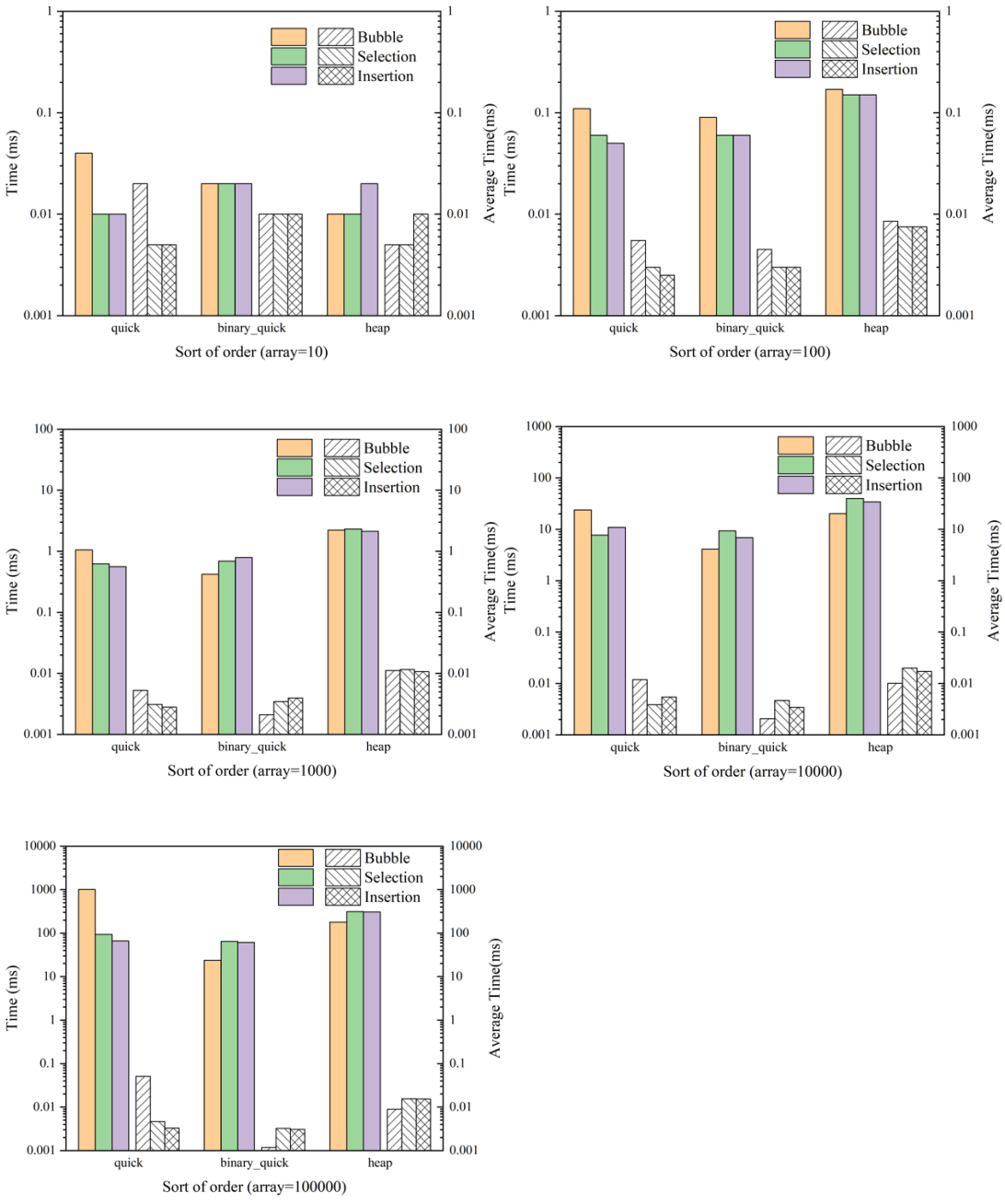
Report of Lab2

This experiment focuses on implementing three different quick sort algorithms. In this experiment, in order to better compare the sorting efficiency of the quick sort with that of the slow sort, I also used random, ascending and descending arrays ranging from 0 to 99 as sorting objects.

The running time and average running time for each type of algorithm are shown below:



The first thing worth noting is the quick sort, in my code, I choose random neutral axis, with the choice of random neutral axis, the quick sort algorithm will be slower for sorting random numbers than for ordered arrays, but the same, if you choose fixed pivot, the quick sort will be faster for sorting random than for sorting ordered arrays, this proves that the choice of pivot is very important in the quick sort, but also shows that the quick sort is an unstable sorting algorithm. The average time complexity of fast sort is O(nlogn), which is a very desirable time complexity, much lower than the average O(n^2) time complexity of slow sort.

For binary quick sort, we can notice that it has the most stable and best performance, and we can see that it is an efficient and stable algorithm that performs in similar time in average, worst and best case with time complexity O(kn), it is a stable quick sort, and it is faster in comparison with quick sort and blocking sort, but it takes extra memory because of its sorting method is out-place. Again, the speed improvement is huge compared to the slow sort.

For the heap sort algorithm, the average speed is slower than the binary quick sort, with an average time complexity of O(nlogn), but a space complexity of only O (1), meaning that no additional memory space is taken up, and overall, the performance is comparable for disordered and ascending arrays, while the quick sort reaches a time complexity of O(n*2) in the worst case, but usually behaves as O(nlogn). Heap sort is in fact an improvement of selection sort, however, to search for the maximal element one needs less comparisons, The speed is also a huge improvement compared to slow sorting.

In general, the time complexity of the fast sorting algorithm is much smaller than that of the slow sorting, and in actual operation, the fast sorting algorithm also runs much faster than the slow sorting, mainly because it avoids meaningless comparisons, and each algorithm has its own advantages and disadvantages in the fast sorting algorithm, which requires stable and fast sorting at the expense of memory space, while a certain amount of stability needs to be sacrificed in order to save memory space.