**2305. Fair Distribution of Cookies :-**

Medium Accepted: 50.9K Submissions: 74.7K Acceptance Rate: 68.1%

You are given an integer array cookies, where cookies[i] denotes the number of cookies in the ith bag. You are also given an integer k that denotes the number of children to distribute **all** the bags of cookies to. All the cookies in the same bag must go to the same child and cannot be split up.

The **unfairness** of a distribution is defined as the **maximum** **total** cookies obtained by a single child in the distribution.

Return *the****minimum****unfairness of all distributions*.

**Example 1:**

**Input:** cookies = [8,15,10,20,8], k = 2

**Output:** 31

**Explanation:** One optimal distribution is [8,15,8] and [10,20]

- The 1st child receives [8,15,8] which has a total of 8 + 15 + 8 = 31 cookies.

- The 2nd child receives [10,20] which has a total of 10 + 20 = 30 cookies.

The unfairness of the distribution is max(31,30) = 31.

It can be shown that there is no distribution with an unfairness less than 31.

**Example 2:**

**Input:** cookies = [6,1,3,2,2,4,1,2], k = 3

**Output:** 7

**Explanation:** One optimal distribution is [6,1], [3,2,2], and [4,1,2]

- The 1st child receives [6,1] which has a total of 6 + 1 = 7 cookies.

- The 2nd child receives [3,2,2] which has a total of 3 + 2 + 2 = 7 cookies.

- The 3rd child receives [4,1,2] which has a total of 4 + 1 + 2 = 7 cookies.

The unfairness of the distribution is max(7,7,7) = 7.

It can be shown that there is no distribution with an unfairness less than 7.

**Constraints:**

* 2 <= cookies.length <= 8
* 1 <= cookies[i] <= 105
* 2 <= k <= cookies.length

**Code :-**

class Solution {

public:

    void func(vector<int> &cookie, int ind, int k, vector<int> &child, int &mini, int &maxi){

        //base case

        if(ind >= cookie.size()){

            mini = min(mini, maxi);

            return;

        }

        //recursion

        for(auto i=0; i<k; i++){

            child[i] += cookie[ind];

            int newmaxi = maxi;

            maxi = max(maxi, child[i]);

            func(cookie, ind+1, k, child, mini, maxi);

            child[i] -= cookie[ind];

            maxi = newmaxi;

        }

        return;

    }

    int distributeCookies(vector<int>& cookie, int k) {

        vector<int> child(k,0);

        int mini=INT\_MAX, maxi=INT\_MIN;

        func(cookie, 0, k, child, mini, maxi);

        return mini;

    }

};

**T.C :- O(2(n\*k))**

**S.C :- O(k) without recursion function call stack size**