Approach

So we're given some bags that contain balls. What we have to do is divide each bag into as many parts as possible but the twist is to divide the bags so that EVERY BAG has the minimum size possible.

REMEMBER: Your penalty is the maximum number of balls in a bag. You want to minimize your penalty after the operations.

So if maximum balls in a bag is penalty, that means we have to store the minimum amount of balls in every bag. So we're going to minimize bag size.

So since this is binary search, our low limit is the number of balls we can keep in each bag. That could be 0 of course, we don't keep any balls in a bag but instead we'll take our low limit as 1. I'll tell you why later: **READ COMEBACK POINT**.

The maximum amount of balls we can keep in a bag, is the maximum out of all the bags we're given. So you either can find the maximum bag out of all bags and use that as a high limit or just 10^9 or 1e9 (which is given in the constraints as the maximum balls in a bag).

Apply regular binary search here. While low is less/equal to high, we calculate mid as usual.
mid is our bag limit, the maximum number of balls we can store in every bag.
Now here's the fun part. We need to do extra stuff to binary search to decide the best answer.

So, we're gonna go through every bag we have and count the number of times we can split the bag so that each split bag contains a maximum of mid balls. It's kinda tricky but also simple to calculate.
So say we have a bag of 8 balls and our mid is 2. What this means is that we have to split the bag so that each bag split from 8 contains maximum of 2 balls. So that can be 4 bags that contain 2 balls yeah? Good so we get 4 bags in total but weight, how many times did we split our bag? That'll be 3 times, not 4 because 8 split once is 6 and 2. 6 split once is 4 and 2. 4 split once is 2 and 2. So in total, we get 4 bags of 2 balls by splitting 3 times.
So number of splits is less than the number of total bags yeah? Easy to calculate. Just take the bag size subtract 1 from it and divide it by mid. So, (8-1)/2 which gives us 3. If we had done only 8/2, we would have gotten 4 bags but we need to find the number of splits, not number of bags after splitting.

Now you'll ask, why did we subtract 1 from 8 and not 8/2. The answer is simple, subtracting 1 from the bag directly handles the case for even and odd numbers. If the number is even, we actually can do 8/2 and subtract 1 later, that's totally fine but if the bag size was 7 (an odd number), we would do 7/2 only and that would give us 3 number of splits. So, while looping through each bag, you can either check for even/odd condition: when even, subtract 1 later, when odd, just divide or you can just subtract 1 from the bag size and divide by mid to handle both cases at the same time.

COME BACK: We didn't start with 0, because if our mid ever got to 0, then we would be dividing bag_size by mid but we can't do that. The compiler would throw error on dividing anything by 0 so we start our lower limit with 1 instead ;)

Good, so we've got the number of splits we've made so that all bags are divided into maximum bag size of mid.

BUT HOLD ON, we were given a limit of how many split operations we can do.
This will be our deciding factor in the binary search.

If our split count is less than the limit of split operations allowed, then we possibly have our answer. So store the mid as our ans but it's not enough, maybe we can do better? So, set high limit ot mid-1, to make sure the bag size is even smaller because remember, our penalty is maximizing the size of the bag so we have to keep it as minimum as possible.

If our split count is greater than the limit, then we know our bag size is wrong and we can probably add more balls in our bags so that the total split operations are less. So our low limit is mid+1.