

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: <u>Milissegundo [ms]</u>)	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
I0	Texto A	0,001	0,034	0,090	0,050
I1	Texto B	2,414	2,767	3,423	1,155
I2	Texto C	4,497	4,044	2,436	1,166
I3	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 demorado 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 $O(n\log(n))$ 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)

```
1  # v1: Com Insertion Sort dos arrays finais (A0) | Especial Relatorio
2  # *** VERSAO RELATORIO *** | Tabela 1
3
4  # ##### BIBLIOTECAS #####
5  import sys
6  import time
7  import msvcrt
8  from io import StringIO
9
10
11 # ##### CONSTANTES #####
12 CMD_IN_GLOBAL = "PESQ_GLOBAL\n"
13 CMD_IN_UTILIZADORES = "PESQ_UTILIZADORES\n"
14 CMD_IN_TERMINADO = "TCHAU\n"
15 CMD_IN_TERMINADO2 = "TCHAU"
16 CMD_IN_PALAVRAS = "PALAVRAS\n"
17 CMD_IN_FIM = "FIM.\n"
18 CMD_OUT_GUARDADO = "GUARDADAS"
19
20
21 # ##### FUNCOES #####
22 def main():
23     # ### FUNCAO ### Funcao Principal
24     array_palavras = [] # [omg, xd, a, ahah] | Input "palavra + ID"
25     array_count_global = [] # [3, 1, 10, 2] ou [[3, 0], [1, 1], [10, 2], [2, 3]] [Count, Indice]
26     array_count_utilizadores = [] # [2, 1, 5, 2] > [[Count, Indice], ...]
27     array_ids_utilizadores = [] # [[109, 114], [109], [455,677,232,124,345], [098,345]] , IDs - Diferentes
28     n_total = 0
29     textos_relatorio = ["A", "B", "C", "D"]
30
31     for n_texto in textos_relatorio:
32         print("# # # # # TEXTO " + n_texto + " # # # # #")
33         nome_fich = "./StdinsCalculaTempos/StdinTexto" + n_texto + "RelatorioF4.txt"
34         acumula_texto = 0
35         n_palavras_distintas = 0
36         n_utilizadores_distintos = 0
37         n_pares_distintas = 0
38         for i in range(20):
39             print("##### Tentativa " + str(i+1) + " #####")
40             array_palavras = [] # [omg, xd, a, ahah] | Input "palavra + ID"
41             array_count_global = [] # [3, 1, 10, 2] ou [[3, 0], [1, 1], [10, 2], [2, 3]] [Count, Indice]
42             array_count_utilizadores = [] # [2, 1, 5, 2] > [[Count, Indice], ...]
43             array_ids_utilizadores = [] # [[109, 114], [109], [455,677,232,124,345], [098,345]] , IDs - Diferentes
44             n_total = 0
45
46             while msvcrt.kbhit(): # Clean stdin Windows
47                 msvcrt.getch()
48
49             my_file = open(nome_fich, "r")
50             my_stdin = my_file.read()
51             my_file.close()
52             sys.stdin = StringIO(my_stdin)
53
54             if sys.stdin.readline() == CMD_IN_PALAVRAS:
55                 start_texto = time.time()
56                 array_palavras, array_count_global, array_count_utilizadores, array_ids_utilizadores, n_total = \
57                     input_palavras(array_palavras, array_count_global, array_count_utilizadores, array_ids_utilizadores)
58                 end_texto = time.time()
59                 tempo_texto = end_texto - start_texto
60                 print("*Tempo em MS - Carregamento/Input TEXTO = " + str(tempo_texto * 1000) + " ||| Start Vs End: " +
61                       str(start_texto) + "|" + str(end_texto) + " *")
62                 acumula_texto = acumula_texto + tempo_texto
63             else:
64                 sys.exit("Erro - Sem Comando Inicial: " + CMD_IN_PALAVRAS)
65
```

```

66     acumula_texto = (acumula_texto / 20.0) * 1000
67     print("**** Tempo Medio em MS - Carregamento/Input TEXTO = " + str(acumula_texto) + " ****")
68     print("#####\n")
69
70     n_palavras_distintas = len(array_palavras)
71     n_pares_distintas = 0
72     for i in range(len(array_ids_utilizadores)):
73         n_pares_distintas = n_pares_distintas + len(array_ids_utilizadores[i])
74     array_utilizadores_distintos = []
75     for i in range(len(array_ids_utilizadores)):
76         for j in range(len(array_ids_utilizadores[i])):
77             if not array_ids_utilizadores[i][j] in array_utilizadores_distintos:
78                 array_utilizadores_distintos.append(array_ids_utilizadores[i][j])
79     n_utilizadores_distintos = len(array_utilizadores_distintos)
80     print("\n#####")
81     print("**** Numero de Palavras Distintas = " + str(n_palavras_distintas) + " ****")
82     print(array_palavras)
83     print("**** Numero de Utilizadores Distintos = " + str(n_utilizadores_distintos) + " ****")
84     print(array_utilizadores_distintos)
85     print("**** Numero de Pares Distintos (Palavra-Utilizador) = " + str(n_pares_distintas) + " ****")
86     print(array_ids_utilizadores)
87     print("**** Numero Total de Palavras/Utilizadores (Entradas) = " + str(n_total) + " ****")
88     print("+++++")
89     input_cmd(array_palavras, array_count_global, array_count_utilizadores)
90     print("+++++")
91     print("#####\n\n")
92
93     return 0
94
95
96 def input_palavras(array_palavras, array_count_global, array_count_utilizadores, array_ids_utilizadores):
97     # ### FUNCAO ### Le e manipula o texto do stdin ate CMD_IN_FIM
98     n_total = 0
99     for linha in sys.stdin:
100         if linha == "\n" or linha == "":
101             sys.exit("Erro - Sem Texto para input")
102         if linha == CMD_IN_FIM:
103             break
104         n_total = n_total + 1
105         palavras = linha.split(" ")
106         palavras[0] = palavras[0].upper()
107         palavras[1] = palavras[1][:-1]
108         if palavras[0] in array_palavras:
109             indice = array_palavras.index(palavras[0])
110             array_count_global[indice][0] += 1
111             if not int(palavras[1]) in array_ids_utilizadores[indice]:
112                 array_ids_utilizadores[indice].append(int(palavras[1]))
113                 array_count_utilizadores[indice][0] += 1
114         else:
115             array_palavras.append(palavras[0])
116             indice = len(array_palavras) - 1
117             array_ids_utilizadores.append([int(palavras[1])])
118             array_count_global.append([1, indice])
119             array_count_utilizadores.append([1, indice])
120     print(CMD_OUT_GUARDADO)
121     return array_palavras, array_count_global, array_count_utilizadores, array_ids_utilizadores, n_total
122

```

```

124 def input_cmd(array_palavras, array_count_global, array_count_utilizadores):
125     # ### FUNCAO ### Le, executa e escreve no stdout os comandos no stdin, ate CMD_IN_TERMINADO
126     for linha in sys.stdin:
127         if linha == CMD_IN_TERMINADO2:
128             break
129         elif linha == CMD_IN_TERMINADO:
130             break
131         elif linha == "":
132             break
133         elif linha == CMD_IN_GLOBAL:
134             array_count_global = ordenacao(array_count_global)
135             string = ""
136             valor = array_count_global[-1][0]
137             start = len(array_palavras) - 1
138             for i in range(len(array_palavras)-1, -1, -1):
139                 if valor == array_count_global[i][0]:
140                     start = i
141                 else:
142                     break
143             alvo = []
144             for i in range(start, len(array_palavras)):
145                 indice = array_count_global[i][1]
146                 alvo.append(array_palavras[indice])
147             alvo.sort()
148             for i in range(len(alvo)):
149                 string = string + str(alvo[i]) + " "
150             print(string[:-1])
151         elif linha == CMD_IN_UTILIZADORES:
152             array_count_utilizadores = ordenacao(array_count_utilizadores)
153             string = ""
154             valor = array_count_utilizadores[-1][0]
155             start = len(array_palavras)-1
156             for i in range(len(array_palavras)-1, -1, -1):
157                 if valor == array_count_utilizadores[i][0]:
158                     start = i
159                 else:
160                     break
161             alvo = []
162             for i in range(start, len(array_palavras)):
163                 indice = array_count_utilizadores[i][1]
164                 alvo.append(array_palavras[indice])
165             alvo.sort()
166             for i in range(len(alvo)):
167                 string = string + str(alvo[i]) + " "
168             print(string[:-1])
169         else:
170             sys.exit("Erro - Interpretacao dos comandos pos-palavras")
171     return 0
172
173
174 def insertion_sort(array):
175     # ### FUNCAO ### Insertion Sort para Inteiros
176     for i in range(1, len(array)):
177         temp = array[i]
178         j = i - 1
179         while j >= 0 and temp[0] < array[j][0]: # ORDEM CRESCENTE
180             array[j + 1] = array[j]
181             j = j - 1
182         array[j + 1] = temp
183     return array
184
185
186 def ordenacao(array):
187     # ### FUNCAO ### *Abstracao* -> Chama a funcao de ordenamento
188     array = insertion_sort(array)
189     return array
190
191
192 if __name__ == '__main__':
193     # ### START ###
194     main()
195

```

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

```
1  # v1: Ordenacao para "arrays alvo" -> Com QuickSort (A1) para array "incial", com InsertionSort para array com os elementos finais
2  # *** VERSAO RELATORIO *** | Tabela 2 e 3
3
4
5  # #### BIBLIOTECAS ####
6  import sys
7  import time
8  import msvcrt
9  from io import StringIO
10
11
12  # #### CONSTANTES ####
13  CMD_IN_GLOBAL = "PESQ_GLOBAL\n"
14  CMD_IN_UTILIZADORES = "PESQ_UTILIZADORES\n"
15  CMD_IN_TERMINADO = "TCHAU\n"
16  CMD_IN_TERMINADO2 = "TCHAU"
17  CMD_IN_PALAVRAS = "PALAVRAS\n"
18  CMD_IN_FIM = "FIM.\n"
19  CMD_OUT_GUARDADO = "GUARDADAS"
20  PARAGEM_CORTE = 30
21
22
23  # #### FUNCOES ####
24  def main():
25      # ### FUNCAO ### Funcao Principal
26      array_palavras = [] # [omg, xd, a, ahah] | Input "palavra + ID"
27      array_count_global = [] # [3, 1, 10, 2] ou [[3, 0], [1, 1], [10, 2], [2, 3]] [Count, Indice]
28      array_count_utilizadores = [] # [2, 1, 5, 2] > [[Count, Indice], ...]
29      # array_utilizadores = [] # [[109, 114], [109], [455,677,232,124,345], [098,345]] , IDs - Diferentes
30      textos_relatorio = ["A", "B", "C", "D"]
31
32      for n_texto in textos_relatorio:
33          print("# # # # # TEXTO " + n_texto + " # # # # #")
34          nome_fich = "./StdinsCalculaTempos/StdinTexto" + n_texto + "RelatorioF4.txt"
35          array_palavras = [] # [omg, xd, a, ahah] | Input "palavra + ID"
36          array_count_global = [] # [3, 1, 10, 2] ou [[3, 0], [1, 1], [10, 2], [2, 3]] [Count, Indice]
37          array_count_utilizadores = [] # [2, 1, 5, 2] > [[Count, Indice], ...]
38          acumula_global = 0
39          acumula_utilizadores = 0
40
41          while msvcrt.kbhit(): # Clean stdin Windows
42              msvcrt.getch()
43
44          my_file = open(nome_fich, "r")
45          my_stdin = my_file.read()
46          my_file.close()
47          sys.stdin = StringIO(my_stdin)
48
49          if sys.stdin.readline() == CMD_IN_PALAVRAS:
50              array_palavras, array_count_global, array_count_utilizadores = input_palavras(array_palavras, array_count_global,
51                                                                                             array_count_utilizadores)
52          else:
53              sys.exit("Erro - Sem Comando Incial: " + CMD_IN_PALAVRAS)
54              print("+++++")
55              print(array_palavras)
56              print("+++++")
57              for i in range(20):
58                  print("##### Tentativa " + str(i+1) + " #####")
59                  temp_array_count_global = []
60                  temp_array_count_utilizadores = []
61                  for j in range(len(array_palavras)):
62                      temp_array_count_global.append(array_count_global[j])
63                      temp_array_count_utilizadores.append(array_count_utilizadores[j])
64                  tempo_global, tempo_utilizadores = input_cmd(array_palavras, temp_array_count_global, temp_array_count_utilizadores)
65                  acumula_global = acumula_global + tempo_global
66                  acumula_utilizadores = acumula_utilizadores + tempo_utilizadores
67                  print("#####\n")
68
69              print("#####")
70              acumula_global = (acumula_global / 20.0) * 1000
71              acumula_utilizadores = (acumula_utilizadores / 20.0) * 1000
72              print("***** Tempo MEDIO em MS - PESQUISA GLOBAL = " + str(acumula_global) + " *****")
73              print("***** Tempo MEDIO em MS - PESQUISA UTILIZADORES = " + str(acumula_utilizadores) + " *****")
74              print("#####\n")
75
76      return 0
```

A1 – Parte 2/3

```
79 def input_palavras(array_palavras, array_count_global, array_count_utilizadores):
80     # ### FUNCAO ### Le e manipula o texto do stdin ate CMD_IN_FIM
81     array_ids_utilizadores = [] # [[109, 114], [109], [455,677,232,124,345], [098,345]] , IDs - Diferentes
82     for linha in sys.stdin:
83         if linha == "\n" or linha == "":
84             sys.exit("Erro - Sem Texto para input")
85         if linha == CMD_IN_FIM:
86             break
87         palavras = linha.split(" ")
88         palavras[0] = palavras[0].upper()
89         palavras[1] = palavras[1][:-1]
90         if palavras[0] in array_palavras:
91             indice = array_palavras.index(palavras[0])
92             array_count_global[indice][0] += 1
93             if not int(palavras[1]) in array_ids_utilizadores[indice]:
94                 array_ids_utilizadores[indice].append(int(palavras[1]))
95                 array_count_utilizadores[indice][0] += 1
96         else:
97             array_palavras.append(palavras[0])
98             indice = len(array_palavras)-1
99             array_ids_utilizadores.append([int(palavras[1])])
100             array_count_global.append([1, indice])
101             array_count_utilizadores.append([1, indice])
102     print(CMD_OUT_GUARDADO)
103     return array_palavras, array_count_global, array_count_utilizadores
104
105
106 def input_cmd(array_palavras, array_count_global, array_count_utilizadores):
107     # ### FUNCAO ### Le, executa e escreve no stdout os comandos no stdin, ate CMD_IN_TERMINADO
108     tempo_global = tempo_utilizadores = 0
109
110     start_cmd = time.time()
111     array_count_global = ordenacao(array_count_global)
112     end_cmd = time.time()
113     tempo_global = end_cmd - start_cmd
114     print("**Tempo em MS - CMD-PesquisaGlobal = " + str(tempo_global * 1000) + " ||| Start Vs End: " + str(start_cmd) +
115           "|" + str(end_cmd) + " **")
116
117     string = ""
118     valor = array_count_global[-1][0]
119     start = len(array_palavras) - 1
120     for i in range(len(array_palavras)-1, -1, -1):
121         if valor == array_count_global[i][0]:
122             start = i
123         else:
124             break
125     alvo = []
126     for i in range(start, len(array_palavras)):
127         indice = array_count_global[i][1]
128         alvo.append(array_palavras[indice])
129     alvo.sort()
130     for i in range(len(alvo)):
131         string = string + str(alvo[i]) + " "
132     print(string[:-1])
133
134     start_cmd = time.time()
135     array_count_utilizadores = ordenacao(array_count_utilizadores)
136     end_cmd = time.time()
137     tempo_utilizadores = end_cmd - start_cmd
138     print("**Tempo em MS - CMD-PesquisaUtilizadores = " + str(tempo_utilizadores * 1000) + " ||| Start Vs End: " +
139           str(start_cmd) + "|" + str(end_cmd) + " **")
140     print(array_count_utilizadores)
141     string = ""
142     valor = array_count_utilizadores[-1][0]
143     start = len(array_palavras)-1
144     for i in range(len(array_palavras)-1, -1, -1):
145         if valor == array_count_utilizadores[i][0]:
146             start = i
147         else:
148             break
149     alvo = []
150     for i in range(start, len(array_palavras)):
151         indice = array_count_utilizadores[i][1]
152         alvo.append(array_palavras[indice])
153     alvo.sort()
154     for i in range(len(alvo)):
155         string = string + str(alvo[i]) + " "
156     print(string[:-1])
157
158     return tempo_global, tempo_utilizadores
```

A1 – Parte 3/3

```
161 def insertion_sort(array, indice_baixo, indice_alto):
162     for i in range(indice_baixo, indice_alto + 1):
163         temp = array[i]          # Elemento a comparar
164         j = i - 1                # Começa com o elemento a baixo do temp
165         while j >= indice_baixo and temp[0] < array[j][0]: # ORDEM CRESCENTE, enquanto temp MENOR array[j] -> os elementos sobem um de
166             array[j + 1] = array[j]
167             j = j - 1
168         array[j + 1] = temp      # temp fica abaixo dos demais que subiram 1 degrau
169     return array
170
171
172 def quick_sort(array, indice_baixo, indice_alto):
173     # ### FUNCAO ### Quick Sort para Inteiros
174     if PARAGEM_CORTE > (indice_alto - indice_baixo): # Array < 30 elementos -> Insertion
175         array = insertion_sort(array, indice_baixo, indice_alto)
176     else:
177         indice_meio = int((indice_baixo + indice_alto) / 2)
178         if array[indice_alto][0] < array[indice_baixo][0]: # Mediana
179             temp = array[indice_alto]
180             array[indice_alto] = array[indice_baixo]
181             array[indice_baixo] = temp
182         if array[indice_meio][0] < array[indice_baixo][0]:
183             temp = array[indice_meio]
184             array[indice_meio] = array[indice_baixo]
185             array[indice_baixo] = temp
186         if array[indice_meio][0] < array[indice_alto][0]:
187             temp = array[indice_meio]
188             array[indice_meio] = array[indice_alto]
189             array[indice_alto] = temp
190
191         pivot = array[indice_meio][0] # Nomeia PIVOT
192         temp = array[indice_meio]     # Troca pivot com (ultimo_elemento - 1)
193         array[indice_meio] = array[indice_alto - 1]
194         array[indice_alto - 1] = temp
195
196         ptr_baixo = indice_baixo
197         ptr_alto = indice_alto - 1
198         while True: # Percorre parte-do-array com ponteiros, a semelhança nos slides 72 a 80 - Cap 6A ORD COM CHAVES
199             ptr_alto = ptr_alto - 1
200             ptr_baixo = ptr_baixo + 1
201             while array[ptr_alto][0] > pivot:
202                 ptr_alto = ptr_alto - 1
203             while array[ptr_baixo][0] < pivot:
204                 ptr_baixo = ptr_baixo + 1
205             if ptr_baixo < ptr_alto: # Troca
206                 temp = array[ptr_alto]
207                 array[ptr_alto] = array[ptr_baixo]
208                 array[ptr_baixo] = temp
209             else:
210                 break
211
212         temp = array[indice_meio] # Troca NOVAMENTE pivot no (ultimo_elemento - 1) com ptr_baixo = aprox= meio (backup)
213         array[indice_meio] = array[indice_alto - 1]
214         array[indice_alto - 1] = temp
215
216         array = quick_sort(array, indice_baixo, ptr_baixo - 1) # Parte o array em dois
217         array = quick_sort(array, ptr_baixo + 1, indice_alto)
218
219     return array
220
221
222 def ordenacao(array):
223     # ### FUNCAO ### *Abstracao* -> Chama a funcao de ordenamento
224     array = quick_sort(array, 0, len(array) - 1)
225     return array
226
227
228 if __name__ == '__main__':
229     # ### START ###
230     main()
```


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

```
161 # v1: Com Radix Sort LSD (A2) (Lado Direito -> Lado Esquerdo) Atua Sobre Dígitos e Não Bits (Para inteiros) |
162 # Ex.: 1999 >>> 1 <- 9 <- 9 <- 9
163 # *** VERSÃO RELATÓRIO *** | Tabela 2 e 3
164
165
166 def radix_sort(array, tamanho):
167     # ### FUNÇÃO ### Radix Sort para Inteiros
168
169     max_dígitos = len(str(max(array[:]))) # Número Maior no Array -> Quantos Dígitos?
170
171     array_contador = [] # Contar cada dígito
172     array_semiordenado = [] # Auxiliar: array -> sort -> array_semiordenado -> cópia -> array
173
174     divisor = 1
175     for j in range(max_dígitos): # Passar por todos os dígitos do número
176         for i in range(10):
177             array_contador.append(0)
178         for i in range(tamanho):
179             array_semiordenado.append(0)
180
181         for i in range(tamanho): # Contar as ocorrências de cada dígito
182             dígito = int((array[i][0]/divisor) % 10)
183             array_contador[dígito] = array_contador[dígito] + 1
184
185         temp = temp_anterior = 0
186         for i in range(1, 10): # array_contador[i] -> fica com a posição onde se colocam os números com
187                                     # este dígito no array_semiordenado
188             temp = array_contador[i]
189             array_contador[i] = array_contador[i-1] + temp_anterior # Ver Exemplo/Explicação nas Notas-iPad
190             temp_anterior = temp
191         array_contador[0] = 0 # Dígitos comecem no Índice 0 (e o primeiro)
192
193         for i in range(tamanho): # Semi-Ordena com base nos dígitos e posição no array_contador
194             dígito = int((array[i][0]/divisor) % 10)
195             array_semiordenado[array_contador[dígito]] = array[i]
196             array_contador[dígito] = array_contador[dígito] + 1
197
198         for i in range(tamanho): # COPIA: De 'array_semiordenado' Para 'array'
199             array[i] = array_semiordenado[i]
200
201         divisor = divisor * 10 # *** Prepara a próxima iteração do ciclo ***
202         array_contador = []
203         array_semiordenado = []
204
205     return array
206
207
208 def ordenacao(array):
209     # ### FUNÇÃO ### *Abstração* -> Chama a função de ordenamento
210     array = radix_sort(array, len(array))
211     return array
212
213
214 if __name__ == '__main__':
215     # ### START ###
216     main()
```