

# Aula 7 – Busca Heurística

22705/1001336 - Inteligência Artificial  
2019/1 - Turma A  
Prof. Dr. Murilo Naldi

[naldi@dc.ufscar.br](mailto:naldi@dc.ufscar.br)

# Agradecimentos

- Parte do material utilizado nesta aula foi inspirado pelo material cedido pelas professoras Ricardo Campello, Ricardo Cerri, Heloisa Arruda, Solange Rezende e Andréia Bonfante, e, por esse motivo, o crédito deste material é delas.

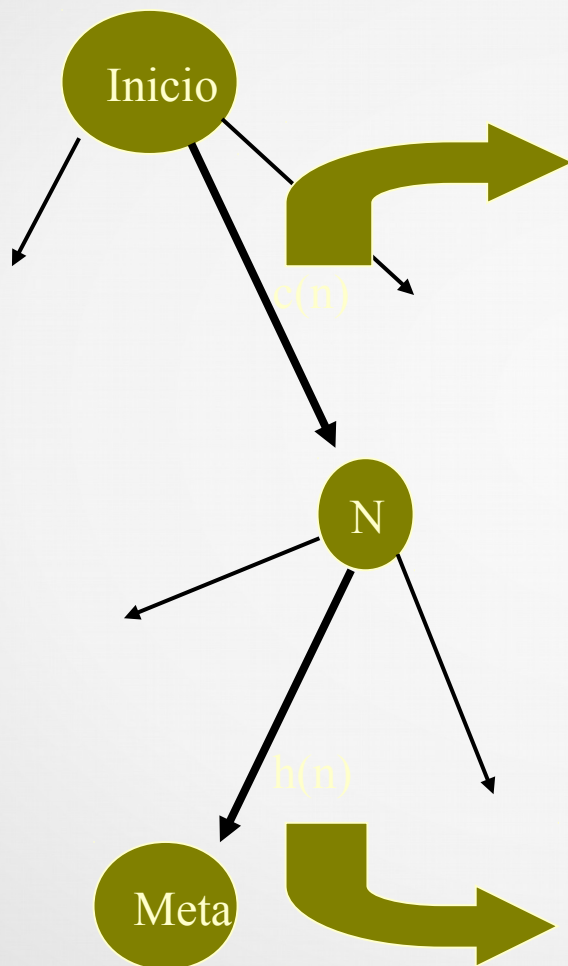
# Busca Heurística

- Conhecida como busca informada
- Usa informações do domínio para decidir qual caminho deve ser percorrido
  - Informação quando aplicada demonstra conhecimento
  - *“A rule of thumb, simplification, or educated guess that reduces or limits the search for solutions in domains that are difficult and poorly understood”.*
- Tende a ser mais eficiente que a busca cega

## Função de avaliação

- **Função de avaliação** – quantifica a qualidade de uma solução
  - retorna um valor de suporte para o próximo passo na busca
- Geralmente, a função de avaliação é composta de duas outras funções:
  - Função de custo
  - Função heurística

# Heurística e Custo



Podemos usar o custo do caminho  $c(n)$  para decidir que nó estender. Já fizemos isto na Busca de Custo Uniforme. Porém, esta medida não dirige a busca para o nó objetivo. Ela é relativa ao caminho passado.

A função heurística calcula o custo estimado para se alcançar o nó objetivo. Denotamos por  $h(n)$  e estima/infero o melhor o caminho do nó  $n$  para o nó objetivo. Ela estima do futuro.



## Best First (BF)

- Explora primeiro os estados considerados “melhores”
- Utiliza uma função de avaliação  $f(n)$ , que avalia o quão bom é um estado
  - pode ser crescente ou decrescente
- A função de avaliação varia com o tipo de algoritmo
  - Exemplo: se  $f(n) = c(n)$ , então é expandido primeiro o estado de melhor custo (busca de custo uniforme)

# Best First (BF)

- 1 Inserir em  $F$  os nós iniciais em ordem de  $f(n)$
- 2 Se  $F$  é vazio
  - 2.1 Então a busca não foi bem sucedida
- 3 Senão seja  $n$  o primeiro estado de  $F$ 
  - 3.1 Se  $n$  é um estado meta então
    - 3.1.1 Retornar caminho do estado inicial até  $n$
  - 3.2 Senão
    - 3.2.1 Remover  $n$  de  $F$  e inserir em  $E$
    - 3.2.2 Adicionar em ordem de  $f(n)$  todos os sucessores de  $n$  que não estão em  $E$  e incrementa o caminho
    - 3.2.3 Voltar ao passo 2

## *Greedy Best First (GBF)*

- A versão gananciosa (gulosa) do Best First consiste em usar a função heurística  $h(n)$  para avaliar qual é o próximo estado.
  - Ou seja,  $f(n) = h(n)$ !
- Geralmente, quanto menor o valor de  $h(n)$ , melhor o estado será em relação ao objetivo!
- Como  $h(n)$  é fixo por estado, não se repete estados na fronteira!



# Greed Best First (GBF)

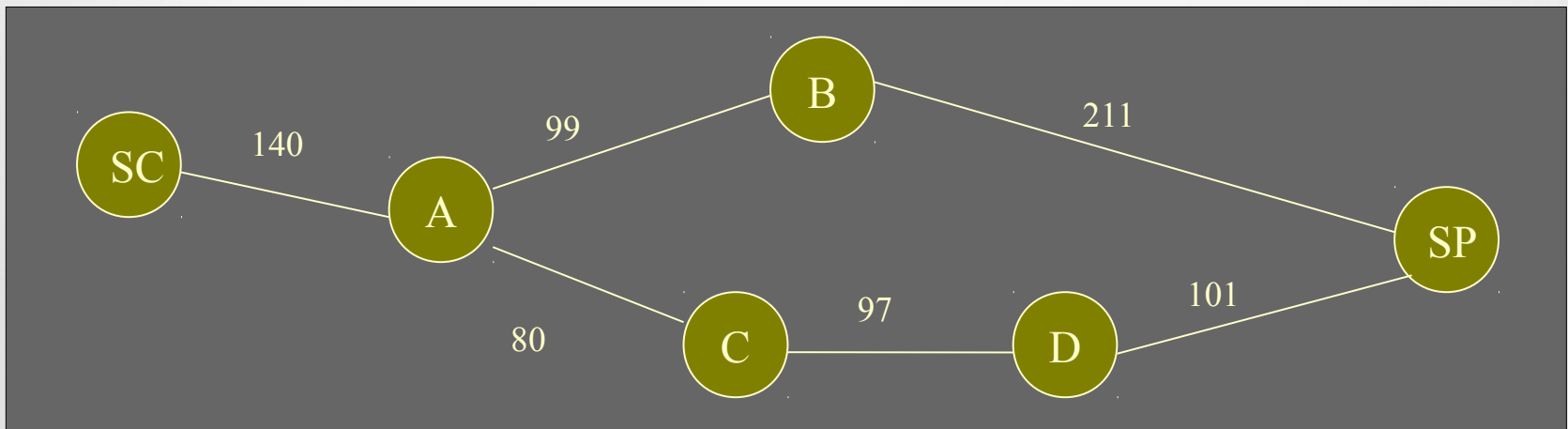
- 1 Inserir em **F** os nós iniciais em **ordem crescente de  $h(n)$**
- 2 Se **F** é vazio
  - 2.1 Então a busca não foi bem sucedida
- 3 Senão seja **n** o primeiro estado de **F**
  - 3.1 Se **n** é um estado meta então
    - 3.1.1 Retornar caminho do estado inicial até **n**
  - 3.2 Senão
    - 3.2.1 Remover **n** de **F** e inserir em **E**
    - 3.2.2 **Adicionar em ordem de  $h(n)$**  todos os sucessores de **n** **que não estão em E** e **incrementa o caminho**
    - 3.2.3 Voltar ao passo 2

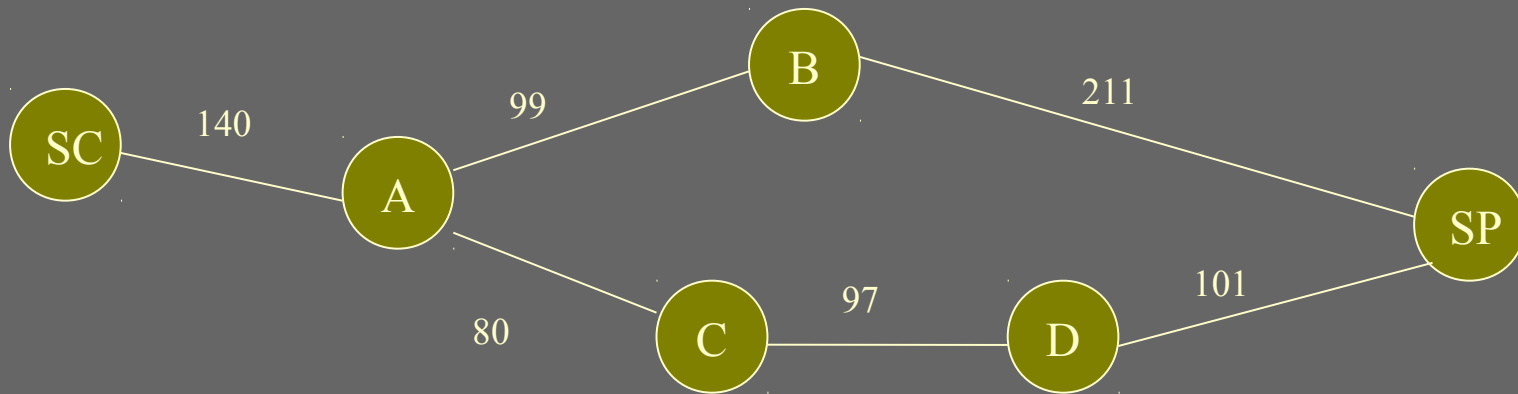
## Propriedades do GBF

- **Completo:** não, pois estende pelo melhor caminho que pode ser infinito
- **Complexidade de Tempo:**  $O(b^m)$ .
- **Complexidade de Espaço:**  $O(b^m)$  - mantém todos os nós.
- **Admissível:** por não ser completo, também não é admissível
- **Vantagem:** pode reduzir espaço e tempo com o uso de uma boa função heurística.

# Exemplo

- Problema do cálculo de rotas
- Uma função heurística bastante utilizada para este tipo de problema consiste na distância em linha reta até o objetivo
- Geralmente, utiliza-se a distância Euclidiana



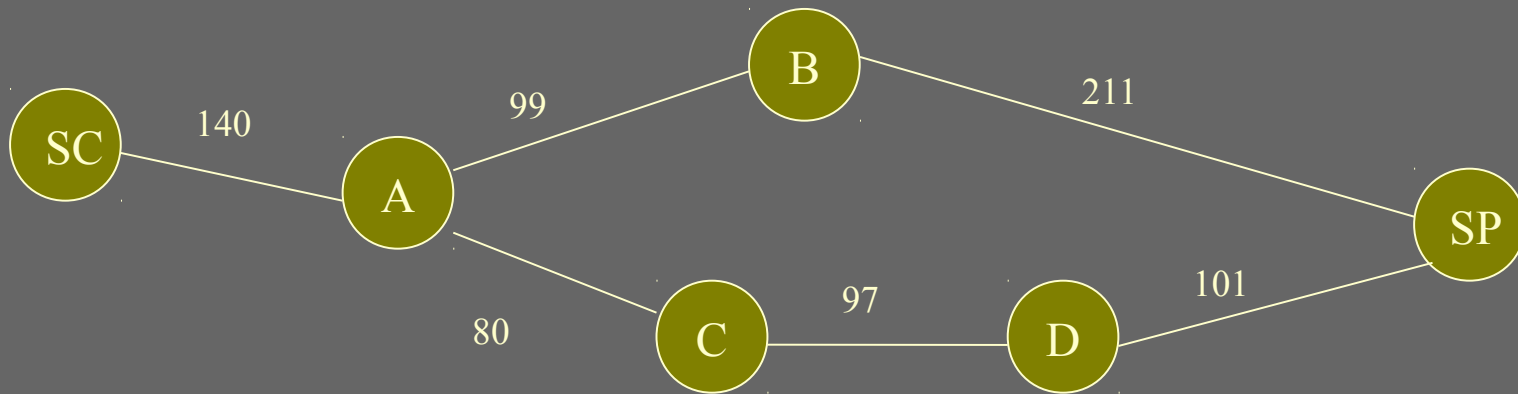


- A distância entre as cidades e São Paulo é calcula por meio de:
  - $D = ((X2 - X1)^2 + (Y2 - Y1)^2)^{1/2}$ .
  - Os resultados estão presentes na tabela ao lado ->
  - Qual é o melhor caminho segundo o algoritmo *GBF*?

Tabela c/  
distância  
em linha  
reta:

SC	- 366
A	- 253
B	- 178
C	- 193
D	- 98
SP	- 0





$F = \{[SC](366)\}, E = \{\}$

$F = ? E = ?$

Tabela c/  
distância  
em linha  
reta:

SC - 366

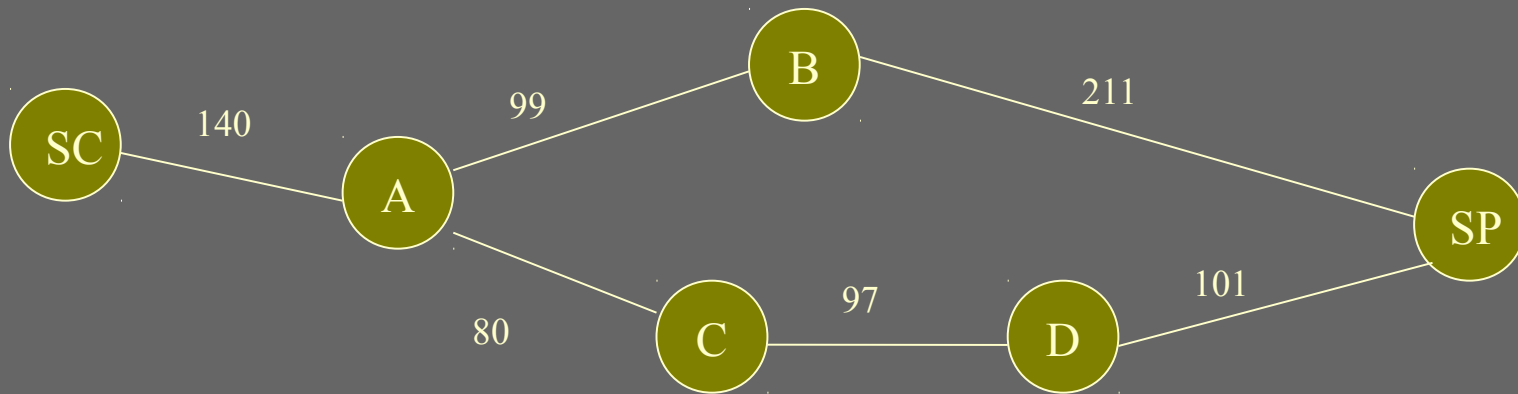
A - 253

B - 178

C - 193

D - 98

SP- 0



$F = \{[SC](366)\}, E = \{\}$

- $F = \{[A, SC](253)\}, E = \{SC\}$

$F = ? \quad E = ?$

Tabela c/  
distância  
em linha  
reta:

SC - 366

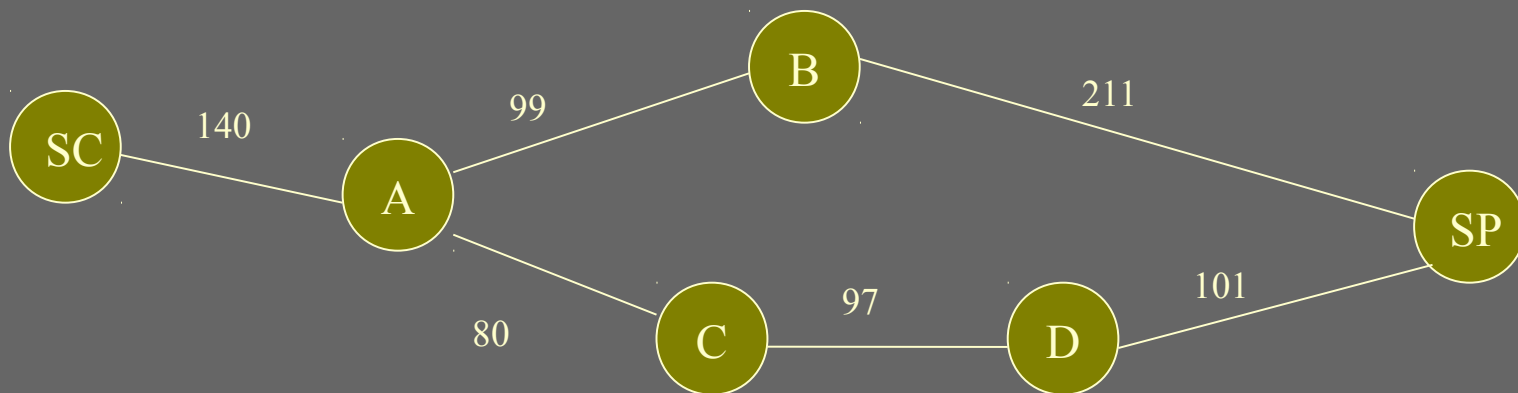
A - 253

B - 178

C - 193

D - 98

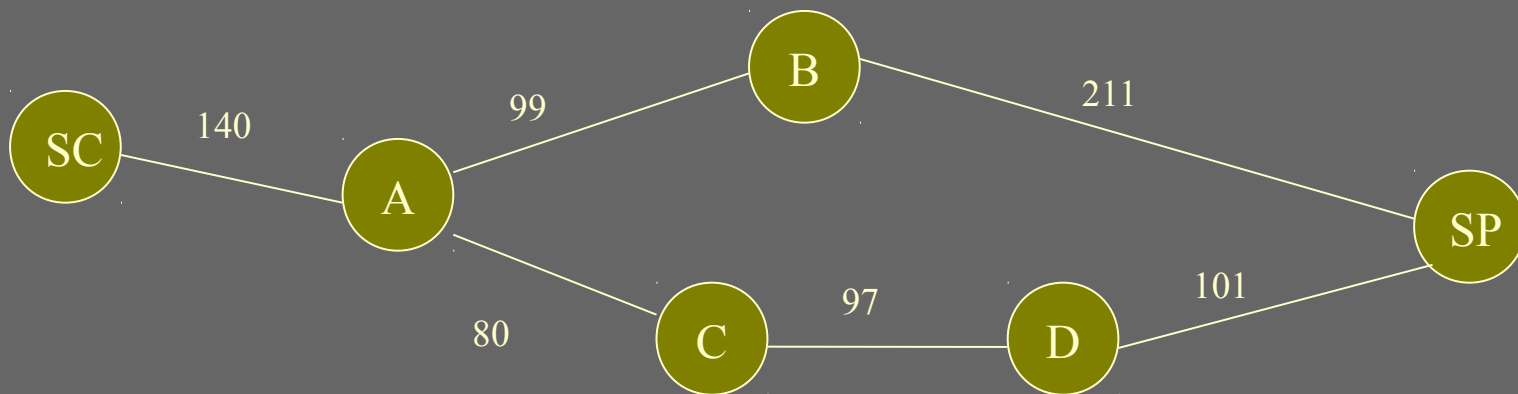
SP- 0



- $F = \{[SC](366)\}$ ,  $E = \{\}$
- $F = \{[A,SC](253)\}$ ,  $E = \{SC\}$
- $F = \{[B,A,SC](178), [C,A,SC](193)\}$ ,
- $E = \{SC, A\}$
- $F = ?$   $E = ?$

Tabela c/  
distância  
em linha  
reta:

SC	- 366
A	- 253
B	- 178
C	- 193
D	- 98
SP	- 0

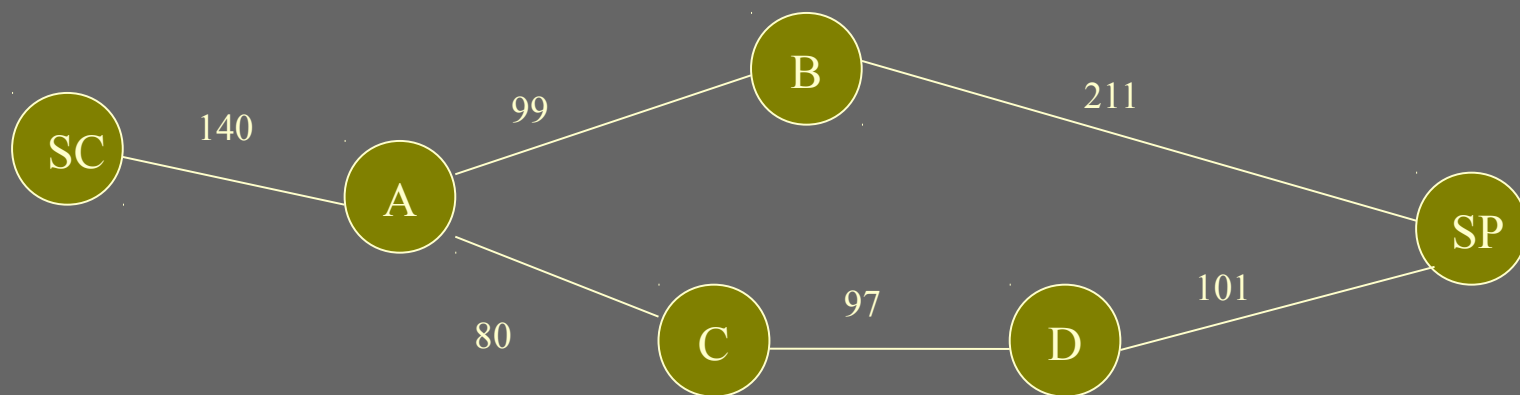


- $F = \{[SC](366)\}$ ,  $E = \{\}$
- $F = \{[A, SC](253)\}$ ,  $E = \{SC\}$
- $F = \{[B, A, SC](178), [C, A, SC](193)\}$ ,
- $E = \{SC, A\}$
- $F = \{[SP, B, A, SC](0), [C, A, SC](193)\}$ ,
- $E = \{SC, A, B\}$
- Retorna SP, caminho SC-A-B

Tabela c/  
distância  
em linha  
reta:

SC	- 366
A	- 253
B	- 178
C	- 193
D	- 98
SP	- 0





- Observe que o caminho SC-A-B-BH que não é o de menor custo!!!!
- Seu custo é 450!
- *GBF* **nem sempre** encontra soluções ótimas, ou seja, de menor custo (não é admissível).

Tabela c/  
distância  
em linha  
reta:

SC	- 366
A	- 253
B	- 178
C	- 193
D	- 98
SP	- 0

## Busca A\*

*GBF* minimiza  $h(n)$ , mas não é completo nem admissível



Custo Uniforme minimiza  $c(n)$ . É admissível e completo, mas pode ser ineficiente

$$f(n) = h(n) + c(n)$$

- Busca A\* é uma variação de *Best First* que mistura busca de custo uniforme com o uso de uma heurística admissível.

## Heurística Admissível

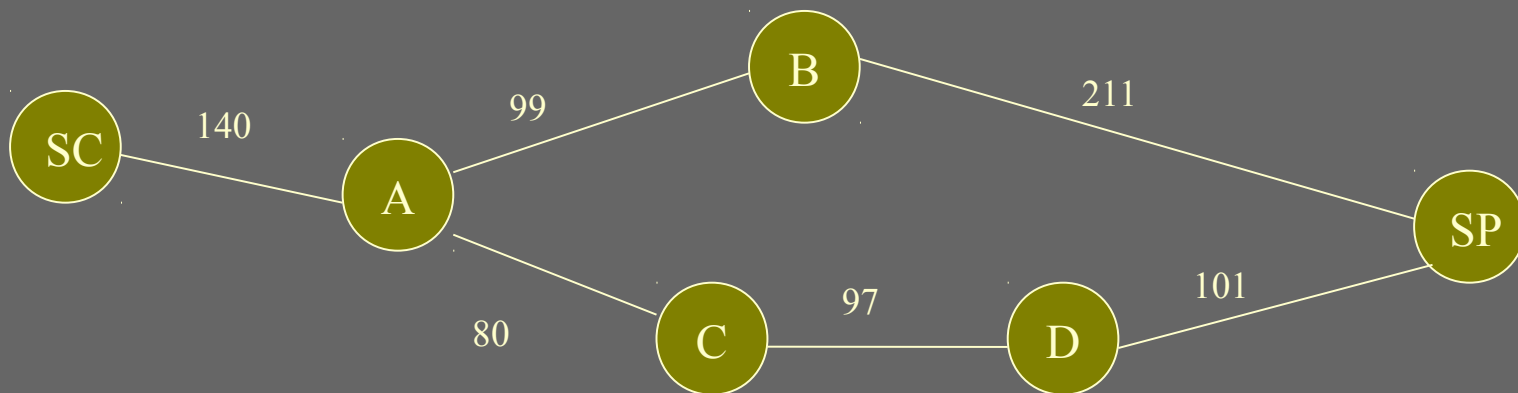
- $h(n)$  é considerada uma heurística admissível se nunca super-estima ou sub-estima o menor custo de se alcançar um objetivo
- Ou seja, os valores da heurística e custo devem influenciar no valor final de  $f(n)$
- Exemplo: funções  $c(n)$  e  $h(n)$  que retornem valores normalizados em um mesmo intervalo

# Algoritmo A\*

1. Inserir em **F** os nós iniciais em **ordem crescente de  $f(n)$**
- 2 Se **F** é vazio
  - 2.1 Então a busca não foi bem sucedida
- 3 Senão seja **n** o primeiro estado de **F**
  - 3.1 Se **n** é um estado meta então
    - 3.1.1 Retornar caminho do estado inicial até **n**
  - 3.2 Senão
    - 3.2.1 Remover **n** de **F** e inserir em **E**
    - 3.2.2 **Adicionar em ordem crescente de  $f(n)$**  todos os sucessores de **n** que não estão em **E** e incrementa o caminho
    - 3.2.3 Voltar ao passo 2

lembrando que  
 $f(n) = c(n) + h(n)$





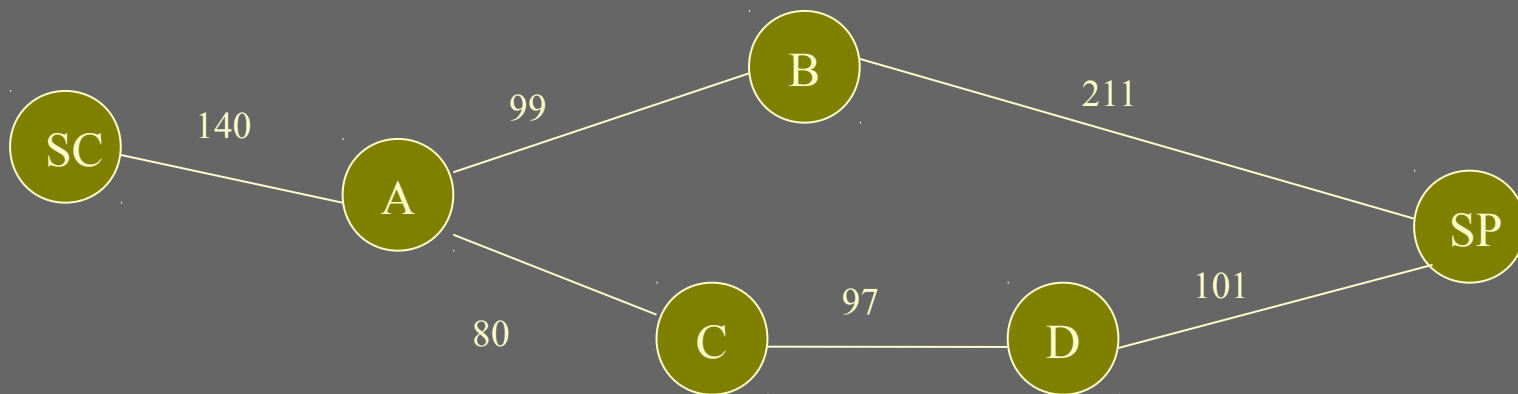
- Qual o caminho do A\* no problema acima (caminho de SC ate SP) ?

$F = \{[SC](366)\}$ ,  $E = \{ \}$

$F = ?$   $E = ?$

Tabela c/  
distância  
em linha  
reta:

SC	- 366
A	- 253
B	- 178
C	- 193
D	- 98
SP	- 0



- Qual o caminho do A\* no problema acima (caminho de SC ate SP) ?

$F = \{[SC](366)\}$ ,  $E = \{ \}$

$F = \{[A,SC](393)\}$ ,  $E = \{SC\}$

$F = ?$   $E = ?$

Tabela c/  
distância  
em linha  
reta:

SC - 366

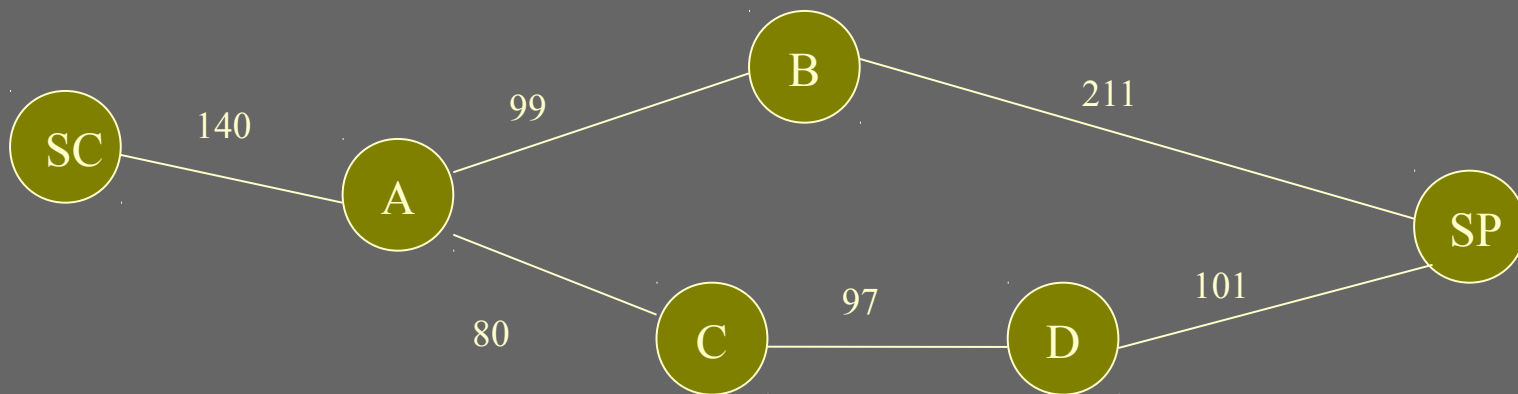
A - 253

B - 178

C - 193

D - 98

SP- 0



- Qual o caminho do A\* no problema acima (caminho de SC ate SP) ?

$F = \{[SC](366)\}$ ,  $E = \{ \}$

$F = \{[A,SC](393)\}$ ,  $E = \{SC\}$

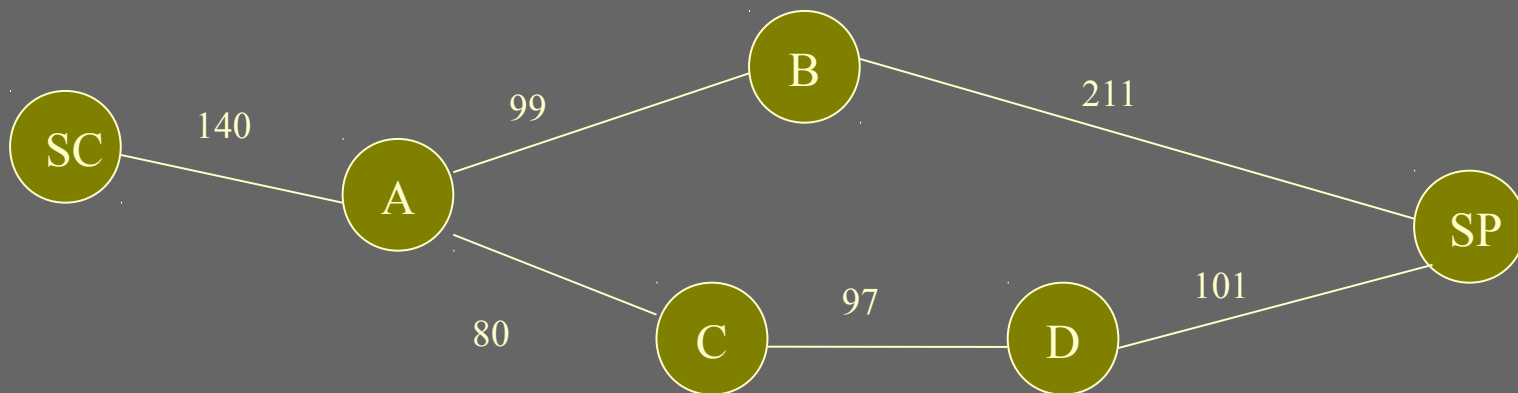
$F = \{[C,A,SC](413), [B,A,SC](417)\}$ ,

$E = \{SC, A\}$

$F = ?$   $E = ?$

Tabela c/  
distância  
em linha  
reta:

SC	- 366
A	- 253
B	- 178
C	- 193
D	- 98
SP	- 0



- Qual o caminho do A\* no problema acima (caminho de SC ate SP) ?

$F = \{[SC](366)\}$ ,  $E = \{ \}$

$F = \{[A,SC](393)\}$ ,  $E = \{SC\}$

$F = \{[C,A,SC](413), [B,A,SC](417)\}$ ,

$E = \{SC, A\}$

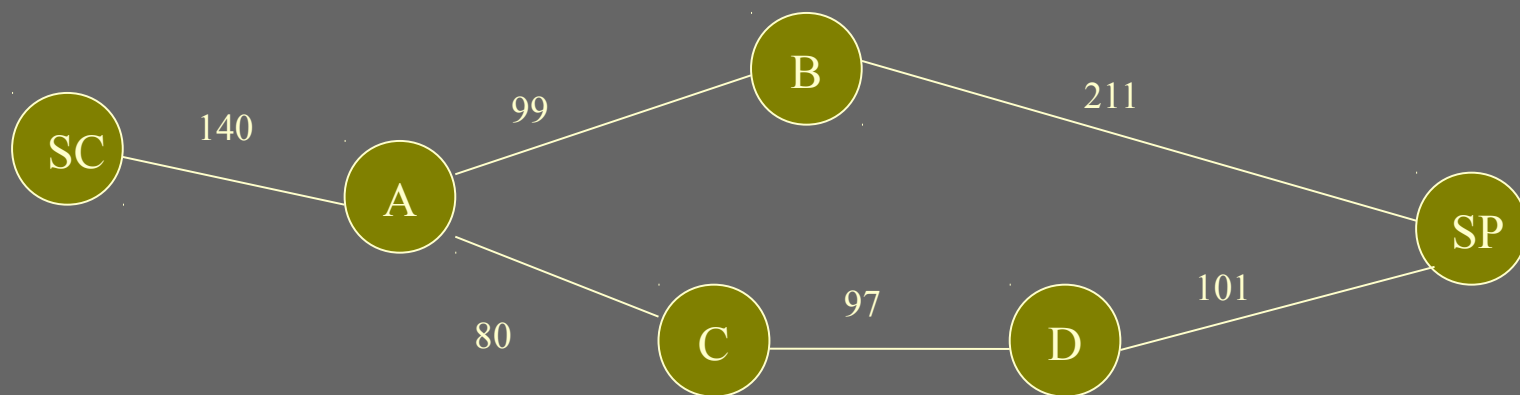
$F = \{[D,C,A,SC](415), [B,A,SC](417)\}$ ,

$E = \{SC, A, C\}$

Tabela c/  
distância  
em linha  
reta:

SC	- 366
A	- 253
B	- 178
C	- 193
D	- 98
SP	- 0



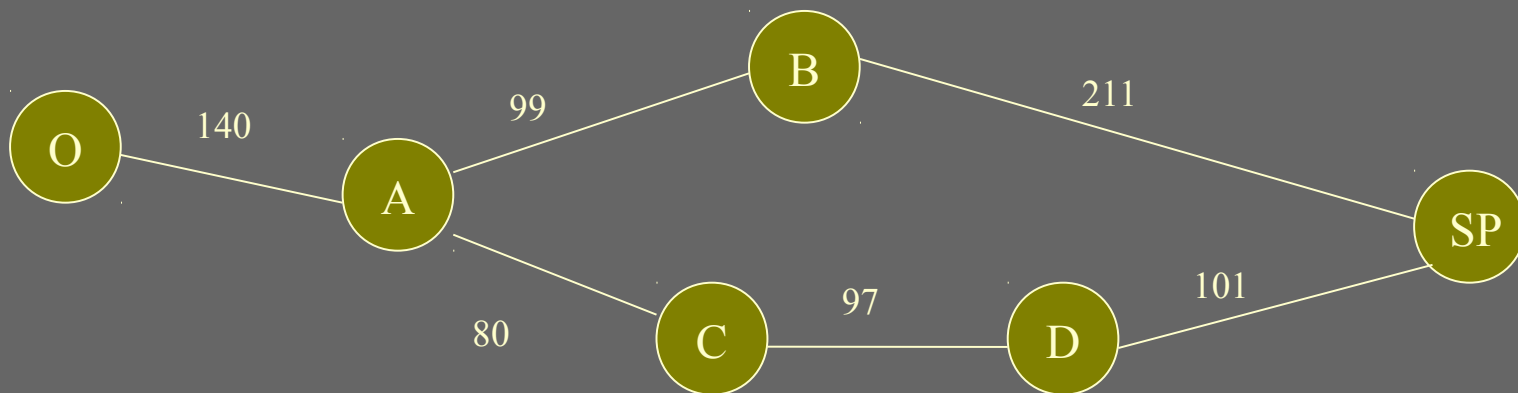


$F = \{[B, A, SC](417), [SP, D, C, A, SC](418)\}$ ,  $E = \{SC, A, C, D\}$

$F = ?$   $E = ?$

Tabela c/  
distância  
em linha  
reta:

SC	- 366
A	- 253
B	- 178
C	- 193
D	- 98
SP	- 0



$F = \{[B, A, O](417), [SP, D, C, A, SC](418)\}$ ,  $E = \{SC, A, C, D\}$

$F = \{[SP, D, C, A, SC](418), [SP, B, A, SC](450)\}$ ,  
 $E = \{SC, A, C, D, B\}$

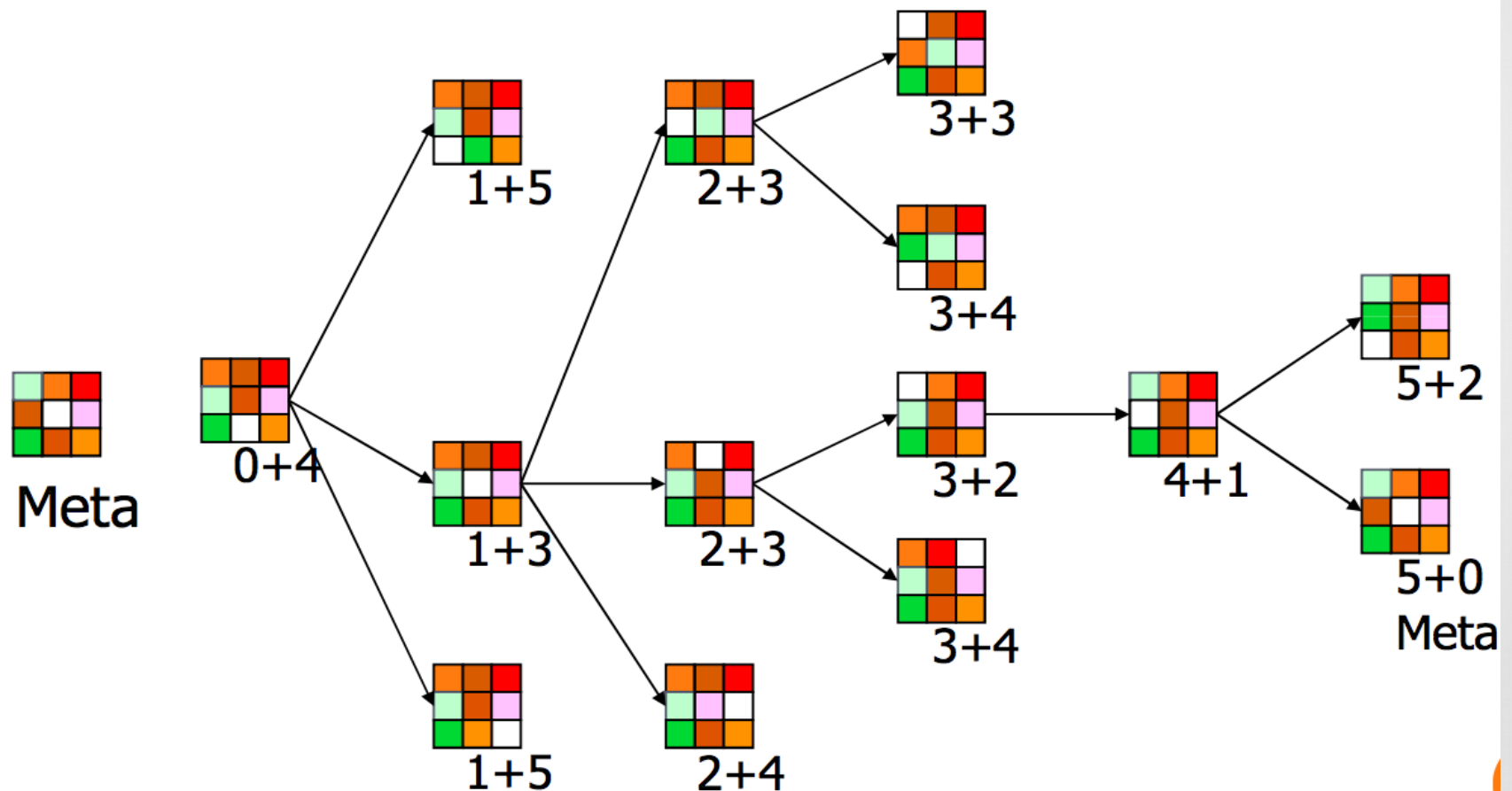
- A busca A\* retorna o caminho SC-A-C-D-SP com função de avaliação de valor 418.

Tabela c/  
distância  
em linha  
reta:

SC	- 366
A	- 253
B	- 178
C	- 193
D	- 98
SP	- 0

# Exemplo: 8-puzzle com A\*

- Custo é dado em movimentos
- Heurística é a semelhança com a meta

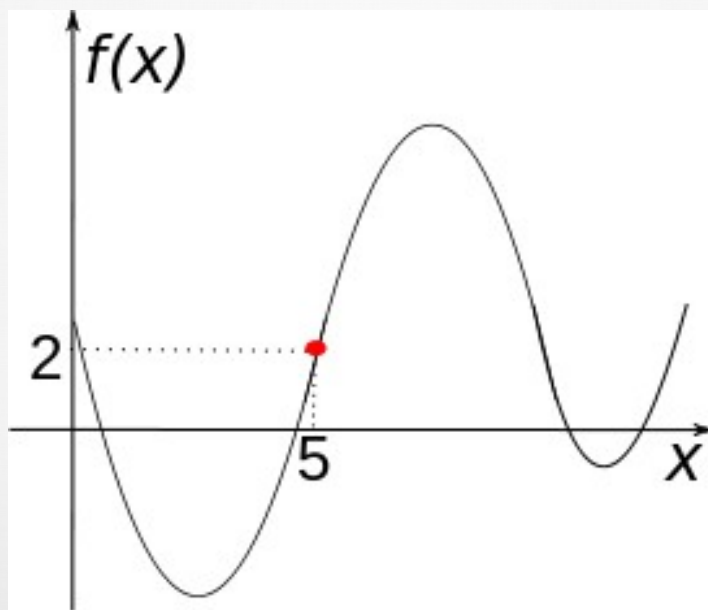


# Busca Local e Otimização

- Algoritmos até agora: exploração sistemática do espaço de busca
  - Caminho para a meta é a solução para o problema
- Nem sempre se conhece a meta
  - Em alguns problemas o objetivo é melhorar
  - Caminho é irrelevante
- Algoritmos de busca local e otimização

# Exemplo

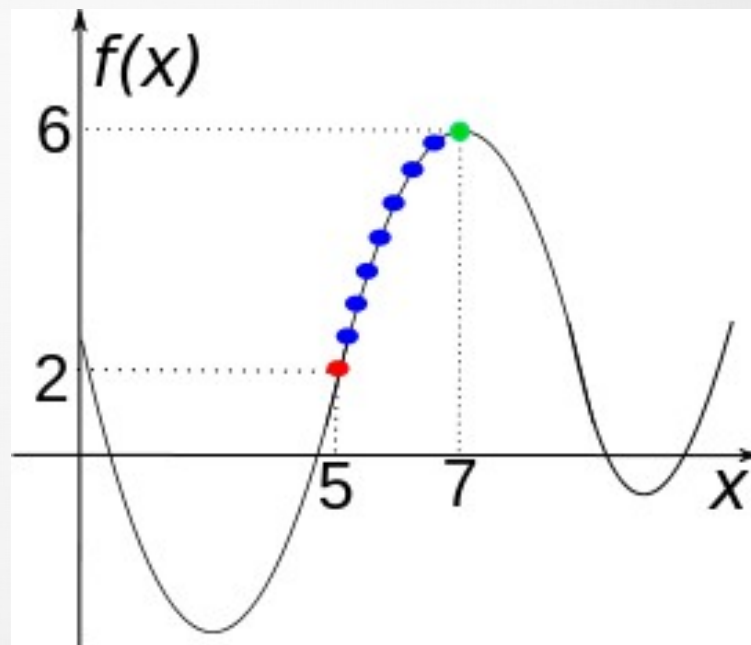
- Por exemplo: o lucro dado pela quantidade  $x$  de um determinado produto, produzido por segundo, é dado pela função  $f(x)$ . Atualmente, a empresa faz 5 desses produtos e fatura 2 reais brutos por segundo.





# Subida de Encosta

- O primeiro algoritmo de otimização que iremos estudar é conhecido como subida de encosta, do inglês *Hill Climbing (HC)*.
- Consiste em mudar sempre para um estado considerado melhor que o atual
- Quando utilizado para minimizar é conhecido como algoritmo de descida de encosta, do inglês *Gradient descent (GD)*.



# Algoritmo Subida de Encosta

1. Para todo estado inicial  $n_0$

1.1. Calcular  $h(n_0)$

2.  $n = \text{argmin}(h(n_0))$

3. objetivo =  $\emptyset$

3. Enquanto objetivo  $\neq n$

3.1 Se  $n$  não possuir sucessor

3.1.1 objetivo =  $n$

3.2 Senão

3.2.1. Para todo  $n_i$  sucessor de  $n$  faça

3.2.1.1. Calcular  $h(n_i)$

3.2.2. Se  $h(n) \leq \min(h(n_i))$

3.2.2.1  $n = \text{argmin}(h(n_i))$

3.2.3. Senão

3.2.3.1 objetivo =  $n$

4. Retorna objetivo

$n$  recebe o nó inicial com menor valor de heurística, ou seja, o de melhor resultado

Se não existe sucessor ou não existe sucessor melhor, então o estado atual é o objetivo

Se existe sucessor com melhor valor de heurística, este se torna o novo estado objetivo

## Caminho com Subida de Encosta

- Se for importante armazenar o caminho até a solução, então é preciso **armazená-lo em uma lista**, como feito com os outros algoritmos de busca
- Contudo, busca de subida de encosta geralmente é utilizado para fazer **buscas locais**, ou seja, procurar por estados melhores próximos do estado inicial
  - Sempre melhores
  - Nunca piora

# Diferença entre *GBF* e *HC*

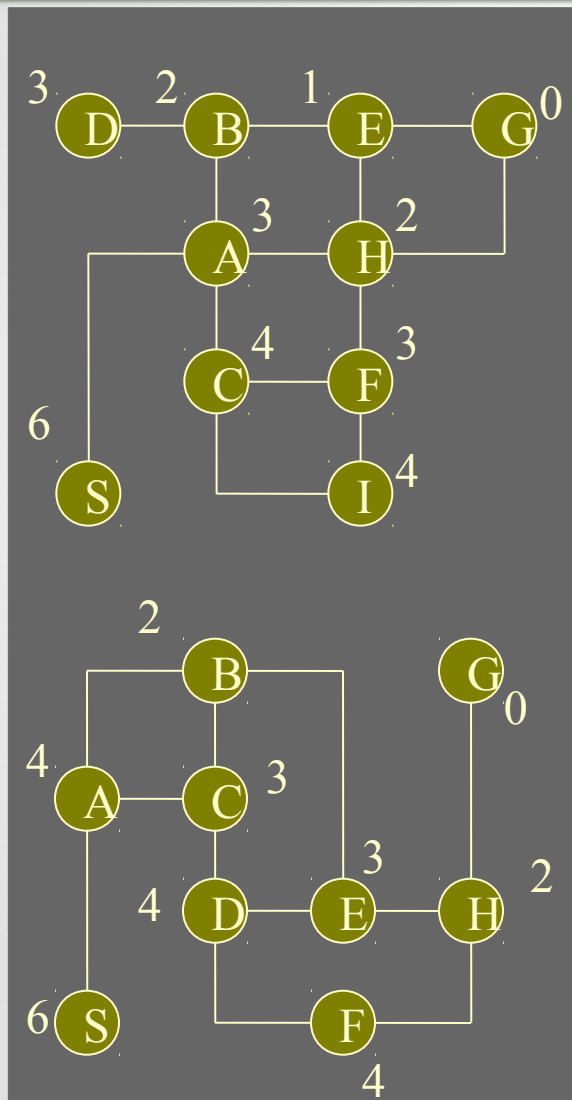


Fig. A

$$\text{Heurística} = |X_g - X_n| + |Y_g - Y_n|$$

Distância em cidade/blocos ou  
Manhattan

GBF tem sucesso???

HC tem sucesso??

Fig. B

$$\text{Heurística} = |X_g - X_n| + |Y_g - Y_n|$$

GBF tem sucesso???

HC tem sucesso??



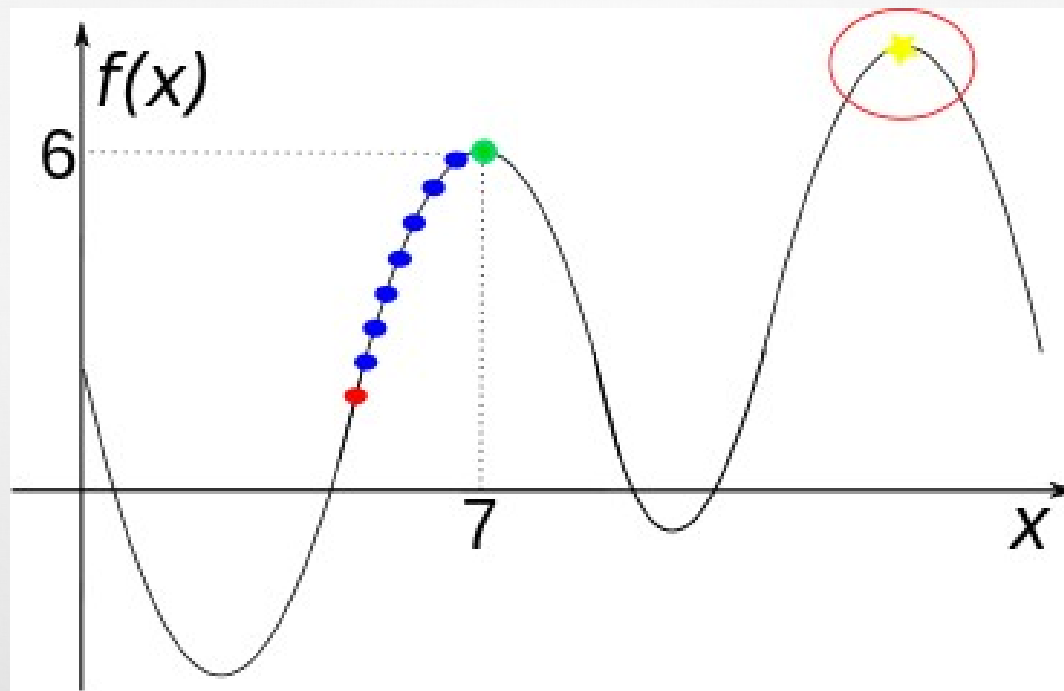
## Diferença entre *GFB* e *HC*

- Fig. A:
  - *GFB* e *HC* tem sucesso.
  - Solução do *GFB* :  $S \rightarrow A^S \rightarrow B^{SA} \rightarrow E^{SAB} \rightarrow G^{SAB}$   
E
  - Solução do *HC*:  $S \rightarrow A^S \rightarrow B^{SA} \rightarrow E^{SAB} \rightarrow G^{SABE}$
- Fig. B:
  - *GFB* tem sucesso; *HC* falha.
  - Solução do *GFB* :  $S \rightarrow A^S \rightarrow B^{SA} \rightarrow E^{SAB} \rightarrow H^{SAB}$   
E  $\rightarrow G^{SABEH}$
  - *HC*:  $S \rightarrow A^S \rightarrow B^{SA}$  (pára)



# Limitação do HC

- A busca pode não conduzir à melhor solução de todas (máximo global), já que os outros caminhos são esquecidos após uma decisão a um estado mais promissor



## Propriedades do *HC*

- Objetivo é atingir o topo da colina
- Se existe um só um máximo local então ele é máximo global e a busca é admissível
- Caso contrário, o algoritmo encontra um máximo local, o que faz dele completo se esse for o objetivo da busca
- Por ser um algoritmo do tipo melhor primeiro, sua complexidade é a mesma!

# Busca Tabu

- Algoritmo capaz de escapar de ótimos locais permitindo movimentos “ruins”
- Ideias:
  - Permitir movimentos que pioram o valor da função de avaliação
  - Proibir, por um dado número de iterações, movimentos que possam retornar a busca a um estado anterior

# Têmpera Simulada

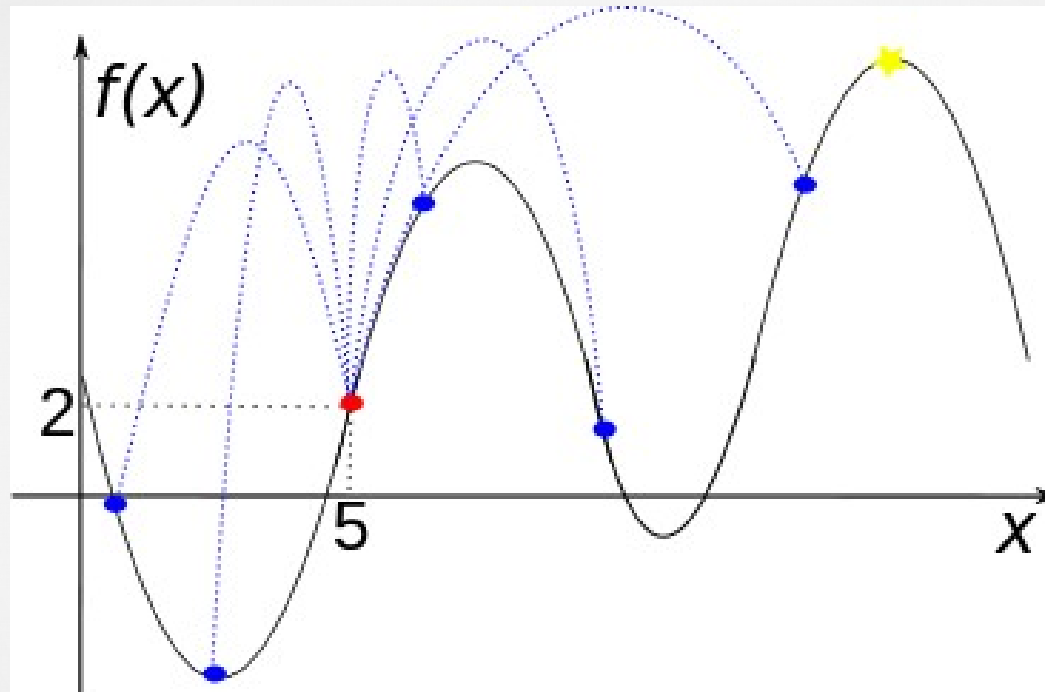
- Chamado de *Simulated Annealing (SA)*
- Inspirado na metalurgia, onde se faz o recozimento para temperar metais e vidro
- Existe uma função  $T$  de temperatura
  - Quanto maior o valor de  $T$ , maior é a amplitude do movimento do estado atual para o próximo
- Utilizado para evitar que a busca fique presa em máximos locais

# Têmpera Simulada

- Ao invés de escolher o melhor movimento, escolhe um movimento. Se o movimento melhorar a situação, ele sempre será aceito. Caso contrário, aceita o movimento com alguma probabilidade menor que 1.
- A probabilidade decresce exponencialmente com a “má qualidade” do movimento
- A probabilidade também decresce a medida que a “temperatura” se reduz.
- Movimentos ruins têm maior probabilidade no início, e se tornam mais improváveis conforme a “temperatura” diminui

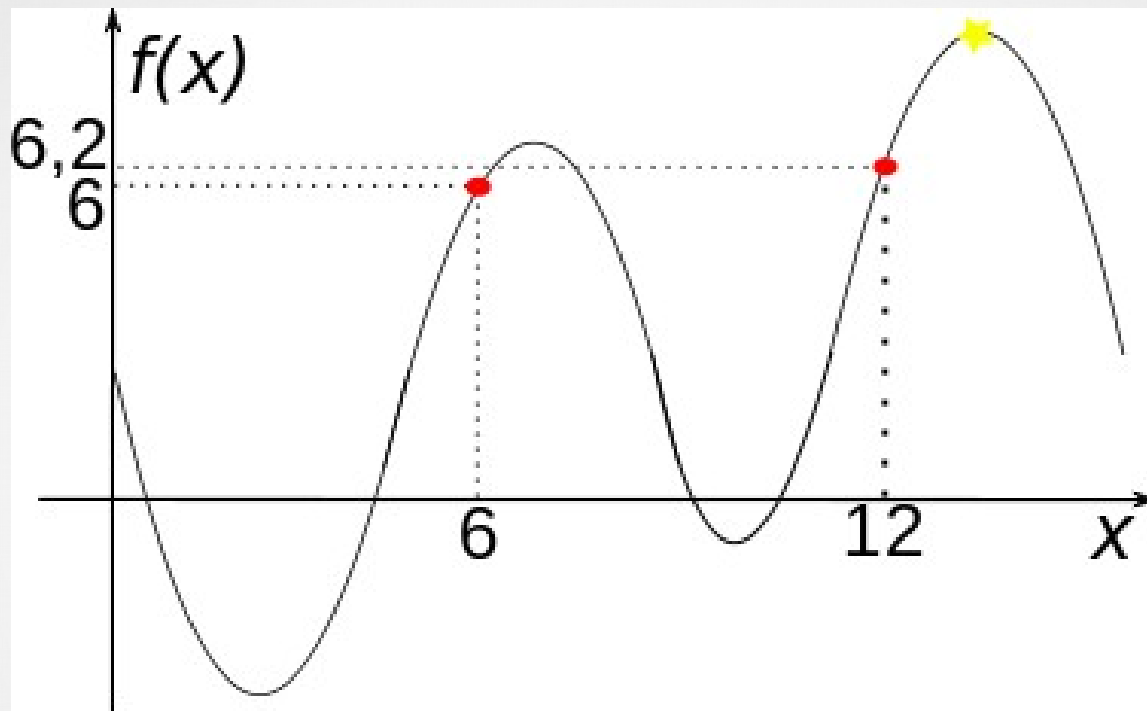


## Exemplo ilustrativo



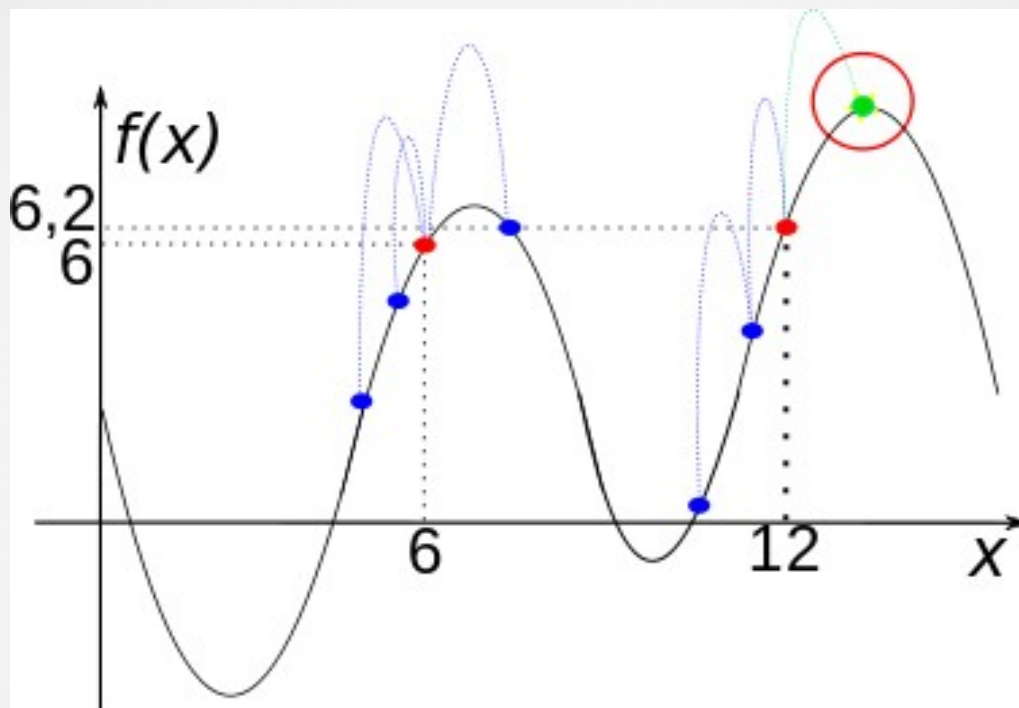
- No início a estratégia possui valor de  $T$  alto, o que faz com que ele busque em uma área maior do espaço de soluções

## Exemplo ilustrativo



- Os melhores estados são selecionados, segundo a função de avaliação escolhida
- Esses pontos servirão de pontos iniciais da próxima etapa do algoritmo

## Exemplo ilustrativo



- O valor de  $T$  é reduzido e o espaço buscado se torna menor
- Eventualmente, o máximo global ou soluções próximas serão encontradas

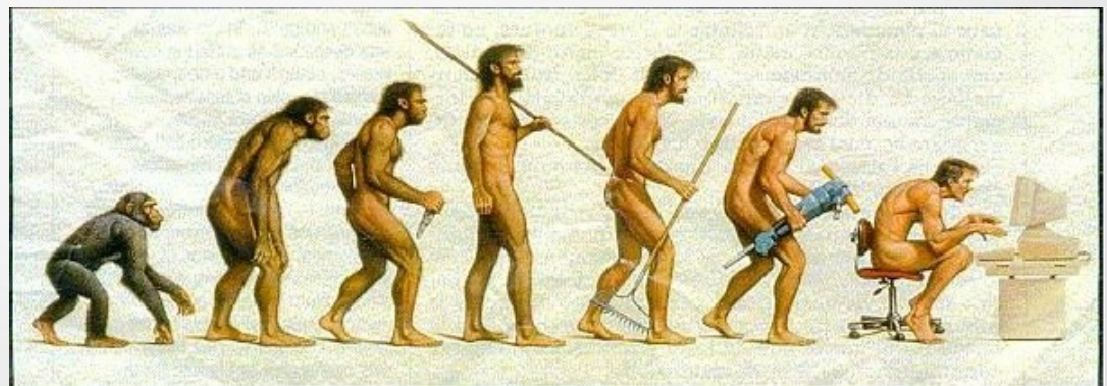
# Propriedades SA

- Fornece um meio de escapar do máximo local, o que pode fazer dele uma estratégia:
  - Completa, e até
  - Admissível
- Contudo, para que isso aconteça é preciso que os valores do parâmetro  $T$  sejam bem escolhidos e haja uma quantidade **suficientemente grande de iterações** para o algoritmo encontrar o(s) objetivo(s)



# Busca Evolutiva

- Também aplicados em problemas de busca por soluções melhores.
- Inspirado na “Teoria da Evolução” de Charles Darwin.
- Na natureza todos os indivíduos dentro de um ecossistema competem entre si por recursos.



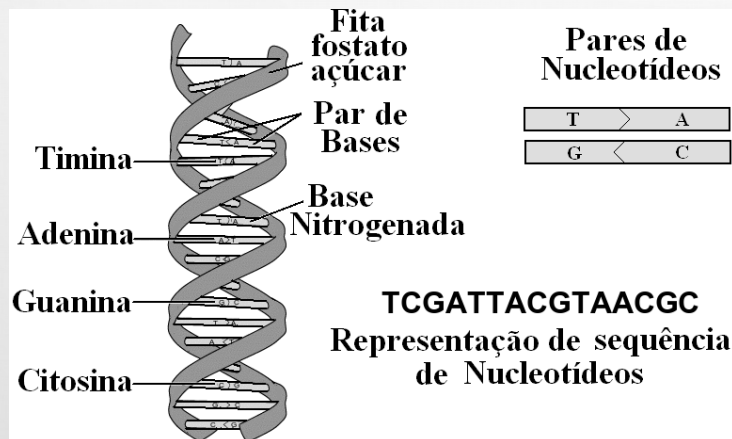


# Busca Evolutiva

- Indivíduos de uma espécie pouco aptos possuem menor chance de gerar prole
  - Essa descendência reduzida faz com que suas características possuam uma menor probabilidade de serem propagadas
  - O contrário ocorre com os indivíduos mais aptos!
- Uma solução é representada como um indivíduo
  - Um conjunto de soluções é uma população
- A ideia é aplicar a seleção natural como processo de busca por soluções melhores!

# Indivíduo

- Na natureza, as características dos indivíduos são codificadas em genes
- Um conjunto específico de genes é chamado de **genótipo**.
- O genótipo é a base do **fenótipo**, que é a expressão das características físicas e mentais codificadas pelos genes.



# Indivíduo

- Genótipo é composto por um **cromossomo**, ou seja, vetores de números:
  - Binários
  - Inteiros
  - Reais
- Fenótipo é a forma que o genótipo é aplicado para solucionar o problema

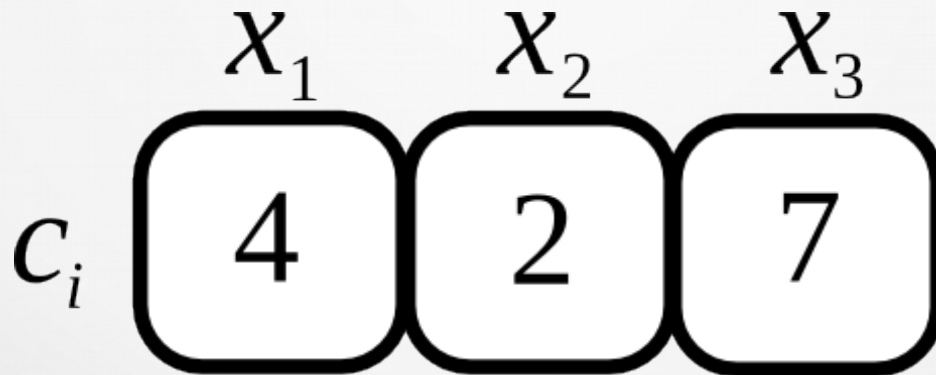
# Exemplo

- Considere o problema de produção dado anteriormente (*só que um pouco mais desafiador*).
- O lucro dado pelas quantidades  $x_1$ ,  $x_2$  e  $x_3$  de três determinados produtos, construídos por segundo, é dado por uma função  $f(\cdot)$  em centenas de reais.
- Exemplo:
  - $i = \{x_1 = 4, x_2 = 2, x_3 = 7\}$ ,  $f(i) = 0,05$
  - ou seja, se produzo 4 unidades do produto 1, 2 do produto 2 e 7 do produto 3, tenho 5 reais de lucro



# Genótipo

- Se  $i$  é uma possível solução do problema, podemos codificar essa solução em um genótipo, ou seja, em um cromossomo.
- Por exemplo, uma sequência de números inteiros:

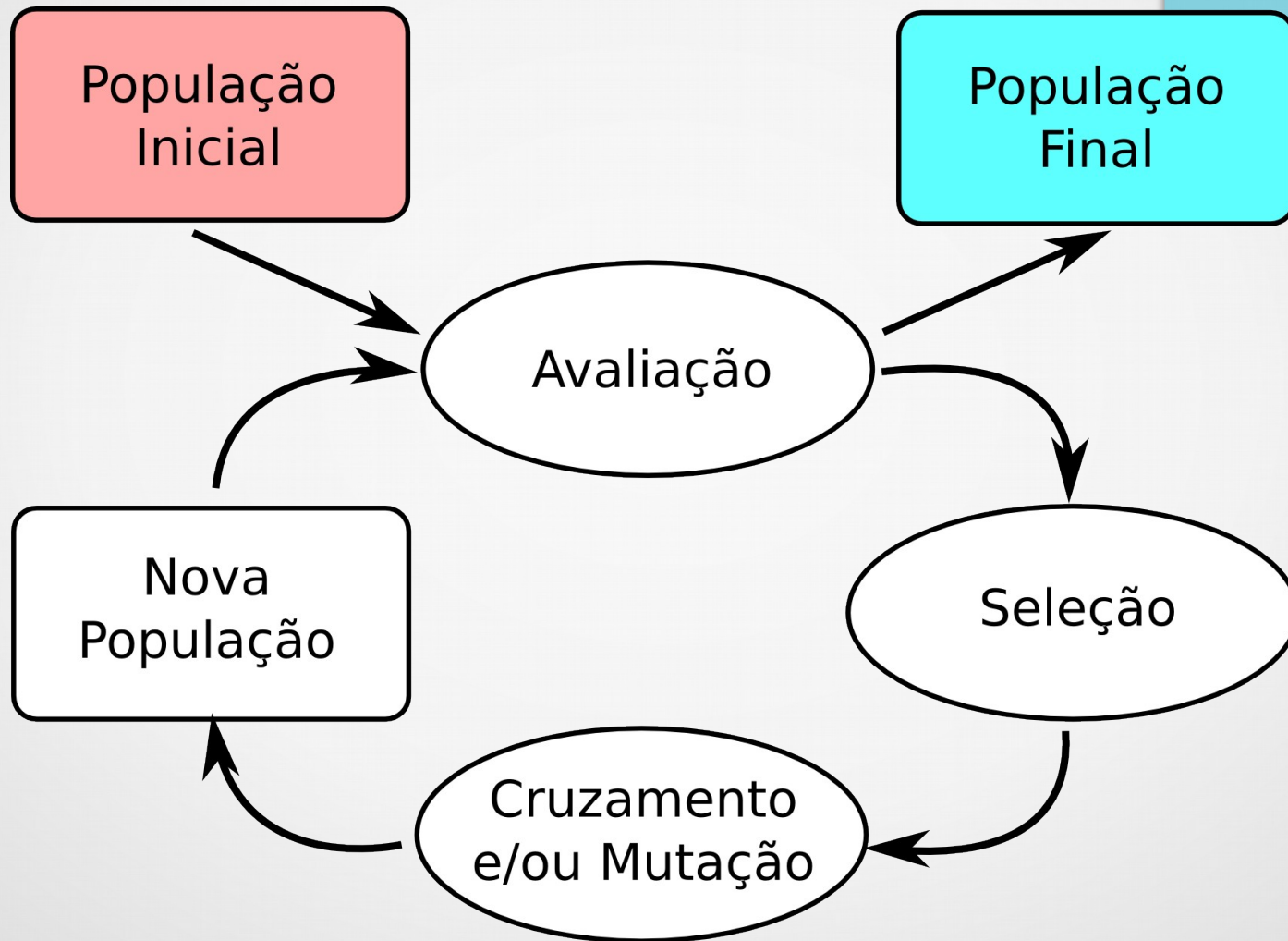




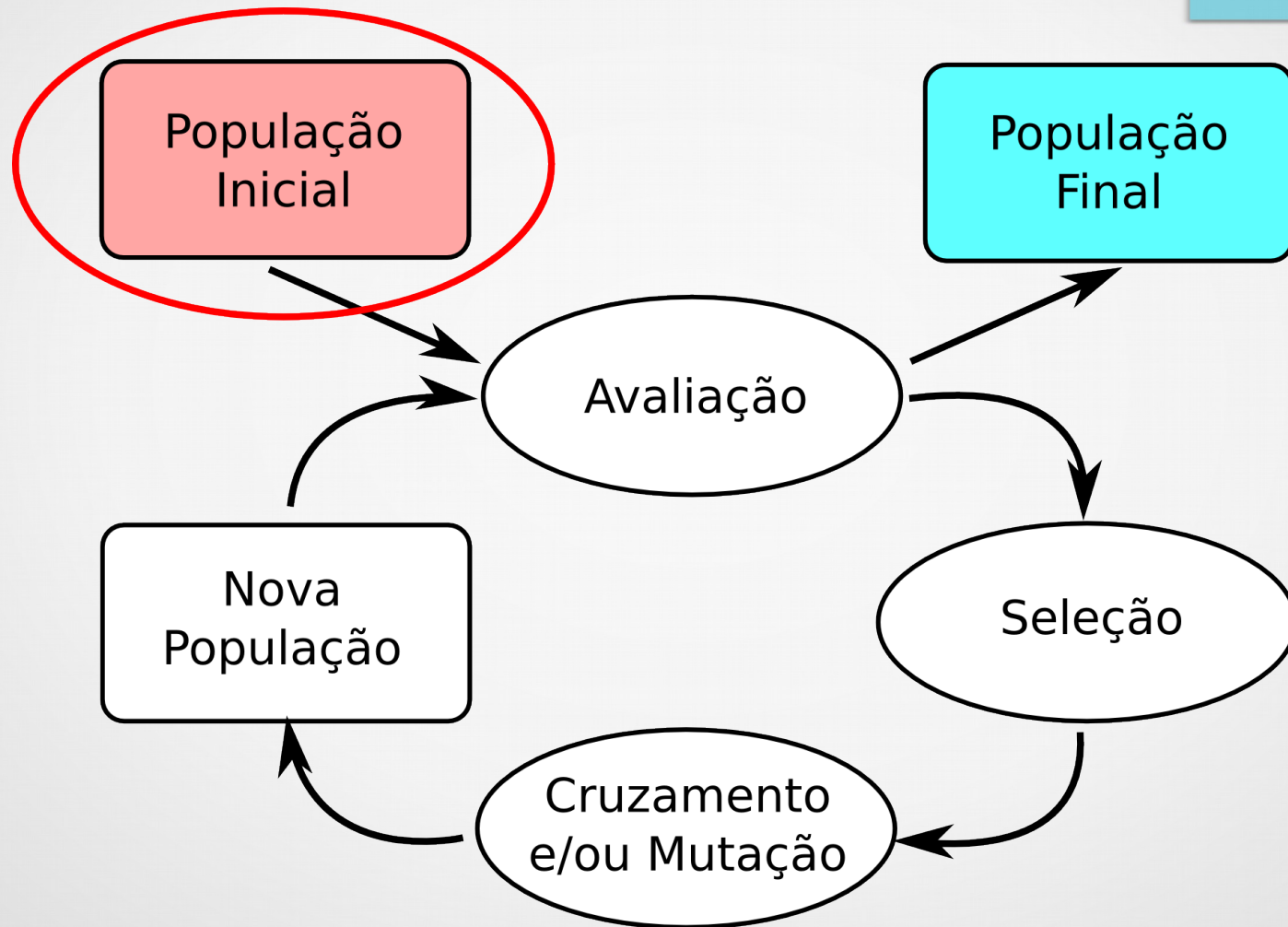
# Fenótipo e aptidão

- O fenótipo está relacionado com o resultado que a solução  $i$ , representada pelo genótipo  $c_i$ , consegue obter.
- Funções são utilizadas para medir o resultado.
  - Quanto melhor o resultado, mais apta está a solução.
- Por esse motivo, chamamos a função utilizada para avaliar um indivíduo como **função de aptidão**.
- Exemplo
  - O indivíduo  $i = \{x_1 = 4, x_2 = 2, x_3 = 7\}$  gera  $f(i) = 0,05$  de lucro e, portanto, sua aptidão é **0,05**.

# Visão geral dos algoritmos evolutivos



# População Inicial



# População

- Para aplicar seleção natural é preciso que haja uma população, ou seja, um conjunto de indivíduos
  - Cada indivíduo representa uma solução
- É importante que haja diversidade entre os indivíduos, para que a busca ocorra em diferentes locais do espaço de soluções (estados).
- Alguns exemplos de indivíduos da população inicial são:
  - Soluções de potencial
  - Soluções conhecidas para o problema
  - Soluções aleatórias



# Exemplo

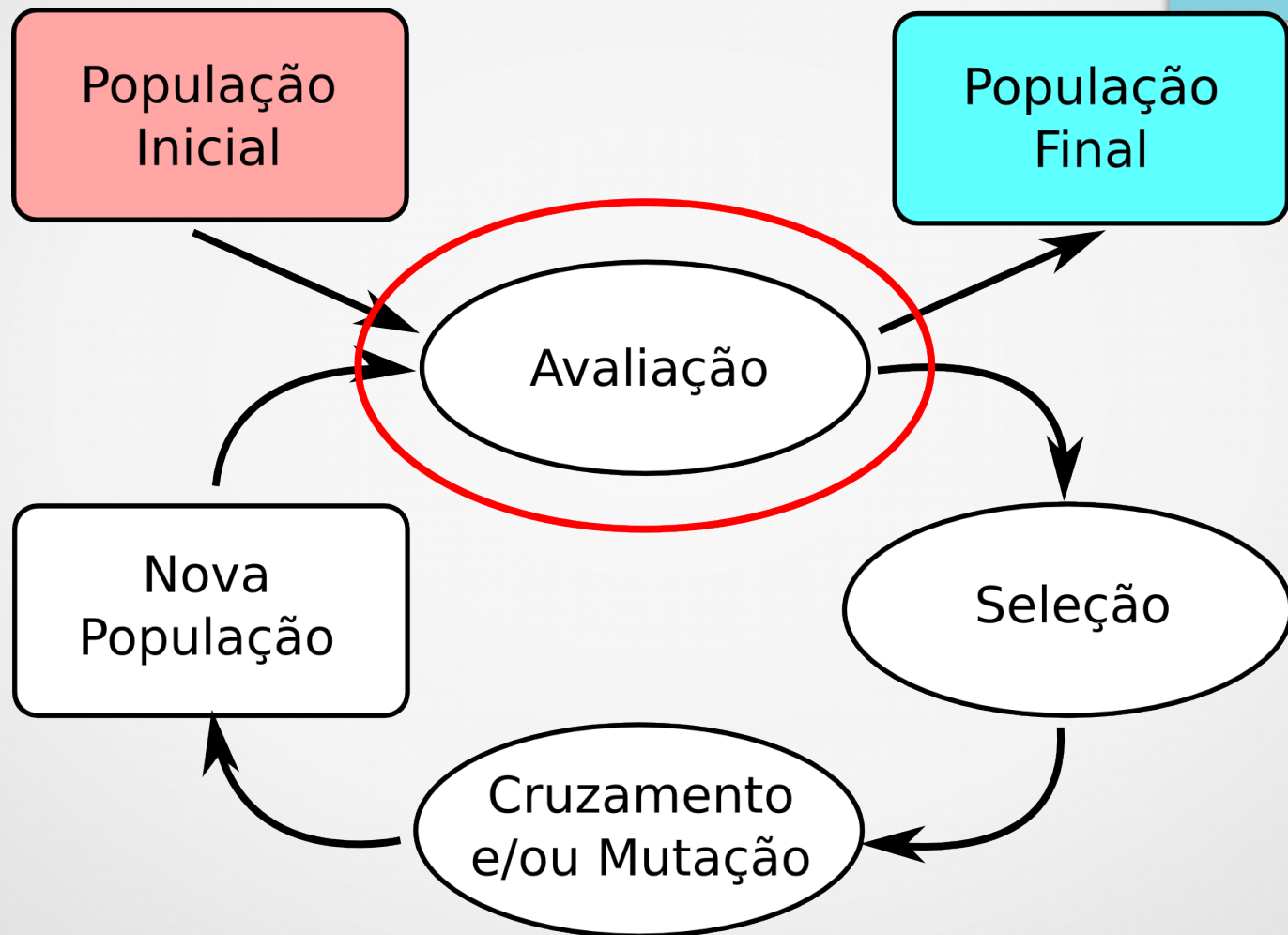
- Considere os três estados a seguir, de forma que cada um seja uma possível solução para o problema de produção.

$c_1$	4	10	8
$c_2$	5	3	12
$c_3$	8	10	9
	$x_1$	$x_2$	$x_3$

- Escolhidos aleatoriamente, essa será nossa população inicial para o problema do lucro



# Avaliação



# Avaliação

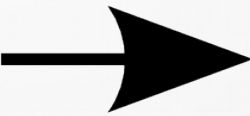
- Função aptidão é utilizada para avaliar os indivíduos
  - Usada para definir o impacto do mesmo
- Exemplo:
  - Suponhamos que o lucro da empresa é calculado pela fórmula:

$$f(i) = \frac{x_1^2 - 2x_1x_2 + x_2x_3}{x_3^3 + x_1}$$

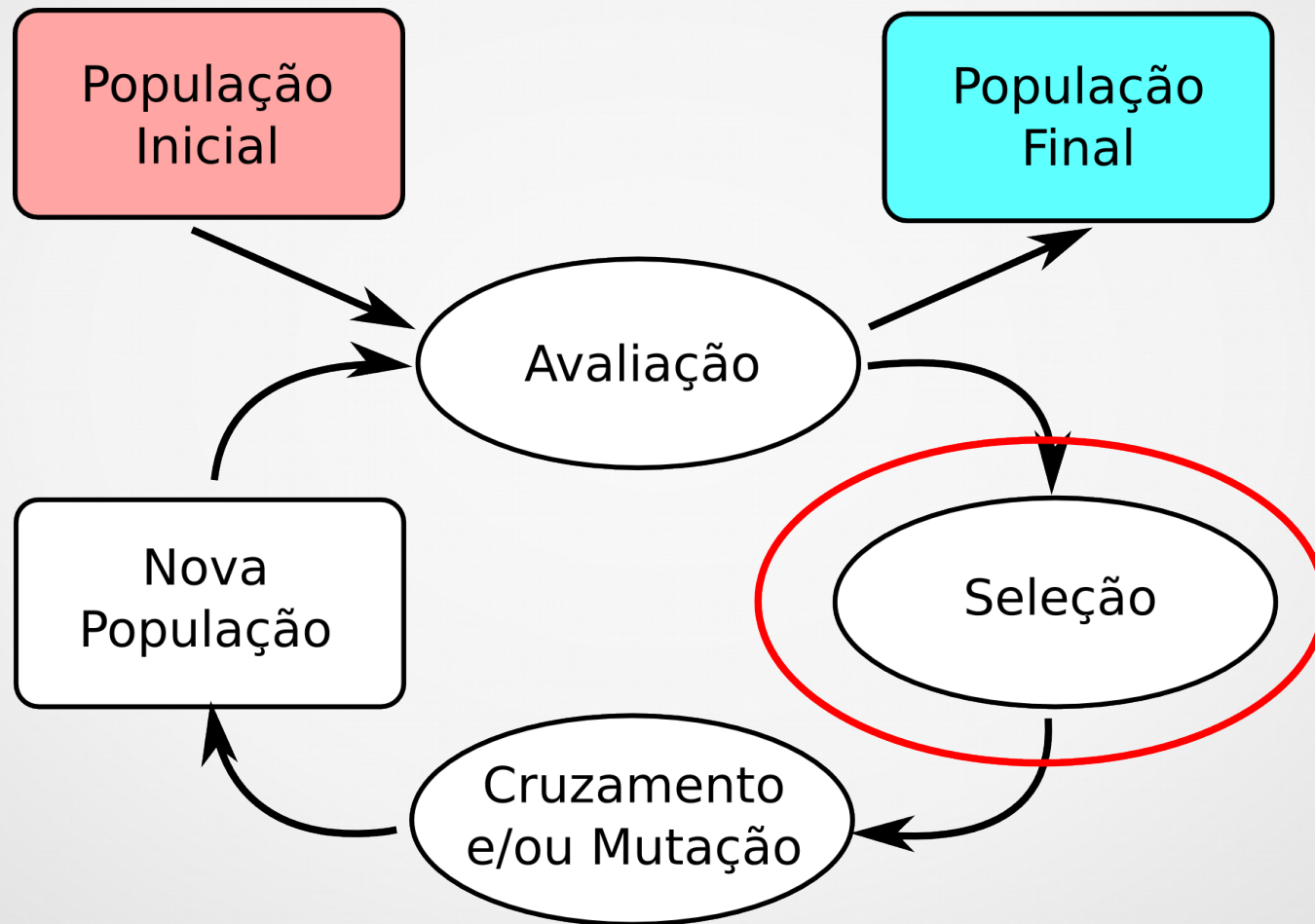
- $f(i)$  é a aptidão do indivíduo  $i$

# Exemplo

- Aplicando a função de aptidão na população a seguir, temos:

$c_1$	4	10	8	$f(c_i)$	0,0310
$c_2$	5	3	12		0,0179
$c_3$	8	10	9		-0,0081
	$x_1$	$x_2$	$x_3$		

# Seleção



# Seleção

- Para gerar uma nova população, é preciso selecionar indivíduos baseados em sua aptidão
  - Os indivíduos mais aptos devem possuir maior probabilidade de seleção.
- Diversos métodos:
  - Seleção proporcional
  - Seleção determinística
  - Outras



# Seleção Proporcional

- Este tipo de seleção é proporcional a aptidão dos indivíduos da população
- A probabilidade de seleção de um indivíduo de uma população de tamanho  $|P|$  é igual à:

$$p_i = \frac{f(c_i)}{\sum_{j=1}^{|P|} f(c_j)}$$



## Exemplo

- Voltemos para as aptidões dos cromossomos  $f(c_1)=0,0310$ ,  $f(c_2)=0,0179$  e  $f(c_3)=-0,0081$ 
  - Não podemos ter probabilidades negativas!
  - Normalizamos os valores do intervalo  $[-0,0081, 0,0310]$  para o intervalo  $[1,10]$

antes		depois
$f(c_1) = 0,0310$	normalização →	10
$f(c_2) = 0,0179$		6,98
$f(c_3) = -0,0081$		1

# Exemplo

- Agora podemos calcular as probabilidades de cada indivíduo ser selecionado
  - Semelhante a uma roleta

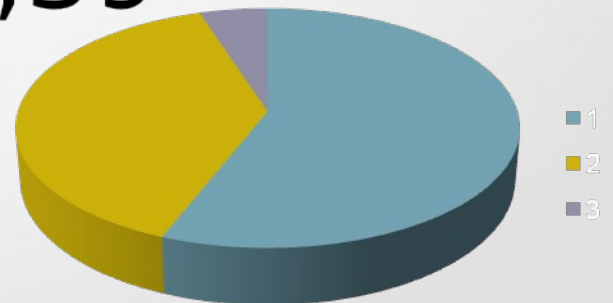
$$T = f(c_1) + f(c_2) + f(c_3)$$

$$T = 10 + 6,98 + 1 = 17,98$$

$$p_1 = 10/17,98 = 0,56$$

$$p_2 = 6,98/17,98 = 0,39$$

$$p_3 = 1/17,98 = 0,05$$



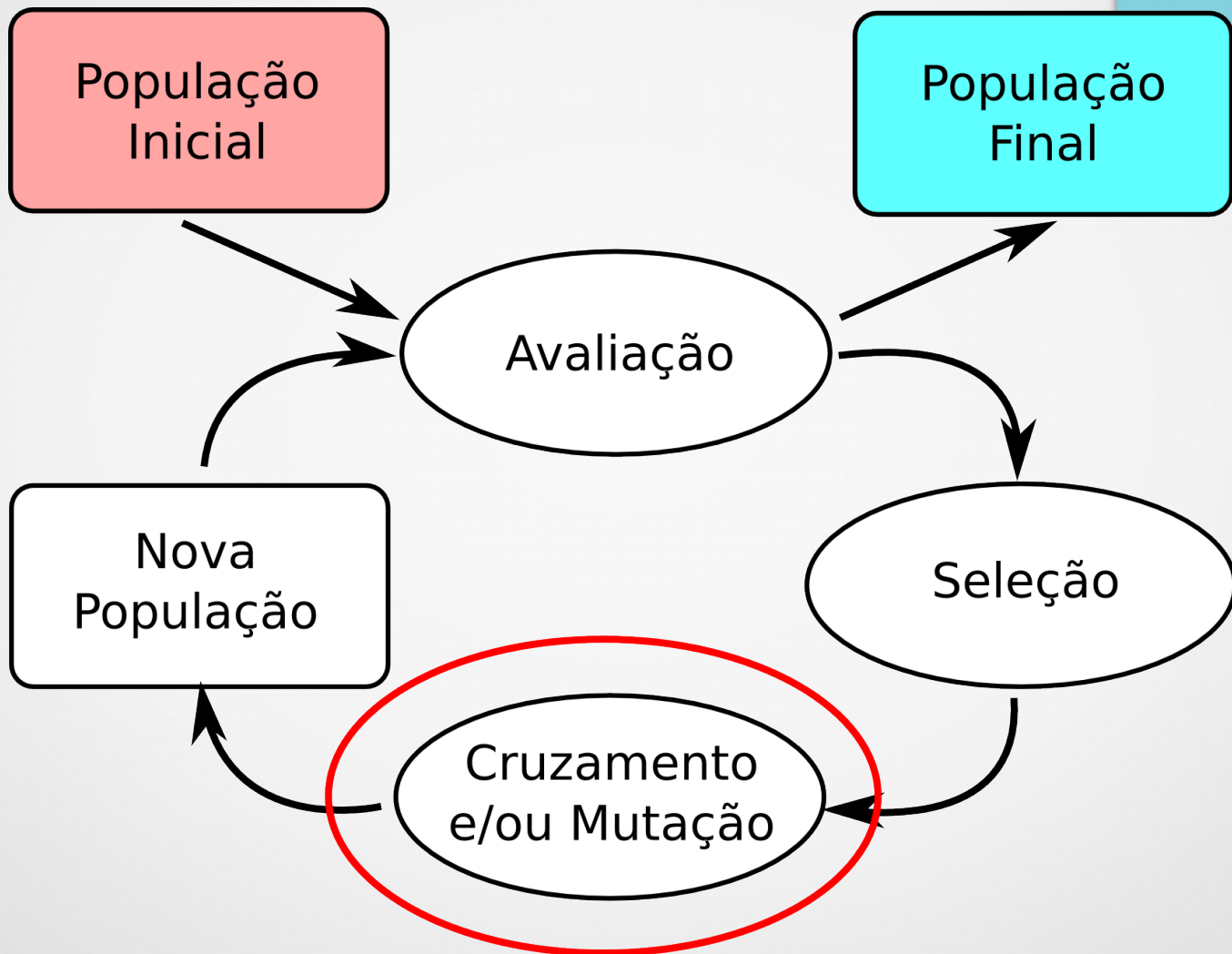


# Seleção Determinística

- Os indivíduos são sorteados aleatoriamente e comparados entre si
- Em seguida, é selecionado o indivíduo que possui maior aptidão dentre os dois comparados
- O processo é repetido



# Operadores Genéticos



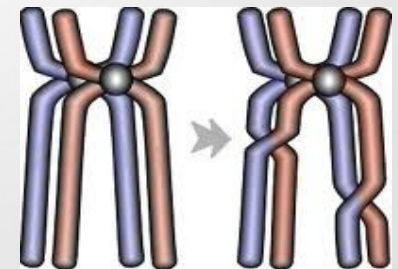


# Operadores Genéticos

- Responsáveis pela modificação dos cromossomos ao longo das gerações
  - Executam a busca de novas soluções
- Podem ser guiados ou não
  - Utilizam algum tipo de heurística
- Vamos estudar dois tipos:
  - Cruzamento ou recombinação
  - Mutação

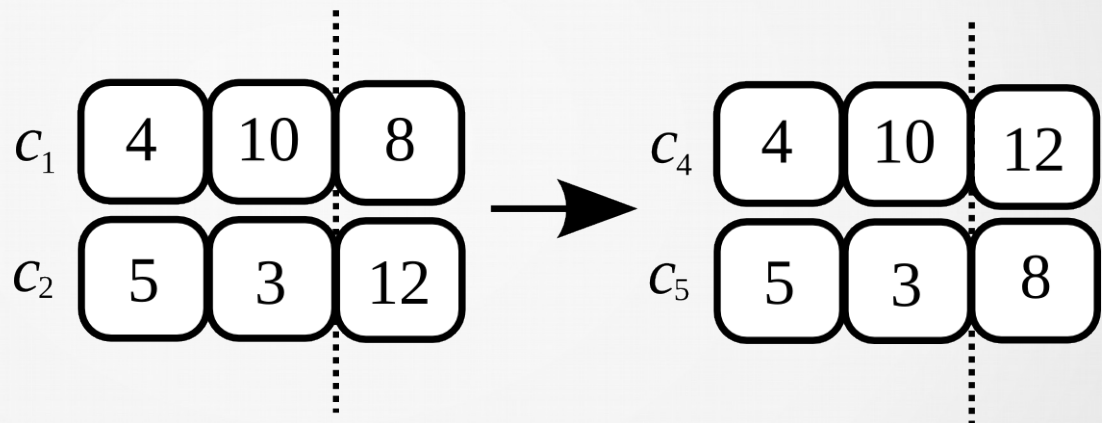
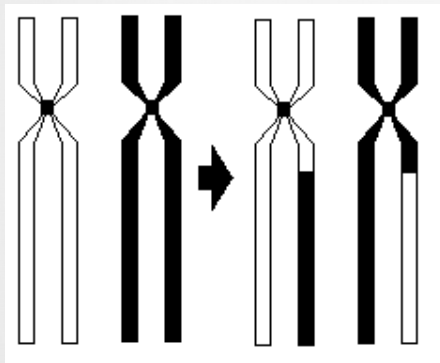
# Cruzamento

- Os indivíduos selecionados são cruzados dois a dois
- Os indivíduos selecionados são chamados de progenitores
- O cruzamento combina características dos indivíduos selecionados para gerar novos indivíduos
- Os indivíduos gerados são chamados de descendentes



# Cruzamento simples

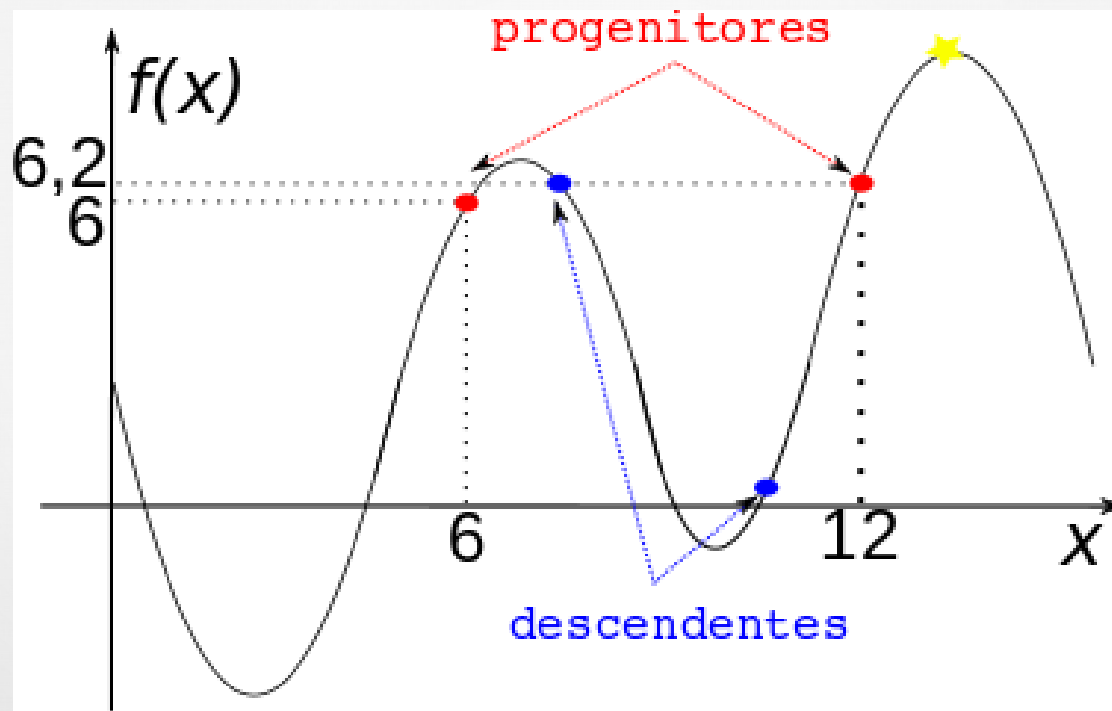
- O cruzamento mais simples consiste em trocar os cromossomos de um par de progenitores em um determinado ponto



- Em um determinado ponto (aleatório ou não) os cromossomos de  $c_1$  e  $c_2$  são trocados para dar origem a  $c_4$  e  $c_5$

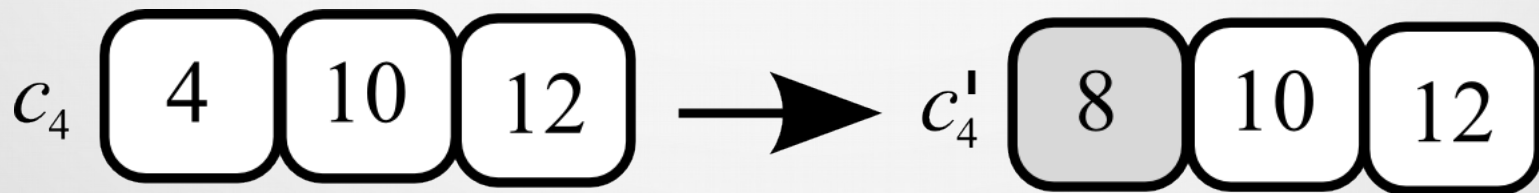
# Cruzamento e busca

- Cruzar soluções geram novas soluções que possuem características dos progenitores



# Mutação

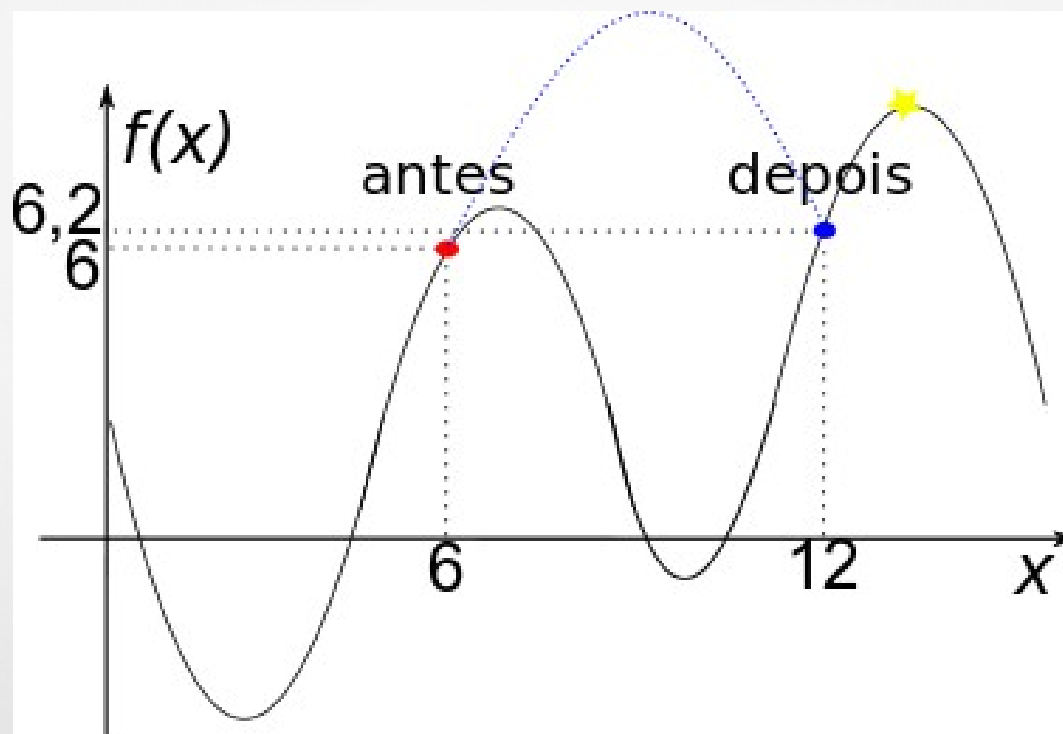
- Modifica parte do cromossomo (aleatoriamente ou por heurística), para gerar uma nova característica no indivíduo que não foi recebida de seus progenitores
  - Objetiva gerar soluções inéditas!
- A taxa de mutação é o parâmetro que define a probabilidade de um cromossomo sofrer mutação
- Exemplo:





# Mutação e busca

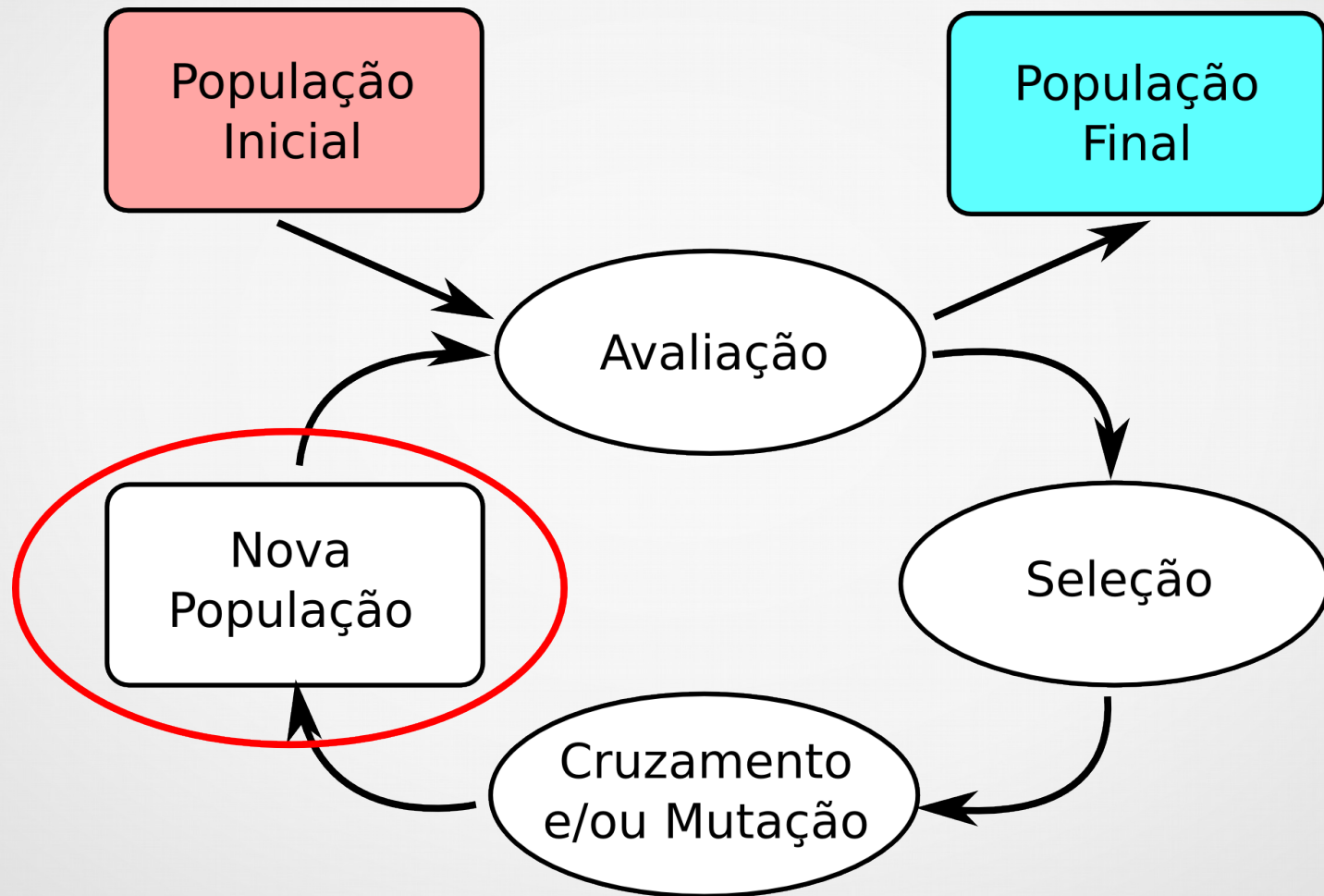
- Aplicar mutação é equivalente a fazer a busca evolutiva “saltar” no espaço do soluções



# Elitismo

- Os operadores de cruzamento e mutação alteram os indivíduos de uma população para outra
- Essa alteração nem sempre melhora a população
- Por isso, é interessante manter um conjunto com o(s) melhor(es) indivíduo(s) de uma população para outra
- Essa prática é chamada elitismo

# Nova População



## Nova População

- Depois que os indivíduos da população anterior são selecionados e os operadores evolutivos são aplicados, uma nova geração é formada
- A partir da nova população, todo o processo é aplicado novamente
- Cada ciclo completo do AE é chamado de **geração**.
- O processo é repetido, criando novas gerações.

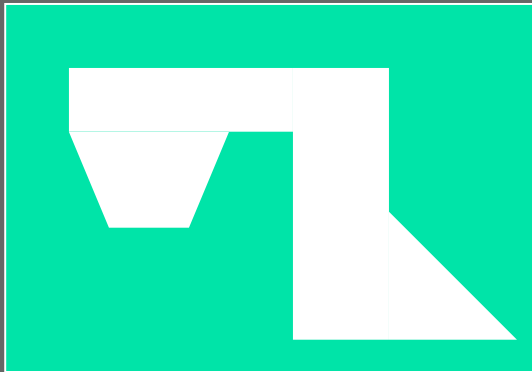
## Critérios de parada

- O algoritmo finaliza quando algum critério de parada é atingido
- Exemplos:
  - Convergência da população
  - Número máximo de gerações
  - Limite de gerações sem melhora

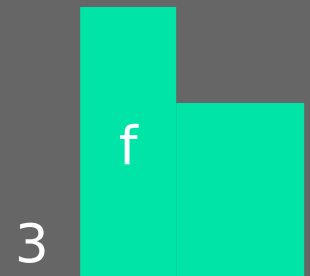
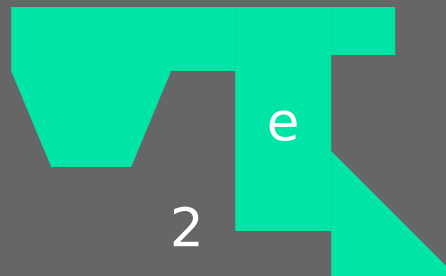
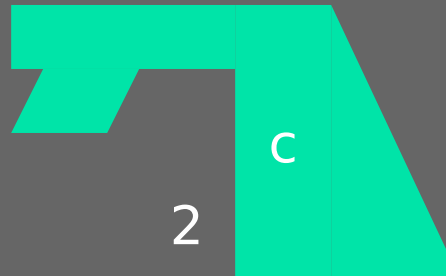
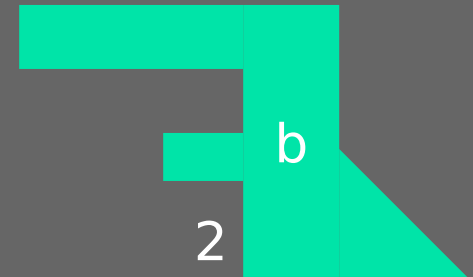
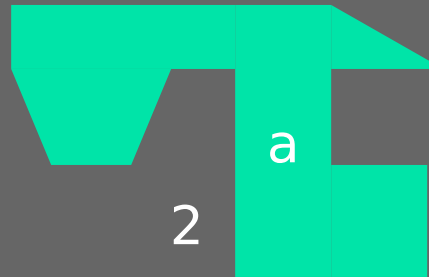


## Exemplo 2: Problema Combinatorial

Encontrar a chave para a fechadura:

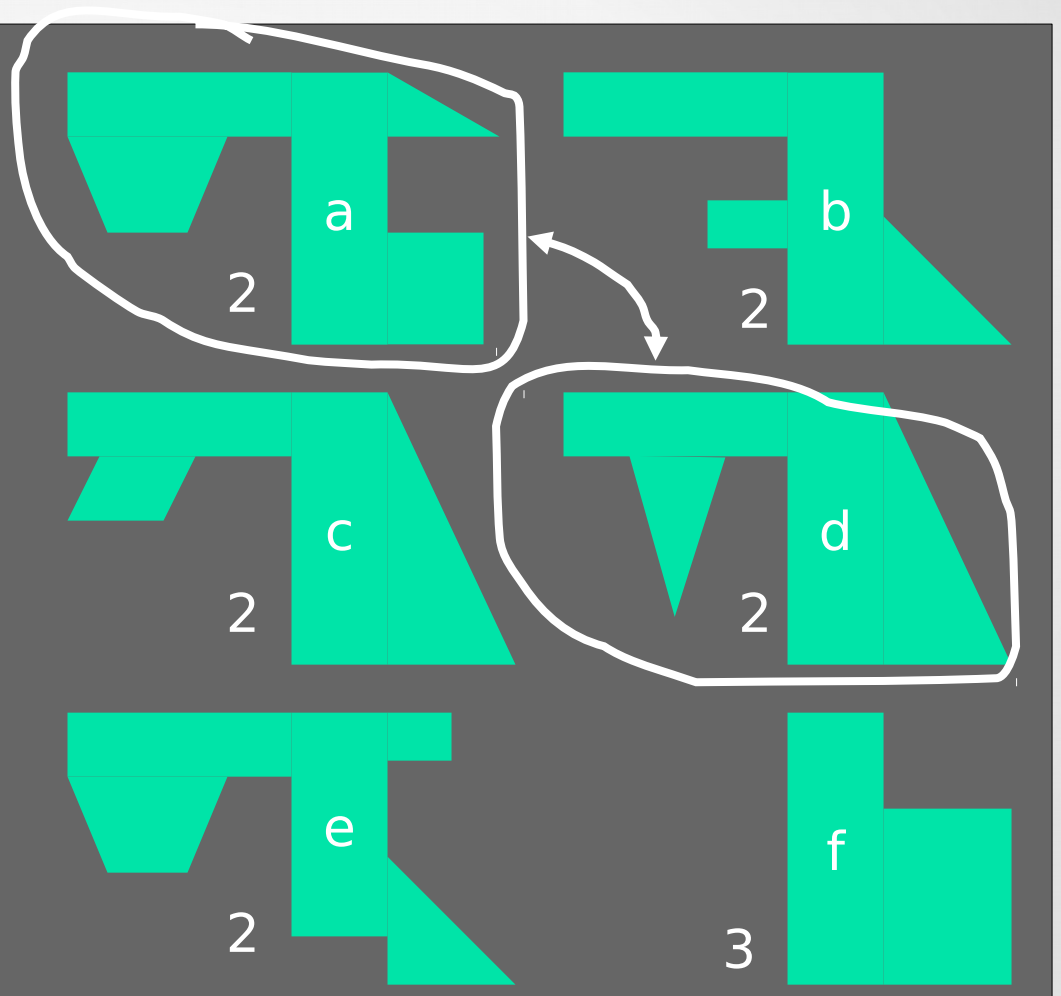
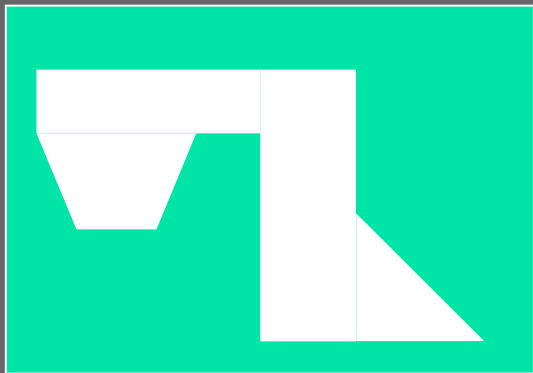


Quanto menor o valor de Aptidão, melhor a chave



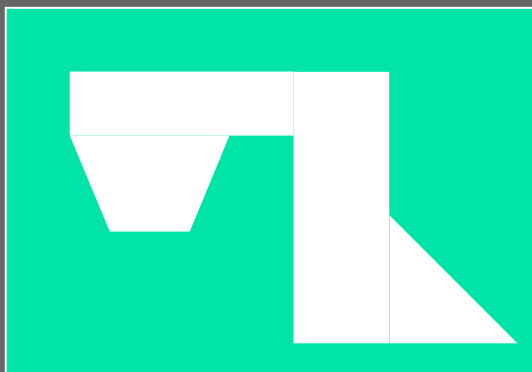
## Seleção de a e d

Encontrar a chave  
para o chaveiro:

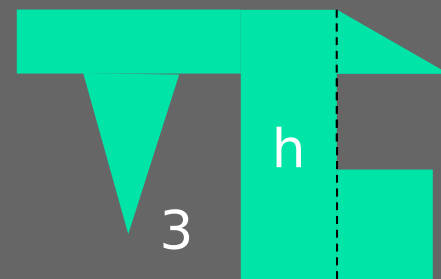
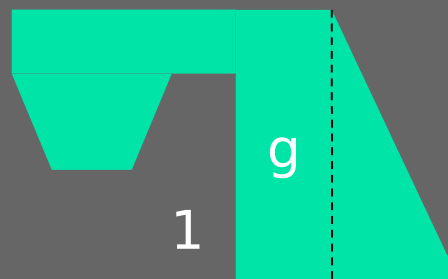
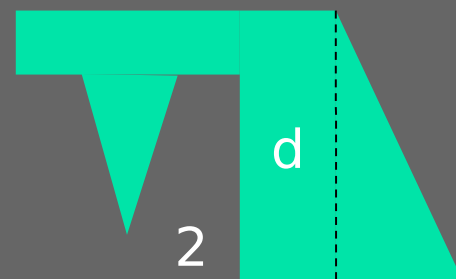
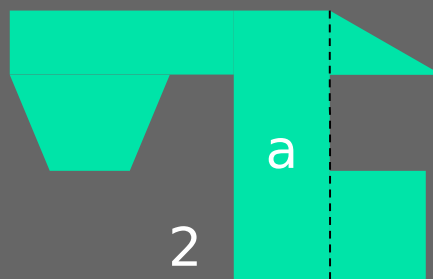


# Cruzamento de a e d

Encontrar a chave  
para o chaveiro:



## Cruzamento

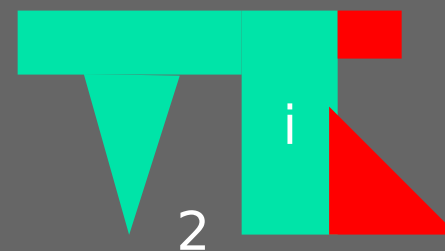
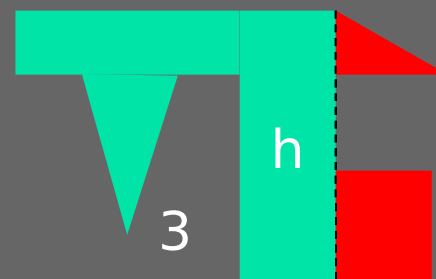


# Mutação de h

Encontrar a chave  
para o chaveiro:

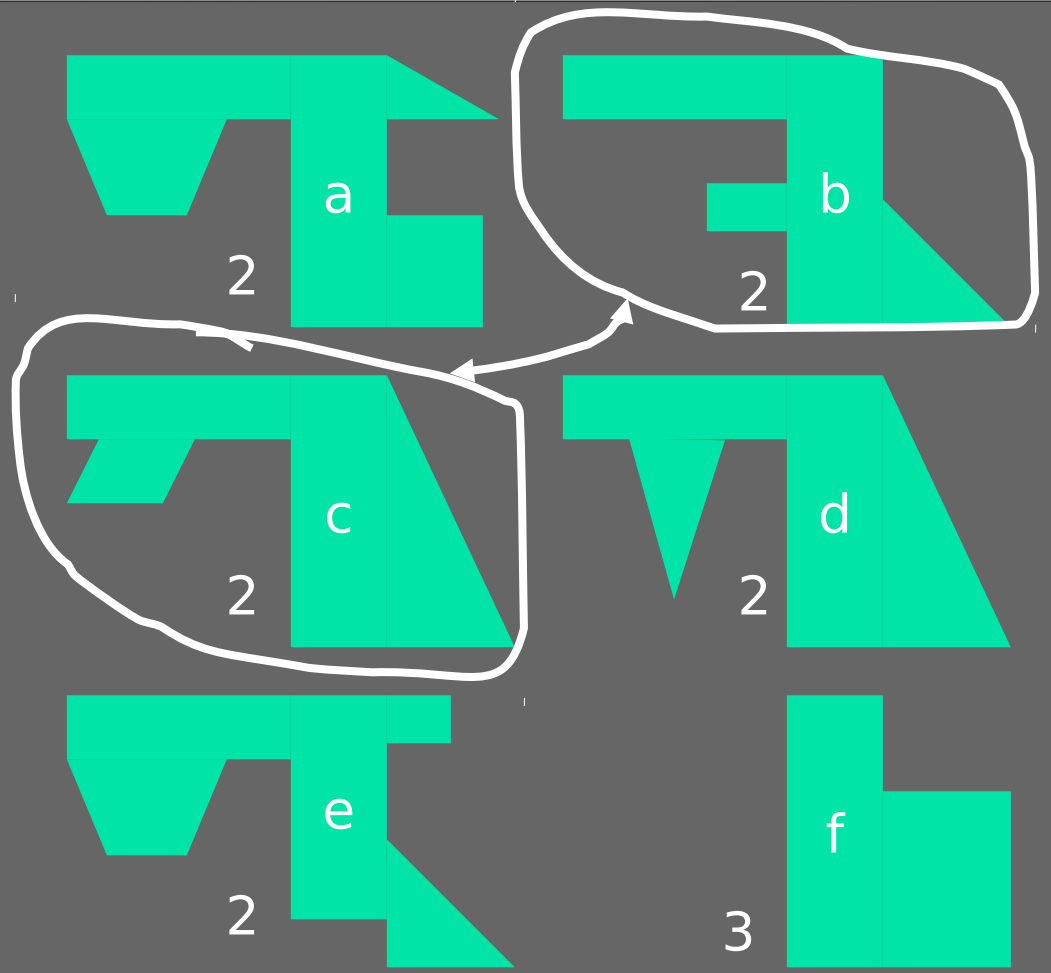
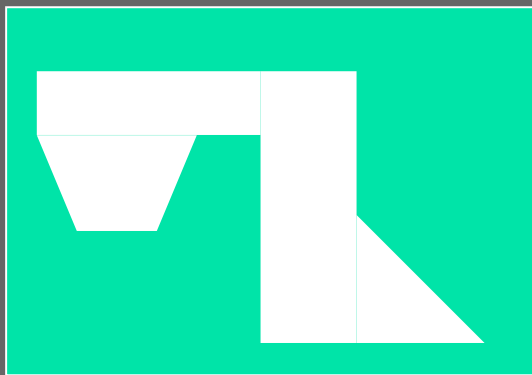


Mutação



## Seleção de c e b

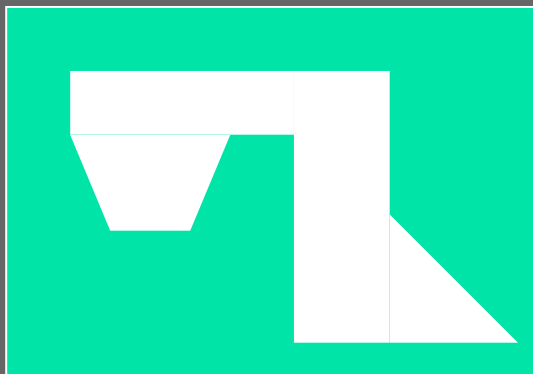
Encontrar a chave  
para o chaveiro:



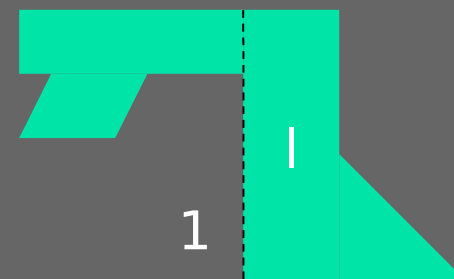
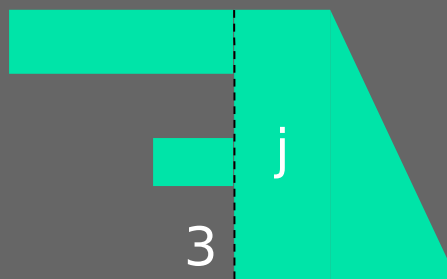
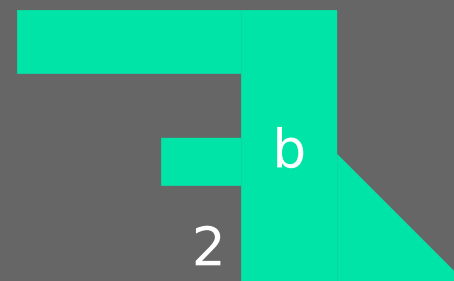
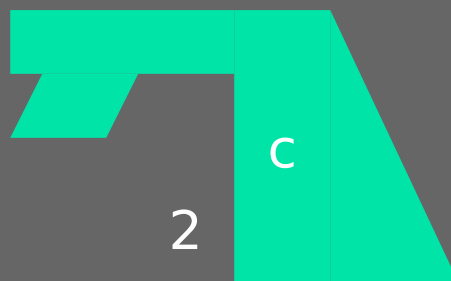


# Cruzamento de c e b

Encontrar a chave  
para o chaveiro:

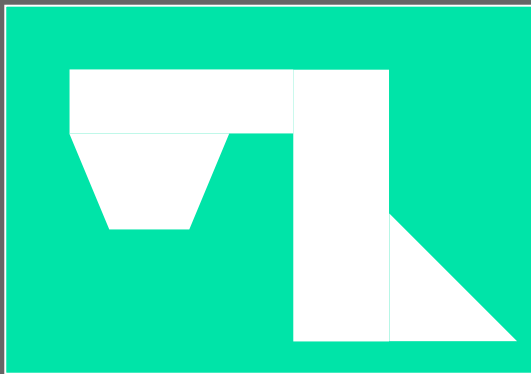


Cruzamento

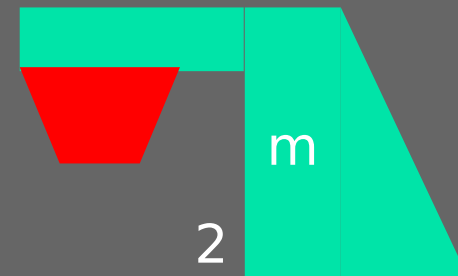
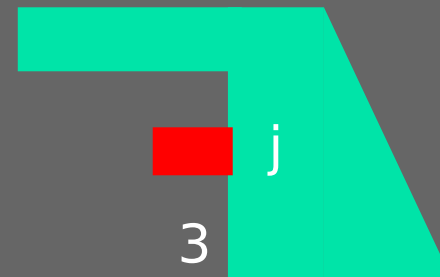


# Mutação de j

Encontrar a chave  
para o chaveiro:

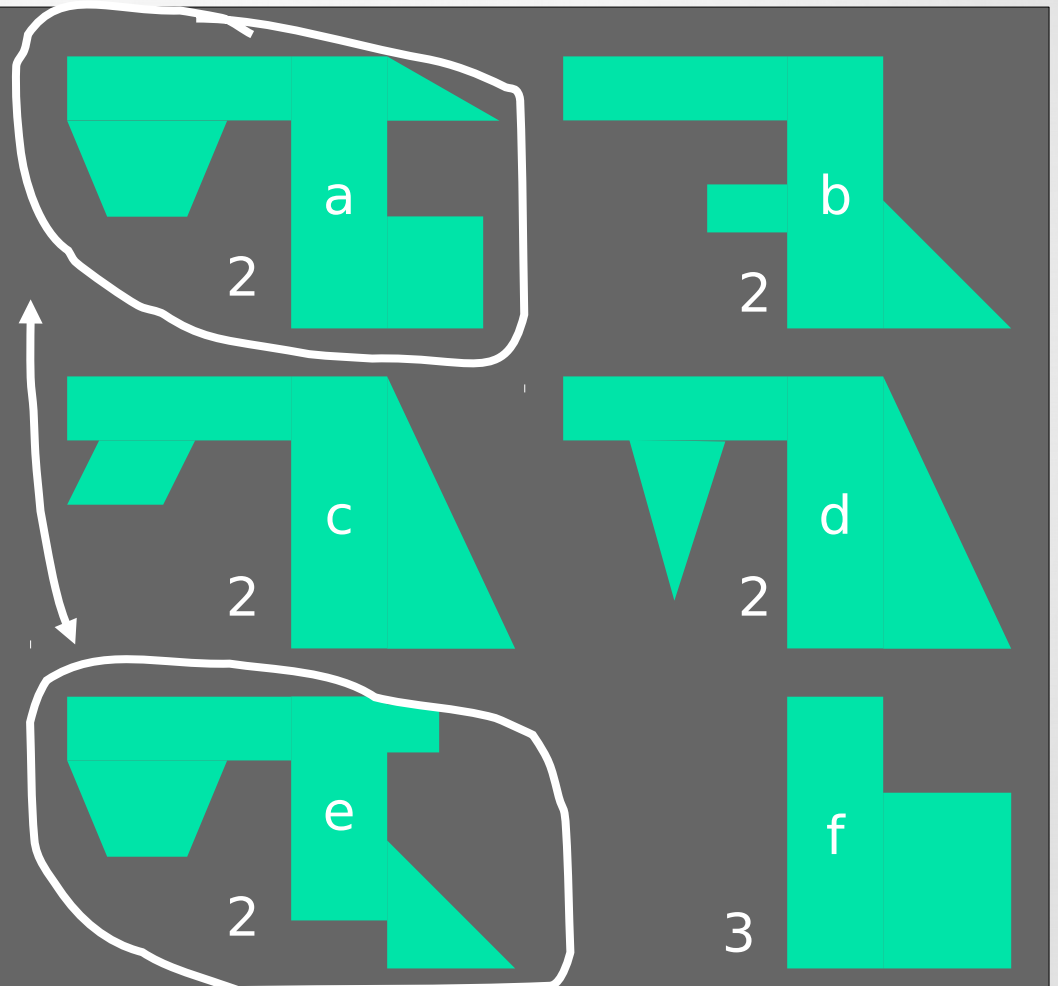
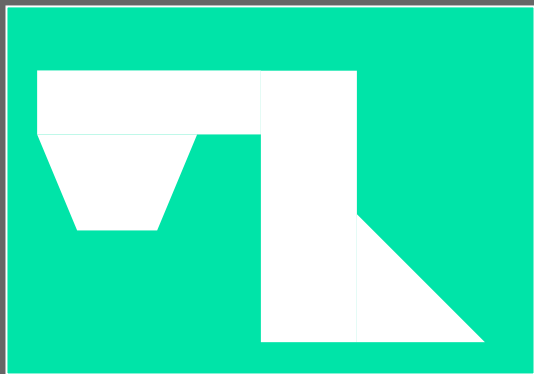


Mutação



## Seleção de a e e

Encontrar a chave  
para o chaveiro:

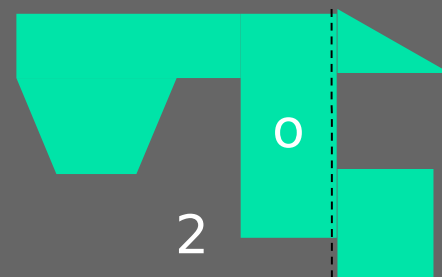
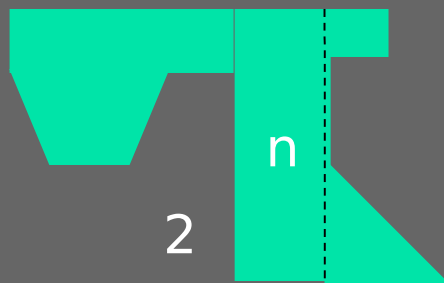
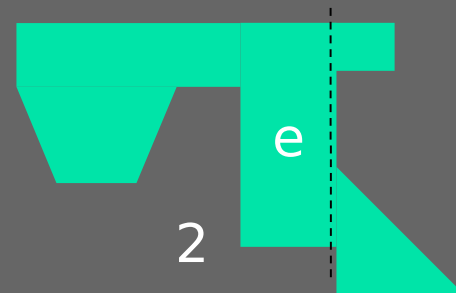
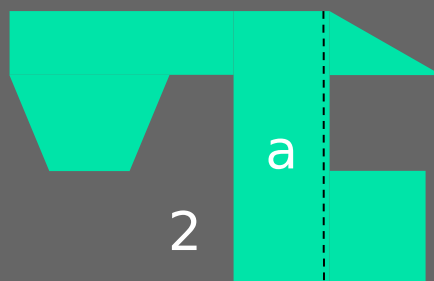


# Cruzamento de a e e

Encontrar a chave  
para o chaveiro:

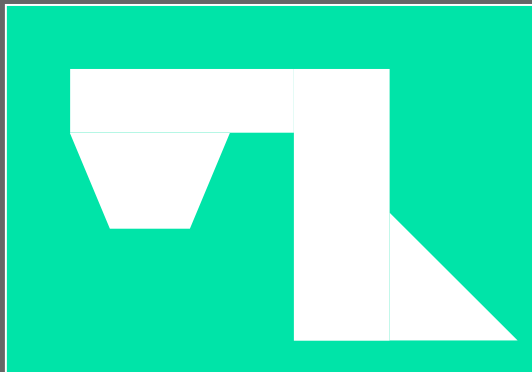


Cruzamento

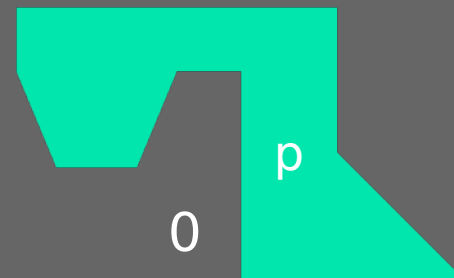
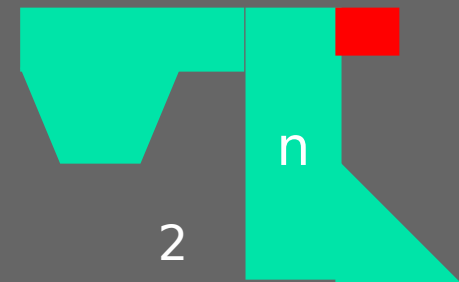


# Mutação de n

Encontrar a chave  
para o chaveiro:



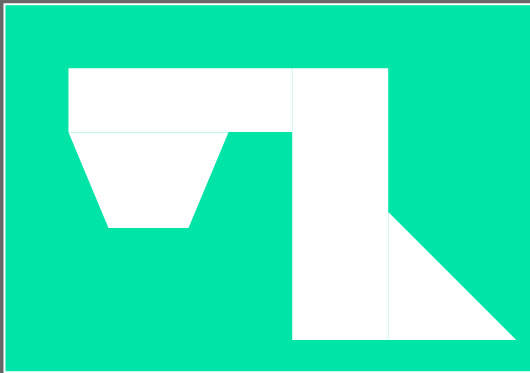
Mutação



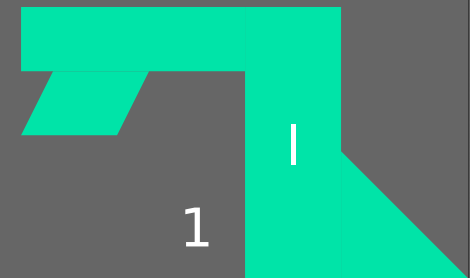
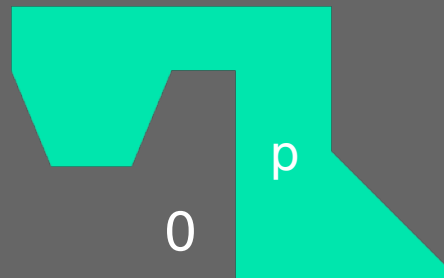
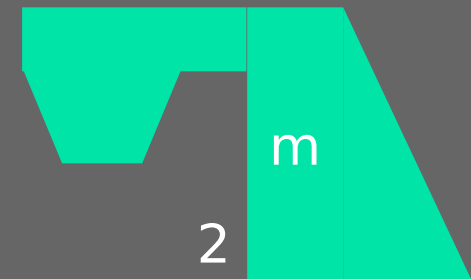
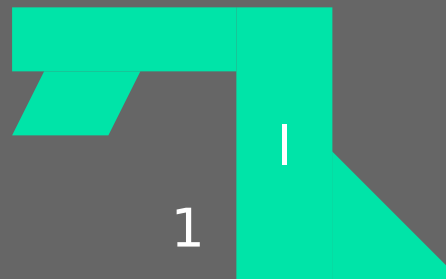
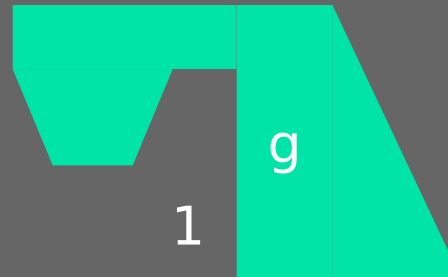


# Nova população

Encontrar a chave  
para a fechadura:

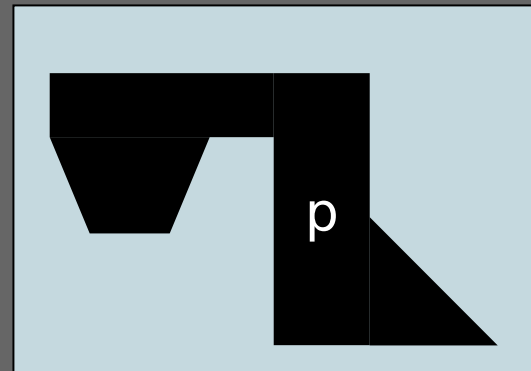
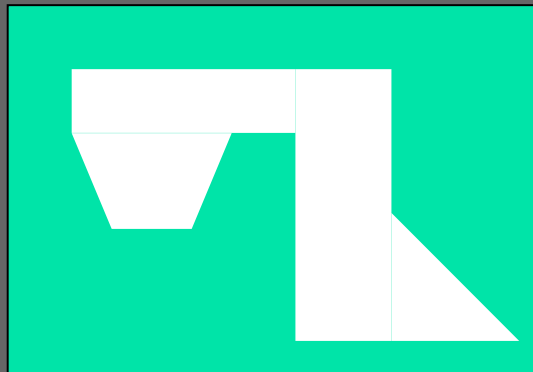


Quanto menor o valor de  
Aptidão, melhor a chave



# Solução

Encontrar a chave  
para o chaveiro:

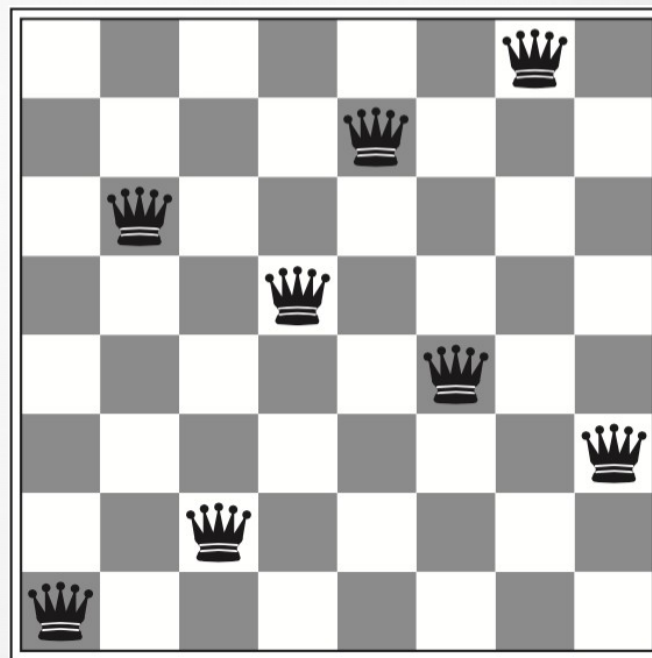


# Exercícios

- Considerando o problema das Oito-Rainhas, como modelar para um problema de busca local/otimização?

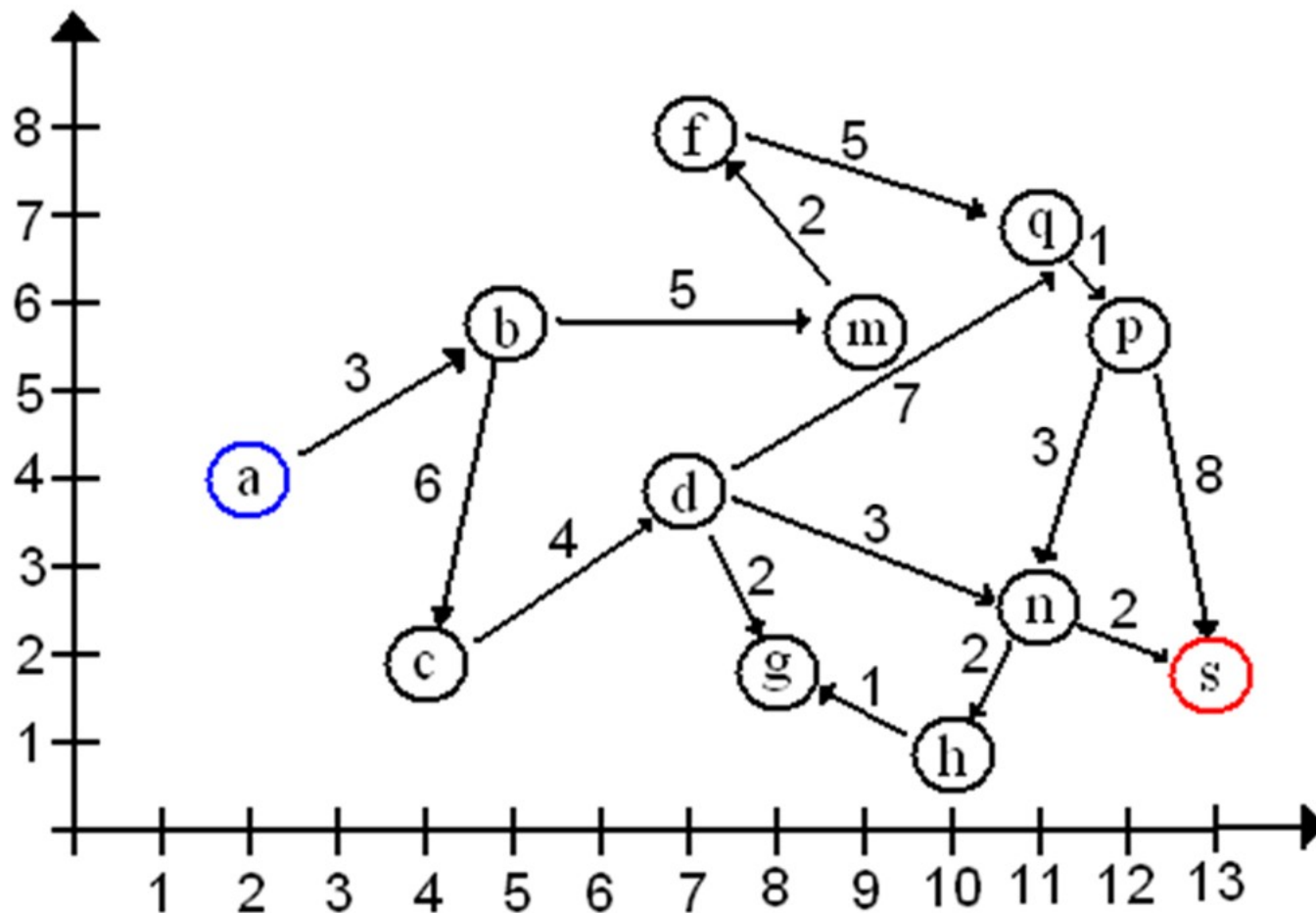
18	12	14	13	13	12	14	14
14	16	13	15	12	14	12	16
14	12	18	13	15	12	14	14
15	14	14	♔	13	16	13	16
♔	14	17	15	♔	14	16	16
17	♔	16	18	15	♔	15	♔
18	14	♔	15	15	14	♔	16
14	14	13	17	12	14	12	18

(a)



(b)

# Exercícios



## Exercícios

- Faça as buscas de custo uniforme e *GBF* para o problema da trajetória entre os nós *a* e *s*.
- Utilize a distância Euclideana como função heurística
- Depois, faça a busca  $A^*$ .
- Mostre como ficam as listas utilizadas pelos algoritmos