Analyzing Merge Sort.

For simplicity, assume that n is a power of 2 so that each divide steps yields two subproblems, both of size exactly $n/2$.

The best case occurs when $n = 1$

when $n \geq 2$, time for merge sort steps:

- Divide: Just compute q as the average of p and r, which takes constant time i.e. $\Theta(1)$.
- Conquer: Recursively solve 2 subproblems, each of size $n/2$, which is $2T(n/2)$.
- Combine: MERGE on an n-element subarray takes $\Theta(n)$ time.

Therefore, the recurrence for merge sort running time is —

$$T(n) = \begin{cases} \Theta(1) & ; \text{if } n = 1 \\ 2T(n/2) + \Theta(n) & ; \text{if } n > 1. \end{cases}$$

By using master theorem, we can show that this recurrence has the solution $T(n) = \Theta(n \log_2 n)$