Problem 1962: Remove Stones to Minimize the Total

Problem Information

Difficulty: Medium
Acceptance Rate: 65.19%
Paid Only: No
Tags: Array, Greedy, Heap (Priority Queue)

Problem Description

You are given a **0-indexed** integer array `piles`, where `piles[i]` represents the number of stones in the `ith` pile, and an integer `k`. You should apply the following operation **exactly** `k` times:

* Choose any `piles[i]` and **remove** `floor(piles[i] / 2)` stones from it.

Notice that you can apply the operation on the **same** pile more than once.

Return _the**minimum** possible total number of stones remaining after applying the _`k` _operations_.

`floor(x)` is the **largest** integer that is **smaller** than or **equal** to `x` (i.e., rounds `x` down).

Example 1:

Input: piles = [5,4,9], k = 2 **Output:** 12 **Explanation:** Steps of a possible scenario are: - Apply the operation on pile 2. The resulting piles are [5,4,_5_]. - Apply the operation on pile 0. The resulting piles are [_3_ ,4,5]. The total number of stones in [3,4,5] is 12.

Example 2:

Input: piles = [4,3,6,7], k = 3 **Output:** 12 **Explanation:** Steps of a possible scenario are: - Apply the operation on pile 2. The resulting piles are [4,3,_3_ ,7]. - Apply the operation on pile 3. The resulting piles are [4,3,3,_4_]. - Apply the operation on pile 0. The resulting piles are [_2_ ,3,3,4]. The total number of stones in [2,3,3,4] is 12.

Constraints:

* `1 <= piles.length <= 105` * `1 <= piles[i] <= 104` * `1 <= k <= 105`

Code Snippets

C++:

```
class Solution {
public:
int minStoneSum(vector<int>& piles, int k) {

}
};
```

Java:

```
class Solution {
public int minStoneSum(int[] piles, int k) {

}
}
```

Python3:

```
class Solution:
def minStoneSum(self, piles: List[int], k: int) -> int:
```