# Introduction

This document presents the problem, solution, and implementation (in Python program) of a puzzle that involves pegs in a board with a pattern of regularly space holes in a straight line. Some of these holes are occupied by pegs while the remaining holes are empty. The number of holes will also vary. The problem involves jumping a peg over an adjacent peg to the following adjacent hole if it’s empty. At every jump, the peg that is jumped over is eliminated. The process continues until the board is left with a single peg or reach a state where no more valid jumps can be performed. If the board is left with a single peg, the puzzle is deemed to have been solved otherwise the puzzle is deemed to be not solvable.

# Problem

The problem is logically modelled as follows.

**Board –** Board is a string of characters with every empty hole represented as “o” and every other hole (filled by a peg) is represented as “X”. Below are a few valid representations of the Board.

* XoXX
* XXoXoX

**Move –** A valid move can be made is possible if the below 2 string patterns contained in the Board can be replaced with respective target substrings, as shown below.

* XXo -> ooX
* oXX -> Xoo

**Target State –** A target state is reached if after a sequence of valid moves, Board is taken to a state where only one peg remains in the board (Solved) or multiple pegs remaining without any further valid moves. Please find below couple of target state representations.

* XXoX -> ooXX -> oXoo (solved)
* XXooX -> ooXoX (not solvable)

# Solution

The solution attempts to use pattern matching linear search algorithm with search and replace of the matched patterns. Board is represented as a string data type with “X” represents a hole with a peg and “o” represents an empty hole.

Substrings of XXo and oXX are linearly searched in master string (representing the Board) from left to right. If a match is found, the substring pattern XXo is replaced with ooX while the pattern oXX is replaced with Xoo in the master string. The search and replace iteration is continued until the Board is reduced to a string with either just one peg (one X) or multiple pegs (multiple Xs) but with no more valid patterns.

Each Move is represented as a list of tuples with each record consisting of 2 data types. The first data type in the tuple is an “integer” data type representing the from location of the peg that is moved to an empty hole. The second data type in the tuple is a “character” data type that represents the direction of transfer.

For e.g., if a peg at location 2 is moved to location 4 over another peg at location 3, a tuple of (2, R) is created to represent this move as shown below.

ooXXo -> ooooX

Similarly, if a peg is moved from location 3 to location 1 over location 2, the resultant move is represented by tuple (3, L) as shown below.

ooXXo -> oXooo

**Algorithm**

Represent Board as a string of “X” and “o”

Initialize a list (Path) to store the moves

While (more valid moves possible)

If (linear search of pattern XXo is true)

Replace XXo with ooX in the Board string

Log the path as a tuple (int location, char “R”)

Append to the path list

Else

If (linear search of pattern oXX is true)

Replace oXX with Xoo in the Board string

Log the path as a tuple (int location, char “L”)

Append to the path list

Else

Break

Return path

# Implementation

Below python functions are developed

* Bool Isonepeg (Board) – The function takes a string as argument and counts the number of pegs. Returns Boolean value true if only one peg is remaining in the board otherwise Boolean false is returned.
* Int Patternfound (Board, pattern) – The function takes 2 strings and find if string 2 is contained as a pattern in string 1. Returns location of the second substring in the first string or returns -1 if no substring is found.
* String Move (Board) – The function takes a string and replaces every pattern found (using Patternfound function defined above), creates a record that contains the location of the pattern found (integer data type) and the direction of the move (string data type).
* Pegsolution (Board) – The function takes Board string as an argument and returns the sequence of tuples that transforms the original Board string to the target string.