Big O: Code Complexity as data grows

Write Better Code Using Python

Yunindyo Prabowo

https://git.io/bigopyid

Yunindyo Prabowo

- **profesi**: Mahasiswa
- **email**: yunindyo.prabowo@gmail.com
- **blog**: ypraw.gitlab.io
- **github/gitlab**: @ypraw

Introduction design and algorithm analysis

Apa itu Algoritma?

Algoritma merupakan langkah-langkah (Prosedur) yang harus dilakukan untuk menyelesaikan suatu masalah

Masalah

yaitu sebuah persoalan yang ingin diselesaikan oleh sebuah algoritma.

Masukan (input)

yaitu contoh data atau keadaan yang menjadi permasalahan.

keluaran (output)

yaitu bentuk akhir dari data atau keadaan setelah algoritma diimplementasikan ke masukan. Keluaran merupakan hasil ideal yang diinginkan dan dianggap telah menyelesaikan masalah.

Algoritma yang Baik

Benar

di mana algoritma menyelesaikan masalah dengan tepat, sesuai dengan definisi masukan / keluaran algoritma yang diberikan.

Efisien

berarti algoritma menyelesaikan masalah tanpa memberatkan bagian lain dari apliikasi. Sebuah algoritma yang tidak efisien akan menggunakan sumber daya (memori, CPU) yang besar dan memberatkan aplikasi yang mengimplementasikan algoritma tersebut

Mudah di Implementasikan

artinya sebuah algoritma yang baik harus dapat dimengerti dengan mudah sehingga implementasi algoritma dapat dilakukan siapapun dengan pendidikan yang tepat, dalam waktu yang masuk akal.

Realita ???

Big O : Asymptotic Notation for Complexity Analysis of Algorithms

Definition & Terminology

- O => stand for Order Of
- bla bla bla N => Banyaknya data
- O(N), O(log n) etc.
- not running time measure
- given 10 times data, how much time to finish???

Examples

Contoh 1

- Anda akan memberikan data penting kepada teman anda, namun anda dan teman anda berbeda kota, dengan ketentuan sebagai berikut, apa yang akan anda lakukan ????
  - anda di kota A dan teman anda di kota B dengan jarak tempuh 2 jam perjalanan.
  - data yang anda berikan berukuran 1TB
  - anda terhubung dengan koneksi internet dengan kecepatan rata-rata 16 Mbps

Solusi ????

Solusi pertama

- Anda akan mentrasfer data menggunakan akses internet tersebut
  - Proses ini digambarkan dalam notasi bigO sebagai O(n) dimana n menunjukan ukuran datanya, semakin banyak data yang ditransfer maka semakin bertambah linier waktu yang di perlukan

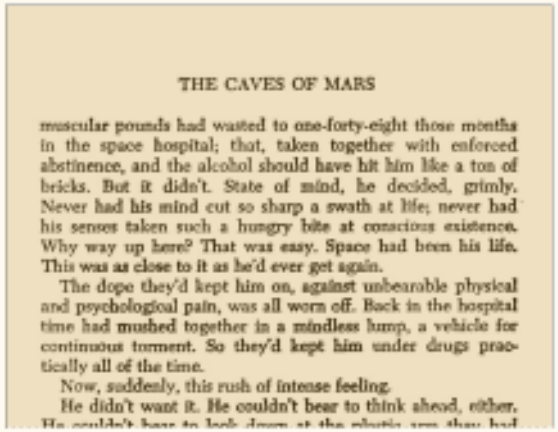
Solusi kedua

- Anda akan memberikan hardisk fisik anda kepada teman anda
  - ○ Proses ini digambarkan dalam notasi bigO sebagai O(1) dimana 1 menunjukan konstanta, nilai konstan 1 menunjukan bahwa operasi ini tidak akan berpengaruh terhadap ukuran data, jika waktu tempuh 2 jam makan 10TB atau 1 rak server berukuran 1 PB pun akan sampai 2 jam.

Another Analogy

Finding Word

Novel



THE CAVES OF MARS

muscular pounds had wasted to one-forty-eight those months in the space hospital; that, taken together with enforced abstinence, and the alcohol should have hit him like a ton of bricks. But it didn't. State of mind, he decided, grimly. Never had his mind cut so sharp a swath at life; never had his senses taken such a hungry bite at conscious existence. Why way up here? That was easy. Space had been his life. This was as close to it as he'd ever get again.

The dope they'd kept him on, against unbearable physical and psychological pain, was all worn off. Back in the hospital time had mushed together in a mindless lump, a vehicle for continuous torment. So they'd kept him under drugs practically all of the time.

Now, suddenly, this rush of intense feeling.

He didn't want it. He couldn't bear to think ahead, either.

O(N)

Encyclopedia

CANTON

O(log N)

contoh 2

```
def pangkat(x, y):
    hasil = 1
    for i in range(0, y):
        hasil = x * hasil
    return hasil
```

pembahasan contoh 2

```
pangkat(2,3)
```

```
hasil = 1              # iterasi ke 0 hasil =1
```

```
hasil = 2 * hasil      # iterasi ke 1 hasil = 2
```

```
hasil = 2 * hasil      # iterasi ke 2 hasil = 4
```

```
hasil = 2 * hasil      # iterasi ke 3 hasil = 8
```

```
    return hasil
```

pembahasan contoh 2 (lanj)

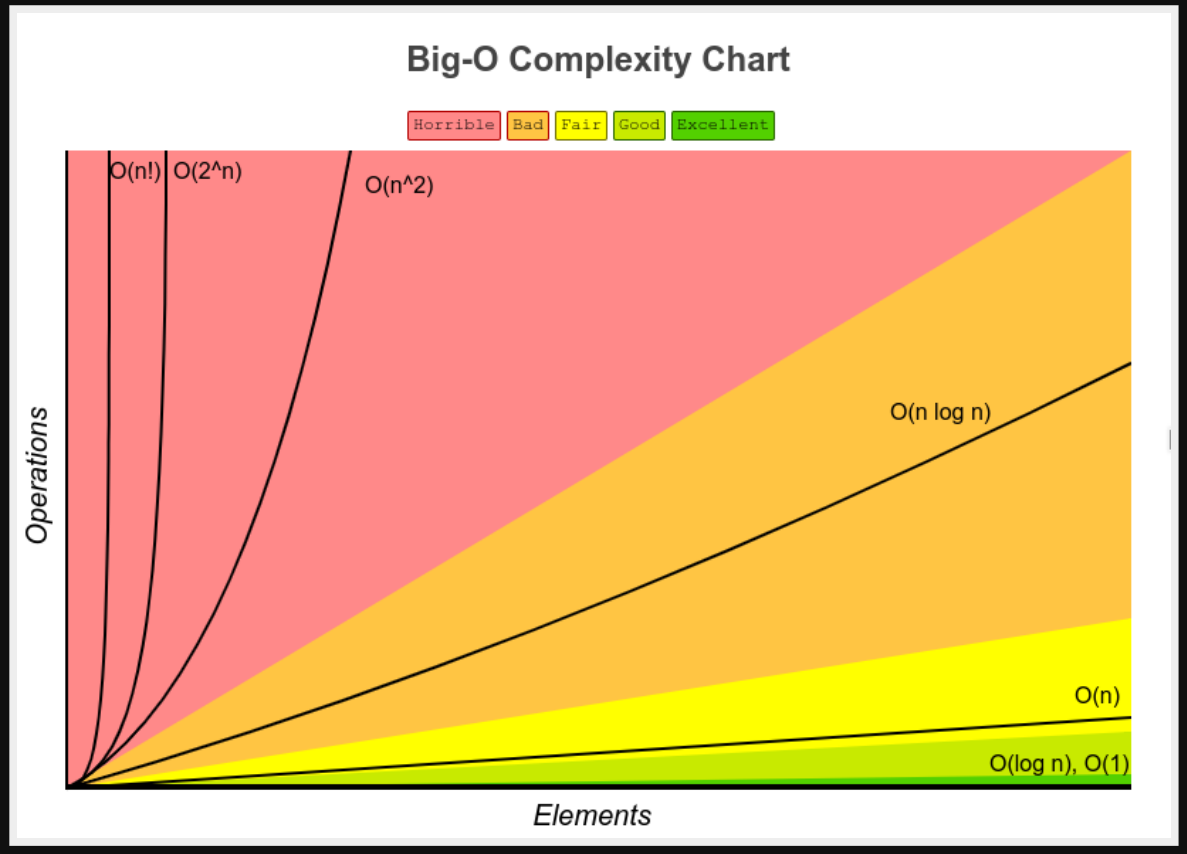
```
    hasil = 1
    for i in range(0, y):
        hasil = x * hasil
    return hasil
```

Baris Kode	Jumlah eksekusi
hasil = 1	1
hasil= x * hasil	y
return hasil	1

Kesimpulan contoh 2

dalam kata lain bahwa fungsi pangkat ini akan selalu di eksekusi sebanyak 2 + y atau bisa disederhanakan menjadi O(y) atau O(N) bigO Linier.

Grafik Pertumbuhan Notasi BigO



Python Time Complexity

List

[1,2,3 ....]	BigO
mylist.append(value)	O(1)
mylist[index]]	O(1)
value in mylist:	O(N)
for value in mylist:	O(N)
mylist.sort()	O(Log N)
append[1]	O(1)
insert	O(n)

Set

{1,2,3.... }	BigO
myset.add(value)	O(1)
value in myset:	O(1)
for value in myset:	O(N)

Dictionary

{key:value, key:value ....}	BigO
mydict[key] = val	O(1)
mydict[key]	O(1)
key in mydict:	O(1)
for key in mydict:	O(N)

Write Better Code using Python

syntax	type	bigO
insert	list	O(N)
append	list	O(1)

pro tips, dahulukan append daripada insert

Bonus

```
def SumOfList(MyList):
    if len(MyList) == 1:
        return MyList[0]
    mid = len(MyList) // 2
    left = SumOfList(MyList[:mid])
    right = SumOfList(MyList[mid:])
    return left + right
```

Pembahasan

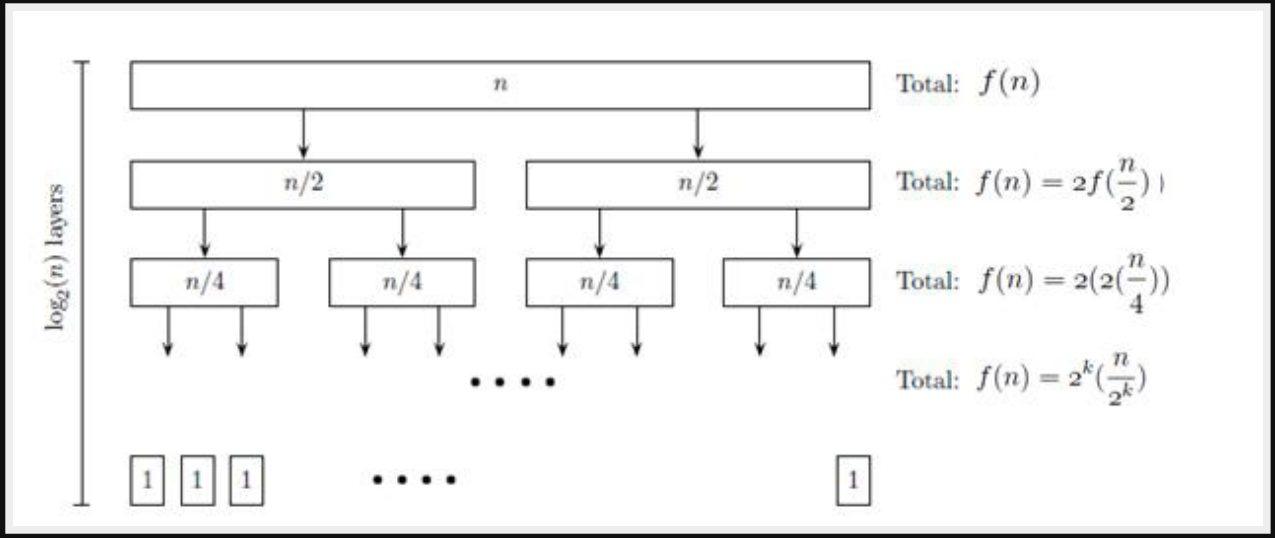
Baris Kode	Jumlah eksekusi
if len(MyList) == 1:	1
return MyList[0]	1
mid = len(MyList) // 2	1
left = SumOfList(MyList[:mid])	f(n/2) + 1
right = SumOfList(MyList[mid:])	f(n/2) + 1

Dalam Fungsi Matematis dapat Ditulis

$$f(n) = 1 + 1 + 1 + f(\frac{n}{2}) + 1 + f(\frac{n}{2}) + 1 + 1$$

$$= 6 + 2f(\frac{n}{2})$$

Ilustrasi


$\log_2(n)$ layers
n
$n/2$
$n/2$
$n/4$
$n/4$
$n/4$
$n/4$
1
1
1
1
Total: $f(n)$
Total: $f(n) = 2f(\frac{n}{2})$
Total: $f(n) = 2(2(\frac{n}{4}))$
Total: $f(n) = 2^k(\frac{n}{2^k})$

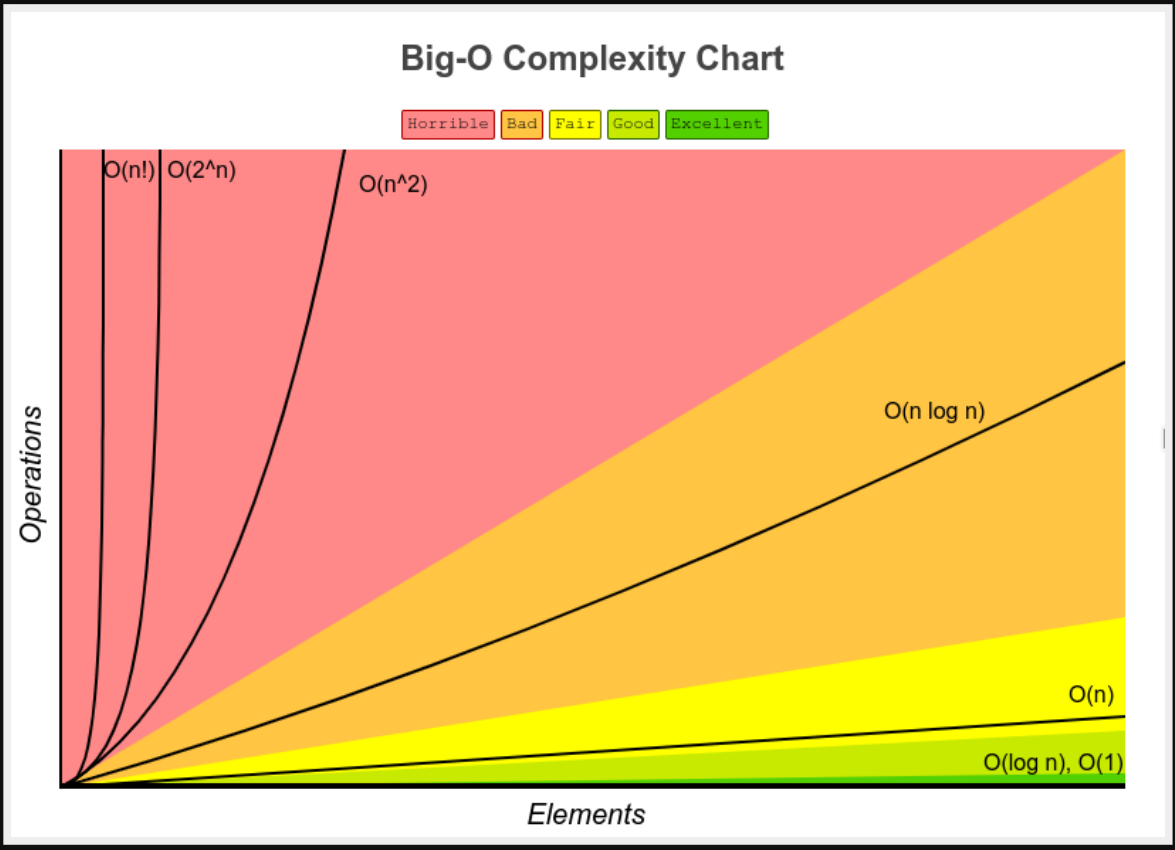
Syarat Berhenti

$$\frac{n}{2^k} = 1$$

$$n = 2^k$$

$$k = log_2 n$$

- atau dengan kata lain bahwa fungsi SumOfList diatas memenuhi syarat O(log N)


Big-O Complexity Chart
Horrible
Bad
Fair
Good
Excellent
O(n!)
O(2^n)
O(n^2)
O(n log n)
O(n)
O(log n), O(1)
Operations
Elements

TERIMA KASIH :)

More Resources

- Big-O: How Code Slows as Data Grows
- Problem Solving with Algorithms and Data Structures
- Analisis Algoritma