BFS Test (H1)

Based on the codes from the book:
Artificial Intelligence : A Modern Approach
The copyrights of the codes belong to
Ravi Mohan, Peter Norvig, Stuart Russell, Ciaran O'Reilly

```
initialState         = NQueensBoard(8)
successorFunction = NQueenSuccessorFunction()
goalTest               = NQueensGoalTest()
```

```
package aima.test.search.searches;

import java.util.List;

import junit.framework.TestCase;
import aima.search.framework.Problem;
import aima.search.framework.Search;
import aima.search.framework.SearchAgent;
import aima.search.framework.TreeSearch;
import aima.search.nqueens.NQueensBoard;
import aima.search.nqueens.NQueensGoalTest;
import aima.search.nqueens.NQueensSuccessorFunction;
import aima.search.uninformed.BreadthFirstSearch;

public class BreadthFirstSearchTest extends TestCase {
        public void testBreadthFirstSuccesfulSearch() throws Exception {
                Problem problem = new Problem(new NQueensBoard(8),
                                new NQueensSuccessorFunction(), new NQueensGoalTest());
                Search search = new BreadthFirstSearch(new TreeSearch());
                SearchAgent agent = new SearchAgent(problem, search);
                List actions = agent.getActions();
                assertCorrectPlacement(actions);
                assertEquals("1965", agent.getInstrumentation().getProperty(
                                "nodesExpanded"));

                problem = new Problem(new NQueensBoard(3),
                                new NQueensSuccessorFunction(), new NQueensGoalTest());
                agent = new SearchAgent(problem, search);
                actions = agent.getActions();
                assertEquals(0, actions.size());
                assertEquals("6", agent.getInstrumentation().getProperty(
                                "nodesExpanded"));
        }

        public void testBreadthFirstUnSuccesfulSearch() throws Exception {
                Problem problem = new Problem(new NQueensBoard(3),
                                new NQueensSuccessorFunction(), new NQueensGoalTest());
                Search search = new BreadthFirstSearch(new TreeSearch());
                SearchAgent agent = new SearchAgent(problem, search);
                List actions = agent.getActions();
                assertEquals(0, actions.size());
                assertEquals("6", agent.getInstrumentation().getProperty(
                                "nodesExpanded"));
        }

        private void assertCorrectPlacement(List actions) {
                assertEquals(8, actions.size());
                assertEquals("placeQueenAt 0  0", actions.get(0));
                assertEquals("placeQueenAt 1  4", actions.get(1));
                assertEquals("placeQueenAt 2  7", actions.get(2));
                assertEquals("placeQueenAt 3  5", actions.get(3));
                assertEquals("placeQueenAt 4  2", actions.get(4));
                assertEquals("placeQueenAt 5  6", actions.get(5));
                assertEquals("placeQueenAt 6  1", actions.get(6));
                assertEquals("placeQueenAt 7  3", actions.get(7));
        }

}
```