In this part we track times while running some heavy programs like Android Studio and A game (like graphical chess) :

Code with auto data, table and graph generate:

```
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:

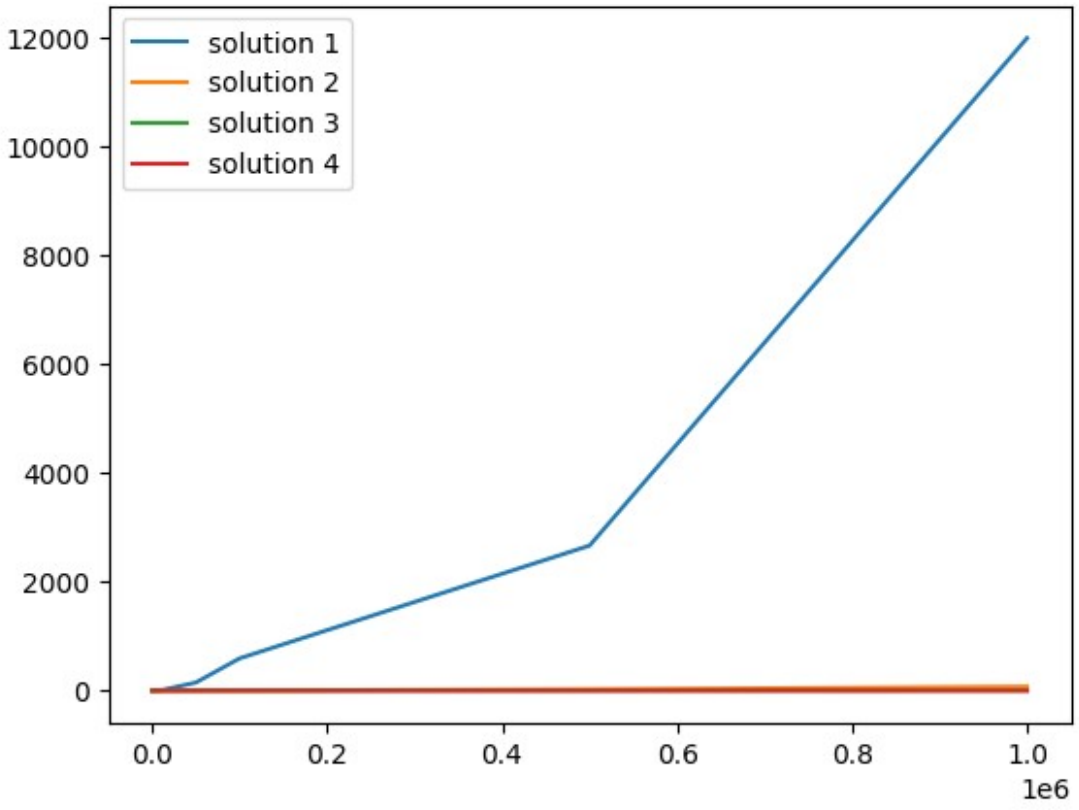
```
import matplotlib.pyplot as plt
import numpy as np

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

times = [[8.106231689453125e-06, 8.344650268554688e-06,
0.00012636184692382812, 0.0004737377166748047, 0.012262821197509766,
0.0542604923248291, 1.4507133960723877, 6.291901350021362,
148.18718767166138, 591.2566955089569, 4.5*591.2566955089569,
4.5*4.5*591.2566955089569], [9.059906005859375e-06,
8.821487426757812e-06, 5.245208740234375e-05, 0.0001227855682373047,
0.0010223388671875, 0.002592325210571289, 0.02563929557800293,
0.07202458381652832, 0.6946070194244385, 1.9673988819122314,
25.00427770614624, 73.84481120109558], [7.867813110351562e-06,
7.390759521484375e-06, 2.8848648071289062e-05, 6.079673767089844e-05,
0.0003972053527832031, 0.0008769035339355469, 0.005219936370849609,
0.012183427810668945, 0.05977797508239746, 0.12654423713684082,
0.7922322750091553, 1.867142915725708], [5.245208740234375e-06,
5.7220458984375e-06, 1.6450881958007812e-05, 2.8133392333984375e-05,
0.00014829635620117188, 0.0003020763397216797, 0.0015347003936767578,
```

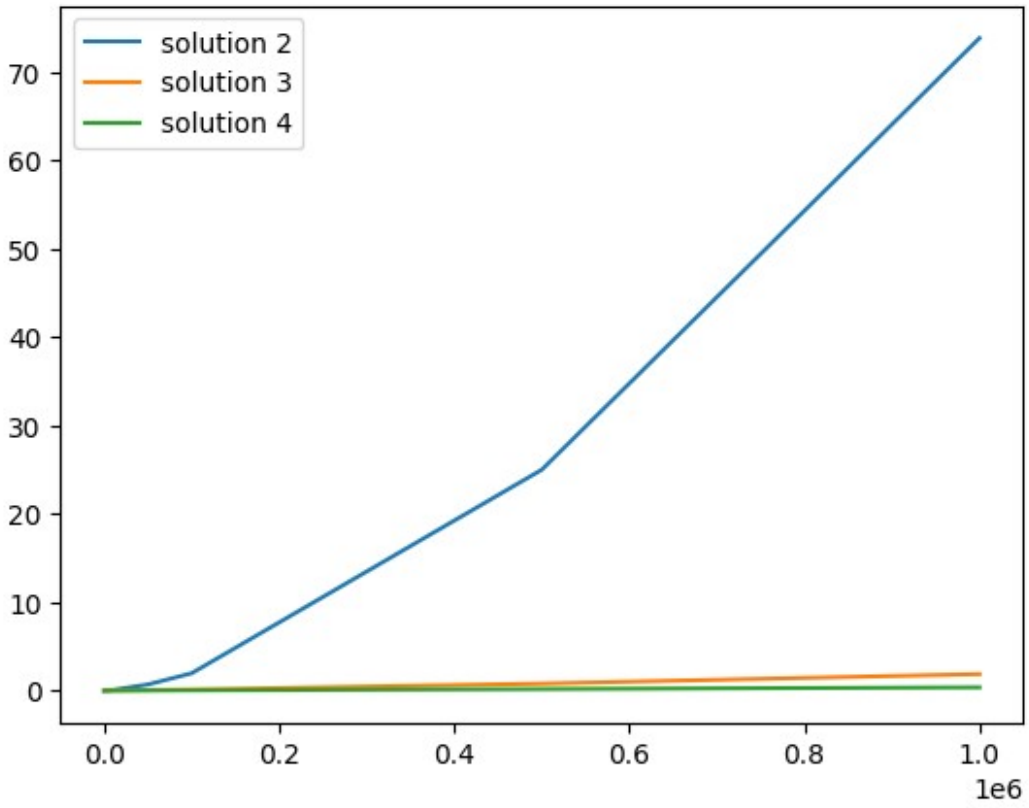
```
0.0031065940856933594, 0.015790224075317383, 0.03180360794067383,
0.17420434951782227, 0.36737847328186035]]
for i in range(len(times)):
    for j in range(len(times[i])):
        times[i][j] = float("{:.5f}".format(times[i][j]))

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()
```

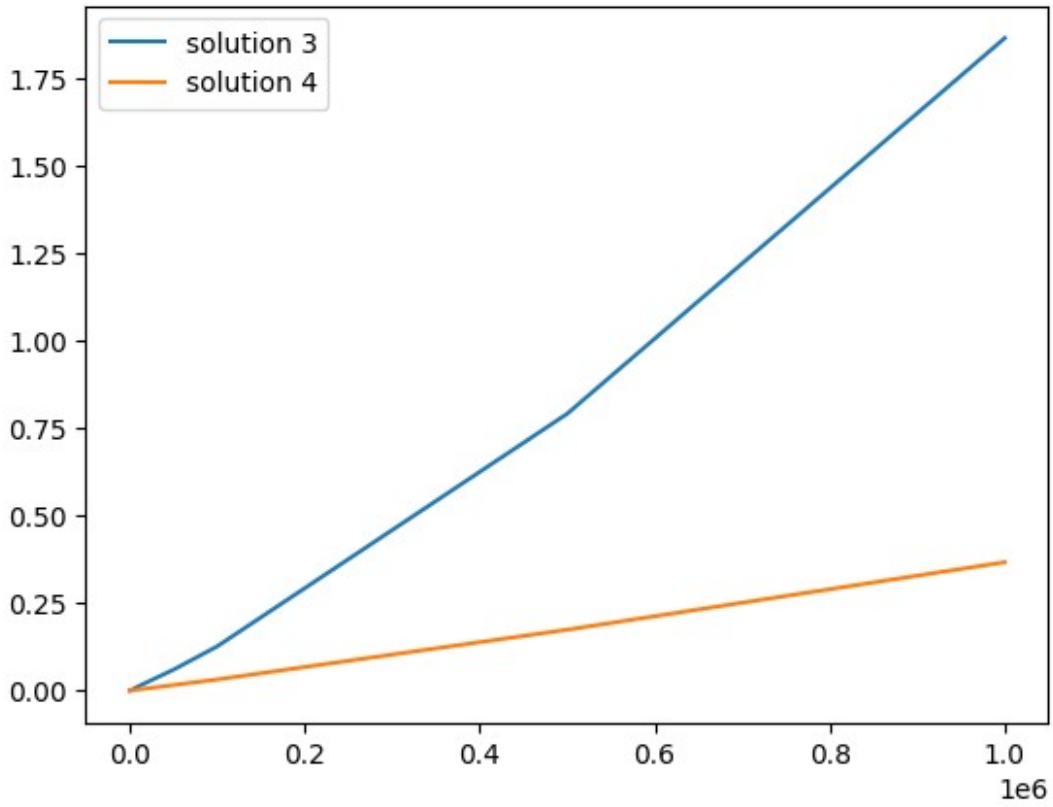
```
    drawAllSingleGraphs()


drawStuff()
```

Summary

	5	10	50	100	500	1000	5000	10000	50000	100000	500000	1000000
Solution 1	1e-05	1e-05	0.00013	0.00047	0.01226	0.05426	1.45071	6.2919	148.18719	591.2567	2660.65513	11972.94808
Solution 2	1e-05	1e-05	5e-05	0.00012	0.00102	0.00259	0.02564	0.07202	0.69461	1.9674	25.00428	73.84481
Solution 3	1e-05	1e-05	3e-05	6e-05	0.0004	0.00088	0.00522	0.01218	0.05978	0.12654	0.79223	1.86714
Solution 4	1e-05	1e-05	2e-05	3e-05	0.00015	0.0003	0.00153	0.00311	0.01579	0.0318	0.1742	0.36738

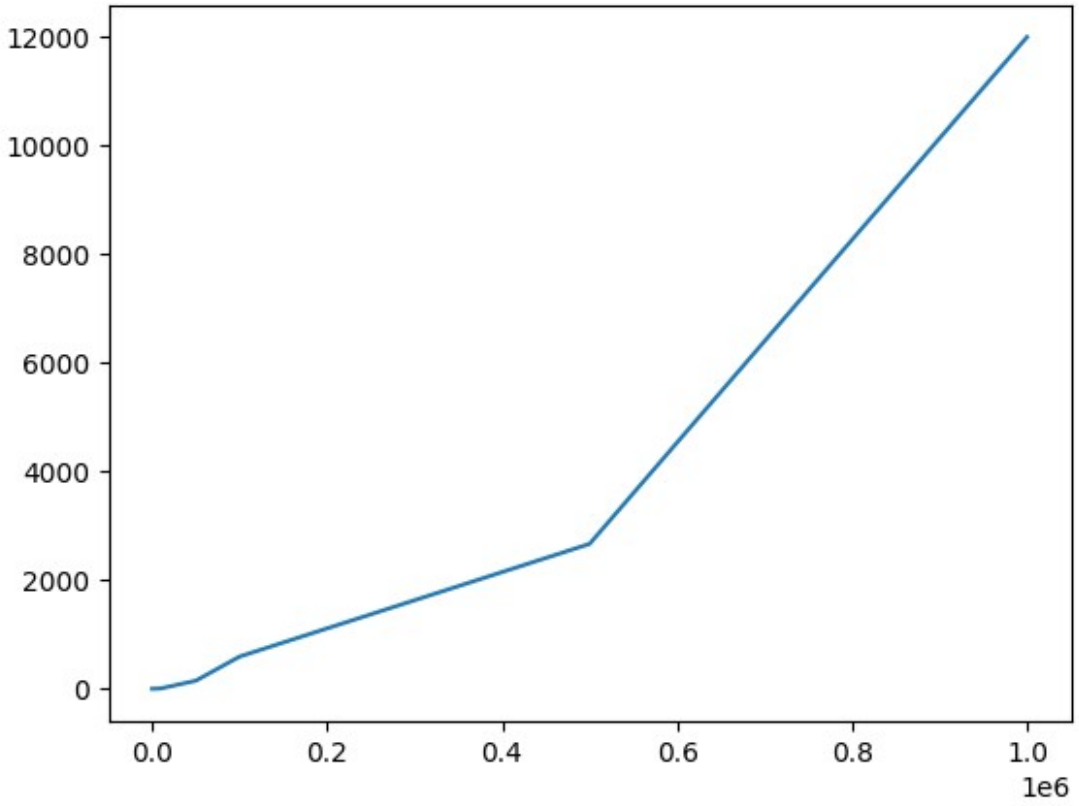



solution 2
solution 3
solution 4
70
60
50
40
30
20
10
0
0.0
0.2
0.4
0.6
0.8
1.0
1e6


solution 3
solution 4
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0.0
0.2
0.4
0.6
0.8
1.0
1e6

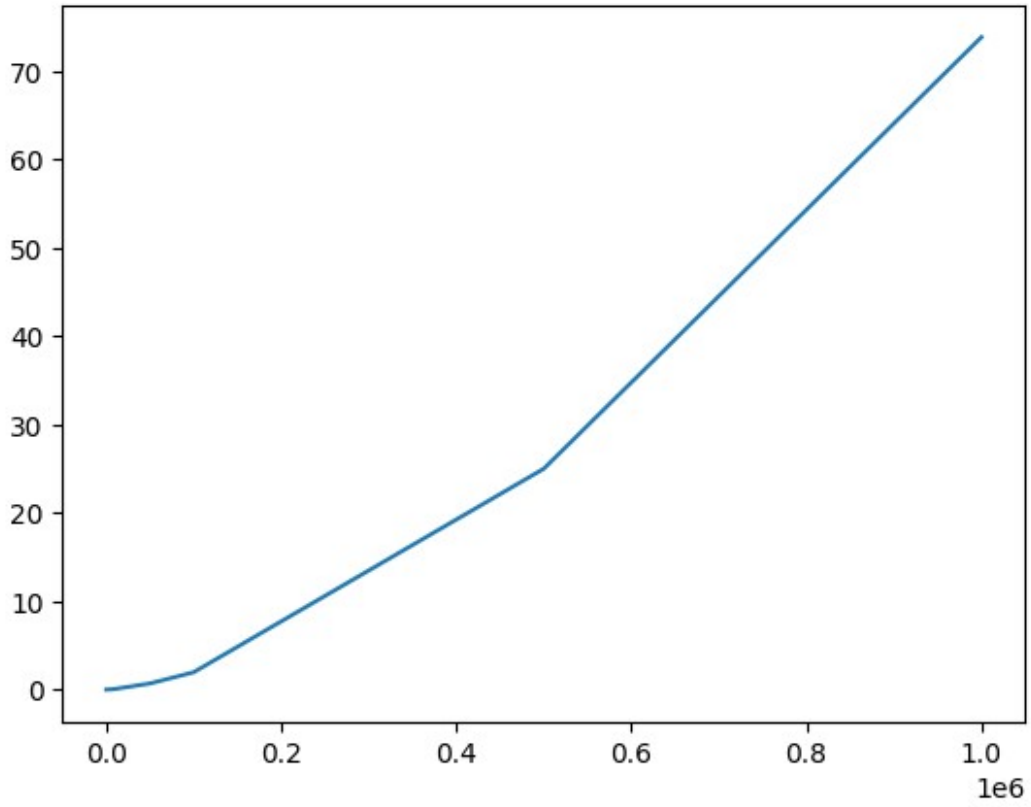
Solution 1

	5	10	50	100	500	1000	5000	10000	50000	100000	500000	1000000
Time	1e-05	1e-05	0.00013	0.00047	0.01226	0.05426	1.45071	6.2919	148.18719	591.2567	2660.65513	11972.94808



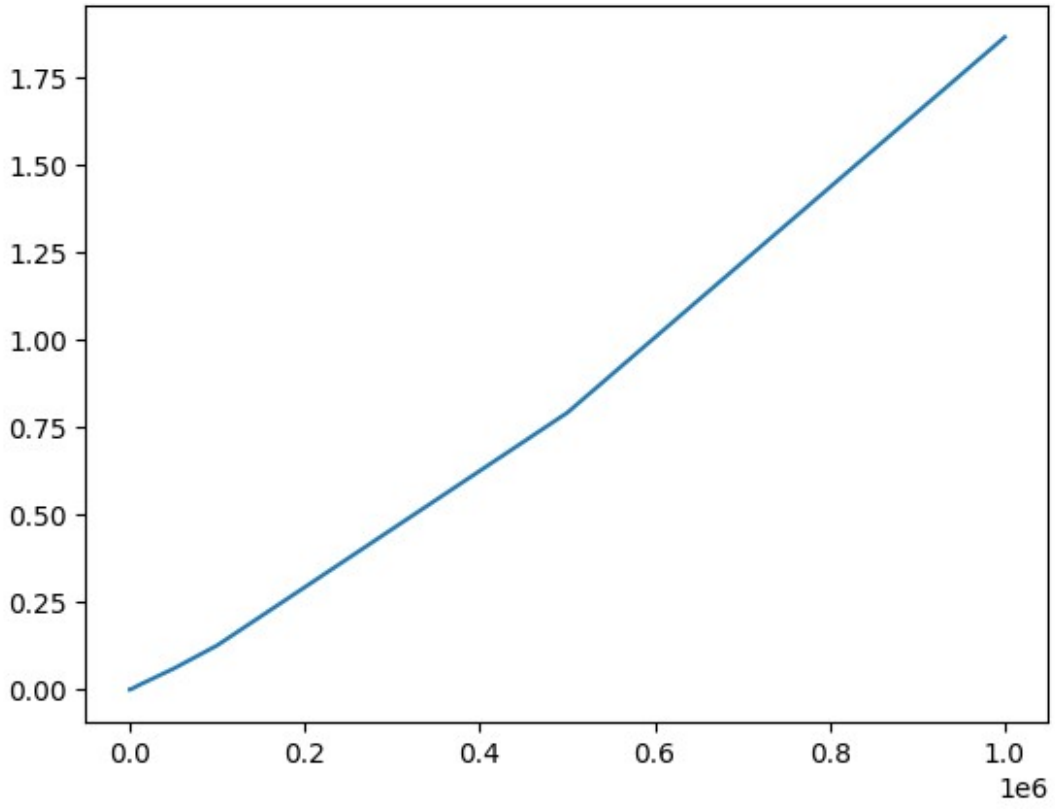
Solution 2

	5	10	50	100	500	1000	5000	10000	50000	100000	500000	1000000
Time	1e-05	1e-05	5e-05	0.00012	0.00102	0.00259	0.02564	0.07202	0.69461	1.9674	25.00428	73.84481



Solution 3

	5	10	50	100	500	1000	5000	10000	50000	100000	500000	1000000
Time	1e-05	1e-05	3e-05	6e-05	0.0004	0.00088	0.00522	0.01218	0.05978	0.12654	0.79223	1.86714



Solution 4

	5	10	50	100	500	1000	5000	10000	50000	100000	500000	1000000
Time	1e-05	1e-05	2e-05	3e-05	0.00015	0.0003	0.00153	0.00311	0.01579	0.0318	0.1742	0.36738