Algorithm 2019 Spring

Homework 2

(Chapter 6~ Chapter 8)

Note: Total 3 pages, full mark would be 100 points.

1. **(10pts) Sort the given list of numbers by radix sort with LSD to ascending order {9527, 8888, 9026, 2596, 2882, 4236, 4582}.**

2. **(10pts) What situation is worst-case for quicksort? Why? Please also derive the time complexity of worst-case.**

3. **In a heap:**
   (a) **(3pts) What are the minimum numbers of elements if the height is h? Show your solution process.**
   (b) **(3pts) What are the maximum numbers of elements if the height is h? Show your solution process.**
   (c) **(4pts) Show that an n-element heap has height $\lfloor \log n \rfloor$.**

4. **(10pts) We know that it is important to how to choose a good pivot in Quick-Sort. Median-of-3 is one way to deal with this problem. Please understand the Median-of-3 by yourself and illustrate the operation of Median-of-3 on the array A(you just need to explain how you choose the pivot) = {13, 19, 9, 5, 12, 8, 7, 4, 11, 2, 6, 21, 35, 8, 13, 2, 5, 6, 37, 12, 24, 26, 3, 8, 9, 10, 54, 56, 10}.**

5. **(10pts) Please show how to sort n integers in the range 0 to $n^3 - 1$ in O(n) time, but the space complexity is in O(n).**

6. **(10pts) Is the array with values [23, 17, 14, 6, 13, 10, 1, 5, 7, 12] a max-heap? Please answer "Yes" or "No" and explain your reason.**

7. (10pts) There is an array which is already heap-sorted.
   [2, 5, 3, 11, 7, 13, 17, 19, 23, 29]
   Please do the following steps and maintain the heap (you can either draw or explain) and give the min you extract.
   (a) Insert 1 to the heap
   (b) Extract min
   (c) Change 19 to 8
   (d) Extract min

8. (10pts) Please fill in the rest of the table and assume they will sort n things.
   For counting sort, the numbers to be sorted are between 0 to k.
   For radix sort, the digits are in range (0 to k) per pass, and there is d pass.

	Time complexity			Additional space complexity	Is it stable?
	Best case	Avg. case	Worst case		
Bubble sort	$\theta(n)$	$\theta(n^2)$	$\theta(n^2)$	$\theta(1)$	Y
Insert sort	(1)	$\theta(n^2)$	(2)	(3)	(4)
Merge sort	$\theta(n\log n)$	$\theta(n\log n)$	(5)	(6)	Y
Quick sort	$\theta(n\log n)$	$\theta(n\log n)$	(7)	$\theta(\log n)$ $\sim\theta(n)$	(8)
Heap sort	(9)	$\theta(n\log n)$	(10)	(11)	(12)
Counting sort	(13)			$\theta(n+k)$	Y
Radix sort	(14)			$O(kn)$	(15)

9. (a) (5pts) Explain what is a stable sorting algorithm, that is, what does it mean for a sorting algorithm to be "stable"?

   (b) (5pts) Explain how come COUNTING-SORT is a stable sorting algorithm.

   (c) (10pts) Below picture is the pseudocode of COUNTING-SORT.

```
COUNTING-SORT(A, B, k)
1  let C[0..k] be a new array
2  for i = 0 to k
3      C[i] = 0
4  for j = 1 to A.length
5      C[A[j]] = C[A[j]] + 1
6  // C[i] now contains the number of elements equal to i.
7  for i = 1 to k
8      C[i] = C[i] + C[i - 1]
9  // C[i] now contains the number of elements less than or equal to i.
10 for j = A.length downto 1
11     B[C[A[j]]] = A[j]
12     C[A[j]] = C[A[j]] - 1
```

Suppose that we were to rewrite the for-loop header in line 10 of the COUNTING-SORT as

```
10 for j = 1 to A.length
```

Is the modified algorithm stable? Please answer "Yes" or "No" and explain your reason.