881. Boats to Save People

24 March 2022 06:36 PM

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.*

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)
```

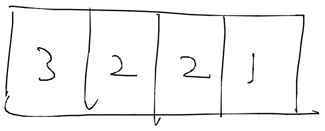
Example 3:

```
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}

limit = 3



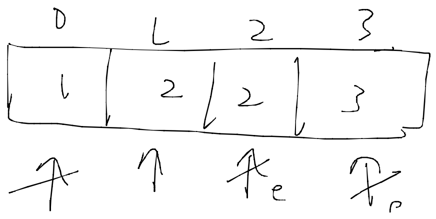
* Sort the array the reason is we know that our limit is 3 and there is at most two people can go in one boat, so two people sum should be equal to 3 (limit).

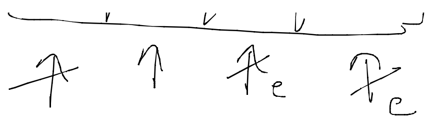
* we are sorting because the end index will have max weight and 1st index with least weight.

* So we can compare two people one with least weight and highest weight, and make them go into same boat with the sum less than or equal to limit.

now taking two pointer

limit = 3



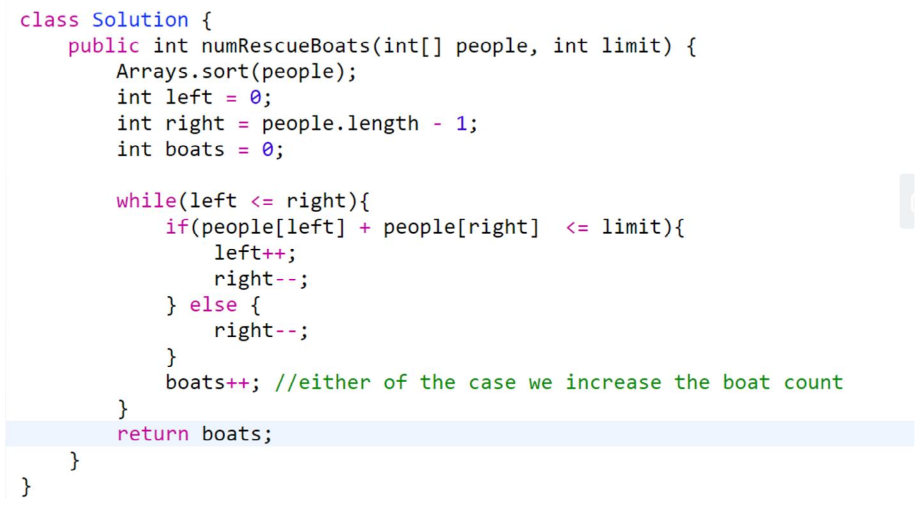


⇒ now we check if the sum is equal to or less than limit

in this case it's not so we decrement the end pointer

now the sum is equal to limit so we increment

and decrement.



```java
class Solution {
    public int numRescueBoats(int[] people, int limit) {
        Arrays.sort(people);
        int left = 0;
        int right = people.length - 1;
        int boats = 0;

        while(left <= right){
            if(people[left] + people[right]  <= limit){
                left++;
                right--;
            } else {
                right--;
            }
            boats++; //either of the case we increase the boat count
        }
        return boats;
    }
}
```

T.C ⇒ $O(n\log n)$ S.C ⇒ $O(1)$

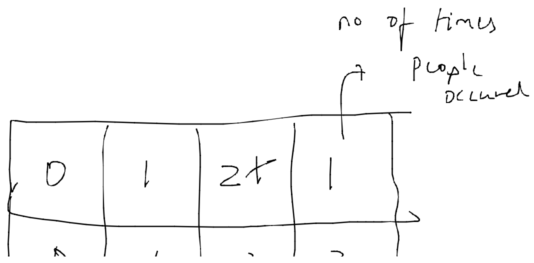
How to do in $O(n)$?

Constraint $1 <= \text{people}[i] \leq= \text{limit} \leq 3000$

we can do count sort

3 2 2 1

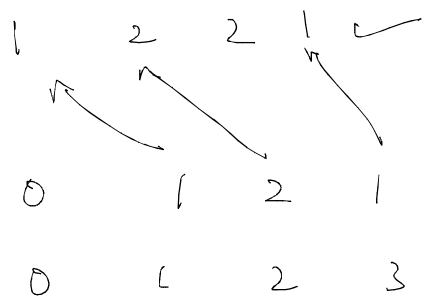
Count sort array



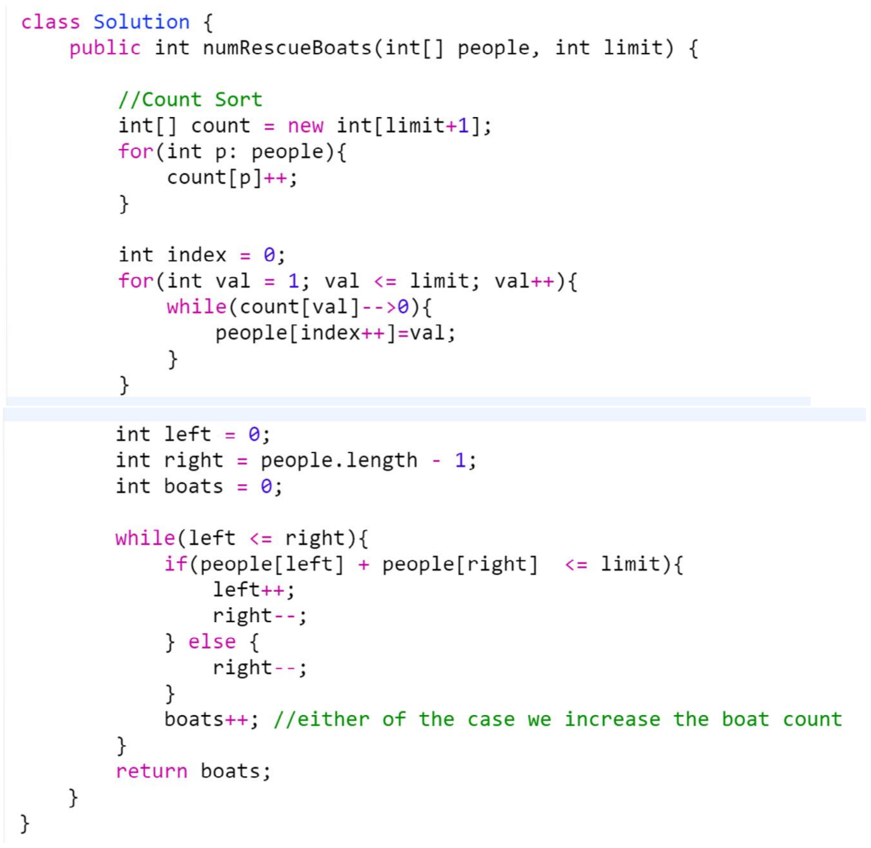
count sort array

0	1	2	1
0	1	2	3

and then now we copy from start of the array
start



without using sorting
but you get space.



```java
class Solution {
    public int numRescueBoats(int[] people, int limit) {

        //Count Sort
        int[] count = new int[limit+1];
        for(int p: people){
            count[p]++;
        }

        int index = 0;
        for(int val = 1; val <= limit; val++){
            while(count[val]-->0){
                people[index++]=val;
            }
        }

        int left = 0;
        int right = people.length - 1;
        int boats = 0;

        while(left <= right){
            if(people[left] + people[right]  <= limit){
                left++;
                right--;
            } else {
                right--;
            }
            boats++; //either of the case we increase the boat count
        }
        return boats;
    }
}
```