You are given a sorted integer array arr containing 1 and **prime** numbers, where all the integers of arr are unique. You are also given an integer k.

For every i and j where 0 <= i < j < arr.length, we consider the fraction arr[i] / arr[j].

Return *the* kth *smallest fraction considered*. Return your answer as an array of integers of size 2, where answer[0] == arr[i] and answer[1] == arr[j].

**Example 1:**

Input: arr = [1,2,3,5], k = 3  
Output: [2,5]  
Explanation: The fractions to be considered in sorted order are:  
1/5, 1/3, 2/5, 1/2, 3/5, and 2/3.  
The third fraction is 2/5.

**Example 2:**

Input: arr = [1,7], k = 1  
Output: [1,7]

**Constraints:**

* 2 <= arr.length <= 1000
* 1 <= arr[i] <= 3 \* 104
* arr[0] == 1
* arr[i] is a **prime** number for i > 0.
* All the numbers of arr are **unique** and sorted in **strictly increasing** order.
* 1 <= k <= arr.length \* (arr.length - 1) / 2

**Follow up:** Can you solve the problem with better than O(n2) complexity?