

UNIVERSIDADE FEDERAL DE ITAJUBÁ

Instituto de Matemática e Computação

Ciência da Computação

--

CMAC03 - Algoritmos em Grafos

Tópico: 1. Introdução à Grafos

Aula Prática - Python Básico

Objetivo: Este notebook tem por objetivo apresentar a linguagem que será adotada para implementação dos algoritmos que serão abordados na disciplina. Ele pode ser usado como ambiente de desenvolvimento alternativo à IDE e ao Moodle. Contudo, visando uma maior autonomia do aluno, a recomendação é que seja utilizada uma IDE instalada localmente no seu equipamento.

Para cada enunciado a seguir implemente os códigos de modo a atender o que se pede.

Matrícula: 2022015139

Nome: Pablo Augusto Matos da Silva

0. Bibliotecas

É recomendável adicionar as bibliotecas necessárias logo no início do código seguindo uma boa prática de programação além de evitar futuros problemas. **Obs:** Use a célula abaixo para as bibliotecas que forem necessárias.

Q1. Instale a biblioteca iGraph (<https://python.igraph.org/en/stable/>).

```
In [59]: from igraph import numpy_to_contiguous_memoryview  
  
# pip install igraph
```

Q2. Importe as seguintes bibliotecas: `numpy`, `time`, `math`, `random`, `igraph`. Faça a montagem de uma pasta no repositório do Google Drive que contenha os *datasets* sobre grafos disponíveis no SIGAA (simples, ponte, zachary).

```
In [60]: # pip install numpy
```

```
In [61]: import numpy as np
import time
import math
import random
import igraph
```

****1. Variáveis e *Strings****

Q3. Crie uma variável do tipo *string* que recebe um valor de entrada dado pelo usuário e retorna um valor do tipo inteiro que é o triplo do valor fornecido.

```
In [62]: valor_str: str = input("Digite um numero do tipo inteiro: ")

numero = int(valor_str)
print(f"o triplo do valor digitado é: {numero * 3}")
```

o triplo do valor digitado é: 9

Q4. Concatene duas *strings* dadas pela entrada do usuário e cujos caracteres estão todos em caixa baixa (minúsculos), de modo que o primeiro caracter de cada string se torne caixa alta (maiúsculo).

```
In [63]: nome: str = input("Digite seu primeiro nome: ").capitalize()
sobrenome: str = input("Digite seu sobrenome: ").capitalize()
nome_completo: str = (nome+" "+sobrenome)
print(f"Nome completo: {nome_completo}")
```

Nome completo: Pablo Augusto

2. Entrada e Saída

Q5. Implemente uma função que recebe como entrada o nome de um *dataset* armazenado no seu Google Drive (ex. simples, ponte, zachary), lê o respectivo arquivo `.txt`, atribua o conteúdo do arquivo a um objeto tipo `numpy.ndarray`, imprime na tela a respectiva matriz e o seu tipo.

```
In [64]: def ler_arquivo_dataset(caminho_arquivo: str):
    matriz = np.genfromtxt(caminho_arquivo)
    print(matriz)
    print(type(matriz))
    return matriz

matriz = ler_arquivo_dataset("datasets/exemplo.txt")
```

```
[[0. 1. 1. 0. 0. 1. 0.]
 [1. 0. 0. 0. 0. 0. 0.]
 [1. 0. 0. 1. 1. 1. 0.]
 [0. 0. 1. 0. 1. 0. 1.]
 [0. 0. 1. 1. 0. 0. 0.]
 [1. 0. 1. 0. 0. 0. 1.]
 [0. 0. 0. 1. 0. 1. 0.]]
<class 'numpy.ndarray'>
```

Q6. Obtenha as dimensões da matriz no formato (linhas, colunas), bem como a lista com os valores constantes na sua diagonal.

```
In [65]: linhas, colunas = matriz.shape
         diagonal = matriz.diagonal()
         print(f"Linhas: {linhas}, Colunas: {colunas}")
         print(f"Diagonal: {diagonal}")
```

```
Linhas: 7, Colunas: 7
Diagonal: [0. 0. 0. 0. 0. 0. 0.]
```

Q7. Usando a matriz obtida em Q5, imprima o elemento da posição `[i][j]` informado pelo usuário. Informe caso algum índice dado na entrada seja maior que a dimensão da matriz.

```
In [66]: i = int(input("Digite a linha: "))
         j = int(input("Digite a coluna: "))

         if i > linhas:
             print(f"0 indice [{i}][{j}] ultrapassa a dimensao da matriz({linhas})
         else:
             print(f"Matriz[{i}][{j}] = {matriz[i][j]}")
```

```
Matriz[2][3] = 1.0
```

Q8. Salve em um arquivo no seu Google Drive com o nome "resultado.txt", lista dos valores da diagonal da matriz obtida em Q6.

```
In [67]: arquivo = open("resultado.txt", "w").write(str(diagonal))
```

3. Estruturas

Q9. A partir da matriz, implemente uma função que constroi um dicionário. Cada chave do dicionário corresponde ao índice da linha da matriz. O valor de cada chave corresponde a uma lista com os índices das colunas associadas à linha em questão, que possuam valor maior que 0. Caso o valor da coluna seja maior que 1 deve-se repetir o índice da coluna conforme o valor (ex. a célula dada pela linha 1 e coluna 2 tem valor 4, logo, o valor 2 na lista da linha 1 precisará ser repetido 4 vezes).

```
In [68]: def matrix_to_dict(matriz):
         return {i: [j for j, valor in enumerate(linha) for _ in range(int(valor))
                    if valor > 0] for i, linha in enumerate(matriz)}

         matrix_to_dict(matriz)
```

```
Out[68]: {0: [1, 2, 5],
          1: [0],
          2: [0, 3, 4, 5],
          3: [2, 4, 6],
          4: [2, 3],
          5: [0, 2, 6],
          6: [3, 5]}
```

Q10. Calcule o tempo total de execução da função que implementou em Q9, considerando precisão de 4 (quatro) casas decimais.

```
In [69]: import time

        inicio = time.perf_counter()
        matrix_to_dict(matriz)
        fim = time.perf_counter()
        print(f"{{(fim - inicio):.4f}} segundos")
```

0.0001 segundos

Q11. Crie uma lista que armazene os valores de uma linha da matriz obtida em Q5, cujo índice da linha é dado como entrada. Imprima na tela essa lista e a sua versão invertida.

```
In [70]: linha = int(input("Digite o indice da linha desejada: "))
        lista = [int(i) for i in matriz[linha]]
        print(lista)
        lista.reverse()
        lista
        lista.reverse()
```

[1, 0, 0, 1, 1, 1, 0]

Q12. Considere a lista obtida em Q11 e altere o elemento que consta na posição 1 dessa lista para o valor 2.

```
In [71]: lista[1] = 2
        lista
```

Out[71]: [1, 2, 0, 1, 1, 1, 0]

Q13. Considerando a lista obtida em Q11, inclua o elemento 5 no final dessa lista e inclua o elemento 3 na posição 1 da lista.

```
In [72]: lista.append(5)
        lista[1] = 3
        lista
```

Out[72]: [1, 3, 0, 1, 1, 1, 0, 5]

4. Estruturas de Repetição

Q14. Crie uma função que recebe um número do usuário e imprime na tela uma contagem regressiva até o valor 1. Utilize o comando `for`.

```
In [80]: def contagem_regressiva_for(valor):  
         [print(i) for i in range(valor,0, -1)]  
  
         valor = int(input("Digite um numero inteiro: "))  
         contagem_regressiva_for(valor)
```

5
4
3
2
1

Q15. Implemente a mesma função solicitada em Q14, mas utilizando o comando `while`.

```
In [81]: def contagem_regressiva_while(valor):  
         while valor > 0:  
             print(valor)  
             valor -= 1  
  
         valor = int(input("Digite um numero inteiro: "))  
         contagem_regressiva_while(valor)
```

5
4
3
2
1

Q16. Implemente a mesma função solicitada em Q14, utilizando uma função recursiva.

```
In [82]: def contagem_regressiva_recursive(valor):  
         if valor > 0:  
             print(valor)  
             contagem_regressiva_recursive(valor-1)  
  
         valor = int(input("Digite um numero inteiro: "))  
         contagem_regressiva_recursive(valor)
```

5
4
3
2
1

5. Resolução de Problemas

Q17. Calcule a área de uma circunferência com base no raio recebido pelo usuário. Imprima na tela o resultado.

```
In [76]: import math
```

```
raio = float(input("Digite o raio da circunferencia: "))
area = math.pi * raio**2
print(f"Area da circunferencia: {area}")
```

Area da circunferencia: 12.566370614359172

Q18. Crie um programa que recebe duas notas do usuário e retorna uma mensagem com base na aprovação ou reprovação considerando média maior ou igual a 6,0.

```
In [77]: nota1 = float(input("Digite a primeira nota: "))
nota2 = float(input("Digite a segunda nota: "))

media = (nota1 + nota2) / 2

if media >= 6.0:
    print(f"Aprovado! Sua media foi {media:.2f}.")
else:
    print(f"Reprovado! Sua media foi {media:.2f}.")
```

Reprovado! Sua media foi 4.50.

Q19. Implemente uma função que recebe uma medida de ângulo em graus e retorna uma mensagem indicando o seu quadrante.

```
In [78]: def quadrante(angulo):
    if 0 <= angulo < 90:
        return "0 angulo esta no 1º quadrante."
    elif 90 <= angulo < 180:
        return "0 angulo esta no 2º quadrante."
    elif 180 <= angulo < 270:
        return "0 angulo esta no 3º quadrante."
    elif 270 <= angulo < 360:
        return "0 angulo esta no 4º quadrante."
    else:
        return "Ângulo invalido. Insira um valor entre 0 e 360 graus."

angulo = float(input("Digite um angulo em graus: "))
print(quadrante(angulo))
```

0 angulo esta no 1º quadrante.