UNIVERSIDAD PERUANA
CAYETANO HEREDIA

Facultad de Ciencias y Filosofía Alberto Cazorla

Talleri

DEPARTAMENTO ACADÉMICO DE CIENCIAS
Ingeniería Biomédica

Curso: Programación Avanzada

PROYECTO FINAL:

Visualización Animada del Problema de las Ocho Reinas

Coordinador: Gervasio Coronel Molina

Alumna: José Alonso Zapata Castro

2023

Índice:

Introducción

El problema de las 8 reinas fue formulado en 1848 por el jugador de ajedrez bávaro Max Bezzel. Planteó la pregunta de cuántas soluciones podrían encontrarse para colocar 8 reinas en un tablero de ajedrez de manera que ninguna de las reinas capture a otra (no coincidan en la misma fila, columna o diagonal).

En este informe se expondrá el desarrollo de mi proyecto final del curso programación avanzada. El tema es el ya mencionado “Visualización animada del problema de las ocho reinas”. Para ello se hará uso de las técnicas aprendidas en clase como el uso de las GUI.

Enunciado del ejercicio:

Visualización Animada del Problema de las Ocho Reinas (Ajedrez):
Usted debe de implementar una aplicación desktop usando el módulo Tkinter. Está en la libertad de incluir los componentes que considere necesario. Trate que vuestro trabajo esté orientado amostrar a los estudiantes el funcionamiento de los algoritmos de iteración y recursión. La aplicación como mínimo debe de permitir:

- Iniciar el juego
- Opción manual, opción automática (simulación)
- Mostrar las soluciones
- Reiniciar el juego.

Puede ver ejemplos de visualización animada en las siguientes direcciones, sin embargo, no está obligado a recrear ninguno de los ejemplos:
http://eightqueen.bechersundstroem.de
https://www.cs.usfca.edu/~galles/visualization/RecQueens.html
http://www.brainmetrix.com/8queens
https://en.wikipedia.org/wiki/Eight_queens_puzzle

Códificación del Programa:

```
#Importación de las librerías:
import tkinter as tk
import time
import tkinter.messagebox as mb

#Programa base:
class ProblemaDeLas8Reinas:
    def __init__(self, maestro):
        #Creación de la ventana
        self.maestro = maestro
        self.maestro.title("Problema de las Ocho Reinas")

        #Longitud del tablero (casillas):
        self.Tablero = 8

        #Listas para guardar las posiciones y soluciones:
        self.Reinas = [-1] * self.Tablero
        self.Soluciones = []

        #Centinelas:
        self.AutoProgre = False
        self.MostrarSoluciones = False

        self.canvas = tk.Canvas(self.maestro, width=400, height=400)
        self.canvas.pack()
        self.canvas.bind("<Button-1>", self.ManualPosi)

        #Creación de los botones
        self.StartButton = tk.Button(self.maestro, text="Iniciar
Juego", command=self.IniciarJuego)
        self.StartButton.pack()

        self.AutomaticButton = tk.Button(self.maestro, text="Opción
Automática", command=self.AutomaticoPosi)
        self.AutomaticButton.pack()

        self.ShowSoluButton = tk.Button(self.maestro, text="Mostrar
Soluciones", command=self.MostrarTodSol)
        self.ShowSoluButton.pack()

        self.RestartButton = tk.Button(self.maestro, text="Reiniciar
Juego", command=self.ReiniciarJuego)
        self.RestartButton.pack()


    #Definición de las funciones:
    def CrearTablero(self):
        self.canvas.delete("all")
        Casilla = 400 // self.Tablero

        #Si la casilla es impar, será blanca. Si es par, será gris
        for i in range(self.Tablero):
            for j in range(self.Tablero):
                if (i + j) % 2 == 0:
                    color = "gray"
                else:
                    color = "white"
```

```
                self.canvas.create_rectangle(j * Casilla, i * Casilla,
(j + 1) * Casilla, (i + 1) * Casilla, fill = color)#Asignación del
color

    def MostrarReinas(self):
        self.canvas.delete("queens")
        Casilla = 400 // self.Tablero

        for rep, col in enumerate(self.Reinas):
            if col != -1:
                x = col * Casilla + Casilla // 2
                y = rep * Casilla + Casilla // 2
                self.canvas.create_text(x, y, text="♕", font=("Arial",
Casilla // 2), tags="queens") #sintaxis sacada de google, también pude
importar

#una imagen descargada.

    #Esta función es clave para el posicionamiento manual, verifica
que una reina no coincida con otra en la misma fila, columna o
diagonal.
    def VerificarPos(self, rep, col):
        for rep_previa in range(rep):
            if (self.Reinas[rep_previa] == col or
                self.Reinas[rep_previa] - rep_previa == col - rep or
                self.Reinas[rep_previa] + rep_previa == col + rep):
                return False
        return True

    #Función recursiva que utiliza el algoritmo de retroceso para
encontrar todas las posibles soluciones al problema.
    def SolucionReinas(self, rep):
        if rep == self.Tablero:
            self.Soluciones.append(self.Reinas.copy())
            return

        for col in range(self.Tablero):
            if self.AutoProgre and not self.MostrarSoluciones:
                if self.VerificarPos(rep, col):
                    self.Reinas[rep] = col
                    self.MostrarReinas()
                    self.maestro.update()
                    time.sleep(2)  #Delay para visualizar las
soluciones

                    self.SolucionReinas(rep + 1) #Esto inicia el
proceso de colocar la siguiente reina en la fila siguiente
            else:
                break #Fin de la iteración

    def Victoria(self):
        return all(col != -1 for col in self.Reinas)
        #Si col es diferente de -1 para todas las filas, significa que
se han colocado todas las reinas.

    def IniciarJuego(self):
        self.CrearTablero()
        self.MostrarReinas()

    def AutomaticoPosi(self):
        self.AutoProgre = True
```

```
        self.MostrarSoluciones = False
        self.Soluciones = []
        self.SolucionReinas(0)

    def MostrarTodSol(self):
        self.AutoProgre = False
        self.MostrarSoluciones = True
        for Solucion in self.Soluciones:
            self.Reinas = Solucion
            self.MostrarReinas()
            self.maestro.update()
            time.sleep(2)  #Delay para visualizar

    def ReiniciarJuego(self):
        #Cambio de lógica de los centinelas para reiniciar el
programa.
        self.AutoProgre = False
        self.MostrarSoluciones = False
        self.Reinas = [-1] * self.Tablero
        self.Soluciones = []
        self.CrearTablero()
        self.MostrarReinas()

    def ManualPosi(self, event):
        #Lo mismo que lo anterior, pero con lógica inversa para los
centinelas
        if not self.AutoProgre and not self.MostrarSoluciones:
            Casilla = 400 // self.Tablero
            col = event.x // Casilla
            rep = event.y // Casilla

            if 0 <= rep < self.Tablero and 0 <= col < self.Tablero and
self.VerificarPos(rep, col):
                self.Reinas[rep] = col
                self.MostrarReinas()

                if self.Victoria():
                    mb.showinfo("Felicitaciones, solución encontrada")
                    #self.ReiniciarJuego()

if __name__ == "__main__":
    root = tk.Tk()
    app = ProblemaDeLas8Reinas(root)
    root.mainloop()
```

Ánalisis y Explicación del código:

Parte 1: Inicialización y Configuración de la Interfaz Gráfica

La importación de las librerías son las necesidades claves para resolver el problema. *import time* me permite establecer un *delay* en el transcurso de las funciones y soluciones mientras que *import tkinter as tk* y *import tkinter.messagebox as mb* permiten la funcionalidad de la clase inicial y el final del problema con el mensaje: “Felicitaciones, solución encontrada”.

Parte 2: Dibujo del Tablero y las Reinas

Las funciones *CrearTablero(self)* y *MostrarReinas(self)* se encargan de dibujar el tablero de ajedrez en el lienzo y graficar las reinas en el tablero según la configuración actual.

Parte 3: Lógica del Problema de las 8 Reinas

Las función *VerificarPos(self, rep, col)* como su nombre indica verifica si es seguro colocar una reina en una posición dada (posición donde no sea capturada). Luego, *SolucionReinas(self, rep)* es una función recursiva para encontrar soluciones al problema de las 8 reinas. Sin embargo, durante el desarrollo de este encontré fallas al finalizarlo, por lo que ya no mostró las soluciones posibles al problema de las 8 reinas.
Por último, en esta parte, la función *Victoria(self)* se encarga de comprobar si se han colocado todas las reinas de manera correcta y muestra un mensaje de victoria.

Parte 4: Interacción Manual con el Usuario

La función *ManualPosi(self, event)* es llamada al presionar el botón de opción manual. Permitendo al usuario colocar reinas haciendo clic en el tablero. No obstante, como cambio al notar la lógica decidí eliminar este botón e implementar la función de manera automática al iniciar el juego, es decir, una vez se comience el juego el usuario puede dar *click* a las casillas de la primera fila para seleccionar la reina de ellas, luego irá bajando de una por una hasta intentar resolver el problema.

Parte 5: Interacción Automática y Visualización de Soluciones

AutomaticoPosi (self) es invocada al presionar el botón de opción automática y se encarga de iniciar la búsqueda automática de soluciones al problema de las 8 reinas. Después, MostrarTodSol (self) es llamada al presionar el botón de mostrar soluciones y permite la visualiza las soluciones encontradas de manera animada (al finalizar el código dejó de funcionar por un error no encontrado).

Parte 6: Reinicio del Juego

ReiniciarJuego(self) Reinicia el juego, deteniendo la opción automática o la visualización de soluciones. Además de regresar el tablero y las configuraciones a su estado inicial.

Parte 7: Ejecución del Programa

Finalmente, las últimas 4 líneas de texto codificado se encargan de inicializar el programa dando comienzo al bucle original con el llamado de la clase *ProblemaDeLas8Reinas*.

Conclusión:

En conclusión, la resolución del problema de las ocho reinas utilizando un programa de Python con Interfaces Gráficas del Usuario de la biblioteca Tkinter se alcanzó. Cumpliendo los objetivos descritos en el análisis realizado con respecto al inicio, reinicio y función tanto automática como manual del juego.

Sin embargo no se logró la muestra de las soluciones obtenidas tras resolver el juego por falta de técnicas o alguna fuente extra que no logré encontrar al solucionar este ejercicio y esto es considerado como una limitación que tuve durante la misma.