**Selection sort**

C = Number of operations before sorting begins

C₁ = Number of operations in the inner loop

C₀ = Number of operations in the outer loop

= Number of operations for each swap

= Probability that a swap occurs

N = Number of elements to be sorted

p = Number of elements selected

**For full selection sort:**

**Analyzing top‑p elements in the list only:**

In this case, we’re not sorting the entire list of n elements, we’re only selecting the **top p smallest elements**. So, the time complexity and operation count depend on how many elements (p) we want to extract, not the full size of the list (n). That’s why p replaces n. We use f(n, p), not just f(p) because the total number of operations can still depend on both

**Merge sort :**

Merge sort’s Big O stays the same because it always splits and merges data, so when sorting just the top p elements, the time is O(p log p) instead of O(N log N).