CS3105 Report - P1

190022658

Overview

1. This practical required us to code multiple parts. They were:
    a. Part 1 - Check potential solutions are correct
    b. Part 2 - Implement and evaluate DFS (depth-first search) for Black Hole
    c. Part 3 - Add a solver for a variant of Black Hole called 'Worm Hole'
2. The code submitted completes all of the parts, and in both variations (recursively and iteratively).

Design

- The starter code had already implemented a BHLayout class and a BHMain class. These already accomplished generating random layouts and storing layouts.

BHLayout

- **isAdjacent(int, int)** : This method simply checks if two cards are adjacent to each other. There's also error checking in case the method receives a card which is not a card but the end of the list. There's use of the '%' operator to make things easier.

Game

- This is an interface that has some methods that are required by the classes that implement it.
- This allows there to be only one method for depth first search as the methods used in the depth first search method are in the interface.
- This also allows me to maintain distinction in Black hole and Worm hole.

BHGame

- This class implements **Game** and extends **BHLayout**.
- There are constructors which just initialise the instance using the super method.

- **play(int, int)** - This method checks if the move (argument) is valid, and then plays it out by changing the holecard and removing the top card of the pile. It returns a boolean depending if the move was successful.
- **gameOver()** - This iterates over the layout and checks if all the piles are empty (which is the condition for game over).
- **possibleMoves()** - Iterates over the piles and if the topcard of the pile is adjacent to the hole card then adds it to the list of possible moves and returns it.

WHGame

- This class implements **Game** and extends **BHLayout**.
- There is a new attribute called **wormhole**.
- There are constructors which just initialise the instance using the super method and initialise the wormhole.
- **topCard(int)** - This method is overridden here, if the pile is -1 then we assume that the user is talking about the wormhole.
- **gameOver()** - Checks if all piles are empty and the wormhole too.
- **play(int, int)** - In a wormhole game, there are three types of move that can be made.
    - If the move is from **wormhole to hole** then check if the wormhole is not empty and if the wormhole card and hole card are adjacent.
    - If the move is from **pile to wormhole** then check if the wormhole is empty and if the pile is not empty (this is very important in this case because we check topCard == -card and topCard(pile) returns -1 in edge cases).
    - And if the move is from **pile to hole** then it is the same as Black hole play.
- **possibleMoves()** - Here if the card in the wormhole is adjacent to hole card then adds that to the list of possible moves, then iterates through the piles and adds the ones adjacent to the holecard to the list of possible moves. If the wormhole is empty and if the pile has a card then moving the card to the wormhole is also added to the list of possible moves.

Move

- This class is a data structure to just store the pile and card involved in a move.

DFSState

- This class is just a way to store a (BH/WH)Game instance with a list of moves played to get to that instance.

BHMain

- Notice that the Game interface comes in handy here.

- **checkSolution(Game, ArrayList<Integer>)** - This is part1 and part of part3. Here I create a copy of the game instance passed as argument (to not make any changes to the original), then just play the game out move by move. If for a move the play method returns false then that move is not possible, if the game is over at this stage then the method returns true (since the practical specifications say to ignore the extra moves). Otherwise the method returns a false. This works for both BlackHole and WormHole.
- **depthfirstsearch(Game)** - This is the iterative implementation. I use a stack to store the states for this implementation. This is a basic iterative algorithm using a stack. If there is no solution the method returns null.

Design choices

Had originally planned to create a tuple-like data structure.

```
public class Tuple<X, Y> {
        public X x;
        public Y y;

        public Tuple(X x, Y y) {
        this.x = x;
        this.y = y;
        }
}
```

This could work as a generic substitute to store moves, Tuple<Integer, Integer>. And store the game state and the moves together.

The resulting code looked very unreadable, so I decided to make the Move class and **DFSState** class.

Another choice was in the beginning, I could just add methods to the BHLayout class. That didn't look like the best option since I knew that I'd have to implement a Worm hole game too and that they have different implementations. So I decided I'd extend BHLayout twice, once for BHGame and then for WHGame.

Testing

Stacscheck output

```
stacscheck /cs/studres/CS3105/Practicals/P1/Tests
Testing CS3105 P1
- Looking for submission in a directory called 'src': found in current directory
 * BUILD TEST - build-all : pass
 * COMPARISON TEST - Part1/prog-run-01Correct.out : pass
 * COMPARISON TEST - Part1/prog-run-02Correct.out : pass
 * COMPARISON TEST - Part1/prog-run-03Wrong.out : pass
 * COMPARISON TEST - Part1/prog-run-04Trivial.out : pass
 * COMPARISON TEST - Part1/prog-run-05Onecard.out : pass
 * COMPARISON TEST - Part1/prog-run-1-5-3-2-1.txt.out : pass
 * COMPARISON TEST - Part1/prog-run-251-26-9-3-10.txt.out : pass
 * COMPARISON TEST - Part1/prog-run-251-5-3-2-2.txt.out : pass
 * COMPARISON TEST - Part1/prog-run-51-26-9-3-10.txt.out : pass
 * COMPARISON TEST - Part1/prog-run-51-5-3-2-1.txt.out : pass
 * COMPARISON TEST - Part1/prog-run-851-26-9-3-10.txt.out : pass
 * COMPARISON TEST - Part1/prog-run-901-26-9-3-10.txt.out : pass
 * COMPARISON TEST - Part1/prog-run-wrong1-51-26-9-3-10.txt.out : pass
 * COMPARISON TEST - Part1/prog-run-wrong2-51-26-9-3-10.txt.out : pass
 * COMPARISON TEST - Part1/prog-run-wrong3-51-26-9-3-10.txt.out : pass
 * COMPARISON TEST - Part1/prog-run-wrong4-51-26-9-3-10.txt.out : pass
 * COMPARISON TEST - Part1/prog-run-wrong5-51-26-9-3-10.txt.out : pass
 * COMPARISON TEST - Part1/prog-run-wrong6-51-26-9-3-10.txt.out : pass
 * COMPARISON TEST - Part2/Solve/prog-run-01Solvable.out : pass
 * COMPARISON TEST - Part2/Solve/prog-run-03Solvable.out : pass
 * COMPARISON TEST - Part2/Solve/prog-run-04Impossible.out : pass
 * COMPARISON TEST - Part2/Solve/prog-run-05Impossible.out : pass
 * COMPARISON TEST - Part2/Solve/prog-run-1-23-6-4-5.txt.out : pass
 * COMPARISON TEST - Part2/Solve/prog-run-1-39-10-4-8.txt.out : pass
 * COMPARISON TEST - Part2/Solve/prog-run-1-5-3-2-1.txt.out : pass
 * COMPARISON TEST - Part2/Solve/prog-run-101-26-9-3-10.txt.out : pass
 * COMPARISON TEST - Part2/Solve/prog-run-101-5-3-2-1.txt.out : pass
 * COMPARISON TEST - Part2/Solve/prog-run-151-39-10-4-8.txt.out : pass
 * COMPARISON TEST - Part2/Solve/prog-run-201-23-6-4-5.txt.out : pass
 * COMPARISON TEST - Part2/Solve/prog-run-251-23-6-4-5.txt.out : pass
 * COMPARISON TEST - Part2/Solve/prog-run-251-26-9-3-10.txt.out : pass
 * COMPARISON TEST - Part2/Solve/prog-run-251-5-3-2-2.txt.out : pass
 * COMPARISON TEST - Part2/Solve/prog-run-301-26-9-3-10.txt.out : pass
 * COMPARISON TEST - Part2/Solve/prog-run-301-5-3-2-2.txt.out : pass
 * COMPARISON TEST - Part2/Solve/prog-run-401-23-6-4-5.txt.out : pass
 * COMPARISON TEST - Part2/Solve/prog-run-51-26-9-3-10.txt.out : pass
 * COMPARISON TEST - Part2/Solve/prog-run-51-26-9-3-6.txt.out : pass
 * COMPARISON TEST - Part2/Solve/prog-run-51-5-3-2-1.txt.out : pass
 * COMPARISON TEST - Part2/Solve/prog-run-751-51-13-4-17.txt.out : pass
 * COMPARISON TEST - Part2/Solve/prog-run-851-26-9-3-10.txt.out : pass
 * COMPARISON TEST - Part2/Solve/prog-run-901-26-9-3-10.txt.out : pass
 * COMPARISON TEST - Part2/SolveAndCheck/prog-run-01Solvable.out : pass
 * COMPARISON TEST - Part2/SolveAndCheck/prog-run-1-23-6-4-5.txt.out : pass
 * COMPARISON TEST - Part2/SolveAndCheck/prog-run-1-5-3-2-1.txt.out : pass
 * COMPARISON TEST - Part2/SolveAndCheck/prog-run-101-26-9-3-10.txt.out : pass
 * COMPARISON TEST - Part2/SolveAndCheck/prog-run-251-26-9-3-10.txt.out : pass
 * COMPARISON TEST - Part2/SolveAndCheck/prog-run-251-5-3-2-2.txt.out : pass
 * COMPARISON TEST - Part2/SolveAndCheck/prog-run-301-26-9-3-10.txt.out : pass
 * COMPARISON TEST - Part2/SolveAndCheck/prog-run-301-5-3-2-2.txt.out : pass
 * COMPARISON TEST - Part2/SolveAndCheck/prog-run-51-26-9-3-10.txt.out : pass
 * COMPARISON TEST - Part2/SolveAndCheck/prog-run-51-5-3-2-1.txt.out : pass
 * COMPARISON TEST - Part2/SolveAndCheck/prog-run-851-26-9-3-10.txt.out : pass
 * COMPARISON TEST - Part2/SolveAndCheck/prog-run-901-26-9-3-10.txt.out : pass
 * COMPARISON TEST - Part2Hard/Solve/prog-run-02Solvable.out : pass
 * COMPARISON TEST - Part2Hard/Solve/prog-run-1-51-13-4-17.txt.out : pass
 * COMPARISON TEST - Part2Hard/Solve/prog-run-151-51-13-4-17.txt.out : pass
 * COMPARISON TEST - Part2Hard/Solve/prog-run-401-26-9-3-10.txt.out : pass
 * COMPARISON TEST - Part2Hard/Solve/prog-run-451-39-10-4-8.txt.out : pass
 * COMPARISON TEST - Part2Hard/Solve/prog-run-51-51-13-4-17.txt.out : !! 60 second timeout reached
 fail
 --- expected output ---
 1
 --- submission output ---
 ... Terminating due to timeout
 ---

 * COMPARISON TEST - Part2Hard/Solve/prog-run-651-51-13-4-17.txt.out : pass
 * COMPARISON TEST - Part2Hard/Solve/prog-run-701-51-13-4-17.txt.out : pass
 * COMPARISON TEST - Part2Hard/Solve/prog-run-951-26-9-3-10.txt.out : pass
 * COMPARISON TEST - Part2Hard/Solve/prog-run-951-51-13-4-17.txt.out : !! 60 second timeout reached
 fail
 --- expected output ---
 1
 --- submission output ---
 ... Terminating due to timeout
 ---

 * COMPARISON TEST - Part2Hard/SolveAndCheck/prog-run-02Solvable.out : pass
 * COMPARISON TEST - Part2Hard/SolveAndCheck/prog-run-1-51-13-4-17.txt.out : pass
 * COMPARISON TEST - Part2Hard/SolveAndCheck/prog-run-151-51-13-4-17.txt.out : pass
 * COMPARISON TEST - Part2Hard/SolveAndCheck/prog-run-51-51-13-4-17.txt.out : !! 60 second
timeout reached
 fail
 --- expected output ---
 true
 --- submission output ---
 ... Terminating due to timeout
 ---

 * COMPARISON TEST - Part2Hard/SolveAndCheck/prog-run-651-51-13-4-17.txt.out : pass
 * COMPARISON TEST - Part2Hard/SolveAndCheck/prog-run-701-51-13-4-17.txt.out : pass
 * COMPARISON TEST - Part2Hard/SolveAndCheck/prog-run-951-26-9-3-10.txt.out : pass
 * COMPARISON TEST - Part2Hard/SolveAndCheck/prog-run-951-51-13-4-17.txt.out : !! 60 second
timeout reached
```

```
fail
--- expected output ---
true
--- submission output ---
... Terminating due to timeout
---

* COMPARISON TEST - Part3/BHCheckClones/prog-run-01Correct.out : pass
* COMPARISON TEST - Part3/BHCheckClones/prog-run-02Correct.out : pass
* COMPARISON TEST - Part3/BHCheckClones/prog-run-03Wrong.out : pass
* COMPARISON TEST - Part3/BHCheckClones/prog-run-04Trivial.out : pass
* COMPARISON TEST - Part3/BHCheckClones/prog-run-05Onecard.out : pass
* COMPARISON TEST - Part3/BHCheckClones/prog-run-1-5-3-2-1.txt.out : pass
* COMPARISON TEST - Part3/BHCheckClones/prog-run-251-26-9-3-10.txt.out : pass
* COMPARISON TEST - Part3/BHCheckClones/prog-run-251-5-3-2-2.txt.out : pass
* COMPARISON TEST - Part3/BHCheckClones/prog-run-51-26-9-3-10.txt.out : pass
* COMPARISON TEST - Part3/BHCheckClones/prog-run-51-5-3-2-1.txt.out : pass
* COMPARISON TEST - Part3/BHCheckClones/prog-run-851-26-9-3-10.txt.out : pass
* COMPARISON TEST - Part3/BHCheckClones/prog-run-901-26-9-3-10.txt.out : pass
* COMPARISON TEST - Part3/BHCheckClones/prog-run-wrong1-51-26-9-3-10.txt.out : pass
* COMPARISON TEST - Part3/BHCheckClones/prog-run-wrong2-51-26-9-3-10.txt.out : pass
* COMPARISON TEST - Part3/BHCheckClones/prog-run-wrong3-51-26-9-3-10.txt.out : pass
* COMPARISON TEST - Part3/BHCheckClones/prog-run-wrong4-51-26-9-3-10.txt.out : pass
* COMPARISON TEST - Part3/BHCheckClones/prog-run-wrong5-51-26-9-3-10.txt.out : pass
* COMPARISON TEST - Part3/BHCheckClones/prog-run-wrong6-51-26-9-3-10.txt.out : pass
* COMPARISON TEST - Part3/WHCheck/prog-run-1-5-3-2-2.txt.out : pass
* COMPARISON TEST - Part3/WHCheck/prog-run-101-5-3-2-1.txt.out : pass
* COMPARISON TEST - Part3/WHCheck/prog-run-151-23-6-4-5.txt.out : pass
* COMPARISON TEST - Part3/WHCheck/prog-run-251-26-9-3-6.txt.out : pass
* COMPARISON TEST - Part3/WHCheck/prog-run-551-23-6-4-5.txt.out : pass
* COMPARISON TEST - Part3/WHCheck/prog-run-wrong1-251-26-9-3-6.txt.out : pass
* COMPARISON TEST - Part3/WHCheck/prog-run-wrong2-251-26-9-3-6.txt.out : pass
* COMPARISON TEST - Part3/WHCheck/prog-run-wrong3-251-26-9-3-6.txt.out : pass
* COMPARISON TEST - Part3/WHCheck/prog-run-wrong4-251-26-9-3-6.txt.out : pass
* COMPARISON TEST - Part3/WHSolve/prog-run-01Solvable.out : pass
* COMPARISON TEST - Part3/WHSolve/prog-run-1-23-6-4-5.txt.out : pass
* COMPARISON TEST - Part3/WHSolve/prog-run-1-26-9-3-6.txt.out : pass
* COMPARISON TEST - Part3/WHSolve/prog-run-1-5-3-2-1.txt.out : pass
* COMPARISON TEST - Part3/WHSolve/prog-run-1-5-3-2-2.txt.out : pass
* COMPARISON TEST - Part3/WHSolve/prog-run-1-51-13-4-2.txt.out : pass
* COMPARISON TEST - Part3/WHSolve/prog-run-101-47-12-4-5.txt.out : pass
* COMPARISON TEST - Part3/WHSolve/prog-run-101-5-3-2-1.txt.out : pass
* COMPARISON TEST - Part3/WHSolve/prog-run-101-51-13-4-3.txt.out : pass
* COMPARISON TEST - Part3/WHSolve/prog-run-151-23-6-4-5.txt.out : pass
* COMPARISON TEST - Part3/WHSolve/prog-run-251-26-9-3-6.txt.out : pass
* COMPARISON TEST - Part3/WHSolve/prog-run-301-23-6-4-5.txt.out : pass
* COMPARISON TEST - Part3/WHSolve/prog-run-351-23-6-4-5.txt.out : pass
* COMPARISON TEST - Part3/WHSolve/prog-run-401-26-9-3-10.txt.out : pass
* COMPARISON TEST - Part3/WHSolve/prog-run-51-26-9-3-6.txt.out : pass
```

```
 * COMPARISON TEST - Part3/WHSolve/prog-run-51-51-13-4-2.txt.out : pass
 * COMPARISON TEST - Part3/WHSolve/prog-run-551-23-6-4-5.txt.out : pass
 * COMPARISON TEST - Part3/WHSolve/prog-run-651-23-6-4-5.txt.out : pass
 * COMPARISON TEST - Part3/WHSolve/prog-run-851-47-12-4-5.txt.out : pass
 * COMPARISON TEST - Part3/WHSolve/prog-run-951-26-9-3-6.txt.out : pass
 * COMPARISON TEST - Part3/WHSolve/prog-run-951-51-13-4-2.txt.out : pass
 * COMPARISON TEST - Part3/WHSolveAndCheck/prog-run-01Solvable.out : pass
 * COMPARISON TEST - Part3/WHSolveAndCheck/prog-run-1-23-6-4-5.txt.out : pass
 * COMPARISON TEST - Part3/WHSolveAndCheck/prog-run-1-26-9-3-6.txt.out : pass
 * COMPARISON TEST - Part3/WHSolveAndCheck/prog-run-1-5-3-2-1.txt.out : pass
 * COMPARISON TEST - Part3/WHSolveAndCheck/prog-run-1-5-3-2-2.txt.out : pass
 * COMPARISON TEST - Part3/WHSolveAndCheck/prog-run-101-5-3-2-1.txt.out : pass
 * COMPARISON TEST - Part3/WHSolveAndCheck/prog-run-151-23-6-4-5.txt.out : pass
 * COMPARISON TEST - Part3/WHSolveAndCheck/prog-run-251-26-9-3-6.txt.out : pass
 * COMPARISON TEST - Part3/WHSolveAndCheck/prog-run-301-23-6-4-5.txt.out : pass
 * COMPARISON TEST - Part3/WHSolveAndCheck/prog-run-351-23-6-4-5.txt.out : pass
 * COMPARISON TEST - Part3/WHSolveAndCheck/prog-run-401-26-9-3-10.txt.out : pass
 * COMPARISON TEST - Part3/WHSolveAndCheck/prog-run-51-26-9-3-6.txt.out : pass
 * COMPARISON TEST - Part3/WHSolveAndCheck/prog-run-551-23-6-4-5.txt.out : pass
 * COMPARISON TEST - Part3/WHSolveAndCheck/prog-run-651-23-6-4-5.txt.out : pass
 * COMPARISON TEST - Part3Hard/WHSolveAndCheckHarder/prog-run-201-51-13-4-17.txt.out : pass
 * COMPARISON TEST - Part3Hard/WHSolveAndCheckHarder/prog-run-251-26-9-3-10.txt.out : pass
 * COMPARISON TEST - Part3Hard/WHSolveAndCheckHarder/prog-run-301-26-9-3-10.txt.out : pass
 * COMPARISON TEST - Part3Hard/WHSolveAndCheckHarder/prog-run-351-26-9-3-6.txt.out : pass
 * COMPARISON TEST - Part3Hard/WHSolveAndCheckHarder/prog-run-651-26-9-3-6.txt.out : pass
 * COMPARISON TEST - Part3Hard/WHSolveAndCheckHarder/prog-run-701-26-9-3-10.txt.out : pass
 * COMPARISON TEST - Part3Hard/WHSolveAndCheckHarder/prog-run-701-26-9-3-6.txt.out : pass
 * COMPARISON TEST - Part3Hard/WHSolveAndCheckHarder/prog-run-851-23-6-4-5.txt.out : pass
 * COMPARISON TEST - Part3Hard/WHSolveAndCheckHarder/prog-run-901-26-9-3-10.txt.out : pass
 * COMPARISON TEST - Part3Hard/WHSolveAndCheckHarder/prog-run-901-26-9-3-6.txt.out : pass
 * COMPARISON TEST - Part3Hard/WHSolveHarder/prog-run-201-51-13-4-17.txt.out : pass
 * COMPARISON TEST - Part3Hard/WHSolveHarder/prog-run-251-26-9-3-10.txt.out : pass
 * COMPARISON TEST - Part3Hard/WHSolveHarder/prog-run-301-26-9-3-10.txt.out : pass
 * COMPARISON TEST - Part3Hard/WHSolveHarder/prog-run-351-26-9-3-6.txt.out : pass
 * COMPARISON TEST - Part3Hard/WHSolveHarder/prog-run-501-26-9-3-6.txt.out : pass
 * COMPARISON TEST - Part3Hard/WHSolveHarder/prog-run-651-26-9-3-6.txt.out : pass
 * COMPARISON TEST - Part3Hard/WHSolveHarder/prog-run-701-26-9-3-10.txt.out : pass
 * COMPARISON TEST - Part3Hard/WHSolveHarder/prog-run-701-26-9-3-6.txt.out : pass
 * COMPARISON TEST - Part3Hard/WHSolveHarder/prog-run-751-23-6-4-5.txt.out : pass
 * COMPARISON TEST - Part3Hard/WHSolveHarder/prog-run-851-23-6-4-5.txt.out : pass
 * COMPARISON TEST - Part3Hard/WHSolveHarder/prog-run-901-26-9-3-10.txt.out : pass
 * COMPARISON TEST - Part3Hard/WHSolveHarder/prog-run-901-26-9-3-6.txt.out : pass
152 out of 156 tests passed
```

- Note that the tests failed because of timeout, they still do give results. They take a bit longer.

General testing

Part 1

```
java BHMain CHECK /cs/studres/CS3105/Practicals/P1/Tests/Instances/BlackHole/1-5-3-2-1.txt
0 6 0 4 0 5 0 3 0 2 0 1 12 09
true
```

Notice that the input after the valid moves are ignored. This is the same for CHECKWORM.

Part 2

There are 4 cases that fail due to timeout but they do still pass

```
java BHMain SOLVE /cs/studres/CS3105/Practicals/P1/Tests/Instances/BlackHole/51-51-13-4-17.txt
1 16 41 5 27 16 52 12 51 13 24 16 49 9 35 15 21 7 20 12 45 14 31 12 32 13 5 1 43 11 44 11 17
4 16 6 2 0 42 5 15 4 14 0 13 9 38 6 50 10 36 10 9 0 34 7 46 13 6 3 33 7 8 1 48 4 23 5 11 6 12 8
39 3 25 8 26 11 40 15 28 14 3 10 30 1 29 9 4 2 18 15 19 3 7 2 47 8 22 2 10 14 37
```

Empty Layout

layout.txt

```
1 1 5
1
-1
-1
-1
-1
-1
```

Terminal

```
java BHMain SOLVE layout.txt
1
java BHMain SOLVEWORM layout.txt
1
```

Evaluation

- Adding a variable that stores the number of cards in play in BHLayout and decrementing it every move **to the hole** will reduce the complexity of iterating all the piles and checking their sizes to just compare the number of cards in play with 0.
- Increase in the number of piles, increase winnability of Black Hole as there are more options to play. At the same time, decrease in the number of piles, also decrease the time taken to find a solution (if there exists one). Note that in this case, the number of cards is assumed to be constant.

- 18, 24, 30 and 36 are each the number of cards used each time (they are numbers with high numbers of factors). The output shows that the previous statement holds true. The stateCount is the number of moves considered. The i in the for loop is the number of piles. They are in increasing order.

```
for i in 1 2 3 6 9 18; do java BHMain GEN 4 18 6 3 $i > layout.txt; time java BHMain SOLVE layout.txt;
done

stateCount: 2
0

real	0m0.123s
user	0m0.132s
sys	0m0.039s

stateCount: 3
0

real	0m0.111s
user	0m0.137s
sys	0m0.031s

stateCount: 9
0

real	0m0.112s
user	0m0.136s
sys	0m0.033s

stateCount: 55
1 3 8 5 9 2 2 2 13 4 6 4 5 1 18 4 11 5 16 1 15 3 14 2 7 0 12 1 17 3 4 0 3 0 10
real	0m0.130s
user	0m0.174s
sys	0m0.032s

stateCount: 38
1 7 6 8 13 6 8 7 15 5 2 6 3 5 16 2 9 0 14 2 7 4 18 4 11 3 12 1 5 3 4 1 17 0 10
real	0m0.133s
user	0m0.154s
```

```
sys	0m0.053s

stateCount: 154
1 16 6 10 5 13 18 8 13 15 8 11 9 14 2 7 15 9 14 2 7 12 12 4 11 5 16 6 3 3 4 1 17 0 10
real	0m0.134s
user	0m0.170s
sys	0m0.041s
```

```
for i in 1 2 3 4 6 8 12 24; do java BHMain GEN 4 24 6 3 $i > layout.txt; time java BHMain SOLVE layout.txt; done
```

```
stateCount: 2
0

real	0m0.124s
user	0m0.136s
sys	0m0.033s

stateCount: 3
0

real	0m0.122s
user	0m0.141s
sys	0m0.028s

stateCount: 9
0

real	0m0.111s
user	0m0.138s
sys	0m0.030s

stateCount: 12
0

real	0m0.112s
user	0m0.128s
sys	0m0.042s

stateCount: 55
1 3 8 5 9 2 2 2 13 4 6 4 5 1 18 4 11 5 16 1 15 3 14 2 7 0 12 1 17 3 4 0 3 0 10
real	0m0.134s
user	0m0.174s
sys	0m0.037s

stateCount: 57
1 7 8 7 15 6 2 6 3 1 14 3 9 3 4 2 5 5 18 2 7 0 6 0 13 4 12 4 11 5 16 1 17 0 10
real	0m0.129s
user	0m0.161s
```

```
sys     0m0.042s

stateCount: 41
1 9 14 11 9 5 16 10 5 4 6 8 13 3 8 7 15 3 4 6 3 2 2 2 7 1 18 4 11 0 12 1 17 0 10
real    0m0.131s
user    0m0.171s
sys     0m0.031s

stateCount: 154
1 16 6 10 5 13 18 8 13 15 8 11 9 14 2 7 15 9 14 2 7 12 12 4 11 5 16 6 3 3 4 1 17 0 10
real    0m0.137s
user    0m0.172s
sys     0m0.043s
```

```
for i in 1 2 3 5 6 10 15 30; do java BHMain GEN 4 30 6 3 $i > layout.txt; time java BHMain SOLVE layout.txt; done
```

```
stateCount: 2
0

real    0m0.112s
user    0m0.132s
sys     0m0.036s

stateCount: 3
0

real    0m0.111s
user    0m0.130s
sys     0m0.038s

stateCount: 9
0

real    0m0.113s
user    0m0.132s
sys     0m0.039s

stateCount: 11
0

real    0m0.113s
user    0m0.138s
sys     0m0.032s

stateCount: 55
1 3 8 5 9 2 2 2 13 4 6 4 5 1 18 4 11 5 16 1 15 3 14 2 7 0 12 1 17 3 4 0 3 0 10
real    0m0.133s
user    0m0.181s
```

```
sys     0m0.029s

stateCount: 40
1 9 14 8 13 6 6 0 5 2 12 2 7 5 8 7 15 5 16 6 3 4 2 1 9 0 10 4 11 3 18 1 17 3 4
real    0m0.132s
user    0m0.174s
sys     0m0.029s

stateCount: 46
1 14 2 11 9 9 14 8 13 13 18 10 5 12 12 4 11 5 16 7 15 3 4 6 3 0 8 2 7 1 6 1 17 0 10
real    0m0.134s
user    0m0.162s
sys     0m0.045s

stateCount: 154
1 16 6 10 5 13 18 8 13 15 8 11 9 14 2 7 15 9 14 2 7 12 12 4 11 5 16 6 3 3 4 1 17 0 10
real    0m0.140s
user    0m0.192s
sys     0m0.029s
```

```
for i in 1 2 3 4 6 9 12 18 36; do java BHMain GEN 4 36 6 3 $i > layout.txt; time java BHMain SOLVE layout.txt; done
```

```
stateCount: 2
0

real    0m0.114s
user    0m0.138s
sys     0m0.034s

stateCount: 3
0

real    0m0.111s
user    0m0.140s
sys     0m0.026s

stateCount: 9
0

real    0m0.113s
user    0m0.130s
sys     0m0.041s

stateCount: 12
0

real    0m0.122s
user    0m0.151s
```

```
sys     0m0.029s

stateCount: 55
1 3 8 5 9 2 2 2 13 4 6 4 5 1 18 4 11 5 16 1 15 3 14 2 7 0 12 1 17 3 4 0 3 0 10
real    0m0.131s
user    0m0.176s
sys     0m0.029s

stateCount: 38
1 7 6 8 13 6 8 7 15 5 2 6 3 5 16 2 9 0 14 2 7 4 18 4 11 3 12 1 5 3 4 1 17 0 10
real    0m0.131s
user    0m0.167s
sys     0m0.031s

stateCount: 41
1 9 14 11 9 5 16 10 5 4 6 8 13 3 8 7 15 3 4 6 3 2 2 2 7 1 18 4 11 0 12 1 17 0 10
real    0m0.129s
user    0m0.159s
sys     0m0.042s

stateCount: 154
1 16 6 10 5 13 18 8 13 15 8 11 9 14 2 7 15 9 14 2 7 12 12 4 11 5 16 6 3 3 4 1 17 0 10
real    0m0.134s
user    0m0.184s
sys     0m0.031s

stateCount: 154
1 16 6 10 5 13 18 8 13 15 8 11 9 14 2 7 15 9 14 2 7 12 12 4 11 5 16 6 3 3 4 1 17 0 10
real    0m0.134s
user    0m0.181s
sys     0m0.038s
```

Conclusion

The practical completes all parts successfully, although it takes some time for some cases. I learn a lot about analysing data and coming up with data sets.

Task	**Done**	**Reported**
Part 1 - Checking Solutions Design + Implementation	Yes	Yes
Provided Tests passed	Yes	Yes
Additional Testing Undertaken	Yes	Yes
Part 2 - Black Hole Search Design + Implementation	Yes	Yes
Provided Tests passed	Yes	Yes
Additional Testing Undertaken	Yes	Yes
Evaluation Performed + Reported	Yes	Yes
Other Evaluation questions answered	No	No
Part 3 - Worm Hole Search Design + Implementation	Yes	Yes
Provided Tests passed	Yes	Yes
Additional Testing Undertaken	Somewhat	Somewhat
Evaluation Performed + Reported		

Other Evaluation questions answered		
Code Code well structured + written	Yes	Yes
build.sh script included and working	Yes	Yes
Bugs / Inefficiencies Identified	Yes	Yes
Report Report written covering required points	Yes	Yes
Any additional material/insights presented	Yes	Yes
Problems Encountered / Overcame in report	Yes	Yes