****

**LAB EXERCISE 3**

Name: Jayannthan P T

Dept: CSE ‘A’

Roll No.: 205001049

1. Use import matplotlib.pyplot as plt and plot the graph for n and complexity for the following recurrence relations:
   1. 𝑇(𝑛)=𝑇(𝑛−1)+𝑛
   2. **Code:**

import matplotlib.pyplot as plt

import math

import numpy as np

xpoints = np.array(range(900))

**def** yarray(k):

    if(k <= 1):

        return 1

    else:

        return yarray(k-1)+k

ypoints = []

for i in xpoints:

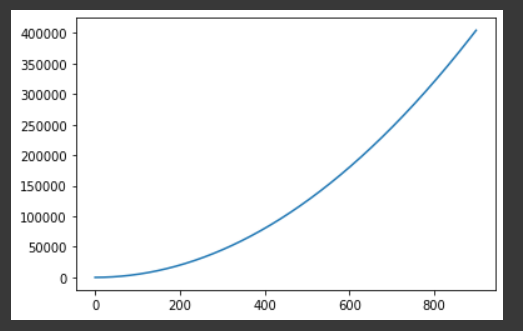
*# print(i)*

    ypoints.append(yarray(i))

plt.plot(xpoints, ypoints)

plt.show()

**Output:**



* 1. b. 𝑇(𝑛)=𝑇(𝑛−1)+𝑛2

**Code:**

*# 𝑇(𝑛)=𝑇(𝑛−1)+𝑛^2*

xpoints = np.array(range(900))

**def** yarray(k):

    if(k <= 1):

        return 1

    else:

        return yarray(k-1)+(k\*k)

ypoints = []

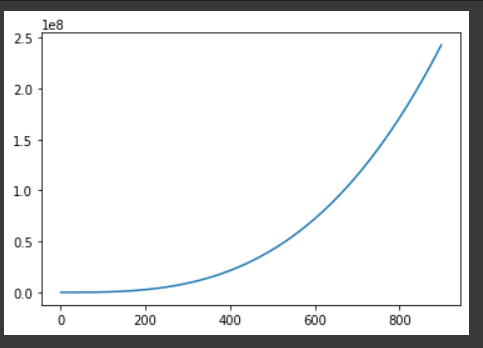
for i in xpoints:

    ypoints.append(yarray(i))

plt.plot(xpoints, ypoints)

plt.show()

**Output:**

****

* 1. c. 𝑇(𝑛)=𝑇(𝑛−1)+𝑙𝑜𝑔𝑛

**Code:**

*# 𝑇(𝑛)=𝑇(𝑛−1)+𝑙𝑜𝑔𝑛*

xpoints = np.array(range(100))

**def** yarray(k):

    if(k <= 1):

        return 1

    else:

        return yarray(k-1)+math.log(k, 2)

ypoints = []

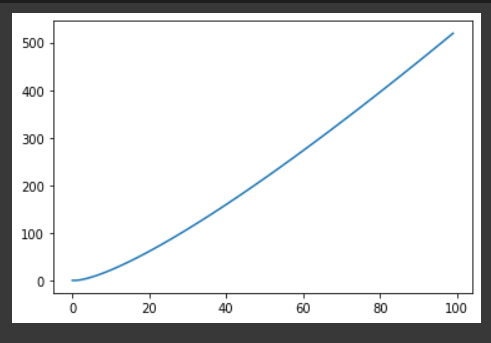
for i in xpoints:

    ypoints.append(yarray(i))

plt.plot(xpoints, ypoints)

plt.show()

**Output:**



* 1. d. 𝑇(𝑛)=𝑇(𝑛/2)+𝑙𝑜𝑔𝑛

**Code:**

*# 𝑇(𝑛) = 𝑇(𝑛/2) + 𝑙𝑜𝑔𝑛*

xpoints = np.array(range(100))

**def** yarray(k):

    if(k <= 1):

        return 1

    else:

        return yarray(k/2)+math.log(k, 2)

ypoints = []

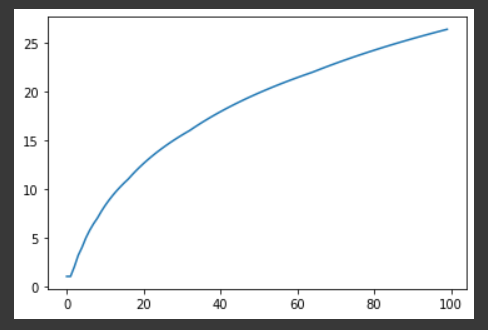
for i in xpoints:

    ypoints.append(yarray(i))

plt.plot(xpoints, ypoints)

plt.show()

**Output:**



* 1. e. 𝑇(𝑛)=𝑇 (√𝑛) +𝑙𝑜𝑔𝑛

**Code:**

*# 𝑇(𝑛) = 𝑇 (√𝑛) +𝑙𝑜𝑔𝑛*

xpoints = np.array(range(100))

**def** yarray(k):

    if(k <= 1):

        return 1

    else:

        return yarray(math.sqrt(k))+math.log(k, 2)

ypoints = []

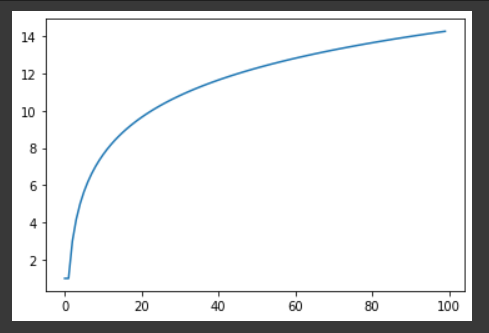
for i in xpoints:

    ypoints.append(yarray(i))

plt.plot(xpoints, ypoints)

plt.show()

**Output:**



1. Implement Strassen Matrix multiplication

**Code:**

import numpy as np

**def** split(mat):

    r, c = mat.shape

    r2, c2 = r // 2, c//2

    return mat[: r2, : c2], mat[: r2, c2:], mat[r2:, : c2], mat[r2:, c2:]

**def** strassen(x, y):

    if len(x) == 1:

        return x \* y

    a, b, c, d = split(x)

    e, f, g, h = split(y)

    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)

    c11 = p5 + p4 - p2 + p6

    c12 = p1 + p2

    c21 = p3 + p4

    c22 = p1 + p5 - p3 - p7

    c = np.vstack((np.hstack((c11, c12)), np.hstack((c21, c22))))

    return c

strassen(np.array([[1, 2], [3, 4]]), np.array([[1, 2], [3, 4]]))

**Output:**

