

# Programação Científica

---

Prof. Dr. Danilo H. Perico

# OTIMIZAÇÃO

---

# RECOZIMENTO SIMULADO

## *SIMULATED ANNEALING*

---

# SIMULATED ANNEALING

- Embora existam variantes que podem melhorar a **subida de encosta**, todas compartilham a mesma falha: **quando o algoritmo atinge um máximo local, ele para de funcionar**
- O **Recozimento Simulado** permite que o algoritmo não fique preso se cair em um máximo local

# SIMULATED ANNEALING

- O **Recozimento Simulado** é um algoritmo **estocástico** de otimização
  - Padrão estocástico é aquele cujo estado é indeterminado, com origem em eventos aleatórios
  - Normalmente, qualquer sistema ou processo analisado com a teoria da probabilidade é estocástico
  - Por exemplo, o lançamento de uma dado resulta num processo estocástico, pois qualquer uma das 6 faces têm iguais probabilidades de ficar para cima

# SIMULATED ANNEALING

- O núcleo do método de Recozimento Simulado é a analogia com a termodinâmica, especificamente com a maneira como os metais resfriam e recozem
- A essência do processo de recozimento é o resfriamento lento, que garante que um estado de baixa energia seja alcançado
- Quando um metal líquido é resfriado rapidamente (*ou temperado*), ele acaba em um estado policristalino, com energia mais alta

# SIMULATED ANNEALING

- A analogia é a seguinte:
  - Se buscarmos otimizações gananciosas, que buscam encontrar a solução o mais rápido possível (*resfriamento rápido*), é bem provável que máximos ou mínimos locais sejam encontrados
  - Resfriamentos mais lentos (*recozimento*), podem levar a máximos ou mínimos globais

# SIMULATED ANNEALING

- No processo do Recozimento Simulado, utilizamos a seguinte metáfora:
  - No início, a temperatura é alta, sendo mais provável a tomada de decisões aleatórias
  - Conforme a temperatura diminui, torna-se menos provável a tomada de decisões aleatórias



# SIMULATED ANNEALING

- Este mecanismo permite que o algoritmo mude seu estado para um vizinho que, eventualmente, pode ser pior do que o estado atual, que é como ele pode escapar dos máximos/mínimos locais

# SIMULATED ANNEALING

- Por ser estocástico, o Recozimento Simulado toma decisões de usar ou não vizinhos que pioram a solução geral baseados na **distribuição de probabilidade de Boltzman**:
  - **$Prob(E) \sim \exp(-E/kT)$**
  - **E** é a energia
  - **T** é temperatura
  - **k** é uma constante que relaciona Energia e Temperatura

# SIMULATED ANNEALING

- **Distribuição de probabilidade de Boltzman:** descreve a maneira como uma dada energia distribui-se ao longo de um número grande de partículas em um dado sistema clássico em equilíbrio térmico
- A figura ao lado exibe a distribuição de velocidade de moléculas de oxigênio para três temperaturas distintas

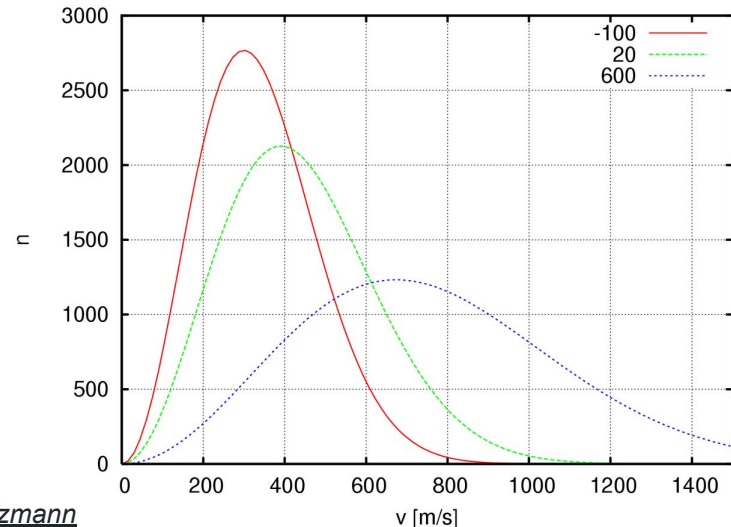


Figura: [https://pt.wikipedia.org/wiki/Distribui%C3%A7%C3%A3o\\_de\\_Maxwell-Boltzmann](https://pt.wikipedia.org/wiki/Distribui%C3%A7%C3%A3o_de_Maxwell-Boltzmann)

# SIMULATED ANNEALING

- Código em aula: exemplo com o Problema do Caixeiro Viajante (TSP)

# Exercício para Entrega

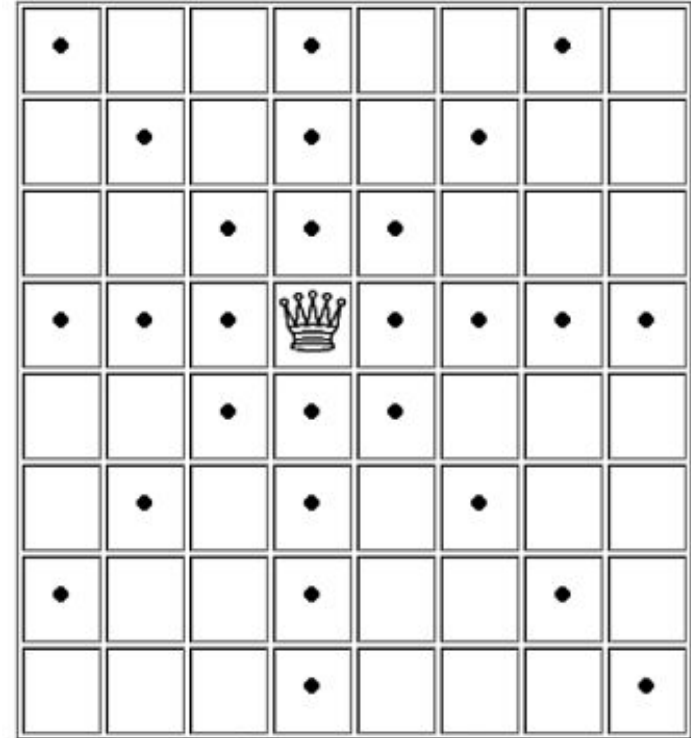
---

# EXERCÍCIO - OTIMIZAÇÃO

- **$n$  rainhas:**
- O problema das  **$n$**  rainhas consiste em posicionar  **$n$**  rainhas em um tabuleiro de mesa  **$n \times n$**  sem que nenhuma delas esteja na linha de ataque entre si
- As rainhas podem se mover horizontalmente, diagonalmente e verticalmente

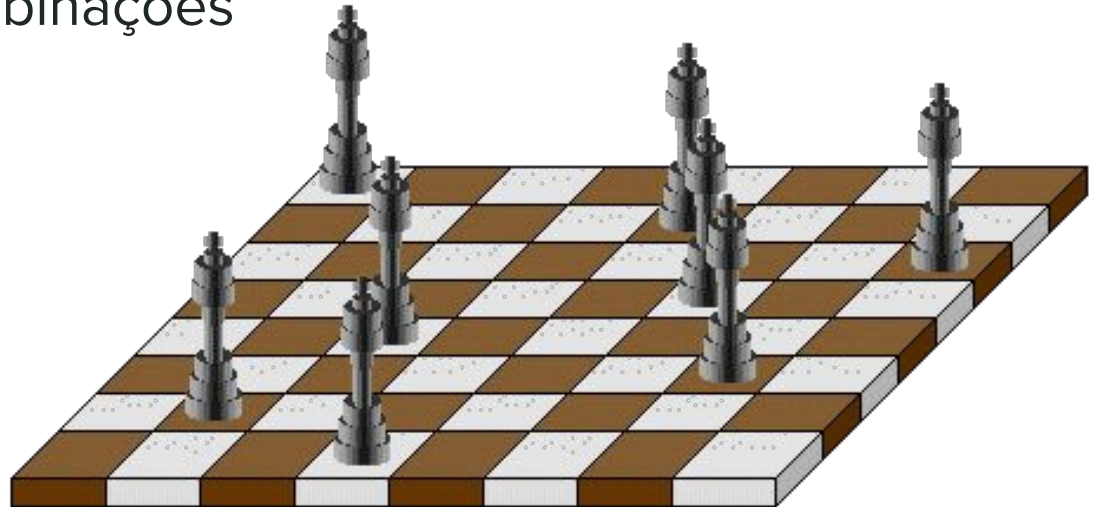
# EXERCÍCIO - OTIMIZAÇÃO

- ***Exemplo: 8 rainhas***
- Possíveis movimentações da rainha:



# EXERCÍCIO - OTIMIZAÇÃO

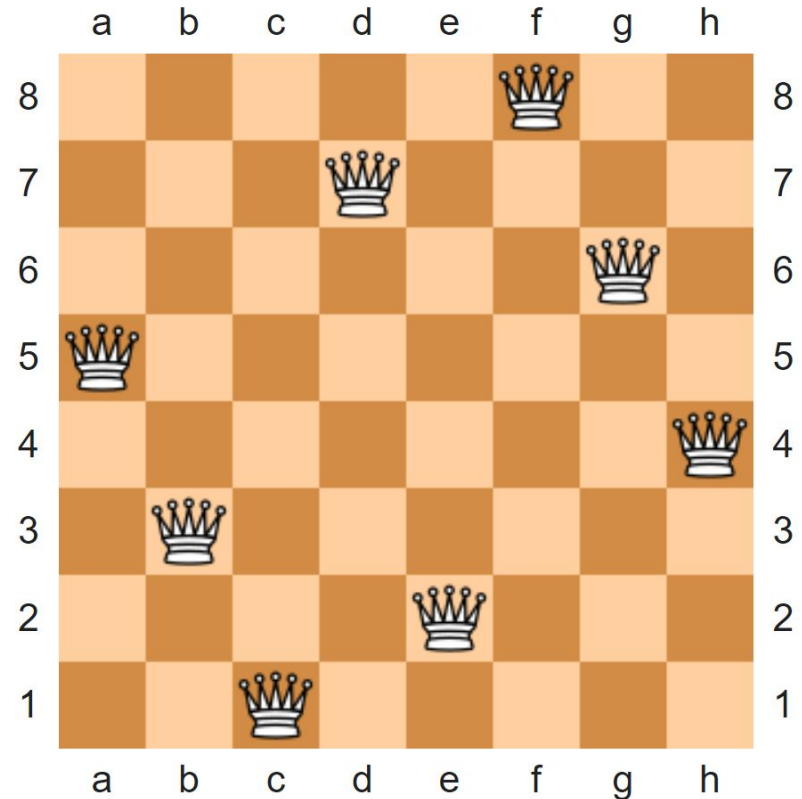
- ***Exemplo: 8 rainhas***
- Encontrar a solução para o problema não é fácil, pois existe um grande número de combinações





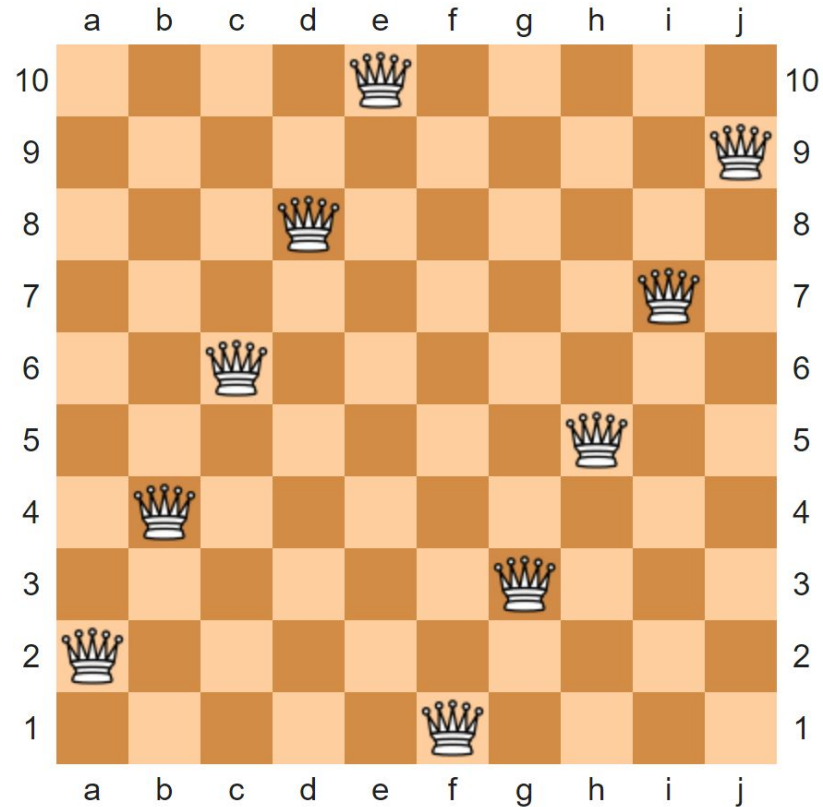
# EXERCÍCIO - OTIMIZAÇÃO

- **Exemplo: 8 rainhas**
- O problema com 8 rainhas apresenta 4.426.165.368 possíveis disposições das 8 rainhas *combinação:  $C(64,8) = 64! / ((64-8)! \times (8!))$*
- Porém, somente 92 soluções
- Uma possível solução pode ser vista ao lado



# ENTREGA - OTIMIZAÇÃO

- Encontrar **uma** solução do problema com **10 rainhas** (*tabuleiro 10x10*)
- $C(100,10) = 100!/((100-10)! \times (10!)) = 1.73 \times 10^{13}$
- Existem 724 soluções nesse caso



# ENTREGA - OTIMIZAÇÃO

- Utilize as seguintes abordagens no problema:
  - Força Bruta
  - Subida de Encosta
  - Recozimento Simulado
- Compare todos os métodos com relação à quantidade de iterações e tempo para encontrar uma solução

# ENTREGA - OTIMIZAÇÃO

- Entregue o código comentado em Python:
  - Arquivo *.ipynb*

# ALGORITMO GENÉTICO - AG

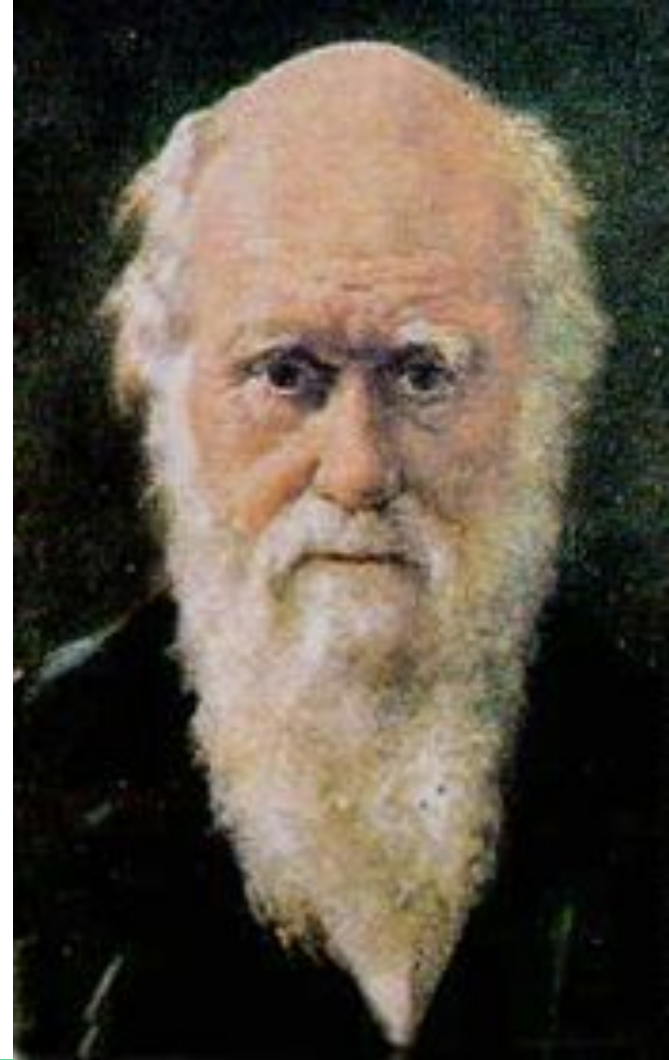
## *GENETIC ALGORITHM - GA*

---



# Teoria da Evolução

- 1859: **Charles Darwin**
  - Existe uma diversidade de seres devido aos contingentes da natureza (comida, clima, ...) e é pela lei da **Seleção Natural** que os seres mais adaptados ao seus ambientes sobrevivem
  - Características adquiridas são herdadas pelas gerações seguintes





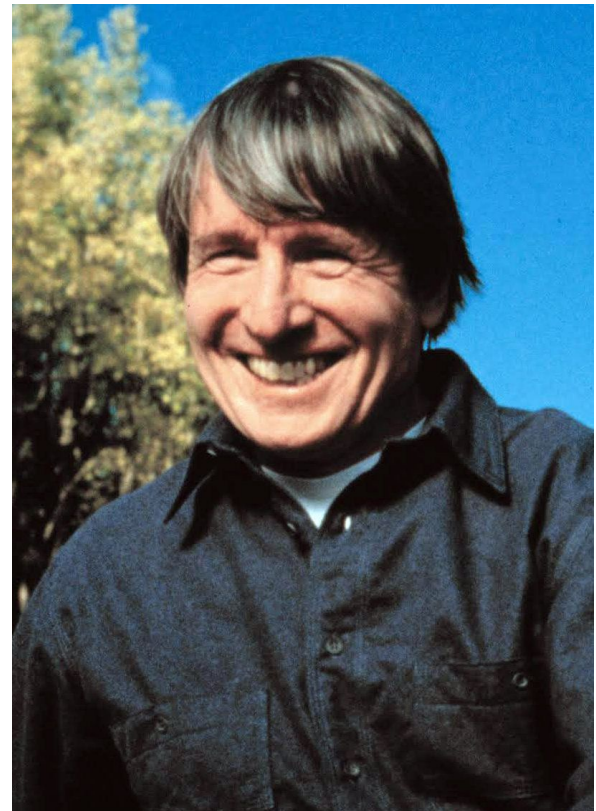
## Teoria da Evolução

*“Quanto melhor um indivíduo se adaptar ao seu meio ambiente, maior será sua chance de sobreviver e gerar descendentes.”*

(DARWIN, 1859)

# Surgimento dos Algoritmos Genéticos

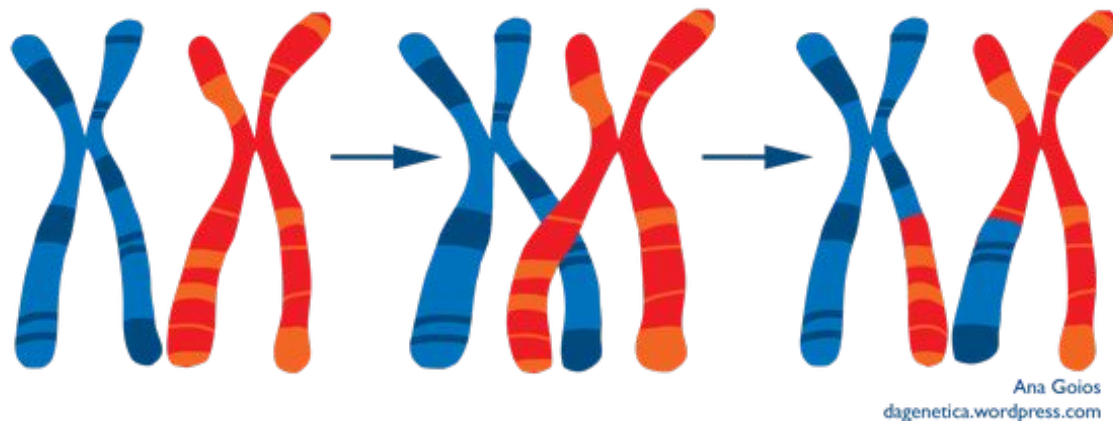
- **John Holland** Idealizou os algoritmos genéticos (1975)
- Por que evolução ?
  - Muitos problemas computacionais:
    - Envolvem busca através de um grande número de possíveis soluções
    - Requerem que o programa seja adaptativo, apto a agir em um ambiente dinâmico
  - A evolução biológica
    - uma busca paralela em um enorme espaço de problemas





# Algoritmos Genéticos - Introdução

- Um Algoritmo Genético (AG) é uma técnica de otimização utilizada na ciência da computação para achar soluções aproximadas em problemas complexos
- Algoritmos Genéticos são **Algoritmos Bioinspirados Evolutivos**



# Algoritmos Genéticos - Introdução

- **indivíduo** = uma possível solução
- muda-se os indivíduos por intermédio de *reprodução* e *mutação*
- *seleciona-se* indivíduos mais adaptados através de sucessivas gerações
- A **avaliação** de cada indivíduo é medida pela “função de aptidão” (*fitness function*)

# Algoritmos Genéticos

## Pseudocódigo

```
1 ✓ Procedimento AG{
2     t = 0;
3     inicia_populacao(P, t)
4     avaliacao(P, t)
5 ✓   repita até(t=d){
6       t = t + 1
7       selecao_dos_pais(P, t)
8       reproducao(P, t)
9       mutacao(P, t)
10      avaliacao(P, t)
11      sobrevivem(P, t)
12    }
13  }
14
15  t = tempo atual
16  d = tempo determinado para finalizar o algoritmo
17  P - populacao
```

Fonte: <https://sites.icmc.usp.br/andre/research/genetic/>

# População Inicial

- Dado um problema, precisamos definir como será a representação dos **indivíduos** (*possíveis soluções*)
- Um **indivíduo** é um **Cromossomo**
  - Os parâmetros do problema de otimização são representados por cadeias de valores
  - Exemplos:
    - Vetores de reais: (2.345, 4.3454, 5.1, 3.4)
    - Cadeias de bits: (111011011)
    - Vetores de inteiros: (1,4,2,5,2,8)

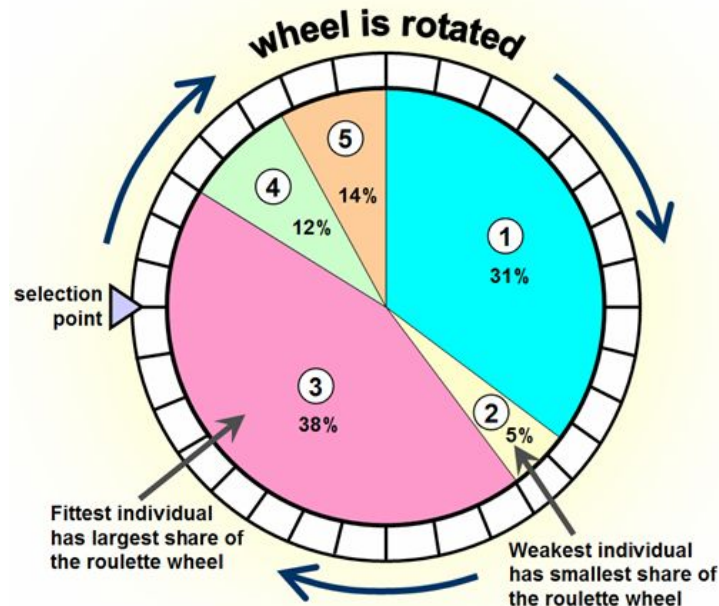
# Função de Avaliação (*Fitness Function*)

- É feita através de uma função que melhor representa o problema e tem por objetivo fornecer uma **medida de aptidão** de cada indivíduo na população corrente que irá dirigir o processo de busca
- Aptidão é uma nota associada ao indivíduo que **avalia quão boa é a solução** por ele representada

# Seleção dos Pais

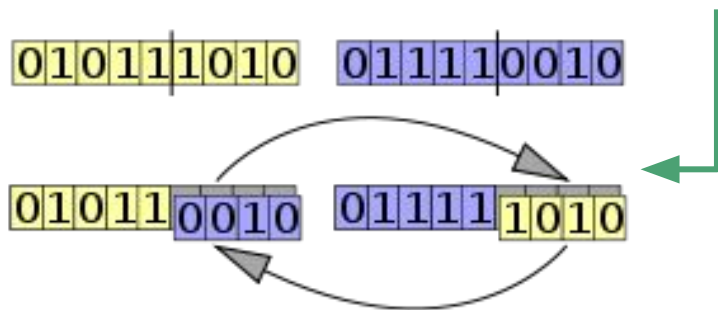
- Como selecionar indivíduos para **Reprodução** ?
- **Método da Roleta**
  - Selecionar indivíduos aleatoriamente, proporcionando chances de reprodução aos mais aptos

Chances de seleção são proporcionais à aptidão



# Reprodução ou Recombinação

- A reprodução/recombinação visa a **combinar** e/ou perpetuar material genético dos indivíduos mais adaptados
  - Tipos:
    - assexuada (**duplicação**) ou sexuada (**crossover**)



- Observação: Existe uma “*taxa de crossover*” que controla a quantidade de reprodução que será feita

# Mutação

- Gera **diversidade** (para escapar de ótimos locais)

depois da reprodução:

010111010 011110010

010110010 011111010

mutação:

010111010 011110010

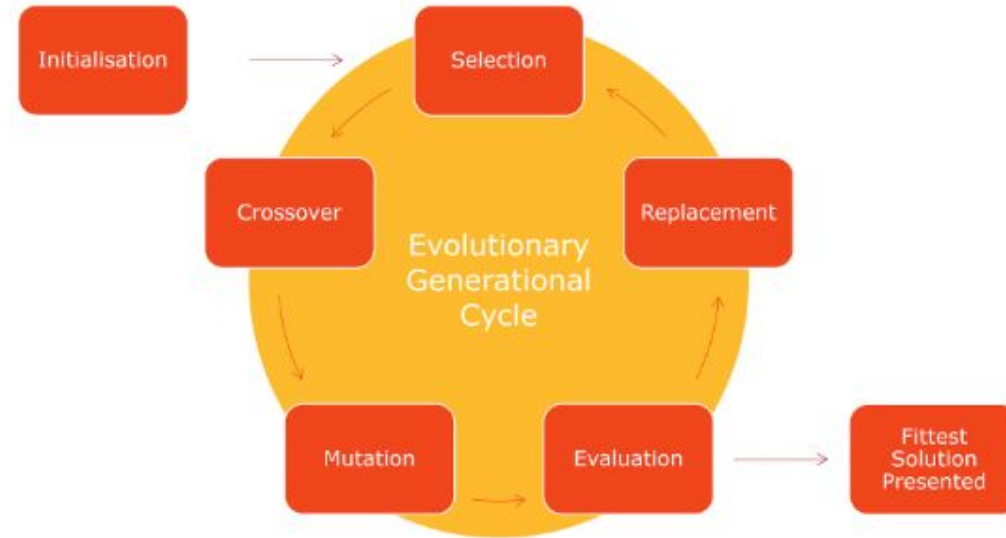
010110010 011111011



- Observação: Existe uma “*taxa de mutação*” que diminui com o tempo (e.g. % da população selecionada) para garantir convergência



# Ciclo do Algoritmo Genético

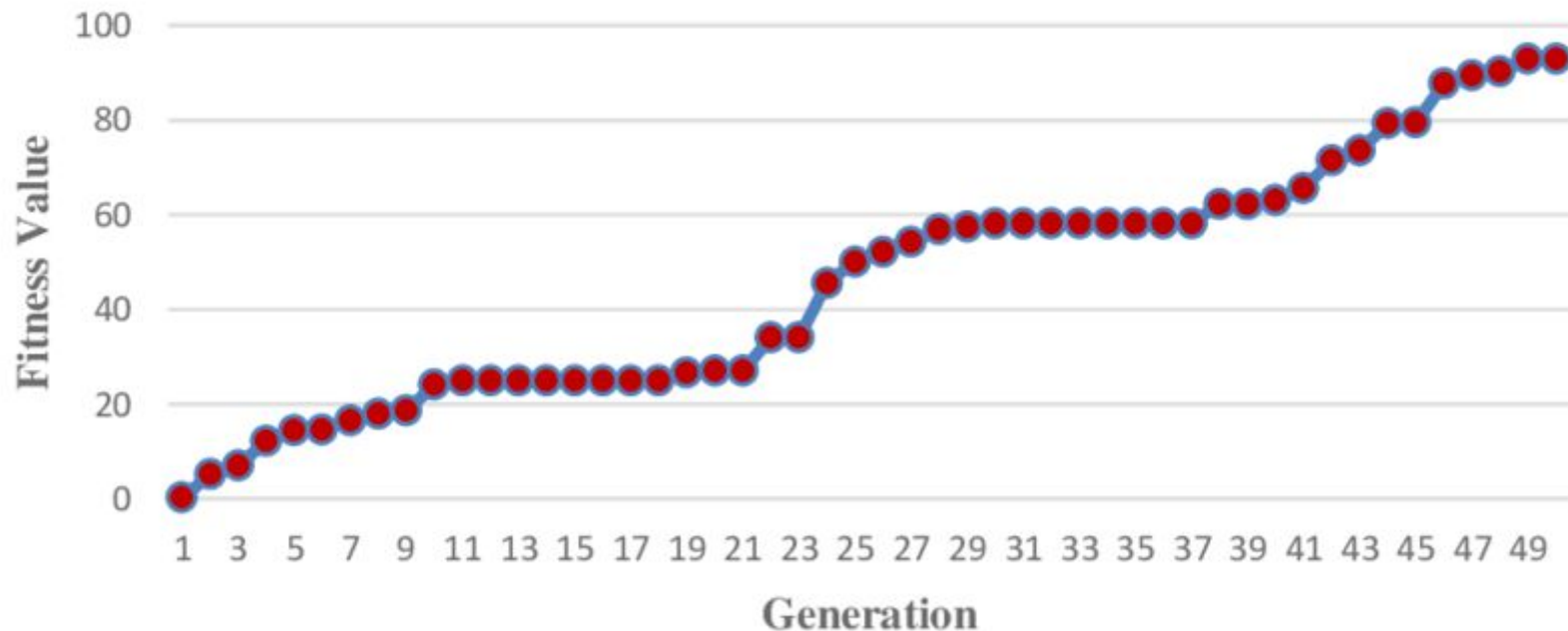


# Como escolher o que fica em cada nova geração?

- Troca de toda população
  - Isso significa que a **população 2** substitui a **população 1**
  - A cada ciclo,  $N/2$  pares são escolhidos gerando  $M=N$  descendentes
- Elitismo
  - A **população 2** substitui a **população 1** ( $M= N-1$ )
  - Acrescenta-se o mais apto da **população 1** na **população 2**
- *Steady State*
  - Gera-se  $M<N$  indivíduos e esses  $M$  substituem os  $N$  piores do conjunto da **população 1** – Existe também o Steady state sem duplicados

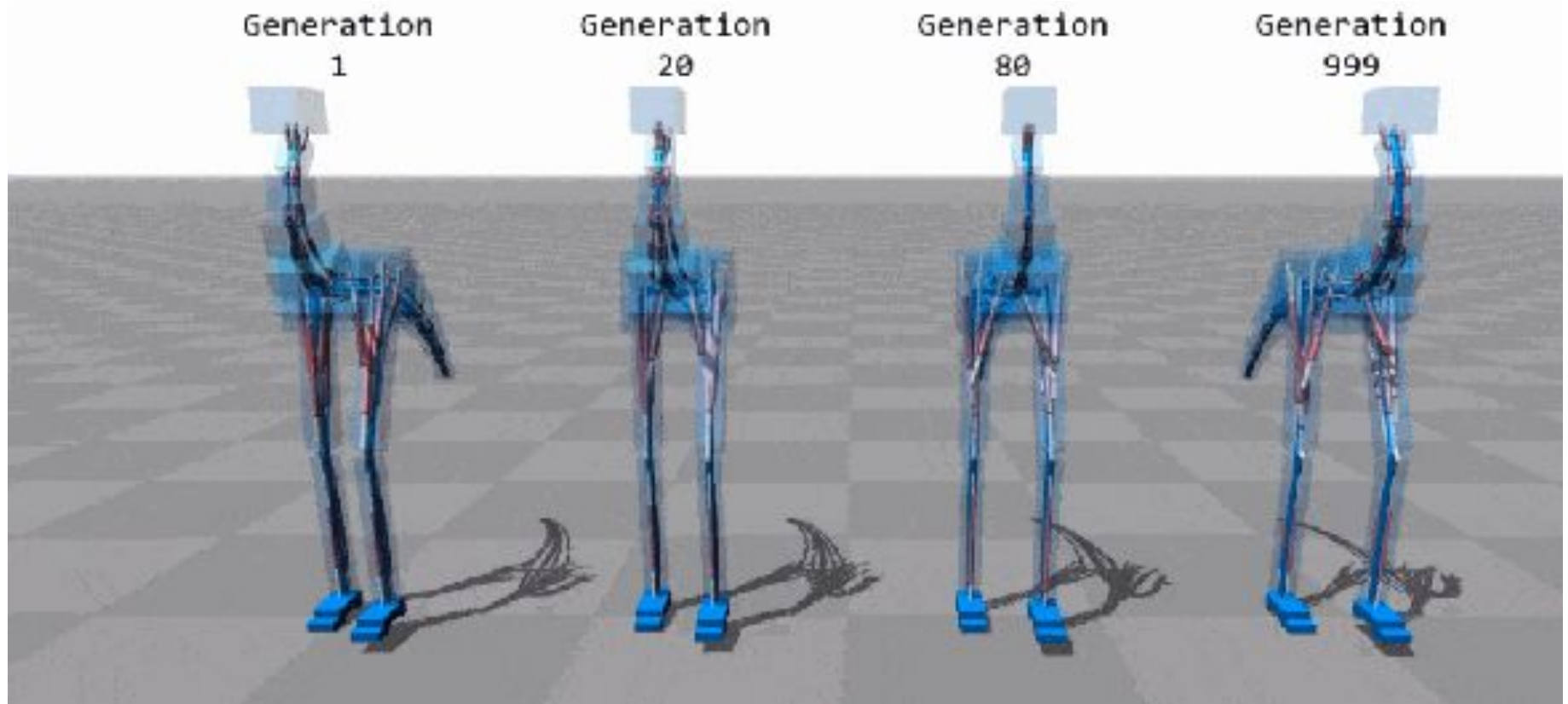


# Convergência das gerações



Fonte:  
[https://www.researchgate.net/publication/332578752\\_An\\_Approach\\_based\\_on\\_Genetic\\_Algorithm\\_for\\_Multi-tenant\\_Resource\\_Allocation\\_in\\_SaaS\\_Applications](https://www.researchgate.net/publication/332578752_An_Approach_based_on_Genetic_Algorithm_for_Multi-tenant_Resource_Allocation_in_SaaS_Applications)

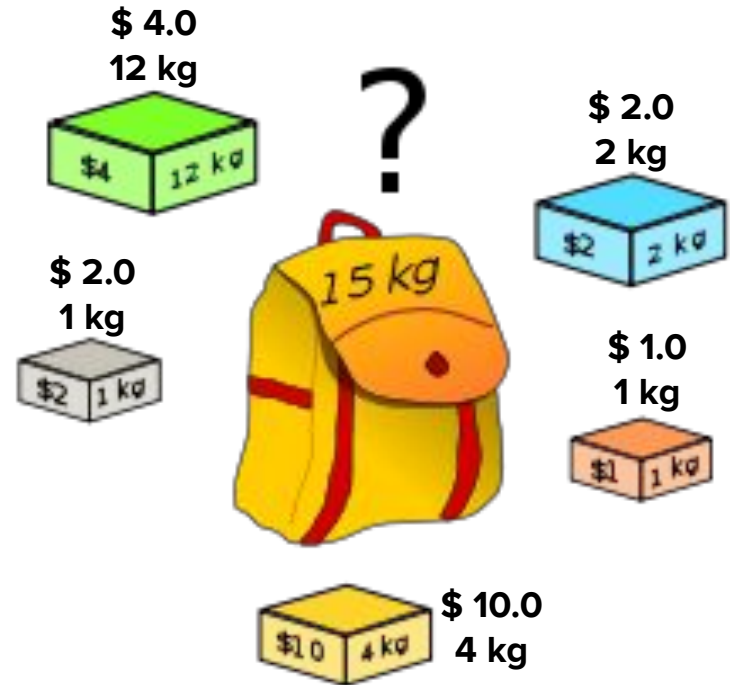
# Algoritmos Genéticos - Após gerações



# Exemplo:

## Problema da Mochila sem Repetições (Knapsack problem)

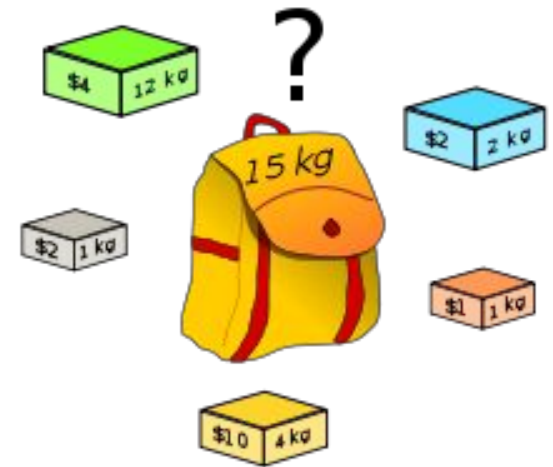
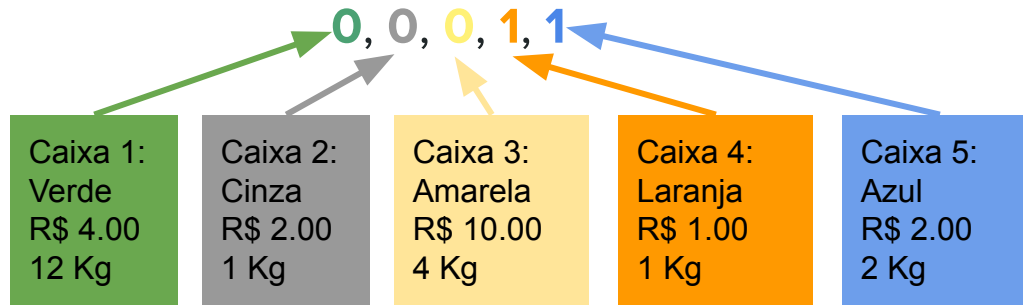
- Problema de **otimização combinatória**
- 5 caixas com pesos e valores distintos
- Quais caixas devem ser escolhidas para **maximizar** a quantidade de dinheiro, **mantendo o peso total abaixo ou igual a 15 kg**?



# Exemplo:

## Problema da Mochila sem Repetições (*Knapsack problem*)

- Indivíduo: cada indivíduo deve representar uma possível solução ao problema
  - Só podemos colocar uma caixa de cada na mochila (sem repetições)
  - Exemplo de **1 indivíduo**:



# Exemplo:

## Problema da Mochila sem Repetições

- Tamanho da população **6 indivíduos**

Verde	Cinza	Amarelo	Laranja	Azul
0	0	0	1	1
0	1	0	0	0
1	1	1	0	1
1	1	1	1	1
0	0	1	0	1
0	1	0	1	0

Exemplo:

## Problema da Mochila sem Repetições

- Função de Aptidão  $f()$ :
  - Valor total em \$ de cada possível resposta
  - Se peso > 15: valor = 0

Verde	Cinza	Amarelo	Laranja	Azul	$f()$
0	0	0	1	1	3.00
0	1	0	0	0	2.00
1	1	1	0	1	0.00
1	1	1	1	1	0.00
0	0	1	0	1	12.00
0	1	0	1	0	5.00



Exemplo:

## Problema da Mochila sem Repetições

- Seleção dos Melhores Indivíduos
  - Roleta:

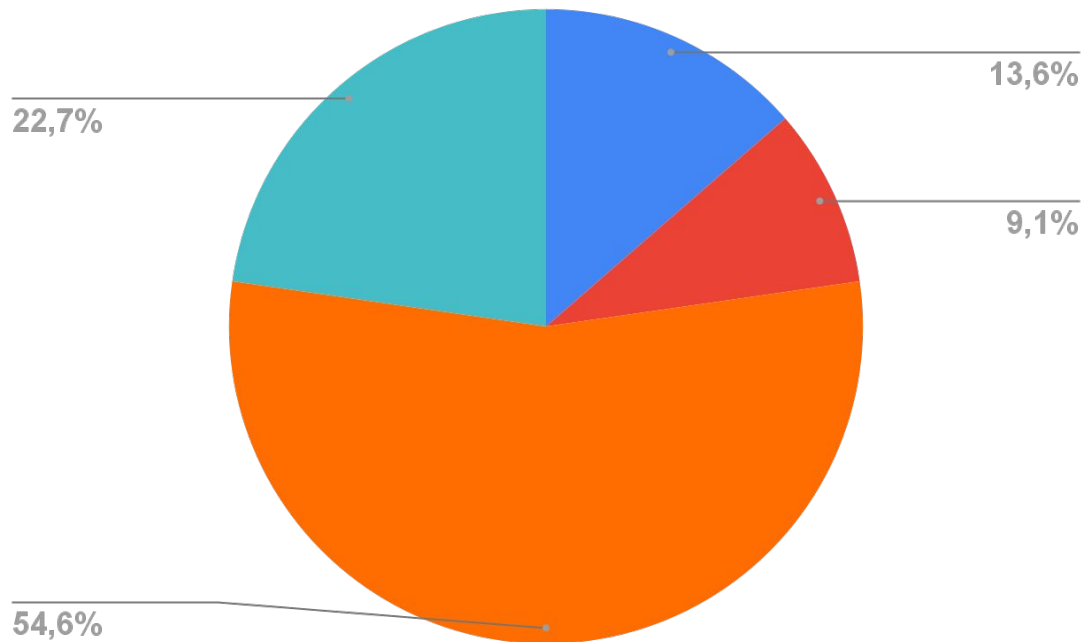
Verde	Cinza	Amarelo	Laranja	Azul	f( )	%
0	0	0	1	1	3.00	13.6%
0	1	0	0	0	2.00	9.1%
1	1	1	0	1	0.00	0.0%
1	1	1	1	1	0.00	0.0%
0	0	1	0	1	12.00	54.5%
0	1	0	1	0	5.00	22.7%

**22.00**

Exemplo:

## Problema da Mochila sem Repetições

- Seleção dos Melhores Indivíduos
  - Roleta:



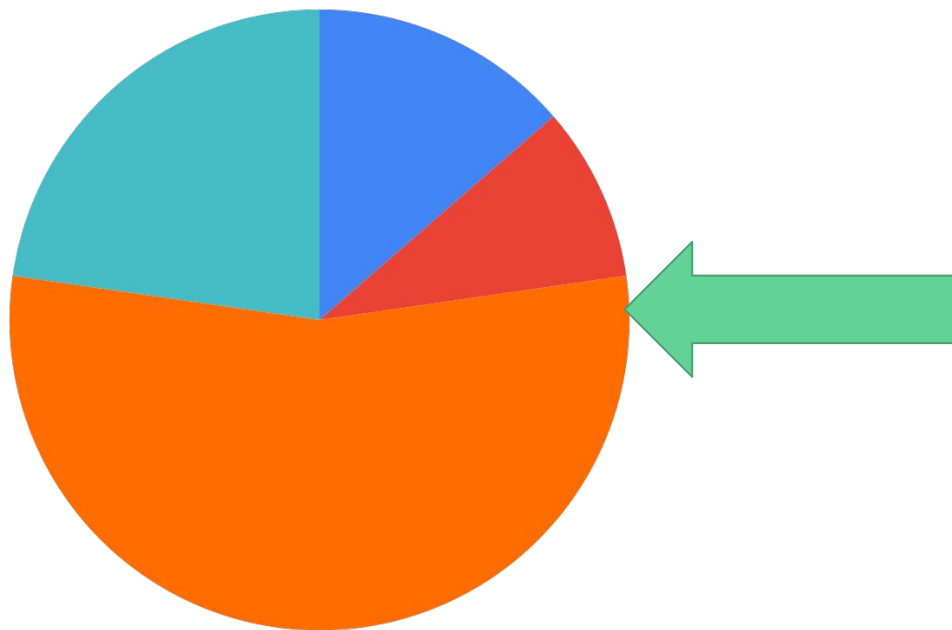
Exemplo:

## Problema da Mochila sem Repetições

- Seleção dos Melhores Indivíduos
  - Roleta:

Girar a roleta para  
selecionar 6  
indivíduos  
novamente

1º giro



Exemplo:

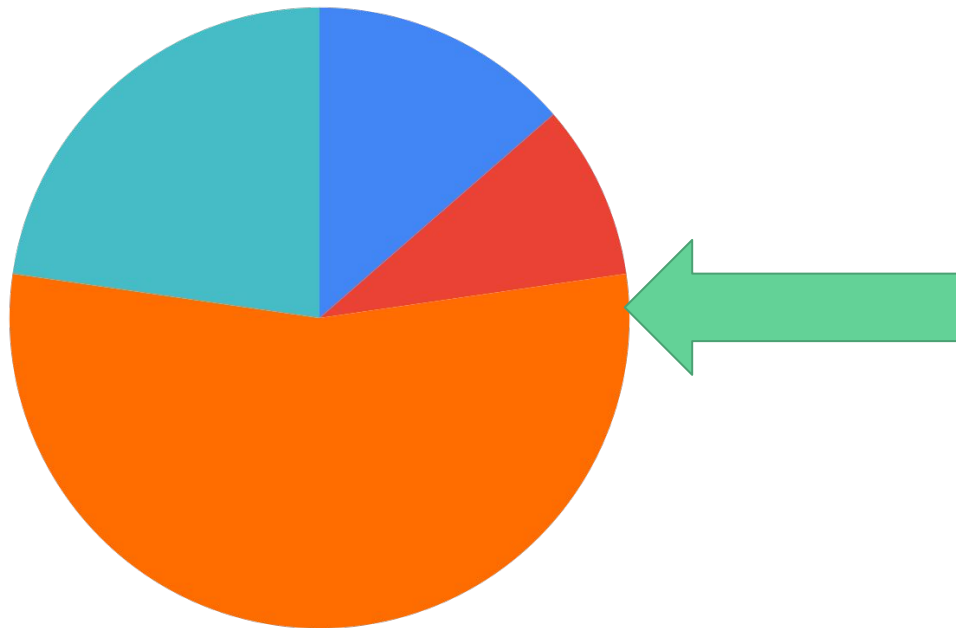
## Problema da Mochila sem Repetições

- Seleção dos Melhores Indivíduos
  - Roleta:

Girar a roleta para  
selecionar 6  
indivíduos  
novamente

1º giro

1º giro: laranja



Exemplo:

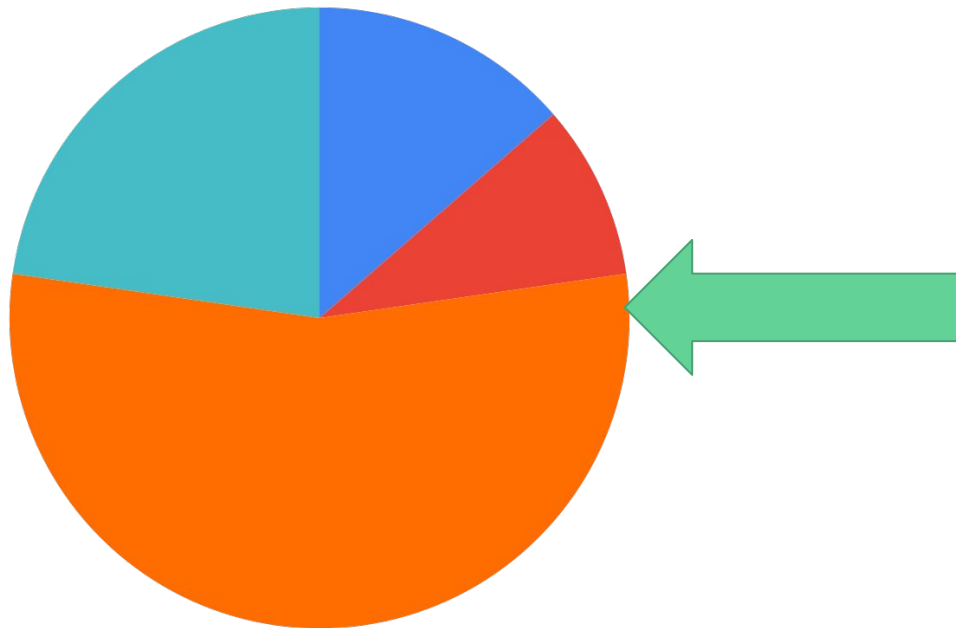
## Problema da Mochila sem Repetições

- Seleção dos Melhores Indivíduos
  - Roleta:

Girar a roleta para  
selecionar 6  
indivíduos  
novamente

2º giro

1º giro: laranja



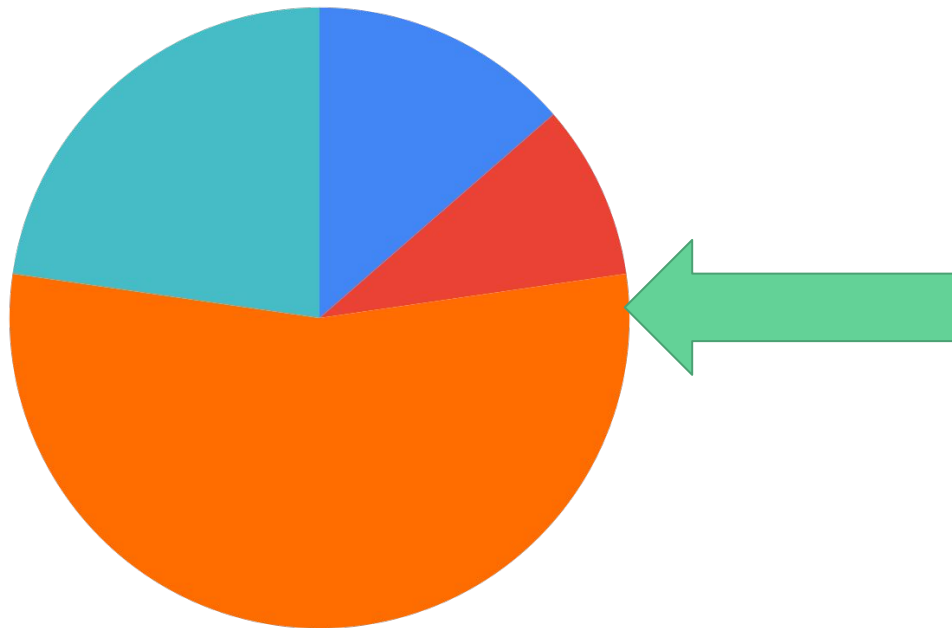
Exemplo:

## Problema da Mochila sem Repetições

- Seleção dos Melhores Indivíduos
  - Roleta:

Girar a roleta para  
selecionar 6  
indivíduos  
novamente

2º giro



1º giro: laranja  
2º giro: laranja

Exemplo:

## Problema da Mochila sem Repetições

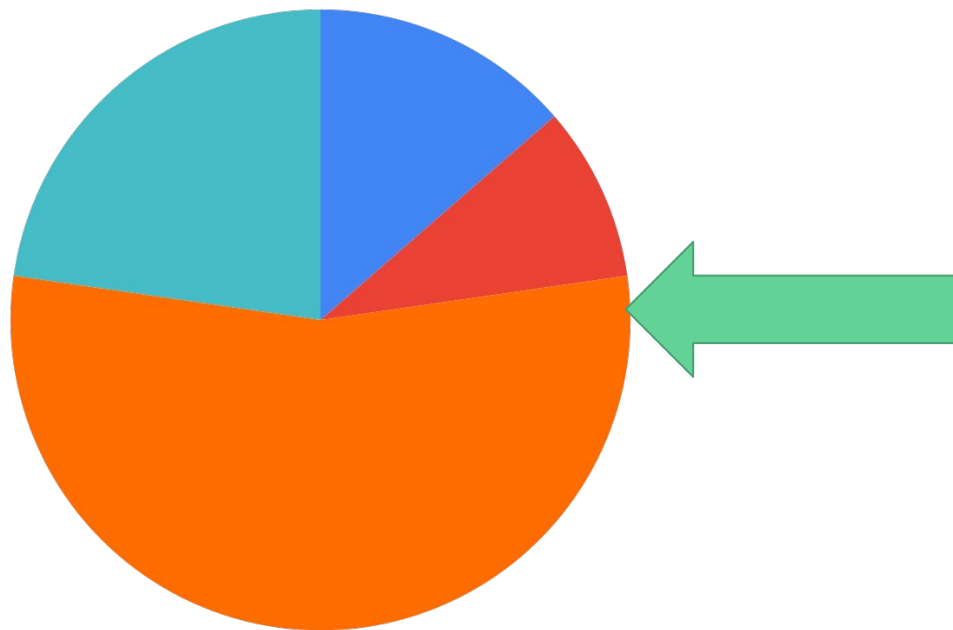
- Seleção dos Melhores Indivíduos
  - Roleta:

Girar a roleta para  
selecionar 6  
indivíduos  
novamente

3º giro

1º giro: laranja

2º giro: laranja



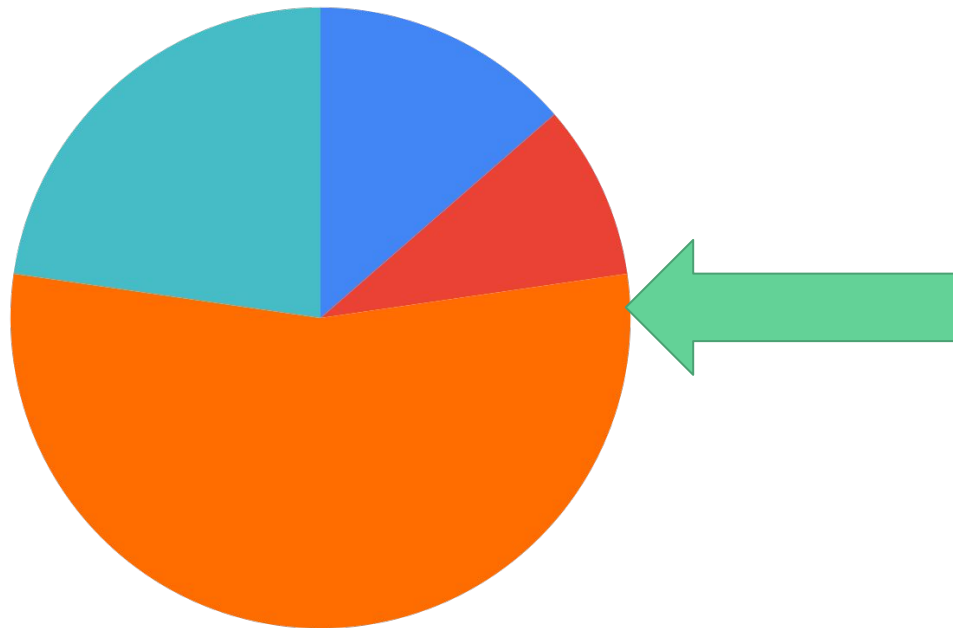
Exemplo:

## Problema da Mochila sem Repetições

- Seleção dos Melhores Indivíduos
  - Roleta:

Girar a roleta para  
selecionar 6  
indivíduos  
novamente

3º giro



1º giro: laranja

2º giro: laranja

3º giro: laranja



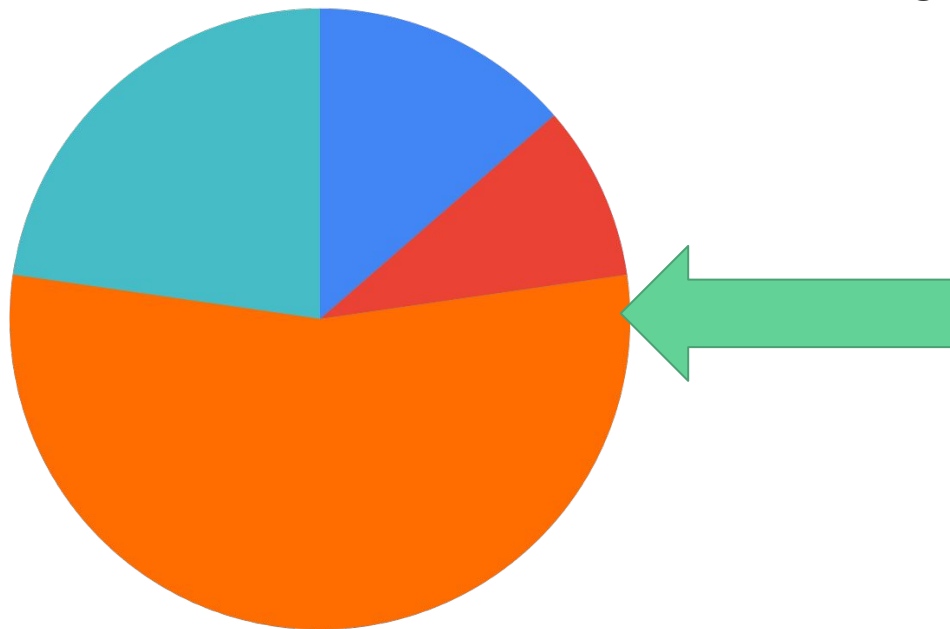
Exemplo:

## Problema da Mochila sem Repetições

- Seleção dos Melhores Indivíduos
  - Roleta:

Girar a roleta para  
selecionar 6  
indivíduos  
novamente

4º giro



1º giro: laranja

2º giro: laranja

3º giro: laranja

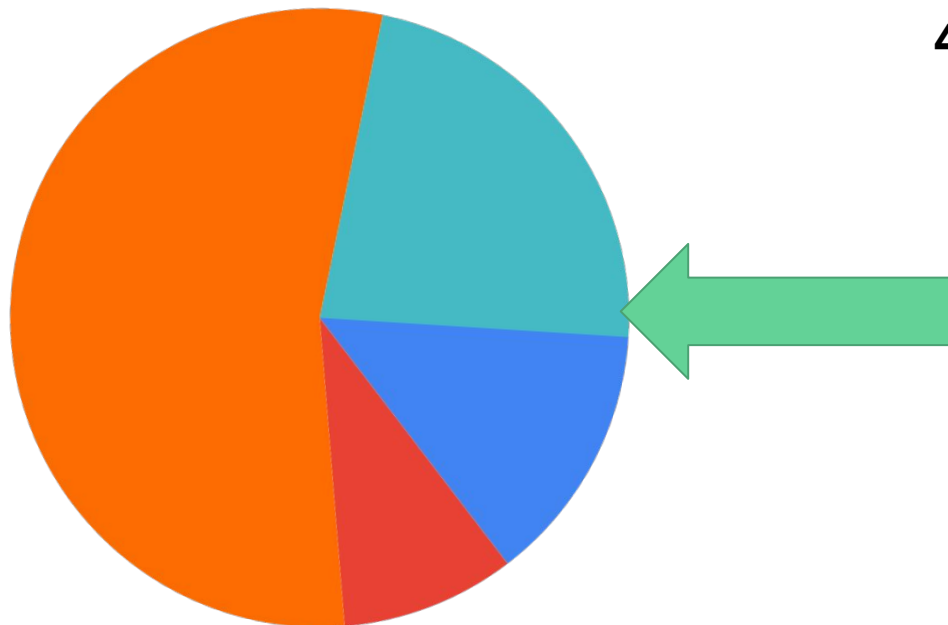
Exemplo:

## Problema da Mochila sem Repetições

- Seleção dos Melhores Indivíduos
  - Roleta:

Girar a roleta para  
selecionar 6  
indivíduos  
novamente

4º giro



1º giro: laranja  
2º giro: laranja  
3º giro: laranja  
4º giro: verde

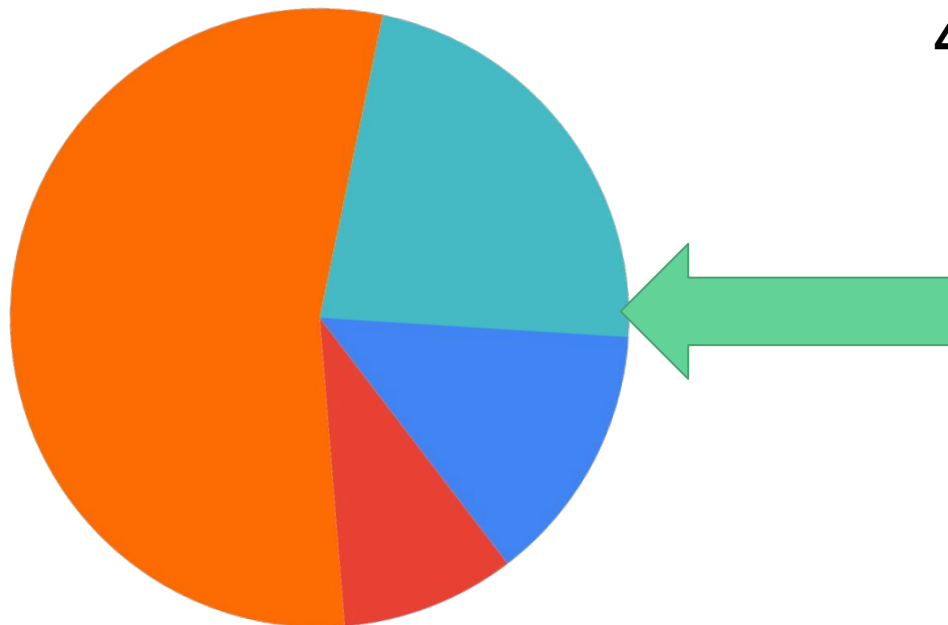
Exemplo:

## Problema da Mochila sem Repetições

- Seleção dos Melhores Indivíduos
  - Roleta:

Girar a roleta para  
selecionar 6  
indivíduos  
novamente

5º giro



1º giro: laranja  
2º giro: laranja  
3º giro: laranja  
4º giro: verde

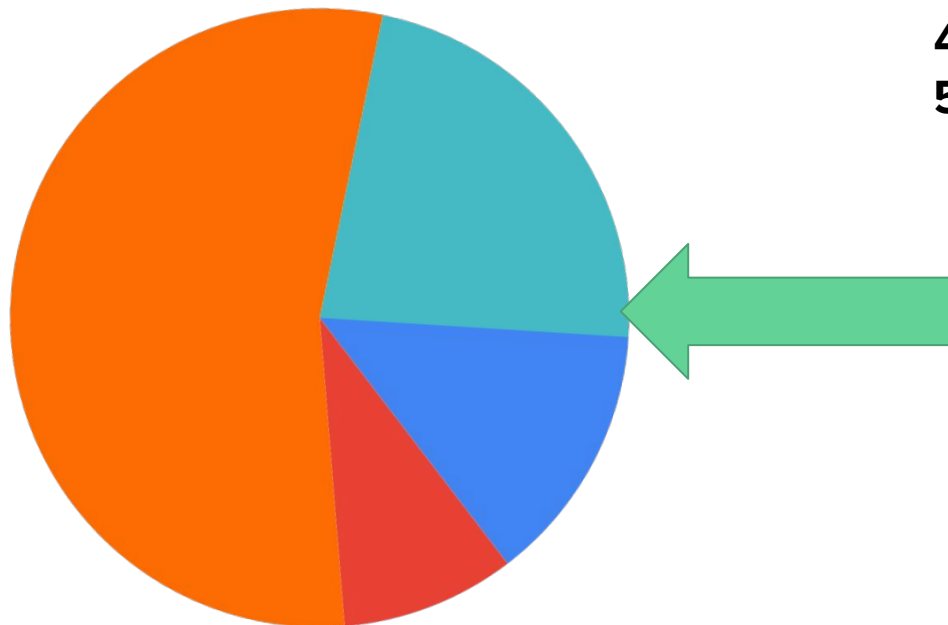
Exemplo:

## Problema da Mochila sem Repetições

- Seleção dos Melhores Indivíduos
  - Roleta:

Girar a roleta para  
selecionar 6  
indivíduos  
novamente

5º giro



1º giro: laranja

2º giro: laranja

3º giro: laranja

4º giro: verde

5º giro: verde

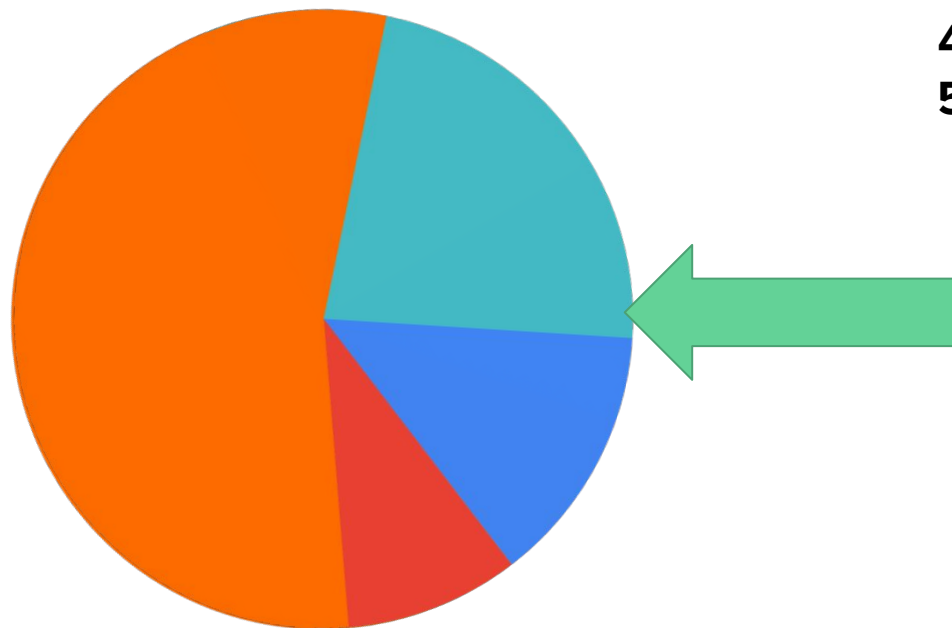
Exemplo:

## Problema da Mochila sem Repetições

- Seleção dos Melhores Indivíduos
  - Roleta:

Girar a roleta para  
selecionar 6  
indivíduos  
novamente

6º giro



1º giro: laranja

2º giro: laranja

3º giro: laranja

4º giro: verde

5º giro: verde

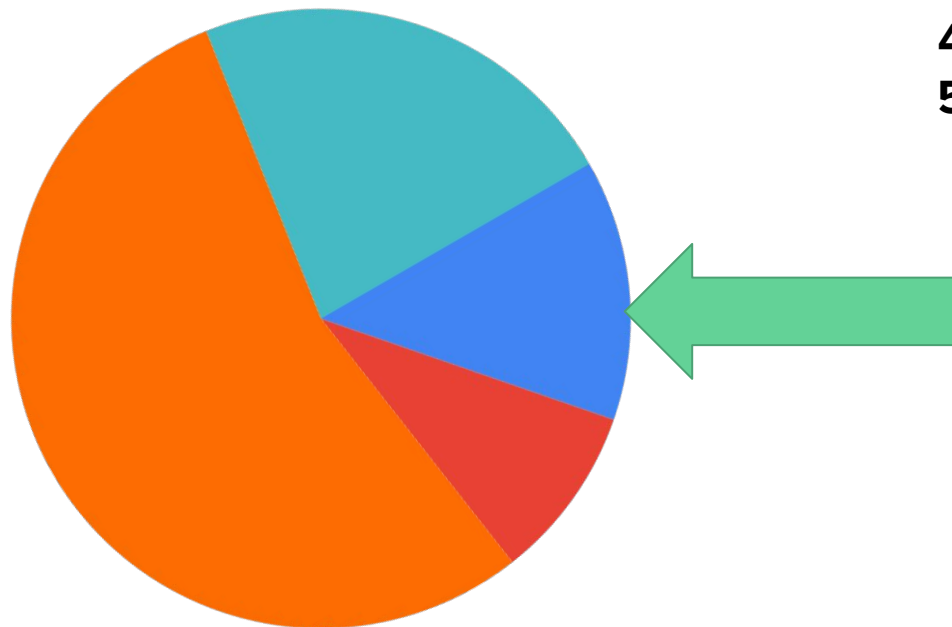
Exemplo:

## Problema da Mochila sem Repetições

- Seleção dos Melhores Indivíduos
  - Roleta:

Girar a roleta para  
selecionar 6  
indivíduos  
novamente

6º giro



1º giro: laranja  
2º giro: laranja  
3º giro: laranja  
4º giro: verde  
5º giro: verde  
6º giro: azul

Exemplo:

## Problema da Mochila sem Repetições

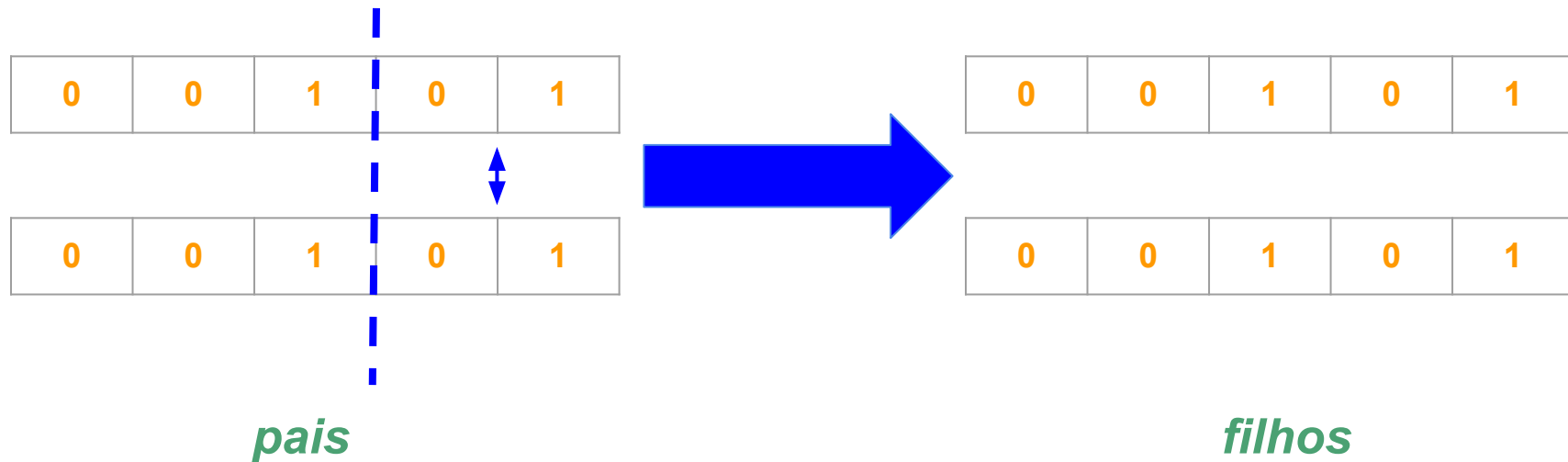
- Seleção dos Melhores Indivíduos

Verde	Cinza	Amarelo	Laranja	Azul
0	0	1	0	1
0	0	1	0	1
0	0	1	0	1
0	1	0	1	0
0	1	0	1	0
0	0	0	1	1

# Exemplo:

## Problema da Mochila sem Repetições

- Taxa de Crossover: **80%**

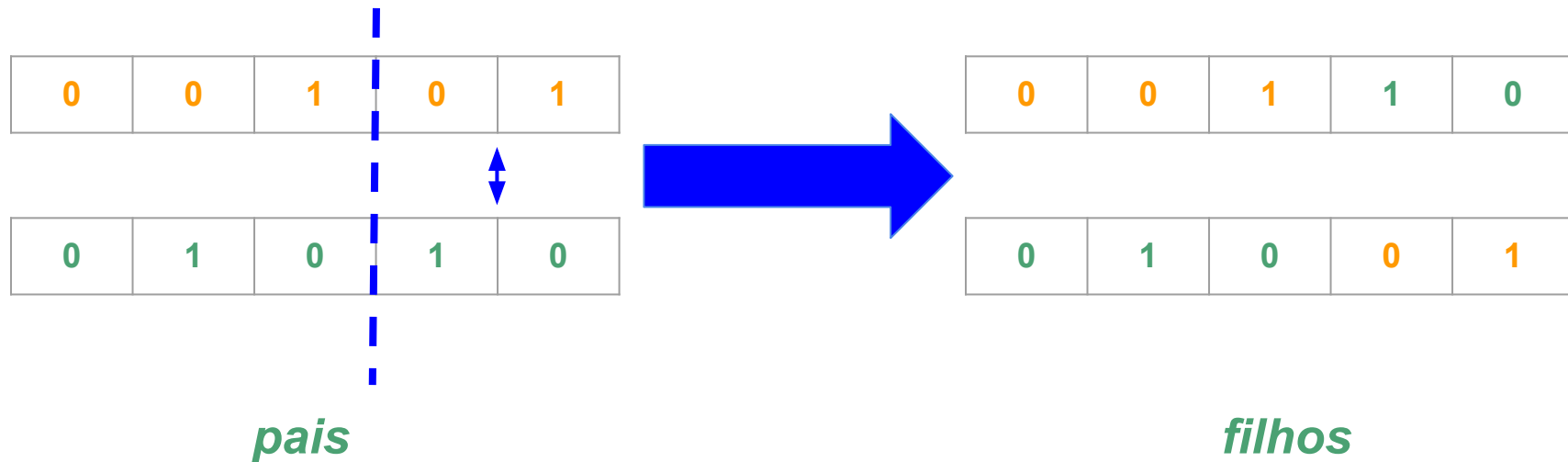




# Exemplo:

## Problema da Mochila sem Repetições

- Taxa de Crossover: **80%**



# Exemplo:

## Problema da Mochila sem Repetições

- Taxa de Crossover: **80%**

0	1	0	1	0
---	---	---	---	---

0	0	0	1	1
---	---	---	---	---



0	1	0	1	0
---	---	---	---	---

0	0	0	1	1
---	---	---	---	---

Taxa de 80%: esses dois  
não realizam crossover

*pais*

*pais*

# Exemplo:

## Problema da Mochila sem Repetições

- População depois do Crossover

Verde	Cinza	Amarelo	Laranja	Azul
0	0	1	0	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	0	0	1	1

Exemplo:

## Problema da Mochila sem Repetições

- Taxa de Mutação: **20%**
  - Nesse caso:  $6 * 0.2 = 1.2 \sim 1$  indivíduo terá mutação

Verde	Cinza	Amarelo	Laranja	Azul
0	0	1	0	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	0	0	1	1

Exemplo:

## Problema da Mochila sem Repetições

- Taxa de Mutação: **20%**
  - Nesse caso:  $6 * 0.2 = 1.2 \sim 1$  indivíduo terá mutação

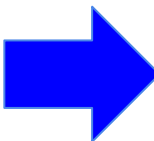


Verde	Cinza	Amarelo	Laranja	Azul
0	0	1	0	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	0	0	1	1

Exemplo:

## Problema da Mochila sem Repetições

- Taxa de Mutação: **20%**
  - Nesse caso:  $6 * 0.2 = 1.2 \sim 1$  indivíduo terá mutação



Verde	Cinza	Amarelo	Laranja	Azul
0	0	1	0	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	1	1	0
0	0	0	1	1

Exemplo:

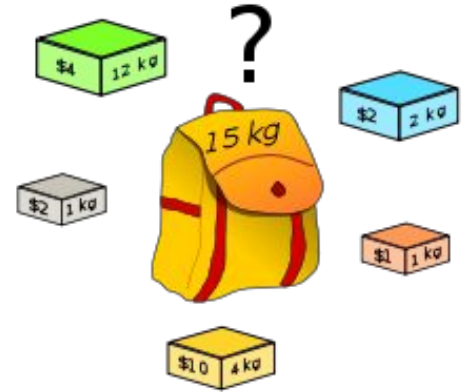
## Problema da Mochila sem Repetições

- Quantas gerações: **10 (por exemplo)**
  - O ciclo recomeça com a nova população, a partir da avaliação (função de aptidão)
  - Então, faz 10 vezes o ciclo completo

# Exemplo:

## Problema da Mochila sem Repetições (*Knapsack problem*)

- O Python tem alguns pacotes para Algoritmos Genéticos
- Vamos usar o **pyeasyga**





# Exercício:

## Problema da Mochila com Repetições (*Knapsack problem*)

- Problema de otimização combinatória
- Você pode escolher um número infinito de qualquer uma das 5 caixas
- Quais caixas devem ser escolhidas para maximizar a quantidade de dinheiro, mantendo o peso total abaixo ou igual a 15 kg?

