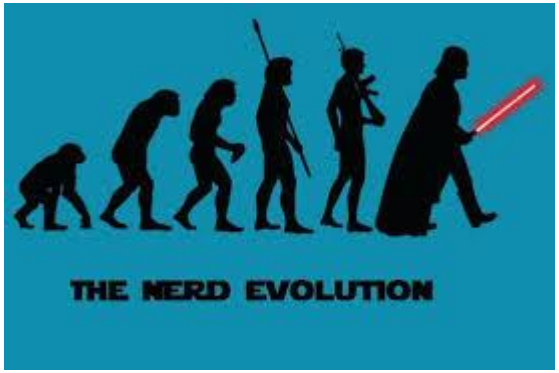



# Inteligência Artificial
# Ciência da Computação

**Prof. Aline Paes / alinepaes@ic.uff.br**

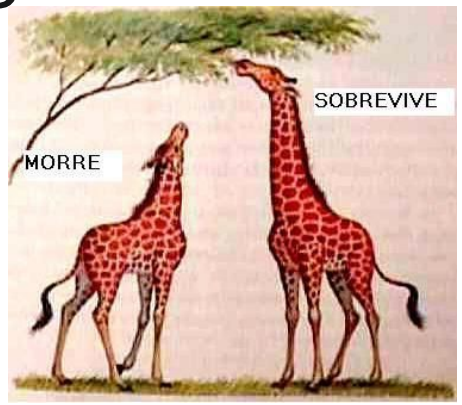
## Algoritmos Genéticos
## RN 4.1

Universidade Federal Fluminense

Instituto de Computação

# Algoritmos genéticos

- Vê otimização como uma analogia à teoria da evolução
  - simulação de seleção natural
  - estados são indivíduos em uma população
  - pontuação é uma função de fitness
  - indivíduos com menor fitness são deixados de fora para morrer
  - indivíduos com melhor fitness vão reproduzir



# Problemas resolvidos por GAs

- Roteamento de veículos
- Bioinformática
- Code breaking
- Problemas de escalonamento
- Comportamento de robôs
- Projeto de carros
- Joke generations e criatividade computacional
- Estratégias de investimento
- the list goes on.....

# Algoritmos genéticos - implementação

- indivíduos são strings


- Analogia:
  - string é o cromossomo que representa o indivíduo
  - strings são feitas de genes
  - genes são passados para os filhos
  - configurações de genes que contribuem para um fitness alto tendem a sobreviver nas gerações

# Algoritmos genéticos - indivíduos

- Exemplo: SAT

$$(P_1 \wedge P_2) \vee (P_1 \wedge \neg P_3) \vee (P_1 \wedge \neg P_4) \vee (\neg P_3 \wedge \neg P_4)$$

- Indivíduo

| P1 | P2 | P3 | P4 |
|---|---|---|---|
| 1 | 0 | 1 | 1 |

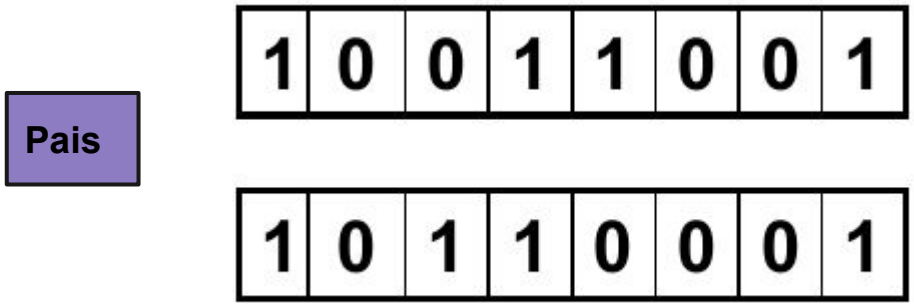
# Algoritmos genéticos - indivíduos

- Indivíduos são soluções para o problema (completas ou parciais)
- Explora o espaço de soluções fazendo os indivíduos
  - interagirem
    - novos indivíduos são produzidos
  - competirem
    - indivíduos mais fracos são eliminados
- Depois de várias gerações, um indivíduo forte é encontrado (i.e., uma solução)

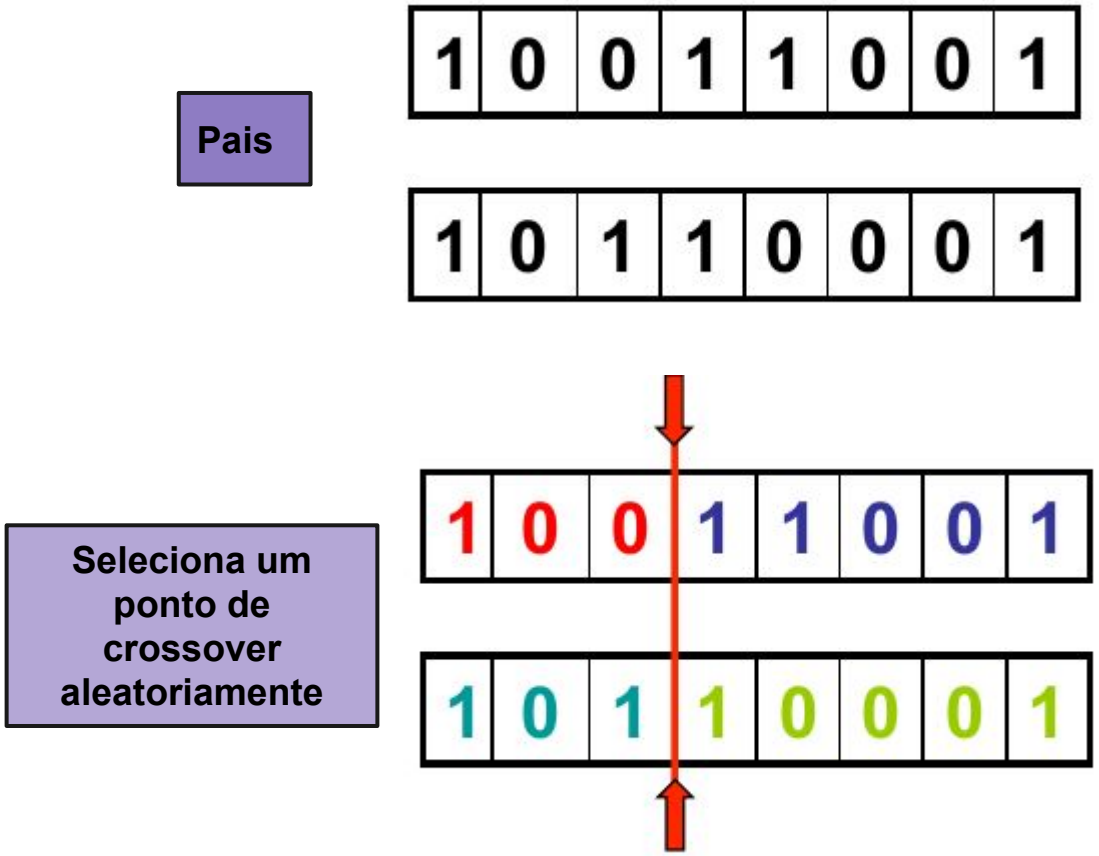
# Algoritmos genéticos

- Inicia com uma população aleatória de *P* indivíduos e executa neles as operações
  - reprodução (crossover)
  - mutação

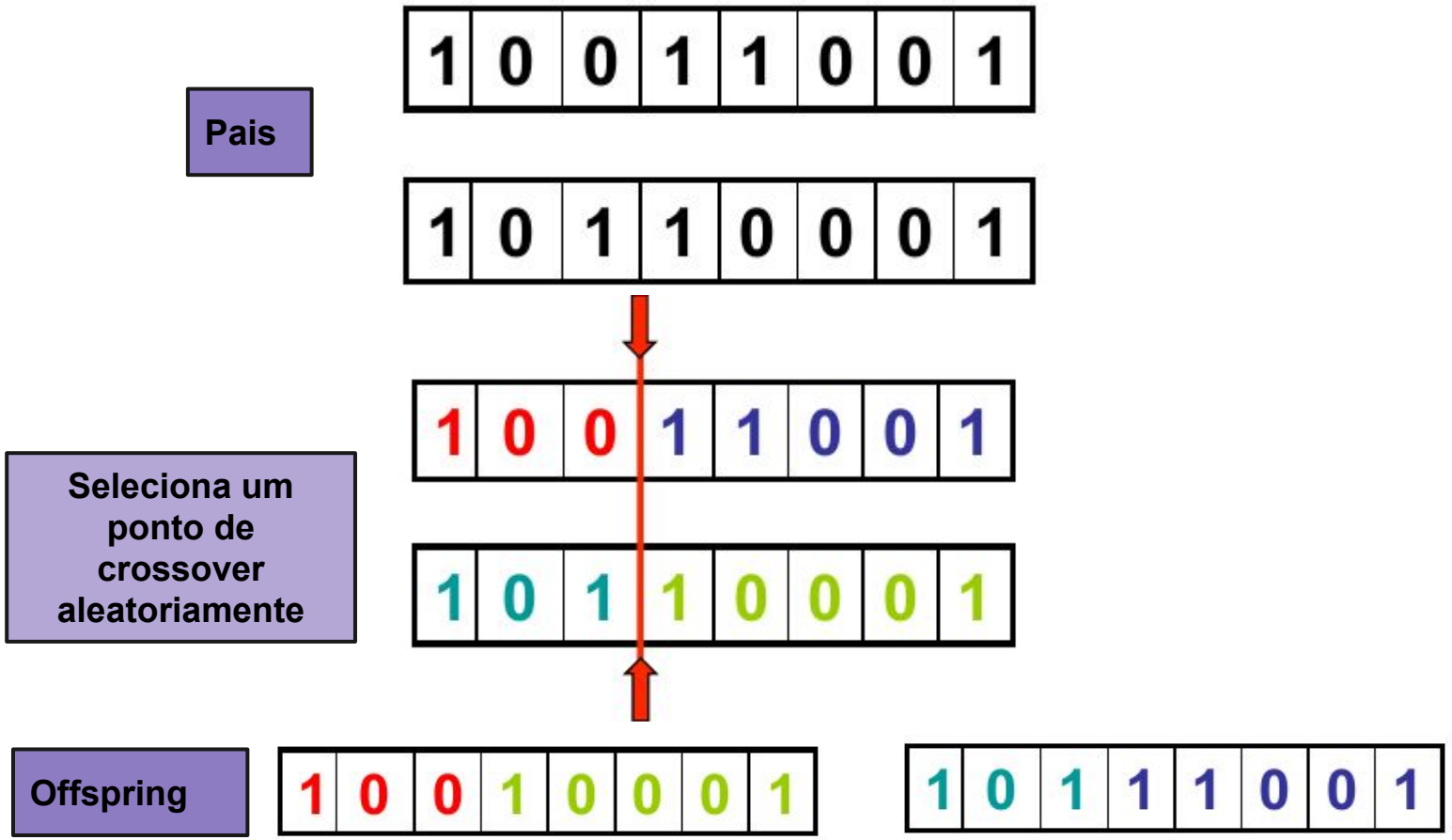
# Algoritmos genéticos - reprodução



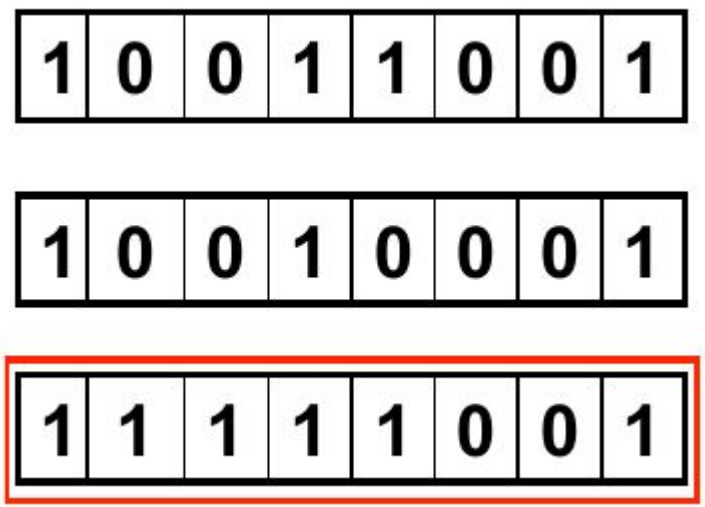
# Algoritmos genéticos - reprodução



# Algoritmos genéticos - reprodução

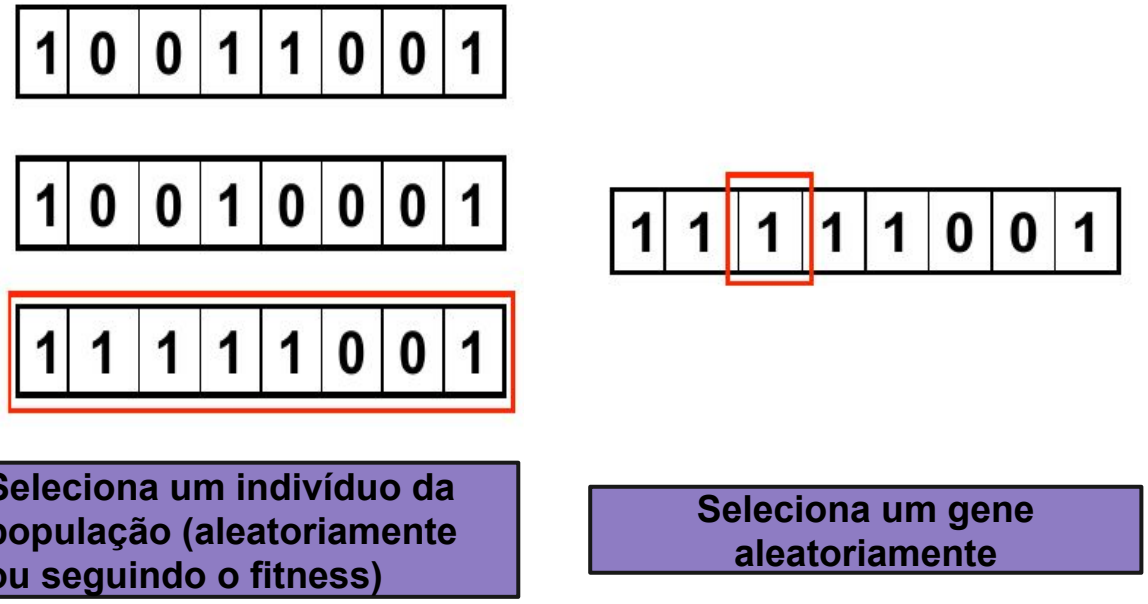


# Algoritmos genéticos: mutação

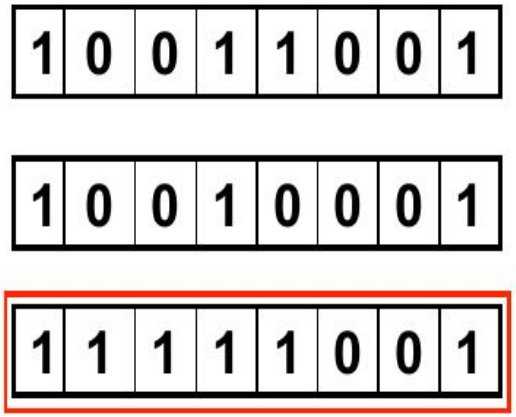



**Seleciona um indivíduo da população (aleatoriamente ou seguindo o fitness)**

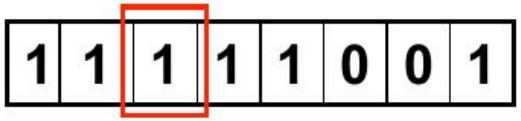
# Algoritmos genéticos: mutação



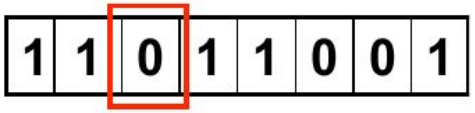
# Algoritmos genéticos: mutação




Seleciona um indivíduo da população (aleatoriamente ou seguindo o fitness)




Seleciona um gene aleatoriamente

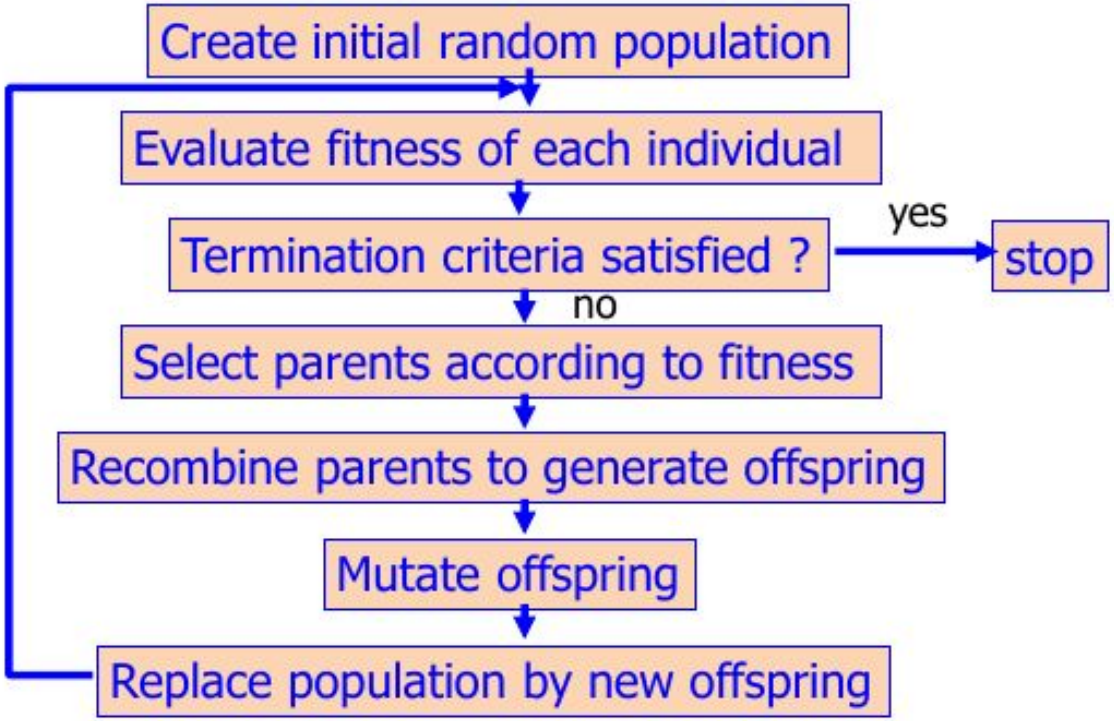



Muda o gene escolhido

# Avaliação

- Medida de fitness corresponde à qualidade das soluções individuais
  - problemas de classificação: $(\text{correct}(i))^2$
  - TSP:  distance(i)
  - SAT: #termosSatisfeitos(i)

# Algoritmo genético



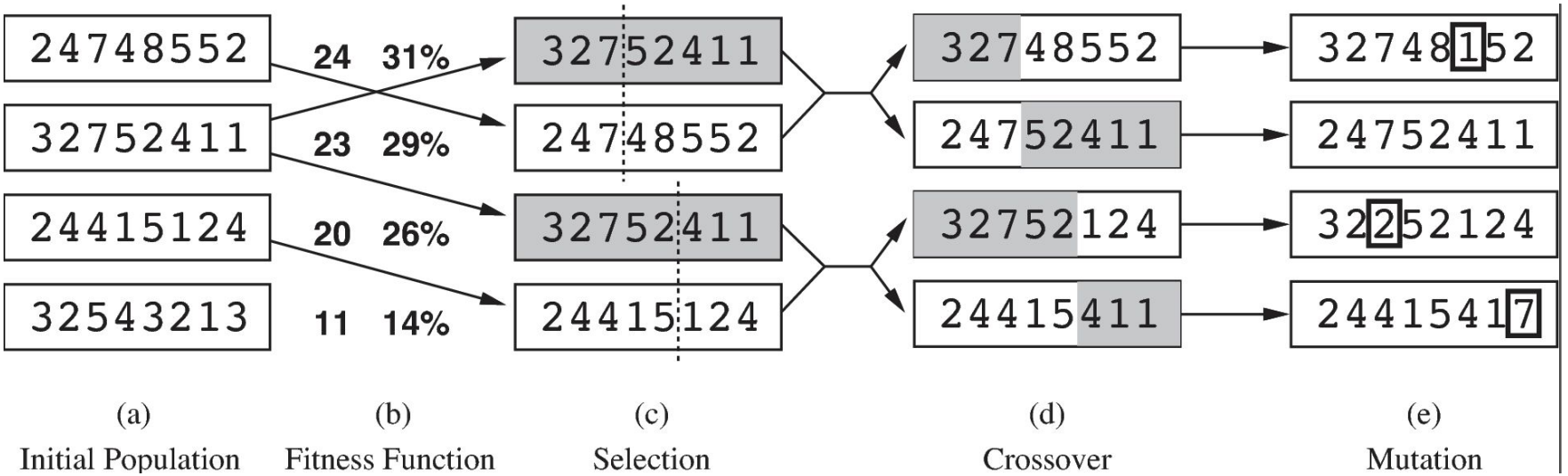
# Algoritmo genético - versão básica

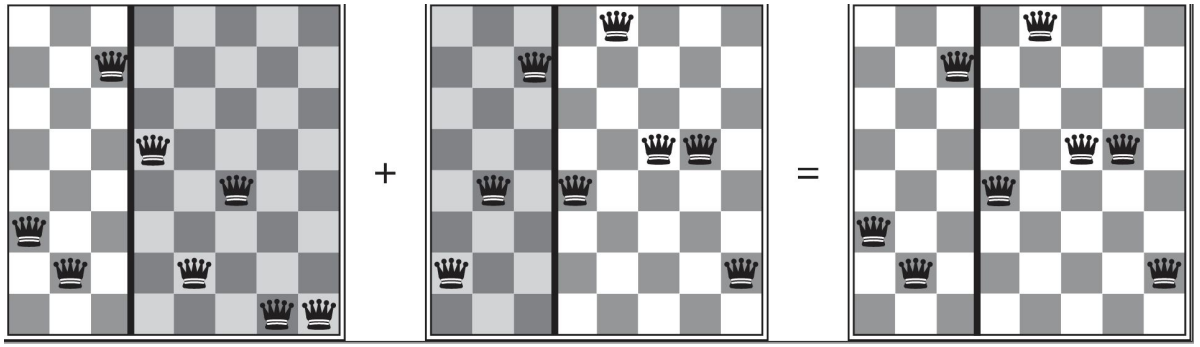
```
function Genetic-Algorithm(population, fitness-fn) returns um indivíduo

  repeat
      new_population = conjunto vazio
      para i de 1 até tamanho(população) faça
          x = selecione um indivíduo na população de acordo com fitness-fn (um pai)
          y = selecione um indivíduo na população de acordo com fitness-fn (outro pai)
          child = reproduce(x,y)
          com uma probabilidade p
              child = mutate(child)
          adicione child em new_population
      population = new_population
  until algum indivíduo tenha um valor de fitness suficiente ou passou tempo suficiente
return melhor indivíduo em population

function reproduce(x,y) returns um indivíduo
      n = tamanho(x)
      c = número aleatório entre 1 e n
      return append(substring(x,1,c), substring(y, c+1, n))
```

# Exemplo

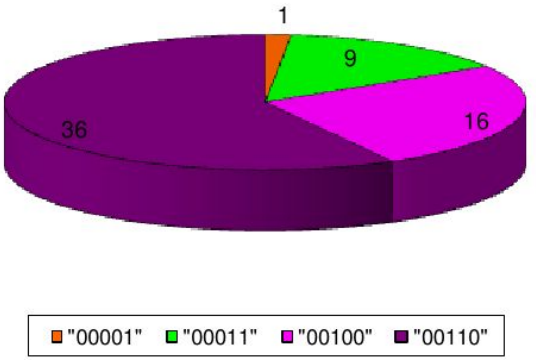

24748552
32752411
24415124
32543213
24 31%
23 29%
20 26%
11 14%
32752411
24748552
32752411
24415124
32748552
24752411
32752124
24415411
32748152
24752411
32252124
24415417
(a)
Initial Population
(b)
Fitness Function
(c)
Selection
(d)
Crossover
(e)
Mutation


+
=

# Seleção

- Método mais comum: roleta
- Cada indivíduo recebe um pedaço na roleta proporcional ao seu fitness

| Indivíduo | Avaliação | Pedaço da roleta (%) |
|---|---|---|
| 00001 | 1 | 1.61 |
| 00011 | 9 | 14.51 |
| 00100 | 16 | 25.81 |
| 00110 | 36 | 58.07 |
| **Total** | 62 | 100.00 |



# População

- Deve ser diversa o suficiente
  - ou pode resultar em convergência prematura
- Como gerar uma população diversa o suficiente?
  - uniformemente aleatório - distribuição uniforme
  - grid  - indivíduos em intervalos regulares
  - non-clustering - indivíduos distanciados
  - otimização local - técnica de busca local para encontrar população em um ótimo local

# Seleção

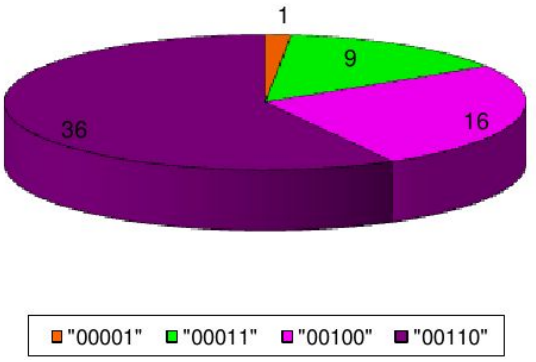
- Seleção depende da função da avaliação (fitness)
- Escolher que indivíduos sobrevivem e reproduzem
  - influencia na escolha de uma solução ótima
  - e também na convergência
- Natureza não elimina todos os genes "unfit"
  - podem se tornar recessivos por um longo período e então mutar para algo mais útil

# Seleção

- Método mais comum: roleta
- Cada indivíduo recebe um pedaço na roleta proporcional ao seu fitness

| Indivíduo | Avaliação | Pedaço da roleta (%) |
|---|---|---|
| 00001 | 1 | 1.61 |
| 00011 | 9 | 14.51 |
| 00100 | 16 | 25.81 |
| 00110 | 36 | 58.07 |
| **Total** | 62 | 100.00 |

Gera-se um número aleatório: 27
Indivíduo selecionado: 00100



# Aplicação dos operadores

- É possível aumentar ou diminuir a incidência de cada
  - mais controle sobre o desenvolvimento dos novos indivíduos
- A porcentagem de aplicação de cada operador não precisa ser fixa
  - Inicialmente: muita diversidade genética
    - Muita reprodução e pouca mutação
  - Depois de um grande número de gerações: pouca diversidade genética
    - Mais mutação e menos reprodução

# Elitismo

- Os *n* melhores indivíduos de cada geração não devem "morrer" junto com a sua geração, mas sim passar para a próxima
- É uma forma de garantir que o algoritmo nunca regride.

# Exercício

Considere a seguinte fórmula em lógica proposicional:

$$(\neg x \vee \neg z \vee y) \wedge (\neg y \vee z) \wedge (x \vee \neg z) \wedge (y) \wedge (\neg y \vee \neg x \vee \neg w \vee z)$$

Mostre pelo menos 4 iterações de um algoritmo genético para resolver um problema SAT, tendo a fórmula acima como base. Para tanto, você deve decidir como codificar os cromossomos e definir uma função de fitness para o problema. Sua população inicial deverá ter 4 indivíduos.

# Exercício

Use um algoritmo genético para resolver o problema da mochila. Assuma que você tem uma mochila de capacidade C e um conjunto N de objetos na sua frente. Cada objeto i em N tem tamanho $s_i$ e valor $v_i$. Uma solução é um conjunto $M \subseteq N$ tal que $sum(s_i) \leq C$, para $i \in M$ e que $sum(v_i)$, $i \in M$ é maximizado. Crie uma população inicial de 4 indivíduos e execute ao menos 5 gerações do algoritmo genético.

# Exercício

Use um algoritmo genético para coletar os 6 itens com o menor custo total possível em um mapa representado pelo grafo abaixo. Decida como os indivíduos serão codificados e gere uma população inicial de 5 indivíduos. Execute pelo menos 5 iterações do algoritmo genético básico.