Algoritmos e Programação de Computadores

Revisão e Exercícios para Prova 2

Inc. material do prof. Ricardo Caceffo (orig. para aula de revisão interativa com uso de *clickers*, A28)
Instituto de Computação (IC/Unicamp)

Resumo ...

Resumo de Tópicos

Assuntos tratados	**Descrição**
Algumas estruturas de dados básicas em Python	Definição e uso de Tuplas e Dicionários. Operações e métodos em `{key:value}`
Funções > **def** `func(a,b,c):`	Definição, Uso e chamada (invocação) de funções, passagem de parâmetros, return
Funções > **def** `main`, **def** `f1`, **def** `f2`	Uso de variáveis globais e locais
Escopo e visibilidade de variáveis	Uso e visibilidades de listas em funções
Matrizes e Vetores Multidimensionais	Matrizes e vetores multi-dimensionais Declaração e uso de matrizes, vetores. Uso de package NumPy e o tipo array

Resumo de Tópicos

Tópicos	Descrição
Algoritmos de Ordenação: selection Sort, bubbleSort e insertionSort	Uso de função def selectionSort(vetor): for i in vetor > indiceMenor(vetor,i): Uso de função def bubbleSort(vetor): Uso de função de insertionSort(vetor):
Algoritmos de busca: busca sequencial e busca binária	Uso de função def buscaSequencial(lista,key): Uso de função de buscaBinaria(lista,key): Exercicios …. Inc. múltiplos itens
***Arquivos (texto) (Não cobrado na prova)	Uso e manipulação de arquivos de tipo texto (leitura e escrita); uso de parâmetros, sys.argv

Resumo de Tópicos

Tópicos	**Descrição**
***Arquivos (binários), pickle.dump/ load (parte I).	Definição, e manipulação (leitura, escrita) de arquivos binários – uso de util. *pickle*
Funções Recursivas (parte II.)	Definição e Uso de ~funções recursivas; Exercícios (soma de elementos, fatorial)
Algoritmo de ordenação quickSort	Descrição do algoritmo, definição de estratégia `divide-conquer` e exercícios
Algoritmo de ordenação mergeSort	Descrição do algoritmo, definição de estratégia `divide-conquer` e exercícios

Exemplo da prova

Exemplo (Matrizes)

- Seja P a propriedade que indica que uma matriz é quadrada e que os elementos em sua diagonal principal são iguais a 1 e os outros são iguais a 0. Maria gostaria de escrever uma função em Python para verificar a propriedade P em uma matriz representada por uma lista de listas contendo números inteiros.
- Inicialmente, Maria precisa identificar a propriedade em algumas matrizes. Indique para cada estrutura de dados apresentada abaixo se esta representa uma matriz que respeita a propriedade P ou não. Em caso de violação da propriedade, escreva uma justificativa.

`[[2, 0, 0],`
`[0, 1, 0],`
`[0, 0, 1]]`
	`[[2, 0, 0, 0],`
	`[0, 1, 0, 0],`
	`[0, 0, 1, 0]]`
		`[[1, 0, 0],`
		`[0, 1, 0],`
		`[0, 0, 1]]`
			`[[1, 0, 0],`
			`[0, 1, 0, 0],`
			`[0, 0, 1]]`

Exemplo (Matrizes)

`[[2, 0, 0],`
`[0, 1, 0],`
`[0, 0, 1]]`
	`[[2, 0, 0, 0],`
	`[0, 1, 0, 0],`
	`[0, 0, 1, 0]]`
		`[[1, 0, 0],`
		`[0, 1, 0],`
		`[0, 0, 1]]`
			`[[1, 0, 0],`
			`[0, 1, 0, 0],`
			`[0, 0, 1]]`

Não garante P. `m[0][0] != 1`	Não garante P. Não é quadrada.	Garante P	Não garante P. Não é quadrada.

Exemplo (Matrizes)

– – –

- Em seguida, escreva a função verifica P(m) que recebe uma lista de listas de inteiros como parâmetro e retorna True se a matriz atende a propriedade P ou False caso contrário.

Exemplo (Matrizes)

– – –

- Em seguida, escreva a função verifica P(m) que recebe uma lista de listas de inteiros como parâmetro e retorna True se a matriz atende a propriedade P ou False caso contrário.

```
def verifica_P(m) :
    n = len(m)
    for i in range(n) :
        if len(m[i]) != n :
            return False
        for j in range(n) :
            if i == j and m[i][j] != 1 :
                return False
            if i != j and m[i][j] != 0 :
                return False
    return True
```

Exemplo (Funções)

- Rafaela começou a estudar o algoritmo de ordenação QuickSort. Antes de analisar as chamadas recursivas, ela vai explorar o algoritmo que posiciona corretamente o pivô a cada passo.
- O elemento escolhido para ser o pivô pode ser, por exemplo, o primeiro elemento do vetor. Após a execução da função particiona, à esquerda do pivô ficarão os elementos com valores menores do que o valor do pivô e à direita do pivô ficarão os elementos com valores maiores. Veja o exemplo a seguir:

12	14	6	7	18	2	21

7	2	6	**12**	18	14	21

Exemplo (Funções)

- Observe o código abaixo, em que a partição pode atuar na lista inteira ou em uma sublista, de acordo com os valores dos índices *e* e *d* passados como parâmetro para a função. Para acompanhar os passos, Rafaela introduziu algumas chamadas ao comando print em pontos estratégicos, logo após as trocas dos elementos.

```
def particiona(lista, e, d) :
    valor_pivo = lista[e]
    pos_pivo = e
    e += 1
    print("e =", e, "d =", d, lista)
    while e < d :
        while e <= d and valor_pivo >= lista[e] :
            e += 1
        while e <= d and valor_pivo <= lista[d] :
            d -= 1
        if e > d :
            break
        lista[e], lista[d] = lista[d], lista[e]
        print("e =", e, "d =", d, lista)
    lista[pos_pivo], lista[d] = lista[d], lista[pos_pivo]
    print("e =", e, "d =", d, lista)
    return d
```

Exemplo (Funções)

- Para a lista e as chamadas abaixo indique o resultado dos comandos print seguindo o modelo. Ao final, indique o valor das variáveis ponto1 e ponto2.
- Execução 0

```
lista = [22, 10, 6, 30, 19, 2, 27]
ponto1 = particiona(lista, 0, len(lista) - 1)
ponto2 = particiona(lista, 0, ponto1 - 1)
```

e = 1 d = 6 [22 , 10 , 6 , 30 , 19 , 2 , 27]

Exemplo (Funções)

- Execução 1

```
def particiona(lista, e, d) :
    valor_pivo = lista[e]
    pos_pivo = e
    e += 1
    print("e =", e, "d =", d, lista)
    while e < d :
        while e <= d and valor_pivo >= lista[e] :
            e += 1
        while e <= d and valor_pivo <= lista[d] :
            d -= 1
        if e > d :
            break
        lista[e], lista[d] = lista[d], lista[e]
        print("e =", e, "d =", d, lista)
    lista[pos_pivo], lista[d] = lista[d], lista[pos_pivo]
    print("e =", e, "d =", d, lista)
    return d
```

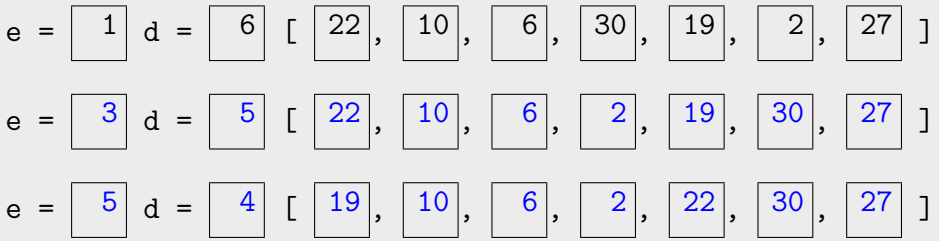
e = 1 d = 6 [22, 10, 6, 30, 19, 2, 27]

e = 3 d = 5 [22, 10, 6, 2, 19, 30, 27]

Exemplo (Funções)

- Execução 2

```
def particiona(lista, e, d) :
    valor_pivo = lista[e]
    pos_pivo = e
    e += 1
    print("e =", e, "d =", d, lista)
    while e < d :
        while e <= d and valor_pivo >= lista[e] :
            e += 1
        while e <= d and valor_pivo <= lista[d] :
            d -= 1
        if e > d :
            break
        lista[e], lista[d] = lista[d], lista[e]
        print("e =", e, "d =", d, lista)
    lista[pos_pivo], lista[d] = lista[d], lista[pos_pivo]
    print("e =", e, "d =", d, lista)
    return d
```



Exemplo (Funções)

- Execução 3

```
def particiona(lista, e, d) :
    valor_pivo = lista[e]
    pos_pivo = e
    e += 1
    print("e =", e, "d =", d, lista)
    while e < d :
        while e <= d and valor_pivo >= lista[e] :
            e += 1
        while e <= d and valor_pivo <= lista[d] :
            d -= 1
        if e > d :
            break
        lista[e], lista[d] = lista[d], lista[e]
        print("e =", e, "d =", d, lista)
    lista[pos_pivo], lista[d] = lista[d], lista[pos_pivo]
    print("e =", e, "d =", d, lista)
    return d
```

e = 1 d = 6 [22, 10, 6, 30, 19, 2, 27]

e = 3 d = 5 [22, 10, 6, 2, 19, 30, 27]

e = 5 d = 4 [19, 10, 6, 2, 22, 30, 27]

e = 1 d = 3 [19, 10, 6, 2, 22, 30, 27]

Exemplo (Funções)

- Execução 4

```
def particiona(lista, e, d) :
    valor_pivo = lista[e]
    pos_pivo = e
    e += 1
    print("e =", e, "d =", d, lista)
    while e < d :
        while e <= d and valor_pivo >= lista[e] :
            e += 1
        while e <= d and valor_pivo <= lista[d] :
            d -= 1
        if e > d :
            break
        lista[e], lista[d] = lista[d], lista[e]
        print("e =", e, "d =", d, lista)
    lista[pos_pivo], lista[d] = lista[d], lista[pos_pivo]
    print("e =", e, "d =", d, lista)
    return d
```

Exemplo (Recursividade)

– – –

- João agora está estudando recursão e elaborou nova série de testes. Como na questão 1, você deve escrever a saída do programa ou indicar “Nada será escrito”. Caso algum erro seja encontrado, indique o motivo e marque na coluna da esquerda a linha em que ele ocorre. Você também deve indicar claramente se o programa tiver entrado em loop.

Exemplo (Recursividade)

Programa	O que será escrito

def rec(n):
  print("n =", n)
  if n == 1 or n == 0:
  return n
  rec(n-3)
rec(9)
	n = 9
	n = 6
	n = 3
	n = 0

def rec(n):
  if (n >= 2):
  return n
  r = rec(n+1) + rec(n+2)
  print ("n =", n, "r =", r)
  return r
rec(0)
	n = 1 r = 5
	n = 0 r = 7

def rec(n):
  if n < 2 :
  n += 1
  return n + rec (n+1)
rec(0)
	Nada será escrito
	Loop infinito (comentário sobre estouro de pilha não é obrigatório)

Exemplo (Recursividade)

- João também fez testes para comparar versões recursivas e iterativas (não recursivas) dos mesmos algoritmos. Seguindo esta ideia, complete a tabela abaixo, escrevendo a versão iterativa do código à esquerda.

Versão recursiva	Versão iterativa

`def rec(n):`
`  print(n)`
`  if n == 0:`
`  return 0`
`  else:`
`  return n + rec(n-1)`
	`def iterativa(n) :`
	`  aux = 0`
	`  while (n >= 0) :`
	`  print(n)`
	`  aux += n`
	`  n = n-1`
	`  return aux`
	ou
	`def iterativa(n) :`
	`  aux = 0`
	`  for i in range(n,-1,-1) :`
	`  print(i)`
	`  aux += i`
	`  return aux`

Exercícios

II revisão ...

Exercício (A18: cópia de listas)

Ref. A28, ex3_clickers_A16-Q34.py, cópia de listas via atribuição **`lista_a = lista_b`**

Em Python, uma cópia de lista via atribuição simples apenas gera uma referência à mesma estrutura em memória. Assim, qualquer modificação à **`lista_a`**, gera a mesma modificação na **`lista_b`**

para gerar cópias independentes, use

1. Slicing> `b = a[:]`

2. Copia via `list`

```
a = [0,1,2]
b =  list(a)
```

3. Via método copy

```
a = [0, 1, 2]
b = a.copy()
```

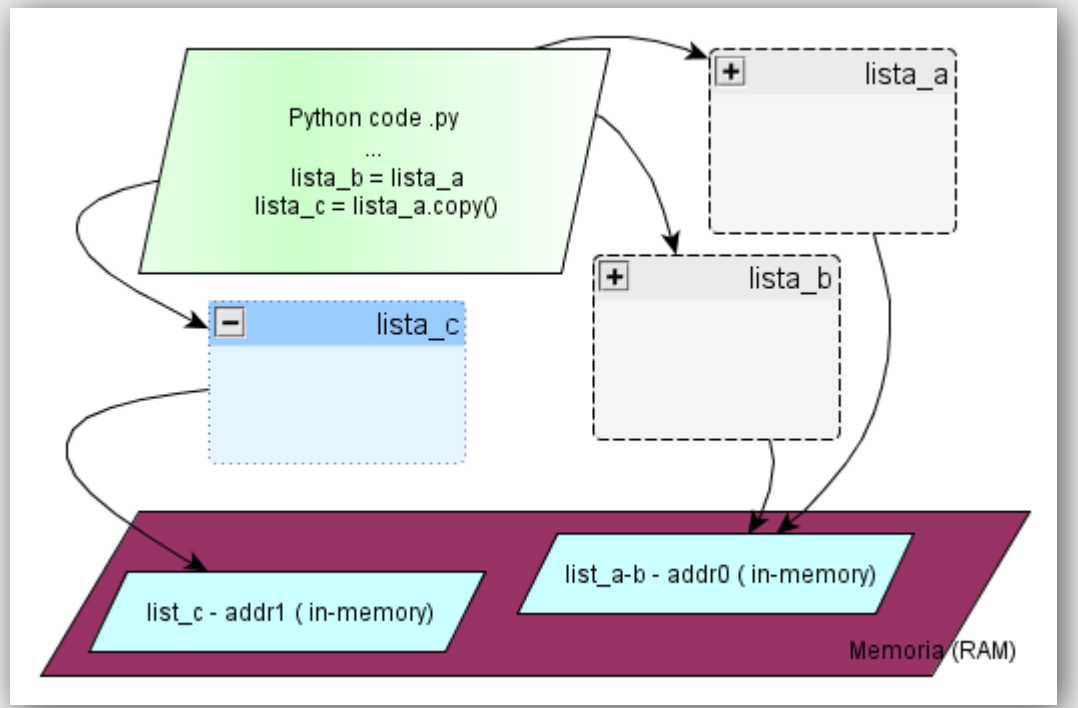
```
#copia de listas via atribuição simples
def adicionaElem(lista,elem):
    lista.append(elem); return lista
# main
lista_a = [20,30]
lista_b = lista_a
adicionaElem(lista_b, 40)
lista_b[0]=-1;lista_b[1]=-2;lista_b[0]=-3
print("la: ",lista_a); print("lb: ",lista_b)
>> la: [-1, -2, -3]
>> lb: [-1, -2, -3]
```

Cópia de listas em Python

```
# copia de listas via
# 1. atribuição simples
lista_a = [0,1,2]
lista_b = lista_a
# as 2 listas referenciam a
# mesma posição de memória

# 2. cópia via método copy
# são geradas cópias
# independentes, localizadas
# em posições de memória
# diferentes (shallow copy)

lista_c = lista_a.copy()
```



Exercício (A15: Tuplas, Dicionários)

definindo tuplas; tupla1 = ('setembro', 26, 9, 2018); tupla2 = (1, 2, 3, 4, 5, 6, 7)

print (...); print(len(tupla1)); print(tupla1[1:3]), ... ;

```
1. dados da tupla 1.
length of 'tupla1' is : 4
tupla1 is : ('setembro', 26, 9, 2018)

2. dados da tupla 2.
length of 'tupla2' is : 7
tupla2 is : (1, 2, 3, 4, 5, 6, 7)
```

```
#definindo lista de tuplas
# ex.
lista_de_tuplas=[ ];
lista_de_tuplas.append((18,20))
lista_de_tuplas.append((novembro,25))
print(lista_de_tuplas)

> [(18,20),(‘setembro’,25)]
```

Exercício (A15: Tuplas, Dicionários)

```
# 1. definindo dicionário; e percorrendo as chaves na sequencia
RA = {“Liz”: 229874, “Hugo”:215793,”Sofia”:199745}
for i in RA:
    print (i)
1.  Sofia, Liz, Hugo
# 2. Verificando se há chaves no dic.
A1 = “Sofia” in RA
A2 = “Aline in RA
print(A1, A2)  >  True, False
print(RA[“Hugo”])  > 215739
```

```
# verificando pares (KEY, VALUE), get
# uso : get(key)
# ex.percorrendo os pares do dicion.
RA.get(“Hugo”)  >  215793

for i, j in RA.items():
    print(i,j,sep=‘ ‘)

# saida do loop acima
```

- Sofia 199745
- Liz 229874
- Hugo 215793

Exemplos (A16: Funções)

Há duas formas ou estruturas para uso e definição de funções em Python

```
#1.  funçoes declaradas antes do seu uso no programa principal
def funcao_nome(parametros ou argumentos):
    corpo/comandos da função
    return <valor/variavel de retorno)
# programa principal  (aqui e onde a funcao_nome eh invocada)
  funcao_nome(x, y, …)
#2. funções declaradas posteriormente, a função main é declarada no inicio
def main()
def funcao_f1(parametros da funçao f1)
def funcao_f2 (parametros da funcao f2)
# programa principal (aqui eh invocada a funcao progrma principal > main)
main()
```

Exemplos (A16: funções)

Revisar exercício sld. 30 e 31, present. Aula 16

```
def leNota(num):
    notas = []
    ….
    return
def calculaMedia(notas):
    soma = 0
    ….
    return
# comandos do programa principal
n=int(input("digitar numero de notas"))
notas = leNota(n)
Media = calculaMedia(notas)
```

```
# funcao 0. def. programa principal
def main():
    n = int(input("digitar notas"))
    notas = leNota(n)
    media = calculaMedia(notas)
    ....
# funcao 1. def. leitura de notas
def leNota(num):
    notas = [ ]
    ....
# funcao 2. def. calcula media
def calculaMedia(notas):
    soma = 0
    ....
# programa principal (execucao)
main()
```

Exemplos (A17: funções, var. globais e locais)

```
# definicao e uso de variaveis globais; escopo: todo o programa
def leNota(num):
x=0 # variavel x local a func. leNota
    notas = [x]
    ….
def calculaM(notas):
y=1 # variavel y local a func. calculaM
    soma = y
    ….
# programa principal, var. global x e z
x=4;z=2 # x e z: globais, criadas fora funcões
notas = leNota(z)
Media = calculaM(notas)
print(x) # sera impresso valor 4 (global)
```

```
# funcao 0. def. programa principal
def main():
# aqui z esta restrita a funcao main
# x eh uma variavel global
    x=5; z = int(input("digitar notas"))
    notas = leNota(z)
    media = calculaMedia(notas)
    ....
# funcao 1. def. leitura de notas
def leNota(num):
    global x
    notas = [ ]
    x=0  # x eh global ...
# funcao 2. def. calcula media
def calculaMedia(notas):
    soma = 0
    ....
# programa principal (execucao)
main()    > imprime x = 0 !!
```

A18: Matrizes e Vetores multi-dimensionais

- Inicialização de uma matriz de dimensões *l* x *c* vazia utilizando listas.
- Exemplo de uma matriz 3 x 4 (com elementos inicialmente vazios):

```
mat = [[] for i in range(3)]
#dentro da lista externa criam-se 3 listas vazias []
mat
[[], [], []]
```

- Assim, em “mat” , cada lista interna representa uma linha da matriz i.e.,
  - `[[a01,a02,a03], [a11, a12, a13], [a21,a22,a23]]`

A18: Matriz e Vetores multi-dimensionais

- Ex. Criar matriz 3 x 4 onde cada posição (*i* , *j*) contém o valor de *i* * *j*.

	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	4	6

A18: Matriz e Vetores multi-dimensionais

- Criar matriz 3 x 4 onde cada posição (*i* , *j*) contém o valor de *i* * *j*.

```
mat = []
for i in range(3): # para cada linha de 0 ate 2
    l = [] # linha começa vazia
    for j in range(4): # para cada coluna de 0 ate 3
        l.append(i*j) # preenche colunas da linha I
    mat.append(l) # adiciona linha na matriz
print(mat)
```

```
[[0, 0, 0, 0], [0, 1, 2, 3], [0, 2, 4, 6]]
```

Exemplo de Declaração de Matriz

- Obtendo o mesmo resultado utilizando compreensão de listas:

```
mat = [[i*j for j in range(4)] for i in range(3)]
print(mat)
```

```
[[0, 0, 0, 0], [0, 1, 2, 3], [0, 2, 4, 6]]
```

Exercício (A18: Matrizes e Vetores multi-D)

Uso de bibliotecas NumPy (Python)

```
# Ex. Array 3x3
import numpy as np
b = np.array([[1,2,3],[4,5,6]])
print(b.ndim)
print(b.size) print(b)

# re-arrange/reshape
a= no.arange(10)
print(a)
A - np.arange(10.reshape(2,5
print(a)
```

Uso de bibliotecas NumPy (Python

Se Mi é uma matriz n-dimensional)

Ex.mult. K0*M0, M1+M2,...

```
# uso de numpy
import numpy as np
a = np.array([[1,2],[4,5]])
print(a.ndim)
b = np.array([[7,8],[2,0]])
c = np.dot(a,b)
c= a*b          [[11  8]
(matricial)      [38 32]]
```

Exemplos RegEx

ref. apresentação orig. Aula 28

Exemplos Algoritmos de Ordenação
BubbleSort
InsertionSort

ref. apresentação orig. em Aula 28

Aula 20

Ordenação:

- Bubble Sort

lista = [3,2,9,7,5,1,8,4]

[A20-Q7] Qual será o conteúdo da lista após uma iteração do algoritmo Bubble Sort?

Índice *i* percorre todas as posições de 0 a tam-2,
trocando lista[i+1] com lista[i] se
lista[i] > lista [i+1]

lista = [3,2,9,7,5,1,8,4]

[A20-Q7] Qual será o conteúdo da lista após uma iteração do algoritmo Bubble Sort?

Índice *i* percorre todas as posições de 0 a tam-2, trocando lista[i+1] com lista[i] se lista[i] > lista [i+1]

A

lista = [3,2,9,7,5,1,8,4]

B

lista = [2,3,7,5,1,8,4,9]

C

lista = [1,2,3,4,5,7,8,9]

D

lista = [2,3,5,7,1,8,4,9]

E

lista = [1,2,3,7,5,8,4,9]

lista = [3,2,9,7,5,1,8,4]

[A20-Q7] Qual será o conteúdo da lista após uma iteração do algorítmo Bubble Sort?

Índice *i* percorre todas as posições de 0 a tam-2, trocando lista[i+1] com lista[i] se lista[i] > lista [i+1]

A

lista = [3,2,9,7,5,1,8,4]

B

lista = [2,3,7,5,1,8,4,9]

C

lista = [1,2,3,4,5,7,8,9]

D

lista = [2,3,5,7,1,8,4,9]

E

lista = [1,2,3,7,5,8,4,9]

lista =
[25,20,15,10,3,2,30,35,39,40,55,9,7,60,80,5,1,8,4]

[A20-Q9] Qual será o conteúdo das 4 últimas posições da lista após 4 iterações do Bubble Sort?

lista = [25,20,15,10,3,2,30,35,39,40,55,9,7,60,80,5,1,8,4]

[A20-Q9] Qual será o conteúdo das 4 últimas posições da lista após 4 iterações do Bubble Sort?

A

lista = […,5,1,8,80]

B

lista = […, 50,55,60,80]

C

lista = […,40,55,60,80]

D

lista = […,39,40,55,60]

D

lista = […,55,40,60,80]

lista =
[25,20,15,10,3,2,30,35,39,40,55,9,7,60,80,5,1,8,4]

[A20-Q9] Qual será o conteúdo das 4 últimas posições da lista após 4 iterações do Bubble Sort?

A

lista = […,5,1,8,80]

B

lista = […, 50,55,60,80]

C

lista = […,40,55,60,80]

D

lista = […,39,40,55,60]

D

lista = […,55,40,60,80]

Aula 21

Ordenação:

- Insertion Sort

[A21-Q1] Este programa implementa o Insertion Sort. Quais linhas **obrigatoriamente** devem ser removidas para que o programa funcione como esperado?

a21_e01.py (no symbol selected)

```
1  lista = [7, 8, 5, 2, 4, 6, 3]
2  i = 1
3  for i in range(1, len(lista)):
4      elem = lista[i]
5      local = i
6      while (local != 0 and lista[local-1] > elem):
7          lista[local] = lista[local-1]
8          local = local - 1
9      lista[local] = elem
10     i = i + 1
11 print("Lista ordenada = ", lista)
```

```
1  lista = [7, 8, 5, 2, 4, 6, 3]
2  i = 1
3  for i in range(1, len(lista)):
4      elem = lista[i]
5      local = i
6      while (local != 0 and lista[local-1] > elem):
7          lista[local] = lista[local-1]
8          local = local - 1
9      lista[local] = elem
10     i = i + 1
11 print("Lista ordenada = ", lista)
```

A

Nenhuma linha. O programa funciona corretamente e assim.

B

Linha 2

C

Linha 10

D

Linha 2 e Linha 10

E

Linha 8

```
1  lista = [7, 8, 5, 2, 4, 6, 3]
2  i = 1
3  for i in range(1, len(lista)):
4      elem = lista[i]
5      local = i
6      while (local != 0 and lista[local-1] > elem):
7          lista[local] = lista[local-1]
8          local = local - 1
9      lista[local] = elem
10     i = i + 1
11 print("Lista ordenada = ", lista)
```

A

Nenhuma linha. O programa funciona corretamente assim.

B

Linha 2

C

Linha 10

D

Linha 2 e Linha 10

E

Linha 8

[A21-Q4] O que será impresso pelo programa abaixo?

a21_e04.py (no symbol selected)

```
1  lista = [7, 8, 5, 2, 4, 6, 3]
2  deslocamento = 0
3  for i in range(1, len(lista)):
4      elem = lista[i]
5      local = i
6      while (local != 0 and lista[local-1] > elem):
7          deslocamento += 1
8          lista[local] = lista[local-1]
9          local = local - 1
10     lista[local] = elem
11 print(deslocamento)
```

Implementação correta do Insertion Sort.

a21_e04.py (no symbol selected)

```
1  lista = [7, 8, 5, 2, 4, 6, 3]
2  deslocamento = 0
3  for i in range(1, len(lista)):
4      elem = lista[i]
5      local = i
6      while (local != 0 and lista[local-1] > elem):
7          deslocamento += 1
8          lista[local] = lista[local-1]
9          local = local - 1
10     lista[local] = elem
11 print(deslocamento)
```

A	B	C	D	E
10	13	15	17	20

a21_e04.py (no symbol selected)

```
1  lista = [7, 8, 5, 2, 4, 6, 3]
2  deslocamento = 0
3  for i in range(1, len(lista)):
4      elem = lista[i]
5      local = i
6      while (local != 0 and lista[local-1] > elem):
7          deslocamento += 1
8          lista[local] = lista[local-1]
9          local = local - 1
10     lista[local] = elem
11 print(deslocamento)
```

A	B	C	D	E
10	13	15	17	20

Exemplos Algoritmos de Busca
Sequencial
Binária

ref. apresentação orig. em Aula 28

Aula 22

Busca:

- Sequencial
- Binária

[A22-Q1] O que será impresso pelo programa abaixo?

a22_q1.py (no symbol selected)

```
1  def buscaSequencial(lista, chave):
2      for i in range(len(lista)):
3          if lista[i] == chave:
4              return i
5      #Só executa essa linha se não achou a chave!
6      return -1
7
8  chave = -1
9  lista = [0, -1, 5, 2, 4, 6, 3]
10 pos = buscaSequencial(lista, chave)
11 if (pos == -1):
12     print ("Não")
13 else     :
14     print ("Pos = ", pos)
```

a22_q1.py (no symbol selected)

```
1  def buscaSequencial(lista, chave):
2      for i in range(len(lista)):
3          if lista[i] == chave:
4              return i
5      #Só executa essa linha se não achou a chave!
6      return -1
7
8  chave = -1
9  lista = [0, -1, 5, 2, 4, 6, 3]
10 pos = buscaSequencial(lista, chave)
11 if (pos == -1):
12     print ("Não")
13 else    :
14     print ("Pos = ", pos)
```

A	B	C	D	E
Não irá compilar.	Não	Pos = 0	Pos = -1	Pos = 1

a22_q1.py (no symbol selected)

```
1  def buscaSequencial(lista, chave):
2      for i in range(len(lista)):
3          if lista[i] == chave:
4              return i
5      #Só executa essa linha se não achou a chave!
6      return -1
7
8  chave = -1
9  lista = [0, -1, 5, 2, 4, 6, 3]
10 pos = buscaSequencial(lista, chave)
11 if (pos == -1):
12     print ("Não")
13 else    :
14     print ("Pos = ", pos)
```

A	B	C	D	E
Não irá compilar.	Não	Pos = 0	Pos = -1	Pos = 1

[A22-Q9] Qual o valor da chave para que o programa seja executado conforme abaixo?


```
23  lista = [2, 5, 6, 8, 10, 12]
24  chave = 
25  pos = buscaBinaria (lista, chave)
26  if (pos == -1):
27      print ("Chave ", chave, " nao existe")
28  else:
29      print ("Pos = ", pos)
```

```
wifi-177-220-85-182:Downloads Ricardo$ python3 binary.py
inicio =  0  fim =  5 meio =  2
inicio =  3  fim =  5 meio =  4
Pos =  4
wifi-177-220-85-182:Downloads Ricardo$
```

[A22-Q9] Qual o valor da chave para que o programa seja executado conforme abaixo?


```
23  lista = [2, 5, 6, 8, 10, 12]
24  chave = 
25  pos = buscaBinaria (lista, chave)
26  if (pos == -1):
27      print ("Chave ", chave, " nao existe")
28  else:
29      print ("Pos = ", pos)
```



```
wifi-177-220-85-182:Downloads Ricardo$ python3 binary.py
inicio =  0  fim =  5 meio =  2
inicio =  3  fim =  5 meio =  4
Pos =  4
wifi-177-220-85-182:Downloads Ricardo$
```

A	B	B	D	E
5	6	8	10	12

[A22-Q9] Qual o valor da chave para que o programa seja executado conforme abaixo?


```
23  lista = [2, 5, 6, 8, 10, 12]
24  chave = 
25  pos = buscaBinaria (lista, chave)
26  if (pos == -1):
27      print ("Chave ", chave, " nao existe")
28  else:
29      print ("Pos = ", pos)
```

```
wifi-177-220-85-182:Downloads Ricardo$ python3 binary.py
inicio =  0  fim =  5 meio =  2
inicio =  3  fim =  5 meio =  4
Pos =  4
wifi-177-220-85-182:Downloads Ricardo$
```

A	B	B	D	E
5	6	8	10	12

Exemplos Algoritmos de Ordenação
MergeSort
QuickSort

ref. apresentação orig. em Aula 28

Aula 27

Merge Sort

[A27-Q2] Qual o resultado da fusão (ordenação por intercalação) realizada com as listas abaixo?

2	5	7	11

-1	2	2	2	30

[A27-Q2] Qual o resultado da fusão (ordenação por intercalação) realizada com as listas abaixo?

2	5	7	11

-1	2	2	2	30

A	B	C	D	E
Não é possível fazer a fusão.	-1 2 2 2 30	-1 2 2 2 2 11 30	-1 2 2 2 5 7 11 30	-1 2 2 2 2 5 7 11 30

[A27-Q2] Qual o resultado da fusão (ordenação por intercalação) realizada com as listas abaixo?

2	5	7	11

-1	2	2	2	30

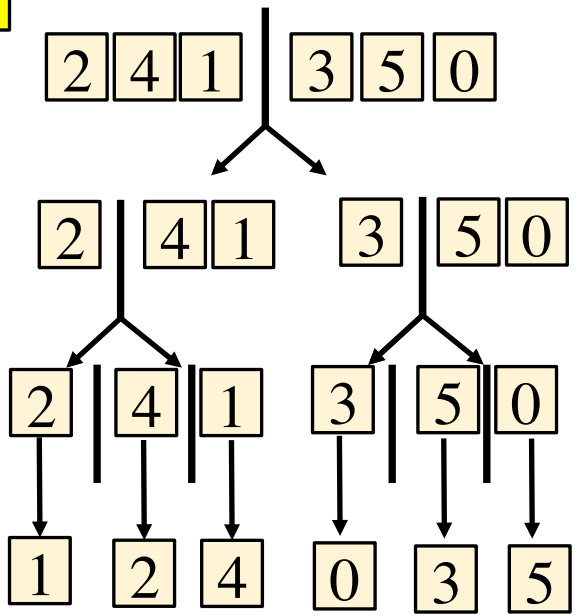
A	B	C	D	E
Não é possível fazer a fusão.	-1 2 2 2 30	-1 2 2 2 2 11 30	-1 2 2 2 5 7 11 30	-1 2 2 2 2 5 7 11 30

[A27-Q3] Considere a seguinte lista. Qual a árvore gerada considerando-se apenas a fase de Divisão e Chamada Recursiva?

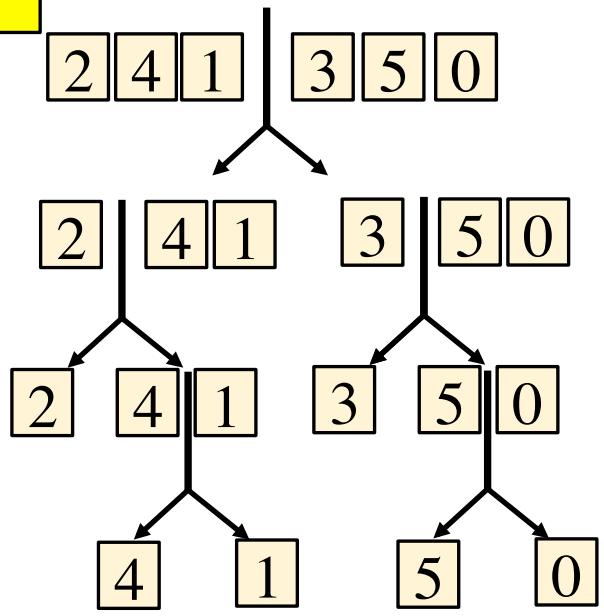
2	4	1	3	5	0

[A27-Q3] Considere a seguinte lista. Qual a árvore gerada considerando-se apenas a fase de Divisão e Chamada Recursiva?

A

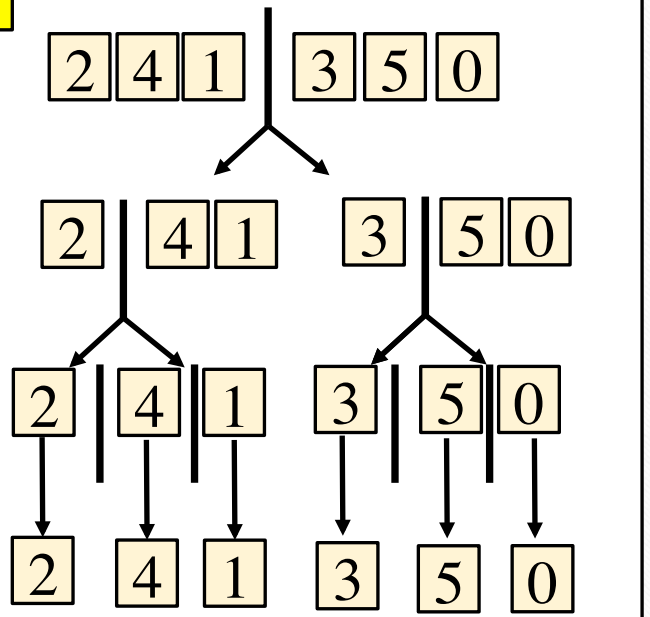


B

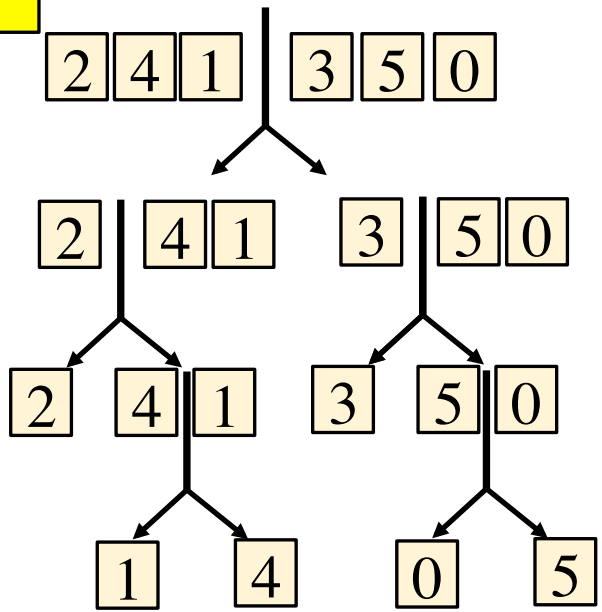


[A27-Q3] Considere a seguinte lista. Qual a árvore gerada considerando-se apenas a fase de Divisão e Chamada Recursiva?

C



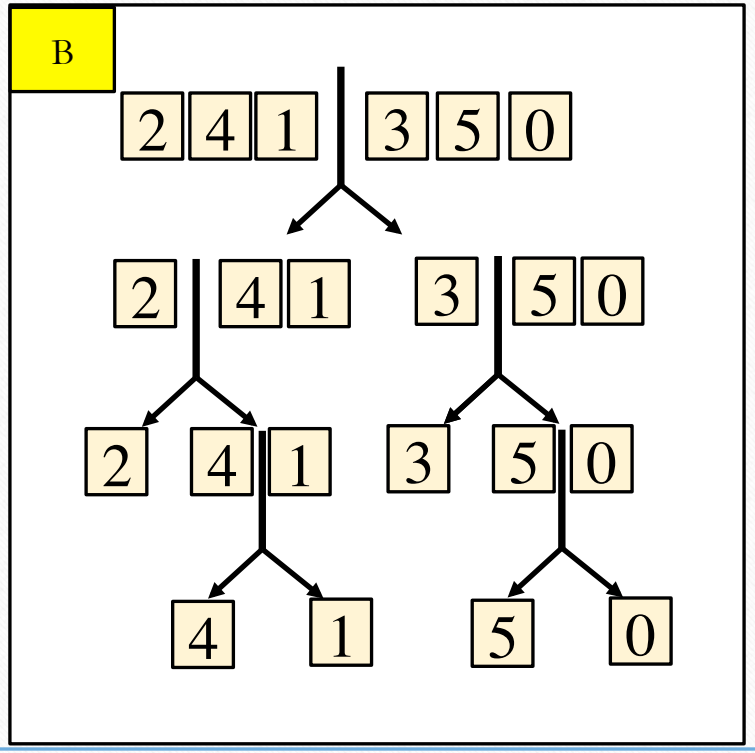
D



E Nenhuma das alternativas anteriores.

[A27-Q3] Revelação:

B



Aula 28

Quick sort

[A28-Q1] Considere a seguinte lista. Qual será a lista após a chamada da função *particiona,* e qual será o valor retornado?

[A28-Q1] Considere a seguinte lista. Qual será a lista após a chamada da função *particiona,* e qual será o valor retornado?

8	7	1	2	4	9	3	5	6

A
8 7 1 2 4 9 3 5 6
Retornará 6

B
5 1 2 4 3 8 9 7 6
Retornará 5

C
1 2 3 4 5 6 7 8
Retornará 5

D
6 5 1 2 4 3 9 7 8
Retornará 6

E
9 1 2 4 3 8 5 7 6
Retornará 5

[A28-Q1] Considere a seguinte lista. Qual será a lista após a chamada da função *particiona,* e qual será o valor retornado?

8	7	1	2	4	9	3	5	6

A
8 7 1 2 4 9 3 5 6
Retornará 6

B
5 1 2 4 3 8 9 7 6
Retornará 5

C
1 2 3 4 5 6 7 8
Retornará 5

6 5 1 2 4 3 9 7 8
Retornará 6

E
5 1 2 4 3 8 9 7 6
Retornará 6

[A28-Q2] Considere a seguinte lista. Qual será a lista após a chamada da função *particiona,* e qual será o valor retornado?

9	0	8	1	7	2	6	3	4

A
4 0 3 1 2 7 6 8 9
Retornará 4

B
4 0 3 1 2 7 6 8 9
Retornará 5

C
0 4 1 3 2 7 6 9 8
Retornará 4

D
0 4 1 3 2 7 6 9 8
Retornará 5

E
0 4 1 3 2 7 6 9 8
Retornará 6

[A28-Q2] Considere a seguinte lista. Qual será a lista após a chamada da função *particiona,* e qual será o valor retornado?

9	0	8	1	7	2	6	3	4

A
4 0 3 1 2 7 6 8 9
Retornará 4

B
4 0 3 1 2 7 6 8 9
Retornará 5

C
0 4 1 3 2 7 6 9 8
Retornará 4

D
0 4 1 3 2 7 6 9 8
Retornará 5

E
0 4 1 3 2 7 6 9 8
Retornará 6

[A28-Q2] Considere a seguinte lista. Qual será a lista após a chamada da função *particiona,* e qual será o valor retornado?

9	0	8	1	7	2	6	3	4

A
4 0 3 1 2 7 6 8 9
Retornará 4

B
4 0 3 1 2 7 6 8 9
Retornará 5

C
0 4 1 3 2 7 6 9 8
Retornará 4

D
0 4 1 3 2 7 6 9 8
Retornará 5

E
0 4 1 3 2 7 6 9 8
Retornará 6

Referências & Exercícios

- https://http://www.algoritmosempython.com.br/capitulos/pesquisa-ordenacao/pesquisa-binaria
- https://panda.ime.usp.br/aulasPython/static/aulasPython/aula23.html
- http://interactivepython.org/runestone/static/pythonds/SortSearch/TheBubbleSort.html
- http://interactivepython.org/courselib/static/pythonds/SortSearch/TheInsertionSort.html
- http://interactivepython.org/courselib/static/pythonds/SortSearch/TheMergeSort.html
- http://interactivepython.org/courselib/static/pythonds/SortSearch/TheQuickSort.html

Apresentação original com exercícios interativos (aula 28), do prof. Ricardo Caceffo em :
https://ic.unicamp.br/~mjara.perez/mc102-z_html/mc102-z/Clickers_revisao-P2_Aula28.pdf

Tópicos principais
Listas, Tuplas, Dicionários, RegEx: Expressões Regulares,
Algoritmos de Busca Binaria e Sequencial
Ordenação: selection Sort, BubbleSort, Insertion Sort, MergeSort, QuickSort