881. Boats to Save People

Medium | Topics | Companies

You are given an array `people` where `people[i]` is the weight of the i^{th} person, and an **infinite number of boats** where each boat can carry a maximum weight of `limit`. Each boat carries at most two people at the same time, provided the sum of the weight of those people is at most `limit`.

Return *the minimum number of boats to carry every given person*.

3 2 2 1

2 1 3 1

Example 1:

```
Input: people = [1,2], limit = 3
Output: 1
Explanation: 1 boat (1, 2)
```

Example 2:

```
Input: people = [3,2,2,1], limit = 3
Output: 3
Explanation: 3 boats (1, 2), (2) and (3)
```

Example 3:

```
Input: people = [3,5,3,4], limit = 5
Output: 4
Explanation: 4 boats (3), (3), (4), (5)
```

Constraints:

- `1 <= people.length <= 5 * 10^4`
- `1 <= people[i] <= limit <= 3 * 10^4`

Approach 1: Sort + Two pointers

$[1 \quad 2 \quad 2 \quad 3]$ limit = 3

l r

$a_l + a_r = 1+3$

$= 4$ i.e. $>$ limit

if you can't take ③ with the minimum ① then its obvious that you can't take ③ combined with any other in array.

So ③ must go alone in a boat

So ③ must go alone in a boat
Hence increase boat count and
decrement r pointer

[1 2 2 3]
  l   r

boats = 1

$a_l + a_r = 1 + 2$
$= 3$ i.e. == limit

So ① & ② can go in a single boat.
Hence increase boat count and move both l and r

[1 2 2 3]
    l
    r

boats = 2

now as $l < r$ is false we stop

The while loop gets terminated with $l == r$ only when there is ① person left to get into the boat.
let

[1 1 2 3]
  l   r

Limit = 4

[1 1 2 3]
    l r

boats = 1

[1 1 2 3]
    r l

boats = 2

As $l < r$ is false we stop.
Is there anyone left to get into a boat.

NO, we can be sure about this because $l > r$.

```cpp
class Solution {
public:
    int numRescueBoats(vector<int>& people, int limit) {
        sort(people.begin(),people.end());

        int boats=0,l=0,r=people.size()-1;

        while(l<r)
        {
            if(people[l]+people[r] <= limit) boats++,l++,r--;
            else boats++,r--;
        }

        return l==r?boats+1:boats;
    }
};
```

here we are allowing only 2 people in a boat at max. i.e. if $P_l + P_r < limit$ we are increasing boats and moving both l & r. we are not checking if boat can hold even more people with P_l and P_r being in the boat

$T(n): O(n\log n) + O(n)$
$S(n): O(1)$

Approach 2: Bucket Sort

?

$T(n): O(n)$
$S(n): O(limit)$