The running time of the algorithm models a recurrence of the form $T(n) = 2 \cdot T(\frac{n}{2}) + O(n \cdot log(n))$. The reason being that we have two subproblems of size $\frac{n}{2}$. The combining portion is $O(n \cdot log(n))$ because the bottleneck is in doing two quicksorts, one for the left and another for the right half of the subarrays. Setting up a table and solving the system.

level	size of a problem	cost of a node	# nodes	row sum
0	n	$n \cdot log(n)$	1	$n \cdot log(n)$
1	$\frac{n}{2}$	$\frac{n}{2} \cdot log(\frac{n}{2})$	2	$n \cdot log(\frac{n}{2})$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
i	$\frac{n}{2^i}$	$\frac{n}{2^i} \cdot log(\frac{n}{2^i})$	2^i	$n \cdot log(\frac{n}{2^i})$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
k	1	1	2^k	2^k

$$\begin{aligned} \frac{n}{2^k} &= 1 \\ k &= log_2(n) \end{aligned}$$

$$\begin{aligned} \sum_{i=0}^{k-1} n \cdot log(\frac{n}{2^i}) + 2^k &= n \cdot \sum_{i=0}^{k-1} log(\frac{n}{2^i}) + 2^k \\ &= n \cdot \sum_{i=0}^{k-1} (log(n) - log(2^i)) + 2^{log_2(n)} \\ &= n \cdot \sum_{i=0}^{k-1} log(n) - n \cdot \sum_{i=0}^{k-1} log(2^i) + n \\ &= n \cdot log(n) \cdot (k - 1 - 0 + 1) - n \cdot \sum_{i=0}^{k-1} i \cdot log(2) + n \\ &= n \cdot log(n) \cdot log(n) - n \cdot \frac{(k-1) \cdot (k)}{2} + n \\ &= n \cdot log^2(n) - n \cdot \frac{(log(n) - 1) \cdot (log(n)}{2} + n \\ &= \Theta(n \cdot log^2(n)) \end{aligned}$$

Thus the algorithm runtime is $\Theta(n \cdot log^2(n))$.