INSTITUTO POLITÉCNICO NACIONAL

ESCOM

INTELIGENCIA ARTIFICIAL

BUSQUEDA ALEATORIA

MARTÍNEZ CHÁVEZ JORGE ALEXIS

6CV3

09 SEPTIEMBRE 2024

Introducción

En el presente documento se presentarán 2 algoritmos de búsqueda aleatoria, la búsqueda aleatoria la cual trata de generar un gran numero de posibles soluciones “ de forma aleatoria” dentro de un rango determinado donde se cumplan las reglas y condiciones del problema, estas posibles soluciones se someten a prueba y se seleccionara el o los mejores resultados que se hayan obtenido.

El primer algoritmo que trataremos es para poder resolver una ecuación la cual debemos poder encontrar el valor mínimo de la función, la cual consta de dos variables que deben de estar en el rango de -4.5 y 4.5 para ellos que manera aleatoria le daremos valor a estas 2 variables, calcularemos el resultado e iremos guardado que el mejor resultado nos dé.

El segundo algoritmo que trataremos es el juego del gato pero en un tablero de 4x4, en este caso el jugador deberá se seleccionar donde tira y la computadora de forma aleatoria deberá de seleccionar una de las casillas disponibles, es decir, no tendrá inteligencia en intentar bloquear al usuario simplemente tirara en donde haya una casilla vacía.

Algoritmos

Algoritmo “Mínimos”

Primero debemos de definir la función de la cual debemos de encontrar los mínimos

```
# PRIMERO DEFINIMOS LA FUNCIÓN
def f(x, y):
    return ((1.5 - x + x * y) ** 2 + (2.25 - x + x * y ** 2) ** 2 + (2.625 - x + x * y ** 3) ** 2)
```

Definimos el rango de “x” hacia abajo y hacia arriba 4.5 respectivamente, así como el rango de “y” el cual es el mismo de “x”. También definimos el numero de intentos que buscaremos en este caso serán 10000 iteraciones las que se realizarán.

```
# Número de iteraciones
n_iteraciones = 10000

# CREAMOS NUESTRAS VARIABLES SOLUCIÓN U ÓPTIMAS
valor_minimo = float('inf')
mejor_x = None
mejor_y = None
```

Ahora creamos las variables en donde almacenaremos cada vez que se encuntre un mejor resultado, almacenaremos el resultado y el valor de las variables que dieron ese resultado.

```
# CREAMOS NUESTRAS VARIABLES SOLUCIÓN U ÓPTIMAS
valor_minimo = float('inf')
mejor_x = None
mejor_y = None
```

Hacemos un for en que va desde el 1 hasta el 10000 que es el numero de iteraciones que realizaremos, dentro de este bucle a nuestro valor de “x” y de “y” les asignaremos un valor aleatorio entre los rangos previamente definidos, posteriormente mandaremos llamar la función en donde pasamos como parametros estos valores

aleatorios para poder evaluar la función y nos regresara el valor de la función. Depues compararemos el valor regresado con nuestra varibales donde guardariamos el mejor resultado, si este es menor, entonces cambiaremos nuestras variables minimos con estos nuevos valores asognados y su resultado, cada vez que se haya cumplido esta condición imprimeremos que se ha encontrado un mejor minimo.

```
# REALIZAMOS LAS ITERACIONES
for iteracion in range(1, n_iteraciones + 1):
    # EN CADA ITERACIÓN GENERAMOS VALORES ALEATORIOS DENTRO DE LOS RANGOS DEFINIDOS
    x = random.uniform(x_minimo, x_maximo)
    y = random.uniform(y_minimo, y_maximo)

    # EVALUAMOS LA FUNCIÓN CON ESTOS VALORES
    valor_actual = f(x, y)

    # SI EL RESULTADO ES MENOR AL VALOR MÍNIMO GUARDADO, LO ACTUALIZAMOS
    if valor_actual < valor_minimo:
        valor_minimo = valor_actual
        mejor_x = x
        mejor_y = y

        # IMPRIMIMOS CADA VEZ QUE ENCONTRAMOS UN NUEVO MÍNIMO
        print(f'Iteración: {iteracion}, Nuevo mínimo encontrado: {valor_minimo}')
        print(f'Valores: x = {mejor_x}, y = {mejor_y}\n')
```

En caso contario simplemeten se continuara con el bucle hasta terminar con el numero de iteraciones que hemos asignado, al final imprimimos cual fue nuestra solucion encontrada.

```
# IMPRIMIMOS EL RESULTADO FINAL
print(f'\nEl valor mínimo encontrado en {n_iteraciones} iteraciones es: {valor_minimo}')
print(f'Valores de x e y que minimizan la función: x = {mejor_x}, y = {mejor_y}')
```

Si ejecutamos el programa nos dara el siguiente resultado.

```
PS C:\Users\jorge\OneDrive\Documentos\JORGE\ESCOM\9_SEMESTRE\IA\LABORATORIO\1_BUSQUEDA_ALEATORIA> python minimos.py
Iteración: 1, Nuevo mínimo encontrado: 39.80526215541863
Valores: x = -2.0228137238114465, y = 0.5002249001857635

Iteración: 7, Nuevo mínimo encontrado: 18.70693005966828
Valores: x = 2.2621915073165955, y = -1.0144911039425546

Iteración: 8, Nuevo mínimo encontrado: 12.197903398876868
Valores: x = 0.03411936789349479, y = -2.4156379055265274

Iteración: 9, Nuevo mínimo encontrado: 2.6695891532013345
Valores: x = 1.3908608104318256, y = -0.39442819660861606

Iteración: 18, Nuevo mínimo encontrado: 1.6970179893195043
Valores: x = 1.542569133005907, y = 0.08376672547523789

Iteración: 111, Nuevo mínimo encontrado: 0.7308931054084804
Valores: x = 3.2260626966231634, y = 0.37418604726659144

Iteración: 200, Nuevo mínimo encontrado: 0.06483344411285046
Valores: x = 2.9966477785307664, y = 0.5500835613905144

Iteración: 834, Nuevo mínimo encontrado: 0.052903510209727354
Valores: x = 2.9386469454599036, y = 0.43264627886356255

Iteración: 2236, Nuevo mínimo encontrado: 0.029774111192836513
Valores: x = 2.8864016893673323, y = 0.4320390280112587

Iteración: 2724, Nuevo mínimo encontrado: 0.01517080932364945
Valores: x = 2.7573364469404265, y = 0.4462602644039251

Iteración: 4656, Nuevo mínimo encontrado: 0.009453093296937023
Valores: x = 2.7863063791061213, y = 0.4446396515557556


El valor mínimo encontrado en 10000 iteraciones es: 0.009453093296937023
Valores de x e y que minimizan la función: x = 2.7863063791061213, y = 0.4446396515557556
```

Algoritmo “Gato 4x4”

Primero debemos de crear una función con la cual crearemos el tablero del juego, el cual consiste en una matriz bidimensional de 4x4.

```
# FUNCIÓN PARA CREAR EL TABLERO DE 4 FILAS X 4 COLUMNAS
def crear_tablero():
    return [[' ' for _ in range(4)] for _ in range(4)]
```

Ahora debemos de crear las reglas del juego, empezando con las condiciones si es que hay algún ganador, los casos donde hay un ganador es cuando hay 4 fichas del mismo jugador en línea. Para ello debemos de comprobar cada fila y cada columna, así como las dos diagonales, para ello se recibí como parámetro el tablero el cual será la matriz y el jugador el cual es el tipo de ficha. Por tanto, compararemos esta ficha con el tablero recibido si se encuentra que es ganador regresara un true en caso de que no haya encontrado un ganador regresara un false.

```
# FUNCIÓN PARA CREAR EL TABLERO DE 4 FILAS X 4 COLUMNAS
def crear_tablero():
    return [[' ' for _ in range(4)] for _ in range(4)]
```

Ahora comprobamos si el tablero esta lleno, se llegara a esto si es que hay un empate, simplemente comprobamos que cada posición del arreglo esta lleno, en caso de que si entonces regresara false y si esta lleno regresara un true.

```
# COMPROBAMOS SI EL TABLERO ESTÁ LLENO
# ESTO SE DARÁ CUANDO NO HAY UN GANADOR, ENTONCES SERÁ UN EMPATE
def tablero_lleno(tablero):
    return all([tablero[i][j] != ' ' for i in range(4) for j in range(4)])
```

Ahora haremos la tirada del jugador, en donde se le solicitara que ingrese el numero de la fila y el numero de la columna en donde desea tirar, para ello debe de digitar entre el 0 y el 3, esto lo comprobaremos un con except que se traerá a la hora de intentar ingresar el valor en el arreglo, en caso de que no esté dentro del rango, entra dentro de la excepción y entonces enviara un mensaje de que la entrada no es válida. En caso de que si, hará una comprobación de que donde se desea tirar esta vacío en caso de que no se pueda tirar ahí mostrara un mensaje en donde indica que la tirada no es válida.

Todo esto estará dentro de un bucle que no se romperá hasta que el usuario ingrese una posición correcta.

```
# HACEMOS LA TIRADA DEL USUARIO
def movimiento_jugador(tablero):
    # HACEMOS UN BUCLE PARA PODER VERIFICAR QUE SE INGRESÓ UNA POSICIÓN CORRECTA
    while True:
        try:
            # PRIMERO LE SOLICITAMOS LA FILA EN UN RANGO DEL 0 AL 3
            fila = int(input("Ingresa la fila (0-3): "))
            # DESPUÉS LE SOLICITAMOS LA COLUMNA EN UN RANGO DEL 0 AL 3
            columna = int(input("Ingresa la columna (0-3): "))
            # COMPROBAMOS LA LEGALIDAD DEL TIRO, PRIMERO QUE LA FILA Y COLUMNA INGRESADA ESTÉN DENTRO DEL RANGO
            # DESPUÉS QUE LA POSICIÓN DONDE SE DESEA TIRAR ESTÉ VACÍA
            if 0 <= fila < 4 and 0 <= columna < 4 and tablero[fila][columna] == ' ':
                # SI LA LEGALIDAD DEL TIRO ES CORRECTA, SE AGREGA UNA X QUE ES LA FICHA DEL JUGADOR
                tablero[fila][columna] = 'X'
                break
            else:
                # EN CASO DE QUE NO SEA VÁLIDO, DEBERÁ VOLVER A PEDIR QUE EL USUARIO INGRESE LAS POSICIONES
                print("Movimiento no válido. Intenta de nuevo.")
        except ValueError:
            print("Entrada no válida. Ingresa números entre 0 y 3.")
```

Ahora la tirada de la computadora será a partir de un bucle en donde de forma aleatoria seleccionara una fila y una columna y comprobara si esta posición está vacía, en caso de que si entonces hará su tirada, en caso de que no entonces buscara de forma aleatoria otra posición.

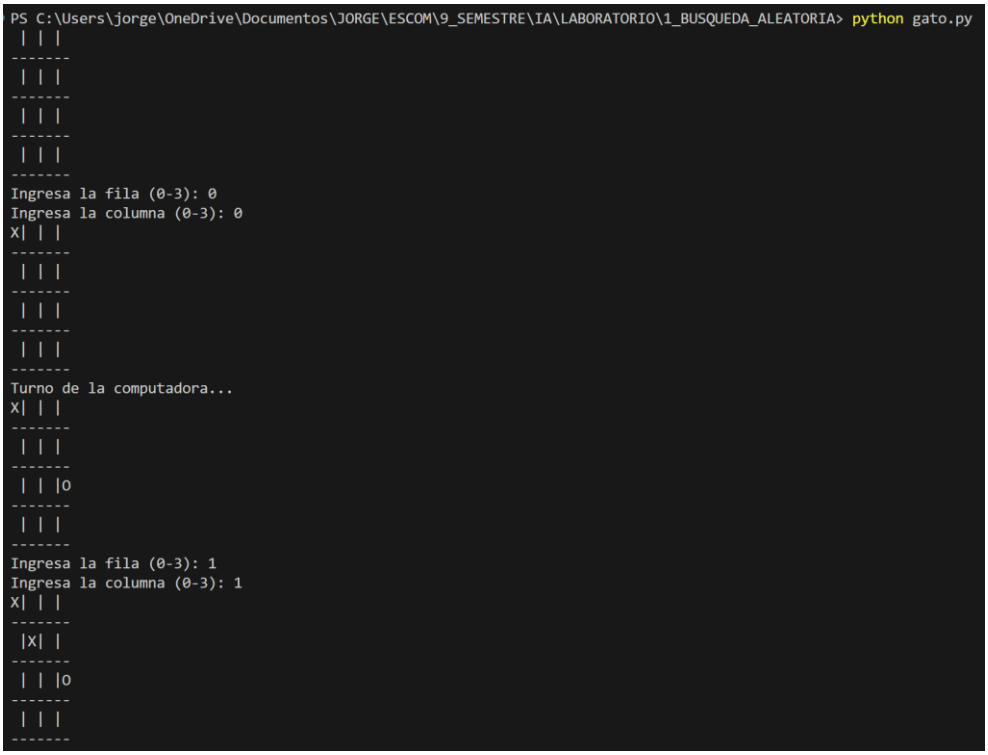
```
# AHORA HAREMOS LA TIRADA DE LA COMPUTADORA
def movimiento_computadora(tablero):
    # HAREMOS UN BUCLE PARA PODER ENCONTRAR UNA POSICIÓN VACÍA
    while True:
        # PRIMERO ELIGE UN NÚMERO ALEATORIO ENTRE 0 Y 3 PARA LA FILA
        fila = random.randint(0, 3)
        # DESPUÉS ELIGE UN NÚMERO ALEATORIO ENTRE 0 Y 3 PARA LA COLUMNA
        columna = random.randint(0, 3)
        # COMPRUEBA SI LA POSICIÓN ESTÁ VACÍA
        if tablero[fila][columna] == ' ':
            # EN CASO DE QUE SÍ, AGREGA UNA O QUE ES LA FICHA DE LA COMPUTADORA
            tablero[fila][columna] = 'O'
            break
```

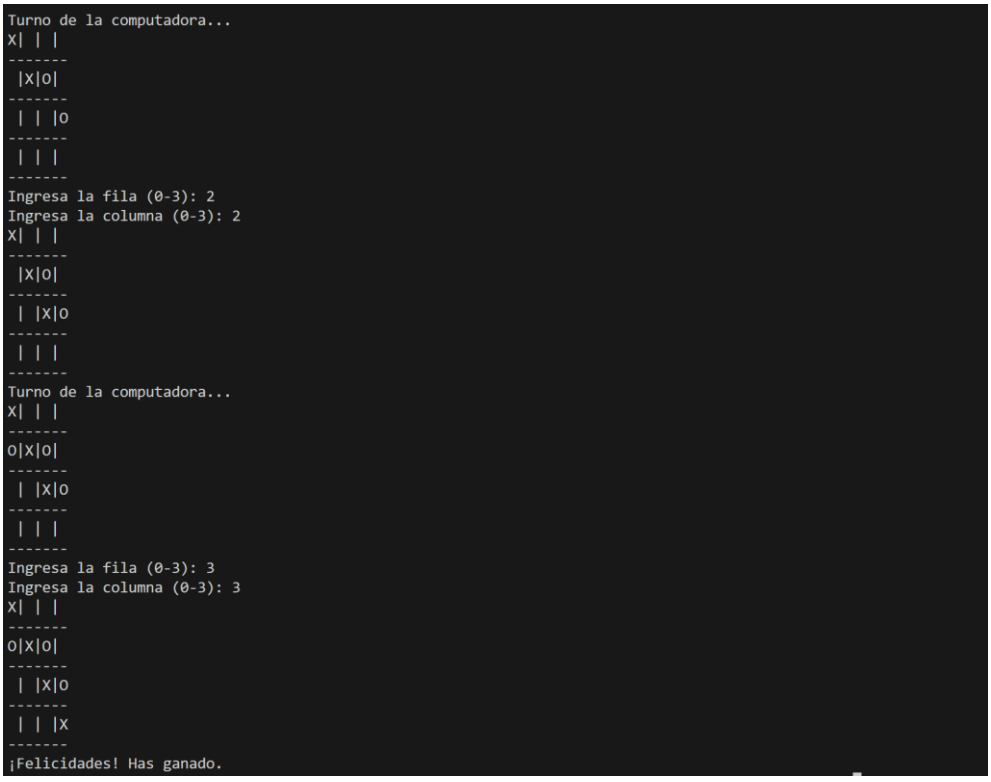
Ahora el juego inicia con la tirada del jugador el cual tiene como ficha una “x”, primero se manada a llamar la función de tirada de jugador, después se debe de imprimir el tablero y se mandara llamar la función de verificar ganador, en caso de que se regrese un true, se imprimirá que el jugador ha ganado, después se mandara llamar la función de que si el tablero esta lleno, en caso de que regrese un true entonces se imprimirá

que hay un empate. Después se hará la tirada de la computadora, primero se mandara llamar la función de tirada de jugador, después la función de verificar un ganador, si está regresa un true, entonces imprimirá que la maquina es la ganadora, después se manda llamar la función para verificar si el tablero esta lleno, en caso de regresar un true imprime que hay empate. Esto es un bucle el cual tienen 3 criterios de break, cuando haya un ganador o haya un empate.

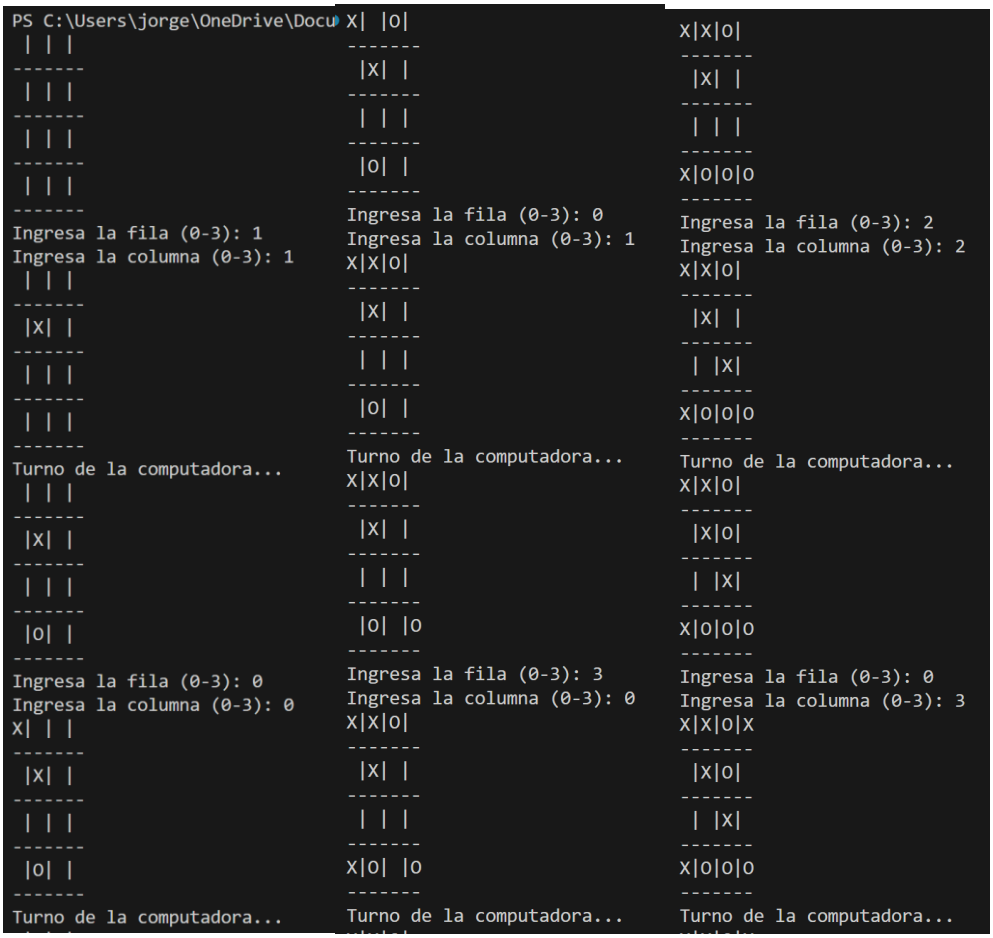
```
# FUNCIÓN DEL JUEGO
def juego_gato():
    # PRIMERO CREAMOS LA MATRIZ DE 4 X 4 Y LA GUARDAMOS EN LA VARIABLE TABLERO
    tablero = crear_tablero()
    # IMPRIMIMOS EL TABLERO
    imprimir_tablero(tablero)

    # INICIAMOS EL BUCLE DEL JUEGO
    while True:
        # PRIMERO HACEMOS LA TIRADA DEL JUGADOR
        # DESPUÉS IMPRIMIMOS EL TABLERO
        # COMPROBAMOS SI GANA EL USUARIO
        movimiento_jugador(tablero)
        imprimir_tablero(tablero)
        if verificar_ganador(tablero, 'X'):
            print("¡Felicidades! Has ganado.")
            break
        if tablero_lleno(tablero):
            print("¡Es un empate!")
            break

        # DESPUÉS HACEMOS LA TIRADA DE LA COMPUTADORA
        # IMPRIMIMOS EL TABLERO
        # COMPROBAMOS SI HA GANADO LA COMPUTADORA
        print("Turno de la computadora...")
        movimiento_computadora(tablero)
        imprimir_tablero(tablero)
        if verificar_ganador(tablero, 'O'):
            print("La computadora ha ganado.")
            break
        if tablero_lleno(tablero):
            print("¡Es un empate!")
            break
```

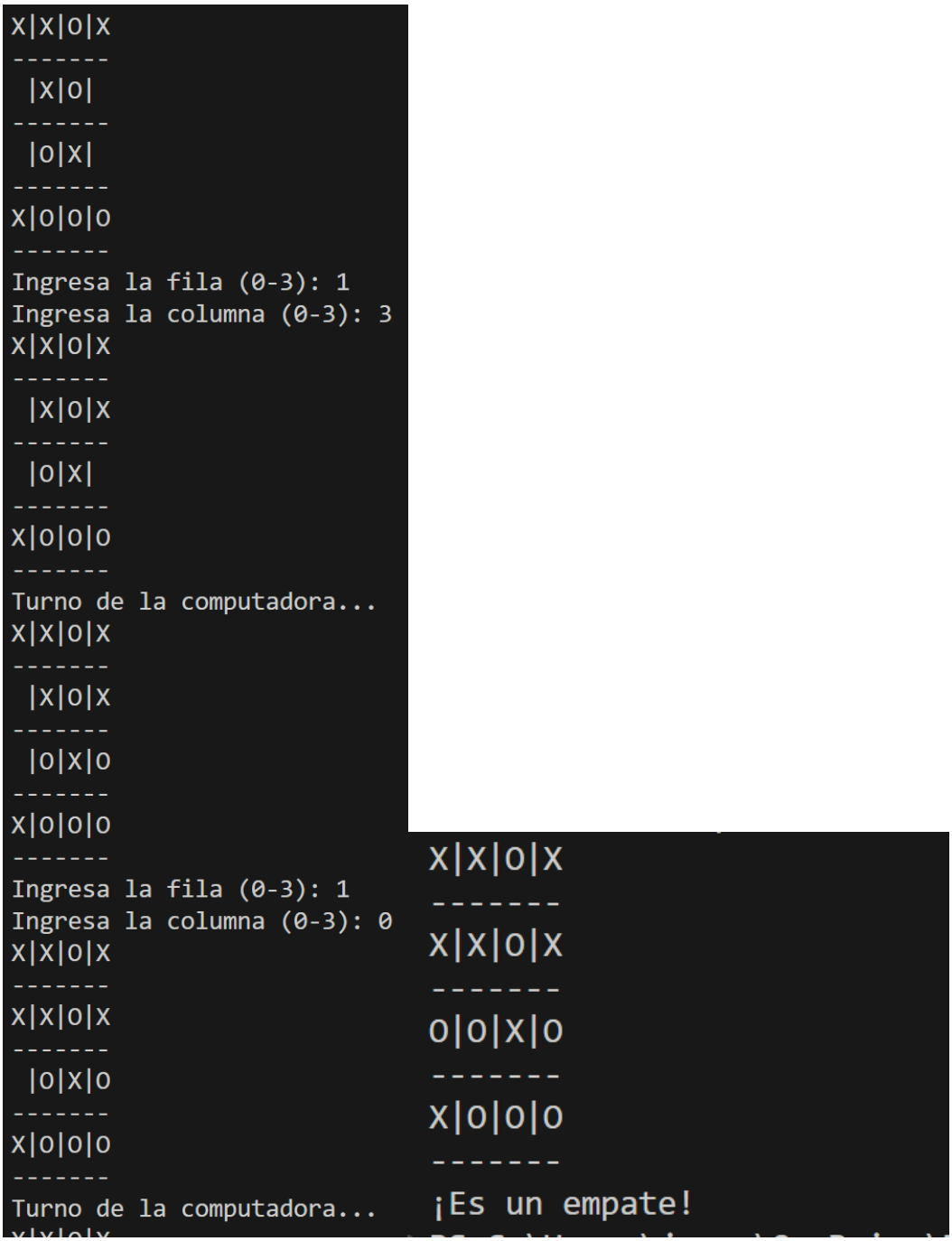
Ahora hacemos una ejecución de prueba de forma en la que ganamos





Ahora hacemos una prueba donde empatamos




X|X|O|X

|X|O|

|O|X|

X|O|O|O

Ingresa la fila (0-3): 1
Ingresa la columna (0-3): 3
X|X|O|X

|X|O|X

|O|X|

X|O|O|O

Turno de la computadora...
X|X|O|X

|X|O|X

|O|X|O

X|O|O|O

Ingresa la fila (0-3): 1
Ingresa la columna (0-3): 0
X|X|O|X

X|X|O|X

|O|X|O

X|O|O|O

Turno de la computadora...
X|X|O|X

X|X|O|X

O|O|X|O

X|O|O|O

¡Es un empate!

Enlace GitHub

A continuación, agregamos el link del GitHub donde se almacena esta practica de laboratorio, es un único link para cada una de las entregas en la cual estará dividida en carpetas correspondientes a la entrega. Para esta entrega es la carpeta LAB_1_BUSQUEDA_ALEATORIA.

https://github.com/Jorge300403/IA_6CV3_MartinezChavez