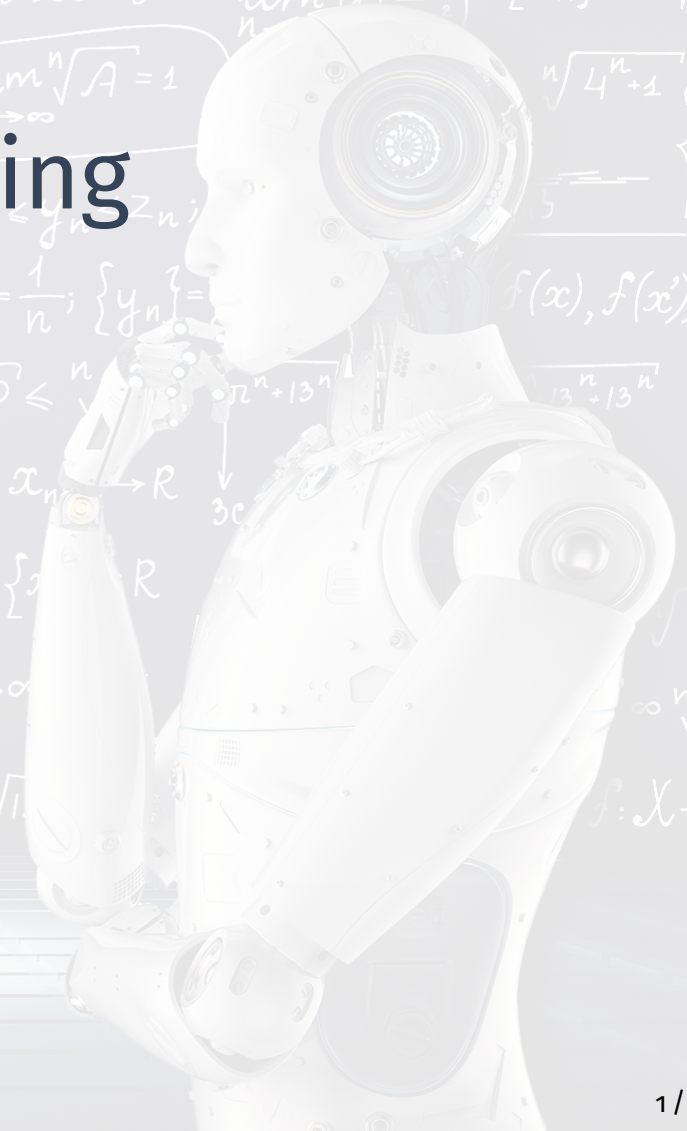
Machine Learning

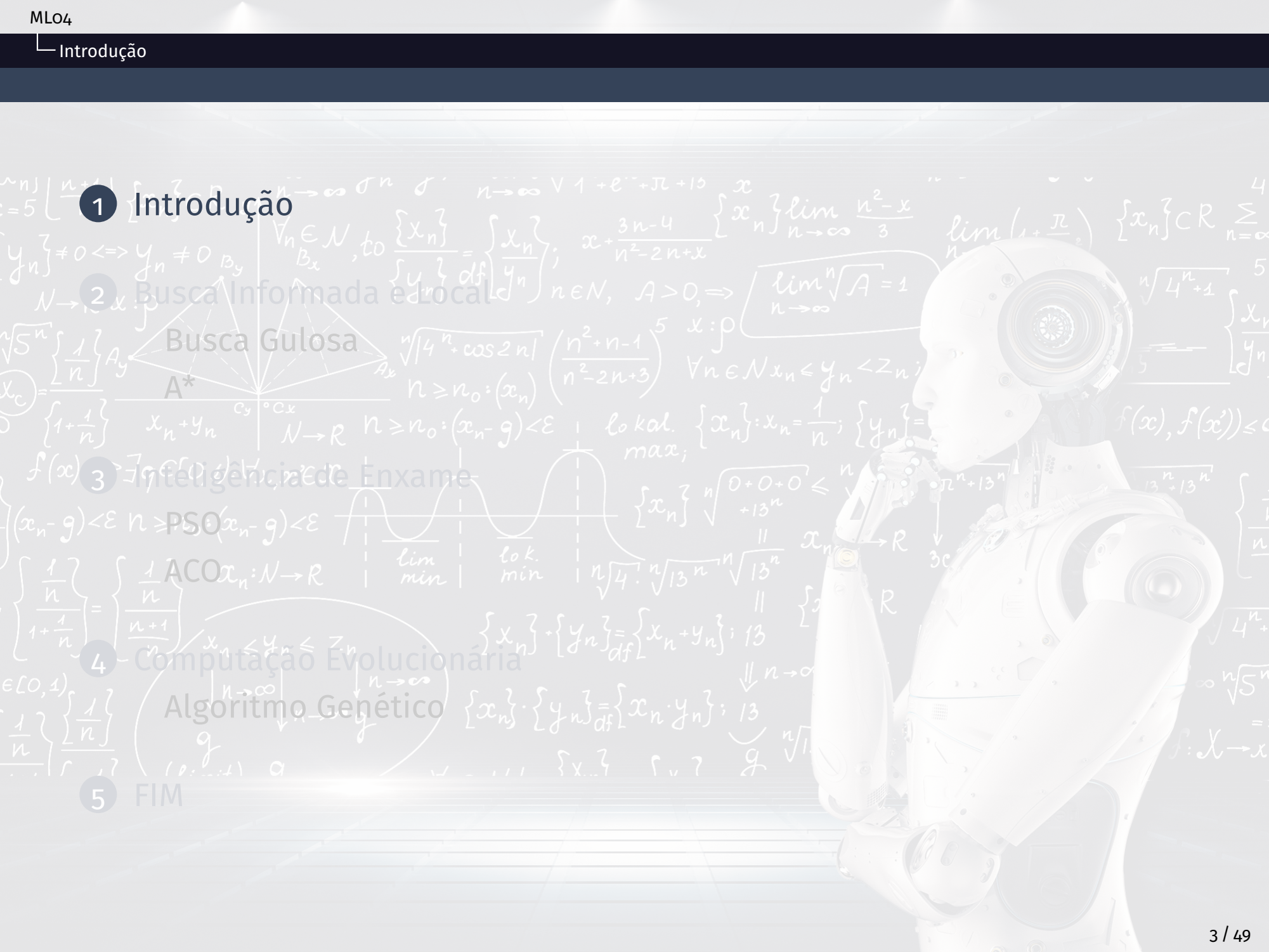
Aula 04

Evandro J.R. Silva

Uninassau Teresina

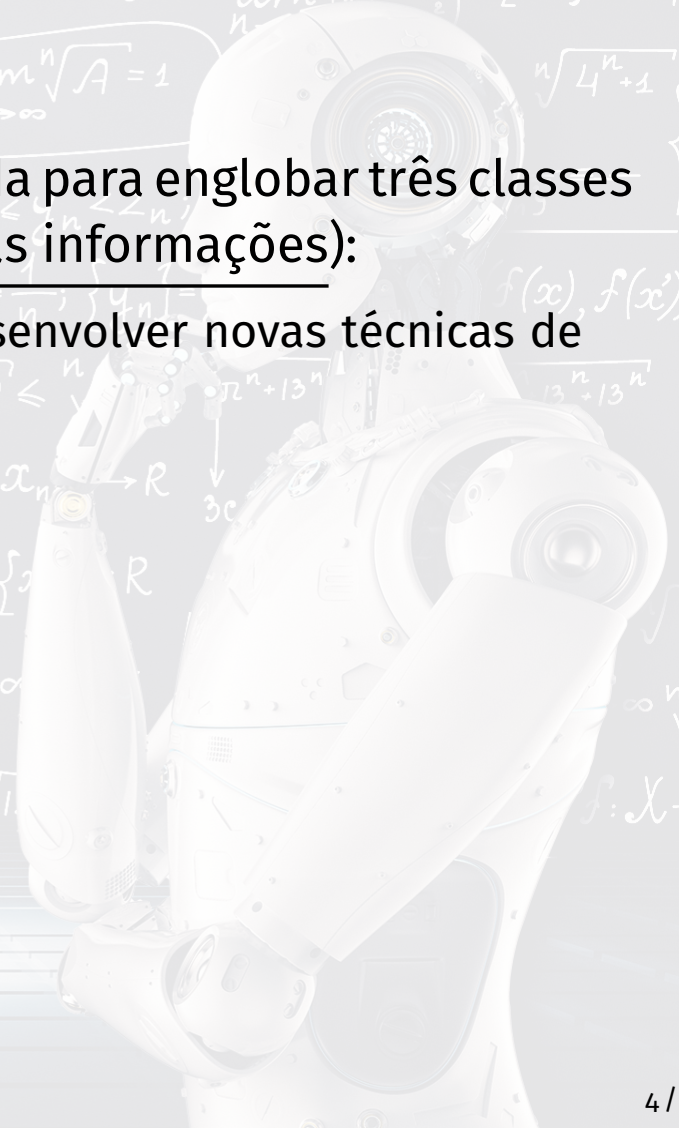


Sumário



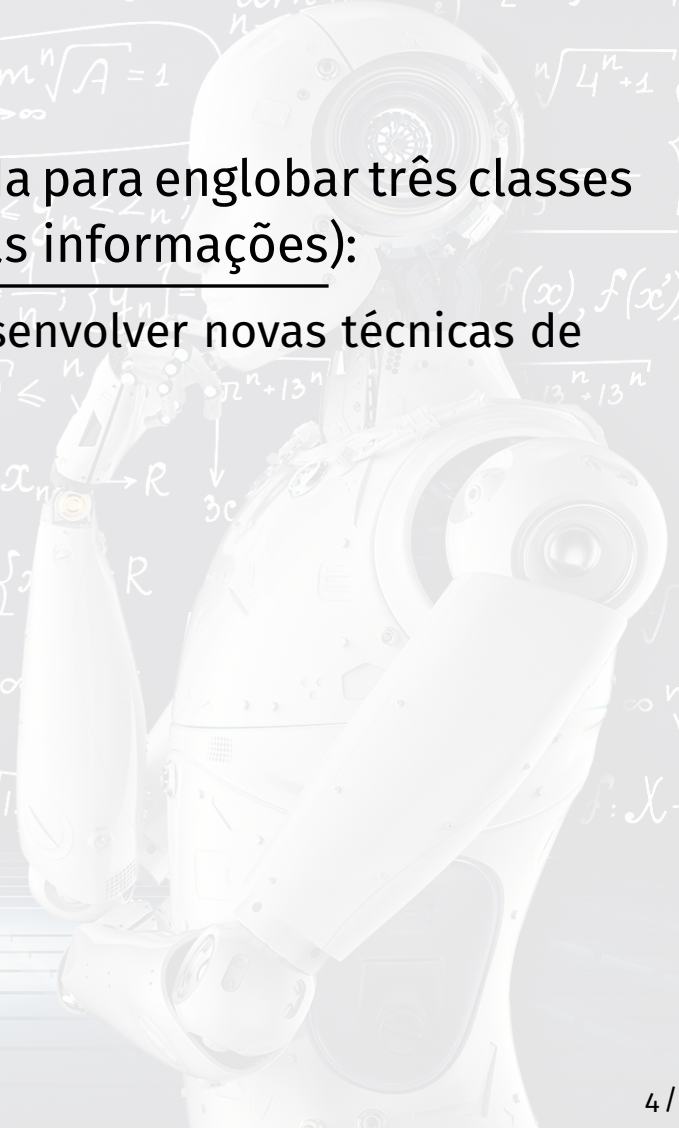
Introdução

- **Computação Natural**: é a terminologia criada para englobar três classes de métodos (artigo da Wikipedia com várias informações):
  1. Os que se inspiram na natureza para desenvolver novas técnicas de resolução de problemas;



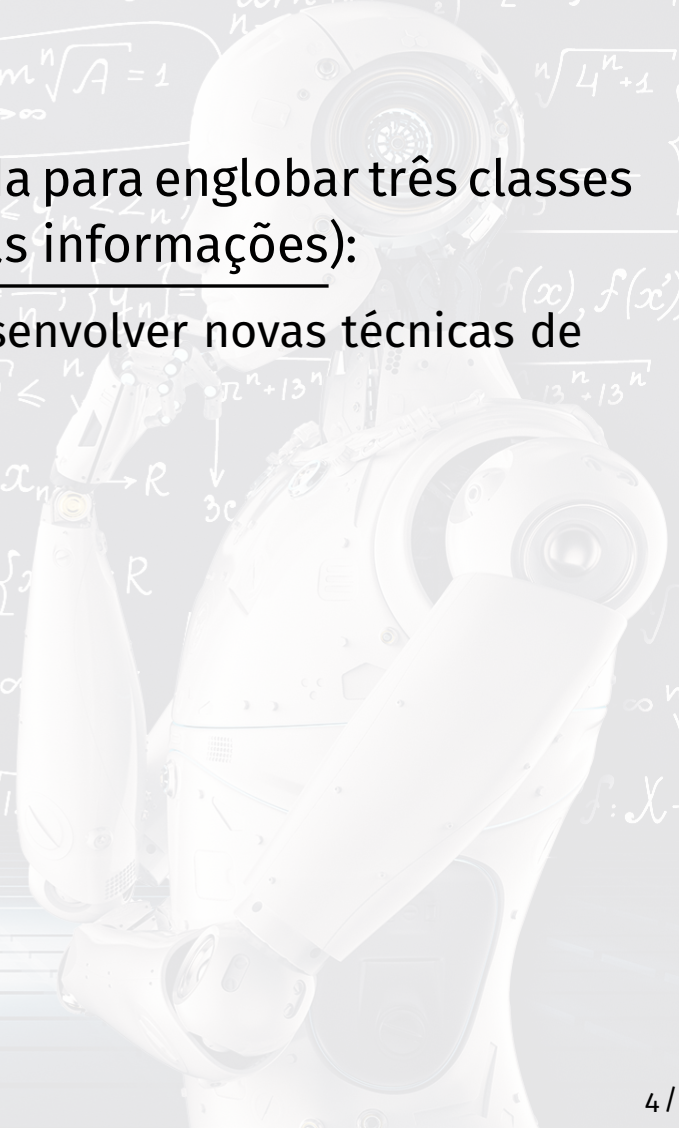
Introdução

- **Computação Natural**: é a terminologia criada para englobar três classes de métodos (artigo da Wikipedia com várias informações):
  1. Os que se inspiram na natureza para desenvolver novas técnicas de resolução de problemas;
  2. Os que sintetizam fenômenos naturais;



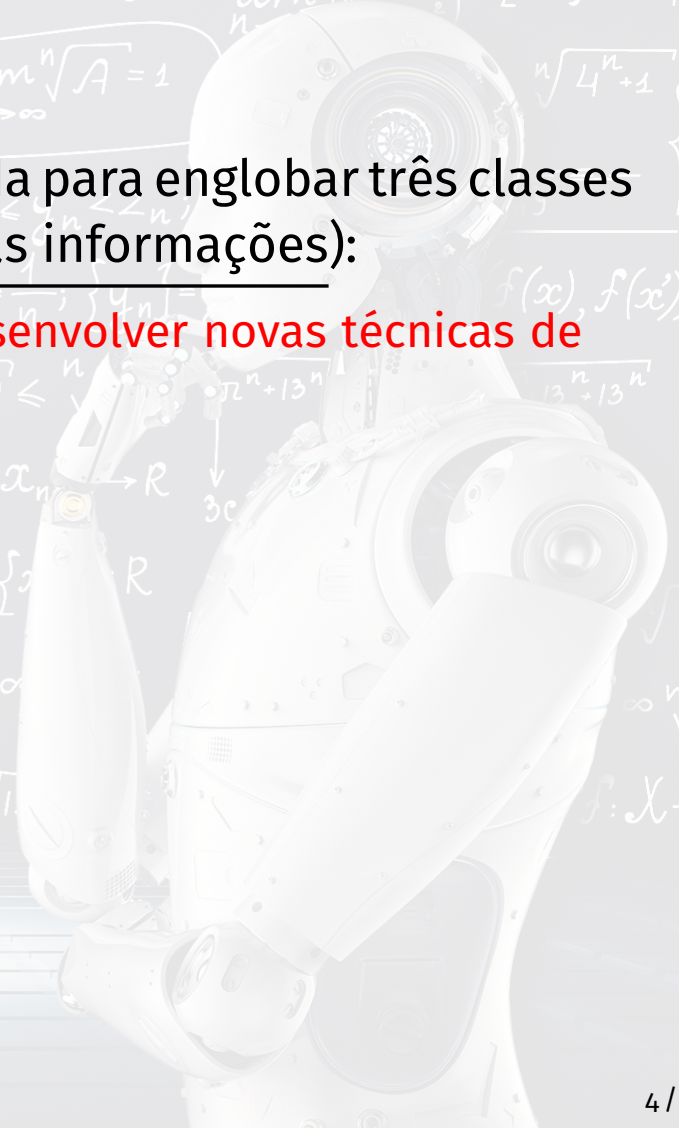
Introdução

- **Computação Natural**: é a terminologia criada para englobar três classes de métodos (artigo da Wikipedia com várias informações):
  1. Os que se inspiram na natureza para desenvolver novas técnicas de resolução de problemas;
  2. Os que sintetizam fenômenos naturais;
  3. Os que utilizam materiais naturais.



Introdução

- **Computação Natural**: é a terminologia criada para englobar três classes de métodos (artigo da Wikipedia com várias informações):
  1. Os que se inspiram na natureza para desenvolver novas técnicas de resolução de problemas;
  2. Os que sintetizam fenômenos naturais;
  3. Os que utilizam materiais naturais.
     - O foco da nossa aula!



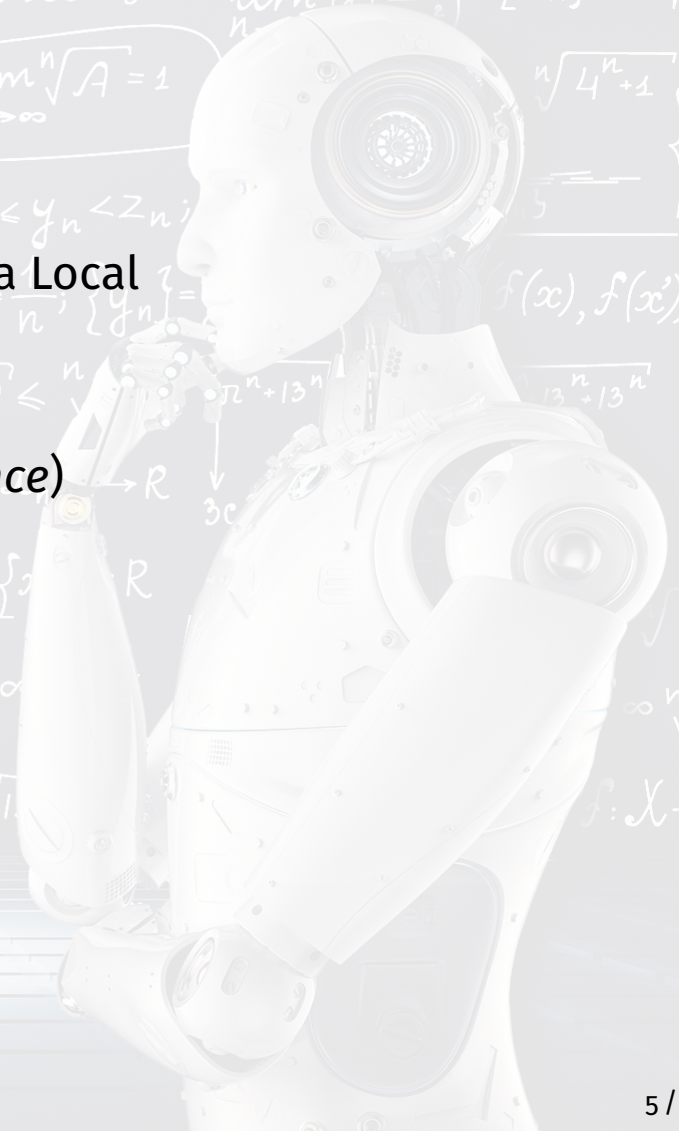
Introdução

- **Computação Natural**: é a terminologia criada para englobar três classes de métodos (artigo da Wikipedia com várias informações):
  1. Os que se inspiram na natureza para desenvolver novas técnicas de resolução de problemas;
  2. Os que sintetizam fenômenos naturais;
  3. Os que utilizam materiais naturais.

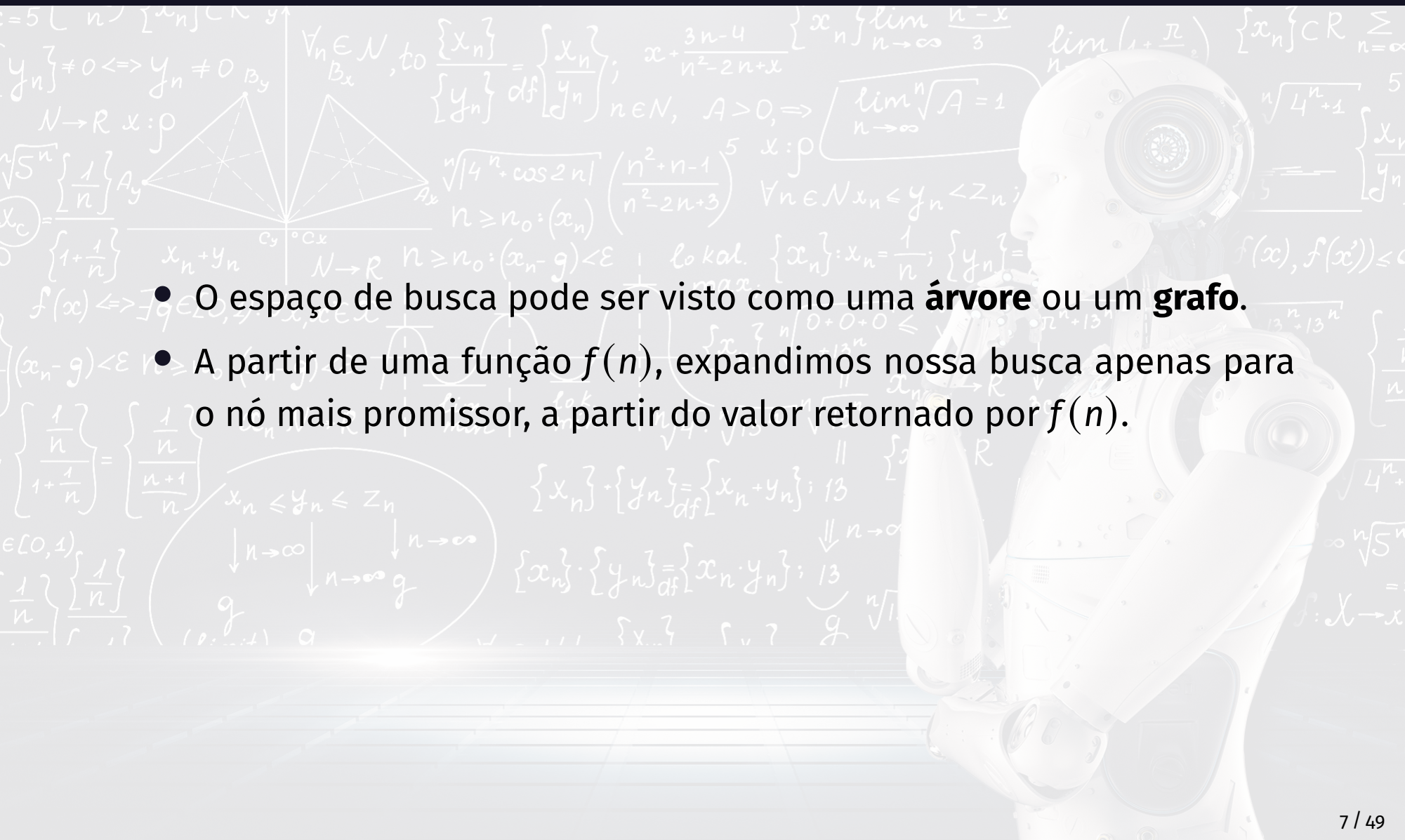
    - Exemplos e artigos de variadas formas de computação bioinspirada para vocês darem uma olhada — não dá para falar um pouco sobre cada um porque é muita coisa!

Introdução

- Algoritmos que vamos ver hoje
  - Busca Informada (com heurística) e Busca Local
    - Busca Gulosa (*greedy search*)
    - A*
  - Inteligência de Enxame (*Swarm Intelligence*)
    - PSO
    - ACO
  - Computação Evolucionária
    - Algoritmo Genético



Busca Informada e Local



- O espaço de busca pode ser visto como uma **árvore** ou um **grafo**.
- A partir de uma função $f(n)$, expandimos nossa busca apenas para o nó mais promissor, a partir do valor retornado por $f(n)$.

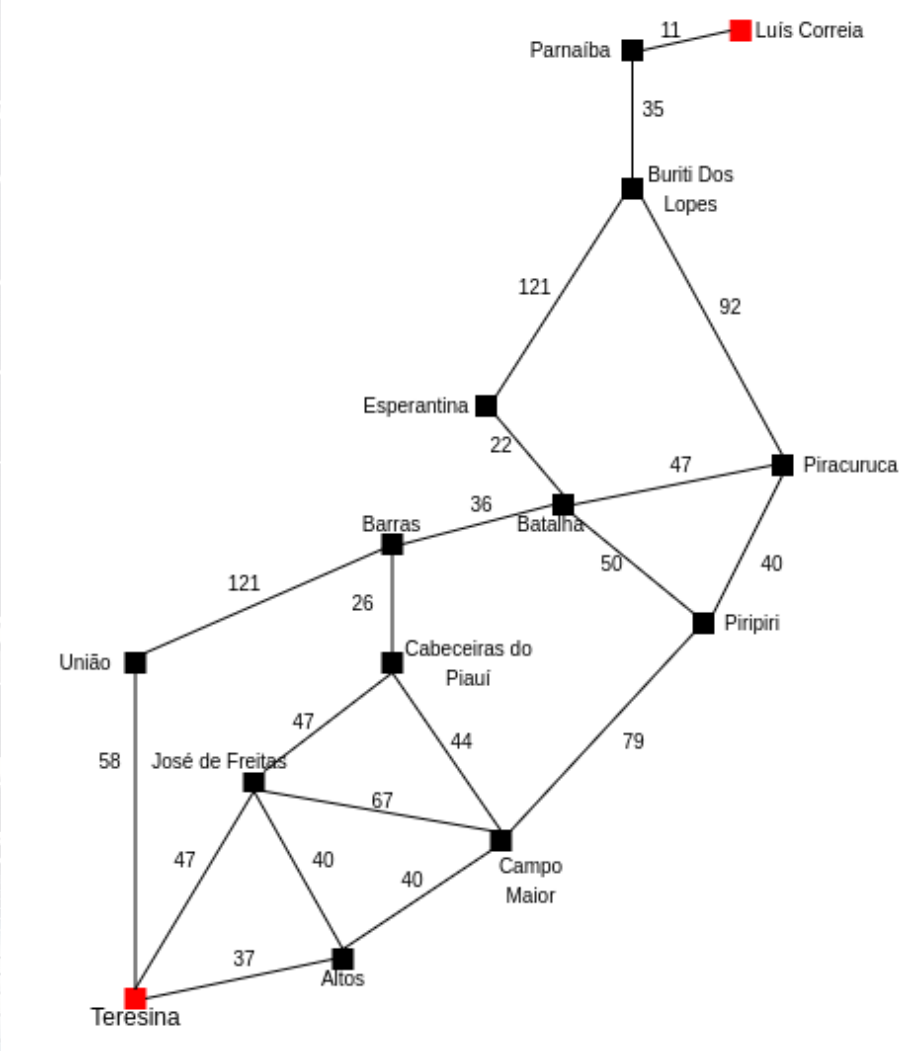
Busca Informada e Local

- Problema: encontrar o menor caminho entre Teresina e Luís Correia.
- Temos a seguinte informação: distância em linha reta de cada cidade para Luís Correia:

Cidade	**Distância**	**Cidade**	**Distância**
Teresina	277	Esperantina	129
Altos	255	Batalha	134
União	231	Piripiri	155
José de Freitas	231	Piracuruca	117
Barras	167	Buriti dos Lopes	40
Cabeceiras do Piauí	191	Parnaíba	10
Campo Maior	223	Luís Correia	0

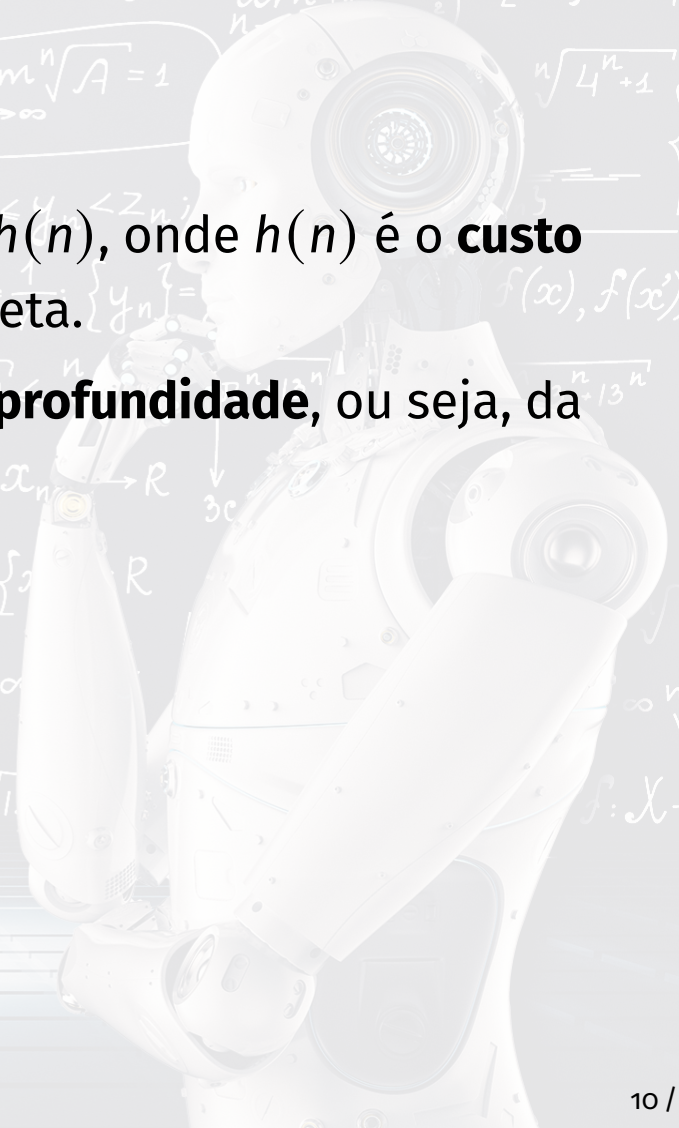
Busca Informada e Local

O mapa



Busca Gulosa

- Este algoritmo é caracterizado por $f(n) = h(n)$, onde $h(n)$ é o **custo estimado** do melhor caminho de n até a meta.
- Seu comportamento é o de uma **busca em profundidade**, ou seja, da raiz até as folhas.
- É um algoritmo **não ótimo** e **incompleto**



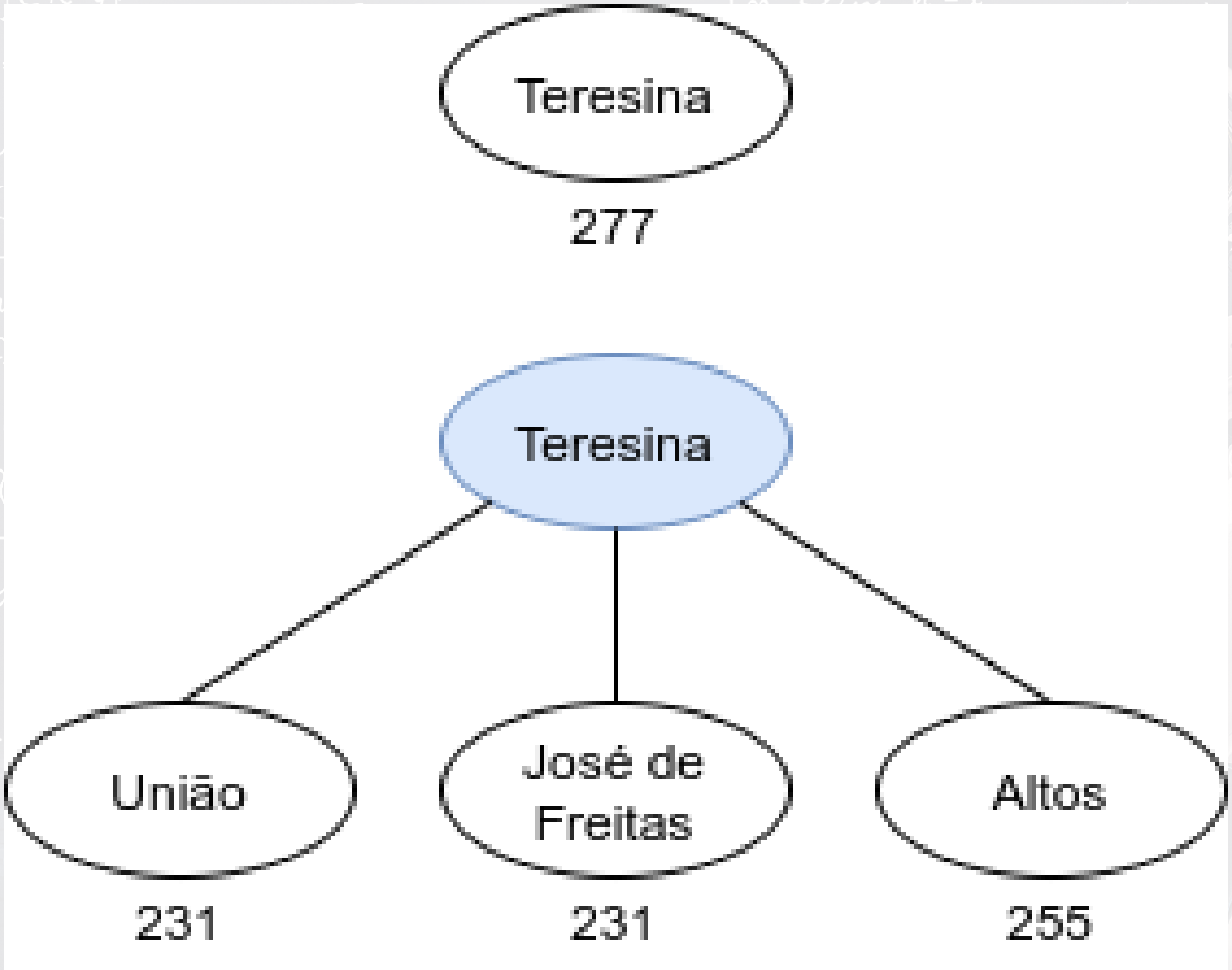
Busca Gulosa

- Este algoritmo é caracterizado por $f(n) = h(n)$, onde $h(n)$ é o **custo estimado** do melhor caminho de n até a meta.
- Seu comportamento é o de uma **busca em profundidade**, ou seja, da raiz até as folhas.
- É um algoritmo **não ótimo** e **incompleto**
  - Não garante encontrar a melhor solução entre todas.

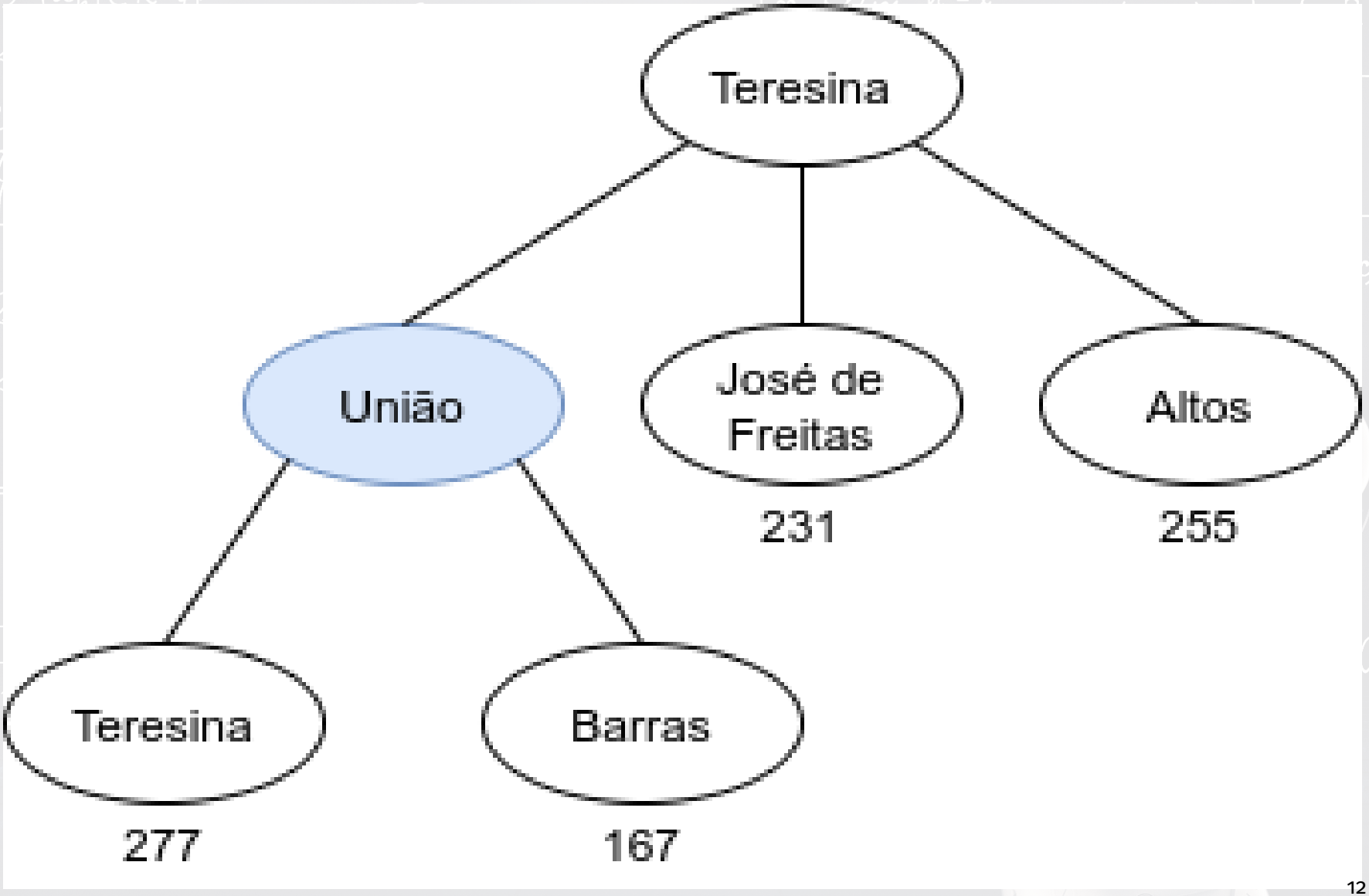
Busca Gulosa

- Este algoritmo é caracterizado por $f(n) = h(n)$, onde $h(n)$ é o **custo estimado** do melhor caminho de n até a meta.
- Seu comportamento é o de uma **busca em profundidade**, ou seja, da raiz até as folhas.
- É um algoritmo **não ótimo** e **incompleto**
  - Não garante encontrar uma solução (pode ficar preso em algum ciclo).

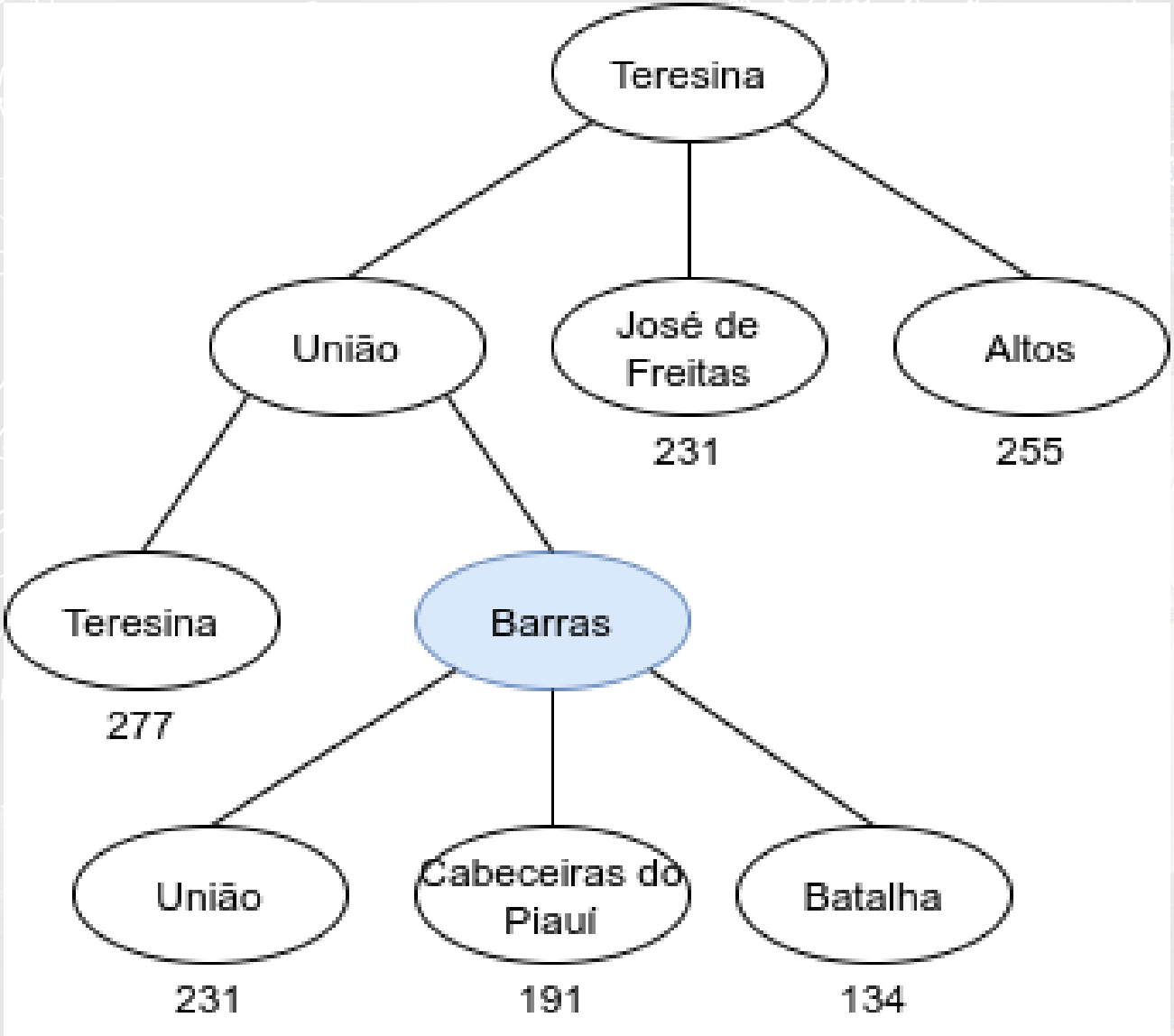
Busca Gulosa



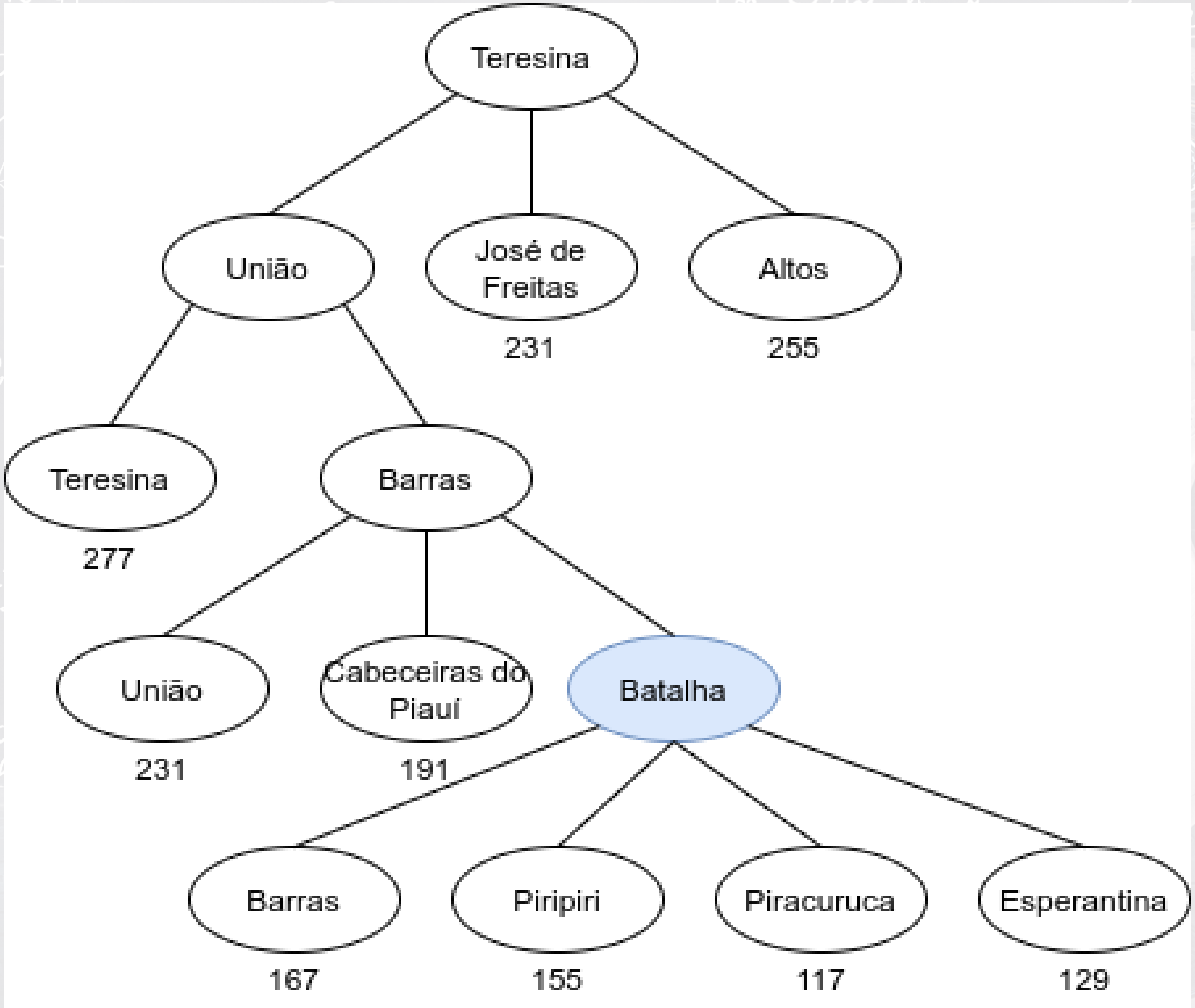
Busca Gulosa



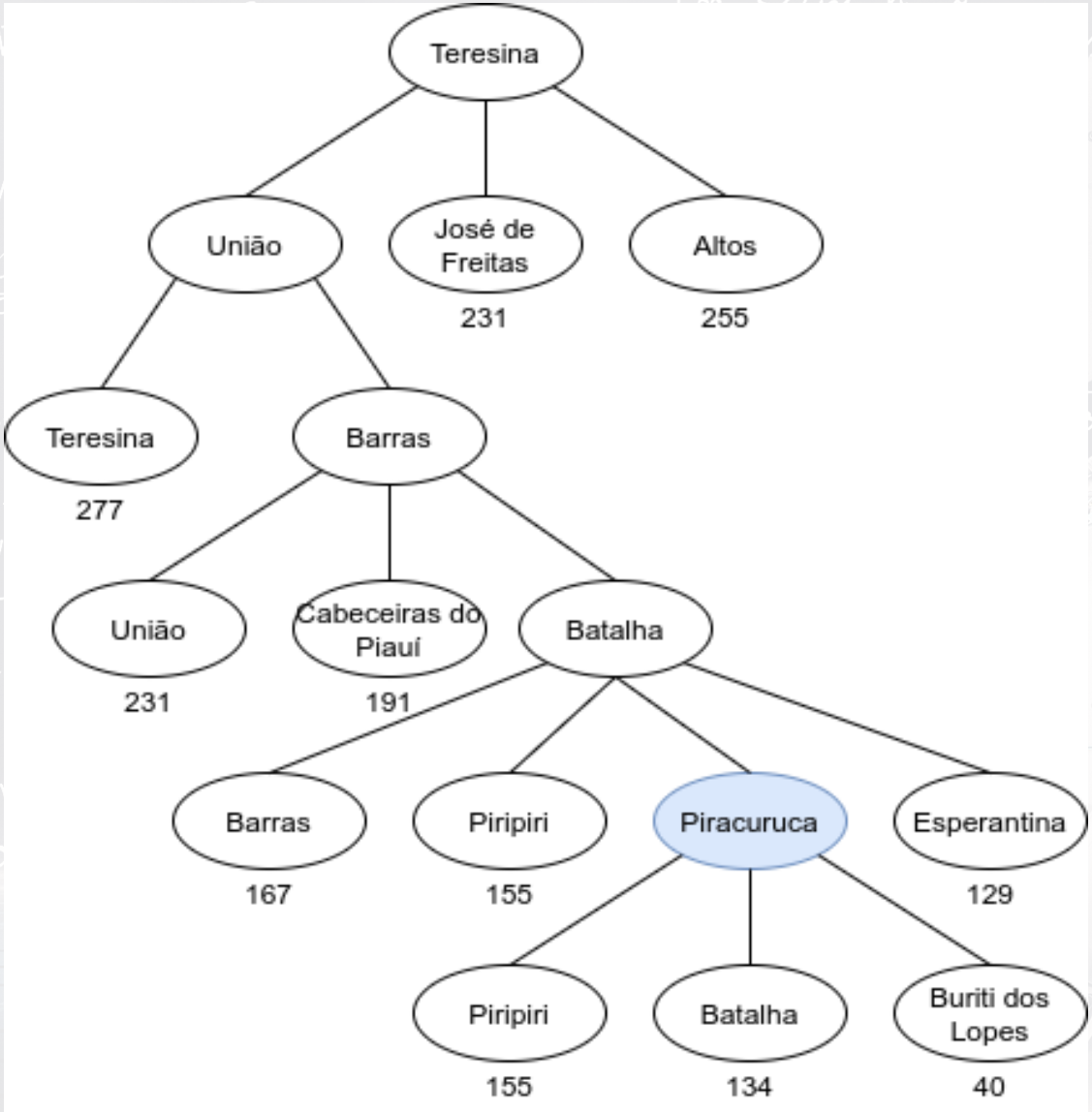
Busca Gulosa



Busca Gulosa

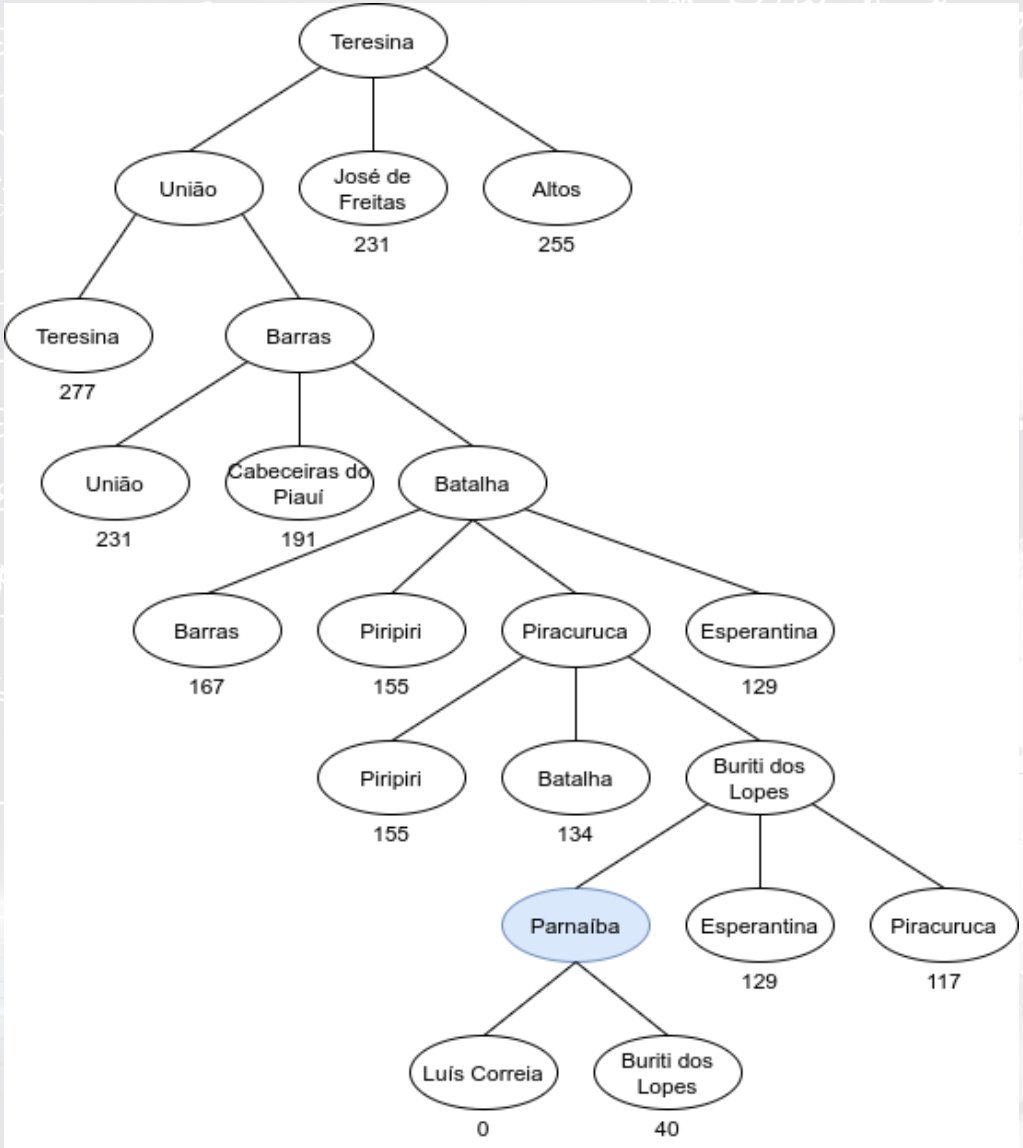


Busca Gulosa

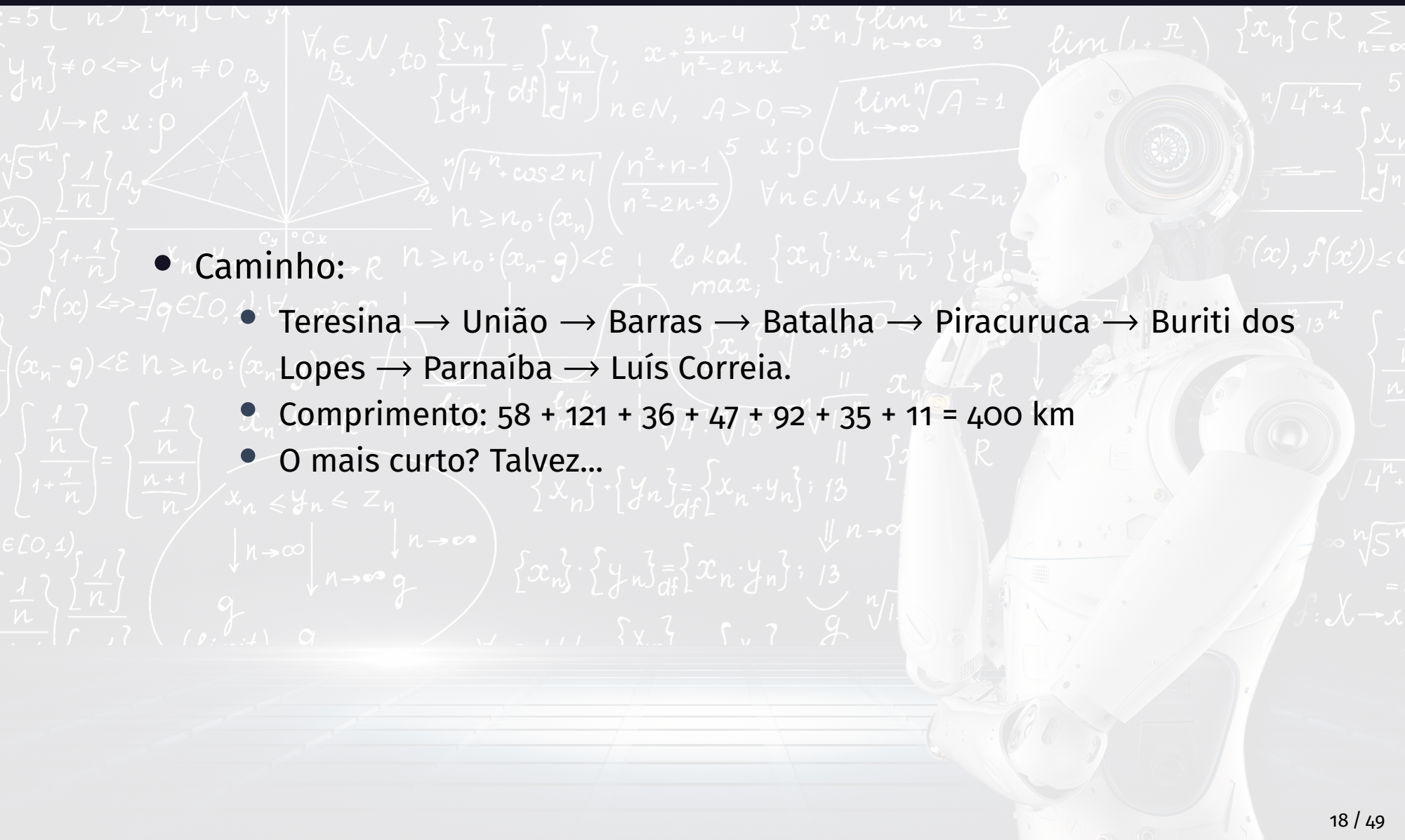


Busca Gulosa

Busca Gulosa



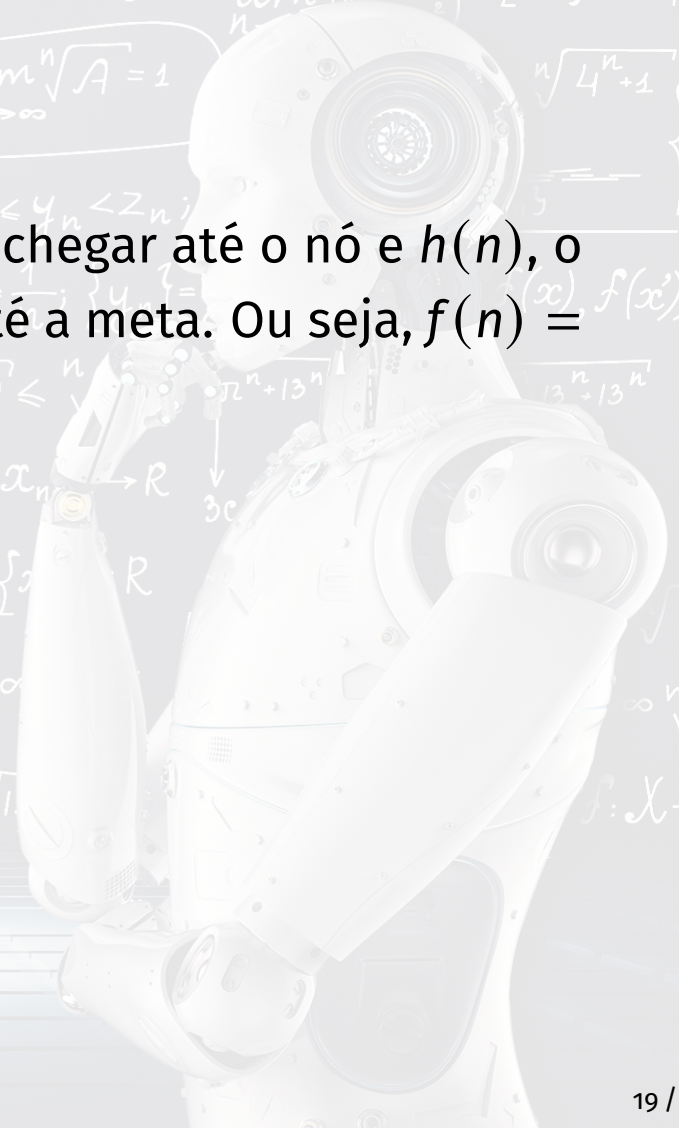
Busca Gulosa



- Caminho:
  - Teresina ⟶ União ⟶ Barras ⟶ Batalha ⟶ Piracuruca ⟶ Buriti dos Lopes ⟶ Parnaíba ⟶ Luís Correia.
  - Comprimento: 58 + 121 + 36 + 47 + 92 + 35 + 11 = 400 km
  - O mais curto? Talvez...

A*

- Combina duas funções: $g(n)$, o custo para chegar até o nó e $h(n)$, o custo estimado do melhor caminho de n até a meta. Ou seja, $f(n) = g(n) + h(n)$
- É um algoritmo ótimo e completo

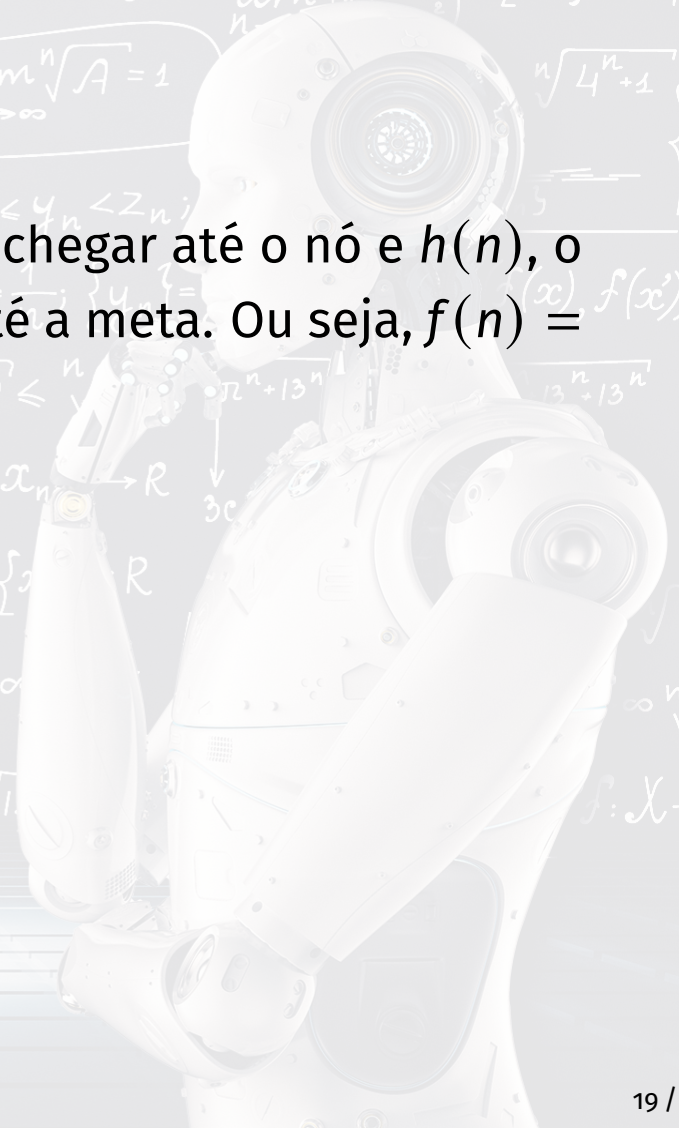


A*

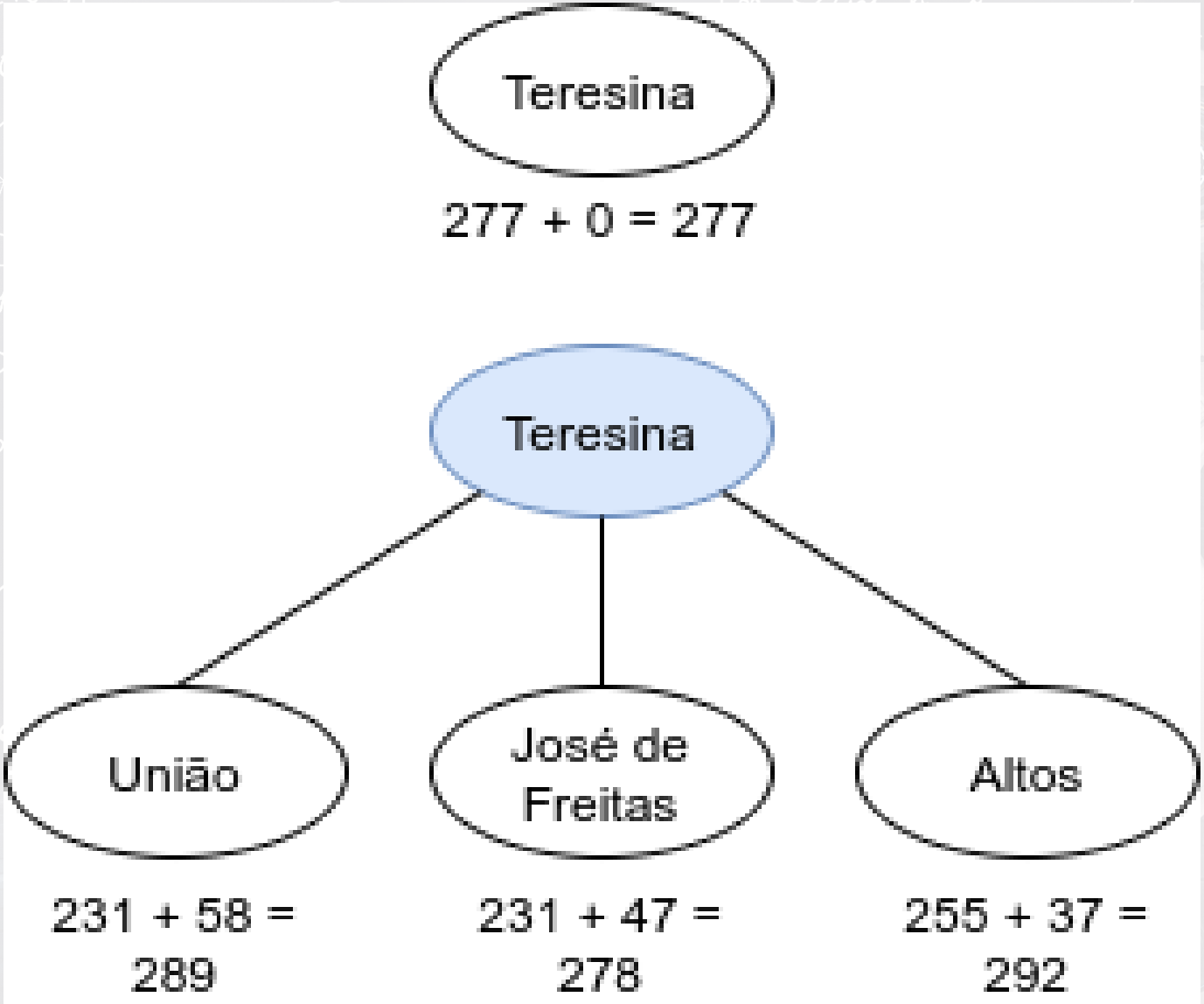
- Combina duas funções: $g(n)$, o custo para chegar até o nó e $h(n)$, o custo estimado do melhor caminho de n até a meta. Ou seja, $f(n) = g(n) + h(n)$
- É um algoritmo ótimo e completo
  - Garante encontrar a melhor solução possível, dependendo de como seja $h(n)$.

A*

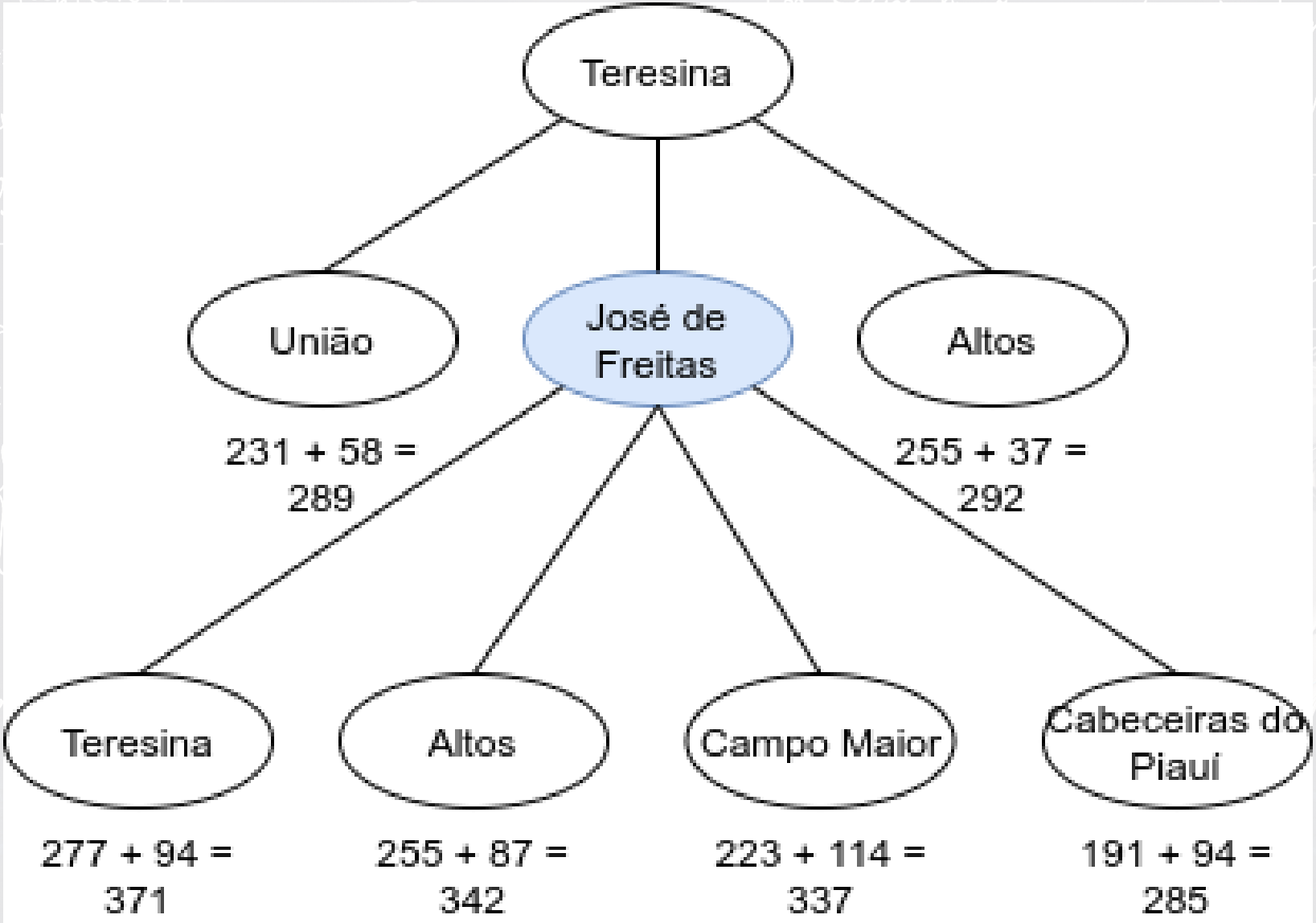
- Combina duas funções: $g(n)$, o custo para chegar até o nó e $h(n)$, o custo estimado do melhor caminho de n até a meta. Ou seja, $f(n) = g(n) + h(n)$
- É um algoritmo ótimo e completo
  - Garante encontrar uma solução.



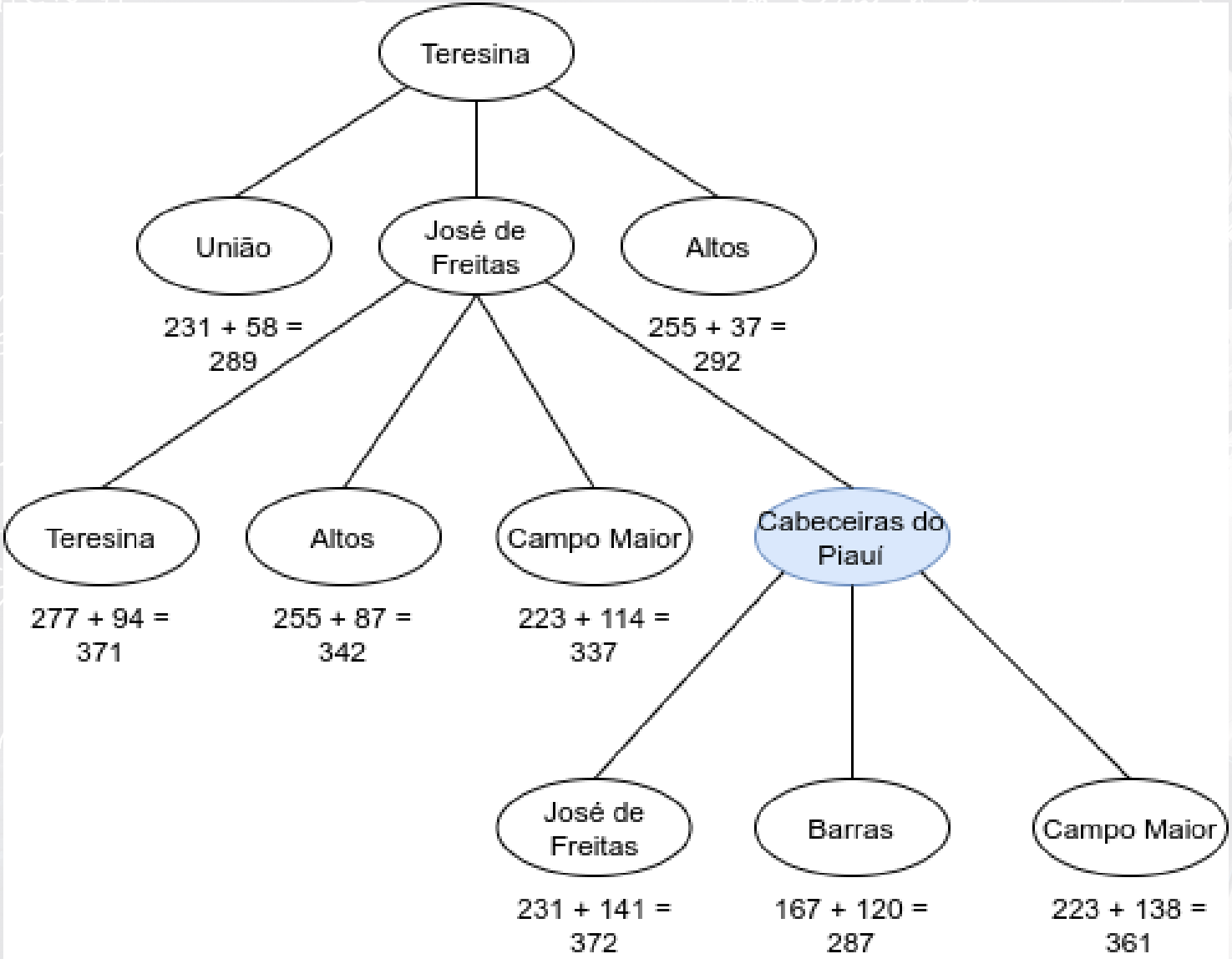
A*


Teresina
277 + 0 = 277
Teresina
União
José de Freitas
Altos
231 + 58 = 289
231 + 47 = 278
255 + 37 = 292

A*


Teresina
União
José de Freitas
Altos
231 + 58 = 289
255 + 37 = 292
Teresina
Altos
Campo Maior
Cabeceiras do Piauí
277 + 94 = 371
255 + 87 = 342
223 + 114 = 337
191 + 94 = 285

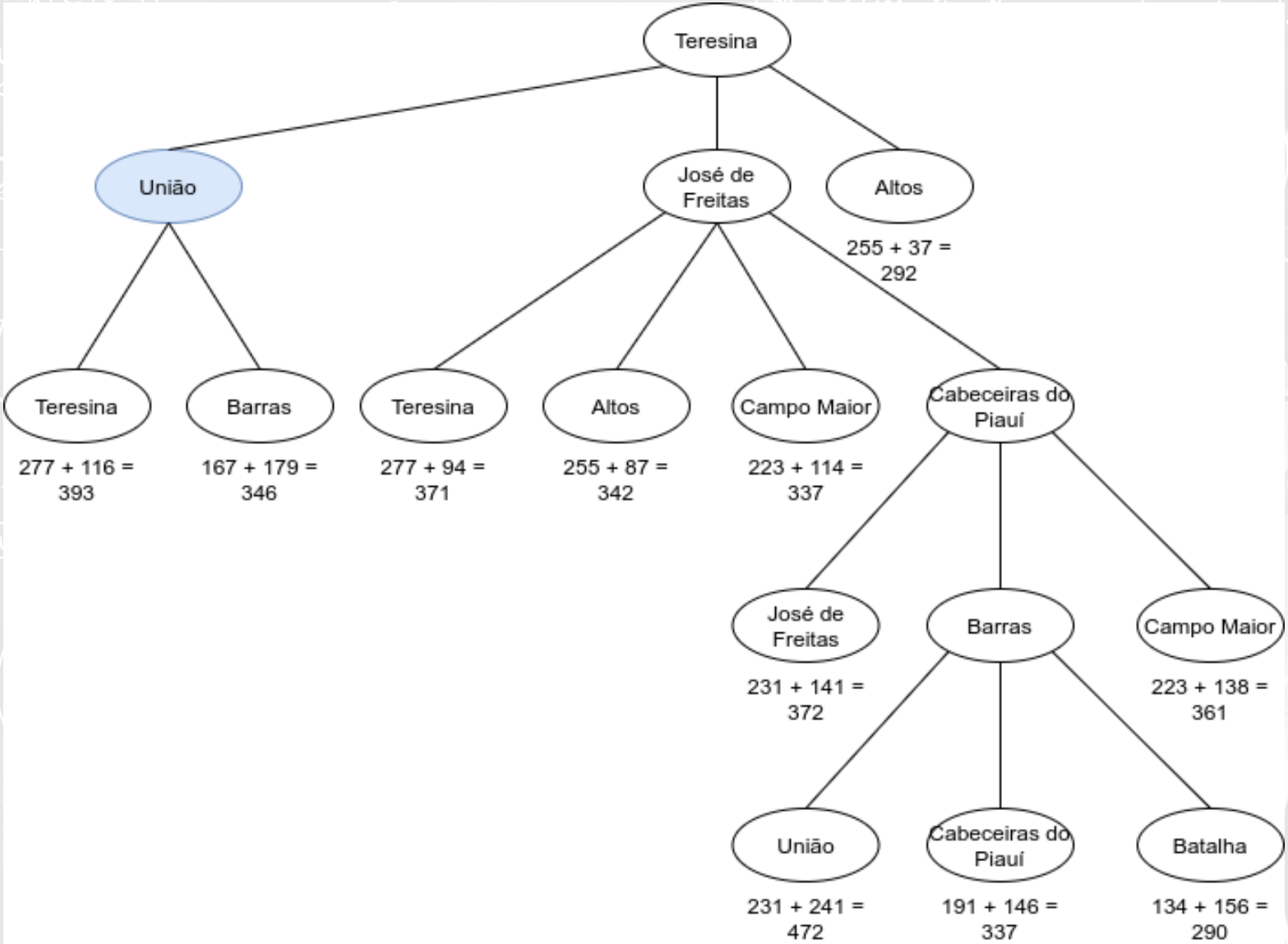
A*


Teresina
União
José de Freitas
Altos
231 + 58 = 289
255 + 37 = 292
Teresina
Altos
Campo Maior
Cabeceiras do Piauí
277 + 94 = 371
255 + 87 = 342
223 + 114 = 337
José de Freitas
Barras
Campo Maior
231 + 141 = 372
167 + 120 = 287
223 + 138 = 361

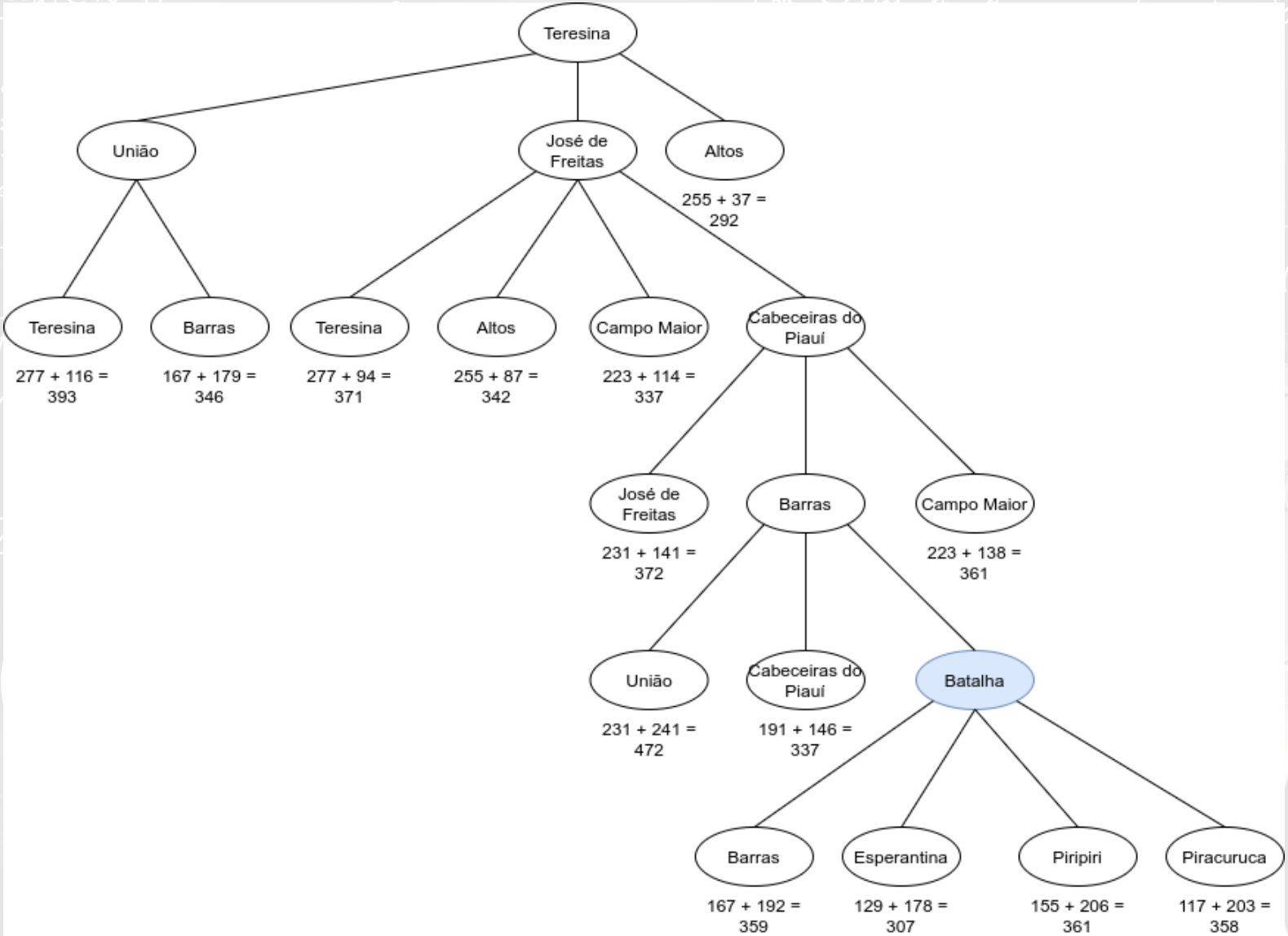
A*


Teresina
União
José de Freitas
Altos
231 + 58 = 289
255 + 37 = 292
Teresina
Altos
Campo Maior
Cabeceiras do Piauí
277 + 94 = 371
255 + 87 = 342
223 + 114 = 337
José de Freitas
Barras
Campo Maior
231 + 141 = 372
223 + 138 = 361
União
Cabeceiras do Piauí
Batalha
231 + 241 = 472
191 + 146 = 337
134 + 156 = 290

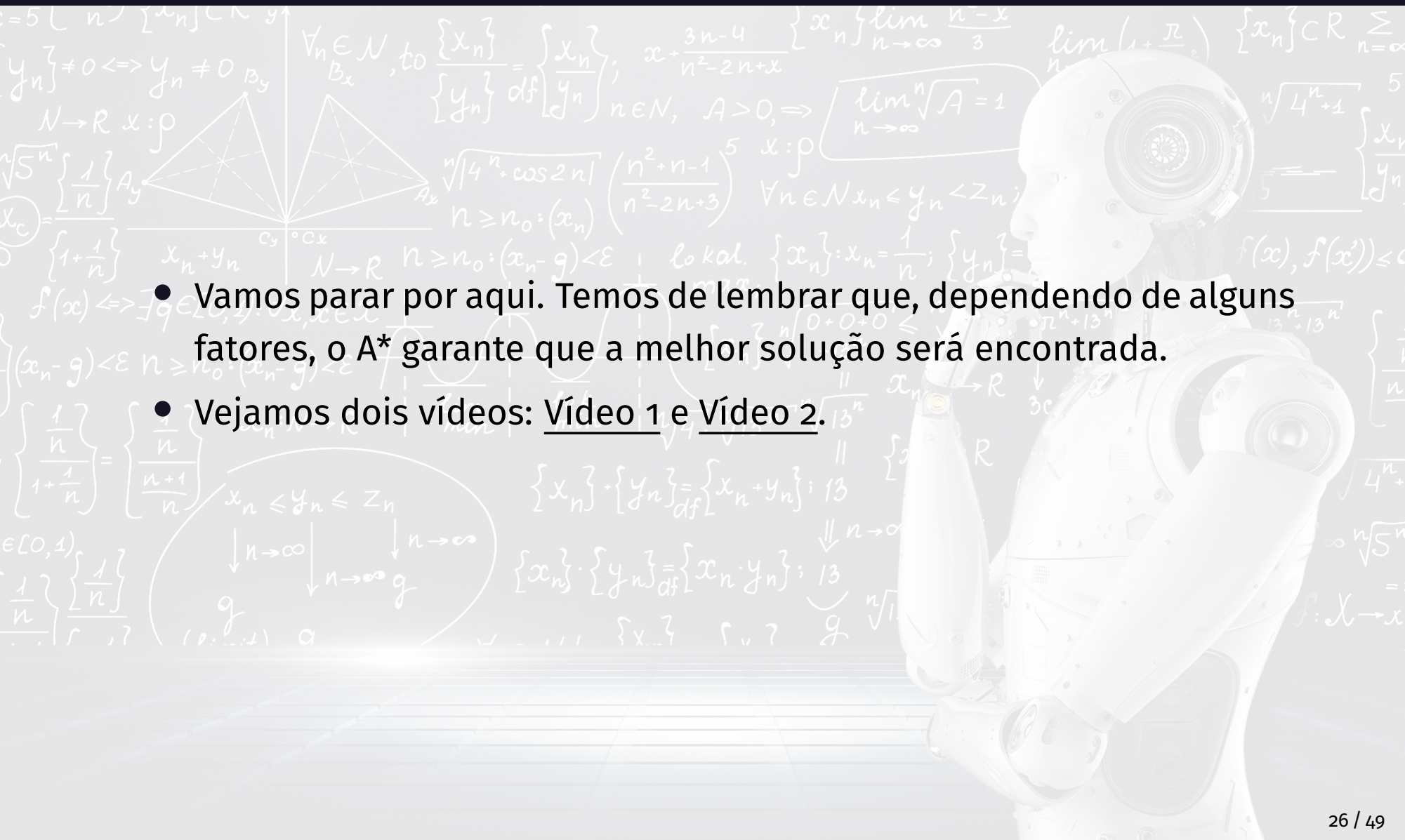
A*


Teresina
União
José de Freitas
Altos
255 + 37 = 292
Teresina
Barras
Teresina
Altos
Campo Maior
Cabeceiras do Piauí
277 + 116 = 393
167 + 179 = 346
277 + 94 = 371
255 + 87 = 342
223 + 114 = 337
José de Freitas
Barras
Campo Maior
231 + 141 = 372
223 + 138 = 361
União
Cabeceiras do Piauí
Batalha
231 + 241 = 472
191 + 146 = 337
134 + 156 = 290

A*


Teresina
União
José de Freitas
Altos
255 + 37 = 292
Teresina
277 + 116 = 393
Barras
167 + 179 = 346
Teresina
277 + 94 = 371
Altos
255 + 87 = 342
Campo Maior
223 + 114 = 337
Cabeceiras do Piauí
José de Freitas
231 + 141 = 372
Barras
Campo Maior
223 + 138 = 361
União
231 + 241 = 472
Cabeceiras do Piauí
191 + 146 = 337
Batalha
Barras
167 + 192 = 359
Esperantina
129 + 178 = 307
Piripiri
155 + 206 = 361
Piracuruca
117 + 203 = 358

A*



- Vamos parar por aqui. Temos de lembrar que, dependendo de alguns fatores, o A* garante que a melhor solução será encontrada.
- Vejamos dois vídeos: Vídeo 1 e Vídeo 2.

Inteligência de Enxame

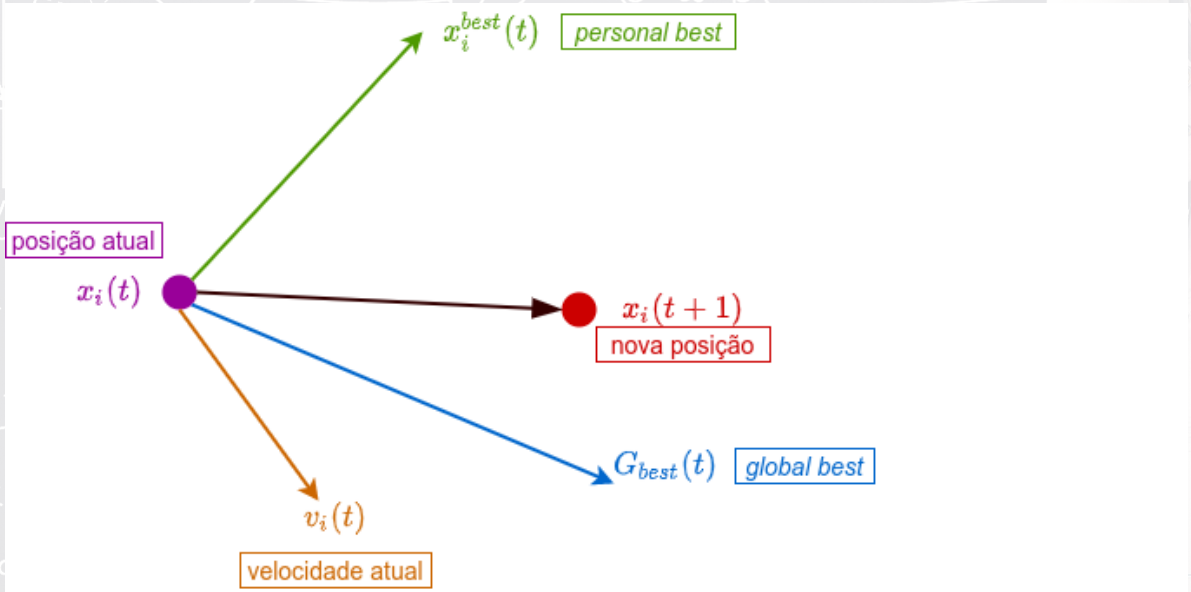
- Esses algoritmos realizam a busca da melhor solução com a utilização de uma **população de indivíduos** (ou enxame). Os **indivíduos interagem entre si**, e **influenciam uns aos outros**.
- São inspirados em comportamentos encontrados na natureza.
- Veremos os **PSO** (*Particle Swarm Optimization*, ou Otimização por Enxame de Partículas) e o **ACO** (*Ant Colony Optimization*, ou Otimização por Colônia de Formigas).

PSO

- Surgiu como uma simulação de comportamento social e, em sua forma simplificada, simula uma revoada ou um cardume.
- Ideia geral
  - Cada partícula busca o **ótimo global**, ou seja, a melhor solução para um determinado problema.
    - Cada partícula está em **movimento** e possui uma **velocidade**.
    - Cada partícula armazena sua melhor posição encontrada (*personal best*).
    - As partículas informam seus *personal best* a todas as outras, possibilitando que todas encontrem o ***global best***, ou seja, **a melhor solução encontrada até então**.

PSO

- A cada nova iteração, as partículas **ajustam** suas **velocidade** e **direção** de acordo com os *personal best* e *global best*.

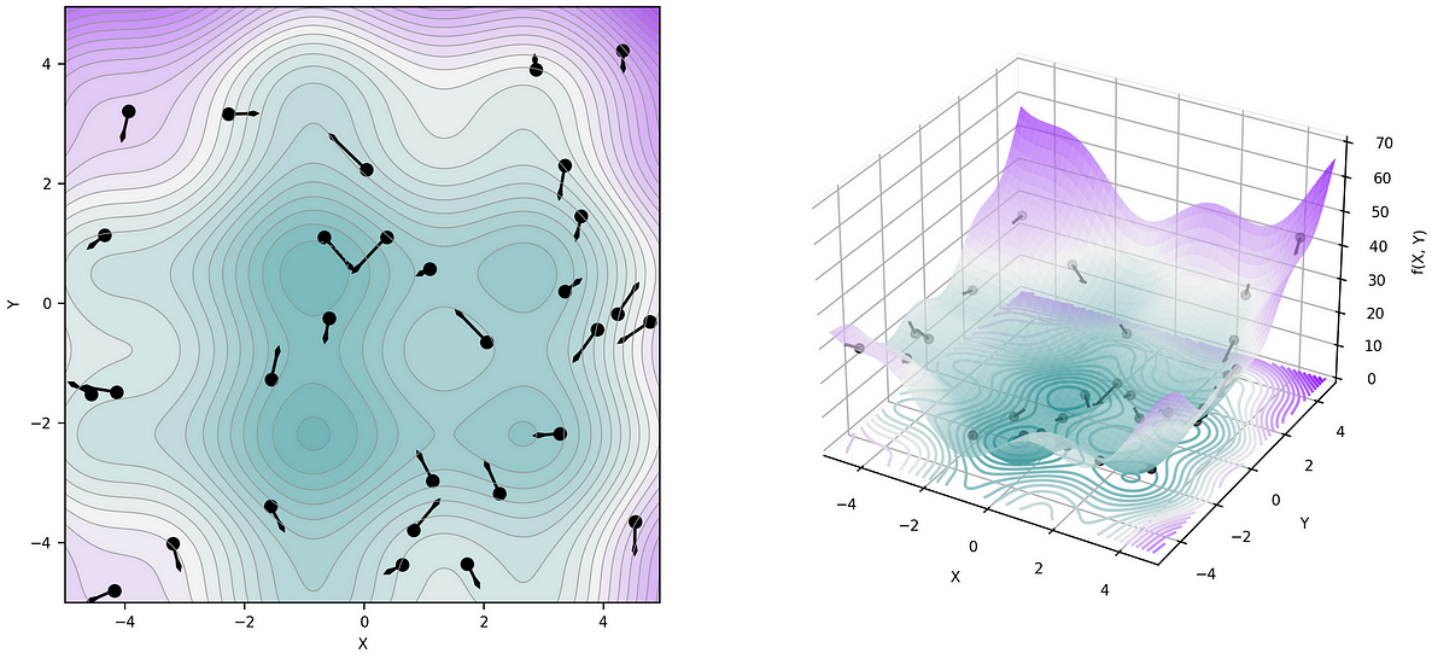


Nova velocidade: $v_i(t+1) = wv_i(t) + c_1 r_1 (x_i^{best}(t) - x_i(t)) + c_2 r_2 (G_{best}(t) - x_i(t))$

Inércia — Componente Cognitivo — Componente Social

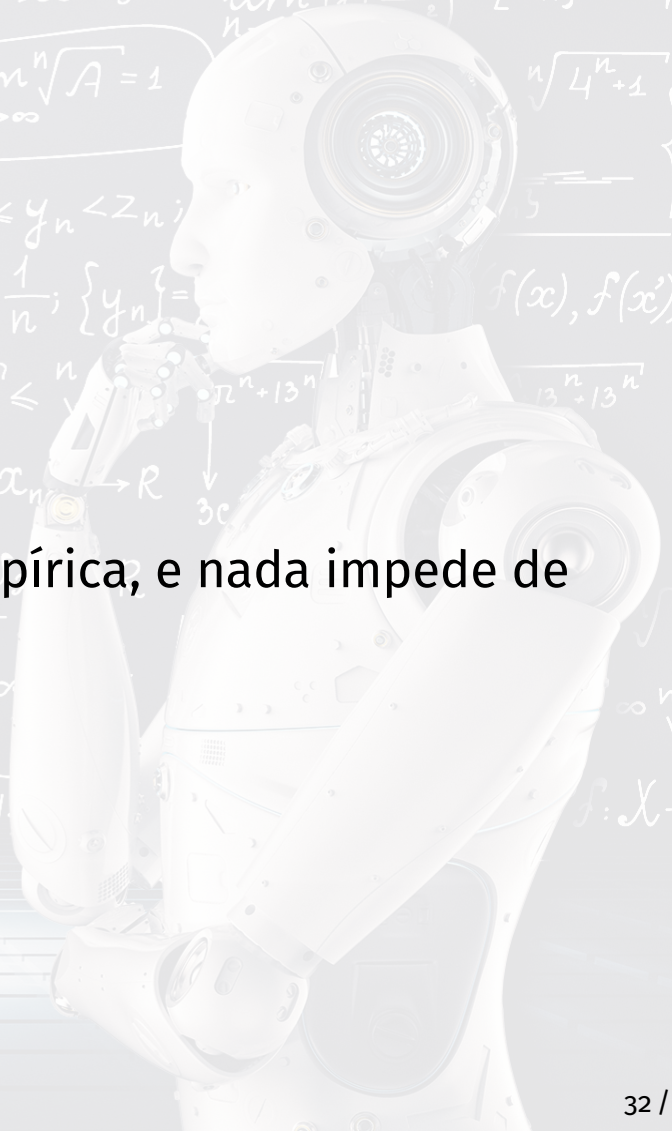
Nova posição: $x_i(t+1) = x_i(t) + v_i(t+1)$

PSO


Random initialization of $N = 30$ particles with velocity
X
Y
f(X, Y)

PSO

- Valores mais usados nos parâmetros
  - $w = 1$
  - $c_1 = c_2 = 2$
  - $r_1 = r_2 =$ número aleatório [0,1]
  - Quantidade de partículas: 10 a 50
- Esses valores foram definidos de forma empírica, e nada impede de você tentar outros valores.
- Vejamos dois vídeos: vídeo 1 e vídeo 2



ACO

- Sua principal aplicação é no encontro de melhores caminhos.
- Ideia geral
  - Cada formiga se move pelo espaço aleatoriamente, construindo uma solução.
  - Enquanto se movem, cada formiga deixa depositado no seu estado anterior um feromônio.
  - A quantidade de feromônio depositado em um dado estado, influencia a decisão de outras formigas que passarem nesse estado.
  - A quantidade de feromônio em um estado é constantemente atualizado.

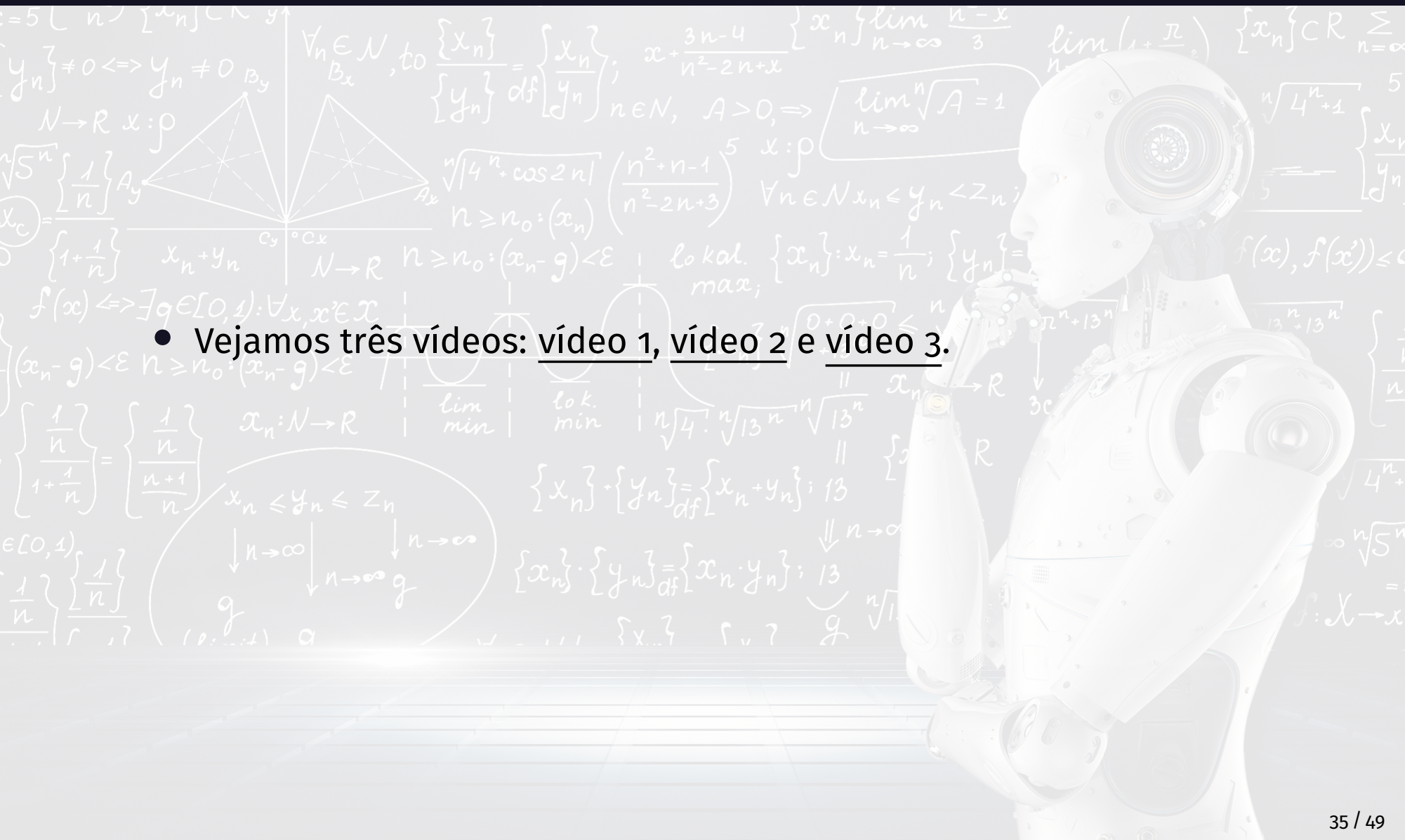
ACO

- Um pouco mais de formalidade
  - Cada formiga k se move do estado x para o y com probabilidade:

$$p_{xy}^{k} = \frac{(\tau_{xy}^{\alpha})(\eta_{xy}^{\beta})}{\sum_{z \in permitido_y} (\tau_{xy}^{\alpha})(\eta_{xy}^{\beta})}$$

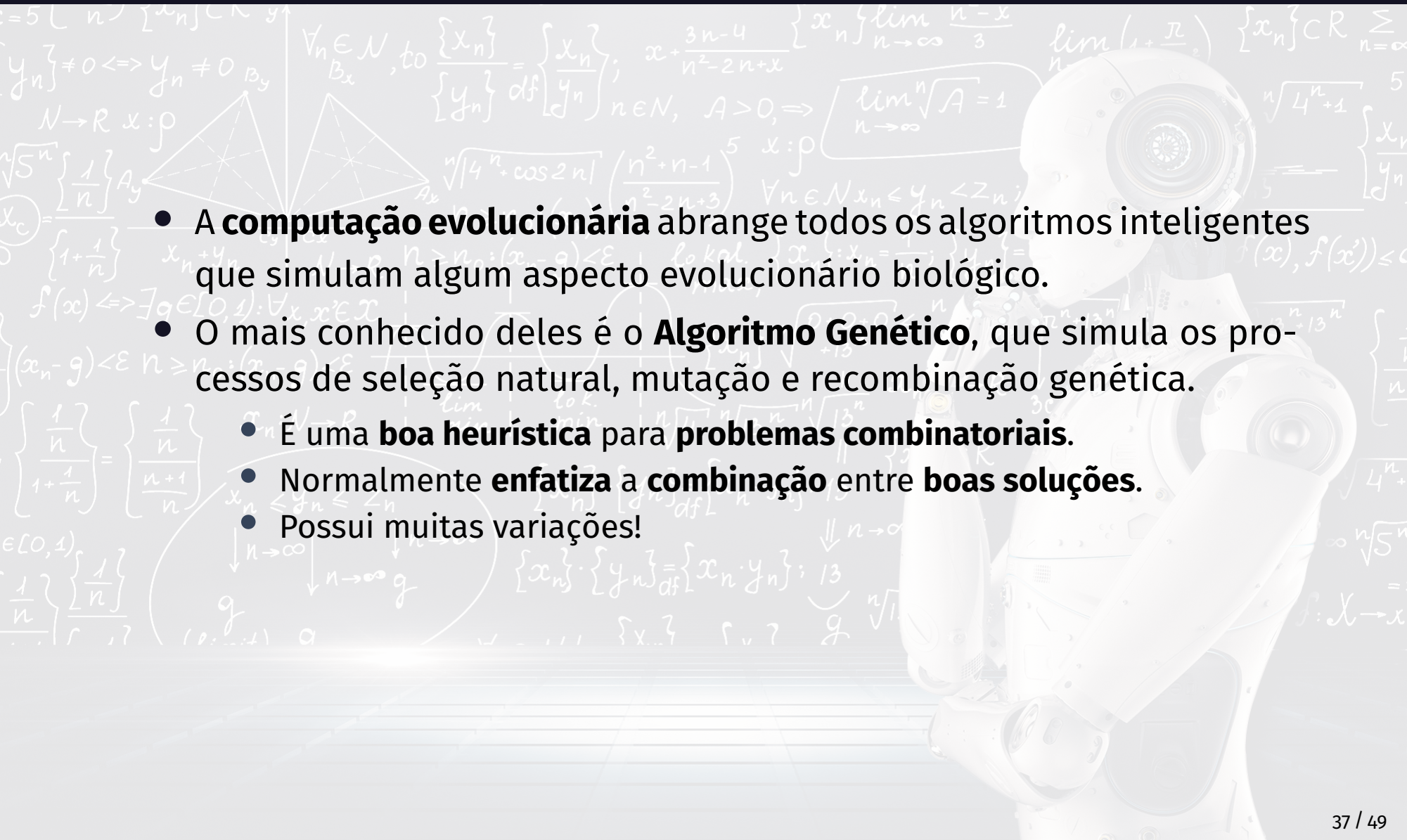
    - τ_{xy} é a quantidade de feromônio depositado na transição de x para y, e α é um parâmetro para controlar a influência de τ_{xy}.
    - η_{xy} é a desejabilidade da transição xy (um conhecimento *a prior*, normalmente $1/d_{xy}$, em que d é a distância) e β é um parâmetro para controlar a influência de η_{xy}.
  - Atualização do feromônio:
$\Delta\tau_{xy} = (1-\rho)\tau_{xy} + \sum_{k}^{m} \Delta\tau_{xy}^{k}$
    - τ_{xy} é a quantidade de feromônio depositado para uma transição de estado xy.
    - ρ é o coeficiente de evaporação do feromônio.
    - m é a quantidade de formigas.
    - $\Delta\tau_{xy}^{k}$ é a quantidade de feromônio depositado pela $k-ésima$ formiga.

ACO



- Vejamos três vídeos: vídeo 1, vídeo 2 e vídeo 3.

Computação Evolucionária



- A **computação evolucionária** abrange todos os algoritmos inteligentes que simulam algum aspecto evolucionário biológico.
- O mais conhecido deles é o **Algoritmo Genético**, que simula os processos de seleção natural, mutação e recombinação genética.
  - É uma **boa heurística** para **problemas combinatoriais**.
  - Normalmente **enfatiza** a **combinação** entre **boas soluções**.
  - Possui muitas variações!

Algoritmo Genético

- Termos técnicos
  - **Cromossomo**: a forma como uma solução é mapeada.

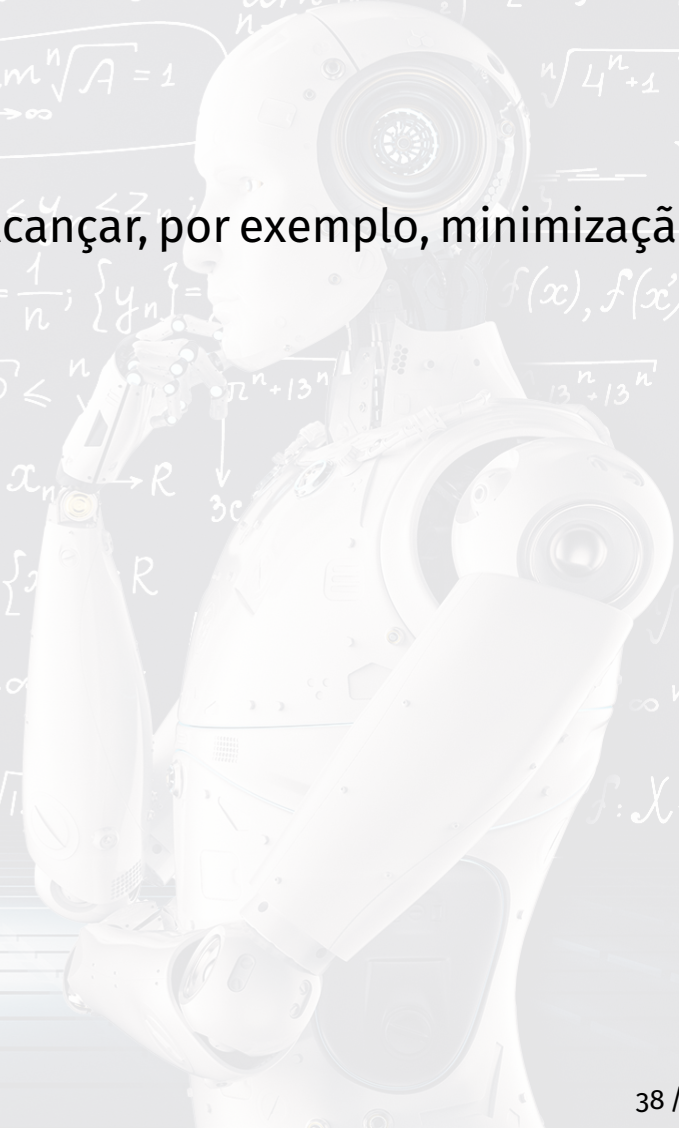
Algoritmo Genético

- Termos técnicos
  - **Fitness** (ou aptidão): o quão boa é uma solução para o problema.

Algoritmo Genético

- Termos técnicos

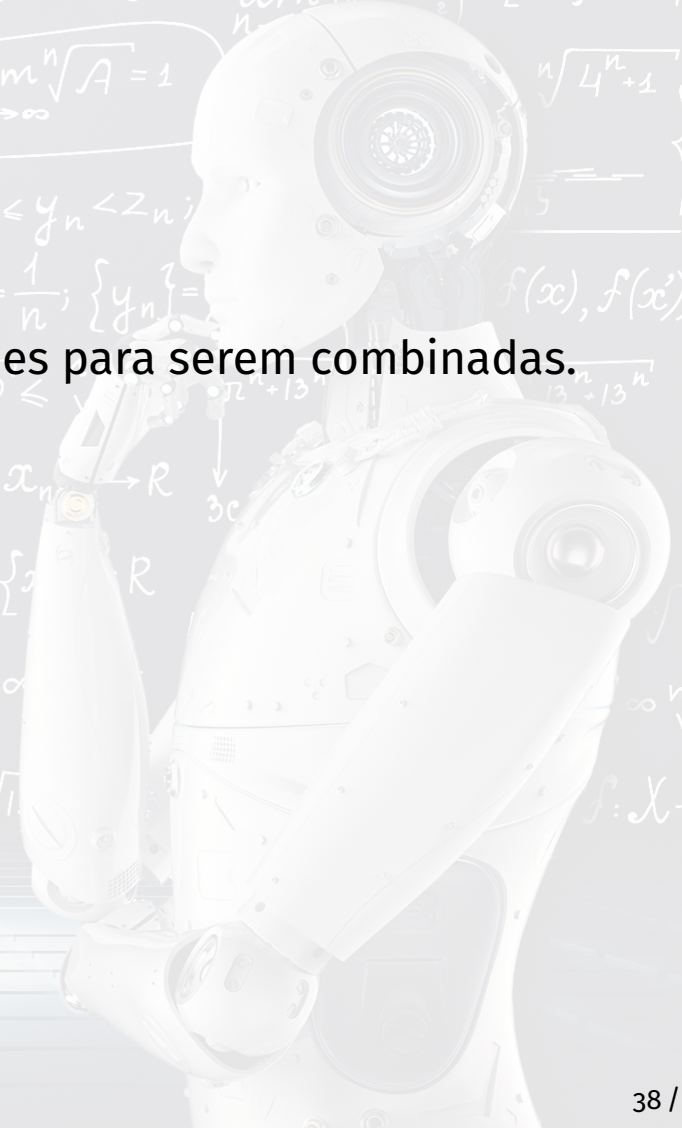
  - **Função objetivo**: o objetivo que se quer alcançar, por exemplo, minimização do valor de uma função.



Algoritmo Genético

- Termos técnicos

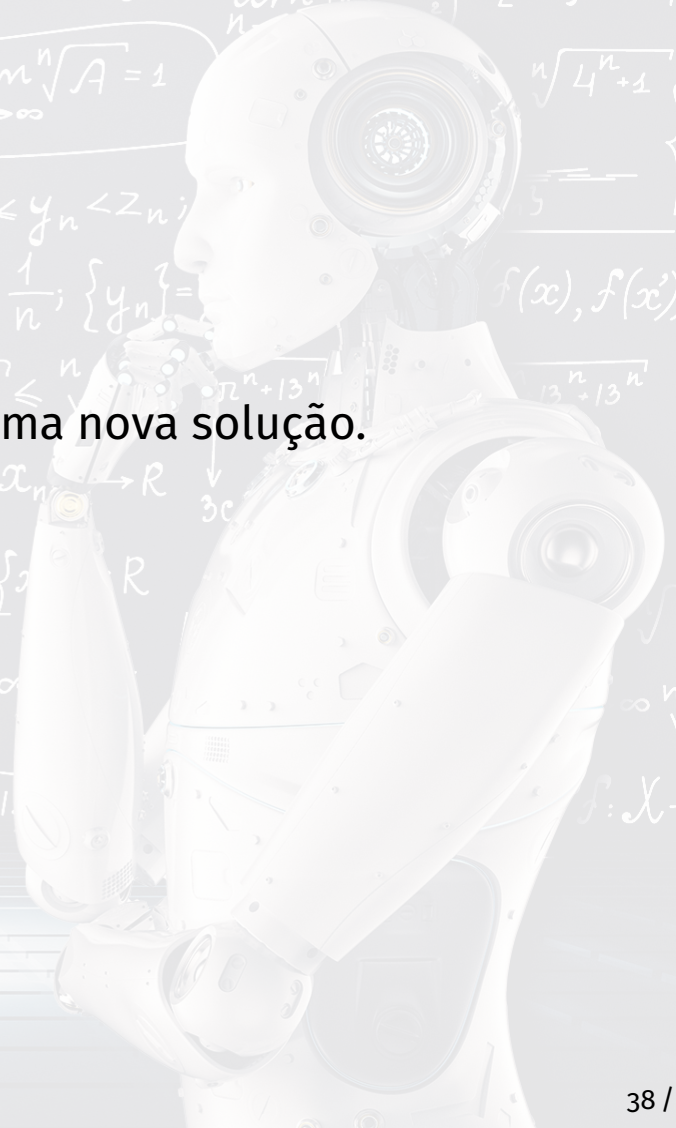
- **Seleção**: escolha de duas ou mais soluções para serem combinadas.



Algoritmo Genético

- Termos técnicos

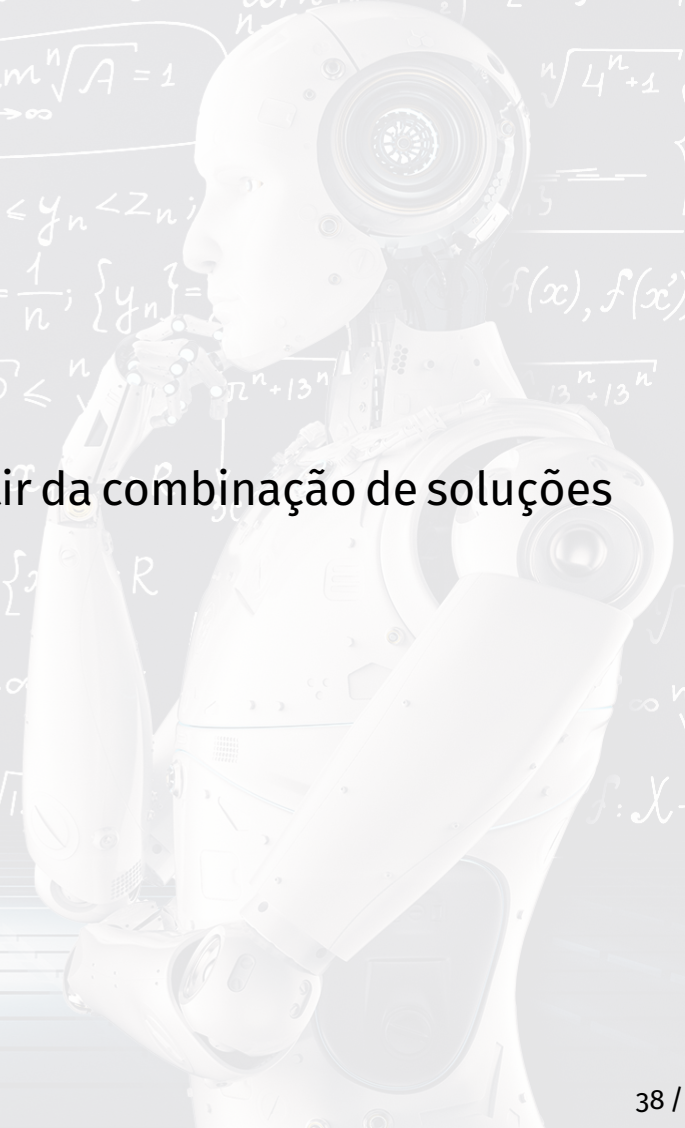
  - **Pais**: soluções escolhidas para gerarem uma nova solução.



Algoritmo Genético

- Termos técnicos

- **Prole**: solução ou soluções geradas a partir da combinação de soluções previamente existentes.



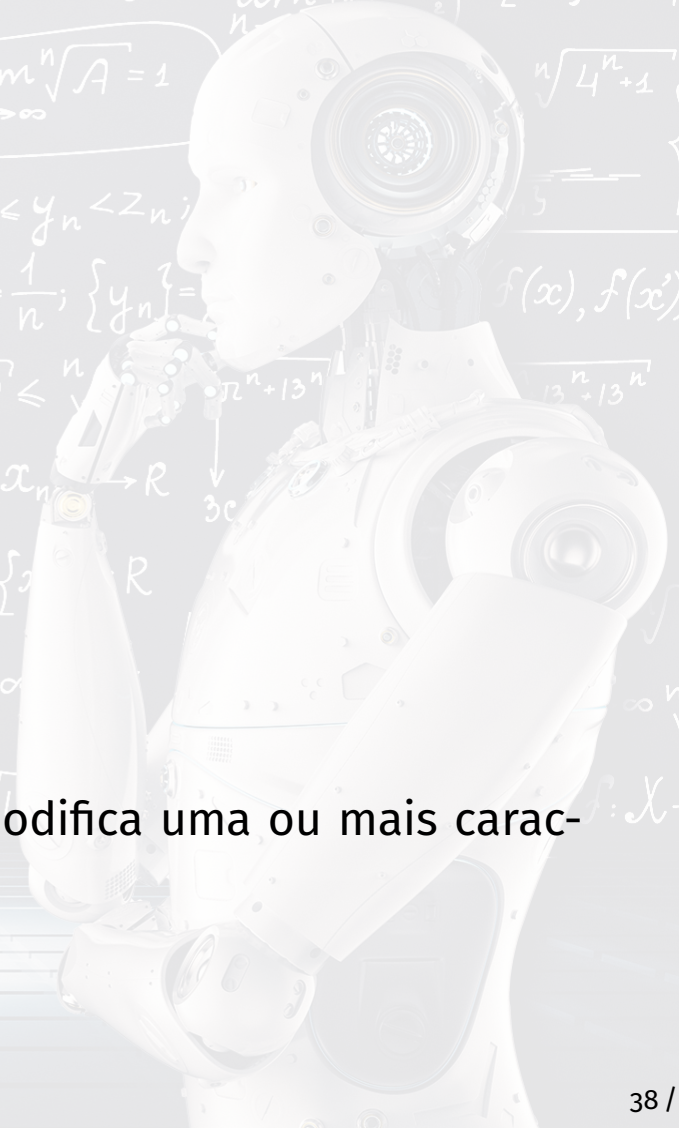
Algoritmo Genético

- Termos técnicos

- **Crossover** (ou recombinação): operação de combinação de duas ou mais soluções para gerar prole(s). Existem vários operadores de recombinação.

Algoritmo Genético

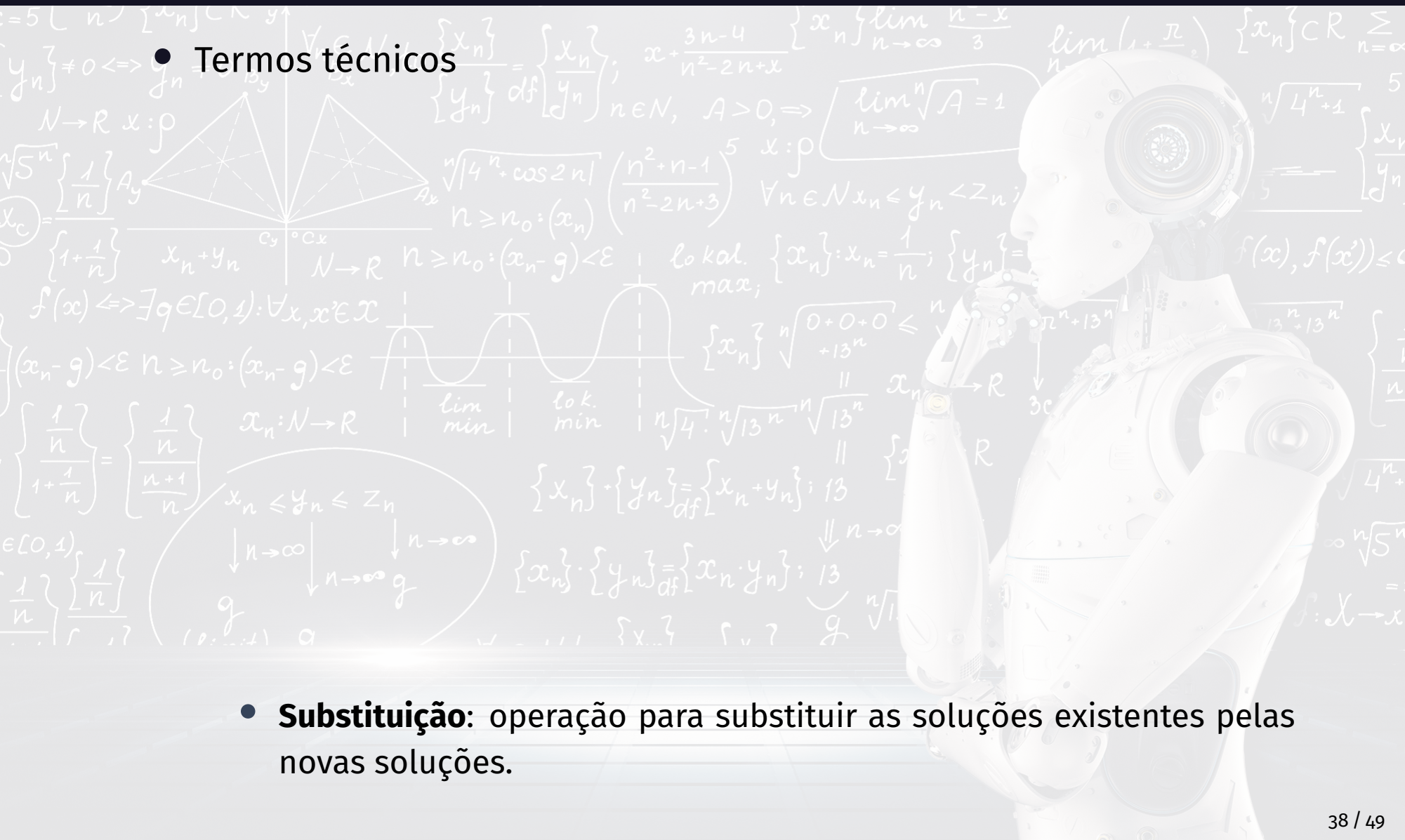
- Termos técnicos



- **Mutação**: operação sobre a prole que modifica uma ou mais características da solução.

Algoritmo Genético

- Termos técnicos

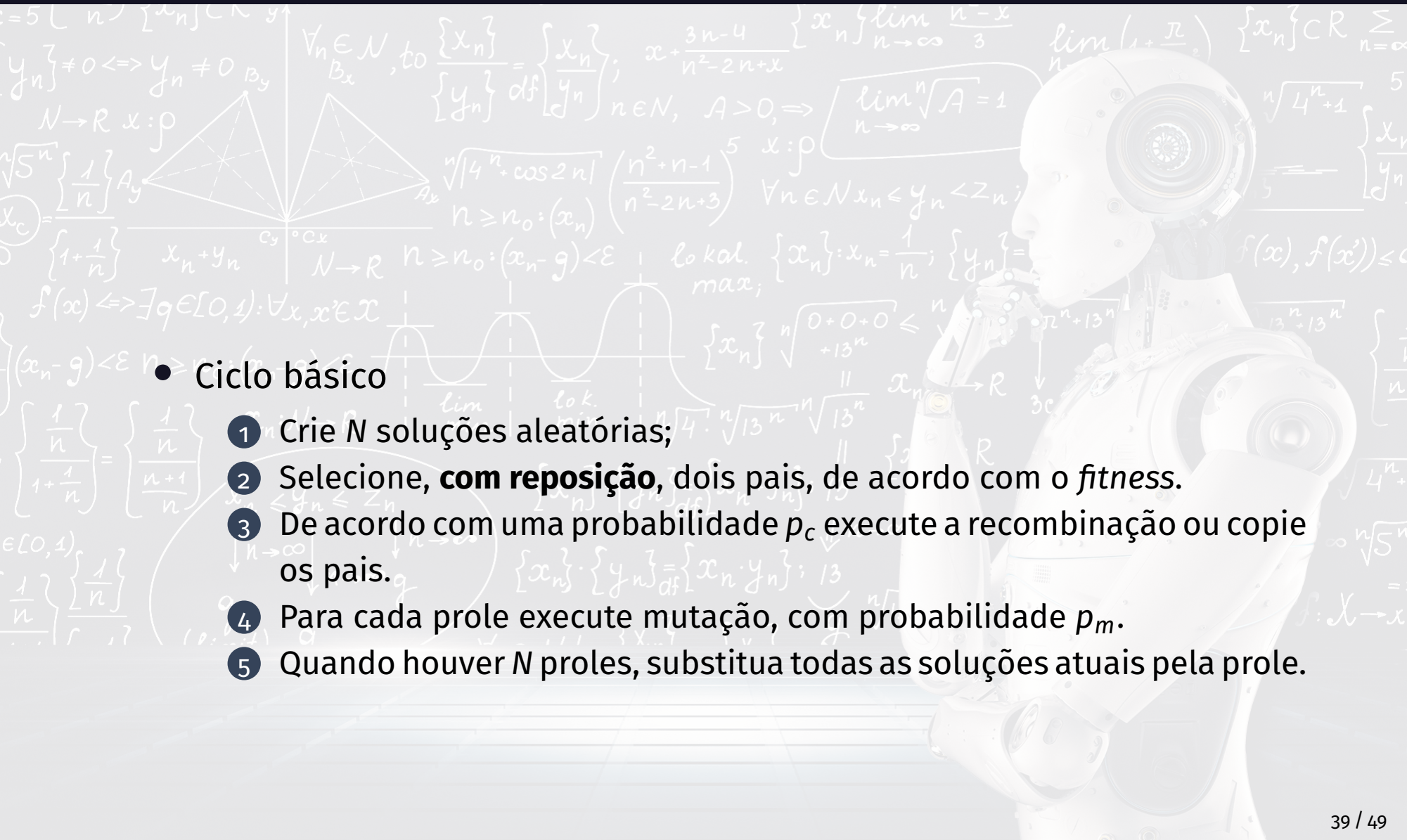


- **Substituição**: operação para substituir as soluções existentes pelas novas soluções.

Algoritmo Genético

- Em sua versão mais básica apresenta as seguintes características
  - Representação binária.
  - Seleção parental proporcional ao *fitness*.
  - Baixa probabilidade de mutação.
  - O esquema de substituição é geracional.

Algoritmo Genético



- Ciclo básico
  1. Crie N soluções aleatórias;
  2. Selecione, **com reposição**, dois pais, de acordo com o *fitness*.
  3. De acordo com uma probabilidade p_c execute a recombinação ou copie os pais.
  4. Para cada prole execute mutação, com probabilidade p_m.
  5. Quando houver N proles, substitua todas as soluções atuais pela prole.

Algoritmo Genético

Exemplo

Maior valor para a função $f(x) = x^2 + 4x$ no limite [0, 15].

Como existem 16 valores na faixa dos valores possíveis, podemos mapear todos os valores com 4 bits.

Começando com $N = 4$
$(1) = 0011 \therefore (1) = 03 \therefore \mathit{fitness} = 21 \therefore 8\%$
$(2) = 0101 \therefore (2) = 05 \therefore \mathit{fitness} = 45 \therefore 17\%$
$(3) = 1100 \therefore (3) = 12 \therefore \mathit{fitness} = 192 \therefore 73\%$
$(4) = 0001 \therefore (4) = 01 \therefore \mathit{fitness} = 5 \therefore 2\%$

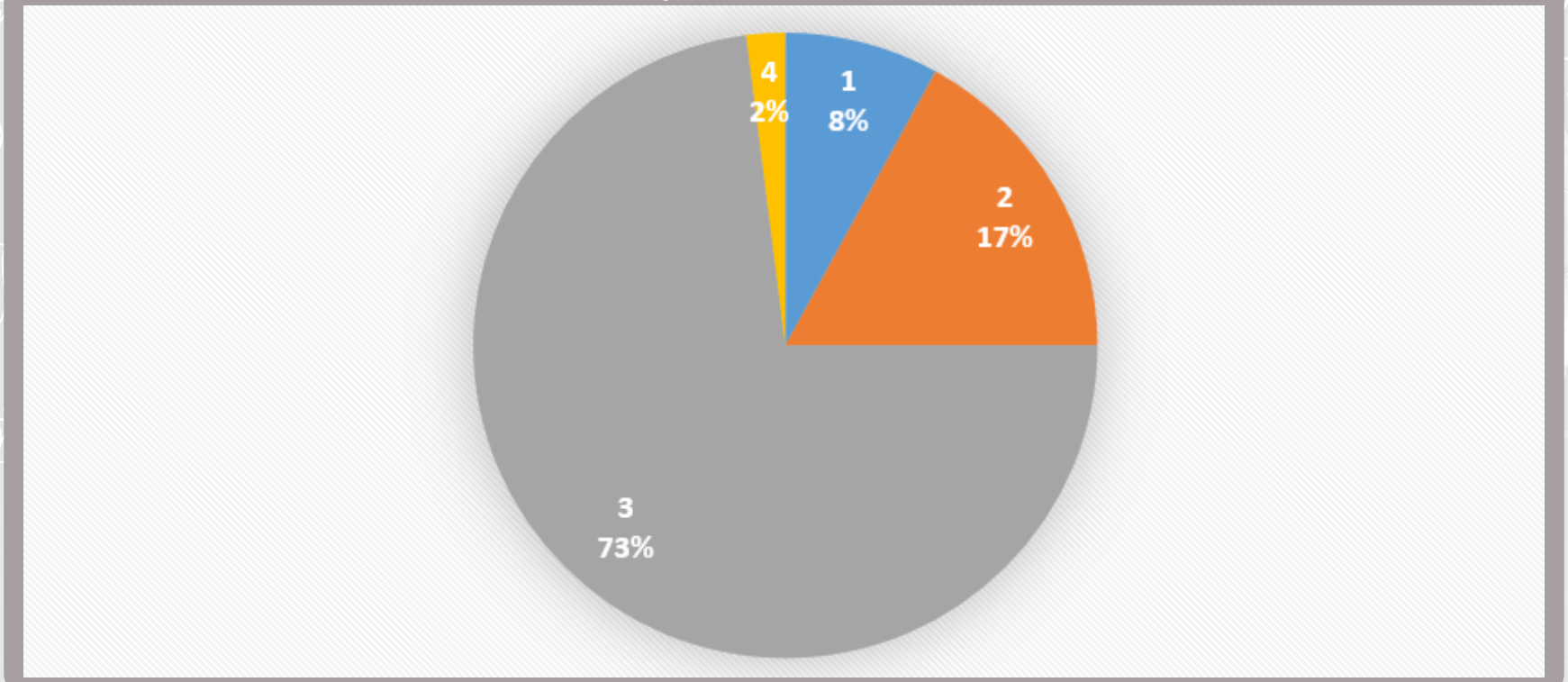
Soma dos fitness = 263

Algoritmo Genético

Exemplo

Maior valor para a função $f(x) = x^2 + 4x$ no limite [0, 15].

Escolha dos pais — método da roleta



Algoritmo Genético

Exemplo

Maior valor para a função $f(x) = x^2 + 4x$ no limite [0, 15].

Pais escolhidos: [(1) (3)] e [(3) (2)]
Probabilidade de recombinação: 100%
Probabilidade de mutação: 1%
Ponto de corte: 2

Recombinação:

(1) **00**11 00**11**
(3) 11**00** **11**00

(3) **11**00 11**00**
(2) 01**01** **01**01

Algoritmo Genético

Exemplo

Maior valor para a função $f(x) = x^2 + 4x$ no limite [0, 15].

Pais escolhidos: [(1) (3)] e [(3) (2)]
Probabilidade de recombinação: 100%
Probabilidade de mutação: 1%
Ponto de corte: 2

Prole:

0000	1101
1111	0100

Algoritmo Genético

Exemplo

Maior valor para a função $f(x) = x^2 + 4x$ no limite [0, 15].

Pais escolhidos: [(1) (3)] e [(3) (2)]

Probabilidade de recombinação: 100%

Probabilidade de mutação: 1%

Ponto de corte: 2

Mutação:

0000	1101
1111	0100

Algoritmo Genético

Exemplo

Maior valor para a função $f(x) = x^2 + 4x$ no limite [0, 15].

Pais escolhidos: [(1) (3)] e [(3) (2)]

Probabilidade de recombinação: 100%

Probabilidade de mutação: 1%

Ponto de corte: 2

Nova População:

1000	1101
1101	0100

Algoritmo Genético

Exemplo

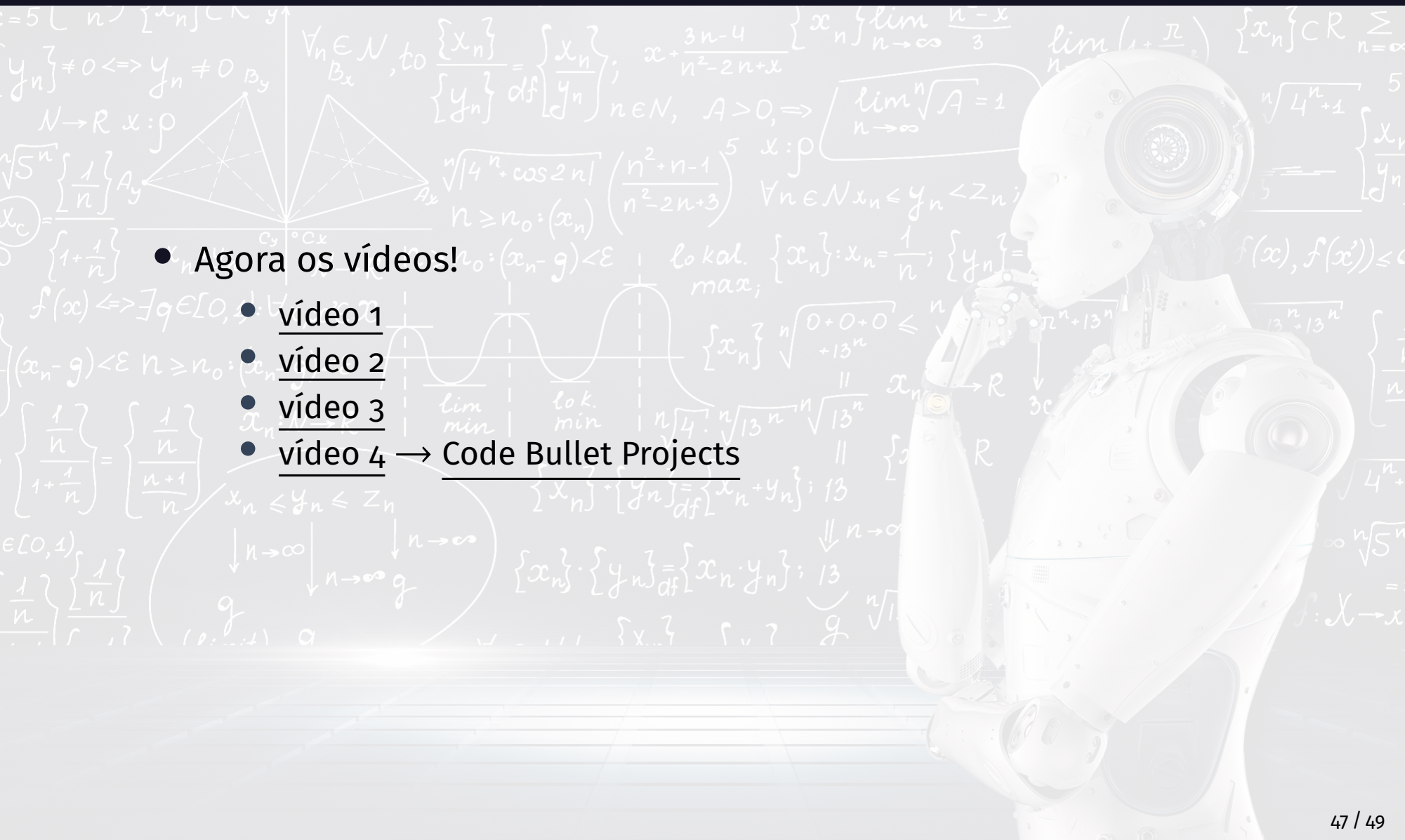
Maior valor para a função $f(x) = x^2 + 4x$ no limite [0, 15].

Como existem 16 valores na faixa dos valores possíveis, podemos mapear todos os valores com 4 bits.

Começando com $N = 4$
$(1) = 1000 \therefore (1) = 08 \therefore \mathit{fitness} = 96 \therefore 17\%$
$(2) = 1101 \therefore (2) = 13 \therefore \mathit{fitness} = 221 \therefore 39\%$
$(3) = 1101 \therefore (3) = 13 \therefore \mathit{fitness} = 221 \therefore 39\%$
$(4) = 0100 \therefore (4) = 04 \therefore \mathit{fitness} = 32 \therefore 5\%$

Soma dos fitness = 570

Algoritmo Genético



- Agora os vídeos!
  - vídeo 1
  - vídeo 2
  - vídeo 3
  - vídeo 4 → Code Bullet Projects

FIM