UCS1403 Design and Analysis of Algorithms
AY: 2021-22

Name: *Krithika Swaminathan*
Reg. no.: *205001057*

DAA Lab – Assignment 3

Divide and Conquer Algorithms and Recurrence Relations

1. Recurrence relations

Code:

```
'''Use import matplotlib.pyplot as plt and plot the graph for n and
complexity for the following recurrence relations:
a.T(n)=T(n−1)+n
b.T(n)=T(n−1)+n2
c.T(n)=T(n−1)+logn
d.T(n)=T(n/2)+logn
e.T(n)=T(√n)+logn'''

import matplotlib.pyplot as plt
import numpy as np
import math

result = []

def func_a(n):
      if (n==1):
            result.append(1)
            return 1
      else:
            fn = func_a(n-1)+n
            result.append(fn)
            return fn

def func_b(n):
      if (n==1):
            result.append(1)
            return 1
      else:
            fn = func_b(n-1)+(n**2)
            result.append(fn)
            return fn

def func_c(n):
      if (n==1):
            result.append(1)
            return 1
      else:
            fn = func_c(n-1)+math.log(n,2)
            result.append(fn)
            return fn
```

Name: *Krithika Swaminathan*
Reg. no.: *205001057*

```
def func_d(n):
      if (n==1):
            result.append(1)
            return 1
      else:
            fn = func_d(n//2)+math.log(n,2)
            result.append(fn)
            return fn

def func_e(n):
      if (n==2):
            result.append(1)
            return 1
      else:
            fn = func_e(math.floor(n**0.5))+math.log(n,2)
            result.append(fn)
            return fn

def getx(n):
      nums = []
      for i in range(1,n+1):
            nums.append(i)
      return np.array(nums)

def getsmoothx(n):
      nums = []
      i = 1
      while (i<=n):
            nums.append(i)
            i *= 2
      return np.array(nums)

def getrootx(n):
      nums = []
      m = 1
      i = 2**m
      while (i<=n):
            nums.append(i)
            m *= 2
            i = 2**m
      return np.array(nums)

print("RECURRENCE RELATIONS: ")
```

Output:

```
RECURRENCE RELATIONS:
```

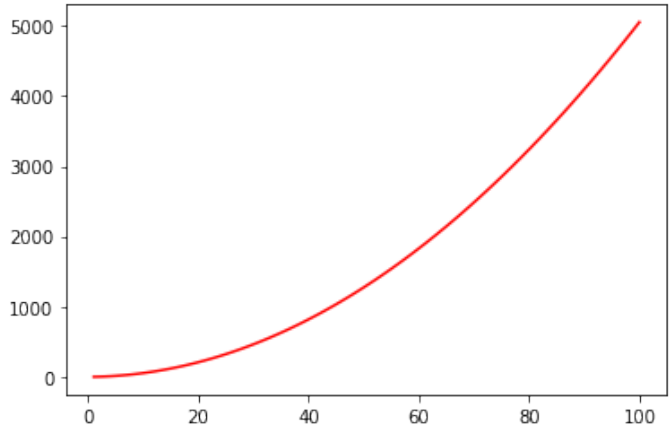
Name: *Krithika Swaminathan*
Reg. no.: *205001057*

Code:

```
print("Subdivision 1: ")
num = int(input("Enter number: "))
xpoints = getx(num)

result = []
print("T(n)=",func_a(num))
ypoints = np.array(result)

plt.plot(xpoints,ypoints,color='r')
plt.show()
```

Output:

```
Subdivision 1:
Enter number: 100
T(n)= 5050
```



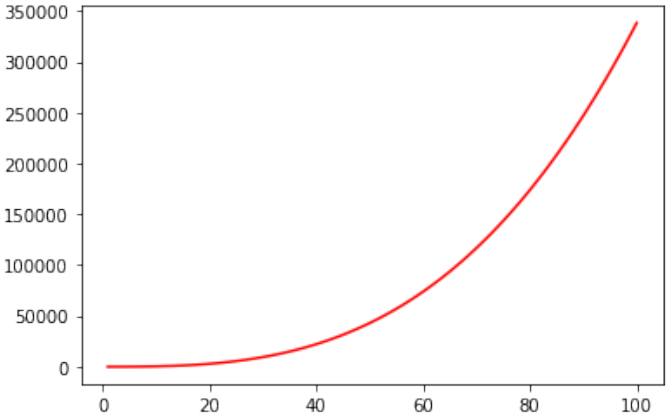
Code:

```
print("Subdivision 2: ")
num = int(input("Enter number: "))
xpoints = getx(num)

result = []
print("T(n)=",func_b(num))
ypoints = np.array(result)

plt.plot(xpoints,ypoints,color='r')
plt.show()
```

Output:

```
Subdivision 2:
Enter number: 100
T(n)= 338350
```

Name: *Krithika Swaminathan*
Reg. no.: *205001057*

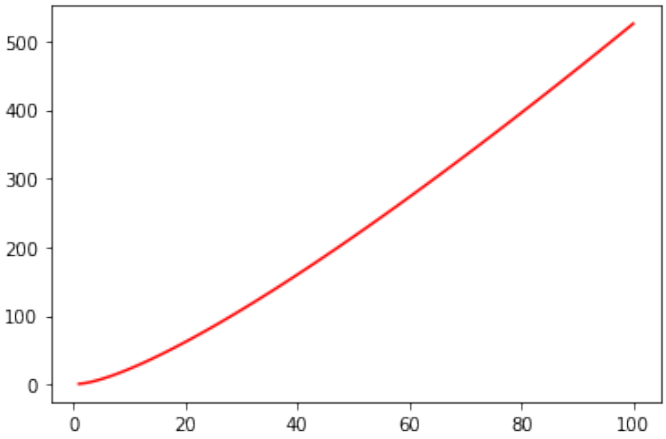


Code:

```
print("Subdivision 3: ")
num = int(input("Enter number: "))
xpoints = getx(num)

result = []
print("T(n)=",func_c(num))
ypoints = np.array(result)

plt.plot(xpoints,ypoints,color='r')
plt.show()
```

Output:

```
Subdivision 3:
Enter number: 100
T(n)= 525.7649932900597
```



Code:

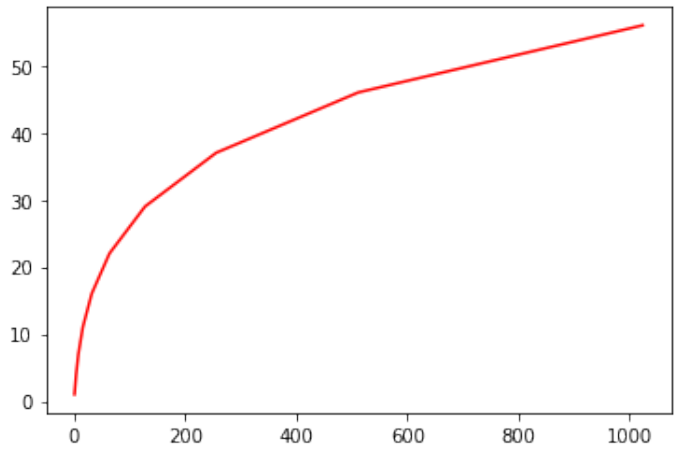
```
print("Subdivision 4: ")
num = int(input("Enter number: "))
xpoints = getsmoothx(num)
```

Name: *Krithika Swaminathan*
Reg. no.: *205001057*

```
result = []
print("T(n)=",func_d(num))
ypoints = np.array(result)

plt.plot(xpoints,ypoints,color='r')
plt.show()
```

Output:

```
Subdivision 4:
Enter number: 1024
T(n)= 56.0
```



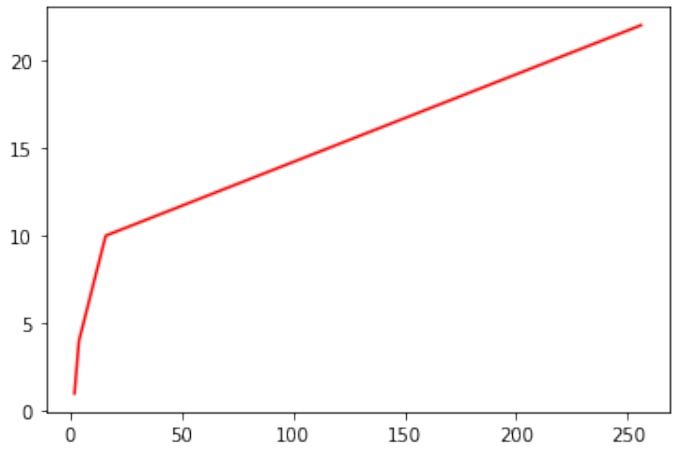
Code:

```
print("Subdivision 5: ")
num = int(input("Enter number: "))

xpoints = getrootx(num)

result = []
print("T(n)=",func_e(num))
ypoints = np.array(result)

plt.plot(xpoints,ypoints,color='r')
plt.show()
```

Output:

```
Subdivision 5:
Enter number: 4096
T(n)= 22.0
```

Name: *Krithika Swaminathan*
Reg. no.: *205001057*



2. Divide and Conquer Multiplication

Code:

```
'''Implement Strassen Matrix multiplication'''
import numpy as np

def split(matrix):
    row, col = matrix.shape
    row2, col2 = row//2, col//2
    return matrix[:row2, :col2], matrix[:row2, col2:], matrix[row2:, :col2],
matrix[row2:, col2:]

def strassen(x, y):
    if len(x) == 1:
        return x * y

    #Splitting the matrices into quadrants. This will be done recursively
until the base case is reached.
    a, b, c, d = split(x)
    e, f, g, h = split(y)

    #Computing the 7 products, recursively (p1, p2...p7)
    p1 = strassen(a, f - h)
    p2 = strassen(a + b, h)
    p3 = strassen(c + d, e)
    p4 = strassen(d, g - e)
    p5 = strassen(a + d, e + h)
    p6 = strassen(b - d, g + h)
    p7 = strassen(a - c, e + f)

    # Computing the values of the 4 quadrants of the final matrix c
    c11 = p5 + p4 - p2 + p6
    c12 = p1 + p2
    c21 = p3 + p4
    c22 = p1 + p5 - p3 - p7
```

Name: *Krithika Swaminathan*
Reg. no.: *205001057*

```
    #Combining the 4 quadrants into a single matrix by stacking horizontally
and vertically.
    c = np.vstack((np.hstack((c11, c12)), np.hstack((c21, c22))))

    return c

m = int(input("Enter no. of rows: "))
n = int(input("Enter no. of columns: "))

matrix1, matrix2 = [],[]

print("Enter matrix1 elements: ")
for i in range(m):
      row = []
      for j in range(n):
            row.append(int(input()))
      matrix1.append(row)
print(matrix1)

print("Enter matrix2 elements: ")
for i in range(m):
      row = []
      for j in range(n):
            row.append(int(input()))
      matrix2.append(row)
print(matrix2)

print("\nProduct:\n",strassen(np.array(matrix1),np.array(matrix2)))
```

Output:

```
~/DAA-Exercise3$ python3 strassen.py
Enter no. of rows: 2
Enter no. of columns: 2
Enter matrix1 elements:
1
2
3
4
[[1, 2], [3, 4]]
Enter matrix2 elements:
5
6
7
8
[[5, 6], [7, 8]]

Product:
 [[19 22]
 [43 50]]
```