Writing a Loop in the Game 2048

Chengyao Li

July 15, 2017

1 OBJECTIVE

1.1 DESCRIPTION

The game 2048 is played on a 4 x 4 board. Players move numbers around on the board using the arrow keys: up, left, right and down. Now consider clicking the left arrow, so that a move is made to the left. This program will achieve moveLeft() function in the game 2048 to perform this move and calculate corresponding scores.

1.2 SPECIFICATION

The values in the game 2048 are kept in square array b, the size of which is b[b.length][b.length]. So moving left consists of moving elements of each row b[i]. Examples are given below to illustrate how the moveLeft function works.

1. Values get moved as far to the left as possible. (0, 8, 0, 2) becomes (8, 2, 0, 0).
2. Two neighboring equal values get added and stored as one value. This is the only time points get produced. The points produced is that one value. (4, 4, 16, 16) becomes (8, 32, 0, 0). The number of points is 8 + 32 = 40.
3. Once a merged value has been produced, it cannot participate in another merge (during this move). Thus, (4, 4, 0, 4) becomes (8, 4, 0, 0), with a point score of 8. Note that the *leftmost* two 4's are added together. It is *wrong* for this move to procedure (4, 8, 0, 0). Also, (4, 4, 4, 4) would become (8, 8, 0, 0).
4. Based on the above rules, it makes sense to process the array from beginning to end.

A function declaration is provided.

```
/*Perform a move in direction left on row b[i].
 *Return the points gained for this move (-1 if no move made).*/
public int moveLeft(int[][] b, int i){
}
```

2 ALGORITHM

2.1 INVARIANT

I try to give a precondition and postcondition below, and then the invariant.

	0		n
Q b[i]	x	?	

	0	h		n
R b[i]	merged	x	0	

	0	h		j	n
P b[i]	merged	x	0	?	

The meaning of each part of the invariant is described below.

- merged: values that have been merged and cannot be changed
- x: the next value to be merged
- 0: blanks
- ?: wait to be merged
- score: points produced in the move
- moveMade: becomes true if a position change happens

Thus I can write the corresponding loop.

```
h, j, score, moveMade,:= 0, 1, 0, false;
do   n+1-j > 0   →
     if   b[i][j] = 0   →   j := j + 1;
     □   b[i][j] ≠ 0 ∧ b[i][h] := b[i][j]   →   b[i][h], b[i][j] := b[i][h] + b[i][j], 0;
                                                score := score + b[i][h];
                                                h, j := h+1, j+1;
                                                moveMade := true;
     □   b[i][j] ≠ 0 ∧ b[i][h] ≠ b[i][j] ∧ b[i][h] = 0   →   b[i][h], b[i][j] := b[i][j], 0;
                                                              j := j+1;
     □   b[i][j] ≠ 0 ∧ b[i][h] ≠ b[i][j] ∧ b[i][h] ≠ 0   →
         if   h+1 = j   →   h, j := h+1, j+1;
         □   h+1 ≠ j   →   h := h+1;
                           b[i][h], b[i][j] := b[i][j], 0;
                           j := j+1;
                           moveMade := true;
         fi
     fi
od
```

2.2 Test Method

I test the moveLeft() method in the main() method using random examples. I randomize the scale of square matrix b to [4, 7], which means that the size of b can be modified by changing the parameters of the random method. In the meantime the values of the element in matrix b is also randomized. Thus every time we run the main() method we can test a completely new example without modifying the size and the elements of matrix b.

3 Results

3.1 Environment

- Windows 10
- Java Development Kit 1.8.0_131
- Eclipse

3.2 Example Output

I run the code in eclipse console and two example outputs are shown below.

```
0:   4   8   16   16   0
32:   4   8   32   0   0

0:   8   16   0   16   16   8   2
32:   8   32   16   8   2   0   0
```

3.3 Thoughts

At first I quickly developed the invariant diagram using what I learned from professor's class. However when I tried to write down the corresponding loop it consumed me lots of time to think about different kinds of situations. After that I gave out a loop considering only element movement and then I found it easy to add moveMade flag and the variable score and polish the whole loop. As what professor said, it really surprised me that how easy it is to write the codes with the loop invariant and the algorithm. Thanks for the classes you gave us in the last three weeks and I am looking forward to your reply if any correction is supposed to be made.