linier sering dengan input ukuran pertumbuhan sementara itu, algoritma dengan input ukuran pertumbuhan. Sementara itu, algoritma dengan kompleksitas $O(n)^2$ berarti waktu eksekusinya akan tumbuh secara kuadrik, yang jauh lebih lambat untuk input besar. Notasi besar O mengabaikan konstanta dan faktor-faktor dengan order lebih rendah karena tidak signifikan untuk input yang lebih besar.

Notasi omega besar (Ω) memberikan batas ~~bawah~~ bawah dari kompleksitas algoritma, yang berarti menggambarkan waktu terbaik atau minimal yang selalu di perlukan algoritma terlepas dari kondisi input. Notasi ini berguna untuk mengetahui seberapa cepat algoritma setidaknya dapat berjalan.

Notasi Theta (Θ) memberikan batas ketat yang berarti kompleksitas algoritma berada tepat diantara batas atas dan batas bawah. Sehingga big theta memberikan gambaran paling ~~kurat~~ akurat tentang perilaku asimptotik algoritma. Selain notasi utama ketiga tersebut, ada juga little o dan little omega yang digunakan untuk menggambarkan hubungan yang lebih ketat, yaitu suatu fungsi tumbuh secara strict lebih lambat atau strict lebih cepat dibandingkan fungsi lainnya.

Cara menghitung kompleksitas waktu.
1). Identifikasi operasi dasar yang paling sering dijalankan (perbandingan a/ arithmatika)
2). hitung berapakali operasi tsb dijalankan sebagai fungsi dari ukuran input (n)
3) Sederhanakan ekspresi dengan mengabaikan konstanta dan faktor berorde "kecil"
4) Nyatakan hasil akhirnya dalam notasi Big O.

Struktur kontrol sangat mempengaruhi kompleksitas
- satu loop dari 1 sampai n → $O(n)$
- Nested loop → dikalikan (misalnya dua loop n → $O(n^2)$).
- If - else → mengikuti cabang dengan kompleksitas terbesar
- Rekursi → dianalisis menggunakan relasi rekurensi.

Nama : Hasnidar. P.
Nim : D022A321
Kelas : Informatika C2A

Analisis algoritma merupakan salah satu cabang fundamental dalam ilmu komputer yang mempelajari cara menghasilkan efisiensi suatu algoritma. Analisis proses ini meliputi pengukuran sumber daya yang diperlukan oleh algoritma ketika dijalankan, terutama dalam hal waktu eksekusi dan penggunaan memori. Dengan memahami analisis algoritma. Seorang Programer dapat memilih sosuli yg paling optimal untuk menyelesaikan suatu masalah, memprediksi kinerja program ketika data yang di olah semakin besar. serta menghindari implementasi algoritma yang tidak efisien. Analisis algoritma juga menjadi dasar penting dalam pengambilan keputusan desain sistem. karena pemilihan algoritma yang tepat dapat memberikan perbedaan kinerja yang sangat signifikan. terutama untuk aplikasi yang memproses data dalam skala besar.

Terdapat dua pendekatan utama dalam menganalisis efisiensi algoritma.
1). Analisis waktu (kopleksitas waktu)
kompleksitas waktu mengukur berapa lama waktu yang di butuhkan algoritma untuk menyelesaikan masalah sebagai fungsi dari ukuran input.
2). Analis ruang (kopleksitas ruang)
kompleksitas ruang mengukur berapa banyak memori yang di perlukan algoritma selama eksekusinya.

Notasi asimptotik
Notasi asimptotik merupakan bahasa matematika yang digunakan untuk mendeskripsikan perilaku suatu fungsi ketika nilai inputnya mendekati nilai tertentu, biasanya menuju tak hingga.

Notasi O Besar (O) merupakan notasi yang paling digunakan dalam analisis algoritma. notasi ini memberikan batas-batas dari kompleksitas waktu atau ruang algoritma. yang berarti menggambarkan waktu terburuk yang mungkin diperlukan algoritma untuk input berukuran tertentu. misalnya. algoritma dengan kopleksitas O(n) berarti waktu eksekusinya akan tumbuh secara kuadratik.