



# LifeInspiration

Projecto de Sistemas de Informação

Life Inspiration V0.3

# Caixeiro Viajante

---

## ► Problema:

- Procura-se encontrar um ou mais caminhos, para percorrer todas as cidades e voltar ao ponto de origem, sem nunca passar numa cidade duas vezes e obter o caminho com menos custo.

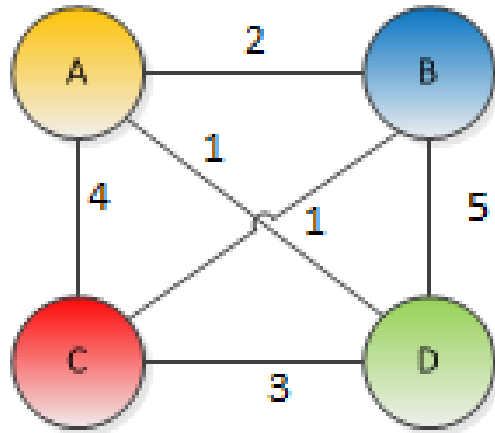
## ► Esquema:

- Tamanho do gene igual ao número de cidades;
- As cidades são indexadas para que as suas representações sejam números inteiros;
- A ordem com que os alelos aparecem no gene é a mesma ordem pela qual passamos nas cidades.



# Caixeiro Viajante

## ► Representação de Caminhos e custos



	A	B	C	D
A	0	2	4	1
B	2	0	1	5
C	4	1	0	3
D	1	5	3	0

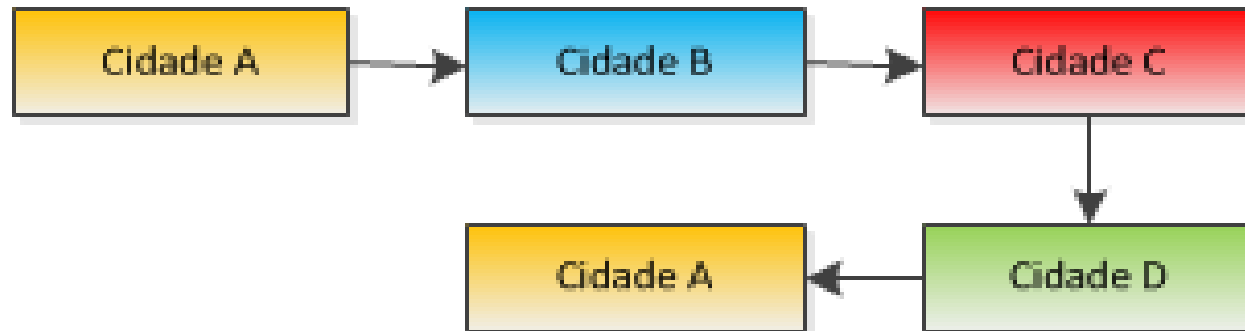
## ► Representação de Genes

0	1	2	3
---	---	---	---

Cidade	A	B	C	D
Índex	0	1	2	3

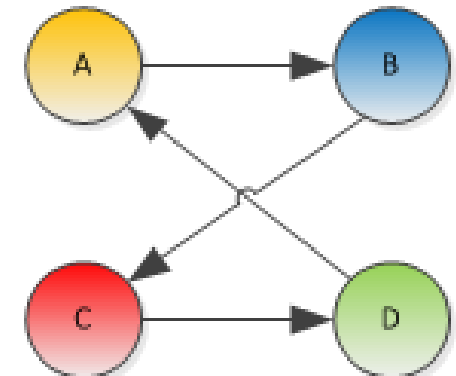
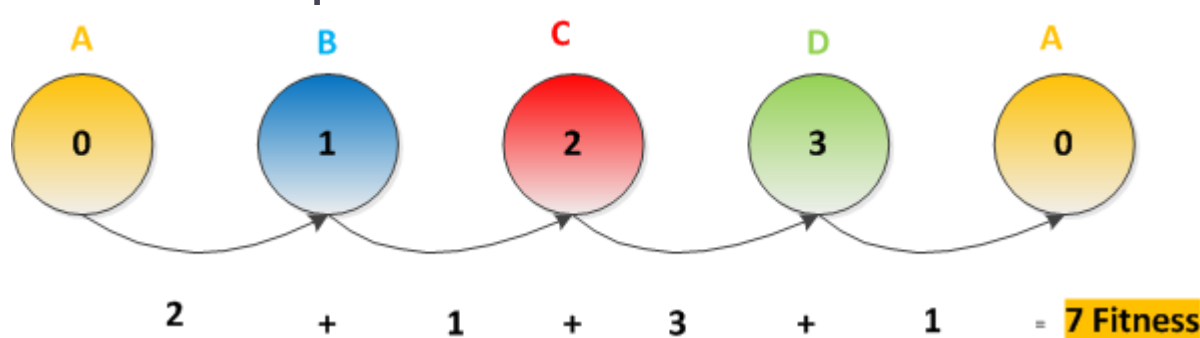
# Caixeiro Viajante

## ► Ordem do caminho



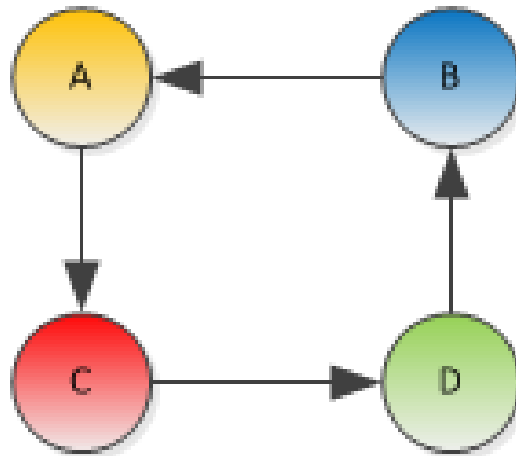
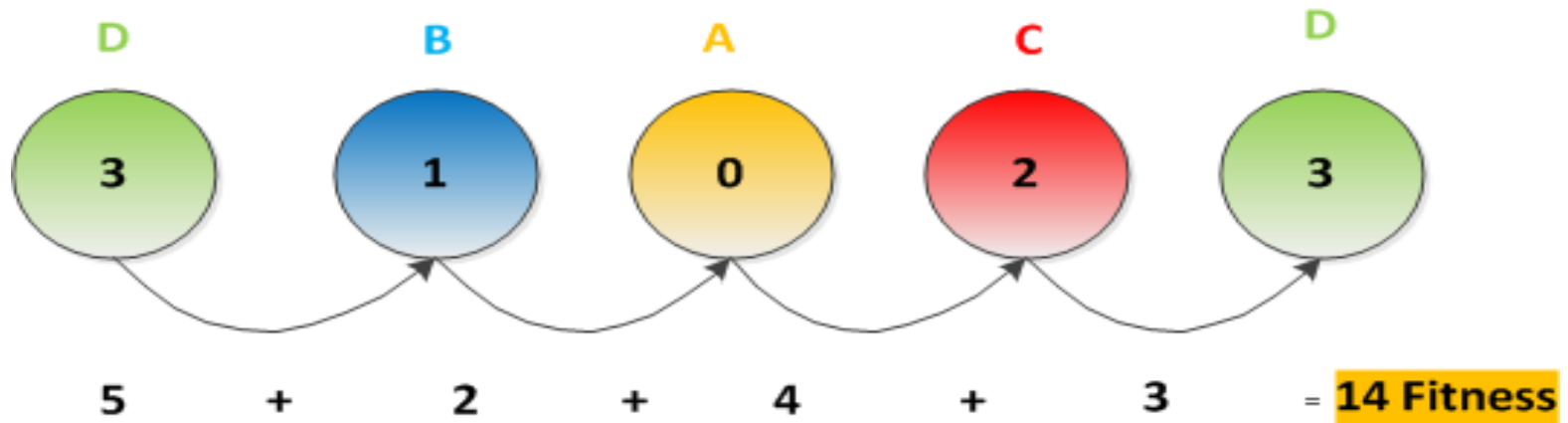
## ► Cálculo de Fitness

### ► Exemplo I:



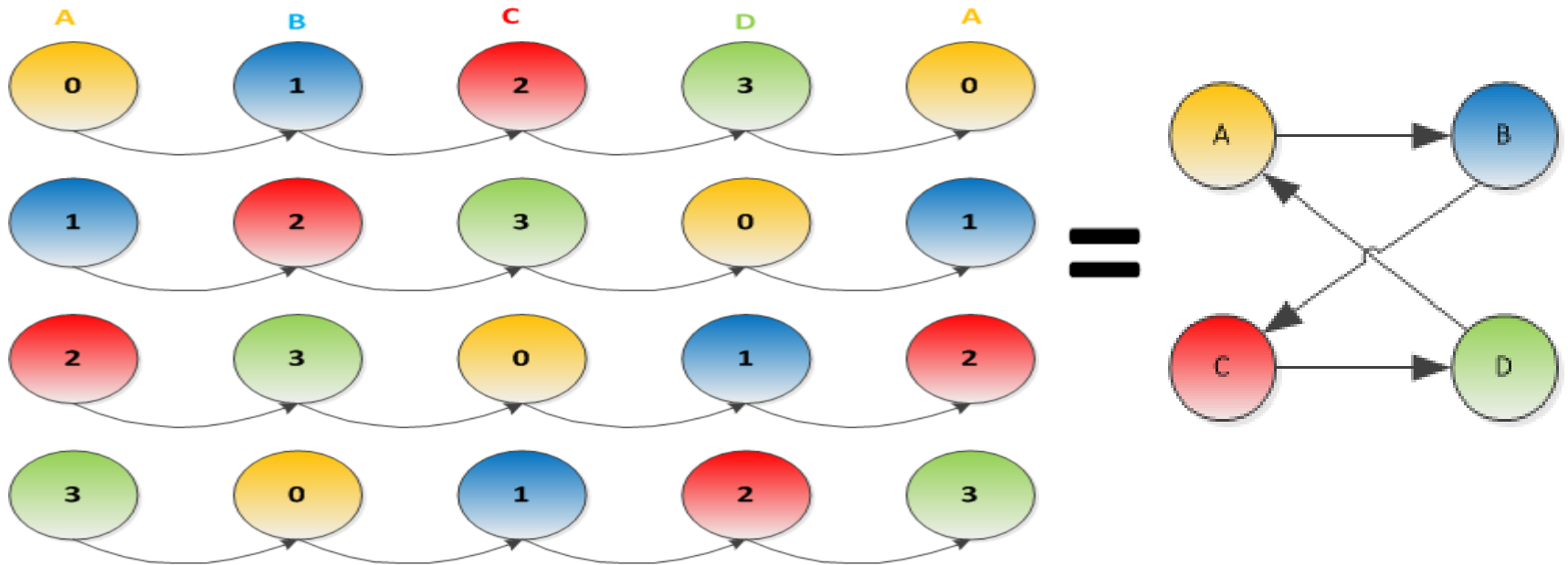
# Caixeiro Viajante

## Exemplo2:



# Caixeiro Viajante

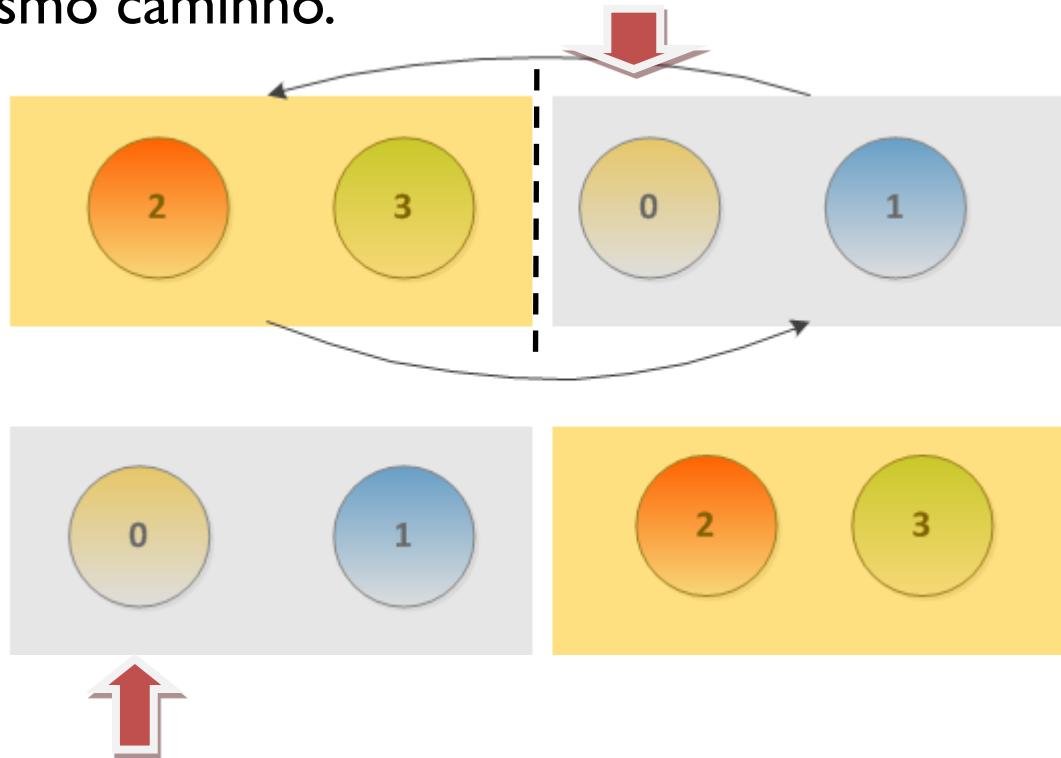
## ► Atenção:



- São sempre o mesmo caminho, a única diferença é o ponto de partida.

# Caixeiro Viajante

- ▶ Reparar o problema anterior:
  - ▶ Pegar na cidade **A** e colocá-la como primeira cidade através de uma rotação, eliminando assim diferentes representações para o mesmo caminho.

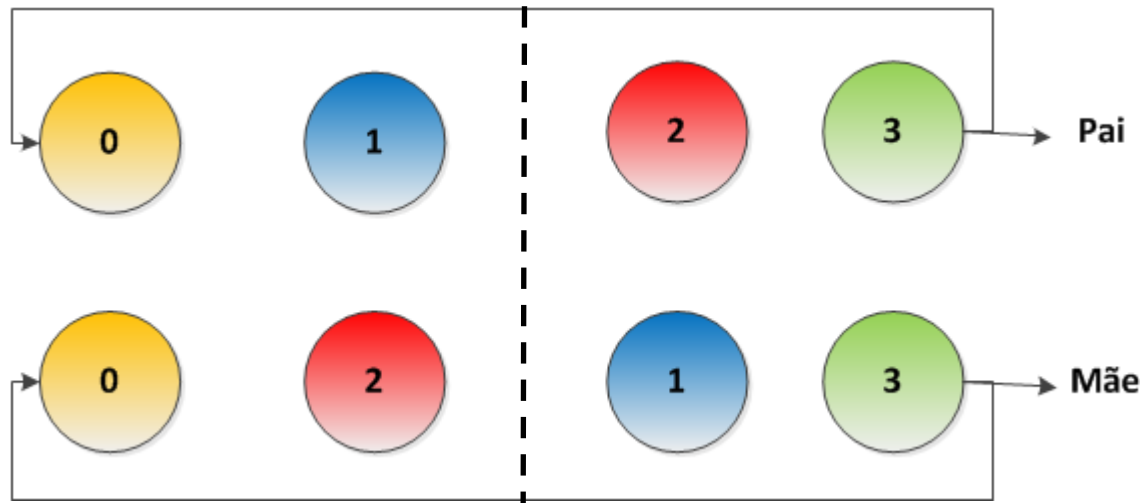


# Operador: OrderCrossover

## ► Descrição:

- Uma parte do pai é mapeada para uma porção da mãe. A partir da porção substituída, o resto dos genes são preenchidos mas omitindo os genes já presentes e respeitando a ordem com que eles se encontram.

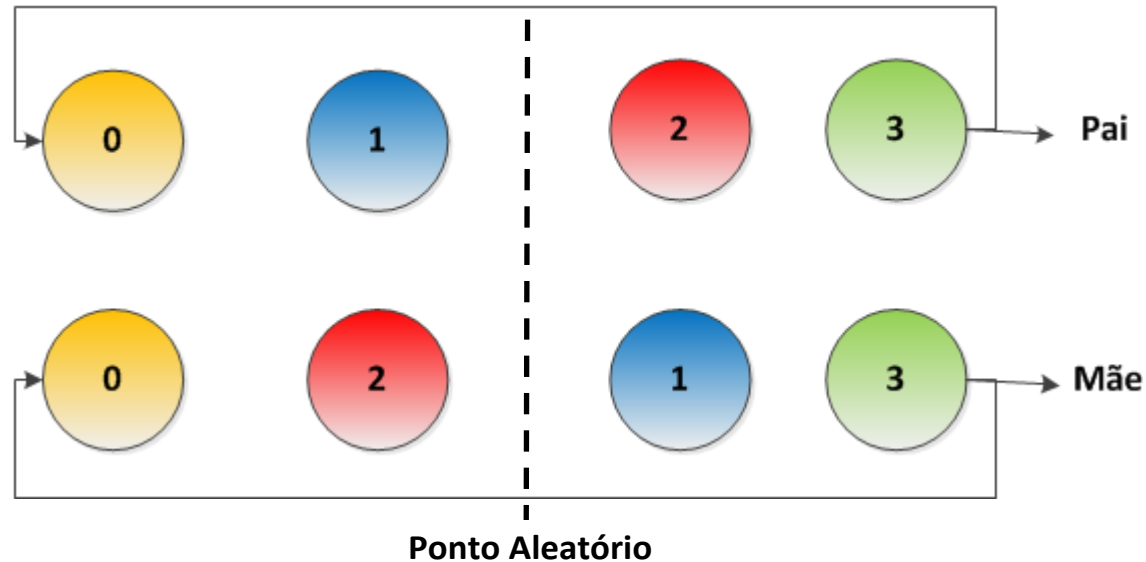
## ► Esquema:



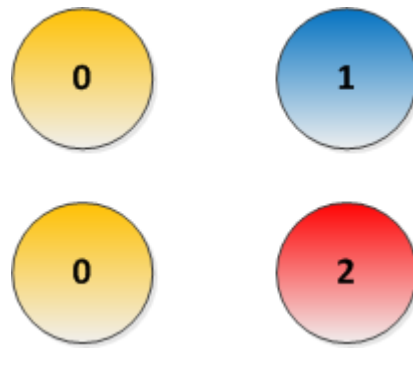


# Operador: OrderCrossover

► Esquema:



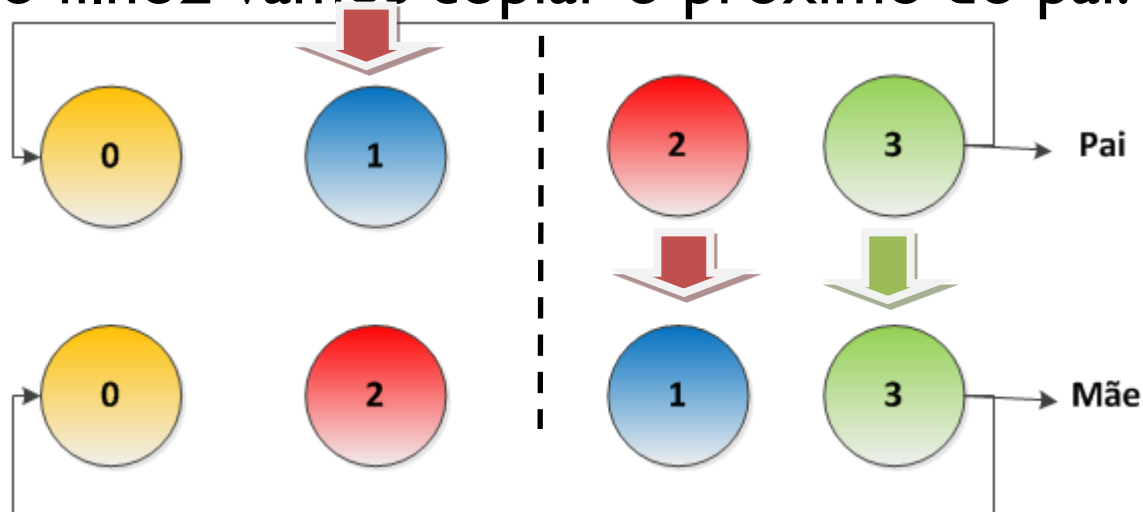
Filhos



Os genes do lado esquerdo  
são iguais aos dos pais.

# Operador: OrderCrossover

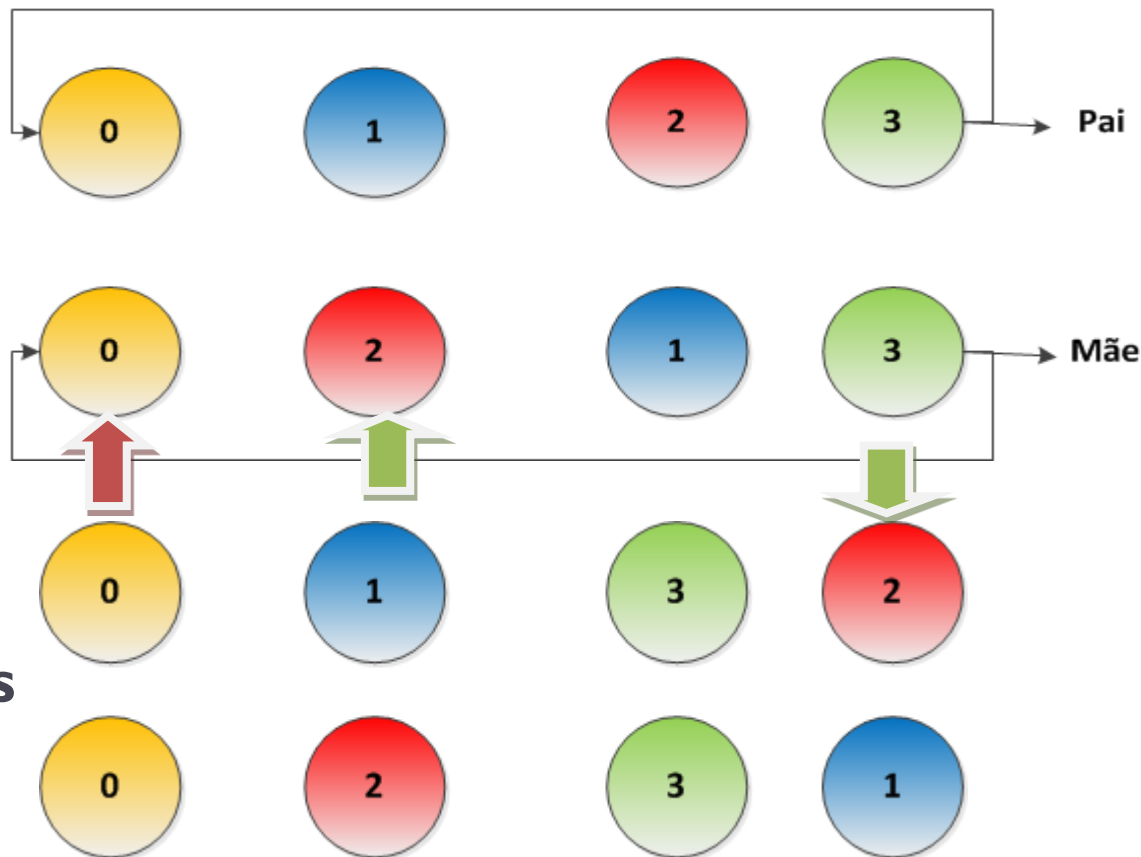
- Para o filho1 vamos copiar o próximo gene da mãe, e para o filho2 vamos copiar o próximo do pai.



- Como o filho1 já possuía o gene 1, não pode ficar com esse gene repetido, por isso coloca-se o próximo gene da mãe que não seja repetido.

# Operador: OrderCrossover

- Quando se chega ao fim do gene e ainda não colocámos todos os genes no filho, começamos no início do gene do pai e repetimos o processo até os filhos terem todos os genes.



# Operador: SUS - Minimização

---

## ► Problema:

- O operador SUS seleciona o fitness mais alto, por isso é preciso adaptá-lo para passar a selecionar o mais baixo e poder assim utilizá-lo no caixeiro viajante.

## ► Esquema:

### População

A	B	C	D	E	F	G	H	I	J
10	5	7	7	5	1	2	8	9	12

- $\text{Max}(\text{População}) = 12$
- $\text{Min}(\text{População}) = 1$



# Operador: SUS - Minimização

---

## ► PseudoCodigo : Calcular novo fitness

início

maxFitness := maiorFitness(individuos[])+1

```
para i := 0 ate i < tamanho(individuos[])  
    individuos[i].fitness := maxFitness-individuos[i].fitness  
    i := i+1
```

fim

maiorFitness(individuos[])

```
    max := 0  
    para i := 0 ate i < tamanho(individuos[])  
        se (max < individuos[i].fitness) entao  
            max := individuos[i].fitness  
        i := i+1
```

retorna max



# Operador: SUS - Minimização

---

- ▶ Novos valores de fitness:

	A	B	C	D	E	F	G	H	I	J
Fitness	10	5	7	7	5	1	2	8	9	12
Novo Fitness	3	8	6	6	8	12	11	5	4	1

- ▶ Novo Fitness =  $(12 + 1) - \text{Fitness}$
- ▶ Novo Fitness A =  $13 - 10 = 3$
- ▶ Fazendo esta conversão do fitness, o fitness com valor mais alto passa a ter o valor mais baixo, o que torna possível a utilização do operador SUS.



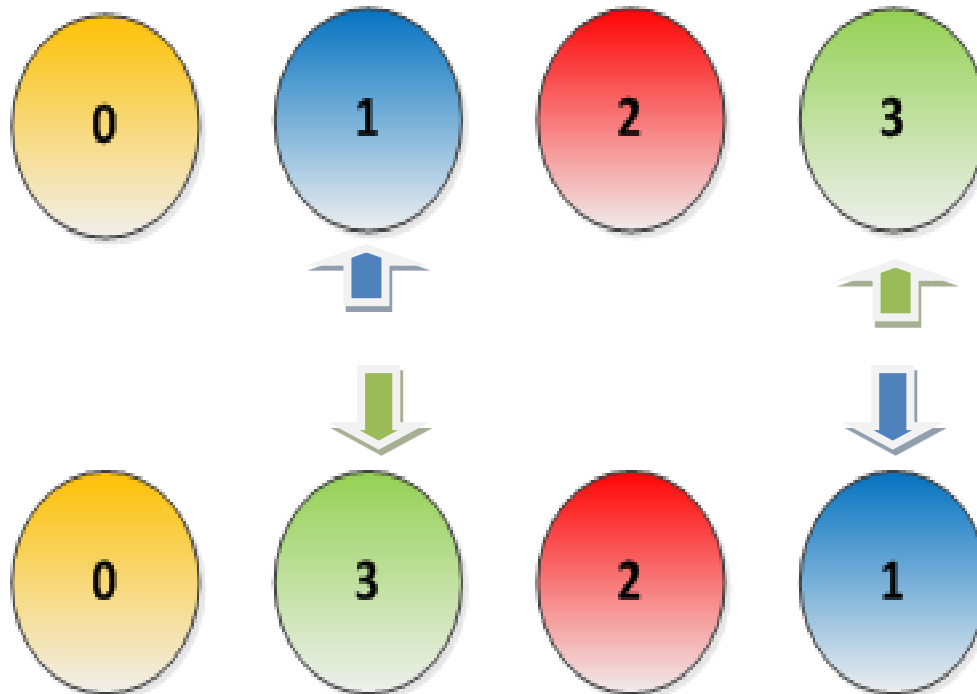
# Operador: Swap Genes - Mutaç o

---

## ► Descri  o:

- Escolher dois alelos aleat rios de um indiv duo, e troc -los de s tio.

## ► Esquema:



# Operador: Swap Genes - Mutação

---

- ▶ Probabilidade de mutação:
  - ▶ Probabilidade =  $1 / \text{TamanhoCromossoma}$

