Divide and Conquer Twice

First we’ll look at the recurrence $T(n) = 2T\left(\frac{n}{2}\right) + O(n)$, the recurrence that is typical of mergesort in all cases, or quicksort average case. We let $T(n)$ be the time for the method/function/algorithm to execute when the input size is $n$.

\[
T(n) = 2T(n/2) + n \\
= 2[2T(n/4) + n/2] + n \\
= 4T(n/4) + n + n \\
= 4[2T(n/8) + n/4] + 2n \\
= 8T(n/8) + n + 2n \\
= 8T(n/8) + 3n
\]

We look for a pattern and generalize. We notice the last line which has a $3n$ and an $8$. Other lines use $2n$ and $4$ or $n$ and $2$. The pattern is as follows, we can substitute $k=1,2, or 3$ and see the equations above.

\[
T(n) = 2^kT\left(\frac{n}{2^k}\right) + kn
\]

Since this equation holds for every value of $k$, we can let $2^k = n$. We choose this so that we get $T(1)$ on the right hand side of the equation, since $\frac{n}{2^k} = \frac{n}{n} = 1$. But taking log of both sides gives $k = \log(n)$. Then our equation is (recall that $n = 2^k$):

\[
T(n) = nT(1) + \log(n) \cdot n
\]

Since $T(1) = 1$, i.e., the time to solve a problem with input size 1 doesn’t depend on $n$, we get that the solution to this recurrence is

\[
T(n) = n + \log(n) \cdot n = O(n \log(n))
\]

Divide and Conquer Once

\[
T(n) = T(n/2) + n \\
= [T(n/4) + n/2] + n \\
= T(n/8) + n/4 + n/2 + n \\
= T(n/16) + n/8 + n/4 + n/2 + n
\]

Again, we look for a pattern and generalize. We see powers of two in the denominator of the $T$ expression and the sum of a geometric series. The pattern is as follows, we can substitute $k=1,2,3, or 4$ and see the equations above.

\[
T(n) = T\left(\frac{n}{2^k}\right) + \sum_{i=0}^{k} \frac{n}{2^i}
\]
Again we want $T(1)$ on the right hand side so we let $n = 2^k$ to get

\[
T(n) = T(n/n) + \sum_{i=0}^{k} \frac{n}{2^i} \\
T(n) = T(1) + n \sum_{i=0}^{k} \frac{1}{2^i}
\]

But the sum is less than 2, it doesn’t matter how many terms there are – although since $n = 2^k$ we have $\log(n) = k$, but the sum is bounded above by 2, yielding:

\[
T(n) = T(1) + 2n \\
= O(n)
\]