Programming Exercise 10

Objectives: Algorithmic complexity
Instructor: Dr. Selim Yılmaz (selimyilmaz@mu.edu.tr)

Tasks

1. You are given a module named `sortingalgorithms.py` that involves two basic sorting algorithms: insertion sort and merge sort[1].

   We already know that the worst case complexities of insertion and merge sort are $O(n^2)$ and $O(nlgn)$, respectively. Well, let's empirically study on it by running them with different input conditions. First, create a file named `sorting_complexities.py` and import these sorting algorithms into your source file file to make use of them:

   ```
   from sortingalgorithms import *
   ```

2. In this exercise, you are expected to run these algorithms by passing them an integer list of length $n \mid n \in \{10, 100, 1000, 10000\}$. Note the numbers are randomly drawn from a uniform distribution from 0 to 1000. You can use `np.random.randint` method from `numpy` library:

   ```
   import numpy as np

   # an integer number from 0 to 10 is randomly generated
   val = np.random.randint(0, 10)
   ```

3. Every time you run each of the algorithms, measure the CPU time and record it. To do that, you may prefer to use the `time` module of Python. A common strategy for measuring the difference in time that results from the execution of a function call is shown below.

   ```
   import time


   tic = time.time() # take the initial clock time of CPU
   call_a_func() # invoke a function you will measure the CPU time cost
   toc = time.time() # take the final clock time of CPU
   cpu_cost = toc - tic # the time difference as second consumed by call_a_func
       .
   ```

   As you see, you can easily take the time difference in second by recording the current clock time of CPU `time.time()` twice. In order to take the difference in terms of μs, for example, you should multiply `cpu_cost` by 10^6. In this assignment, measure the running time of these algorithms in terms of μs as well.

[1] the implementations of these algorithms are borrowed from GFG.

4. After you calculate the time difference of the sorting algorithms according to the varying lengths, you should plot them using the `matplotlib` library. The line plot should look exactly as given below: