PYTHON ASSIGNMENT BOOK

MAKE A MOVE TO PYTHON

ASSIGNMENTS

TASK ON GAME

Submitted By

Farhana Firoz

Date: June 10,2020

Main Python program:
Note: Here we are assuming that you have already taken size of matrix and then values in matrix (0's & 1's only) from user.

As a Developer write a program for matching following constraints:

You are given a two-dimensional array (matrix) of potentially unequal height and width containing only Os and 1s. Each 0 represents land, and each 1 represents part of a river. A river consists of any number of 1s that are either horizontally or vertically adjacent (but not diagonally adjacent). The number of adjacent 1s forming a river determine its size. Write a function that returns an array of the sizes of all rivers represented in the input matrix. Note that these sizes do not need to be in any particular order.

Now from returned array of sizes You need to print "Guess the size of River" on each index, take input from user as guesses for each entry.

If all entered sizes match then show You are the winner. If 60% of the inputs match with sizes in array of river sizes, show you got second position, else Show "Invest more money on Almonds, then come back".

Now console should Ask if you wanted to play again:
IF Yes: Redirect to the Matrix Input.
Else: Exit

Sample input:
[1,0,0,1,0]
(1,0,1,0,0)
[0,0,1,0,1]
[1,0,1,0,1]
[1,0,1,1,0,

Sizes of River = [1,2,2,2,5] note: This should not be visible to the user, this a reference for accuracy of output.

Sample output:
Guess the size of River: input from user: note: This size should be compared with size present on specific or random index in river sizes array.
Guess the size of River: input from user: 5
you are the winner.

Codes for Game:

```
from collections import defaultdict


class River(object):
   def __init__(self, numOfElements=100):
      self.rank = [0 for _ in range(numOfElements)]
      self.parents = [0 for _ in range(numOfElements)]
      self.n = numOfElements

   def init (self, numOfElements):
      self.makeSet()

   def makeSet(self):
      for i in range(self.n):
         self.parents[i] = i

   def union(self, x, y):
      parentX = self.find(x)
      parentY = self.find(y)

      if parentX == parentY:
         return
      if self.rank[parentX] > self.rank[parentY]:
         self.parents[parentY] = parentX
      elif self.rank[parentX] < self.rank[parentY]:
         self.parents[parentX] = parentY
      else:
         self.parents[parentX] = parentY
         self.rank[parentY] += 1

   def find(self, x):
      parentX = self.parents[x]
      if x != parentX:
         parentX = self.find(parentX)
      return parentX


def riverSizes(matrix):
   global i, j
   if not matrix:
      return []

   rowCount, colCount = len(matrix), len(matrix[0])
```

```
djs = River()
for i in range(rowCount):
    for j in range(colCount):
        val = matrix[i][j]
        if val == 0:
            continue

if i + 1 < rowCount and matrix[i + 1][j] == 1:
    djs.union(i * colCount + j, (i + 1) * colCount + j)

if i - 1 >= 0 and matrix[i - 1][j] == 1:
    djs.union(i * colCount + j, (i - 1) * colCount + j)

if j + 1 < colCount and matrix[i][j + 1] == 1:
    djs.union(i * colCount + j, (i) * colCount + j + 1)

if j - 1 >= 0 and matrix[i][j - 1] == 1:
    djs.union(i * colCount + j, (i) * colCount + j - 1)

islands = defaultdict(int)
for i in range(rowCount):
    for j in range(colCount):
        if matrix[i][j] == 1:
            val = i * colCount + j
            parent = djs.find(val)
            islands[parent] += 1

return islands.values()
```