## Assignment 3

1. (10 points) For QUICKSORT, the PARTITION function is called *n* times to sort an array of *n* elements. Prove that when the array is already sorted in ascending or descending order, every call to the PARTITION (A, *p* ,*r*) function generates an empty array and an array of size p – r .

2. (10 points) Is it correct to say that the worst-case running time of RANDOMIZED-QUICKSORT on textbook 7.3 (page 179) is O(nlgn)? If YES, please give your reason. If NO, please explain under which situation the worst-case running time can be triggered.

3. (27 points) See below for Triple-QUICKSORT(A, p, r), which partitions an array into three subarrays A1, A2, and A3, such that all elements in A1 are less than those in A2, and all elements in A2 are less than those in A3.

```
Triple-QUICKSORT(A, p, r)
{
  if (p < r)
  {
    [q1, q2] = Triple-PARTITION(A, p, r);
    Triple-QUICKSORT (A, p, q1-1);
    Triple-QUICKSORT (A, q1+1, q2-1);
    Triple-QUICKSORT (A, q2+1, r);
  }
}
```

Answer the following questions:

(1) (7 points) Triple-PARTITION (A, *p*, *r*) returns two pivots q1 and q2. When this function returns, all elements in A[p…q1-1] are less than or equal to the pivot A[q1], all elements in A[q1+1…q2-1] are larger than A[q1] and less than or equal to the pivot A[q2], and all elements in A[q2+1…r] are larger than the pivot A[q2]. Complete the code of Triple-PARTITION (A, *p*, *r*).

Triple-PARTITION(A, *p*, *r*)

```
{       x = A[r]
        i = p – 1
        for j = p to r – 1
                if A[j] ⩽ x
                        i = i + 1
                        exchange A[i] ↔ A[j]
```

```
            exchange A[i + 1] ↔ A[r]
            q1 = ________

            x = A[r]
            i = q1
            for j = ________ to ________
                    if A[j] ⩽________
                        i = i + 1
                        exchange ________
            exchange ________
            q2 = ________

        return [q1, q2]

    }
```

(2) (10 points) Is it possible for Line 1 or line 9 of Triple-PARTITION choose the same element as the pivots (In other words, q1 = q2)? Please explain your reason.

(3) (10 points) Analyze the worst-case time complexity of Triple-QUICKSORT (A, *p*, *r*).

4. (10 points) Use induction to prove that LSD Radix Sort works. Where does your proof need the assumption that the intermediate sort is stable?

5. (10 points) Given *n* = 80,000,000 numbers and each number of 64 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, $\lceil \lg n \rceil$, and $\lfloor \lg n \rfloor$ bits? Please explain your answer.

6. (10 points) 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?

7. (10 points) Describe an algorithm that, given *n* integers in the rage 0 to *k*, preprocess its input and then answers any query about how many of the *n* integers fall into a range [*a*…*b*] in O(1) time. Your algorithm should use $\Theta(n + k)$ preprocessing time.

8. (10 points) Suppose we use RANDOMIZED-SELECT to select the minimum element of the array A = {3,2,9,0,7,5,4,8,6,1}. Describe a sequence of partitions that results in a worst-case performance of RANDOMIZED-SELECT.

9. (10 points) Analyze SELECT to show that if $n \geq 140$, then at least $\left\lceil \frac{n}{4} \right\rceil$ elements are greater than the median-of-medians *x,* and at least $\left\lceil \frac{n}{4} \right\rceil$ elements are less than *x*

10. (10 points) Describe an O(*n*) algorithm (other than linear sorting algorithm) that, given a set S of *n* distinct numbers and a positive integer $k \leq n$, determines the k numbers in S that are closest to the median of S.