Worst case runtime:

$$T(N) = N + 2T\left(\frac{N}{2}\right)$$

$$\therefore T\left(\frac{N}{2}\right) = \frac{N}{2} + 2T\left(\frac{N}{4}\right)$$

$$\therefore T(N) = N + 2\left(\frac{N}{2} + N T\left(\frac{N}{4}\right)\right)$$

↓ continuing this pattern,

$$T(N) = kN + 2^k T\left(\frac{N}{2^k}\right)$$

$$N = 2^k$$

$$\therefore k = \log_2 N$$

$$\therefore T(N) = \log_2 N \cdot N + N T\left(\frac{N}{N}\right)$$

$$= \log_2 N \cdot N + N$$

$O(\log_2 N \cdot N + N)$

where N represents the number of times the code runs to sort an array of length N

and $2\left(\frac{N}{2}\right)$ represents the two sub arrays that need to be merged and sorted through once again.

where k is the # of recursion through the array thus far.