KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN

POLITEKNIK NEGERI JEMBER

Kompleksitas Algoritma (Big-O)

POLITEKNIK NEGERI JEMBER

Jember, 26 September 2020

Tim Pengampu Analisis dan Desain Algoritma - Zain Fathoni

KEMENTERIAN
PENDIDIKAN DAN KEBUDAYAAN

POLITEKNIK
NEGERI JEMBER

POLITEKNIK
NEGERI JEMBER

Perkenalan

Zain Fathoni, S.T.

KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN

POLITEKNIK NEGERI JEMBER

- Asal Jember, alumni SMA N 1 Jember
- S1 Teknik Informatika ITB
- Pengalaman kerja:

Tahun	Perusahaan	Jabatan
2014	QUN	Programmer
2015 – 2017	Xtremax	Software Developer, Software Development Manager
2017	Bukalapak	Software Engineer
2018 – sekarang	Ninja Van (Ninja Xpress)	Senior Software Engineer

POLITEKNIK NEGERI JEMBER

Zain Fathoni, S.T.

KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN

POLITEKNIK NEGERI JEMBER

- Aktivitas di luar pekerjaan:

Aktivitas	URL	Peran
KawalCOVID19	https://kawalcovid19.id/	Tech Lead
ReactJS Indonesia	https://reactjs.id/	Co-organizer
Frontend Indonesia	https://feid.dev/	Co-organizer
Konferensi & *Meetup*	https://zainf.dev/talk	Pembicara
Blog	https://zainf.dev/blog	Penulis
Pejuang Kode	https://zainf.dev/discord	Pendiri

Informasi selengkapnya ada di https://zainfathoni.com

POLITEKNIK NEGERI JEMBER

KEMENTERIAN
PENDIDIKAN DAN KEBUDAYAAN

POLITEKNIK
NEGERI JEMBER

POLITEKNIK
NEGERI JEMBER

Pendahuluan

Pendahuluan

POLITEKNIK
NEGERI JEMBER

- Sebuah masalah dapat mempunyai banyak algoritma penyelesaian. Contoh: masalah pengurutan (*sort*), ada puluhan algoritma pengurutan
- Sebuah algoritma tidak saja harus benar, tetapi juga harus mangkus (*efisien*).
- Algoritma yang bagus adalah algoritma yang mangkus (*efisien*).
- Kemangkusan (efisiensi) algoritma diukur berdasarkan waktu (*time*) eksekusi algoritma dan kebutuhan ruang (*space*) memori.

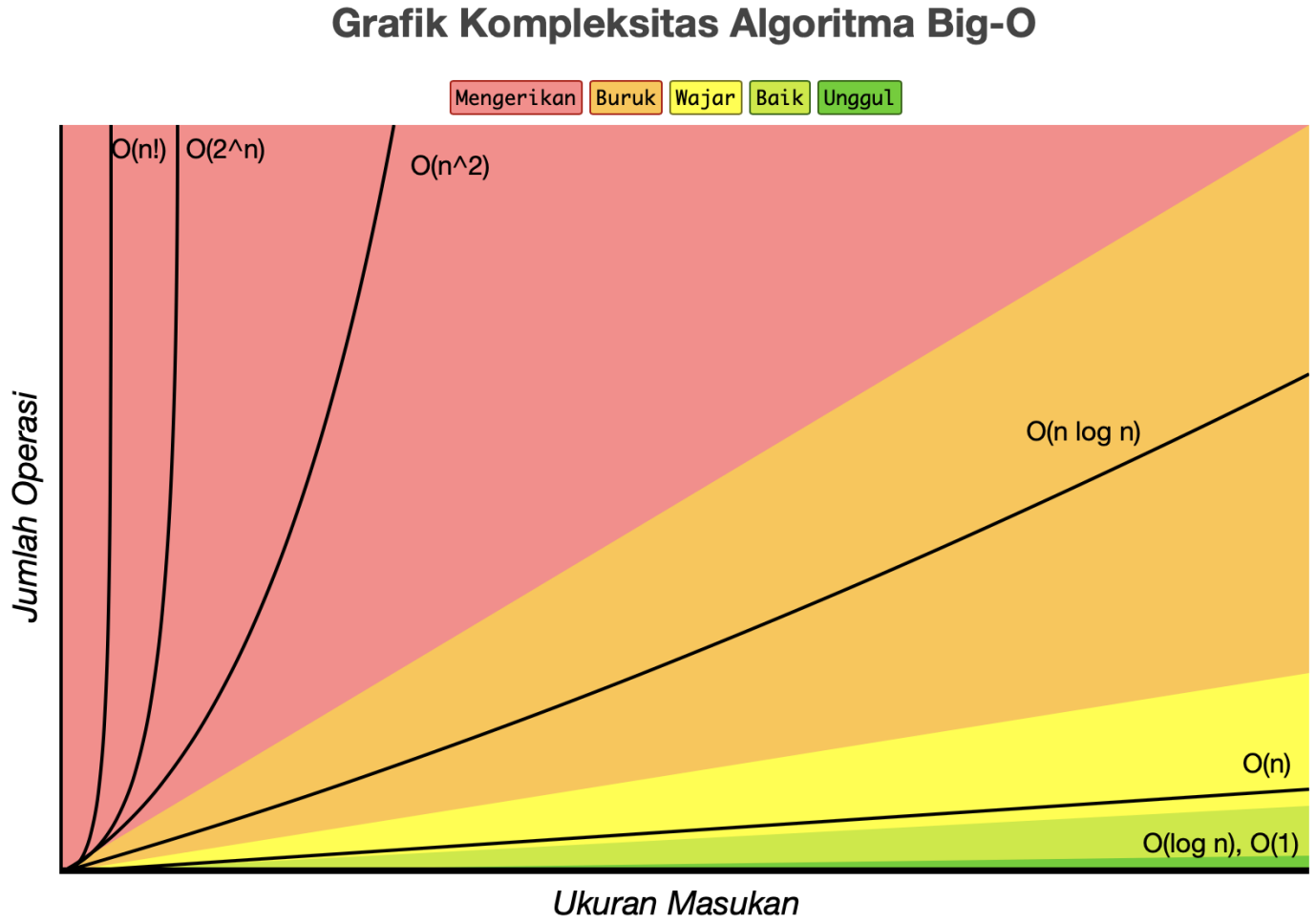
Algoritma yang Mangkus (Efisien)

POLITEKNIK NEGERI JEMBER

- Algoritma yang mangkus ialah algoritma yang meminimumkan kebutuhan waktu dan ruang.
- Kebutuhan waktu dan ruang suatu algoritma bergantung pada ukuran masukan (*n*), yang menyatakan jumlah data yang diproses.
- Kemangkusan algoritma dapat digunakan untuk menilai algoritma yang bagus dari sejumlah algoritma penyelesaian masalah.

Algoritma Mangkus untuk apa?

POLITEKNIK NEGERI JEMBER



Diterjemahkan dari https://www.bigocheatsheet.com

Contoh Kasus

POLITEKNIK
NEGERI JEMBER

- Seorang kurir Ninja Xpress harus mengirimkan sekaligus menjemput paket ke empat alamat rumah yang berbeda dalam satu kota.

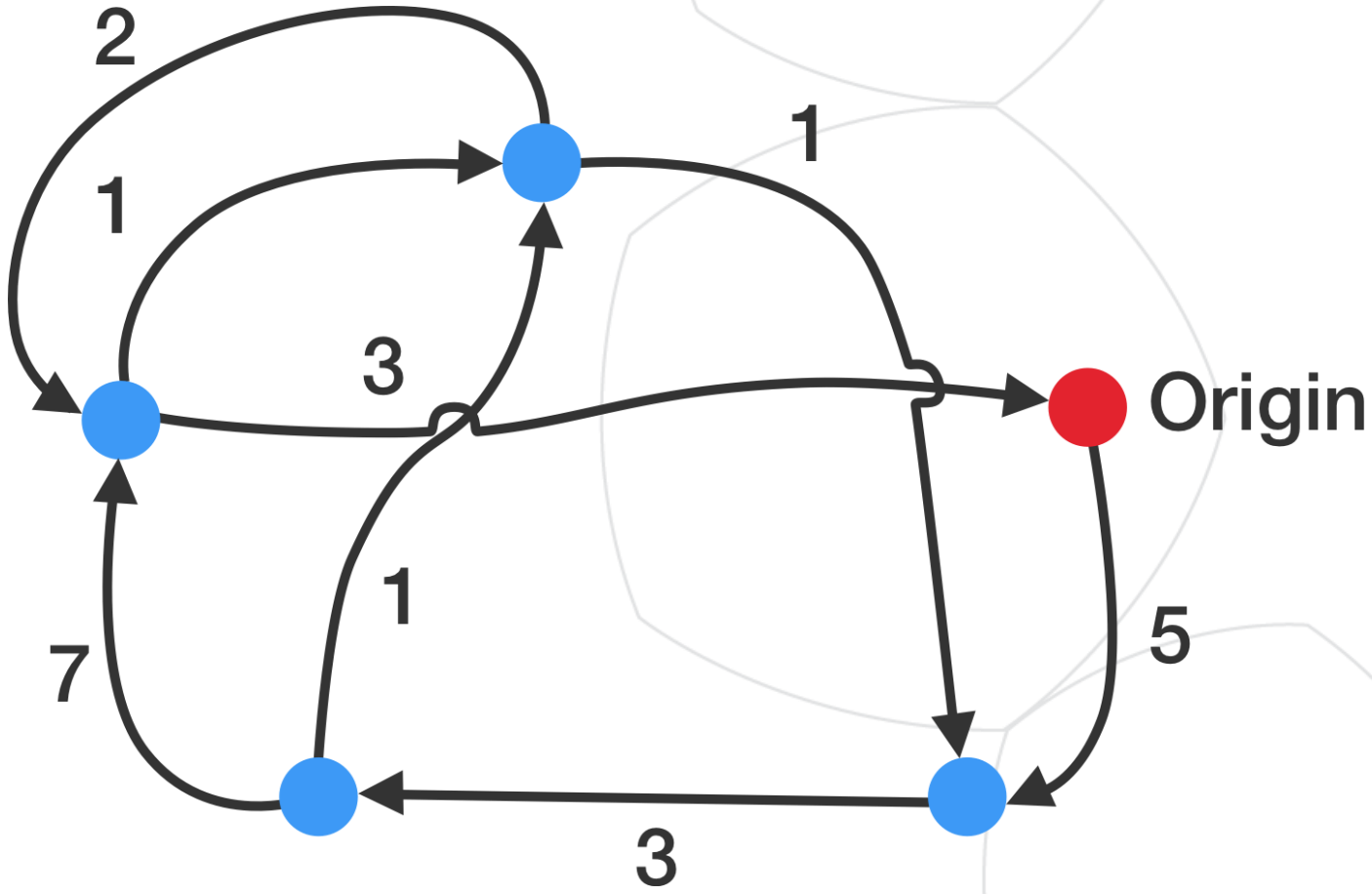
- Dia perlu mencari rute terpendek untuk menelusuri keempat alamat tersebut dalam rangka menghemat waktu dan bensin.

- Dia berangkat dari gudang Ninja Xpress untuk mengambil paket yang hendak diantarkan dan harus kembali lagi ke gudang untuk mengantarkan paket yang telah dijemput

Ilustrasi Jarak Antar Alamat

POLITEKNIK NEGERI JEMBER

- Arah garis mewakili arah jalan satu arah
- Angka pada garis mewakili jarak antar alamat (dalam km)
- *Origin* = Gudang Ninja Xpress
- Titik-titik biru = Alamat yang dituju



Sumber: https://brilliant.org/wiki/traveling-salesperson-problem/

Mari bersama kita cari solusinya!

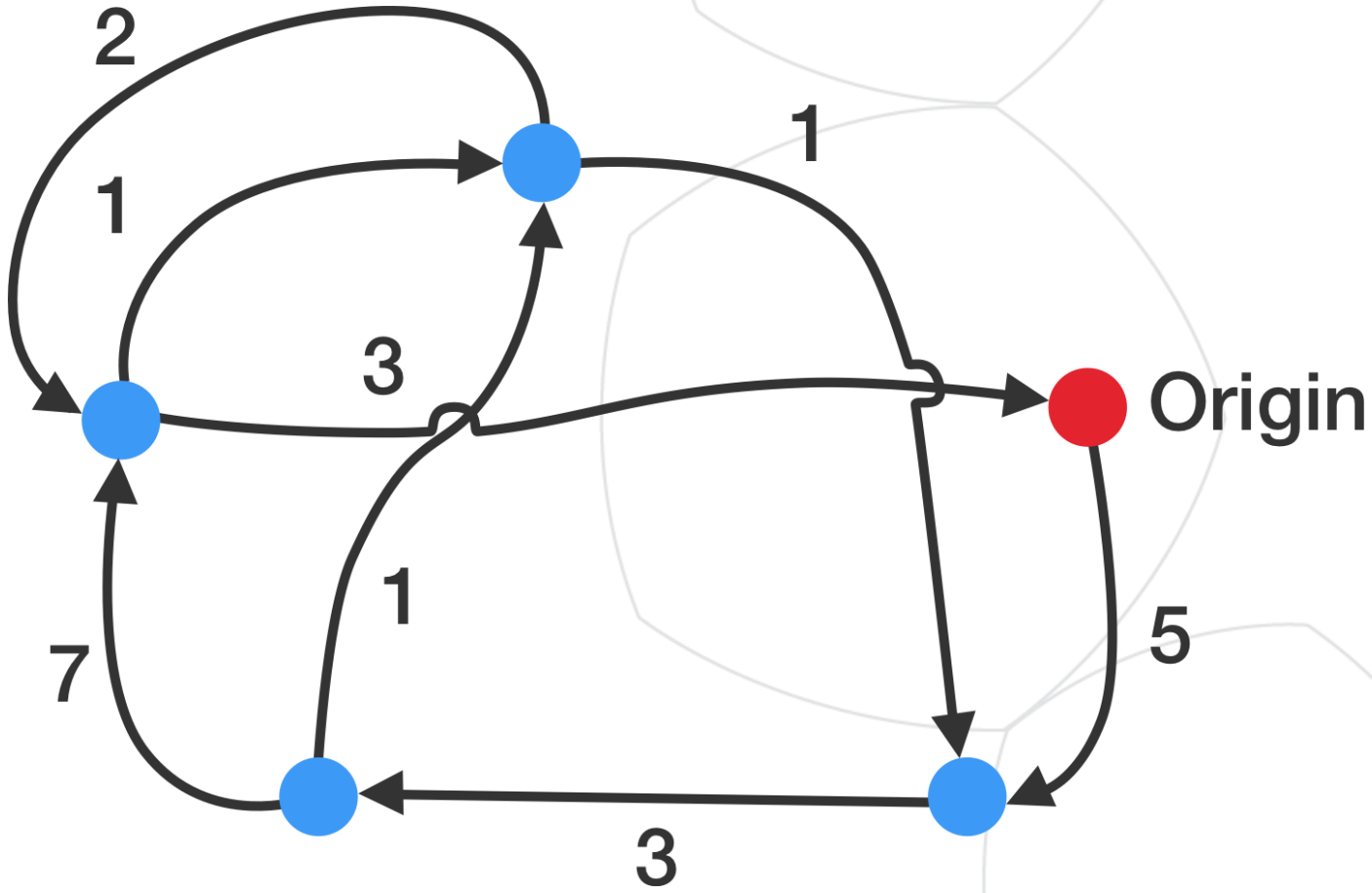
POLITEKNIK NEGERI JEMBER

Algoritma *Brute Force*:

Di setiap percabangan jalan, kita ambil yang kiri terlebih dahulu.

- 5 + 3 + 7 + 1 + 2 + 3 = 21
- 5 + 3 + 7 + 1 + 1 + 3 + 7 + 3 = 30
- 5 + 3 + 7 + 1 + 1 + 3 + 1 + 2 + 3 = 26
- 5 + 3 + 1 + 2 + 1 + 1 + 3 + 1 + 2 + 3 = 22
- 5 + 3 + 1 + 2 + 3 = 14

Total: 5 kali percobaan



Sumber: https://brilliant.org/wiki/traveling-salesperson-problem/

Kita coba cara lainnya!

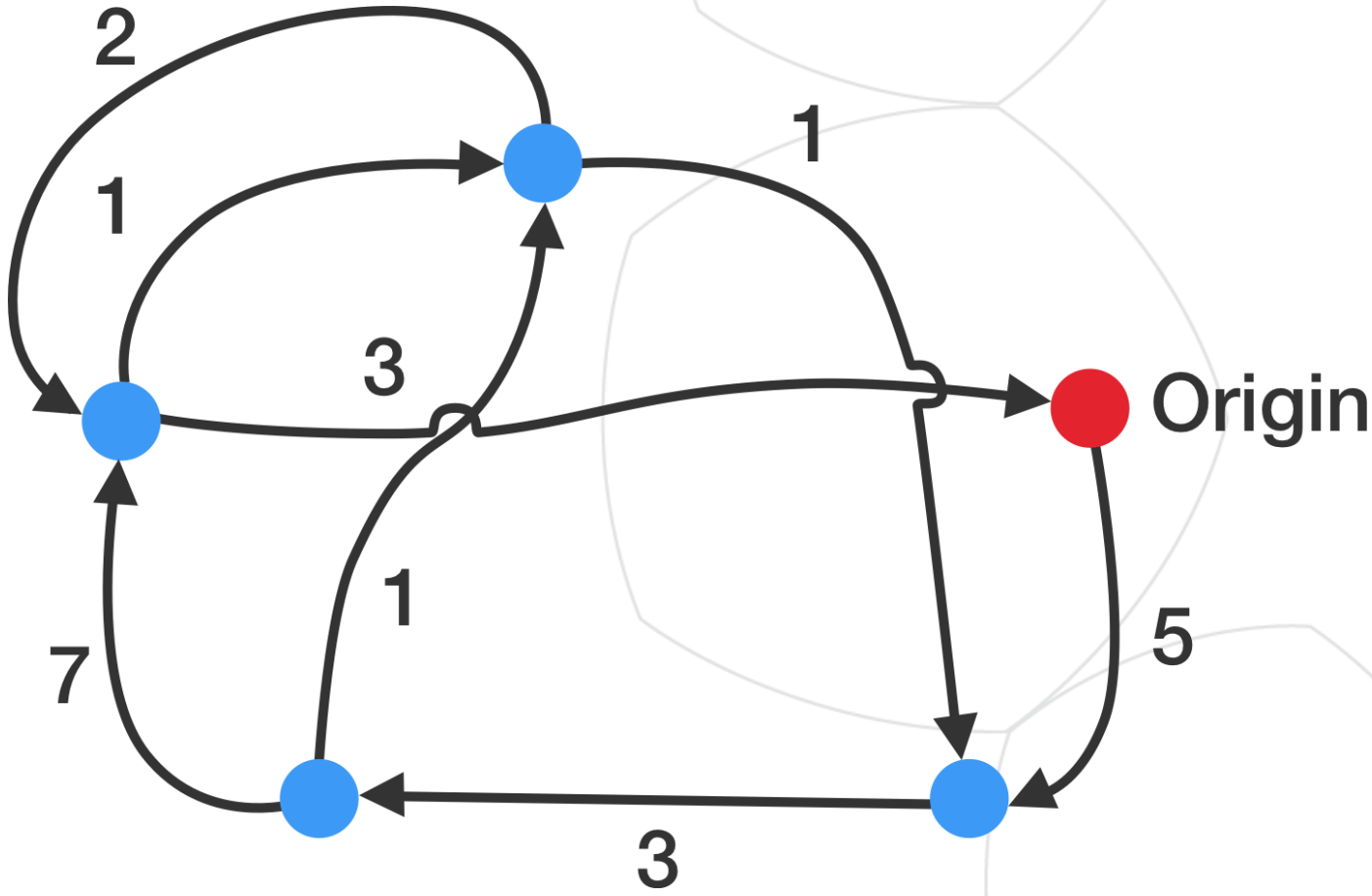
POLITEKNIK NEGERI JEMBER

Algoritma *Greedy*:

Di setiap percabangan jalan, kita ambil yang terpendek terlebih dahulu.

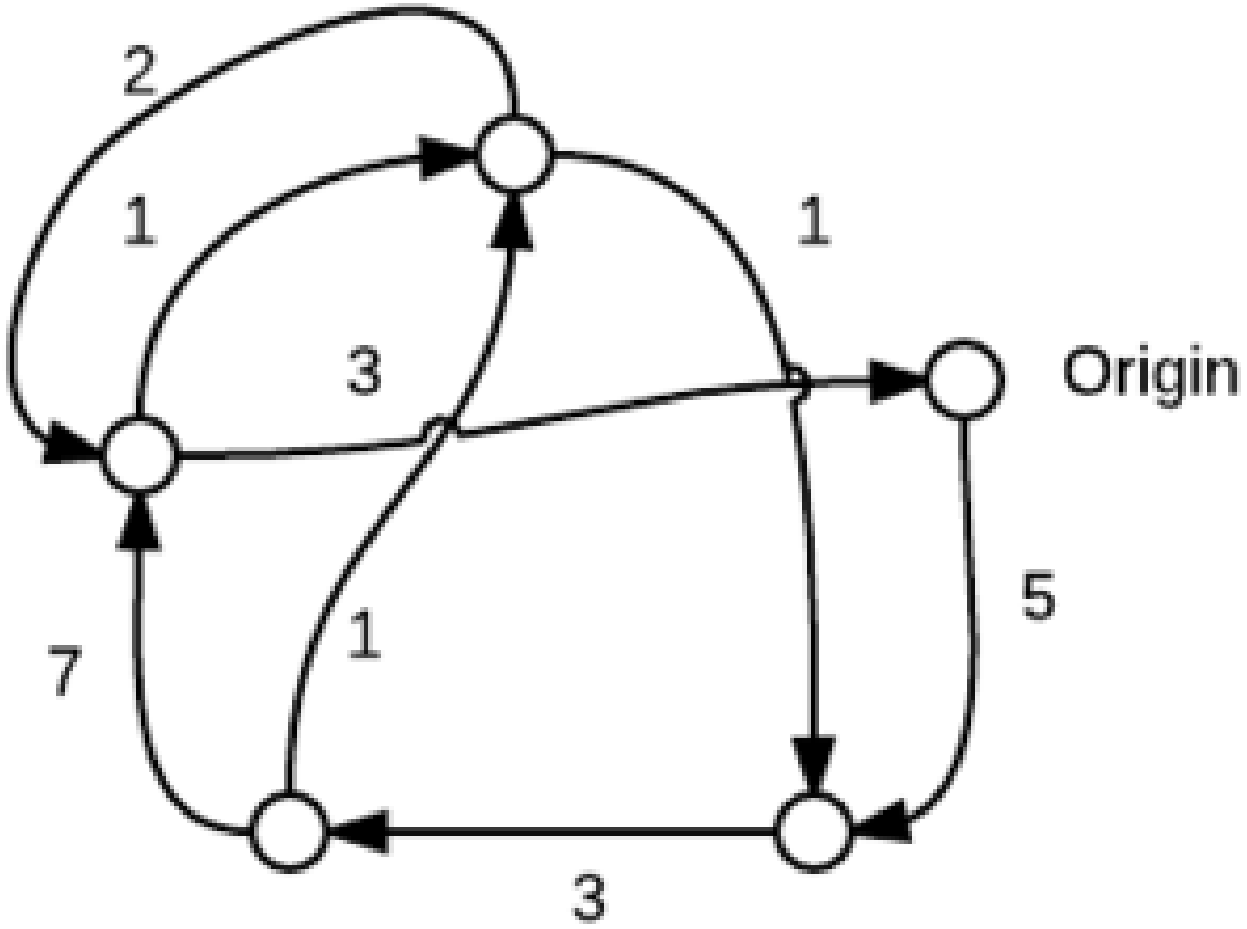
- 5 + 3 + 1 + 1 + 3 + 1 + 2 + 1 + 1 + 3 + 1 + 2 + 3 = 27
- 5 + 3 + 1 + 2 + 1 + 1 + 3 + 1 + 2 + 3 = 22
- 5 + 3 + 1 + 2 + 3 = 14

Total: 3 kali percobaan



Sumber: https://brilliant.org/wiki/traveling-salesperson-problem/

Solusi



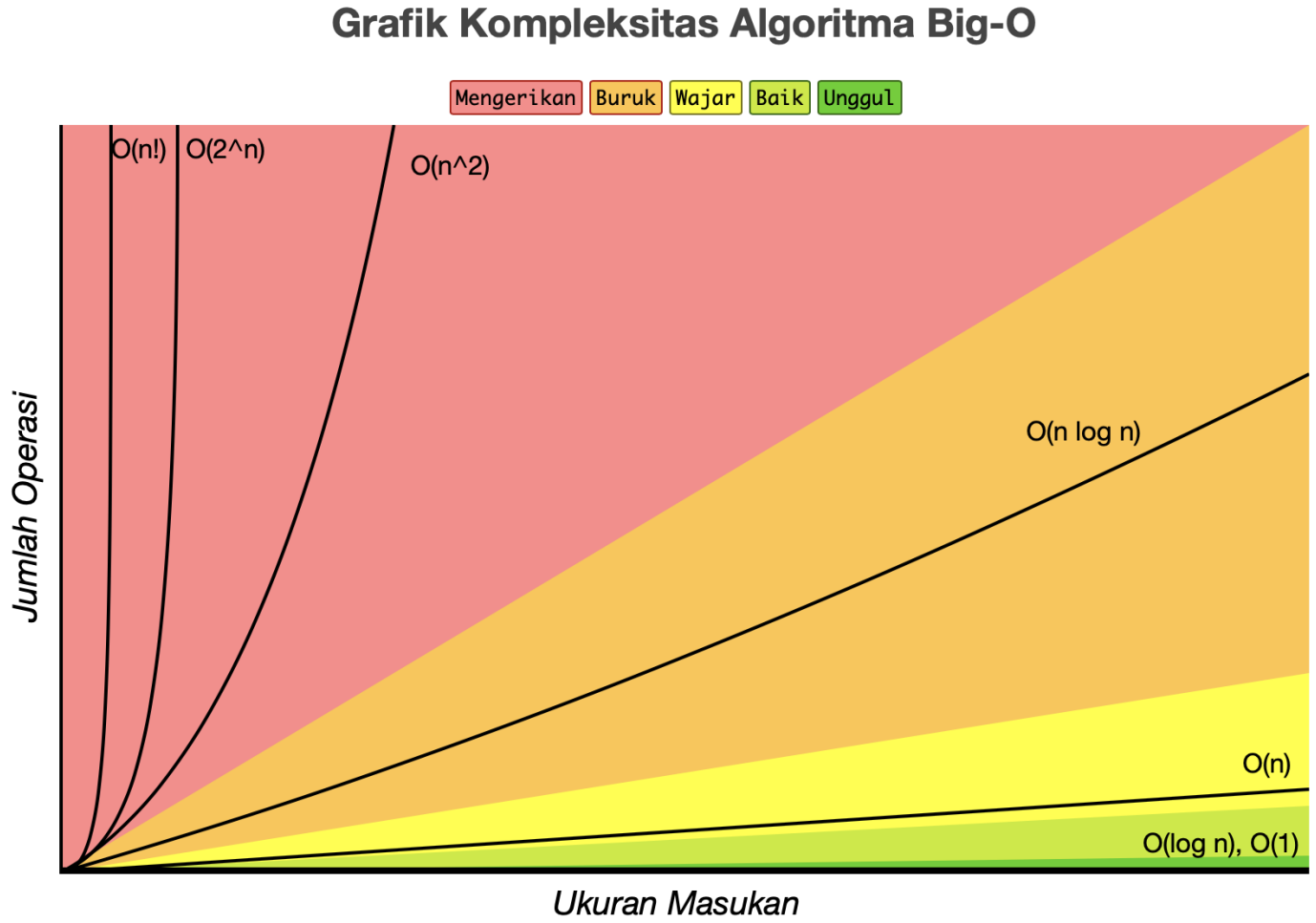
Sumber: https://brilliant.org/wiki/traveling-salesperson-problem/

Traveling Salesperson Problem (TSP)

POLITEKNIK NEGERI JEMBER

- Permasalahan di atas termasuk dalam kategori permasalahan *Traveling Salesperson Problem (TSP)*.
- Ada *(n-1)!* total kemungkinan jalur yang dapat ditempuh oleh *salesperson* tersebut.
- Komputasi yang diperlukan untuk menyelesaikan permasalahan ini tumbuh terlalu cepat seiring dengan semakin banyaknya titik yang harus ditelusuri.
- Jika petanya hanya memiliki lima buah alamat, ada 4!, atau 24 kemungkinan jalur yang dapat ditempuh .
- Namun, jika petanya memiliki 20 buah alamat, akan ada 19! kemungkinan jalur, itu berarti ada $1{,}22 \times 10^{17}$ kemungkinan jalur!

Grafik Kompleksitas Algoritma



Diterjemahkan dari https://www.bigocheatsheet.com

Model Perhitungan Kebutuhan Waktu

POLITEKNIK NEGERI JEMBER

Menghitung kebutuhan waktu algoritma dengan mengukur waktu sesungguhnya (dalam satuan detik) ketika algoritma dieksekusi oleh komputer bukan cara yang tepat.

Alasannya:

1. Setiap komputer dengan arsitektur berbeda mempunyai bahasa mesin yang berbeda → waktu setiap operasi antara satu komputer dengan komputer lain tidak sama.
2. *Compiler* bahasa pemrograman yang berbeda menghasilkan kode mesin yang berbeda → waktu setiap operasi antara compiler dengan compiler lain tidak sama.

Model Abstrak Perhitungan Waktu/Ruang

POLITEKNIK NEGERI JEMBER

- Model abstrak pengukuran waktu/ruang harus independen dari pertimbangan mesin dan *compiler* apapun.
- Besaran yang dipakai untuk menerangkan model abstrak pengukuran waktu/ruang ini adalah **kompleksitas algoritma**.
- Ada dua macam kompleksitas algoritma, yaitu: **kompleksitas waktu** dan **kompleksitas ruang**.

Model Abstrak Perhitungan Waktu/Ruang

POLITEKNIK NEGERI JEMBER

- Kompleksitas waktu, $T(n)$, diukur dari jumlah tahapan komputasi yang dibutuhkan untuk menjalankan algoritma sebagai fungsi dari ukuran masukan n.
- Kompleksitas ruang, $S(n)$, diukur dari memori yang digunakan oleh struktur data yang terdapat di dalam algoritma sebagai fungsi dari ukuran masukan n.
- Dengan menggunakan besaran kompleksitas waktu/ruang algoritma, kita dapat menentukan *laju* peningkatan waktu (ruang) yang diperlukan algoritma dengan meningkatnya ukuran masukan n.

Model Abstrak Perhitungan Waktu/Ruang

POLITEKNIK NEGERI JEMBER

- Ukuran masukan (*n*): jumlah data yang diproses oleh sebuah algoritma.
- Dalam praktek perhitungan kompleksitas, ukuran masukan dinyatakan sebagai variabel *n* saja.

Contoh:

- Algoritma pengurutan 1000 elemen larik, maka *n* = 1000.
- Algoritma *TSP (Traveling Salesperson Problem)* pada sebuah graf lengkap dengan 100 simpul, maka *n* = 100.
- Algoritma perkalian 2 buah matriks berukuran 50 x 50, maka *n* = 50.

Kompleksitas Waktu

POLITEKNIK NEGERI JEMBER

- Jumlah tahapan komputasi dihitung dari berapa kali suatu operasi dilaksanakan di dalam sebuah algoritma sebagai fungsi ukuran masukan (n).

- Di dalam sebuah algoritma terdapat berbagai macam jenis operasi:
  - Operasi baca/tulis
  - Operasi aritmetika (+, -, *, /)
  - Operasi pengisian nilai (*assignment*)
  - Operasi pengaksesan elemen larik (*array*)
  - Operasi pemanggilan fungsi/prosedur
  - dll

- Dalam praktik perhitungan kompleksitas algoritma, kita hanya menghitung jumlah operasi khas (tipikal) yang **mendasari** suatu algoritma.

Contoh Operasi Khas di dalam Algoritma

POLITEKNIK NEGERI JEMBER

Algoritma	Operasi Khas
Pencarian di dalam larik (*array*)	Perbandingan elemen larik (*array*)
Pengurutan	Perbandingan dan pertukaran elemen
Penjumlahan 2 buah matriks	Penjumlahan
Perkalian 2 buah matriks	Perkalian dan penjumlahan

Contoh 1

POLITEKNIK NEGERI JEMBER

Algoritma untuk menghitung rata-rata sebuah larik (*array*)

```
sum ← 0
for i ← 1 to n do
   sum ← sum + a[i]
endfor
rata_rata ← sum/n
```

- Operasi khasnya adalah penjumlahan elemen-elemen a_i (yaitu **`sum ← sum + a[i]`**) yang dilakukan sebanyak *n* kali.
- Kompleksitas waktu: $T(n) = n$.

Contoh 2

Algoritma untuk mencari elemen terbesar di dalam sebuah larik (*array*)

```
max ← a[1]
for i ← 2 to n do
   if a[i] > max then
      max ← a[i]
   endif
endfor
```

- Operasi khasnya adalah perbandingan elemen larik, yaitu **`a[i] > max`**.
- Kompleksitas waktu: $T(n) = n - 1$.

Tiga Macam Kompleksitas Waktu

POLITEKNIK NEGERI JEMBER

Kompleksitas Waktu	Penjelasan
$T_{max}(n)$	Kompleksitas waktu untuk kasus terburuk (*worst case*) → Kebutuhan waktu maksimum
$T_{min}(n)$	Kompleksitas waktu untuk kasus terbaik (*best case*) → Kebutuhan waktu minimum
$T_{avg}(n)$	Kompleksitas waktu untuk kasus rata-rata (*average case*) → Kebutuhan waktu secara rata-rata

Contoh 3

Algoritma untuk mencari indeks suatu elemen *x* di dalam sebuah larik (*array*)

```
i ← 1
found ← false
while (i < n) and (not found) do
  if a[i] = x then
    found ← true
  else
    i ← i + 1
  endif
endwhile
```

```
if found then
  index ← i   // x ditemukan
else
  index ← 0   // x tidak ditemukan
endif
```

- Operasi khasnya adalah perbandingan elemen larik, yaitu **`a[i] = x`**.
- Bagaimana kompleksitas waktunya untuk kasus terbaik, terburuk, dan rata-rata?

Kompleksitas Waktu di Contoh 3

POLITEKNIK NEGERI JEMBER

Jumlah operasi perbandingan elemen tabel:

1. *Kasus terbaik*: ini terjadi bila $a_1 = x$.

   $T_{\min}(n) = 1$

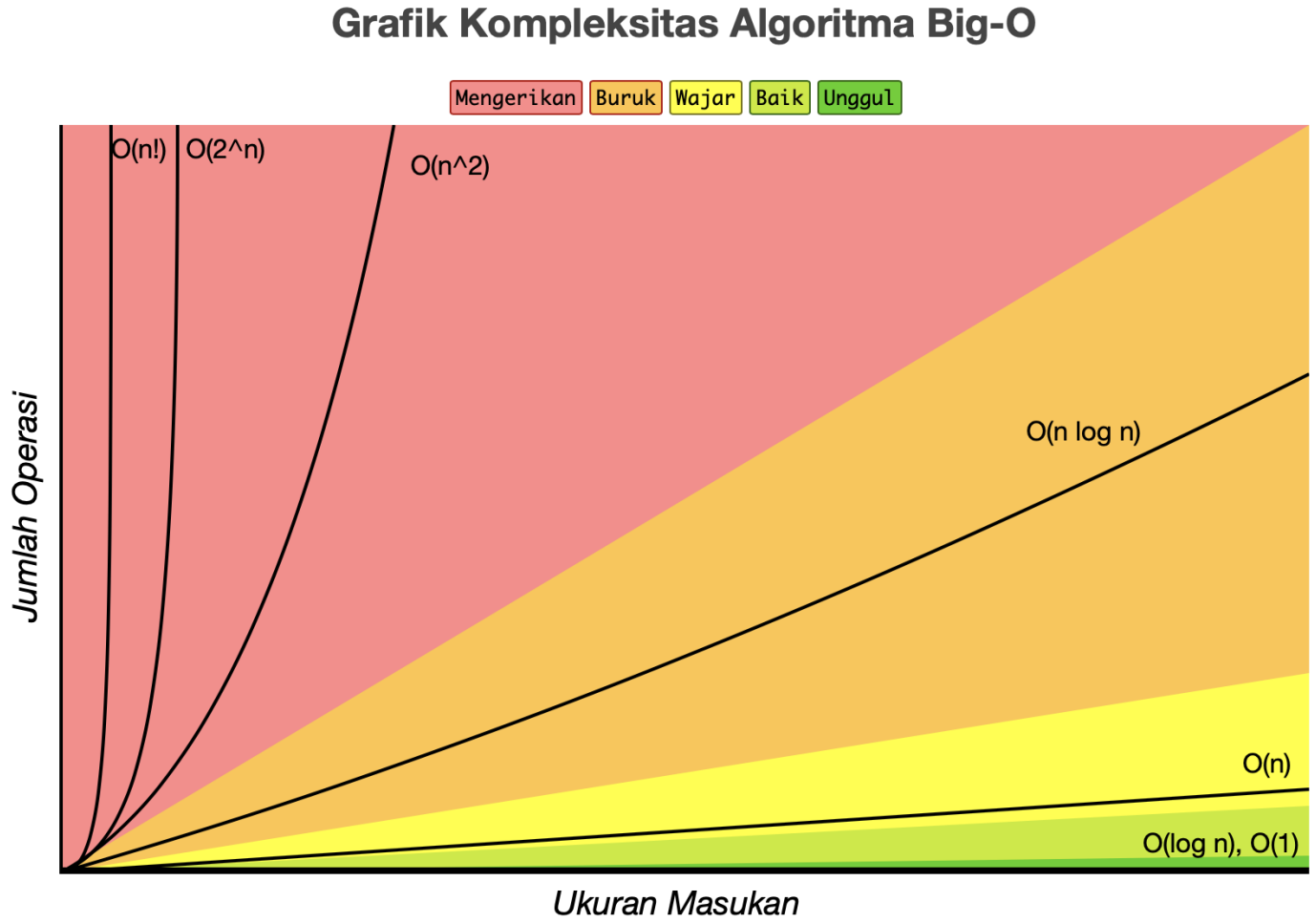
2. *Kasus terburuk*: bila $a_n = x$ atau x tidak ditemukan.

   $T_{\max}(n) = n$

3. *Kasus rata-rata*: Jika x ditemukan pada posisi ke-j, maka operasi perbandingan ($a_k = x$) akan dieksekusi sebanyak j kali.

$$T_{\text{avg}}(n) = \frac{(1+2+3+\ldots+n)}{n} = \frac{\frac{1}{2}n(1+n)}{n} = \frac{(n+1)}{2}$$

Grafik Kompleksitas Algoritma



Diterjemahkan dari https://www.bigocheatsheet.com

Apa itu Big-O?

POLITEKNIK NEGERI JEMBER

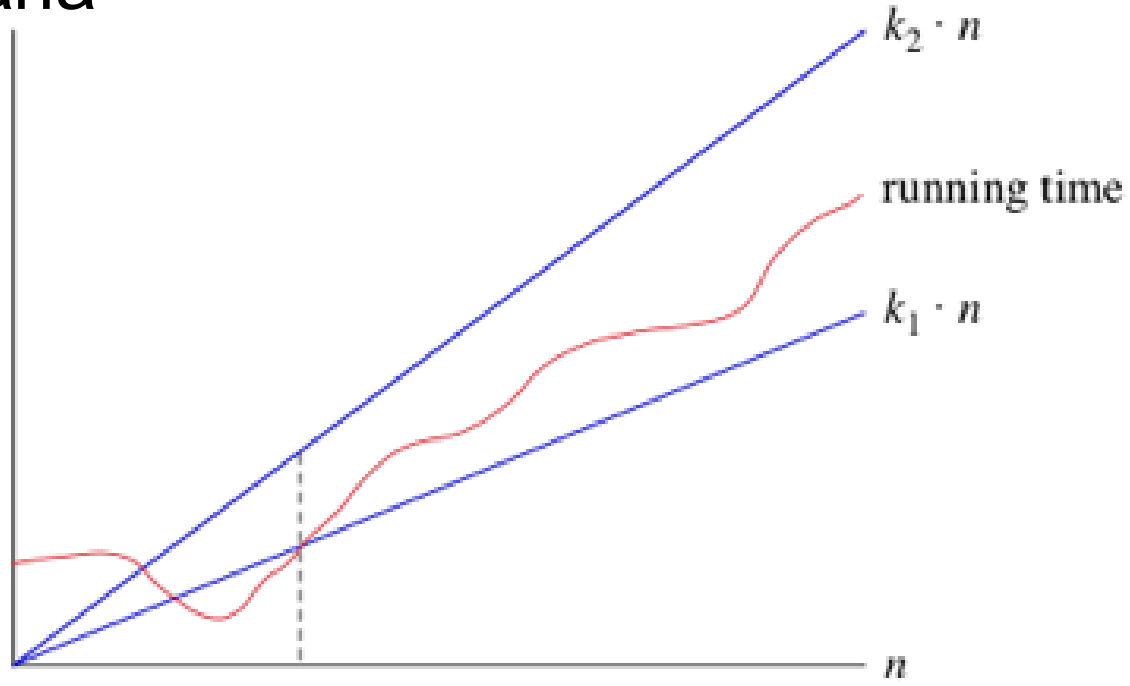
- Big-O adalah notasi matematis yang termasuk kategori **notasi asimtotik** (*asymptotic notation)*.
- Definisi asimtotik = “tidak bertemu”, maksudnya adalah grafiknya tidak bertemu dengan fungsi asal yang diwakilinya
- Notasi asimptotik ini membebaskan kita dari keharusan untuk menghitung fungsi kompleksitas waktu *T(n)* secara presisi.
- Ada tiga macam notasi asimtotik, yaitu:
  - Big-Θ (dibaca Big Teta),
  - Big-O (dibaca Big O), dan
  - Big-Ω (dibaca Big Omega)

Mengenal Big-Θ

POLITEKNIK NEGERI JEMBER

Kompleksitas waktu $\Theta(n)$ artinya:

- untuk nilai n yang cukup besar,
- ada nilai konstanta k_1 & k_2 di mana
- kompleksitas waktu $T(n)$
- diapit oleh fungsi $k_1 \cdot n$ & $k_2 \cdot n$



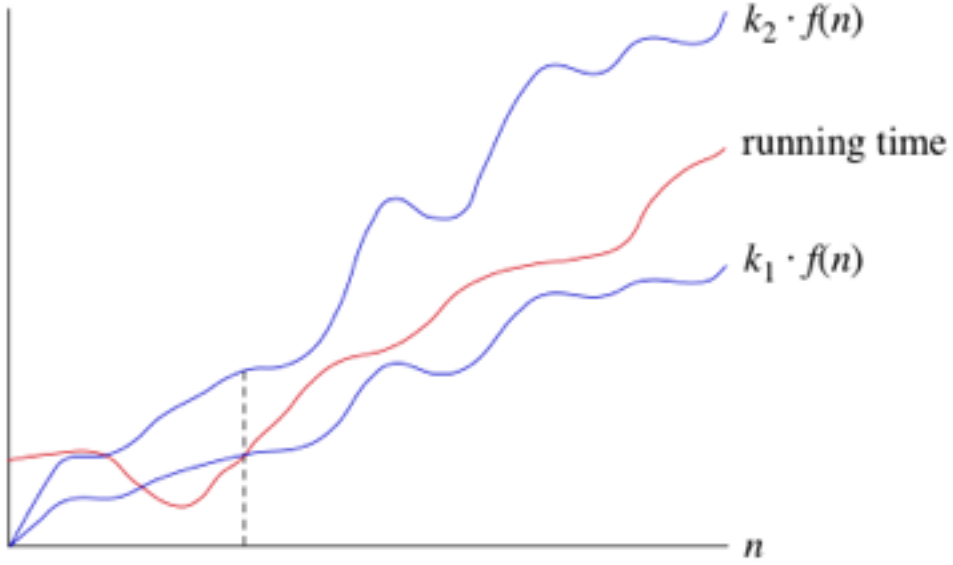
Sumber: https://www.khanacademy.org/computing/computer-science/algorithms/asymptotic-notation/a/big-big-theta-notation

Mengenal Big-Θ

n bisa diganti menjadi fungsi lain

Kompleksitas waktu *Θ(f(n))* artinya:

- untuk nilai *n* yang cukup besar,
- ada nilai konstanta k_1 & k_2 di mana
- kompleksitas waktu *T(f(n))*
- diapit oleh fungsi $k_1 \cdot f(n)$ & $k_2 \cdot f(n)$



Sumber: https://www.khanacademy.org/computing/computer-science/algorithms/asymptotic-notation/a/big-big-theta-notation

Cara Mudah Menghitung Big-Θ

POLITEKNIK NEGERI JEMBER

Buang **konstanta**

Buang suku-suku dengan **orde (pangkat) yang lebih rendah**

Contoh:

- $T(n) = 6n^2 + 100n + 300 \rightarrow \Theta(n) = \cancel{6n^2} + \cancel{100n} + \cancel{300} = n^2$
- $T(n) = 3n^3 + 4n^2 + 300 \rightarrow \Theta(n) = \cancel{3n^3} + \cancel{4n^2} + \cancel{300} = n^3$
- $T(n) = 5 = 5n^0 \rightarrow = \Theta(n^0) = \Theta(1)$

Sederhananya, cukup dengan melihat suku (*term*) yang mempunyai **pangkat terbesar**.

Daftar Fungsi Asimtotik

POLITEKNIK NEGERI JEMBER

- Berhubung banyak unsur yang dibuang dari fungsi asalnya, kemungkinan fungsi yang tersisa jadi semakin sedikit.
- Berikut ini beberapa fungsi asimtotik yang sering muncul ketika kita menganalisis kompleksitas algoritma.

Terurut dari yang paling lambat ke yang paling cepat laju pertumbuhannya:

1. $\Theta(1)$
2. $\Theta(\log n)$
3. $\Theta(n)$
4. $\Theta(n \log n)$
5. $\Theta(n^2)$
6. $\Theta(n^2 \log n)$
7. $\Theta(n^3)$
8. $\Theta(2^n)$
9. $\Theta(n!)$

Mengenal Big-O

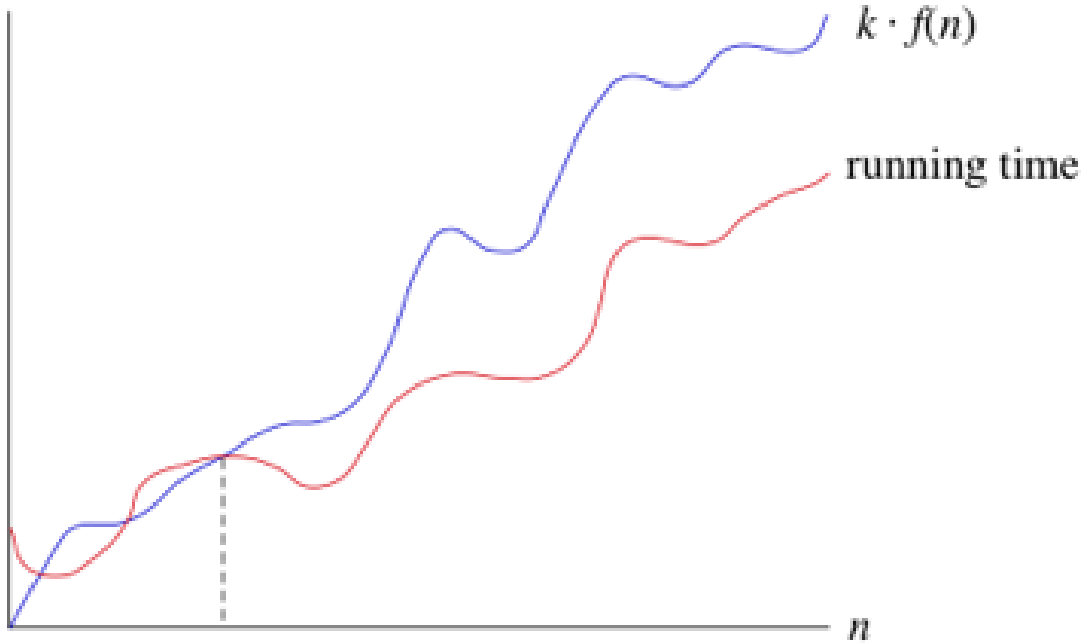
POLITEKNIK NEGERI JEMBER

Kompleksitas waktu *O(f(n))* artinya:

- untuk nilai *n* yang cukup besar,
- ada nilai konstanta *k* di mana
- kompleksitas waktu *T(f(n))*
- selalu lebih kecil daripada $k \cdot f(n)$

Dengan kata lain:

Big-O adalah **batas atas** Big-Θ



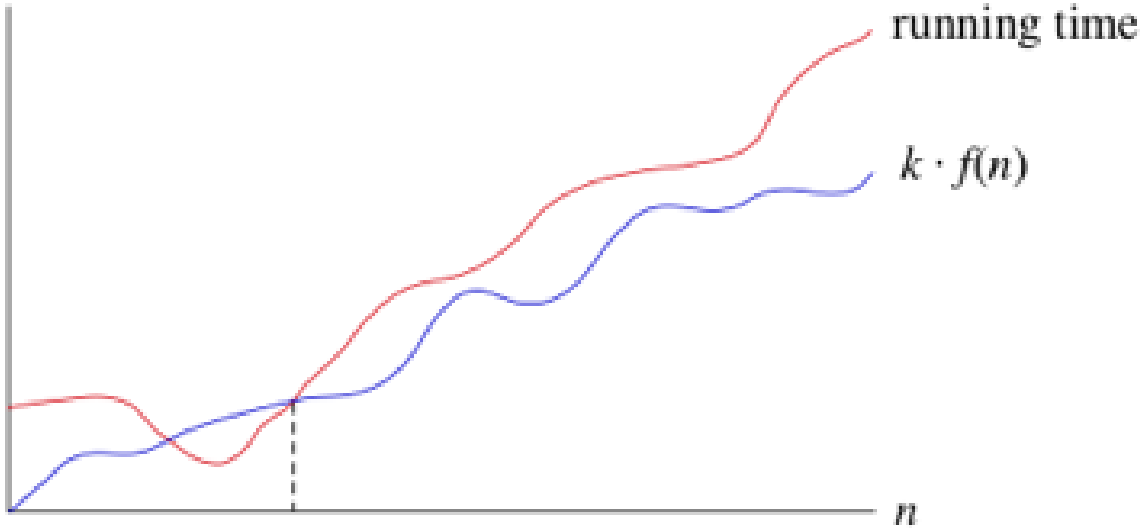
Sumber: https://www.khanacademy.org/computing/computer-science/algorithms/asymptotic-notation/a/big-o-notation

Mengenal Big-Ω

POLITEKNIK NEGERI JEMBER

Kompleksitas waktu *Ω(f(n))* artinya:

- untuk nilai *n* yang cukup besar,
- ada nilai konstanta *k* di mana
- kompleksitas waktu *T(f(n))*
- selalu lebih kecil daripada $k \cdot f(n)$



Sumber: https://www.khanacademy.org/computing/computer-science/algorithms/asymptotic-notation/a/big-big-omega-notation

Dengan kata lain:

Big-Ω adalah **batas bawah** Big-Θ

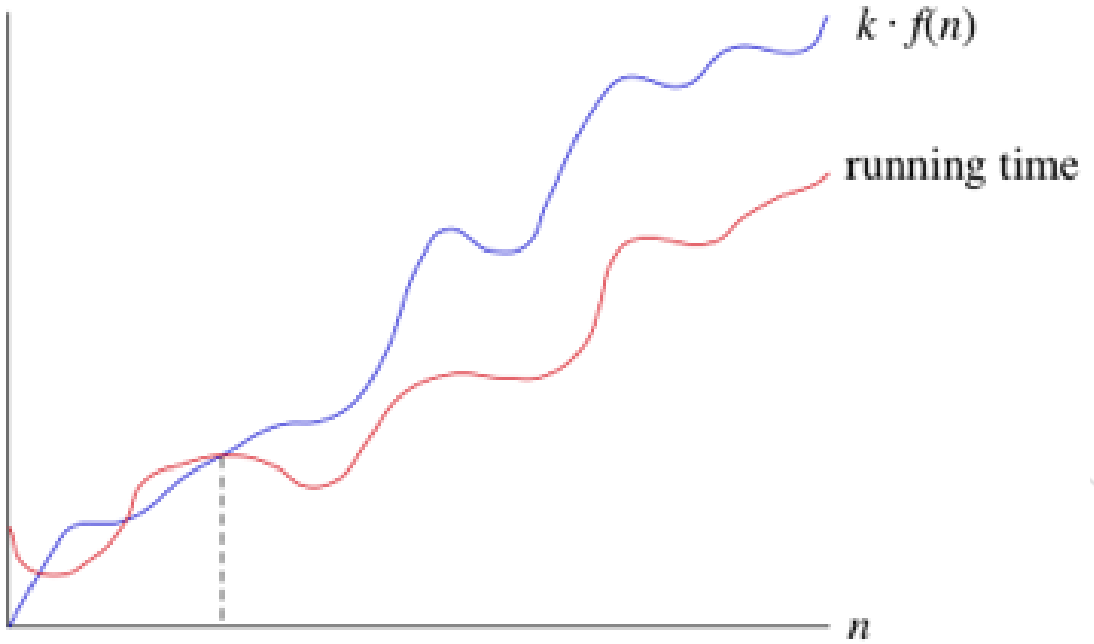
Mengapa Big-O yang Digunakan?

POLITEKNIK NEGERI JEMBER

Karena Big-O mewakili **batas atas** yang berarti selambat-lambatnya algoritma tersebut, tidak akan lebih lambat daripada fungsi Big-O ini.

Ibarat kita hendak pergi jauh, seboros-borosnya saya tidak akan menghabiskan uang lebih banyak dari *n* rupiah.

Sehingga dengan uang saku sebesar *n* rupiah, kita bisa yakin tidak akan kelaparan di perjalanan.



Sumber: https://www.khanacademy.org/computing/computer-science/algorithms/asymptotic-notation/a/big-o-notation

RISTEKDIKTI

POLITEKNIK NEGERI JEMBER

Pertemuan Berikutnya

Berlatih cara menghitung Big-O

RISTEKDIKTI

POLITEKNIK NEGERI JEMBER

Sumber

- https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2008-2009/Kompleksitas%20Algoritma.ppt
- https://www.bigocheatsheet.com
- https://devopedia.org/algorithmic-complexity
- https://brilliant.org/wiki/traveling-salesperson-problem/
- https://zainf.dev/fb-big-o
- https://www.khanacademy.org/computing/computer-science/algorithms/asymptotic-notation/a/asymptotic-notation

RISTEKDIKTI

POLITEKNIK NEGERI JEMBER

Diskusi Lanjutan

Untuk berdiskusi lebih lanjut bersama saya di Discord, silakan bergabung ke sini:

https://zainf.dev/polije

POLITEKNIK
NEGERI JEMBER

Terima Kasih

Politeknik Negeri Jember
Jalan Mastrip Kotak Pos 164 Jember 68101
Telp. (0331) 333532-34
Fax. (0331) 333531

POLITEKNIK
NEGERI JEMBER