UFMA
Universidade Federal do Maranhão
A VIDA É COMBATE

dted
DIRETORIA DE TECNOLOGIAS NA EDUCAÇÃO

Estrutura de dados: Ordenação

Sérgio Souza Costa

EAD PARA VOCÊ

Roteiro

1. Introdução
2. Aplicações
3. Categorias
4. Algoritmos

Roteiro

1. Introdução
2. Aplicações
3. Categorias
4. Algoritmos

Fonte: https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/toctree.html

Introdução

- Ordenação é o processo de colocar elementos de uma coleção em uma determinada ordem.

- Por exemplo:
  - uma lista de palavras poderia estar ordenada alfabeticamente ou por comprimento.
  - uma lista de cidades poderia estar ordenada por população, área ou CEP.

Roteiro

1. Introdução
2. Aplicações
3. Categorias
4. Algoritmos

Fonte: https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/toctree.html

Aplicações

Com dados ordenados, algumas aplicações se tornam mais simples:

- **Teste de unicidade**: verifica se todos os elementos de uma coleção são distintos.
- **Remoção de duplicatas**: remove eventuais duplicatas de uma coleção.
- **Busca**: conforme foi visto, operações de busca são bastante facilitadas quando os dados são ordenados.
- **Encontrar o i-ésimo** maior (ou menor) elemento de uma coleção
- **Contagem de freqüência** (moda): encontra o elemento que ocorre com mais freqüência numa coleção.
- **Encontrar** a **união** ou a **interseção** de dois conjuntos.

Roteiro

1. Introdução
2. Aplicações
3. Categorias
4. Algoritmos

Fonte: https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/toctree.html

Categorias

Podemos classificar os algoritmos por diferentes estratégias. De modo geral, podemos classificar em:

- Por troca como o BubbleSort (Bolha) e QuickSort
- Por seleção como o selection sort e heap sort
- Por inserção o insertion sort".
- Por intercalação como o denominado merge sort.

Fonte: https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/toctree.html

Ordenação por troca

Para entender esse algoritmos, podemos imaginar um baralho de cartas.

Para ordenar as cartas utilizando **troca**, espalhe-as voltadas para cima e então troque as cartas fora de ordem até que todo baralho esteja ordenado;

Fonte: https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/toctree.html

Ordenação por seleção

Para entender esse algoritmos, podemos imaginar um baralho de cartas.

Esses algoritmos partem do princípio de realizar o isolamento de elementos para posições ordenadas.

Utilizando **seleção**, espalhe as cartas na mesa, selecione a carta de menor valor, retire-a do baralho e coloque-a na mão. O processo continua e termina quando todas as cartas estiverem na sua mão;

Fonte: https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/toctree.html

Ordenação por Inserção

Para entender esse algoritmos, podemos imaginar um baralho de cartas.

Para ordenar as cartas por **inserção**, segure todas as cartas na mão. Ponha uma carta por vez na mesa, sempre a inserindo na posição correta.

Fonte: https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/toctree.html

Ordenação por intercalação

Para entender esse algoritmos, podemos imaginar um baralho de cartas.

Esses algoritmos partem do princípio de criar uma seqüência ordenada a partir de duas outras também ordenadas. Esse método é conhecido também como mistura ou *merge* em inglês.

Utilizando intercalação, divida o baralho em pares, até restar apenas 1, depois agrupe-as de forma a gerar um novo grupo ordenado, e assim por diante.

Roteiro

1. Introdução
2. Aplicações
3. Categorias
4. Algoritmos

Fonte: https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/toctree.html

Método da bolha (bubble sort)

- O bubble sort realiza múltiplas passagem por uma lista.
- Ele compara itens adjacentes e troca aqueles que estão fora de ordem. Cada passagem pela lista coloca o próximo maior valor na sua posição correta.

Fonte: https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/toctree.html

Método da bolha (bubble sort)

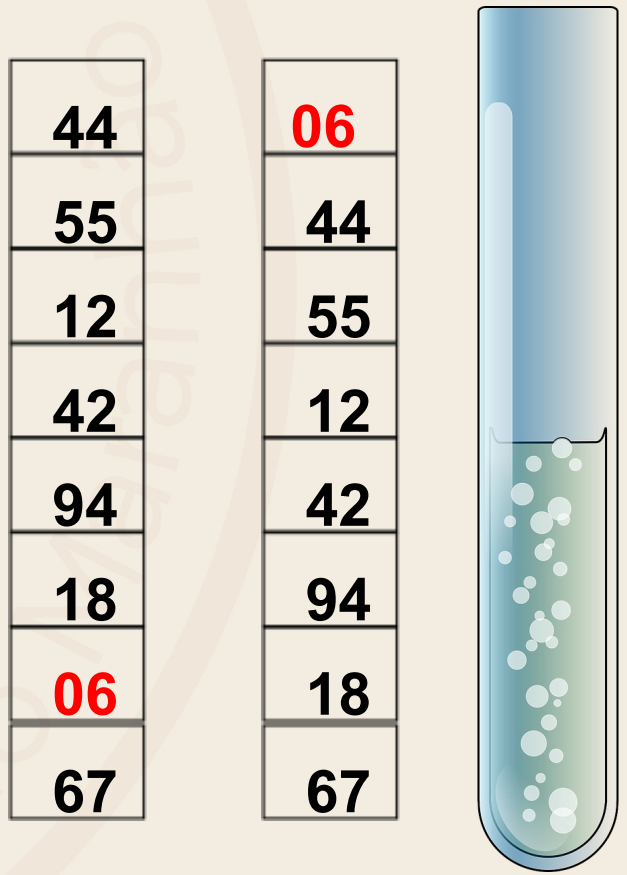
- Por que Bolha?

Fonte: https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/toctree.html

Método da bolha (bubble sort)

- Por que Bolha?

  Se a lista (ou vetor) a ser ordenado for colocado na vertical durante cada passo o menor elemento “sobe” até encontrar um elemento maior ainda, como se uma bolha subisse dentro de um tubo de acordo com sua densidade



Fonte: https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/toctree.html

Método da bolha (bubble sort)

Considerando essa lista inicial [25,48,37,12,57,86,33,92], vamos simular o primeiro passo:

[25,48,37,12,57,86,33,92]

Fonte: https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/toctree.html

Método da bolha (bubble sort)

Considerando essa lista inicial [25,48,37,12,57,86,33,92], vamos simular o primeiro passo:

[**25**,**48**,37,12,57,86,33,92] 25x48

Fonte: https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/toctree.html

Método da bolha (bubble sort)

Considerando essa lista inicial [25,48,37,12,57,86,33,92], vamos simular o primeiro passo:

[25,48,37,12,57,86,33,92] 25x48
[25,48,37,12,57,86,33,92]

Fonte: https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/toctree.html

Método da bolha (bubble sort)

Considerando essa lista inicial [25,48,37,12,57,86,33,92], vamos simular o primeiro passo:

[25,48,37,12,57,86,33,92] 25x48
[25,**48**,**37**,12,57,86,33,92] 48x37

Fonte: https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/toctree.html

Método da bolha (bubble sort)

Considerando essa lista inicial [25,48,37,12,57,86,33,92], vamos simular o primeiro passo:

[25,48,37,12,57,86,33,92]	25x48
[25,**48**,**37**,12,57,86,33,92]	48x37 troca

Fonte: https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/toctree.html

Método da bolha (bubble sort)

Considerando essa lista inicial [25,48,37,12,57,86,33,92], vamos simular o primeiro passo:

[25,48,37,12,57,86,33,92]	25x48
[25,48,37,12,57,86,33,92]	48x37 troca
[25,37,48,12,57,86,33,92]	

Fonte: https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/toctree.html

Método da bolha (bubble sort)

Considerando essa lista inicial [25,48,37,12,57,86,33,92], vamos simular o primeiro passo:

[25,48,37,12,57,86,33,92]	25x48
[25,48,37,12,57,86,33,92]	48x37 troca
[25,37,**48**,**12**,57,86,33,92]	48x12

Fonte: https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/toctree.html

Método da bolha (bubble sort)

Considerando essa lista inicial [25,48,37,12,57,86,33,92], vamos simular o primeiro passo:

[25,48,37,12,57,86,33,92]	25x48
[25,48,37,12,57,86,33,92]	48x37 troca
[25,37,**48**,**12**,57,86,33,92]	48x12 troca

Fonte: https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/toctree.html

Método da bolha (bubble sort)

Considerando essa lista inicial [25,48,37,12,57,86,33,92], vamos simular o primeiro passo:

[25,48,37,12,57,86,33,92]	25x48
[25,48,37,12,57,86,33,92]	48x37 troca
[25,37,48,12,57,86,33,92]	48x12 troca
[25,37,12,**48**,**57**,86,33,92]	48x57

Fonte: https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/toctree.html

Método da bolha (bubble sort)

Considerando essa lista inicial [25,48,37,12,57,86,33,92], vamos simular o primeiro passo:

[25,48,37,12,57,86,33,92]	25x48
[25,48,37,12,57,86,33,92]	48x37 troca
[25,37,48,12,57,86,33,92]	48x12 troca
[25,37,12,48,57,86,33,92]	48x57
[25,37,12,48,**57**,**86**,33,92]	57x86

Fonte: https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/toctree.html

Método da bolha (bubble sort)

Considerando essa lista inicial [25,48,37,12,57,86,33,92], vamos simular o primeiro passo:

[25,48,37,12,57,86,33,92]	25x48
[25,48,37,12,57,86,33,92]	48x37 troca
[25,37,48,12,57,86,33,92]	48x12 troca
[25,37,12,48,57,86,33,92]	48x57
[25,37,12,48,57,86,33,92]	57x86
[25,37,12,48,57,**86**,**33**,92]	86x33 troca

Fonte: https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/toctree.html

Método da bolha (bubble sort)

Considerando essa lista inicial [25,48,37,12,57,86,33,92], vamos simular o primeiro passo:

[25,48,37,12,57,86,33,92]	25x48
[25,48,37,12,57,86,33,92]	48x37 troca
[25,37,48,12,57,86,33,92]	48x12 troca
[25,37,12,48,57,86,33,92]	48x57
[25,37,12,48,57,86,33,92]	57x86
[25,37,12,48,57,86,33,92]	86x33 troca
[25,37,12,48,57,33,**86**,**92**]	86x92

Fonte: https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/toctree.html

Método da bolha (bubble sort)

Considerando essa lista inicial [25,48,37,12,57,86,33,92], vamos simular o primeiro passo:

[25,48,37,12,57,86,33,92]	25x48
[25,48,37,12,57,86,33,92]	48x37 troca
[25,37,48,12,57,86,33,92]	48x12 troca
[25,37,12,48,57,86,33,92]	48x57
[25,37,12,48,57,86,33,92]	57x86
[25,37,12,48,57,86,33,92]	86x33 troca
[25,37,12,48,57,33,86,92]	86x92
[25,37,12,48,57,33,86,92]	**final da primeira passada**

Fonte: https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/toctree.html

Método da bolha (bubble sort)

Considerando essa lista inicial [25,48,37,12,57,86,33,92], vamos simular o primeiro passo:

[25,48,37,12,57,86,33,92]	25x48
[25,48,37,12,57,86,33,92]	48x37 troca
[25,37,48,12,57,86,33,92]	48x12 troca
[25,37,12,48,57,86,33,92]	48x57
[25,37,12,48,57,86,33,92]	57x86
[25,37,12,48,57,86,33,92]	86x33 troca
[25,37,12,48,57,33,86,92]	86x92
[25,37,12,48,57,33,86,92]	**final da primeira passada**

Neste ponto, o maior elemento, 92, já está na sua posição final.

No próximo passo vou parar antes do ultimo elemento.

Fonte: https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/toctree.html

Método da bolha (bubble sort)

Para garantir que entenderam, vamos simular o segundo passo:

[25,37,12,48,57,33,86,92]

Fonte: https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/toctree.html

Método da bolha (bubble sort)

Para garantir que entenderam, vamos simular o segundo passo:

[**25**,**37**,12,48,57,33,86,92] 25x37

Fonte: https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/toctree.html

Método da bolha (bubble sort)

Para garantir que entenderam, vamos simular o segundo passo:

[25,37,12,48,57,33,86,92] 25x37
[25,37,12,48,57,33,86,92]

Fonte: https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/toctree.html

Método da bolha (bubble sort)

Para garantir que entenderam, vamos simular o segundo passo:

[25,37,12,48,57,33,86,92] 25x37
[25,**37**,**12**,48,57,33,86,92] 37x12

Fonte: https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/toctree.html

Método da bolha (bubble sort)

Para garantir que entenderam, vamos simular o segundo passo:

[25,37,12,48,57,33,86,92] 25x37
[25,**37**,**12**,48,57,33,86,92] 37x12 troca

Fonte: https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/toctree.html

Método da bolha (bubble sort)

Para garantir que entenderam, vamos simular o segundo passo:

[25,37,12,48,57,33,86,92]	25x37
[25,37,12,48,57,33,86,92]	37x12 troca
[25,12,**37**,**48**,57,33,86,92]	37x48

Fonte: https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/toctree.html

Método da bolha (bubble sort)

Para garantir que entenderam, vamos simular o segundo passo:

[25,37,12,48,57,33,86,92]	25x37
[25,37,12,48,57,33,86,92]	37x12 troca
[25,12,37,48,57,33,86,92]	37x48
[25,12,37,**48**,**57**,33,86,92]	48x57

Fonte: https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/toctree.html

Método da bolha (bubble sort)

Para garantir que entenderam, vamos simular o segundo passo:

[25,37,12,48,57,33,86,92]	25x37
[25,37,12,48,57,33,86,92]	37x12 troca
[25,12,37,48,57,33,86,92]	37x48
[25,12,37,48,57,33,86,92]	48x57
[25,12,37,48,**57**,**33**,86,92]	57x33 troca

Fonte: https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/toctree.html

Método da bolha (bubble sort)

Para garantir que entenderam, vamos simular o segundo passo:

[25,37,12,48,57,33,86,92]	25x37
[25,37,12,48,57,33,86,92]	37x12 troca
[25,12,37,48,57,33,86,92]	37x48
[25,12,37,48,57,33,86,92]	48x57
[25,12,37,48,57,33,86,92]	57x33 troca
[25,12,37,48,33,**57**,**86**,92]	57x86

Fonte: https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/toctree.html

Método da bolha (bubble sort)

Para garantir que entenderam, vamos simular o segundo passo:

[25,37,12,48,57,33,86,92]	25x37
[25,37,12,48,57,33,86,92]	37x12 troca
[25,12,37,48,57,33,86,92]	37x48
[25,12,37,48,57,33,86,92]	48x57
[25,12,37,48,57,33,86,92]	57x33 troca
[25,12,37,48,33,57,86,92]	57x86
[25,12,37,48,33,57,86,92]	**final do segundo passo**

Fonte: https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/toctree.html

Método da bolha (bubble sort)

Para garantir que entenderam, vamos simular o segundo passo:

[25,37,12,48,57,33,86,92]	25x37
[25,37,12,48,57,33,86,92]	37x12 troca
[25,12,37,48,57,33,86,92]	37x48
[25,12,37,48,57,33,86,92]	48x57
[25,12,37,48,57,33,86,92]	57x33 troca
[25,12,37,48,33,57,86,92]	57x86
[25,12,37,48,33,57,86,92]	**final do segundo passo**

Neste ponto, os elementos 86 e 92 já estão em sua posição correta.

Agora já podemos olhar o codigo.

Fonte: https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/toctree.html

Método da bolha (bubble sort)

Uma implementação possível

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

# testando
l = [54,26,93,17,77,31,44,55,20]
bubbleSort(l)
print(l)
```

Esse range, irá gerar a seguinte sequencia [8,7,6,5 ...]

Observe aqui a troca, em muitas linguagens seria necessário três passos e uma variável auxiliar.

https://replit.com/@sergio_costa/ordenacao#main.py

Fonte: https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/toctree.html

Método da bolha (bubble sort)

Uma implementação possível

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

# testando
l = [54,26,93,17,77,31,44,55,20
bubbleSort(l)
print(l)
```

https://replit.com/@sergio_costa/ordenacao

Esse range, irá gerar a seguinte sequencia [8,7,6,5 ...]

Uma boa dica é testar através do Python Tutor:

https://pythontutor.com/live.html#mode=edit

Observe aqui a troca, em muitas linguagens seria necessário três passos e uma variável auxiliar.

Fonte: https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/toctree.html

Ordenação por seleção (selection sort)

- A ordenação por seleção melhora o bubble sort ao realizar apenas uma troca a cada passagem pela lista.
- Para conseguir isso, uma ordenação por seleção procura pelo valor mais alto enquanto faz uma passagem e, depois de completá-la, coloca-o na posição correta.
- Assim como o bubble sort, depois da primeira passagem, o maior item sempre está na posição correta. Depois da segunda passagem, o próximo maior item também vai para a posição adequada ...

Fonte: https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/toctree.html

Ordenação por seleção (selection sort)

O	R	D	E	N	A

Estado inicial

Fonte: https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/toctree.html

Ordenação por seleção (selection sort)

O	R	D	E	N	A	Estado inicial
A	R	D	E	N	**O**	Fim do passo 1

Fonte: https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/toctree.html

Ordenação por seleção (selection sort)

O	R	D	E	N	A	Estado inicial
A	R	D	E	N	**O**	Fim do passo 1
A	**D**	**R**	E	N	O	Fim do passo 2

Fonte: https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/toctree.html

Ordenação por seleção (selection sort)

O	R	D	E	N	A	Estado inicial
A	R	D	E	N	**O**	Fim do passo 1
A	**D**	**R**	E	N	O	Fim do passo 2
A	D	**E**	**R**	N	O	Fim do passo 3

Fonte: https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/toctree.html

Ordenação por seleção (selection sort)

O	R	D	E	N	A	Estado inicial
A	R	D	E	N	**O**	Fim do passo 1
A	**D**	**R**	E	N	O	Fim do passo 2
A	D	**E**	**R**	N	O	Fim do passo 3
A	D	E	**N**	**R**	O	Fim do passo 4

Fonte: https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/toctree.html

Ordenação por seleção (selection sort)

O	R	D	E	N	A	Estado inicial
A	R	D	E	N	**O**	Fim do passo 1
A	**D**	**R**	E	N	O	Fim do passo 2
A	D	**E**	**R**	N	O	Fim do passo 3
A	D	E	**N**	**R**	O	Fim do passo 4
A	D	E	N	**O**	**R**	Fim do passo 5

Fonte: https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/toctree.html

Ordenação por seleção (selection sort)

Primeiro, podemos considerar uma função que descobre a posição do maior valor em uma dada lista:

```
def posicao_maior (l, fim):
  pos_maior = 0
  for atual in range(1,fim+1):
    if l[atual] > l[pos_maior]:
      pos_maior = atual
  return pos_maio
```

Esse parâmetro é usado para não percorremos toda a lista. Dado que a medida que vai sendo ordenada, não será necessário ir até o final.

https://replit.com/@sergio_costa/ordenacao#main.py

Fonte: https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/toctree.html

Ordenação por seleção (selection sort)

Agora usando a função definida previamente, o algoritmo fica mais fácil para ler.

```
def selectionSort(l):
    for i in range(len(l)-1,0,-1):
        pos_maior = posicao_maior(l, i)
        l[pos_maior], l[i] = l[i], l[pos_maior]

alist = [54,26,93,17,77,31,44,55,20]
selectionSort(alist)
print(alist)
```

Neste algoritmo só ocorre uma troca por passo.

https://replit.com/@sergio_costa/ordenacao#main.py

Fonte: https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/toctree.html

Ordenação por seleção (selection sort)

Agora, considere o algoritmo selection sort sendo aplicado a uma lista já ordenada.

Quantas trocas serão realizadas ? Ele fará nenhuma troca, número menor ou maior de trocas ?

Fonte: https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/toctree.html

Ordenação por seleção (selection sort)

Agora, considere o algoritmo selection sort sendo aplicado a uma lista já ordenada

Quantas trocas serão realizadas nenhuma troca, número menor trocas ?

Para responder, vocês podem testar pelo Python Tutor:

https://pythontutor.com/live.html#mode=edit

Fonte: https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/toctree.html

Ordenação por inserção (inserction sort)

A ordenação por inserção sempre mantém uma sublista ordenada nas posições inferiores da lista. Cada novo item é “inserido” na sublista anterior de modo que a sublista ordenada fique com um item a mais. Formalmente:

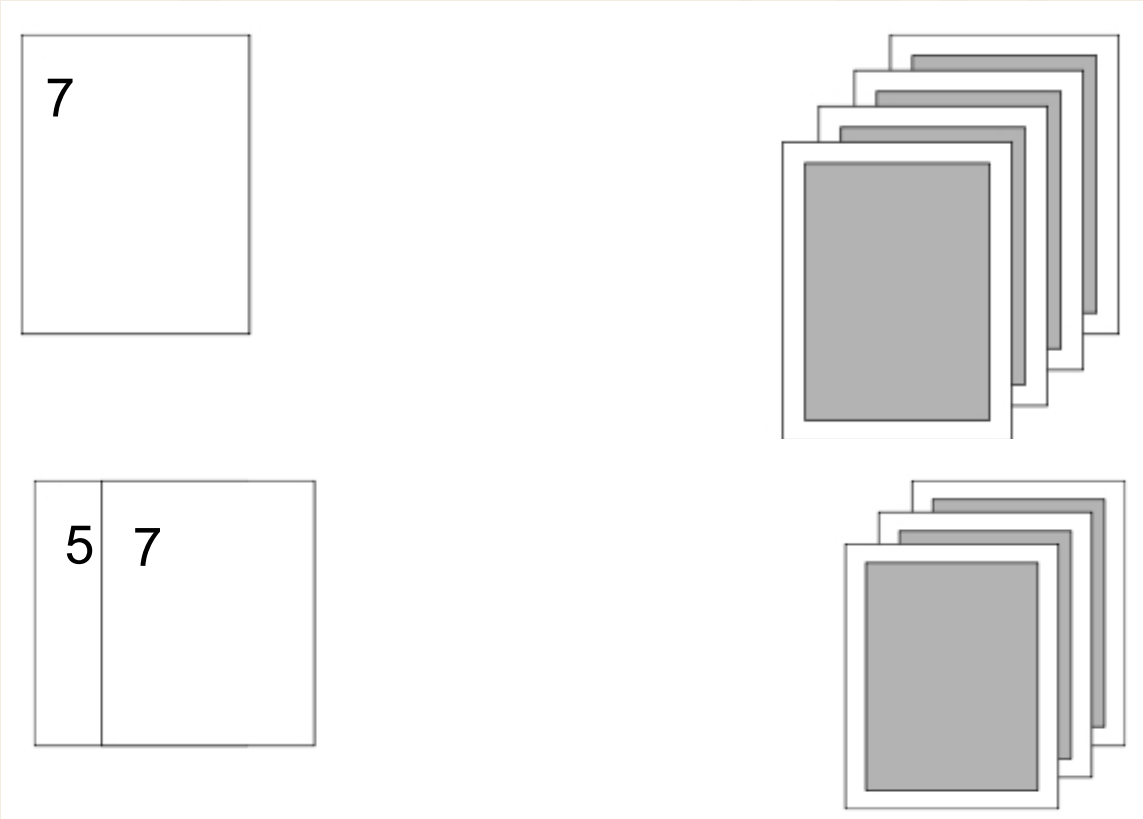
Os elementos são divididos em uma seqüência de destino a_1, ..., a_{i-1} e em uma seqüência fonte a_i, ..., a_n.

Em cada passo, a partir de i =2, o i-ésimo item da seqüência fonte é retirado e transferido para a seqüência destino sendo inserido na posição adequada

Fonte: https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/toctree.html

Ordenação por inserção (inserction sort)

Destino Origem

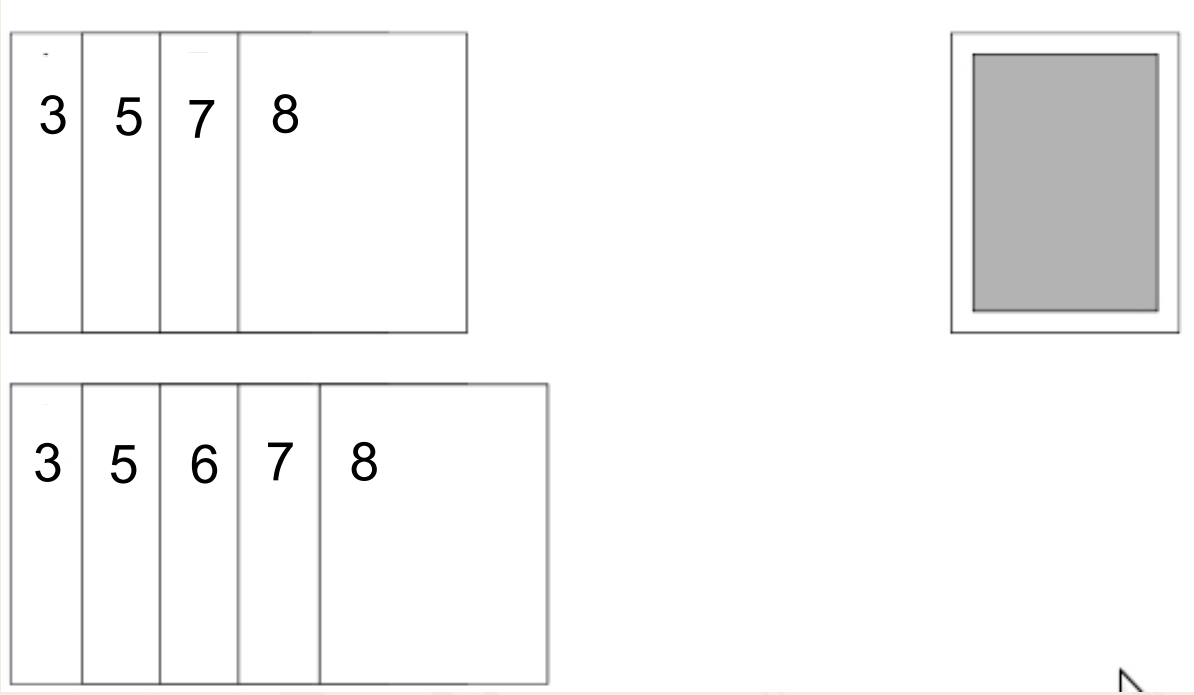


Fonte: https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/toctree.html

Ordenação por inserção (inserction sort)

Destino

Origem

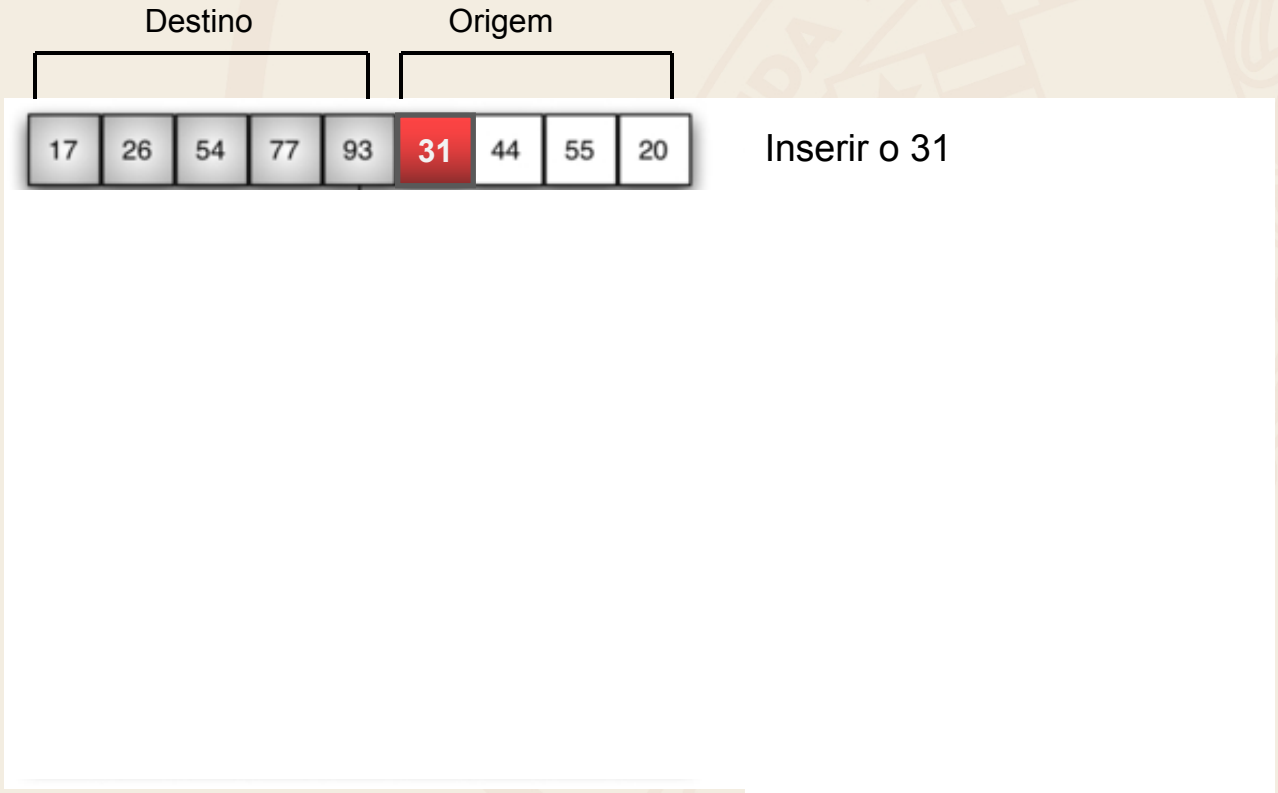


O processo finaliza quando a sublista de origem não tenha mais elementos

Fonte: https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/toctree.html

Ordenação por inserção (inserction sort)

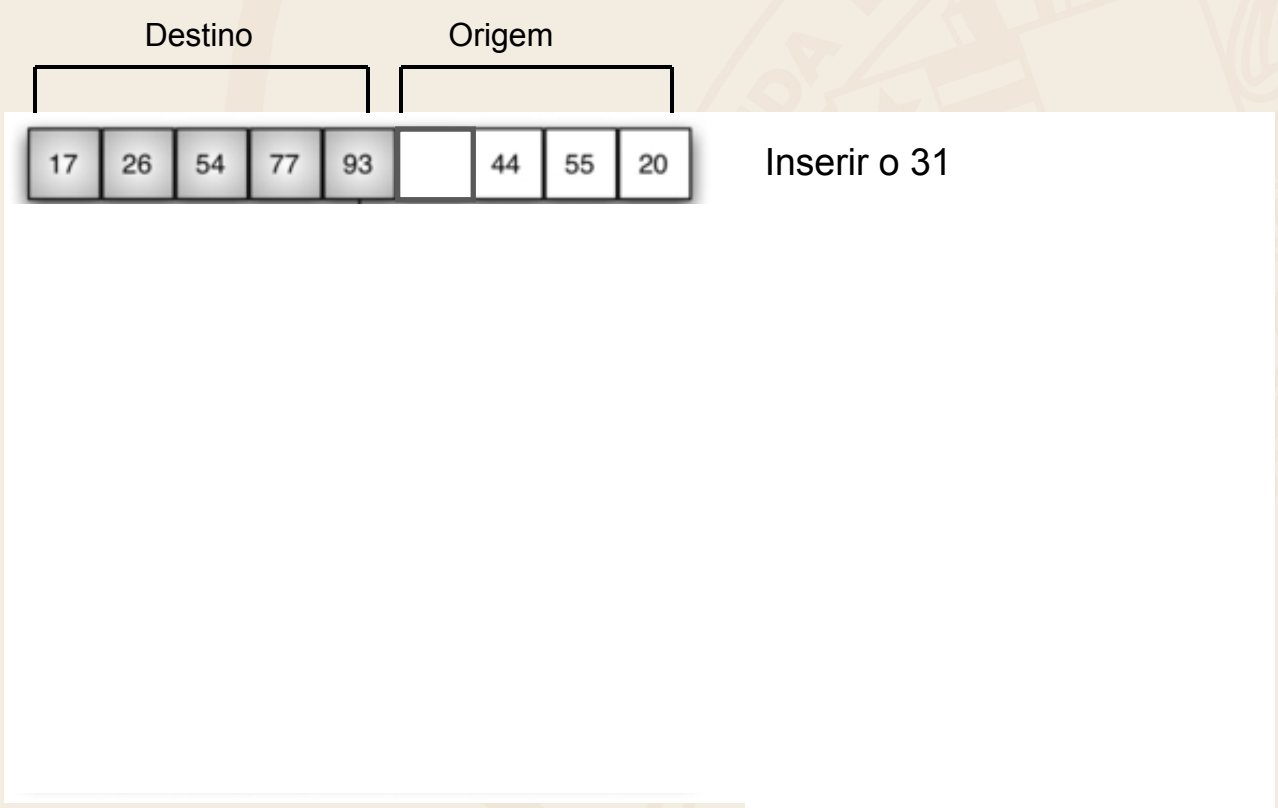
Um conceito básico é a inserção de um elemento em uma lista ordenada. Por exemplo, inserindo 31:



Fonte: https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/toctree.html

Ordenação por inserção (inserction sort)

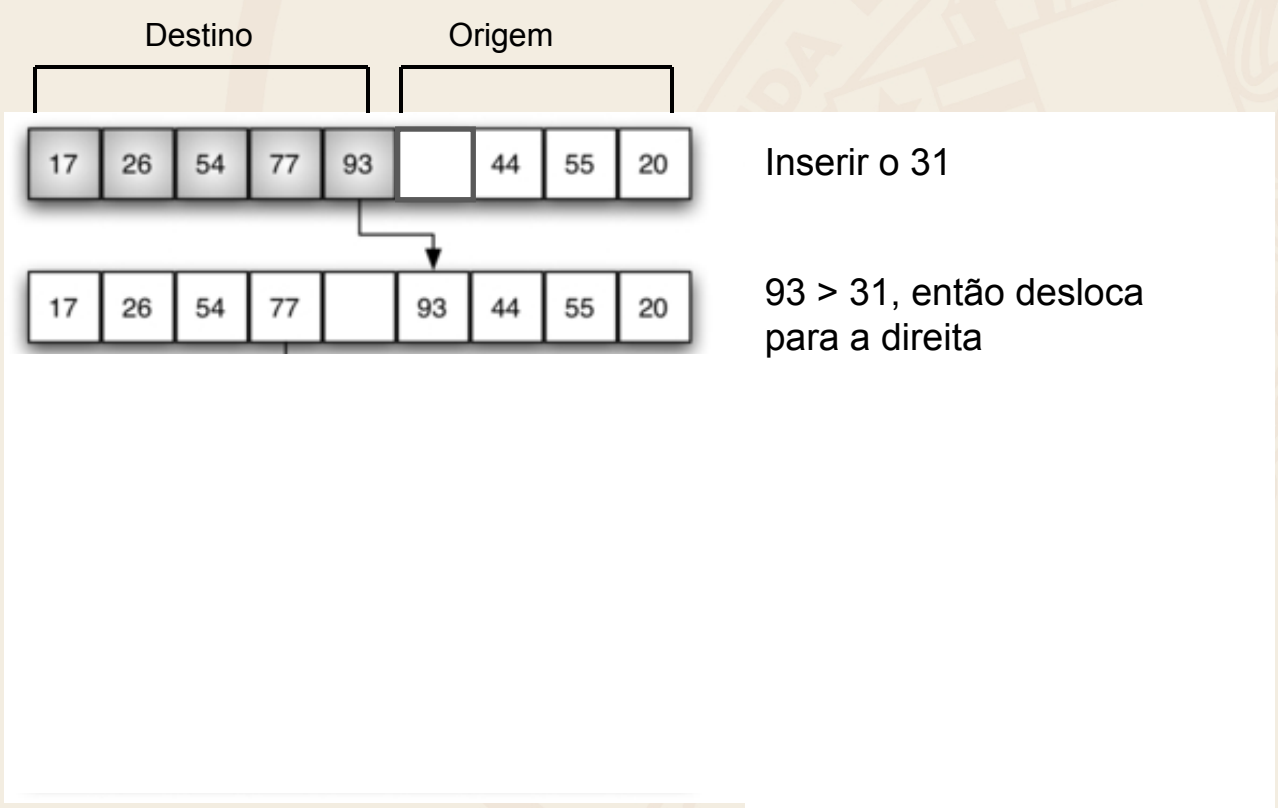
Um conceito básico é a inserção de um elemento em uma lista ordenada. Por exemplo, inserindo 31:



Fonte: https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/toctree.html

Ordenação por inserção (inserction sort)

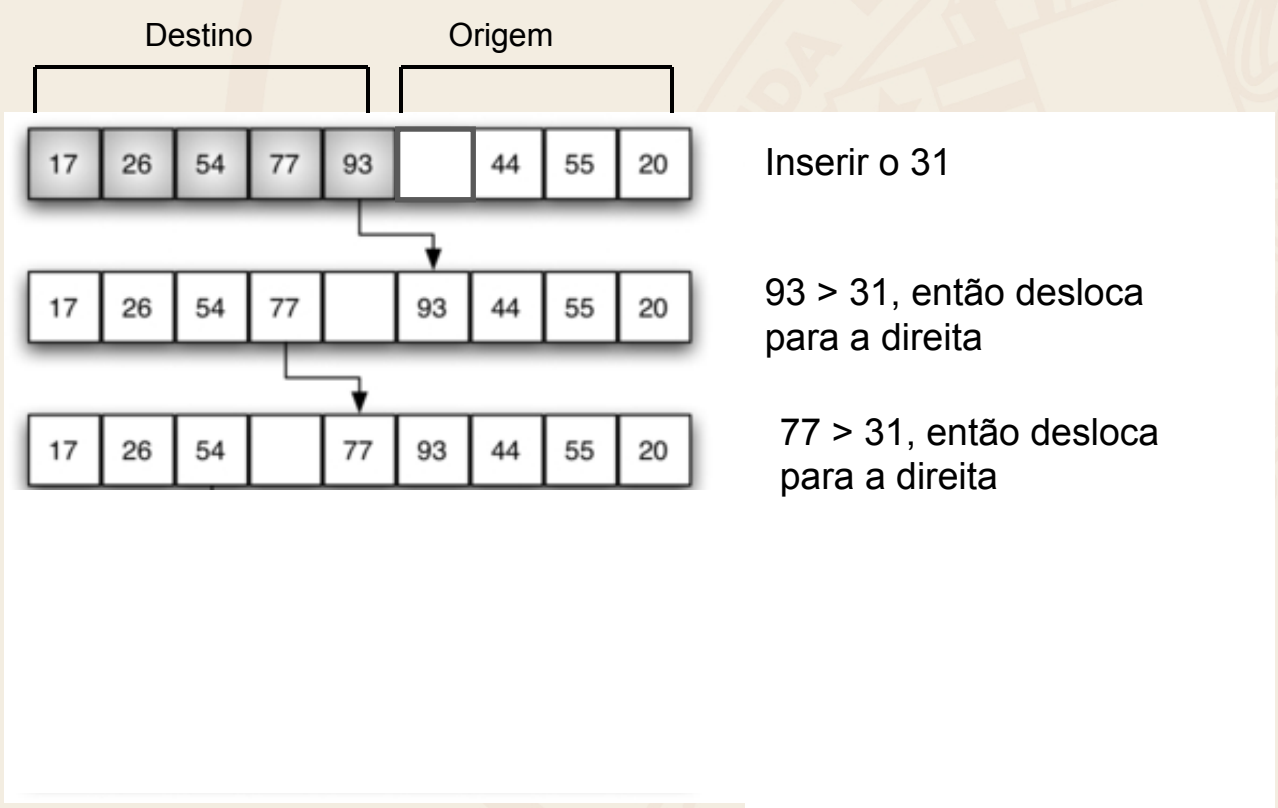
Um conceito básico é a inserção de um elemento em uma lista ordenada. Por exemplo, inserindo 31:



Fonte: https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/toctree.html

Ordenação por inserção (inserction sort)

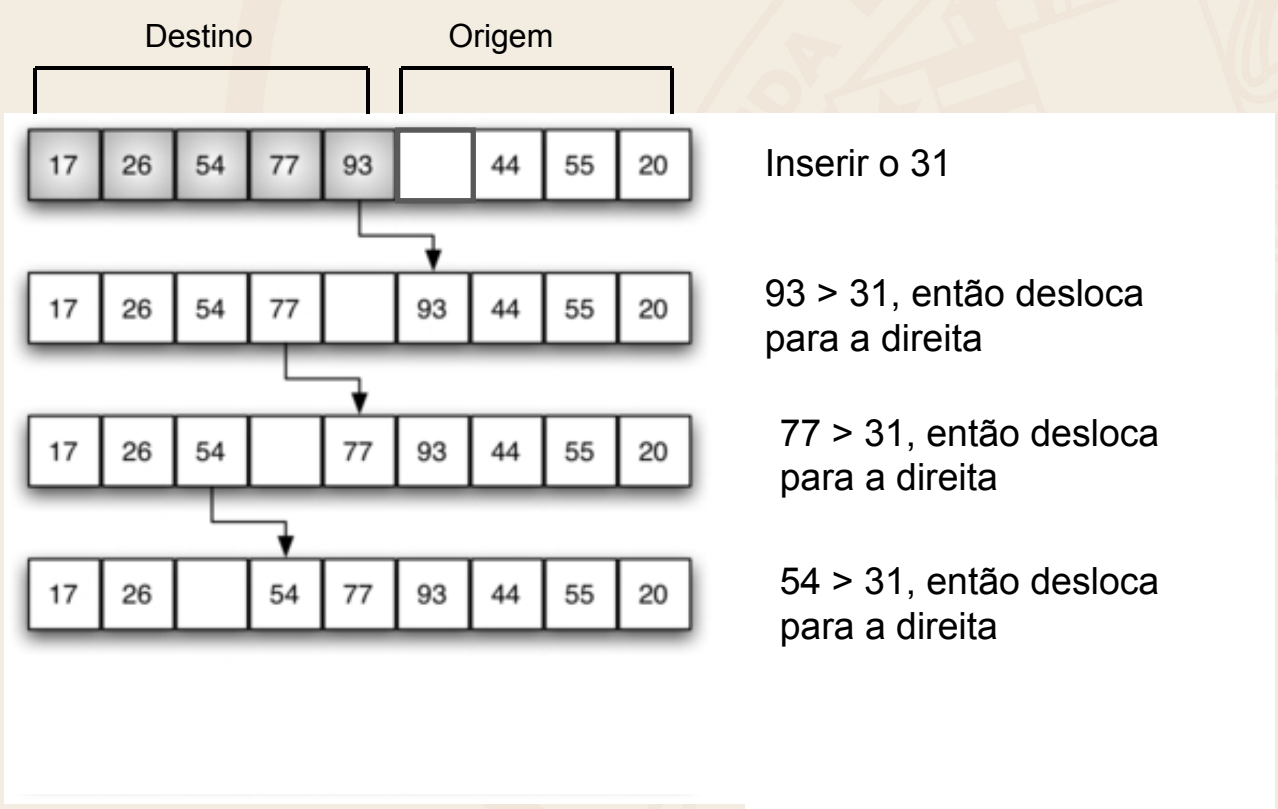
Um conceito básico é a inserção de um elemento em uma lista ordenada. Por exemplo, inserindo 31:



Fonte: https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/toctree.html

Ordenação por inserção (inserction sort)

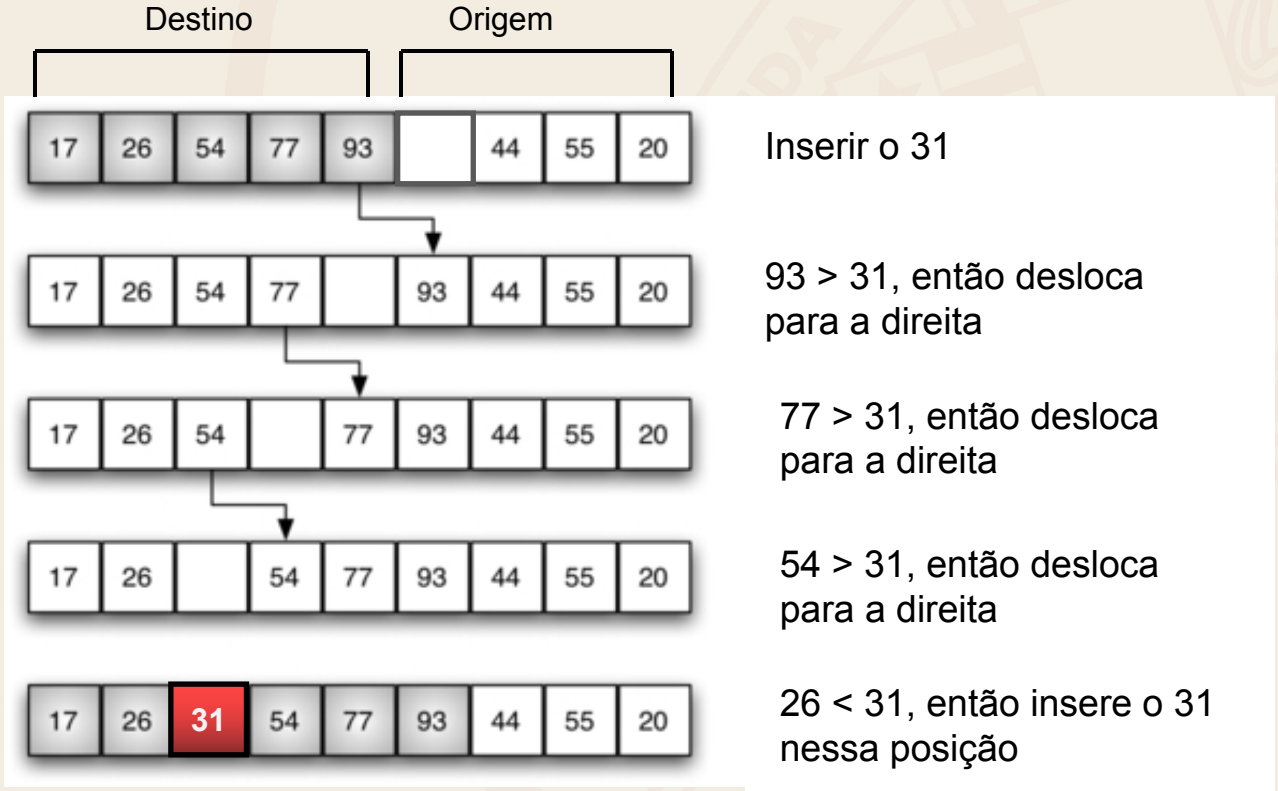
Um conceito básico é a inserção de um elemento em uma lista ordenada. Por exemplo, inserindo 31:



Fonte: https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/toctree.html

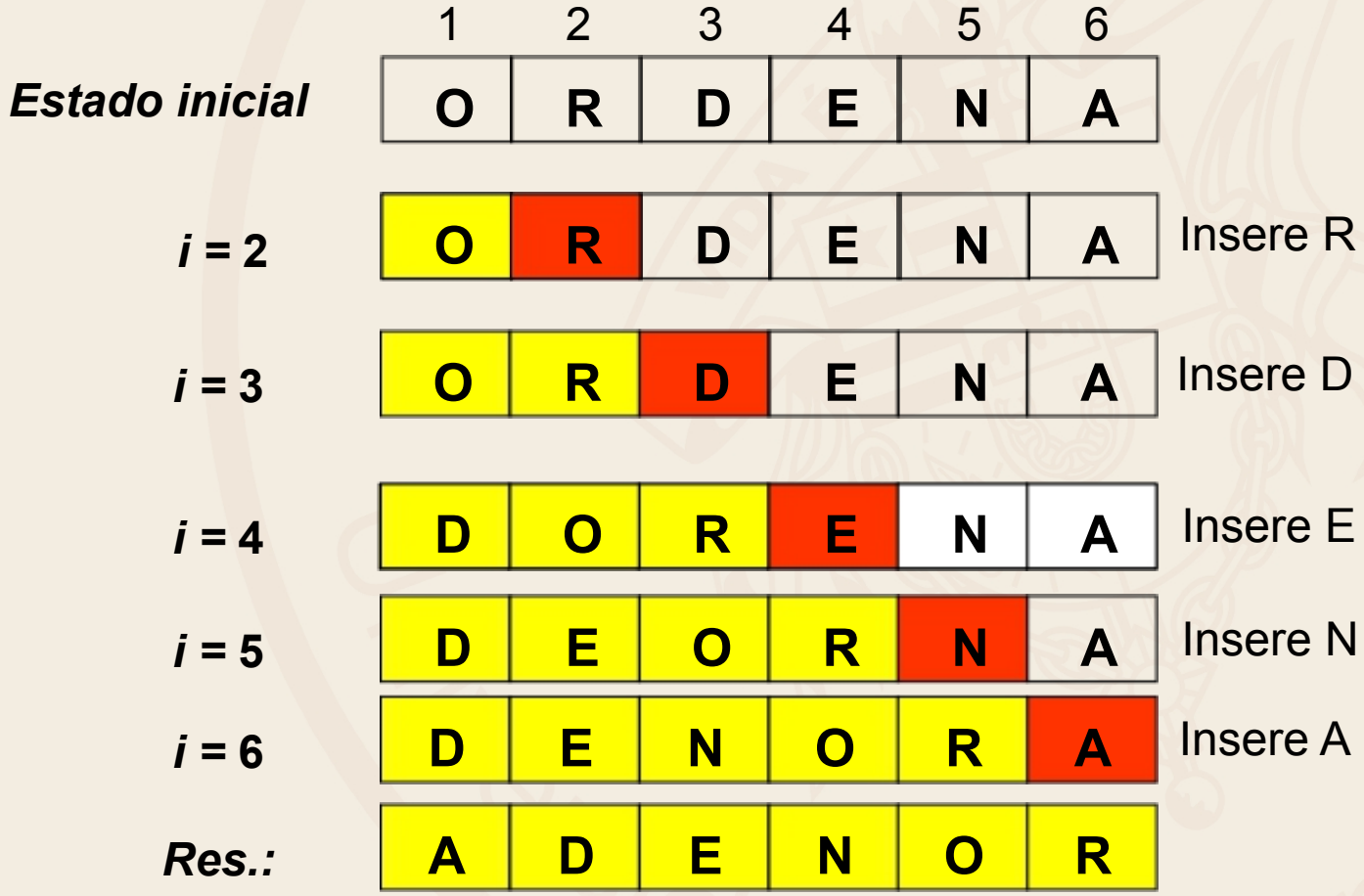
Ordenação por inserção (inserction sort)

Um conceito básico é a inserção de um elemento em uma lista ordenada. Por exemplo, inserindo 31:



Fonte: https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/toctree.html

Ordenação por inserção (inserction sort)



Fonte: https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/toctree.html

Ordenação por inserção (inserction sort)

```
def insertionSort(l):
    for i in range(1,len(l)):
      valor = l[i]
      pos = i
      while pos>0 and l[pos-1]>valor:
          l[pos]=l[pos-1]
          position = pos-1

      l[pos]= valor

l = [54,26,93,17,77,31,44,55,20]
insertionSort(l)
print(l)
```

Começando pelo segundo elemento

O valor que vai ser inserido

Observe que ele compara com os elementos anteriores

https://replit.com/@sergio_costa/ordenacao#main.py

UFMA

Universidade Federal do Maranhão

dted

DIRETORIA DE TECNOLOGIAS
NA EDUCAÇÃO

EAD PARA VOCÊ

www.eadparavc.dinte.ufma.br

@ufma.eadparavoce