

# INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE OCCIDENTE

## FUNDAMENTOS DE CIENCIAS COMPUTACIONALES



### CALCULADORA DE RELACIONES

Presentan:

**Ana Carolina Arellano Valdez**

**Luis Raúl Acosta Mendoza**

**Arlyn Linette Medina García**

Profesora:

**Silvia Piña Romero**

Fecha:

**12/05/2022**

# PROYECTO

## Relaciones y sus Aplicaciones en las Ciencias Computacionales

Una relación es un subconjunto del producto cartesiano, formado por los pares cuyos elementos cumplan una propiedad establecida por la relación y se escriben formalmente se escriben de la siguiente manera:

(Tomando en cuenta los conjuntos A y B)

$$aRb \Rightarrow ***$$

\*\*\* Aquí se escribe la condición que cumplen los elementos ( $a < b$ ,  $a = b$ , etc.).

Las relaciones pueden contar con ciertas propiedades dependiendo de las características de esta.

(Considerando los elementos de  $A \times A$ )

La propiedad reflexiva ocurre cuando para todos los elementos de A existe una pareja ( $a, a$ ) en la relación, la propiedad simétrica si para toda pareja ( $x, y$ ) también existe ( $y, x$ ) en la relación, la propiedad transitiva es cuando para todas las parejas ( $a, b$ ) existe una pareja ( $b, c$ ) tal que ( $a, c$ ) también existe en la relación. Por último, se considera que es una relación de equivalencia cuando cumple con las 3 propiedades mencionadas.

Dentro del área de las ciencias computacionales, las relaciones tienen una gran importancia y son usadas frecuentemente, por ejemplo, en bases de datos, estructuras de datos, redes, autómatas, lenguajes, entre otros. Generalmente el uso que se les da es para poder acceder y manejar la información más fácil y ordenadamente.

## **Explicación técnica de la programación**

- Nuestro programa se realizo con la interfaz Tkinter, y esta principalmente compuesta por varias funciones. Cada función esta relacionada a las 5 diferentes relaciones que escogimos a la vez que la determinación de si son reflexivas, simetricas y/o transitivas.

*Función Inicial: Esta función se podría considerar la más importante ya que con esto se inicializa todo. En pocas palabras, al usuario ingresar el conjunto con el que desea trabajar, esta función crea los pares ordenados para después aplicarles las relaciones.*

### Función Producto Cartesiano:

Esta función cuenta con un parámetro el cual es el conjunto ingresado por el usuario. Después se entran en dos ciclos “for” los cuales revisan valor por valor el mismo conjunto. Cada que se entra a estos dos ciclos, se van agregando las diferentes combinaciones del conjunto, de manera que se van creando los pares ordenados.

*Funciones de las relaciones escogidas: (Todas contienen el mismo parámetro que es el resultado de los pares ordenados de la función anterior. De igual manera en todas las funciones se regresa la lista “pares” la cual es uno de los resultados que le aparecerá al usuario en la interfaz)*

### Función Mayor o Igual que:

Esta función empieza con un ciclo “for” el cual recorre cada uno de los pares ordenados que se resultaron del conjunto que el usuario introdujo. Posteriormente entra en una condicional “if” en la que, si el primer número del par ordenado es mayor o igual que el segundo número, agrega ese correspondiente par ordenado a una lista llamada pares con el comando “append”.

### Función Menor o Igual que:

En esta función se hace lo mismo que en la anterior, con la única diferencia de que la condicional “if”, si el primer número del par ordenado es menor o igual que el segundo número, se agrega ese correspondiente par ordenado a la lista “pares”.

### Función Igual:

En esta función de igual manera se hace lo mismo que en las anteriores, con la diferencia de que la condicional “if”, si el primero número del par ordenado es completamente igual que el segundo, se agrega ese correspondiente par ordenado a la lista “pares”.

### Función Múltiplos:

Esta función principalmente agrega los pares ordenados que cumplan con la condición de que el segundo número del par ordenado sea múltiplo del primero. Esto lo hacemos con el operador “%” el cual revida el residuo de una división. Si el residuo de la división entre el

segundo número y el primero es igual a 0, el correspondiente par ordenado se agrega a la lista “pares”.

#### Función Mitad:

En esta función lo que se busca es encontrar los pares ordenados que cumplan con la condición de que el segundo número del par ordenado sea el doble que el del primero. Esto de igual manera, lo hacemos con una condicional “if” en la cual, si el segundo número del par ordenado entre dos es igual al primero, se agrega a la lista “pares”.

*Funciones para determinar si son Relaciones Reflectiva/Simétricas/Transitivas: (Todas contienen el mismo parámetro el cual es el resultado de los pares que cumplen con la relación seleccionada por el usuario de las funciones de las relaciones. De igual manera en todas se regresa el valor verdadero/falso el cual le aparecerá al usuario en la interfaz)*

#### Función Relación Reflexiva

En esta función primeramente se entra en un ciclo “for” el cual recorre cada uno de los pares ordenados de la lista “pares”. Después se entra en una condicional “if” en la que si por cada par ordenado (a,b), a es diferente de b y se encuentra en la lista pares, se regresa el valor falso de manera inmediato. De no ser así se regresa el valor verdadero. Esto para que se revise que cada uno de los pares cumpla con la condición.

#### Función Relación Simétrica

Esta función consiste en un ciclo “for” el cual recorre cada uno de los pares que formaron parte de la relación. En este se encuentra una condicional “if”, en la que checa si por cada par ordenado (a,b) no se encuentra el par (b,a) en la lista de pares, se regresa el valor de falso de manera inmediata. De no ser así se regresará el valor verdadero. Esto de igual manera nos permite revisar que se cumpla la condición en todos los pares ordenados, sin excepción alguna.

#### Función Relación Transitiva

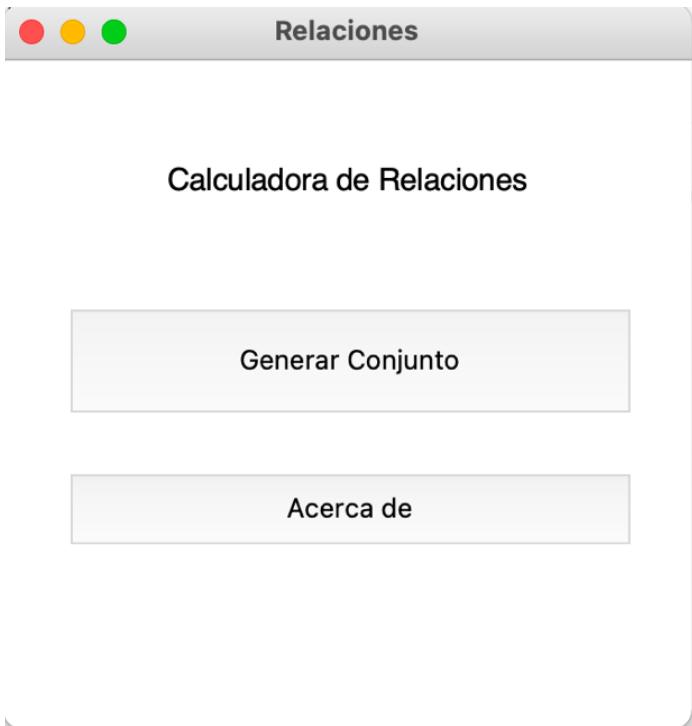
Esta relación consiste en dos ciclos “for”, de los va recorriendo cada uno de los pares ordenados de la lista “pares”. El primero ciclo recorre los pares de manera (a,b) mientras que el segundo de manera (c,d). Después se entra en una condicional “if” en la cual si se cumple que “b” es igual a “c” y que el par ordenado (a,d) está en la lista pares, se regresa el valor verdadero. De no ser así se regresa el valor de falso.

#### Función Relación Equivalencia:

Esta es la última función la cual tiene como parámetro los tres resultados de las anteriores funciones. Esta función consiste en una simple condicional “if” de la cual, si el valor de los tres parámetros es igual a “Verdadero”, regresa verdadero. De no ser así regresa falso.

## Capturas de Pantalla de Ejemplo:

Página Principal:



Crear conjunto:

This screenshot shows the "Opcion crear propio conjunto" (Option to create your own set) section. It includes fields for entering values, buttons for adding and removing values, and a "Conjunto Final" (Final Set) display showing the numbers 1 through 10. Below this is the "Opcion crear conjunto random" (Option to create a random set) section, which has a field for entering the size of the set and a "Generar conjunto random" (Generate random set) button.

**Opcion crear propio conjunto**

Ingrese valores del conjunto:

Agregar valor    Eliminar valor

**Conjunto Final**

Conjunto Creado

1 2 3 4 5 6 7 8 9 10

**Opcion crear conjunto random**

Ingrese el tamaño del conjunto:

Generar conjunto random

Borrar Conjunto    Continuar

5 opciones de las Relaciones escogidas:

● ● ● Opciones de Relaciones

### Tipos de Relaciones

- Mayor o igual que
- Menor o igual que
- Igual
- Multiplos
- Doble

**Regresar**

Opción Mayor o Igual Que:

● ● ● Opciones de Relaciones

Pares que cumplen con la relación:	Determinación de Relación:
1 {1 1}	Reflexiva      Falso
2 {2 1}	Simétrica      Falso
3 {2 2}	Transitiva      Verdadero
4 {3 1}	Equivalencia      Falso
5 {3 2}	
6 {3 3}	
7 {4 1}	
8 {4 2}	
9 {4 3}	
10 {4 4}	

**Regresar**

Opción Menor o Igual Que:

● ○ ●

Opciones de Relaciones

**Pares que cumplen con la relación:**

- 1 {1 1}
- 2 {1 2}
- 3 {1 3}
- 4 {1 4}
- 5 {1 5}
- 6 {1 6}
- 7 {1 7}
- 8 {1 8}
- 9 {1 9}
- 10 {1 10}

**Determinación de Relación:**

Reflexiva	Falso
Simétrica	Falso
Transitiva	Verdadero
Equivalencia	Falso

[Regresar](#)

Opción Igual:

● ○ ●

Opciones de Relaciones

**Pares que cumplen con la relación:**

- 1 {1 1}
- 2 {2 2}
- 3 {3 3}
- 4 {4 4}
- 5 {5 5}
- 6 {6 6}
- 7 {7 7}
- 8 {8 8}
- 9 {9 9}
- 10 {10 10}

**Determinación de Relación:**

Reflexiva	Verdadero
Simétrica	Verdadero
Transitiva	Verdadero
Equivalencia	Verdadero

[Regresar](#)

Opción Múltiplo:

Opciones de Relaciones

Pares que cumplen con la relación:

- 1 {1 1}
- 2 {1 2}
- 3 {1 3}
- 4 {1 4}
- 5 {1 5}
- 6 {1 6}
- 7 {1 7}
- 8 {1 8}
- 9 {1 9}
- 10 {1 10}

Determinación de Relación:

Reflexiva	Falso
Simétrica	Falso
Transitiva	Verdadero
Equivalencia	Falso

[Regresar](#)

Opción Doble:

Opciones de Relaciones

Pares que cumplen con la relación:

- 1 {1 2}
- 2 {2 4}
- 3 {3 6}
- 4 {4 8}
- 5 {5 10}

Determinación de Relación:

Reflexiva	Falso
Simétrica	Falso
Transitiva	Falso
Equivalencia	Falso

[Regresar](#)

## Código Comentado:

#Parte de las funciones

pares = []

def ProductoCartesiano(conj1):

#Lista que se usará en las funciones de las relaciones

cartesiano = []

#Combina cada valor del primer conjunto que escogio el usuario y lo combina con el segundo

for a in conj1:

    for b in conj1:

        cartesiano.append((a, b))

return (cartesiano)

def MayorIgualQue(cartesiano):

#Recorre todos los elementos de los pares ordenados

for i in range (len(cartesiano)):

#Checa que el primer valor del par ordenado sea mayor o igual al segundo

if cartesiano[i][0]>=cartesiano[i][1]:

    pares.append(cartesiano[i])

return (pares)

def MenorIgualQue(cartesiano):

#Recorre todos los elementos de los pares ordenados

for i in range (len(cartesiano)):

if cartesiano[i][0]<=cartesiano[i][1]:

    pares.append(cartesiano[i])

return (pares)

```
def Igual(cartesiano):
    #Recorre todos los elementos de los pares ordenados
    for i in range (len(cartesiano)):
        #Checa que el primer valor del par ordenado sea menor o igual al segundo
        if cartesiano[i][0]==cartesiano[i][1]:
            pares.append(cartesiano[i])
    return (pares)
```

```
def Multiplos(cartesiano):
    #Recorre todos los elementos de los pares ordenados
    for i in range (len(cartesiano)):
        #Checa que el primer valor del par ordenado sea igual al segundo
        if int(cartesiano[i][1]) % int(cartesiano[i][0]) == 0:
            pares.append(cartesiano[i])
    return (pares)
```

```
def Doble(cartesiano):
    #Recorre todos los elementos de los pares ordenados
    for i in range (len(cartesiano)):
        #Checa que el segundo valor del par ordenado sea el doble que el primero
        if (int(cartesiano[i][1]) / 2) == int(cartesiano[i][0]):
            pares.append(cartesiano[i])
    return (pares)
```

```
def RelacionTransitiva(pares):
    #Recorre todos los elementos de los pares que cumplen con la relación
    for a,b in pares:
        #Recorre todos los elementos de los pares que cumplen con la relación con otros parámetros
```

```

for c,d in pares:
    #Condicional que checa que se cumple la relación
    if b == c and ((a,d) in pares):
        return ("Verdadero")
    else:
        return ("Falso")
    return ("Falso")

def RelacionReflexiva(pares):
    #Recorre todos los elementos de los pares que cumplen con la relación
    for a,b in pares:
        #Condicional que checa si no se cumple la relación
        if a != b and ((a,b) in pares):
            return ("Falso")
        return ("Verdadero")

def RelacionSimetrica(pares):
    #Recorre todos los elementos de los pares que cumplen con la relación
    for a,b in pares:
        #Condicional que checa si no se cumple la relación
        if (b,a) not in pares:
            return ("Falso")
        return ("Verdadero")

def RelacionEquivalencia(Reflectiva, Simetrica, Transitiva):
    #Condicional que checa si se todos los valores de las anteriores 3 funciones son verdadero.
    if Reflectiva == "Verdadero" and Simetrica == "Verdadero" and Transitiva == "Verdadero":
        return ("Verdadero")

```

```
else:  
    return ("Falso")  
  
#Parte de la interfaz  
  
import tkinter as tk  
from tkinter import *  
from tkinter import ttk  
from tkinter import scrolledtext  
import functions  
from random import randint  
import random  
import tkinter.font as tkFont  
  
#Lista que el usuario ingresará primero  
conjunto = []  
  
def agregar_elemento(entry, conjunto):  
    elemento = int(entry.get())  
    if elemento not in conjunto:  
        conjunto.append(int(elemento))  
    entry.delete(0,END)  
  
def eliminar_elemento(entry, conjunto):  
    conjunto.pop(-1)  
    entry.delete(0,END)  
  
def agregar_conjunto_random(entry, conjunto, num):  
    conjunto.clear()
```

```
i=0
while i < int(num):
    elemento = randint(1, 100)
    if elemento not in conjunto:
        conjunto.append(int(elemento))
    i+=1

entry.delete(0,END)

def limpiar_conjuntos(conjunto1):
    conjunto.clear()

#Ventana Final que se le presentará al usuario cuando se haya escogido la relación
def Resultado(OpcionesWindow, Pares,):
    ResultadoWindow = tk.Toplevel(OpcionesWindow)
    ResultadoWindow.focus()
    ResultadoWindow.grab_set()
    ResultadoWindow.title('Opciones de Relaciones')
    ResultadoWindow.geometry('640x260')
    OpcionesWindow.withdraw()

    Label1 = Label(ResultadoWindow, text = "Pares que cumplen con la relación:",font= "Helvetica 15 bold")
    Label1.place(x=15, y=5)
    imprimir = scrolledtext.ScrolledText(ResultadoWindow, font= "Helvetica 13", width=20, height=10)
    for i in range(len(Pares)):
        imprimir.insert(tk.INSERT, (i+1,Pares[i]))
```

```
imprimir.insert(tk.END, '\n')
imprimir.configure(state ='disabled')
imprimir.place(x=30, y=50)

Label2 = Label(ResultadoWindow, text= "Determinación de Relación: ", font= "Helvetica
15 bold")
Label2.place(x=350, y=5)

Label3 = Label(ResultadoWindow, text= "Reflexiva ")
Label3.place(x=355, y=40)

Label4 = Label(ResultadoWindow, text= "Simétrica ")
Label4.place(x=355, y=75)

Label5 = Label(ResultadoWindow, text= "Transitiva ")
Label5.place(x=355, y=110)

Label6 = Label(ResultadoWindow, text= "Equivalencia ")
Label6.place(x=355, y=145)

ReturnBoton = Button(ResultadoWindow, text= "Regresar", height=1, width=15, command
= lambda:[ResultadoWindow.destroy(), Pares.clear(),opciones_relaciones(OpcionesWindow,
conjunto)])
ReturnBoton.place(x=45, y= 220)

labelR = Label(ResultadoWindow, text = functions.RelacionReflexiva(Pares))
labelR.place(x=430 , y=40)

labelS = Label(ResultadoWindow, text = functions.RelacionSimetrica(Pares))
labelS.place(x=430 , y=75)

labelT = Label(ResultadoWindow, text = functions.RelacionTransitiva(Pares))
labelT.place(x=430 , y=110)

labelE = Label(ResultadoWindow, text =
functions.RelacionEquivalencia(functions.RelacionReflexiva(Pares),
functions.RelacionSimetrica(Pares), functions.RelacionTransitiva(Pares)))
labelE.place(x=450, y=145)

def opciones_relaciones(mainWindow, conjunto):
```

```

OpcionesWindow = tk.Toplevel(mainWindow)
OpcionesWindow.focus()
OpcionesWindow.grab_set()
OpcionesWindow.title('Opciones de Relaciones')
OpcionesWindow.geometry('335x325')
mainWindow.withdraw()

Label1 = Label(OpcionesWindow, text="Tipos de Relaciones", font= "Helvetica 15 bold")
Label1.place(x=90, y=10)

BotonMayor = tk.Button(OpcionesWindow, text='Mayor o igual que', width=25, height=2,
command = lambda:
[Resultado(OpcionesWindow, functions.MayorIgualQue(functions.ProductoCartesiano(conjunto)
))])
BotonMayor.place(x=50, y=50)

BotonMenor = tk.Button(OpcionesWindow, text='Menor o igual que', width=25, height=2,
command = lambda:
[Resultado(OpcionesWindow, functions.MenorIgualQue(functions.ProductoCartesiano(conjunto)
))])
BotonMenor.place(x=50, y=100)

BotonIgual = tk.Button(OpcionesWindow, text='Igual', width=25, height=2, command =
lambda: [Resultado(OpcionesWindow,
functions.Igual(functions.ProductoCartesiano(conjunto))))]
BotonIgual.place(x=50, y=150)

BotonMultiplos = tk.Button(OpcionesWindow, text='Multiplos', width=25, height=2,
command = lambda: [Resultado(OpcionesWindow,
functions.Multiplos(functions.ProductoCartesiano(conjunto))))]
BotonMultiplos.place(x=50, y=200)

Boton5 = tk.Button(OpcionesWindow, text='Doble', width=25, height=2, command =
lambda: [Resultado(OpcionesWindow,
functions.Doble(functions.ProductoCartesiano(conjunto))))]
Boton5.place(x=50, y=250)

ReturnBoton = Button(OpcionesWindow, text= "Regresar", height=1, width=15, command
= lambda:[OpcionesWindow.destroy(), conjunto.clear(), calcWindow(mainWindow, 0)])

```

```
ReturnBoton.place(x=90, y= 300)

def AcercaDe(parentwindow,mode):
    global focusedDisplay
    mainWindow = tk.Toplevel(parentwindow)
    mainWindow.focus()
    mainWindow.grab_set()
    mainWindow.geometry('500x250')
    parentwindow.withdraw()
    mainWindow.title('Información del programa')
    text2 = tk.StringVar()

    label1 = tk.Label(mainWindow,text="Creado por ", font='Helvetica 16')
    label1.place(x=180,y=20)
    label2 = tk.Label(mainWindow,text="Luis Raúl Acosta Mendoza", font='Helvetica 10')
    label2.place(x=150,y=50)
    label3 = tk.Label(mainWindow,text="Ana Carolina Arellano Valdez", font='Helvetica 10')
    label3.place(x=150,y=75)
    label4 = tk.Label(mainWindow,text="Arlyn Linette Medina García", font='Helvetica 10')
    label4.place(x=150,y=100)

    mainWindow.wait_window()
    parentwindow.deiconify()

def calcWindow(root, mode):
    global focusedDisplay
    mainWindow = tk.Toplevel(root)
    mainWindow.focus()
```

```

mainWindow.grab_set()
mainWindow.geometry('640x300')
root.withdraw()

def guardar_conjuntos():
    conjunto.sort()
    Label1 = Label(mainWindow, text= "Conjunto Final", font= "Helvetica 15 bold")
    Label1.place(x=300,y=75)
    mostrarconjunto = LabelFrame(mainWindow, text= "Conjunto Creado", font=
    "Helvetica 10 italic", width=310, height=34)
    c = Label(mostrarconjunto, text= conjunto, font= "Helvetica 10")
    mostrarconjunto.place(x=300, y=100)
    c.place(x=0, y=0)
    Limpiarboton = Button(mainWindow, text= "Borrar Conjunto", command = lambda:
    [limpiar_conjuntos(conjunto),guardar_conjuntos()])
    Limpiarboton.place(x=330, y= 140)
    Continuarboton = Button(mainWindow, text= "Continuar", command = lambda:
    [opciones_relaciones(mainWindow, conjunto)])
    Continuarboton.place(x=500, y=140)

if mode == 0:
    Label1 = Label(mainWindow, text = "Opcion crear propio conjunto", font="Helvetica
    15 bold")
    Label1.place(x=25, y=5)
    Label2 = Label(mainWindow, text = "Ingrese valores del conjunto:")
    Label2.place(x=40, y=30)
    ConjEntry = Entry(mainWindow)
    ConjEntry.place(x=40,y=60)
    AgregarBoton = Button(mainWindow, text = "Aregar valor", command = lambda:
    [agregar_elemento(ConjEntry, conjunto),guardar_conjuntos()])

```

```
AgregarBoton.place(x=30, y=100)

EliminarBoton = Button(mainWindow, text = "Eliminar valor", command = lambda:
[eliminar_elemento(ConjEntry, conjunto),guardar_conjuntos()])

EliminarBoton.place(x=150, y=100)

Label3 = Label(mainWindow, text = "Opcion crear conjunto random", font="Helvetica
15 bold")

Label3.place(x=25,y=135)

ConjRandomLabel = Label(mainWindow, text= "Ingrese el tamaño del conjunto:")

ConjRandomLabel.place(x=40, y=160)

RandomConjEntry = Entry(mainWindow)

RandomConjEntry.place(x=40, y=190)

ConjRandom = Button(mainWindow, text = "Generar conjunto random", command =
lambda: [agregar_conjunto_random(ConjEntry,
conjunto,RandomConjEntry.get()),guardar_conjuntos()])

ConjRandom.place(x=48, y=230)
```

```
mainWindow.wait_window()  
root.deiconify()
```

```
about = tk.Button(root,
                  text='Acerca de',
                  width=30,
                  height=2,
                  bg= 'SkyBlue4',
                  command=lambda: AcercaDe(root, 1))

title.place(x=75, y=30)
GenerateSet.place(x=32, y=120)
about.place(x=32, y=200)
root.mainloop()
```

## **Referencias:**

Capítulo 6: Relaciones. (2022). Recuperado el 12 May 2022, de  
[https://libroweb.alfaomega.com.mx/book/685/free/ovas\\_statics/presentaciones1/matecompu\\_cap6.pdf](https://libroweb.alfaomega.com.mx/book/685/free/ovas_statics/presentaciones1/matecompu_cap6.pdf)

1.4 Propiedades de las relaciones. (2022). Recuperado el 12 May 2022, de  
<http://mate.cucei.udg.mx/matdis/1rel/1rel4.htm>

Bruno López Takeyas, Relaciones (2022). Recuperado el 12 May 2022, de  
[http://www.itnuevolaredo.edu.mx/takeyas/apuntes/matematicas\\_discretas/apuntes/Relaciones.pdf](http://www.itnuevolaredo.edu.mx/takeyas/apuntes/matematicas_discretas/apuntes/Relaciones.pdf)

Tkinter. (2022). tkinter — Interface de Python para Tcl/Tk — documentación de Python - 3.10.4. Python, Tkinter. Recuperado 25 de abril de 2022, de  
<https://docs.python.org/es/3/library/tkinter.html>