# Playing and solving 2048 optimally using AI

Project Plan

Rayan Miah

Final Year Project

2022/23

---

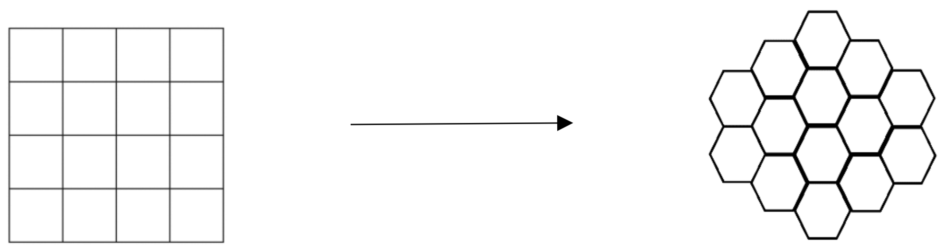
Supervised by: Yun Kuen Cheung

Department of Computer Science

Royal Holloway, University of London

## Abstract:

In this project I plan to create a playable game of 2048, that gives you hints if the player wishes for them, these hints are to be generated by an AI and will be the best possible move for the player to make. I also plan to create an option in the game where you can watch the AI play the game itself to see how high of a score it can get. If you don't know what 2048 is, it is a game where you are given a four by four square where random tiles of two and four spawn. The player then needs to input either left, right, up or down to slide all tiles in that direction. If two matching tiles collide, they combine to make a single tile of their combined values. So, when a four tile and another four tile collide they make a single eight tile. The goal of the game is to get a single tile to 2048, however you can go much further than this albeit it is very difficult, to get a score this high you need a combination of both luck and strategy. It may seem like there isn't much strategy to the game, but this is not true. To achieve the highest score possible there are many heuristics that will have to be favoured by the AI. Two that I know of include: always keep the highest numbered tile in one the corners of the grid and the other is a penalty for having non-monotonic rows/columns, so a row/column of [8 16 32 64] is valued higher than [32 8 16 64]. There may be other heuristics that can be used to further optimize the AI which I will implement to further optimise the AI. If I have time left over, I would like to implement a way for the AI to automatically play the browser version of the game. When it comes to making my project unique there are a few ideas that I can try to implement. These include making the game in a different sized grid, so larger than four by four for example, the rules of the game would need to be changed slightly for this to happen as making the grid larger with the same rules would most likely make the game easier. This could be done by increasing the number of tiles that can spawn every move or by decreasing the chance of higher valued tiles of spawning. The other idea is to make the grid a different shape other than a square, a hexagon for example, this would make the game quite different as the player could now move in six directions instead of four. An example of this:



## Project reports/ proof of concept programs:

### Proof of concept programs:

- Simple colourful GUI with a couple active buttons that take you to different windows. This is to be used to eventually make a menu that lets the user pick between the different options the game presents.
- A 4x4 grid with tiles that can be slid in 4 directions, merging if two of the same tiles collide. This is the base of the 2048 game, and I will need an internal version of the game for the AI to work on. The grid will also need to randomly spawn new tiles every turn.
- A program which shows the AI solving a simple and pre-determined version of the grid.
- An attempt at making the 2048 game on a hexagonal grid. If possible, I would like to make the game on a different shaped grid as opposed to a different sized grid as I am aiming for distinction and changing the size of the grid is not too different from the original game.

### Project reports:

- Report on Minimax and Alpha-Beta pruning as this method can be used for 2048 and may be the method I end up using.
- Report on Expectimax as this is another method that can be used for 2048 and will most likely be the one I use.
- Report on different AI algorithms that can be used to solve 2048 including Minimax, Expectimax and machine learning approaches, this report is most likely the one I will do as there is much more to write about as opposed to the individual decision algorithms.

## Timeline:

### Term 1:

| | |
|---|---|
| Week 1: | Work on Menu GUI for the proof of concept program. |
| Week 2-3: | Work on the base game of 2048 for the proof of concept. |
| Week 4: | Study AI algorithms and their implementations into the game. |
| Week 5-6: | Work on the proof of concept program where the AI plays a pre-determined version of the game. |
| Week 7: | Study optimisation techniques and apply them in the proof of concept. |
| Week 8: | Write report on different AI algorithms and their uses. |
| Week 9: | Attempt to make the hexagonal 2048 proof of concept program. |
| Week 10-11: | Prepare for interim report and presentation. |

**Term 2:**

| | |
|---|---|
| Week 1-2: | Evaluate how the final code should look and what should be included. |
| Week 3-4: | Start putting the proof of concepts together using proper SE practices. |
| Week 5-7: | Re-evaluate final project results and allow time to complete everything if not done already. |
| Week 8-10 | Prepare for final report. |

## Risks and mitigation:

Every project has risks that can occur when you least expect them to. This section will go over the different problems that may pop up and how I will mitigate them.

**Hardware failure:**
It is possible that my computer at home may run into problems causing me to lose a lot or all of my progress on this project. This can happen at any time and therefore instead of taking the approach of making regular backups etc. I should use the GitLab directory the university has provided for me. With this I can always push my work onto the directory, including reports, proof of concept programs, my main programs and diaries to make sure that I always have access to them even when I am not working from home.

**Not giving enough time for tasks:**
It is very important that I do not leave my tasks too late as some may take a long time to do. If this happens I will fall behind schedule and it will be hard to follow my plan thoroughly. To prevent this from happening I need to make sure to keep to my project timeline as this will be a somewhat clear indicator for how long I should be spending on each task. If some tasks take longer than expected, I should make time for them and try to get the easier ones done in a quicker time frame.

**AI algorithm risks:**
I may not leave enough time for the AI algorithms if I get stuck on their implementation and actually getting them to work. To combat this, I should study a couple algorithms I know I want to use and in particular learn how to get them in my code and working. I can then add the implementation of the AI into the game afterwards. To make sure this is done I should leave the AI algorithms proof of concept I decided to do for the middle of the first term and then do the distinctive hexagon proof of concept before my reports.

**Efficiency/Optimisation problems:**
In my program I may come to a point where the game runs very slowly when the AI is playing, which leads to it looking less flashy and the AI not actually being very smart as the obvious solution would be to look less moves ahead. To make sure I do not go with this approach, I should study different ways to make the algorithm more optimised and make sure to have as many of these methods in from the start.

## Bibliography:

[1] The Dstl Biscuit Book – *Artificial Intelligence, Data Science and (mostly) Machine Learning* – 1st Edition revised v1_2
https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/850129/The_Dstl_Biscuit_Book_WEB.pdf

[2] Code Bullet. (2018, July 6th). *AI learns to play 2048*. [Video]. Youtube.
https://www.youtube.com/watch?v=1g1HCYTX3Rg&t=593s

[3] *What is the optimal algorithm for the game 2048*. Stack Overflow. Blog post with many authors all commenting different techniques they have used with AI to beat the game.
https://stackoverflow.com/questions/22342854/what-is-the-optimal-algorithm-for-the-game-2048

[4] *2048 AI*. An online working example of 2048 being played by an AI that achieves winning scores very often.
https://aj-r.github.io/2048-AI/