Question 2:

Let's assume that the time taken by the algorithm to solve a problem of size n is T(n). The algorithm divides the problem into two smaller problems, each of size 3n/4​, and let's assume that the division and the combining of the results take f(n) time, which is a function of n.

Therefore Recurrence Relation is:  
  
**T(n) = 2T (3n/4​ ) + c**

Let’s find the values of “a”, “b” and “c” for the standard form of master Theorem recurrence relation:

1. Because there are 2 sub problems, therefore a = 2
2. Each sub-problem is of size 3n/4​, therefore b = 4/3
3. f(n) is the time complexity of the dividing and combining steps.

There are 3 cases which are required to apply the master theorm,

1. If **a < bd**, f(n) = O(nc) for some constant c < log b(a), then the time complexity is O(nlog b(a)).
2. If **a = bd**, f(n) = O(nlog b(a)), then the time complexity is Theta(nlog\_b(a) \* log(n)).
3. If **a > bd**, f(n) = O(nc) for some constant c > log\_b(a), then the time complexity is O(nf(n)).

Calculation of f(n) = log b(a) + d

Let’s assume here f(n) = 1:

Which makes c = 0.

Since **0 < 0.5 = log 4(2)**, we fall into the first case, and the time complexity is:  
  
O(nlog b(a)) = O(nlog 4(2)) = ***O(n0.5)***