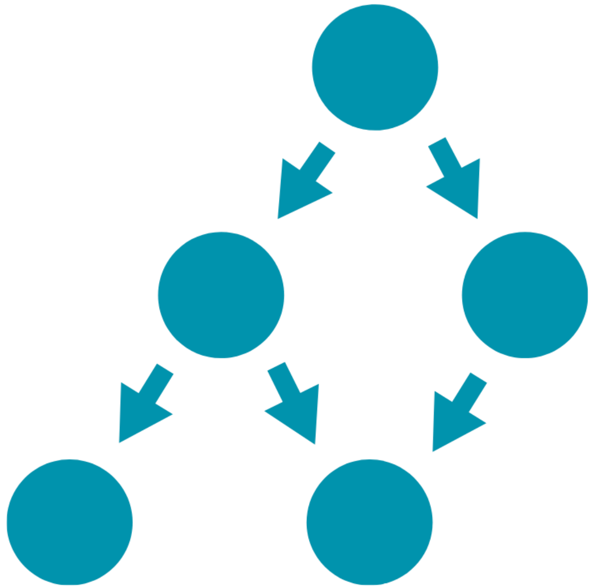
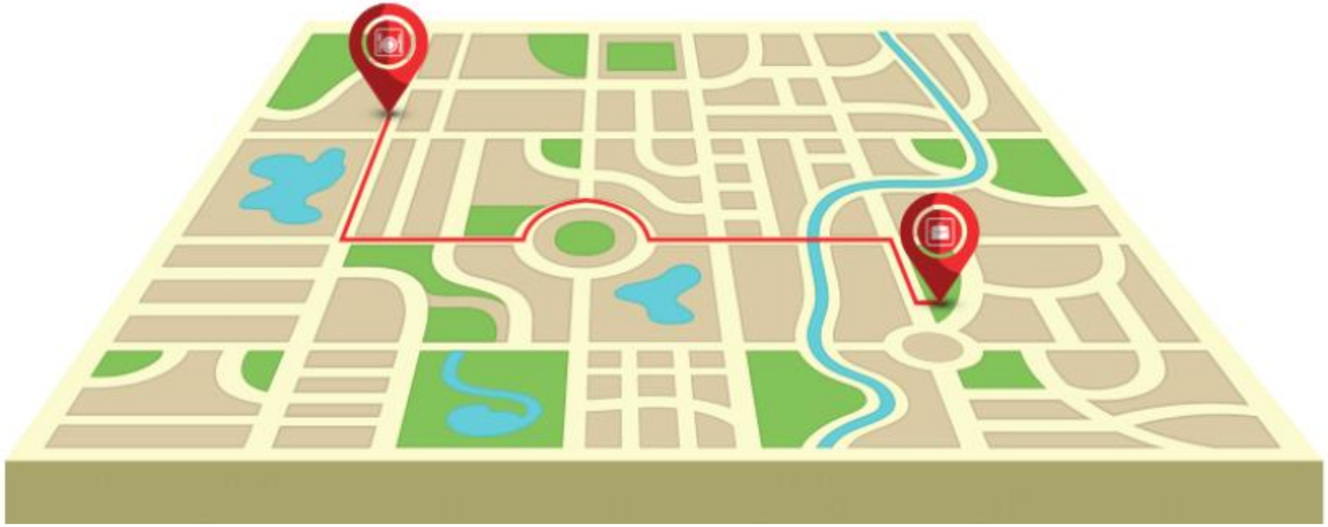


VERİ YAPILARILARI VE ALGORİTMALAR

Algorithm Analysis and Asymptotic Notations

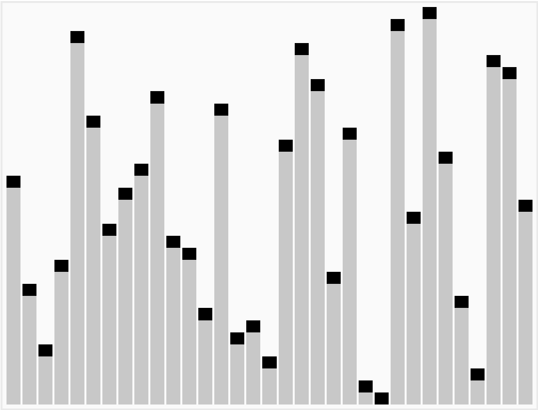
Big-Oh



Algoritmaları neden analiz ederiz?

Big-Oh

- SIRALAMA
  - Insertion
  - Selection
  - Quicksort
  - Binary
  - Bubble



Algoritmaları neden analiz ederiz?

Big-Oh

$$|f(n)| \leq c * |g(n)|, tüm\ n \geq n_0$$

- Pozitif tam sayılardan pozitif tam sayılara kadar $f(n)$ ve $g(n)$ monotonik fonksiyonlar için $c > 0$ ve $n_0 > 0$ sabitleri olduğunda $f(n) = O(g(n))$ şekliden ifade edilir.

Big-Oh

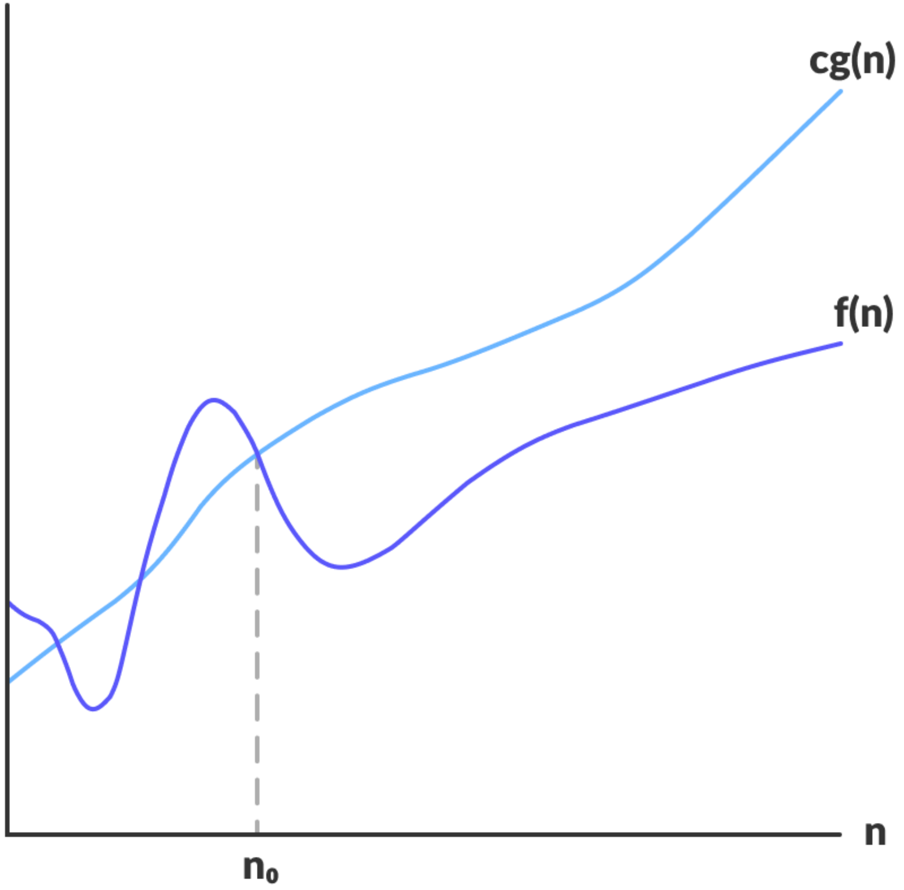
$$f(n) \leq c * g(n), \text{tüm } n \geq n_0$$

- Sezgisel olarak, bu $f(n)$ fonksiyonunun $g(n)$'den daha hızlı büyümediği veya $g(n)$ fonksiyonunun $f(n)$ için, yeterince büyük olan $n \rightarrow \infty$ için bir üst sınır olduğu anlamına gelir.

Big-Oh

$$f(n) = O(g(n))$$

- $f(n) = O(g(n))$ ilişkisinin grafiksel gösterimi



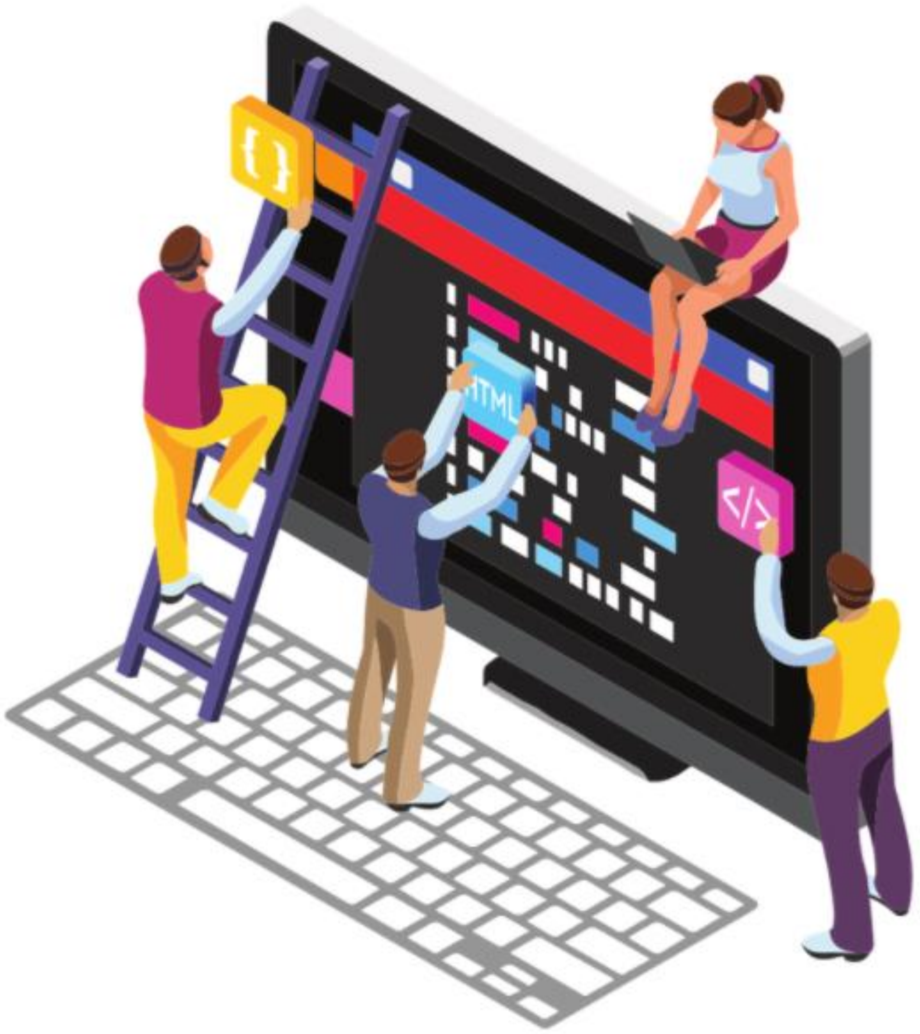
f(n) = O(g(n))

Çalışma Zamanı Analizi

- Çalışma zamanı giriş boyutunun boyutuna bağlı olarak artar.

- Giriş boyutu n'e bağlı olarak varsayım yapılmamalıdır.

- n her zaman küçük olmayabilir.



Çalışma Zamanı Analizi

- **Büyüme Hızı (Rate of Growth)**: Girdinin bir fonksiyonu olarak çalışma süresinin artma hızı.

- **Alt Sıradan Terimler (Lower Order Terms)**: Bir fonksiyonun büyüme oranına ilişkin bir tahmin verildiğinde, daha yüksek dereceli şartlar için daha az önemli olduklarından, düşük dereceden terimleri düşürme eğilimindeyiz.

$$f(n) = n^3 + \cancel{} + \cancel{}$$

$$O(n^3)$$

Lower Order Terms in Layman’s Terms

600.000

600.532

Sabit faktörler ve düşük dereceli terimler gibi ikinci dereceden ayrıntıları gizlemek ve girdi boyutu büyüdükçe bir algoritmanın çalışma süresinin nasıl ölçeklendiğine odaklanmak istiyoruz.

Big-Oh

1. Giriş boyutuna bağlı karmaşıklık
2. Makineden bağımsız
3. Temel bilgisayar adımları

Big-Oh

Ölçüm Türleri

1. En kötü durum (worst-case)
2. En iyi durum (best-case)
3. Ortalama durum (average-case)

Big-Oh

Genel Kurallar

1. Sabitler ihmal edilir (**constant factor**).

   $T(n) = 5n + 3 \quad => \quad O(n)$

   $T(n) = 10n + 99 \quad => \quad O(n)$

   $T(n) = 1000n \quad => \quad O(n)$

Big-Oh

Genel Kurallar

2. Baskın terim dikkate alınır.

O(1) < O(logn) < O(n)

$O(n^2+5n+100) \Rightarrow O(n^2)$

Örnekler

Fonksiyon	Big O
$n^4 + 100n^2 + 10n + 50$	$O(n^4)$
$10n^3 + 2n^2$	$O(n^3)$
$n^3 - n^2$	$O(n^3)$
10	$O(1)$
1273	$O(1)$

Big-Oh

$$f(x) = x^2 + 2x + 1 \text{ is } O(x^2)$$

$$x^2 + 2x + 1 \le C.x^2 \quad \text{when} \quad x > k$$

$$0 \le x^2 + 2x + 1 \le x^2 + 2x^2 + x^2 = 4x^2$$

Big-Oh

$$f(x) = x^2 + 2x + 1 \text{ is } O(x^2)$$

$$x^2 + 2x + 1 \text{ is } O(x^3)$$

Önceki versiyonu tercih ederiz.

Big-Oh

$$n!\ \text{ifadesinin}\ O(n^n)$$

olduğunu gösterelim.

$$\therefore n! \le C.n^n \ bazı\ n > k$$

$$\therefore 1.2.3.\cdots.n \le n.n.n.\cdots.n$$

$$C = 1\ \ ve\ k = 1\ \ \text{olduğunda}\ n! = O(n^n)$$

Big-Oh

$$f(n) = n^2,\ O(n) \ \ olmaz!$$

$$\therefore n^2 \leq C.n \text{ for some } n > k$$

$$\therefore \frac{n^2}{n} \leq \frac{C.n}{n}$$

$$\therefore n \leq C$$

Burada n *değişkendir* ve C ise bu sabittir.

Doğrusal değildir.

Big-Oh

Eğer $f_1(n)$ -> $O(g_1(n))$ and $f_2(n)$ -> $O(g_2(n))$ ise

$\mathbf{f_1(n)*f_2(n)}$ is $\mathbf{O(g_1(n)*g_2(n))}$

Big-Oh

(3n+1)*(2n+log n)

İlgili örneğin derecesi nedir?

$3n+1 \;\; \rightarrow O(n)$

$2n+\log n \;\; \rightarrow O(n)$

$(3n+1)*(2n+\log n) \rightarrow O(n*n)=O(n^2)$

Big-Oh

n – Giriş boyutu
Karmaşıklığını küçükten büyüğe doğru sıralanması.

Sabit Zaman:	O(1)	
Logaritmik Zaman:	O(log(n))	
Doğrusal Zaman:	O(n)	
Doğrusal-logaritmik Zaman:	O(n log(n))	
İkinci Dereceden Zaman:	O(n^2)	
Kubik Zaman:	O(n^3)	
Üstel Zaman:	O(b^n)	b>1
Faktoriyel Zaman:	O(n!)	

Algoritmanın Zaman Karmaşıklığı

Function	n: 10	100	1,000	10,000	100,000	1,000,000
1	1	1	1	1	1	1
$\log_2 n$	3	6	9	13	16	19
n	10	10^2	10^3	10^4	10^5	10^6
$n * \log_2 n$	30	664	9,965	10^5	10^6	10^7
n^2	10^2	10^4	10^6	10^8	10^{10}	10^{12}
n^3	10^3	10^6	10^9	10^{12}	10^{15}	10^{18}
2^n	10^3	10^{30}	10^{301}	$10^{3,010}$	$10^{30,103}$	$10^{301,030}$

Algoritmanın Zaman Karmaşıklığı

- Giriş boyutu 8 olan bir veri için algoritma 1 saniyede yanıt vermektedir, problem boyutunun 16 olması durumunda algoritmanın çalışma zamanın hesaplayınız?

- Algoritmanın derecesine göre:

O(1) ➔ T(n) = 1 saniye

O($\log_2 n$) ➔ T(n) = (1*$\log_2 16$) / $\log_2 8$ = 4/3 saniye

O(n) ➔ T(n) = (1*16) / 8 = 2 saniye

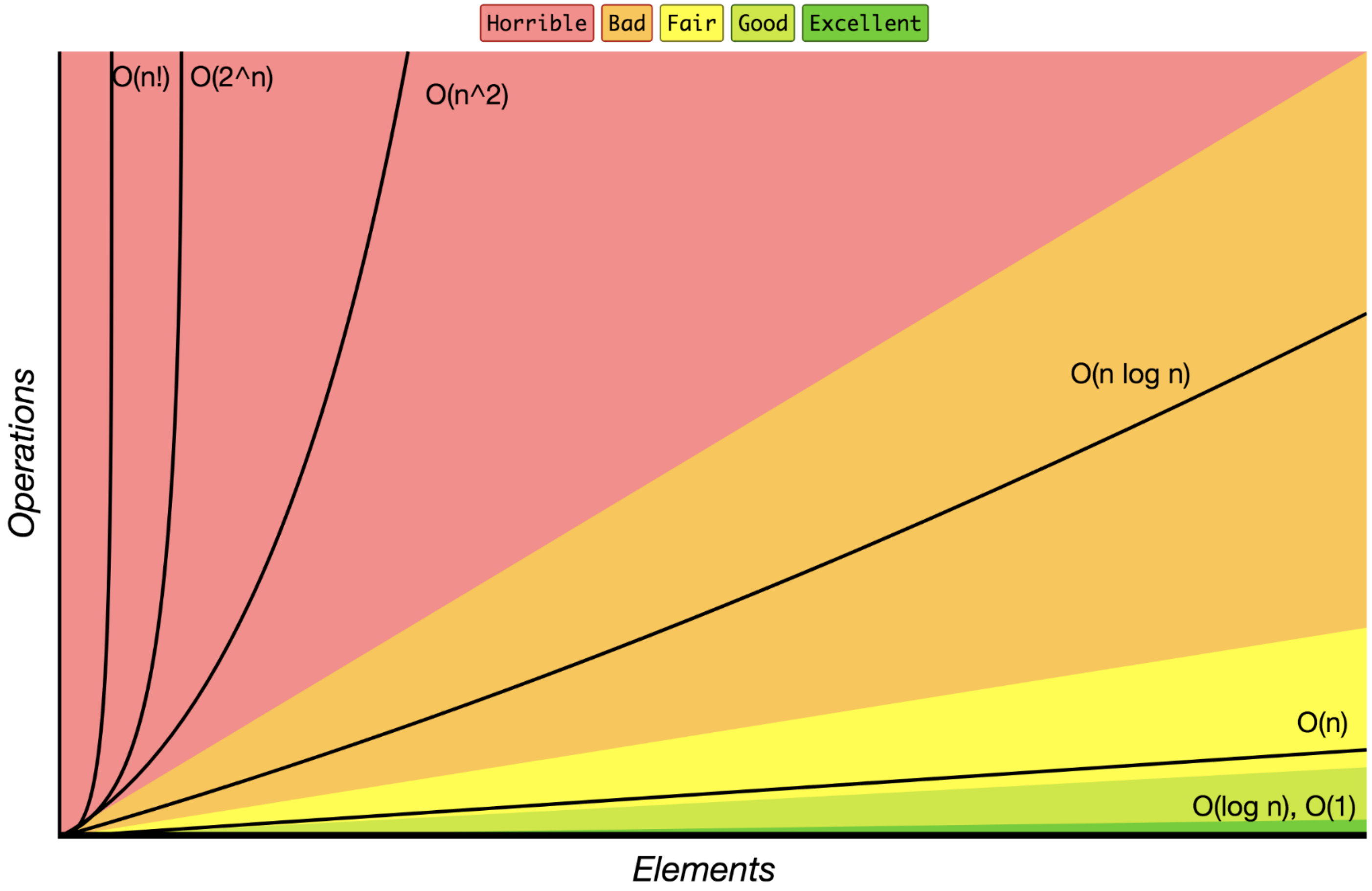
O(n*$\log_2 n$) ➔ T(n) = (1*16*$\log_2 16$) / 8*$\log_2 8$ = 8/3 saniye

O(n^2) ➔ T(n) = (1*16^2) / 8^2 = 4 saniye

O(n^3) ➔ T(n) = (1*16^3) / 8^3 = 8 saniye

O(2^n) ➔ T(n) = (1*2^{16}) / 2^8 = 2^8 saniye = 256 saniye

Algoritmanın Zaman Karmaşıklığı



O-Notasyonu

O(g(n)) =

{

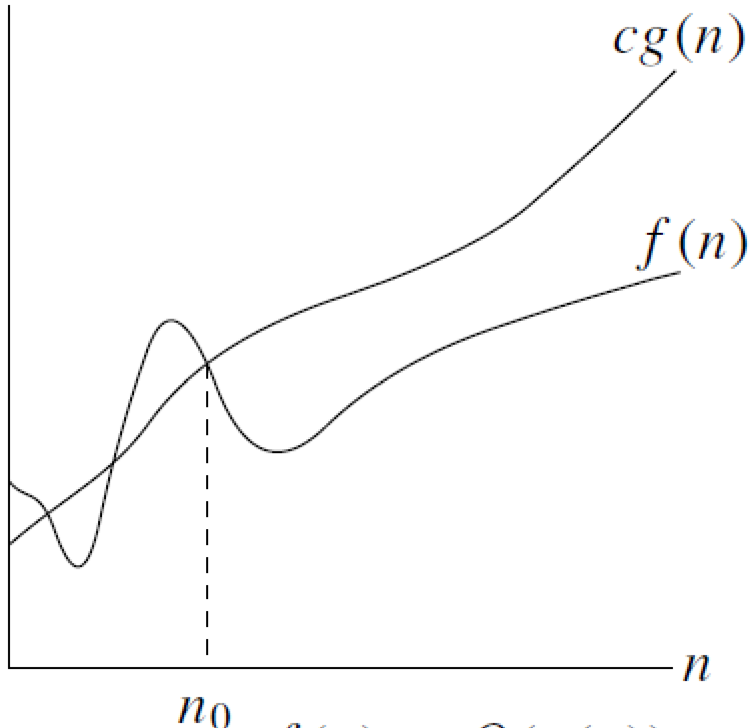
f(n) :

$\exists$ pozitif sabitler c and n_0,

$\forall$n $\geq$ n_0,

0 $\leq$ f(n) $\leq$ cg(n)

}



$f(n) = O(g(n))$

Θ-Notasyonu

Θ(g(n)) =

{

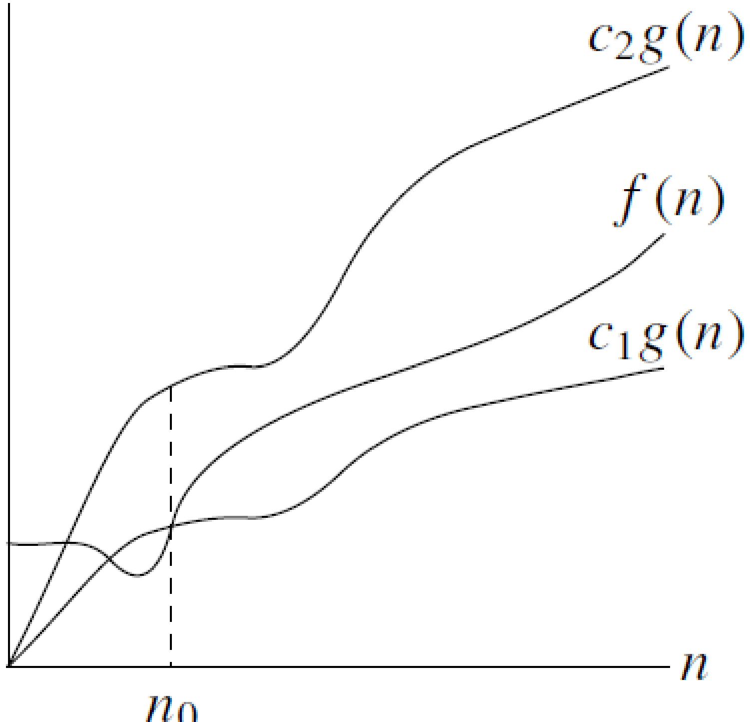
f(n) :

∃ pozitif sabitler c_1, c_2, ve n_0,

∀n ≥ n_0,

$0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$

}



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

Ω -Notasyonu

$\Omega(g(n))$ =

{

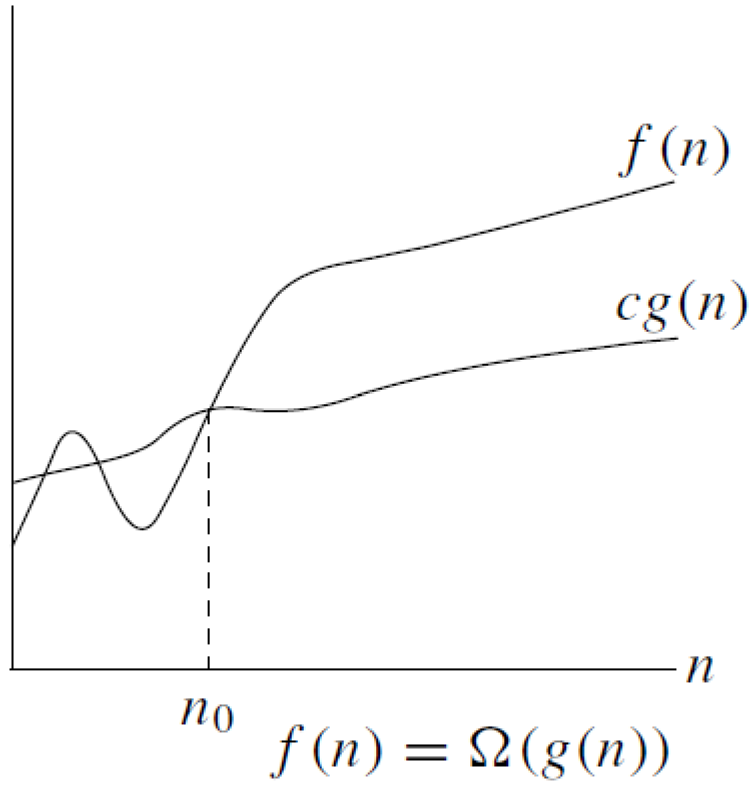
f(n) :

$\exists$ pozitif sabitler c and n_0,

$\forall n \geq n_0$,

$0 \leq cg(n) \leq f(n)$

}



o - Notasyonu

o(g(n)) =

{

f(n): $\forall$ c > 0, $\exists$ n_0 > 0

$\forall$ n $\geq$ n_0,

0 $\leq$ f(n) < cg(n)

}

g(*n*) ***üst sınırı*** *f*(*n*) *için* asimptotik olarak sıkı değildir.

ω -Notasyonu

o(g(n)) =

{

f(n): $\forall$ c > 0, $\exists\, n_0$ > 0

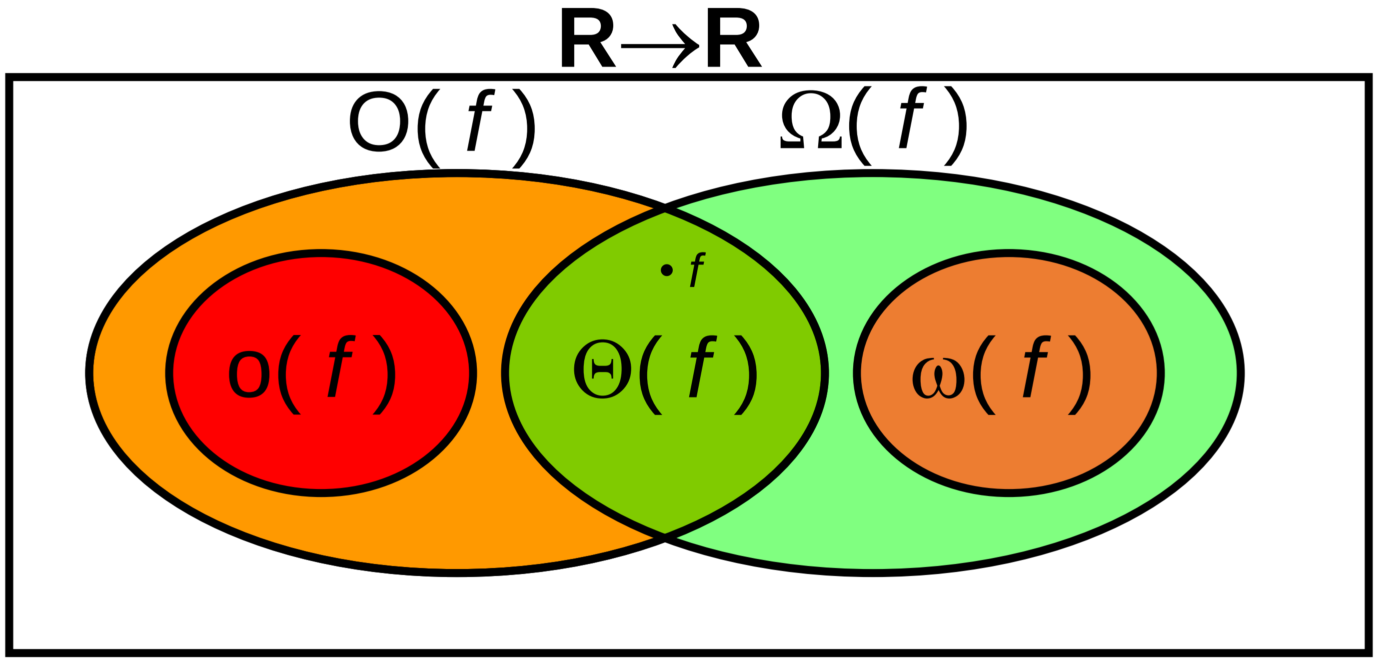
$\forall$ n $\geq$ n_0,

0 $\leq$ f(n) < cg(n)

}

g(*n*) ***alt sınırı*** *f*(*n*) *için* asimptotik olarak sıkı değildir.

Asimptotik Notasyonlar



Fonksiyonların Karşılaştırılması

$$f \leftrightarrow g \approx a \leftrightarrow b$$

$$f(n) = O(g(n)) \approx a \leq b$$

$$f(n) = \Omega(g(n)) \approx a \geq b$$

$$f(n) = \Theta(g(n)) \approx a = b$$

$$f(n) = o(g(n)) \approx a < b$$

$$f(n) = \omega(g(n)) \approx a > b$$

Monotonluk

- $f(n)$ fonksiyonu:
  - **Monoton artandır:** eğer $m \leq n \Rightarrow f(m) \leq f(n)$.
  - **Monoton azalandır:** eğer $m \geq n \Rightarrow f(m) \geq f(n)$.
  - **Kesinlikle artandır:** eğer $m < n \Rightarrow f(m) < f(n)$.
  - **Kesinlikle azalandır:** eğer $m > n \Rightarrow f(m) > f(n)$.

Fonksiyonların Büyüme Derecelerini Karşılaştırmak için Limit Kullanımı

$$\lim_{n\to\infty} t(n)/g(n) = \begin{cases} 0 & \text{büyüme derecesi } \boldsymbol{t(n)} < \text{büyüme derecesi } \boldsymbol{g(n)} \\ c & \text{büyüme derecesi } \boldsymbol{t(n)} = \text{büyüme derecesi } \boldsymbol{g(n)} \\ \infty & \text{büyüme derecesi } \boldsymbol{t(n)} > \text{büyüme derecesi } \boldsymbol{g(n)} \end{cases}$$

Örnekler

Limit kullanarak ispatlayınız.

- $10n - 3n \in O(n^2)$
- $3n^4 \in \Omega(n^3)$
- $n^2/2 - 3n \in \Theta(n^2)$
- $2^{2n} \in \Theta(2^n)$

Örnekler

Limit kullanarak ispatlayınız.

- $10n - 3n \in O(n^2)$ – **Evet!**

  $\lim_{n\to\infty} [\, 10n - 3n \,/\, n^2 \,] = 0$

- $3n^4 \in \Omega(n^3)$ – **Evet!**

  $\lim_{n\to\infty} [\, 3n^4 \,/\, n^3 \,] = \infty$

- $n^2/2 - 3n \in \Theta(n^2)$ – **Evet!**

  $\lim_{n\to\infty} [\, n^2/2 - 3n \,/\, n^2 \,] = 1/2$

- $2^{2n} \in \Theta(2^n)$ – **Hayır!**

  $\lim_{n\to\infty} [\, 2^{2n} \,/\, 2^n \,] = \infty$

Veri Yapıları ve Algoritmalar

ZAFER CÖMERT
Öğretim Üyesi