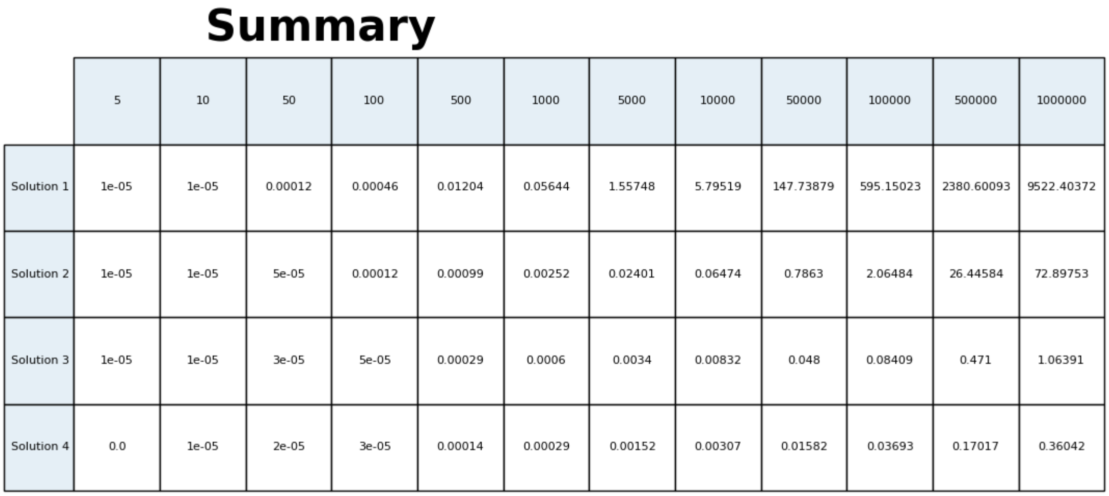
Code with auto data, table and graph generate:

```python
import math
import random
import time
import matplotlib.pyplot as plt
import numpy as np

def solution1(n):
    count = 0
    for i in range(2, n + 1):
        isPrime = True
        for j in range(2, i):
            if(i%j == 0):
                isPrime = False
        if(isPrime):
            count += 1
    return count

def solution2(n):
    count = 0
    for i in range(2, n + 1):
        isPrime = True
        for j in range(2, math.floor(math.sqrt(i))+1):
            if(i%j == 0):
                isPrime = False
        if(isPrime):
            count += 1
    return count

def solution3(n):
    # hasDivisor means divisors except 1 and the number itself
    hasDivisor = [False for i in range(n+1)] # from 0 to n
    for i in range(2, n + 1):
        for j in range(2 * i, n + 1, i):
            hasDivisor[j] = True
    return n - hasDivisor.count(True) - 1 # -1 is for 1

def solution4(n):
    # hasDivisor means divisors except 1 and the number itself
    hasDivisor = [False for i in range(n+1)] # from 0 to n
    for i in range(2, n + 1):
        if(not hasDivisor[i]):
            for j in range(i * i, n + 1, i):
                hasDivisor[j] = True
    return n - hasDivisor.count(True) - 1  #-1 is for 1

def drawGraph(x, y):
    plt.plot(x,y)
    plt.show()

def drawSummaryGraphs():
    for k in range(0, 3):
        for i in range(k, 4):
            plt.plot(inputs, times[i], label = "solution
{}".format(i+1))
        plt.legend()
        plt.show()

def drawTable(rows, columns, data, title):
    fig, ax = plt.subplots()
    ax.set_axis_off()
    rcolors = plt.cm.BuPu(np.full(len(rows), 0.1))
    ccolors = plt.cm.BuPu(np.full(len(columns), 0.1))
    table = ax.table(
        cellText = data,
        rowLabels = rows,
        colLabels = columns,
        rowColours = rcolors,
        colColours = ccolors,
        cellLoc ='center',
        loc ='upper left')

#     table.auto_set_font_size(False)
    table.set_fontsize(30)
    table.scale(2, 5)
    ax.set_title(title,
                 fontweight ="bold", fontdict={'fontsize': 30})

    plt.show()

def drawAllSingleTG():
    for i in range(4):
        drawTable(["Time"], inputs, [times[i]], 'Solution
{}'.format(i+1))
        drawGraph(inputs, times[i])

def drawStuff():
    drawTable(["Solution 1", "Solution 2", "Solution 3", "Solution
4"], inputs, times, 'Summary')
    drawSummaryGraphs()
    drawAllSingleTG()
    print("="*10 + " Times List " + "="*10)
    print(times)

def calculateTimes():
    for n in inputs:
        start_time = time.time()
        solution1(n)
        times[0].append(time.time() - start_time)

        start_time = time.time()
        solution2(n)
        times[1].append(time.time() - start_time)

        start_time = time.time()
        solution3(n)
        times[2].append(time.time() - start_time)

        start_time = time.time()
        solution4(n)
        times[3].append(time.time() - start_time)



inputs = [5, 10, 50, 100, 500, 10**3, 5 * 10**3, 10**4, 5 * 10**4,
10**5, 5 * 10**5, 10**6]
times = [[], [], [], []]
outputs = [[], [], [], []]
calculateTimes()
drawStuff()
```

Code with saved datas:

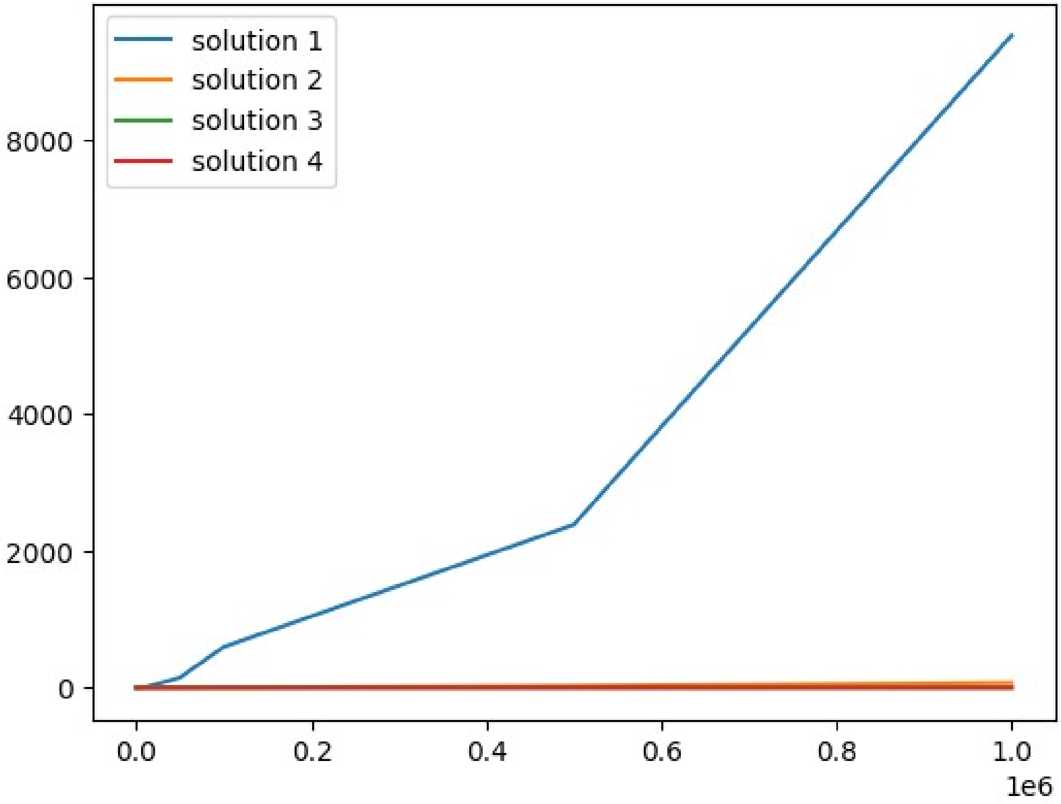
```
import matplotlib.pyplot as plt
import numpy as np

inputs = [5, 10, 50, 100, 500, 10**3, 5 * 10**3, 10**4, 5 * 10**4,
10**5, 5 * 10**5, 10**6]

times = [[8.58306884765625e-06, 8.344650268554688e-06,
0.0001227855682373047, 0.00045990943908691416, 0.01203608512878418,
0.05643773078918457, 1.5574758052825928, 5.795194864273071,
147.73878645896912, 595.1502323150635, 595.1502323150635*4,
595.1502323150635*16], [1.2874603271484375e-05, 9.5367431640625e-06,
5.221366882324219e-05, 0.00012230873107910156, 0.0009887218475341797,
0.002524852752685547, 0.02401137351989746, 0.06473803520202637,
0.7862980365753174, 2.0648396015167236, 26.445839405059814,
72.89753484725952], [8.106231689453125e-06, 6.9141387939453125e-06,
2.5272369384765625e-05, 5.030632019042969e-05, 0.0002853870391845703,
0.0006022453308105469, 0.0033957958221435547, 0.008321762084960938,
0.04799675941467285, 0.08408665657043457, 0.47100043296813965,
1.0639145374298096], [4.76837158203125e-06, 5.4836273193359375e-06,
1.5974044799804688e-05, 2.8371810913085938e-05,
0.00014400482177734375, 0.0002930164337158203, 0.0015158653259277344,
0.0030672550201416016, 0.015823841094970703, 0.03692960739135742,
0.17017102241516113, 0.360424280166626]]
for i in range(len(times)):
    for j in range(len(times[i])):
        times[i][j] = float("{:.5f}".format(times[i][j]))

def drawGraph(x, y):
    plt.plot(x,y)
    plt.show()

def drawSummaryGraphs():
    for k in range(0, 3):
        for i in range(k, 4):
            plt.plot(inputs, times[i], label = "solution
{}".format(i+1))
        plt.legend()
        plt.show()

def drawTable(rows, columns, data, title):
    fig, ax = plt.subplots()
    ax.set_axis_off()
    rcolors = plt.cm.BuPu(np.full(len(rows), 0.1))
    ccolors = plt.cm.BuPu(np.full(len(columns), 0.1))
    table = ax.table(
        cellText = data,
        rowLabels = rows,
        colLabels = columns,
        rowColours = rcolors,
        colColours = ccolors,
        cellLoc ='center',
        loc ='upper left')

#     table.auto_set_font_size(False)
    table.set_fontsize(30)
    table.scale(2, 5)
    ax.set_title(title,
                 fontweight ="bold", fontdict={'fontsize': 30})

    plt.show()

def drawAllSingleGraphs():
    for i in range(4):
        drawTable(["Time"], inputs, [times[i]], 'Solution
{}'.format(i+1))
        drawGraph(inputs, times[i])

def drawStuff():
    drawTable(["Solution 1", "Solution 2", "Solution 3", "Solution
4"], inputs, times, 'Summary')
    drawSummaryGraphs()
    drawAllSingleGraphs()
```

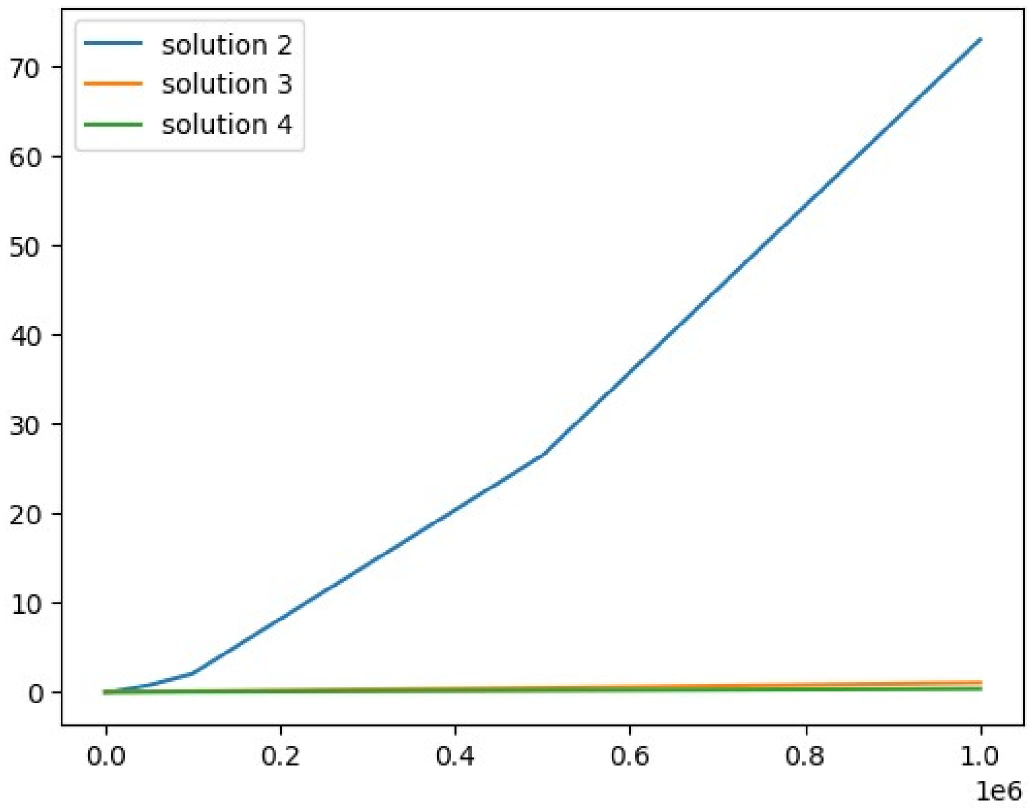
```
drawStuff()
```

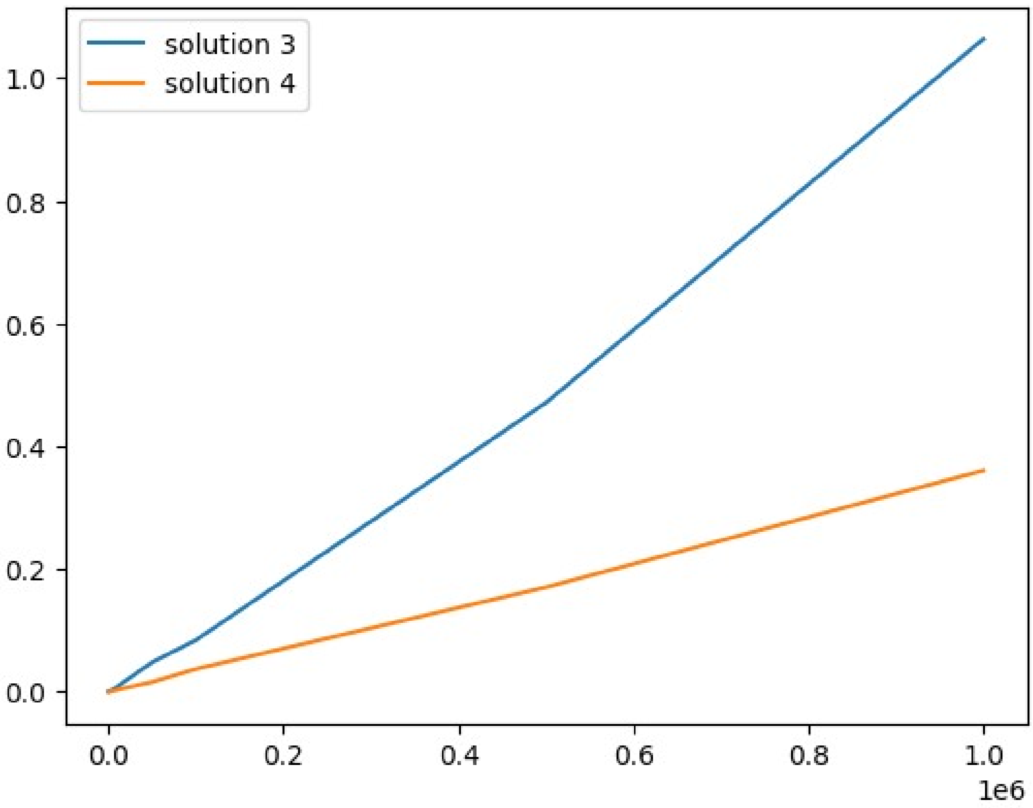


Summary

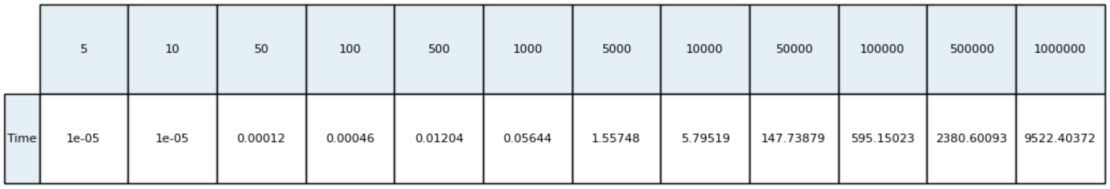
	5	10	50	100	500	1000	5000	10000	50000	100000	500000	1000000
Solution 1	1e-05	1e-05	0.00012	0.00046	0.01204	0.05644	1.55748	5.79519	147.73879	595.15023	2380.60093	9522.40372
Solution 2	1e-05	1e-05	5e-05	0.00012	0.00099	0.00252	0.02401	0.06474	0.7863	2.06484	26.44584	72.89753
Solution 3	1e-05	1e-05	3e-05	5e-05	0.00029	0.0006	0.0034	0.00832	0.048	0.08409	0.471	1.06391
Solution 4	0.0	1e-05	2e-05	3e-05	0.00014	0.00029	0.00152	0.00307	0.01582	0.03693	0.17017	0.36042



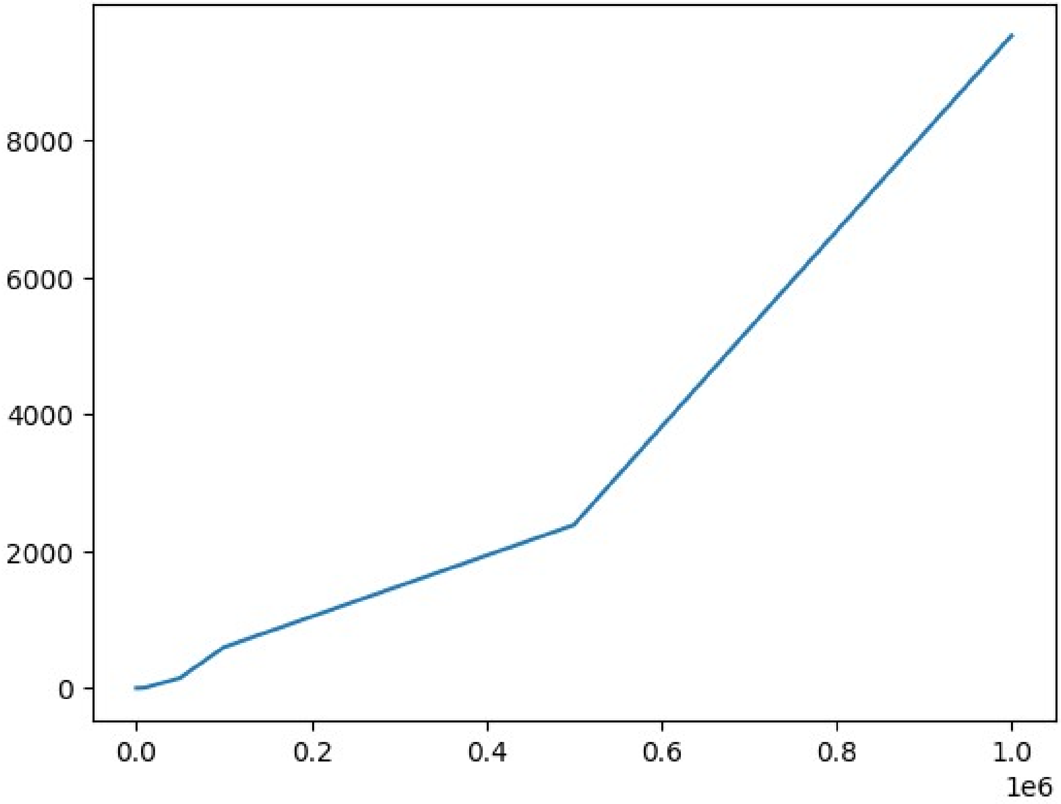

solution 2
solution 3
solution 4
70
60
50
40
30
20
10
0
0.0
0.2
0.4
0.6
0.8
1.0
1e6


solution 3
solution 4
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
1e6

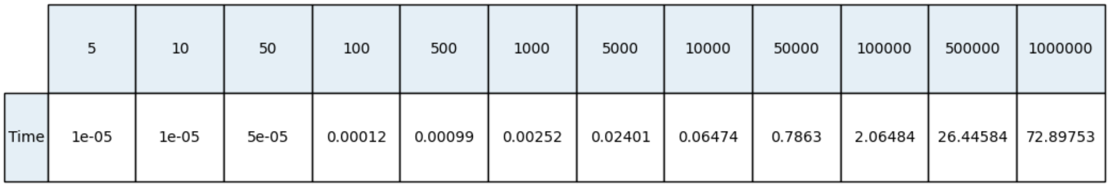
Solution 1



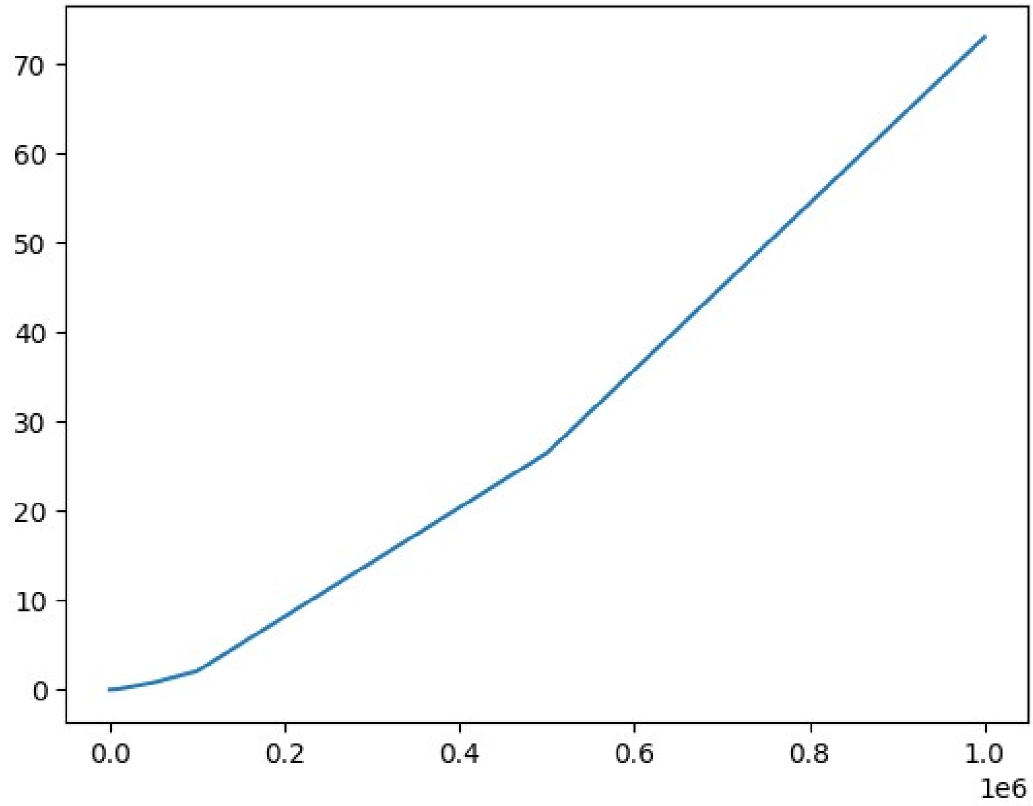
	5	10	50	100	500	1000	5000	10000	50000	100000	500000	1000000
Time	1e-05	1e-05	0.00012	0.00046	0.01204	0.05644	1.55748	5.79519	147.73879	595.15023	2380.60093	9522.40372



Solution 2

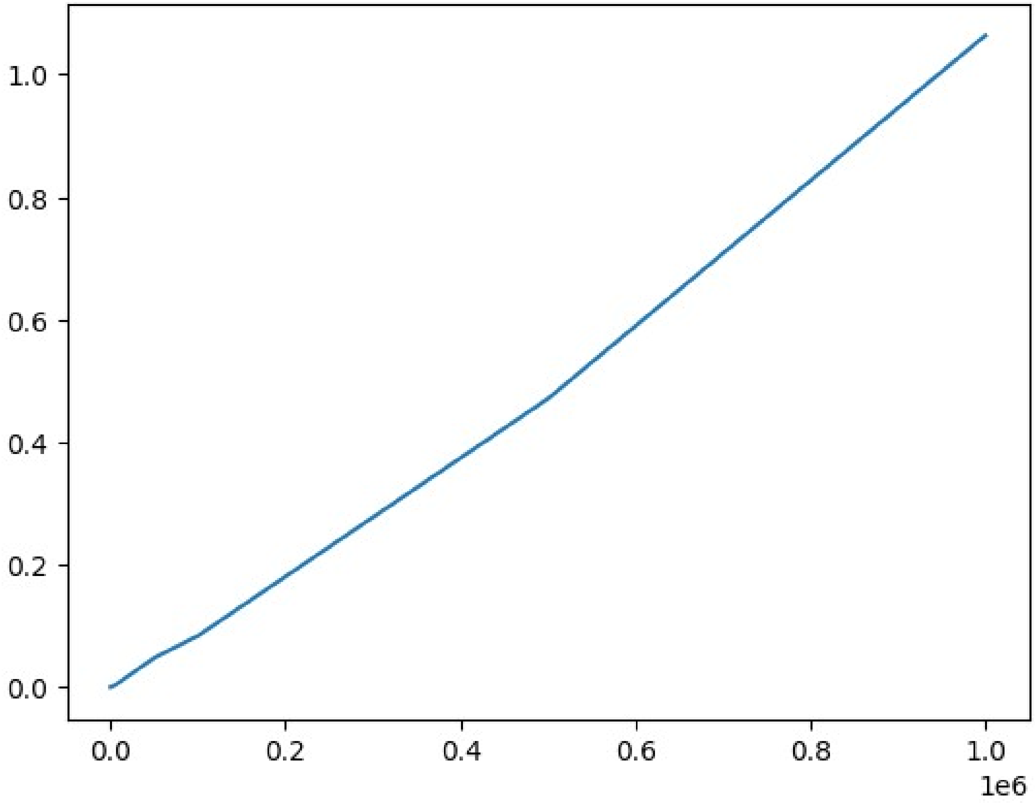


	5	10	50	100	500	1000	5000	10000	50000	100000	500000	1000000
Time	1e-05	1e-05	5e-05	0.00012	0.00099	0.00252	0.02401	0.06474	0.7863	2.06484	26.44584	72.89753

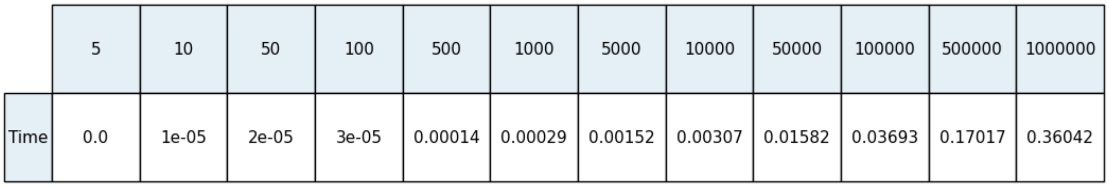


Solution 3

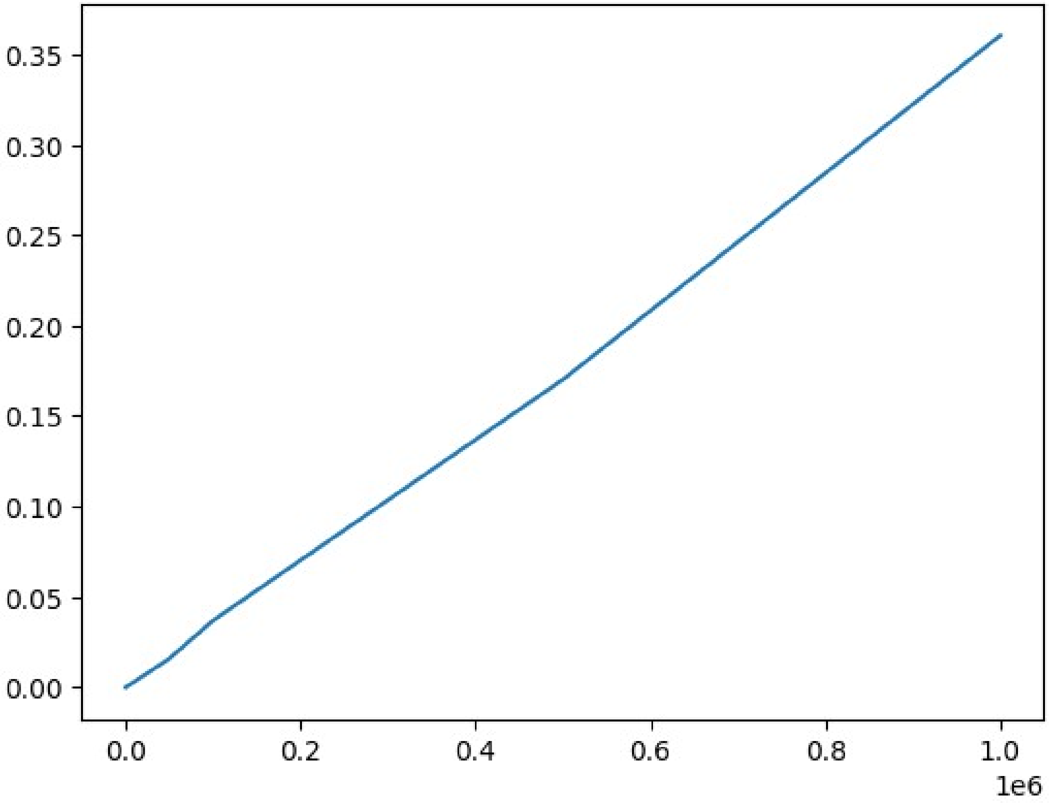
	5	10	50	100	500	1000	5000	10000	50000	100000	500000	1000000
Time	1e-05	1e-05	3e-05	5e-05	0.00029	0.0006	0.0034	0.00832	0.048	0.08409	0.471	1.06391



Solution 4



	5	10	50	100	500	1000	5000	10000	50000	100000	500000	1000000
Time	0.0	1e-05	2e-05	3e-05	0.00014	0.00029	0.00152	0.00307	0.01582	0.03693	0.17017	0.36042


0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0.0
0.2
0.4
0.6
0.8
1.0
1e6