**Relações Binárias**

Trabalho prático de relações binárias - Matemática Discreta - UFMG

**Objetivo:** O objetivo desse trabalho é:

1. Ler um conjunto de pares ordenados;
2. Retornar se as propriedades (Reflexiva, Irreflexiva, Simétrica, Anti-simétrica, Assimétrica, Transitiva) são válidas ou não;
3. Caso as propriedades não sejam válidas, apresentar os pares ordenados que fariam as propriedades serem válidas;
4. Retornar se a relação é de equivalência ou ordem parcial;
5. Retornar o fecho transitivo da relação.

**Entrada:** A entrada deve ser introduzida no arquivo entrada.txt.

A quantidade de elementos no conjunto é representado pelo primeiro número na primeira linha. Os números subsequêntes, ainda na primeira linha, são os "nomes" de cada elemento. Devem ser consecutivos.

A segunda linha em diante representa uma relação de um elemento com outro.

3 20 21 22

20 21

20 22

22 22

**Saída:** A saída será gravada no arquivo: saida.txt.

Reflexiva: F

(20,20); (21,21);

Irreflexiva: F

(22,22);

Simétrica: F

(21,20); (22,20);

Anti-simétrica: V

Assimétrica: F

Transitiva: V

Relação de equivalência: F

Relação de ordem parcial: F

Fecho transitivo da relação: (20,21); (20,22); (22,22);

**Funcionamento:** O programa funciona da seguinte maneira:

* Cria-se uma matriz de adjacência, onde o eixo das linhas representa o ponto de origem da relação, e o eixo das colunas representa o ponto de destino da relação.

Para cada propriedade, foi analisado o seguinte:

* **Reflexiva:** Se toda a diagonal principal da matriz tiver relações, é reflexiva.
* **Irreflexiva:** Se toda a diagonal principal da matriz não tiver relações, é irreflexiva.
* **Simétrica:** Se Matriz[i][j] = Matriz[j][i], então é simétrica.
* **Anti-simétrica:** Se Matriz[i][j] != Matriz[j][i], a menos que i=j, é anti-simétrica.
* **Assimétrica:** É assimétrica se for irreflexiva e se for anti-simétrica.
* **Transitiva:** Se Matriz[i][j] = 1 e Matriz[j][z] = 1 e Matriz[i][z] = 1, então é transitiva.
* **Relação de equivalência:** Se a matriz de adjecência for reflexiva, simétrica e transitiva simultaneamente, é relação de equivalência.
* **Relação de ordem parcial:** Se a matriz de adjecência for reflexiva e transitiva simultaneamente, mas não for anti-simétrica, é relação de ordem parcial.
* **Fecho transitivo da relação:** Se a matriz for transitiva, ela é seu próprio fecho transitivo direto. Se não for transitiva, a matriz juntamente com os itens que faltaram para ser transitiva será o fecho direto.

**Complexidade:** A parte principal do programa é retornar se as propriedades abaixo são válidas ou não. Caso não, retornar os pares ordenados necessários para que seja válida. Logo, a complexidade das partes principais do código são as seguintes:

* **Reflexiva:** Apenas um loop, com duração dependendo exclusivamente da entrada n, logo a complexidade de tempo para o pior caso é O(n).
* **Irreflexiva:** Apenas um loop, com duração dependendo exclusivamente da entrada n, logo a complexidade de tempo para o pior caso é O(n).
* **Simétrica:** Dois loops aninhados, cuja duração se baseia na entrada, possuindo duração O(n²).
* **Anti-simétrica:** Dois loops aninhados, cuja duração se baseia na entrada, possuindo duração O(n²).
* **Assimétrica:** Apenas um condicional.
* **Transitiva:** Três loops aninhados, cuja duração se baseia na entrada, possuindo duração O(n³).
* **Relação de equivalência:** Apenas um condicional.
* **Relação de ordem parcial:** Apenas um condicional.
* **Fecho transitivo da relação:** Dois loops aninhados, cuja duração se baseia na entrada, possuindo duração O(n²).

**Análise geral da complexidade:** O algoritmo contém no máximo três loops aninhados, cuja duração se baseia na entrada, possuindo duração O(n³). O algoritmo utiliza memória baseada numa matriz de adjacência quadrada de n lados, possuindo complexidade de espaço de O(n²).

**Execução:**

1. Adicionar a entrada no arquivo: entrada.txt

2. Compilar: gcc relacao.c -o relacao

3. Executar: ./relacao

4. Resultado será gravado no arquivo: saida.txt