Algoritma Analizi

Ders 3: Asimtotik Notasyon

Doç. Dr. Mehmet Dinçer Erbaş
Bolu Abant İzzet Baysal Üniversitesi
Mühendislik Fakültesi
Bilgisayar Mühendisliği Bölümü

Fonksiyonların büyümesi

- Bir önceki bölümde gördüğümüz üzere algoritmaların çalışma süreleri üzerinde çalıştıkları girdinin miktarına göre değişmektedir.
- Her algoritmanın girdi miktarına göre ne kadar sürede çalıştığını analiz ederek algoritmalarının verimliliklerini hesaplayabiliriz.
- Örneğin
  - Eklemeli Sıralama en kötü çalışma süresi
    - $\Theta(n^2)$
  - Birleştirmeli Sıralama en kötü çalışma süresi
    - $\Theta(n \lg n)$
  - Buna göre girdi miktarı arttıkça Birleştirmeli Sıralama Eklemeli Sıralama ile karşılaştırıldığında daha kısa sürede sonuca ulaşacaktır.
- Girdi miktarı arttıkça katsayı sabitlerin ve düşük-dereceli terimlerin önemi azalmaktadır.

Fonksiyonların büyümesi

- Girdi durumunun arttığında çalışma süresinin ne şekilde değiştiğini hesapladığımızda yaptığımız analiz bir algoritmanın asimtotik verimliliğidir.
  - Yani girdi miktarı giderek arttığında algoritmanın çalışma süresinin değişimini inceliyoruz.
- Bu bölümde bu analizin yapılabilmesi için gerekli matematiksel notasyon gösterilecektir.

Asimtotik notasyon

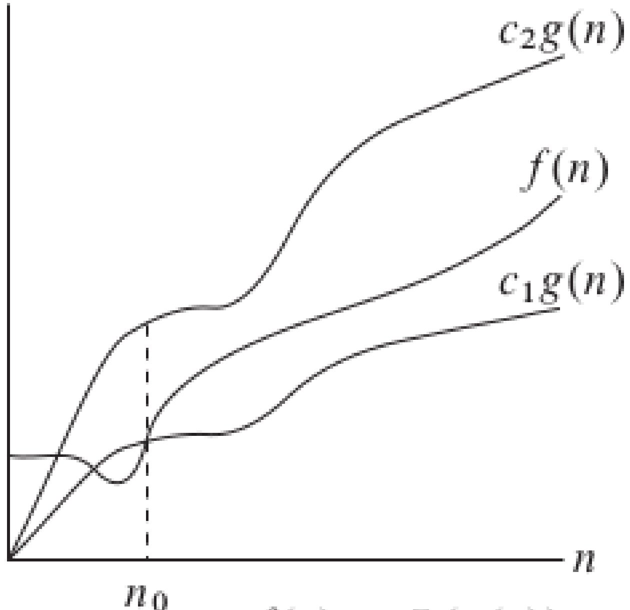
- Algoritmaların çalışma sürelerini tanımlayabilmek için asimtotik notasyon kullanacağız.
  - Örneğin Eklemeli Sıralama en kötü çalışma süresini $\Theta(n^2)$ olarak belirtmiştik.
- Asimtotik notasyon ayrıca fonksiyonların tanımlanmasında kullanılır.
- Yaptığımız analizlerde asimtotik notasyon algoritmanın çalışma süresini tanımlayacak.
  - Asimtotik notasyon kullanarak algoritmanın kullandığı hafıza miktarı tanımlanabilir.
- Asimtotik notasyon ile çalışma süresini tanımlarken hangi çalışma süresinden bahsettiğimizi belirteceğiz.
  - Bazı durumlarda bahsedilen en kötü çalışma süresidir.
  - Ancak istediğimiz her tür girdi için asimtotik notasyon ile çalışma süresini tanımlayabilmektir.

ΘNotasyonu

- Verilen bir g(n) fonksiyonu için, Θ(g(n)) fonksiyon topluluğu şu şekilde tanımlanır:
  - Θ(g(n)) = {f(n) : aşağıda belirtilen eşitsiziliği sağlayan pozitif c_1, c_2 ve n_0 sabitleri bulunabilir.

    $0 \le c_1 g(n) \le f(n) \le c_2 g(n)$ bütün $n \ge n_0$ için}
  - Yukarıda belirtildiği üzere f(n) fonksiyonu için c_1 ve c_2 değerleri bularak fonksiyonun aldığı değerleri c_1g(n) ve c_2g(n) arasına alabiliyorsak, f(n) fonksiyonu Θ(g(n)) kümesine aittir diyoruz.
  - Bu durumda g(n) içim asimtotik sıkı sınır (İng: Asymptotically tight bound) diyoruz.
- Tahtada örnek.

Θ Notasyonu



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

Θ Notasyonu

- Belirtilen Θ(g(n)) tanımının geçerli olabilmesi için f(n) Є Θ(g(n)) fonksiyonlarının asimtotik eksi olmayan (asymptotically nonnegative) olması gerekmektedir.
  - Yani n yeteri kadar büyük olduğnda f(n)'in eksi olmayan olması gerekmektedir.
- Bu bölümde yapacağımız diğer asimtotik notasyon tanımları için aynı koşul gereklidir.

Θ Notasyonu

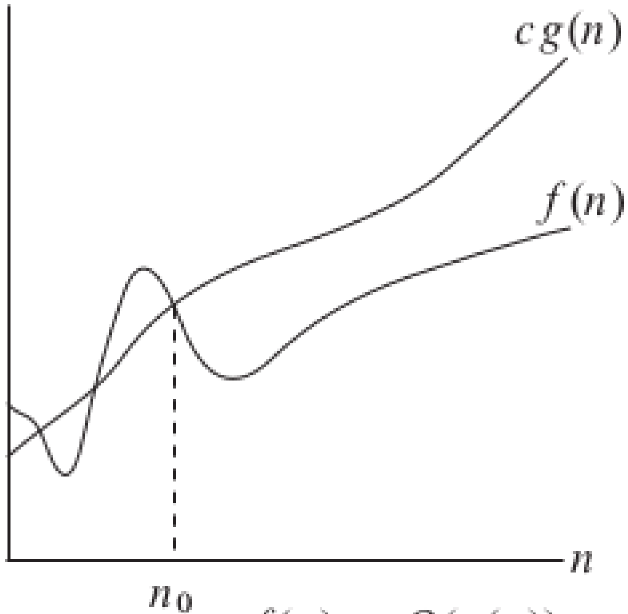
- Örneklerden de anlaşılacağı üzere asimptotik sıkı sınırları bulduğumuzda asimtotik pozitif fonksiyona ait düşük dereceli terimleri görmezden gelebiliriz.
  - Bu terimler n büyüdükçe önemsizleşir.
  - En yüksek dereceye sabit terimin ufak bir parçası bile düşük dereceli terimleri domine eder.
- Bu sebeple c_1 değerini en yüksek dereceli sabitin katsayısından çok az küçük ve c_2 değerini en yüksek dereceli sabitin katsayısında çok az büyük olarak seçmemiz Θ notasyon için oluşturduğumuz eşitsizliklerin doğru olması için yeterli olacaktır.
- Aynı şekilde en yüksek dereceli terimin katsayısı görmezden gelinebilir çünkü bu katsayı sadece c_1 ve c_2 değerlerinin büyüklüğünü sabit bir oranda değiştirmektedir.
- Tahtada örnek.

O Notasyonu

- Önceki bölümde görüldüğü üzere Θ notasyonu bir fonksiyon için üst ve alt sınırları belirliyor.
- Sadece üst sınır belirlemek istediğimizde O (büyük O) notasyonu kullanıyoruz.
- Verilen bir g(n) fonksiyonu için, O(g(n)) fonksiyon topluluğu şu şekilde tanımlanır:
  - O(g(n)) = {f(n) : aşağıda belirtilen eşitsiziliği sağlayan pozitif c ve n_0 sabitleri bulunabilir.

    $0 \leq f(n) \leq cg(n)$ her $n \geq n_0$ için}
- Bir fonksiyona üst sınır belirtmek için O notasyonunu kullanıyoruz.

O Notasyonu



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

O Notasyonu

- $f(n) = O(g(n))$, $f(n)$ $O(g(n))$ set kümesine ait bir fonksiyondur anlamı gelmektedir.
- $f(n) = \Theta(g(n))$ olduğunda $f(n)$ için hem üst hem alt sınır tanımılıyoruz. Bu durumda $f(n) = O(g(n))$ yazabiliriz.
  - Yani $f(n) = \Theta(g(n))$ ise $f(n) = O(g(n))$ yazabiliriz
- $\theta(g(n)) \subseteq O(g(n))$
- Bu sebeple önceki kısımda gösterdiğimiz ispat burada kullanılabilir:
  - an^2+bn+c, $a > 0$ şeklinde gösterilebilen her ikinci derece fonksiyon $\Theta(n^2)$ ise bu fonksiyonlar aynı zamanda $O(n^2)$ kümesine aittir.
- İlginç olan ise $an+b$ şeklinde yazılabilen her birinci derece fonksiyon aynı zamanda $O(n^2)$ kümesine aittir.
  - Bu durumu ispatlama için $c = a + |b|$ ve $n_0 = 1$ değerleri kullanılabilir.

O Notasyonu

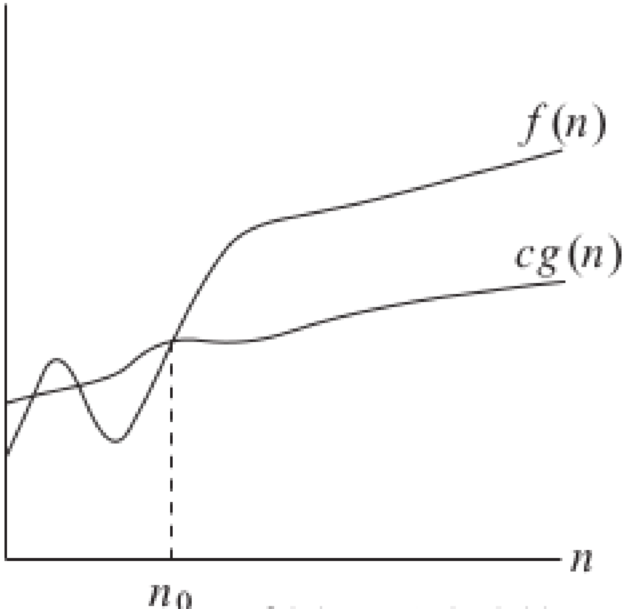
- Bir algoritmanın çalışma süresini bazı durumlarda algoritmanın genel yapısını inceleyerek tanımlayabiliriz.
  - Örneğin Eklemeli Sıralama algoritması
- O notasyonu algoritmanın çalışma süresi için bir üst limit tanımlar.
  - Bu sebeple bir algoritmanın en kötü çalışma süresi için sınır belirlediğimizde bu aynı zamanda algoritmanın herhangi bir girdi ile çalışma süresi için bir üst sınırdır.
  - Aynı durum incelendiğinde Θ notasyonu için geçerli değildir.
    - Eklemeli Sıralama en kötü çalışma süresi $\Theta(n^2)$, en iyi çalışma süresi ise $\Theta(n)$ olarak hesaplanır.

Ω Notasyonu

- O notasyonu fonksiyonların alabileceği değer için asimtotik üst sınır tanımlamak için kullanılırken Ω ise fonksiyonların alabileceği değer için asimtotik alt sınır tanımlar.
- Sadece alt sınır tanımlamak istediğimizde Ω (büyük omega) notasyonunu kullanıyoruz.
- Verilen bir g(n) fonksiyonu için, Ω(g(n)) fonksiyon topluluğu şu şekilde tanımlanır:
  - Ω(g(n)) = {f(n) : aşağıda belirtilen eşitsiziliği sağlayan c ve n_0 sabitleri bulunabilir.

    $0 \le cg(n) \le f(n)$ her $n \ge n_0$ için}
- Bir fonksiyona alt sınır belirlemek için Ω notasyonunu kullanıyoruz.

Ω Notasyonu



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

Ω Notasyonu

- Bu aşamaya kadar gördüğümüz asimtotik notasyon tanımlarını kullanarak aşağıdaki teorimi ispatlayabiliriz:
  - f(n) ve g(n) iki farklı fonksiyon olsun. f(n) = Θ(g(n)) ancak ve ancak f(n) = O(g(n)) ve f(n) = Ω(g(n)) ise doğrudur.
- Bu teoremi kullanarak şu durumu ispatlayabiliriz: an^2+bn+c şeklinde yazılan ikinci dereceden bir fonksiyon $\Theta(n^2)$ ise, bu fonksiyon aynı zamanda $O(n^2)$ ve $\Omega(n^2)$ olmalıdır.
- Ω notasyonu algoritmanın çalışma süresi için alt sınır belirler.
  - Bu sebeple bir algoritmanın en iyi çalışma süresi için sınır belirlediğimizde bu aynı zamanda algoritmanın herhangi bir girdi ile çalışma süresi için bir alt sınırdır.
- Eklemeli Sıralama algoritmasının çalışma süresi Ω(n) ile $O(n^2)$ arasındadır.

Asimtotik Notasyon

- Birleştirmeli Sıralama algoritması için aşağıdaki şekilde bir hesaplama yapmıştık
  - $T(n) = 2T(n/2) + \Theta(n)$
- Bazı durumlarda aşağıdakine benzer formuller görebiliriz.
  - $2n^2 + \Theta(n) = \Theta(n^2)$
- Bir başka kullanım şekli ise şöyle.
  - $2n^2 + 3n + 1 = 2n^2 + \Theta(n)$

    $= \Theta(n^2)$

o Notasyonu

- Önceki bölümde gördüğümüz üzere O notasyonu asimtotik olarak sıkı olabilir veya olmayabilir.
  - Örneğin $2n^2 = O(n^2)$ asimtotik olarak sıkıdır.
  - $2n = O(n^2)$ asimtotik olarak sıkı değildir.
- Bir fonksiyonun alacağı değerler için kesin olarak asimtotik sıkı olmayan bir üst sınır tanımlamak için o (küçük o) notasyonunu kullanıyoruz.
- Verilen bir g(n) fonksiyonu için, o(g(n)) fonksiyon topluluğu şu şekilde tanımlanır:
  - o(g(n)) = {f(n) : her pozisif c sabiti için aşağıda belirtilen eşitsizliği sağlayan $n_0 > 0$ değeri bulunabilir:

$$0 \leq f(n) < cg(n) \quad \text{her } n \geq n_0 \text{ için}\}$$

o Notasyonu

- Bu tanıma göre $2n = o(n^2)$ ancak $2n^2 \neq o(n^2)$
- Görüleceği üzere O ve o notasyonları birbirlerine benzemektedir.
- Ancak iki notasyon arasında önemli bir fark vardır:
  - $f(n) = O(g(n))$ için $0 \leq f(n) \leq cg(n)$ bazı sabit $c > 0$ değerleri için doğrudur.
  - $f(n) = o(g(n))$ için $0 \leq f(n) < cg(n)$ her $c > 0$ sabiti için doğrudur.
- Buna göre o notasyonunda n değeri sonsuza yaklaştığında f(n) fonksiyonu g(n) fonksiyonu ile karşılaştırıldığında önemsiz olur; yani:

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

ω Notasyonu

- Önceki bölümde gördüğümüz üzere Ω notasyonu asimtotik olarak sıkı olabilir veya olmayabilir.
  - Örneğin $2n^2 = \Omega(n^2)$ asimtotik olarak sıkıdır.
  - $2n^2 = \Omega(n)$ asimtotik olarak sıkı değildir.
- Bir fonksiyonun alacağı değerler için kesin olarak asimtotik sıkı olmayan bir alt sınır tanımlamak için ω (küçük omega) notasyonunu kullanıyoruz.
- Verilen bir g(n) fonksiyonu için, ω(g(n)) fonksiyon topluluğu şu şekilde tanımlanır:
  - ω(g(n)) = {f(n) : her pozisif c sabiti için aşağıda belirtilen eşitsizliği sağlayan n_0>0 değeri bulunabilir:

$$0 \le cg(n) < f(n) \quad \text{her } n \ge n_0 \text{ için}\}$$

ω Notasyonu

- Bu tanıma göre $n^2/2 = \omega(n)$ ancak $n^2/2 \neq \omega(n^2)$.
- Buna göre ω notasyonunda n değeri sonsuza yaklaştığında f(n) fonksiyonu g(n) fonksiyonuna göre oldukça büyür; yani:

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

Karşılaştırmalar

- Geçicilik (İng: Transitivity)
  - $f(n) = \Theta(g(n))$ ve $g(n) = \Theta(h(n))$ ise $f(n) = \Theta(h(n))$
  - $f(n) = O(g(n))$ ve $g(n) = O(h(n))$ ise $f(n) = O(h(n))$
  - $f(n) = \Omega(g(n))$ ve $g(n) = \Omega(h(n))$ ise $f(n) = \Omega(h(n))$
  - $f(n) = o(g(n))$ ve $g(n) = o(h(n))$ ise $f(n) = o(h(n))$
  - $f(n) = \omega(g(n))$ ve $g(n) = \omega(h(n))$ ise $f(n) = \omega(h(n))$
- Yansıma özelliği (İng: Reflexivity)
  - $f(n) = \Theta(f(n))$
  - $f(n) = O(f(n))$
  - $f(n) = \Omega(f(n))$

Karşılaştırmalar

- Simetri
  - $f(n) = \Theta(g(n))$ ancak ve ancak $g(n) = \Theta(f(n))$
- Transpoz Simetri
  - $f(n) = O(g(n))$ ancak ve ancak $g(n) = \Omega(f(n))$
  - $f(n) = o(g(n))$ ancak ve ancak $g(n) = \omega(f(n))$