Assignment-3

1. Use master theorem to solve (if master theorem can not be applied, write the reason):
   a. $T(n)=9T(n/3)+n$

      $a=9, b=3, k=1, p=0$

      by comparing a and b^k,
      $9>3$ which is $a>b^k$,

      Therefore, according to master theorem,
      $T(n)=\Theta(n^{\log_3 9})$
      $T(n)=\Theta(n^2)$

   b. $T(n)=9T(n/3)+1000n^2$

      $a=9, b=3, k=2, p=0$

      by comparing a and b^k,
      $9=3^2$ which is $a=b^k$ and $p>-1$,

      Therefore, according to master theorem,
      $T(n)=\Theta(n^{\log_3 9} \log n)$
      $T(n)= \Theta(n^2 \log n)$

   c. $T(n)=9T(n/3)+1000n^3$

      $a=9, b=3, k=3, p=0$

      by comparing a and b^k,
      $9<3^3$ which is $a<b^k$ and $p>=0$,

      Therefore, according to master theorem,
      $T(n)= \Theta(n^3)$

   d. $T(n)=9T(n/3)+n^2\log n$

      $a=9, b=3, k=2, p=1$

      by comparing a and b^k,
      $9=3^2$ which is $a=b^k$ and $p>-1$,

      Therefore, according to master theorem,
      $T(n)= \Theta(n^{\log_3 9} \log^2 n)$
      $T(n)= \Theta(n^2 \log^2 n)$

e. $T(n)=0.5T(n/2)+n$

$a=0.5$, $b=2$, $k=1$, $p=0$

Here, we have $a=0.5$ which doesn't satisfy the master theorems condition(as a should be $a>=1$). So, we can't apply master theorem to this given recurrence term.

f. $T(n)=2T(n/2)-n$

According to master theorem condition, we have to add some work after each step, but as we can see here in given recurrence term doesn't satisfy master theorems condition. So, we can't apply master theorem to given recurrence term.

g. $T(n)=nT(n/2)+n\log n$

$a=n$, $b=2$, $k=1$, $p=1$

here $a=n$, and a has to be constant and $a>=1$ as per master theorem conditions. So, we can't apply master theorem to given recurrence term.

h. $T(n)=T(n-2)+n^2$

As per master theorem condition we need $T(n)$ term in dividing form and here it's in subtraction form which is why we can't apply master theorem to this given recurrence term.

i. $T(n)=T(7n/10)+n$

$a=1$, $b=10/7$, $k=1$, $p=0$

by comparing a and b^k,
$1<10/7$ which is $a<b^k$ and $p>=0$,

Therefore, according to master theorem,
$T(n)= \Theta(n)$

j. $T(n)=2T(n/4)+n^{1/2}$

$a=2$, $b=4$, $k=1/2$, $p=0$

by comparing a and b^k,
$2=4^{1/2}$ which is $a=b^k$ and $p>-1$,

Therefore, according to master theorem,
$T(n)= \Theta(n^{1/2} \log n)$

2. Can master theorem be applied to
   $T(n) = 4T(n/2) + n^2 \log n$?
   Why or why not?

Yes, master theorem can be applied to the given recurrence term,

$T(n)=4T(n/2)+n^2 \log n$

a=4, b=2, k=2, p=1

by comparing a and b^k,
$4=2^2$ which is $a=b^k$ and p>-1,

Therefore, according to master theorem,
$T(n)= \Theta(n^{\log_2 4} \log^2 n)$
$T(n)= \Theta(n^2 \log^2 n)$

Here, the given term satisfy all condition of master theorem like a>=1, b>1, k>=0, p should be a real number, T(n) term in dividing form and f(n) adding work, so that we can apply master theorem to given recurrence term.

3. Use COUNTING_SORT to draw the process of sorting
   A = { 4, 8, 4, 2, 9, 3, 6, 6, 9, 0, 9}
   using C as intermediate array and Result to be the result array in the step of assigning A into Result array after constructing C, what is the difference between traversing A from left to right (from 1 to A.length) and from right to left (from A.length down to 1)?

let array A be, like,

A = | 4 | 8 | 4 | 2 | 9 | 3 | 6 | 6 | 9 | 0 | 9 |

now, by analyzing array A we can construct intermediate array C as below and C should be of 0-9 index.

	0	1	2	3	4	5	6	7	8	9	← index
C =	1	0	1	1	2	0	2	0	1	3	
	1	1	2	3	5	5	7	7	8	11	← count.

Now, using intermediate array count and given array A we need to traverse from right to the left (end to the begining) of the given array A, we can construct result array.

result = | 0 | 2 | 3 | 4 | 4 | 6 | 6 | 8 | 9 | 9 | 9 |

for instance, Checking the last element of array A ~~and~~ which is 9 and the very first index for 9 in result array is 11 as we can see in intermediate array C.

Similarly, we can do for all remaining elements.

The difference between traversing A from left to right and right to left is, if we travers from right to left then the relationship of same values in input list maintain the same in result list which is also known as stable sorting, or else while traversing from left to right the result list looks like same but the relationship of same values in input list doesn't maintain the same in result list, which we can call a unstable sorting.

4. Illustrate the process of using Radix sort to sort:

   DOG
   RUG
   ROW
   BIG
   FOX
   NOW
   BAR
   EAR
   COW

   write each result of each pass.

DOG		DOG		BAR		BAR
RUG		RUG		EAR		BIG
ROW		BIG		BIG		COW
BIG		BAR		DOG		DOG
FOX	pass-1 →	EAR	pass-2 →	ROW	pass-3 →	EAR
NOW		ROW		NOW		FOX
BAR		NOW		COW		NOW
EAR		COW		FOX		ROW
COW		FOX		RUG		RUG

First of all we need to sort this right most significant column in pass-1, then we need to sort middle column in pass-2 and lastly, we need to sort left most significant column in pass-3 to complete this radix sort.

5. Are insertion sort and merge sort stable sort? explain why.

   Yes, insertion sort and merge sort both are stable sort.

   As stable sort simply means the relative order of the duplicate elements should not change.

   In insertion sort, we just pick an element and places it in its correct place and in the code logic we are only swapping the elements if the element is larger than the key, i.e. we are not swapping the element with key when it holds equality condition.

   Merge sort is stable even in an efficient implementation. While merging two halves, we need to just make sure to use “L[i] <= R[j]” so that you favor left-half values over right-half values, means the left most significant element will be placed at left most significant place just before the same element, if they are equal.