**Recurrence Relation**

Let the length of the first list be nnn and the length of the second list be mmm.

* At each step of the merging process, one element is moved to the merged list.
* In the worst case, we need to examine every element in both lists. Therefore, the total number of comparisons and operations is proportional to n+mn + mn+m.

Thus, the recurrence relation for this algorithm can be described as:

T(n,m)=T(n−1,m)+O(1)orT(n,m)=T(n,m−1)+O(1)T(n, m) = T(n-1, m) + O(1) \quad \text{or} \quad T(n, m) = T(n, m-1) + O(1)T(n,m)=T(n−1,m)+O(1)orT(n,m)=T(n,m−1)+O(1)

The base case is when one of the lists is empty, at which point we just append the other list to the merged list.

**Time Complexity**

The time complexity of the algorithm is O(n+m)O(n + m)O(n+m), where:

* nnn is the number of elements in the first list.
* mmm is the number of elements in the second list.

Each element from both lists is processed exactly once, and comparisons between elements occur in constant time.

**Space Complexity**

The space complexity is O(1)O(1)O(1) if we consider only the extra space used by the algorithm (not counting the input lists). However, if we account for the space used by the result list, it will be O(n+m)O(n + m)O(n+m) because we create a new merged list containing all elements of the two input lists.

In summary:

* **Time complexity**: O(n+m)O(n + m)O(n+m)
* **Space complexity**: O(1)O(1)O(1) (excluding the output list, O(n+m)O(n + m)O(n+m) if we count the result)