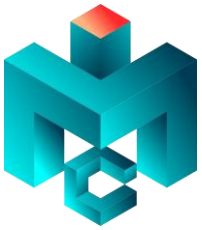
UNIVERSIDADE FEDERAL DE ITAJUBÁ
Instituto de Matemática e Computação



SIN110 – Algoritmos e Grafos

Preparação do ambiente de desenvolvimento

Rafael Frinhani
frinhani@unifei.edu.br

2022

OBJETIVOS

Apresentar a linguagem e respectivo ambiente de desenvolvimento, bem como boas práticas de desenvolvimento de código.

AGENDA

- Ambiente de Desenvolvimento
- Organização do Código
- Modularização
- Entrada e Saída de Dados
- Hibridização de Linguagens
- Estruturas de Dados
- Execuções via script
- Medindo o Tempo
- Geração de números pseudo-aleatórios
- Pythonicamente

Ambiente de Desenvolvimento

Linguagem: Python 3.10.0 ou superior.

Download: www.python.org/downloads/
OBS. Atenção com a incompatibilidade de versões

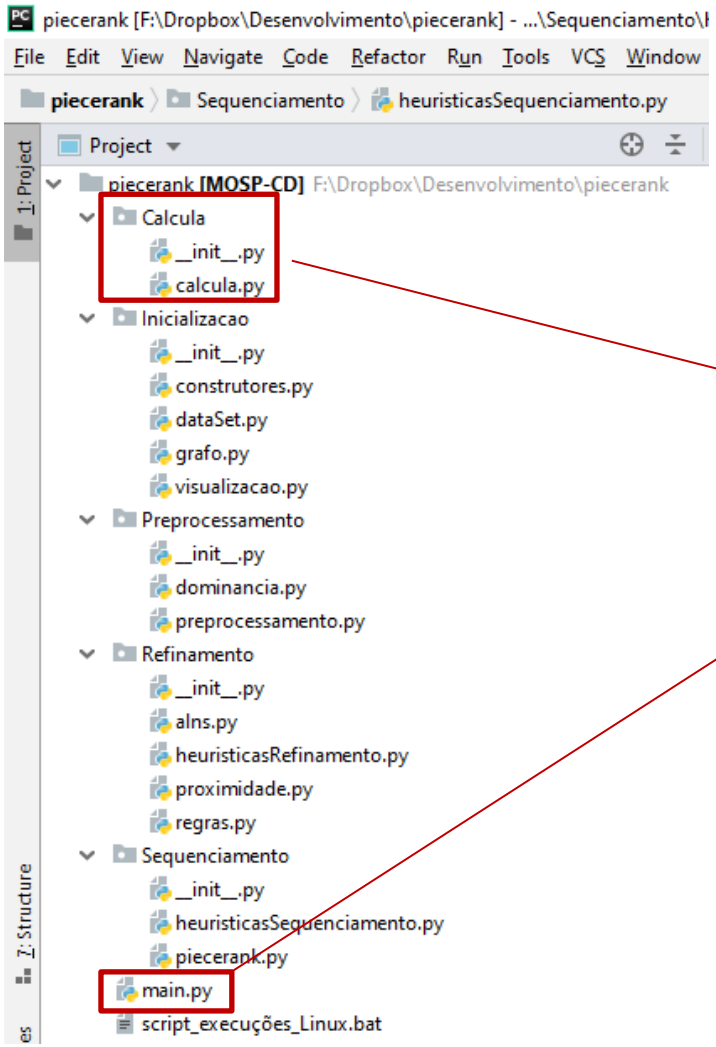
IDE: JetBrains PyCharm Community 2022.2
Download: www.jetbrains.com/pycharm/download

Bibliotecas Úteis:

- Numpy 1.23.1 – Manipulação de matrizes.
- Matplotlib 3.5.2 – Criação de gráficos.
- iGraph 0.9.11 – Manipulação de grafos.
- pyCairo 1.21.0 – Criação de gráficos.
- Pandas 1.4.3 – Manipulação e análise de dados.

OBS. Utilize **pip** para instalação de bibliotecas. No Windows por vezes é necessária a instalação via arquivo (https://www.lfd.uci.edu/~gohlke/pythonlibs/)

Organização do Código

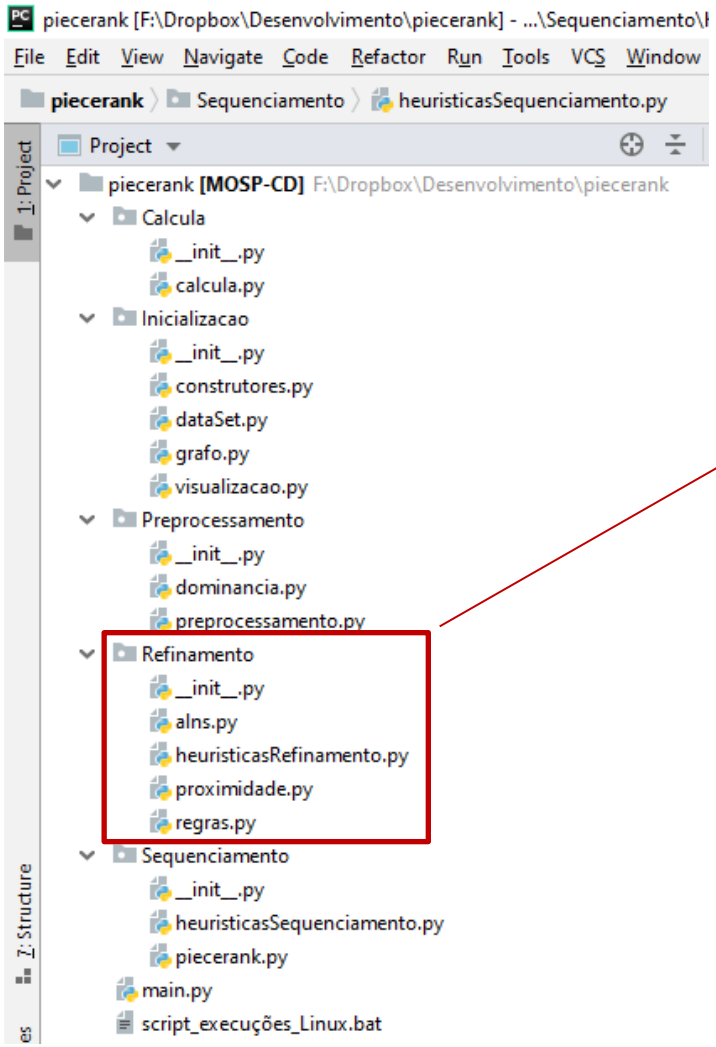


Organize o código em pastas e arquivos de modo a agilizar o desenvolvimento e a localização de itens a serem alterados.

Exemplo:

- A classe “Calcula” contém o arquivo “calcula.py” que contém apenas funções que realizam cálculos.
- O arquivo “main.py” contém o core da aplicação.

Organização do Código



Organize o código em pastas e arquivos de modo a agilizar o desenvolvimento e a localização de itens a serem alterados.

Exemplo:

- Classe “Refinamento” contém apenas métodos de refinamento de soluções. O arquivo “heurísticasRefinamento.py” recebe como parâmetro a função a ser executada.

heuristicasRefinamento.py

```
1  from Refinamento import (regras as rg, alns as al)
2
3  def refinamento(heuristica, listaPecas, LPgeral):
4      if heuristica == 'r12':
5          LPgeral = rg.refinamentoRegrasIe2(LPgeral, listaPecas)
6      if heuristica == 'alns':
7          LPgeral = al.refinamentoALNS(LPgeral, listaPecas)
8      return LPgeral
```

Modularização

Crie funções modularizadas, de modo a possibilitar seu acionamento ou alterações de forma independente, sem comprometer o funcionamento de toda a aplicação.

Seja criterioso com os **comentários**

```
''' Função que constroi uma lista de padroes com respectivas peças que ele contém (idPadrao : [peças] qtdPeças)
    Entrada: Matriz = Matriz Padrao x Peças; l = qtd. linhas matriz; c = qtd. colunas matriz
    Saída: lista Padrões = idPadrao : [pecas, qtdPecas]  '''
def constroiListaPadroes(matriz, l, c):
    listaPadroes = {}
    for i in range(l):...
    return listaPadroes
```

Entrada da função

Saída da função

Altere o que for preciso na função respeitando sempre que possível sua entrada e saída para um menor impacto na quantidade de alterações.

Entrada de Dados

Padronize os arquivos de entrada (*datasets*) de modo a utilizar uma única função para entrada dos dados. Para facilitar o uso de scripts de execução, coloque todos os arquivos em um mesmo local alterando o nome e identificador da instância.

```
6 7
1 1 1 0 0 0 0
0 1 1 1 0 0 0
1 0 0 0 1 0 1
0 0 0 1 0 1 1
1 0 1 0 0 0 0
0 0 1 0 1 0 1
```

Nome do *dataset* (**inst**) e identificador da instância (**id**)

Modo de acesso: **r** = read, **b** = binary

```
def criaMatPadraoPeca(inst, id):
    caminho = 'F:/Dropbox/Desenvolvimento/InstanciasPadroesxPecas/' + inst + id + '.txt'
    with open(caminho, 'rb') as f:
        nrows, ncols = [int(field) for field in f.readline().split()]
        data = np.genfromtxt(f, dtype="int32", max_rows=nrows)
    return data
```

Lê um arquivo .txt f, cria uma matriz de inteiros (32bits)

Os dados são armazenados em uma matriz (data).

Saída de Dados

Crie uma função que possibilite salvar em arquivo o resultado da execução do método. Salve os resultados de modo a facilitar seu planilhamento e obtenção de dados estatísticos para discussão.

resultado armazena o nome da instância, tempo e valor da função objetivo obtidas pelo método

```
resultado = str(inst) + ';' + str("%.4f" % tempo) + ';' + str("%.3f" % FO)
ds.salvaResultadoResumido(resultado)
```

Precisão de 4 casas decimais

Local de criação e nome do arquivo.

```
def salvaResultado(resultado):
    arquivo = open('D:/Dropbox/Desenvolvimento/Resultados/Resultados-MetodoX.csv', 'a+')
    arquivo.writelines(resultado + '\n')
    arquivo.close()
```

Fecha o arquivo após a escrita

Modo de Escrita:
a = Append (acrescenta)
w = Write (apaga e escreve)

Hibridização de Linguagens

Python permite a integração com C possibilitando o aproveitamento de código ou casos em que a função (ex. função objetivo) tem exigências de desempenho.

Comando para criação da biblioteca de vínculo dinâmico a partir do código escrito em C: gcc -shared -o mcnh.so -fPIC mcnh.c

```
import numpy as np
import ctypes

def mcnh(matriz):
    padroes, pecas = np.shape(matriz)
    matriz = np.transpose(matriz)
    lib = ctypes.cdll.LoadLibrary("./Sequenciamento/HNCM/mcnh.so")
    _mcnh = lib.mcnh

    solucao = np.empty((padroes), dtype=np.int32)  # Sequencia de padrões
    _mcnh(ctypes.c_void_p(matriz.ctypes.data), ctypes.c_int(padroes), ctypes.c_int(pecas), ctypes.c_void_p(solucao.ctypes.data))
    LP = solucao.tolist()
    return LP
```

Biblioteca para integração com C

Localização da biblioteca de vínculo dinâmico

Importante associar na entrada o tipo de dados suportado por C

Associação de tipos de dados

```
void mcnh(const int *indata, int padroes, int pecas, int *solucao)
```

Estruturas de Dados

Analise a estrutura mais adequada para manipulação dos dados de modo a facilitar a sua utilização com o melhor desempenho possível.

Utilize a biblioteca numpy que possui uma grande diversidade de comandos para operações sobre matrizes.

A estrutura de dicionários funciona como uma tabela *Hash*, sendo eficiente para armazenar informações de objetos.

Define listaNos como um dicionário

nosAdjacentes é uma lista de inteiros

```
listaNos = {}
nosAdjacentes = []
listaNos[no] = [grauNo, nosAdjacentes, round(centralidade, 4)]
```

```
0 : 4 [1, 2, 4, 6] 0.7308
1 : 3 [0, 2, 3] 0.5881
2 : 5 [0, 1, 3, 4, 6] 0.8822
3 : 4 [1, 2, 5, 6] 0.7256
4 : 3 [0, 2, 6] 0.6006
5 : 2 [3, 6] 0.4349
6 : 5 [0, 2, 3, 4, 5] 0.8698
```

Exemplos de Comandos:

> listaNos[0][0] retorna 4
> listaNos[0][1] retorna [1, 2, 4, 6]
> listaNos[0][1][2] retorna 4

Estruturas de Dados (cont.)

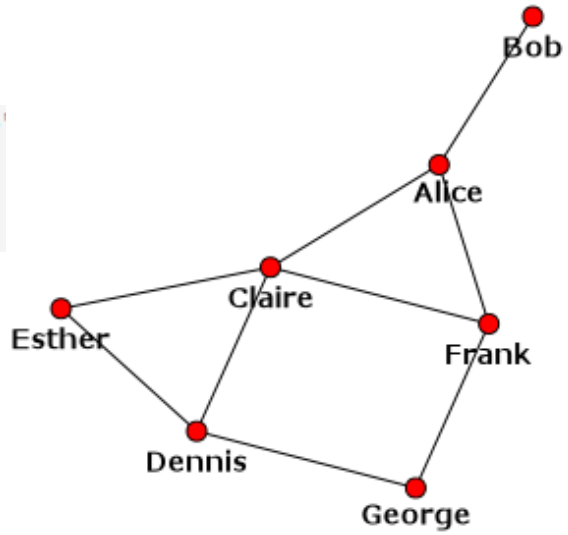
A biblioteca iGraph permite a manipulação de grafos e possui uma grande diversidade de comandos e funções para operações sobre grafos.

Criação de um grafo g que representa uma rede social a partir da definição das ligações

```
>>> g = Graph([(0,1), (0,2), (2,3), (3,4), (4,2), (2,5), (5,0), (6,3), (5,6)])
```

Definição dos atributos dos nós (vertices, vs). O mesmo é possível para as arestas (edges, es)

```
>>> g.vs["name"] = ["Alice", "Bob", "Claire", "Dennis", "Esther
>>> g.vs["age"] = [25, 31, 18, 47, 22, 23, 50]
>>> g.vs["gender"] = ["f", "m", "f", "m", "f", "m", "m"]
```



A biblioteca disponibiliza vários recursos, que vão desde medidas (ex. grau, distâncias) até métodos (ex. agrupamento, detecção de comunidades).

Execução via script

A execução do método via script permite automatizar os experimentos, que além de facilitar os testes é útil nos casos de grandes quantidades de instâncias.

Em main() define-se os parâmetros de entrada do método, que podem ser desde valores necessários para execução do método, como valores utilizados nas análises.

```
'''Chamada à função main()
   Parâmetros: [1]Dataset, [2]qtdInstancias, [3]melhorSol, [4]executarPreprocessamento, [5]metRefinamento'''
if __name__ == '__main__':
    main(str(sys.argv[1]), int(sys.argv[2]), float(sys.argv[3]), int(sys.argv[4]), str(sys.argv[5]))
```

Em um arquivo .bat são definidos os scripts de execução para cada instância do *dataset*.

```
F:\Dropbox\Desenvolvimento\piecerank>type completa_Artigo.bat
python main.py Random-30-30-2- 5 10.80 0 semRef
python main.py Random-30-30-4- 5 17.40 0 semRef
python main.py Random-30-30-6- 5 21.80 0 semRef
python main.py Random-30-30-8- 5 25.40 0 semRef
```

Medindo o Tempo

Trechos de código poderão ser medidos com uma função de tempo a partir de formato pré-configurados.

```
import time # Biblioteca

t_inicio = time.time() # Tempo inicio medição

                << TRECHO DE CÓDIGO A SER MEDIDO >>

t_total = time.time() - t_inicio # Tempo final medição

print (str("%.4f" % t_total)) # Precisão de 4 casas
```

Mais informações sobre a biblioteca e configurações:
https://docs.python.org/3/library/time.html

Geração de números Pseudo-Aleatórios

Muitos métodos necessitam que sejam gerados números pseudoaleatórios para o seu funcionamento.

```
from random import *

# Definir semente para permitir reprodução do experimento
seed(valor_semente)

# Real entre [0, 1) - Ex. 0.0853, 0.9561
("%.4f" % random())

# Escolhe um elemento de uma lista
choice(lista)

# Inteiro entre 0 e 9 – Ex. 3, 7
randrange(0, 9)

# Real entre 1 e 9 – Ex. 4.683, 5.461
("%.3f" % uniform(0, 9))

# Obtém uma nova lista com os k elementos embaralhados.
sample(lista, k=len(lista)) # Para todos os elementos use k = len(lista)
```

Mais informações sobre a biblioteca e configurações:

https://docs.python.org/3/library/random.html

Pythonicamente

Um código mais enxuto e melhores desempenhos são obtidos quando se escreve o código de forma "*pythonica*".

As funções a seguir dobram o valor de cada número par do vetor:

Implementação em C

```
int[] arr = { 1, 2, 3, 4, 5, 6 };
int length = sizeof(arr) / sizeof(arr[0]);

for (int i = 0; i < length; i++) {
    if (arr[i] % 2 == 0) {
        arr[i] *= 2
    }
}
```

Tradução direta em Python

```
arr = [1, 2, 3, 4, 5, 6]
length = len(arr)

for i in range(0, length):
    if arr[i] % 2 == 0:
        arr[i] *= 2
```

Forma Pythonica com *list comprehension*

```
arr = [1, 2, 3, 4, 5, 6]
arr = [x * 2 if x % 2 == 0 else x for x in arr]
```

Pythonicamente (cont.)

Dicas de leituras:

How to be Pythonic and why you should care
https://towardsdatascience.com/how-to-be-pythonic-and-why-you-should-care-188d63a5037e

Effective Python
https://hacktec.gitbooks.io/effective-python/content/en/

Meus truques preferidos em Python – Parte I
https://leportella.com/pt-br/2018/05/07/pytricks-I.html

Python Performance Tips
https://nyu-cds.github.io/python-performance-tips/

Map vs List comprehension in Python
https://dev.to/lyfolos/map-vs-list-comprehension-in-python-2ljj

ATV1 – Preparar ambiente para Atividades

Objetivo: Implementar um protótipo de *software* com funções para entrada de dados e saída dos resultados.

Considerando os arquivos de instâncias de grafos disponibilizados:

- Desenvolver um protótipo de *software* que faça a leitura do arquivo de uma dada instância e salve um determinado resultado em outro arquivo;
- O nome da instância deverá ser passado como argumento para o método no comando de execução;
- Uma função de entrada deverá ler o conteúdo do arquivo da respectiva instância e armazená-lo em uma matriz do tipo *numpy (consulte a documentação da biblioteca em https://numpy.org)*;
- Obter a dimensão da matriz (i.e. quantidade de linhas e de colunas);
- Uma função de saída deverá salvar em arquivo o nome da instância e a dimensão da matriz no formato: *nome_instância = (qtd_linhas, qtd_colunas)*.