Algorítmos (2)

Aula 9

Curso: BIG863 - Basic Python Programming for Ecologists

Professora: Dra. Cecilia F. Fiorini

Supervisor: Prof. Dr. Fernando A. O. Silveira

https://meet.google.com/zdi-ueoz-nsr, 10 de maio de 2023

Roteiro*

1. Exercícios - Ordenanção
2. Exercício - Busca

*Conteúdo adaptado a partir de material desenvolvido pelo Prof. Zanoni Dias e disponível em https://ic.unicamp.br/ mc102.

ECMVS

Exercício 1

- Altere o Bubble Sort para que o algoritmo pare assim que for possível perceber que a lista está ordenada.

ECMVS

Exercício 1 - Resposta

```
1  def bubbleSort(lista):
2      n = len(lista)
3      for i in range(n - 1, 0, -1):
4          trocou = False  # variável de controle
5          for j in range(i):
6              if lista[j] > lista[j + 1]:
7                  lista[j], lista[j + 1] = lista[j + 1], lista[j]
8                  trocou = True  # houve troca
9          if not trocou:  # se não houve troca, a lista já está ordenada
10             break
11     return lista
```


ECMVS

Exercício 2

- Escreva uma função k-ésimo que, dada uma lista de tamanho n e um inteiro k, determine o k-ésimo menor elemento da lista. Nota: $1 \leq k \leq n$

ECMVS

Exercício 2 - Resposta

```
1  def k_esimo(lista, k):
2      for i in range(k):
3          menor = i
4          for j in range(i+1, len(lista)):
5              if lista[j] < lista[menor]:
6                  menor = j
7          lista[i], lista[menor] = lista[menor], lista[i]
8      return lista[k-1]
```


ECMVS

Exercício 3

- Mostre como implementar uma variação da busca binária que retorne um inteiro k entre 0 e n, tal que, ou lista[k] = chave, ou a chave não se encontra na lista, mas poderia ser inserida entre as posições (k-1) e k de forma a manter a lista ordenada. Note que, se k = 0, então a chave deveria ser inserida antes da primeira posição da lista, assim como, se k = n, a chave deveria ser inserida após a última posição da lista.

ECMVS

Exercício 3 - Resposta

```
1  def buscaBinária(lista, chave):
2      pos_ini = 0
3      pos_fim = len(lista) - 1
4
5      while pos_ini <= pos_fim:
6          pos_meio = (pos_ini + pos_fim) // 2
7
8          if lista[pos_meio] == chave:
9              return pos_meio
10
11         if lista[pos_meio] > chave:
12             pos_fim = pos_meio - 1
13         else:
14             pos_ini = pos_meio + 1
15
16 ...
```


ECMVS

Exercício 3 - Resposta (continuação)

```
1  def buscaBinária(lista, chave):
2  ...
3      if pos_ini == 0:
4          return 0
5      elif pos_ini == len(lista):
6          return len(lista)
7      elif lista[pos_ini] > chave:
8          return pos_ini
9      else: #lista[pos_ini] < chave
10         return pos_ini + 1
```


ECMVS