1. **Introduction**

In the game called peg the player has to move the pegs over the holes and take them off until there is only 1 peg left

1. **Problem**

Key:

X = peg

o = hole

How you remove pegs is when there is 2 consecutive pegs with one hole at the end such as “XXo” or “oXX” you move the peg at the end over into the hole and then you take off the middle peg. So the ending X will go to other end where “XXo” will transform to “oXX” to “ooX”. This event keeps happening tell only one peg left in a gameboard Combination (i.e “XoXX” or “XXoo” etc.)

1. **Solution**

Algorithms

1. Finding number of pegs remaining

Steps

* Pass gameboard to function
* Initialize a counter to 0
* Loop through gameboard
* Increase counter when a peg is found

1. Pattern finding

Steps

* Pass board and pattern to the function
* Check if pattern exists in function

1. Getting new combination

Steps

* Check if combination for 2 pegs and one hole exist
* If it doesn’t add all what is currently into gameboard into new board
* If it exist check if combination of one peg and 2 holes exist
* If so change peg hole combination in that part accordingly
* Make adjustments and add those changes in those positions to new board

1. Finding number of pegs remaining

Steps

* Check if one peg is remaining
* Get newboard combination
* Return new board
* If the board has not changed display that the board is unsolvable
* If board has changed with one page remaining display new one

1. **Implementation**

* **Main functionality (solving peg)**

def pegsSolution(Board):

# Program your solution here

#initializing path array

print ("\nGameBoard = ",Board);

#initializing new board string

while Isonepeg(Board) != True:

newBoard = move (Board)

print ("new Board =", newBoard)

if (Board == newBoard):

print ("Board is not solvable; exiting")

return

else:

Board = newBoard

return path

* **Helper Function 1, Checking if it has reached one peg**

def Isonepeg(Board):

#initialize counter and flag

onex = False

countx = 0

#looping through game board

for x in range(0, len(Board), 1):

#appending when find peg

if Board[x] == "X":

countx = countx + 1

#setting flag to true when peg found

if countx == 1:

onex = True

return onex

* **Helper Function 2, Checking if pattern found**

def Patternfound(Board, pattern):

#initialize index to null

index = -1

#search for pattern

for x in range(0,len(Board)-len(pattern)+1,1):

#when pattern found return the index

if (Board[x:x+len(pattern)] == pattern):

index = x

break

return index

* **Helper Function 3, Move around Board**

def move(Board):

#initializing newboard

newBoard=""

#initializing patterns

pattern = "XXo"

newpattern = "ooX"

#checking what the pattern is

index = Patternfound (Board,pattern)

#moving based on pattern

if (index != -1):

for i in range (0, index-2,1):

newBoard = newBoard + Board[i]

for i in range (0,3,1):

newBoard = newBoard + newpattern[i]

for i in range(index+1, len(Board),1):

newBoard = newBoard + Board[i]

path.append((index-2, "R"))

print ("path = (",index-2,"R)")

else:

pattern = "oXX"

newpattern = "Xoo"

index = Patternfound (Board,pattern)

if (index != -1):

for i in range (0, index-2,1):

newBoard = newBoard + Board[i]

for i in range (0,3,1):

newBoard = newBoard + newpattern[i]

for i in range(index+1, len(Board),1):

newBoard = newBoard + Board[i]

path.append((index, "L"))

print ("path = (",index,"L)")

else:

#return current board if no pattern return regular board

return Board

#return newboard

return newBoard