1. Performance comparison: a table lists execution time, speedup (using mode 0 as the baseline) of each mode the excludes the data initialization and result verification.

	Mode 0 (baseline)	Mode 1
Time	16.216 s	9.335 s
Speedup	1x	1.74x

2. What's the complexity of your algorithm?

For my implemented version of bucket sort algorithm, firstly I separate the elements into buckets according to the ranges of the float-point data and the number of buckets. After separating the elements, I use C++ built-in sorting function to sort elements in each bucket.

The best-case of the time complexity will be O(n) because if the number of elements is less than the number of the buckets, and each bucket has at most one element, we don't need to apply another sorting algorithm to sort elements in the bucket.

The average and the worst-case of the time complexity will be O(n*log(n)). First, my implementation distributes elements into buckets, and this part takes O(n). Next, for each bucket, we use a built-in sorting function to sort elements in the buckets. In the average case, we can assume the elements are distributed evenly, so sorting all buckets takes O(10* (n/10) * log (n/10)) = O(n * log (n/10)). In the worst-case, which all elements are in the same bucket, it still only takes O(n*log n). As a result, in the average and the worst-case, the time complexity will be O(n * log(n)).

3. Is the performance improvement as good as you expected? Why or why not?

Yes, the improvement is as good as I expected, because not only I use the bucket sort, which might be slightly faster than conventional sorting algorithms, but also because I use pthread to distribute works to multiple cores. In my implementation, I only use two threads because we only have two cores on the board, and I assign five buckets for each pthread. Although there might be an overhead of creating a new thread, sharing the resources and waiting for the other thread to be done, there is still an improvement because each thread can run at the same time and only needs to run half of the works.

4. Comparing your performance with your board mate, is your algorithm faster than theirs or not? Why or why not?

I share the board with Te I, and he implemented bitonic sort and the result was 12.82 sec. The main reason my algorithm is faster is because bucket sort can evenly distribute the elements into buckets, so the time complexity will be a little bit less than bitonic sort.