```
arestas = [
    # Acre
    [4, 1, 7.8],
    [4, 2, 9],
    [2, 3, 620.5],
    [1, 3, 615],

    # Alagoas
    [7, 5, 38.4],
    [7, 8, 155],
    [5, 6, 266],
    [8, 6, 357],

    # Amapá
    [9, 12, 437],
    [9, 11, 1.6],
    [11, 10, 297.1],
    [12, 10, 272],

    # Amazonas
    [13, 16, 17],
    [13, 15, 303],
    [15, 14, 379.2],
    [16, 14, 660.7],

    # Bahia
    [17, 20, 1.7],
    [17, 19, 268],
    [19, 18, 1673],
    [20, 18, 1829],

    # Ceará
    [21, 24, 2.3],
    [21, 23, 86.6],
    [23, 22, 99.1],
    [24, 22, 32.2],

    # Espírito Santo
    [25, 28, 59.5],
    [25, 26, 92.1],
    [26, 27, 147],
```

```
42      [28, 27, 223],
43
44      # Goiás
45      [32, 30, 63.2],
46      [32, 31, 541],
47      [31, 29, 160],
48      [30, 29, 422],
49
50      # Maranhão
51      [35, 33, 248],
52      [35, 34, 760],
53      [34, 36, 182.2],
54      [33, 36, 605],
55
56      # Mato Grosso
57      [39, 40, 98.8],
58      [39, 37, 66.5],
59      [37, 38, 270],
60      [38, 40, 374.8],
61
62      # Mato Grosso do Sulva
63      [41, 44, 277],
64      [41, 42, 281],
65      [42, 43, 16.6],
66      [43, 44, 43.6],
67
68      # Minas Gerais
69      [45, 47, 341],
70      [45, 46, 48.6],
71      [46, 48, 138],
72      [47, 48, 431],
73
74      # Paraíba
75      [51, 52, 100],
76      [51, 49, 9.1],
77      [52, 50, 117],
78      [49, 50, 128],
79
80      # Paraná
81      [55, 56, 79.1],
82      [55, 53, 119],
```

```
 83      [56, 54, 144],
 84      [53, 54, 69.8],
 85
 86      # Pernambuco
 87      [57, 58, 9.7],
 88      [57, 60, 61.2],
 89      [58, 59, 144],
 90      [60, 59, 162],
 91
 92      # Piauí
 93      [62, 64, 18],
 94      [62, 63, 11],
 95      [64, 61, 334],
 96      [63, 61, 313],
 97
 98      # Rio de Janeiro
 99      [67, 68, 6],
100      [67, 65, 9.5],
101      [68, 66, 13.5],
102      [65, 66, 6.2],
103
104      # Rio Grande do Norte
105      [70, 69, 61, 2],
106      [70, 71, 59, 2],
107      [69, 72, 162],
108      [71, 72, 140],
109
110      # Rio Grande do Sul
111      [74, 75, 15.8],
112      [74, 73, 106],
113      [75, 76, 407],
114      [73, 76, 308],
115
116      # Rondônia
117      [79, 77, 363],
118      [79, 78, 241],
119      [77, 80, 551],
120      [78, 80, 726],
121
122      # Roraíma
123      [83, 81, 0.3],
```

```
124     [83, 84, 49.8],
125     [81, 82, 83.7],
126     [84, 82, 114],
127
128     # Santa Catarina
129     [87, 85, 11.4],
130     [87, 88, 3.6],
131     [85, 86, 110],
132     [88, 86, 123],
133
134     # São Paulo
135     [91, 89, 81.2],
136     [91, 92, 74.6],
137     [89, 90, 42.9],
138     [92, 90, 51.5],
139
140     # Sergipe
141     [94, 93, 7.9],
142     [94, 96, 7.7],
143     [93, 95, 218],
144     [96, 95, 220],
145
146     # Tocatins
147     [100, 98, 111],
148     [100, 99, 27.2],
149     [98, 97, 240],
150     [99, 97, 274],
151
152     # Pará
153     [102, 101, 219],
154     [102, 103, 305],
155     [101, 104, 1188],
156     [103, 104, 1292],
157
158     # Nordeste
159     [57, 49, 115],
160     [57, 33, 1500],
161     [57, 61, 1111],
162     [57, 17, 798],
163     [57, 69, 285],
164     [57, 21, 744],
```

```
165     [57, 93, 490],
166     [57, 5, 248],
167     [33, 61, 429],
168     [33, 21, 806],
169     [33, 69, 1289],
170     [33, 49, 1415],
171     [33, 5, 1527],
172     [33, 93, 1462],
173     [33, 17, 1565],
174     [61, 21, 570],
175     [61, 69, 964],
176     [61, 49, 1066],
177     [61, 5, 1102],
178     [61, 93, 1037],
179     [61, 17, 1138],
180     [21, 69, 509],
181     [21, 49, 636],
182     [21, 5, 878],
183     [21, 93, 982],
184     [21, 17, 1136],
185     [69, 49, 177],
186     [69, 5, 531],
187     [69, 93, 714],
188     [69, 17, 1019],
189     [5, 93, 242],
190     [5, 17, 556],
191     [5, 49, 361],
192     [49, 17, 893],
193     [49, 93, 587],
194     [17, 93, 316],
195
196     # Suldeste
197     [89, 65, 451],
198     [89, 45, 580],
199     [89, 25, 885],
200     [65, 45, 447],
201     [65, 25, 543],
202     [25, 45, 511],
203
204     # Norte
205     [13, 101, 2984],
```

```
206     [13, 1, 1395],
207     [13, 77, 888],
208     [13, 81, 738],
209     [13, 9, 2982],
210     [13, 97, 2924],
211     [101, 1, 2999],
212     [101, 77, 2492],
213     [101, 81, 222],
214     [101, 9, 5963],
215     [101, 97, 1129],
216     [1, 77, 508],
217     [1, 81, 2130],
218     [1, 9, 4374],
219     [1, 97, 2796],
220     [77, 81, 1623],
221     [77, 9, 3867],
222     [77, 97, 2290],
223     [81, 9, 2262],
224     [81, 97, 3659],
225     [9, 97, 3658],
226
227     # Centro-Oeste
228     [37, 41, 781],
229     [37, 29, 885],
230     [41, 29, 827],
231
232     # Sul
233     [73, 85, 474],
234     [73, 53, 683],
235     [53, 85, 302],
236
237     # Inter-Regionais
238     [57, 89, 2572],
239     [57, 13, 4571],
240     [57, 29, 2171],
241     [57, 73, 3651],
242     [13, 29, 3027],
243     [13, 89, 3849],
244     [13, 73, 4420],
245     [89, 29, 939],
246     [89, 73, 1079],
```

```
247     [29, 73, 1791]
248 ]
249
250
251 class Grafo:
252     def __init__(self, vertice):  # cria uma lista
    vazia
253         self.V = vertice
254         self.grafo = []
255
256     def add_aresta(self, u, v, w):  # método para
    adicionar arestas ponderadas no grafo
257         self.grafo.append([u, v, w])
258
259     def busca(self, pai, i):  # Faz uma busca
    recursiva para retornar o 'i'
260         if pai[i] == i:
261             return i
262         return self.busca(pai, pai[i])
263
264     def conectar(self, pai, rank, x, y):  # conecta
    as raizes
265         raiz_x = self.busca(pai, x)
266         raiz_y = self.busca(pai, y)
267         if rank[raiz_x] < rank[raiz_y]:
268             pai[raiz_x] = raiz_y
269         elif rank[raiz_x] > rank[raiz_y]:
270             pai[raiz_y] = raiz_x
271         else:
272             pai[raiz_y] = raiz_x
273             rank[raiz_x] += 1
274
275     # Esse algoritmo vai ordenar uma lista de
    arestas pelo seu peso e inicializa pai e rank
    matrizes
276     # Em seguida, ele itera sobre a lista ordenada
    de arestas, seleciona-as uma a uma e as adiciona à
    árvore resultante.
277     # O algoritmo para quando o número de arestas da
     árvore resultante é igual a (self.V - 1).
278     # A árvore resultante é a árvore geradora mínima
```

```
278  que estamos tentando construir e o peso da soma das
     arestas
279
280      def kruskal(self):
281          total = 0
282          arestas = 0
283          result = []
284          i = 0
285          e = 0
286
287          self.grafo = sorted(self.grafo, key=lambda
    item: item[2])
288          pai = []
289          rank = []
290          for no in range(self.V):
291              pai.append(no)
292              rank.append(0)
293          while e < self.V - 1:
294              s, d, p = self.grafo[i]
295              i = i + 1
296              x = self.busca(pai, s)
297              y = self.busca(pai, d)
298              if x != y:
299                  e = e + 1
300                  result.append([s, d, p])
301                  self.conectar(pai, rank, x, y)
302          print(f"a1    a2    peso")
303          print('================')
304          for s, d, peso in result:
305              arestas += 1
306              print(f'{s:^3} {d:^7}', end=" ")
307              print(f"\33[31m{peso:^4}\33[m")
308              total += peso
309
310          print()
311          print(f"Para construir a Árvore Geradora
    Mínima, foram usadas {arestas} arestas")
312          print(f'O peso da árvore geradora mínima é
    de \33[31m{total}')
313
314
```

```
315 # Esse laço é meramente para conciliar com o
    algoritmo
316 for a in arestas:
317     a[0] -= 1
318     a[1] -= 1
319
320 grafo = Grafo(104)  # Instância de Grafo
321
322 # Esse laço adiciona as arestas ao grafo "grafo"
323 for a in arestas:
324     grafo.add_aresta(a[0], a[1], a[2])
325
326 # Kruskal é aplicado e nos é retornado as arestas da
     Árvore Geradora Mínima e a soma dos pesos das
    arestas
327 grafo.kruskal()
328
```