

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

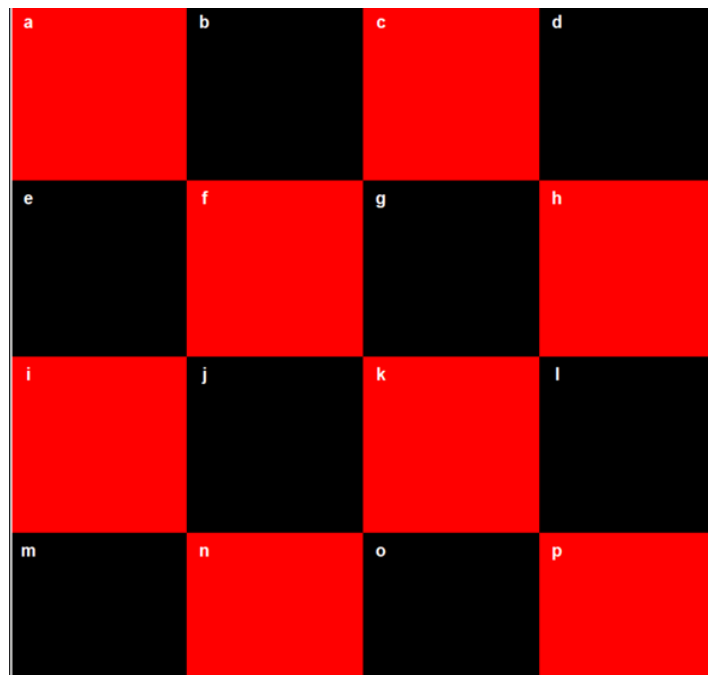


Figura 1: Tablero

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 librería 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: Selecccion del modo de juego y la rutas deseadas

```

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

Elegiendo 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

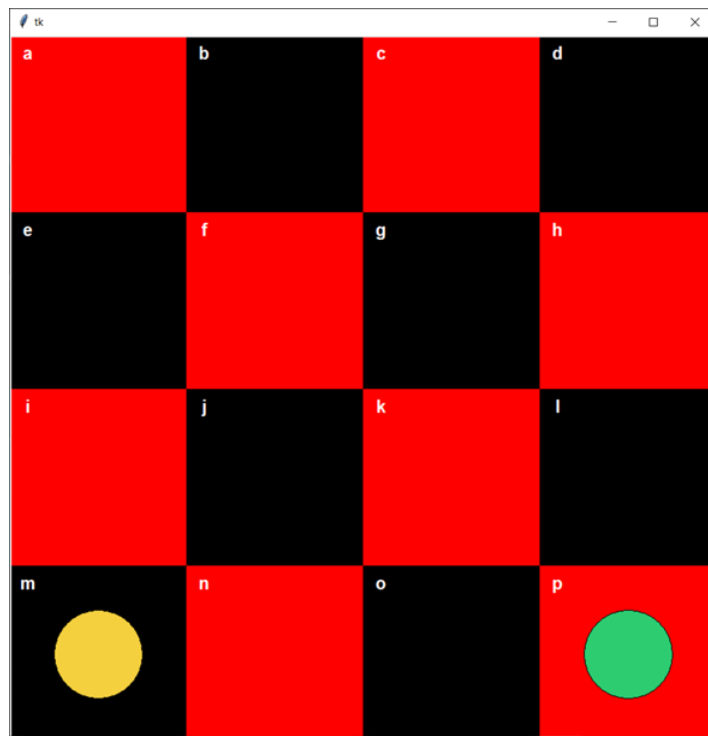


Figura 5: Estado final en la animacion

```

rutasPosiblesPly1.txt
5  ('a', 'f', 'b', 'g', 'f')
6  ('a', 'f', 'b', 'g', 'h')
7  ('a', 'f', 'b', 'g', 'k')
8  ('a', 'f', 'e', 'b', 'a')
9  ('a', 'f', 'e', 'b', 'f')
10 ('a', 'f', 'e', 'b', 'c')
11 ('a', 'f', 'e', 'j', 'f')
12 ('a', 'f', 'e', 'j', 'i')
13 ('a', 'f', 'e', 'j', 'k')
14 ('a', 'f', 'e', 'j', 'n')
15 ('a', 'f', 'g', 'b', 'a')
16 ('a', 'f', 'g', 'b', 'f')
17 ('a', 'f', 'g', 'b', 'c')
18 ('a', 'f', 'g', 'd', 'c')
19 ('a', 'f', 'g', 'd', 'h')
20 ('a', 'f', 'g', 'j', 'f')
21 ('a', 'f', 'g', 'j', 'i')
22 ('a', 'f', 'g', 'j', 'k')
23 ('a', 'f', 'g', 'j', 'n')
24 ('a', 'f', 'g', 'l', 'h')
25 ('a', 'f', 'g', 'l', 'k')
26 ('a', 'f', 'g', 'l', 'p')
27 ('a', 'f', 'j', 'e', 'a')
28 ('a', 'f', 'j', 'e', 'f')
29 ('a', 'f', 'j', 'e', 'i')
30 ('a', 'f', 'j', 'g', 'c')
31 ('a', 'f', 'j', 'g', 'f')
32 ('a', 'f', 'j', 'g', 'h')
33 ('a', 'f', 'j', 'g', 'k')
34 ('a', 'f', 'j', 'm', 'i')
35 ('a', 'f', 'j', 'm', 'n')
36 ('a', 'f', 'j', 'o', 'n')
37 ('a', 'f', 'j', 'o', 'k')
38 ('a', 'f', 'j', 'o', 'p')

```

Figura 6: Rutas posibles del jugador 1

	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

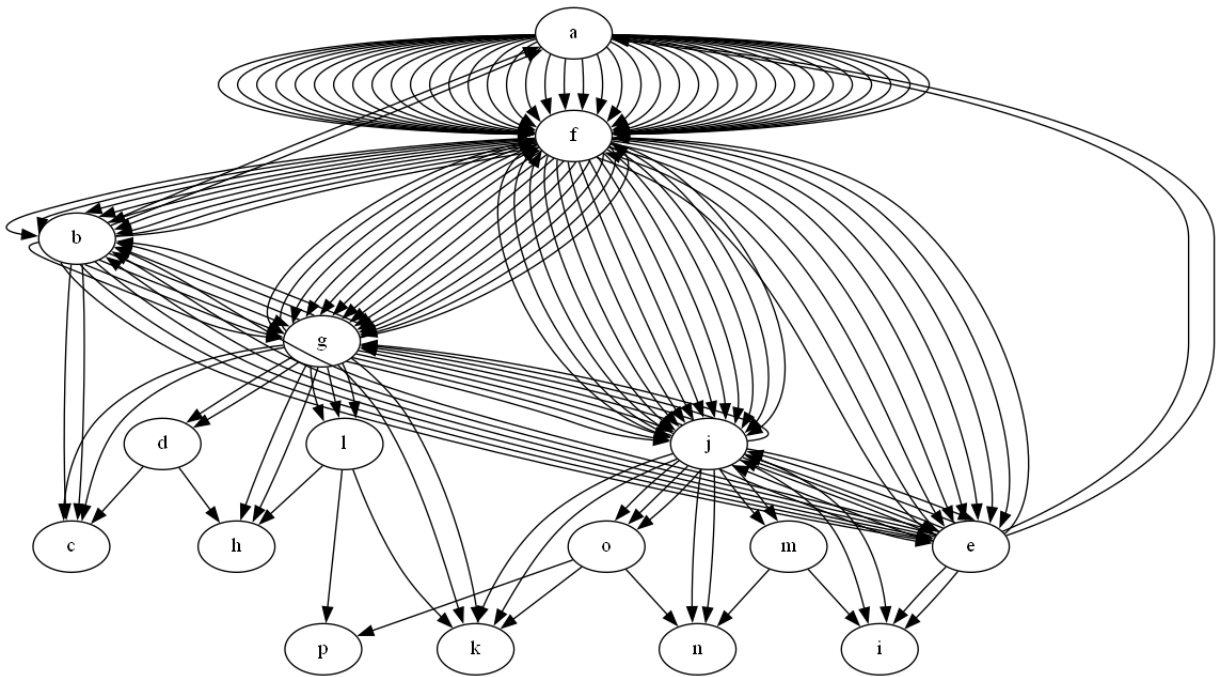


Figura 8: Arbol de las rutas posibles del jugador 1

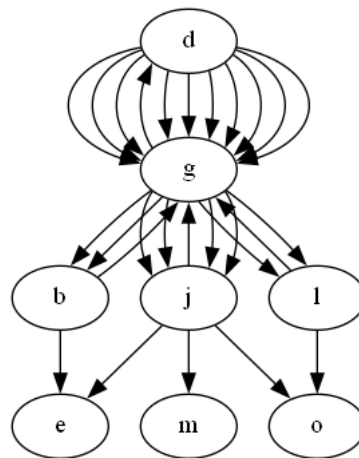


Figura 9: Arbol de las rutas posibles del jugador 2

```

rutasGanadorasPly1.txt
1  ('a', 'f', 'g', 'l', 'p')
2  ('a', 'f', 'j', 'o', 'p')
3

```

Figura 10: Rutas ganadoras del jugador 1

```

rutasGanadorasPly2.txt
1  ('d', 'g', 'j', 'm')
2

```

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:

```

1  # TABLERO NFA
2  from itertools import zip_longest
3  import os
4  import random
5  import re
6  import time
7  from strgen import StringGenerator
8  from tkinter import *
9  import random
10 import networkx as nx
11 import matplotlib.pyplot as plt
12 import graphviz
13
14 switch = {
15     'a': {'B': 'be', 'R': 'f'},
16     'b': {'B': 'eg', 'R': 'afc'},
17     'c': {'B': 'bgd', 'R': 'fh'},
18     'd': {'B': 'g', 'R': 'ch'},
19     'e': {'B': 'bj', 'R': 'afi'},
20     'f': {'B': 'begj', 'R': 'acik'},
21     'g': {'B': 'bdjl', 'R': 'cfhk'},
22     'h': {'B': 'dgl', 'R': 'ck'},
23     'i': {'B': 'ejm', 'R': 'fn'},
24     'j': {'B': 'egmo', 'R': 'fikn'},
25     'k': {'B': 'gjlo', 'R': 'fhnp'},
26     'l': {'B': 'go', 'R': 'hkp'},
27     'm': {'B': 'j', 'R': 'in'},

```

```

28     'n': {'B': 'mjo', 'R': 'ik'},
29     'o': {'B': 'jl', 'R': 'nkp'},
30     'p': {'B': 'lo', 'R': 'k'},
31 }
32
33
34 def routes(current, path):
35     if not path:
36         yield (current,)
37         return
38     first, *newpath = path
39     for state in switch[current][first]:
40         for route in routes(state, newpath):
41             yield (current,) + route
42
43
44 def chess(path, estadoInicial: str, id: str):
45     fileRutasPosibles = open(f'rutasPosibles{id}.txt', 'a')
46     fileRutasGanadoras = open(f'rutasGanadoras{id}.txt', 'a')
47     listSize = 0
48
49     rutas = []
50
51     for i, r in enumerate(routes(estadoInicial, path), 1):
52         fileRutasPosibles.write(str(r) + '\n')
53         rutas.append(list(r))
54
55         if (id == 'Ply1' and r[-1] == 'p'):
56             fileRutasGanadoras.write(str(r) + '\n')
57             listSize += 1
58         if (id == 'Ply2' and r[-1] == 'm'):
59             fileRutasGanadoras.write(str(r) + '\n')
60             listSize += 1
61
62     fileRutasGanadoras.close()
63     fileRutasPosibles.close()
64
65     graficarRuta(rutas, id)
66
67     return listSize
68
69
70 def obtenerRutaGanadoraAleatoria(fileRutas, listSize):
71     fileRutasGanadoras = open(fileRutas, 'r')
72     randomRutaGanadora = 0
73     if (listSize):
74         randomRutaGanadora = random.randint(1, listSize)
75     rutaGanadora = []
76     patron = re.compile('[a-p]')
77
78     count = 1
79     for line in fileRutasGanadoras:
80         if (count == randomRutaGanadora):
81             rutaGanadora = patron.findall(line)
82             break
83         count += 1
84
85     return rutaGanadora
86
87
88 def corregirCruces(rutaGanadora1: list, rutaGanadora2: list):
89     i = 0
90     for l1, l2 in zip_longest(rutaGanadora1, rutaGanadora2, fillvalue=''):
91         if(l1 == l2):
92             print("Se cruzan")
93             rutaGanadora2.insert(i, rutaGanadora2[i-1])

```

```

94         i += 1
95
96     return rutaGanadora2
97
98
99 def recorridoRutasGanadoras(rutaJugadorInicial: list, rutaJugadorSecundario:
100 list):
101     i = 0
102     for l1, l2 in zip_longest(rutaJugadorInicial, rutaJugadorSecundario):
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))
214         #return(jugadorPrincipal, jugadorSecundario)
215         else:
216             print("\n-----Gana el jugador 2-----")
217             return list(zip_longest(rutaGanadoraPly1, rutaGanadoraPly2))
218
219     else:
220         print("\nEmpieza jugador 2 (JUGADOR PRINCIPAL)")
221         rutaGanadoraPly1 = corregirCruces(
222             rutaGanadoraPly2, rutaGanadoraPly1)
223         ganador = recorridoRutasGanadoras(

```

```

223         rutaGanadoraPly2, rutaGanadoraPly1)
224
225         print("Ruta final para jugador1: ", rutaGanadoraPly1)
226         print("Ruta final para jugador2: ", rutaGanadoraPly2)
227
228         if (ganador == "principal"):
229             print("\n-----Gana el jugador 2-----")
230             return list(zip_longest(rutaGanadoraPly2, rutaGanadoraPly1)
231 )
232         else:
233             print("\n-----Gana el jugador 1-----")
234             return list(zip_longest(rutaGanadoraPly2, rutaGanadoraPly1)
235 )
236
237         elif (rutaGanadoraPly1):
238             print("Ruta elegida para jugador1: ", rutaGanadoraPly1)
239             print("\nNo hay ruta ganadora para el jugador 2")
240
241             print("\n-----Gana el jugador 1-----")
242
243             return list(zip_longest(rutaGanadoraPly1, rutaGanadoraPly2))
244
245         elif (rutaGanadoraPly2):
246             print("\nNo hay ruta ganadora para el jugador 1")
247             print("Ruta elegida para jugador2: ", rutaGanadoraPly2)
248
249             print("\n-----Gana el jugador 2-----")
250             return list(zip_longest(rutaGanadoraPly2, rutaGanadoraPly1))
251
252         else:
253             print("\nNo hay rutas ganadoras de ningun jugador")
254             print("\n-----Nadie gana-----")
255
256     if 'rutasGanadorasPly1.txt' in os.listdir('.'):
257         os.remove('./rutasGanadorasPly1.txt')
258     if 'rutasGanadorasPly2.txt' in os.listdir('.'):
259         os.remove('./rutasGanadorasPly2.txt')
260     if 'rutasPosiblesPly1.txt' in os.listdir('.'):
261         os.remove('./rutasPosiblesPly1.txt')
262     if 'rutasPosiblesPly2.txt' in os.listdir('.'):
263         os.remove('./rutasPosiblesPly2.txt')
264
265     print("BIENVENIDO")
266     print("Escoja su modo de juego: ")
267     print("1. Manual")
268     print("2. Automatico")
269     modoJuego = int(input("Inserte el no. de su opci n: "))
270
271     patron = re.compile('[a-p]')
272     rutaAgrupada = []
273
274     if (modoJuego == 1):
275         print("\nElija cuantos jugadores desea: ")
276         print("1. Un jugador")
277         print("2. Dos jugadores")
278         dosJugadores = int(input("Ingrese su opcion: "))
279
280         rutaAgrupada = game(modoJuego, dosJugadores)
281     else:
282         dosJugadores = random.randint(0, 1)
283         rutaAgrupada = game(modoJuego, dosJugadores)
284
285     ventana = Tk()
286     canv = Canvas(ventana, width=800, height=800)
287     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

```

```

351         if (camino[0] == 'c'):
352             canv.delete("circulo1")
353             canv.update()
354             ball = canv.create_oval(450, 50, 550, 150, fill="#f4d03f", tags = "
circulo1")
355             canv.update()
356             time.sleep(1.5)
357
358         if (camino[0] == 'd'):
359             canv.delete("circulo1")
360             canv.update()
361             ball = canv.create_oval(650, 50, 750, 150, fill="#f4d03f", tags = "
circulo1")
362             canv.update()
363             time.sleep(1.5)
364
365         if (camino[0] == 'e'):
366             canv.delete("circulo1")
367             canv.update()
368             ball = canv.create_oval(50, 250, 150, 350, fill="#f4d03f", tags = "
circulo1")
369             canv.update()
370             time.sleep(1.5)
371
372         if (camino[0] == 'f'):
373             canv.delete("circulo1")
374             canv.update()
375             ball = canv.create_oval(250, 250, 350, 350, fill="#f4d03f", tags = "
circulo1")
376             canv.update()
377             time.sleep(1.5)
378
379         if (camino[0] == 'g'):
380             canv.delete("circulo1")
381             canv.update()
382             ball = canv.create_oval(450, 250, 550, 350, fill="#f4d03f", tags = "
circulo1")
383             canv.update()
384             time.sleep(1.5)
385
386         if (camino[0] == 'h'):
387             canv.delete("circulo1")
388             canv.update()
389             ball = canv.create_oval(650, 250, 750, 350, fill="#f4d03f", tags = "
circulo1")
390             canv.update()
391             time.sleep(1.5)
392
393         if (camino[0] == 'i'):
394             canv.delete("circulo1")
395             canv.update()
396             ball = canv.create_oval(50, 450, 150, 550, fill="#f4d03f", tags = "
circulo1")
397             canv.update()
398             time.sleep(1.5)
399
400         if (camino[0] == 'j'):
401             canv.delete("circulo1")
402             canv.update()
403             ball = canv.create_oval(250, 450, 350, 550, fill="#f4d03f", tags = "
circulo1")
404             canv.update()
405             time.sleep(1.5)
406
407         if (camino[0] == 'k'):
408             canv.delete("circulo1")

```

```

409         canv.update()
410         ball = canv.create_oval(450, 450, 550, 550, fill="#f4d03f", tags =
"circulo1")
411         canv.update()
412         time.sleep(1.5)
413
414         if (camino[0] == 'l'):
415             canv.delete("circulo1")
416             canv.update()
417             ball = canv.create_oval(650, 450, 750, 550, fill="#f4d03f", tags =
"circulo1")
418             canv.update()
419             time.sleep(1.5)
420
421         if (camino[0] == 'm'):
422             canv.delete("circulo1")
423             canv.update()
424             ball = canv.create_oval(50, 650, 150, 750, fill="#f4d03f", tags = "
circulo1")
425             canv.update()
426             time.sleep(1.5)
427
428         if (camino[0] == 'n'):
429             canv.delete("circulo1")
430             canv.update()
431             ball = canv.create_oval(250, 650, 350, 750, fill="#f4d03f", tags =
"circulo1")
432             canv.update()
433             time.sleep(1.5)
434
435         if (camino[0] == 'o'):
436             canv.delete("circulo1")
437             canv.update()
438             ball = canv.create_oval(450, 650, 550, 750, fill="#f4d03f", tags =
"circulo1")
439             canv.update()
440             time.sleep(1.5)
441
442         if (camino[0] == 'p'):
443             canv.delete("circulo1")
444             canv.update()
445             ball = canv.create_oval(650, 650, 750, 750, fill="#f4d03f", tags =
"circulo1")
446             canv.update()
447             time.sleep(1.5)
448
449         # CAMINO DEL JUGADOR SECUNDARIO
450         if (camino[1] == 'a'):
451             canv.delete("circulo2")
452             canv.update()
453             ball = canv.create_oval(50, 50, 150, 150, fill="#2ecc71", tags = "
circulo2")
454             canv.update()
455             time.sleep(1.5)
456
457         if (camino[1] == 'b'):
458             canv.delete("circulo2")
459             canv.update()
460             ball = canv.create_oval(250, 50, 350, 150, fill="#2ecc71", tags = "
circulo2")
461             canv.update()
462             time.sleep(1.5)
463
464         if (camino[1] == 'c'):
465             canv.delete("circulo2")
466             canv.update()

```

```

467         ball = canv.create_oval(450, 50, 550, 150, fill="#2ecc71", tags = "
circulo2")
468         canv.update()
469         time.sleep(1.5)
470
471         if (camino[1] == 'd'):
472             canv.delete("circulo2")
473             canv.update()
474             ball = canv.create_oval(650, 50, 750, 150, fill="#2ecc71", tags = "
circulo2")
475             canv.update()
476             time.sleep(1.5)
477
478         if (camino[1] == 'e'):
479             canv.delete("circulo2")
480             canv.update()
481             ball = canv.create_oval(50, 250, 150, 350, fill="#2ecc71", tags = "
circulo2")
482             canv.update()
483             time.sleep(1.5)
484
485         if (camino[1] == 'f'):
486             canv.delete("circulo2")
487             canv.update()
488             ball = canv.create_oval(250, 250, 350, 350, fill="#2ecc71", tags =
"circulo2")
489             canv.update()
490             time.sleep(1.5)
491
492         if (camino[1] == 'g'):
493             canv.delete("circulo2")
494             canv.update()
495             ball = canv.create_oval(450, 250, 550, 350, fill="#2ecc71", tags =
"circulo2")
496             canv.update()
497             time.sleep(1.5)
498
499         if (camino[1] == 'h'):
500             canv.delete("circulo2")
501             canv.update()
502             ball = canv.create_oval(650, 250, 750, 350, fill="#2ecc71", tags =
"circulo2")
503             canv.update()
504             time.sleep(1.5)
505
506         if (camino[1] == 'i'):
507             canv.delete("circulo2")
508             canv.update()
509             ball = canv.create_oval(50, 450, 150, 550, fill="#2ecc71", tags = "
circulo2")
510             canv.update()
511             time.sleep(1.5)
512
513         if (camino[1] == 'j'):
514             canv.delete("circulo2")
515             canv.update()
516             ball = canv.create_oval(250, 450, 350, 550, fill="#2ecc71", tags =
"circulo2")
517             canv.update()
518             time.sleep(1.5)
519
520         if (camino[1] == 'k'):
521             canv.delete("circulo2")
522             canv.update()
523             ball = canv.create_oval(450, 450, 550, 550, fill="#2ecc71", tags =
"circulo2")

```

```

524         canv.update()
525         time.sleep(1.5)
526
527         if (camino[1] == 'l'):
528             canv.delete("circulo2")
529             canv.update()
530             ball = canv.create_oval(650, 450, 750, 550, fill="#2ecc71", tags =
"circulo2")
531             canv.update()
532             time.sleep(1.5)
533
534         if (camino[1] == 'm'):
535             canv.delete("circulo2")
536             canv.update()
537             ball = canv.create_oval(50, 650, 150, 750, fill="#2ecc71", tags = "
circulo2")
538             canv.update()
539             time.sleep(1.5)
540
541         if (camino[1] == 'n'):
542             canv.delete("circulo2")
543             canv.update()
544             ball = canv.create_oval(250, 650, 350, 750, fill="#2ecc71", tags =
"circulo2")
545             canv.update()
546             time.sleep(1.5)
547
548         if (camino[1] == 'o'):
549             canv.delete("circulo2")
550             canv.update()
551             ball = canv.create_oval(450, 650, 550, 750, fill="#2ecc71", tags =
"circulo2")
552             canv.update()
553             time.sleep(1.5)
554
555         if (camino[1] == 'p'):
556             canv.delete("circulo2")
557             canv.update()
558             ball = canv.create_oval(650, 650, 750, 750, fill="#2ecc71", tags =
"circulo2")
559             canv.update()
560             time.sleep(1.5)
561
562         canv.update()
563
564     print("-----RUTAS TERMINADAS-----")
565     canv.place(x=0, y=0)
566     ventana.mainloop()

```