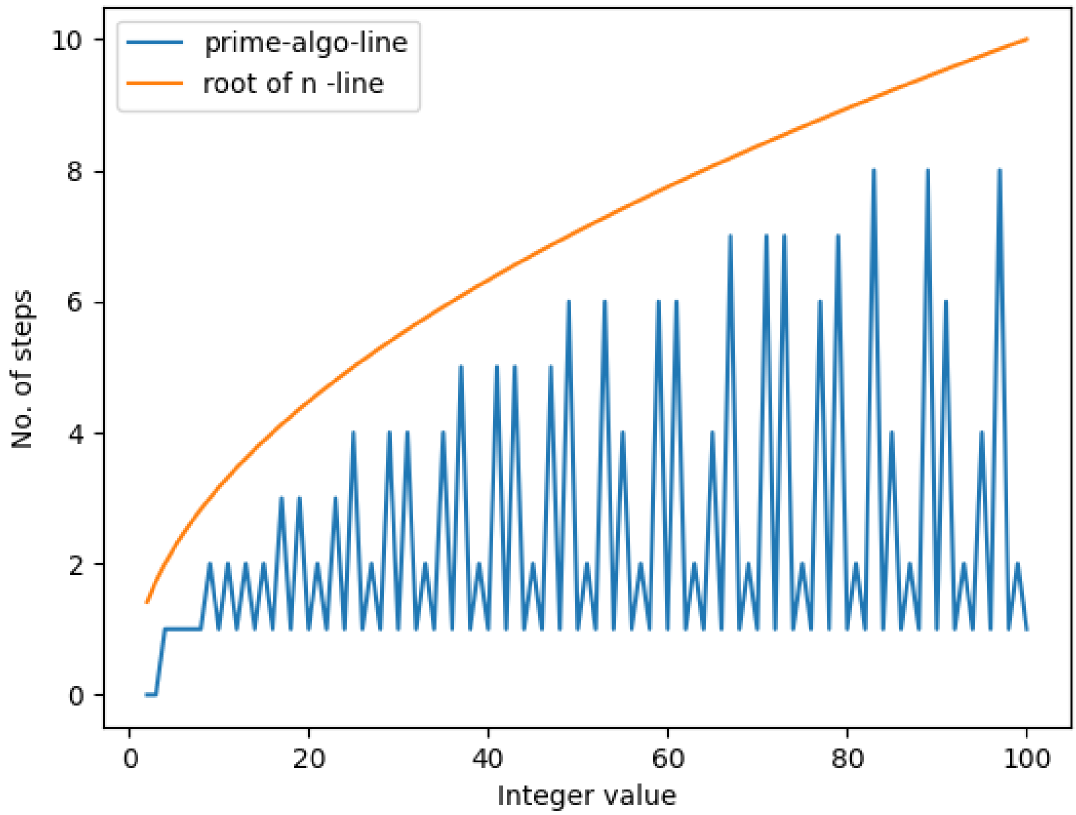
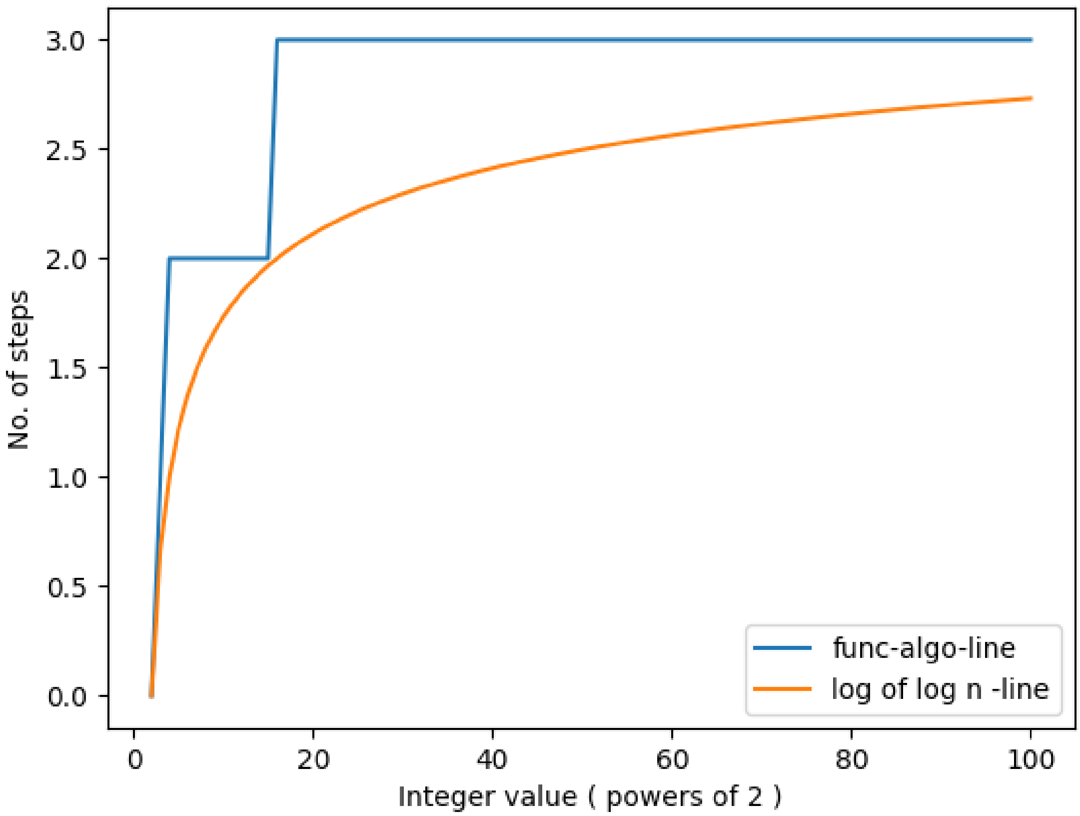
Name : Sabarivasan V
Register Number : 205001085
Clas : CSE - B

```
# 1. Implement an algorithm to find whether a number is prime
# or not for different sizes of 'n'. Write the recurrence relation
# and solve it. Plot the graph with 'n' in x-axis and the number of steps
# required to produce output in y-axis

import matplotlib.pyplot as plt
from math import sqrt

def prime(n):
    count =  0
    for i in range(2,int(sqrt(n))+1):
        count+=1
        if n%i==0:
            return count
    return count


xpoints = [ i for i in range(2,101) ]
ypoints = [ prime(i) for i in range(2,101) ]
zpoints = [ sqrt(i) for i in range(2,101) ]

plt.plot(xpoints, ypoints,label ='prime-algo-line')
plt.plot(xpoints, zpoints,label ='root of n -line')
plt.xlabel("Integer value")
plt.ylabel("No. of steps")
plt.legend()
plt.show()
```



T(n) = sqrt(n)
O(sqrt(n))

```
# 2. Implement an algorithm to find square root of a number ‘n’
recursively
# until it is greater than 2. Write the recurrence relation and solve it.
# Plot the graph with n in x-axis and the number of steps it takes to
reach 2
# in y axis. Let ‘n’ can be in powers of 2.

import matplotlib.pyplot as plt
from math import log2

def function(n,l) :
    x=n
    while (1) :
        root = 0.5 * (x + (n / x))
        if (abs(root - x) < l) :
            break
        x = root
    print (root)
    return root

def final(i,count):
    if(i<=2):
        print("---------------")
        return count
    count+=1
    i=function(i,0.01)
    return final(i,count)

xpoints = [ i for i in range(2,101) ]
ypoints = [ final(i,0) for i in xpoints ]
zpoints = [ log2(log2(i)) for i in range(2,101) ]


plt.plot(xpoints, ypoints,label ='func-algo-line')
plt.plot(xpoints, zpoints,label ='log of log n -line')
plt.xlabel("Integer value ( powers of 2 )")
plt.ylabel("No. of steps")
plt.legend()
plt.show()
```



$T(n) = T(n^{(1/2)}) + 1$
$T(n^{(1/2)}) = T(n^{(1/4)}) + 1$
$T(n^{(1/4)}) = T(n^{(1/8)}) + 1$
$T(n^{(2^{(-k)})}) = T(n^{(2^{(-k-1)})}) + 1$
$T(n) = T(n^{(1/4)}) + 1 + 1$
$T(n) = T(n^{(1/8)}) + 1 + 1 + 1$
$T(n) = T(n^{(2^{(-k)})}) + k$

At $T(2) = 0$, $\log n = 2^{(k)}$, $k = \log(\log n)$
$T(n) = \log(\log n)$

theta(log(logn))