1099. Two Sum Less Than K

Easy | 680 | 74 | Add to List | Share

Given an array `nums` of integers and integer `k`, return the maximum `sum` such that there exists `i < j` with `nums[i] + nums[j] = sum` and `sum < k`. If no `i`, `j` exist satisfying this equation, return `-1`.

Example 1:

```
Input: nums = [34,23,1,24,75,33,54,8], k = 60
Output: 58
Explanation: We can use 34 and 24 to sum 58 which is less than 60.
```

Example 2:

```
Input: nums = [10,20,30], k = 15
Output: -1
Explanation: In this case it is not possible to get a pair sum less that 15.
```

Constraints:

- `1 <= nums.length <= 100`
- `1 <= nums[i] <= 1000`
- `1 <= k <= 2000`

Accepted 76,211 | Submissions 125,986

Seen this question in a real interview before? Yes No

Companies

0 ~ 6 months | 6 months ~ 1 year | 1 year ~ 2 years

Amazon | 5 | Capital One | 3

Related Topics

Array | Two Pointers | Binary Search | Sorting

Similar Questions

Two Sum	Easy
Two Sum II - Input Array Is Sorted	Easy
3Sum Smaller	Medium
Subarray Product Less Than K	Medium

Hide Hint 1

What if we have the array sorted?

Hide Hint 2

Loop the array and get the value A[i] then we need to find a value A[j] such that A[i] + A[j] < K which means A[j] < K - A[i]. In order to do that we can find that value with a binary search.

Java | Autocomplete

```java
class Solution {
    public int twoSumLessThanK(int[] nums, int k) {
        Arrays.sort(nums);
        int answer = -1;
        int left = 0;
        int right = nums.length - 1;
        while (left < right) {
            int sum = nums[left] + nums[right];
            if (sum < k) {
                answer = Math.max(answer, sum);
                left++;
            } else {
                right--;
            }
        }
        return answer;
    }
}

class Solution {
    public int twoSumLessThanK(int[] nums, int k) {
        int answer = -1;
        Arrays.sort(nums);
        for (int i = 0; i < nums.length; ++i) {
            int idx = Arrays.binarySearch(nums, i + 1, nums.length, k - nums[i] - 1);
            int j = (idx >= 0 ? idx : ~idx);
            if (j == nums.length || nums[j] > k - nums[i] - 1) {
                j--;
            }
            if (j > i) {
                answer = Math.max(answer, nums[i] + nums[j]);
            }
        }
        return answer;
    }
}
```