Description | Solution | Discuss (161) | Submissions

Java | Autocomplete

1231. Divide Chocolate

Hard 605 42 Add to List Share

You have one chocolate bar that consists of some chunks. Each chunk has its own sweetness given by the array `sweetness`.

You want to share the chocolate with your `k` friends so you start cutting the chocolate bar into `k + 1` pieces using `k` cuts, each piece consists of some **consecutive** chunks.

Being generous, you will eat the piece with the **minimum total sweetness** and give the other pieces to your friends.

Find the **maximum total sweetness** of the piece you can get by cutting the chocolate bar optimally.

Example 1:

```
Input: sweetness = [1,2,3,4,5,6,7,8,9], k = 5
Output: 6
Explanation: You can divide the chocolate to [1,2,3], [4,5], [6], [7], [8], [9]
```

Example 2:

```
Input: sweetness = [5,6,7,8,9,1,2,3,4], k = 8
Output: 1
Explanation: There is only one way to cut the bar into 9 pieces.
```

Example 3:

```
Input: sweetness = [1,2,2,1,2,2,1,2,2], k = 2
Output: 5
Explanation: You can divide the chocolate to [1,2,2], [1,2,2], [1,2,2]
```

Constraints:

- `0 <= k < sweetness.length <= 10^4`
- `1 <= sweetness[i] <= 10^5`

Accepted 30,433 | Submissions 54,872

Seen this question in a real interview before? Yes No

Companies

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

Google 2 Amazon 2 Facebook 2

Related Topics

Array Binary Search

Similar Questions

Split Array Largest Sum Hard

Capacity To Ship Packages Within D Days Medium

Hide Hint 1

After dividing the array into K+1 sub-arrays, you will pick the sub-array with the minimum sum.

Hide Hint 2

Divide the sub-array into K+1 sub-arrays such that the minimum sub-array sum is as maximum as possible.

Hide Hint 3

Use binary search with greedy check.

```java
class Solution {
    public int maximizeSweetness(int[] sweetness, int k) {
        // Initialize the left and right boundaries.
        // left = 1 and right = total sweetness / number of people.
        int numberOfPeople = k + 1;
        int left = Arrays.stream(sweetness).min().getAsInt();
        int right = Arrays.stream(sweetness).sum() / numberOfPeople;

        while (left < right) {
            // Get the middle index between left and right boundary indexes.
            // cur_sweetness stands for the total sweetness for the current person.
            // people_with_chocolate stands for the number of people that have
            // a piece of chocolate of sweetness greater than or equal to mid.
            int mid = (left + right + 1) / 2;
            int curSweetness = 0;
            int peopleWithChocolate = 0;

            // Start assigning chunks to the current people,.
            for (int s : sweetness) {
                curSweetness += s;

                // If the total sweetness for him is no less than mid, meaning we
                // have done with him and should move on to assigning chunks to the next people.
                if (curSweetness >= mid) {
                    peopleWithChocolate += 1;
                    curSweetness = 0;
                }
            }

            // Check if we successfully give everyone a piece of chocolate with sweetness
            // no less than mid, and eliminate the search space by half.
            if (peopleWithChocolate >= numberOfPeople) {
                left = mid;
            } else {
                right = mid - 1;
            }
        }

        // Once the left and right boundaries coincide, we find the target value,
        // that is, the maximum possible sweetness we can get.
        return right;
    }
}
```

Problems Pick One Prev 1/99 Next Console Contribute i

Run Code Submit