PRAKTIKUM ALGORITMA STRUKTUR DATA

MODUL 10

ANALISIS ALGORITMA

Disusun oleh:

Adinda Aulia Hapsari

L200220037

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS KOMUNIKASI DAN INFORMATIKA

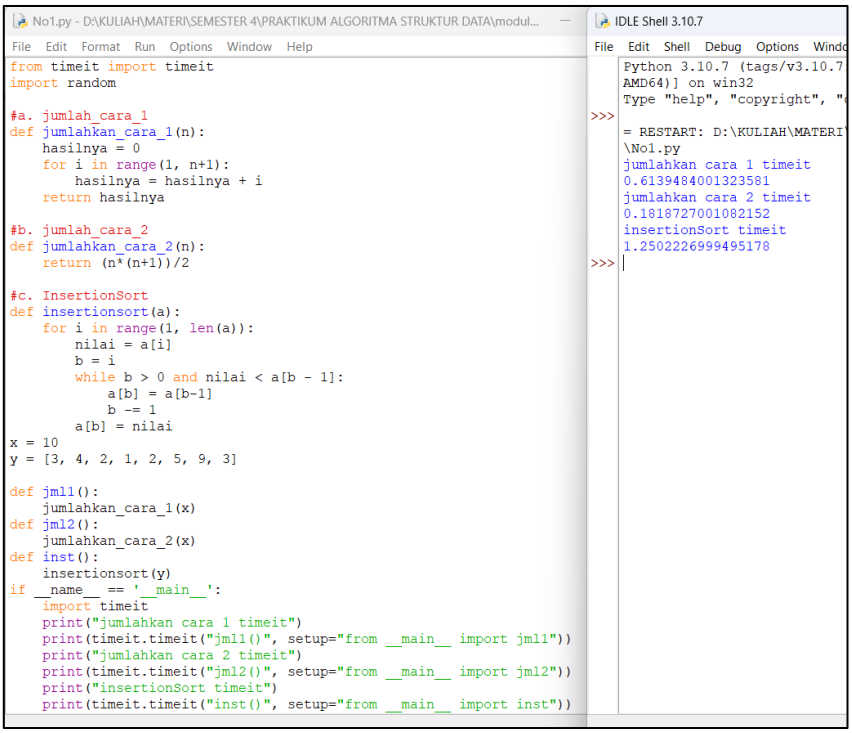
UNIVERSITAS MUHAMMADIYAH SURAKARTA

TAHUN 2024

Setelah kegiatan selesai, lembar kerja ini dicetak (di-print) dan dikumpulkan ke asisten. NIM : L200220037 Nama : Adinda Aulia Hapsari Nama Asisten : Tanggal Praktikum :	(Diisi oleh Asisten) Nilai Praktek : Tanda Tangan :

1. Kerjakan ulang contoh dan Latihan di modul ini menggunakan timeit, yaitu
   a. Jumlahkan_cara_1
   b. Jumlahkan_cara_2
   c. insertionShort

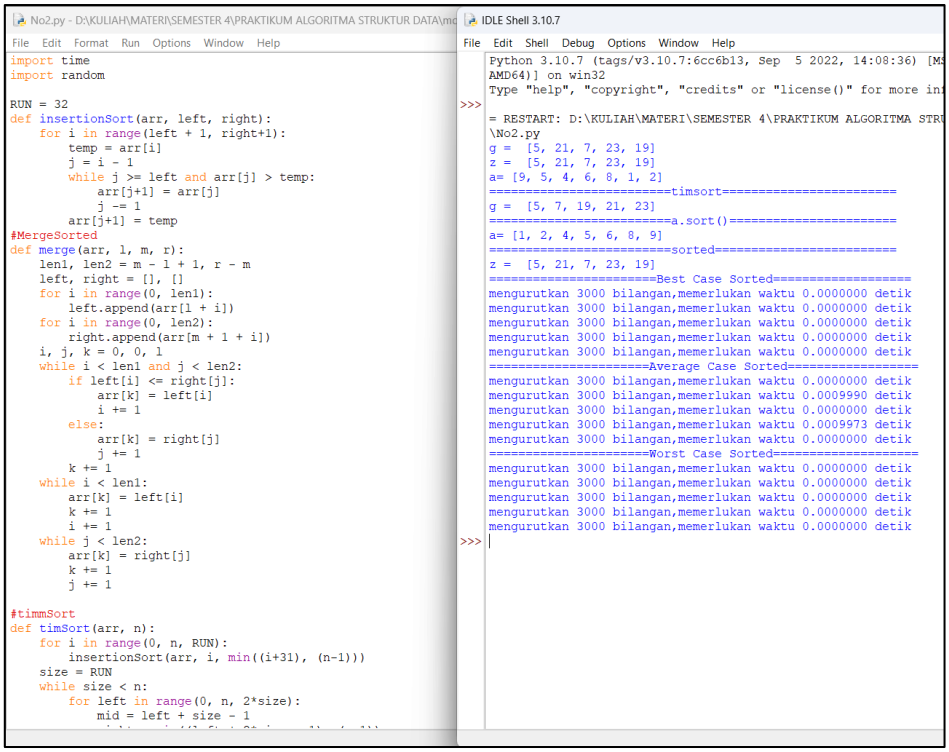
   untuk insertionShort, kerjakan untuk ketiga kasusnya.



```
from timeit import timeit
import random

#a. jumlah_cara_1
def jumlahkan_cara_1(n):
    hasilnya = 0
    for i in range(1, n+1):
        hasilnya = hasilnya + i
    return hasilnya

#b. jumlah_cara_2
def jumlahkan_cara_2(n):
    return (n*(n+1))/2

#c. InsertionSort
def insertionsort(a):
    for i in range(1, len(a)):
        nilai = a[i]
        b = i
        while b > 0 and nilai < a[b - 1]:
            a[b] = a[b-1]
            b -= 1
        a[b] = nilai
x = 10
y = [3, 4, 2, 1, 2, 5, 9, 3]

def jml1():
    jumlahkan_cara_1(x)
def jml2():
    jumlahkan_cara_2(x)
def inst():
    insertionsort(y)
if __name__ == '__main__':
    import timeit
    print("jumlahkan cara 1 timeit")
    print(timeit.timeit("jml1()", setup="from __main__ import jml1"))
    print("jumlahkan cara 2 timeit")
    print(timeit.timeit("jml2()", setup="from __main__ import jml2"))
    print("insertionSort timeit")
    print(timeit.timeit("inst()", setup="from __main__ import inst"))
```

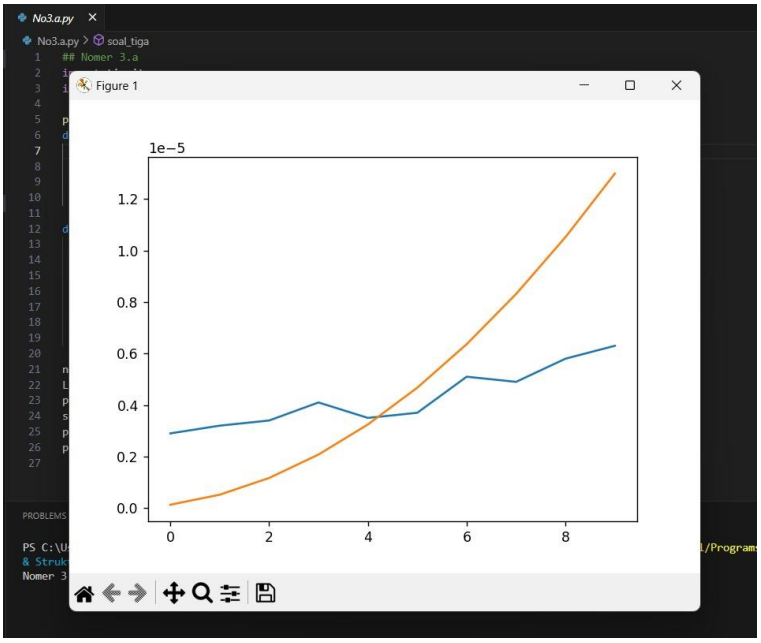
```
Python 3.10.7 (tags/v3.10.7
AMD64)] on win32
Type "help", "copyright", "
>>>
= RESTART: D:\KULIAH\MATERI
\No1.py
jumlahkan cara 1 timeit
0.6139484001323581
jumlahkan cara 2 timeit
0.1818727001082152
insertionSort timeit
1.2502226999495178
>>>
```

2. Python mempunyai perintah untuk mengurutkan suatu list yang memanfaatkan algoritma Timsort. Jika g adalah suatu list berisi bilangan, maka g.sort() kan mengurutkannya. Perintah yang lain, sorted() mengurutkan list dan mengembalikan sebuah list baru yang sudah urut. Selidikilah fungsi sorted ini menggunakan timeit:
   a. Apakah yang merupakan best case dan average case bagi sorted()?
   b. Confirm bahwa data input urutan terbalik bukan kasus terburuk bagi sorted(). Bahkan dia lebih cepat dalam mengurutkannya dari pada data input random.



3. a. kode

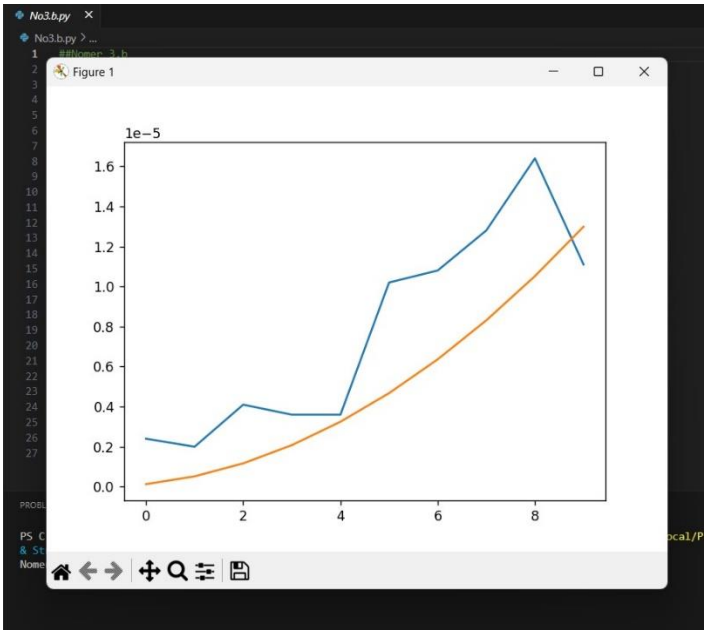
```
## Nomer 3.a
import timeit
import matplotlib.pyplot as plt

print("Nomer 3.a")
def soal_tiga(n):
    test = 0
    for i in range(n):
        for j in range(n):
            test = test + i * j

def uji_soal_tiga(n):
    ls = []
    jangkauan = range(1, n+1)
    siap = "from __main__ import soal_tiga"
    for i in jangkauan:
        t = timeit.timeit("soal_tiga(" + str(i) + ")", setup=siap, number=1)
        ls.append(t)
    return ls

n = 10
LS = uji_soal_tiga(n)
plt.plot(LS)
skala = 7700000
plt.plot([x*x/skala for x in range(1, n+1)])
plt.show()
```



b. kode

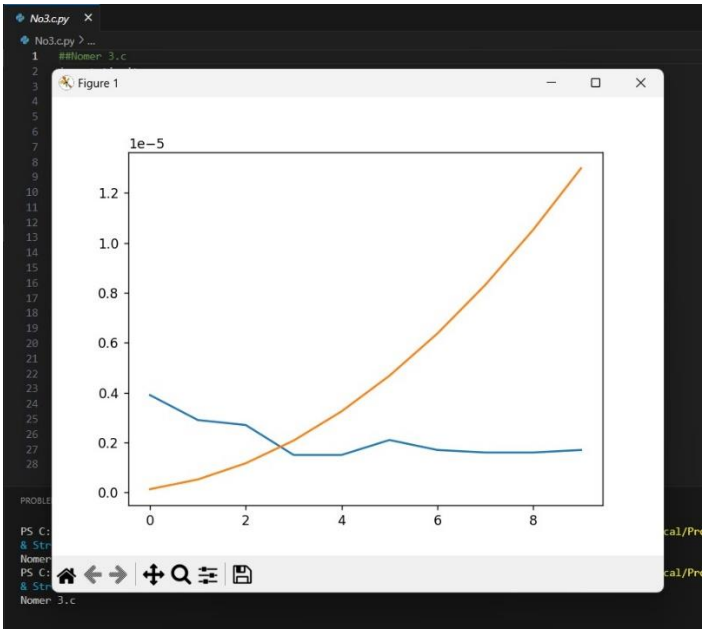
```
##Nomer 3.b
import timeit
import matplotlib.pyplot as plt

print("Nomer 3.b")
def soal_tiga(n):
    test = 0
    for i in range(n):
        for j in range(i):
            test = test + i * j

def uji_soal_tiga(n):
    ls = []
    jangkauan = range(1, n+1)
    siap = "from __main__ import soal_tiga"
    for i in jangkauan:
        t = timeit.timeit("soal_tiga(" + str(i) + ")", setup=siap,
number=1)
        ls.append(t)
    return ls

n = 10
LS = uji_soal_tiga(n)
plt.plot(LS)
skala = 7700000
plt.plot([x*x/skala for x in range(1, n+1)])
plt.show()
```

c.



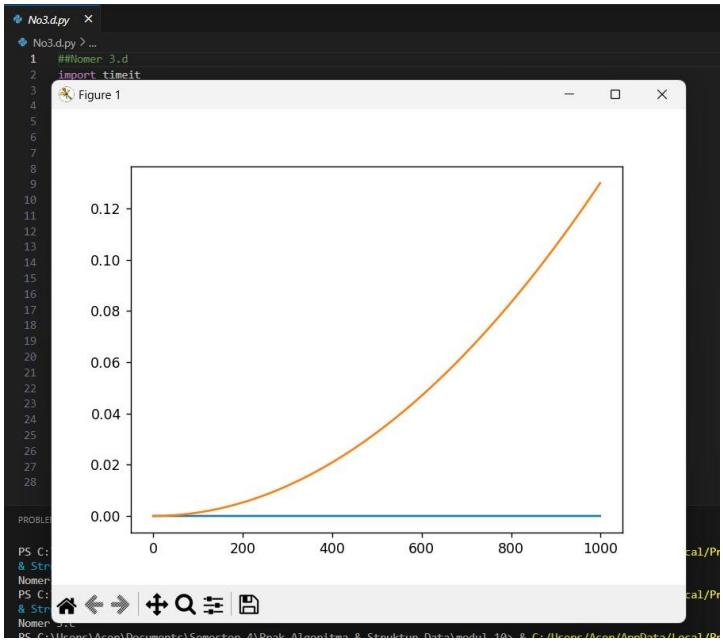
c. Kode

```
##Nomer 3.c
import timeit
import matplotlib.pyplot as plt

print("Nomer 3.c")
def soal_tiga(n):
    test = 0
    for i in range(n):
        test = test+1
    for j in range(n):
        test = test - 1

def uji_soal_tiga(n):
    ls = []
    jangkauan = range(1, n+1)
    siap = "from __main__ import soal_tiga"
    for i in jangkauan:
        t = timeit.timeit("soal_tiga(" + str(i) + ")", setup=siap,
number=1)
        ls.append(t)
    return ls

n = 10
LS = uji_soal_tiga(n)
plt.plot(LS)
skala = 7700000
plt.plot([x*x/skala for x in range(1, n+1)])
plt.show()
```



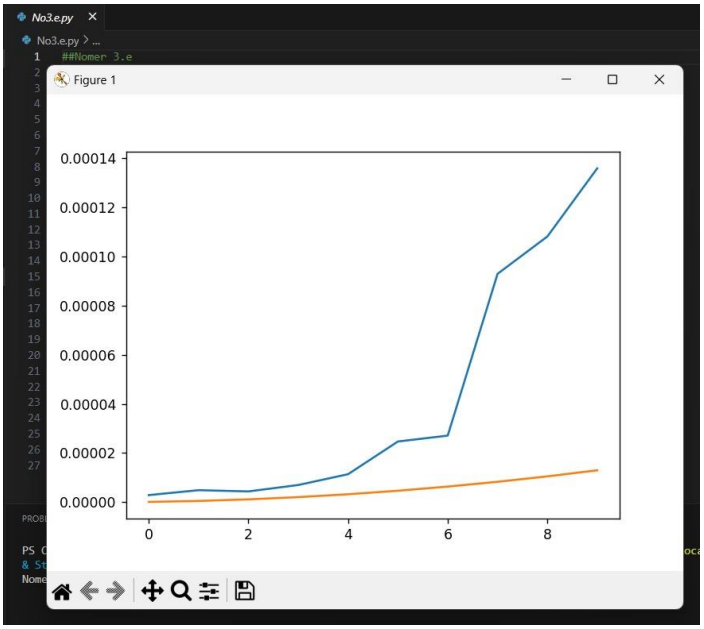
d. Kode

```
##Nomer 3.d
import timeit
import matplotlib.pyplot as plt

print("Nomer 3.d")
def soal_tiga(n):
    i = n
    while i > 0:
        k = 2 + 2
        i = i // 2

def uji_soal_tiga(n):
    ls = []
    jangkauan = range(1, n+1)
    siap = "from __main__ import soal_tiga"
    for i in jangkauan:
        ##print('i = ',i)
        t = timeit.timeit("soal_tiga(" + str(i) + ")", setup=siap,
number=1)
        ls.append(t)
    return ls

n = 1000
LS = uji_soal_tiga(n)
plt.plot(LS)
skala = 7700000
plt.plot([x*x/skala for x in range(1, n+1)])
plt.show()
```



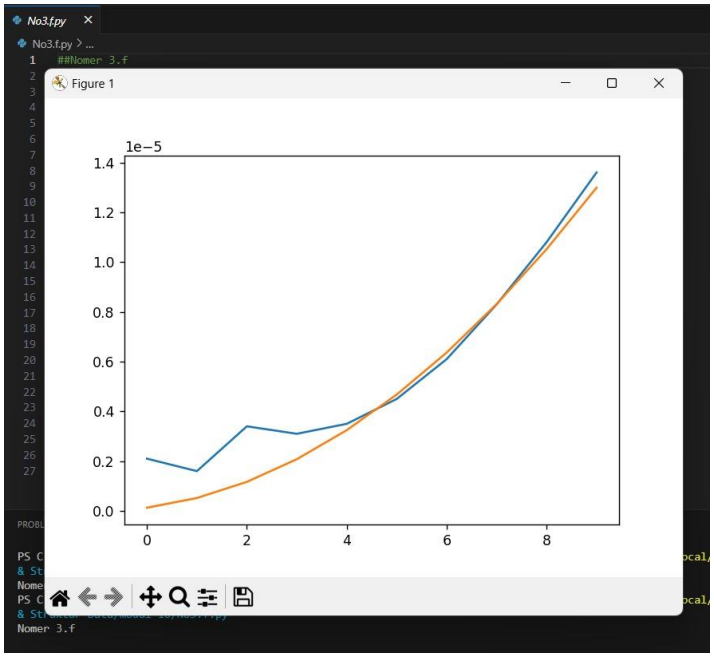
e. Kode

```
##Nomer 3.e
import timeit
import matplotlib.pyplot as plt

print("Nomer3.e")
def soal_tiga(n):
    for i in range(n):
        for j in range(n):
            for k in range(n):
                m = i + j + k + 2019

def uji_soal_tiga(n):
    ls = []
    jangkauan = range(1, n+1)
    siap = "from __main__ import soal_tiga"
    for i in jangkauan:
        t = timeit.timeit("soal_tiga(" + str(i) + ")", setup=siap,
number=1)
        ls.append(t)
    return ls

n = 10
LS = uji_soal_tiga(n)
plt.plot(LS)
skala = 7700000
plt.plot([x*x/skala for x in range(1, n+1)])
plt.show()
```



f. kode

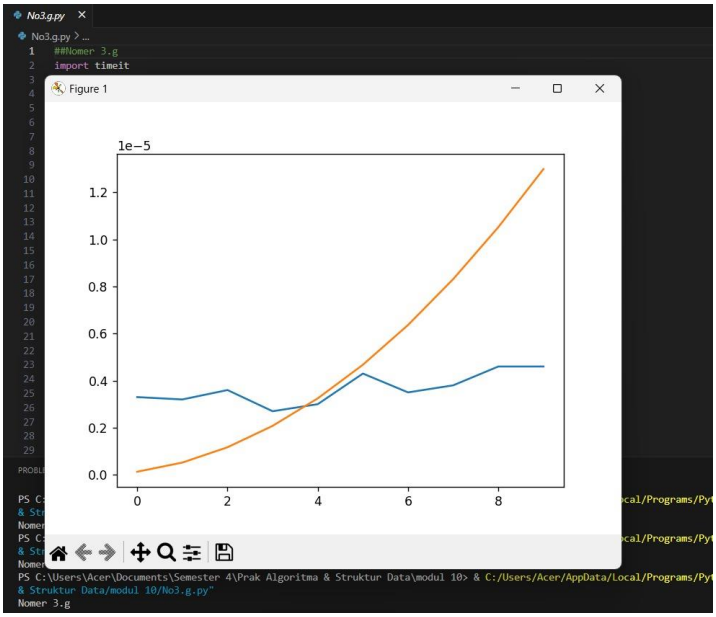
```
##Nomer 3.f
import timeit
import matplotlib.pyplot as plt

print("Nomer 3.f")
def soal_tiga(n):
    for i in range(n):
        for j in range(i):
            for k in range(j):
                m = i + j + k + 2019

def uji_soal_tiga(n):
    ls = []
    jangkauan = range(1, n+1)
    siap = "from __main__ import soal_tiga"
    for i in jangkauan:
        t = timeit.timeit("soal_tiga(" + str(i) + ")", setup=siap,
number=1)
        ls.append(t)
    return ls

n = 10
LS = uji_soal_tiga(n)
plt.plot(LS)
skala = 7700000
plt.plot([x*x/skala for x in range(1, n+1)])
plt.show()
```



g. kode

```
##Nomer 3.g
import timeit
import matplotlib.pyplot as plt

print("Nomer 3.g")
def soal_tiga(n):
    for i in range(n):
        if i % 3 == 0:
            for j in range(n // 2):
                j+=j
        elif i % 2 == 0:
            for j in range(5):
                j+=j
        else:
            for j in range(n):
                j+=j

def uji_soal_tiga(n):
    ls = []
    jangkauan = range(1, n+1)
    siap = "from __main__ import soal_tiga"
    for i in jangkauan:
        t = timeit.timeit("soal_tiga(" + str(i) + ")", setup=siap,
number=1)
        ls.append(t)
    return ls

n = 10
LS = uji_soal_tiga(n)
plt.plot(LS)
```

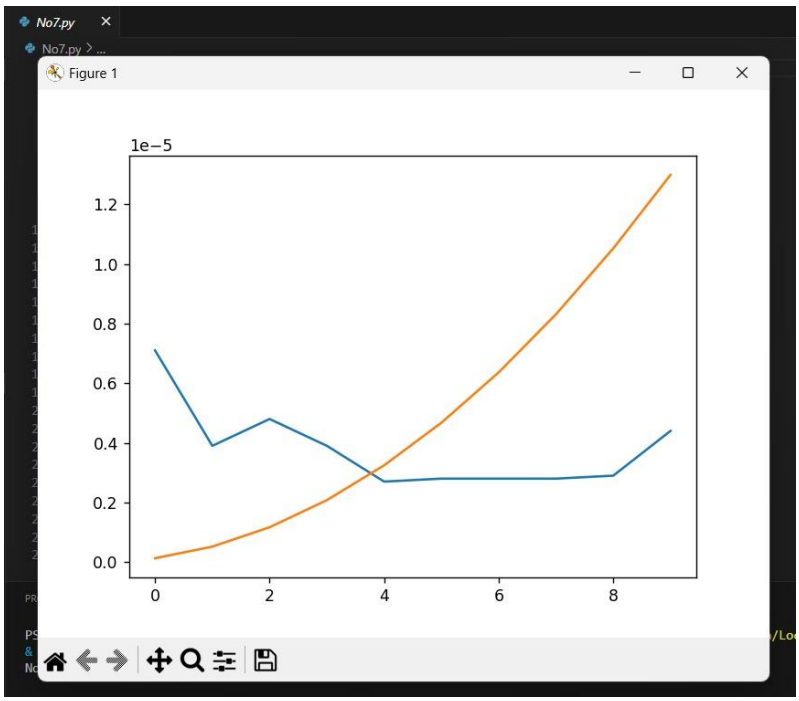
```
skala = 7700000
plt.plot([x*x/skala for x in range(1, n+1)])
plt.show()
```



7. Buatlah suatu uji coba untuk mengkonfirmasi bahwa metode append() adalah O(1). Gu nakan timeit dan matplotlib, seperti sebelumnya.

```
##Nomer 7
import timeit
import matplotlib.pyplot as plt

print("Nomer 7")
def soal_tujuh(n):
    L = list(range(30))
    L = L[::-1]
    for i in range(n):
        L.append(n)

def uji_soal_tujuh(n):
    ls = []
    jangkauan = range(1, n+1)
    siap = "from __main__ import soal_tujuh"
    for i in jangkauan:
        ##print('i = ',i)
        t = timeit.timeit("soal_tujuh(" + str(i) + ")", setup=siap, number=1)
        ls.append(t)
    return ls

n = 10
LS = uji_soal_tujuh(n)
plt.plot(LS)
skala = 7700000
```

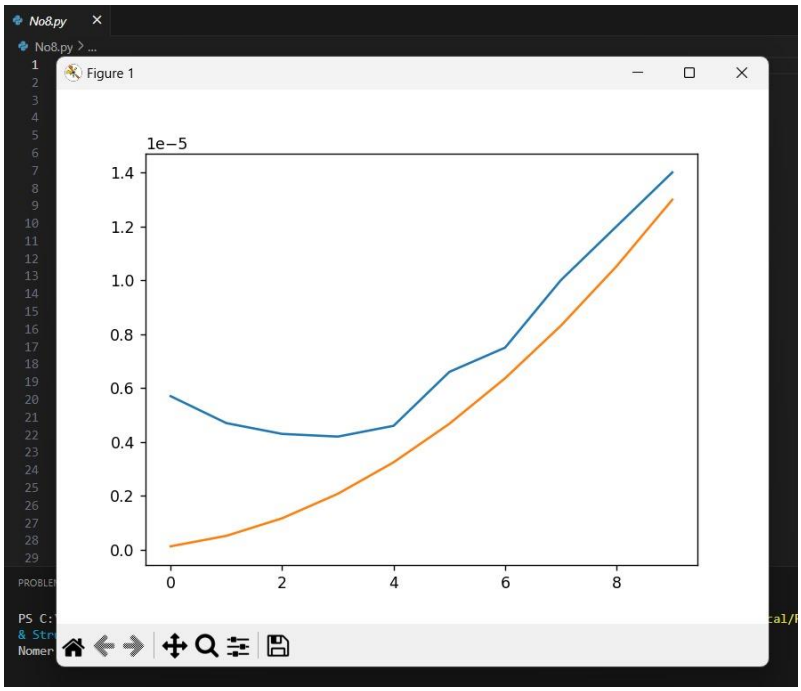
```
plt.plot([x*x/skala for x in range(1, n+1)])
plt.show()
```



8. Buatlah suatu uji coba untuk mengkonfirmasi bahwa metode insert() adalah O(n). Gu nakan timeit dan matplotlib, seperti sebelumnya.

```
##Nomer 8
import timeit
import matplotlib.pyplot as plt

print("Nomer 8")
def soal_tujuh(n):
    L = list(range(30))
    L = L[::-1]
    for i in range(n):
        for b in range(n):
            L.insert(i,b)

def uji_soal_tujuh(n):
    ls = []
    jangkauan = range(1, n+1)
    siap = "from __main__ import soal_tujuh"
    for i in jangkauan:
        ##print('i = ',i)
        t = timeit.timeit("soal_tujuh(" + str(i) + ")", setup=siap, number=1)
        ls.append(t)
    return ls

n = 10
```

```
LS = uji_soal_tujuh(n)
plt.plot(LS)
skala = 7700000
plt.plot([x*x/skala for x in range(1, n+1)])
plt.show()
```



9. Buatlah suatu uji coba untuk mengkonfirmasi bahwa untuk memeriksa apakah-suatu-nilai berada-di-suatu-list mempunyai kompleksitas O(n). Gunakan timeit dan matplotlib, seperti sebelumnya.

```
##Nomer 9
import timeit
import time

print("Nomer 9")
def carilurus(wadah, target):
    n = len(wadah)
    for i in range(n):
        if wadah[i] == target:
            return True
    return False
def tim():
    z=100
    a = [8, 7, 2, 1, 3, 2, 10]
    awal = time.time()
    U = carilurus(a, z)
    akhir=time.time()
    print("==========================worst case ")
```

```
    print("mengurutkan %d bilangan, memerlukan %8.7f detik" %(U,akhir-awal))
tim()

import matplotlib.pyplot as plt
def tim_matlib(n):
    z=100
    a = [8, 7, 2, 1, 3, 2, 10]
    U = carilurus(a, n)

def uji_soal_sembilan(n):
    ls = []
    jangkauan = range(1, n+1)
    siap = "from __main__ import tim_matlib"
    for i in jangkauan:
        t = timeit.timeit("tim_matlib(" + str(i) + ")", setup=siap, number=1)
        ls.append(t)
    return ls

n = 10
LS = uji_soal_sembilan(n)
plt.plot(LS)
skala = 7700000
plt.plot([x*x/skala for x in range(1, n+1)])
plt.show()
```