Monday, March 17, 2025 5:55 PM

Maximum Candies Allocated to K Children

2226. Maximum Candies Allocated to K Children

Solved

Medium Topics Companies Hint

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 be allocated candies from **only one** pile of candies and some piles of candies may go unused.

Return the ***maximum number of candies*** *each child can get.*

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 five piles of candies of sizes 5, 5, 3, 5, and 1. We can allocate the 3 piles of size 5 to 3 children. It can be proven that each child cannot receive more than 5 candies.
```

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 candy. Thus, each child gets no candy and the answer is 0.
```

Constraints:

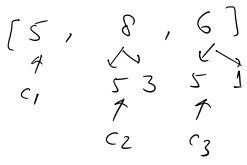
- `1 <= candies.length <= 10^5`
- `1 <= candies[i] <= 10^7`
- `1 <= k <= 10^12`

Seen this question in a real interview before? 1/5

Yes No

candies

eg candies = [5,8,6], k = 3

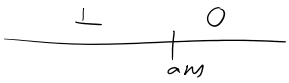


max. candies = 5

Observations:-

1. K must be equal to n.
2. lo = 1
   hi = min(candies).
3. Test function checks if mid is less than/ equal to all elements

Here we need maximum value, so binary search conditions are changed.




```
while(lo<hi){
    int mid = lo+hi+1/2;
    if(test(candies,mid)) ans = mid; lo = mid+1;
     else hi = mid;
    }
return lo;
}
```

[5, 8, 6] (3)

lo = 1, hi = 6

① $\frac{1+6}{2} = 3$ ans = 3 lo = 4

② $\frac{4+6}{2} = 5$ ans = 5 lo = 6

```
class Solution{
  public:
  int maximumCandies(vector<int>& candies,
  long long k){
```

```
long long k){
int n=candies.size();
int lo=1, hi=INT_MIN, ans=0;
for(int i: candies) hi=max(hi,i);
hi++; //exclusive hi
while(lo<hi){
    int mid=lo+(hi-lo)/2;
if(test(candies,mid,k)){
        ans=mid; lo=mid+1;}
    else hi=mid;}
    return ans;}
int test(vector<int>& candies, int mid, long long k)
{ long long ans=0;
  for(int i: candies){
        ans+=(i/mid);}

    return ans>=k;

  }
};
```

T.C. O(n log(max(candies)))

S.C. O(1).