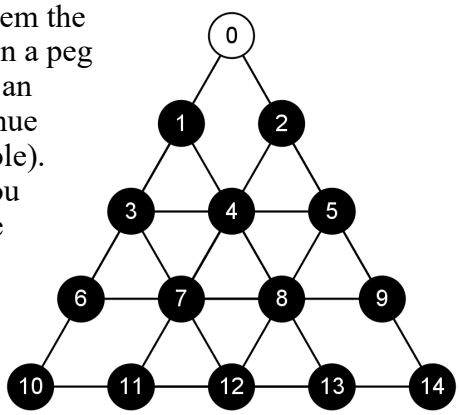
COS 485: Program #5 – PegJump

Objectives: Designing a backtracking algorithm.

You will design an algorithm to solve peg jump puzzles. For this problem the input is a board with a bunch of holes in it. Initially all the holes contain a peg except for the starting hole. You can jump a peg over a filled hole into an empty hole and then remove the jumped over peg. The goal is to continue until only one peg is left (extra credit if the last peg is in the starting hole). This is very difficult to solve by hand, but with a computer program you can write a backtracking search algorithm to explore the vast multitude of possibilities.



Example:
In the example, the first jump could be from hole 3, over hole 1, and into hole 0. This would leave hole 3 empty and remove the peg in hole 1. There is only 1 other choice at this point, but as you proceed the number of choices will increase.

Setting up the project in Eclipse:
Create a new project similar to how you set up program 1:

- It will use PegJumpTester.jar, the same Scaffold jar, and starting code PegJump.java
- In the run configuration set **Main class** to: tester/PegJumpTester

Explanation:
You are writing the method:

long solvePegJump(PegJumpPuzzle puzzle, ArrayList<Jump> jumpList)

The inputs are:

- A PegJumpPuzzle object that holds information about the puzzle, such as the number of holes, the starting hole, and a list of the valid jumps. You can get an iterator to run through the valid jumps. You will need your own array to keep track of the pegs. This is all demonstrated in the starting code.
- An empty ArrayList of Jumps. When you find a correct solution, put the sequence of jumps into this ArrayList to return your solution.

You should also count the number of nodes examined in the search and return this value. This might be quite large, so use type long for your counter.

What to turn in:
Written Report turned in through Brightspace

1. A screen shot of the report tab.
2. A screen shot of your result for the highest numbered test case solved.

Electronic Submit
From a Unix machine in the lab run the program "submit" to submit your files.
Submit your source code (.java files) and compiled code (.class files) to the directory: **prog5**

Grading:

- 50 points – Solving small test cases 1, 2 and 3
- 10 points each – Solving test larger cases 4, 5, 6, 7 and 8.
- +10 points extra credit – Solve all 8 test cases and leave the last peg in the starting hole!