Check if an Array is Sorted

Given an array of integers, determine whether the array is sorted in non-decreasing order.

Input:
- An integer array arr[] of size n.

Output:
- Return true if the array is sorted in non-decreasing order; otherwise, return false.

Constraints:
- $1 \leq n \leq 10^5$
- $-10^9 \leq arr[i] \leq 10^9$

Example:
- Input: arr[] = {1, 2, 3, 4, 5}
  Output: true
- Input: arr[] = {5, 3, 4, 1, 2}
  Output: false

Pair with Given Sum in Sorted Array

Given a sorted array of integers, find if there exists a pair of elements that sum up to a given target value.

Input:

- An integer array `arr[]` of size `n` sorted in non-decreasing order.
- An integer `target` representing the desired sum.

Output:

- Return `true` if such a pair exists; otherwise, return `false`.

Constraints:

- $1 \leq n \leq 10^5$
- $-10^9 \leq arr[i], target \leq 10^9$

Example:

- Input: arr[] = {1, 2, 3, 4, 6}, target = 6 Output: true (because 2 + 4 = 6)
- Input: arr[] = {2, 5, 8, 12, 30}, target = 17 Output: true (because 5 + 12 = 17)

Sort Elements by Frequency

Given an integer array arr[] of size N, sort the array so that elements with higher frequency come first. If two numbers have the same frequency, the smaller number should appear before the larger one.

Input Format:

N
arr[0] arr[1] ... arr[N-1]

- N: the number of elements in the array (integer)
- The next line consists of N space-separated integers representing the array.

Output Format:

sorted_arr[0] sorted_arr[1] ... sorted_arr[N-1]

(Print the elements arranged in decreasing order of frequency, and for ties by increasing numeric value.)

Example:

Input:
9
4 4 5 6 4 2 2 8 5

Output:
4 4 4 2 2 5 5 6 8

Explanation:
Frequencies are:
- 4 → 3 times
- 2 → 2 times
- 5 → 2 times
- 6 → 1 time
- 8 → 1 time

Sorted by decreasing frequency (and for equal frequencies, smaller number first):
→ 4 4 4 2 2 5 5 6 8

Constraints:

$1 \leq N \leq 10^5$
Array elements fit within typical integer limits.

Sort Even-Placed Elements in Increasing and Odd-Placed in Decreasing Order

Given an array of integers, rearrange the array such that the elements at even indices (0-based indexing) are sorted in increasing order and the elements at odd indices are sorted in decreasing order. The rearrangement must be done in-place.

Input Format:

N
arr[0] arr[1] ... arr[N-1]

- N: the number of elements in the array
- arr: space-separated integers representing the array elements

Output Format:

Rearranged array with even-indexed elements sorted in increasing order and odd-indexed elements sorted in decreasing order.

Print the modified array as space-separated integers.

Example:

Input:
N = 6
arr = 1 3 2 2 5 7

Output:
1 7 2 3 5 2

Explanation:
- Even-indexed elements: 1, 2, 5 → sorted → 1, 2, 5

- Odd-indexed elements: 3, 2, 7 → sorted in decreasing → 7, 3, 2
Final arrangement → 1 7 2 3 5 2

Constraints:

$1 \le N \le 10^5$
$-10^9 \le arr[i] \le 10^9$

Sorting a Singly Linked List

Given a singly linked list of N integer nodes, sort the linked list in non-decreasing order (ascending), modifying the node links (not just swapping values). You may use Bubble Sort, Insertion Sort or Selection Sort.

Input Format:

N
val1 val2 ... valN

- N: number of nodes in the list
- Next line: N integers representing node values in the current linked list order

Output Format:

sorted_val1 → sorted_val2 → ... → sorted_valN

(Print the values in ascending order, connected by arrows or space-separated as preferred.)

Examples:

Input:
10 → 30 → 20 → 5
Output:
5 → 10 → 20 → 30

Input:
20 → 4 → 3
Output:
3 → 4 → 20

Constraints:

$1 \leq N \leq 10^5$
Node values: integers within typical limits

Meeting Rooms – Can a Person Attend All Meetings?

Given a list of meeting intervals intervals, where each intervals[i] = [start_i, end_i] represents the start and end times of the i-th meeting, determine whether a person can attend all meetings without any overlap. Meetings are considered non-overlapping if one ends exactly when another begins (i.e. end == start is allowed).

Output

Return true if no two meetings overlap, meaning the individual can attend all meetings; otherwise, return false.

Examples

Input: [[2, 4], [1, 2], [7, 8], [5, 6], [6, 8]]
Output: false
Explanation: Intervals [7,8] and [6,8] overlap.

Input: [[1, 4], [10, 15], [7, 10]]
Output: true
Explanation: All meetings are non-overlapping.

Minimum Increment Operations to Make Array Unique

Given an integer array arr[] of size N, you can choose any index i in one operation and increment arr[i] by 1. Determine the minimum number of operations required to make all elements of the array unique, meaning no duplicates remain.

Output

Make the array elements strictly unique using the minimum number of increment operations.

Examples

Input: [3, 2, 1, 2, 1, 7]
Output: 6
Explanation: After increments, result could be [3, 4, 1, 2, 5, 7].

Input: [1, 2, 2]
Output: 1
Explanation: Increment one of the 2s to 3.

Input: [5, 4, 3, 2, 1]
Output: 0
Explanation: All elements already unique.

Sort an Array of 0s, 1s and 2s

Given an array consisting of only 0s, 1s, and 2s, write an algorithm to sort it in linear time without using any sorting algorithm.

Input:
- An integer array arr[] of size n consisting only of 0s, 1s, and 2s.

Output:
- The array sorted in non-decreasing order.

Constraints:
- $1 \leq n \leq 10^6$
- $arr[i] \in \{0, 1, 2\}$

Example:
- Input: arr[] = {0, 2, 1, 2, 0}
  Output: {0, 0, 1, 2, 2}
- Input: arr[] = {2, 0, 1}
  Output: {0, 1, 2}