```
### CSCI B-505 Applied Algorithms - Assignment - 1
### Submitted by - Vishal

import numpy as np
import timeit
import statistics
from matplotlib import pyplot as plt
import random


class SortingAlgos:

    def __init__(self):

        self.list_lengths = np.arange(5000, 30001, 5000)

    def gen_array(self, n):
        gen_arr = np.random.randint(n, size=n)
        return gen_arr

    def selection_sort(self, A):

        start_time = timeit.default_timer()
        for i in range(len(A)):
            min_idx = i
            for j in range(i+1, len(A)):
                if A[j] < A[min_idx]:
                    min_idx = j
            A[i], A[min_idx] = A[min_idx], A[i]
        time_elapsed = timeit.default_timer() - start_time
        return A, time_elapsed

    def insertion_sort(self, A):

        start_time = timeit.default_timer()
        for i in range(1, len(A)):
            key = A[i]
            j = i - 1
            while j > 0 and A[j] > key:
                A[j+1] = A[j]
                j -= 1
            A[j + 1] = key
        time_elapsed = timeit.default_timer() - start_time
        return A, time_elapsed

    def plot_graph(self, is_array, ss_array, suffix):
        plt.plot(self.list_lengths, is_array, label='Insertion Sort',
                 marker='o')
        plt.plot(self.list_lengths, ss_array, label='Selection Sort',
                 marker='o')
        plt.xlabel('array lengths')
        plt.ylabel('sorting times')
        plt.title('Comparison of running times for sorting algorithms for ' +
                  suffix)
        plt.legend()
        plt.xticks(self.list_lengths)
        plt.show()

    def swap_random(self, arr):
        idx = range(len(arr))
        i1, i2 = random.sample(idx, 2)
        arr[i1], arr[i2] = arr[i2], arr[i1]
```

```
    def generate_runtimes(self, sort=False, rvs=False, avg=False, swap=False):
        if avg == True:
            print("Averaging ...")
        if swap == True:
            print("Swapping...")
        is_runtimes, ss_runtimes = [], []
        for n in self.list_lengths:
            gen_arr = sorting_algos.gen_array(n)
            if sort==True:
                gen_arr = sorted(gen_arr, reverse = rvs)
            if swap == True:
                for s in range(50):
                    self.swap_random(gen_arr)
            if avg == True:
                is_time_avg, ss_time_avg = [], []
                for i in range(3):
                    is_arr_sub, is_time_sub = self.insertion_sort(gen_arr)
                    ss_arr_sub, ss_time_sub = self.selection_sort(gen_arr)
                    is_time_avg.append(is_time_sub)
                    ss_time_avg.append(ss_time_sub)
                is_time = statistics.mean(is_time_avg)
                ss_time = statistics.mean(ss_time_avg)
            else:
                is_arr, is_time = self.insertion_sort(gen_arr)
                ss_arr, ss_time = self.selection_sort(gen_arr)
            is_runtimes.append(is_time)
            ss_runtimes.append(ss_time)
        print("Insertion sort running times:",
              [round(t, 5) for t in is_runtimes])
        print("Selection sort running times:",
              [round(t, 5) for t in ss_runtimes])
        return is_runtimes, ss_runtimes

    def generate_runtimes_i5(self):
        is_runtime_i5 = 0
        ss_runtime_i5 = 0
        for i in range(100000):
            gen_arr_i5 = sorting_algos.gen_array(50)
#            print(gen_arr_i5)
            is_arr_i5, is_time_i5 = self.insertion_sort(gen_arr_i5)
            ss_arr_i5, ss_time_i5 = self.selection_sort(gen_arr_i5)
            is_runtime_i5 += is_time_i5
            ss_runtime_i5 += ss_time_i5
        return is_runtime_i5, ss_runtime_i5



if __name__ == "__main__":

    sorting_algos = SortingAlgos()

    ##Generating runtimes
    print("Input 1")
    is_runtimes_p1, ss_runtimes_p1 = sorting_algos.generate_runtimes(
            sort=False, avg=True)
    print("Input 2")
    is_runtimes_p2, ss_runtimes_p2 = sorting_algos.generate_runtimes(
            sort=True, rvs=False)
    print("Input 3")
    is_runtimes_p3, ss_runtimes_p3 = sorting_algos.generate_runtimes(
            sort=True, rvs=True)
    print("Input 4")
    is_runtimes_p4, ss_runtimes_p4 = sorting_algos.generate_runtimes(
```

```
        sort=True, rvs=False, swap=True, avg=True)
print("Input 5")
is_runtimes_p5, ss_runtimes_p5 = sorting_algos.generate_runtimes_i5()

##Plotting
sorting_algos.plot_graph(is_runtimes_p1, ss_runtimes_p1, "input 1")
sorting_algos.plot_graph(is_runtimes_p2, ss_runtimes_p2, "input 2")
sorting_algos.plot_graph(is_runtimes_p3, ss_runtimes_p3, "input 3")
sorting_algos.plot_graph(is_runtimes_p4, ss_runtimes_p4, "input 4")
print("Total insertion sort running time for input 5:",
      round(is_runtimes_p5, 5))
print("Total selection sort running time for input 5:",
      round(ss_runtimes_p5, 5))
```