COS 226

2048 Assignment: Method Descriptions and Resources

/* written by colton wolk */

`size()` This method returns the instance variable *n* which keeps track of how many numbered tiles are in the array. Numbered tiles are also known as occupied positions or non-zero elements.

`weightedScore()` This method returns the cached instance variable *weighted* which is calculated by multiplying each position in the array of weights by the corresponding position in the board, and summing those values.

`snakeScore()` This score is calculated by examining the tiles above, below, to the left, and to the right, of the largest tile in the board. The closer each tile is to the previous one in the given row or column, the higher the value that is added to the score.

`penalty()` This method goes through each tile and adds the difference between the tile and each of its neighbors to the penalty. A high penalty is bad—this happens when large numbered tiles are scattered around the grid. In other words, when similarly valued tiles are close to each other, the difference is lower so the penalty is less.

`mergeTiles()` This method performs the merges using helper methods. It works by going through each row or column (depending on direction) and finding the first numbered tile and assigning it into a temp variable. From there, it continues along, skipping blank tiles until it reaches another numbered tile. If the tiles have the same value, one is reset to zero and the other is doubled (again, which one depends on the direction). If they are different, the temp value is reassigned to equal the new tile and it continues along this pattern. In a 4x4 grid a maximum of two merges can take place in a single row or column, even if all of the values are the same. (e.g., a row of {2, 2, 2, 2} becomes {4, 4, 0, 0}, NOT {8, 0, 0, 0})

`swipeTiles()` This method performs the swipes using helper methods. AFTER merges are performed, the method maintains a pointer that moves in the opposite direction that the swipe is in (e.g., if swipe = down then pointer moves from bottom to top). As the pointer works its way through, it swaps the first blank tile seen with the first numbered tile seen after the blank one. Then, the pointer is reset to the position after the swapped numbered tile. Sorting doesn't work here, since the order of the numbered tiles must be maintained.

`Solver2048` This class is extremely similar to the **`Solver`** class in 8-Puzzle. The difference here, however, is that a high priority value is a good thing. With 8-Puzzle, we made the decision to use MinPQ since we want the board with the lowest priority (lowest manhattan or hamming distance). In this case, I had the option of either changing MinPQ to MaxPQ (which seemed to make sense but violates the principles of A* search I think) OR subtracting the priority from 0 which would make the lower priority the better one. I elected to do the latter option.

Resources (all accessed in May 2018):

8-Puzzle: http://www.cs.princeton.edu/courses/archive/spring18/cos226/assignments/8puzzle/index.html

2048 actual game: http://gabrielecirulli.github.io/2048/
2048 GitHub: https://github.com/gabrielecirulli/2048

Mathematics behind 2048: http://www.science4all.org/article/2048-game/
More mathematics: http://jdlm.info/articles/2017/12/10/counting-states-enumeration-2048.html
Paper on mathematics and snake pattern: https://www.ripublication.com/aama17/aamav12n1_01.pdf

AI solver 1: http://blog.datumbox.com/using-artificial-intelligence-to-solve-the-2048-game-java-code/
AI solver 2: https://sandipanweb.wordpress.com/2017/03/06/using-minimax-with-alpha-beta-pruning-and-heuristic-evaluation-to-solve-2048-game-with-computer/
AI solver 3: http://www.randalolson.com/2015/04/27/artificial-intelligence-has-crushed-all-human-records-in-2048-heres-how-the-ai-pulled-it-off/
AI solver 4: https://adaickalavan.github.io/portfolio/artifical_intelligence_ansaf_salleb-aouissi/
AI stack overflow discussion: https://stackoverflow.com/questions/22342854/what-is-the-optimal-algorithm-for-the-game-2048

Tufts 2048 C++ assignment (Go Jumbos!): https://www.cs.tufts.edu/comp/11/hw/proj2/2048.pdf

Weight matrix and penalty info: http://iamkush.me/an-artificial-intelligence-for-the-2048-game/