Leetcode 912 — Sort an Array

# Problem Title:

# 912. Sort an Array 🔗 [Leetcode Link](https://leetcode.com/problems/sort-an-array/)

### Problem Statement:

Given an array of integers nums, return the array sorted in ascending order.

You must implement one of the classic sorting algorithms like:

* Merge Sort (🟢 stable, good for linked lists)
* Quick Sort (⚠️ not stable, but fast)
* Heap Sort (good for practice)

### Input:

nums = [5,2,3,1]

### Output:

[1,2,3,5]

### Java Code (Merge Sort):

class Solution {

public int[] sortArray(int[] nums) {

mergeSort(nums, 0, nums.length - 1);

return nums;

}

public void mergeSort(int[] nums, int left, int right) {

if (left >= right) return;

int mid = left + (right - left) / 2;

mergeSort(nums, left, mid);

mergeSort(nums, mid + 1, right);

merge(nums, left, mid, right);

}

public void merge(int[] nums, int left, int mid, int right) {

int[] temp = new int[right - left + 1];

int i = left;

int j = mid + 1;

int k = 0;

while (i <= mid && j <= right) {

if (nums[i] < nums[j]) {

temp[k++] = nums[i++];

} else {

temp[k++] = nums[j++];

}

}

while (i <= mid) temp[k++] = nums[i++];

while (j <= right) temp[k++] = nums[j++];

for (int p = 0; p < temp.length; p++) {

nums[left + p] = temp[p];

}

}

}

### Dry Run:

For nums = [5,2,3,1]

* Split: [5,2] and [3,1]
* Then: [5],[2] → merge → [2,5]
* Then: [3],[1] → merge → [1,3]
* Final merge: [2,5] and [1,3] → [1,2,3,5]

|  |  |  |  |
| --- | --- | --- | --- |
| Iteration | mid | canShip | Action |
| 1 | 55 | ✅ | right = 55 |
| 2 | 32 | ✅ | right = 32 |
| 3 | 20 | ✅ | right = 20 |
| 4 | 13 | ❌ | left = 14 |
| 5 | 17 | ✅ | right = 17 |
| 6 | 15 | ✅ | right = 15 |
| 7 | 14 | ❌ | left = 15 |
| Done | 15 | ✅ | return 15 |

### Time and Space Complexity:

|  |  |  |
| --- | --- | --- |
| Algorithm | Time | Space |
| Merge Sort | O(n log n) | O(n) |
| Quick Sort | O(n log n) | O(log n) avg, worst O(n²) |
| Heap Sort | O(n log n) | O(1) |