

**UNIVERSIDAD TECNOLÓGICA DE PANAMÁ
FACULTAD DE INGENIERÍA DE SISTEMAS COMPUTACIONALES
DEPARTAMENTO DE COMPUTACIÓN Y SIMULACIÓN DE SISTEMAS**

ESTRUCTURAS DISCRETAS PARA LA COMPUTACIÓN

LABORATORIO 4

**Facilitador:
Tomás J. Concepción Miranda**

Estudiantes: **Roger Zurdo 4-764-2276
Joy Nelaton 8-902-1282**

Grupo: **1IL-123**

Fecha: **07/06/2023**

PASO QUE SE USO PARA EL DESARROLLO DE LOS PROBLEMAS

P1

1. Función `es_reflexiva(A, R)`: Esta función verifica si una relación R es reflexiva en un conjunto A . Recorre todos los elementos en A y comprueba si el par (e, e) está presente en R para cada elemento e . Si encuentra algún par faltante, devuelve `False`; de lo contrario, devuelve `True`.
2. Función `es_irreflexiva(A, R)`: Esta función verifica si una relación R es irreflexiva en un conjunto A . Recorre todos los elementos en A y comprueba si el par (e, e) está presente en R para algún elemento e . Si encuentra algún par presente, devuelve `False`; de lo contrario, devuelve `True`.
3. Función `es_simetrica(A, R)`: Esta función verifica si una relación R es simétrica en un conjunto A . Recorre todos los pares (i, j) en R y verifica si el par inverso (j, i) también está presente en R . Si encuentra algún par inverso faltante, devuelve `False`; de lo contrario, devuelve `True`.
4. Función `es_asimetrica(A, R)`: Esta función verifica si una relación R es asimétrica en un conjunto A . Recorre todos los pares (i, j) en R y verifica si el par inverso (j, i) está presente en R . Si encuentra algún par inverso presente, devuelve `False`; de lo contrario, devuelve `True`.
5. Función `es_antisimetrica(A, R)`: Esta función verifica si una relación R es antisimétrica en un conjunto A . Recorre todos los pares (i, j) en R y verifica las siguientes condiciones:

6. Definimos una función llamada "es_transitiva" que toma dos argumentos: A y R .

Dentro de la función, hay dos bucles "for" anidados que recorren los elementos de la relación R y el conjunto A .

En el bucle interno, se realiza una validación de transitividad comprobando si el par (b, c) está en la relación R .

Si se cumple la condición, se verifica si el par (a, c) no está en la relación R .

Si la condición no se cumple, se devuelve `False`, lo que indica que la relación R no es transitiva.

Si se completa el recorrido de ambos bucles sin encontrar ninguna violación de transitividad, se devuelve `True`.

Imprime los valores de $A7$, $R10$ y $R11$.

Llama a la función "es_transitiva" con los argumentos $A7$ y $R10$, e imprime el resultado.

Llama a la función "es_transitiva" con los argumentos A7 y R11, e imprime el resultado.

7. Imprime los valores de A8 y R12.

Llama a una función "es_simetrica" que no está definida en el código.

Llama a la función "es_transitiva" con los argumentos A8 y R12, e imprime el resultado.

Imprime los valores de A9 y R13.

Llama a una función "es_antisimetrica" que no está definida en el código.

Llama a la función "es_transitiva" con los argumentos A9 y R13, e imprime el resultado.

Imprime los valores de A10 y R14.

Llama a una función "es_irreflexiva" que no está definida en el código.

Llama a la función "es_transitiva" con los argumentos A10 y R14, e imprime el resultado.

Imprime los valores de A11 y R15.

Llama a una función "es_reflexiva" que no está definida en el código.

Llama a una función "es_simetrica" que no está definida en el código.

8. Define una función llamada "dominio" que toma un argumento R.

Dentro de la función, se crea un conjunto vacío llamado "dom" para almacenar el dominio.

Se recorren los elementos de la relación R y se agrega el primer elemento de cada par al conjunto "dom".

Finalmente, se devuelve el conjunto "dom".

Imprime el valor de R1 y llama a la función "dominio" con el argumento R1, e imprime el resultado.

Imprime el valor de R2 y llama a la función "dominio" con el argumento R2, e imprime

9. Define una función llamada "codominio" que toma un argumento R.

Dentro de la función, se crea un conjunto vacío llamado "cod" para almacenar el codominio.

Se recorren los elementos de la relación R y se agrega el segundo elemento de cada par al conjunto "cod".

Finalmente, se devuelve el conjunto "cod".

Imprime el valor de R9 y llama a la función "codominio" con el argumento R9, e imprime el resultado.

Imprime el valor de R14 y llama a la función "codominio" con el argumento R14, e imprime el resultado.

10. Define una función llamada "funcion_alea" que toma dos argumentos: S y T.

Dentro de la función, se crea un conjunto vacío llamado "func" para almacenar los pares de la función.

Se recorren los elementos del primer conjunto S y se verifica si existen en el segundo conjunto T.

Si un elemento de S existe en T, se agrega el par (i, i) al conjunto "func".

Finalmente, se devuelve el conjunto "func".

Imprime los valores de A1 y A6.

Llama a la función "funcion_alea" con los argumentos A1 y A6, convierte el resultado a una lista y lo imprime.

11. En esta sección se define una función llamada "definida_todas_partes" que toma dos parámetros: A y R.

La función utiliza la función "dominio" en R, pero no se muestra el código de la función "dominio".

Luego, la función compara si el resultado de la función "dominio(R)" es igual a A.

Si el dominio de R es igual a A, la función retorna True; de lo contrario, retorna False.

Se imprime A13, R17, R18 y los resultados de llamar a la función "definida_todas_partes" con A13 y R17, y con A13 y R18.

12. Aquí se define una función llamada "es_uno_a_uno" que toma un parámetro R.

La función crea una lista llamada "val_cod" para guardar los elementos del codominio.

Luego, itera sobre los pares de la relación R.

En cada iteración, guarda el segundo elemento del par en la variable "cod".

Comprueba si el elemento "cod" ya está en la lista "val_cod".

Si el elemento "cod" ya está en la lista, significa que hay una repetición en el codominio y la función retorna False.

Si no se encuentra una repetición, se agrega el elemento "cod" a la lista "val_cod".

Después de recorrer todos los pares de la relación, si no se encontraron repeticiones, la función retorna True.

Se imprime R18, R19 y los resultados de llamar a la función "es_uno_a_uno" con R18 y con R19.

13. Aquí se define una función llamada "es_exhaustiva" que toma dos parámetros: A y R.

La función utiliza la función "codominio" en R, pero no se muestra el código de la función "codominio".

Luego, la función compara si el resultado de la función "codominio(R)" es igual a A.

Si el codominio de R es igual a A, la función retorna True; de lo contrario, retorna False.

Se imprime A15, R20, R21 y los resultados de llamar a la función "es_exhaustiva" con A15 y R20, y con A15 y R21.

14. Aquí se define una función llamada "es_biyectiva" que toma dos parámetros: A y R.

La función asigna el resultado de llamar a la función "es_uno_a_uno" con R a la variable "r1".

Luego, asigna el resultado de llamar a la función "es_exhaustiva" con A y R a la variable "r2".

Si tanto "r1" como "r2" son True, la función retorna True; de lo contrario, retorna False.
Se imprime A, R, R1 y los resultados de llamar a la función "es_biyectiva" con A y R, y con A y R1.

15. Esta sección contiene cinco partes, cada una resolviendo un problema diferente relacionado con funciones.

Parte "a) Función uno-a-uno y no exhaustiva":

Se define una función llamada "uno_a_uno_no_exhaustiva" que toma un parámetro R.
La función crea un conjunto llamado "codominio_set" para guardar los elementos del codominio.
Luego, itera sobre los pares de la relación R.
En cada iteración, se guarda el segundo elemento del par en la variable "y".
Comprueba si el elemento "y" ya está en el conjunto "codominio_set".
Si el elemento "y" ya está en el conjunto, significa que no es una función uno-a-uno y no exhaustiva, por lo que la función retorna False.
Si no se encuentra repetición, se agrega el elemento "y" al conjunto "codominio_set".
Después de recorrer todos los pares de la relación, si no se encontraron repeticiones, la función retorna True.
Se imprime el conjunto R y el resultado de llamar a la función "uno_a_uno_no_exhaustiva" con R.

Parte "b) Función uno-a-uno y no definida en todas partes":

Se define una función llamada "uno_a_uno_no_definida" que toma dos parámetros: A y R.
La función crea un conjunto llamado "dominio_set" para guardar los elementos del dominio.
Luego, itera sobre los pares de la relación R.
En cada iteración, se guarda el primer elemento del par en la variable "x".
Comprueba si el elemento "x" no está en el conjunto "dominio_set".
Si el elemento "x" no está en el conjunto, significa que no es una función definida en todas partes y uno-a-uno, por lo que la función retorna False.
Si se encuentra una repetición en el dominio, se agrega el elemento "x" al conjunto "dominio_set".
Después de recorrer todos los pares de la relación, si el conjunto "dominio_set" no es igual a A, la función retorna True.
Se imprime el conjunto A, el conjunto R y el resultado de llamar a la función "uno_a_uno_no_definida" con A y R.

Parte "c) Función definida en todas partes y no uno-a-uno":

Se define una función llamada "definida_no_uno_a_uno" que toma dos parámetros: A y R.

La función crea un conjunto llamado "codominio_set" para guardar los elementos del codominio.

Luego, itera sobre los pares de la relación R.

En cada iteración, se guarda el segundo elemento del par en la variable "y".

Comprueba si el elemento "y" ya está en el conjunto "codominio_set".

Si el elemento "y" ya está en el conjunto, significa que no es una función uno-a-uno, pero es definida en todas partes, por lo que la función retorna True.

Si no se encuentra repetición en el codominio, se agrega el elemento "y" al conjunto "codominio_set".

Después de recorrer todos los pares de la relación, si no se encontraron repeticiones, la función retorna False.

Se imprime el conjunto R y el resultado de llamar a la función "definida_no_uno_a_uno" con A y R.

Parte "d) Función exhaustiva y no uno-a-uno":

Se define una función llamada "exhaustiva_no_uno_a_uno" que toma dos parámetros: A y R.

La función crea un conjunto llamado "func" para guardar los elementos del codominio.

Luego, itera sobre los pares de la relación R.

En cada iteración, se agrega el segundo elemento del par al conjunto "func".

Después de recorrer todos los pares de la relación, se compara la longitud del conjunto "func" con la longitud de A.

Si son iguales, significa que la función es exhaustiva pero no uno-a-uno, por lo que la función retorna True.

Si no son iguales, significa que la función no es exhaustiva, por lo que la función retorna False.

Se imprime el conjunto R y el resultado de llamar a la función "exhaustiva_no_uno_a_uno" con A y R.

Parte "e) Función biyectiva":

Se define una función llamada "biyectiva" que toma dos parámetros: A y R.

La función crea un conjunto llamado "func" para guardar los elementos del dominio.

Luego, itera sobre los pares de la relación R.

En cada iteración, se verifica si el primer elemento del par no está en el conjunto "func".

Si no está en el conjunto, se agrega al conjunto "func".

Si está en el conjunto, significa que no es una función biyectiva, por lo que la función retorna False.

Después de recorrer todos los pares de la relación, se compara la longitud del conjunto "func" con la longitud de A y la longitud de R.

Si son iguales, significa que la función es biyectiva, por lo que la función retorna True.

Si no son iguales, significa que la función no es biyectiva, por lo que la función retorna False.

Se imprime el conjunto R y el resultado de llamar a la función "biyectiva" con A y R.

P1

lab4_funciones.py > ...

```
1  A1 = {1,2,3,4,5}
2  R1 = {(1,1),(2,2),(3,3), (4,4), (5,5)}
3  R2 = {(1,1), (2,3), (3,3)}
4
5
6  def es_reflexiva(A, R): # Definicion de la funcion
7      for e in A:         # Ciclo for para recorrer los elementos
8          if (e,e) not in R: # Validacion de coincidencia de los pares iguales en la relacion
9              return False  # Se retorna falso si tras el recorrido no todos los pares igual
10         return True # Se retorna true si tras el recorrido todos los pares iguales se encuentran
11
12  print(A1)
13  print(R1)
14  print(es_reflexiva(A1,R1))
15  print(R2)
16  print(es_reflexiva(A1,R2))
17
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
TA/DISCRETAS/LAB4/lab4_funciones.py
{1, 2, 3, 4, 5}
{(4, 4), (5, 5), (3, 3), (2, 2), (1, 1)}
True
{(2, 3), (1, 1), (3, 3)}
False
```

P2

```
26 def es_irreflexiva(A, R): # Definicion de la funcion
27     for e in A:           # Ciclo para recorrer los elementos
28         if (e,e) in R: # Validacion de coincidencia de los pares iguales en la relacion
29             return False # Se retorna false si tras el recorrido se encuentra algun par igual
30     return True # Se retorna True si no se encuentra ningun par igual tras el recorrido
31
32 print(A2)
33 print(R3)
34 print(R4)
35 print(es_irreflexiva(A2,R3))
36 print(es_irreflexiva(A2,R4))
37
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Python Debug Console + - □ ✕ ... ^ ✕

```
userlub@userlub-pc:/media/userlub/DATA/DISCRETAS/LAB4$ /usr/bin/env /bin/python3 /home/userlub/.vscode/extensions/ms-python.
python-2023.10.1/pythonFiles/lib/python/debugpy/adapters/../../debugpy/launcher 38583 -- /media/userlub/DATA/DISCRETAS/LAB4/la
b4_funciones.py
{'a', 'b', 'c'}
{('a', 'b'), ('c', 'a')}
{('a', 'b'), ('b', 'c'), ('a', 'a')}
True
False
```


P3

lab4_funciones.py > ...

```
45 def es_simetrica(A, R): # Definicion de la funcion
46     for i, j in R: # Ciclo para recorrer los elementos
47         if (j,i) not in R: # Validacion de coincidencia del par inverso
48             return False # Se retorna false si no se encuentra el par inverso
49     return True # Se retorna true si tras terminar el recorrido se encuentran los pares inversos.
50
51 print(A3)
52 print(R5)
53 print(R6) |
54
55 print(es_simetrica(A3,R5))
56 print(es_simetrica(A3,R6))
57
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Python Debug Console + - □ ☒ ... ^ ×

```
userlub@userlub-pc:/media/userlub/DATA/DISCRETAS/LAB4$ /usr/bin/env /bin/python3 /home/userlub/.vscode/extensions/ms-python.
python-2023.10.1/pythonFiles/lib/python/debugpy/adapter/../../debugpy/launcher 50147 -- /media/userlub/DATA/DISCRETAS/LAB4/la
b4_funciones.py
{'w', 'u', 'v'}
{('u', 'v'), ('v', 'u'), ('w', 'u'), ('u', 'w')}
{('x', 'x'), ('x', 'v'), ('u', 'x')}
True
False
```

P4

```
67 def es_asimetrica(A, R): # Definicion de la funcion
68     for i, j in R: # Ciclo para recorrer los elementos
69         if (j,i) in R: # Validacion de coincidencia del par inverso
70             return False # Se retorna false si se encuentra un par inverso
71     return True # Se retorna true si tras el recorrido ningun par inverso fue encontrado
72
73
74 print(A4)
75 print(R7)
76 print(R8)
77 print(es_asimetrica(A4,R7))
78 print(es_asimetrica(A4,R8))
79
80 def es_antisimetrica(A, R):
81     pass
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Python Debug Console + - □ □ ... ^ ×

```
python-2023.10.1/pythonFiles/lib/python/debugpy/adapter/../../debugpy/launcher 50689 -- /media/userlub/DATA/DISCRETAS/LAB4/lab4_funciones.py
{1, 2, 3, 4}
{(3, 1), (1, 2), (3, 4)}
{(1, 2), (3, 4), (4, 3)}
True
False
```

P5

```
126 def es_antisimetrica(A, R): # Definicion de la funcion
127     for i,j in R:           # Ciclo para recorrer los elementos
128         if i!=j and ((i,j) in R and (j,i) not in R): # Validacion de elementos distintos y el par inverso
129             return True      # Se retorna true si para elementos distintos no se encuentra el inverso
130         if i==j and (i,j) in R: # Se retorna true si los elementos de los pares son iguales
131             return True
132     return False             # Se retorna false si se encuentra par inverso tras el recorrido
133
134
135 print(A5)
136 print(R8)
137 print(R9)
138 print(es_antisimetrica(A5,R8))
139 print(es_antisimetrica(A6,R9))
140
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Python Debug Console + -

```
python-2023.10.1/pythonFiles/lib/python/debugpy/adapter/../../debugpy/launcher 60145 -- /media/userlub/DATA/DISCR
b4_funciones.py
{1, 2, 3, 4}
{(2, 3), (3, 2), (4, 1), (1, 4)}
{(2, 3), (1, 2)}
False
True
```

P6

```
111 def es_transitiva(A, R): #Definicion de la funcion
112     for a, b in R: # Recorrido de elementos en la relacion
113         for c in A:
114             if (b, c) in R: # Validacion de transitividad
115                 if (a, c) not in R:
116                     return False # Retorno de false tras no encontrar el par que completa la transitividad
117         return True # Retorno de true tras los recorridos y encontrar el par que cumple la transitividad
118
119 print(A7)
120 print(R10)
121 print(R11)
122
123 print(es_transitiva(A7,R10))
124 print(es_transitiva(A7,R11))
125
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Python Debug Console + - □

```
python-2023.10.1/pythonFiles/lib/python/debugpy/adapter/../../debugpy/launcher 34493 -- /media/userlub/DATA/DISCRE
b4_funciones.py
{1, 2, 3}
{(2, 3), (1, 2), (1, 3)}
{(2, 3), (1, 2), (1, 3), (3, 1)}
True
False
```

P7a)

```
135 # 7a) Validacion de simetria y no transitividad
136
137
138 print(A8)
139 print(R12)
140 print(es_simetrica(A8,R12))
141 print(es_transitiva(A8,R12))
142
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

```
userlub@userlub-pc:/media/userlub/DATA/DISCRETAS/LAB4$ /usr/bin/env /bin
python-2023.10.1/pythonFiles/lib/python/debugpy/adapter/../../debugpy/la
b4_funciones.py
{'c', 'b', 'd', 'a'}
{('c', 'd'), ('d', 'c'), ('a', 'b'), ('b', 'a')}
True
False
```

P7b)

```
149 # 7b) Validacion de antisimetria y transitividad
150
151 print(A9)
152 print(R13)
153 print(es_antisimetrica(A9,R13))
154 print(es_transitiva(A9,R13))
155
156
157
158 def dominio(R):
159     pass
160
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
b4_funciones.py
{'x', 'w', 'y'}
{('x', 'y'), ('w', 'y'), ('w', 'x')}
True
True
```

P7c)

```
163 # 7c) Validacion de irreflexibilidad y no transitividad
164
165 print(A10)
166 print(R14)
167 print(es_irreflexiva(A10,R14))
168 print(es_transitiva(A10,R14))
169
170
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
b4_funciones.py
{1, 2, 3, 4, 5}
{(3, 1), (2, 4), (5, 2), (4, 3)}
True
False
```

P7d)

```
177 # 7d) Validacion de reflexibilidad y no simetria
178
179 print(A11)
180 print(R15)
181 print(es_reflexiva(A11,R15))
182 print(es_simetrica(A11,R15))
183
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
b4_funciones.py
{'f', 'e', 'd'}
{('f', 'd'), ('e', 'd'), ('e', 'e'), ('d', 'd'), ('f', 'f')}
True
False
```


P7e)

```
193 # 7e) Validacion de simetria y antisimetria
194
195 print(A12)
196 print(R16)
197 print(es_simetrica(A12, R16))
198 print(es_antisimetria(A12, R16))
199
200
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

```
userlub@userlub-pc:/media/userlub/DATA/DISCRETAS/LAB4$ /u
python-2023.10.1/pythonFiles/lib/python/debugpy/adapter/..
b4_funciones.py
{1, 2, 3}
{(1, 1), (3, 3), (2, 2)}
True
True
```

P8)

```
203  ✓ def dominio(R): # Definicion de la funcion
204      dom = set() # Creacion del conjunto dom para guardar el dominio
205  ✓   for i in R: # Recorrido de los elementos de la relacion
206      |       dom.add(i[0]) # Agregado del primer elemento del par de la relacion al conjunto dom
207
208      return dom
209
210
211  print (R1)
212  print(dominio(R1))
213
214  print(R2)
215  print(dominio(R2))
216
217  ✓ def codominio(R):
218      |     pass
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Python Debug

```
python-2023.10.1/pythonFiles/lib/python/debugpy/adapter/../../debugpy/launcher 48787 -- /media/userl
b4_funciones.py
{(4, 4), (5, 5), (3, 3), (2, 2), (1, 1)}
{1, 2, 3, 4, 5}
{(2, 3), (1, 1), (3, 3)}
{1, 2, 3}
```

P9)

```
203  ✓ def dominio(R): # Definicion de la funcion
204      dom = set() # Creacion del conjunto dom para guardar el dominio
205  ✓   for i in R: # Recorrido de los elementos de la relacion
206      |       dom.add(i[0]) # Agregado del primer elemento del par de la relacion al conjunto dom
207
208      return dom
209
210
211  print (R1)
212  print(dominio(R1))
213
214  print(R2)
215  print(dominio(R2))
216
217  ✓ def codominio(R):
218      |     pass
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL


Python Debu

```
python-2023.10.1/pythonFiles/lib/python/debugpy/adapter/../../debugpy/launcher 48787 -- /media/userl
b4_funciones.py
{(4, 4), (5, 5), (3, 3), (2, 2), (1, 1)}
{1, 2, 3, 4, 5}
{(2, 3), (1, 1), (3, 3)}
{1, 2, 3}
```

P10)

```
231 def funcion_alea(S, T):
232
233     func = set() # Se crea un conjunto para guardar los pares de la funcion
234
235     for i in S: # Se recorren los elementos del primer conjunto
236         if i in T: # Validacion si el elemento del primer conjunto existe en el segundo conjunto
237             func.add((i,i)) # Se agrega el elemento a la funcion
238
239     return func
240
241 print(A1)
242 print(A6)
243
244 print(list(funcion_alea(A1,A6)))
245
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

 Python Debug Cons

```
userlub@userlub-pc:/media/userlub/DATA/DISCRETAS/LAB4$ /usr/bin/env /bin/python3 /home/userlub/.vscode/
python-2023.10.1/pythonFiles/lib/python/debugpy/adapter/../../debugpy/launcher 60781 -- /media/userlub/D
b4_funciones.py
{1, 2, 3, 4, 5}
{1, 2, 3}
[(1, 1), (3, 3), (2, 2)]
userlub@userlub-pc:/media/userlub/DATA/DISCRETAS/LAB4$
```

P11)

```
254 def definida_todas_partes(A, R):
255     dominio(R) # Uso de la funcion dominio en R
256
257     if(dominio(R)==A): # Se valida si el dominio de R es igual al conjunto
258         return True # Se retorna true es caso afirmativo
259     else:
260         return False # Se retorna false en caso negativo
261
262 print(A13)
263 print(R17)
264 print(R18)
265 print(definida_todas_partes(A13,R17))
266 print(definida_todas_partes(A13,R18))
267
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
{8, 9, 10, 11, 12}
{(8, 9), (11, 11), (9, 9)}
{(11, 9), (10, 9), (12, 10), (9, 12), (8, 10)}
False
True
```

P12)

```

def es_uno_a_uno(R):
    val_cod = [] # Lista para guardar los elementos del codominio

    for i in R: # Recorrido de los pares de la relacion
        cod = i[1] # Indicacion de que el codominio esta en la posicion 1 de cada par

        if cod in val_cod: # Validacion de existencia del elemento del codominio en la lista
            return False # De existir se retorna false ya que significa que estaria
                           # repitiendose en la relacion
        else:
            val_cod.append(cod) # En caso contrario se agrega el elemento del codominio a la lista

    return True # Se retorna true tras termino de recorrido y no encontrar repeticiones
                # de elementos en el codominio

```

```

290 print(R18)
291 print(R19)
292 print(es_uno_a_uno(R18))
293 print(es_uno_a_uno(R19))
294

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```


userlub@userlub-pc:/media/userlub/DATA/DISCRETAS/LAB4$
python-2023.10.1/pythonFiles/lib/python/debugpy/adapter/
b4_funciones.py
{(4, 5), (5, 6), (3, 4)}
{(3, 3), (5, 6), (4, 3), (6, 6), (5, 7)}
True
False

```

P13)


```
306 v def es_exhaustiva(A, R):  
307     codominio(R)      # Uso de la funcion codominio en R  
308 v     if(codominio(R)==A): # Validacion si el codominio de R es igual a los elementos del conjunto  
309         return True # Se retorna true en caso positivo  
310 v     else:  
311         return False # Se retorna false en caso negativo  
312  
313     print(A15)  
314     print(R20)  
315     print(R21)  
316  
317     print(es_exhaustiva(A15,R20))  
318     print(es_exhaustiva(A15,R21))  
319
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

 Python Debug Console

```
python-2023.10.1/pythonFiles/lib/python/debugpy/adapter/../../debugpy/launcher 57277 -- /media/userlub/D  
b4_funciones.py  
{1, 4, 5}  
{(4, 4), (1, 1), (5, 5)}  
{(5, 1), (1, 4)}  
True  
False
```

P14

ión Ver Ir Ejecutar ... lab 4

lab4_funciones.py C:\...\Downloads lab4_funciones.py X ejem_transitiva.py

lab4_funciones.py > ...

```
365
366     # Problemas # 14
367
368     def es_biyectiva(A, R):
369         r1 = es_uno_a_uno(R)
370         r2 = es_exhaustiva(A, R)
371
372         if r1 and r2:
373             return True
374         else:
375             return False
376
377     print(A)
378     print(R)
379     print(R1)
380     print(es_biyectiva(A, R)) # Salida: False
381     print(es_biyectiva(A, R1)) # Salida: False
382
383
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN **TERMINAL**

PROBLEMAS # 14

{1, 2, 3}

{{(2, 3), (1, 2), (1, 3)}}

{{(2, 3), (1, 2), (1, 3), (3, 1)}}

False

False

PROBLEMAS #15

a) Función uno-a-uno y no exhaustiva:

Conjunto R: [(1, 2), (2, 3), (3, 4)]

Resultado: True

Lín. 416, col. 1 Espacios: 4

P15

ArchivoEditarSelecciónVerIrEjecutarlab 4

EXPLORADORLAB 4ejem_antisimetria.pyejem_transitiva.pylab4_funciones.py

lab4_funciones.py > ...

```
384 print("\n")
385 print("PROBLEMAS #15")
386
387 print(" a) Función uno-a-uno y no exhaustiva:")
388
389 def uno_a_uno_no_exhaustiva(R):
390     codominio_set = set()
391     for _, y in R:
392         if y in codominio_set:
393             return False
394         codominio_set.add(y)
395     return True
396
397 R = [(1, 2), (2, 3), (3, 4)] # Ejemplo de conjunto R
398 print("Conjunto R:", R)
399 print("Resultado:", uno_a_uno_no_exhaustiva(R)) # Salida: True
400
401 print("\n")
402 print(" b) Función uno-a-uno y no definida en todas partes:")
403
404 def uno_a_uno_no_definida(A, R):
405     dominio_set = set()
406     for x, _ in R:
407         if x not in dominio_set:
408             dominio_set.add(x)
409     return dominio_set != A
410
411 A = {1, 2, 3} # Ejemplo de conjunto A
412 R = [(1, 2), (2, 3), (4, 5)] # Ejemplo de conjunto R
413 print("Conjunto A:", A)
414 print("Conjunto R:", R)
415 print("Resultado:", uno_a_uno_no_definida(A, R)) # Salida: False
```

0 0 0

Lín. 416, col. 1Espacios: 4UTF-8LFPython 3.11.4 64-bit

ArchivoEditarSelecciónVerIrEjecutarlab 4

EXPLORADOR

LAB 4

ejem_antisimetria.py

ejem_transitiva.py

lab4_funciones.py

ESQUEMA

LÍNEA DE TIEMPO

lab4_funciones.py C:\...\Downloads

lab4_funciones.py

ejem_transitiva.py

lab4_funciones.py > ...

```
415 print("Resultado:", uno_a_uno_no_definida(A, R)) # Salida: False
416
417 print("\n")
418 print("c) Función definida en todas partes y no uno-a-uno:")
419
420 def definida_no_uno_a_uno(A, R):
421     codominio_set = set()
422     for _, y in R:
423         if y in codominio_set:
424             return True
425         codominio_set.add(y)
426     return False
427
428 R = [(1, 2), (2, 3), (2, 4)] # Ejemplo de conjunto R
429 print("Conjunto R:", R)
430 print("Resultado:", definida_no_uno_a_uno(A, R)) # Salida: False
431
432 print("\n")
433 print("d) Función exhaustiva y no uno-a-uno:")
434
435 def exhaustiva_no_uno_a_uno(A, R):
436     func = set()
437     for i in R:
438         func.add(i[1])
439     return len(func) == len(A)
440
441 R = [(1, 2), (2, 3), (3, 4)] # Ejemplo de conjunto R
442 print("Conjunto R:", R)
443 print("Resultado:", exhaustiva_no_uno_a_uno(A, R))
444
445 print("\n")
446 print("Función biyectiva:")
```

Lín. 416, col. 1

Espacios: 4

UTF-8

LF

Python

3.11.4 64-bit

ArchivoEditarSelecciónVerIrEjecutarlab 4

EXPLORADOR

LAB 4

ejem_antisimetria.py
ejem_transitiva.py
lab4_funciones.py

ESQUEMA
LÍNEA DE TIEMPO

lab4_funciones.py C:\...Downloadslab4_funciones.py

```
lab4_funciones.py > ...  
429 print("Conjunto R:", R)  
430 print("Resultado:", definida_no_uno_a_uno(A, R)) # Salida: False  
431  
432 print("\n")  
433 print("d) Función exhaustiva y no uno-a-uno:")  
434  
435 def exhaustiva_no_uno_a_uno(A, R):  
436     func = set()  
437     for i in R:  
438         func.add(i[1])  
439     return len(func) == len(A)  
440  
441 R = [(1, 2), (2, 3), (3, 4)] # Ejemplo de conjunto R  
442 print("Conjunto R:", R)  
443 print("Resultado:", exhaustiva_no_uno_a_uno(A, R))  
444  
445 print("\n")  
446 print("e) Función biyectiva:")  
447  
448 def biyectiva(A, R):  
449     func = set()  
450     for i in R:  
451         if i[0] not in func:  
452             func.add(i[0])  
453         else:  
454             return False  
455     return len(func) == len(A) and len(R) == len(A)  
456  
457 R = [(1, 2), (2, 3), (3, 4)] # Ejemplo de conjunto R  
458 print("Conjunto R:", R)  
459 print("Resultado:", biyectiva(A, R))  
460
```

Lín. 432, col. 12 Espacios: 4 UTF-8 LF Python 3.11.4 64-bit

ArchivoEditarSelecciónVerIrEjecutar...lab 4

EXPLORADOR...lab4_funciones.py C:\...\Downloadslab4_funciones.py \ X

LAB 4ejem_antisimetria.pyejem_transitiva.py lab4_funciones.py

> ESQUEMA> LÍNEA DE TIEMPO

lab4_funciones.py > biyectiva

362262 print("\n")

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓNTERMINAL

```
{(2, 3), (1, 2), (1, 3), (3, 1)}
False
False

PROBLEMAS #15
a) Función uno-a-uno y no exhaustiva:
Conjunto R: [(1, 2), (2, 3), (3, 4)]
Resultado: True

b) Función uno-a-uno y no definida en todas partes:
Conjunto A: {1, 2, 3}
Conjunto R: [(1, 2), (2, 3), (4, 5)]
Resultado: True

c) Función definida en todas partes y no uno-a-uno:
Conjunto R: [(1, 2), (2, 3), (2, 4)]
Resultado: False

d) Función exhaustiva y no uno-a-uno:
Conjunto R: [(1, 2), (2, 3), (3, 4)]
Resultado: True

e) Función biyectiva:
Conjunto R: [(1, 2), (2, 3), (3, 4)]
Resultado: True
PS C:\Users\USER\Downloads\lab 4>
```

PythonPython

0 0 0Lín. 448, col. 21Espacios: 4UTF-8LFPython 3.11.4 64-bit

Conclusión

En conclusión, una función discreta es una relación entre conjuntos finitos de elementos donde cada elemento del dominio se asigna a un único elemento del codominio. Los elementos clave de una función discreta son el dominio, que es el conjunto de posibles valores de entrada, y el codominio, que es el conjunto de posibles valores de salida. Además, existe una relación que establece la asociación entre los elementos del dominio y los elementos del codominio.