Find If Array Can Be Sorted

solved by	Senan
Platform	LeetCode
difficulty	Medium
Serial	3011
tags	Bit Manipulation Sorting
language	C++
solved on	@06/11/2024
link	https://leetcode.com/problems/find-if-array-can-be-sorted/description/
Completion	✓

Intuition

The problem is about sorting an array while only being able to swap adjacent elements under specific conditions. The condition for swapping is that two adjacent numbers must have the same number of set bits (bits that are 1 in their binary representation). The intuition here is to use a modified bubble sort approach where the adjacent swap is allowed only if the number of set bits is the same.

Approach

1. Iterate through the array using a nested loop structure, similar to bubble sort.
2. For each element at index `i`, check the element at index `i - 1`.
3. If the current element (`nums[i]`) is smaller than the previous one (`nums[i - 1]`), check the number of set bits in both elements.
4. Use `__builtin_popcount()` to count the number of set bits.
5. If the number of set bits is the same for both elements, swap them.
6. If the number of set bits differs, return `false` as it is not possible to sort the array under these conditions.
7. Continue the process until the array is checked. If no condition fails, return `true`.

Complexity

Time Complexity:

- **O(n^2)** in the worst case. The approach is similar to bubble sort where each element is compared with adjacent ones, leading to a nested loop structure.

Space Complexity:

- **O(1)**, as the algorithm sorts the array in place without using any extra space.

Code

```
class Solution {
public:
    bool canSortArray(vector<int>& nums) {
        for(int i = 1; i < nums.size(); i++) {
            for(int j = i; j >= 1; j--) {
                if(nums[j] < nums[j - 1]) {
                    int setBit1 = __builtin_popcount(nums[j]);
                    int setBit2 = __builtin_popcount(nums[j - 1]);
                    if(setBit1 == setBit2) {
                        swap(nums[j], nums[j - 1]);
                    } else {
                        return false;
                    }
                } else {
                    break;
                }
            }
        }
        return true;
    }
};
```