NAMA : KEVIN AVICENNA WIDIARTO
NIM : L200200183
Modul : 4

Praktikum Algoritma Struktur Data

MODUL 10

No 1a

E: > PRAK ASD > Modul 10 > no 1a.py > ...

```python
import time
from timeit import timeit

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

for i in range(5):
    a = 'from __main__ import jumlahkan_cara_1'
    j = jumlahkan_cara_1(1000000)
    t = timeit('jumlahkan_cara_1(2000000)', setup = a, number=1)
    print("Jumlah %d perlu %9.8f detik" % (j, t))

```

PROBLEMS OUTPUT DEBUG CONSOLE JUPYTER TERMINAL

```
aconda3\python.exe' 'c:\Users\kevin\.vscode\extensions\ms-python.python-2022
\no 1a.py'
Jumlah 500000500000 perlu 0.20396290 detik
Jumlah 500000500000 perlu 0.18358360 detik
Jumlah 500000500000 perlu 0.18660960 detik
Jumlah 500000500000 perlu 0.18267990 detik
Jumlah 500000500000 perlu 0.18941210 detik
```

No 1b

E: > PRAK ASD > Modul 10 > no 1b.py > jumlahkan_cara_2

```python
import time
from timeit import timeit

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

for i in range(5):
    a = 'from __main__ import jumlahkan_cara_2'
    j = jumlahkan_cara_2(1000000)
    t = timeit('jumlahkan_cara_2(1000000)', setup = a, number=1)
    print("Jumlah %d perlu %9.8f detik" % (j, t))

```

PROBLEMS OUTPUT DEBUG CONSOLE JUPYTER TERMINAL

```
pwsh E:\PRAK ASD\Modul 10
> e:; cd 'e:\PRAK ASD\Modul 10'; & 'C:\Users\kevin\anaconda3\python.exe'
hon\debugpy\launcher' '56424' '--' 'e:\PRAK ASD\Modul 10\no 1b.py'
Jumlah 500000500000 perlu 0.00000200 detik
Jumlah 500000500000 perlu 0.00000180 detik
Jumlah 500000500000 perlu 0.00000170 detik
Jumlah 500000500000 perlu 0.00000170 detik
Jumlah 500000500000 perlu 0.00000170 detik
pwsh E:\PRAK ASD\Modul 10
>
```

No 1c

E: > PRAK ASD > Modul 10 > no 1c.py > insertionSort

```python
import random
from timeit import timeit

def insertionSort(A):
    n = len(A)
    for i in range(1, n):
        nilai = A[i]
        pos = i
        while pos > 0 and nilai < A[pos - 1]:
            A[pos] = A[pos - 1]
            pos = pos -1
        A[pos] = nilai


print("\nskenario kasus rata-rata")

for i in range(5):
    siap = 'from __main__ import insertionSort, L'
    L = list(range(2500))
    random.shuffle(L)
    t = timeit('insertionSort(L)', setup = siap, number=1)
    print("Mengurutkan %d bilangan perlu %8.7f detik" % (len(L), t))

print("\nskenario kasus terburuk")
for i in range(5):
    siap = 'from __main__ import insertionSort, L'
    L = list(range(2500))
    L = L[::-1]
    t = timeit('insertionSort(L)',setup = siap, number=1)
    print("Mengurutkan %d bilangan perlu %8.7f detik" % (len(L), t))

print("\nskenario kasus terbaik")
for i in range(5):
    siap = 'from __main__ import insertionSort, L'
    L = list(range(2500))
    t = timeit('insertionSort(L)', setup = siap, number=1)
    print("Mengurutkan %d bilangan perlu %8.7f detik" % (len(L), t))
```

```
pwsh E:\PRAK ASD\Modul 10
> e:; cd 'e:\PRAK ASD\Modul 10'; & 'C:\Users\kev
.8.0\pythonFiles\lib\python\debugpy\launcher' '564

skenario kasus rata-rata
Mengurutkan 2500 bilangan perlu 0.7082379 detik
Mengurutkan 2500 bilangan perlu 0.6862010 detik
Mengurutkan 2500 bilangan perlu 0.6935810 detik
Mengurutkan 2500 bilangan perlu 0.6857388 detik
Mengurutkan 2500 bilangan perlu 0.6956847 detik

skenario kasus terburuk
Mengurutkan 2500 bilangan perlu 1.3602674 detik
Mengurutkan 2500 bilangan perlu 1.3650545 detik
Mengurutkan 2500 bilangan perlu 1.3492809 detik
Mengurutkan 2500 bilangan perlu 1.3685161 detik
Mengurutkan 2500 bilangan perlu 1.3832764 detik

skenario kasus terbaik
Mengurutkan 2500 bilangan perlu 0.0008447 detik
Mengurutkan 2500 bilangan perlu 0.0008538 detik
Mengurutkan 2500 bilangan perlu 0.0008621 detik
Mengurutkan 2500 bilangan perlu 0.0008724 detik
Mengurutkan 2500 bilangan perlu 0.0008720 detik
pwsh E:\PRAK ASD\Modul 10
>
```

No 2

> PRAK ASD > Modul 10 > no 2.py > ...

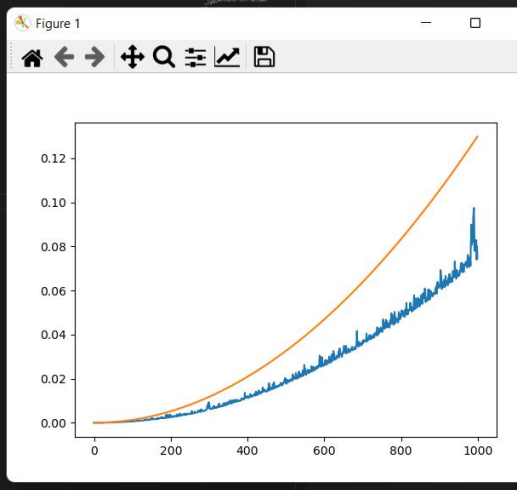
```python
from timeit import timeit
import random

print("\nskenario kasus rata-rata")
for i in range(5):
    g = list(range(2500))
    random.shuffle(g)
    t = timeit('sorted(g)', setup = 'from __main__ import g', number=1)
    print("Mengurutkan %d bilangan perlu %8.7f detik" % (len(g), t))

print("\nskenario kasus terburuk")
for i in range(5):
    g = list(range(2500))
    g = g[::-1]
    t = timeit('sorted(g)', setup = 'from __main__ import g', number=1)
    print("Mengurutkan %d bilangan perlu %8.7f detik" % (len(g), t))

print("\nskenario kasus terbaik")
for i in range(5):
    g = list(range(2500))
    t = timeit('sorted(g)', setup = 'from __main__ import g', number=1)
    print("Mengurutkan %d bilangan perlu %8.7f detik" % (len(g), t))
```

```
pwsh ~
> & C:/Users/kevin/anaconda3/python.exe "e:/PRAK ASD/

skenario kasus rata-rata
Mengurutkan 2500 bilangan perlu 0.0002002 detik
Mengurutkan 2500 bilangan perlu 0.0001954 detik
Mengurutkan 2500 bilangan perlu 0.0002261 detik
Mengurutkan 2500 bilangan perlu 0.0001958 detik
Mengurutkan 2500 bilangan perlu 0.0002063 detik

skenario kasus terburuk
Mengurutkan 2500 bilangan perlu 0.0000156 detik
Mengurutkan 2500 bilangan perlu 0.0000152 detik
Mengurutkan 2500 bilangan perlu 0.0000150 detik
Mengurutkan 2500 bilangan perlu 0.0000150 detik
Mengurutkan 2500 bilangan perlu 0.0000149 detik

skenario kasus terbaik
Mengurutkan 2500 bilangan perlu 0.0000143 detik
Mengurutkan 2500 bilangan perlu 0.0000142 detik
Mengurutkan 2500 bilangan perlu 0.0000143 detik
Mengurutkan 2500 bilangan perlu 0.0000144 detik
Mengurutkan 2500 bilangan perlu 0.0000145 detik
pwsh ~
>
```

No 3a

E: > PRAK ASD > Modul 10 > no 3a.py > ujiKalangBersusuh

```python
import timeit
import matplotlib.pyplot as plt

def kalangBersusuh(n):
    test = 0
    for i in range(n):
        for j in range(n):
            test = test + i * j

def ujiKalangBersusuh(n):
    ls = []
    jangkauan = range(1, n+1)
    a = 'from __main__ import kalangBersusuh'
    for i in jangkauan:
        print('i = ',i)
        t = timeit.timeit('kalangBersusuh(' + str(i) +')', setup = a,number=1)
        ls.append(t)
    return ls

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



PROBLEMS OUTPUT DEBUG CONSOLE JUPYTER

TERMINAL

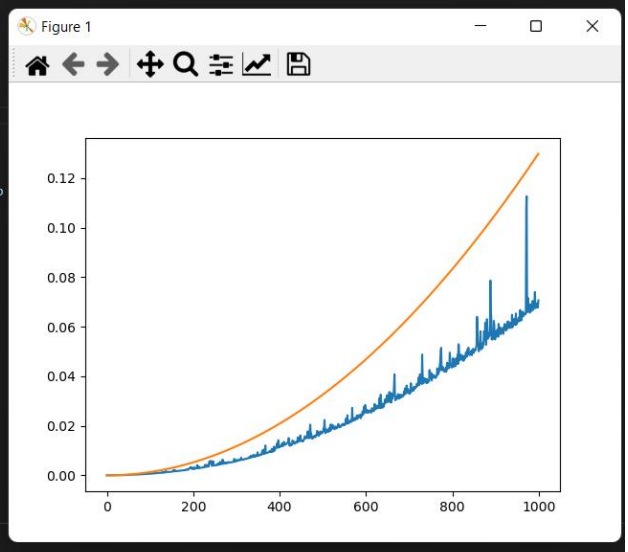
```
i =  992
i =  993
i =  994
i =  995
i =  996
i =  997
i =  998
i =  999
i =  1000
```

DEBUG CONSOLE

No 3b

E: > PRAK ASD > Modul 10 > no 3b.py > ujiKalangBersusuh

```python
import timeit
import matplotlib.pyplot as plt

def kalangBersusuh(n):
    test = 0
    for i in range(n):
        for j in range(i):
            test = test + i * j

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

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



PROBLEMS OUTPUT DEBUG CONSOLE JUPYTER

TERMINAL

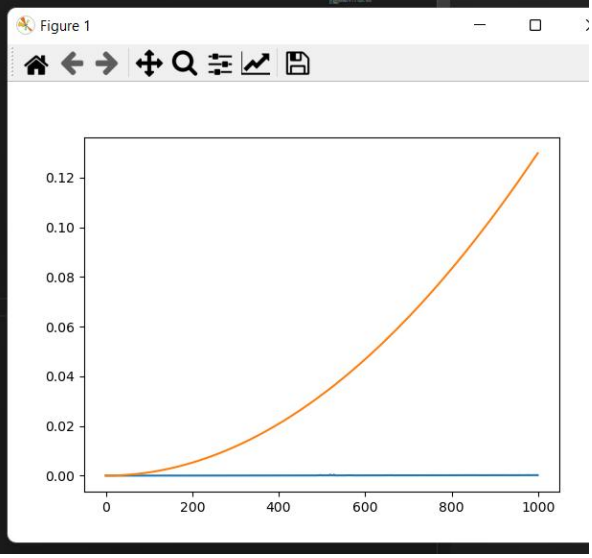
```
i =  992
i =  993
i =  994
i =  995
i =  996
i =  997
i =  998
i =  999
i =  1000
```

DEBUG CONSOLE

No 3c

E: > PRAK ASD > Modul 10 > no 3c.py > ujiKalangBersusuh

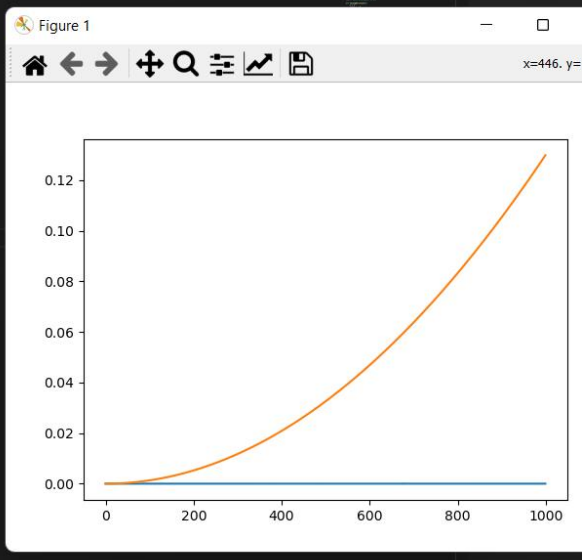
```python
import timeit
import matplotlib.pyplot as plt

def kalangBersusuh(n):
    test = 0
    for i in range(n):
        test = test + 1
    for j in range(n):
        test = test -1

def ujiKalangBersusuh(n):
    ls = []
    jangkauan = range(1, n+1)
    siap = 'from __main__ import kalangBersusuh'
    for i in jangkauan:
        print('i = ',i)
        t = timeit.timeit('kalangBersusuh(' + str(i) +')', setup = siap,number=1)
        ls.append(t)
    return ls
n = 1000
LS = ujiKalangBersusuh(n)
plt.plot(LS)
skala = 7700000
plt.plot([x*x/skala for x in range(1, n+1)])
plt.show()
```



No 3d

E: > PRAK ASD > Modul 10 > no 3d.py > ujiKalangBersusuh

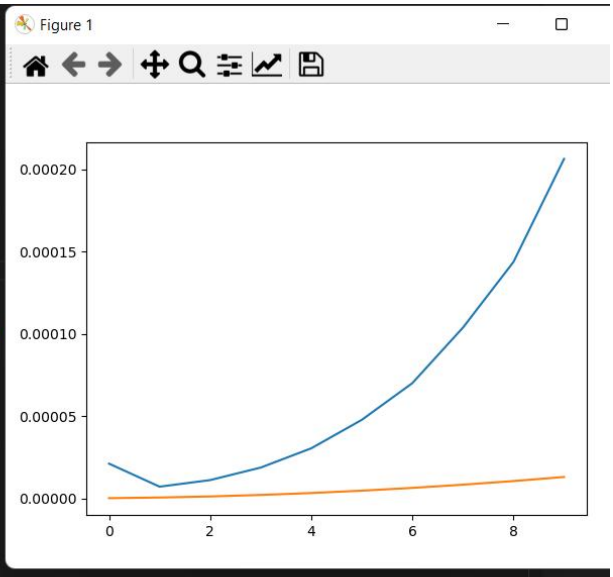
```python
import timeit
import matplotlib.pyplot as plt

def kalangBersusuh(n):
    i = n
    while i > 0:
        k = 2 + 2
        i = i // 2

def ujiKalangBersusuh(n):
    ls = []
    jangkauan = range(1, n+1)
    siap = 'from __main__ import kalangBersusuh'
    for i in jangkauan:
        print('i = ',i)
        t = timeit.timeit('kalangBersusuh(' + str(i) +')', setup = siap,number=1)
        ls.append(t)
    return ls
n = 1000
LS = ujiKalangBersusuh(n)
plt.plot(LS)
skala = 7700000
plt.plot([x*x/skala for x in range(1, n+1)])
plt.show()
```



No 3e

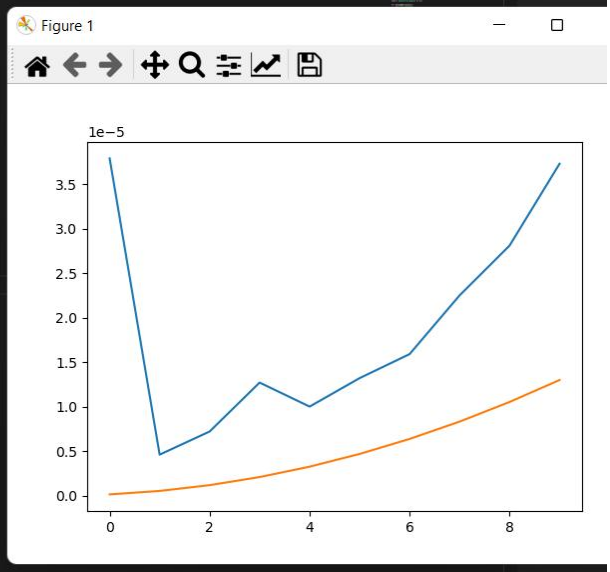
E: > PRAK ASD > Modul 10 > no 3e.py > ujiKalangBersusuh

```python
import timeit
import matplotlib.pyplot as plt

def kalangBersusuh(n):
    for i in range(n):
        for j in range(n):
            for k in range(n):
                m = i + j + k + 2019

def ujiKalangBersusuh(n):
    ls = []
    jangkau = range(1, n+1)
    siap = 'from __main__ import kalangBersusuh'
    for i in jangkau:
        print('i = ',i)
        t = timeit.timeit('kalangBersusuh(' + str(i) +')', setup = siap,number=1)
        ls.append(t)
    return ls

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



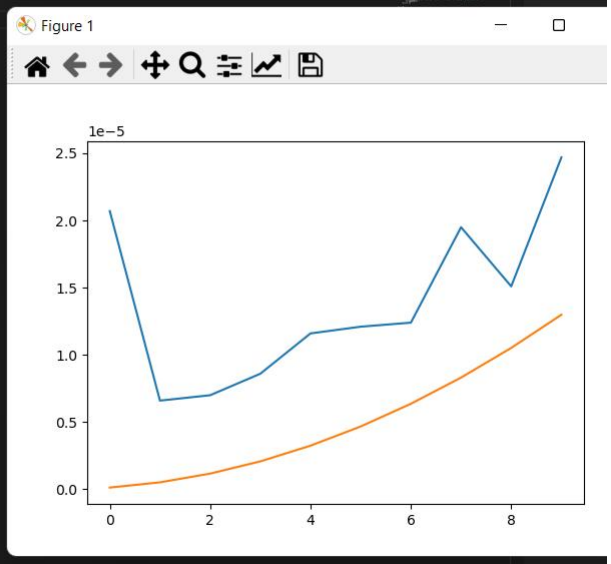
No 3f

```python
import timeit
import matplotlib.pyplot as plt

def kalangBersusuh(n):
    for i in range(n):
        for j in range(i):
            for k in range(j):
                x = i + j + k + 2019

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

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



No 3g

```python
    j += j

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

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



Nomor 4

Mengurutkan kompleksitasnya lambat ke yang cepat.
n log2 n 4n 10 log2 n 5n2 log4 n 12n6 2 log 2n n3
= log4 n 2 log 2n 10 log2 n 5n2 n3 12n6 4n

Nomor 5
Menentukan O(·) dari fungsi-fungsi berikut:
a. T(n) = n2 + 32n + 8
= O(n2)
b. T(n) = 87n + 8n
= O(n)
c. T(n) = 4n + 5n log n + 102
= O(n)
d. T(n) = log n + 3n2 + 88
= O(n2)
e. T(n) = 3(2n) + n2 + 647
= O(n2)

f. T(n, k) = kn + log k
= O(kn)
g. T(n, k) = 8n + k log n + 800
= O(n)
h. T(n, k) = 100kn + n
= O(nk)

NO 6

Array Sorting Algorithms

Algorithm	Time Complexity			Space Complexity
	Best	Average	Worst	Worst
Quicksort	Ω(n log(n))	θ(n log(n))	O(n^2)	O(log(n))
Mergesort	Ω(n log(n))	θ(n log(n))	O(n log(n))	O(n)
Timsort	Ω(n)	θ(n log(n))	O(n log(n))	O(n)
Heapsort	Ω(n log(n))	θ(n log(n))	O(n log(n))	O(1)
Bubble Sort	Ω(n)	θ(n^2)	O(n^2)	O(1)
Insertion Sort	Ω(n)	θ(n^2)	O(n^2)	O(1)
Selection Sort	Ω(n^2)	θ(n^2)	O(n^2)	O(1)
Tree Sort	Ω(n log(n))	θ(n log(n))	O(n^2)	O(n)
Shell Sort	Ω(n log(n))	θ(n(log(n))^2)	O(n(log(n))^2)	O(1)
Bucket Sort	Ω(n+k)	θ(n+k)	O(n^2)	O(n)
Radix Sort	Ω(nk)	θ(nk)	O(nk)	O(n+k)
Counting Sort	Ω(n+k)	θ(n+k)	O(n+k)	O(k)
Cubesort	Ω(n)	θ(n log(n))	O(n log(n))	O(n)

» TimeComplexity

This page documents the time-complexity (aka "Big O" or "Big Oh") of various operations in current CPython. Other Python implementations (or older or still-under development versions of CPython) may have slightly different performance characteristics. However, it is generally safe to assume that they are not slower by more than a factor of O(log n).

Generally, 'n' is the number of elements currently in the container. 'k' is either the value of a parameter or the number of elements in the parameter.

list

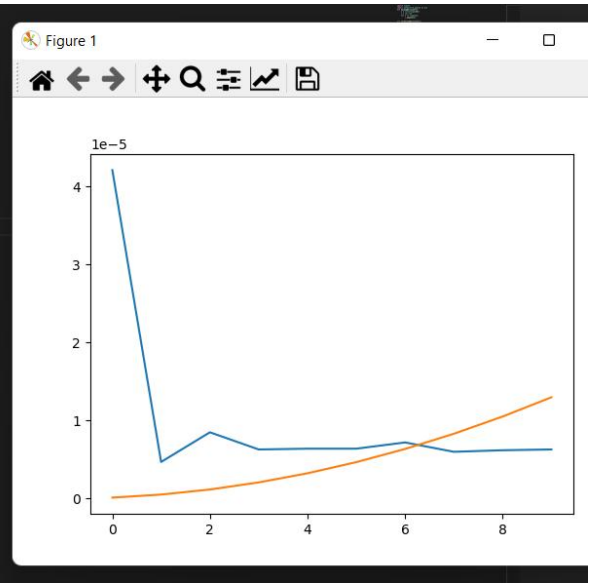
The Average Case assumes parameters generated uniformly at random.

Internally, a list is represented as an array; the largest costs come from growing beyond the current allocation size (because everything must move), or from inserting or deleting somewhere near the beginning (because everything after that must move). If you need to add/remove at both ends, consider using a collections.deque instead.

Operation	Average Case	Amortized Worst Case
Copy	O(n)	O(n)
Append[1]	O(1)	O(1)
Pop last	O(1)	O(1)
Pop intermediate[2]	O(n)	O(n)
Insert	O(n)	O(n)
Get Item	O(1)	O(1)
Set Item	O(1)	O(1)
Delete Item	O(n)	O(n)
Iteration	O(n)	O(n)
Get Slice	O(k)	O(k)
Del Slice	O(n)	O(n)
Set Slice	O(k+n)	O(k+n)
Extend[1]	O(k)	O(k)
Sort	O(n log n)	O(n log n)
Multiply	O(nk)	O(nk)
x in s	O(n)	
min(s), max(s)	O(n)	
Get Length	O(1)	O(1)

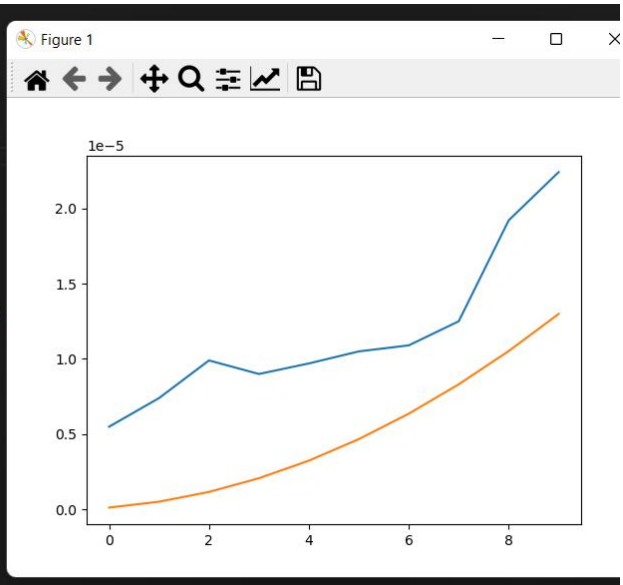
No 7

```python
import timeit
import matplotlib.pyplot as plt
def kalangBersusuh(n):
    L = list(range(30))
    L = L[::-1]
    for i in range(n):
        L.append(n)

def ujiKalangBersusuh(n):
    ls = []
    jangkau = range(1, n+1)
    siap = 'from __main__ import kalangBersusuh'
    for i in jangkau:
        print('i = ',i)
        t = timeit.timeit('kalangBersusuh(' + str(i) +')', setup = siap,number=1)
        ls.append(t)
    return ls

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

```



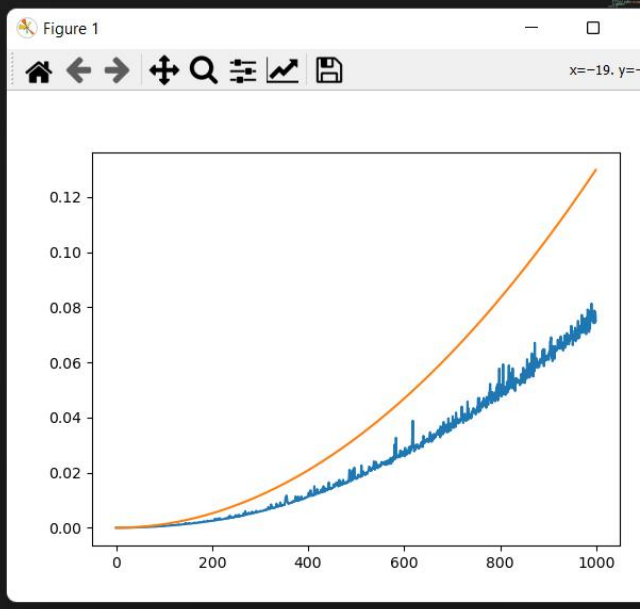
No 8

```python
import timeit
import matplotlib.pyplot as plt
def kalangBersusuh(n):
    L = list(range(30))
    L = L[::-1]
    for i in range(n):
        for x in range(n):
            L.insert(i,x)

def ujiKalangBersusuh(n):
    ls = []
    jangkau = range(1, n+1)
    siap = 'from __main__ import kalangBersusuh'
    for i in jangkau:
        print('i = ',i)
        t = timeit.timeit('kalangBersusuh(' + str(i) +')', setup = siap,number=1)
        ls.append(t)
    return ls

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



No 9

```python
import timeit
import time
import matplotlib.pyplot as plt

def straight(cont, target):
    x = len(cont)
    for i in range(x):
        if cont[i] == target:
            return True
        return False

def tim():
    a = 100
    b = [12, 3, 5, 6, 8, 2, 11]
    awal = time.time()
    U = straight(b, a)
    akhir = time.time()
    print('Worst case')
    print('mengurutkan %d bilangan perlu %8.7f detik' % (U, akhir-awal))

tim()
tim()

def kalangBersusuh(n):
    a = 100
    b = [12, 3, 5, 6, 8, 2, 11]
    U = straight(b, a)

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

n = 100
LS = ujiKalangBersusuh(n)
plt.plot(LS)
skala = 7700000
plt.plot([x*x/skala for x in range(1, n+1)])
plt.show()
```



OBLEMS OUTPUT DEBUG CONSOLE JUPYTER

No 10

dict

The Average Case times listed for dict objects assume that the hash function for the objects is sufficiently robust to make collisions uncommon. The Average Case assumes the keys used in parameters are selected uniformly at random from the set of all keys.

Note that there is a fast-path for dicts that (in practice) only deal with str keys; this doesn't affect the algorithmic complexity, but it can significantly affect the constant factors: how quickly a typical program finishes.

Operation	Average Case	Amortized Worst Case
k in d	O(1)	O(n)
Copy[3]	O(n)	O(n)
Get Item	O(1)	O(n)
Set Item[1]	O(1)	O(n)
Delete Item	O(1)	O(n)
Iteration[3]	O(n)	O(n)

No 11

Big-O O(·)	Big-Theta Θ(·)	Big-Omega Ω(·)
Tingkat pertumbuhan algoritma kurang dari atau sama dengan nilai tertentu (<=).	Tingkat pertumbuhan sama dengan nilai yang ditentukan (==).	Tingkat pertumbuhan lebih bsar dari atau sama dengan nilai yang ditentukan (>=).
Batas atas algoritma diwakili oleh notasi Big O. Hanya fungsi di atas yang dibatasi oleh Big O. ikatan atas asimtotik yang diberikan oleh notasi Big-O.	Pembatas fungsi dari atas dan bawah diwakili oleh notasi theta. Perilaku asimptotik yang tepat dilakukan oleh notasi theta ini.	Batas bawah algoritma diwakili oleh notasi Omega. Ikatan bawah asimptotik diberikan oleh notasi Omega
Didefinisikan sebagai batas atas (upper bound) dan batas atas pada algoritma adalah jumlah waktu terbanyak yang dibutuhkan (kinerja kasus terburuk).	Didefinisikan sebagai terikat paling ketat (tight bound) dan terikat paling ketat adalah yang terbaik dari semua kasus terburuk yang dapat diambil oleh algoritma	Didefinisikan sebagai batas bawah (lower bound) dan batas bawah pada algoritma adalah jumlah waktu paling sedikit yang dibutuhkan (cara seefesien mungkin, dengan kata lain kasus terbaik)

No 12

Amortized Analysis = Metode untuk menganalisis kompleksitas waktu algoritma · Push(S,x) O(1) memasukkan objek x ke dalam stack,digunakan untuk algoritma yang mana operasi terkadang berjalan sangat lambat, tetapi sebagian besar operasi lainnya lebih cepat. Amortized analysis menganalisis uruta operasi dan menjamin waktu rata-rata kasus terburuk yang lebih rendah daripada waktu kasus terburuk dari operasi yang sangat mahal.