

**GRAFICO DE LOGARITMOS**

:

DOCENTE

MSC. CLAUDIA UREÑA HINOJOSA

CARRERA

:

INGENERIA DE SISTEMAS

CÓDIGO

: h421-9

NOMBRE

CAHUAYA COLQIUE TONY

SMESTRE :

SEPTIMO

COCHABAMBA – BOLIVIA



|  |  |
| --- | --- |
| H421-9 | CAHUAYA COLQUE TONY PITER |

 Programe el gráfico del log  L = [⟨a,b,c,d,e,f,b,d,c,e,g⟩, ⟨a,b,d,c,e,g⟩, ⟨a,b,c,d,e,f,b,c,d,e,f,b,d,c,e,g⟩]   
CODIGO DEL PROGRAMA:  
import tkinter as tk

from tkinter import messagebox, scrolledtext

from collections import defaultdict

import graphviz

import os

from PIL import Image, ImageTk  # Para manejar imágenes en Tkinter

# --- Funciones del Algoritmo Alpha ---

def generate\_relation\_matrix(log):

    """Generar la matriz de relaciones a partir del log"""

    elements = sorted({event for sequence in log for event in sequence})

    matrix = {a: {b: '#L' for b in elements} for a in elements}

    direct\_successions = set()

    for sequence in log:

        for i in range(len(sequence) - 1):

            direct\_successions.add((sequence[i], sequence[i + 1]))

    for a in elements:

        for b in elements:

            if (a, b) in direct\_successions:

                if (b, a) in direct\_successions:

                    matrix[a][b] = '∥L'

                    matrix[b][a] = '∥L'

                else:

                    matrix[a][b] = '→L'

                    matrix[b][a] = '←L'

    return matrix

def compute\_XL\_corrected(log):

    """Calcular XL - Pares (A, B) sin filtrar"""

    relations\_matrix = generate\_relation\_matrix(log)

    elements = sorted({event for sequence in log for event in sequence})

    def check\_internal\_relations(set\_elements):

        """Verificar que no haya relaciones de paralelismo o causalidad inversa dentro del conjunto"""

        for i in range(len(set\_elements)):

            for j in range(len(set\_elements)):

                if i != j and relations\_matrix[set\_elements[i]][set\_elements[j]] != '#L':

                    return False

        return True

    def check\_set\_relations(set\_a, set\_b):

        """Verificar que todos los elementos de A estén en relación de causalidad directa con todos los elementos de B"""

        for a in set\_a:

            for b in set\_b:

                if relations\_matrix[a][b] != '→L':

                    return False

        return True

    XL = set()

    n = len(elements)

   for i in range(1, 1 << n):  # Generar todos los subconjuntos no vacíos de A

        set\_a = []

        for j in range(n):

            if i & (1 << j):

                set\_a.append(elements[j])

        if not check\_internal\_relations(set\_a):  # Verificar que A cumple las condiciones

|  |  |
| --- | --- |
| H421-9 | CAHUAYA COLQUE TONY PITER |

    continue

        for k in range(1, 1 << n):  # Generar todos los subconjuntos no vacíos de B

            set\_b = []

            for j in range(n):

                if k & (1 << j):

                    set\_b.append(elements[j])

            if not check\_internal\_relations(set\_b):  # Verificar que B cumple las condiciones

                continue

            if check\_set\_relations(set\_a, set\_b):  # Verificar que A -> B

                XL.add((frozenset(set\_a), frozenset(set\_b)))

    return XL

def compute\_YL(XL):

    """Calcular YL - Encontrar los pares maximales"""

    YL = set()

    for pair1 in XL:

        is\_maximal = True

        for pair2 in XL:

            if pair1 != pair2:

                A1, B1 = pair1

                A2, B2 = pair2

                if A1.issubset(A2) and B1.issubset(B2) and (len(A2) > len(A1) or len(B2) > len(B1)):

                    is\_maximal = False

                    break

        if is\_maximal:

            YL.add(pair1)

    return YL

def compute\_PL(YL, TI, TO):

    """Calcular PL - Construir el conjunto de lugares"""

    PL = set()

    for A, B in YL:

        place\_name = f"p\_{','.join(sorted(A))}\_{','.join(sorted(B))}"

        PL.add(place\_name)

    PL.add('iL')

    PL.add('oL')

    return PL

def compute\_FL(YL, TI, TO):

    """Calcular FL - Construir el conjunto de arcos"""

    FL = set()

    for A, B in YL:

        place\_name = f"p\_{','.join(sorted(A))}\_{','.join(sorted(B))}"

        for a in A:

            FL.add((a, place\_name))

        for b in B:

            FL.add((place\_name, b))

    for t in TI:

        FL.add(('iL', t))

    for t in TO:

        FL.add((t, 'oL'))

    return FL

|  |  |
| --- | --- |
| H421-9 | CAHUAYA COLQUE TONY PITER |

def visualize\_petri\_net(PL, TL, FL):

    """Crear visualización de la red de Petri usando Graphviz"""

    # Especifica la ruta al ejecutable dot de Graphviz

    os.environ["PATH"] += os.pathsep + 'C:\\Program Files\\Graphviz\\bin'

    dot = graphviz.Digraph(comment='Red de Petri')

    dot.attr(rankdir='LR')

    # Agregar lugares (círculos) con nombres

    for p in PL:

        if p == 'iL':

            dot.node(p, 'iL', shape='circle', style='filled', fillcolor='gray')

        elif p == 'oL':

            dot.node(p, 'oL', shape='circle', style='filled', fillcolor='gray')

        else:

            dot.node(p, p, shape='circle')  # Mostrar el nombre del lugar

    # Agregar transiciones (rectángulos)

    for t in TL:

        dot.node(t, t, shape='box')

    # Agregar arcos

    for f in FL:

        dot.edge(str(f[0]), str(f[1]))

    # Guardar el grafo como archivo temporal

    dot.render('temp\_petri\_net', format='png', cleanup=True)

    return 'temp\_petri\_net.png'

def mostrar\_grafica\_en\_ventana(image\_path):

    """Muestra la imagen de la red de Petri en una ventana separada"""

    ventana\_grafica = tk.Toplevel()

    ventana\_grafica.title("Red de Petri - Paso 8")

    # Cargar la imagen

    img = Image.open(image\_path)

    img = ImageTk.PhotoImage(img)

    # Mostrar la imagen en un Label

    label\_imagen = tk.Label(ventana\_grafica, image=img)

    label\_imagen.image = img  # Evitar que la imagen sea eliminada por el recolector de basura

    label\_imagen.pack()

def classify\_relations(log):

    """Clasificar las relaciones en el log"""

    relations = {

        'sucesiones\_directas': set(),

        'causalidades': set(),

        'paralelismo': set(),

        'decisiones': set()

    }

    for relation in log:

        for i in range(len(relation) - 1):

            relations['sucesiones\_directas'].add((relation[i], '>', relation[i + 1]))

    for relation in log:

        for i in range(len(relation) - 1):

            if (relation[i + 1], relation[i]) not in relations['sucesiones\_directas']:

                forward\_relation = (relation[i], relation[i + 1])

|  |  |
| --- | --- |
| H421-9 | CAHUAYA COLQUE TONY PITER |

            inverse\_relation = (relation[i + 1], relation[i])

                if inverse\_relation in relations['causalidades']:

                    relations['paralelismo'].add(forward\_relation)

                    relations['paralelismo'].add(inverse\_relation)

                    relations['causalidades'].remove(inverse\_relation)

                else:

                    relations['causalidades'].add(forward\_relation)

    all\_letters = set()

    for relation in log:

        all\_letters.update(relation)

    all\_letters = sorted(all\_letters)

    for letter in all\_letters:

        for other\_letter in all\_letters:

            if letter != other\_letter:

                decision = (letter, '#', other\_letter)

                if (letter, '>', other\_letter) not in relations['sucesiones\_directas'] and \

                   (other\_letter, '>', letter) not in relations['sucesiones\_directas']:

                    relations['decisiones'].add(decision)

    for letter in all\_letters:

        relations['decisiones'].add((letter, '#', letter))

    return relations

def generate\_table(sucesiones\_directas, causalidades, paralelismo, decisiones):

    """Generar la tabla de relaciones"""

    elementos = set()

    for sucesion in sucesiones\_directas:

        elementos.update(sucesion)

    elementos = sorted(elementos)

    tabla = [['' for \_ in range(len(elementos) + 1)] for \_ in range(len(elementos) + 1)]

    for i, elemento in enumerate(elementos):

        tabla[0][i + 1] = elemento

        tabla[i + 1][0] = elemento

    for causalidad in causalidades:

        tabla[elementos.index(causalidad[0]) + 1][elementos.index(causalidad[1]) + 1] = '→L'

        tabla[elementos.index(causalidad[1]) + 1][elementos.index(causalidad[0]) + 1] = '←L'

    for paralelo in paralelismo:

        tabla[elementos.index(paralelo[0]) + 1][elementos.index(paralelo[1]) + 1] = '∥L'

        tabla[elementos.index(paralelo[1]) + 1][elementos.index(paralelo[0]) + 1] = '∥L'

    for decision in decisiones:

        tabla[elementos.index(decision[0]) + 1][elementos.index(decision[2]) + 1] = '#L'

    return tabla

def mostrar\_tabla\_relaciones(log):

    """Mostrar la tabla de relaciones en formato de texto"""

    relations = classify\_relations(log)

    tabla = generate\_table(relations['sucesiones\_directas'], relations['causalidades'],

                         relations['paralelismo'], relations['decisiones'])

    tabla\_str = "Tabla de Relaciones:\n"

    for fila in tabla:

        tabla\_str += "\t".join(fila) + "\n"

    return tabla\_str

|  |  |
| --- | --- |
| H421-9 | CAHUAYA COLQUE TONY PITER |

# --- Interfaz Gráfica ---

def procesar\_input():

    """Procesa la entrada del usuario y genera la red de Petri"""

    log\_input = entry.get()

    try:

        sequences = log\_input.split('>,<')

        log = [sequence.strip('<>').split(',') for sequence in sequences]

        tabla\_str = mostrar\_tabla\_relaciones(log)

        TL = sorted({evento for seq in log for evento in seq})

        TI = {seq[0] for seq in log}

        TO = {seq[-1] for seq in log}

        XL = compute\_XL\_corrected(log)

        YL = compute\_YL(XL)

        PL = compute\_PL(YL, TI, TO)

        FL = compute\_FL(YL, TI, TO)

        # Generar visualización de la red de Petri

        image\_path = visualize\_petri\_net(PL, TL, FL)

        # Mostrar la gráfica en una ventana separada

        mostrar\_grafica\_en\_ventana(image\_path)

        steps = "===== Algoritmo Alpha =====\n\n"

        steps += tabla\_str + "\n\n"

        steps += "Paso 1: TL (Conjunto de transiciones):\n" + str(TL) + "\n\n"

        steps += "Paso 2: TI (Transiciones iniciales):\n" + str(TI) + "\n\n"

        steps += "Paso 3: TO (Transiciones finales):\n" + str(TO) + "\n\n"

        steps += "Paso 4: XL (Pares (A, B) sin filtrar):\n"

        for A, B in XL:

            steps += f"({set(A)}, {set(B)})\n"

        steps += "\nPaso 5: YL (Pares maximales):\n"

        for A, B in YL:

            steps += f"({set(A)}, {set(B)})\n"

        steps += "\nPaso 6: PL (Lugares):\n" + str(PL) + "\n\n"

        steps += "Paso 7: FL (Arcos):\n" + str(FL) + "\n\n"

        steps += "Paso 8: α(L) = (PL, TL, FL)\n"

        steps += "Red de Petri resultante:\n"

        steps += f"Lugares (P): {PL}\n"

        steps += f"Transiciones (T): {set(TL)}\n"

        steps += f"Arcos (F): {FL}\n"

        steps += "\nSe ha generado una imagen con la visualización de la red de Petri.\n"

        steps += "\n------------------------------------------------\n\n"

        text\_box.config(state=tk.NORMAL)

        text\_box.delete(1.0, tk.END)

        text\_box.insert(tk.END, steps)

        text\_box.config(state=tk.DISABLED)

    except Exception as e:

        messagebox.showerror("Error", f"Ocurrió un error: {e}")

# --- Configuración de la ventana principal ---

root = tk.Tk()

root.title("Algoritmo Alpha - Pasos 1-8 con Visualización")

root.geometry("800x700")

root.configure(bg='lightblue')

|  |  |
| --- | --- |
| H421-9 | CAHUAYA COLQUE TONY PITER |

label = tk.Label(root, text="Ingrese los ejercicios (formato <a,b,e,f>,<a,b,e,c,d,b,f>,...):", bg='lightblue')

label.pack(pady=10)

entry = tk.Entry(root, width=80)

entry.pack(pady=10)

button = tk.Button(root, text="Procesar", command=procesar\_input, bg='lightgreen')

button.pack(pady=10)

text\_box = scrolledtext.ScrolledText(root, width=90, height=25, wrap=tk.WORD)

text\_box.pack(pady=10)

text\_box.config(state=tk.DISABLED)

root.mainloop()

Diagrama

Descripción generada automáticamenteEjemplo: