Yürütme Zamanı (Running Time)

- Algoritmanın belirli bir işleme veya eyleme kaç kez gereksinim duyulduğunu gösteren bağıntıdır ve *T(n)* ile gösterilir.
- Temel hesap birimi olarak, programlama dilindeki deyimler seçilebildiği gibi döngü sayısı, toplama işlemi sayısı, atama sayısı, dosyaya erişme sayısı gibi işler de temel hesap birimi olarak seçilebilir.

Yürütme Zamanı (Running Time)

- Yürütme zamanı bağıntısı fiziksel gerçeğe yakın bir sonuç verir, ancak sapmalar da olabilir.
- Kabul edilen temel hesap birimi tüm hesaplar için aynı olmayabilir.
- ÖR: Bir tam sayı sayacın bir artırılmasıyla iki gerçel sayının çarpımı maliyeti farklı olabilir veya iki tam sayıyı karşılaştırmak ile iki diziyi karşılaştırma maliyetleri farklı olur.

Yürütme Zamanı (Running Time)

- ÖR: Aşağıdaki kodun yürütme zamanı bağıntısını belirleyiniz.

```
float bulorta( float A[], int n)
 {
    float ortalama, toplam=0;
    int k;

1   for (k=0; k<n; k++)
2      toplam+=A[k];
3   ortalama=toplam/n;
4   return ortalama;
 }
```

Yürütme Zamanı (Running Time)

- ÖR: Aşağıdaki kodun yürütme zamanı bağıntısını belirleyiniz.

```
void toplamMatris(A, B, C)
int A[n][m], B[n][m], C[n][m];
 {
    int i,j;
1   for (i=0; i<n; i++)
2       for (j=0; j<m; j++)
3           C[i][j]=A[i][j]+B[i][j]
 }
```

Karmaşıklık

- Algoritmaları karşılaştırabilmek için bir algoritmanın zorluk derecesi ölçümüne "Computational Complexity" denir.
- Computational complexity bir algoritmanın gerçekleştirilmesi için gereken maliyeti veya çabayı ifade eder. Maliyet veya çaba zaman (time) ve kullanılan alan (space) ile ifade edilir.

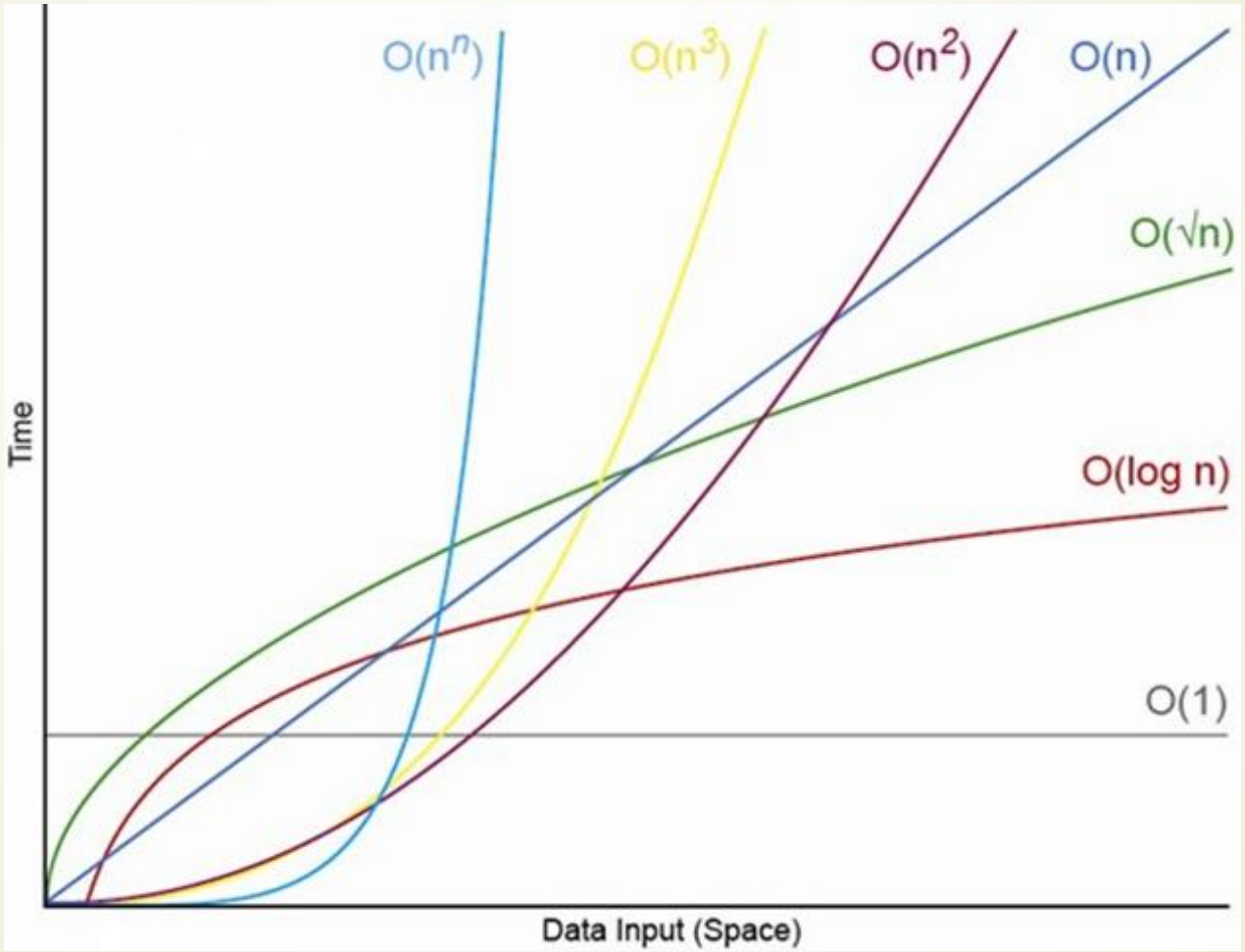
Asimptotik Karmaşıklık

- Algoritmalarda t (süre) ve n (giriş boyutu) arasındaki ilişki çoğu zaman çok karmaşıktır.
- Fonksiyon içerisindeki önemsiz kısımlar ve katsayılar atılarak basitleştirilir ve gerçek fonksiyona göre yaklaşık bir değer bulunur.
- Elde edilen bu yeni etkinlik ölçümüne "Asymptotic Complexity" denir.
- Genellikle girişin büyümesine bağlı olarak fonksiyonun büyümesinde en büyük etkiye sahip olan parametre alınır.

Asimptotik Karmaşıklık

n	$t(n)=60n^2+5n+1$	$60n^2$
10	6.051	6.000
100	600.501	600.000
1.000	60.005.001	60.000.000
10.000	6.000.050.001	6.000.000.000

Asimptotik Karmaşıklık

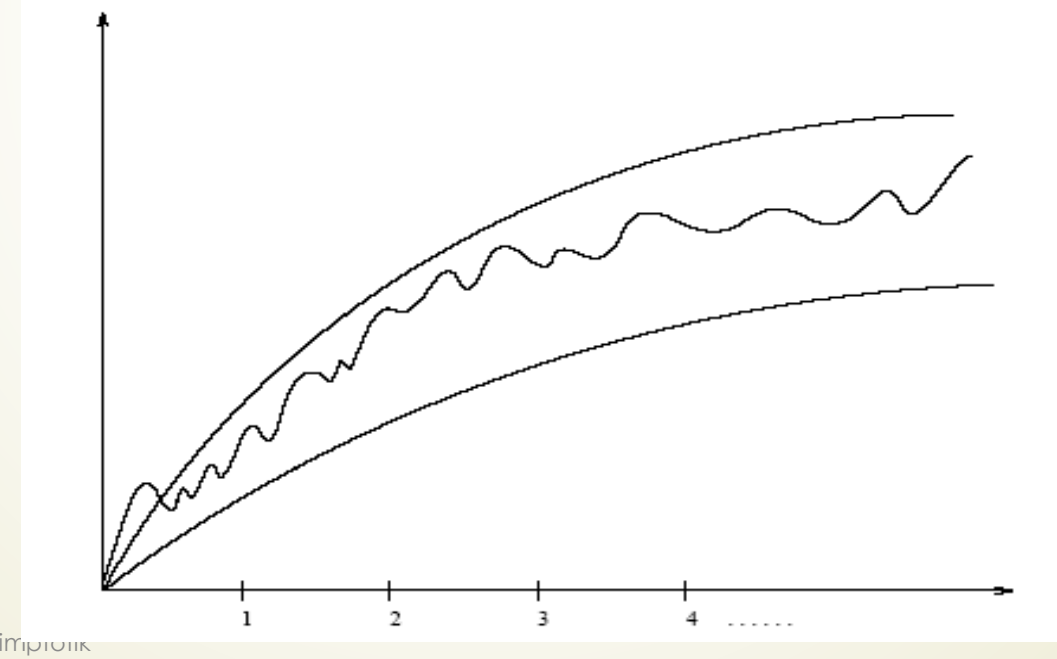


Asimptotik Analiz

- Amaç: detaylardan kurtularak çalışma süresi analizini basitleştirmek
- Sayılar için "rounding" işlemi:1,000,001 ≈ 1,000,000
- Fonksiyonlar için "rounding" işlemi: $3n^2 \approx n^2$
- Niteliğini belirlemek (Capturing the essence): belirlenen limit içerisinde girişin boyutuna göre algoritmanın çalışma süresinin nasıl arttığının bulunması

Asimptotik Analiz

- Algoritmaların best, worst ve average case çalışma süreleri bir fonksiyonla ifade edilebilir.
- Ancak kesin sürenin hesaplanması oldukça zor ve karmaşıktır.

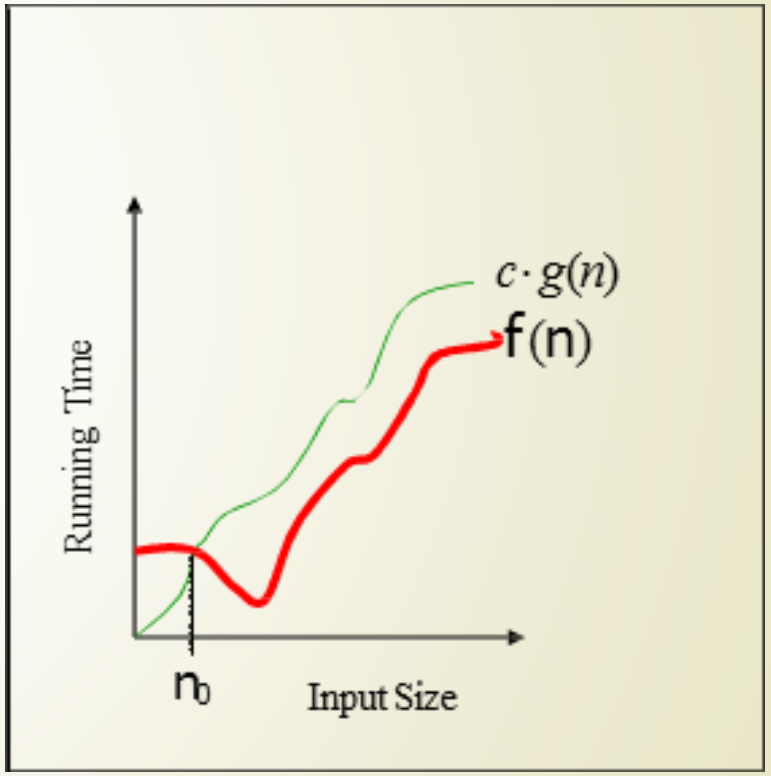


Asimptotik Notasyonlar

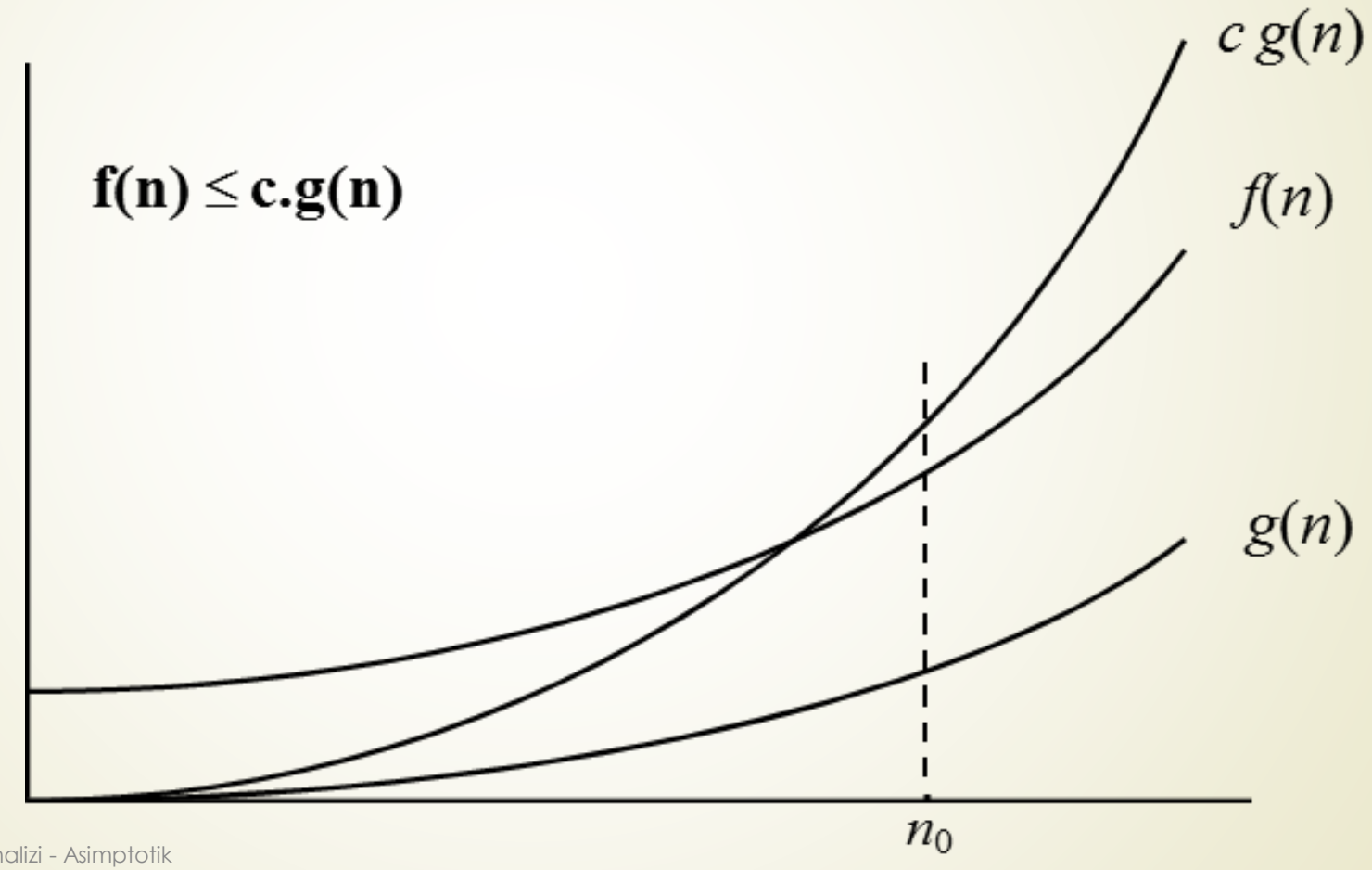
- "big-O" O-ifadesi
  - asymptotic upper bound
  - *f(n) = O(g(n)),* eğer sabit bir c ve n_0 değeri için

    f(n) ≤ c.g(n) bütün $n \geq n_0$

    değerleri için doğruysa
  - *f(n) ve g(n)* pozitif değere sahip fonksiyonlardır.
- worst-case analiz için kullanılır



Asimptotik Notasyonlar – Big O



Asimptotik Notasyonlar – Big O

- Örnek:
- $3n^2+2n+5 = O(n^2)$ olduğunu gösteriniz.
- $10\ n^2 = 3n^2 + 2n^2 + 5n^2$
  $\geq 3n^2 + 2n + 5\ ,\ n \geq 1$
- $c = 10,\ n_0 = 1$

Asimptotik Notasyonlar – Big O

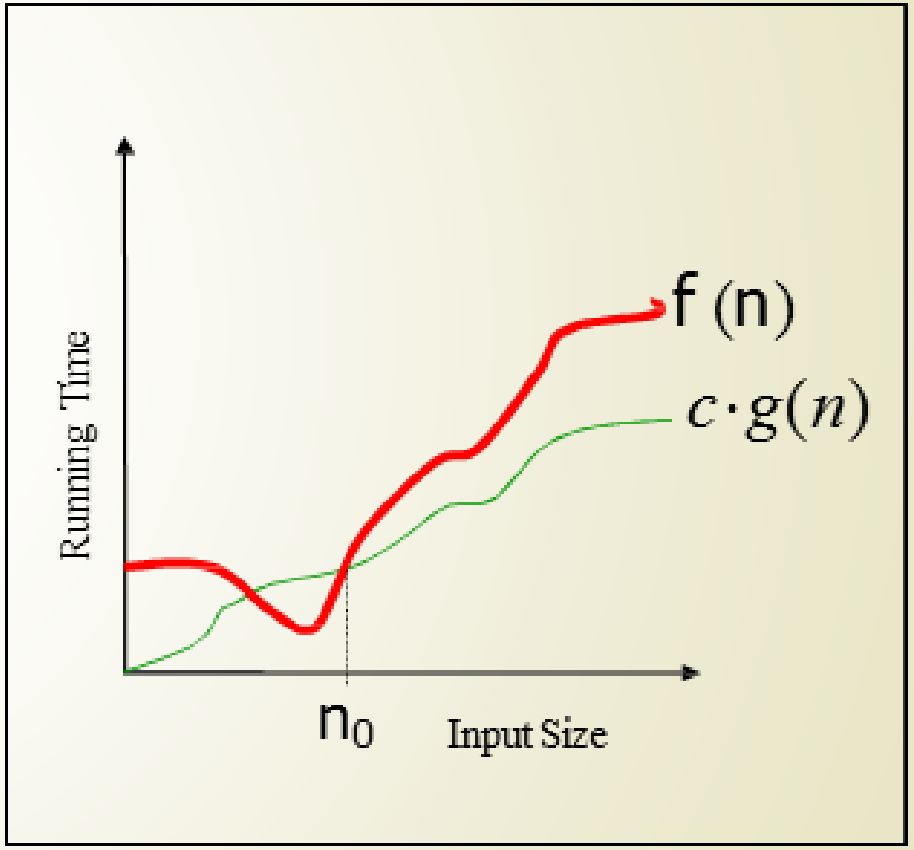
- O-ifadesi için genellikle en basit formül kullanılır.
- Örnek:
  - $3n^2+2n+5 = O(n^2)$
- Aşağıdaki örneklerde doğrudur ancak genellikle kullanılmazlar.
  - $3n^2+2n+5 = O(3n^2+2n+5)$
  - $3n^2+2n+5 = O(n^2+n)$
  - $3n^2+2n+5 = O(3n^2)$

Asimptotik Notasyonlar – Big Ω

- "Big-Omega" Ω ifadesi
- Asymptotic Lower Bound
- f(n) = Ω (g(n)), eğer sabit bir c ve n_0 değeri için

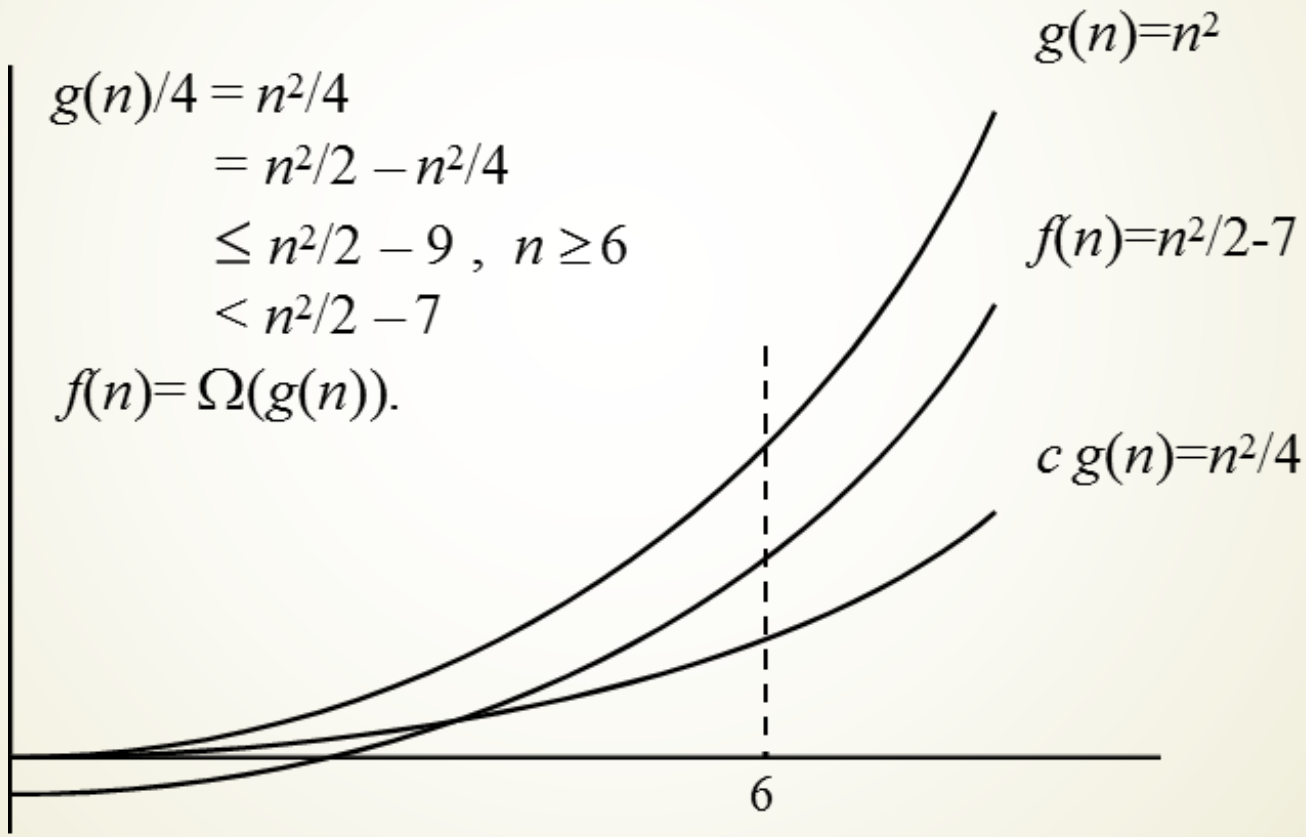
  c.g(n) ≤ f(n) bütün n ≥ n0

  değerleri için doğruysa
- best-case çalışma süresi veya lower bound tanımlamasında kullanılır.



Asimptotik Notasyonlar – Big Ω

$$g(n)/4 = n^2/4$$
$$= n^2/2 - n^2/4$$
$$\leq n^2/2 - 9 \;,\; n \geq 6$$
$$< n^2/2 - 7$$
$$f(n) = \Omega(g(n)).$$



Asimptotik Notasyonlar – Big Ω

- Basit kural: Küçük dereceden terimler ve sabitler atılır.
- 50 *n* log *n* ifadesi O(*n* log *n*) şeklinde gösterilir.
- 7*n* - 3 ifadesi O(*n*)
- $8n^2 \log n + 5n^2 + n$ ifadesi O(n^2 log *n*) şeklinde ifade edilir.

Asimptotik Notasyonlar – Big-Θ

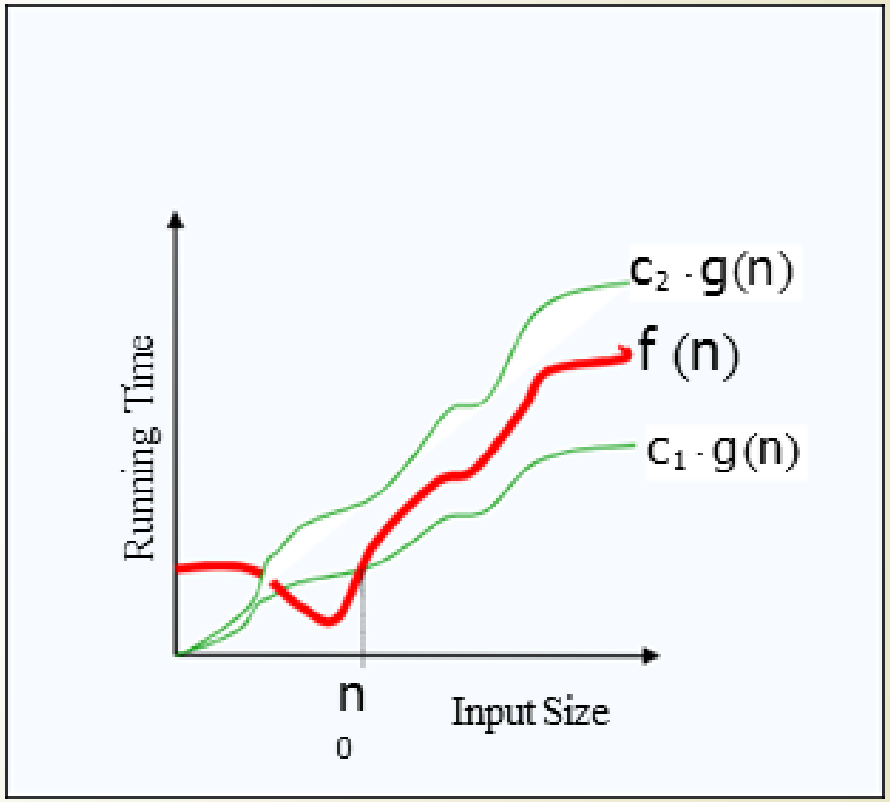
- "Big-Theta" Θ İfadesi
- asymptotic tight bound
- f(n) = Θ(g(n)), eğer sabit

  c_1, c_1 , ve n_0 , değerleri için

  $\mathbf{c_1\, g(n) \le f(n) \le c_2\, g(n)}$

  bütün $n \ge n_0$ değerleri için

  doğruysa.



Asimptotik Notasyonlar – Little-o

- "Little-o" ifadesi f(n)=o(g(n)) Örn: $2n = o(n^2)$
- Her c>0 için, bir n_0 değeri vardır ve

  $0 \leq f(n) < c \cdot g(n)$

  ifadesi bütün $n \geq n_0$ değerleri için doğrudur.
- Çalışma sürelerinin karşılaştırılması için kullanılır. Eğer f(n)=o(g(n)), ise g(n), f(n) fonksiyonundan daha ağırlıklıdır (dominates).

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0$$

Asimptotik Notasyonlar - Little-ω

- "Little-omega" ifadesi f(n)= ω(g(n)) Örn: $n^2/2 = \omega(n)$
- Her c>0 için, bir n_0 değeri vardır ve **f(n) > c.g(n)** ≥ 0 ifadesi bütün $n \geq n_0$ değerleri için doğrudur.
- Çalışma sürelerinin karşılaştırılması için kullanılır. Eğer f(n)= ω(g(n)), ise f(n), g(n) fonksiyonundan daha ağırlıklıdır. (dominates).

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

Asimptotik Notasyonlar

- $f(n) = O(g(n)) \cong f \leq g$
- $f(n) = \Omega(g(n)) \cong f \geq g$
- $f(n) = \Theta(g(n)) \cong f = g$
- $f(n) = o(g(n)) \cong f < g$
- $f(n) = \omega(g(n)) \cong f > g$

- $f(n) = O(g(n))$ gerçekte $f(n) \in O(g(n))$ anlamındadır.