1471. The k Strongest Values in an Array

Given an array of integers `arr` and an integer `k`.

A value `arr[i]` is said to be stronger than a value `arr[j]` if `|arr[i] - m| > |arr[j] - m|` where `m` is the **median** of the array.
If `|arr[i] - m| == |arr[j] - m|`, then `arr[i]` is said to be stronger than `arr[j]` if `arr[i] > arr[j]`.

Return *a list of the strongest* `k` values in the array. return the answer **in any arbitrary order**.

Median is the middle value in an ordered integer list. More formally, if the length of the list is n, the median is the element in position `((n - 1) / 2)` in the sorted list **(0-indexed)**.

- For `arr = [6, -3, 7, 2, 11]`, `n = 5` and the median is obtained by sorting the array `arr = [-3, 2, 6, 7, 11]` and the median is `arr[m]` where `m = ((5 - 1) / 2) = 2`. The median is `6`.
- For `arr = [-7, 22, 17, 3]`, `n = 4` and the median is obtained by sorting the array `arr = [-7, 3, 17, 22]` and the median is `arr[m]` where `m = ((4 - 1) / 2) = 1`. The median is `3`.

Example 1:

Input: arr = [1,2,3,4,5], k = 2
Output: [5,1]
Explanation: Median is 3, the elements of the array sorted by the strongest are [5,1,4,2,3].
The strongest 2 elements are [5, 1]. [1, 5] is also **accepted** answer.
Please note that although |5 - 3| == |1 - 3| but 5 is stronger than 1 because 5 > 1.

Example 2:

Input: arr = [1,1,3,5,5], k = 2
Output: [5,5]
Explanation: Median is 3, the elements of the array sorted by the strongest are [5,5,1,1,3].
The strongest 2 elements are [5, 5].

Example 3:

Input: arr = [6,7,11,7,6,8], k = 5
Output: [11,8,6,6,7]
Explanation: Median is 7, the elements of the array sorted by the strongest are [11,8,6,6,7,7].
Any permutation of [11,8,6,6,7] is **accepted**.

Example 4:

Input: arr = [6,-3,7,2,11], k = 3
Output: [-3,11,2]

Example 5:

Input: arr = [-7,22,17,3], k = 2
Output: [22,17]

```
def getStrongest(self, arr: List[int], k: int) -> List[int]:
    #     n = len(arr)
    #     medidx  = (n-1)//2
    #     arr = sorted(arr)
    #     median = arr[medidx]
    #     heap = []
    #     count = 0
    #     for i in range(len(arr)):
    #         heapq.heappush(heap,(abs(arr[i]-median),arr[i]))
    #         count = count+1
    #         if count == k:
    #             break
    #     for j in range(i+1,len(arr)):
    #         val, element = heap[0]
    #         if val<abs(arr[j]-median):
    #             heapq.heappop(heap)
    #             heapq.heappush(heap,(abs(arr[j]-median),arr[j]))
    #         elif val == abs(arr[j]-median) and element<arr[j]:
    #             heapq.heappop(heap)
    #             heapq.heappush(heap,(abs(arr[j]-median),arr[j]))
    #     res = []
    #     while heap:
    #         val, ele = heapq.heappop(heap)
    #         res.append(ele)
        # return res

        arr = sorted(arr)
        medidx = (len(arr)-1)//2
        median = arr[medidx]
        res = []
        res = sorted(arr,reverse = True, key = lambda x:(abs(x-median),x))
[:k]
        return res
```

```python
def getStrongest(self, arr: List[int], k: int) -> List[int]:
        arr.sort()
        n = len(arr)
        m = (n-1)//2
        median = arr[m]
        res = sorted(arr,key=lambda x:(-abs(x-median),-x))[:k]
        return res
```