

# 3074. Apple Redistribution into Boxes

Solved

Easy Topics Companies Hint

You are given an array `apple` of size  $n$  and an array `capacity` of size  $m$ .

There are  $n$  packs where the  $i^{\text{th}}$  pack contains `apple[i]` apples. There are  $m$  boxes as well, and the  $i^{\text{th}}$  box has a capacity of `capacity[i]` apples.

Return *the minimum number of boxes you need to select to redistribute these  $n$  packs of apples into boxes*.

**Note** that, apples from the same pack can be distributed into different boxes.

## Example 1:

**Input:** `apple = [1,3,2], capacity = [4,3,1,5,2]`

**Output:** 2

**Explanation:** We will use boxes with capacities 4 and 5.

It is possible to distribute the apples as the total capacity is greater than or equal to the total number of apples.

## Example 2:

**Input:** `apple = [5,5,5], capacity = [2,4,2,7]`

**Output:** 4

**Explanation:** We will need to use all the boxes.

## Constraints:

- $1 \leq n == \text{apple.length} \leq 50$
- $1 \leq m == \text{capacity.length} \leq 50$
- $1 \leq \text{apple}[i], \text{capacity}[i] \leq 50$
- The input is generated such that it's possible to redistribute packs of apples into boxes.

## Python:

```
class Solution:  
    def minimumBoxes(self, apple: List[int], cap: List[int]) -> int:  
        tot = sum(apple)
```

```
cap.sort(reverse=True)
```

```
res = 0
while tot > 0:
    tot -= cap[res]
    res += 1
```

```
return res
```

## JavaScript:

```
const minimumBoxes = (apple, cap) => {
    let sum = apple.reduce((a, c) => a + c, 0);
    cap.sort((a, b) => b - a);

    let res = 0;
    while (sum > 0)
        sum -= cap[res++];

    return res;
};
```

## Java:

```
class Solution {
    public int minimumBoxes(int[] apple, int[] cap) {
        int sum = Arrays.stream(apple).sum();

        int[] fq = new int[51];
        int high = 0, low = 51;
        for (int c : cap) {
            fq[c]++;
            high = Math.max(high, c);
            low = Math.min(low, c);
        }

        int res = 0;
        for (int i = high; i >= low && sum > 0; i--) {
            while (fq[i]-- > 0 && sum > 0) {
                sum -= i;
                res++;
            }
        }

        return res;
    }
}
```