CMPUT 175 - Lab 9: Sorting

Goal: Gain an in-depth understanding of the selection sort and merge sort algorithms, and practice using recursion.

Exercise 0:

Download and complete the two practice worksheets on eClass to become more familiar with various sorting algorithms. Be prepared to talk about your work with your TA for any questions related to the selection sort and the merge sort.

Exercise 1:

In this exercise, you will implement two sorting algorithms: selection sort and merge sort. Implement both of the algorithms using recursion, and compute their time to sort different lists of data.

Task 1: Selection Sort

a. Implement a selection sort algorithm using recursion to sort a list of numbers **in descending order**. Start with the file, **`exercise_1.py`**, downloaded from eClass. DO NOT RENAME THIS FILE.

b. Complete the function **`recursive_selection_sort()`** in this file. You may want to define your own additional functions to complete the task, but they must be called inside of **`recursive_selection_sort()`**. Your function should sort a list in-place, so it will not need to return the sorted list.

c. Test your solution with the tests provided in **`test_selection_sort.py`** file. (i.e. just run it.) Make sure your sorting function passes all the tests.

Task 2: Merge Sort

a. Implement merge sort using recursion to sort a list of numbers **in descending order**. Complete the function **`recursive_merge_sort()`** in **`exercise_1.py`**. You may want to define your own additional functions to complete the task, but these functions must be called inside **`recursive_merge_sort()`**. This function does NOT sort the list in-place, so don't forget to return the sorted list from the function.

b. Test your solution with the tests provided in **`test_merge_sort.py`** file. Make sure your sorting function passes all the tests.

Once you have successfully completed Task 1 and 2, you may run the **`exercise_1.py`** file. Its **`__main__`** portion compares how long each sort function takes to sort lists of (a) randomly generated integers, (b) ascending integers, and (c) descending integers. This part is already written for you. **Which sorting algorithm takes the least amount of time to sort each list?**