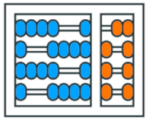


Instituto de Computação

UNIVERSIDADE ESTADUAL DE CAMPINAS

UNICAMP

MC102 - Aula 18

Exemplos sobre Ordenação e Busca

Algoritmos e Programação de Computadores

Turmas

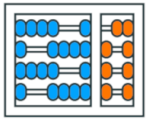
OVXZ

Prof. Lise R. R. Navarrete

lrommel@ic.unicamp.br

Quinta-feira, 26 de maio 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 1
- Exemplo 2
- Exemplo 3
- Exemplo 4
- Exemplo 5

Exemplo 1

Ordenando uma String

```python
def insertion(lista, i):
    j = i - 1
    while (j >= 0) and (lista[j] > lista[i]):
        j = j - 1
    lista[j + 1:i + 1] = [lista[i]] + lista[j + 1:i]
def insertionSort(lista):
    for i in range(1, len(lista)):
        insertion(lista, i)

A=[9,8,2,1,3,6,7,2]
insertionSort(A)
print(A)
```

```
$ python3 exe18001.py
[1, 2, 2, 3, 6, 7, 8, 9]
$
```

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

```python
def insertion(lista, i):
    j = i - 1
    while (j >= 0) and (lista[j] > lista[i]):
        j = j - 1
    lista.insert(j+1,lista.pop(i))
def insertionSort(lista):
    for i in range(1, len(lista)):
        insertion(lista, i)

A=[9,8,2,1,3,6,7,2]
insertionSort(A)
print(A)
```

```
$ python3 exe18001.py
[1, 2, 2, 3, 6, 7, 8, 9]
$
```

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

```python
def insertionSort(lista):
    for i in range(1, len(lista)):
        j = i - 1
        while (j >= 0) and (lista[j] > lista[i]):
            j = j - 1
        lista.insert(j+1,lista.pop(i))

A=[9,8,2,1,3,6,7,2]
insertionSort(A)
print(A)
```

```
$ python3 exe18001.py
[1, 2, 2, 3, 6, 7, 8, 9]
$
```

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

```
1   def insertionSort(lista):
2       for i in range(1, len(lista)):
3           j = i - 1
4           while (j >= 0) and (lista[j] > lista[i]):
5             j = j - 1
6           lista.insert(j+1,lista.pop(i))
7
8   A="98213672"
9   insertionSort(A)
10  print(A)
```

```
  File "/home/nuvemaula/exe18001.py", line 9, in <module>
    insertionSort(A)
  File "/home/nuvemaula/exe18001.py", line 6, in insertionSort
    lista.insert(j+1,lista.pop(i))
AttributeError: 'str' object has no attribute 'insert'
$
```

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

```python
def insertionSort(lista):
    for i in range(1, len(lista)):
        j = i - 1
        while (j >= 0) and (lista[j] > lista[i]):
            j = j - 1
        lista.insert(j+1,lista.pop(i))

A=list("98213672")
insertionSort(A)
print(A)
```

```
$ python3 exe18001.py
['1', '2', '2', '3', '6', '7', '8', '9']
$
```

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

```python
def insertionSort(lista):
    for i in range(1, len(lista)):
        j = i - 1
        while (j >= 0) and (lista[j] > lista[i]):
            j = j - 1
        lista.insert(j+1,lista.pop(i))

A=list("98213672")
insertionSort(A)
print("".join(A))
```

```
$ python3 exe18001.py
12236789
$
```

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

```python
def insertionSort(lista):
    for i in range(1, len(lista)):
        j = i - 1
        while (j >= 0) and (lista[j] > lista[i]):
            j = j - 1
        lista.insert(j+1,lista.pop(i))

A=list("Socorram-me, subi no ônibus em Marrocos!")
insertionSort(A)
print("".join(A))
```

```
$ python3 exe18001.py
```

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

```python
def insertionSort(lista):
    for i in range(1, len(lista)):
        j = i - 1
        while (j >= 0) and (lista[j] > lista[i]):
            j = j - 1
        lista.insert(j+1,lista.pop(i))

A=list("Socorram-me, subi no ônibus em Marrocos!")
insertionSort(A)
print("".join(A))
```

```
$ python3 exe18001.py
     !,-MSaabbcceeiimmmnnooooorrrrsssuuô
$
```

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

Exemplo 2

Ordenação com predicado

```python
def insertionSort(lista):
    for i in range(1, len(lista)):
        j = i - 1
        while (j >= 0) and (lista[j] > lista[i]):
            j = j - 1
        lista.insert(j+1,lista.pop(i))

A=[("casa",1),("carro",2),("mesa",5),("cadeira",4)]

insertionSort(A)
print(A)

```

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

```python
def insertionSort(lista):
    for i in range(1, len(lista)):
        j = i - 1
        while (j >= 0) and (lista[j] > lista[i]):
            j = j - 1
        lista.insert(j+1,lista.pop(i))

A=[("casa",1),("carro",2),("mesa",5),("cadeira",4)]

insertionSort(A)
print(A)
```

```
$ python3 exe18002.py
[('cadeira', 4), ('carro', 2), ('casa', 1), ('mesa', 5)]
$
```

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

```python
def insertionSort(lista):
    for i in range(1, len(lista)):
        j = i - 1
        while (j >= 0) and (lista[j][1] > lista[i][1]):
            j = j - 1
        lista.insert(j+1,lista.pop(i))

A=[("casa",1),("carro",2),("mesa",5),("cadeira",4)]

insertionSort(A)
print(A)

```

```
$ python3 exe18002.py
[('casa', 1), ('carro', 2), ('cadeira', 4), ('mesa', 5)]
$
```

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

```python
def insertionSort(lista,func):
    for i in range(1, len(lista)):
        j = i - 1
        while (j >= 0) and func(lista[j],lista[i]):
            j = j - 1
        lista.insert(j+1,lista.pop(i))

A=[("casa",1),("carro",2),("mesa",5),("cadeira",4)]
def compara(A,B):
    return A[1] > B[1]

insertionSort(A,compara)
print(A)
```

```
$ python3 exe18002.py
[('casa', 1), ('carro', 2), ('cadeira', 4), ('mesa', 5)]
$
```

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

```python
def insertionSort(lista,func):
    for i in range(1, len(lista)):
        j = i - 1
        while (j >= 0) and func(lista[j],lista[i]):
            j = j - 1
        lista.insert(j+1,lista.pop(i))

A=[("casa",1,3),("carro",2,2),("mesa",5,3),("cadeira",4,10)]
def compara(A,B):
    return A[1] > B[1]

insertionSort(A,compara)
print(A)
```

```
$ clear
$ python3 exe18002.py
[('casa', 1, 3), ('carro', 2, 2), ('cadeira', 4, 10), ('mesa', 5, 3)]
$
```

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

```python
def insertionSort(lista,func):
    for i in range(1, len(lista)):
        j = i - 1
        while (j >= 0) and func(lista[j],lista[i]):
            j = j - 1
        lista.insert(j+1,lista.pop(i))

A=[("C",1,3),("CA",2,2),("ME",5,3),("CA",4,10),("A",1.5)]
def compara(A,B):
    return A[1] > B[1]

insertionSort(A,compara)
print(A)
```

```
$ python3 exe18002.py
[('C', 1, 3), ('A', 1.5), ('CA', 2, 2), ('CA', 4, 10), ('ME', 5, 3)]
$
```

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

```
8   A=[("C",1,3),("CA",2,2),("ME",5,3),("CA",4,10),("A",1.5)]
9
10  def fun01(x):
11      return x[1]
12
13
14  A.sort(key=fun01)
15  print(A)
16
```

```
$ python3 exe18002.py
[('C', 1, 3), ('A', 1.5), ('CA', 2, 2), ('CA', 4, 10), ('ME', 5, 3)]
$
```

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

```
8   A=[("C",1,3),("CA",2,2),("ME",5,3),("CA",4,10),("A",1.5)]
9
10
11
12
13
14  A.sort(key=lambda x:x[1])
15  print(A)
16
```

```
$ python3 exe18002.py
[('C', 1, 3), ('A', 1.5), ('CA', 2, 2), ('CA', 4, 10), ('ME', 5, 3)]
$
```

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

Exemplo 3

Altere o Bubble Sort para que o algoritmo pare assim que for possível perceber que a lista está ordenada. Qual o custo deste novo algoritmo em termos do número de comparações entre elementos da lista (tanto no melhor, quanto no pior caso)?

```python
def bubbleSort(lista):
  n = len(lista)
  for i in range(n - 1, 0, -1):
    for j in range(i):
      if lista[j] > lista[j + 1]:
        (lista[j], lista[j + 1]) = (lista[j + 1], lista[j])

A = [1, -10, 13, 7, 67]

bubbleSort(A)
print(A)
```

```
$ python3 exe18003.py
[-10, 1, 7, 13, 67]
$
```

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

```python
def bubbleSort(lista):
    n = len(lista)
    for i in range(n - 1, 0, -1):
        flag = 0
        for j in range(i):
            if lista[j] > lista[j + 1]:
                (lista[j], lista[j + 1]) = (lista[j + 1], lista[j])
                flag = 1
        if flag == 0:
            break
A = [1, -10, 13, 7, 67]
bubbleSort(A)
print(A)
```

```
$ python3 exe18003.py
[-10, 1, 7, 13, 67]
$
```

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

Exemplo 4

Escreva uma função k-ésimo que, dada uma lista de tamanho n e um inteiro k (tal que $1 \leq k \leq n$), determine o k-ésimo menor elemento da lista. Analise o custo da sua função em termos do número de comparações realizadas entre elementos da lista.

```python
def bubbleSort(lista):
    n = len(lista)
    for i in range(n - 1, 0, -1):
        for j in range(i):
            if lista[j] > lista[j + 1]:
                (lista[j], lista[j + 1]) = (lista[j + 1], lista[j])

A = [1, -10, 13, 7, 67]

bubbleSort(A)
print(A)
```

```
$ python3 exe18003.py
[-10, 1, 7, 13, 67]
$
```

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

```python
def bubbleSortK(A,k):
  lista = A.copy()
  nums  = list(range(len(A)))
  n = len(lista)
  for i in range(n - 1, 0, -1):
    for j in range(i):
      if lista[j] > lista[j + 1]:
        (lista[j], lista[j + 1]) = (lista[j + 1], lista[j])
        ( nums[j],  nums[j + 1]) = ( nums[j + 1],  nums[j])
  return nums[k]
A = [1, -10, 13, 7, 67]
print( A[bubbleSortK(A,4)])

```

```
$
```

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

```python
def bubbleSortK(A,k):
  lista = A.copy()
  nums  = list(range(len(A)))
  n = len(lista)
  for i in range(n - 1, 0, -1):
    for j in range(i):
      if lista[j] > lista[j + 1]:
        (lista[j], lista[j + 1]) = (lista[j + 1], lista[j])
        ( nums[j],  nums[j + 1]) = ( nums[j + 1],  nums[j])
  return nums[k]
A = [1, -10, 13, 7, 67]
print( A[bubbleSortK(A,4)])

```

```
$ python3 exe18004.py
67
$
```

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

```
1   def bubbleSortK(A,k):
2     lista = A.copy()
3     nums  = list(range(len(A)))
4     n = len(lista)
5     for i in range(n - 1, 0, -1):
6       for j in range(i):
7         if lista[j] > lista[j + 1]:
8           (lista[j], lista[j + 1]) = (lista[j + 1], lista[j])
9           ( nums[j],  nums[j + 1]) = ( nums[j + 1],  nums[j])
10    return nums[k]
11  A = [1, -10, 13, 7, 67]
12  print( A[bubbleSortK(A,3)])
13
```

```
$ python3 exe18004.py
13
$
```

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

```python
def bubbleSortK(A,k):
  lista = A.copy()
  nums  = list(range(len(A)))
  n = len(lista)
  for i in range(n - 1, 0, -1):
    for j in range(i):
      if lista[j] > lista[j + 1]:
        (lista[j], lista[j + 1]) = (lista[j + 1], lista[j])
        ( nums[j],  nums[j + 1]) = ( nums[j + 1],  nums[j])
  return nums[k]
A = [1, -10, 13, 7, 67]
print( A[bubbleSortK(A,0)])

```

```
$ python3 exe18004.py
-10
$
```

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

```
1  def bubbleSortK(A,k):
2    lista = A.copy()
3    nums  = list(range(len(A)))
4    n = len(lista)
5    for i in range(n - 1, k, -1):
6      for j in range(i):
7        if lista[j] > lista[j + 1]:
8          (lista[j], lista[j + 1]) = (lista[j + 1], lista[j])
9          ( nums[j],  nums[j + 1]) = ( nums[j + 1],  nums[j])
10   return nums[k]
11 A = [1, -10, 13, 7, 67]
12 print( A[bubbleSortK(A,3)])
13
```

temos o k-ésimo

```
$ python3 exe18004.py
13
$
```

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

Exemplo 5

https://ic.unicamp.br/~mc102/aulas/aula11.pdf

1. Refaça as funções de busca sequencial e busca binária assumindo que a lista possui chaves que podem ocorrer múltiplas vezes na lista. Neste caso, você deve retornar uma lista com todas as posições onde a chave foi encontrada. Se a chave não for encontrada na lista, retornar uma lista vazia.

```python
def buscaBinária(lista, chave):
    pos_ini = 0
    pos_fim = len(lista) - 1
    while pos_ini <= pos_fim:
        pos_meio = (pos_ini + pos_fim) // 2
        if lista[pos_meio] == chave:
            return pos_meio
        if lista[pos_meio] > chave:
            pos_fim = pos_meio - 1
        else:
            pos_ini = pos_meio + 1
    return -1
print(buscaBinária([2,4,4,4,4,31],4))
```

```
$ python3 exe18005.py
2
$
```

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

```
2     pos_ini = 0
3     pos_fim = len(lista) - 1
4     while pos_ini <= pos_fim:
5       pos_meio = (pos_ini + pos_fim) // 2
6       if lista[pos_meio] == chave:
7         while lista[pos_meio-1] == chave:
8           pos_meio -= 1
9         L=[]
0         while lista[pos_meio] == chave:
1           L.append(lista[pos_meio])
2           pos_meio += 1
3         return L
4       if lista[pos_meio] > chave:
5         pos_fim = pos_meio - 1
6       else:
7         pos_ini = pos_meio + 1
8     return -1
9   print(buscaBinária([2,4,4,4,4,31],4))
```

```
$ python3 exe18005.py
[4, 4, 4, 4]
$
```

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

```
    pos_ini = 0
    pos_fim = len(lista) - 1
    while pos_ini <= pos_fim:
      pos_meio = (pos_ini + pos_fim) // 2
      if lista[pos_meio] == chave:
        while lista[pos_meio-1] == chave:
            pos_meio -= 1
        L=[]
        while lista[pos_meio] == chave:
            L.append(pos_meio)
            pos_meio += 1
        return L
      if lista[pos_meio] > chave:
        pos_fim = pos_meio - 1
      else:
        pos_ini = pos_meio + 1
    return -1
print(buscaBinária([2,4,4,4,4,31],4))
```

```
$ python3 exe18005.py
[1, 2, 3, 4]
$
```

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

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.