MAC122 – Princípios de Desenvolvimento de Algoritmos
Segundo Semestre de 2019 – BMAC – IMEUSP – Prof. Marcilio
EXERCÍCIO PROGRAMA 3 – Entregar até 16/Nov/2019

Dado um arquivo de texto, onde cada linha contém os seguintes campos separados por vírgula:

<nome>,<data nascimento>,<identificação>

<nome> - máximo de 40 caracteres
<data nascimento> - no formato dd/mm/aa – 8 caracteres
<identificação> - 11 caracteres

Exemplo:

```
Jose de Castro,10/12/2001,32343245100
Maria das Dores e Silva,25/01/2005,43456790236
Antonio Paixao dos Santos,30/05/2012,22589349476
...
```

O programa deve classificar o arquivo (origem) pelo <nome>, dentro deste pela <identificação> e dentro desta, pela <data nascimento>. O resultado deve ser armazenado num outro arquivo (destino).

Para tornar a classificação rápida, é melhor ler o arquivo todo, colocando-o numa lista **TAB,** onde cada elemento é uma string com uma linha do arquivo e classificar esta lista de acordo com o critério estabelecido acima.

A classificação da lista deve ser feita por 3 métodos: Quick recursivo, Quick Não recursivo e usando a o método sort() do Python. O objetivo é mostrar o tempo decorrido para classificar em cada um dos métodos.

Exemplo de saída do programa:

```
Entre com o nome do arquivo origem:meuarq1.txt
Entre com o nome do arquivo destino:meuarq2.txt

Quantidade de registros a classificar: 250 registros

Tempo para classificar a tabela:
Método Quick Recursivo: xxx segundos
Método Quick Não Recursivo: yyy segundos
Método sort() do Python: zzz segundos
```

Escreva então as duas funções de classificação para a tabela TAB:

```
ClassQuickRecursivo(TAB)
ClassQuickNaoRecursivo(TAB)
classMetodoSort(TAB)
```

Escreva também uma função que verifique se a tabela está classificada, devolvendo True ou False (quando houve algum erro na classificação):

```
VerifClass(TAB)
# verifica se TAB[i] ≤ TAB[i+1]
```

O programa fica então:

```
entre com o nome do arquivo origem
entre com o nome do arquivo destino
leia o arquivo 1 e coloque em TAB
classifique TAB pelo primeiro método cronometrando o tempo
verifique se TAB está classificada
classifique TAB pelo segundo método cronometrando o tempo
verifique se TAB está classificada
classifique TAB pelo terceiro método cronometrando o tempo
verifique se TAB está classificada
```

Repita o trecho acima até que seja digitado “fim” com nome do arquivo origem.

O programa gerador de testes

Use o programa abaixo para gerar arquivos de teste para o seu programa.
Gere por exemplo arquivos com 100, 200 ou 1000 registros usando o programa abaixo.

```
from random import seed, randrange
# nomes randômicos
n1 = ["Felicia", "Catulo", "Osmund", "Artmio", "Senizio", "Tilenio"]
n2 = ["Cartuxo", "Olambro", "Romulo", "Ambulo", "Atomon", "Virino"]
n3 = ["Sereno", "Soterno", "Moncoes", "Oscaran", "Topovi", "Talento"]
n4 = ["Lasmia", "Mantega", "Casas", "Lorentao", "Melkioz", "Motivio"]
nn = 6

# gera um registro com NOME[0..39], DATAN[40..47] e IDENT[48..55]
# conteúdo randômico baseado em seu NUSP
# pp = '' - gera um registro completo
# pp != '' - gera apenas uma nova datan + ident ou apenas ident
def GeraRegistro(pp):
    global n1, n2, n3, n4, nn
    # nome, datan e ident
    nome = n1[randrange(nn)] + ' ' + n2[randrange(nn)] + ' ' +
n3[randrange(nn)] + ' ' + n4[randrange(nn)]
    dia = randrange(28) + 1
    mes = randrange(12) + 1
    ano = randrange(17) + 2000
    datan = f'{dia:02}' + '/' + f'{mes:02}' + '/' + f'{ano:04}'
    ident = f'{randrange(100000000000):011}'
    if pp == '':
        # gera um novo registro completo
        registro = nome + ',' + datan + ',' + ident
        return registro
    elif randrange(2) == 0:
        # preserva o nome e gera datan + ident
        campos = pp.split(',')
        registro = campos[0] + ',' + datan + ',' + ident
```

```
        return registro
    else:
        # preserva o nome e datan e gera ident
        campos = pp.split(',')
        registro = campos[0] + ',' + campos[1] + ',' + ident
        return registro

# gera arquivo nomearq com nreg registros
def GeraArquivo(nusp, nomearq, nreg):
    # randomize
    seed(nusp)
    # quantidade de registros - gera 80% do total
    nreg80 = nreg * 80 // 100
    # tabela para guardar registros para repetição
    tab = ['' for k in range(nreg // 20)] # 5% dos registros
    # abre arquivo para gravação
    arq = open(nomearq, "w")
    # grava metade dos registros
    for k in range(nreg80):
        reg = GeraRegistro('')
        arq.write(reg + '\n')
        print(k + 1, " - ", reg)
        # guarda 5% dos registros para repetição
        if k % 20 == 0:
            # guarda em tab
            tab[k // 20] = reg
    # grava o resto dos 20% dos registros
    cont = nreg80 + 1
    for k in range(len(tab)):
        # para cada registro em tab gera 4 outros
        for j in range(4):
            reg = GeraRegistro(tab[k])
            arq.write(reg + '\n')
            print(cont, " - ", reg)
            cont += 1
    # fecha arquivo
    arq.close()

# Entre com seu NUSP - para randomizar
nusp = int(input("Entre com seu NUSP - para randomizar:"))
# Gera arquivo com uma certa quantidade de registros
while True:
    nome_arq = input("Entre com o nome do arquivo.txt:")
    quant_reg = int(input("Entre com a quantidade de registros:"))
    GeraArquivo(nusp, nome_arq, quant_reg)
```