Name: Md Habibur Rony
Student ID: 984582
Weekday: Week 2- Day 6

Answer to the Q. No. R-4.14:

A sorting algorithm is said to be stable if two objects with equal keys appear in the same order in sorted output.

Stable sorting algorithms like Insertion sort, Merge Sort, Bubble Sort, etc.

Not stable sorting algorithms are, like Heap Sort, Quick Sort, etc.

1. Bubble Sort: Stable, because two equal elements will never be swapped.
2. Insertion Sort: Stable, because the relative order of equal keys is not changed
3. Merge Sort: Stable, if we slightly modification of the algorithms.
4. Heap Sort: Not stable, because it is implemented with the next item which is the external nodes.
5. Quick Sort: Not stable, because it is implemented with randomize key, and a tree.

Answer to the Q. No. R-5.16:

No, bucket sort algorithm is not in-place. Because we need to move the items into the buckets for sorting. So as it uses other memory places for sorting that's why it is not in-place.

Answer to the Q. No. R-4.13:

Algorithm IsSameSetElement Input: A,B same set of elemements with different order Output:bool value of hasSameSet	
A<---MargeSort(A, C) A<---MargeSort(A, C)	O(nlogn) O(nlogn)
If A.size() != B.Size() then throw exception	O(1) O(1)
hasSameSet<--True i<--0 while i<A.size() then	O(1) O(1) O(n)

if A[i] !=B[i] then hasSameSet<--false i<--- A.size() continue i<--i+1 return hasSameSet	O(n) O(n) O(n) O(n) O(n) O(1)
	T(n) = O(nlogn)

Answer to the Q. No. R-5.4:

a. T(n) = 2T(n/2) + logn

Here,
a=2
b=2
f(n) = log n
$\log_2 2 = 1$

Now, By case 2 of master theorem
$f(n) = \Theta\ (n^{\log_b a} \log^{k+1} n) = \Theta\ (n^{\log_2 2} \log n) = \Theta(n \log n)$

Then
$T(n) = \Theta\ (n^{\log_b a} \log^{k+1} n)$
$T(n) = \Theta\ (n^{\log_2 2} \log^{k+1} n)$
$T(n) = \Theta\ (n \log^{k+1} n)$
$T(n) = \Theta\ (n \log n)$

b. $T(n) = 8T(n/2) + n^2$

Here,
a=8
b=2
$f(n) = n^2$
$\log_2 8 = 3$

Now, By case 1 of master theorem
$f(n) = O(n^{\log_b a - €}) = O(n^{\log_2 8 - €}) = n^3 \quad [€=0]$

Then
$T(n) = \Theta (n^{\log_b a})$
$T(n) = \Theta (n^{\log_2 8})$
$T(n) = n^3$

c. $T(n) = 16T(n/2) + (n\log n)^4$

Here,
a=16
b=2
$f(n) = (n\log n)^4$
$\log_2 16 = 4$

Now, By case 2 of master theorem

$f(n) = \Theta (n^{\log_b a} \log^k n) = \Theta (n^{\log_2 16} \log n) = \Theta(n\log n)^4$

Then
$T(n) = \Theta (n^{\log_b a} \log^{k+1} n)$
$T(n) = \Theta (n^{\log_2 16} \log n)$
$T(n) = \Theta (n\log n)^4$
$T(n) = \Theta (n\log n)^4$

d. T(n) = 7T(n/3) + n

$\log_b a = \log_3 7 \Rightarrow 1 < \log_3 7 < 2$

case 1:

is $O(n^{1.5-\varepsilon})$ upper bound of n

YES for $0<\varepsilon<=1$, Hence case 1 says T(n) is $\Theta(n^{1.5}) = \Theta(v)$

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

Here,
a=9
b=3
$f(n) = n^3\log n$
$\log_3 9 = 2$

According to case3 in above the Master Theorem

$9(n/3)3\log(n/3)$
$\Rightarrow (n^3/3)\log(n/3)$

==>(n^3/3)(logn-log3)
==>(1/3) n^3 (logn-logn3)
==>1/3 n^3 logn - 1/3 n^3 (log3)
==>n^3logn
For $\delta \geq 1/3$, case 3 says T(n) is $\square$(n^3logn)