# ALGORITMA DAN STRUKTUR DATA

INF1083

## LAPORAN PRAKTIKUM 5 : Analisis Algoritma (2)

Oleh :

*Akhmad Qasim*

*2211102441237*

Teknik Informatika
Fakultas Sains & Teknologi
Universitas Muhammadiyah Kalimantan Timur

Samarinda, 2023

# Laporan Praktikum 5:
# Analisis Algoritma (2)

**Pokok Bahasan:**

- ❖ Komparasi Algoritma
- ❖ List
- ❖ Dictionaries

**Tujuan Pembelajaran:**

- ✓ Memahami bagaimana menganalisis algoritma
- ✓ Memahami algoritma list dan dictionaries

**Percobaan & Latihan:**

1. **List**

a) Berikan tampilan output dari perintah diatas!
Tampilan Output:

```
concat  0.9763643000005686 milliseconds
append  0.04675249999854714 milliseconds
comprehension  0.02519910000046366 milliseconds
list range  0.007917099999758648 milliseconds
```

b) Analisa hasil dari perintah diatas!

```
from timeit import Timer  # import modul Timer dari modul timeit


new *
def test1():  # fungsi test1
    l = []  # membuat list kosong
    for i in range(1000):  # perulangan sebanyak 1000 kali
        l = l + [i]  # menambahkan nilai i ke list l


new *
def test2():  # fungsi test2
    l = []  # membuat list kosong
    for i in range(1000):  # perulangan sebanyak 1000 kali
        l.append(i)  # menambahkan nilai i ke list l


new *
def test3():  # fungsi test3
    l = [i for i in range(1000)]  # membuat list dengan comprehension


```

```
    akhmadqasim *
19  def test4():  # fungsi test4
20      l = list(range(1000))
21
22
23  t1 = Timer("test1()", "from __main__ import test1")  # membuat objek Timer dengan fungsi test1
24  print("concat ", t1.timeit(number=1000), "milliseconds")  # menampilkan waktu eksekusi fungsi test1
25  t2 = Timer("test2()", "from __main__ import test2")  # membuat objek Timer dengan fungsi test2
26  print("append ", t2.timeit(number=1000), "milliseconds")  # menampilkan waktu eksekusi fungsi test2
27  t3 = Timer("test3()", "from __main__ import test3")  # membuat objek Timer dengan fungsi test3
28  print("comprehension ", t3.timeit(number=1000), "milliseconds")  # menampilkan waktu eksekusi fungsi test3
29  t4 = Timer("test4()", "from __main__ import test4")  # membuat objek Timer dengan fungsi test4
30  print("list range ", t4.timeit(number=1000), "milliseconds")  # menampilkan waktu eksekusi fungsi test4
```

## 2. List

a) Gunakan variabel diatas dengan perintah berikut! Berikan output dan analisa!

Syntax:

```
1  import timeit  # import modul timeit
2
3  popzero = timeit.Timer("x.pop(0)", "from __main__ import x")  # membuat objek Timer dengan fungsi pop(0)
4  popend = timeit.Timer("x.pop()", "from __main__ import x")  # membuat objek Timer dengan fungsi pop()
5
6  x = list(range(2000000))  # membuat list dengan range 2000000
7  print(popzero.timeit(number=1000))  # menampilkan waktu eksekusi fungsi pop(0)
```

Tampilan Output:

```
1.6047548999995342
```

b) Gunakan variabel diatas dengan perintah berikut! Berikan output dan analisa!

Syntax:

```
1  import timeit  # import modul timeit
2
3  popzero = timeit.Timer("x.pop(0)", "from __main__ import x")  # membuat objek Timer dengan fungsi pop(0)
4  popend = timeit.Timer("x.pop()", "from __main__ import x")  # membuat objek Timer dengan fungsi pop()
5
6  x = list(range(2000000))  # membuat list dengan range 2000000
7  print(popend.timeit(number=1000))  # menampilkan waktu eksekusi fungsi pop()
```

Tampilan Output:

```
3.640000068116933e-05
```

c) Jelaskan perintah pop() dan pop(0) pada variabel diatas!

Perbedaan perintah pop(0) dan pop() adalah perintah pop(0) akan menghapus elemen pertama dari list, sedangkan perintah pop() akan menghapus elemen terakhir dari list. Perbedaan waktu eksekusi antara perintah pop(0) dan pop() adalah perintah pop(0) akan lebih lama karena perintah pop(0) harus menggeser elemen-elemen yang ada di list ke kiri. Sedangkan perintah pop() hanya menghapus elemen terakhir dari list.

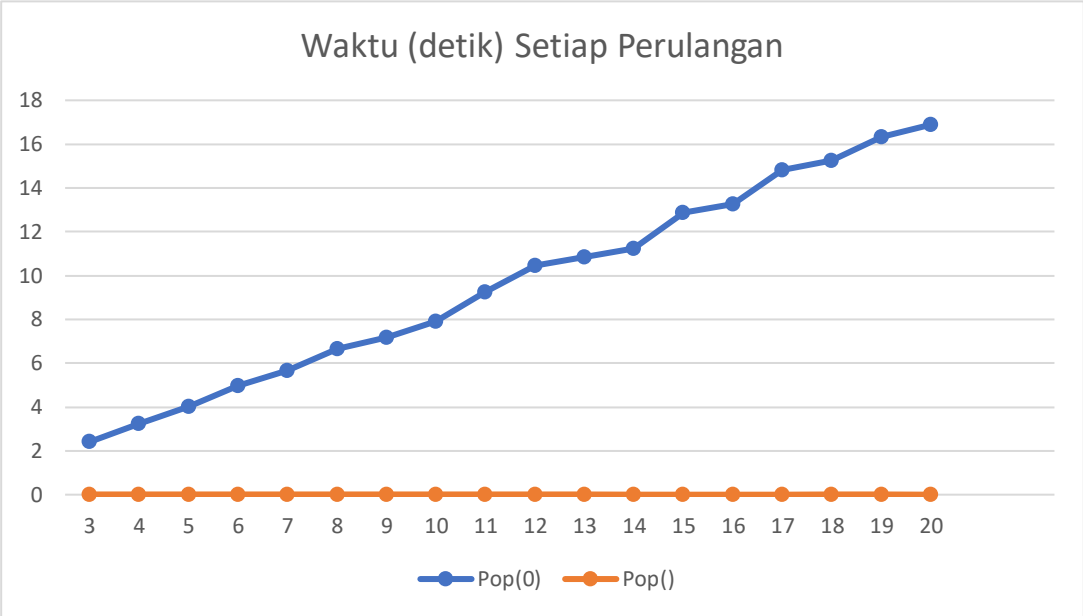
**3. List**

a) Berikan 20 hasil output dari perintah diatas!

Tampilan Output:

```
pop(0)              pop()
        0.81445,            0.00004
        1.57247,            0.00005
        2.41265,            0.00006
        3.24071,            0.00007
        4.00388,            0.00006
        4.98616,            0.00007
        5.65833,            0.00005
        6.66966,            0.00005
        7.16343,            0.00006
        7.90081,            0.00006
        9.26203,            0.00006
       10.47471,            0.00006
       10.84309,            0.00005
       11.22747,            0.00008
       12.87182,            0.00004
       13.24572,            0.00004
       14.83358,            0.00004
       15.26058,            0.00005
       16.32277,            0.00005
       16.88690,            0.00004
```

b) Buatlah grafik dari kedua hasil variabel diatas ( pop(0) & pop() )!




c) Berikan analisa dan tentukan notasi BigO-nya!

Syntax:

```
import timeit  # import modul timeit
popzero = timeit.Timer("x.pop(0)", "from __main__ import x")  # membuat objek Timer dengan fungsi pop(0)
popend = timeit.Timer("x.pop()", "from __main__ import x")  # membuat objek Timer dengan fungsi pop()

print("pop(0)            pop()")  # menampilkan judul tabel
for i in range(1000000, 100000001, 1000000):  # perulangan sebanyak 100000000 dengan interval 1000000
    x = list(range(i))  # membuat list dengan range i
    pt = popend.timeit(number=1000)  # menampilkan waktu eksekusi fungsi pop()
    x = list(range(i))  # membuat list dengan range i
    pz = popzero.timeit(number=1000)  # menampilkan waktu eksekusi fungsi pop(0)
    print("%15.5f, %15.5f" % (pz, pt))  # menampilkan waktu eksekusi fungsi pop(0) dan pop()
```

Notasi Big-O dari fungsi pop(0) adalah O(n) dan fungsi pop() adalah O(1).

## 4. Dictionaries

a) Jelaskan perintah diatas perbaris!

Syntax:

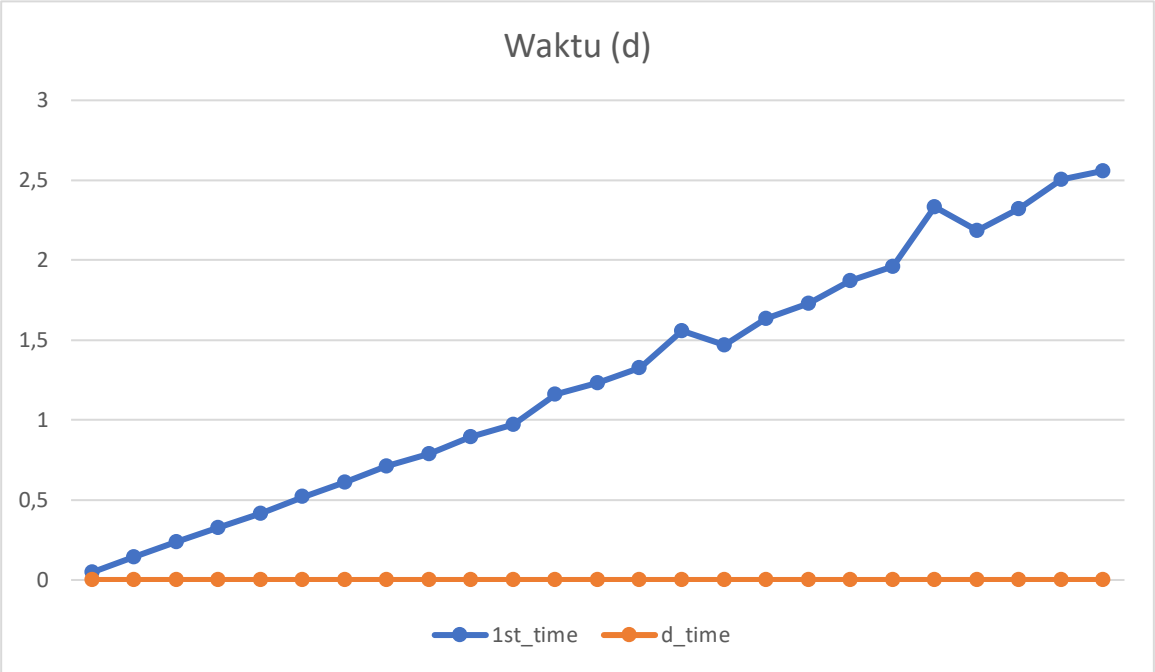
```
import timeit  # import modul timeit
import random  # import modul random

for i in range(10000, 500001, 20000):  # perulangan sebanyak 500000 dengan interval 20000
    # membuat objek Timer dengan fungsi random.randrange()
    t = timeit.Timer("random.randrange(%d) in x" % i, "from __main__ import random, x")
    x1 = list(range(i))  # membuat list dengan range i
    lst_time = t.timeit(number=1000)  # menampilkan waktu eksekusi fungsi random.randrange()
    x2 = {j: None for j in range(i)}  # membuat dictionary dengan range i
    d_time = t.timeit(number=1000)  # menampilkan waktu eksekusi fungsi random.randrange()
    # menampilkan waktu eksekusi fungsi random.randrange() pada list dan dictionary
    print("%d, %10.3f, %10.3f" % (i, lst_time, d_time))
```

b) Berikan hasil output dari perintah diatas!
Tampilan Output:

```
10000,      0.048,      0.001
30000,      0.142,      0.001
50000,      0.238,      0.000
70000,      0.325,      0.001
90000,      0.419,      0.001
110000,      0.520,      0.001
130000,      0.611,      0.001
150000,      0.711,      0.001
170000,      0.787,      0.001
190000,      0.896,      0.001
210000,      0.975,      0.001
230000,      1.161,      0.001
250000,      1.232,      0.001
270000,      1.329,      0.001
290000,      1.557,      0.001
310000,      1.472,      0.001
330000,      1.636,      0.001
350000,      1.729,      0.001
370000,      1.871,      0.001
390000,      1.963,      0.001
410000,      2.332,      0.001
430000,      2.188,      0.001
450000,      2.321,      0.001
470000,      2.505,      0.001
490000,      2.559,      0.001
```

c) Buatlah grafik dari kedua hasil variabel diatas ( 1st_time & d_time )!




d) Berikan analisa dan tentukan notasi BigO-nya!
Program tersebut mengukur waktu eksekusi untuk mencari apakah sebuah angka acak tertentu ada di dalam list atau dictionary. Program tersebut memiliki notasi Big O O(n) karena waktu eksekusinya bergantung pada jumlah elemen dalam list atau dictionary yang diperiksa. Perulangan dilakukan sebanyak 500.000 kali dengan interval 20.00. Pada setiap perulangan, program membuat sebuah objek Timer yang memanggil fungsi random.randrange() dengan parameter i.

Kemudian program membuat sebuah list dengan range i dan memeriksa apakah angka acak yang dihasilkan ada di dalam list tersebut menggunakan fungsi in. Setelah itu, program membuat sebuah dictionary dengan range i dan melakukan pemeriksaan yang sama dengan fungsi in. Dua hasil waktu eksekusi tersebut ditampilkan di console.

Dengan kata lain, program ini mengukur waktu eksekusi untuk operasi pencarian dalam list dan dictionary yang berukuran n elemen. Waktu eksekusi tersebut kemudian digunakan untuk mengamati bagaimana waktu eksekusi tumbuh seiring dengan peningkatan ukuran list atau dictionary, sehingga dapat memberikan gambaran tentang kompleksitas waktu algoritma pencarian yang digunakan.

**Kesimpulan:**

Pada praktikum kali ini kita menguji kecepatan tiga cara berbeda untuk membuat sebuah list berisi angka dari 0 sampai 999: dengan menggabungkan list, dengan append, dan dengan

comprehension yang menghasilkan output yang menunjukkan waktu eksekusi masing-masing cara. Setelah itu kita juga menguji kecepatan dua cara berbeda untuk menghapus elemen dari sebuah list: dengan pop(0) dan dengan pop() dengan menghasilkan output yang menunjukkan waktu eksekusi masing-masing cara untuk list dengan berbagai ukuran. Serta menguji kecepatan pencarian elemen dengan menggunakan fungsi random.randrange() pada list dan dictionary yang menghasilkan output yang menunjukkan waktu eksekusi pencarian pada list dan dictionary dengan berbagai ukuran.