CS5011 A3 Report

170008773

19th November 2017

1 Parts completed

- We successfully implemented all the requirements for part 1
- We again, sucessfully implemented all the requirements for part 2.

2 Parts not completed

- At time of writing we did not attempt to implement the SAT-solver strategy.

3 Literature review

The history

The rules Minesweeper consists of a rectangular board of cells. At the start of the game, all the cells are covered, and some cells will contain mines. The Player/agent can perform two actions in this game: Flagging or uncovering a cell. If a cell containing a mine is uncovered the agent has lost the game. If a cell that does not contain a mine is uncovered it will reveal a number. This number is equal to the number of cells that are adjacent to the uncovered cell and contain a mine. If a cell is uncovered that is not adjacent to any mines, all of its neighbours will be uncovered. The agent has won when all of the cells that do not contain a mine are uncovered.

P vs. NP Much has been written about the complexity of minesweeper. Complexity is a measure of how "hard" a problem is. Kaye (2000) tells us that complexity-theory is a way of extimating the amount of time needed to solve a problem given the *length* of the input. The first class of problems is a class called P, for *polynomial-time computable* problems. These are the porblems that when given an input of length n, that can be solved in n^k steps for some exponent k. Kaye writes that these probelms are precicely the ones that are practically solvable. Conversely NP or *Nondeterministic Polynomail-time computable* is a class of problems that is solvable in polynomial time using "non-deterministic" algortihms (i.e. algorithms where the computer is allowed to make some guesses).

The complexity of minesweeper Kaye (2000) proved that minesweeper is NP-Complete. This

4 Design

5 Examples and Testing

5.1 Testing

Initial testing During the early stages of developemtn we mainly used two forms of testing. Manual inspection of states and outputs and `assert` statements.

Framework

5.2 Examples

A single run A single run of the programm using the easy equation strattegy looks as follows:

```
java -jar Logic2.jar ../worlds/easy/nworld1
                Starting new game
                Probing: (0,0)
                  0  ?  ?  ?  ?
                  ?  ?  ?  ?  ?
                  ?  ?  ?  ?  ?
                  ?  ?  ?  ?  ?
                  ?  ?  ?  ?  ?

                Probing: (1,0)
                  0  0  ?  ?  ?
                  ?  ?  ?  ?  ?
                  ?  ?  ?  ?  ?
                  ?  ?  ?  ?  ?
                  ?  ?  ?  ?  ?

                Probing: (2,0)
                  0  0  0  ?  ?
                  ?  ?  ?  ?  ?
                  ?  ?  ?  ?  ?
                  ?  ?  ?  ?  ?
                  ?  ?  ?  ?  ?

                Probing: (3,0)
                  0  0  0  2  ?
                  ?  ?  ?  ?  ?
                  ?  ?  ?  ?  ?
                  ?  ?  ?  ?  ?
                  ?  ?  ?  ?  ?

                Probing: (1,1)
                  0  0  0  2  ?
                  ?  0  ?  ?  ?
                  ?  ?  ?  ?  ?
                  ?  ?  ?  ?  ?
```

```
  ?  ?  ?  ?  ?

Probing: (0,1)
  0  0  0  2  ?
  0  0  ?  ?  ?
  ?  ?  ?  ?  ?
  ?  ?  ?  ?  ?
  ?  ?  ?  ?  ?

Probing: (0,2)
  0  0  0  2  ?
  0  0  ?  ?  ?
  1  ?  ?  ?  ?
  ?  ?  ?  ?  ?
  ?  ?  ?  ?  ?

Probing: (1,2)
  0  0  0  2  ?
  0  0  ?  ?  ?
  1  2  ?  ?  ?
  ?  ?  ?  ?  ?
  ?  ?  ?  ?  ?

Probing: (2,1)
  0  0  0  2  ?
  0  0  0  ?  ?
  1  2  ?  ?  ?
  ?  ?  ?  ?  ?
  ?  ?  ?  ?  ?

Probing: (3,1)
  0  0  0  2  ?
  0  0  0  2  ?
  1  2  ?  ?  ?
  ?  ?  ?  ?  ?
  ?  ?  ?  ?  ?

Probing: (2,2)
  0  0  0  2  ?
  0  0  0  2  ?
  1  2  1  ?  ?
  ?  ?  ?  ?  ?
  ?  ?  ?  ?  ?

Probing: (3,2)
  0  0  0  2  ?
  0  0  0  2  ?
  1  2  1  2  ?
  ?  ?  ?  ?  ?
  ?  ?  ?  ?  ?

SPS
Checking Cell (1,2)
Checking Cell (2,2)
```

```
Checking Cell (3,2)
Checking Cell (3,1)
Checking Cell (3,0)
Flagging: (4,0)
  0  0  0  2  F
  0  0  0  2  ?
  1  2  1  2  ?
  ?  ?  ?  ?  ?
  ?  ?  ?  ?  ?

Flagging: (4,1)
  0  0  0  2  F
  0  0  0  2  F
  1  2  1  2  ?
  ?  ?  ?  ?  ?
  ?  ?  ?  ?  ?

SPS
Checking Cell (1,2)
Checking Cell (2,2)
Checking Cell (3,2)
Checking Cell (3,1)
Probing: (4,2)
  0  0  0  2  F
  0  0  0  2  F
  1  2  1  2  1
  ?  ?  ?  ?  ?
  ?  ?  ?  ?  ?

SPS
Checking Cell (1,2)
Checking Cell (2,2)
Checking Cell (3,2)
Checking Cell (4,2)
Probing: (3,3)
  0  0  0  2  F
  0  0  0  2  F
  1  2  1  2  1
  ?  ?  ?  2  ?
  ?  ?  ?  ?  ?

Probing: (4,3)
  0  0  0  2  F
  0  0  0  2  F
  1  2  1  2  1
  ?  ?  ?  2  0
  ?  ?  ?  ?  ?

Probing: (3,4)
  0  0  0  2  F
  0  0  0  2  F
  1  2  1  2  1
  ?  ?  ?  2  0
  ?  ?  ?  2  ?
```

```
Probing: (4,4)
  0  0  0  2  F
  0  0  0  2  F
  1  2  1  2  1
  ?  ?  ?  2  0
  ?  ?  ?  2  0

SPS
Checking Cell (1,2)
Checking Cell (3,4)
Flagging: (2,3)
  0  0  0  2  F
  0  0  0  2  F
  1  2  1  2  1
  ?  ?  F  2  0
  ?  ?  ?  2  0

Flagging: (2,4)
  0  0  0  2  F
  0  0  0  2  F
  1  2  1  2  1
  ?  ?  F  2  0
  ?  ?  F  2  0

SPS
Checking Cell (1,2)
Checking Cell (2,2)
Probing: (1,3)
  0  0  0  2  F
  0  0  0  2  F
  1  2  1  2  1
  ?  3  F  2  0
  ?  ?  F  2  0

SPS
Checking Cell (1,2)
Flagging: (0,3)
  0  0  0  2  F
  0  0  0  2  F
  1  2  1  2  1
  F  3  F  2  0
  ?  ?  F  2  0

Probing: (0,4)
  0  0  0  2  F
  0  0  0  2  F
  1  2  1  2  1
  F  3  F  2  0
  1  ?  F  2  0

Probing: (1,4)
  0  0  0  2  F
  0  0  0  2  F
```

```
1  2  1  2  1
F  3  F  2  0
1  3  F  2  0

      Final number of random guesses: 0
      Final number of probes: 20
      Final number of flags: 5
      Number of runs untill success: 1
```

Whereas a run form the ProduceExperimentReport.jar looks like this:

flags

	EASY_EQUATION,	RANDOM_GUESS,	SINGLE_POINT,
../worlds/easy/nworld1	5,	5,	5,
../worlds/easy/nworld2	9,	5,	8,
../worlds/easy/nworld3	8,	5,	7,
../worlds/easy/nworld4	7,	5,	7,
../worlds/easy/nworld5	8,	5,	7,
../worlds/hard/nworld1	20,	0,	20,
../worlds/hard/nworld2	34,	0,	32,
../worlds/hard/nworld3	33,	0,	35,
../worlds/hard/nworld4	34,	0,	34,
../worlds/hard/nworld5	34,	0,	34,
../worlds/medium/nworld1	16,	0,	16,
../worlds/medium/nworld2	10,	0,	10,
../worlds/medium/nworld3	16,	0,	17,
../worlds/medium/nworld4	10,	0,	10,
../worlds/medium/nworld5	16,	0,	16,

probes

	EASY_EQUATION,	RANDOM_GUESS,	SINGLE_POINT,
../worlds/easy/nworld1	20,	20,	20,
../worlds/easy/nworld2	16,	20,	17,
../worlds/easy/nworld3	17,	20,	18,
../worlds/easy/nworld4	18,	20,	18,
../worlds/easy/nworld5	17,	20,	18,
../worlds/hard/nworld1	80,	67,	80,
../worlds/hard/nworld2	66,	11,	68,
../worlds/hard/nworld3	67,	45,	65,
../worlds/hard/nworld4	66,	36,	66,
../worlds/hard/nworld5	66,	25,	66,
../worlds/medium/nworld1	65,	19,	65,
../worlds/medium/nworld2	71,	57,	71,
../worlds/medium/nworld3	65,	34,	64,
../worlds/medium/nworld4	71,	50,	71,
../worlds/medium/nworld5	65,	65,	65,

randomGuesses

	EASY_EQUATION,	RANDOM_GUESS,	SINGLE_POINT,
../worlds/easy/nworld1	0,	4,	0,
../worlds/easy/nworld2	0,	6,	6,
../worlds/easy/nworld3	0,	7,	1,
../worlds/easy/nworld4	0,	4,	0,
../worlds/easy/nworld5	0,	4,	2,

```
../worlds/hard/nworld1          0,      7,      5,
../worlds/hard/nworld2          0,      1,      5,
../worlds/hard/nworld3          0,      1,      1,
../worlds/hard/nworld4          0,      1,      0,
../worlds/hard/nworld5          0,      6,      0,
../worlds/medium/nworld1        0,      2,      0,
../worlds/medium/nworld2        0,      1,      0,
../worlds/medium/nworld3        0,      6,      2,
../worlds/medium/nworld4        0,      2,      0,
../worlds/medium/nworld5        0,      9,      1,

runsUntilSuccess
                        EASY_EQUATION,  RANDOM_GUESS,   SINGLE_POINT,
../worlds/easy/nworld1          1,         72,          1,
../worlds/easy/nworld2          1,        345,          3,
../worlds/easy/nworld3          1,        523,          5,
../worlds/easy/nworld4          1,        161,          1,
../worlds/easy/nworld5          1,        137,          3,
../worlds/hard/nworld1          1,       1000,          4,
../worlds/hard/nworld2          1,       1000,          6,
../worlds/hard/nworld3          1,       1000,          2,
../worlds/hard/nworld4          1,       1000,          1,
../worlds/hard/nworld5          1,       1000,          1,
../worlds/medium/nworld1        1,       1000,          1,
../worlds/medium/nworld2        1,       1000,          1,
../worlds/medium/nworld3        1,       1000,         19,
../worlds/medium/nworld4        1,       1000,          1,
../worlds/medium/nworld5        1,       1000,          3,
```

6 Running

1. Several `.jar` files are included with the submition. All of the `LogicN.jar` files should be run in the same maner: `java -jar LogicN.jar <testDirectory>` The program expects there to be a file in this directory called `map.txt`. The format of this file is as follows. The first three lines of the file should contain just one integer. The first two should be the length and width of the world respectively. The third should be the number of nettles present in the world. Then the array of the world should follow in CSV format (i.e. rows of integers seperated by commas and rows should be seperated by newlines). For example:

   ```
   5
   5
   0, 0, 0, 2, -1
   0, 0, 0, 2,-1
   1, 2, 1, 2, 1
   -1, 3, -1, 2, 0
   1, 3, -1, 2, 0
   ```

   Further examples of the file and directory structure that the programmes expect are included.

2. There is another `.jar` file included with the submition called `ProduceExperimentReport.jar`. This file expects as argument the root directory of the experiments. It will then recursively go through this directory tree looking for files called `map.txt` and running the experiments it finds with all provided

implementations and record the data those experiments report. When all the experiments are done it will output the result in a table format (one for every variable)

7 Evaluation

8 Conclusion

word count:

References

Kaye, R. (2000). Minesweeper is NP-complete. *Mathematical Intelligencer*, 22(2):9.