For Binary Search, the time complexity is given by the following recurrence:

$T(n) = O(1) + T(n/2) \quad \Rightarrow \quad T(n) = O(\log n)$

For Merge Sort, the time complexity is given by the following recurrence:

$T(n) = 2.T(n/2) + O(n) \quad \Rightarrow \quad T(n) = O(n.\log(n))$

Solving Recurrence Relations --

1) Substitution method: Guess a solution, and then check whether it is correct.

Eg. - Let us guess the solution for Binary Search as $T(n) = O(\log n)$, which means that we must have $T(n) <= c.(\log n)$ for large enough n (for all $n >= n0$).

$$
\begin{aligned}
T(n) &= O(1) + T(n/2) \\
&<= O(1) + O(\log(n/2)) \\
&<= c1 + c2.(\log(n/2)) \\
&= c1 + c2.(\log(n) - \log(2)) \\
&= c1 + c2.\log(n) - c2 \\
&= c2.\log(n) - (c2 - c1) \\
&<= c2.\log(n) \\
&= O(\log n)
\end{aligned}
$$

2) Recurrence Tree method: Figure out the solution by studying the recurrence tree.

3) Master's Theorem method:

It is a direct method to get solutions for recurrences of the form $T(n) = a.T(n/b) + O(n^c)$, where $a>=1$ and $b>1$. Then, the following three cases are used to obtain the solution directly -

(i) If $c < \log_b(a)$, then $T(n) = \Theta(n^{(\log_b(a))})$

Eg. - For the recurrence $T(n) = 16.T(n/4) + O(n)$, we have: a=16, b=4, c=1

Therefore, $1 = c < \log_b(a) = \log4(16) = 2$, and so we have: $T(n) = \Theta(n^2)$

(ii) If $c = \log_b(a)$, then $T(n) = \Theta(n^c. \log(n))$

Eg. - For binary search recurrence $T(n) = 1.T(n/2) + O(1)$, we have: a=1, b=2, c=0

Therefore, $c = \log_b(a) = \log2(1) = 0$, and so we have: $T(n) = \Theta(n^0. \log(n))$

Eg. - For merge sort recurrence $T(n) = 2.T(n/2) + O(n)$, we have: a=2, b=2, c=1

Therefore, $c = \log_b(a) = \log2(2) = 1$, and so we have: $T(n) = \Theta(n^1. \log(n))$

(iii) If $c > \log_b(a)$, then $T(n) = \Theta(n^c)$

Eg. - For the recurrence $T(n) = 2.T(n/4) + O(n^2)$, we have: a=

Therefore, $2 = c > \log_b(a) = \log4(2) = 0.5$, and so we have: $T(n) = \Theta(n^2)$

$$
\begin{aligned}
T(n) &= 2.T(n/2) + O(n) \\
&= 2.(2.T(n/4) + O(n/2)) + O(n) \\
&= 2.(2.(2.T(n/8) + O(n/4)) + O(n/2)) + O(n) \\
&\ldots\ldots. \\
&\ldots\ldots. \\
&= 2^k.T(n/(2^k))) + (O(n) + 2.O(n/2) + 4.O(n/4) + \ldots + (2^k).O(n/(2^k))) \\
&= n.T(1) + (O(n) + O(n) + ..(k \text{ times}).. + O(n)) \\
&= n + k.O(n) \\
&= n + O(n).\log(n) = O(n. \log(n))
\end{aligned}
$$