Instituto Politécnico Nacional

Escuela Superior de Cómputo

Ciudad de México, México, 13 de octubre de 2024 .

Programa Tablero

Ponce Fragoso Emmanuel

Teoria de Computaciòn
Genaro Juarez
5BM2

Índice

1. Inicio

1.1. Objetivo de la práctica

El objetivo de la práctica es desarrollar y comprender la lógica detrás de un juego de ajedrez en un tablero de 5x5, mediante la creación de un programa que simule la generación y el análisis de jugadas. A través de esta práctica, se busca que los participantes adquieran habilidades en la programación, específicamente en el manejo de estructuras de datos y algoritmos recursivos. El programa permite ingresar jugadas de manera automática o manual, evaluando los resultados de cada jugada y clasificándolos como ganadores o perdedores.

1.2. Objetivo específico

Es implementar un programa que permita simular un juego de ajedrez en un tablero de 5x5, facilitando la entrada de jugadas de forma manual y automática. Esto incluye la creación de un algoritmo que evalúe cada jugada realizada, si resulta en una victoria o derrota, así como la generación de un archivo que registre las secuencias de jugadas. A través de este proceso, se busca que los participantes comprendan el funcionamiento del juego, mejoren sus habilidades en la programación utilizando estructuras de datos adecuadas y desarrollen su capacidad para aplicar técnicas de análisis lógico en la resolución de problemas.

1.3. Introducción

Autómata finito no determinista
Un autómata finito no determinista consta de:

$$A = \{Q, \Sigma, \delta, q_0, F\}$$

donde:

- Q es un conjunto finito de estados.
- Σ es un conjunto finito de símbolos de entrada.
- δ es una función de transición que toma como argumentos un estado y un símbolo de entrada, y devuelve un conjunto de posibles estados.
- q_0 es el estado inicial.
- F es un conjunto finito de estados finales.

Autómata finito no determinista
Un autómata finito no determinista se define de manera similar a un autómata determinista, pero con la peculiaridad de que la función δ puede retornar un conjunto de estados, permitiendo múltiples posibles transiciones para un símbolo dado.

Propiedades y Aplicaciones del Autómata Finito No Determinista
Los autómatas finitos no deterministas (NFA) son capaces de aceptar un lenguaje regular a través de múltiples caminos de ejecución. Esto significa que una cadena de entrada es aceptada si al menos un camino lleva a un estado de aceptación, lo que les permite reconocer patrones complejos de manera más flexible. Los NFA son especialmente útiles en análisis léxico, como en compiladores, donde se utilizan para identificar palabras clave y operadores.

1.4. Metodología

Para el desarrollo del programa "tablebroz" *buscador* se propuso el uso de archivos `txt`, como archivos de marcado. Estos tienen la finalidad de representar tablas de transiciones de autómatas. De esta manera, estas tablas son interpretadas para su uso en distintos contextos.

A continuación, un ejemplo de uso para un autómata no determinista:

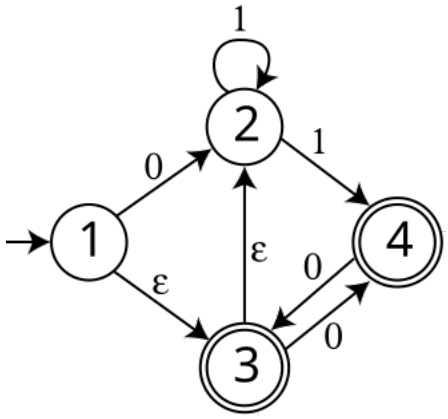

Diagrama de transiciones del NFA.

2. Desarrollo

2.1. NFA_Movimientos.py

2.1.1. Librerías Importadas

```
import os
import random
import networkx as nx
import matplotlib.pyplot as plt
```

Libreria OS
Esta librería permite interactuar con el entorno es muy útil para hacer tareas relacionadas con archivos o directorios, como asegurar de que un archivo o carpeta existe antes de crear o leerlo.

Librería random
Para generar números aleatorios, utilizados en la generación de jugadas.

Librería networkx y matplotlib
Se importan, pero no se utilizan en el código que proporcionaste. Podrían ser parte de una futura extensión del código para graficar o manipular gráficos de alguna manera.

2.1.2. Variables Globales

```
actual_jugada = ""
archivo_jugada = ""
archivo_losers = ""
archivo_winners = ""
jugada_final = ""
lis_jugada = []
```

Fucionamiento:
Se definen variables globales que se usarán en varias partes del código para almacenar información sobre la jugada actual, los nombres de los archivos donde se guardarán las jugadas, y una lista de jugadas.

2.1.3. Función Principal (main)

```
def main():
    global actual_jugada, archivo_jugada, archivo_losers, archivo_winners, jugada_final,
        lis_jugada

    # Menu para elegir la forma de entrada
    print("Seleccione el metodo de entrada para las jugadas:")
    print("1. Generar automaticamente")
    print("2. Ingresar manualmente")
    opcion = input("Ingrese 1 o 2: ")

    # Eliminacion de archivos existentes
    files_to_remove = [
        "Ajedrez 5x5/jugada_t_1.txt",
        "Ajedrez 5x5/jugada_g_1.txt",
        "Ajedrez 5x5/jugada_p_1.txt",
        "Ajedrez 5x5/jugada_t_2.txt",
        "Ajedrez 5x5/jugada_g_2.txt",
        "Ajedrez 5x5/jugada_p_2.txt"
    ]
    for file in files_to_remove:
```

```
        if os.path.exists(file):
            os.remove(file)

    if opcion == '1':
        Inico_Secuenciadad_jugadaimientos = random.randint(5, 100)  # Tamano automatico
        print(f"Tamano de la cadena: {Inico_Secuenciadad_jugadaimientos}")

        # Configuracion de archivos para jugador 1
        archivo_jugada = "Ajedrez 5x5/jugada_t_1.txt"
        archivo_losers = "Ajedrez 5x5/jugada_p_1.txt"
        archivo_winners = "Ajedrez 5x5/jugada_g_1.txt"
        jugada_final = "25,"

        # Obtencion de jugadas para jugador 1
        obt_play(Inico_Secuenciadad_jugadaimientos, 'r')
        obt_play(actual_jugada, 0, Inico_Secuenciadad_jugadaimientos, 1, "1,")
        save_jug()

        # Limpiar lista y actual jugada
        lis_jugada.clear()
        actual_jugada = ""

        # Configuracion de archivos para jugador 2
        obt_play(Inico_Secuenciadad_jugadaimientos, 'b')
        archivo_jugada = "Ajedrez 5x5/jugada_t_2.txt"
        archivo_losers = "Ajedrez 5x5/jugada_p_2.txt"
        archivo_winners = "Ajedrez 5x5/jugada_g_2.txt"
        jugada_final = "21,"

        # Obtencion de jugadas para jugador 2
        obt_play(actual_jugada, 0, Inico_Secuenciadad_jugadaimientos, 5, "5,")
        save_jug()

    elif opcion == '2':
        Inico_Secuenciadad_jugadaimientos = int(input("Ingrese el tamano de la cadena (
            entre 5 y 100): "))
        if Inico_Secuenciadad_jugadaimientos < 5 or Inico_Secuenciadad_jugadaimientos >
            100:
            print("El tamano debe estar entre 5 y 100. Saliendo...")
            return

        # Ingresar manualmente la cadena para el jugador 1
        actual_jugada = input("Ingrese la cadena para el jugador 1 (tamano debe ser
            igual): ")
        if actual_jugada != actual_jugada[:Inico_Secuenciadad_jugadaimientos]:
            print("La cadena debe ser del mismo tamano que el especificado. Saliendo..."
                )
            return

        archivo_jugada = "Ajedrez 5x5/jugada_t_1.txt"
        archivo_losers = "Ajedrez 5x5/jugada_p_1.txt"
        archivo_winners = "Ajedrez 5x5/jugada_g_1.txt"
        jugada_final = "25,"
        obt_play(actual_jugada, 0, Inico_Secuenciadad_jugadaimientos, 1, "1,")
        save_jug()

        lis_jugada.clear()
        actual_jugada = ""

        # Ingresar manualmente la cadena para el jugador 2
```

```
        actual_jugada = input("Ingrese la cadena para el jugador 2 (tamano debe ser
            igual): ")
        if actual_jugada != actual_jugada[:Inico_Secuenciadad_jugadaimientos]:
            print("La cadena debe ser del mismo tamano que el especificado. Saliendo..."
                )
            return

        archivo_jugada = "Ajedrez 5x5/jugada_t_2.txt"
        archivo_losers = "Ajedrez 5x5/jugada_p_2.txt"
        archivo_winners = "Ajedrez 5x5/jugada_g_2.txt"
        jugada_final = "21,"
        obt_play(actual_jugada, 0, Inico_Secuenciadad_jugadaimientos, 5, "5,")
        save_jug()
    else:
        print("Opcion no valida. Saliendo...")
        return
```

Fucionamiento de la Función:
Aquí se define la función main y se declaran varias variables como globales. Esto significa que estas variables pueden ser accedidas y modificadas en otras partes del código fuera de esta función.
Las variables declaradas son:
1. actual_jugada: Almacena la jugada actual.
2 .archivo_jugada: Ruta del archivo donde se guarda la jugada.
3. archivo_losers: Ruta del archivo donde se guardan las jugadas perdedoras.
4. archivo_winners: Ruta del archivo donde se guardan las jugadas ganadoras.
5. jugada_final: Representa el resultado final de la jugada.
6. lis_jugada: Lista que probablemente almacena varias jugadas.

Menú de selección:
Aquí se presenta un menú al usuario, donde puede elegir entre dos opciones: generar las jugadas automáticamente o ingresarlas manualmente.

Eliminación de archivos existentes:
Se define una lista de archivos que deben eliminarse al inicio del programa para asegurarse de que no haya datos previos. El bucle for verifica si cada archivo existe y, si es así, lo elimina utilizando os.remove().

Condicional para selección de jugadas automáticas y Manuales:
Si el usuario elige la opción 1, se genera un tamaño de cadena aleatorio entre 5 y 100 usando random.randint(), y se imprime el tamaño elegido.
Si el usuario elige la opción 2, se le pide que ingrese manualmente el tamaño de la cadena. Si no se encuentra en el rango adecuado, se muestra un mensaje y la función finaliza.
Se solicita la cadena de entrada para el jugador 1, y se valida que su tamaño sea igual al especificado. Si no es así, se muestra un mensaje de error y se termina la función.

Configuración de archivos y jugadas para el jugador 1 y 2:
Se configuran las rutas de los archivos para el jugador 1 o 2, y se establece jugada_final con un valor inicial.
Se llama a la función obt_play para generar las jugadas y guardarlas en el archivo correspondiente. La función save_jug() se utiliza para guardar las jugadas en los archivos.

Manejo de opciones no válidas:
Si el usuario ingresa algo diferente de 1 o 2, se imprime un mensaje de error y se finaliza la función.

2.1.4. Función obt_play()

Primera definición de obt_play()

```
def obt_play(Inico_Secuenciadad_jugadaimientos, ultima):
    global actual_jugada
    actual_jugada = "".join(["b" if random.randint(0, 1) == 0 else "r" for _ in range(
        Inico_Secuenciadad_jugadaimientos - 1)]) + ultima
    print(actual_jugada)
```

Fucionamiento:

Propósito: Generar una cadena de movimientos aleatorios de los jugadores ('b' para un color y 'r' para otro).

Se utiliza una comprensión de lista para generar una cadena aleatoria de 'b' y 'r', añadiendo el último carácter que se pasa como argumento.

Segunda definición de obt_play()

```
def obt_play(jugada, Inico_Secuencia, Final_Secuencia, state, sec_arm):
    global lis_jugada
    if Inico_Secuencia == Final_Secuencia:
        lis_jugada.append(sec_arm)
    else:
        if actual_jugada[Inico_Secuencia] == 'r':
            if state == 1:
                obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 2, sec_arm + "2,"
                    )
                obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 5, sec_arm + "6,"
                    )
            elif state == 2:
                obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 5, sec_arm + "6,"
                    )
                obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 8, sec_arm + "8,"
                    )
            elif state == 3:
                obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 2, sec_arm + "2,"
                    )
                obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 4, sec_arm + "4,"
                    )
                obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 8, sec_arm + "8,"
                    )
            elif state == 4:
                obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 3, sec_arm + "3,"
                    )
                obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 5, sec_arm + "5,"
                    )
                obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 9, sec_arm + "9,"
                    )
            elif state == 5:
                obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 4, sec_arm + "4,"
                    )
                obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 10, sec_arm + "
                    10,")
            elif state == 6:
                obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 1, sec_arm + "1,"
                    )
                obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 7, sec_arm + "7,"
                    )
                obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 11, sec_arm + "
                    11,")
```

```
        elif state == 7:
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 2, sec_arm + "2,"
                )
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 6, sec_arm + "6,"
                )
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 8, sec_arm + "8,"
                )
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 12, sec_arm + "
                12,")
        elif state == 8:
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 3, sec_arm + "3,"
                )
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 7, sec_arm + "7,"
                )
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 9, sec_arm + "9,"
                )
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 13, sec_arm + "
                13,")
        elif state == 9:
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 4, sec_arm + "4,"
                )
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 8, sec_arm + "8,"
                )
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 10, sec_arm + "
                10,")
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 14, sec_arm + "
                14,")
        elif state == 10:
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 5, sec_arm + "5,"
                )
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 9, sec_arm + "9,"
                )
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 15, sec_arm + "
                15,")
        elif state == 11:
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 6, sec_arm + "6,"
                )
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 12, sec_arm + "
                12,")
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 16, sec_arm + "
                16,")
        elif state == 12:
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 7, sec_arm + "7,"
                )
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 11, sec_arm + "
                11,")
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 13, sec_arm + "
                13,")
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 17, sec_arm + "
                17,")
        elif state == 13:
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 8, sec_arm + "8,"
                )
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 12, sec_arm + "
                12,")
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 14, sec_arm + "
                14,")
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 18, sec_arm + "
                18,")
        elif state == 14:
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 9, sec_arm + "9,"
```

```
            )
        obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 13, sec_arm + "
            13,")
        obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 15, sec_arm + "
            15,")
        obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 19, sec_arm + "
            19,")
    elif state == 15:
        obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 10, sec_arm + "
            10,")
        obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 14, sec_arm + "
            14,")
        obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 20, sec_arm + "
            20,")
    elif state == 16:
        obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 11, sec_arm + "
            11,")
        obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 17, sec_arm + "
            17,")
        obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 21, sec_arm + "
            21,")
    elif state == 17:
        obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 12, sec_arm + "
            12,")
        obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 16, sec_arm + "
            16,")
        obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 18, sec_arm + "
            18,")
        obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 22, sec_arm + "
            22,")
    elif state == 18:
        obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 13, sec_arm + "
            13,")
        obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 17, sec_arm + "
            17,")
        obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 19, sec_arm + "
            19,")
        obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 23, sec_arm + "
            23,")
    elif state == 19:
        obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 14, sec_arm + "
            14,")
        obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 18, sec_arm + "
            18,")
        obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 20, sec_arm + "
            20,")
        obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 24, sec_arm + "
            24,")
    elif state == 20:
        obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 15, sec_arm + "
            15,")
        obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 19, sec_arm + "
            19,")
        obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 20, sec_arm + "
            25,")
    elif state == 21:
        obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 16, sec_arm + "
            16,")
        obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 22, sec_arm + "
            22,")
    elif state == 22:
```

```
                obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 17, sec_arm + "
                    17,")
                obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 21, sec_arm + "
                    21,")
                obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 23, sec_arm + "
                    23,")
            elif state == 23:
                obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 18, sec_arm + "
                    18,")
                obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 22, sec_arm + "
                    22,")
                obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 24, sec_arm + "
                    24,")
            elif state == 24:
                obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 19, sec_arm + "
                    19,")
                obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 23, sec_arm + "
                    23,")
                obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 24, sec_arm + "
                    25,")
            elif state == 25:
                obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 20, sec_arm + "
                    20,")
                obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 24, sec_arm + "
                    24,")
        else:
            if state == 1:
                obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 7, sec_arm + "7,"
                    )
            elif state == 2:
                obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 1, sec_arm + "1,"
                    )
                obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 3, sec_arm + "3,"
                    )
                obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 7, sec_arm + "7,"
                    )
            elif state == 3:
                obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 7, sec_arm + "7,"
                    )
                obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 9, sec_arm + "9,"
                    )
            elif state == 4:
                obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 8, sec_arm + "8,"
                    )
                obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 10, sec_arm + "
                    10,")
            elif state == 5:
                obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 9, sec_arm + "9,"
                    )
            elif state == 6:
                obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 2, sec_arm + "2,"
                    )
                obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 12, sec_arm + "
                    12,")
            elif state == 7:
                obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 1, sec_arm + "1,"
                    )
                obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 3, sec_arm + "3,"
                    )
                obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 11, sec_arm + "
                    11,")
```

```
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 13, sec_arm + "
                13,")
        elif state == 8:
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 2, sec_arm + "2,"
                )
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 4, sec_arm + "4,"
                )
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 12, sec_arm + "
                12,")
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 14, sec_arm + "
                14,")
        elif state == 9:
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 3, sec_arm + "3,"
                )
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 5, sec_arm + "5,"
                )
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 13, sec_arm + "
                13,")
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 15, sec_arm + "
                15,")
        elif state == 10:
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 4, sec_arm + "4,"
                )
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 14, sec_arm + "
                14,")
        elif state == 11:
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 7, sec_arm + "7,"
                )
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 17, sec_arm + "
                17,")
        elif state == 12:
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 6, sec_arm + "6,"
                )
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 8, sec_arm + "8,"
                )
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 16, sec_arm + "
                16,")
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 18, sec_arm + "
                18,")
        elif state == 13:
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 7, sec_arm + "7,"
                )
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 9, sec_arm + "9,"
                )
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 17, sec_arm + "
                17,")
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 19, sec_arm + "
                19,")
        elif state == 14:
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 8, sec_arm + "8,"
                )
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 10, sec_arm + "
                10,")
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 18, sec_arm + "
                18,")
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 20, sec_arm + "
                20,")
        elif state == 15:
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 9, sec_arm + "9,"
                )
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 19, sec_arm + "
```

```
            19,")
        elif state == 16:
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 12, sec_arm + "
                12,")
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 22, sec_arm + "
                22,")
        elif state == 17:
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 11, sec_arm + "
                11,")
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 13, sec_arm + "
                13,")
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 21, sec_arm + "
                21,")
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 23, sec_arm + "
                23,")
        elif state == 18:
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 12, sec_arm + "
                12,")
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 14, sec_arm + "
                14,")
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 22, sec_arm + "
                22,")
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 24, sec_arm + "
                24,")
        elif state == 19:
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 13, sec_arm + "
                13,")
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 15, sec_arm + "
                15,")
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 23, sec_arm + "
                23,")
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 25, sec_arm + "
                25,")
        elif state == 20:
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 14, sec_arm + "
                14,")
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 24, sec_arm + "
                24,")
        elif state == 21:
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 17, sec_arm + "
                17,")
        elif state == 22:
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 16, sec_arm + "
                16,")
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 18, sec_arm + "
                18,")
        elif state == 23:
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 17, sec_arm + "
                17,")
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 19, sec_arm + "
                19,")
        elif state == 24:
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 18, sec_arm + "
                18,")
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 20, sec_arm + "
                20,")
        elif state == 25:
            obt_play(jugada, Inico_Secuencia + 1, Final_Secuencia, 19, sec_arm + "
                19,")
```

Fucionamiento:

Propósito: Procesar y generar las jugadas posibles en función del estado actual y la secuencia.
Parámetros:
jugada: la cadena de movimientos.
Inico_Secuencia: índice actual en la jugada.
Final_Secuencia: longitud total de la jugada.
state: estado actual del proceso.
sec_arm: secuencia de movimientos acumulados.
La función usa una lógica recursiva para explorar todas las combinaciones de jugadas posibles. Dependiendo del carácter actual en la secuencia (ya sea 'r' o 'b'), llama a sí misma para generar las siguientes jugadas.

2.1.5. Función save_jug()

```
def save_jug():
    global lis_jugada
    try:
        with open(archivo_jugada, "a") as fw1, open(archivo_winners, "a") as fw2, open(
            archivo_losers, "a") as fw3:
            fw2.write(actual_jugada + "\n")
            for x in lis_jugada:
                if x[-3:] == jugada_final:
                    fw2.write(x + "\n")
                else:
                    fw3.write(x + "\n")
                    fw1.write(x + "\n")
    except Exception as e:
        print(e)
```

Fucionamiento de la Función:
Propósito: Guardar las jugadas generadas en archivos correspondientes.

1. Se abren tres archivos para escritura.
2. La jugada actual se guarda en el archivo de ganadores.
3. Se itera sobre lis_jugada para guardar las jugadas finales y las que no cumplieron las condiciones.

2.1.6. Función main

Se llama la funciòn main().

```
if __name__ == "__main__":
    main()
```

2.2. Arbol.py

2.2.1. Librerías Importadas

```
import networkx as nx
import matplotlib.pyplot as plt
```

Librería networkx
Se utiliza para crear, manipular y estudiar la estructura, dinámica y funciones de grafos.

Libreria matplotlib.pyplot
Se utiliza para crear visualizaciones gráficas.

2.2.2. Función leer_y_filtrar_rutas_y_movimientos(archivo)

```
# Funcion para leer y filtrar rutas y movimientos de cada archivo
def leer_y_filtrar_rutas_y_movimientos(archivo):
    secuencias_movimientos = []
    secuencias_rutas = []
    lineas_validas = []

    with open(archivo, 'r') as f:
        for line in f:
            line = line.strip()
            if line:
                # Verificar si la linea tiene una coma para dividirla en dos partes
                if ',' in line:
                    partes = line.split(',')
                    if len(partes) >= 2:
                        movimientos = partes[0].strip()
                        ruta = partes[1:]  # Ruta numerica

                        # Filtrar lineas segun el estado final
                        if ruta.count('25') >= 1 or ruta.count('21') >= 1:
                            # Encontrar el indice de la primera ocurrencia de 25 o 21
                            indice_final = min(
                                (ruta.index('25') if '25' in ruta else len(ruta)),
                                (ruta.index('21') if '21' in ruta else len(ruta))
                            )
                            # Mantener solo la parte de la ruta hasta el indice final
                            ruta_corregida = ruta[:indice_final + 1]  # Incluir el
                                estado final
                        else:
                            ruta_corregida = ruta

                        secuencias_movimientos.append(movimientos)
                        secuencias_rutas.append(ruta_corregida)
                        lineas_validas.append(f"{movimientos}," + ','.join(
                            ruta_corregida))  # Agregar la linea valida
                else:
                    print(f"Formato no esperado en la linea: {line}")

    # Escribir lineas validas de vuelta al archivo
    with open(archivo, 'w') as f:
        for linea in lineas_validas:
            f.write(linea + '\n')

    return secuencias_movimientos, secuencias_rutas
```

Propósito: Leer un archivo de texto que contiene movimientos y rutas, filtrar la información relevante y guardar las líneas válidas de vuelta en el archivo.
Variables Locales:
secuencias_movimientos: Almacena los movimientos extraídos.
secuencias_rutas: Almacena las rutas asociadas a los movimientos.
lineas_validas: Almacena las líneas que son válidas para ser escritas de nuevo al archivo.
Lógica de la función:
1. Abre el archivo y itera sobre cada línea.
2. Limpia espacios en blanco al principio y al final.
3. Verifica si la línea contiene una coma, dividiéndola en dos partes: movimientos y ruta.
4. Filtra las líneas según si la ruta contiene '25' o '21', manteniendo solo la parte de la ruta hasta el índice final.
5. Agrega los movimientos y la ruta corregida a las listas.
6. Reescribe el archivo solo con las líneas válidas.
Retorno de la función: Devuelve las listas de movimientos y rutas.

2.2.3. Función graficar_ruta(movimientos, rutas, jugador)

```
# Funcion para graficar el arbol de movimientos
def graficar_ruta(movimientos, rutas, jugador):
    G = nx.DiGraph()
    estado_inicial, estado_final = (1, 25) if jugador == 1 else (5, 21)

    # Anadir nodo inicial y final
    G.add_node(estado_inicial)  # Nodo inicial
    G.add_node(estado_final)    # Nodo final

    # Crear las transiciones segun los movimientos y rutas
    for secuencia_mov, secuencia_ruta in zip(movimientos, rutas):
        estado_actual = estado_inicial
        for transicion in secuencia_ruta:
            try:
                transicion_int = int(transicion)
                if transicion_int:  # Asegurarse de que la transicion no sea cero
                    G.add_edge(estado_actual, transicion_int)
                    estado_actual = transicion_int
            except ValueError:
                print(f"Transicion no valida: {transicion}")

    # Solo agregar la arista de entrada a los estados finales
    if estado_actual != estado_final:
        G.add_edge(estado_actual, estado_final)

    return G
```

Propósito: Crear un grafo dirigido (DiGraph) para representar las transiciones de los movimientos de un jugador y se definen el estado inicial y el estado final según el jugador.
Creación del Grafo: Itera a través de las secuencias de movimientos y rutas, creando aristas (edges) en el grafo entre los estados según las rutas proporcionadas y agrega una arista desde el estado actual al estado final si no se ha llegado a él.
Retorno de la función: Devuelve el grafo creado.

2.2.4. Función dibujar_y_guardar_grafo(G, jugador)

```
# Funcion para dibujar y guardar el grafo
def dibujar_y_guardar_grafo(G, jugador):
    plt.figure(figsize=(12, 10))

    # Usar el spring_layout con mayor separacion entre nodos
    pos = nx.spring_layout(G, k=2, iterations=50)  # Aumentar k para mas separacion
    labels = {nodo: str(nodo) for nodo in G.nodes()}  # Etiquetar nodos con su
        identificador

    # Dibujar nodos y bordes (edges) con flechas
    nx.draw(G, pos, with_labels=True, node_size=500, node_color='skyblue', font_size=10,
         font_weight='bold', edge_color='gray', width=1.0, arrows=True)

    # Anadir etiquetas sin color rojo
    nx.draw_networkx_labels(G, pos, labels, font_size=12)  # Eliminado font_color='red'

    # Guardar en PNG
    plt.title(f"arbol de Movimientos del Jugador {jugador}")
    plt.savefig(f"jugador_{jugador}_arbol_movimientos.png", bbox_inches='tight')
    plt.close()
```

Propósito: Dibujar y guardar el grafo en un archivo PNG y se establece el tamaño de la figura para el gráfico..
1. Usa spring_layout para determinar la posición de los nodos en el grafo, asegurando una mayor separación.
2. Dibuja el grafo con los nodos y aristas, configurando colores y tamaños.
3. Establece un título para la gráfica y guarda la figura en un archivo PNG.

2.2.5. Lectura y Filtrado de Archivos

```
movimientos_j1, rutas_j1 = leer_y_filtrar_rutas_y_movimientos('Ajedrez 5x5/jugada_t_1.
    txt')
movimientos_j2, rutas_j2 = leer_y_filtrar_rutas_y_movimientos('Ajedrez 5x5/jugada_t_2.
    txt')
```

Lee y filtra los movimientos y rutas de los archivos de texto para los dos jugadores.

2.2.6. Graficar y Guardar Árbol de Movimientos

```
G1 = graficar_ruta(movimientos_j1, rutas_j1, jugador=1)
dibujar_y_guardar_grafo(G1, jugador=1)

G2 = graficar_ruta(movimientos_j2, rutas_j2, jugador=2)
dibujar_y_guardar_grafo(G2, jugador=2)
```

Genera y guarda los grafos de movimientos para ambos jugadores.

2.2.7. Mensaje Final

```
print("Se han generado los archivos PNG para ambos jugadores y los archivos .txt han
    sido actualizados.")
```

Imprime un mensaje al final indicando que los archivos se han generado y actualizado correctamente.

3. tablero5x5.py

3.1. Librerías Importadas

```
import time
from tkinter import *
from tkinter import messagebox as MessageBox
from turtle import Turtle, Screen
```

Librería time
Para controlar el tiempo en las animaciones (usado para sleep)..

Libreria tkinter
Para crear ventanas y mostrar mensajes de información.

Libreria turtle
Para dibujar gráficos en una ventana.

3.2. Función dibujar_tablero

```
def dibujar_tablero(tortuga):
    tortuga.speed(0)
    tortuga.penup()
    tortuga.setpos(-250, 250)  # Ajustar posicion inicial
    tortuga.pendown()
    casilla = 1

    for fila in range(5):
        for columna in range(5):
            tortuga.pencolor('gold')  # Color dorado para el borde
            tortuga.fillcolor('black' if (fila + columna) % 2 != 0 else 'white')
            tortuga.begin_fill()

            # Dibujar cada casilla
            for _ in range(4):
                tortuga.forward(100)
                tortuga.right(90)
            tortuga.end_fill()

            tortuga.penup()
            # Ajustar la posicion del numero
            tortuga.forward(10)  # Ajuste horizontal
            tortuga.right(90)
            tortuga.forward(25)
            tortuga.left(90)

            # Escribir el numero en dorado
            tortuga.pencolor('gold')
            tortuga.write(casilla, False, align="left", font=("Arial", 18, "bold"))
            casilla += 1

            # Regresar a la posicion del cuadrado
            tortuga.left(90)
            tortuga.forward(25)
            tortuga.right(90)
            tortuga.backward(10)  # Volver al inicio
```

```
            tortuga.pendown()
            # Moverse a la siguiente casilla
            tortuga.forward(100)

        # Ir a la siguiente fila
        tortuga.penup()
        tortuga.backward(500)  # Regresar al inicio de la fila
        tortuga.right(90)
        tortuga.forward(100)
        tortuga.left(90)
        tortuga.pendown()

    tortuga.penup()
    tortuga.setpos(-250, 250)  # Reposicionar para el siguiente elemento
    tortuga.pendown()
```

Dibuja un tablero de ajedrez de 5x5 utilizando la tortuga. Alterna colores entre negro y blanco para cada casilla y escribe números del 1 al 25 en cada casilla.

3.3. Función desplazar_ficha

```
def desplazar_ficha(tortuga, valor, color_ficha):
    if valor == '' or not valor.isdigit():  # Validacion adicional
        print(f"Valor invalido: '{valor}'")
        return

    # Coordenadas de las posiciones en el tablero
    coordenadas = [[-200 + (100 * (i % 5)), 200 - (100 * (i // 5))] for i in range(25)]

    # Configurar la posicion inicial de la ficha
    tortuga.penup()
    tortuga.clear()

    # Animar el movimiento hacia la nueva posicion
    for paso in range(10):  # Numero de pasos de la animacion
        x = coordenadas[int(valor) - 1][0] * (paso + 1) / 10
        y = coordenadas[int(valor) - 1][1] * (paso + 1) / 10
        tortuga.setpos(x, y)
        time.sleep(0.05)  # Controlar la velocidad de la animacion

    # Dibujar la ficha en la posicion final
    tortuga.setpos(coordenadas[int(valor) - 1])  # Ir a la posicion final
    tortuga.pendown()
    tortuga.color(color_ficha)  # Cambiar el color de la ficha
    tortuga.begin_fill()
    tortuga.circle(25)  # Tamano de la ficha
    tortuga.end_fill()
    time.sleep(1)
```

Mueve una ficha en el tablero a la posición correspondiente al valor dado. Usa una animación para mover la ficha suavemente y la dibuja en la posición final.

3.4. Función cargar_jugadas

```
def cargar_jugadas(archivo, jugadas):
    movimientos = ""
    with open(archivo) as arch:
        for contador, linea in enumerate(arch):
            if contador == 0:
```

```
                movimientos += linea.strip()  # Eliminar espacios y saltos de linea
            else:
                jugada = linea.strip().split(',')  # Eliminar espacios y saltos de linea
                jugada = [j.strip() for j in jugada if j.strip()]  # Limpiar entradas
                    vacias
                if jugada:  # Solo anadir si no esta vacio
                    jugadas.append(jugada)
    return movimientos
```

3.5. Función reiniciar_juego

```
def iniciar_juego(jugadas1, jugadas2, movimientos):
    if jugadas1 and jugadas2:
        ganador = 1
        turno = 1
        estado1 = 0
        estado2 = 0
        fila1 = 0
        fila2 = 0
        col1 = 0
        col2 = 0
        contador1 = 0  # Contador de movimientos para jugador 1
        contador2 = 0  # Contador de movimientos para jugador 2

        while ganador != 0:
            if turno == 1:
                temp = 0
                estado1 = jugadas1[fila1][col1] if col1 < len(jugadas1[fila1]) else None

                if estado1 is None:
                    break  # Si no hay mas movimientos disponibles

                salto = 0

                if estado1 == estado2:
                    contador = 0
                    for i in jugadas1:
                        temp = i[col1] if col1 < len(i) else None
                        contador += 1
                        if temp != estado2:
                            fila1 = contador - 1
                            salto = 0
                            break
                        salto = 1
                    estado1 = temp

                if salto == 0:
                    desplazar_ficha(j1, estado1, 'red')  # Ficha roja para el jugador 1
                    col1 += 1
                    contador1 += 1  # Incrementar el contador del jugador 1

                turno = 2

                # Verificar condiciones de victoria
                if contador1 > 0 and estado1 == '25':  # Player 1 wins
                    ganador = 0
                    MessageBox.showinfo("Felicitaciones!", "Ha ganado el jugador 1")
                    reiniciar_juego()
                    return  # Salir del juego
```

```
            elif turno == 2:
                temp = 0
                estado2 = jugadas2[fila2][col2] if col2 < len(jugadas2[fila2]) else None

                if estado2 is None:
                    break  # Si no hay mas movimientos disponibles

                salto = 0

                if estado2 == estado1:
                    contador = 0
                    for i in jugadas2:
                        temp = i[col2] if col2 < len(i) else None
                        contador += 1
                        if temp != estado1:
                            fila2 = contador - 1
                            salto = 0
                            break
                        salto = 1
                    estado2 = temp

                if salto == 0:
                    desplazar_ficha(j2, estado2, 'blue')  # Ficha azul para el jugador 2
                    col2 += 1
                    contador2 += 1  # Incrementar el contador del jugador 2

                turno = 1

                # Verificar condiciones de victoria
                if contador2 > 0 and estado2 == '21':  # Player 2 wins
                    ganador = 0
                    MessageBox.showinfo("Felicitaciones!", "Ha ganado el jugador 2")
                    reiniciar_juego()
                    return  # Salir del juego

        # Verificar si ambos han terminado sin ganar
        if contador1 > 0 and contador2 > 0 and (estado1 != '25' and estado2 != '21'):
            MessageBox.showinfo("Fin del juego", "Ninguno gano, ambos alcanzaron sus
                limites de movimiento.")
            reiniciar_juego()

    elif not jugadas1 and not jugadas2:
        MessageBox.showinfo("Mensaje", "No hay jugadas ganadoras para los jugadores")
    elif not jugadas1:
        MessageBox.showinfo("Felicitaciones!", "Gana el jugador 2 por default")
    elif not jugadas2:
        MessageBox.showinfo("Felicitaciones!", "Gana el jugador 1 por default")
```

Esta función maneja el flujo del juego. Alterna entre los turnos de los dos jugadores, mueve las fichas según las jugadas, y verifica si hay un ganador. En caso de que ambos jugadores terminen sin ganar, muestra un mensaje de que el juego ha terminado.

3.6. Configuración de la ventana y ejecución del juego

```
# Crear la pantalla y el tablero
pantalla = Screen()  # crear la pantalla
pantalla.setup(width=500, height=500)  # Establecer tamano de la ventana

jugadas1 = []
jugadas2 = []
```

```
movj1 = cargar_jugadas(archivo="Ajedrez 5x5/jugada_t_1.txt", jugadas=jugadas1)
movj2 = cargar_jugadas(archivo="Ajedrez 5x5/jugada_t_2.txt", jugadas=jugadas2)

# Crear una tortuga para dibujar el tablero
tortuga_tablero = Turtle()
dibujar_tablero(tortuga_tablero)

# Crear tortugas para los jugadores
j1 = Turtle()
j2 = Turtle()
j1.speed(0)  # Velocidad de la tortuga mas rapida para la animacion
j2.speed(0)

# Ocultar las tortugas
j1.hideturtle()
j2.hideturtle()

# Iniciar el juego
iniciar_juego(jugadas1=jugadas1, jugadas2=jugadas2, movimientos=movj1)

pantalla.mainloop()  # Mantener la ventana abierta
```

1. Configuración de la ventana: Se crea la ventana del juego con dimensiones específicas.
2. Carga de jugadas: Se cargan las jugadas de dos archivos de texto y se inicializa el tablero.
3. Tortugas: Se crean tortugas para el tablero y los jugadores, configurando la velocidad y ocultándolas para que no interfieran con la visualización.
4. Ejecutar el juego: Se inicia el juego con las jugadas cargadas.

4. Salidas

4.1. Grafos

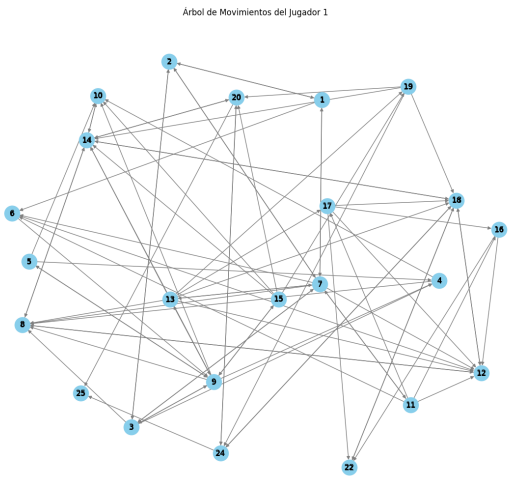

Figura 1: Grafo de las Posibles Jugadas Ganadoras del Jugador 1

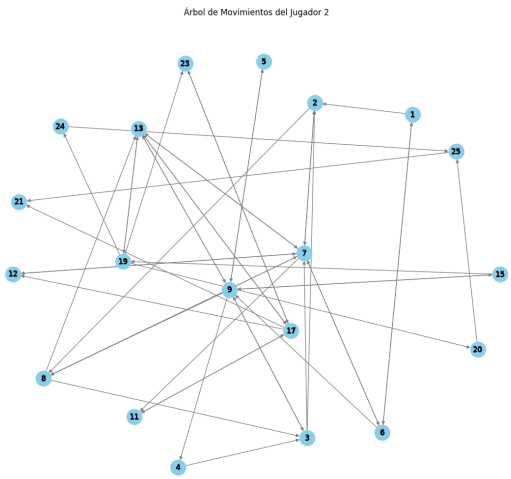

Figura 2: Grafo de las Posibles Jugadas Ganadoras del Jugador 2

Fragmento de Rutas Ganadoras del Jugador 2: bbbrrrbbbb
5,9,3,7,2,8,3,7,11,17,21
5,9,3,7,2,8,3,7,13,17,21
5,9,3,7,2,8,3,9,13,17,21
5,9,3,7,2,8,13,7,11,17,21
5,9,3,7,2,8,13,7,13,17,21
5,9,3,7,2,8,13,9,13,17,21
5,9,3,7,2,8,13,17,11,17,21
5,9,3,7,2,8,13,17,13,17,21
5,9,3,7,2,8,13,17,21
5,9,3,7,2,8,13,17,23,17,21
5,9,3,7,2,8,13,19,13,17,21
5,9,3,7,2,8,13,19,23,17,21
5,9,3,7,6,1,2,7,11,17,21
5,9,3,7,6,1,2,7,13,17,21
5,9,3,7,6,1,6,9,13,17,21
5,9,3,7,6,7,2,7,11,17,21
5,9,3,7,6,7,2,7,13,17,21
5,9,3,7,8,3,2,7,11,17,21
5,9,3,7,8,3,2,7,13,17,21
5,9,3,7,8,7,2,7,11,17,21
5,9,3,7,8,7,2,7,13,17,21
5,9,3,7,12,7,2,7,11,17,21
5,9,3,7,12,7,2,7,13,17,21
5,9,3,9,4,3,2,7,11,17,21
5,9,3,9,4,3,2,7,13,17,21
5,9,3,9,8,3,2,7,11,17,21
5,9,3,9,8,3,2,7,13,17,21

5,9,3,9,8,7,2,7,11,17,21
5,9,3,9,8,7,2,7,13,17,21
5,9,5,9,4,3,2,7,11,17,21
5,9,5,9,4,3,2,7,13,17,21
5,9,5,9,8,3,2,7,11,17,21
5,9,5,9,8,3,2,7,13,17,21
5,9,5,9,8,7,2,7,11,17,21
5,9,5,9,8,7,2,7,13,17,21
5,9,13,7,2,8,3,7,11,17,21
5,9,13,7,2,8,3,7,13,17,21
5,9,13,7,2,8,3,9,13,17,21
5,9,13,7,2,8,13,7,11,17,21
5,9,13,7,2,8,13,7,13,17,21
5,9,13,7,2,8,13,9,13,17,21
5,9,13,7,2,8,13,17,11,17,21
5,9,13,7,2,8,13,17,13,17,21
5,9,13,7,2,8,13,17,21
5,9,13,7,2,8,13,17,23,17,21
5,9,13,7,2,8,13,19,13,17,21
5,9,13,7,2,8,13,19,23,17,21
5,9,13,7,6,1,2,7,11,17,21
5,9,13,7,6,1,2,7,13,17,21
5,9,13,7,6,1,6,9,13,17,21
5,9,13,7,6,7,2,7,11,17,21
5,9,13,7,6,7,2,7,13,17,21
5,9,13,7,8,3,2,7,11,17,21
5,9,13,7,8,3,2,7,13,17,21
5,9,13,7,8,7,2,7,11,17,21
5,9,13,7,8,7,2,7,13,17,21
5,9,13,7,12,7,2,7,11,17,21
5,9,13,7,12,7,2,7,13,17,21
5,9,13,9,4,3,2,7,11,17,21
5,9,13,9,4,3,2,7,13,17,21
5,9,13,9,8,3,2,7,11,17,21
5,9,13,9,8,3,2,7,13,17,21
5,9,13,9,8,7,2,7,11,17,
5,9,13,9,8,7,2,7,13,17,21
5,9,13,17,12,7,2,7,11,17,21
5,9,13,17,12,7,2,7,13,17,21
5,9,15,9,4,3,2,7,11,17,21
5,9,15,9,4,3,2,7,13,17,21
5,9,15,9,8,3,2,7,11,17,21
5,9,15,9,8,3,2,7,13,17,21
5,9,15,9,8,7,2,7,11,17,21
5,9,15,9,8,7,2,7,13,17,21

4.2. Tablero

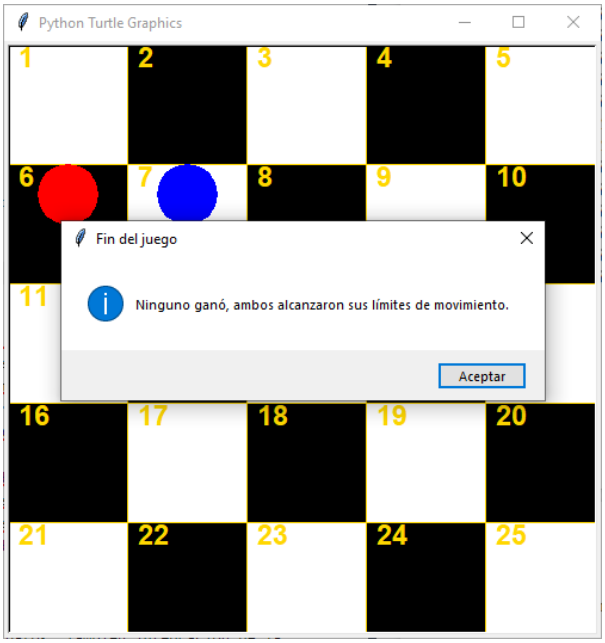

Figura 3: Tablero Caso Perdedor Ambos

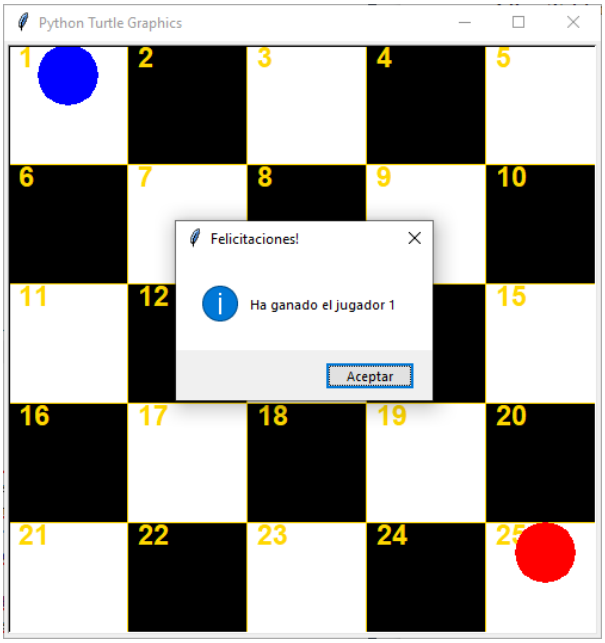

Figura 4: Tablero Caso Ganador

5. Conclusion

Los tres códigos proporcionan ejemplos de cómo implementar autómatas no deterministas (NFA) en Python utilizando la biblioteca PLY (Python Lex-Yacc). Cada código representa un enfoque para construir y evaluar un NFA que reconoce un conjunto específico de cadenas. El primer código muestra una implementación básica de un NFA, donde se definen los estados, las transiciones y el estado de aceptación. Esta implementación permite realizar la transición entre estados según los caracteres de la cadena de entrada, destacando el principio fundamental de los NFA: pueden tener múltiples transiciones posibles para un mismo estado y símbolo. El segundo código introduce un NFA más complejo que reconoce expresiones regulares simples, mostrando la flexibilidad de estos autómatas al manejar patrones más variados en las cadenas de entrada. El tercer código lleva esto un paso más allá al permitir la inclusión de un conjunto de transiciones epsilon, que permite cambios de estado sin consumir un símbolo de entrada. Este último aspecto resalta una de las características distintivas de los NFA, donde la no determinación se puede dar tanto por la multiplicidad de transiciones como por la posibilidad de realizar transiciones vacías. En conjunto, estos ejemplos ilustran la capacidad de los NFA para reconocer lenguajes complejos y su utilidad en aplicaciones como el análisis léxico y la compilación, proporcionando una base sólida para entender cómo funcionan estos autómatas en la práctica.

6. Referencias

@bookhopcroft2001introduction, author = John E. Hopcroft and Rajeev D. Ullman and Jeffrey D. Ullman, title = Introduction to Automata Theory, Languages, and Computation, publisher = Addison-Wesley, year = 2001, url = https://www-2.dc.uba.ar/staff/becher/Hopcroft-Motwani-Ullman-2001.pdf

@miscullman2010jeffslides, author = Jeffrey D. Ullman, title = Jeff's Slides for CS154, Spring 2010, year = 2010, note = Stanford University, url = http://infolab.stanford.edu/ ullman/ialc/spr10/spr10.htmlLECTURE