EECS 293 - Programming Assignment 6

Christian Bonnell and Jason Shin

February 28, 2019

Solver Class:

- Instance Variables:
  - Board
- Methods:
  - solveCurrentBoard

Board Class:

- Instance Variables:
  - 2D-Array of Pebbles
- Methods:
  - addPebble - add pebble to specified location on board

Pebble Class:

- Instance Variables:
  - Color
  - Iteration Value
  - Left Pebble
  - Right Pebble
  - Up Pebble
  - Down Pebble
- Methods:
  - flipColor - if black, set Color to white, else set Color to black

Algorithm 1 Solve Current Board: Belongs to Solver Class

1: **procedure** SOLVECURRENTBOARD
2: **Input:** A board representation that contains a 2 dimensional array of pebble representations.
3: **Output:** Number of iterations that pebble representations are replaced and whether or not there are any black pebble representations remaining.
4: $Q \leftarrow \emptyset$ (Queue)
5: *blackPebbles* $\leftarrow 0$
6: **for** each *pebble* on board **do**
7: **if** *pebble* is white **then**
8: add *pebble* to Q
9: set *pebble* iteration level to *0*
10: **else**
11: *blackPebbles* $\leftarrow$ *blackPebbles* $+ 1$
12: *currentIteration* $\leftarrow 0$
13: **while** Q is not empty **do**
14: *currentPebble* $\leftarrow$ pebble deqeued from Q
15: *currentIteration* $\leftarrow$ *currentPebble*'s *iterationLevel*
16: Change color of *currentPebble*'s black neighbors to white and enqueue them to Q with iteration level of one greater than currentPebble's iteration level.
17: Decrement *blackPebbles* by 1 for each black neighbor flipped
18: **return** *currentIteration* and True if blackPebbles > 0, False otherwise

Runtime:

- First loop iterates through each pebble at most once, this O(n)
- Second loop goes through entirety of queue, which holds at most all the pieces on the board since pieces cannot be added to the queue more than once, this is O(n)
- O(n) + O(n) = O(n), where n is the number of pieces on the board

Correctness:

- The correctness of the algorithm can be justified by checking every type of board configuration
  - Board with no pebbles - Since there are no pebbles, no pebbles will be added to Q, so blackPebbles will stay at 0 and iteration will stay at 0. Therefore, the algorithm will return 0 and False.
  - Board with only white pebbles - Since there are only white pebbles, every pebble will be added to Q with iteration level 0, and blackPebbles will stay at 0. currentIteration will never be set to anything other than 0, since there are no black pebbles. Therefore, the algorithm will return 0 and False.
  - Board with only black pebbles - Since there are only black pebbles, no pebbles will be added to Q and blackPebbles will equal the number of pebbles. Therefore, the algorithm will return 0 and True.
  - Board with black pebbles not touching any white pebbles - Since there are no black pebbles touching white pebbles, the only pebbles that would be added to Q would be white pebbles with iteration level 0 have no neighboring black pebbles. currentIteratioin will stay at 0 and blackPebbles will be greater than 0. Therefore, the algorithm will return 0 and True.
  - Board with black pebbles touching white pebbles
    * Case 1: No islands of solely back pebbles - All of the white pebbles will be added to Q and blackPebbles will start at the number of black pebbles in the beginning. Since each black pebble does not exist in an island it will eventually be added to the queue. Because of this each one will be flipped and the blackPebbles total will be reduced to 0. This will result in the game board being returned with the accurate amount of iterations and False.
    * Case 2: Exists island(s) of solely of black pebble - Identical to case 1, but because there exists these isolated black pebbles, then blackPebbles will always be greater than 0, since this island of pebbles will never be added to Q, so they can't be flipped. Therefore the algorithm will return the appropriate iteration level and True.

Class Abstraction Explanations

- Class Pebbles: The pebbles class is an abstraction that represents the physical game pieces used in the game of Gone. Each piece must have its color and the ability to be flipped to the opposite color if the correct circumstances are met. For our purposes it also includes references to its neighboring pebbles.
- Class Board: The board class is an abstraction that represents the physical game board used in Gone. It simply contains a 2-dimensional array thats size is specified by the user. We will be storing Pebbles on the board.
- Class Solver: The solver class represents the results of a single turn in Gone after all the pieces have been set on the board and the first player begins to replace black pieces with white pieces according to the rules of Gone.