Practica 4. Programa - tablero

Autor: Victor Ulises Miranda Chávez. Grupo: 5BM1

Fecha de entrega: 06/11/2022

1. Introducción

Los autómatas no determinísticos (NDAs) son una variante de los autómatas finitos que permiten tener más de una transición posible para un mismo símbolo de entrada y un mismo estado. Esto se logra mediante la adición de un conjunto de transiciones para cada par (símbolo de entrada, estado) en lugar de tener una única transición.

Un ejemplo es el tablero de ajedrez donde tenemos:

1. Estados: Casillas
2. Entradas:
   - R (Moverse a una casilla adyacente de color Rojo)
   - B (Moverse a una casilla adyacente de color Negro)
3. El estado inicial, el estado final en las esquinas opuestas

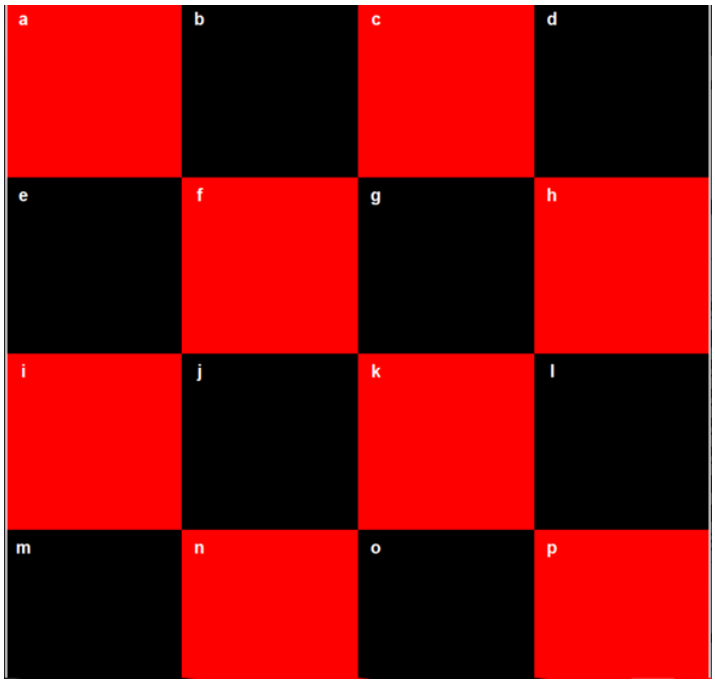

Figura 1: Tablero

1.1. Objetivo

Elaborar un programa para realizar movimientos ortogonales y diagonales en un tablero de ajedrez de 4x4 con dos piezas. Los movimientos y las reglas están explicadas en las láminas del curso de Stanford.

Adicionalmente, el programa debe de contar con las siguientes características:

1. Debe de correr en modo automático (todo) y forma manual.
2. En el caso de forma manual, el usuario podrá introducir la cadena de movimientos o generarla aleatoriamente.
3. El programa puede correr con una pieza o dos.
4. En el caso de dos piezas, la segunda iniciará en el estado 4 y su estado final es el estado 13.
5. Cuando inicie el juego, de manera aleatoria el programa debe decidir quién inicia primero.
6. Una vez definida la cadena de movimientos para uno o dos piezas, se deben generar los archivos de todos los movimientos posibles por pieza, generar otro archivo con todos los movimientos ganadores por pieza. Estos dos últimos archivos servirán para reconfigurar las rutas.
7. Si se reconfigura una ruta y aún así no se puede avanzar, entonces habrá que esperar una iteración para continuar.
8. Graficar el tablero y mostrar los movimientos de una pieza o dos piezas.
9. Si se escoge el modo automático las cadenas generadas no deben ser mayores a 10 movimientos para la animación.
10. El número máximo de movimientos deberá de ser de a lo más 100 símbolos.
11. En un archivo de salida (imagen) dibujar el árbol(les) de la cadena(s) evaluadas en esa corrida.
12. Incluir en el reporte el código fuente.

2. Desarrollo

2.1. Lenguaje utilizado

- Python:
- La libreria utilizada para dibujar fue tkinter

2.2. Capturas de resultados:

```
BIENVENIDO
Escoja su modo de juego:
1. Manual
2. Automatico
Inserte el no. de su opción: 1

Elija cuantos jugadores desea:
1. Un jugador
2. Dos jugadores
Ingrese su opcion: 2

Se eligieron dos jugadores:
Ingrese su ruta jugador 1: RBBR
Ingrese su ruta jugador 2: BBB
```

Figura 2: Seleccion del modo de juego y la rutas deseadas

```
Generando rutas...
-------Rutas generadas-------

Eligiendo ruta ganadora de cada jugador...
Ruta elegida para jugador1:  ['a', 'f', 'j', 'o', 'p']
Ruta elegida para jugador2:  ['d', 'g', 'j', 'm']

Empieza jugador 2 (JUGADOR PRINCIPAL)
Se cruzan
Ruta final para jugador1:  ['a', 'f', 'f', 'j', 'o', 'p']
Ruta final para jugador2:  ['d', 'g', 'j', 'm']

----------Gana el jugador 2-----------
```

Figura 3: Rutas generadas

```
Ruta agrupada de los jugadores:  [('d', 'a'), ('g', 'f'), ('j', 'f'), ('m', 'j'), (None, 'o'), (None, 'p')]

Ruta del jugador PRINCIPAL:  d
Ruta del jugador SECUNDARIO:  a

Ruta del jugador PRINCIPAL:  g
Ruta del jugador SECUNDARIO:  f

Ruta del jugador PRINCIPAL:  j
Ruta del jugador SECUNDARIO:  f

Ruta del jugador PRINCIPAL:  m
Ruta del jugador SECUNDARIO:  j

Ruta del jugador PRINCIPAL:  None
Ruta del jugador SECUNDARIO:  o

Ruta del jugador PRINCIPAL:  None
Ruta del jugador SECUNDARIO:  p
-------RUTAS TERMINADAS-------
```

Figura 4: Movimientos por turno

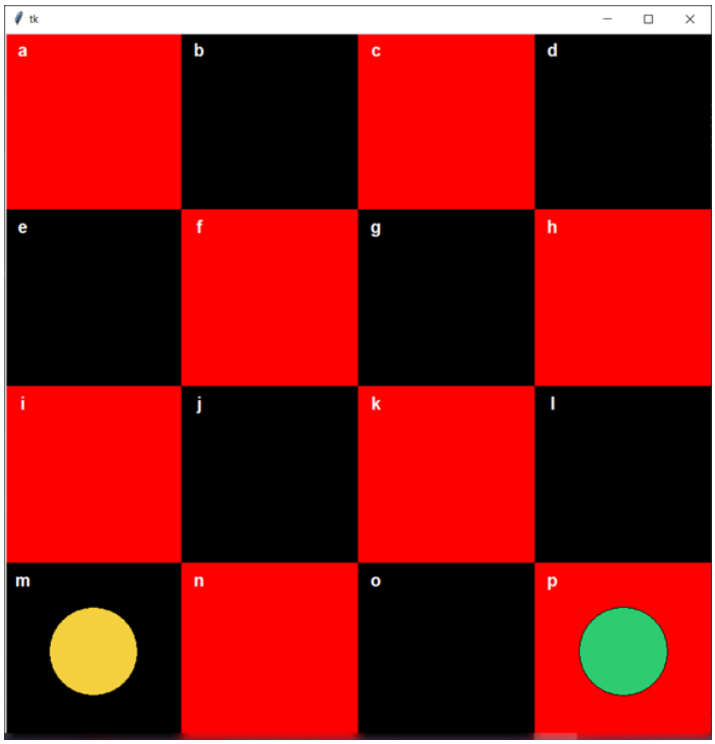

Figura 5: Estado final en la animacion

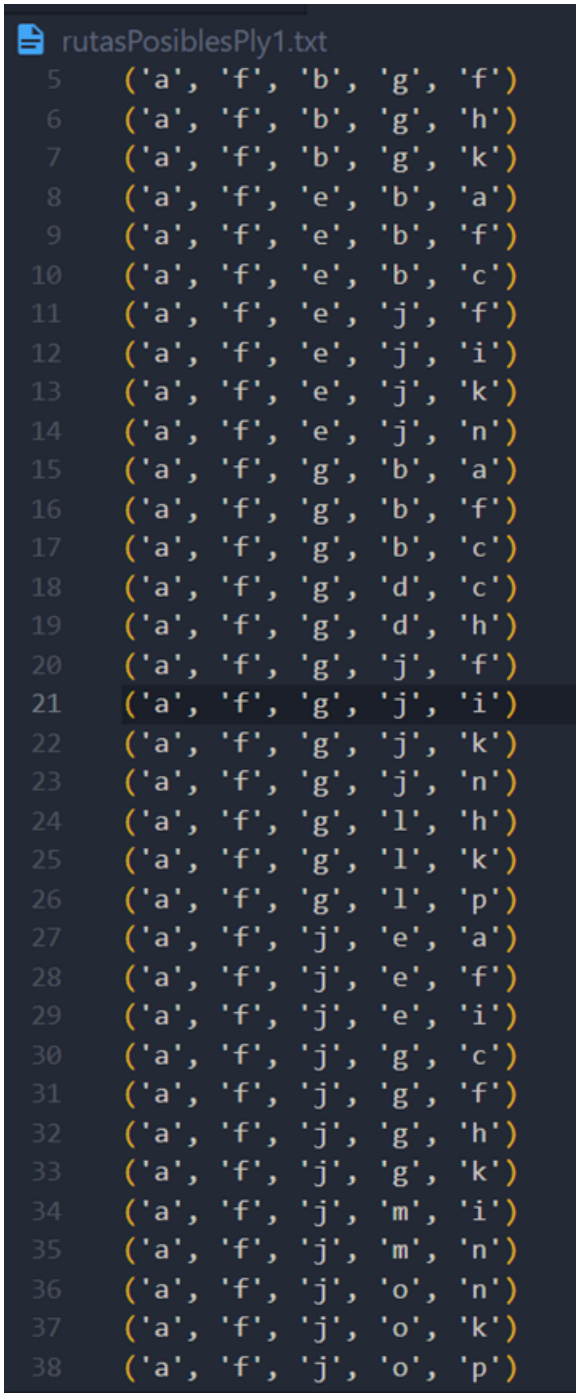

Figura 6: Rutas posibles del jugador 1

rutasPosiblesPly2.txt

```
1  ('d', 'g', 'b', 'e')
2  ('d', 'g', 'b', 'g')
3  ('d', 'g', 'd', 'g')
4  ('d', 'g', 'j', 'e')
5  ('d', 'g', 'j', 'g')
6  ('d', 'g', 'j', 'm')
7  ('d', 'g', 'j', 'o')
8  ('d', 'g', 'l', 'g')
9  ('d', 'g', 'l', 'o')
```

Figura 7: Rutas posibles del jugador 2

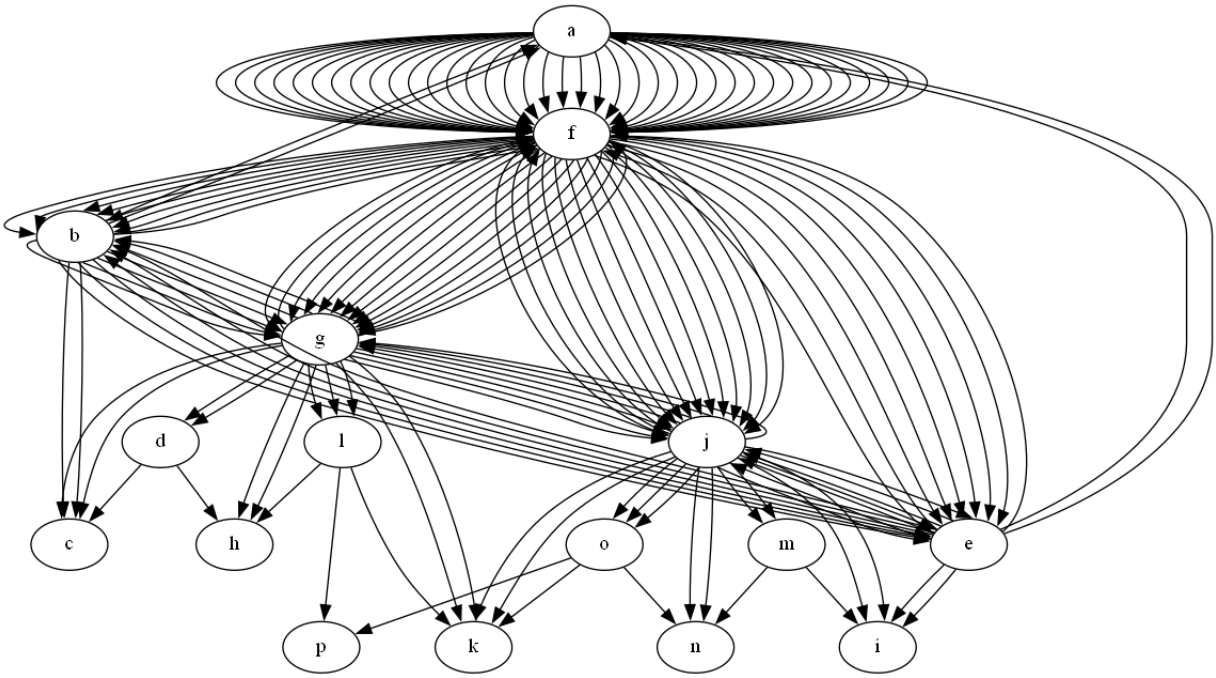

Figura 8: Arbol de las rutas posibles del jugador 1

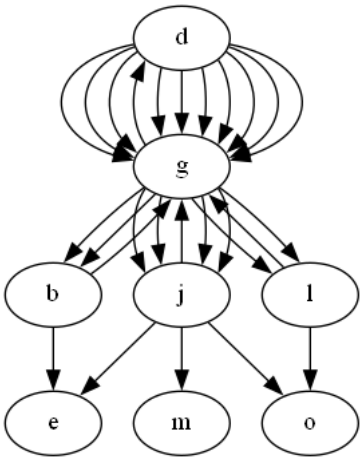

Figura 9: Arbol de las rutas posibles del jugador 2

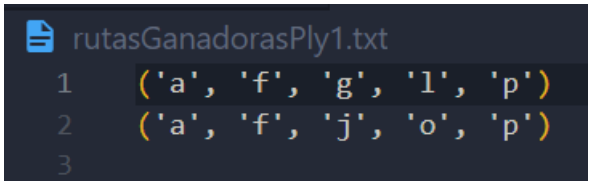

Figura 10: Rutas ganadoras del jugador 1

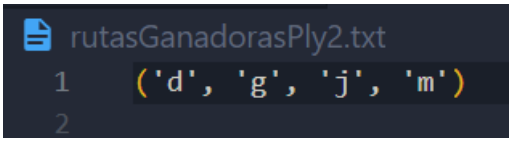

Figura 11: Rutas ganadoras del jugador 2

3. Conclusiones:

Desarrollar esta práctica fue retroalimentativa, puesto que pude conocer la finalidad de los autómatas no determinísticos. Por la parte práctica tuve la oportunidad de poder fortalecer mis habilidades con la creación de un algoritmo lo suficientemente rápido para poder crear todas las posibles rutas que el usuario desea para sus jugadas; fue interesante saber la cantidad increíble de posibilidades de movimientos que se pueden crear a medida que la ruta crece.

4. Bibliografía:

Hopcroft, J. E., Motwani, R. & Ullman, J. D. (2008). Teoría de autómatas, lenguajes y computación 3/E. Pearson Educación.

5. Código:

```
# TABLERO NFA
from itertools import zip_longest
import os
import random
import re
import time
from strgen import StringGenerator
from tkinter import*
import random
import networkx as nx
import matplotlib.pyplot as plt
import graphviz

switch = {
    'a': {'B': 'be', 'R': 'f'},
    'b': {'B': 'eg', 'R': 'afc'},
    'c': {'B': 'bgd', 'R': 'fh'},
    'd': {'B': 'g', 'R': 'ch'},
    'e': {'B': 'bj', 'R': 'afi'},
    'f': {'B': 'begj', 'R': 'acik'},
    'g': {'B': 'bdjl', 'R': 'cfhk'},
    'h': {'B': 'dgl', 'R': 'ck'},
    'i': {'B': 'ejm', 'R': 'fn'},
    'j': {'B': 'egmo', 'R': 'fikn'},
    'k': {'B': 'gjlo', 'R': 'fhnp'},
    'l': {'B': 'go', 'R': 'hkp'},
    'm': {'B': 'j', 'R': 'in'},
```

```
    'n': {'B': 'mjo', 'R': 'ik'},
    'o': {'B': 'jl', 'R': 'nkp'},
    'p': {'B': 'lo', 'R': 'k'},
}


def routes(current, path):
    if not path:
        yield (current,)
        return
    first, *newpath = path
    for state in switch[current][first]:
        for route in routes(state, newpath):
            yield (current,) + route


def chess(path, estadoInicial: str, id: str):
    fileRutasPosibles = open(f'rutasPosibles{id}.txt', 'a')
    fileRutasGanadoras = open(f'rutasGanadoras{id}.txt', 'a')
    listSize = 0

    rutas = []

    for i, r in enumerate(routes(estadoInicial, path), 1):
        fileRutasPosibles.write(str(r) + '\n')
        rutas.append(list(r))

        if (id == 'Ply1' and r[-1] == 'p'):
            fileRutasGanadoras.write(str(r) + '\n')
            listSize += 1
        if (id == 'Ply2' and r[-1] == 'm'):
            fileRutasGanadoras.write(str(r) + '\n')
            listSize += 1

    fileRutasGanadoras.close()
    fileRutasPosibles.close()

    graficarRuta(rutas, id)

    return listSize


def obtenerRutaGanadoraAleatoria(fileRutas, listSize):
    fileRutasGanadoras = open(fileRutas, 'r')
    randomRutaGanadora = 0
    if (listSize):
        randomRutaGanadora = random.randint(1, listSize)
    rutaGanadora = []
    patron = re.compile('[a-p]')

    count = 1
    for line in fileRutasGanadoras:
        if (count == randomRutaGanadora):
            rutaGanadora = patron.findall(line)
            break
        count += 1

    return rutaGanadora


def corregirCruces(rutaGanadora1: list, rutaGanadora2: list):
    i = 0
    for l1, l2 in zip_longest(rutaGanadora1, rutaGanadora2, fillvalue=' '):
        if(l1 == l2):
            print("Se cruzan")
            rutaGanadora2.insert(i, rutaGanadora2[i-1])
```

```
94          i += 1
95
96      return rutaGanadora2
97
98
99  def recorridoRutasGanadoras(rutaJugadorInicial: list, rutaJugadorSecundario:
        list):
100     i = 0
101     for l1, l2 in zip_longest(rutaJugadorInicial, rutaJugadorSecundario):
102
103         if not l1:
104             return "principal"
105         if not l2:
106             return "secundario"
107     return "principal"
108
109 def graficarRuta(rutas, id):
110
111     # Creamos un grafo
112     g = graphviz.Digraph()
113
114     # Agregamos los nodos y las aristas a partir de las rutas
115     for ruta in rutas:
116         for nodo in ruta:
117             g.node(nodo)
118         for i in range(len(ruta)-1):
119             g.edge(ruta[i], ruta[i+1])
120
121     # Renderizamos el grafo
122     g.render(format='png', filename=f'grafo{id}')
123
124 def game(modoJuego: int, dosJugadores: bool, ):
125     listSize1 = 0
126     listSize2 = 0
127
128     if dosJugadores == 1:
129         print("\nSe eligi  un jugador: ")
130
131         if (modoJuego == 2):
132             size = str(random.randint(1, 10))  # Para evitar morir
133             print("tama o: ", size)
134
135             cadenaAleatoria = StringGenerator("[RB]{%s}" % size).render_list(1)
136             print("Cadena generada: ", cadenaAleatoria)
137
138             print("\nGenerando rutas... ")
139             listSize1 = chess(cadenaAleatoria[0], 'a', 'Ply1')
140         else:
141             cadena = input("Ingrese su ruta: ")
142
143             print("\nGenerando rutas... ")
144             listSize1 = chess(cadena, 'a', 'Ply1')
145
146         print("-------Rutas generadas-------")
147
148         rutaGanadora1 = obtenerRutaGanadoraAleatoria(
149             "rutasGanadorasPly1.txt", listSize1)
150
151         if(rutaGanadora1):
152             print("Ruta elegida para jugador1: ", rutaGanadora1)
153             return list(zip_longest(rutaGanadora1, ['']))
154         else:
155             print("No hay ruta ganadora")
156
157     else:
158         print("\nSe eligieron dos jugadores: ")
```

```
159
160         if modoJuego == 2:
161             sizeJug1 = str(random.randint(1, 10))
162             sizeJug2 = str(random.randint(1, 10))
163 
164             print("Tama o1: ", sizeJug1)
165             print("Tama o2: ", sizeJug2)
166 
167             cadenaAleatoria1 = StringGenerator(
168                 "[RB]{%s}" % sizeJug1).render_list(1)
169             print("Cadena generada para jug 1: ", cadenaAleatoria1)
170 
171             cadenaAleatoria2 = StringGenerator(
172                 "[RB]{%s}" % sizeJug2).render_list(1)
173             print("Cadena generada para jug 2: ", cadenaAleatoria2)
174 
175             print("\nGenerando rutas... ")
176             listSize1 = chess(cadenaAleatoria1[0], 'a', 'Ply1')
177             listSize2 = chess(cadenaAleatoria2[0], 'd', 'Ply2')
178         else:
179             cadena1 = input("Ingrese su ruta jugador 1: ")
180             cadena2 = input("Ingrese su ruta jugador 2: ")
181 
182             print("\nGenerando rutas... ")
183             listSize1 = chess(cadena1, 'a', 'Ply1')
184             listSize2 = chess(cadena2, 'd', 'Ply2')
185 
186         print("-------Rutas generadas-------")
187 
188         print("\nEligiendo ruta ganadora de cada jugador...")
189 
190         rutaGanadoraPly1 = obtenerRutaGanadoraAleatoria(
191             "rutasGanadorasPly1.txt", listSize1)
192         rutaGanadoraPly2 = obtenerRutaGanadoraAleatoria(
193             "rutasGanadorasPly2.txt", listSize2)
194 
195         if (rutaGanadoraPly1 and rutaGanadoraPly2):
196             print("Ruta elegida para jugador1: ", rutaGanadoraPly1)
197             print("Ruta elegida para jugador2: ", rutaGanadoraPly2)
198 
199             startPlayer = random.randint(0,1)
200 
201             if (startPlayer == 0):
202                 print("\nEmpieza jugador 1 (JUGADOR PRINCIPAL)")
203                 rutaGanadoraPly2 = corregirCruces(
204                     rutaGanadoraPly1, rutaGanadoraPly2)
205                 ganador = recorridoRutasGanadoras(
206                     rutaGanadoraPly1, rutaGanadoraPly2)
207 
208                 print("Ruta final para jugador1: ", rutaGanadoraPly1)
209                 print("Ruta final para jugador2: ", rutaGanadoraPly2)
210 
211                 if (ganador == "principal"):
212                     print("\n----------Gana el jugador 1-----------")
213                     return list(zip_longest(rutaGanadoraPly1, rutaGanadoraPly2)
    ) #return(jugadorPrincipal, jugadorSecundario)
214                 else:
215                     print("\n----------Gana el jugador 2-----------")
216                     return list(zip_longest(rutaGanadoraPly1, rutaGanadoraPly2)
    )
217 
218             else:
219                 print("\nEmpieza jugador 2 (JUGADOR PRINCIPAL)")
220                 rutaGanadoraPly1 = corregirCruces(
221                     rutaGanadoraPly2, rutaGanadoraPly1)
222                 ganador = recorridoRutasGanadoras(
```

```
                    rutaGanadoraPly2, rutaGanadoraPly1)

                print("Ruta final para jugador1: ", rutaGanadoraPly1)
                print("Ruta final para jugador2: ", rutaGanadoraPly2)

                if (ganador == "principal"):
                    print("\n----------Gana el jugador 2----------")
                    return list(zip_longest(rutaGanadoraPly2, rutaGanadoraPly1)
    )
                else:
                    print("\n----------Gana el jugador 1----------")
                    return list(zip_longest(rutaGanadoraPly2, rutaGanadoraPly1)
    )

        elif (rutaGanadoraPly1):
            print("Ruta elegida para jugador1: ", rutaGanadoraPly1)
            print("\nNo hay ruta ganadora para el jugador 2")

            print("\n----------Gana el jugador 1----------")

            return list(zip_longest(rutaGanadoraPly1, rutaGanadoraPly2))

        elif (rutaGanadoraPly2):
            print("\nNo hay ruta ganadora para el jugador 1")
            print("Ruta elegida para jugador2: ", rutaGanadoraPly2)

            print("\n----------Gana el jugador 2----------")
            return list(zip_longest(rutaGanadoraPly2, rutaGanadoraPly1))

        else:
            print("\nNo hay rutas ganadoras de ningun jugador")
            print("\n----------Nadie gana----------")


if 'rutasGanadorasPly1.txt' in os.listdir('.'):
    os.remove('./rutasGanadorasPly1.txt')
if 'rutasGanadorasPly2.txt' in os.listdir('.'):
    os.remove('./rutasGanadorasPly2.txt')
if 'rutasPosiblesPly1.txt' in os.listdir('.'):
    os.remove('./rutasPosiblesPly1.txt')
if 'rutasPosiblesPly2.txt' in os.listdir('.'):
    os.remove('./rutasPosiblesPly2.txt')

print("BIENVENIDO")
print("Escoja su modo de juego: ")
print("1. Manual")
print("2. Automatico")
modoJuego = int(input("Inserte el no. de su opci n: "))

patron = re.compile('[a-p]')
rutaAgrupada = []

if (modoJuego == 1):
    print("\nElija cuantos jugadores desea: ")
    print("1. Un jugador")
    print("2. Dos jugadores")
    dosJugadores = int(input("Ingrese su opcion: "))

    rutaAgrupada = game(modoJuego, dosJugadores)
else:
    dosJugadores = random.randint(0, 1)
    rutaAgrupada = game(modoJuego, dosJugadores)

ventana = Tk()
canv = Canvas(ventana, width=800, height=800)
ventana.geometry("800x800")
```

```
287
288 canv.create_rectangle(0, 0, 200, 200, width=0, fill='red')
289 canv.create_rectangle(200, 0, 400, 200, width=0, fill='black')
290 canv.create_rectangle(400, 0, 600, 200, width=0, fill='red')
291 canv.create_rectangle(600, 0, 800, 200, width=0, fill='black')
292 canv.create_text(20, 20, text="a", fill="white", font=('Helvetica 15 bold'))
293 canv.create_text(220, 20, text="b", fill="white", font=('Helvetica 15 bold'))
294 canv.create_text(420, 20, text="c", fill="white", font=('Helvetica 15 bold'))
295 canv.create_text(620, 20, text="d", fill="white", font=('Helvetica 15 bold'))
296
297 canv.create_rectangle(0, 200, 200, 400, width=0, fill='black')
298 canv.create_rectangle(200, 200, 400, 400, width=0, fill='red')
299 canv.create_rectangle(400, 200, 600, 400, width=0, fill='black')
300 canv.create_rectangle(600, 200, 800, 400, width=0, fill='red')
301 canv.create_text(20, 220, text="e", fill="white", font=('Helvetica 15 bold'))
302 canv.create_text(220, 220, text="f", fill="white", font=('Helvetica 15 bold'))
303 canv.create_text(420, 220, text="g", fill="white", font=('Helvetica 15 bold'))
304 canv.create_text(620, 220, text="h", fill="white", font=('Helvetica 15 bold'))
305
306 canv.create_rectangle(0, 400, 200, 600, width=0, fill='red')
307 canv.create_rectangle(200, 400, 400, 600, width=0, fill='black')
308 canv.create_rectangle(400, 400, 600, 600, width=0, fill='red')
309 canv.create_rectangle(600, 400, 800, 600, width=0, fill='black')
310 canv.create_text(20, 420, text="i", fill="white", font=('Helvetica 15 bold'))
311 canv.create_text(220, 420, text="j", fill="white", font=('Helvetica 15 bold'))
312 canv.create_text(420, 420, text="k", fill="white", font=('Helvetica 15 bold'))
313 canv.create_text(620, 420, text="l", fill="white", font=('Helvetica 15 bold'))
314
315 canv.create_rectangle(0, 600, 200, 800, width=0, fill='black')
316 canv.create_rectangle(200, 600, 400, 800, width=0, fill='red')
317 canv.create_rectangle(400, 600, 600, 800, width=0, fill='black')
318 canv.create_rectangle(600, 600, 800, 800, width=0, fill='red')
319 canv.create_text(20, 620, text="m", fill="white", font=('Helvetica 15 bold'))
320 canv.create_text(220, 620, text="n", fill="white", font=('Helvetica 15 bold'))
321 canv.create_text(420, 620, text="o", fill="white", font=('Helvetica 15 bold'))
322 canv.create_text(620, 620, text="p", fill="white", font=('Helvetica 15 bold'))
323
324 canv.pack()
325 canv.update()
326 time.sleep(3)
327
328 print("\nRuta agrupada de los jugadores: ", rutaAgrupada)
329 if rutaAgrupada:
330
331     for camino in rutaAgrupada:
332
333         print("\nRuta del jugador PRINCIPAL: ", camino[0])
334         print("Ruta del jugador SECUNDARIO: ", camino[1])
335
336
337         if (camino[0] == 'a'):
338             canv.delete("circulo1")
339             canv.update()
340             ball = canv.create_oval(50, 50, 150, 150, fill="#f4d03f", tags = "
    circulo1")
341             canv.update()
342             time.sleep(1.5)
343
344         if (camino[0] == 'b'):
345             canv.delete("circulo1")
346             canv.update()
347             ball = canv.create_oval(250, 50, 350, 150, fill="#f4d03f", tags = "
    circulo1")
348             canv.update()
349             time.sleep(1.5)
350
```

```
        if (camino[0] == 'c'):
            canv.delete("circulo1")
            canv.update()
            ball = canv.create_oval(450, 50, 550, 150, fill="#f4d03f", tags = "
    circulo1")
            canv.update()
            time.sleep(1.5)

        if (camino[0] == 'd'):
            canv.delete("circulo1")
            canv.update()
            ball = canv.create_oval(650, 50, 750, 150, fill="#f4d03f", tags = "
    circulo1")
            canv.update()
            time.sleep(1.5)

        if (camino[0] == 'e'):
            canv.delete("circulo1")
            canv.update()
            ball = canv.create_oval(50, 250, 150, 350, fill="#f4d03f", tags = "
    circulo1")
            canv.update()
            time.sleep(1.5)

        if (camino[0] == 'f'):
            canv.delete("circulo1")
            canv.update()
            ball = canv.create_oval(250, 250, 350, 350, fill="#f4d03f", tags =
    "circulo1")
            canv.update()
            time.sleep(1.5)

        if (camino[0] == 'g'):
            canv.delete("circulo1")
            canv.update()
            ball = canv.create_oval(450, 250, 550, 350, fill="#f4d03f", tags =
    "circulo1")
            canv.update()
            time.sleep(1.5)

        if (camino[0] == 'h'):
            canv.delete("circulo1")
            canv.update()
            ball = canv.create_oval(650, 250, 750, 350, fill="#f4d03f", tags =
    "circulo1")
            canv.update()
            time.sleep(1.5)

        if (camino[0] == 'i'):
            canv.delete("circulo1")
            canv.update()
            ball = canv.create_oval(50, 450, 150, 550, fill="#f4d03f", tags = "
    circulo1")
            canv.update()
            time.sleep(1.5)

        if (camino[0] == 'j'):
            canv.delete("circulo1")
            canv.update()
            ball = canv.create_oval(250, 450, 350, 550, fill="#f4d03f", tags =
    "circulo1")
            canv.update()
            time.sleep(1.5)

        if (camino[0] == 'k'):
            canv.delete("circulo1")
```

```
            canv.update()
            ball = canv.create_oval(450, 450, 550, 550, fill="#f4d03f", tags =
    "circulo1")
            canv.update()
            time.sleep(1.5)

        if (camino[0] == 'l'):
            canv.delete("circulo1")
            canv.update()
            ball = canv.create_oval(650, 450, 750, 550, fill="#f4d03f", tags =
    "circulo1")
            canv.update()
            time.sleep(1.5)

        if (camino[0] == 'm'):
            canv.delete("circulo1")
            canv.update()
            ball = canv.create_oval(50, 650, 150, 750, fill="#f4d03f", tags = "
    circulo1")
            canv.update()
            time.sleep(1.5)

        if (camino[0] == 'n'):
            canv.delete("circulo1")
            canv.update()
            ball = canv.create_oval(250, 650, 350, 750, fill="#f4d03f", tags =
    "circulo1")
            canv.update()
            time.sleep(1.5)

        if (camino[0] == 'o'):
            canv.delete("circulo1")
            canv.update()
            ball = canv.create_oval(450, 650, 550, 750, fill="#f4d03f", tags =
    "circulo1")
            canv.update()
            time.sleep(1.5)

        if (camino[0] == 'p'):
            canv.delete("circulo1")
            canv.update()
            ball = canv.create_oval(650, 650, 750, 750, fill="#f4d03f", tags =
    "circulo1")
            canv.update()
            time.sleep(1.5)

        # CAMINO DEL JUGADOR SECUNDARIO
        if (camino[1] == 'a'):
            canv.delete("circulo2")
            canv.update()
            ball = canv.create_oval(50, 50, 150, 150, fill="#2ecc71", tags = "
    circulo2")
            canv.update()
            time.sleep(1.5)

        if (camino[1] == 'b'):
            canv.delete("circulo2")
            canv.update()
            ball = canv.create_oval(250, 50, 350, 150, fill="#2ecc71", tags = "
    circulo2")
            canv.update()
            time.sleep(1.5)

        if (camino[1] == 'c'):
            canv.delete("circulo2")
            canv.update()
```

```
            ball = canv.create_oval(450, 50, 550, 150, fill="#2ecc71", tags = "
    circulo2")
            canv.update()
            time.sleep(1.5)

        if (camino[1] == 'd'):
            canv.delete("circulo2")
            canv.update()
            ball = canv.create_oval(650, 50, 750, 150, fill="#2ecc71", tags = "
    circulo2")
            canv.update()
            time.sleep(1.5)

        if (camino[1] == 'e'):
            canv.delete("circulo2")
            canv.update()
            ball = canv.create_oval(50, 250, 150, 350, fill="#2ecc71", tags = "
    circulo2")
            canv.update()
            time.sleep(1.5)

        if (camino[1] == 'f'):
            canv.delete("circulo2")
            canv.update()
            ball = canv.create_oval(250, 250, 350, 350, fill="#2ecc71", tags =
    "circulo2")
            canv.update()
            time.sleep(1.5)

        if (camino[1] == 'g'):
            canv.delete("circulo2")
            canv.update()
            ball = canv.create_oval(450, 250, 550, 350, fill="#2ecc71", tags =
    "circulo2")
            canv.update()
            time.sleep(1.5)

        if (camino[1] == 'h'):
            canv.delete("circulo2")
            canv.update()
            ball = canv.create_oval(650, 250, 750, 350, fill="#2ecc71", tags =
    "circulo2")
            canv.update()
            time.sleep(1.5)

        if (camino[1] == 'i'):
            canv.delete("circulo2")
            canv.update()
            ball = canv.create_oval(50, 450, 150, 550, fill="#2ecc71", tags = "
    circulo2")
            canv.update()
            time.sleep(1.5)

        if (camino[1] == 'j'):
            canv.delete("circulo2")
            canv.update()
            ball = canv.create_oval(250, 450, 350, 550, fill="#2ecc71", tags =
    "circulo2")
            canv.update()
            time.sleep(1.5)

        if (camino[1] == 'k'):
            canv.delete("circulo2")
            canv.update()
            ball = canv.create_oval(450, 450, 550, 550, fill="#2ecc71", tags =
    "circulo2")
```

```
            canv.update()
            time.sleep(1.5)

        if (camino[1] == 'l'):
            canv.delete("circulo2")
            canv.update()
            ball = canv.create_oval(650, 450, 750, 550, fill="#2ecc71", tags = "circulo2")
            canv.update()
            time.sleep(1.5)

        if (camino[1] == 'm'):
            canv.delete("circulo2")
            canv.update()
            ball = canv.create_oval(50, 650, 150, 750, fill="#2ecc71", tags = "circulo2")
            canv.update()
            time.sleep(1.5)

        if (camino[1] == 'n'):
            canv.delete("circulo2")
            canv.update()
            ball = canv.create_oval(250, 650, 350, 750, fill="#2ecc71", tags = "circulo2")
            canv.update()
            time.sleep(1.5)

        if (camino[1] == 'o'):
            canv.delete("circulo2")
            canv.update()
            ball = canv.create_oval(450, 650, 550, 750, fill="#2ecc71", tags = "circulo2")
            canv.update()
            time.sleep(1.5)

        if (camino[1] == 'p'):
            canv.delete("circulo2")
            canv.update()
            ball = canv.create_oval(650, 650, 750, 750, fill="#2ecc71", tags = "circulo2")
            canv.update()
            time.sleep(1.5)

        canv.update()

print("-------RUTAS TERMINADAS-------")
canv.place(x=0, y=0)
ventana.mainloop()
```