ZAMAN KARMAŞIKLIĞI

Bir algoritmanın işlemcide yönetilmesi için gerçekleştirdiği sürenin analiz edilmesine "zaman karmaşıklığı" denir.

Selection Sort

$\frac{n\cdot(n-1)}{2}$ karşılaştırma

işlem sayısı = karşılaştırma

Merge Sort

$n\cdot\log n-n+1$ işlem sayısı

 # Deney1

128 adet veri var diyelim

Bir karşılaştırma işlemi 10^{-6} sn

$n=128=2^7$ adet veri

Bottom up için;

$=n.\log n-n+1$

$=(128.\log 128-128+1).10^{-6}$

$\cong 0,0008sn$

Selection sort için;

$=\left(\frac{n.(n-1)}{2}\right).10^{-6}$

$=\left(\frac{128.(128-1)}{2}\right).10^{-6}$

$\cong 0,008sn$

Selection sort 10 kat daha yavaş

 # Deney 2

$n=2^{20}$ lik bir veri için

$n=2^{20}=1048576$

Bottom up için;

$=n.\log n-n+1$

$=(2^{20}.\log 2^{20}-2^{20}+1).10^{-6}$

$\cong 20sn$

Selection sort için;

$=\left(\frac{n.(n-1)}{2}\right).10^{-6}$

$=\left(\frac{2^{20}.(2^{20}-1)}{2}\right).10^{-6}$

$\cong 6,4gün$

sn – gün birimlere dikkat

ALGORİTMA ANALİZİ (ZAMAN ANALİZİ)

Önemli 2 husus

1. Diğer algortimalar ile karşılaştır
2. (Her zaman) geçerli bir gösterim olmalı
3. Veri artışına göre zaman analizi nasıl değiştiğini ölçebileceğin ifadelere ihtiyaç var.

Asimptotik Yaklaşım

A algoritmasında cn^2 çalışıyorsa c ihmal edilebilir

B algoritmasında cn^3 çalışıyorsa c çok büyük verilerde ihmal edilebilir.

$f(x) = n^2 \log n + 10n^2 + n$ böyle bir durumda n'nin çok büyük değerlerinde $n^2 \log n$ baz alınabilir.

Asimptotik çalışma zamanı

Fonksiyonda küçük değerleri ihmal eder

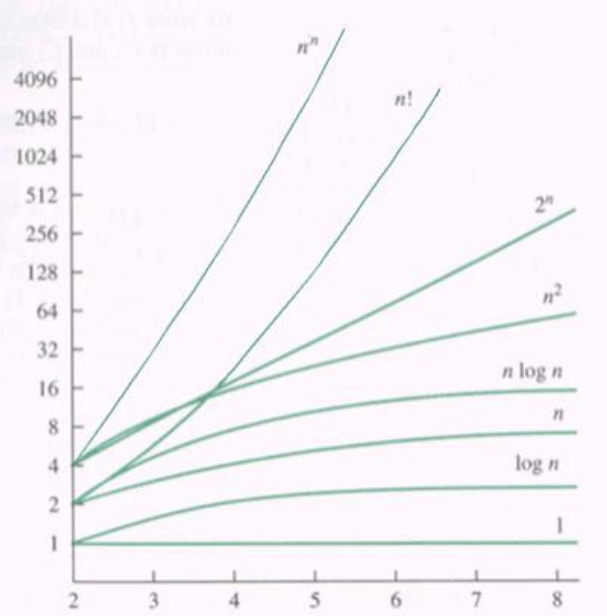
Örnek:

A algoritması $5n^2 + 3n - 7$ çalışma zamanı

B algoritması $9n^2 + 60n - 100$ çalışma zamanı

A ve B algoritması aynı zamanda karmaşıklığa sahip n^2

Büyük O		**Karmaşıklığın değişimi**
O(1)	En iyi	Değişmiyor
O(logn)		Logaritmik artıyor
O(n)		Doğrusal artıyor
O(nlogn)		Doğrusal çarpanlı logaritmik
$O(n^2)$		Karesel artıyor
$O(n^3)$		Kübik artıyor
O(n!)		Faktöriyel olarak artıyor
$O(2^n)$		İki tabanından üssel artıyor
$O(10^n)$	En Kötü	On tabanında üssel artıyor



En kötüden – En iyiye sıralama

$n^n - n! - 2^n - n^2 - n\log n - n - c$

Big O Gösterimi (notasyonu)

Matematiksel fonksiyonların asimptotik yaklaşımını ifade etmek için kullanılan bir notasyon. Üst sınır hakkında bilgi verir.

```
def listedeki eleman sayısı (aranan liste)
{
        for eleman in liste
                if eleman==aranan return true;
        return false;
}
```

#Max karşılaştırma sayısı : Listenin eleman sayısı, zamanı en çok eleman sayısı etkiler BigO ya göre; zaman analizi O(n)'dir. En iyi durum arananın ilk elemanda olması O(1)

Insertion Sort için
En iyi durum;
Dizinin sıralı olması
$T(n) = Cn - C$ **BigO** O(n)

En kötü durum;
Dizi tersten sıralıdır
$T(n) = C\frac{n^2}{2} - C\frac{n}{2}$ **BigO** $O(n^2)$

BigO Tanım

$f(n) \quad ve \quad g(n) \quad N \to R^+ \; \forall \, n \geq n_0 \quad için$

Eğer $f(n) \leq (Cg(n)) \quad n > 0$

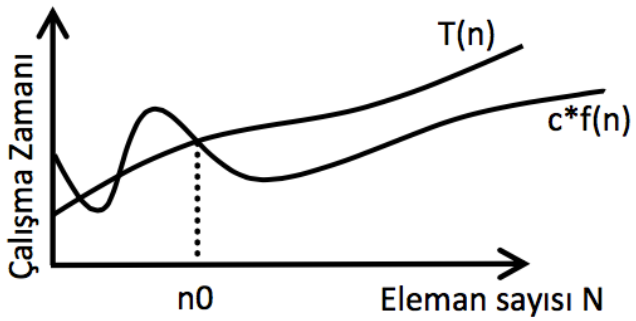
$f(n) = O(g(n))$ // $g(n), f(n)$ *için üst sınırdır*

$\lim_{n\to\infty} \frac{f(n)}{g(n)}$ $n \infty$'*dan farklı* // $f(n) = O(g(n))$ diyebiliriz.

Ω Omega Gösterimi (notasyonu)

Alt sınır hakkında bilgi verir. Üst sınır belirtilmeden (bazı durumlarda) algoritma en az şu kadar adım çalışır dememiz gerekebilir. Burada devreye Ω gösterimi girer.

Bir algoritmanın çalışma zamanı $\Omega(f(n))$ ise yeterince büyük bir n için k bir sabit olmak üzere çalışma zamanı en az $k(f(n))$ dir



Verinin sayısına ve türüne göre değişen bir durum söz konusu, bize alt sınırı gösterir

Insertion Sort için
En iyi durum;
Dizi zaten sıralıdır, dıştaki döngü çalışır O(n)
Ω(n) dir.

BigO gibi değişmiyor her durumda Ω(n)'dir

Ω Tanım

$f(n)$ *ve* $g(n)$ aynı fonksiyon türü $N \to R^+ \; \forall \, n \geq n_0 \quad c > 0$ *ise bu durumda*

$f(n) \geq c.g(n)$ *bu şartlar sağlanıyorsa*

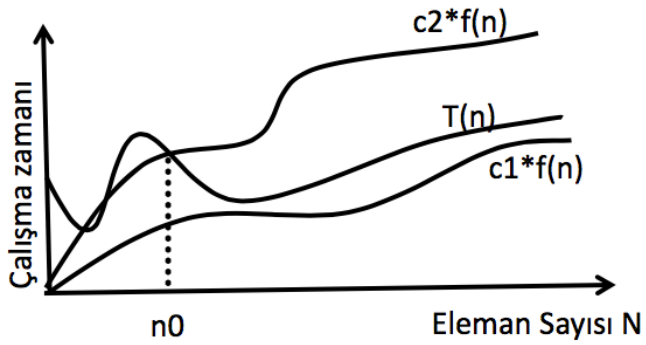
$f(n) = \Omega(g(n))$

$\lim_{n\to\infty} \frac{f(n)}{g(n)}$ $n \infty$'*dan farklı* $\lim_{n\to\infty} \frac{f(n)}{g(n)} \neq 0$ // $f(n) = \Omega(g(n))$ diyebiliriz.

Θ Gösterimi (notasyonu)

Çalışma zamanı hakkında tam bilgi verir.

BigO	O	üst sınır
Omega	Ω	alt sınır
Theta	Θ	Her ikisini verir



Buble Sort için	$\Theta(n^2)$	Her durumda n^2 çalışma zamanı
Selection sort için	$\Theta(n^2)$	Her durumda n^2 çalışma zamanı
BottomUp	$\Theta(n.logn)$	

Θ Tanım

$f(n) \quad ve \quad g(n) \quad N \to R^+ \; \forall \, n \, için \quad n \geq n_0$

C_1 ve C_2 $\quad C_1 > 0 \quad C_2 > 0$

$$C_1 g(n) \leq C_2 f(n)$$

$$f(n) = \Theta(g(n))$$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} \quad n \infty' dan \; farklı \quad \lim_{n\to\infty} \frac{f(n)}{g(n)} \neq 0$$

BigO	$f(n) = O(g(n))$
Omega	$f(n) = \Omega(g(n))$
Theta	$f(n) = \Theta(g(n))$

Örnek:

$f(n) = 10n^2 + 20n$

$n=n_0$

n=1 için $f(1) = 10 \cdot 1^2 + 20 \cdot 1 = 30 \cdot 1^2$

n=2 için $f(2) = 10 \cdot 2^2 + 20 \cdot 2 = 30 \cdot 2^2$ 80<120

n=3 için $f(3) = 10 \cdot 3^2 + 20 \cdot 3 = 30 \cdot 3^2$ 150<270

30'u C sabiti olarak alırsak;

$$f(n) \leq C \cdot g(n)$$
$$10 \cdot n^2 + 20 \cdot n \leq 30 \cdot n^2$$
$$10 \cdot n^2 + 20 \cdot n \leq 1 \cdot n^2$$

BigO $O(n^2)$ üst sınır
Ω $\Omega(n^2)$ alt sınır

$$\Omega(n^2) \leq \overset{?}{f(n)} \leq O(n^2)$$
$$1(n^2) \leq \underset{\Theta(n^2)}{f(n)} \leq 30(n^2)$$

$\Theta(n^2)$ dir diyebiliriz
Her zaman sağlayabileceği durum

Örnek

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

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

Örnek

$$\sum_{j=1}^{n} \log j = \log 1 + \log 2 + \cdots + \log n$$

$$\log 1 + \log 2 + \cdots + \log n \leq \log n + \log n + \cdots + \log n$$

$$\sum_{j=1}^{n} \log j \leq n \cdot \log n$$ // O(nlogn)

$$\log n + \log n + \cdots + \log n \geq \log \frac{n}{2} + \log \frac{n}{2} + \cdots + \log \frac{n}{2}$$

$$\sum_{j=1}^{n} \log j \geq n \cdot (\log n - \log 2)$$

O(nlogn)
Ω(nlogn)
Θ(nlogn)

Örnek

$f(n) = \log n!$

$\log n! = \log(n \cdot (n-1) \cdot (n-2) \cdots 1)$

$\log n! = \log n + \log(n-1) + \log(n-2) + \cdots \log 1$

$\log n! = \sum_{j=1}^{n} \log j$

O(nlogn)
Ω(nlogn)
Θ(nlogn)

Küçük o Gösterimi (notasyonu)

Genellikle teorik bir hesaplamadır

Tanım

$f(n)$ ve $g(n)$ $N \to R^+$ $\forall\, n\, için$ $n > n_0$ $c > 0$ olmak üzere

$f(n) < c \cdot (g(n))$ ise

$f(n) = o(g(n))$

	Notasyonlar	Karşılaştırma notasyonu	lim Gösterimi
o	$f \in o(g(n))$	$f < g$	$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$
O	$f \in O(g(n))$	$f \leq g$	$\lim_{n\to\infty} \frac{f(n)}{g(n)} < \infty$
Θ	$f \in \Theta(g(n))$	$f = g$	$\lim_{n\to\infty} \frac{f(n)}{g(n)} \in R^+$
Ω	$f \in \Omega(g(n))$	$f \geq g$	$\lim_{n\to\infty} \frac{f(n)}{g(n)} > 0$

ALAN KARMAŞIKLIĞI (BELLEK KARMAŞIKLIĞI)

Bir algoritma tarafından kullanılan alan, Giriş verisini tutmak için kullanılan alan hariç tutulur

Problemin çözüm adımları için kullanılan bellek alanı

Selection Sort Alan analizi

O(1) tüm işlemler dizi üzerinde olur

Bottom up Algoritması Alan analizi

O(n)

Sıralı Arama Alan analizi

O(1)

BİR ALGORİTMA NASIL ANALİZ EDİLİR?

1. Adımları sayarak analiz
2. Temel işlem frekansına göre (en fazla hangi işlem yapılıyorsa)
    a. Arama Algoritmalarında (***Karşılaştırma***)
    b. Matris Çarpımı (***Skaler Yönlü Çarpma***)
    c. Bağıl Liste Dolaşma (***Pointer Aritmetiği, Temel İşlem***)
    d. Bir graf dolaşmak (***Düğüm Ziyareti***)
3. Rekürsif bağlantısını kurulması

Faktöriyel Hesabı

```
fak(n)
        if n=0 return 1
        return n*fak(n-1)
```

n=0 ise T(0)=1

n=1 ise T(1)=1

⋮

n=3 ise fak(3)

↳ 3.fak(2)

↳ 2.fak(1)

↳ 1.fak(0)

3 Çarpma
2 Çıkarma
3 Karşılaştırma

1 çağrımda; 1 karşılaştırma
1 çarpma
1 çıkarma
} 3 işlem

T(n) = T(n-1)+3 // Her adımda 3 işlem

T(n-1) = T(n-2)+3+3

⋮

T(n) = T(n-k)+3k+1

T(n) = T(n-n)+3n // T(0)=1

T(n) = 3n+1

Faktöriyelde en iyi en kötü durumdan bahsetmiyoruz
O(n) Ω(n) Θ(n)

Faktöriyel Alan Karmaşıklığı

n=5 ise fak(5)

↳ 5.fak(4)

↳ 4.fak(3)

↳ 3.fak(2)

↳ 2.fak(1)

↳ 1.fak(0)

BELLEK	
	Son eleman
f(0)	***RTS***
f(1)	*Son giren*
f(2)	*ilk çıkar*
f(3)	*n+1' lik yer*
f(4)	*açıldı*
f(5)	İlk eleman

O(n)'lik bir alan maliyeti