1. Multiple Merges

a) It is simple to see that the time complexity of merging two sorted arrays of size n takes $O(n+n) = 2n$ time. This is because you can always compare the left most element and pick the smallest one. Each element will only be went through once.

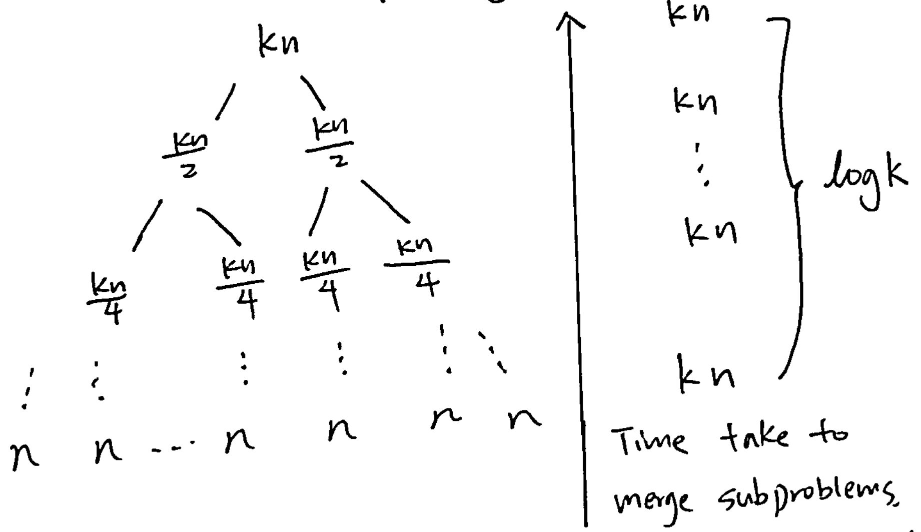
As a result: first merge: $n+n = 2n$ times
Second merge: $2n+n = 3n$ times
third merge: $3n+n = 4n$ times
k th merge: $(k-1)n+n = kn$ times.

sum them together, we have:

$$2n + 3n + 4n + \cdots + kn = n \cdot \left(\frac{(2+k)(k-1)}{2}\right) = O(k^2 n)$$

b) Using a similar approach in merge sort, we merge the first list with the second, then merge the third list with the fourth .... Finally, my approach will merge into one list.

Consider the following Tree:



Time complexity: Each row of subproblems takes (kn) time to finish. Since there are k sorted list, the height of this tree is $(\log k)$. As a resut, the overall time complexity is $O(kn \log k)$.