1 2 9 0 FACULDADE DE CIÊNCIAS E TECNOLOGIA UNIVERSIDADE D COIMBRA DEPARTAMENTO DE ENGENHARIA INFORMÁTICA	**Trabalho no 4 v2 – Palavras mais procuradas** **Algoritmos e Estruturas de Dados** **2019/2020 – 2º Semestre** Upload: (link a disponibilizar no infoestudante) Data Limite: até 15mn depois do fim da respetiva 4ª aula pratica

O RELATÓRIO E LISTAGEM DO CÓDIGO DESENVOLVIDO DEVEM SER SUBMETIDOS NUM ÚNICO DOCUMENTO PDF

Nome: _Dário Filipe Torres Félix_ nº _2018275530_ PL: _PL2_

Nº de horas de trabalho: Aulas Práticas de Laboratório: _6_H Fora de Sala de Aula: _5_H

CLASSIFICAÇÃO: (A Preencher pelo Docente)

Análise Empírica de Complexidade

- Faça o download dos 4 textos. Preencha a Tabela 1 com os valores para cada texto. Use o tempo médio de 20 execuções. Considere essa caracterização na análise qualitativa mais adiante.
- Preencha a Tabela 2 com informação sobre as estruturas de dados e algoritmos que escolheu.
- Preencha a Tabela 3 com os tempos de ordenamento para cada algoritmo.

#	Operação	Texto A	Texto B	Texto C	Texto D
1	Carregamento (tempo em: _Milissegundo [ms]_)	0,387	2 431,561	3 038,701	2 077,105
2	Núm. palavras distintas	75	1354	1354	1354
3	Núm. utilizadores distintos	2	632	632	632
4	Núm. Pares (palavra,utilizador) distintos	150	4 159	4 159	4 159
5	Núm. Total de palavras (Entradas)	156	140 000	140 000	140 000
6	Núm. Total de utilizadores (Entradas)	156	140 000	140 000	140 000
7	Faz algum ordenamento? Caracterize.	Poucas palavras. Palavras e IDs ordenados.	Muitas palavras Palavras e IDs aleatórios.	Muitas palavras, repetidas e ordenadas.	Palavras e IDs repetidas e ordenadas.

#	Tarefa	Estrutura de dados	Algoritmo de ordenamento
8	A1	Arrays	Quicksort (e Insertion Sort)
9	A2	Arrays	Radix Sort

#	Tempos de Ordenamento em [Milissegundo (ms)]	A1 PESQ_GLOBAL	A1 PESQ_UTILIZADORES	A2 PESQ_GLOBAL	A2 PESQ_UTILIZADORES
10	Texto A	0,001	0,034	0,090	0,050
11	Texto B	2,414	2,767	3,423	1,155
12	Texto C	4,497	4,044	2,436	1,166
13	Texto D	1,962	2,020	3,671	1,394

Reflexão sucinta sobre os resultados obtidos

(Formato de referência: Arial 10pt; texto para além do número de linhas não é considerado e desvaloriza o relatório)

1. Comente como as estruturas de armazenamento definidas na Tabela 2 influenciaram os tempos de armazenamento e processamento.

Sim. Sendo a estrutura de armazenamento (vários) arrays e não havendo qualquer tipo de ordenação por inserção, (e por isso) não havendo, por exemplo, pesquisa binária, a cada inserção, a função fica limitada a percorrer todo o array (maior O).

2. Analise os resultados dos algoritmos para as tarefas A1 e A2 e os resultados. Os resultados foram os esperados? O input mostrou-se relevante? Justifique.

Para o texto A, o QuickSort (A1) mostrou-se mais competente do que o Radix Sort (A2), pois por serem contagens pequenas, ao trabalhar ao nível da base fica mais demoroso do que uma simples comparação de chaves. Para os textos B, C e D, na Pesq_Utilizadores, entre o A1 e o A2, o A2 teve melhor desempenho, pois as contagens aqui são menores ou iguais do que na Pesq_Global, e que o Quicksort tem complexidade O(nlog(n)) e o Radix Sort tem complexidade O(kn), em que k é o número de dígitos da chave, e n o tamanho do array. O Quicksort não tem o melhor desempenho quando as chaves não são únicas, ou seja, quando há muitas chaves iguais.

3. Compare as duas formas de ordenamento: por registo vs. por endereço. Quando será apropriada a escolha de cada uma?

É usado ordenamento por registo ao invés de ordenamento por endereço, pois o tamanho dos registos neste trabalho não é relativamente grande que justifique ordenamento por endereços. Os registos contêm um número inteiro indicativo do número de vezes da ocorrência de uma determinada palavra.

Bom trabalho, os Docentes da Disciplina,

Carlos L Bento e Catarina Silva

A0 – Parte 1/3 (Usado na Tabela 1 – Adaptado do A0, Insertion Sort)

```python
# v1: Com Insertion Sort dos arrays finais (A0) | Especial Relatorio
# *** VERSAO RELATORIO *** | Tabela 1

# #### BIBLIOTECAS ####
import sys
import time
import msvcrt
from io import StringIO


# #### CONSTANTES ####
CMD_IN_GLOBAL = "PESQ_GLOBAL\n"
CMD_IN_UTILIZADORES = "PESQ_UTILIZADORES\n"
CMD_IN_TERMINADO = "TCHAU\n"
CMD_IN_TERMINADO2 = "TCHAU"
CMD_IN_PALAVRAS = "PALAVRAS\n"
CMD_IN_FIM = "FIM.\n"
CMD_OUT_GUARDADO = "GUARDADAS"


# #### FUNCOES ####
def main():
    # ### FUNCAO ### Funcao Principal
    array_palavras = []                     # [omg, xd, a, ahah] | Input "palavra + ID"
    array_count_global = []                 # [3, 1, 10, 2] ou [[3, 0], [1, 1], [10, 2], [2, 3]] [Count, Indice]
    array_count_utilizadores = []           # [2, 1, 5, 2] > [[Count, Indice], ...]
    array_ids_utilizadores = []             # [[109, 114], [109], [455,677,232,124,345], [098,345]] , IDs - Diferentes
    n_total = 0
    textos_relatorio = ["A", "B", "C", "D"]

    for n_texto in textos_relatorio:
        print("# # # # # # TEXTO " + n_texto + " # # # # # #")
        nome_fich = "./StdinsCalculaTempos/StdinTexto" + n_texto + "RelatorioF4.txt"
        acumula_texto = 0
        n_palavras_distintas = 0
        n_utilizadores_distintos = 0
        n_pares_distintas = 0
        for i in range(20):
            print("###### Tentativa " + str(i+1) + " ######")
            array_palavras = []  # [omg, xd, a, ahah] | Input "palavra + ID"
            array_count_global = []  # [3, 1, 10, 2] ou [[3, 0], [1, 1], [10, 2], [2, 3]] [Count, Indice]
            array_count_utilizadores = []  # [2, 1, 5, 2] > [[Count, Indice], ...]
            array_ids_utilizadores = []  # [[109, 114], [109], [455,677,232,124,345], [098,345]] , IDs - Diferentes
            n_total = 0

            while msvcrt.kbhit():  # Clean stdin Windows
                msvcrt.getch()

            my_file = open(nome_fich, "r")
            my_stdin = my_file.read()
            my_file.close()
            sys.stdin = StringIO(my_stdin)

            if sys.stdin.readline() == CMD_IN_PALAVRAS:
                start_texto = time.time()
                array_palavras, array_count_global, array_count_utilizadores, array_ids_utilizadores, n_total = \
                    input_palavras(array_palavras, array_count_global, array_count_utilizadores, array_ids_utilizadores)
                end_texto = time.time()
                tempo_texto = end_texto - start_texto
                print("*Tempo em MS - Carregamento/Input TEXTO = " + str(tempo_texto * 1000) + " ||| Start Vs End: " +
                      str(start_texto) + "|" + str(end_texto) + " *")
                acumula_texto = acumula_texto + tempo_texto
            else:
                sys.exit("Erro - Sem Comando Incial: " + CMD_IN_PALAVRAS)

```

```
        acumula_texto = (acumula_texto / 20.0) * 1000
        print("***** Tempo Medio em MS - Carregamento/Input TEXTO = " + str(acumula_texto) + " *****")
        print("########################################\n")

        n_palavras_distintas = len(array_palavras)
        n_pares_distintas = 0
        for i in range(len(array_ids_utilizadores)):
            n_pares_distintas = n_pares_distintas + len(array_ids_utilizadores[i])
        array_utilizadores_distintos = []
        for i in range(len(array_ids_utilizadores)):
            for j in range(len(array_ids_utilizadores[i])):
                if not array_ids_utilizadores[i][j] in array_utilizadores_distintos:
                    array_utilizadores_distintos.append(array_ids_utilizadores[i][j])
        n_utilizadores_distintos = len(array_utilizadores_distintos)
        print("\n########################################")
        print("***** Numero de Palavras Distintas = " + str(n_palavras_distintas) + " *****")
        print(array_palavras)
        print("***** Numero de Utilizadores Distintos = " + str(n_utilizadores_distintos) + " *****")
        print(array_utilizadores_distintos)
        print("***** Numero de Pares Distintos (Palavra-Utilizador) = " + str(n_pares_distintas) + " *****")
        print(array_ids_utilizadores)
        print("***** Numero Total de Palavras/Utilizadores (Entradas) = " + str(n_total) + " *****")
        print("+++++++++")
        input_cmd(array_palavras, array_count_global, array_count_utilizadores)
        print("+++++++++")
        print("########################################\n\n")

    return 0


def input_palavras(array_palavras, array_count_global, array_count_utilizadores, array_ids_utilizadores):
    # ### FUNCAO ### Le e manipula o texto do stdin ate CMD_IN_FIM
    n_total = 0
    for linha in sys.stdin:
        if linha == "\n" or linha == "":
            sys.exit("Erro - Sem Texto para input")
        if linha == CMD_IN_FIM:
            break
        n_total = n_total + 1
        palavras = linha.split(" ")
        palavras[0] = palavras[0].upper()
        palavras[1] = palavras[1][:-1]
        if palavras[0] in array_palavras:
            indice = array_palavras.index(palavras[0])
            array_count_global[indice][0] += 1
            if not int(palavras[1]) in array_ids_utilizadores[indice]:
                array_ids_utilizadores[indice].append(int(palavras[1]))
                array_count_utilizadores[indice][0] += 1
        else:
            array_palavras.append(palavras[0])
            indice = len(array_palavras) - 1
            array_ids_utilizadores.append([int(palavras[1])])
            array_count_global.append([1, indice])
            array_count_utilizadores.append([1, indice])
    print(CMD_OUT_GUARDADO)
    return array_palavras, array_count_global, array_count_utilizadores, array_ids_utilizadores, n_total

```

```python
def input_cmd(array_palavras, array_count_global, array_count_utilizadores):
    # ### FUNCAO ### Le, executa e escreve no stdout os comandos no stdin, ate CMD_IN_TERMINADO
    for linha in sys.stdin:
        if linha == CMD_IN_TERMINADO2:
            break
        elif linha == CMD_IN_TERMINADO:
            break
        elif linha == "":
            break
        elif linha == CMD_IN_GLOBAL:
            array_count_global = ordenacao(array_count_global)
            string = ""
            valor = array_count_global[-1][0]
            start = len(array_palavras) - 1
            for i in range(len(array_palavras)-1, -1, -1):
                if valor == array_count_global[i][0]:
                    start = i
                else:
                    break
            alvo = []
            for i in range(start, len(array_palavras)):
                indice = array_count_global[i][1]
                alvo.append(array_palavras[indice])
            alvo.sort()
            for i in range(len(alvo)):
                string = string + str(alvo[i]) + " "
            print(string[:-1])
        elif linha == CMD_IN_UTILIZADORES:
            array_count_utilizadores = ordenacao(array_count_utilizadores)
            string = ""
            valor = array_count_utilizadores[-1][0]
            start = len(array_palavras)-1
            for i in range(len(array_palavras)-1, -1, -1):
                if valor == array_count_utilizadores[i][0]:
                    start = i
                else:
                    break
            alvo = []
            for i in range(start, len(array_palavras)):
                indice = array_count_utilizadores[i][1]
                alvo.append(array_palavras[indice])
            alvo.sort()
            for i in range(len(alvo)):
                string = string + str(alvo[i]) + " "
            print(string[:-1])
        else:
            sys.exit("Erro - Interpretacao dos comandos pos-palavras")
    return 0


def insertion_sort(array):
    # ### FUNCAO ### Insertion Sort para Inteiros
    for i in range(1, len(array)):
        temp = array[i]
        j = i - 1
        while j >= 0 and temp[0] < array[j][0]:   # ORDEM CRESCENTE
            array[j + 1] = array[j]
            j = j - 1
        array[j + 1] = temp
    return array


def ordenacao(array):
    # ### FUNCAO ### *Abstracao* -> Chama a funcao de ordenamento
    array = insertion_sort(array)
    return array


if __name__ == '__main__':
    # ### START ###
    main()

```

A1 – Parte 1/3 (Para Tabela 2 e 3 – Quicksort + Insertion Sort)

```python
# v1: Ordenacao para "arrays alvo" -> Com QuickSort (A1) para array "incial", com InsertionSort para array com os elementos finais
# *** VERSAO RELATORIO *** | Tabela 2 e 3


# #### BIBLIOTECAS ####
import sys
import time
import msvcrt
from io import StringIO


# #### CONSTANTES ####
CMD_IN_GLOBAL = "PESQ_GLOBAL\n"
CMD_IN_UTILIZADORES = "PESQ_UTILIZADORES\n"
CMD_IN_TERMINADO = "TCHAU\n"
CMD_IN_TERMINADO2 = "TCHAU"
CMD_IN_PALAVRAS = "PALAVRAS\n"
CMD_IN_FIM = "FIM.\n"
CMD_OUT_GUARDADO = "GUARDADAS"
PARAGEM_CORTE = 30


# #### FUNCOES ####
def main():
    # ### FUNCAO ### Funcao Principal
    array_palavras = []                     # [omg, xd, a, ahah] | Input "palavra + ID"
    array_count_global = []                 # [3, 1, 10, 2] ou [[3, 0], [1, 1], [10, 2], [2, 3]] [Count, Indice]
    array_count_utilizadores = []           # [2, 1, 5, 2] > [[Count, Indice], ...]
    # array_utilizadores = []               # [[109, 114], [109], [455,677,232,124,345], [098,345]] , IDs - Diferentes
    textos_relatorio = ["A", "B", "C", "D"]

    for n_texto in textos_relatorio:
        print("# # # # # # # TEXTO " + n_texto + " # # # # # # #")
        nome_fich = "./StdinsCalculaTempos/StdinTexto" + n_texto + "RelatorioF4.txt"
        array_palavras = []  # [omg, xd, a, ahah] | Input "palavra + ID"
        array_count_global = []  # [3, 1, 10, 2] ou [[3, 0], [1, 1], [10, 2], [2, 3]] [Count, Indice]
        array_count_utilizadores = []  # [2, 1, 5, 2] > [[Count, Indice], ...]
        acumula_global = 0
        acumula_utilizadores = 0

        while msvcrt.kbhit():  # Clean stdin Windows
            msvcrt.getch()

        my_file = open(nome_fich, "r")
        my_stdin = my_file.read()
        my_file.close()
        sys.stdin = StringIO(my_stdin)

        if sys.stdin.readline() == CMD_IN_PALAVRAS:
            array_palavras, array_count_global, array_count_utilizadores = input_palavras(array_palavras, array_count_global,
                                                                                          array_count_utilizadores)
        else:
            sys.exit("Erro - Sem Comando Incial: " + CMD_IN_PALAVRAS)
        print("+++++++++")
        print(array_palavras)
        print("+++++++++")
        for i in range(20):
            print("###### Tentativa " + str(i+1) + " ######")
            temp_array_count_global = []
            temp_array_count_utilizadores = []
            for j in range(len(array_palavras)):
                temp_array_count_global.append(array_count_global[j])
                temp_array_count_utilizadores.append(array_count_utilizadores[j])
            tempo_global, tempo_utilizadores = input_cmd(array_palavras, temp_array_count_global, temp_array_count_utilizadores)
            acumula_global = acumula_global + tempo_global
            acumula_utilizadores = acumula_utilizadores + tempo_utilizadores
            print("##############\n")

        print("########################################")
        acumula_global = (acumula_global / 20.0) * 1000
        acumula_utilizadores = (acumula_utilizadores / 20.0) * 1000
        print("***** Tempo MEDIO em MS - PESQUISA GLOBAL = " + str(acumula_global) + " *****")
        print("***** Tempo MEDIO em MS - PESQUISA UTILIZADORES = " + str(acumula_utilizadores) + " *****")
        print("########################################\n\n")

    return 0
```

```python
def input_palavras(array_palavras, array_count_global, array_count_utilizadores):
    # ### FUNCAO ### Le e manipula o texto do stdin ate CMD_IN_FIM
    array_ids_utilizadores = []             # [[109, 114], [109], [455,677,232,124,345], [098,345]] , IDs - Diferentes
    for linha in sys.stdin:
        if linha == "\n" or linha == "":
            sys.exit("Erro - Sem Texto para input")
        if linha == CMD_IN_FIM:
            break
        palavras = linha.split(" ")
        palavras[0] = palavras[0].upper()
        palavras[1] = palavras[1][:-1]
        if palavras[0] in array_palavras:
            indice = array_palavras.index(palavras[0])
            array_count_global[indice][0] += 1
            if not int(palavras[1]) in array_ids_utilizadores[indice]:
                array_ids_utilizadores[indice].append(int(palavras[1]))
                array_count_utilizadores[indice][0] += 1
        else:
            array_palavras.append(palavras[0])
            indice = len(array_palavras)-1
            array_ids_utilizadores.append([int(palavras[1])])
            array_count_global.append([1, indice])
            array_count_utilizadores.append([1, indice])
    print(CMD_OUT_GUARDADO)
    return array_palavras, array_count_global, array_count_utilizadores


def input_cmd(array_palavras, array_count_global, array_count_utilizadores):
    # ### FUNCAO ### Le, executa e escreve no stdout os comandos no stdin, ate CMD_IN_TERMINADO
    tempo_global = tempo_utilizadores = 0

    start_cmd = time.time()
    array_count_global = ordenacao(array_count_global)
    end_cmd = time.time()
    tempo_global = end_cmd - start_cmd
    print("*Tempo em MS - CMD-PesquisaGlobal = " + str(tempo_global * 1000) + " ||| Start Vs End: " + str(start_cmd) +
          "|" + str(end_cmd) + " *")

    string = ""
    valor = array_count_global[-1][0]
    start = len(array_palavras) - 1
    for i in range(len(array_palavras)-1, -1, -1):
        if valor == array_count_global[i][0]:
            start = i
        else:
            break
    alvo = []
    for i in range(start, len(array_palavras)):
        indice = array_count_global[i][1]
        alvo.append(array_palavras[indice])
    alvo.sort()
    for i in range(len(alvo)):
        string = string + str(alvo[i]) + " "
    print(string[:-1])

    start_cmd = time.time()
    array_count_utilizadores = ordenacao(array_count_utilizadores)
    end_cmd = time.time()
    tempo_utilizadores = end_cmd - start_cmd
    print("*Tempo em MS - CMD-PesquisaUtilizadores = " + str(tempo_utilizadores * 1000) + " ||| Start Vs End: " +
          str(start_cmd) + "|" + str(end_cmd) + " *")
    print(array_count_utilizadores)
    string = ""
    valor = array_count_utilizadores[-1][0]
    start = len(array_palavras)-1
    for i in range(len(array_palavras)-1, -1, -1):
        if valor == array_count_utilizadores[i][0]:
            start = i
        else:
            break
    alvo = []
    for i in range(start, len(array_palavras)):
        indice = array_count_utilizadores[i][1]
        alvo.append(array_palavras[indice])
    alvo.sort()
    for i in range(len(alvo)):
        string = string + str(alvo[i]) + " "
    print(string[:-1])

    return tempo_global, tempo_utilizadores
```

```python
def insertion_sort(array, indice_baixo, indice_alto):
    for i in range(indice_baixo, indice_alto + 1):
        temp = array[i]         # Elemento a comparar
        j = i - 1               # Começa com o elemento a baixo do temp
        while j >= indice_baixo and temp[0] < array[j][0]:  # ORDEM CRESCENTE, enquanto temp MENOR array[j] -> os elementos sobem um de
            array[j + 1] = array[j]
            j = j - 1
        array[j + 1] = temp             # temp fica abaixo dos demais que subiram 1 degrau
    return array


def quick_sort(array, indice_baixo, indice_alto):
    # ### FUNCAO ### Quick Sort para Inteiros
    if PARAGEM_CORTE > (indice_alto - indice_baixo):        # Array < 30 elementos -> Insertion
        array = insertion_sort(array, indice_baixo, indice_alto)
    else:
        indice_meio = int((indice_baixo + indice_alto) / 2)
        if array[indice_alto][0] < array[indice_baixo][0]:    # Mediana
            temp = array[indice_alto]
            array[indice_alto] = array[indice_baixo]
            array[indice_baixo] = temp
        if array[indice_meio][0] < array[indice_baixo][0]:
            temp = array[indice_meio]
            array[indice_meio] = array[indice_baixo]
            array[indice_baixo] = temp
        if array[indice_meio][0] < array[indice_alto][0]:
            temp = array[indice_meio]
            array[indice_meio] = array[indice_alto]
            array[indice_alto] = temp

        pivot = array[indice_meio][0]         # Nomeia PIVOT
        temp = array[indice_meio]                # Troca pivot  com (ultimo_elemento - 1)
        array[indice_meio] = array[indice_alto - 1]
        array[indice_alto - 1] = temp

        ptr_baixo = indice_baixo
        ptr_alto = indice_alto - 1
        while True:   # Percorre parte-do-array com ponteiros, a semelhanca nos slides 72 a 80 - Cap 6A ORD COM CHAVES
            ptr_alto = ptr_alto - 1
            ptr_baixo = ptr_baixo + 1
            while array[ptr_alto][0] > pivot:
                ptr_alto = ptr_alto - 1
            while array[ptr_baixo][0] < pivot:
                ptr_baixo = ptr_baixo + 1
            if ptr_baixo < ptr_alto:     # Troca
                temp = array[ptr_alto]
                array[ptr_alto] = array[ptr_baixo]
                array[ptr_baixo] = temp
            else:
                break

        temp = array[indice_meio]    # Troca NOVAMENTE pivot  no (ultimo_elemento - 1) com ptr_baixo =aprox= meio (backup)
        array[indice_meio] = array[indice_alto - 1]
        array[indice_alto - 1] = temp

        array = quick_sort(array, indice_baixo, ptr_baixo - 1)    # Parte o array em dois
        array = quick_sort(array, ptr_baixo + 1, indice_alto)

    return array


def ordenacao(array):
    # ### FUNCAO ### *Abstracao* -> Chama a funcao de ordenamento
    array = quick_sort(array, 0, len(array) - 1)
    return array


if __name__ == '__main__':
    # ### START ###
    main()
```

A2 – Parte 1/1 (As funções em falta são iguais ao A1 | Para Tabela 2 e 3 – Radix Sort)

```python
# v1: Com Radix Sort LSD (A2) (Lado Direito -> Lado Esquerdo) Atua Sobre Digitos e Nao Bits (Para inteiros) |
# Ex.: 1999 >>> 1 <- 9 <- 9 <- 9
# *** VERSAO RELATORIO *** | Tabela 2 e 3


def radix_sort(array, tamanho):
    # ### FUNCAO ### Radix Sort para Inteiros

    max_digitos = len(str(max(array[:][0])))            # Numero Maior no Array -> Quantos Digitos?

    array_contador = []                                 # Contar cada digito
    array_semiordenado = []                             # Auxiliar: array -> sort -> array_semiordenado -> copia -> array

    divisor = 1
    for j in range(max_digitos):                        # Passar por todos os digitos do numero
        for i in range(10):
            array_contador.append(0)
        for i in range(tamanho):
            array_semiordenado.append(0)

        for i in range(tamanho):                        # Contar as ocorrencias de cada digito
            digito = int((array[i][0]/divisor) % 10)
            array_contador[digito] = array_contador[digito] + 1

        temp = temp_anterior = 0
        for i in range(1, 10):                          # array_contador[i] -> fica com a posicao onde se colocam os numeros com
                                                                                                    # este digito no array semiordenado
            temp = array_contador[i]
            array_contador[i] = array_contador[i-1] + temp_anterior     # Ver Exemplo/Explicacao nas Notas-iPad
            temp_anterior = temp
        array_contador[0] = 0                           # Digitos com comecam no Indice 0 (e o primeiro)

        for i in range(tamanho):                        # Semi-Ordena com base nos digitos e posicao no array_contador
            digito = int((array[i][0]/divisor) % 10)
            array_semiordenado[array_contador[digito]] = array[i]
            array_contador[digito] = array_contador[digito] + 1

        for i in range(tamanho):                        # COPIA: De 'array_semiordenado' Para 'array'
            array[i] = array_semiordenado[i]

        divisor = divisor * 10                          # *** Prepara a proxima iteracao do ciclo ***
        array_contador = []
        array_semiordenado = []

    return array


def ordenacao(array):
    # ### FUNCAO ### *Abstracao* -> Chama a funcao de ordenamento
    array = radix_sort(array, len(array))
    return array


if __name__ == '__main__':
    # ### START ###
    main()
```