```
 1 from prettytable import PrettyTable
 2 import numpy as np
 3 import heapq
 4 import time
 5
 6
 7 def create_array(n, bound):
 8     """
 9     Creates a random array
10     :param bound: int
11     :param n: int
12     :return: random array
13     """
14     array = [np.random.randint(0, bound) for x
   in range(n)]
15     return array
16
17
18 def insertion_sort(array):
19     for i, num in enumerate(array):
20         j = i
21         while j != 0 and array[j] < array[j-1]:
22             temp = array[j]
23             array[j] = array[j-1]
24             array[j-1] = temp
25             j -= 1
26
27     return array
28
29
30 def merge_sort(num_array):
31     if len(num_array) < 2:
32         return num_array[:len(num_array)]
33
34     mid = len(num_array) // 2
35     left = merge_sort(num_array[:mid])
36     right = merge_sort(num_array[mid:])
```

```
37
38      sorted_array = merge(left, right)
39
40      return sorted_array
41
42
43  def merge(left, right):
44
45      c = []
46      i = j = 0
47
48      while i < len(left) and j < len(right):
49          if left[i] < right[j]:
50              c.append(left[i])
51              i += 1
52          else:
53              c.append(right[j])
54              j += 1
55
56      c.extend(left[i:])
57      c.extend(right[j:])
58      return c
59
60
61  def heap_sort(num_array):
62      h = []
63      for value in num_array:
64          heapq.heappush(h, value)
65
66      return[heapq.heappop(h) for i in range(len(
    h))]
67
68
69  if __name__=="__main__":
70      bound = 100
71      input_sizes = [100, 1000, 10000, 100000,
    200000]
```

```
 72      insertion_time = [0.0, 0.36, 14.83, 1476.
    58, 5664.06]
 73      Table = PrettyTable(["Input Size", "
    Insertion Sort", "Merge Sort", "Heap Sort"])
 74      for i, size in enumerate(input_sizes):
 75          array = create_array(size, bound)
 76          array_copy = list(array)
 77
 78          # Calculations for heap sort
 79          start_time = time.process_time()
 80          heap_sort(array)
 81          end_time = time.process_time()
 82          time_heap_sort = end_time - start_time
 83
 84          # Calculations for insertion sort
 85          start_time = time.process_time()
 86          insertion_sort(array)
 87          end_time = time.process_time()
 88          time_insertion_sort = end_time -
    start_time
 89
 90          # Calculations for merge sort
 91          start_time = time.process_time()
 92          merge_sort(array_copy)
 93          end_time = time.process_time()
 94          time_merge_sort = end_time -
    start_time
 95
 96          Table.add_row([size, round(
    time_insertion_sort, 2), round(time_merge_sort
    , 2), round(time_heap_sort, 2)])
 97
 98      print(Table)
 99
100
101
102
```

```
103
104
```