18. Given an array of integers nums, sort the array in ascending order and return it.You must solve the problem without using any built-in functions in O(nlog(n)) time complexity and with the smallest space complexity possible.

Program:

```
def merge_sort(nums):
    if len(nums) <= 1:
        return nums

    # Divide the array into two halves
    mid = len(nums) // 2
    left_half = nums[:mid]
    right_half = nums[mid:]

    # Recursively sort each half
    left_half = merge_sort(left_half)
    right_half = merge_sort(right_half)

    # Merge the sorted halves
    return merge(left_half, right_half)

def merge(left, right):
    merged = []
    i = j = 0

    # Merge the two halves while preserving order
    while i < len(left) and j < len(right):
        if left[i] < right[j]:
            merged.append(left[i])
```

```
            i += 1
        else:
            merged.append(right[j])
            j += 1

    # Append remaining elements from left and right
halves
    merged.extend(left[i:])
    merged.extend(right[j:])

    return merged

# Example usage
nums = [5, 2, 9, 1, 6, 4]
sorted_nums = merge_sort(nums)
print("Sorted array:", sorted_nums)
```

Output:

```
"C:\Program Files\Python312\python.exe" "C:\Work Space\DAA COADS.PYTHON\program 18.py"
Sorted array: [1, 2, 4, 5, 6, 9]

Process finished with exit code 0
```

Time complexity:

O(n log n)