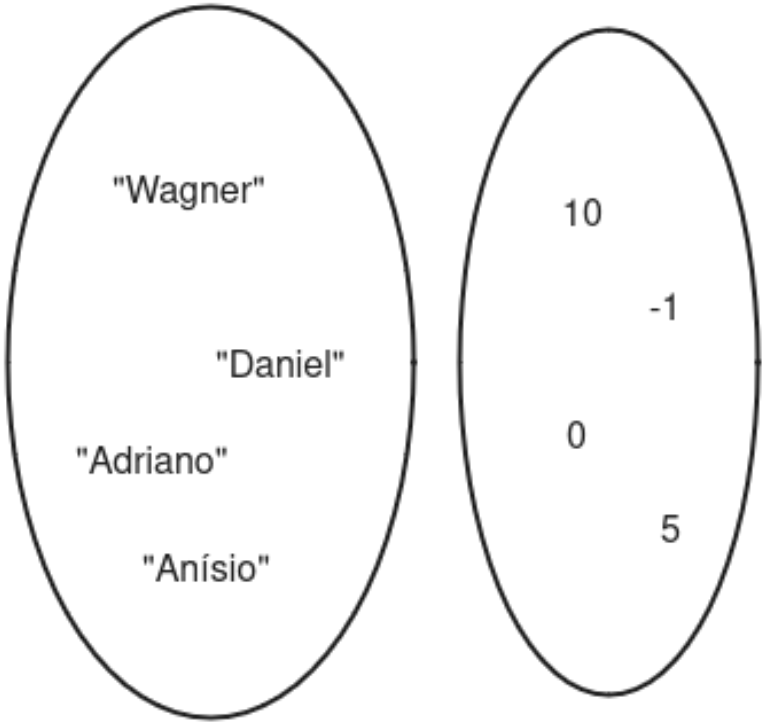
Funções e Algoritmos

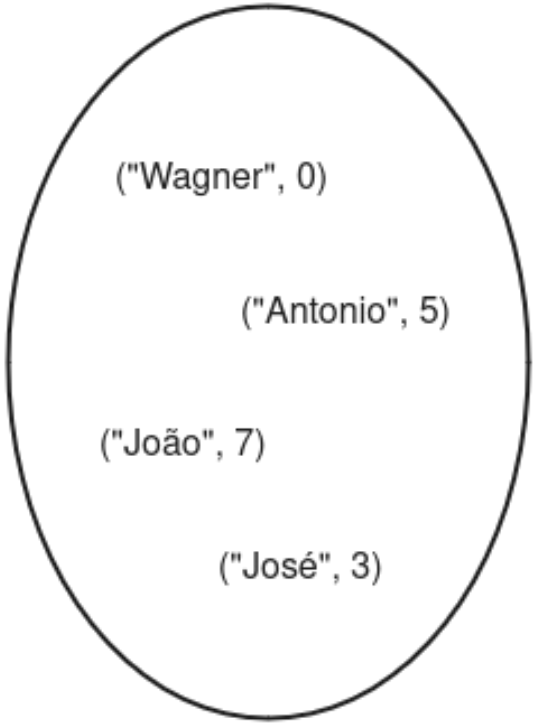
Recaptulando: Conjuntos

- Coleções (não ordenadas)
- Tipos similares
- Implementados por tipos



Recaptulando: Tuplas

- Definidos pelo “produto” dedois conjuntos
- Chamadas “estruturas compostas”
- Definidos por structs

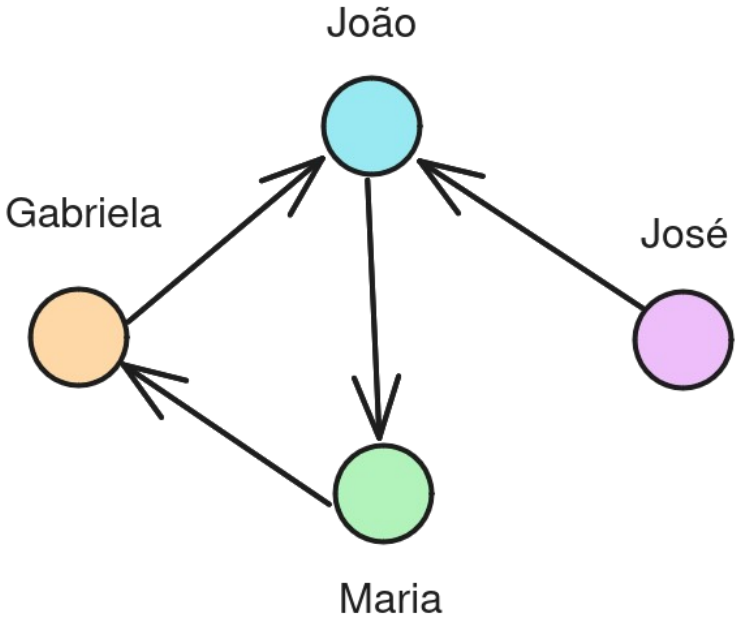


Relações

- É qualquer conjunto de tuplas definido por multiplos conjuntos
- Implica algum tipo de **relação** entre os dados
- Pode ter repetição de algum elemento em dois ou mais pares
- Também pode não conter elementos do conjunto

Exemplo: “Melhor Amigo”

{
(José, João),
(João, Maria),
(Maria, Gabriela),
(Gabriela, João),
}



Classes de Relações

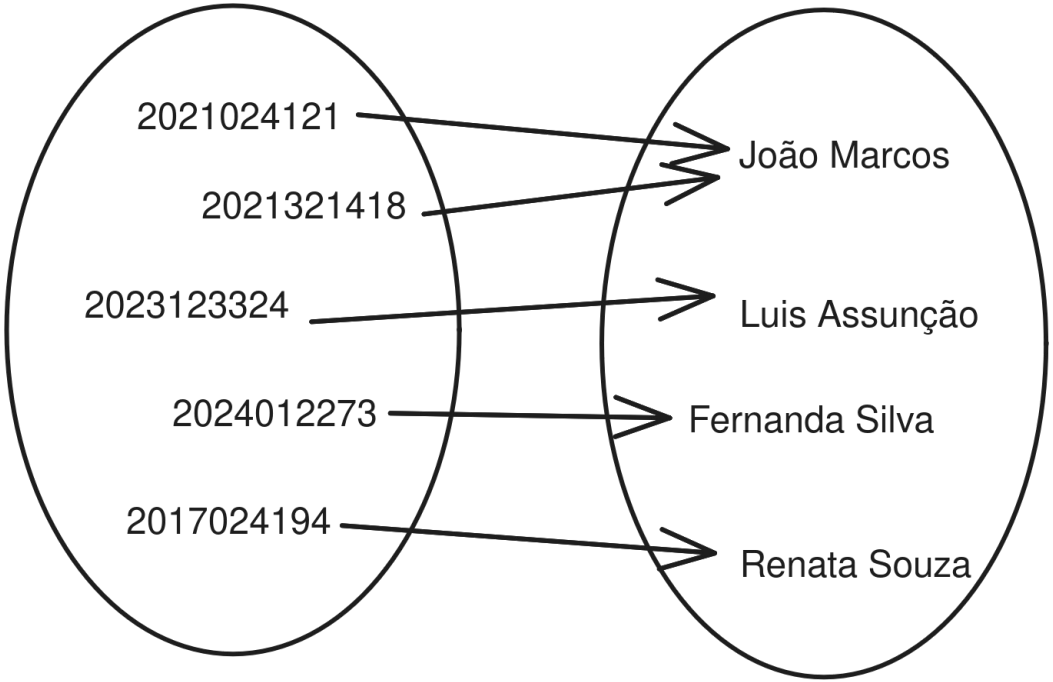
- Existem varias propriedades de Relações relevantes
  - Transitividade, Simetria, Reflexividades
- Uma que vocês podem ouvir falar é a relação de equivalência
- Mas a classe de relações mais importante agora são as funções

Funções (em matemática discreta)

- Relações que contêm **todos** elementos de **um dos** conjuntos **1 vez**
  - Mas podem conter multiplos do outro conjunto
  - Ou faltar alguns elementos dele
- “Mapeiam” desse conjunto (domínio) para o outro (imagem)
- Podem ser definidas por listagem ou descrição
- Essa descrição pode se tornar um procedimento

Funções Listadas

(2021024121, "João Marcos")
(2021321418, "João Marcos")
(2023123324, "Luis Assunçao")
(2024012273, "Fernanda Silva")
(2017024194, "Renata Souza")



Funções Implícitas

- Podem ter notação matemática
  - $y = f(x) = x^2 + 4x - 1$
- Podem ser definidas em linguagem
  - “É o nome associado àquela matrícula”
- Ou podem ser definidas por procedimentos/algoritmos
  - Esse é o foco

Procedimentos

- Basicamente o nome “formal” de uma função em C
- Define uma sequência de passos na linguagem
- Tem um tipo da entrada (domínio), e um tipo da saída (imagem)
- Geralmente implementa um Algoritmo

Algoritmos

- Sequência de etapas para realizar um processo
  - Com mínimo de ambiguidade possível
- Geralmente definido de forma “agnóstica” de linguagem
  - Diagramas, descrições em texto ou Pseudo-Código
- Pode ser analisado em eficiência teórica sem implementar

Pseudo-Código

- Descrição semelhante a código de computador
- Feito pra ser mais fácil de ler
- Abstrai “esquisitices” de linguagem
- “Basicamente python”

```
A: Array
func find_max(A):
    max = A[0]
    for elemento in A:
        if elemento > max:
            max = elemento
    return max
```

Encontra maior o elemento de **A**

Exemplo: Busca

```
A: Array, buscado: Int
func find_normal(A, buscado):
    size = A.size()
    for indice in 0..size:
        elemento = A[indice]
        if elemento == buscado:
            return indice
```

Encontra o indice do valor **buscado** no array **A**

Algoritmos Comuns

- Algoritmos de ordenação
  - Bubblesort, Mergesort, Quicksort
- Algoritmos de busca
  - Busca linear, busca binária, buscas em grafos
- Algoritmos numéricos
  - Raiz quadrada, Método de Newton, Auto-decomposição

Eficiência (próxima aula)

- Algoritmos podem implementar a mesma função
  - Mas alguns podem executar a função melhor
- Esse “melhor” geralmente é definido no tempo de execução
  - Mas também pode ser pela memória utilizada
- Uma forma de análisar perfomance teórica é a Análise Assintótica
  - Veremos ela na próxima aula

Bom Carnaval!

Até a outra semana!