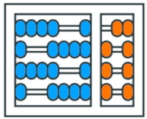


Instituto de Computação

UNIVERSIDADE ESTADUAL DE CAMPINAS

UNICAMP

MC102 - Aula 22

Exemplos sobre Recursão (parte 3)

Algoritmos e Programação de Computadores

Turmas

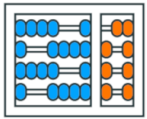
OVXZ

Prof. Lise R. R. Navarrete

lrommel@ic.unicamp.br

Quinta-feira, 09 de junho de 2022

19:00h - 21:00h (CB06)



Instituto de Computação

UNIVERSIDADE ESTADUAL DE CAMPINAS

UNICAMP

MC102 – Algoritmos e Programação de Computadores

Turmas

OVXZ

https://ic.unicamp.br/~mc102/

Site da Coordenação de MC102

Aulas teóricas:

Terça-feira, 21:00h - 23:00h (CB06)

Quinta-feira, 19:00h - 21:00h (CB06)

Conteúdo

Exemplo 13

Encontrar o maior elemento de uma lista.

https://ic.unicamp.br/~mc102/aulas/aula04.pdf

- Encontrando o máximo de um conjunto de números quaisquer.

```
1  numeros = [-3, -1, -7, -9, -4]
2  maximo = numeros[0]
3
4  i = 1
5  while i < len(numeros):
6    if numeros[i] > maximo:
7      maximo = numeros[i]
8    i = i + 1
9
10 print(maximo) # -1
```

```
1 def fatorial(n):
2   if n == 0:
3     return 1
4
5   return n * fatorial(n - 1)
6
```

21

```python
def maximo(lista):
  if len(lista) == 1:
    return  lista[0]



numeros = [-3, -1, -7, -9, -4]
print(maximo(numeros))
```

```
$
```

https://shell.cloud.google.com/

```
1  def maximo(lista):
2    if len(lista) == 1:
3      return  lista[0]
4
5    # retiramos o primeiro elemento
6    primeiro = lista[0]
7
8
9
10
11
12
13
14
15
16
17
18
19 numeros = [-3, -1, -7, -9, -4]
20 print(maximo(numeros))
```

```
$
```

https://shell.cloud.google.com/

```
1  def maximo(lista):
2    if len(lista) == 1:
3      return  lista[0]
4
5    # retiramos o primeiro elemento
6    primeiro = lista[0]
7
8    # a lista sem o primeiro é lista[1:]
9
10
11
12
13
14
15
16
17
18
19 numeros = [-3, -1, -7, -9, -4]
20 print(maximo(numeros))
```

```
$
```

https://shell.cloud.google.com/

```python
def maximo(lista):
  if len(lista) == 1:
    return  lista[0]

  # retiramos o primeiro elemento
  primeiro = lista[0]

  # a lista sem o primeiro é lista[1:]
  # assumimos que conhecemos maximo(lista[1:])









numeros = [-3, -1, -7, -9, -4]
print(maximo(numeros))
```

```
$
```

https://shell.cloud.google.com/

```python
def maximo(lista):
    if len(lista) == 1:
        return  lista[0]

    # retiramos o primeiro elemento
    primeiro = lista[0]

    # a lista sem o primeiro é lista[1:]
    # assumimos que conhecemos maximo(lista[1:])
    # Quem é o maximo entre primeiro e maximo(lista[1:])?









numeros = [-3, -1, -7, -9, -4]
print(maximo(numeros))
```

```
$
```

https://shell.cloud.google.com/

```python
def maximo(lista):
    if len(lista) == 1:
        return  lista[0]

    # retiramos o primeiro elemento
    primeiro = lista[0]

    # a lista sem o primeiro é lista[1:]
    # assumimos que conhecemos maximo(lista[1:])
    # Quem é o maximo entre primeiro e maximo(lista[1:])?
    aux = maximo(lista[1:])
    if aux>primeiro:
        return aux
    else:
        return primeiro



numeros = [-3, -1, -7, -9, -4]
print(maximo(numeros))
```

```
$
```

https://shell.cloud.google.com/

```python
def maximo(lista):
    if len(lista) == 1:
        return  lista[0]

    # retiramos o primeiro elemento
    primeiro = lista[0]

    # a lista sem o primeiro é lista[1:]
    # assumimos que conhecemos maximo(lista[1:])
    # Quem é o maximo entre primeiro e maximo(lista[1:])?
    aux = maximo(lista[1:])
    if aux>primeiro:
        return aux
    else:
        return primeiro



numeros = [-3, -1, -7, -9, -4]
print(maximo(numeros))
```

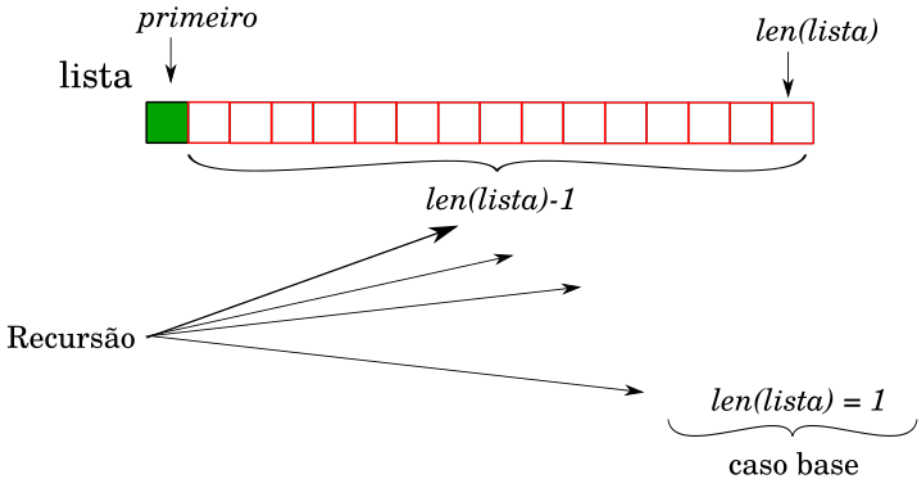
```
$ python3 exe22001.py
-1
$
```

https://shell.cloud.google.com/

```python
def maximo(lista):
    if len(lista) == 1:
        return  lista[0]

    aux = maximo(lista[1:])
    if aux>lista[0]:
        return aux
    else:
        return lista[0]



numeros = [-3, -1, -7, -9, -4]
print(maximo(numeros))
```

```
$ python3 exe22001.py
-1
$
```

https://shell.cloud.google.com/


primeiro
lista
len(lista)
len(lista)-1
Recursão
len(lista) = 1
caso base

https://ic.unicamp.br/~mc102/aulas/aula12.pdf

- Encontrar o maior elemento de uma `lista`.

```
1 def max_lista(lista):
2   if len(lista) == 1:
3     return lista[0]
4   else:
5     aux = max_lista(lista[:-1])
6     if aux > lista[-1]:
7       return aux
8     else:
9       return lista[-1]
```

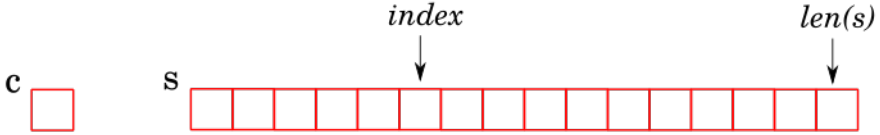
https://ic.unicamp.br/~mc102/aulas/aula12.pdf

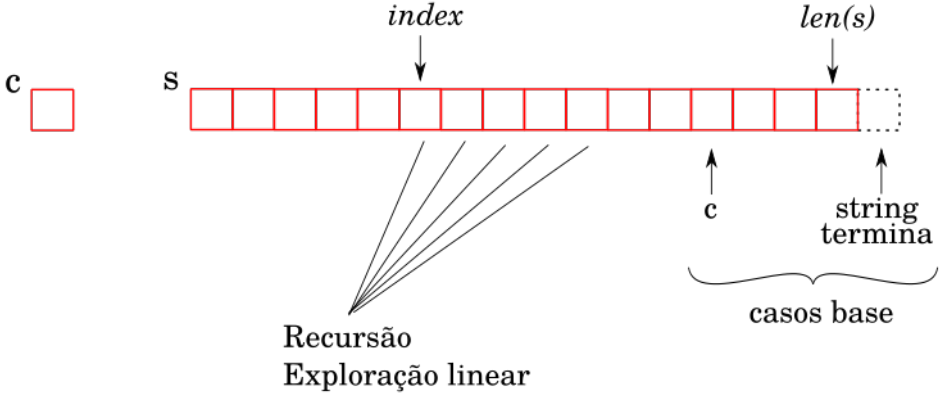
- Encontrar o maior elemento de uma `lista` com `n` números.

```
1 def max_lista(lista, n):
2   if n == 1:
3     return lista[0]
4   else:
5     aux = max_lista(lista, n - 1)
6     if aux > lista[n - 1]:
7       return aux
8     else:
9       return lista[n - 1]
```

Exemplo 14

Buscar um caractere c em uma string s a partir de uma posição $index$ e retornar a posição da primeira ocorrência deste caractere (caso encontre) ou -1 (caso contrario).


index
len(s)
c
s


index
len(s)
c
s
c
string
termina
casos base
Recursão
Exploração linear

```python
def strchr(s, c, index):
  if index >= len(s):
    return -1
  if s[index] == c:
    return index

s = "abracadabra"
print(strchr(s,"b",5))
```

```
$ python3 exe22002.py
None
$
```

https://shell.cloud.google.com/

```python
def strchr(s, c, index):
  if index >= len(s):
    return -1
  if s[index] == c:
    return index

  return strchr(s,c,index+1)









s = "abracadabra"
print(strchr(s,"b",5))
```

```
$ python3 exe22002.py
8
$
```

https://shell.cloud.google.com/

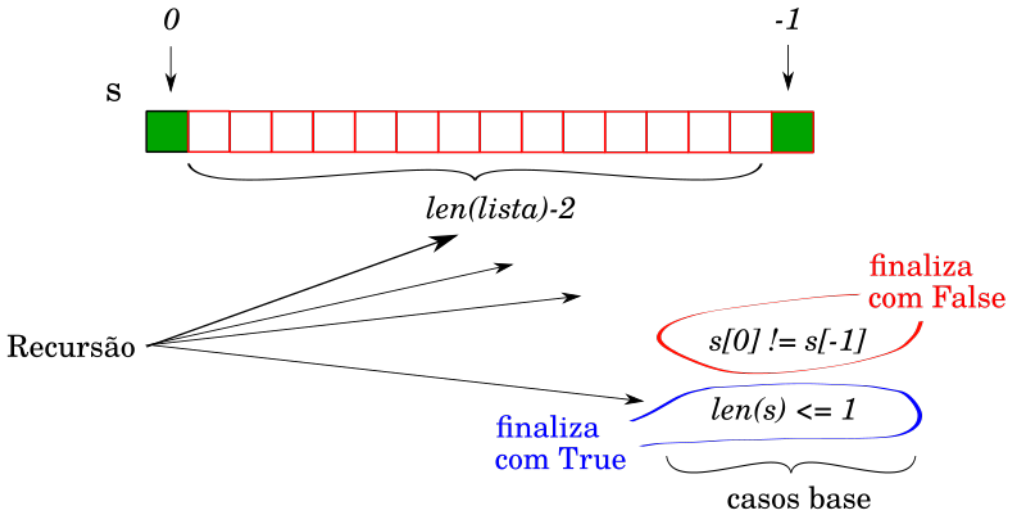
https://ic.unicamp.br/~mc102/aulas/aula12.pdf

- Buscar um caractere `c` em uma string `s` a partir de uma posição `index` e retornar a posição da primeira ocorrência deste caractere (caso encontre) ou -1 (caso contrário).

```
1 def strchr(s, c, index):
2   if index >= len(s):
3     return -1
4   if s[index] == c:
5     return index
6   else:
7     return strchr(s, c, index + 1)
```

Exemplo 15

Verificar se uma *string* é um palíndromo.


0
-1
s
len(lista)-2
Recursão
finaliza
com False
s[0] != s[-1]
len(s) <= 1
finaliza
com True
casos base

https://ic.unicamp.br/~mc102/aulas/aula12.pdf

- Verificar se uma string é um palíndromo.

```
1 def palindromo(s):
2   if len(s) <= 1:
3     return True
4
5   if s[0] != s[-1]:
6     return False
7
8   return palindromo(s[1:-1])
```

Exemplo 16

Inverter uma string dada.

https://ic.unicamp.br/~mc102/aulas/aula12.pdf

- Inverter uma string dada.

```
1 def inverte(s):
2   if len(s) <= 1:
3     return s
4   else:
5     return s[-1] + inverte(s[:-1])
```

https://ic.unicamp.br/~mc102/aulas/aula12.pdf

- Inverter uma string dada.

```
1 def inverte(s):
2   if len(s) <= 1:
3     return s
4   else:
5     return s[-1] + inverte(s[1:-1]) + s[0]
```

```python
def inverte(s):
  if len(s) <= 1:
    return s

  return s[-1] + inverte(s[:-1])









s = "abracadabra"
print(inverte(s))
```

```
$ python3 exe22016.py
arbadacarba
$
```

https://shell.cloud.google.com/

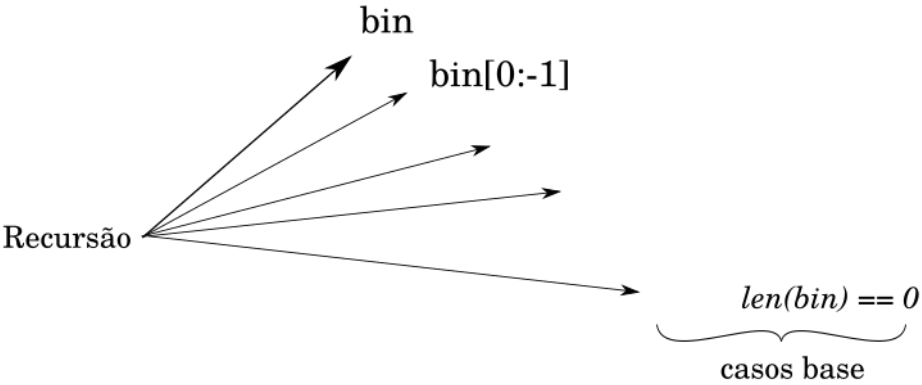
```python
def inverte(s):
  if len(s) <= 1:
    return s

  return s[-1] + inverte(s[1:-1]) + s[0]



s = "abracadabra"
print(inverte(s))
```

```
$ python3 exe22016.py
arbadacarba
$
```

https://shell.cloud.google.com/

Exemplo 17

Converter um número binário (string) num número decimal(inteiro).


bin
bin[0:-1]
Recursão
len(bin) == 0
casos base

https://ic.unicamp.br/~mc102/aulas/aula12.pdf

- Converter um número binário (string) num número decimal (inteiro).

```
1 def converte(bin):
2   if len(bin) == 0:
3     return 0
4   return 2 * converte(bin[0:-1]) + int(bin[-1])
```

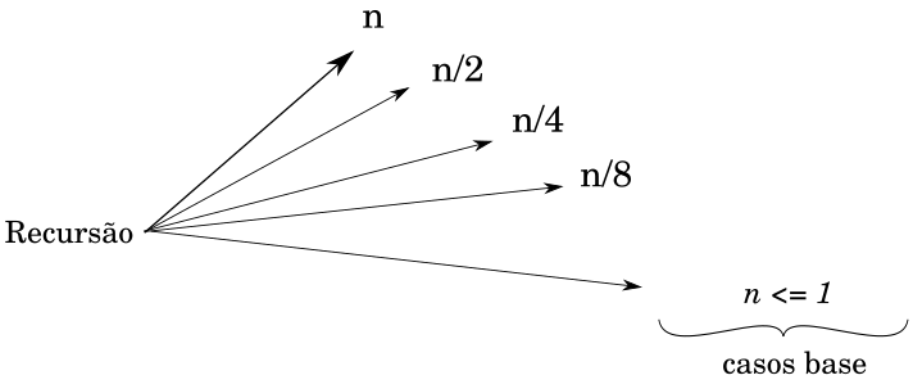
```python
def converte(bin):
  if len(bin) == 0:
    return 0

  return 2 * converte(bin[0:-1]) + int(bin[-1])









num1 =  "1100110"
num2 =  "11111100100"
print(converte(num1))
print(converte(num2))
```

```
$ python3 exe22017.py
102
2020
$
```

https://shell.cloud.google.com/

Exemplo 18

Escreva uma função recursiva que, dado um número inteiro positivo n, retorne a representação binária de n.

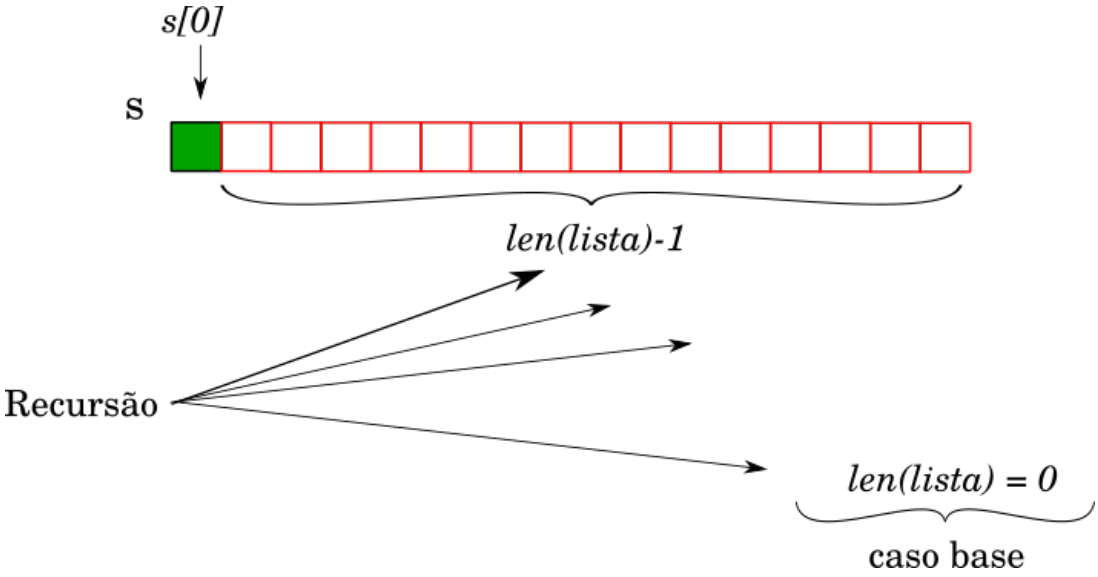

n
n/2
n/4
n/8
Recursão
n <= 1
casos base

https://ic.unicamp.br/~mc102/aulas/aula12.pdf

```
1 def binario(n):
2   if n <= 1:
3     return str(n)
4   else:
5     return binario(n // 2) + str(n % 2)
```

Exemplo 19

Escreva uma função recursiva que, dada uma string s e um caractere c, conte o número de ocorrências do caractere c na string s.


s[0]
s
len(lista)-1
Recursão
len(lista) = 0
caso base

https://ic.unicamp.br/~mc102/aulas/aula12.pdf

```
1 def conta(s, c):
2   if len(s) == 0:
3     return 0
4   aux = conta(s[1:], c)
5   if s[0] == c:
6     return 1 + aux
7   else:
8     return aux
```

https://ic.unicamp.br/~mc102/aulas/aula12.pdf

```
1 def conta(s, c):
2   if s == "":
3     return 0
4   aux = conta(s[1:], c)
5   if s[0] == c:
6     return 1 + aux
7   else:
8     return aux
```

https://ic.unicamp.br/~mc102/aulas/aula12.pdf

```
1 def conta(s, c):
2   if s == "":
3     return 0
4
5   return (s[0] == c) + conta(s[1:], c)
```

Exemplo 20

Escreva uma função recursiva que, dada uma lista l de n números inteiros ordenados $(n \geq 1)$ e um inteiro x, retorne, usando uma busca binária, o índice de x na lista ou o valor -1, caso x não pertença à lista.

https://ic.unicamp.br/~mc102/aulas/aula12.pdf

```
1  def busca_binaria(lista, inicio, fim, chave):
2    if inicio > fim:
3      return -1
4
5    meio = (inicio + fim) // 2
6    if chave == lista[meio]:
7      return meio
8
9    if chave < lista[meio]:
10     return busca_binaria(lista, inicio, meio - 1, chave)
11   else:
12     return busca_binaria(lista, meio + 1, fim, chave)
```

Exemplo 21

Escreva versões recursivas das funções

indiceMenor e selectionSort.

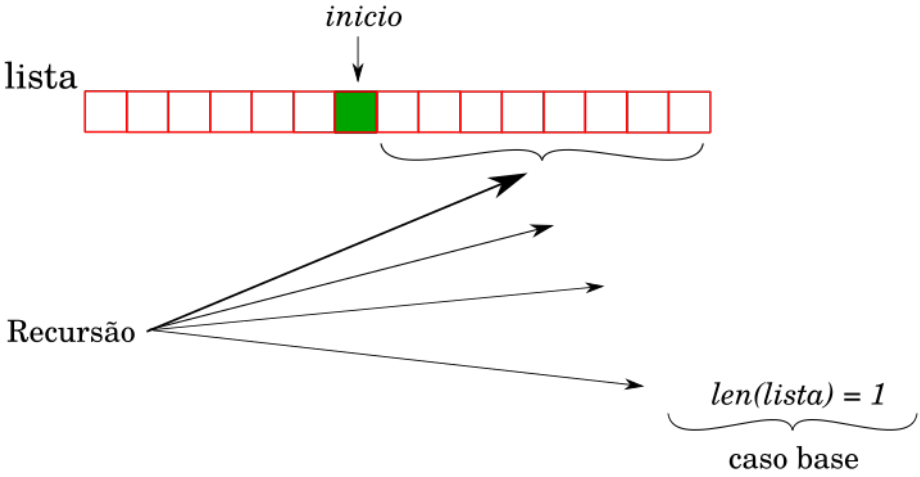
Importante: todas as funções acima devem ser implementadas sem qualquer comando de repetição (for, while, etc).

indiceMenor iterativo

https://ic.unicamp.br/~mc102/aulas/aula10.pdf

- Podemos criar uma função que retorna o índice do menor elemento de uma lista (formado por n números inteiros) a partir de uma posição inicial dada:

```
1 def indiceMenor(lista, inicio):
2   minimo = inicio
3   n = len(lista)
4   for j in range(inicio + 1, n):
5     if lista[minimo] > lista[j]:
6       minimo = j
7   return minimo
```


inicio
lista
Recursão
len(lista) = 1
caso base

https://ic.unicamp.br/~mc102/aulas/aula12.pdf

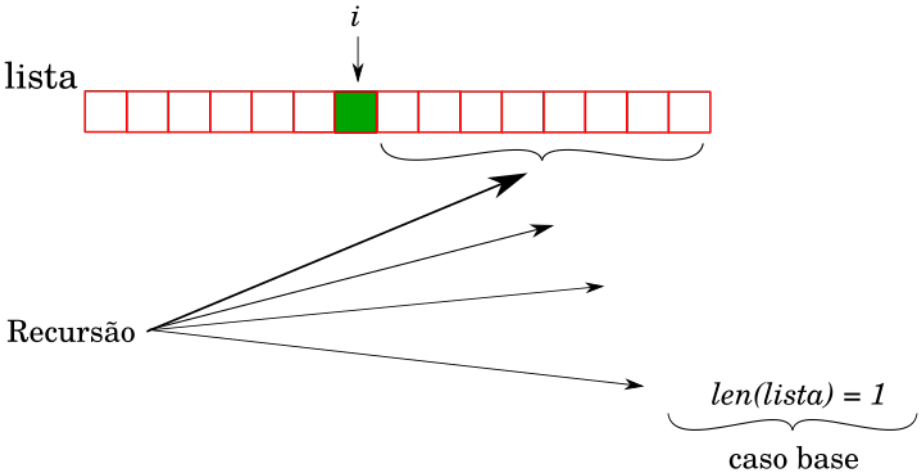
```
1 def indiceMenor(lista, inicio):
2   if inicio == len(lista) - 1:
3     return inicio
4
5   aux = indiceMenor(lista, inicio + 1)
6   if lista[inicio] < lista[aux]:
7     return inicio
8   else:
9     return aux
```

selectionSort iterativo

https://ic.unicamp.br/~mc102/aulas/aula10.pdf

- Usando a função auxiliar `indiceMenor` podemos implementar o Selection Sort da seguinte forma:

```
1 def selectionSort(lista):
2   n = len(lista)
3   for i in range(n - 1):
4     minimo = indiceMenor(lista, i)
5     (lista[i], lista[minimo]) = (lista[minimo], lista[i])
```


i
lista
Recursão
len(lista) = 1
caso base

https://ic.unicamp.br/~mc102/aulas/aula12.pdf

```
1 def selectionSort(lista, i):
2   if i == len(lista) - 1:
3     return
4   else:
5     minimo = indiceMenor(lista, i)
6     (lista[i], lista[minimo]) = (lista[minimo], lista[i])
7     selectionSort(lista, i + 1)
```

https://ic.unicamp.br/~mc102/aulas/aula12.pdf

```
1 def selectionSort(lista, i):
2   if i < len(lista) - 1:
3     minimo = indiceMenor(lista, i)
4     (lista[i], lista[minimo]) = (lista[minimo], lista[i])
5     selectionSort(lista, i + 1)
```

Perguntas ....

Referências

- Zanoni Dias, MC102, Algoritmos e Programação de Computadores, IC/UNICAMP, 2021. https://ic.unicamp.br/~mc102/

  Aula Introdutória [slides] [vídeo]
  Primeira Aula de Laboratório [slides] [vídeo]
  Python Básico: Tipos, Variáveis, Operadores, Entrada e Saída [slides] [vídeo]
  Comandos Condicionais [slides] [vídeo]
  Comandos de Repetição [slides] [vídeo]
  Listas e Tuplas [slides] [vídeo]
  Strings [slides] [vídeo]
  Dicionários [slides] [vídeo]
  Funções [slides] [vídeo]
  Objetos Multidimensionais [slides] [vídeo]
  Algoritmos de Ordenação [slides] [vídeo]
  Algoritmos de Busca [slides] [vídeo]
  Recursão [slides] [vídeo]
  Algoritmos de Ordenação Recursivos [slides] [vídeo]
  Arquivos [slides] [vídeo]
  Expressões Regulares [slides] [vídeo]
  Execução de Testes no Google Cloud Shell [slides] [vídeo]
  Numpy [slides] [vídeo]
  Pandas [slides] [vídeo]

- Panda - Cursos de Computação em Python (IME -USP) https://panda.ime.usp.br/

  Como Pensar Como um Cientista da Computação https://panda.ime.usp.br/pensepy/static/pensepy/
  Aulas de Introdução à Computação em Python https://panda.ime.usp.br/aulasPython/static/aulasPython/

- Fabio Kon, Introdução à Ciência da Computação com Python http://bit.ly/FabioKon/
- Socratica, Python Programming Tutorials http://bit.ly/SocraticaPython/
- Google - online editor for cloud-native applications (Python programming) https://shell.cloud.google.com/
- w3schools - Python Tutorial https://www.w3schools.com/python/
- Outros, citados nos Slides.