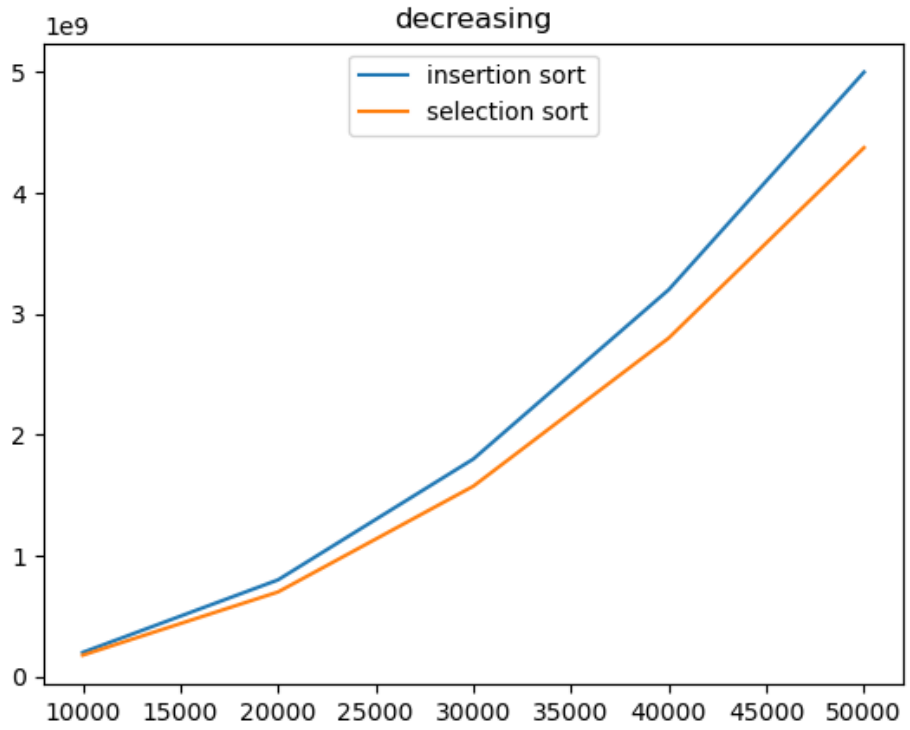


While sorting data in decreasing order, the performance of two sorting functions **become similar** because:

**For insertion sort,** when the function iterates every element and start checking its former elements, it will always not stop until reaching the head of the data because the data is in a decreasing order. Basically, every element's former element is larger, so the procedure to check and assign new values will always continue until reaching the head. This is, technically, a loop inside a loop.

**For selection sort**, the function's performance doesn't change much from that for data in increasing order. When iterating every element, it still needs to check every element after it to see if it's smaller than the current element no matter how's the data presented. Therefore, it's still considered a loop inside a loop, and thus the performance of selection sort is similar with insertion sort when it comes to data in decreasing order.

The insertion sort now has to loop inside a loop so the number of instructions will increase quadratically with the increase of size, so the input case has a more dramatic impact on its performance.