See the Assessment Guide for information on how to interpret this report.

ASSESSMENT SUMMARY

```
Compilation:  PASSED
API:          PASSED

SpotBugs:     PASSED
PMD:          PASSED
Checkstyle:   FAILED (0 errors, 1 warning)

Correctness:  51/51 tests passed
Memory:       22/22 tests passed
Timing:       104/125 tests passed

Aggregate score: 96.64%
[ Compilation: 5%, API: 5%, Style: 0%, Correctness: 60%, Timing: 10%, Memory: 20% ]
```

ASSESSMENT DETAILS

```
The following files were submitted:
----------------------------------
8.6K Mar 20 11:40 Board.java
4.5K Mar 20 11:40 Solver.java


*================================================================================
*  COMPILING
*================================================================================


% javac Board.java
*-----------------------------------------------------------

% javac Solver.java
*-----------------------------------------------------------


================================================================


Checking the APIs of your programs.
*-----------------------------------------------------------
Board:

Solver:

================================================================


*================================================================================
*  CHECKING STYLE AND COMMON BUG PATTERNS
*================================================================================


% spotbugs *.class
*-----------------------------------------------------------
```

```
================================================================


% pmd .
*-----------------------------------------------------------


================================================================


% checkstyle *.java
*-----------------------------------------------------------
[WARN] Board.java:171:9: Conditional logic can be removed. [SimplifyBooleanReturn]
Checkstyle ends with 0 errors and 1 warning.

% custom checkstyle checks for Board.java
*-----------------------------------------------------------

% custom checkstyle checks for Solver.java
*-----------------------------------------------------------


================================================================


********************************************************************************
*  TESTING CORRECTNESS
********************************************************************************

Testing correctness of Board
*-----------------------------------------------------------
Running 26 total tests.

Tests 4-7 and 14-17 rely upon toString() returning results in prescribed format.

Test 1a: check hamming() with file inputs
  * puzzle04.txt
  * puzzle00.txt
  * puzzle07.txt
  * puzzle17.txt
  * puzzle27.txt
  * puzzle2x2-unsolvable1.txt
==> passed

Test 1b: check hamming() with random n-by-n boards
  * 2-by-2
  * 3-by-3
  * 4-by-4
  * 5-by-5
  * 9-by-9
  * 10-by-10
  * 127-by-127
==> passed

Test 2a: check manhattan() with file inputs
  * puzzle04.txt
  * puzzle00.txt
  * puzzle07.txt
  * puzzle17.txt
  * puzzle27.txt
  * puzzle2x2-unsolvable1.txt
==> passed

Test 2b: check manhattan() with random n-by-n boards
  * 2-by-2
  * 3-by-3
  * 4-by-4
```

```
   * 5-by-5
   * 9-by-9
   * 10-by-10
   * 127-by-127
 ==> passed

 Test 3: check dimension() with random n-by-n boards
   * 2-by-2
   * 3-by-3
   * 4-by-4
   * 5-by-5
   * 6-by-6
 ==> passed

 Test 4a: check toString() with file inputs
   * puzzle04.txt
   * puzzle00.txt
   * puzzle06.txt
   * puzzle09.txt
   * puzzle23.txt
   * puzzle2x2-unsolvable1.txt
 ==> passed

 Test 4b: check toString() with random n-by-n boards
   * 2-by-2
   * 3-by-3
   * 4-by-4
   * 5-by-5
   * 9-by-9
   * 10-by-10
   * 127-by-127
 ==> passed

 Test 5a: check neighbors() with file inputs
   * puzzle04.txt
   * puzzle00.txt
   * puzzle06.txt
   * puzzle09.txt
   * puzzle23.txt
   * puzzle2x2-unsolvable1.txt
 ==> passed

 Test 5b: check neighbors() with random n-by-n boards
   * 2-by-2
   * 3-by-3
   * 4-by-4
   * 5-by-5
   * 9-by-9
   * 10-by-10
   * 127-by-127
 ==> passed

 Test 6a: check neighbors() of neighbors() with file inputs
   * puzzle04.txt
   * puzzle00.txt
   * puzzle06.txt
   * puzzle09.txt
   * puzzle23.txt
   * puzzle2x2-unsolvable1.txt
 ==> passed

 Test 6b: check neighbors() of neighbors() with random n-by-n boards
   * 2-by-2
   * 3-by-3
   * 4-by-4
   * 5-by-5
   * 9-by-9
   * 10-by-10
 ==> passed
```

```
Test 7a: check twin() with file inputs
  * puzzle04.txt
  * puzzle00.txt
  * puzzle06.txt
  * puzzle09.txt
  * puzzle23.txt
  * puzzle2x2-unsolvable1.txt
==> passed

Test 7b: check twin() with random n-by-n boards
  * 2-by-2
  * 3-by-3
  * 4-by-4
  * 5-by-5
  * 9-by-9
  * 10-by-10
==> passed

Test 8a: check isGoal() with file inputs
  * puzzle00.txt
  * puzzle04.txt
  * puzzle16.txt
  * puzzle06.txt
  * puzzle09.txt
  * puzzle23.txt
  * puzzle2x2-unsolvable1.txt
  * puzzle3x3-unsolvable1.txt
  * puzzle3x3-00.txt
  * puzzle4x4-00.txt
==> passed

Test 8b: check isGoal() on n-by-n goal boards
  * 2-by-2
  * 3-by-3
  * 4-by-4
  * 5-by-5
  * 6-by-6
  * 100-by-100
==> passed

Test 9: check that two Board objects can be created at the same time
  * random 3-by-3 and 3-by-3 boards
  * random 4-by-4 and 4-by-4 boards
  * random 2-by-2 and 2-by-2 boards
  * random 3-by-3 and 4-by-4 boards
  * random 4-by-4 and 3-by-3 boards
==> passed

Test 10a: check equals()
  * reflexive
  * symmetric
  * transitive
  * argument is null
  * argument is of type String
  * argument is of type UncastableString
  * Board object stored in a variable of type Object
==> passed

Test 10b: check correctness of equals() on random n-by-n boards
  * n = 2
  * n = 3
  * n = 4
  * 5 <= n < 10
==> passed

Test 10c: check equals() when board sizes m and n are different
  * m = 4, n = 5
  * m = 2, n = 5
```

```
   * m = 5, n = 3
   * m = 2, n = 3
   * m = 3, n = 2
 ==> passed

 Test 11: check that Board is immutable by changing argument array after
          construction and making sure Board does not mutate
 ==> passed

 Test 12: check that Board is immutable by testing whether methods
          return the same value, regardless of order in which called
   * puzzle10.txt
   * puzzle20.txt
   * puzzle30.txt
   * 2-by-2
   * 3-by-3
   * 4-by-4
 ==> passed

 Test 13: check dimension() on a board that is kth neighbor of a board
   * 0th neighbor of puzzle27.txt
   * 1st neighbor of puzzle27.txt
   * 2nd neighbor of puzzle27.txt
   * 13th neighbor of puzzle27.txt
   * 13th neighbor of puzzle00.txt
   * 13th neighbor of puzzle2x2-unsolvable1.txt
 ==> passed

 Test 14: check hamming() on a board that is kth neighbor of a board
   * 0th neighbor of puzzle27.txt
   * 1st neighbor of puzzle27.txt
   * 2nd neighbor of puzzle27.txt
   * 13th neighbor of puzzle27.txt
   * 13th neighbor of puzzle00.txt
   * 13th neighbor of puzzle2x2-unsolvable1.txt
 ==> passed

 Test 15: check manhattan() on a board that is a kth neighbor of a board
   * 0th neighbor of puzzle27.txt
   * 1st neighbor of puzzle27.txt
   * 2nd neighbor of puzzle27.txt
   * 13th neighbor of puzzle27.txt
   * 13th neighbor of puzzle00.txt
   * 13th neighbor of puzzle2x2-unsolvable1.txt
 ==> passed

 Test 16: check hamming() on a board that is a kth twin of a board
   * 0th twin of puzzle27.txt
   * 1st twin of puzzle27.txt
   * 2nd twin of puzzle27.txt
   * 13th twin of puzzle27.txt
   * 13th twin of puzzle00.txt
   * 13th twin of puzzle2x2-unsolvable1.txt
 ==> passed

 Test 17: check manhattan() on a board that is a kth twin of a board
   * 0th twin of puzzle27.txt
   * 1st twin of puzzle27.txt
   * 2nd twin of puzzle27.txt
   * 13th twin of puzzle27.txt
   * 13th twin of puzzle00.txt
   * 13th twin of puzzle2x2-unsolvable1.txt
 ==> passed


 Total: 26/26 tests passed!


 ================================================================
```

```
*********************************************************************************
*  MEMORY
*********************************************************************************

Analyzing memory of Board
*-----------------------------------------------------------
Running 10 total tests.

Memory usage of an n-by-n board
[ must be at most 4n^2 + 32n + 64 bytes ]


              n       student (bytes)    reference (bytes)
---------------------------------------------------------
=> passed     2           144                 128
=> passed     3           208                 192
=> passed     4           256                 240
=> passed     8           576                 560
=> passed    12          1024                1008
=> passed    16          1600                1584
=> passed    20          2304                2288
=> passed    37          6872                6856
=> passed    72         23104               23088
=> passed   120         61504               61488
==> 10/10 tests passed

Total: 10/10 tests passed!

Student   memory = 4.00 n^2 + 32.00 n + 64.00   (R^2 = 1.000)
Reference memory = 4.00 n^2 + 32.00 n + 48.00   (R^2 = 1.000)


=================================================================



*********************************************************************************
*  TESTING CORRECTNESS (substituting reference Board)
*********************************************************************************

Testing correctness of Solver
*-----------------------------------------------------------
Running 25 total tests.

Test 1a: check moves() with file inputs
  * puzzle00.txt
  * puzzle01.txt
  * puzzle02.txt
  * puzzle03.txt
  * puzzle04.txt
  * puzzle05.txt
  * puzzle06.txt
  * puzzle07.txt
  * puzzle08.txt
  * puzzle09.txt
  * puzzle10.txt
  * puzzle11.txt
  * puzzle12.txt
  * puzzle13.txt
==> passed

Test 1b: check solution() with file inputs
  * puzzle00.txt
  * puzzle01.txt
  * puzzle02.txt
  * puzzle03.txt
  * puzzle04.txt
  * puzzle05.txt
  * puzzle06.txt
```

```
    * puzzle07.txt
    * puzzle08.txt
    * puzzle09.txt
    * puzzle10.txt
    * puzzle11.txt
    * puzzle12.txt
    * puzzle13.txt
  ==> passed

  Test 2a: check moves() with more file inputs
    * puzzle14.txt
    * puzzle15.txt
    * puzzle16.txt
    * puzzle17.txt
    * puzzle18.txt
    * puzzle19.txt
    * puzzle20.txt
    * puzzle21.txt
    * puzzle22.txt
    * puzzle23.txt
    * puzzle24.txt
    * puzzle25.txt
    * puzzle26.txt
    * puzzle27.txt
    * puzzle28.txt
    * puzzle29.txt
    * puzzle30.txt
    * puzzle31.txt
  ==> passed

  Test 2b: check solution() with more file inputs
    * puzzle14.txt
    * puzzle15.txt
    * puzzle16.txt
    * puzzle17.txt
    * puzzle18.txt
    * puzzle19.txt
    * puzzle20.txt
    * puzzle21.txt
    * puzzle22.txt
    * puzzle23.txt
    * puzzle24.txt
    * puzzle25.txt
    * puzzle26.txt
    * puzzle27.txt
    * puzzle28.txt
    * puzzle29.txt
    * puzzle30.txt
    * puzzle31.txt
  ==> passed

  Test 3a: check moves() with random solvable n-by-n boards
    * 1000 random 3-by-3 boards that are exactly 1 move from goal
    * 1000 random 3-by-3 boards that are exactly 2 moves from goal
    * 1000 random 3-by-3 boards that are exactly 3 moves from goal
    * 1000 random 3-by-3 boards that are exactly 4 moves from goal
    * 1000 random 3-by-3 boards that are exactly 5 moves from goal
    * 1000 random 3-by-3 boards that are exactly 6 moves from goal
    * 1000 random 3-by-3 boards that are exactly 7 moves from goal
    * 1000 random 3-by-3 boards that are exactly 8 moves from goal
    * 1000 random 3-by-3 boards that are exactly 9 moves from goal
    * 1000 random 3-by-3 boards that are exactly 10 moves from goal
    * 1000 random 3-by-3 boards that are exactly 11 moves from goal
    * 1000 random 3-by-3 boards that are exactly 12 moves from goal
  ==> passed

  Test 3b: check solution() with random solvable n-by-n boards
    * 1000 random 3-by-3 boards that are exactly 1 move from goal
    * 1000 random 3-by-3 boards that are exactly 2 moves from goal
```

```
   * 1000 random 3-by-3 boards that are exactly 3 moves from goal
   * 1000 random 3-by-3 boards that are exactly 4 moves from goal
   * 1000 random 3-by-3 boards that are exactly 5 moves from goal
   * 1000 random 3-by-3 boards that are exactly 6 moves from goal
   * 1000 random 3-by-3 boards that are exactly 7 moves from goal
   * 1000 random 3-by-3 boards that are exactly 8 moves from goal
   * 1000 random 3-by-3 boards that are exactly 9 moves from goal
   * 1000 random 3-by-3 boards that are exactly 10 moves from goal
   * 1000 random 3-by-3 boards that are exactly 11 moves from goal
   * 1000 random 3-by-3 boards that are exactly 12 moves from goal
 ==> passed

 Test 4: create two Solver objects at the same time
   * puzzle04.txt and puzzle04.txt
   * puzzle00.txt and puzzle04.txt
   * puzzle04.txt and puzzle00.txt
 ==> passed

 Test 5a: call isSolvable() with file inputs
   * puzzle01.txt
   * puzzle03.txt
   * puzzle04.txt
   * puzzle17.txt
   * puzzle3x3-unsolvable1.txt
   * puzzle3x3-unsolvable2.txt
   * puzzle4x4-unsolvable.txt
 ==> passed

 Test 5b: call isSolvable() on random n-by-n boards
   * 100 random 2-by-2 boards
 ==> passed

 Test 6: check moves() on unsolvable puzzles
   * puzzle2x2-unsolvable1.txt
   * puzzle2x2-unsolvable2.txt
   * puzzle3x3-unsolvable1.txt
   * puzzle3x3-unsolvable2.txt
   * puzzle4x4-unsolvable.txt
 ==> passed

 Test 7: check solution() on unsolvable puzzles
   * puzzle2x2-unsolvable1.txt
   * puzzle2x2-unsolvable2.txt
   * puzzle3x3-unsolvable1.txt
   * puzzle3x3-unsolvable2.txt
   * puzzle4x4-unsolvable.txt
 ==> passed

 Test 8a: check that Solver is immutable by testing whether methods
          return the same value, regardless of order in which called
   * puzzle3x3-00.txt
   * puzzle3x3-01.txt
   * puzzle3x3-05.txt
   * puzzle3x3-10.txt
   * random 2-by-2 solvable boards
 ==> passed

 Test 8b: check that Solver is immutable by testing whether methods
          return the same value, regardless of order in which called
   * puzzle3x3-unsolvable1.txt
   * puzzle3x3-unsolvable2.txt
   * puzzle4x4-unsolvable.txt
   * random 2-by-2 unsolvable boards
 ==> passed

 Test 9a: check that equals() method in Board is called
   * puzzle04.txt
   * puzzle05.txt
   * puzzle10.txt
```

```
==> passed

Test 9b: check that equals() method in Board is called only
         with an argument of type Board
  * puzzle00.txt
  * puzzle04.txt
  * puzzle05.txt
  * puzzle10.txt
==> passed

Test 9c: check that equals() method in Board is called only
         with a neighbor of a neighbor as an argument
  * puzzle00.txt
  * puzzle04.txt
  * puzzle05.txt
  * puzzle10.txt
  * puzzle27.txt
==> passed

Test 10: check that constructor throws exception if board is null
==> passed

Test 11a: check moves() with 2-by-2 file inputs
  * puzzle2x2-00.txt
  * puzzle2x2-01.txt
  * puzzle2x2-02.txt
  * puzzle2x2-03.txt
  * puzzle2x2-04.txt
  * puzzle2x2-05.txt
  * puzzle2x2-06.txt
==> passed

Test 11b: check solution() with 2-by-2 file inputs
  * puzzle2x2-00.txt
  * puzzle2x2-01.txt
  * puzzle2x2-02.txt
  * puzzle2x2-03.txt
  * puzzle2x2-04.txt
  * puzzle2x2-05.txt
  * puzzle2x2-06.txt
==> passed

Test 12a: check moves() with 3-by-3 file inputs
  * puzzle3x3-00.txt
  * puzzle3x3-01.txt
  * puzzle3x3-02.txt
  * puzzle3x3-03.txt
  * puzzle3x3-04.txt
  * puzzle3x3-05.txt
  * puzzle3x3-06.txt
  * puzzle3x3-07.txt
  * puzzle3x3-08.txt
  * puzzle3x3-09.txt
  * puzzle3x3-10.txt
  * puzzle3x3-11.txt
  * puzzle3x3-12.txt
  * puzzle3x3-13.txt
  * puzzle3x3-14.txt
  * puzzle3x3-15.txt
  * puzzle3x3-16.txt
  * puzzle3x3-17.txt
  * puzzle3x3-18.txt
  * puzzle3x3-19.txt
  * puzzle3x3-20.txt
  * puzzle3x3-21.txt
  * puzzle3x3-22.txt
  * puzzle3x3-23.txt
  * puzzle3x3-24.txt
  * puzzle3x3-25.txt
```

```
   * puzzle3x3-26.txt
   * puzzle3x3-27.txt
   * puzzle3x3-28.txt
   * puzzle3x3-29.txt
   * puzzle3x3-30.txt
 ==> passed

 Test 12b: check solution() with 3-by-3 file inputs
   * puzzle3x3-00.txt
   * puzzle3x3-01.txt
   * puzzle3x3-02.txt
   * puzzle3x3-03.txt
   * puzzle3x3-04.txt
   * puzzle3x3-05.txt
   * puzzle3x3-06.txt
   * puzzle3x3-07.txt
   * puzzle3x3-08.txt
   * puzzle3x3-09.txt
   * puzzle3x3-10.txt
   * puzzle3x3-11.txt
   * puzzle3x3-12.txt
   * puzzle3x3-13.txt
   * puzzle3x3-14.txt
   * puzzle3x3-15.txt
   * puzzle3x3-16.txt
   * puzzle3x3-17.txt
   * puzzle3x3-18.txt
   * puzzle3x3-19.txt
   * puzzle3x3-20.txt
   * puzzle3x3-21.txt
   * puzzle3x3-22.txt
   * puzzle3x3-23.txt
   * puzzle3x3-24.txt
   * puzzle3x3-25.txt
   * puzzle3x3-26.txt
   * puzzle3x3-27.txt
   * puzzle3x3-28.txt
   * puzzle3x3-29.txt
   * puzzle3x3-30.txt
 ==> passed

 Test 13a: check moves() with 4-by-4 file inputs
   * puzzle4x4-00.txt
   * puzzle4x4-01.txt
   * puzzle4x4-02.txt
   * puzzle4x4-03.txt
   * puzzle4x4-04.txt
   * puzzle4x4-05.txt
   * puzzle4x4-06.txt
   * puzzle4x4-07.txt
   * puzzle4x4-08.txt
   * puzzle4x4-09.txt
   * puzzle4x4-10.txt
   * puzzle4x4-11.txt
   * puzzle4x4-12.txt
   * puzzle4x4-13.txt
   * puzzle4x4-14.txt
   * puzzle4x4-15.txt
   * puzzle4x4-16.txt
   * puzzle4x4-17.txt
   * puzzle4x4-18.txt
   * puzzle4x4-19.txt
   * puzzle4x4-20.txt
   * puzzle4x4-21.txt
   * puzzle4x4-22.txt
   * puzzle4x4-23.txt
   * puzzle4x4-24.txt
   * puzzle4x4-25.txt
   * puzzle4x4-26.txt
```

```
   * puzzle4x4-27.txt
   * puzzle4x4-28.txt
   * puzzle4x4-29.txt
   * puzzle4x4-30.txt
 ==> passed

 Test 13b: check solution() with 4-by-4 file inputs
   * puzzle4x4-00.txt
   * puzzle4x4-01.txt
   * puzzle4x4-02.txt
   * puzzle4x4-03.txt
   * puzzle4x4-04.txt
   * puzzle4x4-05.txt
   * puzzle4x4-06.txt
   * puzzle4x4-07.txt
   * puzzle4x4-08.txt
   * puzzle4x4-09.txt
   * puzzle4x4-10.txt
   * puzzle4x4-11.txt
   * puzzle4x4-12.txt
   * puzzle4x4-13.txt
   * puzzle4x4-14.txt
   * puzzle4x4-15.txt
   * puzzle4x4-16.txt
   * puzzle4x4-17.txt
   * puzzle4x4-18.txt
   * puzzle4x4-19.txt
   * puzzle4x4-20.txt
   * puzzle4x4-21.txt
   * puzzle4x4-22.txt
   * puzzle4x4-23.txt
   * puzzle4x4-24.txt
   * puzzle4x4-25.txt
   * puzzle4x4-26.txt
   * puzzle4x4-27.txt
   * puzzle4x4-28.txt
   * puzzle4x4-29.txt
   * puzzle4x4-30.txt
 ==> passed

 Test 14a: check moves() with random solvable n-by-n boards
   * 100 random 2-by-2 boards that are <= 6 moves from goal
   * 200 random 3-by-3 boards that are <= 20 moves from goal
   * 200 random 4-by-4 boards that are <= 20 moves from goal
   * 200 random 5-by-5 boards that are <= 20 moves from goal
 ==> passed

 Test 14b: check solution() with random solvable n-by-n boards
   * 100 random 2-by-2 boards that are <= 6 moves from goal
   * 200 random 3-by-3 boards that are <= 20 moves from goal
   * 200 random 4-by-4 boards that are <= 20 moves from goal
   * 200 random 5-by-5 boards that are <= 20 moves from goal
 ==> passed


 Total: 25/25 tests passed!


 ================================================================
 ********************************************************************************
 *  MEMORY (substituting reference Board)
 ********************************************************************************

 Analyzing memory of Solver
 *-----------------------------------------------------------
 Running 12 total tests.

 Maximum allowed time per puzzle is 5.0 seconds.
 Maximum allowed memory per puzzle = 200000000 bytes.
```

```
Test 1: Measure memory of Solver.

               filename   moves   memory
-----------------------------------------
=> passed  puzzle10.txt      10      416
=> passed  puzzle15.txt      15      344
=> passed  puzzle20.txt      20      128
=> passed  puzzle25.txt      25      128
=> passed  puzzle30.txt      30      128
=> passed  puzzle35.txt      35      152
==> 6/6 tests passed



Test 2: Measure memory of MinPQ.

                            deep             max          ending
               filename     memory            size            size
-------------------------------------------------------------------
=> passed  puzzle10.txt      23600              34              33
=> passed  puzzle15.txt      29864              52              51
=> passed  puzzle20.txt     294080             800             799
=> passed  puzzle25.txt    2398176            6492            6491
=> passed  puzzle30.txt    9574432           25274           25273
=> passed  puzzle35.txt  142973744          417915          417914
==> 6/6 tests passed


Total: 12/12 tests passed!


================================================================



*********************************************************************************
*  TIMING (substituting reference Board)
*********************************************************************************

Timing Solver
*-----------------------------------------------------------
Running 125 total tests.

Maximum allowed time per puzzle is 5.0 seconds.

Test 1: Measure CPU time and check correctness

               filename   moves    n  seconds
-----------------------------------------------
=> passed  puzzle20.txt      20    3     0.01
=> passed  puzzle22.txt      22    3     0.01
=> passed  puzzle21.txt      21    3     0.01
=> passed  puzzle23.txt      23    3     0.01
=> passed  puzzle24.txt      24    3     0.01
=> passed  puzzle25.txt      25    3     0.01
=> passed  puzzle27.txt      27    3     0.01
=> passed  puzzle29.txt      29    3     0.01
=> passed  puzzle26.txt      26    3     0.01
=> passed  puzzle28.txt      28    3     0.02
=> passed  puzzle30.txt      30    3     0.03
=> passed  puzzle31.txt      31    3     0.02
=> passed  puzzle39.txt      39    4     0.26
=> passed  puzzle41.txt      41    5     0.07
=> passed  puzzle34.txt      34    4     0.15
=> passed  puzzle37.txt      37    4     0.16
=> passed  puzzle44.txt      44    5     0.15
=> passed  puzzle32.txt      32    4     1.23
=> passed  puzzle35.txt      35    4     0.39
```

```
=> passed  puzzle33.txt      33    4     0.37
=> passed  puzzle43.txt      43    4     0.76
=> passed  puzzle46.txt      46    4     0.77
=> passed  puzzle40.txt      40    4     0.82
=> passed  puzzle36.txt      36    4     1.55
=> passed  puzzle45.txt      45    4     1.01
==> 25/25 tests passed
```

Test 2: Count MinPQ operations

```
               filename   insert()          delMin()
-------------------------------------------------------
=> passed  puzzle20.txt       1986              1187
=> passed  puzzle22.txt       4689              2779
=> FAILED  puzzle21.txt       6620   (1.1x)     3901   (1.1x)
=> FAILED  puzzle23.txt      10078   (1.1x)     6006   (1.1x)
=> passed  puzzle24.txt       8438              5071
=> passed  puzzle25.txt      15972              9481
=> passed  puzzle27.txt      17382             10462
=> FAILED  puzzle29.txt      30806   (1.5x)    18547   (1.5x)
=> passed  puzzle26.txt      19125             11425
=> passed  puzzle28.txt      36653             21988
=> passed  puzzle30.txt      63765             38492
=> passed  puzzle31.txt      64014             38659
=> FAILED  puzzle39.txt     580501   (4.6x)   283785   (4.6x)
=> passed  puzzle41.txt     121471             51664
=> FAILED  puzzle34.txt     345920   (1.3x)   166577   (1.3x)
=> FAILED  puzzle37.txt     374342   (1.3x)   179492   (1.3x)
=> passed  puzzle44.txt     291178            128926
=> FAILED  puzzle32.txt    2499196   (2.7x)  1189911   (2.7x)
=> passed  puzzle35.txt     814237            396323
=> passed  puzzle33.txt     842367            401748
=> passed  puzzle43.txt    1723691            830984
=> passed  puzzle46.txt    1713607            849967
=> passed  puzzle40.txt    1823729            890752
=> passed  puzzle36.txt    3322561           1597683
=> passed  puzzle45.txt    2273677           1110433
==> 18/25 tests passed
```

Test 3: Count Board operations (that should not get called)

```
               filename    hamming()   toString()
-------------------------------------------------
=> passed  puzzle20.txt            0            0
=> passed  puzzle22.txt            0            0
=> passed  puzzle21.txt            0            0
=> passed  puzzle23.txt            0            0
=> passed  puzzle24.txt            0            0
=> passed  puzzle25.txt            0            0
=> passed  puzzle27.txt            0            0
=> passed  puzzle29.txt            0            0
=> passed  puzzle26.txt            0            0
=> passed  puzzle28.txt            0            0
=> passed  puzzle30.txt            0            0
=> passed  puzzle31.txt            0            0
=> passed  puzzle39.txt            0            0
=> passed  puzzle41.txt            0            0
=> passed  puzzle34.txt            0            0
=> passed  puzzle37.txt            0            0
=> passed  puzzle44.txt            0            0
=> passed  puzzle32.txt            0            0
=> passed  puzzle35.txt            0            0
=> passed  puzzle33.txt            0            0
=> passed  puzzle43.txt            0            0
=> passed  puzzle46.txt            0            0
```

```
=> passed  puzzle40.txt            0            0
=> passed  puzzle36.txt            0            0
=> passed  puzzle45.txt            0            0
==> 25/25 tests passed
```

Test 4a: Count Board operations (that should get called)

	filename	Board()		equals()		manhattan()
=> passed	puzzle20.txt	3167		3154		3173
=> passed	puzzle22.txt	7462		7455		7468
=> FAILED	puzzle21.txt	10515	(1.1x)	10505	(1.1x)	10521
=> FAILED	puzzle23.txt	16078	(1.1x)	16068	(1.1x)	16084
=> passed	puzzle24.txt	13503		13490		13509
=> passed	puzzle25.txt	25447		25437		25453
=> passed	puzzle27.txt	27838		27828		27844
=> FAILED	puzzle29.txt	49347	(1.5x)	49337	(1.5x)	49353
=> passed	puzzle26.txt	30544		30537		30550
=> passed	puzzle28.txt	58635		58622		58641
=> passed	puzzle30.txt	102251		102244		102257
=> passed	puzzle31.txt	102667		102657		102673
=> FAILED	puzzle39.txt	864280	(4.6x)	864270	(4.6x)	864286
=> passed	puzzle41.txt	173129		173116		173135
=> FAILED	puzzle34.txt	512491	(1.3x)	512484	(1.3x)	512497
=> FAILED	puzzle37.txt	553828	(1.3x)	553818	(1.3x)	553834
=> passed	puzzle44.txt	420098		420085		420104
=> FAILED	puzzle32.txt	3689101	(2.7x)	3689088	(2.7x)	3689107
=> passed	puzzle35.txt	1210554		1210541		1210560
=> passed	puzzle33.txt	1244109		1244099		1244115
=> passed	puzzle43.txt	2554669		2554659		2554675
=> passed	puzzle46.txt	2563568		2563558		2563574
=> passed	puzzle40.txt	2714475		2714468		2714481
=> passed	puzzle36.txt	4920238		4920225		4920244
=> passed	puzzle45.txt	3384104		3384094		3384110

==> 18/25 tests passed

Test 4b: count Board operations (that should get called),
rejecting if doesn't adhere to stricter caching limits

	filename	Board()		equals()		manhattan()	
=> passed	puzzle20.txt	3167		3154		3173	
=> passed	puzzle22.txt	7462		7455		7468	
=> FAILED	puzzle21.txt	10515	(1.1x)	10505	(1.1x)	10521	(1.1x)
=> FAILED	puzzle23.txt	16078	(1.1x)	16068	(1.1x)	16084	(1.1x)
=> passed	puzzle24.txt	13503		13490		13509	
=> passed	puzzle25.txt	25447		25437		25453	
=> passed	puzzle27.txt	27838		27828		27844	
=> FAILED	puzzle29.txt	49347	(1.5x)	49337	(1.5x)	49353	(1.5x)
=> passed	puzzle26.txt	30544		30537		30550	
=> passed	puzzle28.txt	58635		58622		58641	
=> passed	puzzle30.txt	102251		102244		102257	
=> passed	puzzle31.txt	102667		102657		102673	
=> FAILED	puzzle39.txt	864280	(4.6x)	864270	(4.6x)	864286	(4.6x)
=> passed	puzzle41.txt	173129		173116		173135	
=> FAILED	puzzle34.txt	512491	(1.3x)	512484	(1.3x)	512497	(1.3x)
=> FAILED	puzzle37.txt	553828	(1.3x)	553818	(1.3x)	553834	(1.3x)
=> passed	puzzle44.txt	420098		420085		420104	
=> FAILED	puzzle32.txt	3689101	(2.7x)	3689088	(2.7x)	3689107	(2.7x)
=> passed	puzzle35.txt	1210554		1210541		1210560	
=> passed	puzzle33.txt	1244109		1244099		1244115	
=> passed	puzzle43.txt	2554669		2554659		2554675	
=> passed	puzzle46.txt	2563568		2563558		2563574	
=> passed	puzzle40.txt	2714475		2714468		2714481	
=> passed	puzzle36.txt	4920238		4920225		4920244	

```
=> passed  puzzle45.txt    3384104             3384094             3384110
==> 18/25 tests passed


Total: 104/125 tests passed!


================================================================
```