**Exercise2 report**

**Yu Sun**

In this exercise, we implement different sort methods, they are : Selection sort, Insertion sort, merge sort, quick sort and heap sort. We will discuss the performance of those method in this report

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Selection sort | Insertion sort | merge sort | quick sort | heap sort |
| Best | O(n^2) | O(n) | O(n\*logn) | O(n\*logn) | O(n) |
| Worst | O(n^2) | O(n^2) | O(n\*logn) | O(n^2) | O(n\*logn) |
| Average | O(n^2) | O(n^2) | O(n\*logn) | O(n\*logn) | O(n\*logn) |

1:Selection sort: Selection sort is the most east-understanding algorithm in sort. No matter whether the original array is sorted. Selection sort need to itinerate the whole array twice, So the time complexity is always O(n^2)

2: Insertion Sort: Insertion sort sometimes performs better than Selection sort when the sort is already sorted, since it need to itinerate the array only once if there is no need to change. So the best case the time complexity is O(n), But in worst case and average case, the time complexity is the same as selection sort.

3:merge sort: Merge sort implement algorithm based on the divide and conquer. It divides array to small part and combine them again. So the time complexity is O(n\*logn) and it’s a stable algorithm since the best and the worst situation is almost the same.

4.Quick sort: quick sort use pivot to find the right position of this pivot. It also implement the divide and conquer and the recursion(in the test data the recursion will cause the stackoverflow), So it’s fast in sorting. The best case is O(n\*logn). It’s not stable since the time complexity is depend on how mass the data set is. So the worst situation is O(n^2)

5: Heap sort: heap sort is a algorithm based on the special data structure –maxheap. Based on the max heap, always pop the biggest value in the array and sort it. It’s a unstable algorithm it takes O(n) to build the heap and sort it. So the time Complexity is O(n\*logn )

. 