```
# This is a sample Python script.
import random
import time
import matplotlib.pyplot as plt

# Press Shift+F10 to execute it or replace it with your code.
# Press Double Shift to search everywhere for classes, files, tool windows,
actions, and settings.
# def print_hi(name):
#     # Use a breakpoint in the code line below to debug your script.
#     print(f'Hi, {name}')  # Press Ctrl+F8 to toggle the breakpoint.

def detect_intersection(arr, low, high):
    random_integer = random.randint(low, high)
    arr[random_integer], arr[high] = arr[high], arr[random_integer]
    min_interval = arr[high][0]
    max_interval = arr[high][1]
    for i in range(low, high - 1):
        if arr[i][0] <= max_interval and arr[i][1] >= min_interval:
            if arr[i][0] > min_interval:
                min_interval = arr[i][0]
            if arr[i][1] < max_interval:
                max_interval = arr[i][1]
    return min_interval, max_interval


def right_partition(arr, min_interval, low, high):
    index = low - 1
    for i in range(low, high):
        if arr[i][0] <= min_interval:
            index = index + 1
            arr[index], arr[i] = arr[i], arr[index]
    arr[index + 1], arr[high] = arr[high], arr[index + 1]
    return index + 1


def left_partition(arr, max_interval, low, high):
    index = low - 1
    for i in range(low, high):
        if arr[i][1] < max_interval:
            index = index + 1
            arr[index], arr[i] = arr[i], arr[index]
    arr[index + 1], arr[high] = arr[high], arr[index + 1]
    return index + 1


def sort_fuzzy(intervals_list, low, high):
    if low >= high:
        return

    (min_interval, max_interval) = detect_intersection(intervals_list, low, high)
    pivot_right = right_partition(intervals_list, min_interval, low, high)
    pivot_left = left_partition(intervals_list, max_interval, low, pivot_right)
    sort_fuzzy(intervals_list, low, pivot_left - 1)
    sort_fuzzy(intervals_list, pivot_right + 1, high)


def create_all_overlap_intervals(number_of_elements):
    with open(r"C:\Users\suman\Desktop\all_overlap.txt", "r") as all_overlap:
```

```
        all_overlaps_list = []
        for i in range(number_of_elements):
            all_overlap_value = all_overlap.readline()
            min_interval, max_interval = all_overlap_value.split(' ')
            max_interval = max_interval[:-1]
            min_interval, max_interval = float(min_interval), float(max_interval)
            all_overlaps_list.append((min_interval, max_interval))
        return all_overlaps_list


def create_small_overlap_intervals(number_of_elements):
    with open(r"C:\Users\suman\Desktop\small_overlap.txt", "r") as small_overlap:
        small_overlaps_list = []
        for i in range(number_of_elements):
            small_overlap_value = small_overlap.readline()
            min_interval, max_interval = small_overlap_value.split(' ')
            max_interval = max_interval[:-1]
            min_interval, max_interval = float(min_interval), float(max_interval)
            small_overlaps_list.append((min_interval, max_interval))
        return small_overlaps_list


def produce_graphs(number_of_elements_list, execution_durations, is_all):
    plt.plot(number_of_elements_list, execution_durations)
    if is_all:
        plt.title('Execution duration of All Overlap Intervals')
        plt.legend(['All Overlap Intervals'])
    else:
        plt.title('Execution duration of Small Overlap Intervals')
        plt.legend(['Small Overlap Intervals'])
    plt.ylabel('Execution Duration in seconds')
    plt.xlabel('Number of Elements')
    plt.show()


def all_overlap_execution_duration(number_of_elements):
    all_overlap_intervals = create_all_overlap_intervals(number_of_elements)
    start = time.time()
    sort_fuzzy(all_overlap_intervals, 0, number_of_elements - 1)
    end = time.time()
    return end - start


def small_overlap_execution_duration(number_of_elements):
    small_overlap_intervals = create_small_overlap_intervals(number_of_elements)
    start = time.time()
    sort_fuzzy(small_overlap_intervals, 0, number_of_elements - 1)
    end = time.time()
    return end - start


# Press the green button in the gutter to run the script.
if __name__ == '__main__':
    # print_hi('PyCharm')
    number_of_elements_arr = [10, 100, 500, 1000, 5000, 10000, 50000, 100000,
500000, 1000000]
    # number_of_elements_arr = [10, 100, 1000, 10000, 100000, 1000000]
    execution_durations_all_overlap = []
    execution_durations_small_overlap = []
```

```
    user_input = str(input("Please enter all for all_overlap or small for
small_overlap: "))
    if user_input == "all":
        for i in number_of_elements_arr:
            execution_duration = all_overlap_execution_duration(i)
            execution_durations_all_overlap.append(execution_duration)
        produce_graphs(number_of_elements_arr, execution_durations_all_overlap,
True)
    elif user_input == "small":
        for i in number_of_elements_arr:
            execution_duration = small_overlap_execution_duration(i)
            execution_durations_small_overlap.append(execution_duration)
        produce_graphs(number_of_elements_arr, execution_durations_small_overlap,
False)

# See PyCharm help at https://www.jetbrains.com/help/pycharm/
```