yproblems) o o permutatic

5219. Maximum Candies Allocated to K Children

My Submissions (/contest/weekly-contest-287/problems/maximum-candies-allocated-to-k-children/submissions/)

Back to Contest (/contest/weekly-contest-287/)

You are given a **0-indexed** integer array candies. Each element in the array denotes a pile of candies of size candies [i]. You can divide each pile into any number of **sub piles**, but you **cannot** merge two piles together.

You are also given an integer k. You should allocate piles of candies to k children such that each child gets the **same** number of candies. Each child can take **at most one** pile of candies and some piles of candies may go unused.

Return the maximum number of candies each child can get.

User Accepted:	0
User Tried:	0
Total Accepted:	0
Total Submissions:	0
Difficulty:	Medium

Example 1:

```
Input: candies = [5,8,6], k = 3
Output: 5
Explanation: We can divide candies[1] into 2 piles of size 5 and 3, and candies[2] into 2 piles of size 5 and 1. We now have file
```

Example 2:

```
Input: candies = [2,5], k = 11
Output: 0
Explanation: There are 11 children but only 7 candies in total, so it is impossible to ensure each child receives at least one
```

Constraints:

- 1 <= candies.length <= 10⁵
 1 <= candies[i] <= 10⁷
- 1 $<= k <= 10^{12}$

```
JavaScript
                                                                                                                                 \display
                                                                                                                                        \mathfrak{C}
    const sm = (a) \Rightarrow a.reduce(((x, y) \Rightarrow x + y), 0);
 1
 2
 3
    let a, k;
 4 1
    const maximumCandies = (A, K) \Rightarrow \{
 5
         a = A, k = K;
 6
         let sum = sm(a);
         if (sum < k) {
 7,
 8
              return 0;
 9 •
         } else if (sum == k) {
10
              return 1;
11 ▼
         } else {
12
              let max = Math.max(...a);
13
              return Math.max(1, BinarySearch(0, max));
14
15
    };
16
17 ▼
    const BinarySearch = (low, high) => {
18 ▼
         while (low <= high) {</pre>
19
              let mid = low + parseInt((high - low) / 2);
20 •
              if (possible(mid)) {
21
                   low = mid + 1;
22
              } else {
23
                  high = mid - 1;
24
25
26
         // pr(low, high)
27
         return high;
28
    };
29
30 v const possible = (each) ⇒ {
```

```
4/2/22, 11:19 PM
                                                             Maximum Candies Allocated to K Children - LeetCode Contest
    31
             let canHave = 0;
             for (const x of a) canHave += parseInt(x / each);
    32
             // pr("canHave", canHave, k)
    33
    34
             return canHave >= k;
    35
        };
  ☐ Custom Testcase
                          Use Example Testcases
                                                                                                                                              △ Submit
                                                                                                                                    Run
  Submission Result: Accepted (/submissions/detail/672666300/) 2
                                                                                   More Details > (/submissions/detail/672666300/)
  Share your acceptance!
                ₫1
   Copyright © 2022 LeetCode
   Help Center (/support) | Jobs (/jobs) | Bug Bounty (/bugbounty) | Online Interview (/interview/) | Students (/student) | Terms (/terms) | Privacy Policy (/privacy)
   United States (/region)
```