Intro to Algorithm - Homework 4

Q1:

Master theorem: If $T(n) = aT(\lceil n/b \rceil) + O(n^d)$ for some constants $a > 0$, $b > 1$, and $d \geq 0$, then

$$T(n) = \begin{cases} O(n^d) & \text{if } d > \log_b a \\ O(n^d \log n) & \text{if } d = \log_b a \\ O(n^{\log_b a}) & \text{if } d < \log_b a \end{cases}$$

a. $T(n) = 8T(n/4) + O(n)$

$d = 1$; $a = 8$; $b = 4$

$\log_4 8 = \frac{3}{2} > 1 \longrightarrow d < \log_b a$

Therefore, it is $O(n^{\log_4 8}) = O(n^{\frac{3}{2}})$

b. $T(n) = 2T(n/4) + O(\sqrt{n})$

$d = \frac{1}{2}$; $a = 2$; $b = 4$

$\log_4 2 = \frac{1}{2} = \frac{1}{2} \longrightarrow d = \log_b a$

Hence, it is $O(n^d \log n) = O(\sqrt{n} \log n)$

c. $T(n) = T(n-4) + O(n^2)$

$d = 2 \; ; \quad a = 1 \; ; \quad b = ?$

$a^k \times O\left(\frac{n}{b^k}\right)^d = O(n^d) \times \left(\frac{a}{b^d}\right)^k$) the ratio is less than 1 and result to $O(n)$

$= O(n^2) \times O(n)$

$= O(n^3)$

Hence the result is $O(n^3)$

d. $T(n) = T(\sqrt{n}) + O(n)$

$T(n) = T(n^{\frac{1}{2}}) + O(n)$

n will be larger than $n^{\frac{1}{2}}$.

we take $O(n)$

which the result is $O(n)$.

Q2:

For this question, I can do count sort algorithm.

Because in count sort algorithm, I need to make a list that store the number of count of each integer in an array A. Then, I will need to create another loop to modify the number of count in a count array and sum the count before it.

count:

2	2	0	1	1

count:

2	4	4	5	6

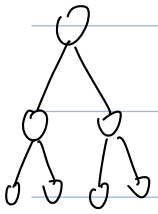
Moreover, I will place the integer as its current position and decrease the count by one base on the number of count.

Finally, I will copy the array into array A to have a sort array. Which is result in $O(n+R)$ time complexity.

Q3.

a. The number of comparision in the above algorithm in the worst case is $\log n$, But we need to split them into 2 parts. Hence, it should be $2 \log n$.

b. The algorithm more sounds like in a binary tree structure, to find the second smallest element, And in the tree, there are 2 sides. First, we can remove the root, and makesure the tree structure is properly. And then the next smallest value will be comeup as the root. This is $O(\log n)$. Because, only 1 side effect



c. $[n + \lceil \log(n) \rceil - 2]$ time complexity.

In this case, $O(\lceil \log(n) \rceil)$ can be know that only one side get effect, because the structure is binary tree. For the n, we need to pass through the array to see it is the second smallest or not. which is $O(n)$. The -2 as I discuss in part b, we need to retrieve two values, in order to start do the new algorithm in the "new" binary tree structure. [change the root element]

Hence, this is result in $O(n + \lceil \log n \rceil - 2)$.