### 3.10第 K 个最小的素数分数

#### 3.10.1题目及描述

给你一个按递增顺序排序的数组 *arr* 和一个整数 *k* 。数组 *arr* 由 1 和若干素数组成，且其中所有整数互不相同。

对于每对满足的 *i* 和 *j* ，可以得到分数。

那么第 *k* 个最小的分数是多少呢?

请输出相应的*arr*[*i*]和*arr*[*j*] 。

#### 3.10.2概要设计

设计函数*vector<int>kthSmallestPrimeFraction(int k)*负责处理问题及返回结果

设计函数内结构体*struct my\_cmp*用来为优先队列提供基于*pair*的比较函数

设计全局变量*arr*用来存放每个元素



#### 3.10.3算法分析

**算法说明:**

当分母给定为时,分子可以在中进行选择,由于数组是严格递增的,那么记分子为那么随着的增加,分数的值也是严格递增的,举个例子,假定分母是,那么这几个一定是严格递增的

因此将每个分母看成一个长度为的列表,包含了

可见对于数组中的每个数字,均有自己的一个列表,这里因为题目要求,其自身的列表可以忽略,那么我们的目标就是从分母所对应的列表中,找到第小的那个分数

初始时,优先队列中储存了个分数,将带入,容易观察到这些数是,容易证明的一定是所有分数中最小的那个分数,举个例子,有素数,那么一定是最小的,所以第一个从优先队列中取出的一定是,但是凭借直觉和之间一定存在着一些额外的分数,故对于当前队列中每一个被取出的分数,如果,则需要将重新放回队列中进行排序,它或许不会被马上取出,但是也是一种可能性.

**对于一些细节:**

优先队列*priority\_queu*e并没有提供基于*pair*的比较函数,同时,这里我们不需要将所有的个分数先合成再比较,故我们需要的是基于位序在数组中所对应的分数的值的比较,所以在函数中每个*pair*对应的并不是一个真是的分数,而是分子分母在数组中的位序,故我们使用如下结构体对优先队列进行初始化:

***struct my\_cmp  
{  
 bool operator ()(const pair<int,int>&x,const pair<int,int>&y)  
 { return arr[x.first]\*arr[y.second]>arr[x.second]\*arr[y.first]; }  
};***

***priority\_queue<pair<int,int>,vector<pair<int,int>>,my\_cmp>que;***

比较分数会损失精度,对此可以将用来判断

**时间复杂度分析:**

第一个*for*循环插入次单次插入,故是

第二个*for*循环插入与删除次,单次插入,故是

一般的,*n*会比*k*大,故总的时间复杂度是

#### 3.10.4使用说明

输入说明 :

输入三行：

第一行输入一个整数*n*表示数组*arr*的长度。

第二行输入*n*个整数表示数组的元素。

第三行输入一个整数表示*k*.

提示：



*arr*[*i*] 是一个素数 ，*i* > 0

*arr* 中的所有数字互不相同 ，且按严格递增排序



#### 3.10.5测试结果及分析

* 第一组数据:

IN:

20

1 2 7 11 19 29 31 43 53 67 71 79 83 97 103 113 131 139 157 173

136

OUT:

31 53

* 第二组数据:

IN:

25

1 3 5 7 17 29 41 47 59 61 67 73 83 101 109 131 139 149 157 167 181 197 211 227 239

82

OUT:

17 149

* 第三组数据:

IN:

40

1 5 13 19 23 31 41 43 59 67 71 79 97 107 109 131 139 151 167 181 191 197 211 227 229 241 251 269 271 277 293 307 311 313 337 353 373 383 389 397

541

OUT:

181 293

* 第四组数据:

IN:

40

1 5 7 17 23 29 31 41 43 47 59 67 79 89 101 103 109 131 139 157 163 167 181 193 211 223 233 241 257 271 283 293 313 337 349 359 379 383 389 409

577

OUT:

139 223

* 第五组数据:

IN:

40

1 5 13 19 23 31 43 59 61 67 79 97 101 107 109 113 131 137 149 163 173 191 197 211 229 239 257 271 277 283 307 317 337 347 353 359 379 383 401 419

202

OUT:

59 337

* 第六组数据:

IN:

70

1 11 197 487 683 727 887 1223 1583 1657 1823 2129 2207 2251 2399 2753 3089 3301 3623 3691 4007 4079 4253 4283 4597 4871 5227 5483 5527 5743 5927 6007 6359 6421 6607 6673 6737 6841 6869 7247 7549 7841 8101 8179 8243 8329 8363 8539 8741 9127 9391 9643 9719 10007 10273 10723 10733 11093 11251 11551 11807 12007 12011 12281 12289 12539 12739 12791 13127 13183

1645

OUT:

6421 9643

* 第七组数据:

IN:

500

1 2 3 5 13 23 29 41 43 47 53 59 67 79 83 89 97 107 113 131 139 151 163 167 173 179 193 211 229 241 257 271 283 311 313 337 347 353 359 367 373 383 397 419 421 439 443 449 457 467 479 491 499 521 541 563 577 593 607 613 631 641 647 659 677 691 709 727 743 751

769 797 809 821 829 857 859 863 877 887 911 937 953 967 983 991 1009 1013 1031 1033 1049 1061 1087 1097 1103 1123 1129 1163 1181 1193 1217 1231 1237 1259 1283 1297 1301 1319 1327 1367 1381 1399 1427 1429 1447 1459 1481 1489 1511 1523 1549 1567 1583 1607 1619 1621 1627 1657 1663 1667 1697 1699 1721 1741 1753 1783 1789 1811 1823 1831 1867 1871 1873 1877 1889 1901 1931 1933 1973 1979 1987 1999 2011 2027 2029 2063 2083 2099 2113 2129 2141 2161 2179 2203 2213 2239 2243 2267 2269 2287 2293 2311 2339 2351 2357 2377 2389 2399 2423 2437 2441 2447 2467 2503 2531 2539 2551 2579 2591 2609 2617 2633 2647 2663 2683 2687 2699 2707 2711 2729 2731 2753 2789 2801 2833 2851 2857 2887 2909 2939 2953 2963 2999 3001 3011 3019 3037 3049 3067 3079 3083 3089 3121 3167 3169 3181 3187 3191 3217 3229 3251 3259 3301 3319 3331 3347 3359 3371 3391 3433 3461 3463 3491 3499 3511 3527 3539 3557 3559 3581 3583 3607 3613 3631 3643 3659 3671 3677 3691 3697 3709 3719 3739 3767 3793 3797 3821 3847 3851 3863 3881 3907 3917 3919 3923 3931 3947 3967 4001 4003 4019 4049 4057 4091 4111 4133 4139 4153 4159 4177 4217 4231 4253 4271 4273 4297 4327 4337 4349 4363 4373 4391 4421 4423 4447 4463 4493 4507 4513 4519 4549 4583 4603 4621 4637 4643 4649 4657 4679 4721 4723 4733 4751 4787 4799 4813 4831 4871 4903 4909 4931 4937 4943 4951 4967 4969 4973 4993 5003 5009 5023 5059 5081 5101 5119 5167 5171 5179 5209 5233 5261 5281 5303 5323 5347 5387 5399 5417 5431 5443 5449 5477 5501 5519 5531 5569 5573 5623 5639 5641 5653 5659 5683 5701 5711 5717 5741 5743 5749 5783 5807 5813 5827 5839 5843 5849 5857 5867 5881 5897 5903 5927 5981 5987 6011 6043 6053 6073 6079 6101 6121 6143 6163 6199 6203 6211 6217 6247 6269 6271 6287 6299 6301 6317 6323 6343 6361 6373 6379 6389 6421 6427 6449 6473 6521 6529 6553 6569 6581 6599 6637 6659 6661 6679 6689 6691 6709 6733 6737 6779 6793 6827 6841 6869 6899 6917 6949 6967 6971 6977 6991 6997 7001 7027 7043 7069 7109 7129 7159 7177 7207 7211 7213 7219 7243 7247 7283 7297 7309 7321 7333 7351 7411 7451 7477 7489 7517 7537 7541 7559 7561 7583 7591 7607 7639 7649 7669 7687 7703 7727 7753 7759 7789 7817 7829 7841 7873 7879 7901 7927

114248

OUT:

4513 5009