Algoritma Analizi Çerçevesi

- Algoritma Analizinde Göz Önünde Bulundurulması Gerekenler Neler?
  - Algoritmanın Doğruluğu (Correctness)
  - Zaman Verimliliği (Time Efficiency)
  - Bellek Alanı Verimliliği (Space Efficiency)
    - Gelişen donanım teknolojisi ile artık çok önemli değil
  - Uygunluk, en iyilik (Optimality)

Algoritma Analizi Çerçevesi

- Girdi Büyüklüğünün Ölçülmesi
- Çalışma Zamanı Ölçü Biriminin Belirlenmesi
- Büyüme Derecesi (Order of Growth)
- En kötü durum, en iyi durum, ortalama durum değerlendirmeleri

Girdi Büyüklüğünün Ölçülmesi

- Girdi Büyüklüğünün Ölçülmesi
  - Tüm algoritmalar için büyük girdiler üzerinde çalışma süresi daha uzundur
    - Büyük dizi içinde arama
    - Büyük boyutlu matrisleri çarpma
  - Algoritmaya girilen veriyi belirten n değerinin belirlenmesi önemli
    - Bazı algoritmalar için kolay
      - Arama, sıralama, eleman sayısı bulma
    - Bazı algoritmalar için değil
      - İki matrisin çarpımı
        - » Matrislerin derecesi mi? Eleman Sayıları mı?
      - Yazım hatası programı
        - » Karakter Sayısı mı? Kelime Sayısı mı?

Çalışma Zamanı Ölçü Biriminin Belirlenmesi

- Bir Algoritmanın Çalışma Zamanı
  - Çalıştığı bilgisayar sisteminin hızı
  - Algoritmanın kullanıldığı programın kalitesine
  - Makine kodunu üreten derleyici gibi etkenlere bağımlıdır
- Bu yüzden dış etkenlere bağımlı olmayan bir ölçüm yolu bulunmalıdır
  - Bir yaklaşım: Algoritma içerisindeki tüm işlemlerin kaç kere gerçekleştiğinin sayılması
    - Zor ve Gereksiz
  - Bir diğer yaklaşım: Algoritma içerisindeki temel işlemin belirlenip, kaç kere gerçekleştiğinin sayılması

Çalışma Zamanı Ölçü Biriminin Belirlenmesi

- Zaman verimliliğinin teorik incelenmesi
  - Zaman verimliliği girdi üzerindeki temel işlemin tekrar sayısı üzerinden değerlendirilir
  - Temel işlem
    - Algoritmanın çalışma süresince en çok gerçekleştirilen işlem

Girdi Büyüklüğü

$$T(n) \approx c_{op} C(n)$$

Koşma Süresi

Temel işlemin Çalışma süresi

Temel işlemin gerçekleşme Sayısı

Çalışma Zamanı Ölçü Biriminin Belirlenmesi

- Temel işlemin belirlenmesi
  - Genellikle en çok gerçekleşen işlem
  - Genellikle en iç döngüde yapılan işlem
    - Sıralama algoritmaları için karşılaştırma
    - Matematiksel işlemler için genellikle 4 işlem
      - / en uzun işlem, sonra *, + ve –

Çalışma Zamanı Ölçü Biriminin Belirlenmesi

- c_{op} :
  - Bir algoritmanın temel işleminin bir bilgisayardaki çalışma süresi
- C(n) :
  - Temel işlemin gerçekleşme sayısı
- T(n) :
  - Algoritmanın uygulandığı programın çalışma süresi
- $$T(n) \approx c_{op}\, C(n)$$
  - Bu formülün yaklaşık ve tahmini bir değer verdiği unutulmamalı

Çalışma Zamanı Ölçü Biriminin Belirlenmesi

- $T(n) \approx c_{op}\, C(n)$
  - Bu programı 10 kat hızlı bir bilgisayarda çalıştırsak ne kadar hızlı sonuç alırız?
    - Cevap : 10 kez

$C(n) = \frac{1}{2}n(n-1)$ ise girdiyi iki kat büyütürsek çalışma zamanı ne kadar artar?

$$C(n) = \frac{1}{2}n(n-1) = \frac{1}{2}n^2 - \frac{1}{2}n \approx \frac{1}{2}n^2$$

$$\frac{T(2n)}{T(n)} \approx \frac{c_{op}C(2n)}{c_{op}C(n)} \approx \frac{\frac{1}{2}(2n)^2}{\frac{1}{2}n^2} = 4.$$

Büyüme Derecesi

- Büyüme derecesi
  - Küçük girdi boyutları ile bir algoritmanın etkinliğinin değerlendirilmesi sağlıklı değil

Values (some approximate) of several functions important for analysis of algorithms

n	$\log_2 n$	n	$n \log_2 n$	n^2	n^3	2^n	$n!$
10	3.3	10^1	$3.3 \cdot 10^1$	10^2	10^3	10^3	$3.6 \cdot 10^6$
10^2	6.6	10^2	$6.6 \cdot 10^2$	10^4	10^6	$1.3 \cdot 10^{30}$	$9.3 \cdot 10^{157}$
10^3	10	10^3	$1.0 \cdot 10^4$	10^6	10^9		
10^4	13	10^4	$1.3 \cdot 10^5$	10^8	10^{12}		
10^5	17	10^5	$1.7 \cdot 10^6$	10^{10}	10^{15}		
10^6	20	10^6	$2.0 \cdot 10^7$	10^{12}	10^{18}		

En İyi, En Kötü, Ortalama Durum

- En kötü durum
  - n boyutunda girdi üzerinden en yüksek değer
- En iyi durum
  - n boyutunda girdi üzerinden en düşük değer
- Ortalama durum
  - n boyutunda girdi için «ortalama» değer
    - Tipik bir girdi için temel işlemin kaç kere olduğu
    - En kötü ile en iyinin ortalaması değil
    - Temel işlem sayısı olarak bir olasılık dağılımı içerisinden rastgele bir değişken değeri beklenir

En Kötü Durum

```
ALGORITHM  SequentialSearch(A[0..n − 1], K)
    //Searches for a given value in a given array by sequential search
    //Input: An array A[0..n − 1] and a search key K
    //Output: The index of the first element in A that matches K
    //          or −1 if there are no matching elements
    i ← 0
    while i < n and A[i] ≠ K do
        i ← i + 1
    if i < n return i
    else return −1
```

- Sıralı arama için
  - En kötü durum: Aranan değerin dizide bulunmaması
    - N boyutunda girdi için maksimum sayıda karşılaştırma
      - $C_{worst}(n) = n$
    - En kötü durum incelemesi bir algoritmanın çalışma zamanı açısından üst sınırını belirler

En İyi Durum İncelemesi

```
ALGORITHM  SequentialSearch(A[0..n − 1], K)
    //Searches for a given value in a given array by sequential search
    //Input: An array A[0..n − 1] and a search key K
    //Output: The index of the first element in A that matches K
    //          or −1 if there are no matching elements
    i ← 0
    while i < n and A[i] ≠ K do
        i ← i + 1
    if i < n return i
    else return −1
```

- Sıralı arama için
  - En iyi durum: Aranan değerin ilk karşılaştırmada bulunması
    - N boyutunda girdi için maksimum sayıda karşılaştırma
      - $C_{best}(n) = 1$
    - En iyi durum incelemesi en kötü durum kadar önemli bir veri sağlamaz.

Ortalama Durum İncelemesi

- En iyi durum incelemesi de en kötü durum incelemesi de bir algoritmayı değerlendirmek için yeterli veri sağlamaz
  - Algoritmanın tipik veya rastgele bir girdi karşısındaki davranışı?

Ortalama Durum İncelemesi

ALGORITHM *SequentialSearch*$(A[0..n-1], K)$

```
//Searches for a given value in a given array by sequential search
//Input: An array A[0..n − 1] and a search key K
//Output: The index of the first element in A that matches K
//          or −1 if there are no matching elements
i ← 0
while i < n and A[i] ≠ K do
    i ← i + 1
if i < n return i
else return −1
```

$$
\begin{aligned}
C_{avg}(n) &= [1 \cdot \frac{p}{n} + 2 \cdot \frac{p}{n} + \cdots + i \cdot \frac{p}{n} + \cdots + n \cdot \frac{p}{n}] + n \cdot (1-p) \\
&= \frac{p}{n}[1 + 2 + \cdots + i + \cdots + n] + n(1-p) \\
&= \frac{p}{n}\frac{n(n+1)}{2} + n(1-p) = \frac{p(n+1)}{2} + n(1-p).
\end{aligned}
$$

Sıralı arama için

- Standart kabule göre başarılı bir aramanın olasılığı p (0<=p<=1)
- İlk karşılaştırmada bulma olasılığı her hangi bir i değeri için aynı. (p/n)
- Bulunamama olasılığı
  - n.(1-p)

Ortalama Durum

$$C_{avg}(n) = [1 \cdot \frac{p}{n} + 2 \cdot \frac{p}{n} + \cdots + i \cdot \frac{p}{n} + \cdots + n \cdot \frac{p}{n}] + n \cdot (1-p)$$

$$= \frac{p}{n}[1 + 2 + \cdots + i + \cdots + n] + n(1-p)$$

$$= \frac{p}{n}\frac{n(n+1)}{2} + n(1-p) = \frac{p(n+1)}{2} + n(1-p).$$

- p=1 ise (Başarılı arama)
  - $C_{avg}(n) = (n+1) / 2$ olur.
  - Bu durum başarılı bir aramda algoritmanın ortalama olarak dizinin yarısına kadar aranan elemanı bulacağı kabul edilir.

1. n elemanın toplamı
2. n! değerinin hesaplanması
3. n elemanlı dizinin en büyük elemanının bulunması
4. Öklid'in EBOB algoritması
5. n*n boyutlu iki matrisin toplanması
6. n*n boyutlu iki matrisin çarpımı

- Yukarıdaki algoritmaların temel işlemlerini ve gerçekleşme sayılarını bulunuz. C(n)

- Sıralı aramayı aranan verinin yer aldığı indis listesi veren versiyonunu tasarlayıp(birden fazla kez yer alma durumu), verimliliğini klasik sıralı aramayla karşılaştırın. (C(n), T(n), En kötü, en iyi durumlar)

Büyüme Derecesinin Asimptotik İncelemesi

- Algoritma verimliliğinin değerlendirilmesinde büyüme derecesi temel işlemin gerçekleşme sayısı ile ilgilidir
- Büyüme derecesi değerlendirilirken 3 farklı notasyon kullanılır
  - O (Big Oh)
  - Ω (Big Omega)
  - Θ (Big Theta)

Big Oh Notasyonu

- *O(g(n))* :
  - Bir g(n) fonksiyonu ile aynı veya daha düşük büyüme derecesine sahip fonksiyonların tümü
    - Bir sabit katsayı ve n değeri sonsuza giderken

$$n \in O(n^2), \qquad 100n + 5 \in O(n^2), \qquad \frac{1}{2}n(n-1) \in O(n^2).$$

$$n^3 \notin O(n^2), \qquad 0.00001n^3 \notin O(n^2), \qquad n^4 + n + 1 \notin O(n^2).$$

Big Oh Notasyonu

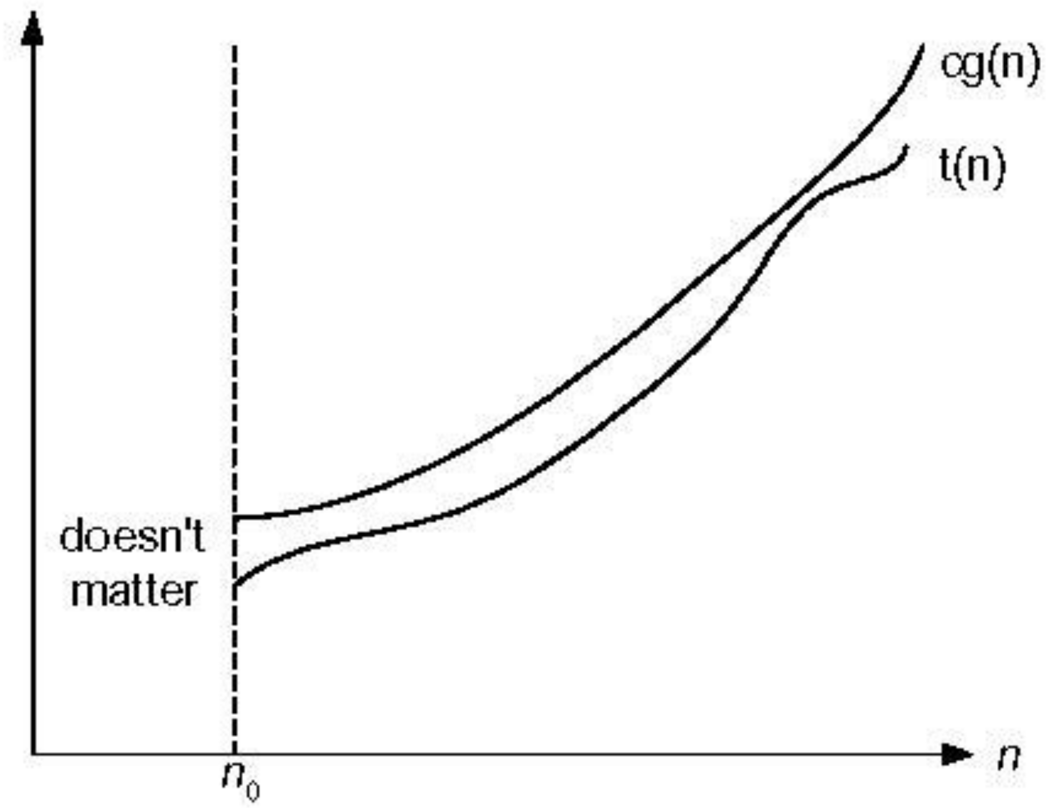

Figure 2.1 Big-oh notation: $t(n) \in O(g(n))$

Big Oh Notasyonu

- Bir $t(n)$ fonksiyonu için $O(g(n))$ içerisindededir diyebilmek için (- $t(n) \in O(g(n))$ -)
  - $T(n)$, Daha büyük n değerleri için üstten, sabit çarpanlı bir $g(n)$ fonksiyonu ile sınırlandırılmış olmalıdır.
  - Pozitif sabit katsayı c, n_0 negatif olmayan bir tamsayı ise
    - $t(n) \leq cg(n)$, $n \geq n_0$ ise
  - $100n + 5 \in O(n^2)$ için ispat
    - $100n + 5 \leq 100n + n$, $n \geq 5$ ise
    - $100n + 5 \leq 101n$
    - $101n \leq 101n^2$
  - $c = 101$, $n_0 = 5$ alınabilir

Big Omega Notasyonu

- $\Omega(g(n))$
  - Bir g(n) fonksiyonu ile aynı veya daha büyük büyüme derecesine sahip fonksiyonların tümü
    - Bir sabit katsayı ve n değeri sonsuza giderken

$$n^3 \in \Omega(n^2), \qquad \frac{1}{2}n(n-1) \in \Omega(n^2), \qquad \text{but } 100n + 5 \notin \Omega(n^2).$$

Big Omega Notasyonu

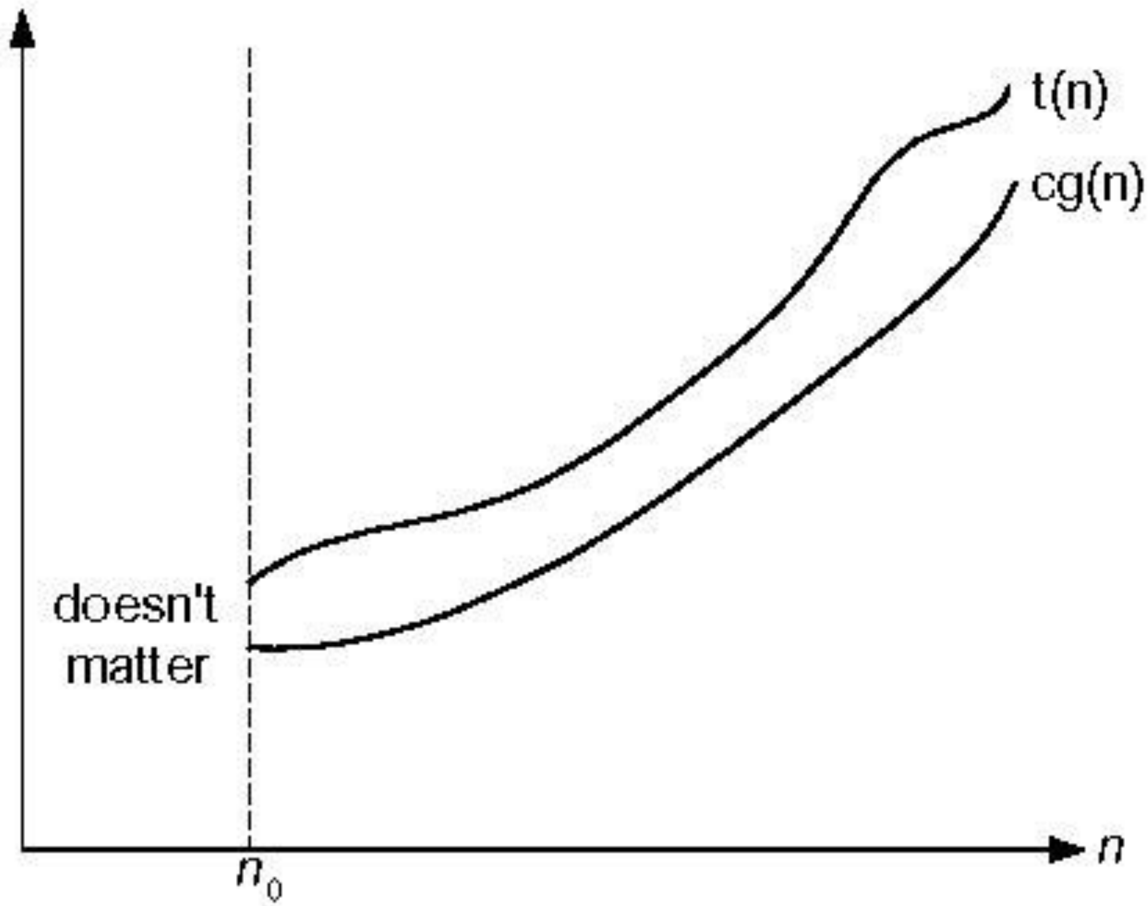

Fig. 2.2 Big-omega notation: $t(n) \in \Omega(g(n))$

Big Omega Notasyonu

- Bir t(n) fonksiyonu için Ω(g(n)) içerisindedir diyebilmek için (- $t(n) \in \Omega(g(n))$ -)
  - T(n), Daha büyük n değerleri için alttan, sabit çarpanlı bir g(n) fonksiyonu ile sınırlandırılmış olmalıdır.
  - Pozitif sabit katsayı c, n_0 negatif olmayan bir tamsayı ise
    - $t(n) \geq cg(n)$, $n \geq n_0$ ise
  - $n^3 \in \Omega(n^2)$ ispatı için
    - $n^3 \geq n^2$, $n \geq 0$ *ise*
  - $c = 1$, $n_0 = 0$ alınabilir

Big Theta Notasyonu

- Θ(g(n))
  - Bir g(n) fonksiyonu ile aynı büyüme derecesine sahip fonksiyonların tümü
    - Bir sabit katsayı ve n değeri sonsuza giderken
  - $an^2 + bn + c$, $a > 0$ *ise* $\Theta(n^2)$ içerisindedir

Big Theta Notasyonu

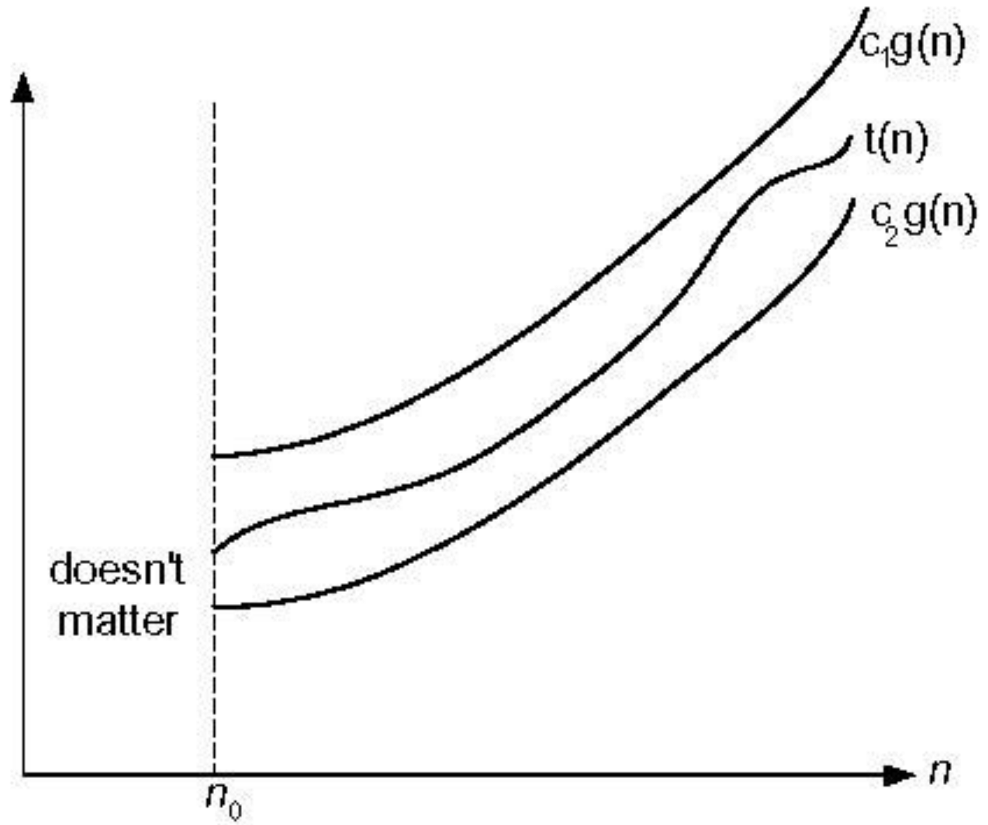

Figure 2.3 Big-theta notation: $t(n) \in \Theta(g(n))$

Big Theta Notasyonu

- Bir t(n) fonksiyonu için *Θ* (g(n)) içerisindedir diyebilmek için (- $t(n) \in \Theta(g(n))$ -)
  - t(n), Daha büyük n değerleri için alttan ve üstten, sabit çarpanlı bir g(n) fonksiyonu ile sınırlandırılmış olmalıdır.
  - Pozitif sabit katsayı c_1 ve c_2, n_0 negatif olmayan bir tamsayı ise
    - $c_2 g(n) \le t(n) \le c_1 g(n),\ n \ge n_0$ ise

$$\tfrac{1}{2}n(n-1) \in \Theta(n^2)$$

$$\frac{1}{2}n(n-1) = \frac{1}{2}n^2 - \frac{1}{2}n \le \frac{1}{2}n^2 \quad \text{for all } n \ge 0.$$

$$\frac{1}{2}n(n-1) = \frac{1}{2}n^2 - \frac{1}{2}n \ge \frac{1}{2}n^2 - \frac{1}{2}n\frac{1}{2}n \ (\text{for all } n \ge 2) = \frac{1}{4}n^2.$$

$$c_2 = \tfrac{1}{4},\ c_1 = \tfrac{1}{2},\ n_0 = 2.$$

Asimptotik Büyüme Derecesi İçin Bazı Özellikler

- $f(n) \in O(f(n))$
- $f(n) \in O(g(n))$, eğer $g(n) \in \Omega(f(\mathrm{n}))$
- Eğer $f(n) \in O(g(n))$ ve $g(n) \in O(h(n))$, ise
  - $f(n) \in O(h(n))$
- If $f_1(n) \in O(g_1(n))$ ve $f_2(n) \in O(g_2(n))$, ise
  - $f_1(n) + f_2(n) \in O(\max\{g_1(n), g_2(n)\})$

Büyüme Derecelerinin Karşılaştırılması

- *O, Ω, ve Θ* yaklaşımları tanımlarının birbirinden bağımsız olması sebebiyle iki fonksiyonun büyüme derecelerinin karşılaştırılmasında pek kullanılmaz

- Bu işlem için iki fonksiyonlarının oranının sonsuza giderken limit yaklaşımı daha uygundur

Limit Yöntemi İle Karşılaştırma

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

- İlk iki durum : *$t(n) \in O(g(n))$,*
- *İkinci durum : $t(n) \in \Theta(g(n))$*
- *Son iki durum : $t(n) \in \Omega(g(n))$,*

Limit Yöntemi İle Karşılaştırma

L'Hôpital Kuralı:

Eğer $lim_{n\to\infty} f(n) = lim_{n\to\infty} g(n) = \infty$ ve f', g' türevleri varsa

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{f'(n)}{g'(n)}$$

Çok büyük n değerleri için Stirling Formülü

$$n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$$

Örnek

- $\frac{1}{2}n(n-1)$ ile n^2 fonksiyonlarının büyüme derecelerini karşılaştırınız

$$\lim_{n\to\infty} \frac{\frac{1}{2}n(n-1)}{n^2} \quad = \frac{1}{2}\lim_{n\to\infty}\frac{n^2-n}{n^2} \quad = \frac{1}{2}\lim_{n\to\infty}(1-\frac{1}{n}) = \frac{1}{2}.$$

- Sonuç sabit olduğu için eşit büyüme derecelerine sahipler

$$\frac{1}{2}n(n-1) \in \Theta(n^2)$$

Örnek

- $\text{Log}_2 n$ ve $\sqrt{n}$ fonksiyonlarının büyüme derecelerini karşılaştırınız.

$$\lim_{n\to\infty} \frac{\log_2 n}{\sqrt{n}} = \lim_{n\to\infty} \frac{(\log_2 n)'}{(\sqrt{n})'} = \lim_{n\to\infty} \frac{(\log_2 e)\,\frac{1}{n}}{\frac{1}{2\sqrt{n}}} = 2\log_2 e \lim_{n\to\infty} \frac{1}{\sqrt{n}} = 0.$$

- $\text{Log}_2 n$ büyüme derecesi daha küçüktür

Örnek

- n! ve 2^n fonksiyonlarının büyüme derecelerini karşılaştırınız

$$\lim_{n\to\infty} \frac{n!}{2^n} = \lim_{n\to\infty} \frac{\sqrt{2\pi n}\left(\frac{n}{e}\right)^n}{2^n} = \lim_{n\to\infty} \sqrt{2\pi n}\frac{n^n}{2^n e^n} = \lim_{n\to\infty} \sqrt{2\pi n}\left(\frac{n}{2e}\right)^n = \infty.$$

- n! Daha hızlı büyümektedir.
- $n! \in \Omega(2^n)$

Örnek

1	constant
$\log n$	logarithmic
n	linear
$n \log n$	linearithmic
n^2	quadratic
n^3	cubic
2^n	exponential
$n!$	factorial