

Algoritma Analizi – Big O (Büyük O) Hesaplama

Big O nedir?

Algoritma Analizi işleminde kullanılan en önemli kavramlardan biri de Big O notasyonudur. Türkçeye Büyük O olarak çevirdiğimiz bu kavram bir fonksiyonun büyümesini ifade eder.

Basit bir şekilde matematiksel olarak anlatmaya çalışırsak f ve g fonksiyonları reel sayı olursa;

$f(x)$ fonksiyonu c ve k sabit olmak üzere büyük O fonksiyonu $O(g(x))$ şeklinde tanımlanır.

$$\bullet |f(x)| \leq c|g(x)| + k$$

$x > k$ olmalı...

Matematiksel ifadeler kafanızı karıştırabilir. Gelin örnek üzerinde anlatalım.

Big O nedir?



$f(x) = x^2 + 2x + 1$ fonksiyonunun büyüme fonksiyonunun x^2 (x kare) olduğunu gösterelim.

- İlk olarak $x > 1$ için;
- $x^2 + 2x + 1 \leq x^2 + 2x^2 + x^2$
- $x^2 + 2x + 1 \leq 4x^2$
- $C = 4$ ve $k = 1$ için;
- $f(x) \leq Cx^2 + k$, $x > k$ durumunun sağlandığını görürüz.

Big O nedir?



Eğer $f(x)$ için $O(x^2)$ ise aynı zamanda $O(x^3)$ diyebiliriz. Ancak bizim niyetimiz en küçük büyüme fonksiyonunu bulmaktır.

Big O nedir?

$O(1)$

İçerik [Gizle]

$O(1)$

$O(n)$

$O(nc)$

$O(\text{Log}n)$

Birden Fazla döngü var ve iç içe değilse

$O(1)$ ile ifade edilen fonksiyonun büyüme sayısı 1'dir. Bunun kod olarak karşılığı döngü içermeyen, tek seferlik karar yapısı bulunan yazılımlardır. Yani `if(n == 3)` satırı tahmin edeceğiniz üzere 1 kere çalışır. Doğal olarak da Big O değeri $O(1)$ olarak ifade edilir. Yine aynı mantıkla değişken atama, switch case gibi ifadelerde de $O(1)$ geçerlidir. Ezber yapmanıza gerek yok bir kod parçasına baktığınızda "burası kaç kere çalışır" diye bir soru sormanız yeterli.

Big O nedir?

O(n)

Big O değeri n olan ifadeler genelde döngülerdir. n nedir derseniz, n eleman sayısının karşılığıdır. Bir döngü n kere dönüyorsa o döngünün algoritma karşılığı O(n)'dir diyebiliriz.

```
for (int i = 1; i <= n; i++) {  
    // herhangi bir atama işlemi  
}
```

Yukarıdaki döngü tahmin edeceğiniz üzere n kez döner. yani n kez işlem yapar. Burada dikkat etmeniz gereken nokta for döngüsünün içerisindeki işlemlerin sonucu değiştirebileceğidir. Biz döngü içerisinde başka döngülerin bulunmadığı durum için O(n) diyoruz.

Big O nedir?

$O(n^c)$

Karmaşıklığı n üzeri c olan ifadeler iç içe döngülerin karmaşıklığıdır. n en içteki döngünün dönme sayısıdır. c ise üstteki döngünün dönme sayısıdır.

```
for (int i = 1; i <= n; i += c) {  
    for (int j = 1; j <= n; j += c) {  
  
    }  
}
```

Yukarıdaki döngü $O(n^2)$ karmaşıklığa sahiptir. n^2 karmaşıklık matris işlemlerinde sıklıkla karşımıza çıkmaktadır.

Big O nedir?

$O(\text{Log}n)$

Eğer bir kodda sürekli olarak bir çarpım ya da bir bölüm işlemi yapılıyorsa bu durumda $\text{log}n$ karmaşıklığı oluşur. Daha sonra bahsedeceğimiz divide and conqueror mantalitesine sahip fonksiyonlarda $O(\text{Log}n)$ karmaşıklığı görülmektedir.

```
for (int i = 1; i <= n; i /= c) {  
  
}
```

Yukarıdaki kod parçasığında i 'nin her döngüde c değerine bölündüğüne dikkat edin. Bu işlem adımını oldukça azaltan bir faktördür. (çarpıldığında tersi olur tabii)



Algoritma Karmaşıklığı ne kadar düşükse o kadar iyidir. Program hızlı çalışır demektir.

Big O nedir?

Birden Fazla döngü var ve iç içe değilse

Eğer bir yazılımda birden fazla döngü var ve bu döngüler iç içe değilse algoritma karmaşıklığını hesaplamak için bu döngülerin karmaşıklıkları toplanır.

.

```
for (int i = 1; i <=m; i += c) {  
  
}  
for (int i = 1; i <=n; i += c) {  
  
}
```

Yukarıdaki Döngülerin karmaşıklıkları $O(m) + O(n)$ 'dir yani $O(m+n)$ olur. (döngülerin dönme sayılarına dikkat edin.