Juego De la Vida

April 30, 2019

1 Juego de La Vida de Conway

1.1 El presente trabajo contiene una Simulacion del Juego de la Vida

1.1.1 Ing. Sistemas

1.1.2 Simulacion

Pedro Bermeo

- ***Historia***

El juego de la vida es un autómata celular diseñado por el matemático británico John Horton Conway en 1970. Hizo su primera aparición pública en el número de octubre de 1970 de la revista Scientific American, en la columna de juegos matemáticos de Martin Gardner. Desde un punto de vista teórico, es interesante porque es equivalente a una máquina universal de Turing, es decir, todo lo que se puede computar algorítmicamente se puede computar en el juego de la vida. Desde su publicación, ha atraído mucho interés debido a la gran variabilidad de la evolución de los patrones. Se considera que la vida es un buen ejemplo de emergencia y auto organización. Es interesante para los científicos, matemáticos, economistas y otros observar cómo patrones complejos pueden provenir de la implementación de reglas muy sencillas. La vida tiene una variedad de patrones reconocidos que provienen de determinadas posiciones iniciales. Poco después de la publicación, se descubrieron el pentaminó R, el planeador o caminador (en inglés, glider, conjunto de células que se desplazan) y el explosionador (células que parecen formar la onda expansiva de una explosión), lo que atrajo un mayor interés hacia el juego. Contribuyó a su popularidad el hecho de que se publicó justo cuando se estaba lanzando al mercado una nueva generación de miniordenadores baratos, lo que significaba que se podía jugar durante horas en máquinas que, por otro lado, no se utilizarían por la noche. Para muchos aficionados, el juego de la vida solo era un desafío de programación y una manera divertida de usar ciclos de la CPU. Para otros, sin embargo, el juego adquirió más connotaciones filosóficas. Desarrolló un seguimiento casi fanático a lo largo de los años 1970 hasta mediados de los 80.

- ***Resultados esperados***
- Con la ejecucion del Juego de la vida, pretendo descubirir con diferentes patrones de entrada ver cual es la diferencia para la supervivencia de cada celda inicial.
- Determinar cual es el patron que mas celdas sobrevivientes me da al final de la ejecucion.
- Determinar la diferencia de tiempo que se genera a partir de las pruebas con diferentes formas de juego y patrones de entrada.

- ***Objetivo***

Con las figuras iniciales que van a trabajar cuales son los patrones que permiten que se genere un oscilador o explorador

- *A continuacion Ingrese el Numero de Celdas*

- *De un Enter para Continuar con la Ejecucion*

[3]:
```
%matplotlib inline
import random
import os
import time
import matplotlib.pyplot as pp
import xlsxwriter

import sys


class JuegoVida():

    def __init__(self):
        self.contI = 0
        self.resultados = []
        self.nCeldas = 0
    def game(self):
        print("--------------------------------------------------------------------------------")
        rows = int(input("Ingrese la Cantidad de Filas:\n"))
        columns = int(input("Ingrese la Cantidad de Columas:\n"))
        e = False
        while(e == False):
            try:
                r = int(input("Desea que se genere Aleatoriamente las Celdas:\n 1-> Si\n 2-> No\n"))
                if(r >0 and r<3):
                    e = True
                else:
                    print("###Error###")
                    print("Por favor, ingrese el Número correspondiente a su Elección")
            except:
                print("###Error###")
                print("Por favor, ingrese el Número correspondiente a su Elección")
        print("--------------------------------------------------------------------------------")
```

```
       if(r==2):

           self.nCeldas = int(input("Ingrese el # de Celdas:\n"))
           e = False
           while (e == False):
               try:
                   r2 = int(input("Desea que las Casillas se generen␣
↪Aleatoriamente:\n 1-> Si\n 2-> No\n"))
                   if (r2 > 0 and r2 < 3):
                       e = True
                   else:
                       print("###Error###")
                       print("Por favor, ingrese el Número correspondiente a su␣
↪Elección")
               except:
                   print("###Error###")
                   print("Por favor, ingrese el Número correspondiente a su␣
↪Elección")
           ␣
↪print("--------------------------------------------------------------------------------")
           if(r2==1):
               print("Va a generar Las Casillas")
               container = [
                   [0, 0, 0],
                   [0, 0, 0],
                   [0, 0, 0]
               ]

               actualTable = []
               futureTable = []

               self.crearTablaActualCasillasAleatorias(actualTable, rows,␣
↪columns)
               self.cargarTablaFutura(futureTable, rows, columns)

               self.validarJuego(actualTable, futureTable, container, rows,␣
↪columns)

           else:
               container = [
                   [0, 0, 0],
                   [0, 0, 0],
                   [0, 0, 0]
               ]

               actualTable = []
```

```
                futureTable = []

                self.crearTablaActualIngresaUsuario(actualTable, rows, columns)
                self.cargarTablaFutura(futureTable, rows, columns)

                self.validarJuego(actualTable, futureTable, container, rows,␣
↪columns)

        else:
            container = [
                [0, 0, 0],
                [0, 0, 0],
                [0, 0, 0]
            ]

            actualTable = []
            futureTable = []

            self.crearTablaActual(actualTable, rows, columns)
            self.cargarTablaFutura(futureTable, rows, columns)

            self.validarJuego(actualTable, futureTable, container, rows, columns)


    def validarJuego(self, actualTable, futureTable, container, rows, cols):
        while True:

            for row in range(rows - 2):
                for col in range(cols - 2):

                    cell1 = actualTable[row][col]
                    cell2 = actualTable[row][col + 1]
                    cell3 = actualTable[row][col + 2]

                    cell4 = actualTable[row + 1][col]
                    cell5 = actualTable[row + 1][col + 1]
                    cell6 = actualTable[row + 1][col + 2]

                    cell7 = actualTable[row + 2][col]
                    cell8 = actualTable[row + 2][col + 1]
                    cell9 = actualTable[row + 2][col + 2]

                    container[0][0] = cell1
                    container[0][1] = cell2
                    container[0][2] = cell3

                    container[1][0] = cell4
```

```
                    container[1][1] = cell5
                    container[1][2] = cell6

                    container[2][0] = cell7
                    container[2][1] = cell8
                    container[2][2] = cell9

                    cellCounter = 0

                    for rowContainer in range(3):
                        for colContainer in range(3):

                            if not (rowContainer == 1 and colContainer == 1):
                                if container[rowContainer][colContainer] == 1:
                                    cellCounter = cellCounter + 1

                    if cellCounter < 2 and container[1][1] == 1:
                        futureTable[row + 1][col + 1] = 0

                    elif cellCounter > 3 and container[1][1] == 1:
                        futureTable[row + 1][col + 1] = 0

                    elif cellCounter == 3 and container[1][1] == 0:
                        futureTable[row + 1][col + 1] = 1

                    elif cellCounter == 3 and container[1][1] == 1:
                        futureTable[row + 1][col + 1] = 1

                    elif cellCounter == 2:
                        futureTable[row + 1][col + 1] = actualTable[row + 1][col␣
↪+ 1]

            auxiliarTable = actualTable
            actualTable = futureTable
            futureTable = auxiliarTable

            os.system('clear')
            #os.system('CLS')
            self.imprimirTabla(futureTable, rows, cols)
            self.reiniciarTabla(futureTable, rows, cols)
            time.sleep(.4)


    def imprimirTabla(self,actualTable, rows, columns):
        res = []
        self.contI+=1
        cont = 0
```

```
        for row in range(rows):
            for col in range(columns):

                if actualTable[row][col] == 1:
                    print("| O ", end='')
                    cont += 1
                else:
                    print("|   ", end='')

                if col == columns - 1:
                    print('')

        self.resultados.append([self.contI,cont])
        print("Número de Iteración, ",self.contI)
        print("Número de Celdas Vivas, ",cont)
        ␣
↪print("--------------------------------------------------------------------------------")

        if (cont == 0):
            print("Juego Terminado")
            print("Resutlados Finales",self.resultados)

            self.crearGrafica()
            self.crearExcel()
            #exit()
            sys.exit("Juego Terminado")


    def crearTablaActual(self,actualTable, rows, columns):
        for row in range(rows):
            for col in range(columns):

                newArray = []
                for zeroValue in range(columns):
                    newArray.append(0)

                actualTable.append(newArray)

        for row in range(rows):
            for col in range(columns):
                value = random.randint(0, 1)

                if value == 1:
                    self.nCeldas +=1
                    actualTable[row][col] = 1
```

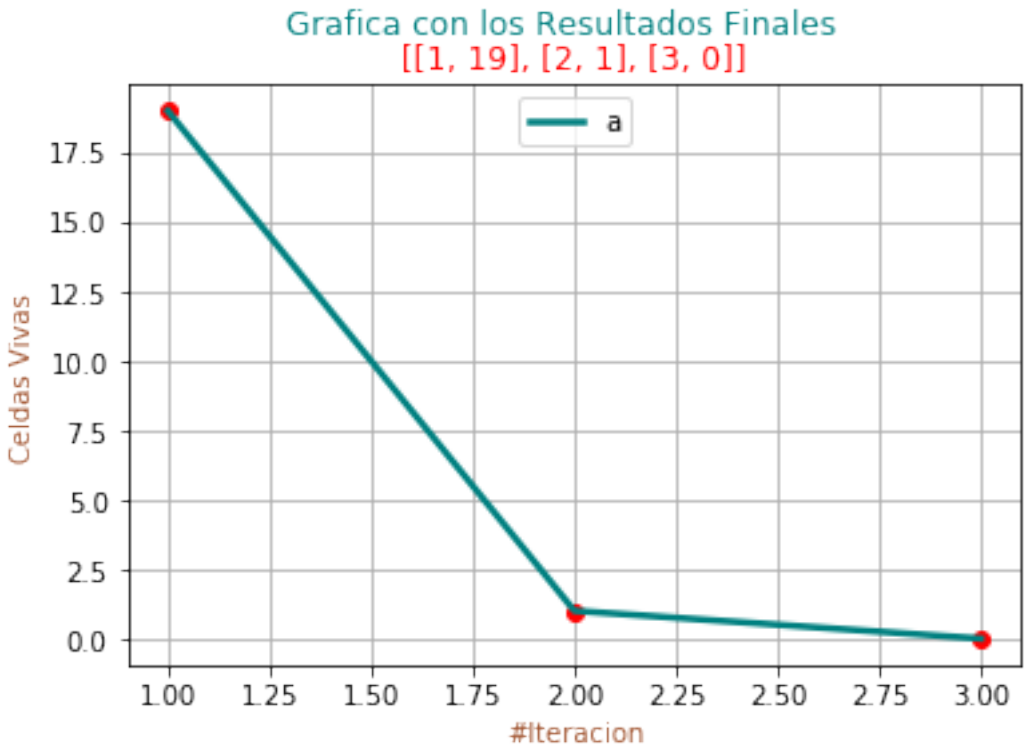
```
    def cargarTablaFutura(self,futureTable, rows, columns):
        for row in range(rows):
            for col in range(columns):

                newArray = []
                for zeroValue in range(columns):
                    newArray.append(0)

                futureTable.append(newArray)

        for row in range(rows):
            for col in range(columns):
                futureTable[row][col] = 0

    def crearTablaActualCasillasAleatorias(self,actualTable, rows, columns):
        cont =0
        for row in range(rows):
            for col in range(columns):

                newArray = []
                for zeroValue in range(columns):
                    newArray.append(0)

                actualTable.append(newArray)

        for row in range(rows):
            for col in range(columns):
                value = random.randint(0, 1)

                if value == 1:
                    cont +=1
                    if(cont<=self.nCeldas):
                        actualTable[row][col] = 1
                    else:
                        actualTable[row][col] = 0


    def crearTablaActualIngresaUsuario(self,actualTable, rows, columns):
        cont = 0
        print("Para ingresar su Celdas, ingrese los siguientes Valores:\n 0-> Si␣
↪no desea una Celda\n 1-> Si desea ubicar una Celda ")
        for row in range(rows):
            for col in range(columns):

                newArray = []
                for zeroValue in range(columns):
                    newArray.append(0)
```

```
                actualTable.append(newArray)

        for row in range(rows):
            for col in range(columns):
                if(cont<self.nCeldas):
                    print(("Desea una Celda en la posición: [",row, col,"] de la␣
↪Matriz: "))
                    value = int(input())
                    #value = random.randint(0, 1)

                    if value == 1:
                        cont+=1
                        actualTable[row][col] = 1

    def reiniciarTabla(self,table, rows, cols):
        for row in range(rows):
            for col in range(cols):
                table[row][col] = 0

    def crearGrafica(self):
        x = []
        y = []
        for i in self.resultados:
            x.append(i[0])
            y.append(i[1])
            # print("i",i)
        pp.plot(x, y, 'ro')
        pp.suptitle("Grafica con los Resultados Finales", color="teal")
        pp.title(self.resultados, color='red')
        pp.plot(x, y, color="teal", linewidth=2.5, linestyle="-", label="a")
        pp.xlabel("#Iteracion", color="sienna")
        pp.ylabel("Celdas Vivas", color="sienna")
        pp.legend(loc='upper center')
        pp.grid(True)
        # pp.set_title('Grafica Final')
        pp.savefig('grafico.png')
        pp.show()

    def crearExcel(self):
        workbook = xlsxwriter.Workbook('ArchivoPrueba.xlsx')
        worksheet = workbook.add_worksheet()
        cell_format = workbook.add_format()
        cell_format.set_bold()
        cell_format.set_italic()
        cell_format.set_font_color('red')
        cell_format.set_font_size(13)
```

```
        cell_format2 = workbook.add_format()
        cell_format2.set_align("center")
        cell_format2.set_bold()

        cell_format3 = workbook.add_format()
        cell_format3.set_align("center")

        worksheet.set_column('A:A', 15)
        worksheet.set_column('B:B', 15)

        bold = workbook.add_format({'bold': True})

        worksheet.write(0, 0, 'Realizado por:',cell_format)
        worksheet.write(0, 1, 'Pedro Bermeo', cell_format)
        worksheet.write(2, 0, 'RESULTADOS FINALES JUEGO DE LA VIDA',bold)
        worksheet.write(4, 0, '# Celdas Iniciales:',bold)
        worksheet.write(4, 2, self.nCeldas,cell_format2)
        worksheet.write(5, 0,␣
 ↪'------------------------------------------------------------')
        worksheet.write(6, 0, '# Iteracion',cell_format2)
        worksheet.write(6, 1, '# Celdas Vivas',cell_format2)
        worksheet.insert_image(0, 5, 'grafico.png')
        # Start from the first cell. Rows and columns are zero indexed.
        row = 7
        col = 0

        # Iterate over the data and write it out row by row.
        for i, c in self.resultados:
            worksheet.write(row, col, i,cell_format3)
            worksheet.write(row, col + 1, c,cell_format3)
            row += 1

        workbook.close()


if __name__ == "__main__":
    juegoVida = JuegoVida()
    juegoVida.game()
```

```
--------------------------------------------------------------------------------
Ingrese la Cantidad de Filas:
6
Ingrese la Cantidad de Columas:
6
Desea que se genere Aleatoriamente las Celdas:
 1-> Si
```

```
 2-> No
1
--------------------------------------------------------------------------------
| O |   | O | O |   |
|   |   |   |   |   | O
| O | O |   | O | O | O
| O | O | O |   | O |
|   |   | O | O |   | O
|   | O |   | O | O |
Número de Iteración,  1
Número de Celdas Vivas,  19
--------------------------------------------------------------------------------
|   |   |   |   |   |   |
|   |   |   |   |   |   |
|   |   |   | O |   |   |
|   |   |   |   |   |   |
|   |   |   |   |   |   |
|   |   |   |   |   |   |
Número de Iteración,  2
Número de Celdas Vivas,  1
--------------------------------------------------------------------------------
|   |   |   |   |   |   |
|   |   |   |   |   |   |
|   |   |   |   |   |   |
|   |   |   |   |   |   |
|   |   |   |   |   |   |
|   |   |   |   |   |   |
Número de Iteración,  3
Número de Celdas Vivas,  0
--------------------------------------------------------------------------------
Juego Terminado
Resutlados Finales [[1, 19], [2, 1], [3, 0]]
```




```
An exception has occurred, use %tb to see the full traceback.

SystemExit: Juego Terminado
```

[ ]:

[ ]: