Laboratorio 1 Complejidad Ciclomática

Moisés David Canaria Martínez.

Hilder Rocha Quiroz.

Pedro Luis Barrios Silgado.

Corporación Universitaria del Caribe – CECAR

Facultad De Ciencias Básicas, Ingenierías Y Arquitectura

Ingeniería de Sistemas

Sincelejo sucre

2022

Ing. Laudith María Lambraño.

Corporación Universitaria del Caribe – CECAR

Facultad De Ciencias Básicas, Ingenierías Y Arquitectura

Ingeniería de Sistemas

Sincelejo sucre

2022

Tabla de contenido

DESCRIPCIÓN DE ACTIVIDADES:

1) Analizar la siguiente situación problemática

Una empresa que fabrica de zapatos, paga a sus empleados de acuerdo a las horas trabajadas y a una tarifa de pago por hora, de acuerdo a las categorías que se indican en la siguiente tabla:

Categoría	tarifa
1	12000
2	17000
3	22000

Si la cantidad de horas trabajadas es mayor a 48 horas, la tarifa se incrementa en un 20% para las horas extras. Pero si las horas trabajadas están entre 24 y 48 se le adiciona un 10% al salario para subsidio de transporte y si es menos de 24 un 5%. Calcular el salario del trabajador dadas las horas trabajadas y la categoría.

2) Construir una función que resuelva el caso presentado en la situación problemática e implementarla en el lenguaje de programación deseado

ejercicio2.py > ...

```python
def CalcularPago(): # (Inicio)
    pago = 0
    horasTrabajadas = int(input("Digite las horas trabajadas: "))
    C = str(input("Seleccione la categoria: \n|    1:    | 12000\n|    2:    | 17000\n|    3:    | 22000\n\n:"))
    TarifaDePago = {"1":12000,"2":17000,"3":22000 }
    if C in TarifaDePago: # Si la categoria esta en el diccionario (A)
      if(horasTrabajadas > 48): # Si las horas trabajadas son mayores a 48 (B)
          tarifa_extra = (TarifaDePago[C] * 0.20) + (TarifaDePago[C]) # (C)
          horas_extras = horasTrabajadas - 48; # (D)
          pago = horas_extras * tarifa_extra + 48 * TarifaDePago[C] # (E)
      elif(horasTrabajadas >= 24): # Si las horas trabajadas son mayores o iguales a 24 (F)
        subsidio=(TarifaDePago[C] * horasTrabajadas) * 0.10 # (G)
        pago = (TarifaDePago[C] * horasTrabajadas) + subsidio # (H)
      else: # Si las horas trabajadas son menores a 24
        subsidio= (TarifaDePago[C] * horasTrabajadas) * 0.05 # (I)
        pago = (TarifaDePago[C] * horasTrabajadas) + subsidio # (J)
    else : # Si la categoria no esta en el diccionario
      print("La categoria no existe")  # (K)
    return print("El pago es: ", pago)
CalcularPago() # llamada a la funcion  Complejidad ciclomatica:  4 MOISES-CANARIA final
```

Imagen #1 Algoritmo

3) Calcular la complejidad ciclomática, determinar los caminos resultantes, y casos de prueba según la técnica de prueba del camino básico.

ejercicio.py > ...

```python
from radon.visitors import ComplexityVisitor
complejidad = ComplexityVisitor.from_code('''
def CalcularPago(): # (Inicio)
    pago = 0
    horasTrabajadas = int(input("Digite las horas trabajadas: "))
    C = str(input("Seleccione la categoria: |    1:    | 12000|    2:    | 17000|    3:    | 22000:"))
    TarifaDePago = {"1":12000,"2":17000,"3":22000 }
    if C in TarifaDePago: # Si la categoria esta en el diccionario (A)
      if(horasTrabajadas > 48): # Si las horas trabajadas son mayores a 48 (B)
          tarifa_extra = (TarifaDePago[C] * 0.20) + (TarifaDePago[C]) # (C)
          horas_extras = horasTrabajadas - 48; # (D)
          pago = horas_extras * tarifa_extra + 48 * TarifaDePago[C] # (E)
      elif(horasTrabajadas >= 24): # Si las horas trabajadas son mayores o iguales a 24 (F)
        subsidio=(TarifaDePago[C] * horasTrabajadas) * 0.10 # (G)
        pago = (TarifaDePago[C] * horasTrabajadas) + subsidio # (H)
      else: # Si las horas trabajadas son menores a 24
        subsidio= (TarifaDePago[C] * horasTrabajadas) * 0.05 # (I)
        pago = (TarifaDePago[C] * horasTrabajadas) + subsidio # (J)
    else : # Si la categoria no esta en el diccionario
      print("La categoria no existe")  # (K)
    return print("El pago es: ", pago)
CalcularPago() # llamada a la funcion  Complejidad ciclomatica:  4 MOISES-CANARIA final
''')
print("Complejidad ciclomatica: ", complejidad.functions[0].complexity)
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL JUPYTER

```
PS C:\Users\HP\OneDrive\Escritorio\Otros\Taller en clase> python ejercicio.py
Complejidad ciclomatica:  4
PS C:\Users\HP\OneDrive\Escritorio\Otros\Taller en clase>
```

Imagen #2 comprobación de complejidad ciclomatica con la librería Radon de Python

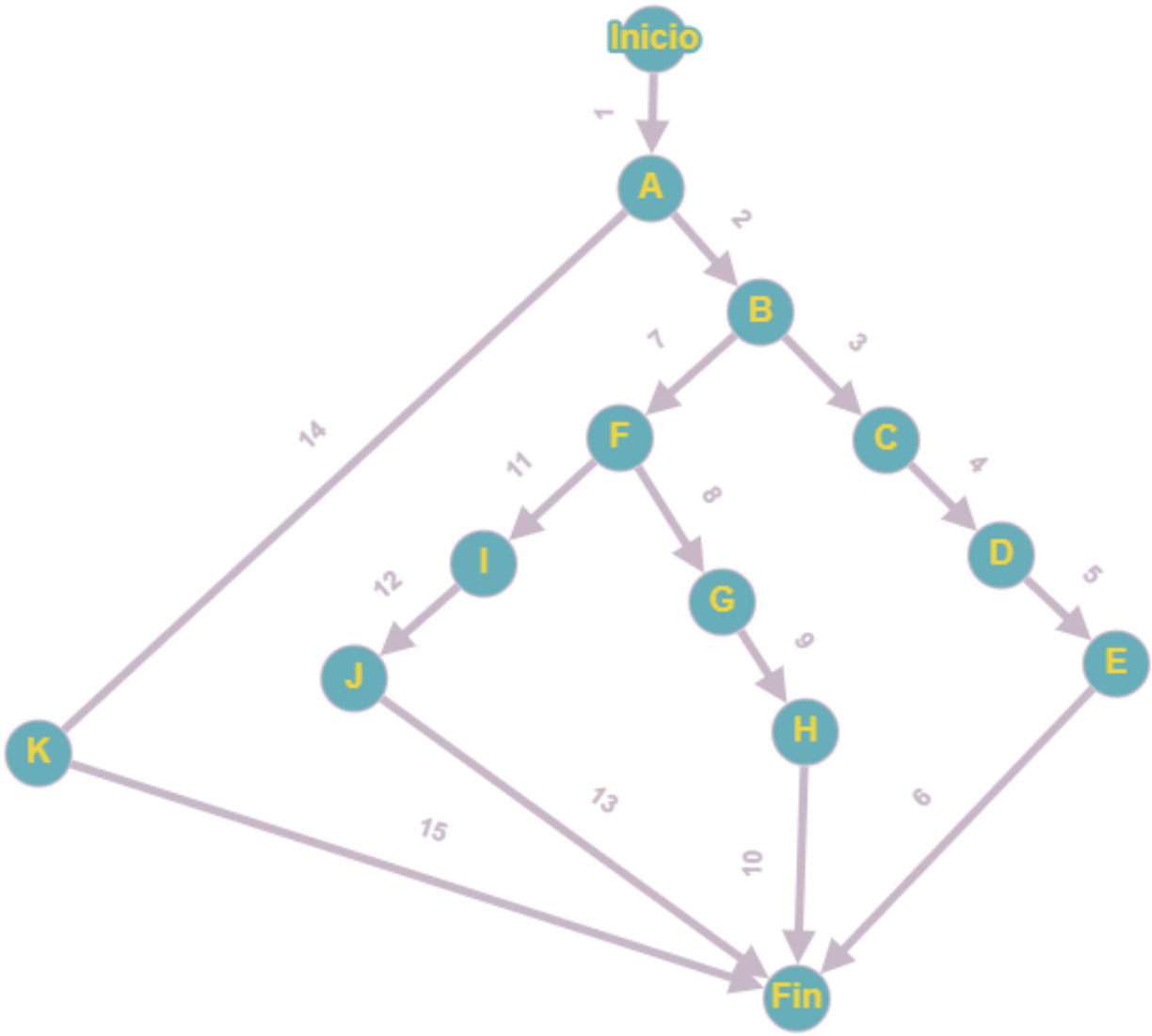

Imagen #4. Grafo de flujo 1

Complejidad Ciclomática:

$$V(G) = \#Aristas - \#Nodos + 2$$
$$V(G) = 15 - 13 + 2$$
$$V(G) = 4$$

Como la complejidad ciclomática es 4, entonces existen cuadro caminos resultantes:

Nota: if C in TarifaDePago: *# Si la categoría está en el diccionario de TarifaPago*

Caso 1: A-B-C-D-E-FIN (C, horasTrabajadas >= 24, tarifa_extra, horas_extras, return = pago)

Caso 2: A-B-F-G-H-FIN (C, horasTrabajadas <24, subsidio, return = pago)

Caso 3: A-B-F-I-J-FIN (C, horasTrabajadas>48, tarifa_extra, horas_extras, return = pago)

Caso 4: A-K-FIN (La categoría no existe, return = pago)

	Aristas															
Caminos	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	**Casos de prueba**
A-B-C-D-E-FIN	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	C=1, horasTrabajadas=49, tarifa_extra=14400, horas_extras=1, pago=576000, return = 590400
A-B-F-G-H-FIN	1	1	0	0	0	0	1	1	1	1	0	0	0	0	0	C=2, horasTrabajadas =23, subsidio=19550, pago=391000, return = 410550
A-B-F-I-J-FIN	1	1	0	0	0	0	1	0	0	0	1	1	1	0	0	C=3, horasTrabajadas =25, subsidio=55000, pago=550000, return =605000
A-K-FIN	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	C4=4, horasTrabajadas=10, Return = La categoria no existe

4) Aplica técnica de prueba de cobertura decisión/condición

$D1\ (C\ in\ TarifaDePago)$

$C1.1 \bullet C = 1$

$C1.2 \bullet C = 2$

$C1.3 \bullet C = 3$

$D2\ (horasTrabajadas > 48)$

$C2.1 \bullet horasTrabajadas > 48$

$D3\ (horasTrabajadas >= 24)$

$C3.1 \bullet horasTrabajadas >= 24$

$D4\ (horasTrabajadas < 24)$

$C4.1 \bullet horasTrabajadas < 24$

cobertura total de decisión/ condición:

CASO	Valor verdadero	Valor falso
C1.1	C=1	C=4
C1.2	C=2	C=5
C1.3	C=3	C=6
C2.1	horasTrabajadas=49	horasTrabajadas=48
C3.1	horasTrabajadas=24	horasTrabajadas=23
C4.1	horasTrabajadas =23	horasTrabajadas =24

Bibliografía

Using radon programmatically — Radon 4.1.0 documentation. (s/f). Readthedocs.Io. Recuperado el 10 de octubre de 2022, de https://radon.readthedocs.io/en/latest/api.html