Slide 3 - Conjuntos

1. Listagem de Elementos de um Conjunto

Como você pode representar o conjunto (V = {a, e, i, o, u}) em Python e imprimir seus elementos?

```
V = {'a', 'e', 'i', 'o', 'u'}
print("Os elementos do conjunto V são:", V)
```

```
Os elementos do conjunto V são: {'a', 'e', 'o', 'i', 'u'}
```

2. Verificação de Elementos em um Conjunto

Dado o conjunto (V = {a, e, i, o, u}), como você pode verificar se o elemento 'a' está presente em (V)?

```
V = {'a', 'e', 'i', 'o', 'u'}
print("'a' está em V?", 'a' in V)
```

```
'a' está em V? True
```

3. Criação de um Conjunto com Propriedades Específicas

Como você pode criar um conjunto (B) em Python que contém todos os números inteiros pares maiores que 10 e menores que 20?

```
B = {x for x in range(11, 20) if x % 2 == 0}
print("Conjunto B com pares entre 10 e 20:", B)
```

```
Conjunto B com pares entre 10 e 20: {16, 18, 12, 14}
```

4. Comparação de Conjuntos

Se (V = {a, e, i, o, u}) e (C = {i, o, u}), como você pode verificar se todos os elementos de (C) também estão em (V)?

```
C = {'i', 'o', 'u'}
print("C está contido em V?", C.issubset(V))
```

```
C está contido em V? True
```

5. Descrição de Conjuntos por Compreensão

Como você pode representar um conjunto (D) em Python que contém todos os números inteiros de 1 a 10 que são divisíveis por 3?

```
D = {x for x in range(1, 11) if x % 3 == 0}
print("D =", D)
```

```
D = {9, 3, 6}
```

6. União de Conjuntos

Dados dois conjuntos (A = {1, 2, 3}) e (B = {3, 4, 5}), como você pode obter a união de (A) e (B)?

```
A = {1, 2, 3}
B = {3, 4, 5}
print("A ∪ B =", A.union(B))
```

```
A ∪ B = {1, 2, 3, 4, 5}
```

7. Interseção de Conjuntos

Dado os conjuntos (A = {1, 2, 3}) e (B = {3, 4, 5}), como você pode encontrar a interseção entre (A) e (B)?

```
print("A ∩ B =", A.intersection(B))
```

```
A ∩ B = {3}
```

8. Diferença entre Conjuntos

Como você pode determinar os elementos que estão em (A = {1, 2, 3}) mas não estão em (B = {3, 4, 5})?

```
print("A - B =", A.difference(B))
```

```
A - B = {1, 2}
```

9. Simetria de Diferença entre Conjuntos

Se (A = {1, 2, 3}) e (B = {3, 4, 5}), como você pode obter a diferença simétrica entre (A) e (B)?

```
print("A Δ B =", A.symmetric_difference(B))
```

```
A Δ B = {1, 2, 4, 5}
```

10. Subconjuntos e Superconjuntos

Dado (A = {1, 2, 3}) e (B = {1, 2, 3, 4, 5}), como você pode verificar se (A) é um subconjunto de (B) e se (B) é um superconjunto de (A)?

```python
A = {1, 2, 3}
B = {1, 2, 3, 4, 5}
print("A é subconjunto de B?", A.issubset(B))
print("B é superconjunto de A?", B.issuperset(A))
```

```
A é subconjunto de B? True
B é superconjunto de A? True
```

11. Números Pares Maiores que 10

$B = \{x : x \text{ é um número par}, x > 10\}$

```python
B = {x for x in range(11, 101) if x % 2 == 0}
print("Pares > 10 e <= 100:", B)
```

```
Pares > 10 e <= 100: {12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34,
36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68,
70, 72, 74, 76, 78, 80, 82, 84, 86, 88, 90, 92, 94, 96, 98, 100}
```

12. Números Primos Menores que 20

$P = \{x : x \text{ é um número primo}, x < 20\}$

```python
def is_prime(n):
    if n < 2: return False
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0: return False
    return True
P = {x for x in range(2, 20) if is_prime(x)}
print("Primos < 20:", P)
```

```
Primos < 20: {2, 3, 5, 7, 11, 13, 17, 19}
```

13. Números Ímpares Divisíveis por 3 até 30

$I = \{x : x \text{ é um número ímpar divisível por } 3, x \leq 30\}$

```python
I = {x for x in range(1, 31) if x % 2 == 1 and x % 3 == 0}
print("Ímpares divisíveis por 3 até 30:", I)
```

```
Ímpares divisíveis por 3 até 30: {3, 9, 15, 21, 27}
```

14. Quadrados Perfeitos Menores que 100

$Q = \{x^2 : x \text{ é um número inteiro}, x^2 < 100\}$

```python
Q = {x**2 for x in range(1, 10)}
print("Quadrados perfeitos < 100:", Q)
```

```
Quadrados perfeitos < 100: {64, 1, 4, 36, 9, 16, 49, 81, 25}
```

15. Múltiplos de 5 entre 10 e 50

$M = \{ x : x \text{ é um múltiplo de } 5, 10 < x < 50 \}$

```
M = {x for x in range(11, 50) if x % 5 == 0}
print("Múltiplos de 5 entre 10 e 50:", M)
```

```
Múltiplos de 5 entre 10 e 50: {35, 40, 45, 15, 20, 25, 30}
```

15. Subconjunto próprio e não próprio

Dados dois conjunto em caa um dos cenários abaixo, escreva um script em python para verificar:

- se A é subconjunto de B e se C é subconjunto próprio de D
- Gerar um diagrama de Venn que ilustre as relações entre os conjuntos em cada um dos cenários
- Explicar porque o diagrama pode mostrar "0"e o que isso significa em termos dos elementos dos conjuntos.

Cenário 1

- Considere (A = {1, 2}) e (B = {1, 2, 3}).

Cenário 2

- Considere (C = {1, 2, 3}) e (D = {1, 2, 3}).

```
import matplotlib.pyplot as plt
from matplotlib_venn import venn2

# Definindo os conjuntos
A = {1, 2}
B = {1, 2, 3}
C = {1, 2, 3}
D = {1, 2, 3}

# Verificando subconjunto próprio
is_subset_proper_A_B = A < B
print(f"A é subconjunto próprio de B: {is_subset_proper_A_B}")

# Verificando subconjunto não próprio
is_subset_C_D = C.issubset(D) and C == D
print(f"C é subconjunto não próprio (igual) de D: {is_subset_C_D}")

# Gerando diagrama de Venn para subconjunto próprio
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
venn2([A, B], ('A', 'B'))
```

```
plt.title("A é subconjunto próprio de B")

# Gerando diagrama de Venn para subconjunto não próprio
plt.subplot(1, 2, 2)
venn2([C, D], ('C', 'D'))
plt.title("C é subconjunto não próprio (igual) de D")

plt.show()

----------------------------------------------------------------------
-----
ModuleNotFoundError                       Traceback (most recent call
last)
Cell In[34], line 1
----> 1 import matplotlib.pyplot as plt
      2 from matplotlib_venn import venn2
      4 # Definindo os conjuntos

ModuleNotFoundError: No module named 'matplotlib'
```

Slide 19 - Operação Produto Cartesiano

O **produto cartesiano** dos conjuntos A e B, denotado por $A \times B$, consiste do conjunto de todos os pares ordenados (a, b) com a primeira componente em A e a segunda componente em B. Ou seja, $A \times B = \{(x, y) : x \in \text{A e y} \in \text{B}\}$.

Por exemplo, sejam A = {1, 2} e B = {3, 4}, então:

- A × B = {(1, 3), (1, 4), (2, 3), (2, 4)}
- B × A = {(3, 1), (3, 2), (4, 1), (4, 2)}
- A × A = A^2 = {(1, 1), (1, 2), (2, 1), (2, 2)}

Aqui está o código Python para calcular o produto cartesiano:

```
from itertools import product

# Definindo os conjuntos
A = {1, 2}
B = {3, 4}

# Produto cartesiano A × B
A_x_B = list(product(A, B))

# Produto cartesiano B × A
B_x_A = list(product(B, A))

# Produto cartesiano A × A (A ao quadrado)
A_squared = list(product(A, A))

# Exibindo os resultados
```

```
print("A × B =", A_x_B)
print("B × A =", B_x_A)
print("A × A = A² =", A_squared)
```

```
A × B = [(1, 3), (1, 4), (2, 3), (2, 4)]
B × A = [(3, 1), (3, 2), (4, 1), (4, 2)]
A × A = A² = [(1, 1), (1, 2), (2, 1), (2, 2)]
```

Slide 22 - Relações

Slide 27 - Relação Reflexiva

Vamos examinar se as seguintes relações são reflexivas, fornecendo exemplos para cada caso:

1. **Relação ≤ (menor ou igual) no conjunto Z:**
   - Explicação: No conjunto dos inteiros Z, todo número é menor ou igual a si mesmo, $a \leq a$. Por exemplo, $5 \leq 5$, $-3 \leq -3$. Portanto, a relação ≤ é reflexiva.
2. **Inclusão de conjuntos ⊆ em uma coleção C de conjuntos:**
   - Explicação: Na relação de inclusão de conjuntos, todo conjunto é um subconjunto de si mesmo, $A \subseteq A$. Por exemplo, se $A = \{1,2\}$, então $A \subseteq A$ é verdadeiro. Assim, a relação de inclusão de conjuntos é reflexiva.
3. **Relação ⊥ (perpendicularidade) em um conjunto L de retas no plano:**
   - Explicação: Uma reta não é perpendicular a si mesma. Portanto, a relação de perpendicularidade ⊥ não é reflexiva, pois não existe $l \perp l$ para uma reta l em L.
4. **Relação ∥ (paralelismo) em um conjunto L de retas no plano:**
   - Explicação: Toda reta é paralela a si mesma no plano. Se l é uma reta em L, então $l \parallel l$ é sempre verdadeiro. Assim, a relação de paralelismo é reflexiva.
5. **Relação ¿ de divisibilidade no conjunto N:**
   - Explicação: No conjunto dos números naturais N, todo número é divisível por si mesmo, $a \vee a$. Por exemplo, $6 \vee 6$, $1 \vee 1$. Assim, a relação de divisibilidade é reflexiva.

Portanto, as relações ≤, ⊆, ∥ e ¿ são reflexivas, enquanto a relação ⊥ (perpendicularidade) não é reflexiva.

18. Relação Reflexiva: Vamos examinar se as seguintes relações são reflexivas, fornecendo exemplos para cada caso:

- Relação ≤ no conjunto Z:
- Inclusão de conjuntos ⊆ em uma coleção C de conjuntos:
- Relação ⊥ (perpendicularidade) em um conjunto L de retas no plano:
- Relação || (paralelismo) em um conjunto L de retas no plano:
- Relação | de divisibilidade no conjunto N:

```python
def is_reflexive(conjunto, relacao):
    for a in conjunto:
        if not relacao(a, a):
            return False
    return True

# 1. Relação ≤ no conjunto dos inteiros
conjunto_Z = [-3, 0, 5]
relacao_menor_igual = lambda a, b: a <= b

# 2. Inclusão de conjuntos
colecao_C = [{1}, {1, 2}, {1, 2, 3}]
relacao_inclusao = lambda A, B: A.issubset(B)

# 3. Perpendicularidade - simulação com dicionário
retas = ['r1', 'r2', 'r3']
perpendiculares = {
    ('r1', 'r2'), ('r2', 'r1'),
    ('r2', 'r3'), ('r3', 'r2'),
    # Não incluímos (r1, r1), (r2, r2)... para simular que não é
reflexiva
}
relacao_perpendicular = lambda l1, l2: (l1, l2) in perpendiculares

# 4. Paralelismo - toda reta é paralela a si mesma
relacao_paralela = lambda l1, l2: True if l1 == l2 else False  #
simples para reflexividade

# 5. Divisibilidade no conjunto dos naturais
naturais = [1, 2, 3, 6, 9]
relacao_divisibilidade = lambda a, b: b % a == 0 if a != 0 else False

# Testes
print("≤ é reflexiva em ℤ?", is_reflexive(conjunto_Z,
relacao_menor_igual))
print("⊆ é reflexiva em C?", is_reflexive(colecao_C,
relacao_inclusao))
print("⊥ é reflexiva em L?", is_reflexive(retas,
relacao_perpendicular))
print("∥ é reflexiva em L?", is_reflexive(retas, relacao_paralela))
print("| é reflexiva em ℕ?", is_reflexive(naturais,
relacao_divisibilidade))
```

```
≤ é reflexiva em ℤ? True
⊆ é reflexiva em C? True
⊥ é reflexiva em L? False
∥ é reflexiva em L? True
| é reflexiva em ℕ? True
```

Slide 28 - Relação Simétrica

Uma relação R em um conjunto A é **simétrica** se $a\,R\,b$ implica $b\,R\,a$, isto é, se $(a,b) \in R$ implica $(b,a) \in R$.

Analisando as relações dadas:

1. $R_1 = \{(1,1),(1,2),(2,3),(1,3),(4,4)\}$ não é simétrica, pois contém pares como $(1,2)$ sem o par inverso $(2,1)$.

2. $R_2 = \{(1,1),(1,2),(2,1),(2,2),(3,3),(4,4)\}$ é simétrica, pois cada par tem seu inverso na relação.

3. $R_3 = \{(1,3),(2,1)\}$ não é simétrica, já que $(1,3)$ está na relação, mas $(3,1)$ não.

4. $R_4 = \varnothing$, a relação vazia, é simétrica por definição, pois não existem pares que falhem em atender à condição de simetria.

   – A relação vazia não contém nenhum par ordenado. Na teoria das relações, a relação vazia é considerada simétrica,
   pois não há pares que possam violar a condição de simetria. Simetria significa que se (a,b) está em R, então (b,a)
   também deve estar em R. Na relação vazia, não existem pares para contradizer essa propriedade, então ela é trivialmente simétrica.

5. $R_5 = A \times A$, a relação universal, não é necessariamente simétrica a menos que cada elemento em A seja relacionado apenas consigo mesmo.

   – A relação universal em um conjunto A contém todos os pares possíveis (a,b) onde a e b são elementos de A.
   Para que uma relação seja simétrica, cada par (a,b) em R deve ter o par inverso (b,a) também em R.

   – Na relação universal, todos os pares possíveis estão incluídos, o que implica que para cada elemento a relacionado a b,
   b também está relacionado a a. No entanto, a verdadeira simetria exige que cada par e seu inverso estejam explicitamente
   presentes na relação. Assim, a relação universal é simétrica se, para cada par (a,b), o par (b,a) também está presente.

   – Por exemplo, se $A = \{1,2\}$, então $A \times A = \{(1,1),(1,2),(2,1),(2,2)\}$, que é simétrica porque cada par tem seu inverso na relação.

   – Agora, um exemplo onde a relação 5 não é simétrica: Considere o conjunto $A = \{1,2,3\}$ e a relação $R_5 = A \times A$ formada sem pares inversos explícitos, como: $R_5 = \{(1,1),(1,2),(2,3)\}$
   Neste caso, R_5 inclui o par $(1,2)$, mas não inclui o par inverso $(2,1)$, e inclui $(2,3)$ mas não $(3,2)$. Portanto, essa relação R_5 não é simétrica, pois não satisfaz a condição de que para todo (a,b) em R, (b,a) também deve estar em R.

Em resumo, a relação vazia R_4 é simétrica por definição, enquanto a relação universal R_5 sobre um conjunto A é simétrica se todos os pares possíveis e seus inversos estão presentes na relação.

19. Relação Simétrica: Uma relação R em um conjunto A é simétrica se aRb implica bRa, isto é, se (a,b) E R implica (b,a) E R.

Analise as relações:

1. R1 = {(1,1), (1,2), (2,3), (1,3), (4,4)}
2. R2 = {(1,1), (1,2), (2,1), (2,2), (3,3), (4,4)}
3. R3 = {(1,3), (2,1)}
4. R = Ø
5. R5 = A x A

```
def is_symmetric(relacao):
    for a, b in relacao:
        if (b, a) not in relacao:
            return False
    return True

# Definindo as relações
R1 = {(1,1), (1,2), (2,3), (1,3), (4,4)}
R2 = {(1,1), (1,2), (2,1), (2,2), (3,3), (4,4)}
R3 = {(1,3), (2,1)}
R4 = set()  # Conjunto vazio
A = {1, 2, 3}
R5 = {(a, b) for a in A for b in A}  # Produto cartesiano A x A

# Verificando se cada uma é simétrica
print("R1 é simétrica?", is_symmetric(R1))
print("R2 é simétrica?", is_symmetric(R2))
print("R3 é simétrica?", is_symmetric(R3))
print("R (vazio) é simétrica?", is_symmetric(R4))
print("R5 = A x A é simétrica?", is_symmetric(R5))
```

```
R1 é simétrica? False
R2 é simétrica? True
R3 é simétrica? False
R (vazio) é simétrica? True
R5 = A x A é simétrica? True
```

Slide 29 - Transitividade

Analisando a transitividade das relações:

1. $R_1 = \{(1,1),(1,2),(2,3),(1,3),(4,4)\}$:
   - Uma relação é transitiva se $a R b$ e $b R c$ implicam $a R c$.
   - Em R_1, temos que $1 R 2$ e $2 R 3$ implicam $1 R 3$, e $(1,3)$ está presente em R_1, logo R_1 é transitiva.

- $1R1$ (reflexividade) não afeta a transitividade e é compatível com a definição.
- Não há outras combinações em R_1 que desafiem a transitividade, então R_1 é considerada transitiva.

2. $R_2 = \{(1,1),(1,2),(2,1),(2,2),(3,3),(4,4)\}$:
   - Em R_2, todas as combinações que seguem aRb e bRc resultam em aRc que também estão presentes em R_2.
   - Temos $1R2$ e $2R1$, e como $1R1$ está em R_2, a relação é transitiva para esses elementos.
   - Da mesma forma, $2R1$ e $1R2$ implicam $2R2$, que também está presente em R_2.
   - Todas as relações reflexivas como $1R1$, $2R2$, $3R3$, e $4R4$ também suportam a transitividade porque um elemento está sempre relacionado a si mesmo.
3. $R_3 = \{(1,3),(2,1)\}$:
   - R_3 não apresenta uma sequência direta para testar a transitividade (não temos um par onde o segundo elemento de um par é o primeiro elemento do outro), e sem elementos contraditórios, podemos considerar R_3 transitiva por definição.
4. $R_4 = \varnothing$, a relação vazia:
   - A relação vazia é considerada transitiva porque não há elementos para violar a condição de transitividade. Não existem pares em R_4 que contradigam a definição de transitividade, então R_4 é trivialmente transitiva.
5. $R_5 = A \times A$, a relação universal:
   - Na relação universal, todos os pares possíveis estão presentes. Para quaisquer a,b,c em A, os pares (a,b), (b,c), e (a,c) estão em R_5. Isso satisfaz a condição de transitividade, tornando R_5 transitiva.
   - A relação universal em um conjunto A inclui todos os pares possíveis (a,b) onde a e b são elementos de A.
   - Isso significa que para quaisquer elementos a, b, e c em A, as relações aRb e bRc implicam aRc, simplesmente porque todos os pares possíveis estão presentes em R_5.
   - Por exemplo, se A é o conjunto $1,2$, então R_5 incluirá $(1,1)$, $(1,2)$, $(2,1)$, e $(2,2)$. Para qualquer par (a,b) e (b,c), o par (a,c) também estará em R_5.
   - Portanto, R_5 é transitiva porque contém todas as combinações possíveis de pares, atendendo à definição de transitividade.

Portanto, R_1, R_2, R_3, R_4 e R_5 são todas relações transitivas.

20. Transitividade: Agora faça o código para analisar a transitividade das relações:

- R1 = {(1,1), (1,2), (2,3), (1,3), (4,4)}
- R2 = {(1,1), (1,2), (2,1), (2, 2), (3,3), (4,4)}
- R3 = {(1, 3), (2, 1)}
- R4 =Ø
- R5 = AxA

```python
def is_transitive(relacao):
    for (a, b) in relacao:
        for (c, d) in relacao:
            if b == c and (a, d) not in relacao:
                return False
    return True

# Relações
R1 = {(1,1), (1,2), (2,3), (1,3), (4,4)}
R2 = {(1,1), (1,2), (2,1), (2,2), (3,3), (4,4)}
R3 = {(1,3), (2,1)}
R4 = set()
A = {1, 2, 3}
R5 = {(a, b) for a in A for b in A}  # Produto cartesiano AxA

# Verificando transitividade
print("R1 é transitiva?", is_transitive(R1))
print("R2 é transitiva?", is_transitive(R2))
print("R3 é transitiva?", is_transitive(R3))
print("R (vazio) é transitiva?", is_transitive(R4))
print("R5 = A x A é transitiva?", is_transitive(R5))
```

```
R1 é transitiva? True
R2 é transitiva? True
R3 é transitiva? False
R (vazio) é transitiva? True
R5 = A x A é transitiva? True
```

Slide 30 - Relações de Equivalência

Uma relação R em um conjunto S é uma **relação de equivalência** se ela é reflexiva, simétrica e transitiva:

- **Reflexiva**: Para todo $a \in S$, temos $a\,R\,a$.
- **Simétrica**: Se $a\,R\,b$, então $b\,R\,a$.
- **Transitiva**: Se $a\,R\,b$ e $b\,R\,c$, então $a\,R\,c$.

Exemplos de Relações de Equivalência

1. **Classificação de animais em espécies**
   - A relação "é da mesma espécie que" é reflexiva, pois todo animal é da mesma espécie que ele mesmo.
   - É simétrica, porque se o animal A é da mesma espécie que o animal B, então B é da mesma espécie que A.
   - É transitiva, pois se A é da mesma espécie que B, e B é da mesma espécie que C, então A é da mesma espécie que C.
2. **Relação** $\{(1,1),(2,2),(3,3),(1,2),(2,1)\}$ **em** $\{1,2,3\}$
   - Reflexiva, pois cada elemento $1, 2, 3$ está relacionado a si mesmo.
   - Simétrica, pois para o par $(1,2)$ existe o par $(2,1)$.

– Transitiva, pois não existem pares que violem a transitividade nesta relação.

3. **Relação "$x+y$ é par" em N**
   – **Reflexiva**: Para todo a em N, $a+a$ resulta em um número par, portanto, é reflexiva.
   – **Simétrica**: Se $a+b$ é par, então $b+a$ também é par, dado que a adição é comutativa.
   – **Transitiva**: Esta propriedade é satisfeita. Considerando a, b, c em N, se $a+b$ e $b+c$ são pares, implica que $a+c$ seja par. Por exemplo, com $a=1$, $b=1$, e $c=3$, temos que $a+b$ é par, $b+c$ é par e $a+c$ também é par.
4. **Relação "$x=y^2$" em $\{0,1\}$**
   – Não forma uma relação de equivalência em um conjunto maior, pois não é simétrica nem transitiva. Por exemplo, $1=1^2$, mas não existe 1 tal que $1=0^2$ no conjunto $\{0,1\}$.

21. Relações de Equivalência: Uma relação R em um conjunto S é uma relação de equivalência se ela é reflexiva, simétrica e transitiva, sendo assim, analise os exemplos a seguir:

- Relação {(1,1), (2,2), (3,3), (1,2), (2,1)} em {1,2,3}
- Relação "x + y é par" em N
- Relação "x = y²" em {0,1}

```
def reflexiva(relacao, conjunto):

    return True

def simetrica(relacao):

    return True

def transitiva(relacao):

    return True

def equivalente(relacao, conjunto):


R1 = {(1, 1), (2, 2), (3, 3), (1, 2), (2, 1)}
S1 = {1, 2, 3}
print(f'Relação = {R1} em {S1} é equivalente? {equivalente(R1, S1)}.')

R2 = {(x, y) for x in range(100) for y in range(100) if (x + y) % 2 ==
0}
S2 = set(range(100))
print(f'Relação = {R2} em {S2} é equivalente? {equivalente(R2, S2)}.')

R3 = {(x, y) for x in {0, 1} for y in {0, 1} if x == y ** 2}
S3 = {0, 1}
print(f'Relação = {R3} em {S3} é equivalente? {equivalente(R3, S3)}.')
```

```
Relação = {(1, 2), (3, 3), (2, 1), (2, 2), (1, 1)} em {1, 2, 3} é
equivalente? True.
Relação = {(71, 29), (90, 42), (6, 48), (92, 88), (83, 39), (2, 50),
(4, 96), (36, 48), (55, 61), (30, 4), (29, 45), (48, 58), (64, 2),
(25, 47), (66, 48), (85, 61), (98, 0), (1, 67), (59, 45), (78, 58),
(94, 2), (10, 8), (31, 67), (3, 5), (22, 18), (24, 64), (43, 77), (40,
8), (17, 61), (58, 62), (77, 75), (61, 67), (80, 80), (33, 5), (52,
18), (93, 19), (54, 64), (73, 77), (67, 3), (86, 16), (70, 8), (89,
21), (88, 62), (5, 27), (79, 13), (63, 5), (82, 18), (26, 86), (0,
70), (19, 83), (32, 22), (51, 35), (35, 27), (12, 80), (53, 81), (72,
94), (56, 86), (9, 11), (28, 24), (47, 37), (46, 78), (65, 91), (49,
83), (68, 96), (21, 21), (62, 22), (81, 35), (42, 80), (83, 81), (74,
32), (95, 91), (92, 22), (27, 41), (30, 46), (7, 99), (20, 38), (39,
51), (4, 30), (23, 43), (41, 97), (13, 35), (25, 89), (16, 40), (57,
41), (76, 54), (34, 94), (37, 99), (50, 38), (69, 51), (71, 97), (84,
36), (87, 41), (22, 60), (96, 46), (99, 51), (15, 57), (18, 62), (31,
1), (8, 54), (11, 59), (52, 60), (43, 11), (45, 57), (64, 70), (77,
9), (61, 1), (80, 14), (38, 54), (79, 55), (98, 68), (82, 60), (73,
11), (91, 65), (75, 57), (94, 70), (10, 76), (3, 73), (47, 79), (0,
4), (19, 17), (21, 63), (40, 76), (12, 14), (53, 15), (72, 28), (33,
73), (74, 74), (93, 87), (46, 12), (65, 25), (49, 17), (68, 30), (67,
71), (86, 84), (70, 76), (89, 89), (42, 14), (83, 15), (5, 95), (63,
73), (95, 25), (39, 93), (14, 36), (13, 77), (32, 90), (16, 82), (35,
95), (7, 33), (26, 46), (9, 79), (28, 92), (69, 93), (41, 31), (60,
44), (25, 23), (44, 36), (62, 90), (34, 28), (37, 33), (78, 34), (97,
47), (96, 88), (99, 93), (71, 31), (90, 44), (6, 50), (92, 90), (8,
96), (2, 52), (4, 98), (36, 50), (55, 63), (27, 1), (30, 6), (38, 96),
(29, 47), (64, 4), (66, 50), (85, 63), (57, 1), (98, 2), (1, 69), (59,
47), (94, 4), (10, 10), (87, 1), (31, 69), (3, 7), (22, 20), (24, 66),
(43, 79), (40, 10), (58, 64), (77, 77), (61, 69), (80, 82), (33, 7),
(52, 20), (93, 21), (54, 66), (67, 5), (86, 18), (70, 10), (89, 23),
(88, 64), (5, 29), (79, 15), (63, 7), (82, 20), (26, 88), (0, 72),
(19, 85), (32, 24), (51, 37), (35, 29), (12, 82), (53, 83), (72, 96),
(56, 88), (9, 13), (28, 26), (47, 39), (46, 80), (65, 93), (49, 85),
(68, 98), (21, 23), (62, 24), (81, 37), (83, 83), (74, 34), (92, 24),
(27, 43), (30, 48), (20, 40), (39, 53), (4, 32), (23, 45), (41, 99),
(13, 37), (25, 91), (16, 42), (57, 43), (76, 56), (34, 96), (50, 40),
(69, 53), (71, 99), (84, 38), (1, 3), (87, 43), (22, 62), (96, 48),
(99, 53), (15, 59), (18, 64), (31, 3), (8, 56), (11, 61), (52, 62),
(24, 0), (43, 13), (45, 59), (64, 72), (77, 11), (61, 3), (80, 16),
(38, 56), (79, 57), (98, 70), (82, 62), (54, 0), (73, 13), (91, 67),
(75, 59), (94, 72), (10, 78), (26, 22), (3, 75), (47, 81), (0, 6),
(19, 19), (21, 65), (40, 78), (12, 16), (53, 17), (72, 30), (56, 22),
(33, 75), (74, 76), (93, 89), (46, 14), (65, 27), (49, 19), (68, 32),
(67, 73), (86, 86), (70, 78), (89, 91), (42, 16), (83, 17), (5, 97),
(63, 75), (95, 27), (39, 95), (14, 38), (13, 79), (32, 92), (16, 84),
(35, 97), (7, 35), (48, 36), (9, 81), (28, 94), (69, 95), (41, 33),
(60, 46), (25, 25), (44, 38), (62, 92), (34, 30), (37, 35), (78, 36),
(97, 49), (96, 90), (99, 95), (71, 33), (90, 46), (6, 52), (92, 92),
(8, 98), (2, 54), (43, 55), (17, 39), (36, 52), (55, 65), (38, 98),
```

```
(29, 49), (73, 55), (64, 6), (66, 52), (85, 65), (98, 4), (1, 71),
(59, 49), (91, 1), (94, 6), (10, 12), (12, 58), (31, 71), (3, 9), (47,
15), (24, 68), (65, 69), (68, 74), (40, 12), (58, 66), (77, 79), (42,
58), (61, 71), (80, 84), (33, 9), (74, 10), (93, 23), (54, 68), (95,
69), (67, 7), (86, 20), (70, 12), (89, 25), (88, 66), (5, 31), (63,
9), (7, 77), (26, 90), (39, 29), (0, 74), (19, 87), (60, 88), (13,
13), (32, 26), (51, 39), (16, 18), (35, 31), (34, 72), (53, 85), (72,
98), (37, 77), (56, 90), (9, 15), (28, 28), (69, 29), (46, 82), (49,
87), (90, 88), (62, 26), (81, 39), (83, 85), (96, 24), (99, 29), (2,
96), (92, 26), (8, 32), (27, 45), (11, 37), (30, 50), (29, 91), (20,
42), (4, 34), (23, 47), (64, 48), (25, 93), (38, 32), (57, 45), (76,
58), (59, 91), (50, 42), (91, 43), (94, 48), (84, 40), (1, 5), (87,
45), (3, 51), (22, 64), (15, 61), (18, 66), (31, 5), (33, 51), (52,
64), (24, 2), (45, 61), (86, 62), (58, 0), (77, 13), (89, 67), (61,
5), (80, 18), (79, 59), (98, 72), (63, 51), (82, 64), (54, 2), (95,
3), (75, 61), (88, 0), (10, 80), (26, 24), (28, 70), (47, 83), (0, 8),
(19, 21), (21, 67), (40, 80), (81, 81), (53, 19), (72, 32), (56, 24),
(74, 78), (93, 91), (46, 16), (49, 21), (68, 34), (67, 75), (70, 80),
(83, 19), (5, 99), (20, 84), (39, 97), (14, 40), (13, 81), (32, 94),
(16, 86), (35, 99), (7, 37), (48, 38), (9, 83), (50, 84), (69, 97),
(41, 35), (60, 48), (25, 27), (44, 40), (62, 94), (78, 38), (97, 51),
(96, 92), (99, 97), (71, 35), (6, 54), (92, 94), (2, 56), (43, 57),
(18, 0), (17, 41), (36, 54), (55, 67), (29, 51), (73, 57), (64, 8),
(66, 54), (85, 67), (98, 6), (1, 73), (91, 3), (94, 8), (10, 14), (12,
60), (31, 73), (3, 11), (47, 17), (24, 70), (65, 71), (68, 76), (21,
1), (40, 14), (58, 68), (77, 81), (42, 60), (61, 73), (80, 86), (33,
11), (74, 12), (93, 25), (54, 70), (95, 71), (67, 9), (86, 22), (70,
14), (89, 27), (88, 68), (5, 33), (63, 11), (7, 79), (26, 92), (39,
31), (0, 76), (19, 89), (60, 90), (13, 15), (32, 28), (51, 41), (16,
20), (35, 33), (34, 74), (53, 87), (37, 79), (56, 92), (9, 17), (28,
30), (69, 31), (46, 84), (49, 89), (90, 90), (62, 28), (81, 41), (83,
87), (96, 26), (99, 31), (2, 98), (92, 28), (8, 34), (27, 47), (11,
39), (30, 52), (29, 93), (20, 44), (4, 36), (23, 49), (64, 50), (55,
1), (25, 95), (38, 34), (57, 47), (76, 60), (59, 93), (50, 44), (91,
45), (94, 50), (85, 1), (84, 42), (1, 7), (87, 47), (3, 53), (22, 66),
(15, 63), (18, 68), (31, 7), (33, 53), (52, 66), (24, 4), (65, 5),
(68, 10), (45, 63), (86, 64), (58, 2), (77, 15), (89, 69), (61, 7),
(80, 20), (79, 61), (98, 74), (63, 53), (82, 66), (54, 4), (95, 5),
(75, 63), (88, 2), (10, 82), (51, 83), (7, 13), (26, 26), (28, 72),
(47, 85), (0, 10), (19, 23), (60, 24), (21, 69), (40, 82), (81, 83),
(34, 8), (53, 21), (72, 34), (37, 13), (56, 26), (74, 80), (93, 93),
(46, 18), (49, 23), (90, 24), (67, 77), (70, 82), (83, 21), (2, 32),
(20, 86), (39, 99), (23, 91), (14, 42), (55, 43), (13, 83), (32, 96),
(16, 88), (29, 27), (48, 40), (9, 85), (50, 86), (69, 99), (41, 37),
(25, 29), (44, 42), (85, 43), (62, 96), (59, 27), (78, 40), (97, 53),
(96, 94), (99, 99), (71, 37), (6, 56), (92, 96), (22, 0), (24, 46),
(43, 59), (18, 2), (17, 43), (36, 56), (77, 57), (80, 62), (52, 0),
(54, 46), (73, 59), (89, 3), (66, 56), (98, 8), (82, 0), (1, 75), (19,
65), (10, 16), (51, 17), (12, 62), (31, 75), (72, 76), (28, 6), (47,
19), (46, 60), (65, 73), (49, 65), (68, 78), (21, 3), (40, 16), (81,
```

```
17), (58, 70), (42, 62), (61, 75), (74, 14), (93, 27), (95, 73), (67,
11), (70, 16), (14, 84), (88, 70), (5, 35), (7, 81), (26, 94), (20,
20), (39, 33), (23, 25), (0, 78), (41, 79), (60, 92), (13, 17), (32,
30), (44, 84), (16, 22), (35, 35), (76, 36), (34, 76), (53, 89), (37,
81), (56, 94), (97, 95), (9, 19), (50, 20), (69, 33), (71, 79), (90,
92), (62, 30), (6, 98), (83, 89), (96, 28), (99, 33), (92, 30), (36,
98), (8, 36), (27, 49), (11, 41), (30, 54), (29, 95), (4, 38), (45,
39), (64, 52), (55, 3), (25, 97), (66, 98), (38, 36), (57, 49), (98,
50), (59, 95), (91, 47), (75, 39), (94, 52), (84, 44), (1, 9), (87,
49), (3, 55), (22, 68), (31, 9), (33, 55), (52, 68), (24, 6), (65, 7),
(68, 12), (67, 53), (86, 66), (58, 4), (70, 58), (77, 17), (89, 71),
(61, 9), (80, 22), (79, 63), (63, 55), (82, 68), (95, 7), (88, 4),
(10, 84), (51, 85), (7, 15), (26, 28), (28, 74), (47, 87), (0, 12),
(19, 25), (60, 26), (21, 71), (62, 72), (81, 85), (34, 10), (53, 23),
(72, 36), (37, 15), (56, 28), (74, 82), (46, 20), (49, 25), (90, 26),
(92, 72), (83, 23), (2, 34), (20, 88), (23, 93), (14, 44), (55, 45),
(13, 85), (16, 90), (29, 29), (48, 42), (50, 88), (41, 39), (25, 31),
(44, 44), (85, 45), (84, 86), (59, 29), (78, 42), (97, 55), (96, 96),
(71, 39), (6, 58), (22, 2), (24, 48), (43, 61), (18, 4), (17, 45),
(36, 58), (77, 59), (80, 64), (52, 2), (54, 48), (73, 61), (86, 0),
(89, 5), (66, 58), (98, 10), (82, 2), (1, 77), (19, 67), (10, 18),
(51, 19), (12, 64), (31, 77), (72, 78), (28, 8), (47, 21), (46, 62),
(65, 75), (49, 67), (68, 80), (21, 5), (40, 18), (81, 19), (58, 72),
(42, 64), (61, 77), (74, 16), (93, 29), (95, 75), (67, 13), (70, 18),
(14, 86), (88, 72), (5, 37), (7, 83), (26, 96), (20, 22), (39, 35),
(23, 27), (0, 80), (41, 81), (60, 94), (13, 19), (32, 32), (44, 86),
(16, 24), (35, 37), (76, 38), (34, 78), (53, 91), (37, 83), (56, 96),
(97, 97), (9, 21), (50, 22), (69, 35), (71, 81), (90, 94), (62, 32),
(83, 91), (96, 30), (99, 35), (15, 41), (18, 46), (92, 32), (8, 38),
(27, 51), (11, 43), (30, 56), (29, 97), (4, 40), (45, 41), (64, 54),
(25, 99), (38, 38), (57, 51), (98, 52), (59, 97), (91, 49), (75, 41),
(94, 54), (10, 60), (84, 46), (1, 11), (87, 51), (3, 57), (22, 70),
(19, 1), (40, 60), (31, 11), (72, 12), (33, 57), (52, 70), (93, 71),
(65, 9), (49, 1), (68, 14), (67, 55), (86, 68), (58, 6), (70, 60),
(89, 73), (61, 11), (5, 79), (79, 65), (63, 57), (82, 70), (95, 9),
(14, 20), (88, 6), (32, 74), (51, 87), (35, 79), (7, 17), (26, 30),
(9, 63), (28, 76), (47, 89), (0, 14), (41, 15), (60, 28), (44, 20),
(21, 73), (62, 74), (81, 87), (34, 12), (53, 25), (37, 17), (56, 30),
(97, 31), (74, 84), (71, 15), (90, 28), (6, 34), (92, 74), (83, 25),
(27, 93), (30, 98), (2, 36), (20, 90), (4, 82), (23, 95), (36, 34),
(55, 47), (13, 87), (16, 92), (57, 93), (29, 31), (48, 44), (50, 90),
(25, 33), (66, 34), (85, 47), (84, 88), (1, 53), (87, 93), (59, 31),
(78, 44), (96, 98), (31, 53), (22, 4), (24, 50), (43, 63), (17, 47),
(58, 48), (77, 61), (61, 53), (80, 66), (52, 4), (93, 5), (54, 50),
(73, 63), (86, 2), (89, 7), (88, 48), (82, 4), (26, 72), (0, 56), (19,
69), (32, 8), (51, 21), (12, 66), (53, 67), (72, 80), (56, 72), (28,
10), (47, 23), (46, 64), (65, 77), (49, 69), (68, 82), (21, 7), (62,
8), (81, 21), (42, 66), (83, 67), (74, 18), (95, 77), (92, 8), (14,
88), (5, 39), (7, 85), (48, 86), (20, 24), (39, 37), (23, 29), (41,
83), (60, 96), (13, 21), (25, 75), (44, 88), (16, 26), (76, 40), (34,
```

```
80), (37, 85), (78, 86), (97, 99), (50, 24), (69, 37), (71, 83), (90,
96), (84, 22), (87, 27), (96, 32), (99, 37), (15, 43), (18, 48), (17,
89), (8, 40), (27, 53), (11, 45), (30, 58), (29, 99), (45, 43), (64,
56), (80, 0), (38, 40), (79, 41), (98, 54), (59, 99), (91, 51), (75,
43), (94, 56), (10, 62), (1, 13), (3, 59), (22, 72), (19, 3), (40,
62), (12, 0), (31, 13), (72, 14), (33, 59), (52, 72), (93, 73), (65,
11), (49, 3), (68, 16), (67, 57), (86, 70), (58, 8), (70, 62), (89,
75), (42, 0), (61, 13), (5, 81), (63, 59), (95, 11), (14, 22), (88,
8), (32, 76), (51, 89), (35, 81), (7, 19), (26, 32), (9, 65), (28,
78), (47, 91), (0, 16), (41, 17), (60, 30), (44, 22), (21, 75), (62,
76), (81, 89), (34, 14), (53, 27), (37, 19), (56, 32), (97, 33), (71,
17), (90, 30), (6, 36), (92, 76), (83, 27), (27, 95), (2, 38), (20,
92), (4, 84), (23, 97), (36, 36), (55, 49), (13, 89), (16, 94), (57,
95), (29, 33), (48, 46), (50, 92), (25, 35), (66, 36), (85, 49), (84,
90), (1, 55), (87, 95), (59, 33), (78, 46), (31, 55), (22, 6), (24,
52), (43, 65), (17, 49), (58, 50), (77, 63), (61, 55), (80, 68), (52,
6), (93, 7), (54, 52), (73, 65), (86, 4), (89, 9), (88, 50), (5, 15),
(79, 1), (82, 6), (26, 74), (0, 58), (19, 71), (32, 10), (51, 23),
(35, 15), (12, 68), (53, 69), (72, 82), (56, 74), (28, 12), (47, 25),
(46, 66), (65, 79), (49, 71), (68, 84), (21, 9), (62, 10), (81, 23),
(42, 68), (83, 69), (74, 20), (95, 79), (92, 10), (14, 90), (27, 29),
(30, 34), (7, 87), (48, 88), (20, 26), (39, 39), (4, 18), (23, 31),
(41, 85), (60, 98), (13, 23), (25, 77), (44, 90), (16, 28), (57, 29),
(76, 42), (34, 82), (37, 87), (78, 88), (50, 26), (69, 39), (71, 85),
(90, 98), (84, 24), (87, 29), (22, 48), (96, 34), (99, 39), (15, 45),
(18, 50), (17, 91), (8, 42), (11, 47), (52, 48), (45, 45), (64, 58),
(80, 2), (38, 42), (79, 43), (98, 56), (82, 48), (91, 53), (75, 45),
(94, 58), (10, 64), (26, 8), (3, 61), (47, 67), (19, 5), (21, 51),
(40, 64), (12, 2), (53, 3), (72, 16), (56, 8), (33, 61), (74, 62),
(93, 75), (46, 0), (65, 13), (49, 5), (68, 18), (67, 59), (86, 72),
(70, 64), (89, 77), (42, 2), (83, 3), (5, 83), (63, 61), (95, 13),
(39, 81), (14, 24), (13, 65), (32, 78), (51, 91), (16, 70), (35, 83),
(7, 21), (48, 22), (9, 67), (28, 80), (69, 81), (41, 19), (60, 32),
(25, 11), (44, 24), (62, 78), (81, 91), (34, 16), (37, 21), (78, 22),
(97, 35), (96, 76), (99, 81), (71, 19), (90, 32), (6, 38), (92, 78),
(8, 84), (27, 97), (11, 89), (2, 40), (43, 41), (20, 94), (4, 86),
(23, 99), (17, 25), (36, 38), (55, 51), (38, 84), (57, 97), (29, 35),
(73, 41), (50, 94), (91, 95), (66, 38), (85, 51), (84, 92), (1, 57),
(87, 97), (59, 35), (12, 44), (31, 57), (22, 8), (24, 54), (65, 55),
(68, 60), (58, 52), (77, 65), (42, 44), (61, 57), (80, 70), (93, 9),
(54, 54), (95, 55), (86, 6), (89, 11), (88, 52), (5, 17), (7, 63),
(26, 76), (0, 60), (19, 73), (60, 74), (32, 12), (51, 25), (35, 17),
(34, 58), (53, 71), (72, 84), (37, 63), (56, 76), (9, 1), (28, 14),
(47, 27), (46, 68), (49, 73), (90, 74), (62, 12), (81, 25), (83, 71),
(96, 10), (92, 12), (14, 92), (27, 31), (30, 36), (29, 77), (48, 90),
(20, 28), (39, 41), (4, 20), (23, 33), (41, 87), (25, 79), (44, 92),
(85, 93), (16, 30), (57, 31), (76, 44), (59, 77), (78, 90), (50, 28),
(69, 41), (71, 87), (84, 26), (87, 31), (22, 50), (15, 47), (18, 52),
(17, 93), (8, 44), (11, 49), (52, 50), (43, 1), (54, 96), (45, 47),
(64, 60), (80, 4), (79, 45), (98, 58), (82, 50), (73, 1), (75, 47),
```

```
(94, 60), (10, 66), (26, 10), (3, 63), (47, 69), (19, 7), (21, 53),
(40, 66), (12, 4), (53, 5), (72, 18), (56, 10), (33, 63), (74, 64),
(93, 77), (46, 2), (65, 15), (49, 7), (68, 20), (67, 61), (86, 74),
(70, 66), (89, 79), (42, 4), (83, 5), (5, 85), (63, 63), (95, 15),
(39, 83), (14, 26), (13, 67), (32, 80), (51, 93), (16, 72), (35, 85),
(7, 23), (48, 24), (9, 69), (28, 82), (69, 83), (41, 21), (60, 34),
(25, 13), (44, 26), (62, 80), (81, 93), (34, 18), (37, 23), (78, 24),
(97, 37), (96, 78), (99, 83), (71, 21), (90, 34), (6, 40), (92, 80),
(8, 86), (27, 99), (11, 91), (2, 42), (43, 43), (20, 96), (4, 88),
(17, 27), (36, 40), (55, 53), (38, 86), (57, 99), (29, 37), (73, 43),
(50, 96), (91, 97), (66, 40), (85, 53), (84, 94), (1, 59), (87, 99),
(59, 37), (10, 0), (12, 46), (31, 59), (47, 3), (24, 56), (65, 57),
(68, 62), (40, 0), (58, 54), (77, 67), (42, 46), (61, 59), (80, 72),
(93, 11), (54, 56), (95, 57), (86, 8), (70, 0), (89, 13), (88, 54),
(5, 19), (7, 65), (26, 78), (39, 17), (0, 62), (19, 75), (60, 76),
(13, 1), (32, 14), (51, 27), (16, 6), (35, 19), (34, 60), (53, 73),
(72, 86), (37, 65), (56, 78), (9, 3), (28, 16), (69, 17), (46, 70),
(49, 75), (90, 76), (62, 14), (81, 27), (83, 73), (96, 12), (99, 17),
(2, 84), (92, 14), (14, 94), (55, 95), (8, 20), (27, 33), (11, 25),
(30, 38), (29, 79), (48, 92), (20, 30), (4, 22), (23, 35), (64, 36),
(41, 89), (25, 81), (44, 94), (85, 95), (38, 20), (57, 33), (76, 46),
(59, 79), (78, 92), (50, 30), (91, 31), (94, 36), (71, 89), (84, 28),
(87, 33), (3, 39), (22, 52), (24, 98), (15, 49), (18, 54), (17, 95),
(33, 39), (52, 52), (54, 98), (45, 49), (86, 50), (77, 1), (89, 55),
(80, 6), (79, 47), (98, 60), (63, 39), (82, 52), (75, 49), (10, 68),
(51, 69), (26, 12), (28, 58), (47, 71), (19, 9), (60, 10), (21, 55),
(40, 68), (81, 69), (53, 7), (72, 20), (56, 12), (74, 66), (93, 79),
(46, 4), (49, 9), (90, 10), (67, 63), (70, 68), (83, 7), (5, 87), (20,
72), (39, 85), (23, 77), (14, 28), (55, 29), (13, 69), (32, 82), (16,
74), (35, 87), (76, 88), (29, 13), (48, 26), (9, 71), (50, 72), (69,
85), (41, 23), (25, 15), (44, 28), (85, 29), (62, 82), (59, 13), (78,
26), (97, 39), (96, 80), (99, 85), (71, 23), (15, 91), (6, 42), (18,
96), (92, 82), (8, 88), (11, 93), (24, 32), (43, 45), (4, 90), (45,
91), (17, 29), (36, 42), (77, 43), (38, 88), (54, 32), (73, 45), (91,
99), (75, 91), (66, 42), (84, 96), (1, 61), (10, 2), (12, 48), (31,
61), (47, 5), (46, 46), (65, 59), (49, 51), (68, 64), (40, 2), (58,
56), (42, 48), (61, 61), (74, 0), (93, 13), (95, 59), (70, 2), (89,
15), (88, 56), (5, 21), (7, 67), (26, 80), (39, 19), (0, 64), (60,
78), (13, 3), (32, 16), (51, 29), (16, 8), (35, 21), (34, 62), (53,
75), (37, 67), (56, 80), (9, 5), (28, 18), (69, 19), (71, 65), (90,
78), (62, 16), (81, 29), (83, 75), (96, 14), (99, 19), (2, 86), (92,
16), (14, 96), (55, 97), (8, 22), (27, 35), (11, 27), (30, 40), (29,
81), (48, 94), (20, 32), (4, 24), (23, 37), (64, 38), (25, 83), (44,
96), (85, 97), (38, 22), (57, 35), (76, 48), (59, 81), (78, 94), (50,
32), (91, 33), (94, 38), (84, 30), (87, 35), (3, 41), (22, 54), (15,
51), (18, 56), (17, 97), (33, 41), (52, 54), (45, 51), (86, 52), (77,
3), (89, 57), (80, 8), (79, 49), (98, 62), (63, 41), (82, 54), (75,
51), (10, 70), (51, 71), (7, 1), (26, 14), (28, 60), (47, 73), (19,
11), (60, 12), (21, 57), (40, 70), (81, 71), (53, 9), (72, 22), (37,
1), (56, 14), (74, 68), (93, 81), (46, 6), (49, 11), (90, 12), (67,
```

```
65), (70, 70), (83, 9), (5, 89), (2, 20), (20, 74), (39, 87), (23,
79), (14, 30), (55, 31), (13, 71), (32, 84), (16, 76), (35, 89), (76,
90), (29, 15), (48, 28), (9, 73), (50, 74), (69, 87), (41, 25), (25,
17), (44, 30), (85, 31), (62, 84), (59, 15), (78, 28), (97, 41), (96,
82), (99, 87), (71, 25), (15, 93), (6, 44), (18, 98), (92, 84), (8,
90), (11, 95), (24, 34), (43, 47), (4, 92), (45, 93), (17, 31), (36,
44), (77, 45), (80, 50), (38, 90), (54, 34), (73, 47), (75, 93), (66,
44), (84, 98), (1, 63), (19, 53), (10, 4), (51, 5), (12, 50), (31,
63), (72, 64), (47, 7), (46, 48), (65, 61), (49, 53), (68, 66), (40,
4), (81, 5), (58, 58), (42, 50), (61, 63), (74, 2), (93, 15), (95,
61), (70, 4), (14, 72), (88, 58), (5, 23), (7, 69), (26, 82), (20, 8),
(39, 21), (23, 13), (0, 66), (41, 67), (60, 80), (13, 5), (32, 18),
(44, 72), (16, 10), (35, 23), (76, 24), (34, 64), (53, 77), (37, 69),
(56, 82), (97, 83), (9, 7), (50, 8), (69, 21), (71, 67), (90, 80),
(62, 18), (6, 86), (83, 77), (96, 16), (99, 21), (2, 88), (15, 27),
(18, 32), (92, 18), (36, 86), (55, 99), (8, 24), (27, 37), (11, 29),
(30, 42), (29, 83), (48, 96), (4, 26), (45, 27), (64, 40), (25, 85),
(66, 86), (85, 99), (38, 24), (57, 37), (98, 38), (59, 83), (78, 96),
(91, 35), (75, 27), (94, 40), (10, 46), (84, 32), (87, 37), (3, 43),
(22, 56), (40, 46), (17, 99), (33, 43), (52, 56), (93, 57), (68, 0),
(67, 41), (86, 54), (70, 46), (89, 59), (5, 65), (79, 51), (63, 43),
(82, 56), (32, 60), (51, 73), (35, 65), (7, 3), (26, 16), (9, 49),
(28, 62), (47, 75), (0, 0), (41, 1), (60, 14), (21, 59), (62, 60),
(81, 73), (53, 11), (37, 3), (56, 16), (74, 70), (71, 1), (90, 14),
(92, 60), (83, 11), (27, 79), (30, 84), (2, 22), (20, 76), (39, 89),
(4, 68), (23, 81), (14, 32), (55, 33), (13, 73), (16, 78), (57, 79),
(76, 92), (29, 17), (48, 30), (50, 76), (69, 89), (25, 19), (66, 20),
(85, 33), (84, 74), (87, 79), (59, 17), (78, 30), (96, 84), (99, 89),
(15, 95), (6, 46), (8, 92), (11, 97), (52, 98), (24, 36), (43, 49),
(45, 95), (17, 33), (77, 47), (80, 52), (38, 92), (79, 93), (82, 98),
(54, 36), (73, 49), (75, 95), (88, 34), (1, 65), (19, 55), (10, 6),
(51, 7), (12, 52), (31, 65), (72, 66), (47, 9), (46, 50), (65, 63),
(49, 55), (68, 68), (40, 6), (81, 7), (42, 52), (74, 4), (93, 17),
(95, 63), (67, 1), (70, 6), (14, 74), (5, 25), (7, 71), (26, 84), (20,
10), (39, 23), (23, 15), (0, 68), (41, 69), (60, 82), (13, 7), (32,
20), (44, 74), (16, 12), (35, 25), (76, 26), (34, 66), (53, 79), (37,
71), (56, 84), (97, 85), (9, 9), (50, 10), (69, 23), (71, 69), (90,
82), (62, 20), (6, 88), (96, 18), (99, 23), (2, 90), (15, 29), (18,
34), (92, 20), (36, 88), (8, 26), (27, 39), (11, 31), (30, 44), (29,
85), (48, 98), (4, 28), (45, 29), (64, 42), (25, 87), (66, 88), (38,
26), (57, 39), (98, 40), (59, 85), (78, 98), (91, 37), (75, 29), (94,
42), (10, 48), (84, 34), (87, 39), (3, 45), (22, 58), (40, 48), (72,
0), (33, 45), (52, 58), (93, 59), (68, 2), (67, 43), (86, 56), (70,
48), (89, 61), (5, 67), (79, 53), (63, 45), (82, 58), (14, 8), (32,
62), (51, 75), (35, 67), (7, 5), (26, 18), (9, 51), (28, 64), (47,
77), (0, 2), (41, 3), (60, 16), (44, 8), (21, 61), (62, 62), (81, 75),
(34, 0), (53, 13), (37, 5), (56, 18), (97, 19), (74, 72), (71, 3),
(90, 16), (6, 22), (92, 62), (83, 13), (27, 81), (30, 86), (2, 24),
(20, 78), (39, 91), (4, 70), (23, 83), (36, 22), (55, 35), (13, 75),
(16, 80), (57, 81), (76, 94), (29, 19), (48, 32), (50, 78), (69, 91),
```

```
(25, 21), (66, 22), (85, 35), (84, 76), (1, 41), (87, 81), (59, 19),
(78, 32), (96, 86), (99, 91), (15, 97), (31, 41), (8, 94), (11, 99),
(24, 38), (43, 51), (45, 97), (17, 35), (58, 36), (77, 49), (61, 41),
(80, 54), (38, 94), (79, 95), (54, 38), (73, 51), (75, 97), (88, 36),
(5, 1), (26, 60), (0, 44), (19, 57), (51, 9), (35, 1), (12, 54), (53,
55), (72, 68), (56, 60), (47, 11), (46, 52), (65, 65), (49, 57), (68,
70), (81, 9), (42, 54), (83, 55), (74, 6), (95, 65), (14, 76), (27,
15), (30, 20), (7, 73), (48, 74), (20, 12), (39, 25), (4, 4), (23,
17), (41, 71), (60, 84), (13, 9), (25, 63), (44, 76), (16, 14), (57,
15), (76, 28), (34, 68), (37, 73), (78, 74), (97, 87), (50, 12), (69,
25), (71, 71), (90, 84), (84, 10), (87, 15), (6, 90), (22, 34), (96,
20), (99, 25), (2, 92), (43, 93), (15, 31), (18, 36), (17, 77), (36,
90), (8, 28), (11, 33), (52, 34), (29, 87), (73, 93), (45, 31), (64,
44), (66, 90), (38, 28), (79, 29), (98, 42), (82, 34), (59, 87), (91,
39), (75, 31), (94, 44), (10, 50), (1, 1), (12, 96), (3, 47), (47,
53), (21, 37), (40, 50), (72, 2), (42, 96), (33, 47), (74, 48), (93,
61), (68, 4), (67, 45), (86, 58), (70, 50), (89, 63), (5, 69), (63,
47), (14, 10), (13, 51), (32, 64), (51, 77), (16, 56), (35, 69), (7,
7), (26, 20), (9, 53), (28, 66), (69, 67), (41, 5), (60, 18), (44,
10), (62, 64), (81, 77), (34, 2), (37, 7), (56, 20), (97, 21), (96,
62), (99, 67), (71, 5), (90, 18), (6, 24), (92, 64), (8, 70), (27,
83), (30, 88), (2, 26), (20, 80), (4, 72), (23, 85), (36, 24), (55,
37), (38, 70), (57, 83), (76, 96), (29, 21), (48, 34), (50, 80), (91,
81), (94, 86), (66, 24), (85, 37), (84, 78), (1, 43), (87, 83), (59,
21), (15, 99), (31, 43), (24, 40), (43, 53), (45, 99), (17, 37), (58,
38), (77, 51), (61, 43), (80, 56), (79, 97), (63, 89), (54, 40), (73,
53), (75, 99), (88, 38), (5, 3), (26, 62), (0, 46), (19, 59), (51,
11), (35, 3), (12, 56), (53, 57), (72, 70), (56, 62), (28, 0), (47,
13), (46, 54), (65, 67), (49, 59), (68, 72), (81, 11), (42, 56), (83,
57), (74, 8), (95, 67), (14, 78), (27, 17), (30, 22), (7, 75), (48,
76), (20, 14), (39, 27), (4, 6), (23, 19), (41, 73), (60, 86), (13,
11), (25, 65), (44, 78), (16, 16), (57, 17), (76, 30), (34, 70), (37,
75), (78, 76), (97, 89), (50, 14), (69, 27), (71, 73), (90, 86), (84,
12), (87, 17), (6, 92), (22, 36), (96, 22), (99, 27), (2, 94), (43,
95), (15, 33), (18, 38), (17, 79), (36, 92), (8, 30), (11, 35), (52,
36), (29, 89), (73, 95), (45, 33), (64, 46), (66, 92), (38, 30), (79,
31), (98, 44), (82, 36), (59, 89), (91, 41), (75, 33), (94, 46), (10,
52), (12, 98), (3, 49), (47, 55), (21, 39), (40, 52), (72, 4), (42,
98), (33, 49), (74, 50), (93, 63), (65, 1), (68, 6), (67, 47), (86,
60), (70, 52), (89, 65), (5, 71), (63, 49), (95, 1), (39, 69), (14,
12), (13, 53), (32, 66), (51, 79), (16, 58), (35, 71), (7, 9), (48,
10), (9, 55), (28, 68), (69, 69), (41, 7), (60, 20), (44, 12), (62,
66), (81, 79), (34, 4), (37, 9), (78, 10), (97, 23), (96, 64), (99,
69), (71, 7), (90, 20), (6, 26), (92, 66), (8, 72), (27, 85), (11,
77), (30, 90), (2, 28), (43, 29), (20, 82), (4, 74), (23, 87), (64,
88), (17, 13), (36, 26), (55, 39), (38, 72), (57, 85), (76, 98), (29,
23), (73, 29), (50, 82), (91, 83), (94, 88), (66, 26), (85, 39), (84,
80), (1, 45), (87, 85), (59, 23), (3, 91), (12, 32), (31, 45), (33,
91), (24, 42), (65, 43), (68, 48), (58, 40), (77, 53), (42, 32), (61,
45), (80, 58), (79, 99), (63, 91), (54, 42), (95, 43), (88, 40), (5,
```

```
5), (7, 51), (26, 64), (39, 3), (0, 48), (19, 61), (60, 62), (32, 0),
(51, 13), (35, 5), (34, 46), (53, 59), (72, 72), (37, 51), (56, 64),
(28, 2), (69, 3), (46, 56), (49, 61), (90, 62), (62, 0), (81, 13),
(83, 59), (99, 3), (2, 70), (92, 0), (14, 80), (55, 81), (8, 6), (27,
19), (11, 11), (30, 24), (29, 65), (48, 78), (20, 16), (4, 8), (23,
21), (64, 22), (41, 75), (25, 67), (44, 80), (85, 81), (38, 6), (57,
19), (76, 32), (59, 65), (78, 78), (97, 91), (50, 16), (91, 17), (94,
22), (71, 75), (84, 14), (87, 19), (6, 94), (22, 38), (24, 84), (43,
97), (15, 35), (18, 40), (17, 81), (36, 94), (77, 95), (33, 25), (52,
38), (54, 84), (73, 97), (45, 35), (86, 36), (89, 41), (66, 94), (79,
33), (98, 46), (63, 25), (82, 38), (75, 35), (10, 54), (47, 57), (46,
98), (21, 41), (40, 54), (72, 6), (74, 52), (93, 65), (65, 3), (68,
8), (67, 49), (70, 54), (5, 73), (39, 71), (14, 14), (13, 55), (32,
68), (51, 81), (16, 60), (35, 73), (7, 11), (48, 12), (9, 57), (50,
58), (69, 71), (41, 9), (60, 22), (25, 1), (44, 14), (62, 68), (34,
6), (37, 11), (78, 12), (97, 25), (96, 66), (99, 71), (71, 9), (90,
22), (6, 28), (92, 68), (8, 74), (27, 87), (11, 79), (30, 92), (2,
30), (43, 31), (4, 76), (23, 89), (64, 90), (17, 15), (36, 28), (55,
41), (38, 74), (57, 87), (29, 25), (73, 31), (91, 85), (94, 90), (66,
28), (85, 41), (84, 82), (1, 47), (87, 87), (59, 25), (3, 93), (12,
34), (31, 47), (33, 93), (24, 44), (65, 45), (68, 50), (58, 42), (77,
55), (42, 34), (61, 47), (80, 60), (63, 93), (54, 44), (95, 45), (89,
1), (88, 42), (5, 7), (7, 53), (26, 66), (39, 5), (0, 50), (19, 63),
(60, 64), (32, 2), (51, 15), (35, 7), (34, 48), (53, 61), (72, 74),
(37, 53), (56, 66), (28, 4), (69, 5), (46, 58), (49, 63), (90, 64),
(62, 2), (81, 15), (83, 61), (96, 0), (99, 5), (2, 72), (92, 2), (14,
82), (55, 83), (8, 8), (27, 21), (11, 13), (30, 26), (29, 67), (48,
80), (20, 18), (4, 10), (23, 23), (64, 24), (41, 77), (25, 69), (44,
82), (85, 83), (38, 8), (57, 21), (76, 34), (59, 67), (78, 80), (97,
93), (50, 18), (91, 19), (94, 24), (71, 77), (84, 16), (87, 21), (6,
96), (3, 27), (22, 40), (24, 86), (43, 99), (15, 37), (18, 42), (17,
83), (36, 96), (77, 97), (33, 27), (52, 40), (54, 86), (73, 99), (45,
37), (86, 38), (89, 43), (66, 96), (79, 35), (98, 48), (63, 27), (82,
40), (75, 37), (10, 56), (51, 57), (26, 0), (28, 46), (47, 59), (21,
43), (40, 56), (81, 57), (72, 8), (56, 0), (74, 54), (93, 67), (67,
51), (70, 56), (5, 75), (2, 6), (20, 60), (39, 73), (23, 65), (14,
16), (55, 17), (13, 57), (32, 70), (16, 62), (35, 75), (76, 76), (29,
1), (48, 14), (9, 59), (50, 60), (69, 73), (41, 11), (25, 3), (44,
16), (85, 17), (62, 70), (59, 1), (78, 14), (97, 27), (96, 68), (99,
73), (71, 11), (15, 79), (6, 30), (18, 84), (92, 70), (8, 76), (27,
89), (11, 81), (30, 94), (24, 20), (43, 33), (4, 78), (45, 79), (64,
92), (17, 17), (36, 30), (77, 31), (80, 36), (38, 76), (57, 89), (98,
90), (54, 20), (73, 33), (91, 87), (75, 79), (94, 92), (66, 30), (10,
98), (84, 84), (1, 49), (87, 89), (3, 95), (19, 39), (40, 98), (12,
36), (31, 49), (72, 50), (33, 95), (46, 34), (65, 47), (49, 39), (68,
52), (67, 93), (58, 44), (70, 98), (42, 36), (61, 49), (93, 1), (63,
95), (95, 47), (14, 58), (88, 44), (5, 9), (7, 55), (26, 68), (39, 7),
(0, 52), (41, 53), (60, 66), (32, 4), (44, 58), (35, 9), (34, 50),
(53, 63), (37, 55), (56, 68), (97, 69), (69, 7), (71, 53), (90, 66),
(62, 4), (6, 72), (83, 63), (96, 2), (99, 7), (2, 74), (92, 4), (36,
```

```
72), (55, 85), (8, 10), (27, 23), (11, 15), (30, 28), (29, 69), (48,
82), (4, 12), (64, 26), (25, 71), (66, 72), (85, 85), (38, 10), (57,
23), (98, 24), (59, 69), (78, 82), (91, 21), (75, 13), (94, 26), (84,
18), (87, 23), (31, 91), (3, 29), (22, 42), (24, 88), (15, 39), (18,
44), (17, 85), (58, 86), (77, 99), (61, 91), (33, 29), (52, 42), (54,
88), (67, 27), (86, 40), (89, 45), (88, 86), (79, 37), (63, 29), (82,
42), (0, 94), (10, 58), (51, 59), (26, 2), (28, 48), (47, 61), (60,
0), (21, 45), (40, 58), (81, 59), (72, 10), (56, 2), (74, 56), (93,
69), (90, 0), (92, 46), (5, 77), (2, 8), (20, 62), (39, 75), (23, 67),
(14, 18), (55, 19), (13, 59), (32, 72), (16, 64), (35, 77), (76, 78),
(29, 3), (48, 16), (9, 61), (50, 62), (69, 75), (41, 13), (25, 5),
(44, 18), (85, 19), (59, 3), (78, 16), (97, 29), (96, 70), (99, 75),
(71, 13), (15, 81), (6, 32), (18, 86), (8, 78), (27, 91), (11, 83),
(30, 96), (24, 22), (43, 35), (4, 80), (45, 81), (64, 94), (17, 19),
(36, 32), (77, 33), (80, 38), (38, 78), (57, 91), (98, 92), (54, 22),
(73, 35), (91, 89), (75, 81), (94, 94), (66, 32), (1, 51), (87, 91),
(3, 97), (19, 41), (12, 38), (31, 51), (72, 52), (33, 97), (46, 36),
(65, 49), (49, 41), (68, 54), (67, 95), (58, 46), (42, 38), (61, 51),
(93, 3), (63, 97), (95, 49), (14, 60), (88, 46), (5, 11), (7, 57),
(26, 70), (39, 9), (23, 1), (0, 54), (41, 55), (60, 68), (32, 6), (44,
60), (35, 11), (76, 12), (34, 52), (53, 65), (37, 57), (56, 70), (97,
71), (69, 9), (71, 55), (90, 68), (62, 6), (6, 74), (83, 65), (96, 4),
(99, 9), (2, 76), (15, 15), (18, 20), (92, 6), (36, 74), (55, 87), (8,
12), (27, 25), (11, 17), (30, 30), (29, 71), (48, 84), (4, 14), (45,
15), (64, 28), (25, 73), (66, 74), (85, 87), (38, 12), (57, 25), (98,
26), (1, 93), (59, 71), (78, 84), (91, 23), (75, 15), (94, 28), (10,
34), (84, 20), (87, 25), (31, 93), (3, 31), (22, 44), (24, 90), (40,
34), (17, 87), (58, 88), (61, 93), (33, 31), (52, 44), (93, 45), (54,
90), (67, 29), (86, 42), (70, 34), (89, 47), (88, 88), (5, 53), (79,
39), (63, 31), (82, 44), (0, 96), (32, 48), (51, 61), (35, 53), (26,
4), (9, 37), (28, 50), (47, 63), (60, 2), (21, 47), (62, 48), (81,
61), (56, 4), (97, 5), (74, 58), (90, 2), (6, 8), (92, 48), (27, 67),
(30, 72), (2, 10), (20, 64), (39, 77), (4, 56), (23, 69), (36, 8),
(55, 21), (13, 61), (16, 66), (57, 67), (76, 80), (29, 5), (48, 18),
(50, 64), (69, 77), (25, 7), (66, 8), (85, 21), (84, 62), (1, 27),
(87, 67), (59, 5), (78, 18), (22, 86), (96, 72), (99, 77), (15, 83),
(18, 88), (31, 27), (8, 80), (11, 85), (52, 86), (24, 24), (43, 37),
(45, 83), (64, 96), (17, 21), (58, 22), (77, 35), (61, 27), (80, 40),
(38, 80), (79, 81), (98, 94), (82, 86), (54, 24), (73, 37), (91, 91),
(75, 83), (94, 96), (88, 22), (3, 99), (0, 30), (19, 43), (21, 89),
(12, 40), (53, 41), (72, 54), (56, 46), (33, 99), (46, 38), (65, 51),
(49, 43), (68, 56), (67, 97), (42, 40), (83, 41), (63, 99), (95, 51),
(14, 62), (5, 13), (7, 59), (48, 60), (39, 11), (23, 3), (41, 57),
(60, 70), (25, 49), (44, 62), (16, 0), (35, 13), (76, 14), (34, 54),
(37, 59), (78, 60), (97, 73), (69, 11), (71, 57), (90, 70), (6, 76),
(96, 6), (99, 11), (2, 78), (15, 17), (18, 22), (17, 63), (36, 76),
(55, 89), (8, 14), (27, 27), (11, 19), (30, 32), (29, 73), (73, 79),
(4, 16), (45, 17), (64, 30), (66, 76), (85, 89), (38, 14), (57, 27),
(98, 28), (1, 95), (59, 73), (91, 25), (75, 17), (94, 30), (10, 36),
(31, 95), (3, 33), (22, 46), (24, 92), (40, 36), (58, 90), (42, 82),
```

```
(61, 95), (33, 33), (52, 46), (93, 47), (54, 92), (95, 93), (67, 31),
(86, 44), (70, 36), (89, 49), (88, 90), (5, 55), (63, 33), (82, 46),
(0, 98), (32, 50), (51, 63), (35, 55), (26, 6), (9, 39), (28, 52),
(47, 65), (60, 4), (21, 49), (62, 50), (81, 63), (53, 1), (56, 6),
(97, 7), (74, 60), (90, 4), (6, 10), (92, 50), (83, 1), (27, 69), (30,
74), (2, 12), (20, 66), (39, 79), (4, 58), (23, 71), (36, 10), (55,
23), (13, 63), (16, 68), (57, 69), (76, 82), (29, 7), (48, 20), (50,
66), (69, 79), (25, 9), (66, 10), (85, 23), (84, 64), (1, 29), (87,
69), (59, 7), (78, 20), (22, 88), (96, 74), (99, 79), (15, 85), (18,
90), (31, 29), (8, 82), (11, 87), (52, 88), (24, 26), (43, 39), (45,
85), (64, 98), (17, 23), (58, 24), (77, 37), (61, 29), (80, 42), (38,
82), (79, 83), (98, 96), (82, 88), (54, 26), (73, 39), (91, 93), (75,
85), (94, 98), (88, 24), (26, 48), (0, 32), (19, 45), (21, 91), (12,
42), (53, 43), (72, 56), (56, 48), (46, 40), (65, 53), (49, 45), (68,
58), (67, 99), (42, 42), (83, 43), (95, 53), (14, 64), (27, 3), (30,
8), (7, 61), (48, 62), (20, 0), (39, 13), (23, 5), (41, 59), (60, 72),
(25, 51), (44, 64), (16, 2), (57, 3), (76, 16), (34, 56), (37, 61),
(78, 62), (97, 75), (50, 0), (69, 13), (71, 59), (90, 72), (87, 3),
(6, 78), (22, 22), (96, 8), (99, 13), (2, 80), (43, 81), (15, 19),
(18, 24), (17, 65), (36, 78), (55, 91), (8, 16), (11, 21), (52, 22),
(29, 75), (73, 81), (45, 19), (64, 32), (66, 78), (85, 91), (38, 16),
(79, 17), (98, 30), (82, 22), (1, 97), (59, 75), (91, 27), (75, 19),
(94, 32), (10, 38), (12, 84), (31, 97), (3, 35), (47, 41), (24, 94),
(65, 95), (21, 25), (40, 38), (58, 92), (42, 84), (61, 97), (33, 35),
(74, 36), (93, 49), (54, 94), (95, 95), (67, 33), (86, 46), (70, 38),
(89, 51), (88, 92), (5, 57), (63, 35), (39, 55), (13, 39), (32, 52),
(51, 65), (16, 44), (35, 57), (34, 98), (9, 41), (28, 54), (69, 55),
(60, 6), (62, 52), (81, 65), (97, 9), (96, 50), (99, 55), (90, 6), (6,
12), (92, 52), (8, 58), (27, 71), (11, 63), (30, 76), (2, 14), (43,
15), (20, 68), (4, 60), (23, 73), (64, 74), (36, 12), (55, 25), (38,
58), (57, 71), (76, 84), (29, 9), (73, 15), (50, 68), (91, 69), (94,
74), (66, 12), (85, 25), (84, 66), (1, 31), (87, 71), (59, 9), (3,
77), (22, 90), (15, 87), (18, 92), (12, 18), (31, 31), (33, 77), (52,
90), (24, 28), (65, 29), (45, 87), (86, 88), (58, 26), (77, 39), (89,
93), (42, 18), (61, 31), (80, 44), (79, 85), (98, 98), (63, 77), (82,
90), (54, 28), (95, 29), (75, 87), (88, 26), (26, 50), (28, 96), (0,
34), (19, 47), (21, 93), (34, 32), (53, 45), (72, 58), (37, 37), (56,
50), (47, 1), (46, 42), (49, 47), (90, 48), (83, 45), (14, 66), (27,
5), (30, 10), (48, 64), (20, 2), (39, 15), (23, 7), (41, 61), (25,
53), (44, 66), (16, 4), (57, 5), (76, 18), (59, 51), (78, 64), (97,
77), (50, 2), (69, 15), (71, 61), (84, 0), (87, 5), (6, 80), (22, 24),
(99, 15), (2, 82), (43, 83), (15, 21), (18, 26), (17, 67), (36, 80),
(55, 93), (8, 18), (11, 23), (52, 24), (73, 83), (45, 21), (64, 34),
(66, 80), (38, 18), (79, 19), (98, 32), (82, 24), (1, 99), (91, 29),
(75, 21), (94, 34), (10, 40), (12, 86), (31, 99), (3, 37), (47, 43),
(24, 96), (65, 97), (21, 27), (40, 40), (58, 94), (42, 86), (61, 99),
(33, 37), (74, 38), (93, 51), (95, 97), (67, 35), (86, 48), (70, 40),
(89, 53), (88, 94), (5, 59), (63, 37), (39, 57), (14, 0), (13, 41),
(32, 54), (51, 67), (16, 46), (35, 59), (9, 43), (28, 56), (69, 57),
(60, 8), (44, 0), (62, 54), (81, 67), (97, 11), (96, 52), (99, 57),
```

```
(90, 8), (6, 14), (92, 54), (8, 60), (27, 73), (11, 65), (30, 78), (2,
16), (43, 17), (20, 70), (4, 62), (23, 75), (64, 76), (17, 1), (36,
14), (55, 27), (38, 60), (57, 73), (76, 86), (29, 11), (73, 17), (50,
70), (91, 71), (94, 76), (66, 14), (85, 27), (84, 68), (1, 33), (87,
73), (59, 11), (3, 79), (22, 92), (15, 89), (18, 94), (12, 20), (31,
33), (33, 79), (52, 92), (24, 30), (65, 31), (68, 36), (45, 89), (86,
90), (58, 28), (77, 41), (89, 95), (42, 20), (61, 33), (80, 46), (79,
87), (63, 79), (82, 92), (54, 30), (95, 31), (75, 89), (88, 28), (7,
39), (26, 52), (28, 98), (0, 36), (19, 49), (60, 50), (51, 1), (21,
95), (34, 34), (53, 47), (72, 60), (37, 39), (56, 52), (46, 44), (49,
49), (90, 50), (81, 1), (83, 47), (2, 58), (14, 68), (55, 69), (27,
7), (30, 12), (29, 53), (48, 66), (20, 4), (23, 9), (64, 10), (41,
63), (25, 55), (44, 68), (85, 69), (57, 7), (76, 20), (59, 53), (78,
66), (97, 79), (50, 4), (91, 5), (94, 10), (71, 63), (84, 2), (87, 7),
(6, 82), (3, 13), (22, 26), (24, 72), (43, 85), (15, 23), (18, 28),
(17, 69), (36, 82), (77, 83), (80, 88), (33, 13), (52, 26), (54, 72),
(73, 85), (45, 23), (86, 24), (89, 29), (66, 82), (79, 21), (98, 34),
(63, 13), (82, 26), (75, 23), (19, 91), (10, 42), (51, 43), (12, 88),
(28, 32), (47, 45), (46, 86), (65, 99), (49, 91), (21, 29), (40, 42),
(81, 43), (58, 96), (42, 88), (74, 40), (93, 53), (95, 99), (67, 37),
(70, 42), (88, 96), (5, 61), (20, 46), (39, 59), (23, 51), (14, 2),
(13, 43), (32, 56), (16, 48), (35, 61), (76, 62), (48, 0), (9, 45),
(50, 46), (69, 59), (44, 2), (85, 3), (62, 56), (78, 0), (97, 13),
(96, 54), (99, 59), (15, 65), (6, 16), (18, 70), (92, 56), (8, 62),
(27, 75), (11, 67), (30, 80), (2, 18), (43, 19), (4, 64), (45, 65),
(64, 78), (17, 3), (36, 16), (38, 62), (57, 75), (98, 76), (54, 6),
(73, 19), (91, 73), (75, 65), (94, 78), (66, 16), (84, 70), (1, 35),
(87, 75), (3, 81), (22, 94), (40, 84), (12, 22), (31, 35), (33, 81),
(52, 94), (93, 95), (65, 33), (68, 38), (67, 79), (86, 92), (58, 30),
(70, 84), (89, 97), (42, 22), (61, 35), (80, 48), (79, 89), (63, 81),
(82, 94), (95, 33), (88, 30), (32, 98), (7, 41), (26, 54), (9, 87),
(0, 38), (19, 51), (60, 52), (51, 3), (21, 97), (62, 98), (34, 36),
(53, 49), (72, 62), (37, 41), (56, 54), (90, 52), (81, 3), (92, 98),
(83, 49), (2, 60), (14, 70), (55, 71), (27, 9), (11, 1), (30, 14),
(29, 55), (48, 68), (20, 6), (23, 11), (64, 12), (41, 65), (25, 57),
(44, 70), (85, 71), (57, 9), (76, 22), (59, 55), (78, 68), (97, 81),
(50, 6), (91, 7), (94, 12), (84, 4), (87, 9), (6, 84), (3, 15), (22,
28), (24, 74), (43, 87), (15, 25), (18, 30), (17, 71), (36, 84), (77,
85), (80, 90), (33, 15), (52, 28), (54, 74), (73, 87), (45, 25), (86,
26), (89, 31), (66, 84), (79, 23), (98, 36), (63, 15), (82, 28), (75,
25), (19, 93), (10, 44), (51, 45), (12, 90), (28, 34), (47, 47), (46,
88), (49, 93), (21, 31), (40, 44), (81, 45), (58, 98), (42, 90), (74,
42), (93, 55), (67, 39), (70, 44), (88, 98), (5, 63), (20, 48), (39,
61), (23, 53), (14, 4), (55, 5), (13, 45), (32, 58), (16, 50), (35,
63), (76, 64), (48, 2), (9, 47), (50, 48), (69, 61), (44, 4), (85, 5),
(62, 58), (78, 2), (97, 15), (96, 56), (99, 61), (15, 67), (6, 18),
(18, 72), (92, 58), (8, 64), (27, 77), (11, 69), (30, 82), (24, 8),
(43, 21), (4, 66), (45, 67), (64, 80), (17, 5), (36, 18), (77, 19),
(80, 24), (38, 64), (57, 77), (98, 78), (54, 8), (73, 21), (91, 75),
(75, 67), (94, 80), (66, 18), (10, 86), (84, 72), (1, 37), (87, 77),
```

```
(3, 83), (22, 96), (19, 27), (40, 86), (12, 24), (31, 37), (72, 38),
(33, 83), (52, 96), (93, 97), (46, 22), (65, 35), (49, 27), (68, 40),
(67, 81), (86, 94), (58, 32), (70, 86), (89, 99), (42, 24), (61, 37),
(79, 91), (63, 83), (82, 96), (95, 35), (14, 46), (88, 32), (7, 43),
(26, 56), (9, 89), (0, 40), (41, 41), (60, 54), (44, 46), (21, 99),
(34, 38), (53, 51), (37, 43), (56, 56), (97, 57), (71, 41), (90, 54),
(6, 60), (83, 51), (2, 62), (15, 1), (18, 6), (36, 60), (55, 73), (27,
11), (11, 3), (30, 16), (29, 57), (48, 70), (4, 0), (45, 1), (64, 14),
(25, 59), (66, 60), (85, 73), (57, 11), (98, 12), (1, 79), (59, 57),
(78, 70), (91, 9), (75, 1), (94, 14), (10, 20), (84, 6), (87, 11),
(31, 79), (3, 17), (22, 30), (24, 76), (43, 89), (40, 20), (17, 73),
(58, 74), (77, 87), (61, 79), (80, 92), (33, 17), (52, 30), (93, 31),
(54, 76), (73, 89), (67, 15), (86, 28), (70, 20), (89, 33), (88, 74),
(79, 25), (63, 17), (82, 30), (26, 98), (0, 82), (19, 95), (32, 34),
(51, 47), (35, 39), (12, 92), (53, 93), (56, 98), (9, 23), (28, 36),
(47, 49), (46, 90), (49, 95), (21, 33), (62, 34), (81, 47), (42, 92),
(83, 93), (74, 44), (92, 34), (20, 50), (39, 63), (4, 42), (23, 55),
(14, 6), (55, 7), (13, 47), (16, 52), (57, 53), (76, 66), (48, 4),
(50, 50), (69, 63), (44, 6), (85, 7), (84, 48), (87, 53), (78, 4),
(97, 17), (96, 58), (99, 63), (15, 69), (6, 20), (18, 74), (8, 66),
(11, 71), (24, 10), (43, 23), (45, 69), (64, 82), (17, 7), (36, 20),
(77, 21), (80, 26), (38, 66), (79, 67), (98, 80), (82, 72), (54, 10),
(73, 23), (91, 77), (75, 69), (94, 82), (10, 88), (1, 39), (3, 85),
(22, 98), (19, 29), (40, 88), (12, 26), (31, 39), (72, 40), (33, 85),
(74, 86), (93, 99), (46, 24), (65, 37), (49, 29), (68, 42), (67, 83),
(86, 96), (58, 34), (70, 88), (42, 26), (61, 39), (63, 85), (95, 37),
(14, 48), (7, 45), (26, 58), (9, 91), (0, 42), (41, 43), (60, 56),
(44, 48), (76, 0), (34, 40), (53, 53), (37, 45), (56, 58), (97, 59),
(71, 43), (90, 56), (6, 62), (83, 53), (2, 64), (15, 3), (18, 8), (36,
62), (55, 75), (8, 0), (27, 13), (11, 5), (30, 18), (29, 59), (48,
72), (4, 2), (45, 3), (64, 16), (25, 61), (66, 62), (85, 75), (38, 0),
(57, 13), (98, 14), (1, 81), (59, 59), (78, 72), (91, 11), (75, 3),
(94, 16), (10, 22), (84, 8), (87, 13), (31, 81), (3, 19), (22, 32),
(24, 78), (43, 91), (40, 22), (17, 75), (58, 76), (77, 89), (61, 81),
(80, 94), (33, 19), (52, 32), (93, 33), (54, 78), (73, 91), (67, 17),
(86, 30), (70, 22), (89, 35), (88, 76), (5, 41), (79, 27), (63, 19),
(82, 32), (0, 84), (19, 97), (32, 36), (51, 49), (35, 41), (12, 94),
(53, 95), (9, 25), (28, 38), (47, 51), (46, 92), (49, 97), (21, 35),
(62, 36), (81, 49), (42, 94), (83, 95), (74, 46), (92, 36), (27, 55),
(30, 60), (20, 52), (39, 65), (4, 44), (23, 57), (55, 9), (13, 49),
(16, 54), (57, 55), (76, 68), (48, 6), (50, 52), (69, 65), (85, 9),
(84, 50), (1, 15), (87, 55), (78, 6), (22, 74), (96, 60), (99, 65),
(15, 71), (18, 76), (31, 15), (8, 68), (11, 73), (52, 74), (24, 12),
(43, 25), (45, 71), (64, 84), (17, 9), (58, 10), (77, 23), (61, 15),
(80, 28), (38, 68), (79, 69), (98, 82), (82, 74), (54, 12), (73, 25),
(91, 79), (75, 71), (94, 84), (88, 10), (10, 90), (26, 34), (3, 87),
(47, 93), (0, 18), (19, 31), (21, 77), (40, 90), (12, 28), (53, 29),
(72, 42), (56, 34), (33, 87), (74, 88), (46, 26), (65, 39), (49, 31),
(68, 44), (67, 85), (86, 98), (70, 90), (42, 28), (83, 29), (63, 87),
```

```
(95, 39), (14, 50), (13, 91), (16, 96), (7, 47), (48, 48), (9, 93),
(41, 45), (60, 58), (25, 37), (44, 50), (76, 2), (34, 42), (37, 47),
(78, 48), (97, 61), (71, 45), (90, 58), (6, 64), (2, 66), (43, 67),
(15, 5), (18, 10), (17, 51), (36, 64), (55, 77), (8, 2), (11, 7), (52,
8), (29, 61), (73, 67), (45, 5), (64, 18), (66, 64), (85, 77), (38,
2), (79, 3), (98, 16), (82, 8), (1, 83), (59, 61), (91, 13), (75, 5),
(94, 18), (10, 24), (12, 70), (31, 83), (3, 21), (24, 80), (65, 81),
(68, 86), (21, 11), (40, 24), (58, 78), (77, 91), (42, 70), (61, 83),
(80, 96), (33, 21), (74, 22), (93, 35), (54, 80), (95, 81), (67, 19),
(86, 32), (70, 24), (89, 37), (88, 78), (5, 43), (63, 21), (7, 89),
(0, 86), (19, 99), (13, 25), (32, 38), (51, 51), (35, 43), (34, 84),
(53, 97), (37, 89), (9, 27), (28, 40), (46, 94), (49, 99), (62, 38),
(81, 51), (83, 97), (96, 36), (99, 41), (92, 38), (27, 57), (30, 62),
(2, 0), (20, 54), (39, 67), (4, 46), (23, 59), (55, 11), (38, 44),
(57, 57), (76, 70), (48, 8), (50, 54), (91, 55), (85, 11), (84, 52),
(1, 17), (87, 57), (78, 8), (22, 76), (15, 73), (18, 78), (31, 17),
(11, 75), (52, 76), (24, 14), (43, 27), (45, 73), (64, 86), (17, 11),
(58, 12), (77, 25), (61, 17), (80, 30), (79, 71), (98, 84), (82, 76),
(54, 14), (73, 27), (75, 73), (88, 12), (10, 92), (26, 36), (3, 89),
(47, 95), (0, 20), (19, 33), (21, 79), (40, 92), (12, 30), (53, 31),
(72, 44), (56, 36), (33, 89), (74, 90), (46, 28), (65, 41), (49, 33),
(68, 46), (67, 87), (70, 92), (42, 30), (83, 31), (95, 41), (14, 52),
(13, 93), (16, 98), (7, 49), (48, 50), (39, 1), (9, 95), (41, 47),
(60, 60), (25, 39), (44, 52), (76, 4), (34, 44), (37, 49), (78, 50),
(97, 63), (69, 1), (71, 47), (90, 60), (6, 66), (22, 10), (99, 1), (2,
68), (43, 69), (15, 7), (18, 12), (17, 53), (36, 66), (55, 79), (8,
4), (11, 9), (52, 10), (29, 63), (73, 69), (45, 7), (64, 20), (66,
66), (85, 79), (38, 4), (79, 5), (98, 18), (82, 10), (1, 85), (59,
63), (91, 15), (75, 7), (94, 20), (10, 26), (12, 72), (31, 85), (3,
23), (47, 29), (24, 82), (65, 83), (68, 88), (21, 13), (40, 26), (58,
80), (77, 93), (42, 72), (61, 85), (80, 98), (33, 23), (74, 24), (93,
37), (54, 82), (95, 83), (67, 21), (86, 34), (70, 26), (89, 39), (88,
80), (5, 45), (63, 23), (7, 91), (39, 43), (0, 88), (13, 27), (32,
40), (51, 53), (16, 32), (35, 45), (34, 86), (53, 99), (37, 91), (9,
29), (28, 42), (69, 43), (46, 96), (62, 40), (81, 53), (83, 99), (96,
38), (99, 43), (6, 0), (92, 40), (8, 46), (27, 59), (11, 51), (30,
64), (2, 2), (43, 3), (20, 56), (4, 48), (23, 61), (64, 62), (36, 0),
(55, 13), (38, 46), (57, 59), (76, 72), (73, 3), (50, 56), (91, 57),
(94, 62), (66, 0), (85, 13), (84, 54), (1, 19), (87, 59), (3, 65),
(22, 78), (15, 75), (18, 80), (12, 6), (31, 19), (33, 65), (52, 78),
(24, 16), (65, 17), (68, 22), (45, 75), (86, 76), (58, 14), (77, 27),
(89, 81), (42, 6), (61, 19), (80, 32), (79, 73), (98, 86), (63, 65),
(82, 78), (54, 16), (95, 17), (75, 75), (88, 14), (10, 94), (51, 95),
(7, 25), (26, 38), (28, 84), (47, 97), (0, 22), (19, 35), (60, 36),
(21, 81), (40, 94), (81, 95), (34, 20), (53, 33), (72, 46), (37, 25),
(56, 38), (74, 92), (46, 30), (49, 35), (90, 36), (67, 89), (70, 94),
(83, 33), (2, 44), (20, 98), (14, 54), (55, 55), (13, 95), (29, 39),
(48, 52), (9, 97), (50, 98), (41, 49), (25, 41), (44, 54), (85, 55),
(76, 6), (59, 39), (78, 52), (97, 65), (71, 49), (6, 68), (22, 12),
```

```
(24, 58), (43, 71), (15, 9), (18, 14), (17, 55), (36, 68), (77, 69),
(80, 74), (52, 12), (54, 58), (73, 71), (45, 9), (86, 10), (66, 68),
(79, 7), (98, 20), (82, 12), (1, 87), (75, 9), (19, 77), (10, 28),
(12, 74), (31, 87), (72, 88), (3, 25), (47, 31), (46, 72), (65, 85),
(49, 77), (68, 90), (21, 15), (40, 28), (58, 82), (42, 74), (61, 87),
(74, 26), (93, 39), (95, 85), (67, 23), (70, 28), (88, 82), (5, 47),
(7, 93), (39, 45), (0, 90), (41, 91), (13, 29), (32, 42), (51, 55),
(16, 34), (35, 47), (34, 88), (37, 93), (9, 31), (28, 44), (69, 45),
(71, 91), (62, 42), (81, 55), (96, 40), (99, 45), (6, 2), (92, 42),
(8, 48), (27, 61), (11, 53), (30, 66), (2, 4), (43, 5), (20, 58), (4,
50), (23, 63), (64, 64), (36, 2), (55, 15), (38, 48), (57, 61), (76,
74), (73, 5), (91, 59), (94, 64), (66, 2), (85, 15), (84, 56), (1,
21), (87, 61), (3, 67), (22, 80), (15, 77), (18, 82), (12, 8), (31,
21), (33, 67), (52, 80), (24, 18), (65, 19), (68, 24), (45, 77), (86,
78), (58, 16), (77, 29), (89, 83), (42, 8), (61, 21), (80, 34), (79,
75), (98, 88), (63, 67), (82, 80), (54, 18), (95, 19), (75, 77), (88,
16), (10, 96), (51, 97), (7, 27), (26, 40), (28, 86), (47, 99), (0,
24), (19, 37), (60, 38), (21, 83), (40, 96), (81, 97), (34, 22), (53,
35), (72, 48), (37, 27), (56, 40), (74, 94), (46, 32), (49, 37), (90,
38), (67, 91), (70, 96), (83, 35), (2, 46), (14, 56), (55, 57), (13,
97), (30, 0), (29, 41), (48, 54), (9, 99), (41, 51), (25, 43), (44,
56), (85, 57), (76, 8), (59, 41), (78, 54), (97, 67), (71, 51), (6,
70), (3, 1), (22, 14), (24, 60), (43, 73), (15, 11), (18, 16), (17,
57), (36, 70), (77, 71), (80, 76), (33, 1), (52, 14), (54, 60), (73,
73), (45, 11), (86, 12), (89, 17), (66, 70), (79, 9), (98, 22), (63,
1), (82, 14), (1, 89), (75, 11), (19, 79), (10, 30), (51, 31), (12,
76), (31, 89), (72, 90), (28, 20), (47, 33), (46, 74), (65, 87), (49,
79), (68, 92), (21, 17), (40, 30), (81, 31), (58, 84), (42, 76), (61,
89), (74, 28), (93, 41), (95, 87), (67, 25), (70, 30), (14, 98), (88,
84), (5, 49), (7, 95), (20, 34), (39, 47), (23, 39), (0, 92), (41,
93), (13, 31), (32, 44), (44, 98), (16, 36), (35, 49), (76, 50), (34,
90), (37, 95), (9, 33), (50, 34), (69, 47), (71, 93), (62, 44), (97,
1), (96, 42), (99, 47), (15, 53), (6, 4), (18, 58), (92, 44), (8, 50),
(27, 63), (11, 55), (30, 68), (43, 7), (4, 52), (45, 53), (64, 66),
(36, 4), (77, 5), (80, 10), (38, 50), (57, 63), (98, 64), (73, 7),
(91, 61), (75, 53), (94, 66), (66, 4), (10, 72), (84, 58), (1, 23),
(87, 63), (3, 69), (22, 82), (19, 13), (40, 72), (12, 10), (31, 23),
(72, 24), (33, 69), (52, 82), (93, 83), (46, 8), (65, 21), (49, 13),
(68, 26), (67, 67), (86, 80), (58, 18), (70, 72), (89, 85), (42, 10),
(61, 23), (5, 91), (79, 77), (63, 69), (82, 82), (95, 21), (88, 18),
(32, 86), (51, 99), (35, 91), (7, 29), (26, 42), (9, 75), (28, 88),
(0, 26), (41, 27), (60, 40), (44, 32), (21, 85), (62, 86), (81, 99),
(34, 24), (53, 37), (37, 29), (56, 42), (97, 43), (74, 96), (71, 27),
(90, 40), (92, 86), (83, 37), (2, 48), (4, 94), (36, 46), (55, 59),
(13, 99), (30, 2), (29, 43), (48, 56), (64, 0), (25, 45), (66, 46),
(85, 59), (76, 10), (59, 43), (78, 56), (94, 0), (3, 3), (22, 16),
(24, 62), (43, 75), (15, 13), (18, 18), (17, 59), (58, 60), (77, 73),
(61, 65), (80, 78), (33, 3), (52, 16), (54, 62), (73, 75), (45, 13),
(86, 14), (89, 19), (88, 60), (79, 11), (63, 3), (82, 16), (1, 91),
```

```
(19, 81), (10, 32), (51, 33), (12, 78), (72, 92), (28, 22), (47, 35),
(46, 76), (65, 89), (49, 81), (68, 94), (21, 19), (40, 32), (81, 33),
(42, 78), (83, 79), (74, 30), (93, 43), (95, 89), (70, 32), (5, 51),
(7, 97), (20, 36), (39, 49), (23, 41), (41, 95), (13, 33), (32, 46),
(16, 38), (35, 51), (76, 52), (34, 92), (37, 97), (9, 35), (50, 36),
(69, 49), (71, 95), (62, 46), (97, 3), (96, 44), (99, 49), (15, 55),
(6, 6), (18, 60), (8, 52), (27, 65), (11, 57), (30, 70), (43, 9), (4,
54), (45, 55), (64, 68), (36, 6), (77, 7), (80, 12), (38, 52), (57,
65), (98, 66), (73, 9), (91, 63), (75, 55), (94, 68), (66, 6), (10,
74), (84, 60), (1, 25), (87, 65), (3, 71), (22, 84), (19, 15), (40,
74), (12, 12), (31, 25), (72, 26), (33, 71), (52, 84), (93, 85), (46,
10), (65, 23), (49, 15), (68, 28), (67, 69), (86, 82), (58, 20), (70,
74), (89, 87), (42, 12), (61, 25), (5, 93), (79, 79), (63, 71), (82,
84), (95, 23), (14, 34), (88, 20), (32, 88), (35, 93), (7, 31), (26,
44), (9, 77), (28, 90), (0, 28), (41, 29), (60, 42), (44, 34), (21,
87), (62, 88), (34, 26), (53, 39), (37, 31), (56, 44), (97, 45), (74,
98)} em {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,
18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34,
35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51,
52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68,
69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85,
86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99} é equivalente?
True.
Relação = {(1, 1), (0, 0)} em {0, 1} é equivalente? True.
```

Slide 31 - Funções

Slide 34 - Definição

Uma função $f: A \to B$ é uma relação de A para B (um subconjunto de $A \times B$) tal que cada $a \in A$ está na primeira posição de um único par ordenado (a, b) em f. O gráfico de f é definido como:

$$\text{Gráfico de } f = \{(a, b) : a \in A, b = f(a)\}$$

Exemplos de Relações e Funções

Dado o conjunto $A = \{1, 2, 3\}$, vamos verificar quais das seguintes relações são funções:

1. $f = \{(1,3), (2,3), (3,1)\}$
   - f é uma função de A em A porque cada elemento de A aparece exatamente uma vez como o primeiro elemento de um par ordenado em f.
   - **Prova**: Observamos que:
     - 1, 2, e 3 são únicos na primeira posição de cada par, e
     - Cada um desses elementos é mapeado para um único valor em A.
   - Portanto, f satisfaz a definição de função.
2. $g = \{(1,2), (3,1)\}$
   - g não é uma função de A para A porque o elemento $2 \in A$ não aparece como o primeiro elemento de nenhum par ordenado em g.

- **Prova**: Para que g fosse uma função, cada elemento de A deveria aparecer como o primeiro elemento em um par único, o que não ocorre aqui.

3. $h = \{(1,3),(2,1),(1,2),(3,1)\}$
   - h não é uma função porque o elemento $1 \in A$ aparece como o primeiro elemento em mais de um par ordenado, o que viola a definição de função.
   - **Prova**: Uma função requer que cada elemento de A esteja relacionado a um único elemento em B. Aqui, 1 está relacionado a 3 e também a 2, o que não é permitido em funções.

Concluindo, apenas f é uma função no conjunto $A=\{1,2,3\}$, enquanto g e h não são funções devido à violação dos critérios de definição de função.

22. Relação de Funções: Dado o conjunto A = {1,2,3}, verifique as funções:

1. f = {(1,3), (2, 3), (3,1)}
2. g = {(1,2), (3,1)}
3. h = {(1,3), (2,1), (1,2), (3,1)}

```
def verificar_funcao(relacao, conjunto):
  dominio = set([x for x, y in relacao])
  for elem in dominio:
        if len([y for x, y in relacao if x == elem]) > 1:
          return False
  return dominio == conjunto

A = {1, 2, 3}

f = {(1, 3), (2, 3), (3, 1)}
g = {(1, 2), (3, 1)}
h = {(1, 3), (2, 1), (1, 2), (3, 1)}

print(f'f é uma função? {verificar_funcao(f,A)}')
print(f'g é uma função? {verificar_funcao(g,A)}')
print(f'h é uma função? {verificar_funcao(h,A)}')
```

Slide 37 - Teoremas e Demonstrações

Slide 42 - Indução Matemática: exemplo 1

Objetivo: Provar que $1+3+5+\ldots+(2n-1)=n^2$ para todo inteiro positivo n.

1. **Base da Indução** $P(1)$:
   - Quando $n=1$, a soma dos primeiros n números ímpares é 1.
   - A fórmula dá $1^2=1$.
   - Portanto, $P(1)$ é verdadeira porque ambos os lados são iguais a 1.
2. **Hipótese Indutiva (Suponha** $P(k)$**)**:
   - Suponha que a fórmula é verdadeira para um certo inteiro positivo k:

$$1+3+5+\ldots+(2k-1)=k^2$$

3. **Passo Indutivo (Prove $P(k+1)$)**:
   - Precisamos provar que a fórmula também é válida para $k+1$:

     $$1+3+5+\ldots+(2k-1)=k^2$$

   - Partindo da Hipótese Indutiva, adicionamos o próximo número ímpar na sequência, $2(k+1)-1$, aos lados esquerdo e direito:

     $$1+3+5+\ldots+(2k-1)+[2(k+1)-1) \quad \text{¿}k^2+[2(k+1)-1)$$
     $$\text{¿} \qquad \text{¿}k^2+2k+1$$
     $$\text{¿} \qquad \text{¿}$$

   - O último passo segue da contração de k^2+2k+1 que é igual a $(k+1)^2$.
   - Portanto, $P(k+1)$ é verdadeira.

Dado que $P(1)$ é verdadeira e $P(k+1)$ é verdadeira assumindo que $P(k)$ é verdadeira, por indução matemática, a equação $1+3+5+\ldots+(2n-1)=n^2$ é verdadeira para todo inteiro positivo n.

Slide 43 - Indução Matemática: exemplo 2

Exemplo: Provar que para todo inteiro positivo n, a seguinte equação é verdadeira: $1+2+3+\ldots+n=\frac{n(n+1)}{2}$.

Etapas da Prova Indutiva

1. **Base da Indução**:
   - Verifique se a equação é verdadeira para $n=1$.
   - $1=\frac{1(1+1)}{2}=1$, que é verdadeiro. Assim, a base da indução é válida.
2. **Hipótese Indutiva** (Suponha $P(k)$):
   - Assuma que a fórmula é verdadeira para algum inteiro positivo k.
   - $1+2+3+\ldots+k=\frac{k(k+1)}{2}$
3. **Passo Indutivo** (Prove $P(k+1)$):
   - Precisamos mostrar que a fórmula também é verdadeira para $k+1$.
   - Começamos com $1+2+3+\ldots+k+(k+1)$.
   - Com base na hipótese indutiva, podemos reescrever isso como $\frac{k(k+1)}{2}+(k+1)$.
   - Combinando os termos, obtemos $\frac{k(k+1)}{2}+\frac{2(k+1)}{2}$.
   - Simplificando, isso se torna $\frac{k(k+1)+2(k+1)}{2}$.
   - Fatorando $(k+1)$, temos $\frac{(k+1)(k+2)}{2}$.
   - Que é igual a $\frac{(k+1)((k+1)+1)}{2}$, confirmando que a fórmula é verdadeira para $k+1$.

- Uma outra forma de ver é:

$$P(k+1): 1+2+3+\ldots+k+(k+1)=\frac{(k+1)((k+1)+1)}{2}$$

- Partimos da soma até k e adicionamos $k+1$ a ambos os lados da equação $P(k)$:

$$1+2+3+\ldots+k+(k+1)=\frac{k(k+1)}{2}+(k+1)$$

- Para unificar o lado direito da equação, encontramos um denominador comum e combinamos os termos:

$$\text{¿}\frac{k(k+1)}{2}+\frac{2(k+1)}{2}$$

- Somando as frações, temos:

$$\text{¿}\frac{k(k+1)+2(k+1)}{2}$$

- Fatorando $(k+1)$ do numerador, obtemos:

$$\text{¿}\frac{(k+1)(k+2)}{2}$$

Agora, reconhecemos que $k+2$ é o mesmo que $(k+1)+1$, portanto, a equação se torna:

$$\text{¿}\frac{(k+1)((k+1)+1)}{2}$$

Portanto, por indução matemática, provamos que $1+2+3+\ldots+n=\frac{n(n+1)}{2}$ é verdadeiro para qualquer inteiro positivo n.

23. Indução matemática: Faça um codigo em python que:

a) Provar que 1+3+5+...+(2n-1) = n² para todo inteiro positivo n

b) provar que para todo inteiro positivo n, a seguinte equação é verdadeira: 1+2+3+...+n = (n(n+1))/2

```python
def soma_impares(n):
    return sum(2*i - 1 for i in range(1, n+1))

def inducao1():
    for n in range(1, 11):
        soma = soma_impares(n)
        formula = n**2
        print(f"Para n = {n}: Soma = {soma}, Fórmula n^2 = {formula}")
        assert soma == formula, f"Falha para n = {n}"
```

```
# Chama a função para realizar a indução
print("Letra a):")
inducao1()

def soma_inteiros(n):
    return sum(range(1, n+1))

def inducao2():
    for n in range(1, 11):
        soma = soma_inteiros(n)
        formula = (n * (n + 1)) // 2
        print(f"Para n = {n}: Soma = {soma}, Fórmula (n(n+1))/2 = 
{formula}")
        assert soma == formula, f"Falha para n = {n}"
print("\nLetra b):")
inducao2()
```

Slide 45 - Prova por contradição - exemplo 1

Objetivo: Provar que $\sqrt{2}$ é um número irracional.

1. **Suposição inicial**:
   - Suponha, por contradição, que $\sqrt{2}$ é racional.
   - Isso significa que existem inteiros p e q, sem fatores comuns (ou seja, p e q são coprimos), tal que $\sqrt{2}=\frac{p}{q}$.
2. **Desenvolvimento da contradição**:
   - Se $\sqrt{2}=\frac{p}{q}$, então elevando ambos os lados ao quadrado, obtemos $2=\frac{p^2}{q^2}$, ou $p^2=2q^2$.
   - Da equação $p^2=2q^2$, p^2 deve ser par porque é duas vezes um número inteiro ($2q^2$).
   - Se p^2 é par, então p também deve ser par (pois somente o quadrado de um número par é par).
   - Se p é par, então existe um inteiro m tal que $p=2m$.
3. **Aprofundamento na contradição**:
   - Substituindo p por $2m$ na equação $p^2=2q^2$, temos $(2m)^2=2q^2$, ou $4m^2=2q^2$, o que simplifica para $2m^2=q^2$.
   - Isso implica que q^2 também é par, e consequentemente, q deve ser par.
4. **Conclusão da contradição**:
   - Se tanto p quanto q são pares, então eles têm pelo menos o fator comum 2, contradizendo a suposição inicial de que p e q são coprimos (não têm fatores comuns).
   - Portanto, nossa suposição inicial de que $\sqrt{2}$ é racional leva a uma contradição.

Conclusão:

- Dado que a suposição de que $\sqrt{2}$ é racional conduz a uma contradição, devemos concluir que $\sqrt{2}$ é irracional.

Slide 46 - Prova por contradição - exemplo 2

Objetivo: Provar que 0 é o único elemento neutro da adição em N.

1. **Suposição Inicial**:
   - Suponha, para fins de contradição, que exista um $e \in N$, com $e \neq 0$, que também é um elemento neutro da adição.
2. **Desenvolvimento**:
   - Sabe-se que 0 é o elemento neutro da adição, então para qualquer $n \in N$, temos $n = n + 0$. Em particular, se escolhermos $n = e$, obtemos $e = e + 0$.
   - Além disso, pela nossa suposição, e também é um elemento neutro. Isso significa que para qualquer $n \in N$, $n = e + n$. Especificamente, escolhendo $n = 0$, obtemos $0 = e + 0$.
3. **Contradição**:
   - Agora, temos duas expressões: $e = e + 0$ e $0 = e + 0$. Isso implica que $e = 0$.
   - No entanto, isso contradiz nossa suposição inicial de que $e \neq 0$.

Conclusão:

- A suposição de que existe um elemento neutro diferente de 0 leva a uma contradição.
- Portanto, 0 é o único elemento neutro da adição em N.

24. Prova por contradição: Faça a prova por contradição dos seguintes exemplos:

a) provar que raiz de 2 é um número irracional

b) provar que 0 é o único elemento neutro da adição em N

```python
def sqrt2():
    for a in range(1, 21):
        for b in range(1, 21):
            if a / b == (2 ** 0.5):
                return f"Sqrt(2) pode ser representado como {a}/{b}, o que é um número racional."
    return "Sqrt(2) não pode ser representado como uma fração, é irracional."

result1 = sqrt2()
print(f'Letra a):{result1}')


def elemento_neutro():
    for n in range(1, 21):
        if n + 0 != n:
            return f"Contradição: Para n = {n}, n + 0 != n."
```

```
    return "0 é o elemento neutro da adição em N."

result2 = elemento_neutro()
print(f'Letra b): {result2}')
```

Slide 47 - Grafos

Slide 48 - Definição de Grafo

Um **grafo** é um par ordenado (V, A), onde:

- V é o conjunto de **vértices** (ou nós) do grafo.
- A é a relação binária sobre V, que especifica os **arcos** (ou arestas) do grafo.

Vertices $v_i, v_j \in V$ tais que $(v_i, v_j) \in A$ são ditos adjacentes, significando que eles são conectados por um arco.

Exemplo de um Grafo

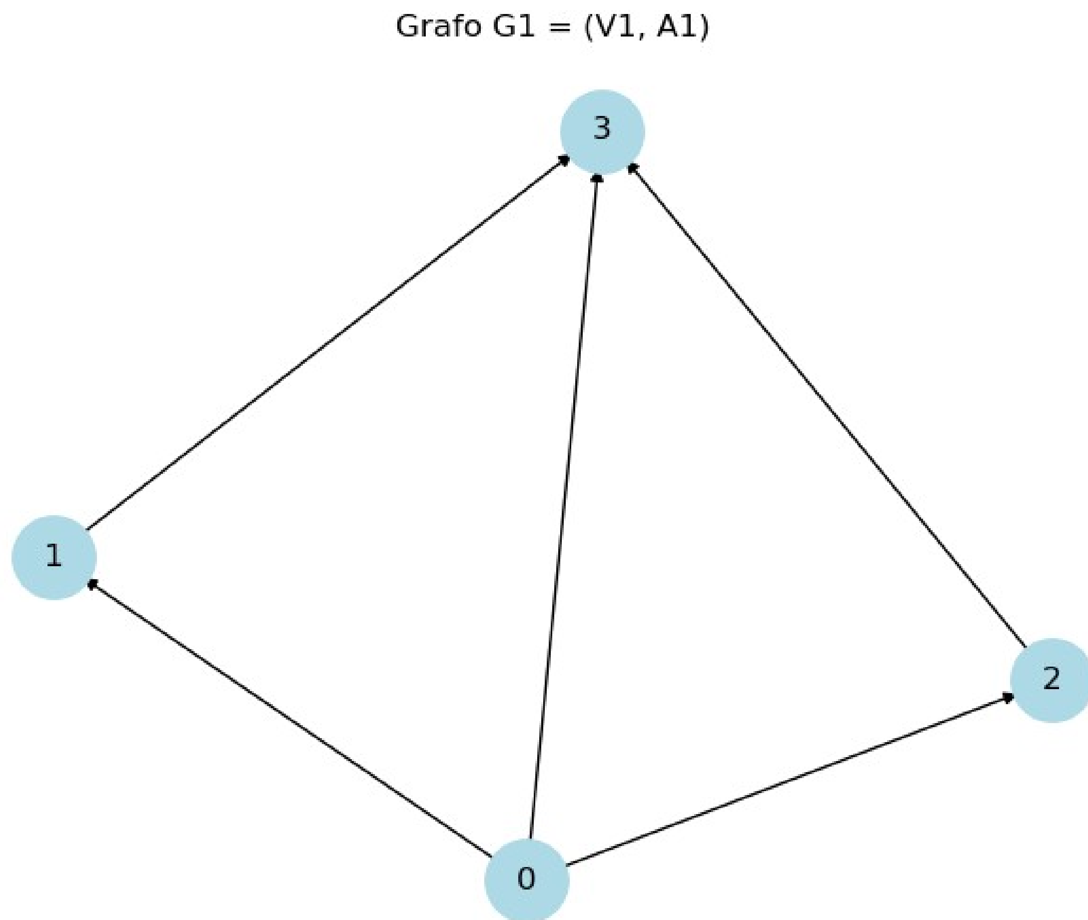
Considere o grafo G_1 representado textualmente e graficamente:

- $G_1 = (V_1, A_1)$
- $V_1 = \{0, 1, 2, 3\}$
- $A_1 = \{(0,1), (0,2), (0,3), (1,3), (2,3)\}$

Graficamente, o grafo G_1 pode ser representado com círculos denotando os vértices e linhas denotando os arcos entre eles. Infelizmente, não posso renderizar ou incluir imagens diretamente aqui, mas a representação seria um ponto para cada vértice conectado por linhas que representam os arcos especificados em A_1.

Cada vértice em V_1 é um ponto numerado de 0 a 3, e cada par em A_1 indica uma linha conectando dois vértices. Por exemplo, o par $(0, 1)$ representa uma linha conectando os vértices 0 e 1.

```
import matplotlib.pyplot as plt
import networkx as nx

G1 = nx.DiGraph()

G1.add_nodes_from([0, 1, 2, 3])
G1.add_edges_from([(0, 1), (0, 2), (0, 3), (1, 3), (2, 3)])
nx.draw(G1, with_labels=True, node_size=800, node_color='skyblue',
font_size=12, edge_color='black', linewidths=4, font_color='black')
plt.title('Grafo G1 = (V1, A1)')
plt.show()
```



Slide 49 - Grafo Orientado

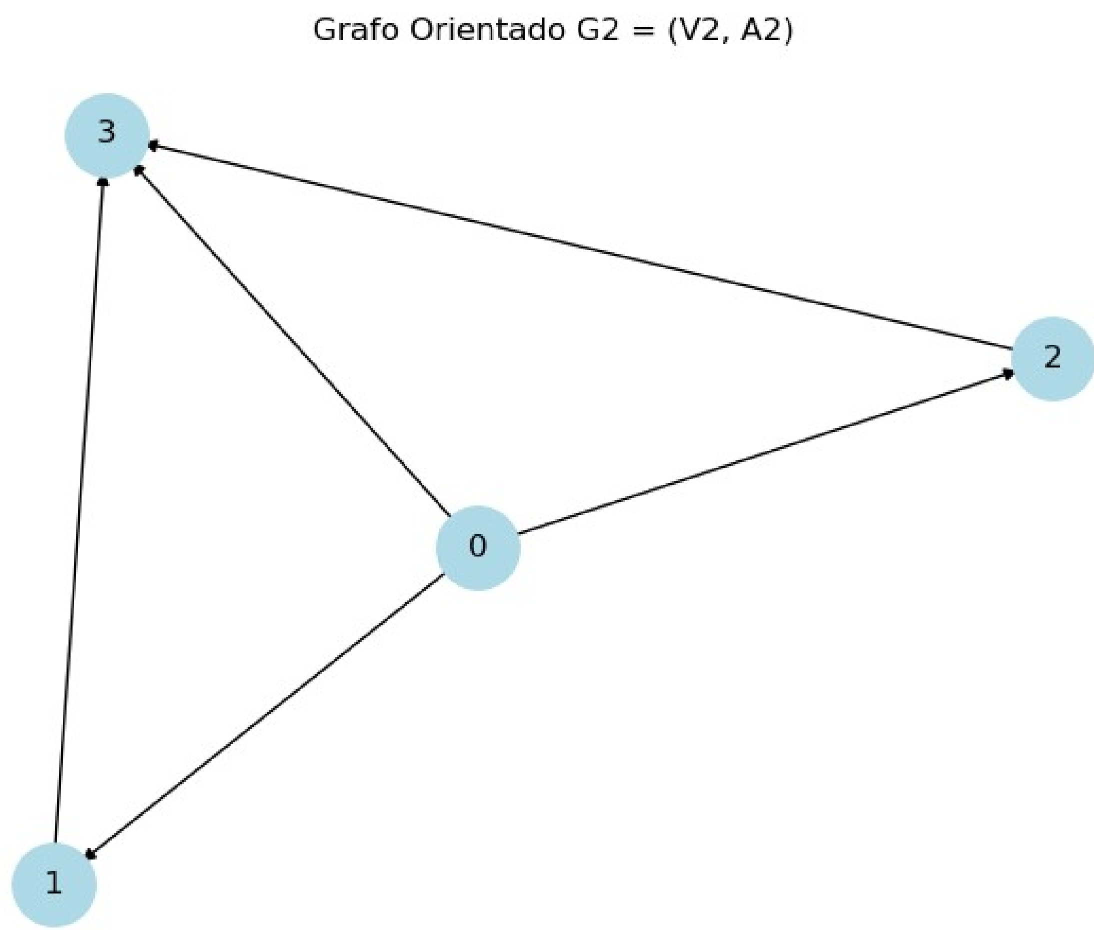
Definição de Grafo Orientado

- Um grafo orientado, ou digrafo, é aquele em que os pares ordenados $(v_i, v_j) \in A$ representam arcos com uma direção específica, de v_i para v_j.
- Em um grafo orientado:
  - Se $(v_i, v_j) \in A$, então v_i é chamado de predecessor de v_j, e v_j é chamado de sucessor de v_i.
- Considere o grafo orientado G_2:
  - $V_2 = \{0, 1, 2, 3\}$ representa o conjunto de vértices.
  - $A_2 = \{(0,1), (0,2), (0,3), (1,3), (2,3)\}$ representa o conjunto de arcos, indicando a direção da relação entre os vértices.

Grafo G_2

- Vértices: $V_2 = \{0, 1, 2, 3\}$
- Arcos: $A_2 = \{(0,1), (0,2), (0,3), (1,3), (2,3)\}$

```
G2 = nx.DiGraph()

V2 = [0, 1, 2, 3]
G2.add_nodes_from(V2)

A2 = [(0, 1), (0, 2), (0, 3), (1, 3), (2, 3)]
G2.add_edges_from(A2)

pos = nx.spring_layout(G2)
nx.draw(G2, pos, with_labels=True, node_size=800,
node_color='skyblue', font_size=12, edge_color='black', linewidths=4,
font_color='black')
plt.title("Grafo Orientado G2 = (V2, A2)")
plt.show()
```



Slide 50 - Grafo Ordenado

Um grafo é considerado **ordenado** se existe uma relação de ordem pré-definida sobre os arcos que saem de cada vértice. No grafo G_3, temos:

- **Conjunto de Vértices**: $V_3 = \{a, b, c, d\}$
- **Conjunto de Arcos**: $A_3 = \{(a,b), (b,a), (a,c), (a,d), (c,b), (d,c), (c,d)\}$

A relação de ordem entre os arcos é dada por:

$$(a,b) < (b,a) < (a,c) < (a,d) < (c,b) < (d,c) < (c,d)$$

Isso significa que, para o vértice a, os arcos são ordenados da seguinte forma:

- Saindo de a: (a,b), (a,c), (a,d)

Para o vértice b:

- Saindo de b: (b,a)

Para o vértice c:

- Saindo de c: (c,b), (c,d)

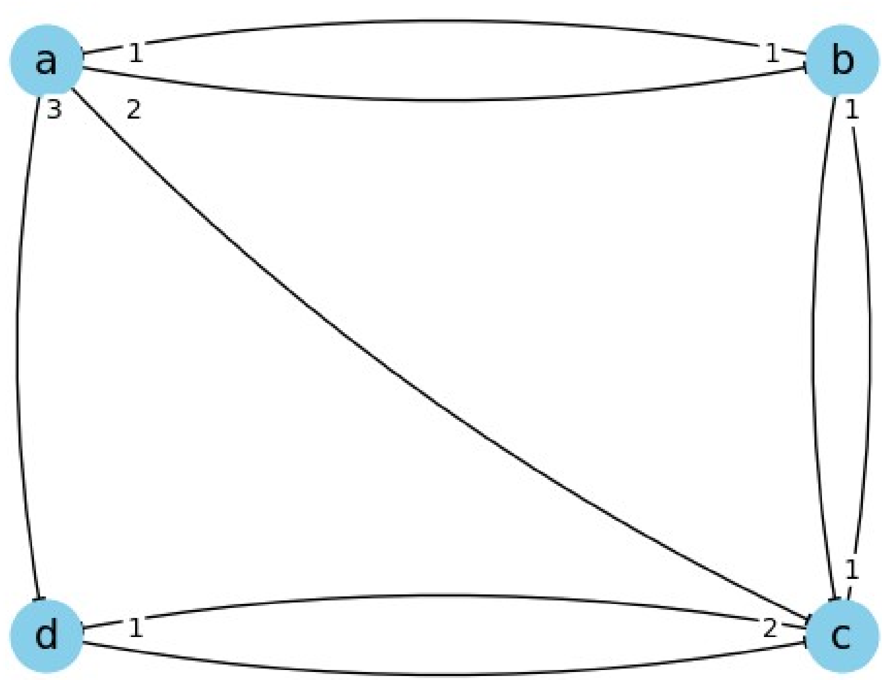
Para o vértice d:

- Saindo de d: (d,c)

```
import networkx as nx
import matplotlib.pyplot as plt

# Criando um grafo direcionado
G = nx.MultiDiGraph()

# Adicionando vértices e arestas com pesos
G.add_edge('a', 'b', weight=1)
G.add_edge('b', 'a', weight=1)
G.add_edge('a', 'c', weight=2)
G.add_edge('a', 'd', weight=3)
G.add_edge('c', 'b', weight=1)
G.add_edge('d', 'c', weight=1)
G.add_edge('c', 'd', weight=2)
G.add_edge('b', 'c', weight=1)

# Definindo a posição dos vértices
pos = {'a': (0, 1), 'b': (1, 1), 'c': (1, 0), 'd': (0, 0)}

# Desenhando o grafo com arcos curvos para arestas bidirecionais
nx.draw_networkx_nodes(G, pos, node_size=700, node_color='skyblue')
nx.draw_networkx_labels(G, pos, font_size=15)

# Desenhar as arestas com curvas suaves para distinguir as
bidirecionais
for (u, v, key) in G.edges(keys=True):
    style = 'arc3, rad=0.1' if key == 0 else 'arc3, rad=-0.1'
    nx.draw_networkx_edges(G, pos, edgelist=[(u, v)],
connectionstyle=style)
```

```
# Adicionando rótulos de aresta no início de cada aresta
edge_labels = {(u, v): d['weight'] for u, v, d in G.edges(data=True)}
for (u, v), weight in edge_labels.items():
    # Define o deslocamento para mover os rótulos para perto do nó de
origem
    edge_pos = pos[u]
    text_pos = (edge_pos[0] * 0.9 + pos[v][0] * 0.1, edge_pos[1] * 0.9
+ pos[v][1] * 0.1)

    # Desenha os rótulos das arestas com fundo branco para melhor
visibilidade
    plt.text(text_pos[0], text_pos[1], s=weight,
bbox=dict(facecolor='white', edgecolor='none',
boxstyle='round,pad=0.1'))

# Mostrando o grafo
plt.axis('off')  # Desliga os eixos
plt.show()
```

```
C:\Users\selah\AppData\Local\Temp\ipykernel_28092\1524887123.py:27:
DeprecationWarning: `alltrue` is deprecated as of NumPy 1.25.0, and
will be removed in NumPy 2.0. Please use `all` instead.
  nx.draw_networkx_edges(G, pos, edgelist=[(u, v)],
connectionstyle=style)
```



Slide 51 - Conceitos de Grafos Orientados

Em um grafo orientado, podemos definir conceitos importantes baseados nas conexões entre os vértices:

- **Ramificação de Saída (N_S)**: Número de arcos que partem de um vértice.
- **Ramificação de Entrada (N_E)**: Número de arcos que chegam a um vértice.
- **Vértices-base ou Vértices-raiz**: Vértices que não têm arcos chegando a eles ($N_E = 0$).
- **Vértices-folha**: Vértices que não têm arcos partindo deles ($N_S = 0$).

Considerando o grafo G_3, temos:

- Vértice a: Ramificação de Saída $N_S(a) = 3$, Ramificação de Entrada $N_E(a) = 1$
- Vértice b: Ramificação de Saída $N_S(b) = 1$, Ramificação de Entrada $N_E(b) = 2$
- Vértice c: Ramificação de Saída $N_S(c) = 2$, Ramificação de Entrada $N_E(c) = 2$
- Vértice d: Ramificação de Saída $N_S(d) = 1$, Ramificação de Entrada $N_E(d) = 2$

Desta forma, no grafo G_3, não temos vértices-base ou vértices-folha, pois todos os vértices têm pelo menos uma ramificação de saída e uma de entrada.

```python
# Função para encontrar vértices-base/raiz e vértices-folha
def find_root_and_leaf_nodes(G):
    root_nodes =
    leaf_nodes =
    return root_nodes, leaf_nodes

# Encontrar e imprimir os vértices-base/raiz e vértices-folha

print(f'Vertices-base/raiz: {root_nodes}')
print(f'Vertices-folha: {leaf_nodes}')

# Calcula a ramificação de saída e de entrada para cada vértice
ramificacao_saida =
ramificacao_entrada =

# Exibe as ramificações para cada vértice
print("Ramificação de Saída:", ramificacao_saida)
print("Ramificação de Entrada:", ramificacao_entrada)
```

```
Vertices-base/raiz: []
Vertices-folha: []
Ramificação de Saída: {'a': 3, 'b': 1, 'c': 2, 'd': 1}
Ramificação de Entrada: {'a': 1, 'b': 2, 'c': 2, 'd': 2}
```

Slide 52 - Caminhos e Ciclos em Grafos

Um **caminho** em um grafo é uma sequência de arcos que conectam uma série de vértices, começando no vértice inicial e terminando no vértice final, de tal forma que cada arco está diretamente conectado ao próximo.

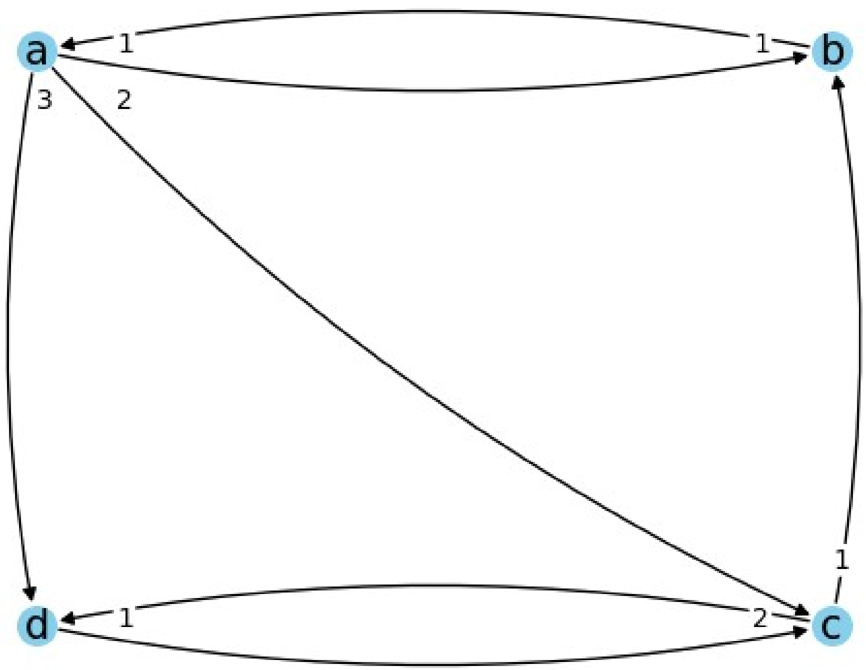
Um **ciclo** é um caminho particular que começa e termina no mesmo vértice.

- **Caminho**: Para o grafo G_3, a sequência $(a,c)(c,b)$ é um caminho válido de comprimento 2.
- **Ciclo**: O grafo G_3 é cíclico, pois contém ciclos, por exemplo, $(a,b)(b,a)$.
- **Grafo Cíclico**: Um grafo é cíclico se contém pelo menos um ciclo.
- **Grafo Acíclico**: Um grafo é acíclico se não contém nenhum ciclo.

```
import networkx as nx
import matplotlib.pyplot as plt

# Criando um grafo direcionado
G = nx.MultiDiGraph()

# Adicionando vértices e arestas com pesos
G.add_edge('a', 'b', weight=1)
G.add_edge('b', 'a', weight=1)
G.add_edge('a', 'c', weight=2)
G.add_edge('a', 'd', weight=3)
G.add_edge('c', 'b', weight=1)
G.add_edge('d', 'c', weight=1)
G.add_edge('c', 'd', weight=2)

# Definindo a posição dos vértices
pos = {'a': (0, 1), 'b': (1, 1), 'c': (1, 0), 'd': (0, 0)}

# Desenhando o grafo com arcos curvos para arestas bidirecionais
nx.draw_networkx_nodes(G, pos, node_size=200, node_color='skyblue')
nx.draw_networkx_labels(G, pos, font_size=15)

# Desenhar as arestas com curvas suaves para distinguir as
bidirecionais
for (u, v, key) in G.edges(keys=True):
    style = 'arc3, rad=0.1' if key == 0 else 'arc3, rad=-0.1'
    nx.draw_networkx_edges(G, pos, edgelist=[(u, v)],
connectionstyle=style)

# Adicionando rótulos de aresta no início de cada aresta
edge_labels = {(u, v): d['weight'] for u, v, d in G.edges(data=True)}
for (u, v), weight in edge_labels.items():
    # Define o deslocamento para mover os rótulos para perto do nó de
origem
    edge_pos = pos[u]
    text_pos = (edge_pos[0] * 0.9 + pos[v][0] * 0.1, edge_pos[1] * 0.9
```

```
+ pos[v][1] * 0.1)

    # Desenha os rótulos das arestas com fundo branco para melhor
visibilidade
    plt.text(text_pos[0], text_pos[1], s=weight,
bbox=dict(facecolor='white', edgecolor='none',
boxstyle='round,pad=0.1'))

# Mostrando o grafo
plt.axis('off')  # Desliga os eixos
plt.show()

# Encontrando caminhos
print("Caminhos de 'a' para 'b':")
for path in nx.all_simple_paths(G, source='a', target='b'):
    print(path)

# Verificando a existência de ciclos
has_cycles = nx.is_directed_acyclic_graph(G)
print(f"O grafo G3 é cíclico? {'Não' if has_cycles else 'Sim'}")

# Identificando um ciclo (se houver)
try:
    cycle = nx.find_cycle(G)
    print("Um ciclo em G3:", cycle)
except nx.NetworkXNoCycle:
    print("G3 é acíclico.")
```

```
C:\Users\selah\AppData\Local\Temp\ipykernel_15908\763802617.py:26:
DeprecationWarning: `alltrue` is deprecated as of NumPy 1.25.0, and
will be removed in NumPy 2.0. Please use `all` instead.
  nx.draw_networkx_edges(G, pos, edgelist=[(u, v)],
connectionstyle=style)
```




```
Caminhos de 'a' para 'b':
['a', 'b']
['a', 'c', 'b']
['a', 'd', 'c', 'b']
O grafo G3 é cíclico? Sim
Um ciclo em G3: [('a', 'b', 0), ('b', 'a', 0)]
```

Slide 53 - Grafo Rotulado

Um **grafo rotulado** é aquele em que seus vértices ou arcos têm rótulos associados que representam informações adicionais.

- **Rotulação de Vértices**: É uma função f_V que associa cada vértice do conjunto V a um rótulo do conjunto R_V.
- **Rotulação de Arcos**: É uma função f_A que associa cada arco do conjunto A a um rótulo do conjunto R_A.

Exemplo de Grafo Rotulado G_4:

Considere o grafo G_4:

- **Vértices**: $V_4 = \{0, 1, 2\}$
- **Arcos**: $A_4 = \{(0,1), (1,2), (0,2)\}$

Uma possível rotulação para G_4 é:

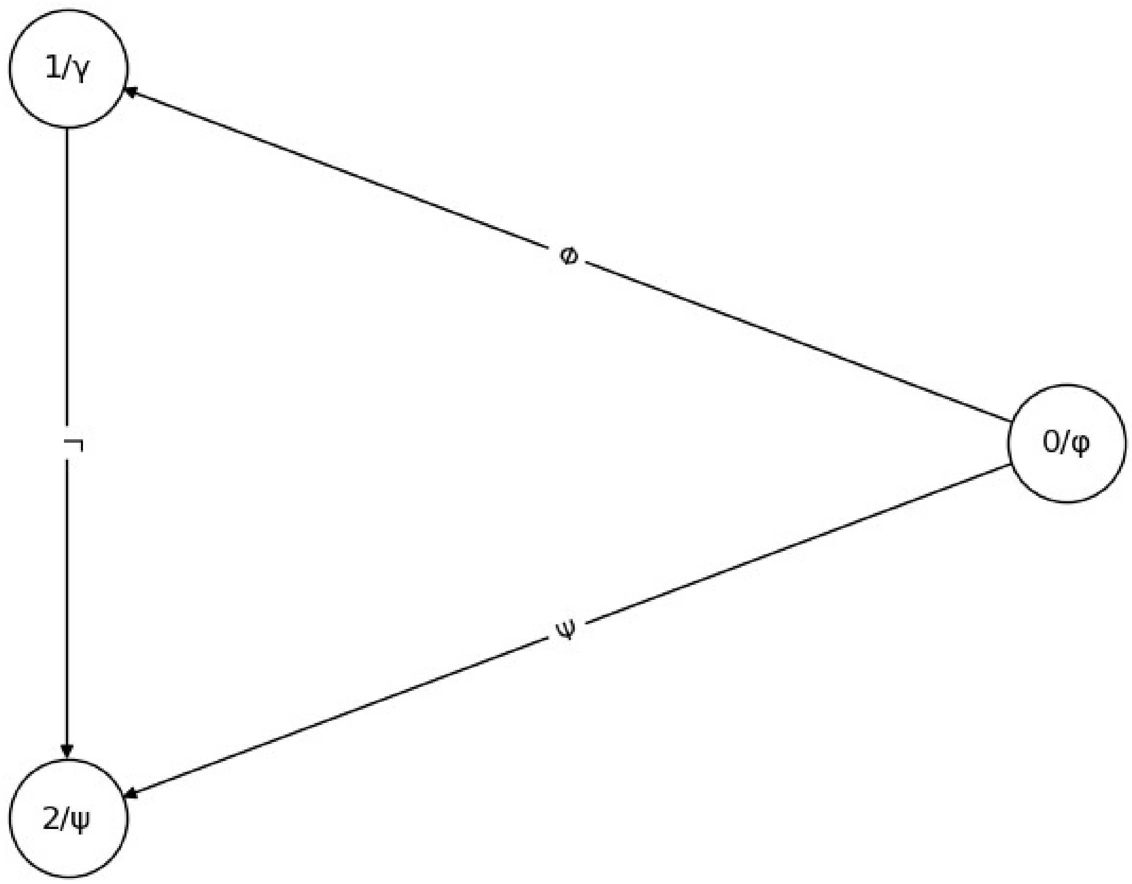
- **Rotulação de Vértices:**

$$f_V = \{(0,\phi),(1,\gamma),(2,\psi)\} \text{ com } R_V = \{\phi,\gamma,\psi\}$$

- **Rotulação de Arcos:**

$$f_A = \{((0,1),\Phi),((1,2),\Gamma),((0,2),\Psi)\} \text{ com } R_A = \{\Phi,\Gamma,\Psi\}$$

```
import networkx as nx
import matplotlib.pyplot as plt

# Criando um grafo direcionado
G4 = nx.DiGraph()

# Adicionando vértices e arestas do grafo G4
G4.add_edge('0', '1', label='Φ')
G4.add_edge('1', '2', label='Γ')
G4.add_edge('0', '2', label='Ψ')

# Rotulação dos vértices
vertex_labels = {'0': 'φ', '1': 'γ', '2': 'ψ'}
for node, label in vertex_labels.items():
    G4.nodes[node]['label'] = label

# Desenhando o grafo
pos = nx.circular_layout(G4)  # Posicionamento circular para os nós
nx.draw(G4, pos, with_labels=False, node_size=2000,
node_color='white', edgecolors='black')

# Desenhando os rótulos dos vértices
for node, (x, y) in pos.items():
    plt.text(x, y, f'{node}/{G4.nodes[node]["label"]}', fontsize=12,
ha='center', va='center')

# Desenhando os rótulos das arestas
edge_labels = nx.get_edge_attributes(G4, 'label')
nx.draw_networkx_edge_labels(G4, pos, edge_labels=edge_labels,
font_color='black')

# Mostrando o grafo
plt.axis('off')  # Desliga os eixos
plt.show()
```



Slides 54 a 59 - Árvores

Definição de Árvore

- Uma **árvore** é um grafo acíclico (sem ciclos) e orientado, onde:
  - Existe exatamente um vértice chamado **raiz** com $N_E = 0$ (sem arestas de entrada).
  - Todos os outros vértices têm exatamente uma aresta de entrada ($N_E = 1$).
  - Existe um único caminho de qualquer vértice para a raiz.

Conceitos em Árvores

- **Ancestral e Descendente**:
  - Se a é ancestral de b, então é possível percorrer um caminho da raiz até b passando por a.
  - Se b é descendente de a, então a vem antes de b no caminho da raiz até b.
- **Pai e Filho**:
  - Se não houver vértices intermediários entre a e b, e se a é ancestral direto de b, então a é o pai de b e b é o filho de a.
- **Folhas e Nós Internos**:
  - Vértices sem filhos são chamados de folhas.

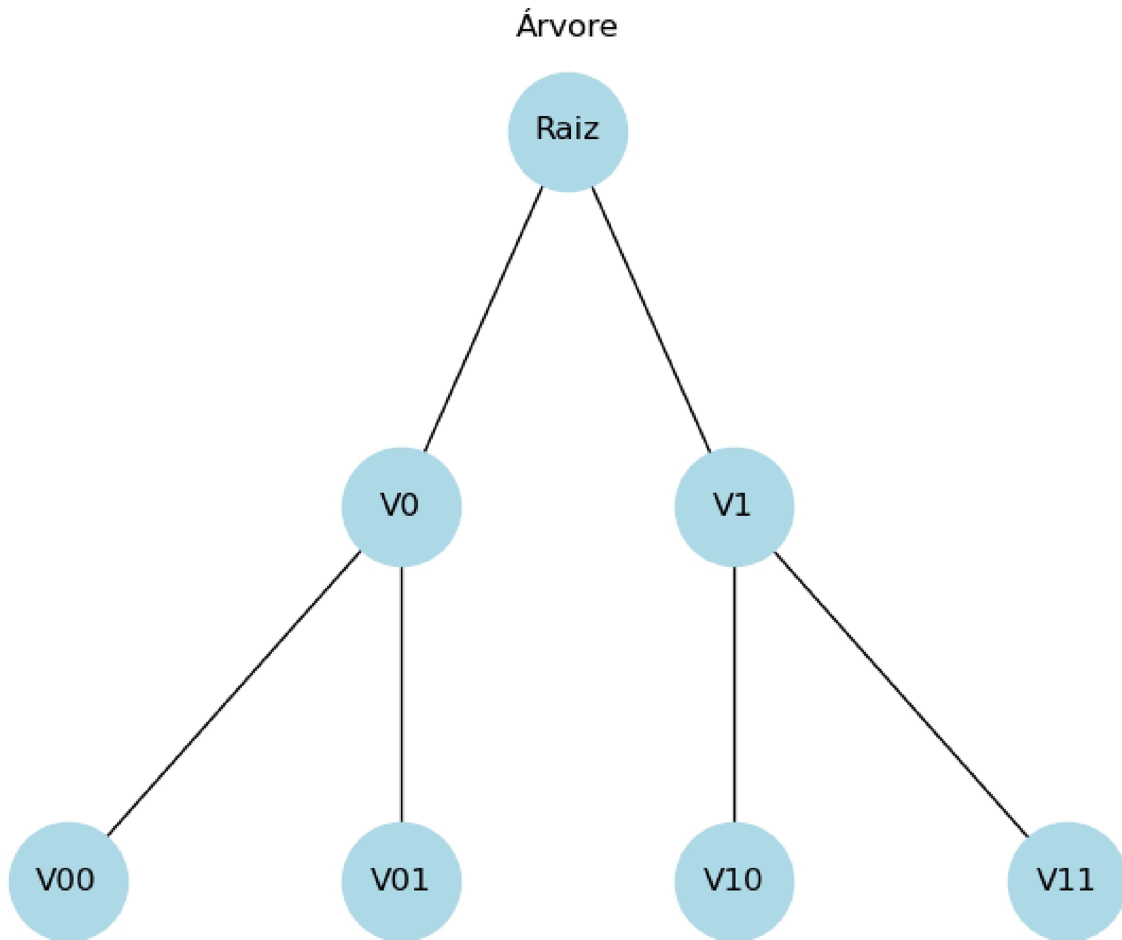
    - Vértices com pelo menos um filho são chamados de nós internos.
- **Profundidade**:
    - A profundidade de um nó é o número de arestas no caminho da raiz até esse nó.

Exemplo

Considerando uma árvore com vértices nomeados como `Raiz`, `V1`, `V11`, `V00`, `V01`, `V0`, `V10`:

- `V1` é o pai de `V11`.
- `Raiz` é ancestral de todos os nós.
- `V00` e `V11` são folhas.
- `V0` e `V1` são nós internos.
- `V01` e `V10` têm profundidade 2.

```
import networkx as nx
import matplotlib.pyplot as plt

# Criando uma árvore
T = nx.DiGraph()

# Adicionando vértices e arestas
T.add_edges_from([
    ('Raiz', 'V0'),
    ('Raiz', 'V1'),
    ('V0', 'V00'),
    ('V0', 'V01'),
    ('V1', 'V10'),
    ('V1', 'V11'),
])

# Desenhando a árvore
pos = nx.nx_agraph.graphviz_layout(T, prog='dot')  # Usando o layout do Graphviz
nx.draw(T, pos, with_labels=True, node_size=2000, node_color='lightblue', arrows=False)

# Mostrando a árvore
plt.title("Árvore")
plt.show()
```




```
# Supondo que G seja o grafo definido anteriormente

# Função para encontrar a raiz, folhas, nós de entrada e saída
def graph_properties(G):
    # Identificar se o grafo é acíclico
    is_acyclic = []
    print(f"O grafo G é acíclico? {is_acyclic}")

    # Identificar a raiz (vértices com grau de entrada igual a zero)
    root = []
    print(f"Raiz do grafo: {root}")

    # Identificar as folhas (vértices com grau de saída igual a zero)
    leaves = []
    print(f"Folhas do grafo: {leaves}")

    # Calcular os graus de entrada e saída
    in_degrees = {}
    out_degrees = {}
    print(f"Graus de entrada: {in_degrees}")
```

```
    print(f"Graus de saída: {out_degrees}")

    return is_acyclic, root, leaves, in_degrees, out_degrees

properties = graph_properties(T)
```

```
O grafo G é acíclico? True
Raiz do grafo: ['Raiz']
Folhas do grafo: ['V00', 'V01', 'V10', 'V11']
Graus de entrada: {'Raiz': 0, 'V0': 1, 'V1': 1, 'V00': 1, 'V01': 1,
'V10': 1, 'V11': 1}
Graus de saída: {'Raiz': 2, 'V0': 2, 'V1': 2, 'V00': 0, 'V01': 0,
'V10': 0, 'V11': 0}
```

- ***Identificando nós folha***: Para identificar os nós folha em um grafo usando a biblioteca NetworkX em Python, utilizamos a seguinte linha de código:

```
leaves = [node for node, deg in G.out_degree() if deg == 0]
```

A linha de código em questão cria uma lista dos nós folha de um grafo `G`. Os nós folha são aqueles que não têm arestas saindo deles. Vamos decompô-la:

- `G.out_degree()`: Este método retorna um conjunto de pares (nó, grau de saída) para todos os nós no grafo `G`. O "grau de saída" é o número de arestas que saem de um nó.

- `[node for node, deg in G.out_degree() if deg == 0]`: Esta é uma compreensão de lista, uma forma concisa de construir uma lista em Python. O que ela faz é:

  - Iterar sobre cada par `(nó, grau de saída)` gerado por `G.out_degree()`.
  - Checar se o grau de saída (`deg`) é `0`, o que significa que não há arestas saindo desse nó.
  - Se o grau de saída for `0`, incluir o `nó` na lista `leaves`.

Portanto, `leaves` será uma lista contendo todos os nós do grafo `G` que são nós folha.

- ***Identificando a raiz***: Para identificar a raiz em um grafo usando a biblioteca NetworkX em Python, utilizamos a seguinte linha de código:

```
root = [node for node, deg in G.in_degree() if deg == 0]
```

A linha de código em questão cria uma lista dos nós raiz de um grafo `G`. Os nós folha são aqueles que não têm arestas entrando neles. Vamos decompô-la:

- `G.in_degree()`: Este método retorna um conjunto de pares (nó, grau de entrada) para todos os nós no grafo `G`. O "grau de entrada" é o número de arestas que entrasm em cada nó.

- `[node for node, deg in G.in_degree() if deg == 0]`: Esta é uma compreensão de lista, uma forma concisa de construir uma lista em Python. O que ela faz é:
  - Iterar sobre cada par `(nó, grau de entrada)` gerado por `G.in_degree()`.
  - Checar se o grau de entrada (`deg`) é `0`, o que significa que não há arestas entrandonesse nó.
  - Se o grau de entrada for `0`, incluir o `nó` na lista `root`.

Portanto, `root` será uma lista contendo todos os nós do grafo `G` que são nós raiz.

- ***Calculando Graus de Entrada e Saída em um Grafo***: Para calcular os graus de entrada e saída de cada nó em um grafo utilizando a biblioteca NetworkX, usamos as seguintes linhas de código:

```
in_degrees = {node: deg for node, deg in G.in_degree()}
out_degrees = {node: deg for node, deg in G.out_degree()}
```

`G.in_degree()`: Este método retorna um iterador sobre os pares (nó, grau de entrada) para todos os nós no grafo `G`. O grau de entrada é o número total de arestas direcionadas para um nó.

`G.out_degree()`: Este método retorna um iterador sobre os pares (nó, grau de saída) para todos os nós no grafo `G`. O grau de saída é o número total de arestas que saem de um nó.

`{node: deg for node, deg in G.in_degree()}`: Esta é uma dictionary comprehension que cria um dicionário `in_degrees`, onde cada chave é um nó do grafo, e seu valor é o grau de entrada desse nó.

`{node: deg for node, deg in G.out_degree()}`: De forma semelhante, cria um dicionário `out_degrees`, onde cada chave é um nó do grafo, e seu valor é o grau de saída desse nó.