

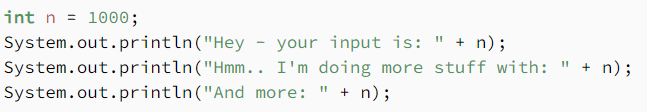
Nama : Muamar

NPM : 22552011119

Mata Kuliah : Struktur Data

Kelas : TIF – RM 22A

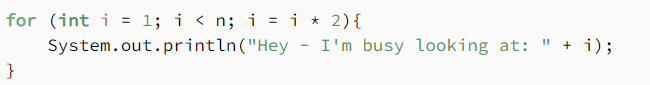
**1. O(1) — Constant Time**

****

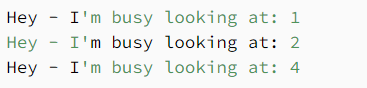
Contoh diatas, terdapat sebuah fungsi untuk mengambil elemen pertama dari sebuah input array. Kita bisa melihat bahwa berapapun jumlah array yang diberikan kepada fungsi tersebut, dia akan selalu melakukan 1 hal, yaitu mengambil elemen pertama. Itu artinya **jumlah input yang diberikan tidak mempengaruhi waktu proses (***runtime***) dari algoritma tersebut**.

1. **O(log n) — Logarithmic Time**

Yang penting di sini adalah bahwa waktu berjalan bertambah sebanding dengan logaritma input (dalam hal ini, log ke basis 2):

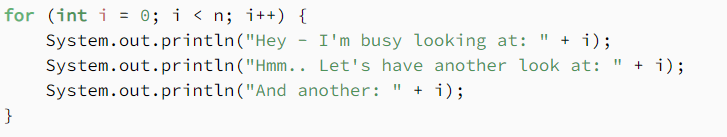


Jika n adalah 8, outputnya adalah sebagai berikut:

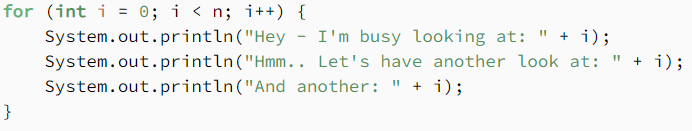


**3. O(n) — Linear Time**

Jika kita mengatakan sesuatu tumbuh secara linier, yang kita maksudkan adalah bahwa itu tumbuh berbanding lurus dengan ukuran inputnya.



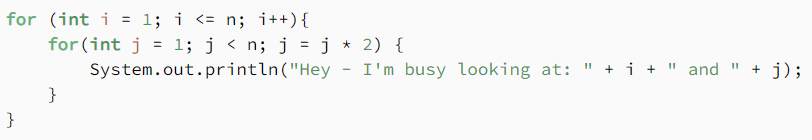
Kita ketahui bahwa algoritma sederhana yang disajikan di atas akan tumbuh secara linier dengan ukuran inputnya.



Runtime akan tetap linier dalam ukuran inputnya, n.

**4. **N Log N Time Algorithms –**O(n log n)**

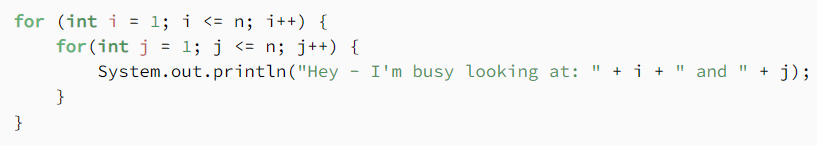
n log n adalah kelas algoritma lanjut. Waktu berjalan bertambah sebanding dengan n log n input:



Misalnya, jika n adalah 8, maka algoritma ini akan menjalankan 8 \* log(8) = 8 \* 3 = 24 kali.

## ****Polynomial Time Algorithms –**O(np)**

Istilah polinomial adalah istilah umum yang mengandung fungsi kuadrat (n2), kubik (n3), kuartik (n4), dll. Yang penting untuk diketahui adalah bahwa O(n2) lebih cepat dari O(n3) yang lebih cepat dari O(n4), dll.

****

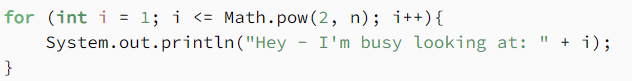
Algoritma ini akan berjalan 82= 64 kali. Catatan, jika kita membuat sarang untuk loop lain, ini akan menjadi algoritma O(n3).

## ****Exponential Time Algorithms –**O(**kn**)**

Algoritma ini tumbuh secara proporsional dengan beberapa faktor yang dieksponenkan oleh ukuran input.

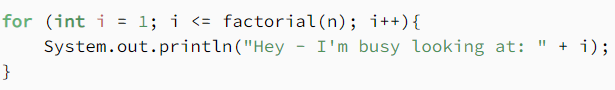
Misalnya, algoritma O(2n) berlipat ganda dengan setiap input tambahan. Jadi, jika n = 2, algoritma ini akan berjalan empat kali; jika n = 3, mereka akan berjalan delapan kali (seperti kebalikan dari algoritma waktu logaritmik).

Contoh sederhana dari algoritma waktu O(2n).



## ****Factorial Time Algorithms –**O(n!)**

Kelas algoritma ini memiliki waktu berjalan yang sebanding dengan faktorial dari ukuran input.



di mana faktorial(n) cukup menghitung n!. Jika n adalah 8, algoritma ini akan menjalankan 8! = 40320 kali.