CITS3001 Algorithms and Artificial Intelligence

Threes Bot - Artificial Intelligence Project

Mitchell Pomery (21130887)
Kieran Hannigan (21151118)

May 30, 2014

Introduction

Threes[1] is a simple mobile game where you are given a four by four grid occupied by tiles. The aim is to merge tiles until the board is full and no more merges are possible, or until there are no tiles left to be added to the board. Each move (left, right, up or down) moves each tile one space on the grid if it is free, or merges it with the tile it moves into if conditions are met. You are only able to merge a one tile with a two tile, and any other tile with one of the same number. We were tasked with the creation of an artificial intelligence to play the game Threes, aiming to get the highest score possible given a board and a list of upcoming tiles.

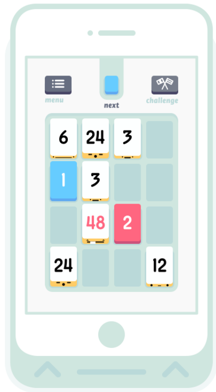

Figure 1: Threes Mobile Game Promotional Image

We were required to investigate algorithms and implimentations to create an artificial intelligence, or bot, to play Threes with the intent of getting as high a score as possible. The specifications for the bot stated three main points that were considered during the design and implimentation.

1. All upcoming tiles are known
2. Tile placement is deterministic
3. Inputs will be large

Design Choices

Programming Language

Python was chosen as the initial implementation language due to the rapid prototyping of different algorithms and data structures that it facilitates. Its handling of memory is convenient, and its formatting requirements make it easy to read and intuit - valuable attributes when codeveloping an artificial intelligence. Subsequently, we began a rewrite of the agent to C. This allowed us to evaluate the strength an weaknesses of each language - an analysis which would have to be conducted if the code was to be hard optimised for production. Whilst python is conducive to comprehensible code, C outperforms any high level language when it comes to space and speed requirements. We reprogrammed the board mechanics, priority queue, and greedy algorithm in C for evaluation. The result was verbose, but faster.

AI Considerations

To make our artificial intelligence achieve the highest score, we need to score each board layout against several criteria. These criteria are essentially questions that can be asked to determine how good the board is, and therefore can be used to rank several boards. The boards score was the initial measure used to rank different board combinations, due to the fact we are aiming for the maximum board score. The number of empty locations on the board is a useful measure that we considered. Less tiles on the board means that we can add more tiles, and hence more points, without merging tiles to achieve a higher score.

The location of the high number tiles is important in a boards configuration. Having the high number tiles in the central four squares on a board means that they are stopping the lower tiles from residing next to each other. Smoothness of the board was consideed after reading a Stack Overflow thread on 2048 artificial intelligences[3]. Smooth board layouts mean that high numbered tiles are clustered in one corner and These two criteria are similar as both relate to the layout of the board, so they are be converted into a single measure.

Algorithms

Making Moves

There are two ways the moves can be implimented in the bot, either each possible move seperately, or by rotating the board and treating all moves as the one direction. In our python implimentation we employ the latter of the two algorithms, as it allows us to reduce the amount of code written. However rotating a sixteen by sixteen board can take several operations, so to avoid this, instead of rotating the board, we rotate the co-ordinates. In our C experiment, we move the board using loops and array operations. We also experimented with large integer bitboards. A 64 bit bitboard allows for each tile to be encoded in a 4 bit nybble. Unfortunately this means a maximum tile size of 12288 which we predicted would be readily achievable by a fully optimised algorithm. Bitboards are optimal because they provide the CPU with a memory unit that it can make extremely efficient shift and addition operations on. Unfortunately this advanatage is largely lost when using multiple boards or 128 bit boards are used – as would be required for the tile size – especially on machines with 32 bit registers. We opted to use more portible code here.

Naive

Initially an naive algorithm was implimented to play Threes one move at a time. It would look at the four possible moves, left, right, up and down, and take the one that gives it the highest score. The naive algorithm is useful as a heuristic later in the A* implimentation. In C, we provided a simple "count the zeros" heuristic to the naive algorithm. We found that by minimising the tiles on the board instead of chasing high tiles, the algorithm produces a better final result.

Minimax

Both minimax and expectiminimax implimentations were investigated due to the similarities between Threes and 2048, and the already existing 2048 AI by Matt Overlan[2]. In 2048 the next tile is placed randomly, while the specifications for this project state that the next tile will be placed in the lexographically lowest location. This difference means that the Threes bot does not end up playing against anything – it becomes a deterministic problem as opposed to a stochastic one – and so minimax is not a relevant algorithm.

A Star

A*, by definition, finds the least-cost path from the initial state to a goal state providing that its algorithms are correct and admissible. It is easy to devise monotonic heuristics for Threes, making A* particularly promising. The goal state in a game of Threes is the highest scoring full board, meaning that the algorithm needs to avoid filling the board for as long as possible.

Bibliography

[1] THREES - A tiny puzzle that grows on you. 2014. THREES - A tiny puzzle that grows on you. [ONLINE] Available at: http://asherv.com/threes/. [Accessed 28 May 2014].

[2] 2048. 2014. 2048. [ONLINE] Available at: http://ov3y.github.io/2048-AI/. [Accessed 28 May 2014].

[3] logic - What is the optimal algorithm for the game, 2048? - Stack Overflow. 2014. logic - What is the optimal algorithm for the game, 2048? - Stack Overflow. [ONLINE] Available at: http://stackoverflow.com/questions/22342854/what-is-the-optimal-algorithm-for-the-game-2048. [Accessed 29 May 2014].