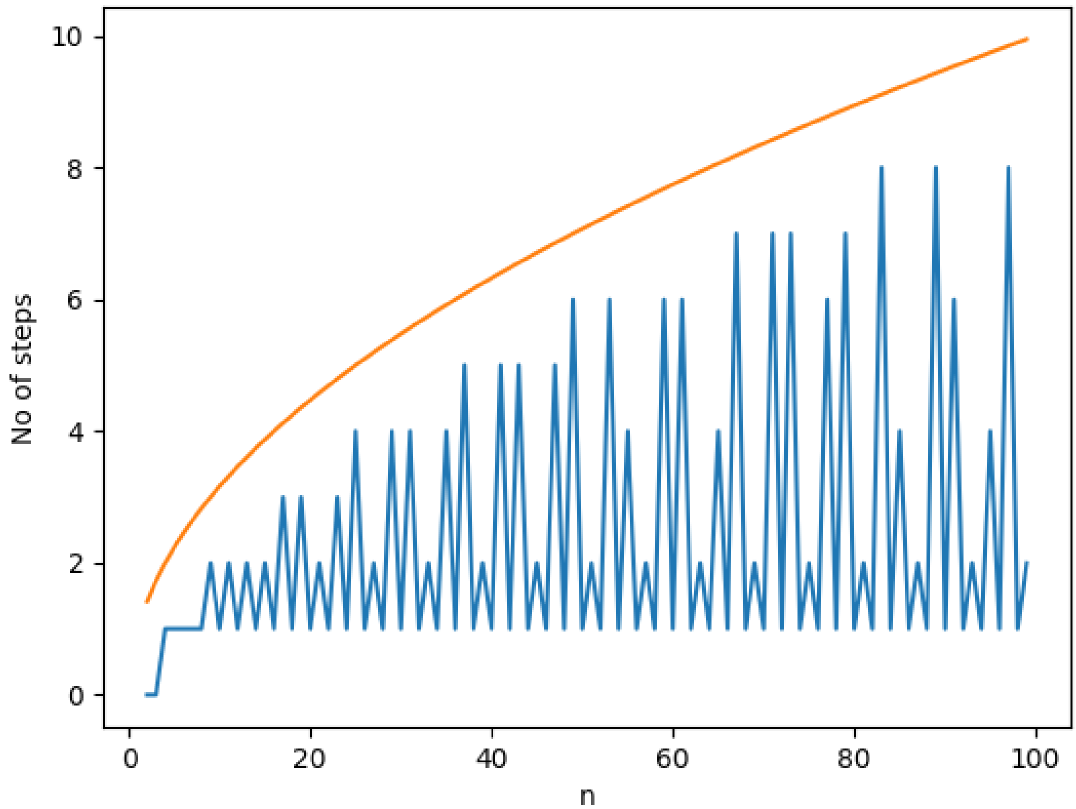
```python
import matplotlib.pyplot as plt
import math
def fun(n):
    c=0
    for i in range (2,(int)(math.sqrt(n)+1)):
        c+=1
        if n%i==0:
            return c
    return c




x1=[]
y1=[]
y2=[]
for i in range(2,100):
    c=fun(i)
    x1.append(i)
    y1.append(c)
    y2.append(math.sqrt(i))

plt.plot(x1,y1)
plt.plot(x1,y2)
plt.xlabel("n")
plt.ylabel("No of steps")
plt.show()
```

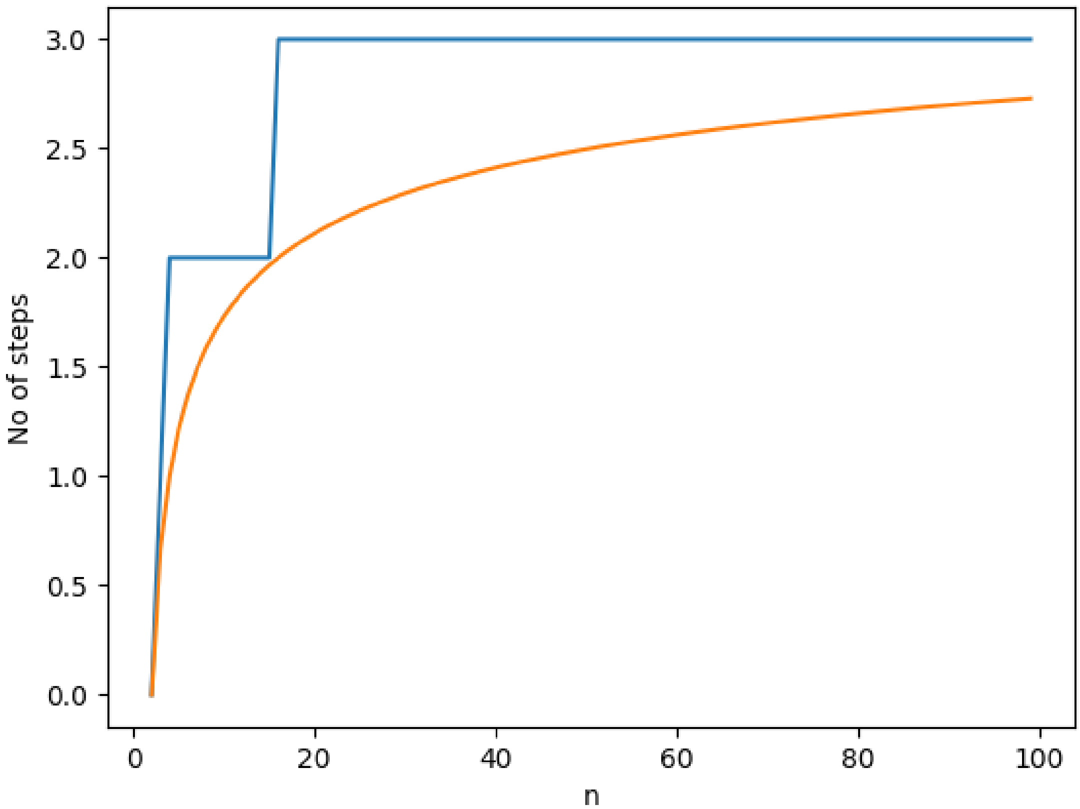


T(n) = sqrt(n)
O(sqrt(n))

2)

```
import matplotlib.pyplot as plt
import math

def fun(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,c):
    if(i<=2):
        return c
    c+=1
    i=fun(i,0.01)
    return final(i,c)

x1=[]
y1=[]
y2=[]
for i in range(2,100):
    c=final(i,0)
    x1.append(i)
    y1.append(c)
    y2.append(math.log2(math.log2(i)))

plt.plot(x1,y1)
plt.plot(x1,y2)
plt.xlabel("n")
plt.ylabel("No of steps")
plt.show()
```



T(n) = T(n^(½)) + 1
T(n^(½)) = 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^(¼)) + 1 + 1
T(n) = T(n^(⅛)) + 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))