* Submit pseudo-code
* State classes to which each method belongs
* Explain the abstraction your classes capture
* Write down a justification of your algorithm’s correctness, and an analysis of its running time
* Sketch a few examples of the algorithm’s operation
* Specify the format of the input and the output

|  |  |
| --- | --- |
| **Description**  This method takes a configuration of black and white pebbles on a board at a given time as an input, and returns the number of rounds the game can be played for until no more pebbles can be replaced. | |
| **Inputs**   * List of coordinates of white pebbles * 2D array of booleans, of the same width/height as the board, indicating which squares contain black pebbles | **Output**   * The result of the game, describing:   + The number of rounds which the game can proceed with the given configuration of the board   + Whether or not there are black pebbles left over at the end of the game |
| **Algorithm**  If the list of white pebbles is empty, no more changes will be made (base case)  Iterate over the array of black pebbles, to check if there are any left  Return a game result indicating that there are no more rounds, and whether or  not any black pebbles remain  Make a new list to keep track of the new white pebbles  Iterate over all of the white pebbles  Check each of the four directions around each one  If there’s a black pebble there, set that value in the array to false and  add those coordinates to the new list of white pebbles  Call the algorithm with the new list of white pebbles and the updated map of black pebbles, and store the result  Increase the number of rounds in the stored result by one, and return it | |
| **Runtime**  O(n2) - if the list of white pebbles has n elements, the complexity of the recursive part of the function will be ~n, and iterating over all of the white pebbles in each function will be n which makes the total complexity n2 | |

|  |  |
| --- | --- |
| **Description**  This method iterates over a 2D array of booleans, and returns whether or not the array contains any true values. | |
| **Inputs**   * 2D array of booleans | **Output**   * True if the input contains any true values, and false otherwise |
| **Algorithm**  Iterate over the array of arrays  Iterate over each sub-array in the list of arrays  If the element is true, return true  If no true values were found, return false | |
| **Runtime**  O(nm) | |

|  |  |
| --- | --- |
| **Description**  This method checks if there is a black pebble at a given location, and replaces it with a white pebble if there is | |
| **Inputs**   * Location of the pebble to check * List of locations of (new) white pebbles * 2D array of booleans representing locations of black pebbles | **Output**   * None |
| **Algorithm**  Check if the corresponding value in the array is true  If so, set it to false and add the new location to the list of white pebbles | |
| **Runtime**  O(1) | |