Prime Factorization

Given a number n, write an efficient function to print all prime factors of n. For example, if the input number is 12, then output should be “2 2 3”. And if the input number is 315, then output should be “3 3 5 7”.

Following are the steps to find all prime factors.
1) While n is divisible by 2, print 2 and divide n by 2.
2) After step 1, n must be odd. Now start a loop from i = 3 to square root of n. While i divides n, print i and divide n by i, increment i by 2 and continue.
3) If n is a prime number and is greater than 2, then n will not become 1 by above two steps. So print n if it is greater than 2.

Python

```
# Python program to print prime factors

import math

# A function to print all prime factors of
# a given number n
def primeFactors(n):

    # Print the number of two\'s that divide n
    while n % 2 == 0:
        print 2,
        n = n / 2

    # n must be odd at this point
    # so a skip of 2 ( i = i + 2) can be used
    for i in range(3,int(math.sqrt(n))+1,2):

        # while i divides n , print i ad divide n
        while n % i== 0:
            print i,
            n = n / i

    # Condition if n is a prime
    # number greater than 2
    if n > 2:
        print n
```

Dash

All

Articles

Videos

Problems

Quiz

```
# Driver Program to test above function

n = 315
primeFactors(n)
```

Output:

```
3 3 5 7
```

How does this work?
The steps 1 and 2 take care of composite numbers and step 3 takes care of prime numbers. To prove that the complete algorithm works, we need to prove that steps 1 and 2 actually take care of composite numbers. This is clear that step 1 takes care of even numbers. And after step 1, all remaining prime factor must be odd (difference of two prime factors must be at least 2), this explains why i is incremented by 2.
Now the main part is, the loop runs till square root of n not till. To prove that this optimization works, let us consider the following property of composite numbers.
Every composite number has at least one prime factor less than or equal to square root of itself.

This property can be proved using counter statement. Let a and b be two factors of n such that a*b = n. If both are greater than √n, then a.b > √n, * √n, which contradicts the expression "a * b = n".

In step 2 of the above algorithm, we run a loop and do following in loop
a) Find the least prime factor i (must be less than √n,)
b) Remove all occurrences i from n by repeatedly dividing n by i.
c) Repeat steps a and b for divided n and i = i + 2. The steps a and b are repeated till n becomes either 1 or a prime number.

Mark as Read