategies can be attempted in this step, adapting those seen in the lectures for square grid sweeper games:

- **Single point strategy for hexagonal worlds (SPX)**: adapt the single point reasoning strategy for hexagonal worlds. The agent should use this strategy to flag or probe the next cell.

- **Satisfiability Strategy for hexagonal worlds (SATX)**: adapt the satisfiability test reasoning strategy for hexagonal worlds. The agent should be able to transform its current

[3] `http://docs.oracle.com/javase/tutorial/java/javaOO/enum.html`

partial view of the world into a logic sentence, and use satisfiability results to select the next move. A SAT solver should be used to prove that a given cell does or does not contain a Tornado.

These strategies can be used individually or in combination. When no other moves are available the agent might make a random step before continuing with the strategy. For SATX, SAT4J Core[4] is to be used. In addition, you may use additional libraries such as those discussed in the lectures to parse and convert a propositional formula to CNF.

3.6 Additional Advanced Logical Agent

It is strongly recommended that you ensure you have completed the requirements for the basic and intermediate agent before attempting any of these additional requirements. You may consider one or two additional functionalities:

1. Suggest an additional strategy to improve the performance of the logic agent (some inspiration may be drawn from the Minesweeper wiki [5]).
2. Include additional items or rules in the game, write the rules for these items and extend the implementation of the game to include these new rules (some inspiration may be drawn from the Wumpus game).
3. Permit the agent to play on boards of a different shape (e.g. triangular, but excluding the classic square grid ones).
4. Use the satisfiability approach to solve other similar games on a grid, such as Hitori[6] or Soduku[7]. Test this approach only on small environments/reduced boards of those games.
5. Suggest your own additional requirements: for a significant extension, you may increase the complexity of the game and extend your reasoning strategies to guide the agent to solve the problem, or identify additional games that can be solved with similar logic strategies.

4 Code and Report Specifications

4.1 Code Submission and Running

The program must be written in Java and your implementation must compile and run on the School lab Machines without the use of an IDE. Please do not use libraries that implement minesweeper algorithms, but other libraries that are secondary to the objectives of the practical can be used. Your source code should be placed in a directory called **A2src/** and should include all non-standard external libraries. Your code should run using the following command for the basic and intermediate agent:

```
java A2main <RPX|SPX|SATX> <ID>  [any param]
```

where ID is the world name. For example to run a random probing agent on the board in Figure 1 we will use:

```
java A2main RPX TEST0
```

For advanced agent functionalities, please include clear running instructions. Please note that code that does not adhere to these instructions will not be accepted.

[4] http://www.sat4j.org/products.php#core
[5] http://www.minesweeper.info/wiki/Windows_Minesweeper
[6] https://en.wikipedia.org/wiki/Hitori
[7] https://en.wikipedia.org/wiki/Sudoku

4.2 Report

You are required to submit a report describing your submission in PDF with the structure and requirements presented in the additional document *CS5011_A_Reports*. The core sections (Design, Implementation and Evaluation) have a limit of 1500 words in total. The report should include clear instructions on how to run your code. Consider the following points in your report:

1. Describe the implementation of the game infrastructure.
2. Describe the implementation of agents and their strategies.
3. Evaluate the agent performance on the given Tornado worlds. Comment on the amount of random probing required and any other performance feature of the agents using different strategies.

5 Deliverables

A single ZIP file must be submitted electronically via MMS by the deadline. Submissions in any other format will be rejected.

Your ZIP file should contain:

1. A PDF report as discussed in Section 4.2
2. Your code as discussed in Section 4.1

6 Assessment Criteria

Marking will follow the guidelines given in the school student handbook. The following issues will be considered:

- Achieved requirements
- Quality of the solution provided
- Examples and testing
- Insights and analysis demonstrated in the report

Some guideline descriptors for this assignment are given below:

- For a mark of 8 or higher: the submission implements part of the basic agent, printing the current agent knowledge and is able to probe the cells in some order, adequately documented or reported.
- For a mark of 11 or higher: the submission implements fully the basic agent with ability of randomly selecting the next cell (RPX), and determining whether the game is over. The code submitted is of an acceptable standard, and the report describes clearly what was done, with good style.
- For a mark of 14 or higher: the submission implements fully the basic agent, and at least one of the SPX,SATX strategies of the intermediate agent. It contains clear and well-structured code and a clear report showing a good level of understanding of design and evaluation of logic agents.

- For a mark of 17: the submission implements fully the basic and intermediate agent with a combination of the two SPX,SATX strategies. It contains clear, well-designed code, together with a clear, insightful and well-written report, showing in-depth understanding of design and evaluation of logic agents.
- Above 17: the submission implements fully the basic and intermediate agent and one or two (max) advanced agent functionalities. The submission should demonstrate unusual clarity of implementation, together with an excellent and well-written report showing evidence of extensive understanding of design and evaluation of logic agents.

7 Policies and Guidelines

Marking: See the standard mark descriptors in the School Student Handbook

`https://info.cs.st-andrews.ac.uk/student-handbook/learning-teaching/feedback.html#Mark_-Descriptors`

Lateness Penalty: The standard penalty for late submission applies (Scheme B: 1 mark per 8 hour period, or part thereof):

`https://info.cs.st-andrews.ac.uk/student-handbook/learning-teaching/assessment.html#latenesspenalties`

Good Academic Practice: The University policy on Good Academic Practice applies:

`https://www.st-andrews.ac.uk/students/rules/academicpractice/`

Alice Toniolo
(a.toniolo@st-andrews.ac.uk)
October 13, 2019