

**Universidad de Guadalajara**

Centro universitario de ciencias exactas e ingenierías

**CUCEI**

**Análisis de algoritmos**

Jorge Ernesto López Arce Delgado

D06

2024A

Actividad #6

Laberinto divide y vencerás

Chávez Velasco Cristian

**218532484**

Hernández Martínez Luis Yael

**215408324**

11/03/24

**Introducción**

Se nos presenta un laberinto representado por una matriz cuadrada (en nuestro caso de una dimensión mínima de 6x6), donde algunas celdas son caminos (representadas por 0) y otras son paredes (representadas por 1). El objetivo es encontrar un camino desde la entrada (siempre en la posición 0, 0) hasta la salida (con el valor 2) utilizando el algoritmo "divide y vencerás".

Para entender esto primero hay que hacer una breve introducción a este algoritmo, hay que entender la pregunta ¿Qué es el algoritmo divide y vencerás?

Divide y vencerás hace refencia al refrán que para resolver un problema difícil se tiene que dividir en partes mas pequeñas, tantas veces sean necesario hasta que la respuesta sea obvia, el método esta basado en la resolución recursiva de un problema como se mencionaba dividiendo el problema en dos o mas subproblemas, se continua este proceso hasta llegar a ser lo suficientemente sencillo para que se resuelva, al final las soluciones de cada uno de los subproblemas se combinan para dar con la solución final al problema original, es un buen algoritmo para casi cualquier tipo de problemas como algoritmos de ordenamiento(Quicksort), multiplicar números grandes(karatsuba), la transformada discreta de Fourier entre otros ejemplos.

**Objetivos**

El objetivo principal de esta actividad es crear un laberinto con una matriz cuadrada, en la que algunas celdas son caminos y otras paredes, se tiene que implementar el algoritmo “divide y vencerás” para encontrar el camino desde la casilla de entrada, también se puede recorrer el laberinto con celdas especiales tales como teletransportes, en esta actividad al ser elaborada en equipo se dividió en tres roles los cuales tenían un objetivo propio que creo cada quien, el frontend tenía tres objetivos principales, los cuales eran elaborar la matriz, ajustar su tamaño y diseñarla correctamente en las ventanas usando tkinter, el backend tenía cuatro objetivos, tales como definir las funciones, validad las celdas y movimiento de posición del jugador, implementar el algoritmo divide y vencerás y generar la matriz con el camino resuelto, el ultimo rol que se implemento fue el de administrador del proyecto, el cual estaba atento que se subieran los avances, saber en todo momento en que fase del proyecto estaba la actividad y la elaboración del reporte y exposición.

**Desarrollo:**

**Código:**

**MAIN:**

from tkinter import Tk, Canvas, mainloop

from tkinter import messagebox, simpledialog

import mapa as m

def mostrarMapaTkinter(mapaJuego, pasos, laberinto\_resuelto=False):

    # Crear una ventana principal

    root = Tk()

    canvas = Canvas(root, width=600, height=600)

    canvas.pack()

    root.geometry("500x400")  # Establecer el tamaño de la ventana principal

    # Canvas para el mapa original

    canvas\_original = Canvas(root, width=600, height=600)

    canvas\_original.pack(side="left")

    # Canvas para el mapa con el recorrido

    canvas\_recorrido = Canvas(root, width=600, height=600)

    canvas\_recorrido.pack(side="right")

    # Creamos un conjunto para las coordenadas especiales

    coordenadas\_especiales = set()

    coordenadas\_especiales.add(tuple(mapaJuego.\_posicionInicial))

    coordenadas\_especiales.add(tuple(mapaJuego.\_salida))

    for portal in mapaJuego.\_cordenadasPortal:

        coordenadas\_especiales.add(tuple(portal))

    # Iterar sobre cada celda del mapa

    for i in range(mapaJuego.\_tamanio):

        for j in range(mapaJuego.\_tamanio):

            x1, y1 = j \* 30, i \* 30

            x2, y2 = x1 + 30, y1 + 30

            # Colorear ubicaciones especiales primero

            if (i, j) in coordenadas\_especiales:

                if (i, j) == tuple(mapaJuego.\_posicionInicial):

                    canvas.create\_rectangle(x1, y1, x2, y2, fill="yellow")

                elif (i, j) == tuple(mapaJuego.\_salida):

                    canvas.create\_rectangle(x1, y1, x2, y2, fill="green")

                elif (i, j) == tuple(mapaJuego.\_cordenadasPortal[0]):

                    canvas.create\_rectangle(x1, y1, x2, y2, fill="blue")

                elif (i, j) == tuple(mapaJuego.\_cordenadasPortal[1]):

                    canvas.create\_rectangle(x1, y1, x2, y2, fill="red")

                elif (i, j) == (1, 1):  # Nueva casilla "trivia"

                    canvas.create\_rectangle(x1, y1, x2, y2, fill="orange")

            # Colorear el recorrido

            elif [i, j] in pasos:

                if (i, j) not in coordenadas\_especiales:  # Evitar colorear ubicaciones especiales

                    canvas.create\_rectangle(x1, y1, x2, y2, fill="pink")

            # Colorear las celdas según su tipo

            elif mapaJuego.\_mapa[i][j] == 0:

                canvas.create\_rectangle(x1, y1, x2, y2, fill="white")

            elif mapaJuego.\_mapa[i][j] == 1:

                canvas.create\_rectangle(x1, y1, x2, y2, fill="black")

            elif mapaJuego.\_mapa[i][j] == 2:

                canvas.create\_rectangle(x1, y1, x2, y2, fill="green")

            elif mapaJuego.\_mapa[i][j] == 111:  # Casilla "trivia"

                canvas.create\_rectangle(x1, y1, x2, y2, fill="orange")

    # Mostrar mensaje si el laberinto ha sido resuelto

    if laberinto\_resuelto:

        texto = "¡Laberinto resuelto! Se encontró una solución para el laberinto."

        canvas.create\_text(250, 380, text=texto, font=("Arial", 12), fill="blue")

    root.mainloop()

if \_\_name\_\_ == "\_\_main\_\_":

    # Crear la matriz de mapa con un tamaño de 10x10

    mapa = m.MapaJuego(10)

    mapa.dibujarMapa()

    mostrarMapaTkinter(mapa, [])  # Pasamos una lista vacía para que al principio no haya pasos

    resuelto = False

    pasos = []

    contador = 0

    # Intenta resolver el laberinto hasta 5 intentos

    while (not resuelto) and (contador <= 5):

        contador += 1

        try:

            pasos = mapa.resolverMapa(mapa.\_posicionInicial)

            resuelto = True

        except Exception as e:

            print(e)

    print(pasos)

    # Mostrar el laberinto con la solución si se encontró una

    if resuelto:

        mostrarMapaTkinter(mapa, pasos, laberinto\_resuelto=True)

    else:

        messagebox.showerror("Laberinto no resuelto", "No se pudo encontrar un camino en el laberinto.")

**UTILIDADES:**

from math import sqrt

def distancia(p1,p2):

    return (sqrt(((p2[0]-p1[0])\*\*2)+((p2[1]-p1[1])\*\*2)))

**MAPA:**

import numpy as np

from math import copysign as cs, ceil as ce

from random import randint as ri, shuffle as sf

from time import sleep as sl

from math import sqrt

from tkinter import messagebox, simpledialog

import utilidades as ut

class MapaJuego:

    def \_\_init\_\_(self, tamanio=6, posicionInicial=[0, 0]):

        self.\_tamanio = tamanio

        self.\_mapa = np.zeros((tamanio, tamanio), dtype=int)

        self.\_posicionInicial = posicionInicial

        self.\_posicionJugador = posicionInicial

        self.\_salida = [ri(0, tamanio - 1), tamanio - 1]

        self.\_cordenadasPortal = []

        self.preguntas\_respuestas = {

            "¿Cuál es el río más largo del mundo?": "Amazonas",

            #etc etc

        }

    def celdaValida(self,celda):

        if (celda[0] < -1) or (celda[1] < -1):

            return False

        if (celda[0] > self.\_tamanio) or (celda[1] > self.\_tamanio):

            return False

        if (celda == self.\_posicionInicial)or(celda == self.\_salida):

            return False

        return True

    def celdaVisitada(self, celda):

        if (self.\_mapa[celda[0],celda[1]]==1):

            return True

        else:

            return False

    def mostrarTrivia(self):

        pregunta = ri(0, len(self.preguntas\_respuestas) - 1)

        pregunta\_texto = list(self.preguntas\_respuestas.keys())[pregunta]

        respuesta\_correcta = self.preguntas\_respuestas[pregunta\_texto]

        respuesta\_jugador = simpledialog.askstring("Trivia", pregunta\_texto)

        if respuesta\_jugador is not None and respuesta\_jugador.lower() == respuesta\_correcta.lower():

            messagebox.showinfo("Respuesta correcta", "¡Respuesta correcta! Puedes pasar por la puerta.")

            return True

        else:

            messagebox.showerror("Respuesta incorrecta", "Respuesta incorrecta. Debes responder correctamente para pasar.")

            return False

    def crearCamino(self, densidadParedes):

        densidadParedes = densidadParedes \*(self.\_tamanio \*\* 2)//4

        direcciones = [(2, 0), (-2, 0), (0, 2), (0, -2)]

        for i in range(0, densidadParedes):

            x = (ri(2,self.\_tamanio-3)//2)\*2

            y = (ri(2,self.\_tamanio-3)//2)\*2

            if(self.celdaValida([x,y]) and (not self.celdaVisitada([x,y]))):

                self.\_mapa[x][y]= 1

                for dx,dy in direcciones:

                    nx = x + dx

                    ny = y + dy

                    if(self.celdaValida([nx,ny])):

                        if self.\_mapa[nx][ny] == 0:

                            self.\_mapa[nx][ny] =1

                            if(x== nx):

                                ny += int((cs(1,(ny - y))\*-1))

                            if(y == ny):

                                nx += int((cs(1,(nx - x))\*-1))

                            self.\_mapa[nx][ny] =1

        return

    def crearPuertas(self):

        for i in range(0, ce(self.\_tamanio/10)):

            puertaCreada = False

            while not puertaCreada:

                x = ri(1,self.\_tamanio-2)

                y = ri(1,self.\_tamanio-2)

                if (self.\_mapa[x][y] == 1):

                    if (((self.\_mapa[x-1][y] == 0) and  (self.\_mapa[x+1][y] == 0)) or

                       ((self.\_mapa[x][y-1] == 0) and  (self.\_mapa[x][y+1] == 0))):

                        self.\_mapa[x][y] = 111

                        puertaCreada = True

    def crearPortal(self):

        entradaCreada = False

        salidaCreada = False

        while not entradaCreada:

            x = ri(0,self.\_tamanio-1)

            y = ri(0,self.\_tamanio-1)

            if (self.\_mapa[x][y] == 0):

                self.\_mapa[x][y] = 3

                entradaCreada = True

                self.\_cordenadasPortal.append([x,y])

        while not salidaCreada:

            x = ri(0,self.\_tamanio-1)

            y = ri(0,self.\_tamanio-1)

            if (self.\_mapa[x][y] == 0):

                self.\_mapa[x][y] = 4

                salidaCreada = True

                self.\_cordenadasPortal.append([x,y])

    def dibujarMapa(self):

        self.\_mapa[self.\_posicionInicial[0],self.\_posicionInicial[1]]= 8

        self.\_mapa[0][2]=1

        self.crearCamino(500)

        self.crearPuertas()

        self.crearPortal()

        self.\_mapa[self.\_salida[0],self.\_salida[1]]= 2

        return self

    def resolverMapa(self, posicion, pasos=[[], []]):

        menorDistancia = self.\_tamanio \*\* 2

        print(pasos)

        if posicion == self.\_salida:

            return pasos[0]

        # Verificar si el jugador está en una casilla de trivia

        if self.\_mapa[posicion[0]][posicion[1]] == 111:

            if self.mostrarTrivia():  # Mientras la respuesta sea incorrecta, seguir mostrando la trivia

                pasos[0].append(posicion)

            else:

                return pasos[0]

        movimientos = [(1, 0), (0, 1), (-1, 0), (0, -1)]

        sf(movimientos)

        for dx, dy in movimientos:

            nx, ny = posicion[0] + dx, posicion[1] + dy

            if (not ((nx < 0) or (ny < 0) or (nx >= self.\_tamanio) or (ny >= self.\_tamanio))) and (

                    not ([nx, ny] in pasos[0]) and not ([nx, ny] in pasos[1])):

                if ((self.\_mapa[nx][ny] == 0) or (self.\_mapa[nx][ny] == 111) or ((self.\_mapa[nx][ny] == 2))):

                    if (ut.distancia([nx, ny], self.\_salida) <= menorDistancia):

                        siguientePaso = [nx, ny]

                        menorDistancia = ut.distancia([nx, ny], self.\_salida)

                elif (self.\_mapa[nx][ny] == 3):

                    if (ut.distancia([nx, ny], self.\_salida) <= menorDistancia):

                        pasos[0].append([nx, ny])

                        nx, ny = self.\_cordenadasPortal[1][0], self.\_cordenadasPortal[1][1]

                        siguientePaso = [nx, ny]

                        menorDistancia = ut.distancia([nx, ny], self.\_salida)

                elif (self.\_mapa[nx][ny] == 4):

                    if (ut.distancia([nx, ny], self.\_salida) <= menorDistancia):

                        pasos[0].append([nx, ny])

                        nx, ny = nx, ny = self.\_cordenadasPortal[0][0], self.\_cordenadasPortal[0][1]

                        siguientePaso = [nx, ny]

                        menorDistancia = ut.distancia([nx, ny], self.\_salida)

        try:

            pasos[0].append(siguientePaso)

            self.resolverMapa(siguientePaso, pasos)

        except Exception as er:

            pasos[1].append(pasos[0].pop())

            self.resolverMapa(pasos[0][-2], pasos)

        return pasos[0]

**Capturas del funcionamiento del código:**

**Imagen en blanco y negro

Descripción generada automáticamente con confianza baja**

**Una captura de pantalla de un celular

Descripción generada automáticamente con confianza baja**

**Un conjunto de letras blancas en un fondo blanco

Descripción generada automáticamente con confianza baja**

**Imagen que contiene juego, texto

Descripción generada automáticamente**

**Imagen que contiene juego, texto

Descripción generada automáticamente**

****

**Imagen que contiene juego, texto

Descripción generada automáticamente**

**Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico, Sitio web

Descripción generada automáticamente**

**Imagen que contiene juego, dibujo

Descripción generada automáticamente**

****

**Observaciones:**

**Conclusiones Cristian:**

En lo personal al tener el puesto del administrador pude saber de primera mano que no es tan fácil como parece, pues es quien básicamente da la cara por el equipo en todo momento, tiene que lidiar con los problemas que se presenten, este supervisando a los demás y al final creo que es algo estresante en algunas ocasiones, quizás si nos hace falta tal como lo menciono el profe el desarrollo de estas cualidades blandas para en el futuro poder ser mucho mejores, tanto profesionalmente como personalmente. Con esta actividad demuestra que muchas de las veces es posible encontrar una solución dividiendo el problema hasta su parte mas fundamental, que es bastante buena está alternativa pero como todo tiene limitaciones

**Conclusiones Luis:** El análisis del problema me llevo a tener que entender o buscar primero una manera de crear un laberinto que fuera posible resolver de manera aleatoria en cada vez que se ejecutara el programa que en si mismo llevo un reto y el segundo reto ya fue entender el problema de la resolución del laberinto basándome en que tengo un punto de salida y siempre una posible solución ir dividiendo el problema hasta su parte mas fundamental que fue el moverse a una casilla y de esta manera junto con saber que casillas ya había visitado y ver que movimiento seria el mas optimo para llegar a la salida fue como hice posible la resolución del problema

**Referencias**

Giraldo Acosta, M. (2016). Ambiente interactivo para la visualización de conceptos asociados a las técnicas de diseño de algoritmos.

Del Estado de Hidalgo, U. A. (s. f.). *Propuesta para la generación de laberintos ampliados en 2d*. https://www.uaeh.edu.mx/scige/boletin/huejutla/n1/a4.html

*Laberintos. Generador interactivo de laberintos*. (2014, 13 diciembre). Recursos Educativos Digitales. https://www3.gobiernodecanarias.org/medusa/ecoescuela/recursosdigitales/2014/12/13/laberintos-generador-interactivo-de-laberintos/