Nama : William Pradana Putra

NPM : 22552011121

Kelas : TIF RM 22A

Matkul : Struktur Data

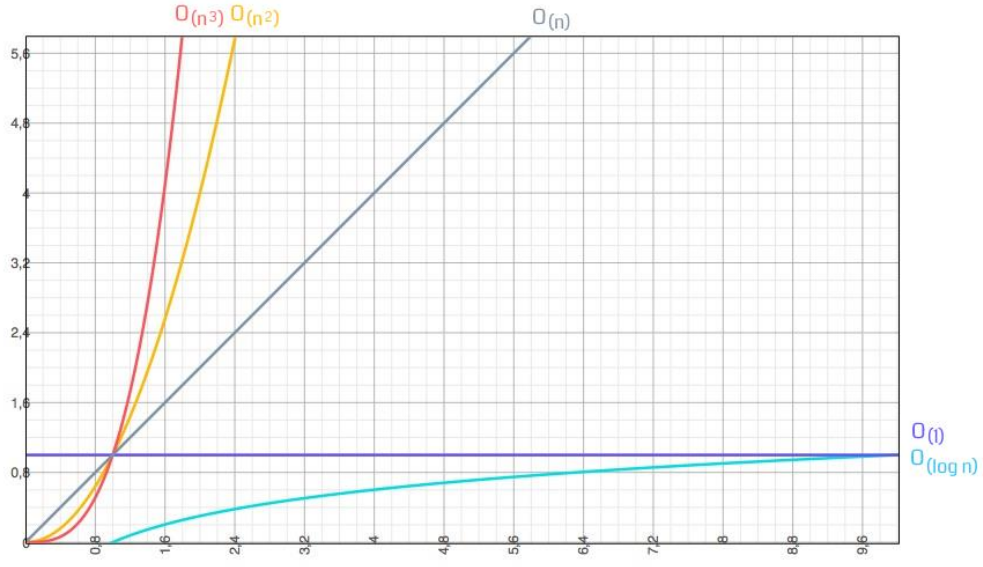
Big O Notation

Sekilas Tentang Big O Notation

Notasi O besar atau yang lazim disebut dengan Big-O Notation adalah sebuah cara atau metode untuk melakukan analisa terhadap sebuah algoritma pemrograman terhadap waktu eksekusi. Tentang seberapa efisien dan kompleksitas barisan kode dalam dimensi waktu.

Di dalam sebuah program komputer pada umumnya, kita sudah lazim dengan istilah masukan-proses-keluaran.

Notasi O besar merupakan skenario terburuk dari sebuah algoritma, dan biasanya terdapat notasi n yang merepresentasikan jumlah masukan. Berikut adalah diagram notasi O besar dengan masukan yang dimulai dari 0 hingga tak terhingga



Dari diagram diatas dapat kita lihat beberapa notasi yang kerap muncul, yaitu: O(1), O(log n), O(n), O(n2) atau

O(nn). Untuk membahasnya, mari kita berandai-andai membangun sebuah aplikasi travel. Dan kita diminta untuk membuat fitur baru yang menampilkan daftar kisaran harga dari hotel-hotel yang berada di area tertentu.

Untuk membuat fitur tersebut, sederhananya kita mencari harga terendah (min) dan harga tertinggi (max) dari daftar hotel. Tentunya semakin banyak datanya, akan semakin lama pula proses untuk mencari harga terendah dan tertinggi.

Dan berikut versi sederhana dari representasi data yang akan kita gunakan. Kita akan menggunakan JavaScript sebagai contoh. Dapat juga diaplikasikan ke bahasa pemrograman lain.

```
const hotels = [
{ price: 180, brand: "Hotel Tugu Lombok" },
{ price: 78, brand: "Sheraton Senggigi Beach Resort" },
...
...
{ price: 317, brand: "The Oberio" }
]
```

Pengulangan di Dalam Pengulangan: O(N2)
Sekarang, bagaimana caranya mencari harga terendah dan tertinggi? Cara yang paling naif adalah dengan membandingkan harga satu-per-satu. Berikut potongan kode sebagai ilustrasi.

```
for (let i = 0; i < hotels.length; i++) {
  for (let j = 0; j < hotels.length; j++) {
    // kode untuk membandingkan satu harga dengan harga lainnya...
  }
}
```

Mari kita simulasikan ketika jumlah data semakin banyak.

n	3	5	10	100	
Jumlah Operasi		9	25	100	10000

Lihat tabel diatas, antara 3 dan 5 cukup dekat perbedaannya hanya dua. Namun jumlah operasi yang dijalankan sangat signifikan bedanya. Semakin banyak datanya semakin signifikan jumlah operasi yang dijalankan.

Dengan kata lain kita membutuhkan n2 dan untuk mencari harga terendah (min) dan harga tertinggi (max). Itulah sebabnya kita menyebutnya dengan notasi O(n2). Algoritma seperti ini sangatlah lambat dan tidak optimal.
Pengulangan Dari Sebuah Set: O(N)

Jika solusi pertama tidak optimal, adakah solusi lain? Tentu. Misalnya kita bisa mencari apakah harga adalah yang terendah atau tertinggi dalam satu kali pengulangan saja seperti ilustrasi kode berikut.

```
for (let i = 0; i < hotels.length; i++) {
  // cari harga paling kecil...
  // cari harga paling besar...
}
```

Ketika kita membutuhkan proses "pengulangan untuk setiap item", merupakan notasi O(N).

Hanya Satu Operasi: O(1)
Jika dengan asumsi bahwa data hotels sudah diurutkan berdasarkan harga terendah ke harga tertinggi, maka pencarian harga terendah dan tertinggi menjadi semakin efisien.

```
const hotels = [
  { price: 78, brand: "Sheraton Senggigi Beach Resort" },
  { price: 180, brand: "Hotel Tugu Lombok" },
  ...
  ...
  { price: 317, brand: "The Oberio" }
]
```

Untuk mendapatkan harga terendah, kita tinggal memanggil hotels[0].price. Sementara untuk harga tertinggi kita bisa menggunakan hotels[hotels.length-1].price. Selain lebih mudah, eksekusi baris kode juga menjadi lebih cepat dan efisien.

Ketika kode menjalankan sebuah operasi, misalkan: baris kode untuk mencari item barang teratas atau mendapatkan nilai sebuah array dengan indeks ke-5. Operasi seperti ini kita bisa sebut sebagai notasi O(1).

Tidak masalah sebanyak apapun isi dari sebuah array atau sebanyak apapun jumlah baris di basis data, untuk mengambil nilai array dengan mengakses indeks maka algoritma berjalan secara konstan atau constant time.

Kita sudah melihat beberapa contoh algoritma untuk menampilkan kisaran harga dari yang terendah hingga tertinggi dan hasilnya sebagai berikut.

Notasi O	Istilah Lain	Jumlah Operasi	Algoritma
$O_{(n^2)}$	Quadratic	n^2	Komparasi seluruh harga. Pengulangan dalam pengulangan
$O_{(n)}$	Linear	2n	Mencari harga terendah dan tertinggi. 1 kali pengulangan
$O_{(1)}$	Constant	2	Asumsi sudah diurut berdasarkan harga, tinggal mencari elemen pertama dan elemen terakhir

Dan secara umum berikut urutan dari notasi O besar diurutkan dari yang tercepat hingga yang terpelan.

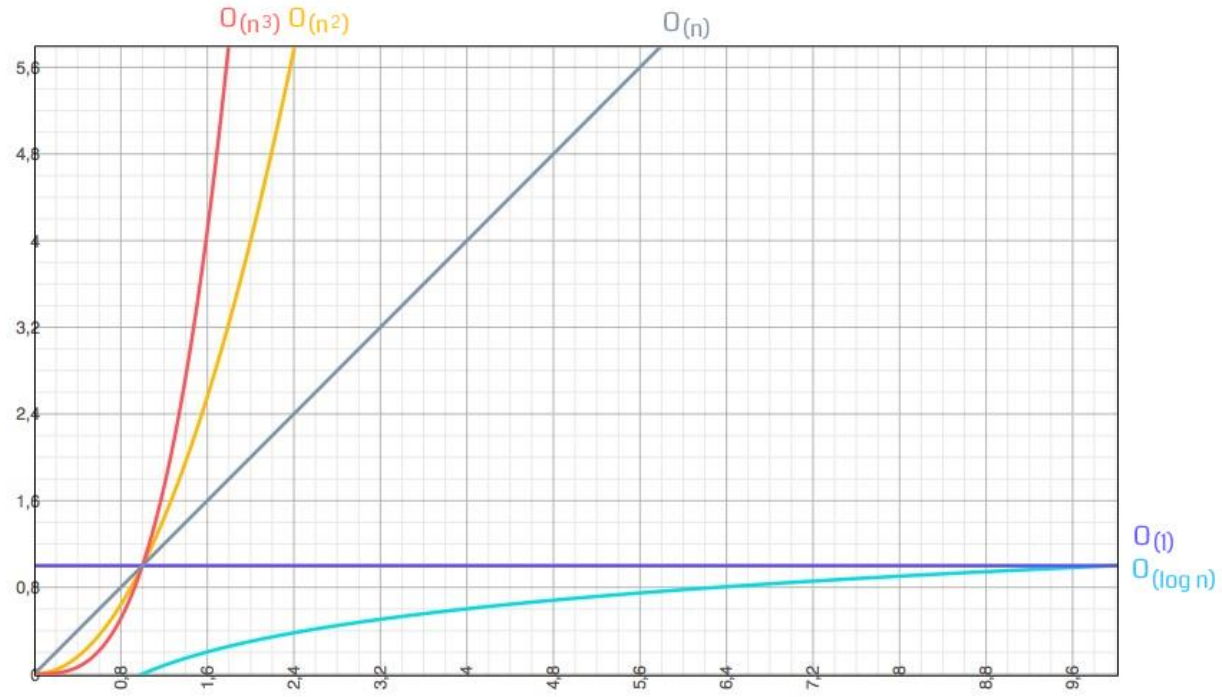
<--- Super Cepat ---------- Super Lambat --------->

Nama	Constant	Logaritmic	Linear	Quadratic	Exponential
Notasi	$O_{(1)}$	$O_{(\log n)}$	$O_{(n)}$	$O_{(n^2)}$	$O_{(n^n)}$

Dan secara umum berikut urutan dari notasi O besar diurutkan dari yang tercepat hingga yang terpelan.

<--- Super Cepat ---------- Super Lambat --------->

Nama Constant Logaritmic Linear Quadratic Exponential
Notasi O(1) O(log n) O(n) O(n2) O(nn)



Beberapa Contoh Notasi O Besar

Mari kita melihat contoh notasi O besar dari fungsi, ekspresi dan operasi JavaScript yang sederhana.

1. Array.push()
push() merupakan sebuah metode untuk menambahkan item baru kedalam sebuah array. Item yang ditambahkan akan berada diakhir array tersebut. Contoh penggunaan dapat dilihat sebagai berikut.

```
const animals = ['ants', 'goats', 'cows'];
animals.push('fish');
console.log(animals); // ['ants', 'goats', 'cows', 'fish']
```

Apakah notasi yang tepat untuk baris kode animals.push('fish');? Karena metode push() tidak peduli dengan seberapa banyak atau sedikit jumlah item yang ada, artinya operasi yang berjalan tetap sama, maka metode push() ini dapat diwakilkan dengan notasi O(1) atau konstan.

2. Array.pop()
pop() merupakan sebuah metode yang mengambil item terakhir dari array sehingga jumlah item yang ada di array akan berkurang satu. Berikut contoh penggunaannya.

```
const plants = ['broccoli', 'cauliflower', 'cabbage', 'tomato'];
plants.pop();
console.log(plants); // ["broccoli", "cauliflower", "cabbage"]
```

Apakah notasi yang tepat untuk baris kode plants.pop();? Mirip seperti metode push() diatas, metode pop() juga tidak mempermasalahkan jumlah item yang ada, artinya operasi yang berjalan tetap sama, maka metode pop() ini juga dapat diwakilkan dengan notasi O(1) atau konstan.

3. Array.unshift()
unshift() adalah sebuah metode untuk menambahkan satu atau beberapa item ke bagian awal dari sebuah array. Contoh penggunaannya sebagai berikut.

```
const array1 = [1, 2, 3];
array1.unshift(4, 5);
console.log(array1); // [4, 5, 1, 2, 3]
```

Sekilas operasi unshift() ini terlihat seperti operasi yang konstan seperti push() dan pop() namun jika kita melakukan implementasi ulang metode ini, maka akan terlihat notasi yang sebenarnya. Berikut kira-kira implementasi dari unshift(), implementasi naif tentunya sekedar gambaran.

```
function unshift(arr, newItem) {
  let newArr = [];
  newArr[0] = newItem;
  for (let i = 1; i < arr.length + 1; i++) {
    newArr[i] = arr[i - 1];
  }
  return newArr;
}
```

Hal yang menambah kompleksitas adalah ketika kita harus mengubah indeks dari array karena kita akan menempatkan item baru di indeks ke-0. Secara otomatis indeks akan bergeser sebanyak satu langkah. Dan karena itu kita menggunakan pengulangan for hingga menjadikan operasi unshift() dapat diwakilkan oleh notasi linear atau O(n). Kita harus menyadari apa yang dilakukan oleh sebuah fungsi, operasi ataupun pustaka sehingga kita dapat memprediksi kira-kira seberapa tingkat kompleksitasnya.