

Homework 6: Quadratic sorting algorithms

Part 1: Implementing Recursive Sorting Algorithms

In this part of the homework, you will implement three sorting algorithms: bubble sort, selection sort, and insertion sort, **recursively**. Each sorting algorithm should return the sorted array and the number of swaps made during the sorting process. Your implementation should aim to be as efficient as possible, with no unnecessary comparison steps and no swaps if not needed. Additionally, you should ensure that both bubble sort and insertion sort maintain $O(n)$ time complexity in the best case.

Example on number of swaps:

```
>>> sort_list = [i for i in range(5)]
>>> sorted_arr, n_swaps = bubble_sort(sort_list)
>>> print(n_swaps)
0

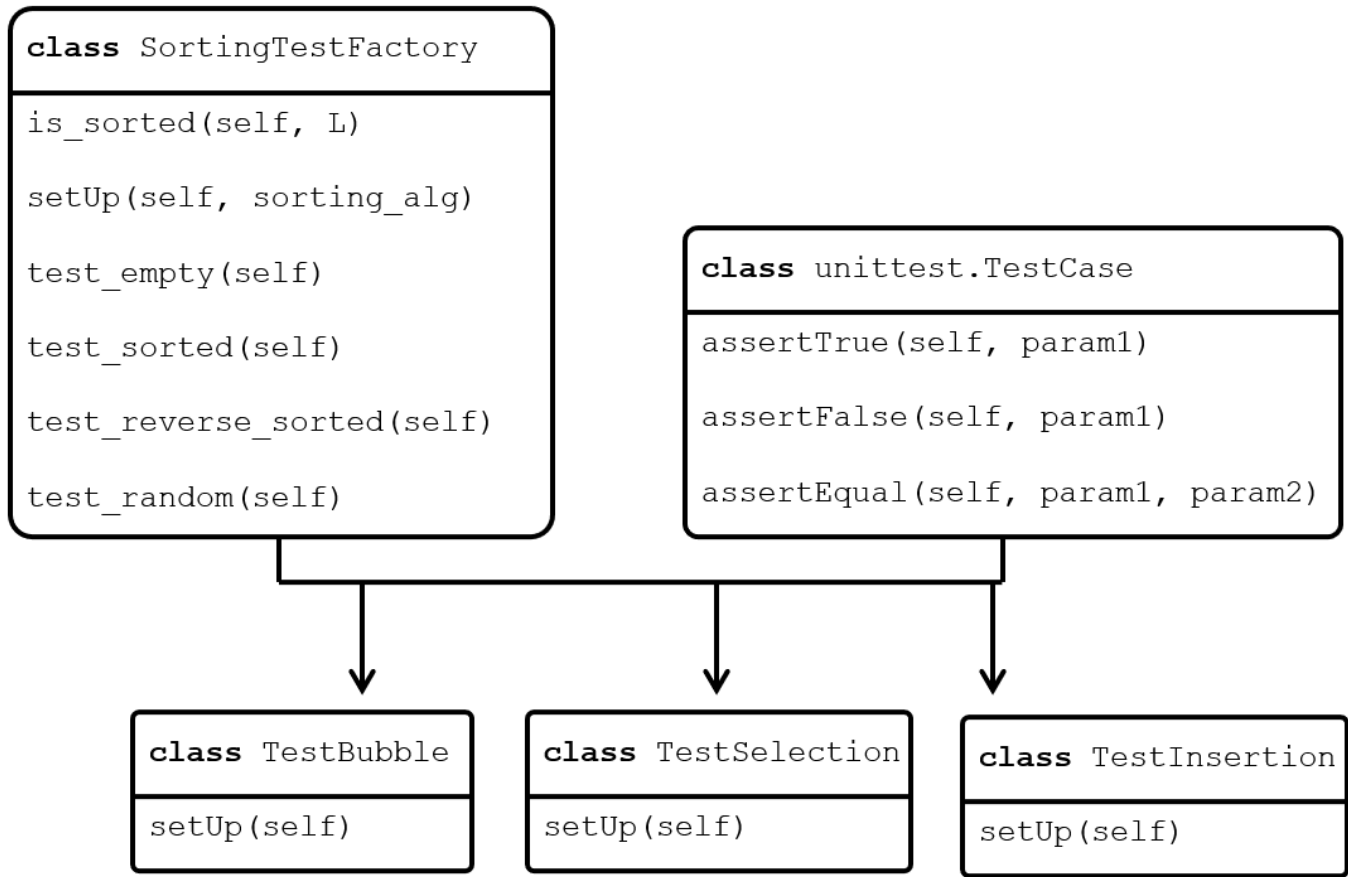
>>> rev_sorted = [i for i in range(5, 0, -1)]
>>> sorted_arr, n_swaps = bubble_sort(rev_sorted)
>>> print(n_swaps)
10

>>> random_list = [3, 2, 5, 1, 4]
>>> sorted_arr, n_swaps = bubble_sort(random_list)
>>> print(n_swaps)
5
```

Part 2: Building Test Factory for Sorting Algorithms

In this part of the homework, you will build a test factory to test the recursive sorting algorithms implemented in Part 1. The test factory should include test cases for an empty list, a sorted list, a reverse sorted list, and a randomly shuffled list. By using a test factory, you can avoid repeating the same tests for different sorting algorithms.

Below is a graphical representation of how to structure the test factory:



A test factory is a design pattern commonly used in software testing to streamline the creation of test cases and promote code reuse. It encapsulates the logic for generating test cases, making it easier to manage and maintain test suites, especially when dealing with multiple test scenarios or similar tests for different functions or classes.

In the context of sorting algorithms, a test factory can be particularly useful for generating test cases to verify the correctness and performance of different sorting implementations. Instead of manually writing test cases for each sorting algorithm, a test factory allows us to define common test scenarios (e.g., empty list, sorted list, reverse sorted list, random list) once and reuse them across multiple sorting algorithms.

It's important to note that when using a test factory, certain test cases may need to be overridden or modified for specific classes or functions. For example, in the case of sorting algorithms like bubble sort and insertion sort, the test cases may be similar. However, for algorithms like selection sort, which works differently, some test cases in the TestSelection class may need to be overridden or adapted to accommodate these differences. This ensures that the test cases accurately reflect the behavior of each sorting algorithm being tested.

Submission

Submit the following files:

- `hw6.py`
- `test_hw6.py`

Students must submit to Gradescope individually by the due date (typically Tuesday at 11:59 pm EST) to receive credit.