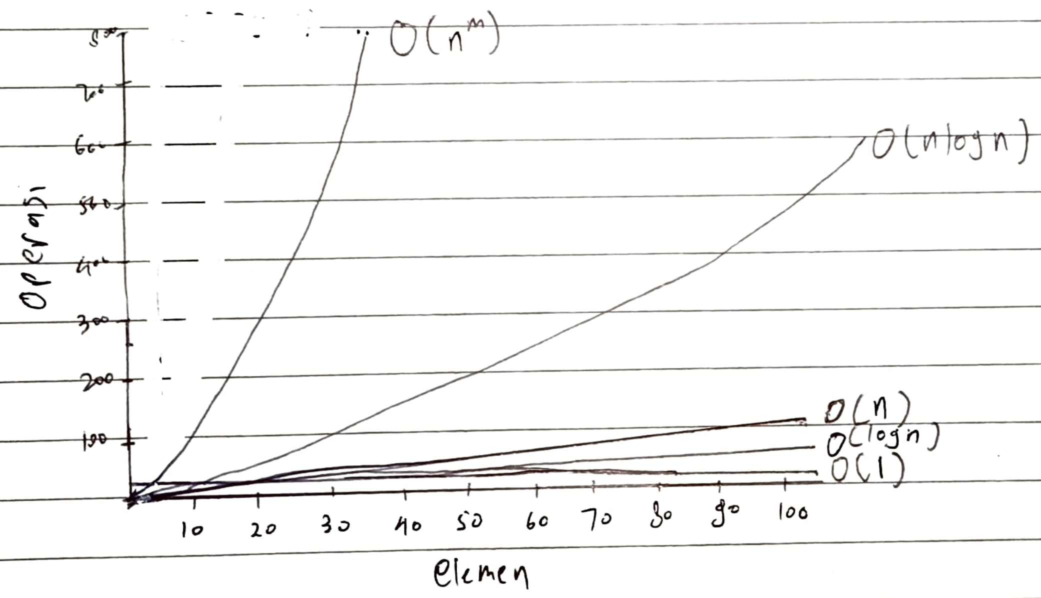
① 5 Kelas Kompleksitas waktu

- O(1) Kompleksitas Konstan
  ↳ Berapun ukuran data yang diterima akan tetap memiliki jumlah langkah yang sama untuk dieksekusi
- O(log n) Kompleksitas logaritma
  ↳ Algoritma dengan kompleksitas ini menyelesaikan masalah dengan membagi masalah menjadi bagian sehingga dapat diselesaikan tanpa harus melakukan komputasi pada seluruh masukan
- O(n) Komplelsitas Linear
  ↳ Bertumbuh selaras dengan pertumbuhan ukuran data
- O(n log n)
  ↳ Algoritma log n yang dijalankan n kali
- $O(n^m)$ Kompleksitas Polinomial
  ↳ Memerlukan jumlah langkah yang jauh lebih besar daripada jumlah data sehingga tidak efisien

Grafik perbandingan



② a.) $T(n) = 2019 + 8\log 10^n = \Omega(n)$

$2019 + 8\log 10^n \geq 8\log 10^n$ untuk semua $n \geq 1$

$2019 + 8\log 10^n \geq 8n \log_{10} 1$

$2019 + 8\log 10^n \geq 8n \rightarrow T(n) \geq C \cdot g(n)$

$C = 8 \quad n_0 = 1$

GELATIK

b.) $T(n) = 10n^3 + 6n^2 + 20n + 14 = \Theta(n^3)$

※ $T(n) = \Theta(h(n))$ jika $O(h(n))$ dan $\Omega(h(n))$

•) $T(n) = 10n^3 + 6n^2 + 20n + 14 = O(n^3)$

$10n^3 + 6n^2 + 20n + 14 \leq 10n^3 + 6n^3 + 20n^3 + 14n^3$ untuk semua $n \geq 1$

$10n^3 + 6n^2 + 20n + 14 \leq 50n^3$

$c = 50$ $\quad n_0 = 1$, maka $T(n) = O(n^3)$

•) $T(n)$ $10n^3 + 6n^2 + 20n + 4 \geq 10n^3$ untuk semua $n \geq 1$

$c = 10$ $\quad n_0 = 1$, maka $T(n) = \Omega(n^3)$

※ Karna $T(n) = O(n^3)$ & $T(n) = \Omega(n^3)$ maka $T(n) = \Theta(n^3)$

c) $T(n) = n + 2n^2 + 4^n = O(2^n)$

$n + 2n^2 + 4^n \leq 2^n + 8^n + 4^n$ untuk semua $n \geq 1$

$n + 2n^2 + 4^n \leq 11(2^n)$

$c = 11$ $\quad n_0 = 1$

$T(n) = n + 2n^2 + 4^n = O(2^n)$

③ a.) $T(n) = 4T(n/2) + n$, $T(1) = 1$

$a = 4$, $b = 2$, $c = 1$

${}^b\log a = {}^2\log 4 = 2$

${}^b\log a > c$

maka $T(n) = \Theta(n^{{}^a\log b}) = \Theta(n^2)$

b.) $T(n) = 4T(n/2) + n^2$, $T(1) = 1$

${}^b\log a = {}^2\log 4 = 2$ $\quad a = 4, b = 2, c = 2$

${}^b\log a = c$

Maka $T(n) = \Theta(n^{{}^a\log b} \log^{k+1} n)$

$= \Theta(n^2 \log n)$

c.) $T(n) = 4T(n/2) + n^3$, $T(1) = 1$

$a = 4$, $b = 2$, $c = 3$

${}^b\log a < c$

Maka $T(n) = \Theta(f(n))$

$= \Theta(n^3)$

④ a.) * Algo 1:

$$T(n) \begin{cases} 0 & , n=1 \\ 1 & , n=2 \\ 3n/2 - 2 & , n > 2 \end{cases}$$

* Algo 2

$$T(n) \begin{cases} 0 & , n=1 \\ 1 & , n=2 \\ 2n-2 & , n > 2 \end{cases}$$

b.) * Algo 1

$$T(n) = \frac{3n}{2} - 2$$

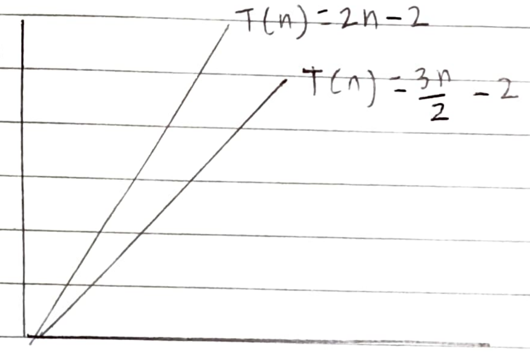
$$= O(n)$$

* Algo 2

$$T(n) = 2n - 2$$

$$= O(n)$$

c.)



d.) Algoritma yang lebih efisien ialah algoritma pertama dikarenakan dapat terlihat pada grafik bahwa Algo 1 memiliki jumlah operasi / fungsi pertumbuhan yang lebih rendah.

$3n/2 - 2 \leq 2n - 2$ untuk semua $n \geq 2$

⑤ a) loop 1 I N F O R M A T I K A
loop 2 F I N O R M A T I K A
loop 3 F I N O R M A T I K A
loop 4 F I N O R M A T I K A
loop 5 F I N M O R A T I K A
loop 6 A F I N M O R T I K A
loop 7 A F I N M O R T I K A
loop 8 A F I I N M O R T K A
loop 9 A F I I K N M O R T A
loop 10 A A F I I K N M O R T

b.) Ya, benar karna Tmin ialah saat best case yaitu array telah terurut. Saat terurut, tidak perlu ditukar sehingga hanya menyelusuri elemennya sebanyak n.

Sedangkan Tmax adalah tiap putaran harus menukar elemen ke depan array, sehingga saat index 8 tukar 8 kali agar elemen yang ditukar bisa di awal array sehingga $T(n) = n \times n = n^2$