UNIVERSIDADE DE VASSOURAS

PRÓ-REITORIA DE CIÊNCIAS TECNOLÓGICAS E EXATAS

CURSO DE ENGENHARIA DE SOFTWARE

Trabalho P1 – Estrutura de Dados

Professor: Caio Jannuzzi

Aluno: Jeziel Luiz Monteiro Farani

Matrícula: 202313146

Vassouras

2024

Aula 2: Introdução ao Python

Python é uma linguagem de programação simples, de fácil aprendizado e portátil criada em 1991, por Guido Van Rossum, muito utilizada para desenvolvimento web, IA, Big Data, Ciência de Dados, Computação Gráfica, etc.
Essa linguagem de programação teve um aumento exponencial em sua popularidade, já que é uma linguagem bem fácil de se aprender. De acordo com o site TIOBE, atualmente Python é a linguagem mais popular do mundo, ficando na primeira posição do ranking, logo a frente das linguagens C e C++. Qualquer pessoa consegue instalá-la, basta acessar o site do Python.
O próximo passo para codar é ter em sua máquina um editor de código ou um IDE, utilizando o Anaconda ou de forma tradicional. No Anaconda, além de conseguir instalar seu editor de código ou seu IDE, ele possibilita a criação de ambientes virtuais para seus projetos e a instalação de bibliotecas para tais projetos. Mas, afinal, qual a diferença entre um editor de código e um IDE? Os editores de código são basicamente editores de texto com recursos e funcionalidades para simplificar e agilizar o processo de edição de código. Já um IDE, são várias ferramentas para desenvolvimento de software com a finalidade de facilitar o processo de codificação.
Caso a pessoa não queira criar um ambiente virtual de desenvolvimento e ter que configurá-lo, pode-se utilizar o Google Colab. Nele, não é preciso configurar o ambiente, é só logar e usar!

Manipulação de variáveis e constantes

Variáveis

Por meio delas, os algoritmos guardam os dados do problema. Pode ser classificados em: float (valores positivos ou negativos decimais), caracteres (char ou string, elementos do teclado) e lógico (boleano).

Manipulação de Strings

.upper - transcreve toda a string em letra maiúscula;

.lower - transcrever toda a string em letra minúscula;

.replace - substitui uma parte específica de uma string por outra;

.find - acha a posição exata de um termo da string;

.strip - remove espaços em branco antes e depois da string;

capitalize() - transcreve apenas a primeira letra da string em letra maiúscula;

len() - exibe o tamanho da string;

Operações Matemáticas

(+) - soma dois ou mais termos;

```
1
2    a = 10
3    b = 20
4
5    soma = (a + b)
6    print(soma)
7
```

(-) - subtrai dois ou mais termos;

```
1
2    a = 10
3    b = 20
4
5    subtracao = (b - a)
6    print(subtracao)
7
```

(/) - divide dois termos;

```
1
2    a = 10
3    b = 20
4
5    divisao = (b / a)
6    print(divisao)
7
```

(%) - exibe o resto da divisão;

```
1
2    a = 9
3    b = 20
4
5    resto = (b % a)
6    print(resto)
7
```

(*) - multiplica dois ou mais termos;

```
1
2    a = 10
3    b = 20
4
5    multiplicacao = (b * a)
6    print(multiplicacao)
7
```

(**) - eleva um termo ao número especificado;

```
1
2    a = 10
3    b = 20
4
5    potenciacao = (b ** a)
6    print(potenciacao)
7
```

round() - arredonda um valor decimal;

```
1
2    a = 9
3    b = 20
4
5    arredondamento = (b / a)
6    print(round(arredondamento, 1))
7
```

Operadores Condicionais

> - maior que;

< - menor que;

>= - maior ou igual;

<= - menor ou igual;

!= - diferente de.

Aula 3: (Continuando)

Estruturas de Repetição

for - seu ciclo será executado por um tempo ou condição pré-determinados e em uma quantidade de vezes que determinamos;

```
1
2   for numero in range(1, 6):
3       print(numero)
4
```

while (enquanto) - é usada quando não sabemos quantas vezes um determinado bloco de instruções precisa ser repetido; com ele, a execução das instruções vai continuar até que uma condição seja verdadeira;

```
1
2   numero = 1
3   while numero < 6:
4       print(numero)
5       numero += 1
6   print("---")
7
```

Coleções: Tuplas e Listas

Tuplas

São sequências de valores de qualquer tipo, separados por vírgulas e, opcionalmente, entre parênteses.

```
1
2   tupla = ("João", "Maria", "José")
3
```

Listas

São um tipo de estrutura de dados que permite armazenar uma coleção de itens em uma única variável.

```
1
2   lista = ["João", "Maria", "José"]
3
4   lista2 = ["Carlos", "Matheus", "Márcia"]
5
```

.append - adiciona um elemento ao final da lista;

.remove - remove um elemento específico da lista;

Coleções: Dicionários e Conjuntos

Dicionários

São coleções que guardam valores multidimensionais para cada índice.

```
1
2    rank_forca = {"João":35, "Maria":26, "José":32}
3
```

del - deleta um elemento do dicionário;

```
3
4    del (rank_forca)["João"]
5
```

.update - agrupa dois dicionário em um só;

.items() - mostra os elementos do dicionário;

Conjuntos(set)

São coleções de itens desordenados, parcialmente imutáveis e que não podem conter elementos duplicados (por isso utilizamos o set()).

```
1
2   bolo = (
3       "farinha",
4       "água",
5       "sal",
6       "ovo"
7       "sal"
8       "ovo"
9       "ovo"
10      "baunilha"
11      "manteiga"
12  )
13
14  print(
15      set(bolo)
16  )
17
```

.intersection - exibe a interseção de dois ou mais conjuntos, ou seja, os elementos em comum entre eles;

.difference - exibe os elementos diferentes entre dois ou mais conjuntos;

Matrizes

numpy (ou np) - biblioteca para matrizes;

```
1
2   import numpy as np
3
```

.array - formar sua matriz;

.shape - exibe o formato da matriz (linhas, colunas);

Funções

```
1
2   def mensagem():
3       return "Olá, mundo!"
4
```

Funções com passagem de parâmetro

```
1
2   def mensagem(texto):
3       print(texto)
4
5   mensagem('Olá, mundo!')
6
```

```
1
2   def soma(a, b):
3       print(a + b)
4
5   soma(10, 50)
6
```

Aula 4: Notação Big-O

É uma notação matemática que descreve o comportamento limitante de uma função quando o argumento tende a um valor específico ou ao infinito. Ela descreve a complexidade do seu código e compara algoritmos para ver qual é mais eficiente.

%timeit - exibe o tempo de execução do algoritmo;

Funções Big-O

O(1) (constante) - o número de operações não cresce;

O(log n) (logaritmo) - crescimento do número de operações < número de itens;

O(n) (linear) - crescimento do número de operações = crescimento do número de itens;

O(n log n) (linearitmica ou quasilinear) - log n executada n vezes;

O(n^2) (quadrático) - processamento de itens de dados aos pares com repetições;

O(2^n) (exponencial) - quando n aumenta, o fator analisado(tempo/espaço) aumenta de forma exponencial;

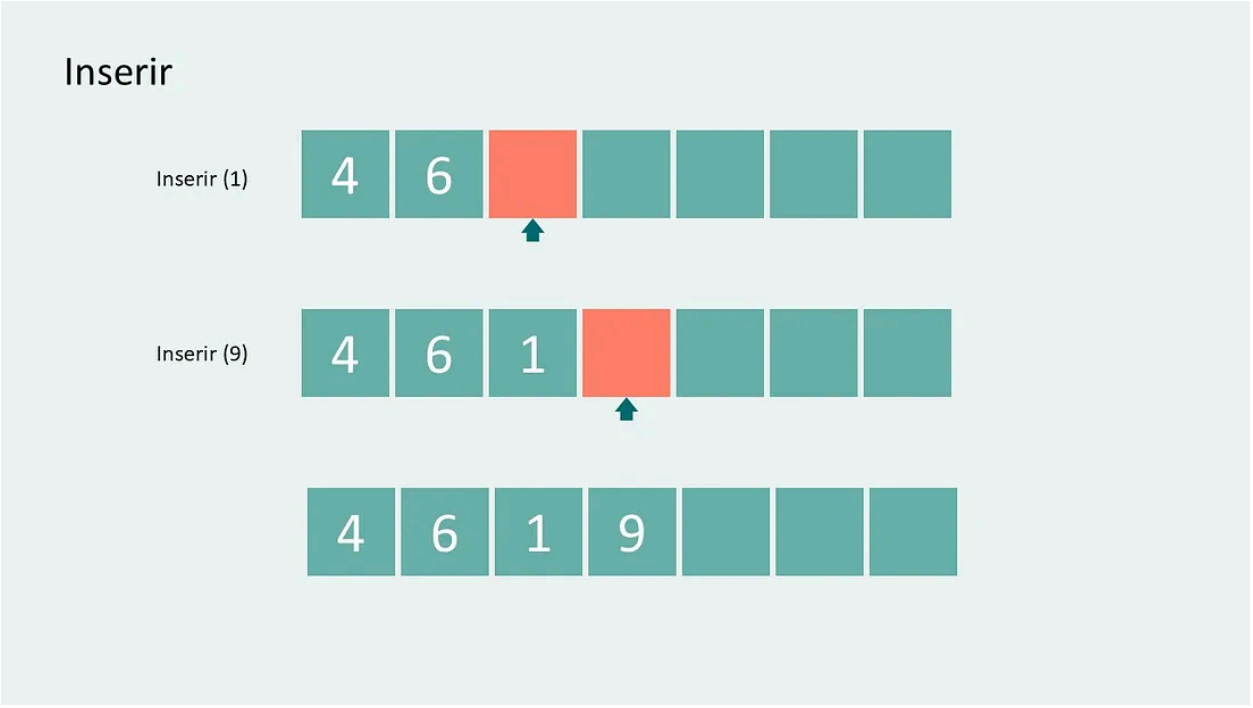
O(n!) (fatorial) - +++número de instruções executada -pequeno número de dados

.ones - retorna uma nova matriz de determinado formato e tipo de dados, onde o valor do elemento é definido como 1;

Aula 5: Vetores Não Ordenados

Inserção

Nos vetores não ordenados, a inserção costuma ser realizada em um único passo, uma vez que os elementos são adicionados ao primeiro espaço disponível (O(1) - Big-O constante).

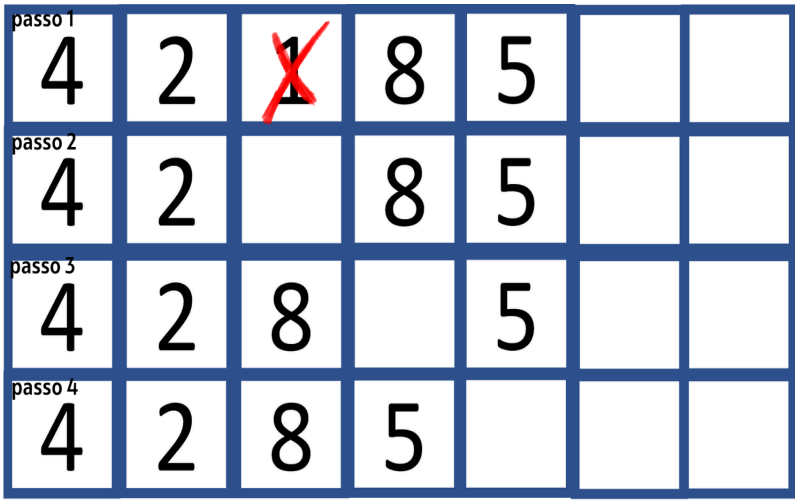


Pesquisa Linear

Faz-se a pesquisa do elemento dentro do vetor, tendo que passar por cada posição do vetor (O(n) - Big-O linear).

Exclusão

O número é excluído e, logo em seguida, seu espaço é sobrescrito pelo próximo item do vetor, e assim por diante (pesquisa linear) (O(2n) = O(n) - Big-O).



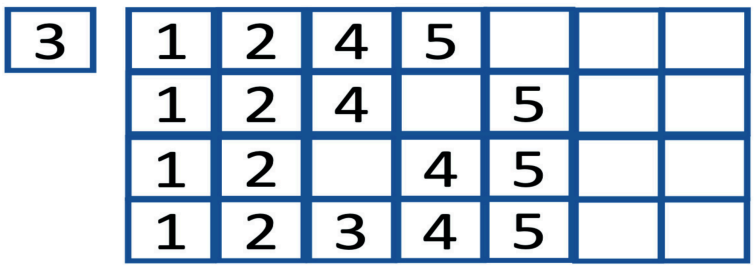
Duplicatas

Com o uso de duplicatas, terá que se percorrer todo o vetor para verificar se o elemento já existe.

Aula 6: Vetores Ordenados

Inserção

É necessário mais passos para a inserção de um elemento no vetor, diferente do não ordenado, onde o elemento apenas seria inserido no último espaço/posição do vetor (pesquisa linear) (O(2n) = O(n) - Big-O).

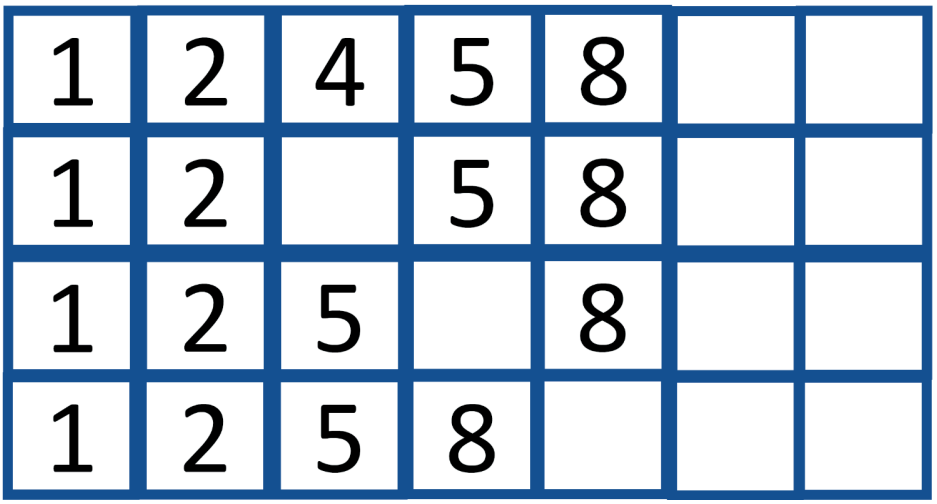


Pesquisa Linear

Termina quando: primeiro item > valor de pesquisa (atingido) (O(n) - Big-O).

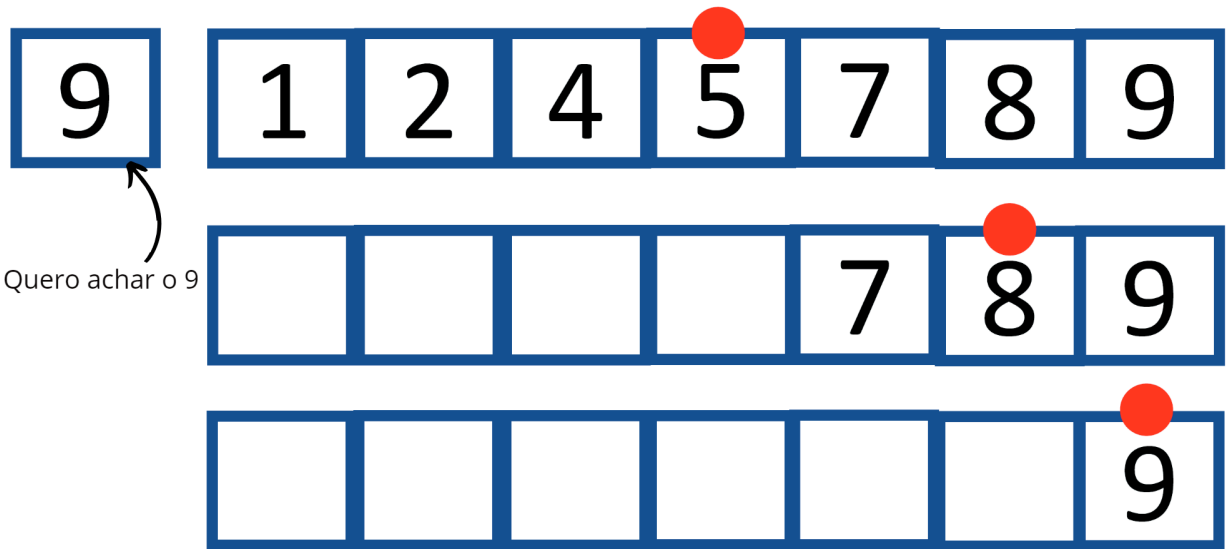
Exclusão

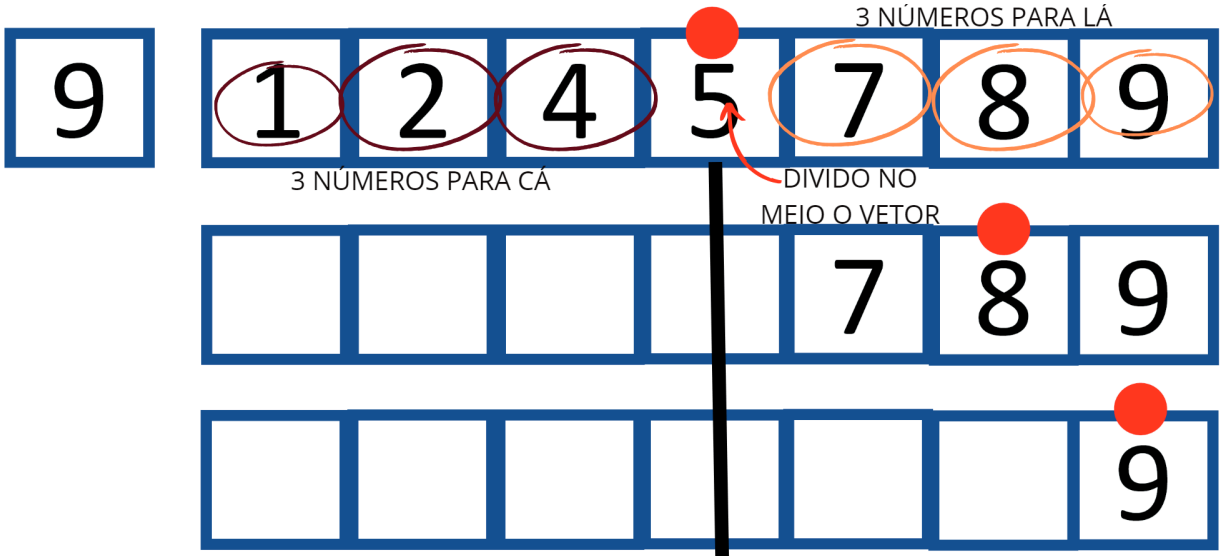
Se o algoritmo não encontrar o elemento, pode terminar a pesquisa linear no meio do caminho (O(2n) = O(n) - Big-O).

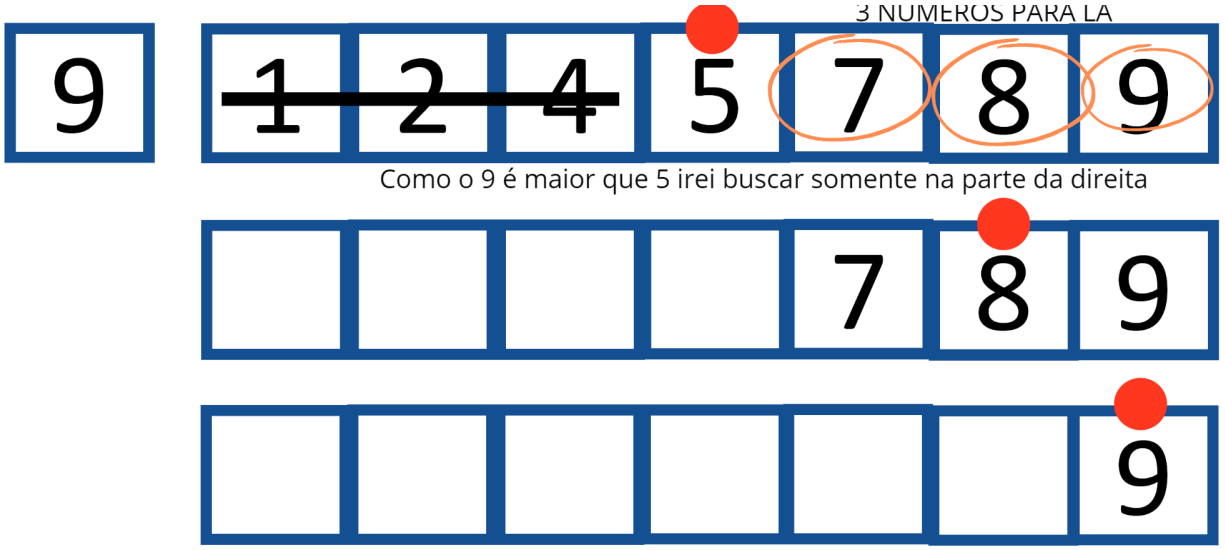


Pesquisa Binária

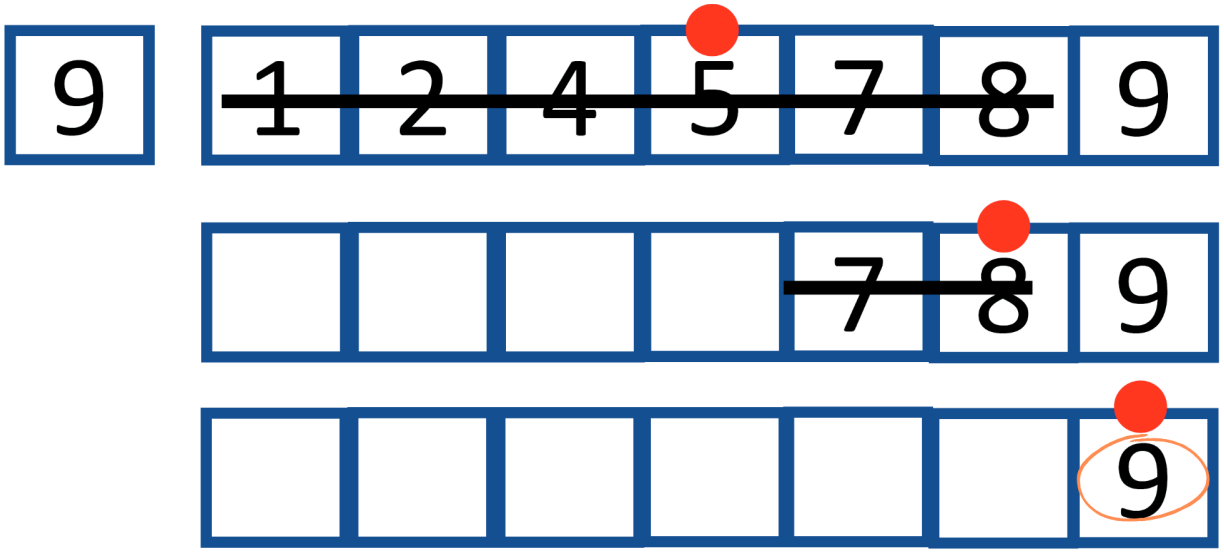
Dividi o problema por 2, facilitando a pesquisa pelo elemento. Ao contrário dos vetores não ordenados, os tempos de pesquisa são muito mais rápidos.




9
1 2 4 5 7 8 9
3 NÚMEROS PARA LÁ
3 NÚMEROS PARA CÁ
DIVIDO NO MEIO O VETOR
7 8 9
9


9
1 2 4 5 7 8 9
Como o 9 é maior que 5 irei buscar somente na parte da direita
7 8 9
9


9
1 2 4 5 7 8 9
DIVIDO NO MEIO O VETOR
7 8 9
1 NÚMERO PARA CÁ
1 NÚMERO PARA LÁ
9


9
1 2 4 5 7 8 9
7 8 9
9