10 points for each question

1. Illustrate the operation of Quick Sort on a nearly sorted array A = {1, 7, 13, 15, 14, 16, 27, 28, 32, 35, 50, 122, 68, 17, 41, 211, 268}.

2. For array A = [4, 6, 7, 8, 9, 10, 11, 12, 1, 2, 3, 4, -2, -1, 0], what is the number of comparisons when Insertion Sort is used to sort this array? What is the number of comparisons when Quick Sort is used to sort this array? Please explain your reason.

3. Revise the PARTITION algorithm to implement a TRIPLE-PARTITION algorithm, which partitions the array A[*p*..*r*] into three subarrays A[*p*..*q1*-1], A[*q1*+1..*q2*], and A[*q2*+1..*r*] such that each element of A[*p*..*q1*-1] is less than or equal to A[*q1*+1..*q2*], and each element of A[*q1*+1..*q2*] is less than or equal to A[*q2*+1..*r*].

4. We improve Quick Sort in the following way. Upon calling Quick Sort on a subarray with k or fewer than k elements, let it simply return without sorting the subarray. After the top-level call to quicksort returns, run Insertion Sort on each and then combines the sorted subarrays with pivots to get the final sorted results. Assume the splits at every level of Quicksort are in the proportion 1 - α to α, where 0 < $\alpha \leq$ 1/2 is a constant and each subarray has exactly k elements. Analyze the running time of this sorting algorithm.

5. (textbook 7.3-2) When RANDOMIZED-QUICKSORT runs, how many calls are made to the random number generator RANDOM in the worst case? How about in the best case? Give your answer in terms of Θ-notation.

6. (textbook 8.3-3) Use induction to prove that LSD Radix Sort works. Where does your proof need the assumption that the intermediate sort is stable?

7. Given n = 40,000,000 numbers and each number of 128 bits. We first divide each number into *d* digits and then use LSD Radix Sort to sort these numbers. What's the running time if each digit is of 32 bits, 16 bits, 8 bits, $\lceil lgn \rceil$, and $\lfloor lgn \rfloor$ bits? Please explain your answer.

8. Each element of an array A of *n* elements falls in the range of [0... $k * n^{100} - 1$], where *k* is a constant that is less than *n*. Can we sort these numbers in O(*n*) time? Why?

9. What's the worst case running time of Bucket Sort? Can we make changes to Bucket Sort to further improve the worst case running time of Bucket Sort?

10. Bucket Sort uses Insertion Sort to sort numbers in a bucket. Hence, comparing numbers is an necessary operation of Bucket Sort. However, we proved that any comparison based sorting Algorithm requires $\Omega(\mathrm{nlgn})$ time complexity, and the textbook says that Bucket Sort runs in linear time. Is this a contradiction? Please explain your thoughts.