Evaluación de propiedades de relaciones en conjuntos

Materia: Matemáticas Discretas

Alumno: Juan Pablo Palmero Valdés

26 de septiembre de 2025

Objetivo de la actividad

Desarrollar un programa que pueda analizar una relación sobre dos conjuntos iguales y determinar si cumple las propiedades de reflexividad, antirreflexividad, simetría, antisimetría, asimetría y transitividad. El sistema debe indicar si la relación es de equivalencia o de orden parcial y, en cada caso, mostrar la representación gráfica correspondiente.

Índice

1. Introducción

Las relaciones en conjuntos son un tema de gran importancia en matemáticas discretas, pues describen cómo se conectan los elementos entre sí. Analizar sus propiedades permite clasificarlas y entender estructuras como las equivalencias y los órdenes parciales.

En este trabajo se presenta un programa en Python que pueda recibir una relación y verificar automáticamente qué propiedades cumple. Con esto se busca hacer más sencillo el análisis, complementando la parte teórica con una herramienta práctica.

2. Planeación de actividades y lenguaje de programación

Planeación de actividades

1. Revisar las instrucciones y definir claramente los objetivos del programa.
2. Analizar el problema y comprender las propiedades de las relaciones a evaluar.
3. Diseñar el algoritmo y la estructura de clases para organizar la lógica del programa.
4. Implementar el programa en Python, creando funciones para cada propiedad de la relación.
5. Generar las visualizaciones correspondientes: el grafo general y el diagrama de Hasse.
6. Realizar pruebas con distintos tipos de relaciones: equivalencia, orden parcial y ninguna.
7. Redactar el informe incluyendo resultados, discusión, conclusiones y referencias.

Justificación del lenguaje de programación

Python fue elegido porque ofrece una sintaxis clara y sencilla, lo que facilita enfocarse en la lógica del problema. Además, brinda facilidades para manejar listas, conjuntos y diccionarios, que son estructuras necesarias en este tipo de análisis.

Otra ventaja es la disponibilidad de librerías como networkx y matplotlib, que permiten representar gráficamente las relaciones sin necesidad de implementar gráficos desde cero. Esto lo convierte en una opción práctica y eficiente para este proyecto.

3. Análisis del problema

Dada una relación R definida sobre un conjunto A (donde A = B), es posible clasificarla en función de las propiedades que cumpla. El programa debe verificar automáticamente las siguientes propiedades fundamentales:

- Reflexividad: todos los elementos de A están relacionados consigo mismos.
- Antirreflexividad (irreflexividad): ningún elemento de A está relacionado consigo mismo.
- Simetría: si un elemento x está relacionado con y, entonces y también está relacionado con x.
- Antisimetría: si x está relacionado con y y y está relacionado con x, entonces necesariamente x = y.
- Asimetría: si x está relacionado con y, nunca ocurre que y esté relacionado con x.
- Transitividad: si x está relacionado con y y y con z, entonces x también debe estar relacionado con z.

Con base en estas propiedades, se puede determinar la clasificación de la relación:

- Relación de equivalencia: cuando es reflexiva, simétrica y transitiva. En este caso, los elementos se agrupan en clases de equivalencia, formando particiones del conjunto.
- Orden parcial: cuando es reflexiva, antisimétrica y transitiva. En este caso, los elementos se pueden organizar en un diagrama de Hasse que refleja la jerarquía entre ellos.
- Ninguna de las anteriores: si la relación no cumple con las condiciones de equivalencia ni de orden parcial, simplemente se reporta con las propiedades que sí cumple, pero no se clasifica en ninguna de estas dos categorías.

En resumen, el problema consiste en diseñar un sistema capaz de recibir cualquier relación sobre un conjunto, analizar sus propiedades y determinar de manera clara si se trata de una relación de equivalencia, un orden parcial o una relación sin clasificación específica.

4. Diseño del algoritmo

El algoritmo se diseñó de manera modular, dividiendo las responsabilidades en pasos bien definidos y encapsulando la lógica principal dentro de clases. El flujo general del programa es el siguiente:

1. Entrada de datos

- El usuario introduce el conjunto de nodos A (que es igual a B) y la relación R como una lista de pares ordenados.
- Si no se especifican los nodos explícitamente, el programa los infiere a partir de los pares dados.
- Se valida la entrada para asegurar que los datos tengan el formato correcto.

2. Construcción de la relación

- Se crea una instancia de la clase Relation, que almacena los nodos y pares.
- Esta clase concentra toda la lógica necesaria para analizar la relación.

3. Verificación de propiedades

- Usando los métodos de Relation, el programa recorre los pares y determina si la relación es:
- Reflexiva o antirreflexiva.
- Simétrica, antisimétrica o asimétrica.
- Transitiva.
- En cada caso, además de un resultado booleano, se listan los pares o caminos que cumplen las propiedades (por ejemplo, pares reflexivos o caminos transitivos).

4. Clasificación de la relación

- Si la relación resulta reflexiva, simétrica y transitiva, se clasifica como relación de equivalencia y se generan las clases de equivalencia.
- Si es reflexiva, antisimétrica y transitiva, se clasifica como orden parcial.
- En cualquier otro caso, se concluye que la relación no pertenece a estas dos categorías.
- 5. Generación de representaciones gráficas
- Para cualquier relación, se puede generar su grafo dirigido usando networkx.

- Si la relación es de equivalencia, se muestran también sus clases y particiones.
- Si es un orden parcial, se construye un diagrama de Hasse usando la clase HasseDiagram, eliminando pares redundantes y organizando los nodos por niveles.

6. Salida de resultados

- Se imprime de manera clara y estructurada:
    - La lista de pares de la relación.
    - La matriz de la relación en formato de tabla (con pandas).
    - El cumplimiento de cada propiedad, con detalles adicionales cuando aplica.
    - La clasificación de la relación (equivalencia, orden parcial o ninguna).
- Finalmente, se muestran las gráficas correspondientes en pantalla.

5. Implementación del programa

El programa se organizó en varias clases y funciones. Las clases se usan para encapsular información y responsabilidades. Este enfoque modular hace que el código sea más fácil de organizar, leer y mantener.

5.1 Clase Relation

La clase Relation encapsula todo lo relacionado con la relación matemática sobre un conjunto de elementos. En otras palabras, aquí guardamos los nodos y los pares de la relación y decidimos “qué propiedades cumple”. Sus responsabilidades principales son:

- Almacenar los datos: guarda los nodos y los pares (x, y) que definen la relación.
- Verificar propiedades: tiene métodos como is_reflexive(), is_symmetric(), is_antisymmetric(), is_transitive(), etc., que recorren los pares y devuelven True o False según las definiciones matemáticas.
- Detalles de propiedades: métodos como get_reflexive_pairs() o get_transitive_paths() permiten inspeccionar qué pares cumplen con cada propiedad, útil para debug y visualización.
- Clasificación automática: is_equivalence() y is_partial_order() determinan si la relación es de equivalencia o un orden parcial, combinando las propiedades necesarias.
- Matriz de relación: get_relation_matrix() construye la matriz binaria de la relación, útil para visualizar la relación de manera tabular con pandas.

En resumen, Relation es la “caja negra” que sabe todo sobre la relación y sus propiedades.

5.2 Clase HasseDiagram

- La clase HasseDiagram se encarga de representar órdenes parciales de forma gráfica usando un diagrama de Hasse. La lógica es:
- Filtrado de pares redundantes: elimina pares que no se deben mostrar para simplificar el diagrama.

- Cálculo de niveles: determina la altura de cada nodo para que los elementos mínimos queden abajo y los máximos arriba.
- Preparación del grafo: genera la estructura que luego se dibuja con networkx.

De esta manera, la clase separa la lógica matemática del diagrama de la visualización, manteniendo el código limpio y modular.

5.3 Funciones de visualización

- plot_graph(): dibuja cualquier relación como grafo dirigido, mostrando todos los nodos y aristas.
- plot_hasse_diagram(): dibuja el diagrama de Hasse si la relación es un orden parcial.
- show_properties(): es la función “todo en uno” que:
- Imprime los pares de la relación.
- Muestra qué pares cumplen reflexividad, simetría y transitividad.
- Genera la matriz de relación con pandas.
- Clasifica la relación como equivalencia, orden parcial o ninguna.
- Llama a las funciones de visualización correspondientes.

5.4 Función main()

main() se encarga de la interacción con el usuario:

1. Pide los nodos del conjunto y los pares de la relación.

2. Crea una instancia de Relation.

3. Llama a show_properties() para analizar y mostrar todo.

4. Gestiona errores de entrada para asegurar que los datos sean válidos y consistentes.

5.5 Diseño por clases: ventajas prácticas

- Encapsulación: cada clase tiene un solo trabajo (Relation analiza propiedades, HasseDiagram genera diagramas), evitando mezclar lógica de negocio con visualización.
- Reutilización: se pueden crear nuevas relaciones y diagramas sin tocar la lógica interna.
- Mantenimiento: es fácil agregar nuevas propiedades o gráficos porque la modularidad separa responsabilidades.
- Claridad para programadores: leer el código es casi como seguir un flujo narrativo: primero definimos la relación, luego analizamos, clasificamos y finalmente visualizamos.

6. Pruebas y resultados

Relación 1: orden parcial:

Conjuntos A = B = {2, 4, 5, 10, 12, 20, 25}

Relación R = {(2, 2), (2, 4), (2, 10), (2, 12), (2, 20), (4, 4), (4, 12), (4, 20), (5, 5), (5, 10), (5, 20), (5, 25), (10, 10), (10, 20), (12, 12), (20, 20), (25, 25)}

Historial de consola:

```
Introduce el conjunto de nodos, separados por comas. Ejemplo: 1,2,3,4
Nodos: 2, 4, 5, 10, 12, 20, 25

Introduce los pares de la relación como una lista de tuplas. Ejemplo: [(1,1), (2,1), (3,2)]
Pares: [(2, 2), (2, 4), (2, 10), (2, 12), (2, 20), (4, 4), (4, 12), (4, 20), (5, 5), (5, 10), (5, 20), (5, 25), (10, 10), (10, 20), (12,
 12), (20, 20), (25, 25)]
||||||||||||||||||||||||||||||||
------  Relación introducida  ------
- Relation pairs:  [(2, 2), (2, 4), (2, 10), (2, 12), (2, 20), (4, 4), (4, 12), (4, 20), (5, 5), (5, 10), (5, 20), (5, 25), (10, 10), (1
0, 20), (12, 12), (20, 20), (25, 25)]
- Is reflexive:  True
    - Pairs:  [(2, 2), (4, 4), (5, 5), (10, 10), (12, 12), (20, 20), (25, 25)]
- Is symmetric:  False
- Is transitive:  True
    - Paths:  [((2, 4), (4, 12), (2, 12)), ((2, 4), (4, 20), (2, 20)), ((2, 10), (10, 20), (2, 20)), ((5, 10), (10, 20), (5, 20))]
- Is asymmetric:  False
- Is antisymmetric:  True

Relation matrix:
    2   4   5   10  12  20  25
2    1   1   0   1   1   1   0
4    0   1   0   0   1   1   0
5    0   0   1   1   0   1   1
10   0   0   0   1   0   1   0
12   0   0   0   0   1   0   0
20   0   0   0   0   0   1   0
25   0   0   0   0   0   0   1
- Partial order:  True
- Hasse diagram:  [(2, 4), (2, 10), (4, 12), (4, 20), (5, 10), (5, 25), (10, 20)]
```

Grafo:

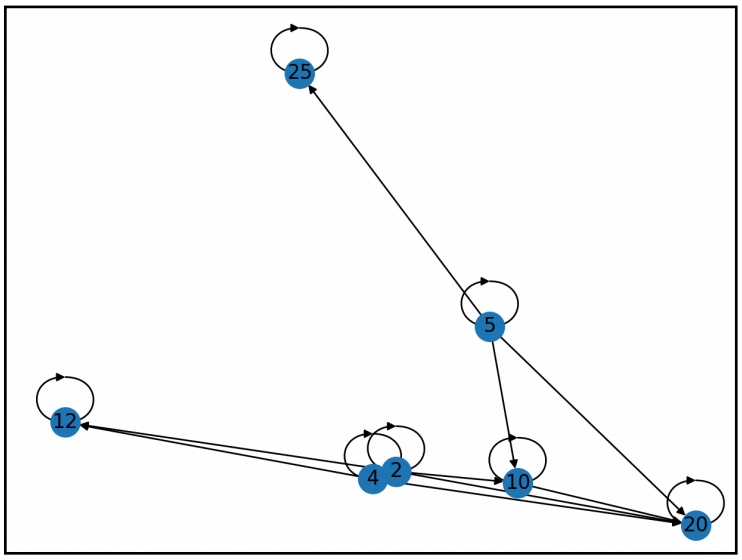
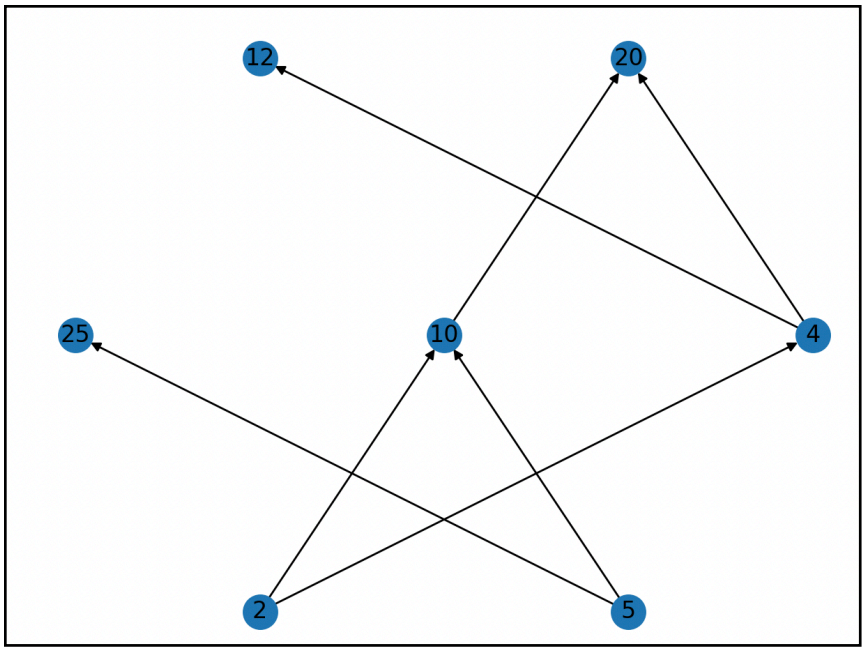


Diagrama de Hasse:


12
20
25
10
4
2
5

Relación 2: orden parcial

Conjunto A = B = {1,2,3,4,5,6}

Relación R = {(1,1),(1,2),(1,3),(1,4),(1,5),(1,6),(2,2),(2,3),(2,4),(2,5),(2,6),(3,3),(3,4),(3,5),(3,6),(4,4),(4,5),(4,6),(5,5),(5,6),(6,6)}

Historial de consola:

```
Introduce el conjunto de nodos, separados por comas. Ejemplo: 1,2,3,4
Nodos: 1, 2, 3, 4, 5, 6

Introduce los pares de la relación como una lista de tuplas. Ejemplo: [(1,1), (2,1), (3,2)]
Pares: [(1,1),(1,2),(1,3),(1,4),(1,5),(1,6),(2,2),(2,3),(2,4),(2,5),(2,6),(3,3),(3,4),(3,5),(3,6),(4,4),(4,5),(4,6),(5,5),(5,6),(6,6)]
||||||||||||||||||||||||||||||||
------  Relación introducida  ------
- Relation pairs:  [(1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (1, 6), (2, 2), (2, 3), (2, 4), (2, 5), (2, 6), (3, 3), (3, 4), (3, 5), (3,
6), (4, 4), (4, 5), (4, 6), (5, 5), (5, 6), (6, 6)]
- Is reflexive:  True
    - Pairs:  [(1, 1), (2, 2), (3, 3), (4, 4), (5, 5), (6, 6)]
- Is symmetric:  False
- Is transitive:  True
    - Paths:  [((1, 2), (2, 3), (1, 3)), ((1, 2), (2, 4), (1, 4)), ((1, 2), (2, 5), (1, 5)), ((1, 2), (2, 6), (1, 6)), ((1, 3), (3, 4),
(1, 4)), ((1, 3), (3, 5), (1, 5)), ((1, 3), (3, 6), (1, 6)), ((1, 4), (4, 5), (1, 5)), ((1, 4), (4, 6), (1, 6)), ((1, 5), (5, 6), (1, 6)
), ((2, 3), (3, 4), (2, 4)), ((2, 3), (3, 5), (2, 5)), ((2, 3), (3, 6), (2, 6)), ((2, 4), (4, 5), (2, 5)), ((2, 4), (4, 6), (2, 6)), ((2
, 5), (5, 6), (2, 6)), ((3, 4), (4, 5), (3, 5)), ((3, 4), (4, 6), (3, 6)), ((3, 5), (5, 6), (3, 6)), ((4, 5), (5, 6), (4, 6))]
- Is asymmetric:  False
- Is antisymmetric:  True

Relation matrix:
   1  2  3  4  5  6
1  1  1  1  1  1  1
2  0  1  1  1  1  1
3  0  0  1  1  1  1
4  0  0  0  1  1  1
5  0  0  0  0  1  1
6  0  0  0  0  0  1
- Partial order:  True
- Hasse diagram:  [(1, 2), (2, 3), (3, 4), (4, 5), (5, 6)]
||||||||||||||||||||||||||||||||
```

Grafo:

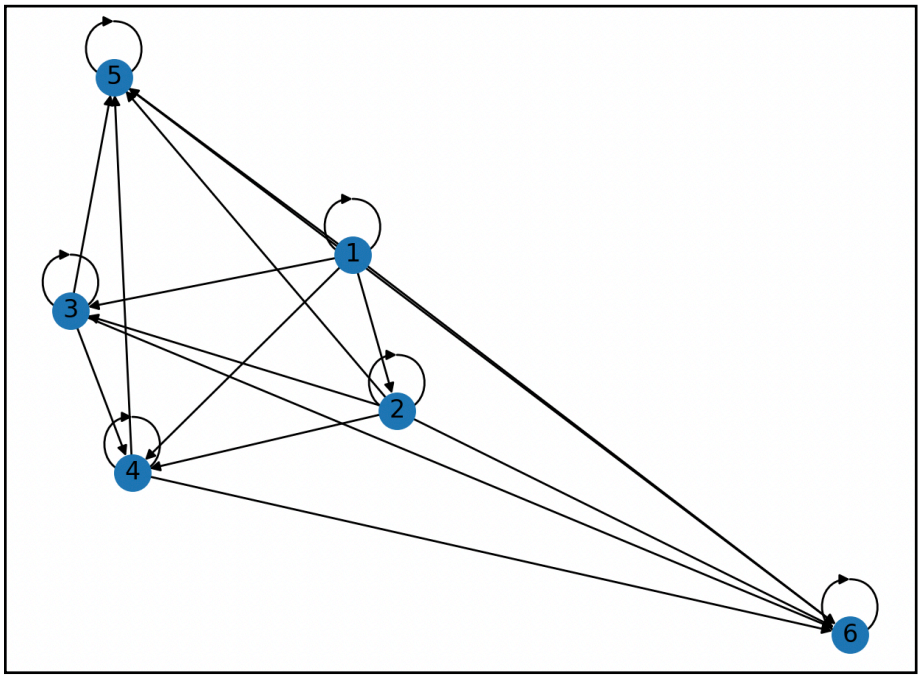
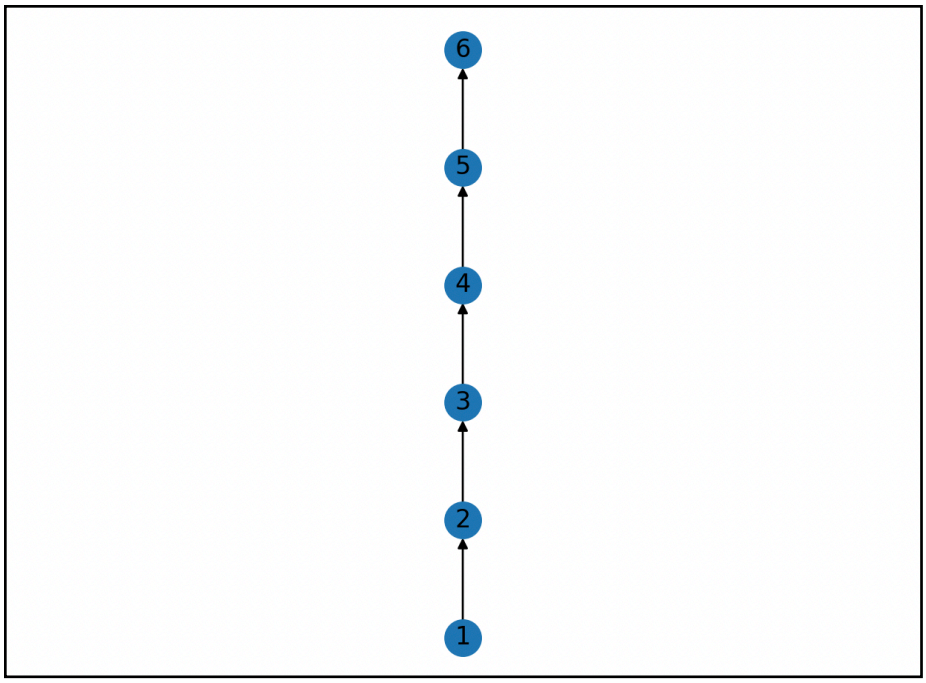


Diagrama de Hasse


6
5
4
3
2
1

Relación 3: orden parcial

Conjunto A = B = {1,2,3,4,5,6,7}

Relación R = {(1,1),(2,2),(3,3),(4,4),(5,5),(6,6),(7,7),(1,2),(1,3),(1,4),(1,6),(1,7),(2,4),(2,6),(3,6)}

Historial de consola:

```
Introduce el conjunto de nodos, separados por comas. Ejemplo: 1,2,3,4
Nodos: 1,2,3,4,5,6,7

Introduce los pares de la relación como una lista de tuplas. Ejemplo: [(1,1), (2,1), (3,2)]
Pares: [(1,1),(2,2),(3,3),(4,4),(5,5),(6,6),(7,7),(1,2),(1,3),(1,4),(1,6),(1,7),(2,4),(2,6),(3,6)]
||||||||||||||||||||||||||||||||
------  Relación introducida  ------
- Relation pairs:  [(1, 1), (2, 2), (3, 3), (4, 4), (5, 5), (6, 6), (7, 7), (1, 2), (1, 3), (1, 4), (1, 6), (1, 7), (2, 4), (2, 6), (3,
6)]
- Is reflexive:  True
   - Pairs:  [(1, 1), (2, 2), (3, 3), (4, 4), (5, 5), (6, 6), (7, 7)]
- Is symmetric:  False
- Is transitive:  True
   - Paths:  [((1, 2), (2, 4), (1, 4)), ((1, 2), (2, 6), (1, 6)), ((1, 3), (3, 6), (1, 6))]
- Is asymmetric:  False
- Is antisymmetric:  True

Relation matrix:
   1  2  3  4  5  6  7
1  1  1  1  1  0  1  1
2  0  1  0  1  0  1  0
3  0  0  1  0  0  1  0
4  0  0  0  1  0  0  0
5  0  0  0  0  1  0  0
6  0  0  0  0  0  1  0
7  0  0  0  0  0  0  1
- Partial order:  True
- Hasse diagram:  [(1, 2), (1, 3), (1, 7), (2, 4), (2, 6), (3, 6)]
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

Grafo:

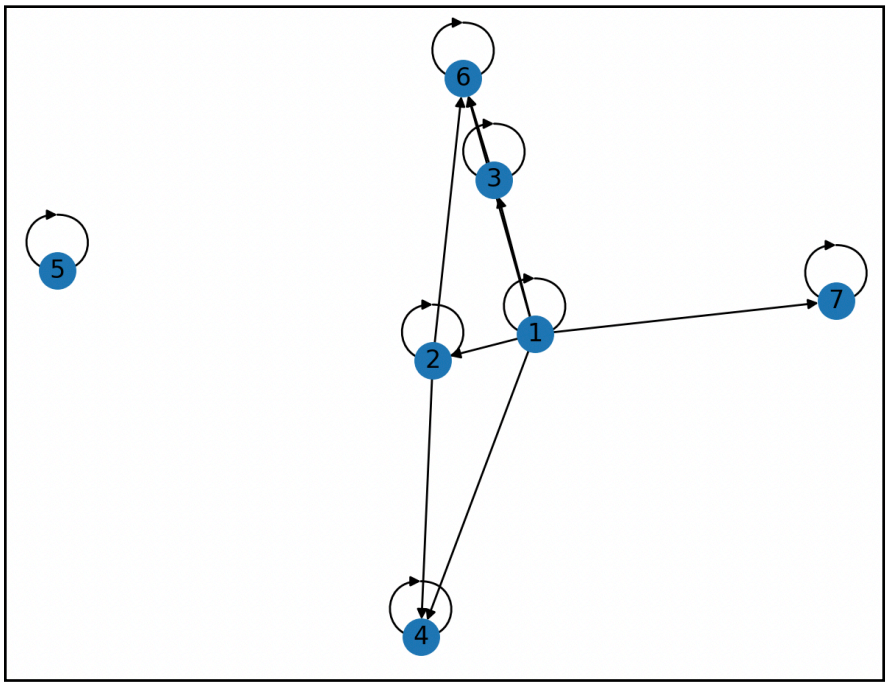
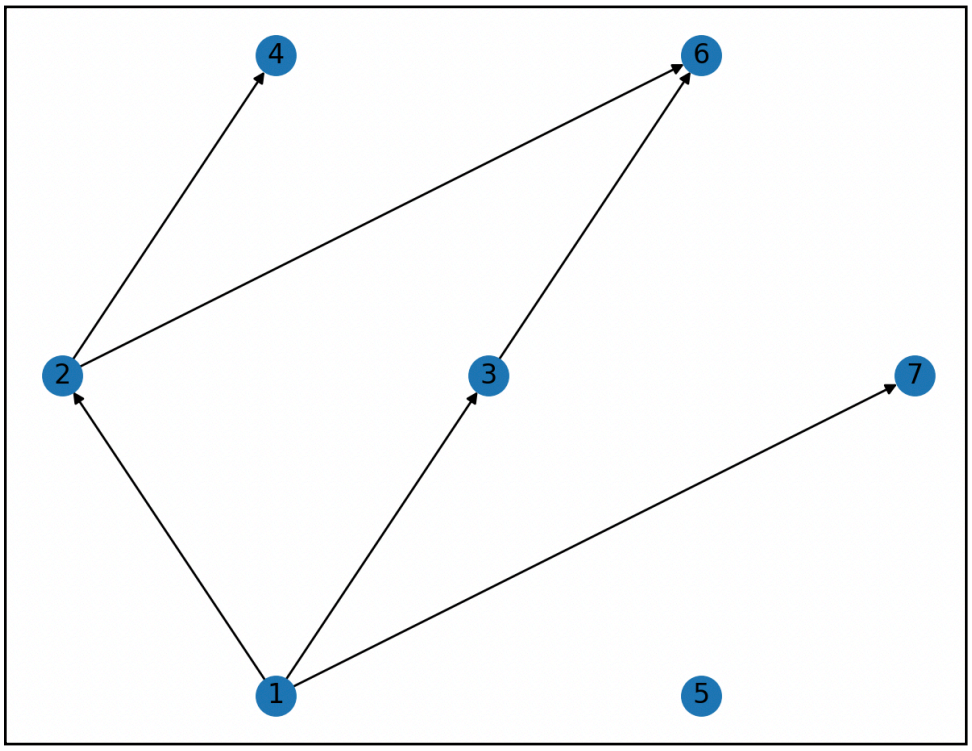


Diagrama de Hasse:


4
6
2
3
7
1
5

Relación 4: equivalencia:

Conjunto A = B = {1,2,3,4,5,6,7,8,9,10,11}

Relación R = {(1,1),(1,2),(1,3),(2,1),(2,2),(2,3),(3,1),(3,2),(3,3), (4,4),(4,5),(4,6), (4,7),(5,4),(5,5),(5,6),(5,7), (6,4),(6,5),(6,6),(6,7),(7,4),(7,5),(7,6),(7,7), (8,8),(8,9), (8,10),(8,11),(9,8),(9,9),(9,10),(9,11), (10,8),(10,9),(10,10),(10,11),(11,8),(11,9), (11,10),(11,11)}

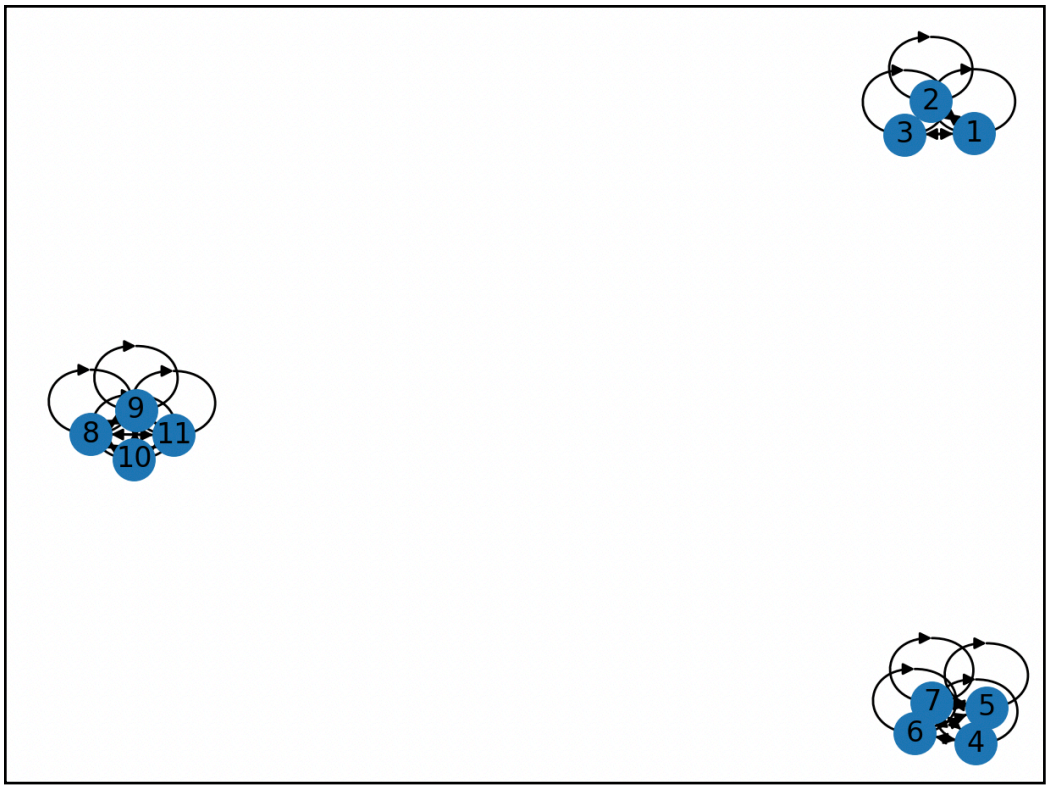
Historial de consola:

```
Introduce el conjunto de nodos, separados por comas. Ejemplo: 1,2,3,4
Nodos: 1,2,3,4,5,6,7,8,9,10,11

Introduce los pares de la relación como una lista de tuplas. Ejemplo: [(1,1), (2,1), (3,2)]
Pares: [(1,1),(1,2),(1,3),(2,1),(2,2),(2,3),(3,1),(3,2),(3,3),(4,4),(4,5),(4,6),(4,7),(5,4),(5,5),(5,6),(5,7),(6,4),(6,5),(6,6),(6,7),(7,4),(7,5),(7,6),(7,7),(8,8),(8,9),(8,10),(8,11),(9,8),(9,9),(9,10),(9,11),(10,8),(10,9),(10,10),(10,11),(11,8),(11,9),(11,10),(11,11)]
||||||||||||||||||||||||||||||||
------  Relación introducida  ------
- Relation pairs:  [(1, 1), (1, 2), (1, 3), (2, 1), (2, 2), (2, 3), (3, 1), (3, 2), (3, 3), (4, 4), (4, 5), (4, 6), (4, 7), (5, 4), (5, 5), (5, 6), (5, 7), (6, 4), (6, 5), (6, 6), (6, 7), (7, 4), (7, 5), (7, 6), (7, 7), (8, 8), (8, 9), (8, 10), (8, 11), (9, 8), (9, 9), (9, 10), (9, 11), (10, 8), (10, 9), (10, 10), (10, 11), (11, 8), (11, 9), (11, 10), (11, 11)]
- Is reflexive:  True
    - Pairs:  [(1, 1), (2, 2), (3, 3), (4, 4), (5, 5), (6, 6), (7, 7), (8, 8), (9, 9), (10, 10), (11, 11)]
- Is symmetric:  True
    - Pairs:  [((1, 2), (2, 1)), ((1, 3), (3, 1)), ((2, 1), (1, 2)), ((2, 3), (3, 2)), ((3, 1), (1, 3)), ((3, 2), (2, 3)), ((4, 5), (5, 4)), ((4, 6), (6, 4)), ((4, 7), (7, 4)), ((5, 4), (4, 5)), ((5, 6), (6, 5)), ((5, 7), (7, 5)), ((6, 4), (4, 6)), ((6, 5), (5, 6)), ((6, 7), (7, 6)), ((7, 4), (4, 7)), ((7, 5), (5, 7)), ((7, 6), (6, 7)), ((8, 9), (9, 8)), ((8, 10), (10, 8)), ((8, 11), (11, 8)), ((9, 8), (8, 9)), ((9, 10), (10, 9)), ((9, 11), (11, 9)), ((10, 8), (8, 10)), ((10, 9), (9, 10)), ((10, 11), (11, 10)), ((11, 8), (8, 11)), ((11, 9), (9, 11)), ((11, 10), (10, 11))]
- Is transitive:  True
    - Paths:  [((1, 2), (2, 3), (1, 3)), ((1, 3), (3, 2), (1, 2)), ((2, 1), (1, 3), (2, 3)), ((2, 3), (3, 1), (2, 1)), ((3, 1), (1, 2), (3, 2)), ((3, 2), (2, 1), (3, 1)), ((4, 5), (5, 6), (4, 6)), ((4, 5), (5, 7), (4, 7)), ((4, 6), (6, 5), (4, 5)), ((4, 6), (6, 7), (4, 7)), ((4, 7), (7, 5), (4, 5)), ((4, 7), (7, 6), (4, 6)), ((5, 4), (4, 6), (5, 6)), ((5, 4), (4, 7), (5, 7)), ((5, 6), (6, 4), (5, 4)), ((5, 6), (6, 7), (5, 7)), ((5, 7), (7, 4), (5, 4)), ((5, 7), (7, 6), (5, 6)), ((6, 4), (4, 5), (6, 5)), ((6, 4), (4, 7), (6, 7)), ((6, 5), (5, 4), (6, 4)), ((6, 5), (5, 7), (6, 7)), ((6, 7), (7, 4), (6, 4)), ((6, 7), (7, 5), (6, 5)), ((7, 4), (4, 5), (7, 5)), ((7, 4), (4, 6), (7, 6)), ((7, 5), (5, 4), (7, 4)), ((7, 5), (5, 6), (7, 6)), ((7, 6), (6, 4), (7, 4)), ((7, 6), (6, 5), (7, 5)), ((8, 9), (9, 10), (8, 10)), ((8, 9), (9, 11), (8, 11)), ((8, 10), (10, 9), (8, 9)), ((8, 10), (10, 11), (8, 11)), ((8, 11), (11, 9), (8, 9)), ((8, 11), (11, 10), (8, 10)), ((9, 8), (8, 10), (9, 10)), ((9, 8), (8, 11), (9, 11)), ((9, 10), (10, 8), (9, 8)), ((9, 10), (10, 11), (9, 11)), ((9, 11), (11, 8), (9, 8)), ((9, 11), (11, 10), (9, 10)), ((10, 8), (8, 9), (10, 9)), ((10, 8), (8, 11), (10, 11)), ((10, 9), (9, 8), (10, 8)), ((10, 9), (9, 11), (10, 11)), ((10, 11), (11, 8), (10, 8)), ((10, 11), (11, 9), (10, 9)), ((11, 8), (8, 9), (11, 9)), ((11, 8), (8, 10), (11, 10)), ((11, 9), (9, 8), (11, 8)), ((11, 9), (9, 10), (11, 10)), ((11, 10), (10, 8), (11, 8)), ((11, 10), (10, 9), (11, 9))]
- Is asymmetric:  False
- Is antisymmetric:  False

Relation matrix:
    1   2   3   4   5   6   7   8   9   10  11
1    1   1   1   0   0   0   0   0   0   0   0
2    1   1   1   0   0   0   0   0   0   0   0
3    1   1   1   0   0   0   0   0   0   0   0
4    0   0   0   1   1   1   1   0   0   0   0
5    0   0   0   1   1   1   1   0   0   0   0
6    0   0   0   1   1   1   1   0   0   0   0
7    0   0   0   1   1   1   1   0   0   0   0
8    0   0   0   0   0   0   0   1   1   1   1
9    0   0   0   0   0   0   0   1   1   1   1
10   0   0   0   0   0   0   0   1   1   1   1
11   0   0   0   0   0   0   0   1   1   1   1
- Equivalence:  True
    - Classes:  {1: {1, 2, 3}, 2: {1, 2, 3}, 3: {1, 2, 3}, 4: {4, 5, 6, 7}, 5: {4, 5, 6, 7}, 6: {4, 5, 6, 7}, 7: {4, 5, 6, 7}, 8: {8, 9, 10, 11}, 9: {8, 9, 10, 11}, 10: {8, 9, 10, 11}, 11: {8, 9, 10, 11}}
    - Partitions:  [{1, 2, 3}, {4, 5, 6, 7}, {8, 9, 10, 11}]
||||||||||||||||||||||||||||||||
```

Grafo:


2
3
1
9
8
11
10
7
5
6
4

Relación 5: equivalencia:

Conjunto A = B = {3, 7, 4, 8, 5, 6, 0}

Relación R = {(0,0),(0,4),(0,6),(0,8),(4,0),(4,4),(4,6),(4,8),(6,0),(6,4),(6,6),(6,8),(8,0),(8,4),(8,6),(8,8),(3,3),(3,5),(3,7),(5,3),(5,5),(5,7),(7,3),(7,5),(7,7)}

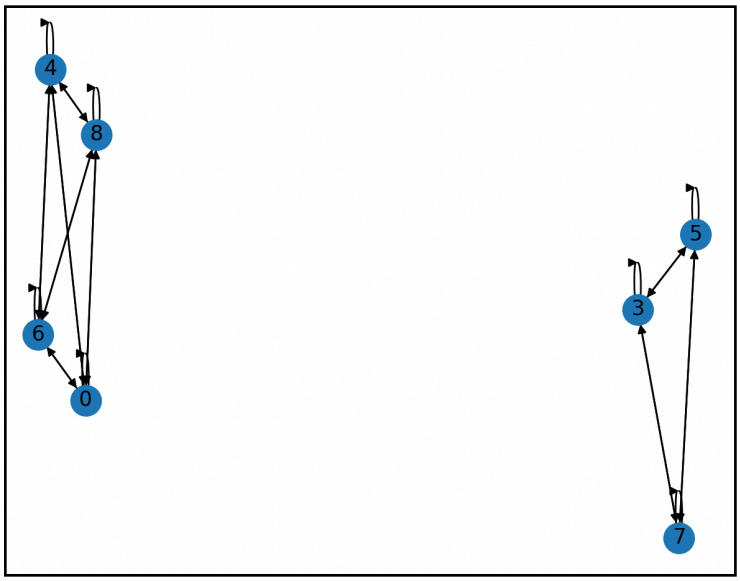
Historial de consola:

```
Introduce el conjunto de nodos, separados por comas. Ejemplo: 1,2,3,4
Nodos: 3, 7, 4, 8, 5, 6, 0

Introduce los pares de la relación como una lista de tuplas. Ejemplo: [(1,1), (2,1), (3,2)]
Pares: [(0,0),(0,4),(0,6),(0,8),(4,0),(4,4),(4,6),(4,8),(6,0),(6,4),(6,6),(6,8),(8,0),(8,4),(8,6),(8,8),(3,3),(3,5),(3,7),(5,3),(5,5),(5,7),(7,3),(7,5),(7,7)]
||||||||||||||||||||||||||||||||
------  Relación introducida  ------
- Relation pairs:  [(0, 0), (0, 4), (0, 6), (0, 8), (4, 0), (4, 4), (4, 6), (4, 8), (6, 0), (6, 4), (6, 6), (6, 8), (8, 0), (8, 4), (8, 6), (8, 8), (3, 3), (3, 5), (3, 7), (5, 3), (5, 5), (5, 7), (7, 3), (7, 5), (7, 7)]
- Is reflexive:  True
    - Pairs:  [(0, 0), (4, 4), (6, 6), (8, 8), (3, 3), (5, 5), (7, 7)]
- Is symmetric:  True
    - Pairs:  [((0, 4), (4, 0)), ((0, 6), (6, 0)), ((0, 8), (8, 0)), ((4, 0), (0, 4)), ((4, 6), (6, 4)), ((4, 8), (8, 4)), ((6, 0), (0, 6)), ((6, 4), (4, 6)), ((6, 8), (8, 6)), ((8, 0), (0, 8)), ((8, 4), (4, 8)), ((8, 6), (6, 8)), ((3, 5), (5, 3)), ((3, 7), (7, 3)), ((5, 3), (3, 5)), ((5, 7), (7, 5)), ((7, 3), (3, 7)), ((7, 5), (5, 7))]
- Is transitive:  True
    - Paths:  [((0, 4), (4, 6), (0, 6)), ((0, 4), (4, 8), (0, 8)), ((0, 6), (6, 4), (0, 4)), ((0, 6), (6, 8), (0, 8)), ((0, 8), (8, 4), (0, 4)), ((0, 8), (8, 6), (0, 6)), ((4, 0), (0, 6), (4, 6)), ((4, 0), (0, 8), (4, 8)), ((4, 6), (6, 0), (4, 0)), ((4, 6), (6, 8), (4, 8)), ((4, 8), (8, 0), (4, 0)), ((4, 8), (8, 6), (4, 6)), ((6, 0), (0, 4), (6, 4)), ((6, 0), (0, 8), (6, 8)), ((6, 4), (4, 0), (6, 0)), ((6, 4), (4, 8), (6, 8)), ((6, 8), (8, 0), (6, 0)), ((6, 8), (8, 4), (6, 4)), ((8, 0), (0, 4), (8, 4)), ((8, 0), (0, 6), (8, 6)), ((8, 4), (4, 0), (8, 0)), ((8, 4), (4, 6), (8, 6)), ((8, 6), (6, 0), (8, 0)), ((8, 6), (6, 4), (8, 4)), ((3, 5), (5, 7), (3, 7)), ((3, 7), (7, 5), (3, 5)), ((5, 3), (3, 7), (5, 7)), ((5, 7), (7, 3), (5, 3)), ((7, 3), (3, 5), (7, 5)), ((7, 5), (5, 3), (7, 3))]
- Is asymmetric:  False
- Is antisymmetric:  False

Relation matrix:
   0  3  4  5  6  7  8
0  1  0  1  0  1  0  1
3  0  1  0  1  0  1  0
4  1  0  1  0  1  0  1
5  0  1  0  1  0  1  0
6  1  0  1  0  1  0  1
7  0  1  0  1  0  1  0
8  1  0  1  0  1  0  1
- Equivalence:  True
    - Classes:  {0: {0, 8, 4, 6}, 4: {0, 8, 4, 6}, 6: {0, 8, 4, 6}, 8: {0, 8, 4, 6}, 3: {3, 5, 7}, 5: {3, 5, 7}, 7: {3, 5, 7}}
    - Partitions:  [{0, 8, 4, 6}, {3, 5, 7}]
||||||||||||||||||||||||||||||||
```

Grafo

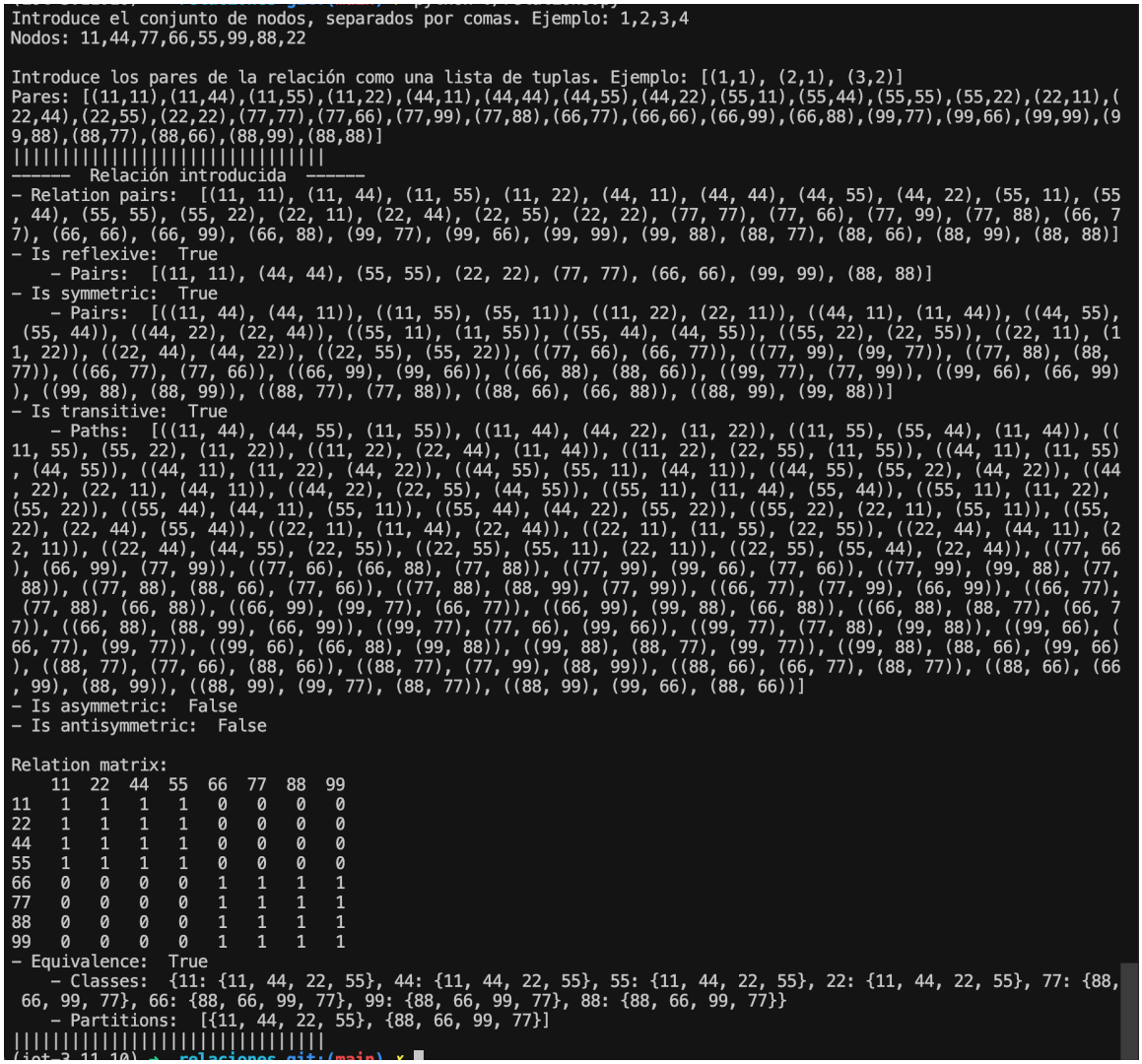


Relación 6: equivalencia:

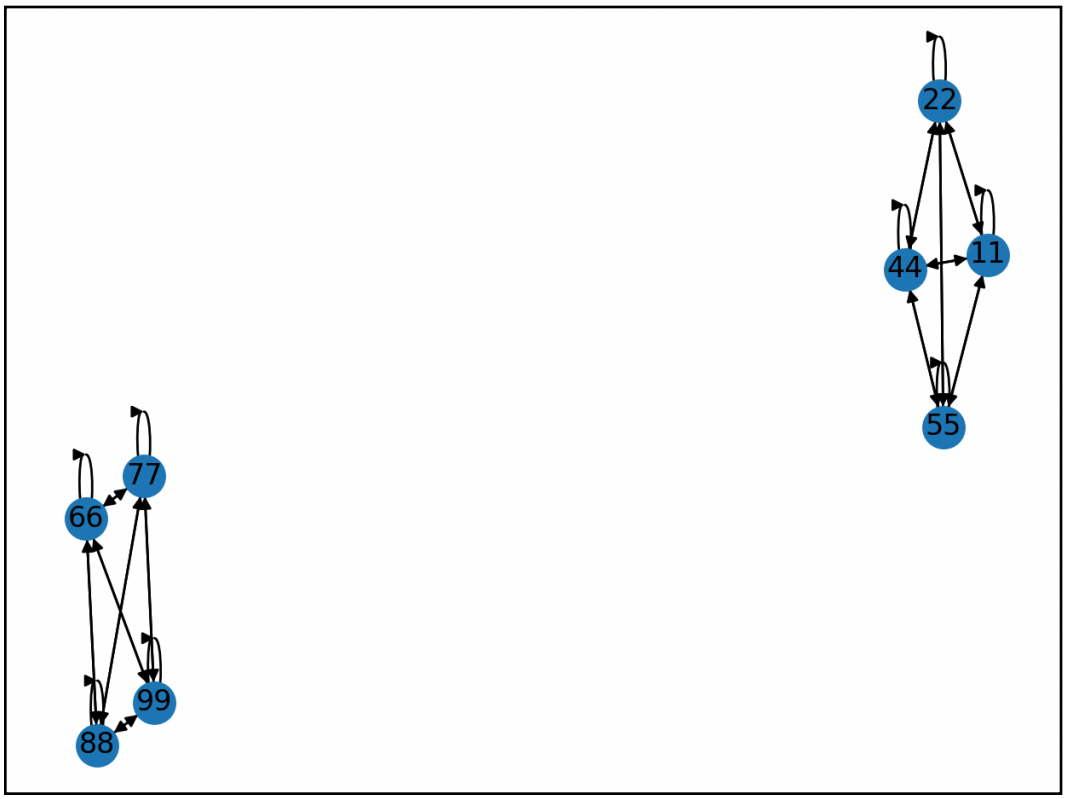
Conjunto A = B = {11,44,77,66,55,99,88,22}

Relación R = {(11,11),(11,44),(11,55),(11,22),(44,11),(44,44),(44,55),(44,22),(55,11),(55,44),(55,55),(55,22),(22,11),(22,44),(22,55),(22,22),(77,77),(77,66),(77,99),(77,88),(66,77),(66,66),(66,99),(66,88),(99,77),(99,66),(99,99),(99,88),(88,77),(88,66),(88,99),(88,88)}

Historial de consola:



Grafo:



Relación 7: no orden parcial y no equivalencia:

Conjuntos A = B = {1,2,3,4,5}

Relación R = {(1,2),(2,1), (2,3), (4,5)}

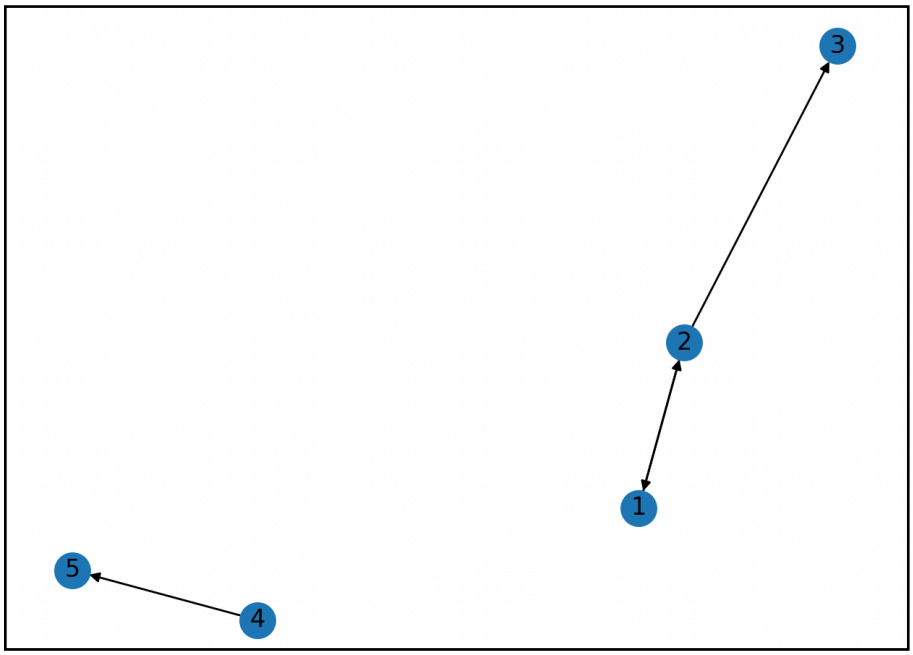
Historial de consola:

```
Introduce el conjunto de nodos, separados por comas. Ejemplo: 1,2,3,4
Nodos: 1,2,3,4,5

Introduce los pares de la relación como una lista de tuplas. Ejemplo: [(1,1), (2,1), (3,2)]
Pares: [(1,2),(2,1),(2,3),(4,5)]
||||||||||||||||||||||||||||||||
------  Relación introducida  ------
- Relation pairs:  [(1, 2), (2, 1), (2, 3), (4, 5)]
- Is reflexive:  False
- Is symmetric:  False
- Is transitive:  False
- Is asymmetric:  False
- Is antisymmetric:  False

Relation matrix:
   1  2  3  4  5
1  0  1  0  0  0
2  1  0  1  0  0
3  0  0  0  0  0
4  0  0  0  0  1
5  0  0  0  0  0
||||||||||||||||||||||||||||||||
```

Grafo:



Relación 8: no orden parcial y no equivalencia:

Conjuntos A = B = {1,2,3,4,5,6,7,8}

Relación R = {(1,2),(2,1),(2,3),(3,5),(4,6),(5,4),(6,7),(7,3),(8,5)}

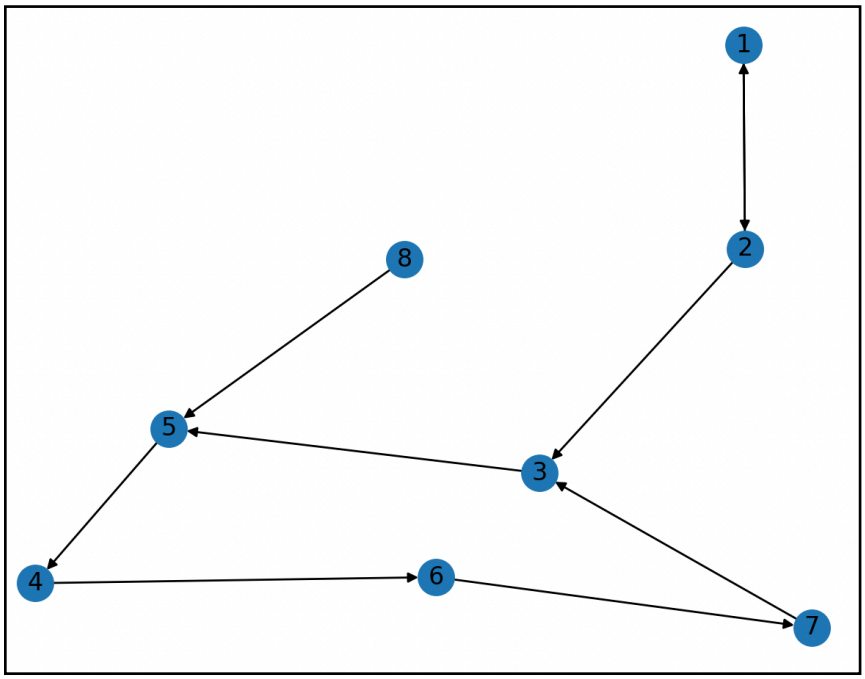
Historial de consola:

```
Introduce el conjunto de nodos, separados por comas. Ejemplo: 1,2,3,4
Nodos: 1,2,3,4,5,6,7,8

Introduce los pares de la relación como una lista de tuplas. Ejemplo: [(1,1), (2,1), (3,2)]
Pares: [(1,2),(2,1),(2,3),(3,5),(4,6),(5,4),(6,7),(7,3),(8,5)]
||||||||||||||||||||||||||||||||
------  Relación introducida  ------
- Relation pairs:  [(1, 2), (2, 1), (2, 3), (3, 5), (4, 6), (5, 4), (6, 7), (7, 3), (8, 5)]
- Is reflexive:  False
- Is symmetric:  False
- Is transitive:  False
- Is asymmetric:  False
- Is antisymmetric:  False

Relation matrix:
   1  2  3  4  5  6  7  8
1  0  1  0  0  0  0  0  0
2  1  0  1  0  0  0  0  0
3  0  0  0  0  1  0  0  0
4  0  0  0  0  0  1  0  0
5  0  0  0  1  0  0  0  0
6  0  0  0  0  0  0  1  0
7  0  0  1  0  0  0  0  0
8  0  0  0  0  1  0  0  0
||||||||||||||||||||||||||||||||
```

Grafo:



Relación 9: no orden parcial y no equivalencia:

Conjuntos A = B = {10,11,12,13,14,15,16,17}

Relación R = {(10,12),(12,14),(14,11),(11,13),(13,15),(15,10),(16,17),(17,16),(12,16)}

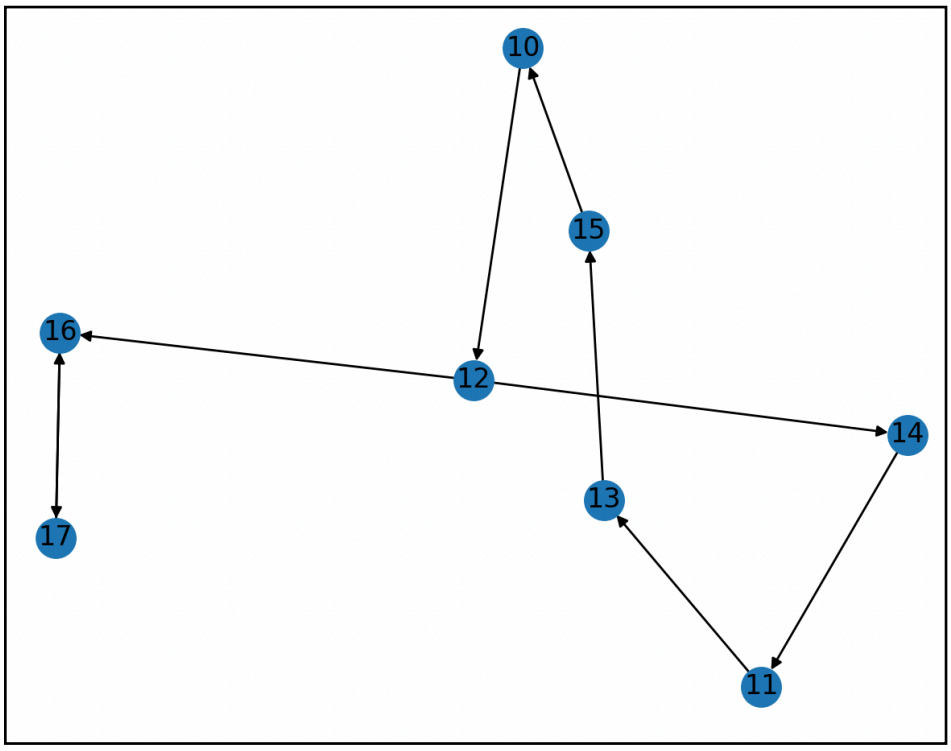
Historial de consola:

```
Introduce el conjunto de nodos, separados por comas. Ejemplo: 1,2,3,4
Nodos: 10,11,12,13,14,15,16,17

Introduce los pares de la relación como una lista de tuplas. Ejemplo: [(1,1), (2,1), (3,2)]
Pares: [(10,12),(12,14),(14,11),(11,13),(13,15),(15,10),(16,17),(17,16),(12,16)]
||||||||||||||||||||||||||||||||
------  Relación introducida  ------
- Relation pairs:  [(10, 12), (12, 14), (14, 11), (11, 13), (13, 15), (15, 10), (16, 17), (17, 16), (12, 16)]
- Is reflexive:  False
- Is symmetric:  False
- Is transitive:  False
- Is asymmetric:  False
- Is antisymmetric:  False

Relation matrix:
    10  11  12  13  14  15  16  17
10   0   0   1   0   0   0   0   0
11   0   0   0   1   0   0   0   0
12   0   0   0   0   1   0   1   0
13   0   0   0   0   0   1   0   0
14   0   1   0   0   0   0   0   0
15   1   0   0   0   0   0   0   0
16   0   0   0   0   0   0   0   1
17   0   0   0   0   0   0   1   0
||||||||||||||||||||||||||||||||
```

Grafo:



7. Discusión

El programa cumple con la tarea de identificar y clasificar relaciones de manera automática. Los resultados coinciden con lo esperado y las gráficas ayudan a interpretar visualmente los casos de equivalencias y órdenes parciales.

Como limitación, si el conjunto es muy grande y las relaciones son muchas, las funciones pueden tardar un poco ya que suelen tener complejidades cuadráticas. Además, las gráficas se vuelven poco prácticas, pero para fines académicos y conjuntos pequeños funciona correctamente.

8. Conclusiones

- El programa identifica de manera confiable las propiedades de una relación y su clasificación.
- Python resultó útil por la claridad de su sintaxis y las librerías gráficas disponibles.
- El proyecto permitió aplicar los conceptos de matemáticas discretas de forma práctica y visual.
- La herramienta es sencilla pero efectiva como apoyo en el estudio de relaciones en conjuntos.

9. Referencias

- Johnsonbaugh, R. Discrete Mathematics. Pearson, 8th edition, 2017.
- Rosen, K. Discrete Mathematics and Its Applications. McGraw Hill, 7th edition, 2012.
- Matousek, J., Nesetril, J. Invitation to Discrete Mathematics. Oxford University Press, 2nd edition, 2008.
- Kun, J. A Programmer´s Introduction to Mathematics. Create Space, 2018.
- MIT OpenCourseWare. Mathematics for Computer Science. https://ocw.mit.edu