INSTITUTO TECNOLÓGICO DE LAS AMÉRICAS

Las Americas Institute of Technology

Inteligencia Artificial

Portafolio - 2

2024 C-2

Docente:

Risaldy José Rodriguez Jimenez

Estudiante:

Luis Montoya **2023-0919**

Viernes 12/07/2024

Introducción a la Inteligencia Artificial
Programación en Python Prof.
Carlos B. Ogando M.

Programación estructurada

Repositorio del código:

https://github.com/0TSH0/Segundo-Portafolio

Dados los siguientes problemas de programación estructurada seleccione uno y envíe el código con la solución. (archivo .py o enlace de prueba en línea) NO TEXTO. 1.- Diseñe una solución que convierte cualquier número romano a decimal. Ejemplo:

```
Entrada: XXXIV Salida: 34
```

```python
# 1.- Diseñe una solución que convierte cualquier número romano a decimal. Ejemplo:
# Entrada: XXXIV
# Salida: 34

def conversor(romano):
    roman_numerals = {'I': 1, 'V': 5, 'X': 10, 'L': 50, 'C': 100, 'D': 500, 'M': 1000}
    total = 0
    prev_value = 0

    for char in reversed(romano):
        value = roman_numerals[char]
        if value < prev_value:
            total -= value
        else:
            total += value
        prev_value = value

    return total


entrada = input("Introduce un número romano: ")
salida = conversor(entrada)
print(f"Entrada: {entrada}\nSalida: {salida}")
```

```
PS D:\BACK UP 17 DE MARZO\ITLA\4to cuatrimestre\IA\ejercicios\python> & C:/Us
 17 DE MARZO/ITLA/4to cuatrimestre/IA/ejercicios/python/main.py"
Introduce un número romano: XXXIV
Entrada: XXXIV
Salida: 34
PS D:\BACK UP 17 DE MARZO\ITLA\4to cuatrimestre\IA\ejercicios\python>
```

2.- Diseñe una solución que determine si una cadena de paréntesis, llaves y corchetes es válida. Ejemplo:

Entrada: (){[]()}

Salida: Válido

```python
def verif(s):
    corchetes = []
    cadena = {')': '(', '}': '{', ']': '['}

    for char in s:
        if char in cadena.values():
            corchetes.append(char)
        elif char in cadena.keys():
            if corchetes == [] or cadena[char] != corchetes.pop():
                return False
        else:
            return False

    return corchetes == []


entrada = input("Introduce una cadena de paréntesis, llaves y corchetes: ")
salida = verif(entrada)
print(f"Entrada: {entrada}\nSalida: {'Válido' if salida else 'Inválido'}")
```

```
//////////////////////////////////////2//////////////////////////////////////////
Introduce una cadena de paréntesis, llaves y corchetes: ({[]})
Entrada: ({[]})
Salida: Válido
PS D:\BACK UP 17 DE MARZO\ITLA\4to cuatrimestre\IA\ejercicios\python>
```

3.- Diseñe una solución que a partir de una lista de números retorne la cantidad de números primos que contiene la misma.

Entrada: [2, 8, 10, 15, 13, 29, 50, 149]
Salida: 4

```python
def n_primo(num):
    if num <= 1:
        return False
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            return False
    return True

def contador(nums):
    return sum(1 for num in nums if n_primo(num))

# Entrada desde la consola:
entrada = input("Introduce una lista de números separados por comas: ")
numeros = list(map(int, entrada.split(',')))
salida = contador(numeros)
print(f"Entrada: {numeros}\nSalida: {salida}")
```

```
PS D:\BACK UP 17 DE MARZO\ITLA\4to cuatrimestre\IA\ejercicios\python> & C:/Users/lama9/Ap
 17 DE MARZO/ITLA/4to cuatrimestre/IA/ejercicios/python/main.py"
Introduce una lista de números separados por comas: 2, 15, 31, 37,115
Entrada: [2, 15, 31, 37, 115]
Salida: 3
////////////////////////////////////4////////////////////////////////////////
```

4.- Diseñe una solución que a partir de una lista de calificaciones retorne una lista con su valor equivalente en letras.

```
Entrada:     [92, 68, 77, 75, 82] Salida:
[A, D, C, C, B]
```

```python
def nota_conversor(calificacion):
    if calificacion >= 90:
        return 'A'
    elif calificacion >= 80:
        return 'B'
    elif calificacion >= 70:
        return 'C'
    elif calificacion >= 60:
        return 'D'
    else:
        return 'F'

def calificacion(total):
    return [nota_conversor(grade) for grade in total]


entrada = input("Introduce una lista de calificaciones separadas por comas: ")
calificaciones = list(map(int, entrada.split(',')))
salida = calificacion(calificaciones)
print(f"Entrada: {calificaciones}\nSalida: {salida}")
```

```
PS D:\BACK UP 17 DE MARZO\ITLA\4to cuatrimestre\IA\ejercicios\python> & C:/U
 17 DE MARZO/ITLA/4to cuatrimestre/IA/ejercicios/python/main.py"
Introduce una lista de calificaciones separadas por comas: 98,32,88,70,79
Entrada: [98, 32, 88, 70, 79]
Salida: ['A', 'F', 'B', 'C', 'C']
```

5.- Diseñe una solución que a partir de una lista de palabras retorne otra lista conteniendo únicamente las que son palíndromos.

Entrada: [“agua”, “rotor”, “perla”, “reconocer”, “ojo”, “peso”] Salida: [“rotor”, “reconocer”, “ojo”]

```python
def palidromo(palabra):
    return palabra == palabra[::-1]

def verificar(palabras):
    return [palabras for palabras in palabras if palidromo(palabras)]

# al momento de ingresar una palabra que sea un palídromo, trata de colocar todo el texto en mayusculas o minusculas
entrada = input("Introduce una lista de palabras separadas por comas: ")
palabras = [palabra.strip() for palabra in entrada.split(',')]
salida = verificar(palabras)
print(f"Entrada: {palabras}\nSalida: {salida}")
```

```
Introduce una lista de palabras separadas por comas: hannah,ana,cielo, perla, ojo
Entrada: ['hannah', 'ana', 'cielo', 'perla', 'ojo']
Salida: ['hannah', 'ana', 'ojo']
PS D:\BACK UP 17 DE MARZO\ITLA\4to cuatrimestre\IA\ejercicios\python>
```

Programación ORIENTADA A OBJETOS

Dados los siguientes problemas de programación orientada a objetos seleccione uno e implemente una clase que cumpla lo que se pide (archivo .py o enlace de prueba en línea) NO TEXTO.

1.- Diseñe una clase que represente un triángulo rectángulo y permita determinar el área, el perímetro, la hipotenusa y los ángulos de este a partir de sus dos catetos. Ejemplo:

Entrada: 3, 4

Salida: Área:6
Perímetro: 12
Hipotenusa 5

Ángulo 1: 53.13°
Ángulo 2: 36.86°

```python
class TrianguloRectangulo:
    def __init__(self, cateto_a, cateto_b):
        self.cateto_b = cateto_b
        self.hipotenusa = self.calcular_hipotenusa()
        self.angulo_a = self.calcular_angulo_a()
        self.angulo_b = self.calcular_angulo_b()

    def calcular_hipotenusa(self):
        return math.sqrt(self.cateto_a**2 + self.cateto_b**2)

    def calcular_area(self):
        return (self.cateto_a * self.cateto_b) / 2

    def calcular_perimetro(self):
        return self.cateto_a + self.cateto_b + self.hipotenusa

    def calcular_angulo_a(self):
        return math.degrees(math.atan(self.cateto_a / self.cateto_b))

    def calcular_angulo_b(self):
        return math.degrees(math.atan(self.cateto_b / self.cateto_a))

    def ver_res(self):
        print(f"Cateto A: {self.cateto_a}")
        print(f"Cateto B: {self.cateto_b}")
        print(f"Área: {self.calcular_area()}")
        print(f"Perímetro: {self.calcular_perimetro()}")
        print(f"Hipotenusa: {self.hipotenusa}")
        print(f"Ángulo A (opuesto a cateto A): {self.angulo_a:.2f} grados")
        print(f"Ángulo B (opuesto a cateto B): {self.angulo_b:.2f} grados")

# Entrada desde la consola:
cateto_a = float(input("Introduce el valor del primer cateto: "))
cateto_b = float(input("Introduce el valor del segundo cateto: "))

triangulo = TrianguloRectangulo(cateto_a, cateto_b)
triangulo.ver_res()
```

```
Introduce el valor del primer cateto: 3
Introduce el valor del segundo cateto: 4
Cateto A: 3.0
Cateto B: 4.0
Área: 6.0
Perímetro: 12.0
Hipotenusa: 5.0
Ángulo A (opuesto a cateto A): 36.87 grados
Ángulo B (opuesto a cateto B): 53.13 grados
PS D:\BACK UP 17 DE MARZO\ITLA\4to cuatrimestre\IA\ejercicios\python>
```

2.- Diseñe una clase que represente una esfera y permita determinar el área de la superficie, el volumen y el diámetro a partir del radio. Ejemplo:

```
Entrada:     5

Salida:      Área de la superficie: 314.16
             Volumen: 523.6 Diámetro: 10
```

```python
import math

class Esfera:
    def __init__(self, radio):
        self.radio = radio

    def calcular_area_superficie(self):
        return 4 * math.pi * self.radio**2

    def calcular_volumen(self):
        return (4/3) * math.pi * self.radio**3

    def calcular_diametro(self):
        return 2 * self.radio

    def res(self):
        area_superficie = self.calcular_area_superficie()
        volumen = self.calcular_volumen()
        diametro = self.calcular_diametro()

        print(f"Radio: {self.radio}")
        print(f"Área de la superficie: {area_superficie:.2f}")
        print(f"Volumen: {volumen:.2f}")
        print(f"Diámetro: {diametro}")

radio = float(input("Introduce el valor del radio de la esfera: "))

esfera = Esfera(radio)
esfera.res()
```

```
Introduce el valor del radio de la esfera: 5
Radio: 5.0
Área de la superficie: 314.16
Volumen: 523.60
Diámetro: 10.0
```

4.- Diseñe una clase que represente a una pizza con su nombre, listado de ingredientes, tamaño, precio de venta, costo de producción y tiempo de producción, y que permita determinar la ganancia neta y comprobar si contiene un ingrediente específico.

Entrada:

```
Nombre: Vegetariana
Ingredientes: [“Ají”, “Cebolla”, “Tomate”, “Champiñón”]
Tamaño: Mediano
Precio de venta: RD$725
Costo de producción: RD$450
Tiempo de cocción: 7 minutos
```

Salida: Ganancia neta: RD$275
¿Contiene “Ají”?: Verdadero

```python
class Pizza:
    def __init__(self, nombre, ingredientes, tamano, precio_venta, costo_produccion, tiempo_coc
        self.nombre = nombre
        self.ingredientes = ingredientes
        self.tamano = tamano
        self.precio_venta = precio_venta
        self.costo_produccion = costo_produccion
        self.tiempo_coccion = tiempo_coccion

    def ganancia_neta(self):
        return self.precio_venta - self.costo_produccion

    def ingredientes(self, ingrediente):
        return ingrediente in self.ingredientes

    def res(self):
        ganancia_neta = self.ganancia_neta()
        print(f"Nombre: {self.nombre}")
        print(f"Ingredientes: {', '.join(self.ingredientes)}")
        print(f"Tamaño: {self.tamano}")
        print(f"Precio de venta: RD${self.precio_venta}")
        print(f"Costo de producción: RD${self.costo_produccion}")
        print(f"Tiempo de cocción: {self.tiempo_coccion} minutos")
        print(f"Ganancia neta: RD${ganancia_neta:.2f}")

nombre = input("Introduce el nombre de la pizza: ")
ingredientes = input("Introduce los ingredientes de la pizza, separados por comas: ").split(','
ingredientes = [ingrediente.strip() for ingrediente in ingredientes]
tamano = input("Introduce el tamaño de la pizza: ")
precio_venta = float(input("Introduce el precio de venta de la pizza (RD$): "))
costo_produccion = float(input("Introduce el costo de producción de la pizza (RD$): "))
tiempo_coccion = int(input("Introduce el tiempo de cocción de la pizza (minutos): "))
```

```
Introduce el nombre de la pizza: classic
Introduce los ingredientes de la pizza, separados por comas: queso, oregano, peppe
Introduce el tamaño de la pizza: 8
Introduce el precio de venta de la pizza (RD$): 400
Introduce el costo de producción de la pizza (RD$): 150
Introduce el tiempo de cocción de la pizza (minutos): 7
Nombre: classic
Ingredientes: queso, oregano, pepperoni
Tamaño: 8
Precio de venta: RD$400.0
Costo de producción: RD$150.0
Tiempo de cocción: 7 minutos
Ganancia neta: RD$250.00
```

Algoritmos de búsqueda

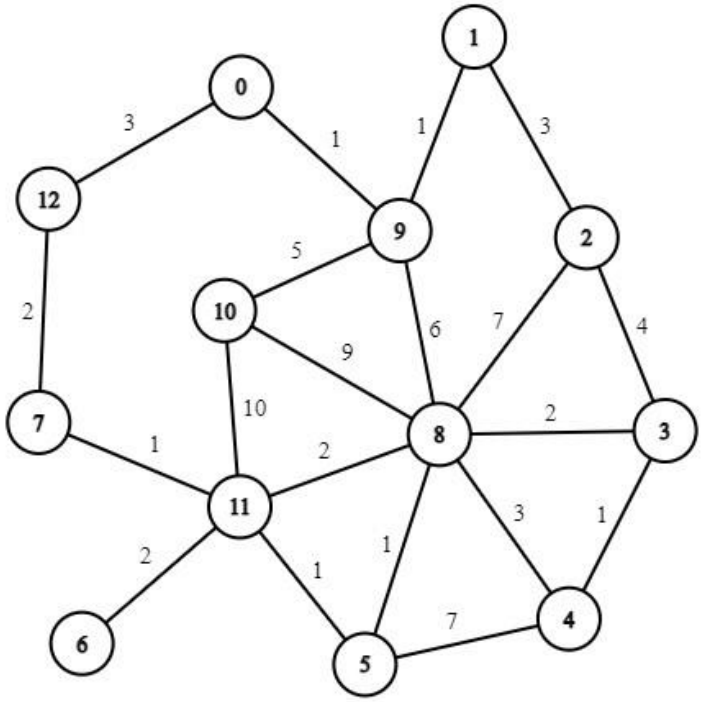
Seleccione uno de los siguientes ejercicios de búsqueda de ruta para llegar de un nodo a otro en un grafo ponderado e indique lo que se le pide más adelante empleando estos tres algoritmos de búsqueda: DFS, BFS y UCS.

En su respuesta debe mostrar por cada algoritmo:

- Nodos explorados (en orden de exploración)
- Nodos en frontera (en orden de exploración) [Valores del queue/stack] • Ruta final obtenida con el algoritmo. (DFS, BFS y UCS)
- Árbol de búsqueda generado con el algoritmo.

Ejercicio 1. Grafo 1.

Partida: Nodo 1
Destino: Nodo 6



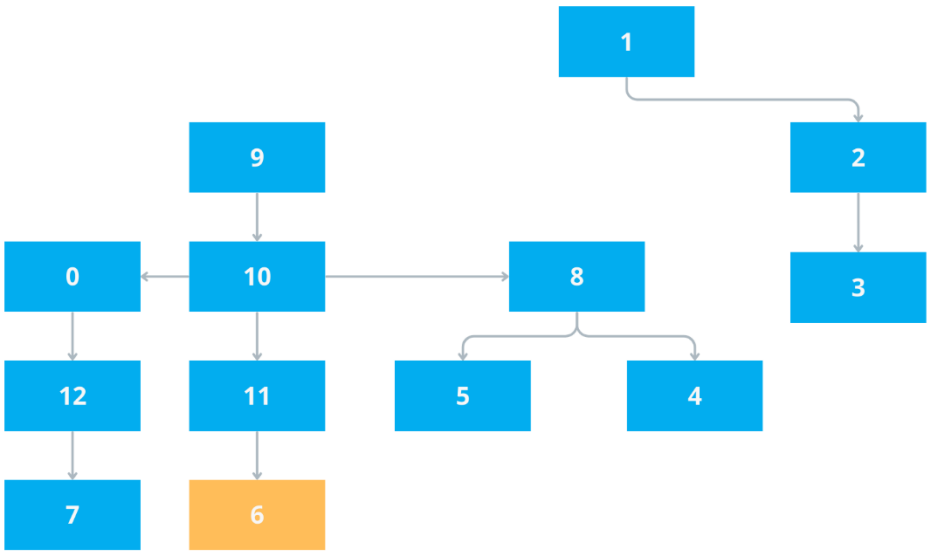
Nodos Explorados (en orden de Exploracion)

BFS:

Nodos explorados: 1,9,2,0,10,8,3,12,11,5,4,7,6

Nodos en frontera: 1,9,2,0,10,8,3,12,11,5,4,7,6 : : : Ɛ

Ruta final:



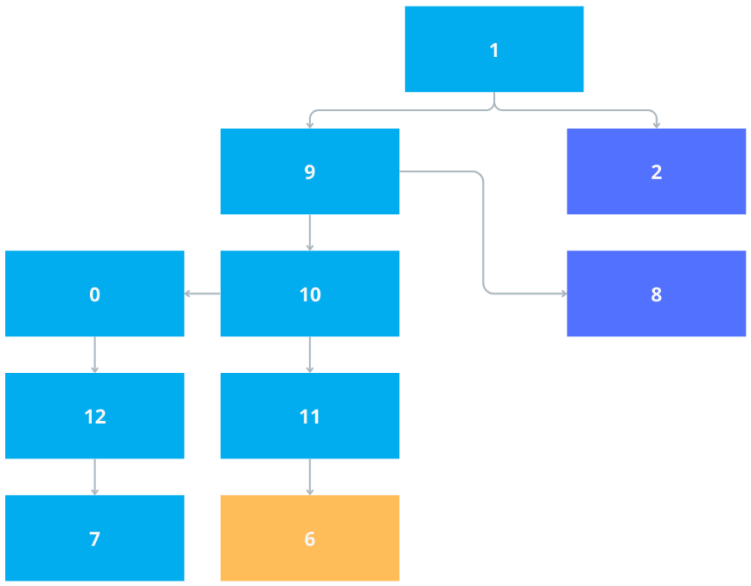
1,9,10,11,6

DFS:

NODOS EXPLORADOS: 1,9,0,10,12,11,7,6

NODOS EN FRONTERA: 1,9,2,0,10,8,12,11,7,6 : : : Ɛ

RUTA FINAL:



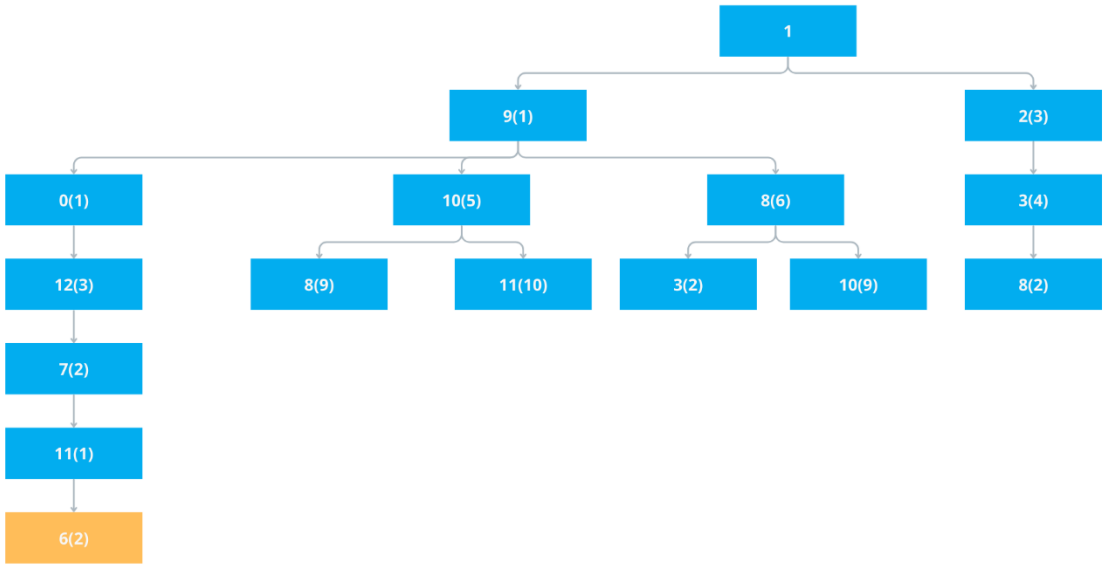
1,9,10,11,6

UCS:

NODOS EXPLORADOS: 1,9,0,10,12,11,7,6

NODOS EN FRONTERA: 1,9,2,0,10,8,3,12,11,5,4,7,6 ::: Ɛ

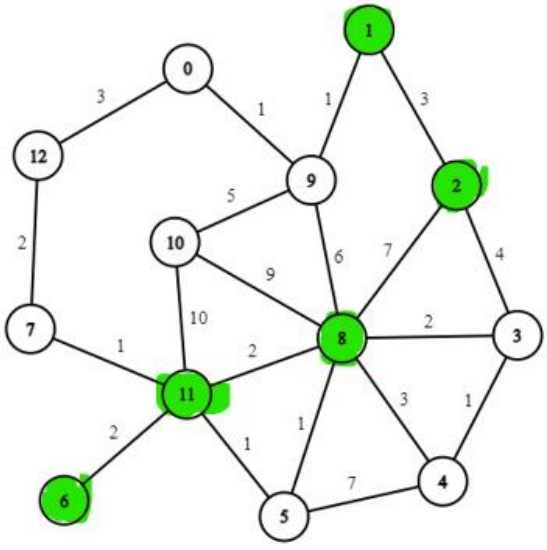
RUTA FINAL:



1,9,0,12,7,6

VALOR DE COSTO: 10

Árbol de búsqueda



Nodos > 1 > 2 > 8 > 11 > 6