int g(arr1, arr2, k)
ordered ordered

int f(arr1, arr2) getUpperMedian
ordered and same length
two arr s

f(arr1, L1, R1, arr2, L2, R2)

arr1[L1, ··· R1] make sure ordered
arr2[L2 ··· R2]

1) arr1[1, 2, 3, 4 ···]
2) arr2[1', 2', 3', 4' ···]

length
even

1) arr1[1, 2, 3, 4]
2) arr2[1', 2', 3', 4']

To return upper median, 4th smallest

if 2 == 2' return either value
(1, 1') (2, 2') (···)

if 2 > 2'
arr1 [3] [4] X won't be the best value
because [3] > [2] > [2'] > [1']
> [1]

[2] > [2']
> [1'] ? [1]

[1, 2] [3', 4'] are possible

if 3' ≤ 1 ≤ 4'
1 2'

fourth smallest number

1 ≤ 2 ≤ 3', 4'
1', 2' 3, 4

conclusion
arr1 [1, 2 / 3' 4'] the second smallest number in the potential array is the fourth smallest number in original array

e.g. 3' ≤ 1 2 ≤ 4'

1 ≤ 2 ≤ 3', 4'

odd
arr1 [1, 2, 3, 4, 5]
arr2 [1', 2', 3', 4', 5']
10 numbers

arr1 [1, 2, 3, 4, 5]
arr2 [1', 2', 3, 4', 5']

if 3' == 3 return
if 3 > 3'
Can't directly call f function because require same length
2 ≤ 3'
verify whether [3'] is fifth small or not
if [3'] > [2] return [3'] directly
else [1', 2'] [4, 5] in f function same length
second largest == fifth largest in total

g function
g(arr1, arr2) both sorted, may not equal length
return kth smallest after sorted combine

arr1 10 nums
arr2 17 nums

kth
7th smallest

Condition [1] 1 ≤ k ≤ 10 ← within below smallest length
obtain first 7th from arr1, arr2
getUpperMedian

Condition [2] len(s) < k ≤ len(L) k=15

arr1: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
arr2: 1', 2', 3', 4', 5', 6', 7', 8', 9', 10', 11', 12', 13', 14', 15', 16', 17'

consider all arr
arr1: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
arr2: 1', 2', 3', 4', 5', 6', 7', 8', 9', 10', 11', 12', 13', 14', 15', 16', 17'
but this time still can't call getUpmedian
f (_ _) equal length

consider all arr
arr1: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
arr2: 1', 2', 3', 4', 5', 6', 7', 8', 9', 10', 11', 12', 13', 14', 15', 16', 17'
Check whether [5'] is 15th smallest
if yes: return
if not: remove

1, 2, 3, 4, 5, 6, 7, 8, 9, 10
5', 6', 7', 8', 9', 10', 11', 12', 13', 14', 15'
f(10 10)

Condition [3] 17 < k ≤ 27
len(L) < k ≤ len(s) + len(L) if k == 23

5+17=22
arr1: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
arr2: 1', 2', 3', 4', 5', 6', 7', 8', 9', 10', 11', 12', 13', 14', 15', 16', 17'
12+10=22

can't merge [6, 7, 8, 9, 10] 5
[13', 14', 15', 16', 17'] to f function

because (5 / 12) 17 + 5 = 22 smallest

manually purge 2 nums
[6]
[13']

6
arr1: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
arr2: 1', 2', 3', 4', 5', 6', 7', 8', 9', 10', 11', 12', 13', 14', 15', 16', 17'
13

(6 / 13) 19 + 4 = 23

f(=) equal length

O(log(len(small)))

```python
def findKthNum(arr1, arr2, kth):
    longs = arr1 if len(arr1) >= len(arr2) else arr2
    shorts = arr1 if len(arr1) < len(arr2) else arr2
    l, s  = len(longs), len(shorts)
    if kth <= s:
        return getUpMedian(shorts, 0, kth - 1, longs, 0, kth - 1)
    if kth > l:  but smaller than sum
        if shorts[kth - l - 1] >= longs[l - 1]: exclude by hand
            return shorts[kth - l - 1]
        if longs[kth - s - 1] >= shorts[s - 1]:  exclude by hand
            return longs[kth - s - 1]
        return getUpMedian(shorts, kth - l, s - 1, longs, kth - s, l - 1)
    if longs[kth - s - 1] >= shorts[s - 1]:
        return longs[kth - s - 1]
    return getUpMedian(shorts, 0, s - 1, longs, kth - s, kth - 1)
```