

912. Sort an Array

The screenshot shows a coding problem interface. On the left, the problem description for '912. Sort an Array' is displayed, including its medium difficulty, a 'Topics' tag, and a 'Companies' tag. The description states: '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 $O(n \log(n))$ time complexity and with the smallest space complexity possible.' Two examples are provided: Example 1 with input `nums = [5,2,3,1]` and output `[1,2,3,5]`, and Example 2 with input `nums = [5,1,1,2,0,0]` and output `[0,0,1,1,2,5]`. On the right, the 'Code' editor shows a C++ solution using a merge sort algorithm. The code includes a `merge` function and a `sortArray` function. Below the code editor, the 'Testcase' section shows 'Accepted' status with a runtime of 0 ms. Two test cases are listed: 'Case 1' and 'Case 2'. The input for Case 1 is `nums = [5,2,3,1]`.

Code:

```
class Solution {
public:
    void merge(vector<int>& nums, int s, int e) {
        int m = (s + e) / 2;
        vector<int> first(m - s + 1), second(e - m);
        for (int i = 0; i < first.size(); i++) {
            first[i] = nums[s + i];
        }
        for (int i = 0; i < second.size(); i++) {
            second[i] = nums[m + 1 + i];
        }
        int i1 = 0, i2 = 0, maindex = s;
        while (i1 < first.size() && i2 < second.size()) {
            if (first[i1] < second[i2]) {
                nums[maindex++] = first[i1++];
            } else {
                nums[maindex++] = second[i2++];
            }
        }
        while (i1 < first.size()) {
```

```

        nums[maindex++] = first[i1++];
    }
    while (i2 < second.size()) {
        nums[maindex++] = second[i2++];
    }
}

void mergesort(vector<int>& nums, int s, int e) {
    if (s >= e) {
        return;
    }
    int m = (s + e) / 2;
    mergesort(nums, s, m);
    mergesort(nums, m + 1, e);
    merge(nums, s, e);
}

vector<int> sortArray(vector<int>& nums) {
    mergesort(nums, 0, nums.size() - 1);
    return nums;
}

};

```