PRAKTIKUM ALGORITMA DAN STRUKTUR DATA

MODUL 10

ANALISIS ALGORITMA

Oleh : **Fariz Taufiqul Hafidz**

Nim : **L200210192**

Kelas : **F**

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS KOMUNIKASI DAN INFORMATIKA

UNIVERSITAS MUHAMMADIYAH SURAKARTA

2023

A. Soal-soal untuk Mahasiswa

1. Kerjakan ulang contoh dan latihan di modul ini menggunakan modul timeit, yakni

   (a) jumlahkan_cara_1 (b) jumlahkan_cara_2 (c) insertionSort

   Untuk insertionSort, kerjakan untuk ketiga kasusnya.

```python
from timeit import timeit

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

def jumlahkan_cara_2(n):
    return (n * (n + 1)) / 2

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

n = 10000
waktu_jml_cara_1 = timeit(lambda: jumlahkan_cara_1(n), number=1)
print("Waktu Jumlahkan Cara 1:", waktu_jml_cara_1)

n = 10000
waktu_jml_cara_2 = timeit(lambda: jumlahkan_cara_2(n), number=1)
print("Waktu Jumlahkan Cara 2:", waktu_jml_cara_2)

a = list(range(10000))
waktu_insertion_sort = timeit(lambda: insertionsort(a), number=1)
print("Waktu insertionSort:", waktu_insertion_sort)
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS E:\Kuliah\Semester 4\Praktikum ASD\Modul 10>  & 'C:\Users\A
Files\lib\python\debugpy\adapter/../..\debugpy\launcher' '4962
Waktu Jumlahkan Cara 1: 0.004276899999240413
Waktu Jumlahkan Cara 2: 0.000287900009425357
Waktu insertionSort: 0.022694499988574535
PS E:\Kuliah\Semester 4\Praktikum ASD\Modul 10>
```

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

```python
import time
import random


#Average Case Scenario
def averagecase():
    for i in range(5):
        L = list(range(90000))
        random.shuffle(L)
        awal = time.time()
        U = sorted(L)
        akhir = time.time()
        print("mengurutkan %d bilangan,memerlukan waktu %8.7f detik" % (len(L), akhir - awal))

#Worst Case Scenario
def worstcase():
    for i in range(5):
        L=list(range(90000))
        L=L[::-1]
        awal=time.time()
        U = sorted(L)
        akhir=time.time()
        print("mengurutkan %d bilangan,memerlukan waktu %8.7f detik" %(len(L), akhir - awal))

#Best Case Scenario
def bestcase():
    for i in range(5):
        L = list(range(90000))
        awal = time.time()
        U = sorted(L)
        akhir = time.time()
        print("mengurutkan %d bilangan,memerlukan waktu %8.7f detik" %(len(L), akhir - awal))



#Best Case Sorted g
print("========== Waktu Bestcase Sorted  ==========")
bestcase()

#Average Case Sorted g
print("========== Waktu Average Sorted  ==========")
averagecase()

#Worst Case Sorted g
print("========== Worst Average Sorted  ==========")
worstcase()

```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
PS E:\Kuliah\Semester 4\Praktikum ASD\Modul 10>  & 'C:\Users\Asus\AppData\Loca
Files\lib\python\debugpy\adapter/../..\debugpy\launcher' '49643' '--' 'E:\Kuli
========== Waktu Bestcase Sorted  ==========
mengurutkan 90000 bilangan,memerlukan waktu 0.0019953 detik
mengurutkan 90000 bilangan,memerlukan waktu 0.0029938 detik
mengurutkan 90000 bilangan,memerlukan waktu 0.0029922 detik
mengurutkan 90000 bilangan,memerlukan waktu 0.0029912 detik
mengurutkan 90000 bilangan,memerlukan waktu 0.0029922 detik
========== Waktu Average Sorted  ==========
mengurutkan 90000 bilangan,memerlukan waktu 0.0678194 detik
mengurutkan 90000 bilangan,memerlukan waktu 0.0269291 detik
mengurutkan 90000 bilangan,memerlukan waktu 0.0359056 detik
mengurutkan 90000 bilangan,memerlukan waktu 0.0528610 detik
mengurutkan 90000 bilangan,memerlukan waktu 0.0349078 detik
========== Worst Average Sorted  ==========
mengurutkan 90000 bilangan,memerlukan waktu 0.0029948 detik
mengurutkan 90000 bilangan,memerlukan waktu 0.0079789 detik
mengurutkan 90000 bilangan,memerlukan waktu 0.0029914 detik
mengurutkan 90000 bilangan,memerlukan waktu 0.0059872 detik
mengurutkan 90000 bilangan,memerlukan waktu 0.0049853 detik
PS E:\Kuliah\Semester 4\Praktikum ASD\Modul 10>
```

3. Untuk tiap kode berikut, tentukan running time-nya, O(1), O(log n), O(n), O(n log n), O(n2), atau O(n3), atau yang lain. Untuk memulai analisis, ambil suatu nilai n tertentu, lalu ikuti apa yang terjadi di kode itu.
   a) loop di dalam loop, keduanya sebanyak n:

      test = 0

      for i in range(n):

            for j in range(n):

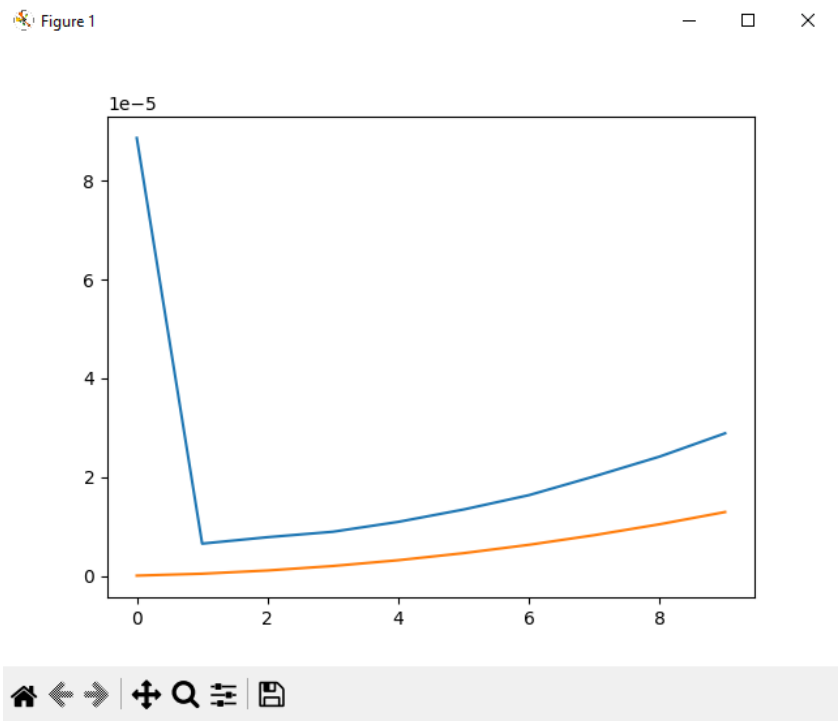
                  test = test + i * j

```python
## 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) loop di dalam loop, yang dalam bergantung nilai i loop luar:

test = 0

for i in range(n):

for j in range(i):

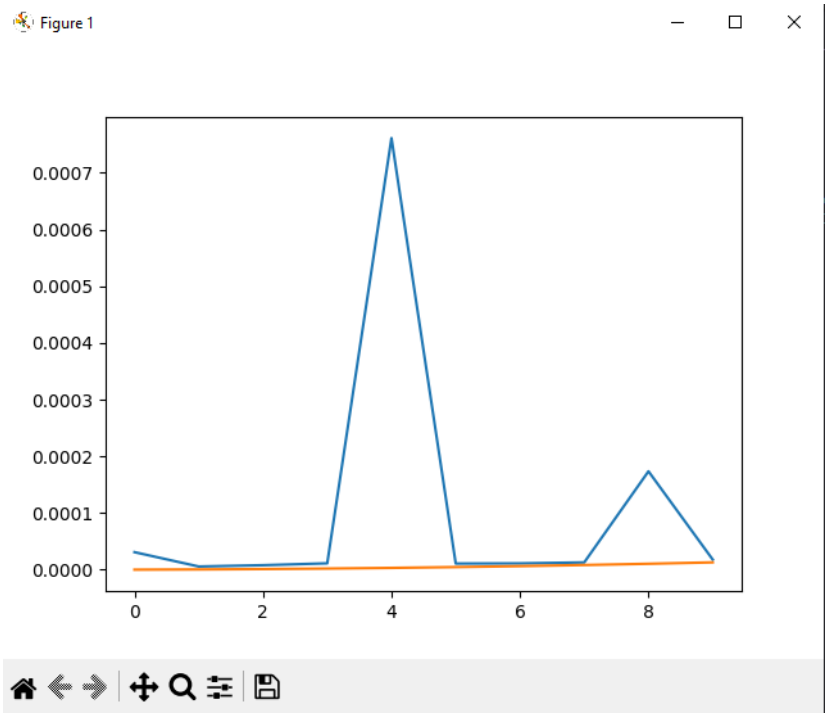
test = test + i * j

```
##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) dua loop terpisah:

test = 0

for i in range(n):

test = test + 1

for j in range(n):

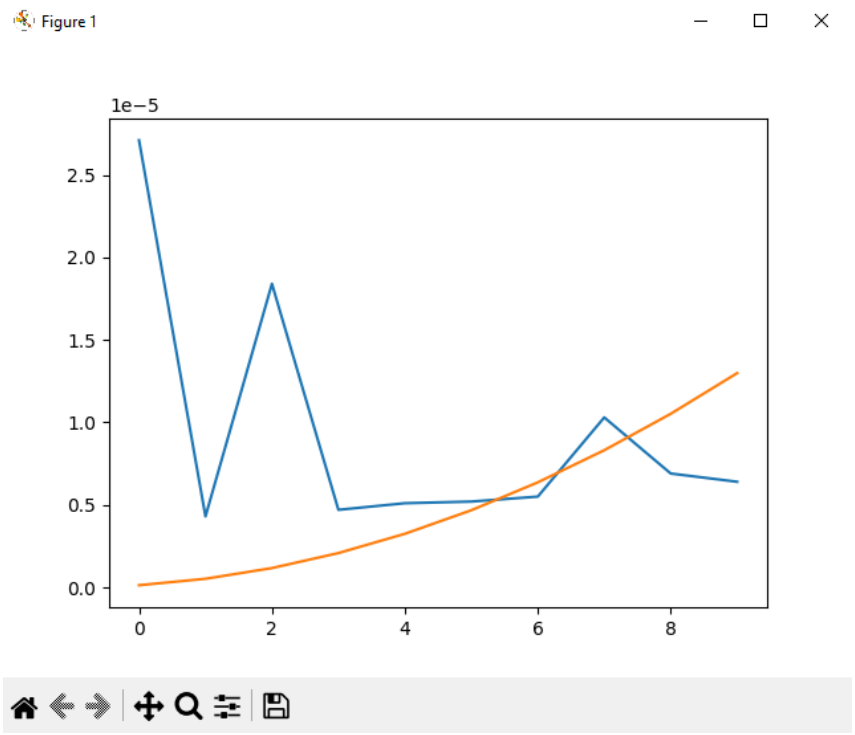
test = test – 1

```python
##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) while loop yang dipangkas separuh tiap putaran:

i=n

while i > 0:

k=2+2

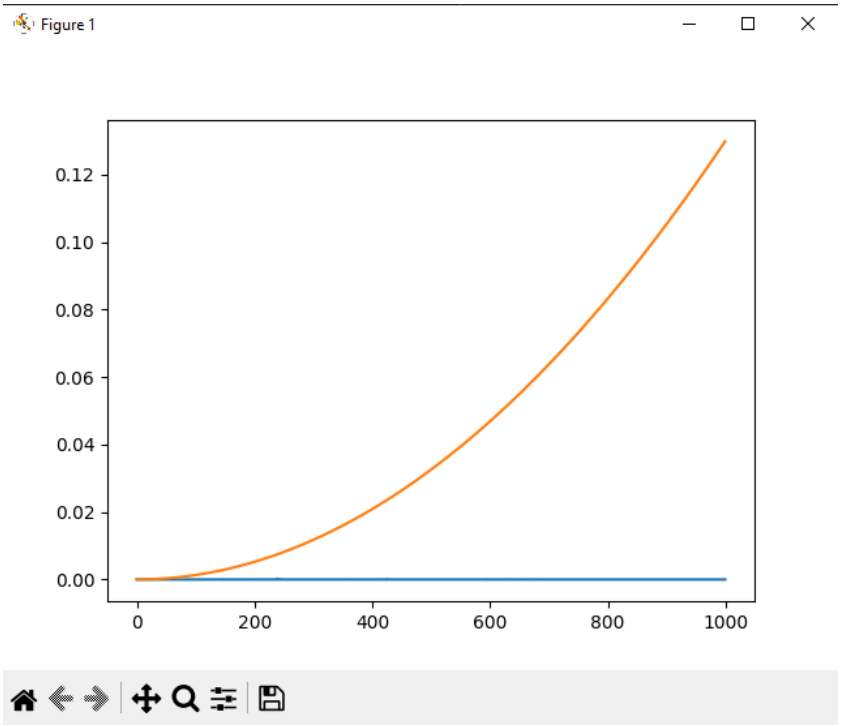
i = i // 2

```
##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) loop in a loop in a loop, ketiganya sebanyak n:

for i in range(n):

for j in range(n):

for k in range(n):

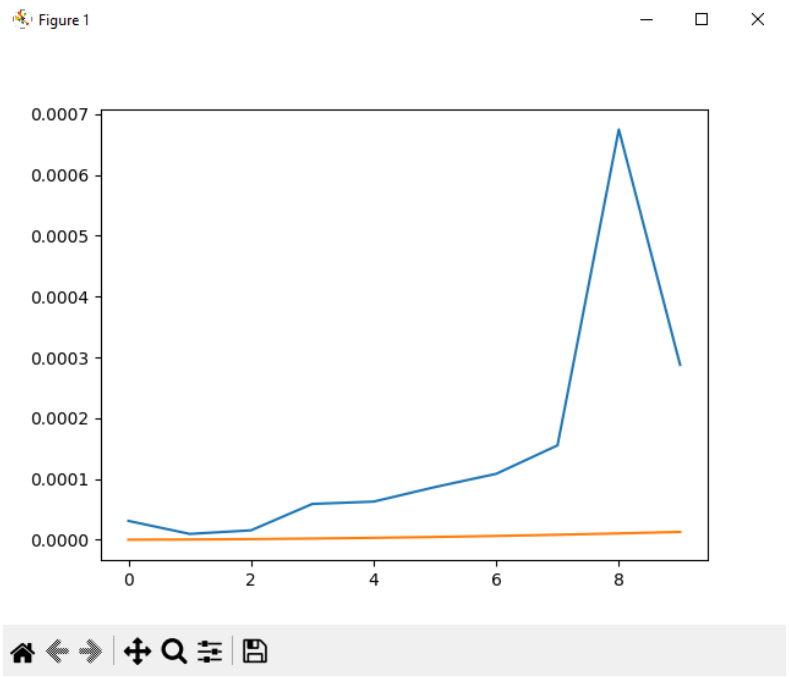
m = i + j + k + 2019

```
##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) loop in a loop in a loop, dengan loop dalam sebanyak nilai loop luar terdekat:

for i in range(n):

for j in range(i):

for k in range(j):

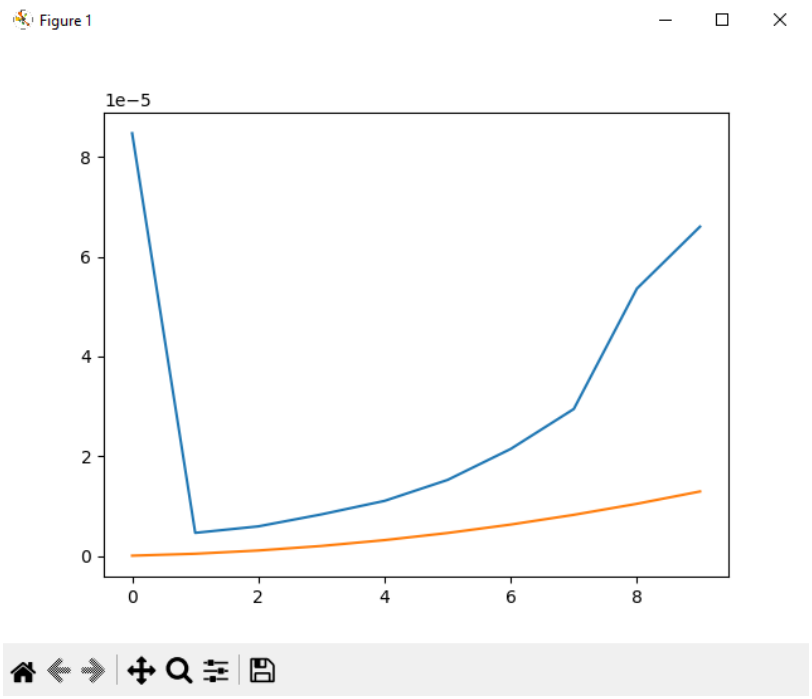
m = i + j + k + 2019

```python
##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) fungsi ini:

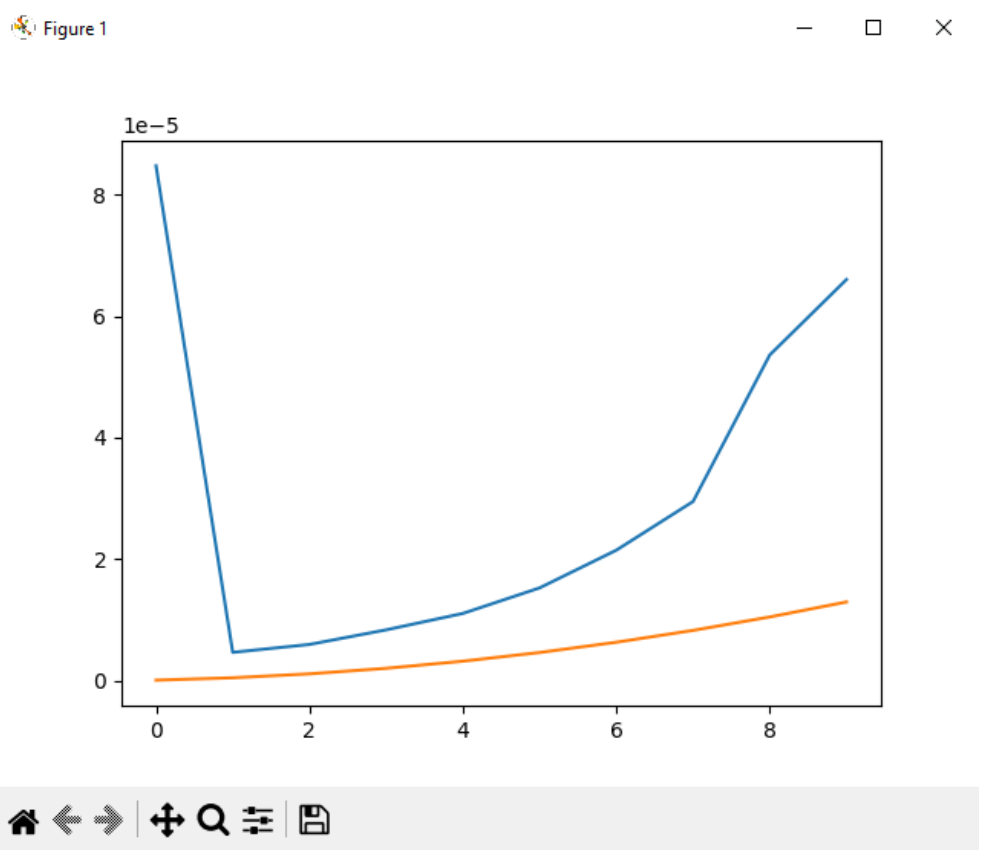
```
for i in range(n):
        if i % 3 == 0 :
                for j in range(n/2):
                        sum += j
        elif i % 2 == 0 :
                for j in range(5):
                        sum += j
        else :
                for j in range(n):
                        sum += j
```

```python
##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()
```



4. Urutkan dari yang pertumbuhan kompleksitasnya lambat ke yang cepat:

   $n \log_2 n$ 4^n $10 \log_2 n$ $5n^2$ $\log_4 n$ $12n^6$ $2^{\log_2 n}$ n^3

   Urutkan :

   $\log_4 n$ $2^{\log_2 n}$ $10 \log_2 n$ 4^n $n \log_2 n$ $5n^2$ n^3 $12n^6$

5. Tentukan O(·) dari fungsi-fungsi berikut, yang mewakili banyaknya langkah yang diperlukan untuk beberapa algoritma.

   a) $T(n) = n^2 + 32n + 8$ $= O(n^2)$

   b) $T(n) = 87n + 8n$ $= O(n)$

   c) $T(n) = 4n + 5n \log n + 102$ $= O(n \log n)$

   d) $T(n) = \log n + 3n^2 + 88$ $= O(n^2)$

   e) $T(n) = 3(2^n) + n^2 + 647$ $= O(n^2)$

   f) $T(n, k) = kn + \log k$ $= O(k^n)$

   g) $T(n, k) = 8n + k \log n + 800$ $= O(k \log n)$

   h) $T(n, k) = 100kn + n$ $= O(k^n)$