

题目描述:

RSA 加密算法在网络安全世界中无处不在,它利用了极大整数因数分解的困难度,数据越大,安全系数越高,给定一个 32 位正整数,请对其进行因数分解,找出是哪两个素数的乘积。

输入描述:

一个正整数 num

$0 < \text{num} \leq 2147483647$

输出描述:

如果成功找到,以单个空格分割,从小到大输出两个素数,分解失败,请输出-1 -1

补充说明:

示例 1

输入:

15

输出:

3 5

说明:

因数分解后,找到两个素数 3 和 5,使得  $3*5=15$ ,按从小到大排列后,输出 3 5

示例 2

输入:

27

输出:

-1 -1

说明:

通过因数分解,找不到任何素数,使得他们的乘积为 27,输出-1 -1

# 判断是否为素数

```
def is_prime(n):  
    if n < 2:  
        return False  
    for i in range(2, int(n ** 0.5) + 1):  
        if n % i == 0:  
            return False  
    return True
```

```
# 找到素数乘积
def find_prime_product(num):
    for i in range(2, int(num ** 0.5) + 1):
        if is_prime(i) and num % i == 0:
            return i, num // i
    return -1, -1

# test
num = int(input())
num1, num2 = find_prime_product(num)

if num1 != -1 and num2 != -1:
    print(num1, num2)
else:
    print("-1 -1")
```