Elementary Sorting Algorithms

1. Which method runs faster for an array with all keys identical, selection sort or insertion sort? Why?

2. Which method runs faster for an array in reverse order, selection sort or insertion sort? Why?

3. Suppose that we use insertion sort on a randomly ordered array where elements have only one of three values. Is the running time linear, quadratic, or something in between?

4. A colleague suggests you use insertion sort to sort a singly linked list in ascending order. Why is this a bad idea? How does insertion sort's runtime complexity change if you were to use it to sort a linked list?

5. A colleague suggests you use selection sort for h-sorting in shellsort? Why is this a bad idea?

6. A clerk at a shipping company is charged with the task of rearranging a number of large crates in order of the time they are to be shipped out. Thus, the cost of compares is very low (just look at the labels) relative to the cost of exchanges (moving the crates). The warehouse is nearly full: there is extra space sufficient to hold any one of the crates, but not two. What sorting method should the clerk use? Justify your answer.

Merge Sort

7. A colleague thinks that input arrays to the merge operation of merge sort don't need to be in sorted order. Is your colleague right or wrong? Why?

   .....

   .....

   .....

   .....

8. Give traces showing how the keys `69 65 83 89 81 85 69 83 84 73 79 78` are sorted with top-down mergesort.

9. Give traces showing how the keys `69 65 83 89 81 85 69 83 84 73 79 78` are sorted with bottom-up mergesort.

10. An array contains n numbers, and you want to determine whether two of the numbers sum to a given number k. For instance, if the input is 8, 4, 1, 6 and k is 10, the answer is yes (4 and 6). A number may be used twice.

    (a) Describe an $O(n^2)$ algorithm to solve this problem.

    .....

    .....

    .....

    .....

    (b) Describe an $O(n \log_2 n)$ algorithm to solve this problem. Hint: sort the items first. After doing so,

you can solve the problem in linear time.

. . . . . .

. . . . . .

. . . . . .

. . . . . .

11. Suppose instead of dividing in half at each step of merge sort, you divide into thirds, sort each third, and combine using a 3-way merge. We call this new sorting method 3-way merge sort. What is the time complexity of 3-way merge sort? Is it worth it to use 3-way merge sort over 2-way merge sort?

. . . . . .

. . . . . .

. . . . . .

. . . . . .

Quick Sort

12. Show the result of standard quicksort (with no optimizations or improvements) partitioning a subarray containing `69 65 83 89 81 85 69 83 84 73 79 78`.

13. Show a trace of how standard quicksort (with no optimizations or improvements) sorts an array containing `69 65 83 89 81 85 69 83 84 73 79 78`. For the purposes of this exercise, ignore the initial shuffle.

14. Explain what happens when standard quicksort (with no optimizations or improvements) is run on an

array having items with only two distinct keys.

15. On average, standard quicksort (with no optimizations or improvements) runs in $O(n \log_2 n)$ time. However, its worst case time complexity is still listed as $O(n^2)$. Under what scenarios will quick sort perform so poorly?

16. What benefits does 3-way quicksort provide over the traditional 2-way quicksort? Remember that 3-way quicksort is a flavor of quicksort where partitioning is done based on three relational buckets (less than, equal to, and greater than the subarray).