Zurich University
of Applied Sciences

zh aw

School of Engineering

P02 – 2048 game playing agent

1. Background

The game 2048 was developed by Italian programmer Gabriele Cirulli during a week-end, based on 1024 by Veewo Studio and conceptually similar to Threes by Asher Vollmer (you may read his story of the making & rip-offs in game development here: http://asherv.com/threes/threemails/).

The gameplay for 2048 is as follows: You can move the tiles (all of them at once) with your arrow keys. When two tiles with the same number touch, they merge into one, with the new tile having twice the old value. After each board-changing move, a new random

tile spawns, having a value of 2 (90% chance) or 4 (10% chance). You win by creating a tile with the value 2048.

In this lab, you will develop a simple (part 4) as well as a sophisticated AI agent (part 5) to remote control the 2048 game running in your browser. Before, you have to learn (and previously: install) the Python programming environment that we will be using throughout this course (part 2), and configure your browser (part 3).

2. Installing and Using Anaconda

Continue with part 2 if you already have Anaconda installed. Other Python distributions may be used, but are not supported.

2.1 Background

Anaconda is a scientific Python distribution by Continuum Analytics. Anaconda offers:
- Easy installers for Windows, MacOS & Linux with all necessary libraries included
- A good-enough and clear integrated development environment called "Spyder"

2.2 Downloading Anaconda

- Download the correct version for your system from here (ca. 600 MB):
- https://www.anaconda.com/distribution/
  - Be sure to use the Version for Python 3.x, *not* Python 2.x
  - Take the 64-bit variant if you have a 64 bit system

2.3 Installing Anaconda

- Follow the instructions in the setup process
  - You can safely accept all standard choices
  - On Windows: If you are asked if you want to install for everyone or just yourself, choose "all users"
  - On Windows: If you are asked if Anaconda should be your standard Python 3.x environment, conform with "yes"
  - On MacOS X: if the installer says something like 'cannot install', choose 'just install for me'.

2.4 Working with Spyder

Try the following things:
- Go to the interactive console tab (bottom right, tab "Console") and use it as a calculator
- Go to `temp.py` in the code editor (left) and write your individual "hello world" script. Run it, modify it, re-run it.

2.5 Update your Anaconda Python installation

Anaconda (Python) needs to be up to date to be used for the labs accompanying this module. Please update it using either the Anaconda Launcher or the following two console commands (from Anaconda install directory):

```
conda update conda
conda update anaconda
```

See also: http://docs.continuum.io/anaconda/install/update-version/

2.6 Learning Python

As a computer scientist it should not be too difficult for you to understand short Python scripts, and from there to start writing own code.

Here are some additional resources to guide your self-study (in order of increasing sophistication):
- Beginner's cheat sheet: https://groklearning-cdn.com/resources/cheatsheet-python-1.pdf
- Programmer's cheat sheet: https://perso.limsi.fr/pointal/_media/python:cours:mementopython3-english.pdf
- Official Python 3 tutorial: https://docs.python.org/3/tutorial/
- Python for analytics: https://www.analyticsvidhya.com/learning-paths-data-science-business-analytics-business-intelligence-big-data/learning-path-data-science-python/

3. Configure your Browser (Firefox or Chrome)

To play the game you can either use a special version of either Firefox (section 3.1), or Chrome (section 3.2), the latter being recommended.

3.1 Install and configure Firefox

1. Install a version of Firefox compatible with the "Remote Control" add-on:
   1.1. Windows: download and install Firefox Developer Portable 57.0 Beta 3 from e.g. https://sourceforge.net/projects/portableapps/files/Mozilla%20Firefox%20Developer%20Portable/
   1.2. MacOS: download and install Firefox Developer 56.0b9 from e.g. https://download-installer.cdn.mozilla.net/pub/devedition/releases/56.0b9/mac/en-US/

2. *Don´t update the browser when it opens, it will render the plugin useless!*

- Immediately, disable automatic software updates in "Preferences" before restarting Firefox!

3. Go to `about:config` (type it into the URL window), search for `extensions.legacy.enabled` and set it to `true`

4. Open Firefox→Extras→Add-ons→"Install from file" (or the page "about:addons",

   there in the "tools" menu using the gear-wheel icon [⚙·]) and select `remote_control-1.2-fx.xpi` (ships with this lab description)

5. If you experience an error because it is not signed, you have to disable signing temporarily. Go to `about:config` (type it into the URL window), search for `xpinstall.signatures.required` and set it to `false`

6. After the installation restart Firefox

7. To enable the plugin for the 2048 game:
   - Use Firefox to open the official game website https://play2048.co/ (or start it locally using the `2048_original.zip` archive shipping with this lab description)
   - Enable the Firefox Remote Control for this website: click the remote control button [▦] on the top right

3.2 Install and configure Chrome

1. Download and install Chrome from https://www.google.com/chrome/ (if it is not already installed)

2. To be able to remote control the browser, you need to start Chrome with the command line argument `--remote-debugging-port=9222`
   On Windows:
   - create a new shortcut on your desktop to the Chrome browser
   - Right click the shortcut, open "Properties"
   - In the field "Target", add `--remote-debugging-port=9222` at the end

3. Install the required dependency "websocket-client" in your Anaconda Python environment:
   ```
   pip install websocket-client
   ```

4. Start Chrome with the command line argument (see above), and open the website of the game: https://play2048.co/

You are now able to run the Python code templates provided with this lab description, which will control the browser game.

4. Getting started with 2048

Be sure that your Firefox or Chrome browser is open (Remote Control is activated in case of Firefox, command line parameter used in case of Chrome) and correctly configured (see previous section), otherwise it will fail.

4.1 Running the code template `2048.py`

To run the Python script you have to open your console and write either
```
python 2048.py -b firefox
```
or
```
python 2048.py -b chrome
```

4.2 Modifying the code template `heuristicai.py`

To program your first own agent that will solve the game you have to modify the file `heuristicai.py`.

The goal of this task is to build an agent which is (at the minimum) better than a random player, by coming up with some rules/heuristics *apart from the algorithms treated in the lecture*. This task could also be abbreviated as *"create an AI agent manually"* – don't use AI you learned about, but hard-code your own smartness.

To do so you have to implement the method `find_best_move(board)` in a better way than it is at the moment, where it will just move the board in a random way. You are provided with a 2D list of the game state, and you have to return the direction in which the game should move.
- The parameter `board` of `find_best_move` is a numpy 2D matrix which you can access like a normal 2D array. It represents the current gamestate.
- The sample solution achieves between 7'000 and 12'000 points and in most of the cases a 512/1024 tile as its maximum.

Hints: Some ideas for useful heuristics are
- When many tiles are on the board, the chance that you can't move anymore is greatly increased; thus, one of your goals is to have the board as empty as possible.
- If tiles of big value are at the corner of the board, they don't block the merging of the "smaller" tiles.
- A move bringing two tiles with the same value next to each other is preferable over a move that won't give you this advantage.

Can you come up with some rules to master the game? C'mon!

Before you continue with the next task, make a self-assessment: How well does your agent play? How does it compare to your classmates? To results you see on the web (if your search for "2048 AI")? How does it compare to you as a human player (not only result-wise, but also if you compare the choice of moves)? What are reasons that explain what you observe? What makes implementing human game-playing (depending on your achieved results) particularly easy or hard?

5. AI for 2048

2 Funktionen, die sich rekursiv aufrufen
es muss berücksichtigt werden, dass es zufalls 2 zu 90% oder 4 zu 10% auftreten kann
Man muss die Wahrscheinlichkeit von allen Boards berücksichtigen
Def => 1. max Tiefe dann heuristic soll benutzt werden.

In this task your goal is to build a game-playing 2048 agent based on the expectimax algorithm. To do so, you will modify the script `searchai.py`. Additionally, please modify 2048.py so that it calls `find_best_move(board)` from `searchai.py`, not `heuristicai.py` as in the previous task.

Man kann auch eine Abbruch

To implement expectimax, you still need a good heuristic to score a current board. You can re-use ideas (or code) here from what you tried in the previous task, or implement others. Probably a big difference to your previous implementation will be that now your heuristics are combined with proven and effective systematic search capability.

[Optional: If you are not satisfied with your own heuristics, check this post for some ideas: http://stackoverflow.com/questions/22342854/what-is-the-optimal-algorithm-for-the-game-2048. It also provides you with other algorithms and ideas to beat the game. If you are interested, give it a try!]

Assess the performance of your solution: What makes the difference to your agent of the previous task (if any)? What differentiates it from a human player? From state of the art (see e.g. http://www.cs.put.poznan.pl/mszubert/pub/szubert2014cig.pdf)? You can compete with your peers using the following online leaderboard: https://goo.gl/meh3Ro

Hints: How to use expectimax for 2048

The main challenge in playing 2048 is the randomness of where and which kind of tile will spawn. This makes the game nondeterministic. Expectimax is a good choice for games which have a nondeterministic model of operation.

Figure 1 shows an excerpt of how expectimax will work with `DEPTH=2`. Recall that after a successful move (which alters the board), a new tile will spawn on an empty field, with the chance of a value-2 tile is 90% that of a value-4 tile is 10%. Therefore, when you have three empty fields after the move, you can have six possible outcomes as new board states after the subsequent spawning (as depicted in Figure 1).

The heuristic to score the board is only needed at the leafs of the tree. The sample solution can reach 2048 most of the time with DEPTH=2.

Example: Imagine that when you move the current board UP, you will have 3 empty fields (say). This means that 6 possible new board positions exist after spawning.

To get the score of the UP move, we sum up all the scores of the new possible boards, weighted by their probability.

$1/3 * (0.9 * 14 + 0.1 * 7 + 0.9 * 3 + 0.1 * 0 + 0.9 * 5 + 0.1 * 5) = 7$
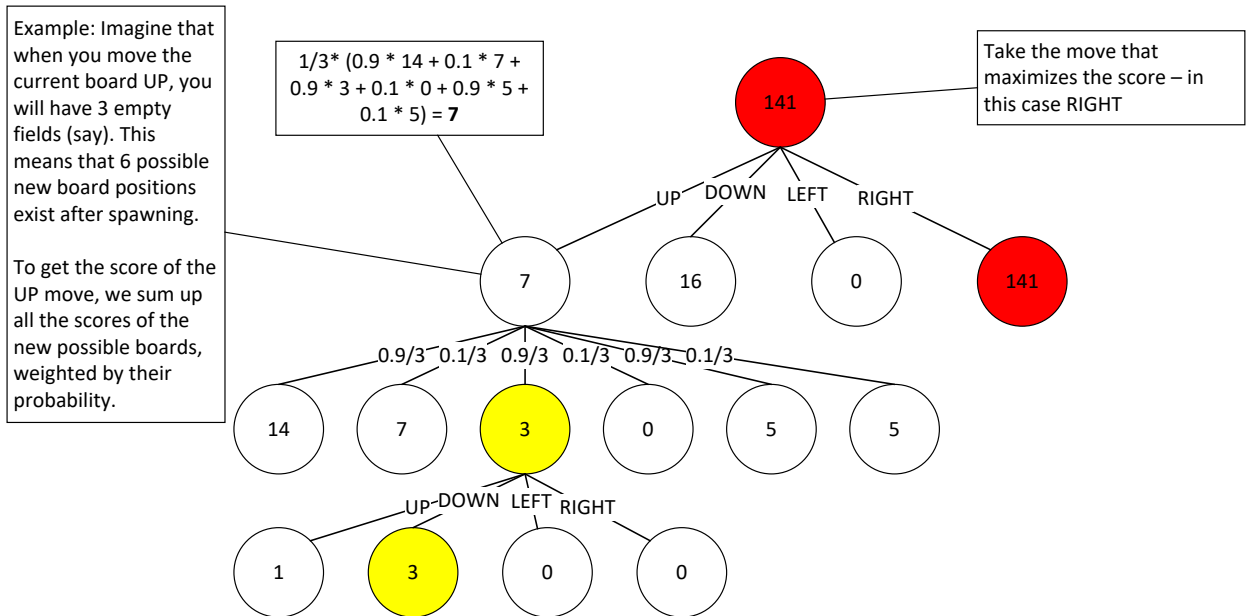
Take the move that maximizes the score – in this case RIGHT

Figure 1: A probabilistic search tree for 2048.