786. K-th Smallest Prime Fraction

A sorted list `A` contains 1, plus some number of primes. Then, for every p < q in the list, we consider the fraction p/q.

What is the `K`-th smallest fraction considered? Return your answer as an array of ints, where `answer[0] = p` and `answer[1] = q`.

```
Examples:
Input: A = [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, 2/3.
The third fraction is 2/5.

Input: A = [1, 7], K = 1
Output: [1, 7]
```

Note:

- `A` will have length between `2` and `2000`.
- Each `A[i]` will be between `1` and `30000`.
- `K` will be between `1` and `A.length * (A.length + 1) / 2`.

This solution probably doesn’t have the best runtime but it’s really simple and easy to understand.

Says if the list is `[1, 7, 23, 29, 47]`, we can easily have this table of relationships

```
1/47  < 1/29    < 1/23 < 1/7
7/47  < 7/29    < 7/23
23/47 < 23/29
```

29/47

So now the problem becomes “find the kth smallest element of (n-1) sorted list”

Following is my implementation using PriorityQueue, running time is ~~O(nlogn)~~ O(max(n,k) * logn), space is O(n):

```
//java
 public int[] kthSmallestPrimeFraction(int[] a, int k) {
        int n = a.length;
        // 0: numerator idx, 1: denominator idx
        PriorityQueue<int[]> pq = new PriorityQueue<>(new Comparator<int[]>() {
            @Override
            public int compare(int[] o1, int[] o2) {
                int s1 = a[o1[0]] * a[o2[1]];
                int s2 = a[o2[0]] * a[o1[1]];
                return s1 - s2;
            }
        });
        for (int i = 0; i < n-1; i++) {
            pq.add(new int[]{i, n-1});
        }
        for (int i = 0; i < k-1; i++) {
            int[] pop = pq.remove();
            int ni = pop[0];
            int di = pop[1];
            if (pop[1] - 1 > pop[0]) {
                pop[1]--;
                pq.add(pop);
            }
        }

        int[] peek = pq.peek();
        return new int[]{a[peek[0]], a[peek[1]]};
    }
```

```
//c++
#include<stdio.h>
#include<iostream>
#include<vector>
#include<algorithm>
#include<limits.h>
#include<queue>
using namespace std;
vector<int> Global_A;
struct compare_func
{
     bool operator()(pair<int,int> &a, pair<int,int> &b)
     {
          int s1 = Global_A[a.first]*Global_A[b.second];
          int s2 = Global_A[b.first]*Global_A[a.second];
          return s1>s2;
     }
};
```

```
vector<int> kthSmallestPrimeFraction(vector<int>& A, int K) {
     Global_A = A;
     int n = A.size();
     priority_queue<pair<int,int>,vector<pair<int,int>>,compare_func>
pq;
     for(int i=0;i<n-1;++i)
     {
          pair<int,int> elem = {i,n-1};
          pq.push(elem);
     }
     for(int i=0;i<K-1;i++)
     {
          pair<int,int> elem = pq.top();
          pq.pop();
          int x = elem.first;
          int y = elem.second;
          if(x+1<y)
          {
               y--;
               pair<int,int> elemnew = {x,y};
               pq.push(elemnew);
          }
     }
     pair<int,int> elem = pq.top();
     vector<int> res;
     res.push_back(A[elem.first]);
     res.push_back(A[elem.second]);
     return res;
}

int main(int argc, char *argv[])
{
     vector<int> A = {1, 2, 3, 5};int K = 3;
     vector<int> ans = kthSmallestPrimeFraction(A,K);
     for(auto elem : ans) cout << elem << endl;
     return 0;
}
```