Algoritma Analizi ve Büyük O Notasyonu

Şadi Evren ŞEKER

YouTube: Bilgisayar Kavramları

Algoritmalar

- Algoritmaların Özellikleri

- **Input** Girdi, bir kümedir,
- **Output** Çıktı, bir kümedir (çözümdür)
- **Definiteness** Kesinlik (algoritmanın adımlarının belirli olması),
- **Correctness** Doğruluk (bütün girdiler için algoritmanın tanımlı olması),
- **Finiteness** Sonluluk (çalışma adımlarının sonlu olması),
- **Effectiveness** Verimlilik (Her adımın ve sonuca ulaşan yolun)
- **Generality** Genellenebilirlik (bir problem kümesi için).

Örnek

- Kaba Kod (pseudocode, müsvette kod).
- Örnek: Bir kümede bulunan en büyük değeri bulan kod

- **procedure** max(a_1, a_2, …, a_n: integers)
- max := a_1
- **for** i := 2 **to** n
- **if** max < a_i **then** max := a_i
- {max en büyük elemandır}

Örnek 2

- Doğrusal arama (linear search): Verilen bir girdi içerisinde baştan sona kadar elemanlara teker teker bakarak bir elemanın aranması
- **procedure** linear_search(x: integer; a_1, a_2, …, a_n: integers)
- i := 1
- **while** (i ≤ n and x ≠ a_i)
- i := i + 1
- **if** i ≤ n **then** konum := i
- **else** konum := 0
- {konum, aranan elemanın sırasını göstermektedir, şayet 0 ise aranan eleman bulunamamıştır}

Örnek 3

- Şayet aranan liste sıralı ise ikili arama (binary search) daha iyi sonuç verir

- Sayı oyunu

Algorithm Examples

- **procedure** binary_search(x: integer; a_1, a_2, …, a_n: integers)
- i := 1 {i arama aralığının sol sınırı}
- j := n {j arama aralığının sağ sınırı}
- **while** (i < j)
- **begin**
- m := $\lfloor (i + j)/2 \rfloor$
- **if** x > a_m **then** i := m + 1
- **else** j := m
- **end**
- **if** x = a_i **then** konum := i
- **else** konum := 0
- {konum aranan x değerinin bulunduğu konumu gösterir, şayet 0 ise aranan x değeri bulunamamıştır}

Karmaşıklık (Complexity)

- Şayet girdi küçükse hafıza (Space) ve zaman (time) karmaşıklıkları çok önemli değildir.

- Örneğin n = 10, için ikili veya doğrusal arama kullanılması çok önemli değildir ama n = 2^{30} gibi eleman sayılarında aralarında yıllarca fark olabilir.

Karmaşıklık

- Örneğin, aynı problemi çözen A ve B gibi iki farklı algoritma olsun.
- A için zaman karmaşıklığı 5,000n, ve B için $\lceil 1.1^n \rceil$ olsun.
- n = 10 için A algoritması 50,000 adımda biterken, B sadece 3, adımda bitmektedir. B çok daha iyidir denebilir mi?
- n = 1000 için A 5,000,000 adımda biterken B requires $2.5 \cdot 10^{41}$ adımda bitmektedir.
- Peki hangisi iyidir?

Karmaşıklık

- A, büyük veri kümeleri için de kullanışlıyken B'nin kullanılamayacağı anlaşılmaktadır.
- Karmaşıklığın "büyümesi" (growth) çok daha önemlidir.
- Algoritmaların karşılaştırılması için algoritmaların zaman ve hafıza karmaşıklıklarındaki büyüme karşılaştırılır.

Karmaşıklık

- A ve B algoritmalarının karmaşıklıkları

Girdi Boyu	Algoritma A	Algoritma B
n	5,000n	$\lceil 1.1^n \rceil$
10	50,000	3
100	500,000	13,781
1,000	5,000,000	$2.5 \cdot 10^{41}$
1,000,000	$5 \cdot 10^9$	$4.8 \cdot 10^{41392}$

Bir fonksiyonun büyümesi (Growth)

- Fonksiyonların büyüme hızını matematikte big-o olarak ifade edilen fonksiyon gösterir.

- Tanım: f ve g fonksiyonları reel sayılardan reel sayılara tanımlı iki fonksiyon olsun.
- f(x) fonksiyonu için c ve k sabit olmak üzere O(g(x)) aşağıdaki şekilde tanımlanır

- $|f(x)| \leq c|g(x)|+k$

- x > k olmak koşuluyla

Fonksiyonların Büyümesi

- Karmaşıklık fonksiyonlarının büyümesini karşılaştırırken, f(x) ve g(x) fonksiyonları her zaman pozitif kabul edilir (neden?).
- Dolayısıyla gösterim sadeleştirilebilir
- $f(x) \leq C \cdot g(x)+k$,$x > k$ şartı ile.
- Şayet f(x)'in O(g(x)) olduğunu göstermek istiyorsak bunu sağlayan bir (C,k) ikilisi bulmamız yeterlidir.

Örnek

$f(x) = x^2 + 2x + 1$ fonksiyonunun büyüme fonksiyonunun x^2 olduğunu (yani $O(x^2)$ olduğunu) gösteriniz

- $x > 1$ için (bu aynı zamanda $x>k$):
- $x^2 + 2x + 1 \leq x^2 + 2x^2 + x^2$
- $\Rightarrow x^2 + 2x + 1 \leq 4x^2$
- Öyleyse $C = 4$ ve $k = 1$ için:
- $f(x) \leq Cx^2 + k$, $x > k$ durumu sağlanır.
- $\Rightarrow f(x)$ is $O(x^2)$.

Fonksiyonların Büyüme

- Soru: Şayet f(x) için $O(x^2)$'dir deniyorsa aynı zamanda $O(x^3)$'tür de denebilir mi?

- **Evet.** x^3 fonksiyonu x^2'den daha hızlı büyür dolayısıyla x^3 fonksiyonu da f(x)'den daha hızlı büyüyecektir.

- Ancak karşılaştırma yapabilmek için her zaman bulunabilecek en küçük büyüme fonksiyonunu bulmak isteriz.

Fonksiyonların Büyümesi

- Sık karşılaşılan fonksiyonlar:
- $n \log n$, 1, 2^n, n^2, $n!$, n, n^3, $\log n$

- En yavaştan hızlıya doğru sıralanmış hali:
- 1
- $\log n$
- n
- $n \log n$
- n^2
- n^3
- 2^n
- $n!$

Uysal Karmaşıklık

- Polinom zamanda çözülebilen algoritmalara uysal (tractable) ismi verilir.

- Polinom zamandan daha hızlı büyüyen fonksiyonlara uysal olmayan (untractable) ismi verilir.

Örnek 1

- Aşağıdaki algoritma ne işe yarar?
- **Procedure** gizemli(a_1, a_2, …, a_n: integers)
- m := 0
- **for** i := 1 to n-1
- **for** j := i + 1 to n
- **if** $|a_i - a_j| > m$ **then** $m := |a_i - a_j|$
- {m verilen girdideki herhangi iki sayı arasındaki en uzak mesafeyi verir}
- Karşılaştırma: n-1 + n-2 + n-3 + … + 1
- $= (n - 1)n/2 = 0.5n^2 - 0.5n$
- Zaman karmaşıklığı $O(n^2)$.

Örnek 2

- Aynı problemi çözen farklı bir algoritma
- **procedure** max_diff(a_1, a_2, …, a_n: integers)
- min := a1
- max := a1
- **for** i := 2 to n
- **if** a_i < min **then** min := a_i
- **else** if a_i > max **then** max := a_i
- m := max - min
- Karşılaştırma Sayıları: 2n - 2
- Zaman Karmaşıklığı O(n).