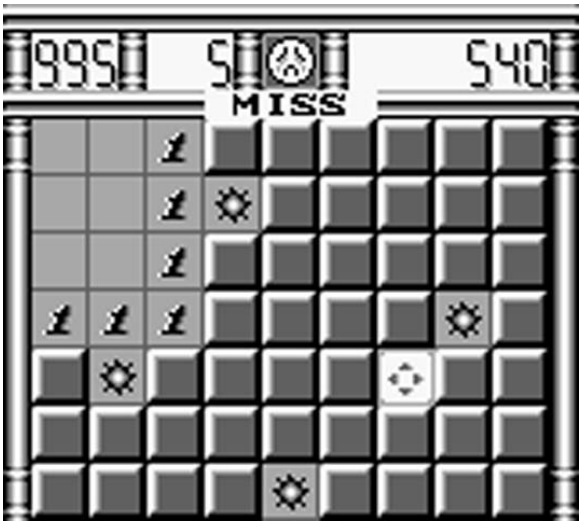


Minesweeper
Student Manual

For Introduction to Artificial Intelligence

By: Justin Chung, John Lu, Sui Ting Yeung, Jian Li, Abdullah Younis, Jan Christian
Contact: Jan Christian (janc2@uci.edu)
Last Updated: 15 September 2018 by Jan Christian

Table of Contents

I. Introduction

In this programming assignment, you will be tasked with implementing a Minesweeper AI Agent, which should be able to play and solve the Minesweeper game. You will have the choice of programming in C++, Java, or Python. Much of the code has already been written for you. Your agent should be able to take in percepts and act accordingly. Your grade will depend on your agent's performance measure. At the end of the quarter, your agent will compete against your peers' agents in a class-wide tournament.

II. Minesweeper Game Mechanics

This version of Minesweeper is based on the classic computer game. You can easily find out the details of the game from the web, for example: https://en.wikipedia.org/wiki/Minesweeper_(video_game)

However, there are a few differences designed to better evaluate your agent which will become apparent as you play with and familiarize yourself with the game. In Minesweeper, you are given a board that is set up as a 2D grid of tiles. Each tile covers either: (1) a hint number (that tells how many mines are around that tile) or (2) a mine. Ultimately, your agent's goal is to uncover all tiles which do not contain a mine. A more concrete definition of the game is given by the following PEAS description.

Performance Measure

- The performance measure of your agent will be a score calculated based on number of worlds your agent has completed. Points are awarded to your agent only if it successfully solves the entire world. Each difficulty has different weight.
- The game ends when your agent chooses to leave the game or if your agent uncovers a mine. In either of these cases you'll get a zero.

Environment

- Each difficulty has a different dimension and number of mines:
    - Beginner: 8 row x 8 column with 10 mines
    - Intermediate: 16x16 with 40 mines
    - Expert: 16x30 with 99 mines
- The board begins with 1 random tile already uncovered and presumably safe
- Mines are randomly placed throughout the board.
- Your agent dies when it uncovers a mine.

Actuators

- Your agent has 4 moves:
  - (1) The action UNCOVER reveals a covered tile.
  - (2) The action FLAG places a flag on a tile.
  - (3) The action UNFLAG removes a flag from a tile if that tile has a flag.
  - (4) The action LEAVE ends the game immediately.
- The actions UNCOVER, FLAG, and UNFLAG are to be coupled with a pair of coordinates which allows the agent to act on a single tile.

Sensors

- Your agent will receive only one percept:
  - Following an UNCOVER action, your argent will perceive the hint number associated with the previous UNCOVER action. This number represents how many mines are within that tile's immediate neighbors.
  - Following a FLAG or UNFLAG action, your agent will perceive -1.

III. Tasks to Complete

Setup Your Environment

NOTE THAT THE CODES YOU WILL SUBMIT MUST RUN IN OPENLAB! This project will take advantage of UCI's openlab; any other coding environment is not supported.

- Install Required Applications

To connect to openlab, you will need to use SSH. SSH stands for Secure Shell. It is a program designed to allow users to log into another computer over a network, to execute commands on that computer and to move files to and from that computer. A Mac user can use the terminal application, whereas, a Windows user will need to install PuTTY. You can download PuTTY from here. Download the MSI installer for Windows, and run the installer for PuTTY.

- Connect to Openlab

Connecting to openlab is as easy as SSHing into the open lab server. If you are on Windows and using PuTTY, type "openlab.ics.uci.edu" into the Host Name box; make sure the port is 22 and the SSH flag is ticked. Click open, and login using your ICS account info. If you are using Mac, open the terminal found under Application -> Utilities. Enter 'ssh yourICSusername@openlab.ics.uci.edu' and login using your into ICS account.

- Download the shells on Openlab

To download the shells on Openlab, you will use Git. On openlab, whether through PuTTY or terminal, execute the following git clone command:

```
Git clone [address]
```

Extra Information about Openlab:
http://www.ics.uci.edu/~lab/students/#unix
https://www.ics.uci.edu/computing/linux/hosts.php
Extra Information about UNIX:
https://cgi.math.princeton.edu/compudocwiki/index.php?title=Documentation_and _Information:Getting_started_with_Linux

Program Your AI

Once you have your environment setup, you can start to program your agent. In the 'src' folder of your shell you will find the source code of the project. **You are only allowed to make changes to the MyAI class.** You are not allowed to copy source codes from any source because it violates Academic Honesty, however, you are allowed to include standard libraries.

Compile Your AI

Compiling your program is easy as executing the command make from the shell's root directory (the directory with the makefile in it).

Test Your AI

To run your program after you have compiled it, navigate to the bin folder. You should find the compiled program inside. Refer to the Shell Manual Appendix for help running it. To generate large amounts of worlds to use with the folder option, refer to the World Generator. If you are using the Python Shell make sure you are using Python 3.5.2. On openlabs, run the command `"module load python/3.5.2"` to load Python 3.5.2.

Write Your Project Report

Write a report according to your Professor's instructions. Make sure your report is in pdf format and place it inside the 'doc' folder. The report template is at the bottom of this doc.

Submit Your Project

At this point you should have your most up-to-date source code in the ‘src’ folder, your report in pdf format in the ‘doc’ folder, and your compiled project in the ‘bin’ folder. Navigate to your shell’s root directory and execute the command make submission. It will ask you for some information and create a zip file inside the folder. Submit this zip file to EEE or Canvas.

IV. WorldGenerator Manual

The Minesweeper World File

If you’d like to create your own custom Minesweeper World, you can create a text file using the following format:

> [rowDimension][space][colDimension]
> [startingRow][space][startingColumn]
> [2D grid of board]

• “StartingRow” and “startingColumn” represent the coordinates of the first tile that the world uncovers for you. This feature is designed to guarantee your agent will be safe on its’ starting tile.

• The 2D grid should be a sequence of 0’s and 1’s. Columns are separated by a single space while rows are separated by a new line. 0’s represent safe tiles while 1’s represent tiles with mines. The bottom left-most tile should be interpreted as (1, 1). When selecting starting tile, the tiles around it will always be safe, namely 0.

• If your 2D grid does not contain at least a 3x3 square of 0’s (because the starting tile must be a safe tile), the world is considered invalid.

Using the Python script

● There is a Python script “WorldGenerator.py” that you can use to easily generate a set of worlds in the form of txt files.

● If you see "Error opening file" in your console when running the script, that means you haven't create a folder called "Problems" in the directory you are running this script, hence the script fails to locate.

● To run the script, issue the command:

```
python3 WorldGenerator.py [numFiles] [filename] [rowDimension]
[colDimension] [numMines]
```

The arguments in square brackets are in that order and represent the following:

numFiles	- The number of files to generate
filename	- The base name of the file
rowDimension	- The number of rows
colDimension	- The number of columns
numMines	- The number of mines

Note that all arguments are required and have certain restrictions, which are listed below:

(1) The minimum number of rows is 4.
(2) The minimum number of columns is 4.
(3) The minimum number of mines is 1.
(4) The number of mines must also be less than or equal to
(rowDimension)*(colDimension) - 9

If any of these conditions are not met, the script will not generate any worlds. Another thing to note is that you can only generate a set worlds of the same dimensions. In order to generate worlds of different sizes, you need to rerun the script with different command-line arguments.

Using the bash script

- Another way to generate worlds is to use the bash script. There are two bash scripts provided. You can easily change the script to generate a different number of worlds or worlds of different dimensions. You may also write your own bash script.

- To run the scripts, issue the command:

```
./generateTournament
```

This script will be used to generate the tournament set of worlds and is a good simulation for your agent's actual performance in the tournament. This script creates a "Problems" folder and generate a random set of 1000 Beginner, 1000 Intermediate, and 1000 Expert worlds.

```
./generateSuperEasy
```

This script will be used to generate the set of worlds for the Minimal submission. It generates 1000 random Easy worlds.

- Run `chmod +x generateTournament.sh` or `chmod +x generateSuperEasy.sh` if you run into a "permission denied" message

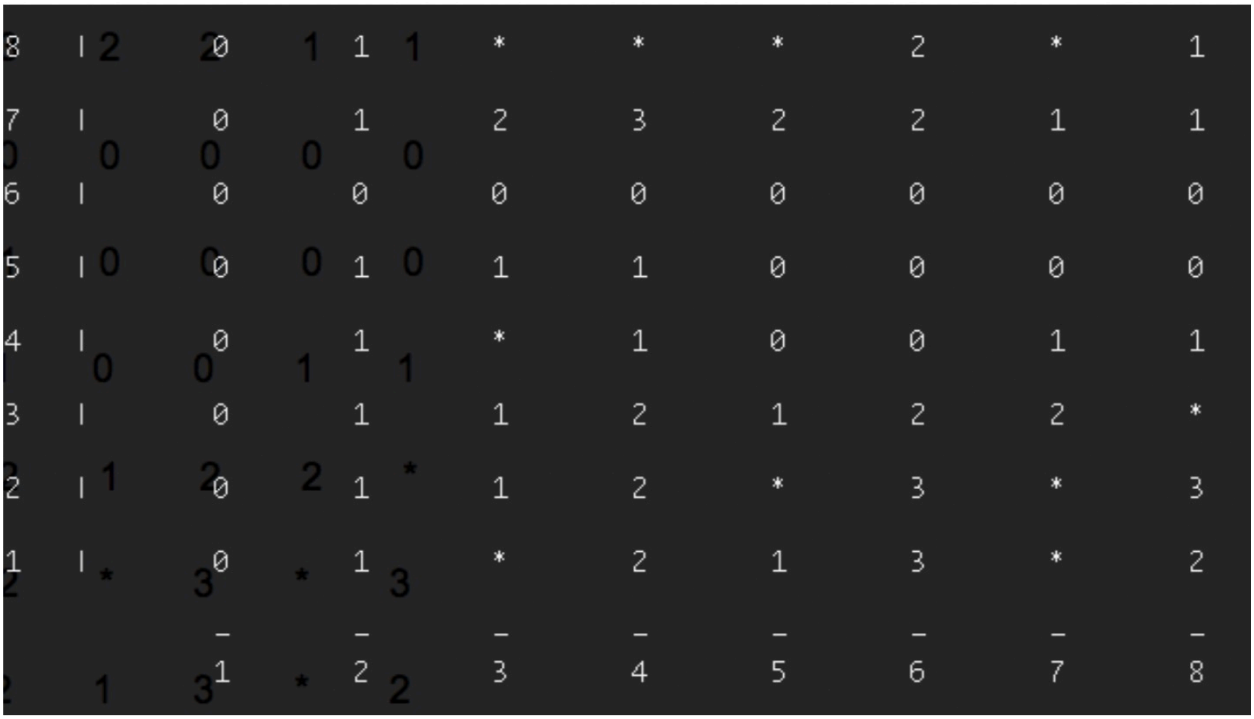
- As a side note, these bash scripts are written to automatically delete any previous file/ folder called "Problems" and create a new one.

Example World

A txt file like this:

```
88
13
00111010
00000000
00000000
00000000
00100000
00000001
00001010
00100010
```

will generate (in console):



Remember that arrays start with the index 0 while the minesweeper's dimension starts with the index of 1.

V. Understanding the Tournament

- After you submit your Final submission and the deadline passes, your agent will be entered into a tournament with your classmates.
- The tournament checks to make sure you followed all the instructions correctly, then runs your agent across 3N random worlds with three different difficulty levels of N worlds each. Every agent is run on the same 3N worlds to ensure fairness.
- Your agent's total score is calculated and a scoreboard is constructed that will be made available.
- Your agent will be timed-out if it hangs for longer than 2 hours.
- After the scoreboard is constructed, scores are checked for any illegal submissions. These include two agents with the same score.
- Late submissions won't be entered into the Tournament, and therefore, receive no bonus points.

VII. Scoring Explanation

The scoring will depend on:
1) Team Formation
2) Submissions: Minimal , Draft , Final
3) Final Report
4) Tournament Bonus
For (1), (2), and (3): You will lose 10% credit for each late day after the deadline and a fraction thereof. For (4), late submissions get zero bonus.

Team Formation

The submission of Team Formation must follow strict rules:
- Team names must use only numbers and alphabetic characters. Any symbols including whitespace, is not allowed. Capital letters are allowed.
- The submission text must follow this format:

> TeamName: <enter team name>
> Member1: <enter member 1 name only>
> Member2: <enter member 2 name only>

- If done solo, leave <enter member2 name only> blank

You will lose points (up to 100%) if you don't follow the stated rules above.

Submissions

- **Minimal:** Complete 20% out of N Easy worlds (5x5 with 1 mine)
- **Draft:** Complete 30% out of N Beginner worlds (8x8 with 10 mines) and 15% out of N Intermediate worlds (16x16 with 40 mines)
- **Final:** Complete 60% out of N Beginner worlds (8x8 with 10 mines), 50% out of N Intermediate worlds (16x16 with 40 mines), and 10% out of N Expert worlds (16x30 with 99 mines)

- You will get full credit if you meet these requirements per submission.
- You will lose point if you don't meet the requirement based on the fractional weight of your completed worlds against the required completion.
- Each difficulty requirement has the same weight that sums up to 100%, for example: in Minimal, Easy has a weight of 100%. In Draft, Beginner and Intermediate has a weight of 50% each. In Final, each difficulty has a weight of approximately 33.3%
- For example, for Final, if your AI completes >= 60% Beginner worlds, you will get 100% out of 33.3% (Beginner weight). Else, you get (completed/required) x 100% out of 33.3%. This applies to all three difficulties.

Final Report

If you write each section in clear, logical, technical prose, you will get 100% credit. The template for the Final Report is given at the bottom of this document.

Tournament Bonus

The tournament score percentage is calculated as follow:

- The maximum completion score in class for each difficulty will be set as the bar, which is 100%. Therefore, the bars can originate from different teams.
- An example bar would be: 99% Beginner worlds completed (by Team A), 85% Intermediate worlds completed (by Team X), 40% Expert worlds completed (by Team Z).
- The weight for each difficulty will follow Final submission explained above.
- Sample calculation: Team B achieves 66% Beginner completion, 68% Intermediate completion, and 20% Expert completion. Thus, Team B's tournament score percentage will be (66/99) x 33.3% + (68/85) x 33.3% + (20/40) x 33.3%, which is 65.49%.
- This percentage is not an absolute score and will only be used to rank your AI from others.

Your AI's percentage score will be compared to other students in a ranking system. The tournament bonus is between 1-10, where the top 10% highest scoring teams will get a 10, the second 10% will get a 9, and so on. Generally, a 10 means a 10% bonus to your Final AI submission.

Common Mistakes

Below are the common mistakes that students often make when submitting.

- Leaving 'print' or 'cout' statements in the source codes. Please delete these because it will create problem for the grader in both time and script.

- Giving additional information or text in the Team Formation submission. Please only fill the required info.

- Not testing in openlab. This seldom happens but it does: it runs in your computer environment, but it doesn't run in openlab (and openlab is where the grading will take place), so please test it for consistency.

- Only submitting the source code(s)/ Not submitting a zip file. Please use the make/ make all command provided, as it will make a zip file with your team name (<team_name>.zip).

VI. Appendix: Shell Manual

Synopsis

`[Name] [Options] [InputFile] [OutputFile]`

Name

The command line name used to invoke this program will change depending on the shells:

`python3 Main.pyc`	if using python shell
`java –jar mine.jar`	if using java Shell
`./Minesweeper`	if using cpp shell

Options

`–m` Use the ManualAI instead of MyAI. If both `–m` and `–r` specified, ManualAI will be turned off.
`–r` Use the RandomAI instead of MyAI.
`–d` Debug mode, which displays the game board after every move.
`–v` Verbose mode, which displays name of world files as they are loaded.
`–f` Depending on the InputFile format supplied, this operand will trigger program **1)** Treats the InputFile as a folder containing many worlds. The program will then construct a world for every valid world file found. The program to display total score instead of a single score. The InputFile operand must be specified with this option **2)** Threats the inputFile as a file. The program will then construct a world for a single valid world file found. The program to display a single score.

Operands

[`InputFile`]: A path to a valid Minesweeper World file, or folder with –f. This operand is optional unless used with –f or OutputFile.
[`OutputFile`]: A path to a file where the results will be written. This is optional.

Examples

You can change "`python 3 Main.pyc`" to other [name] depending on the language you use

`python3 Main.pyc`	Constructs a random 8x8 with 10 mines world, runs the MyAI agent on the world, and prints output to console.
`python3 Main.pyc -m`	Constructs a random 8x8 with 10 mines world, runs the ManualAI agent on the world, and prints output to console.
`python3 Main.pyc -d`	Constructs a random 8x8 with 10 mines world, runs the MyAI agent on the world, and prints output to console. After every turn, the game pauses and prints the current game state to the console.
`python3 Main.pyc -r`	Constructs a random 8x8 with 10 mines world, runs the RandomAI on the world, and prints output to console.
`python3 Main.pyc -rd`	Constructs a random 8x8 with 10 mines world, runs the RandomAI agent on the world using debug mode, and prints output to console. After every turn, the game pauses and prints the current game state to the console.
`python3 Main.pyc -f /path/to/world/file.txt`	Constructs the world specified in the file, runs the MyAI agent on the world, and prints output to console.
`python3 Main.pyc -f /path/to/world/files/`	Constructs all the worlds specified in the folder, runs the MyAI agent on all the worlds, and prints output to console.

`python3 Main.pyc -f /path/to/world/files/ /path/to/outputfile/yourscores.txt`

Constructs all the worlds specified in the folder, runs the MyAI agent on all the worlds, and write output to txt file.

`python3 Main.pyc -fv /path/to/world/files/`

Constructs all worlds specified in the folder, runs the MyAI agent on all the worlds, and prints output to console. Before running every world, print a message indicating which specific world is running.

Notes

The Python shell uses Python version 3.5.2. When using debug mode or ManualAI, the board will printed to the console. Each tile is represented as a full stop potentially followed by a series of characters. Every board that gets displayed starts with 1-indexing and bottom left.

Minesweeper Final AI Report

Team name_____
Member #1 (name/id)_____
Member #2 (name/id)_____

I. Minimal AI

I.A. Briefly describe your Minimal AI algorithm. What did you do that was fun, clever, or creative?

I.B Describe your Minimal AI algorithm's performance:

Board Size	Sample Size	Score	Worlds Complete
5x5			
8x8			
16x16			
16x30			
Total Summary			

II. Final AI

II.A. Briefly describe your Final AI algorithm, focusing mainly on the changes since Minimal AI:

II.B Describe your Final AI algorithm's performance:

Board Size	Sample Size	Score	Worlds Complete
5x5			
8x8			
16x16			
16x30			
Total Summary			

III. In about 1/4 page of text or less, provide suggestions for improving this project (*this section does <u>NOT</u> count as past of your two-page total limit.*)