BİLGİSAYAR BİLİMLERİNE GİRİŞ DERSİ ALGORİTMA KISMI SLAYT DEĞERLENDİRME VE ÇÖZÜM REHBERİ

Slayt İncelemeleri, Algoritma Analizleri, Soruların Detaylı Çözümleri ve Java Karşılaştırmaları

Bu doküman, sunulan 14 adet algoritma slaytının her birini detaylı olarak incelemek, slaytlarda yer alan teorik soruları akademik düzeyde yanıtlamak, sunulan algoritmaların çalışma mantıklarını adım adım izlemek (trace etmek) ve Java dilinde zaman-hafıza karmaşıklığı analizlerini sunmak amacıyla hazırlanmıştır.

Slayt 1: Algoritma Nedir ve Özellikleri Nelerdir?

1. Slayt Sorusu:

• Algoritma ve özellikleri nelerdir? Açıklayınız.

2. Detaylı Açıklama ve Cevap:

Algoritma Tanımı: Algoritma, belirli bir problemi çözmek, bir görevi yerine getirmek ya da bir sonuca ulaşmak için takip edilen, başlangıcı ve sonu net olarak tanımlanmış, sonlu sayıda mantıksal adımdan oluşan işlem dizisidir.

Algoritmanın Temel Özellikleri:
1. Kesinlik (Definiteness): Algoritmanın her bir adımı açık, net ve tek bir anlama gelecek şekilde tanımlanmalıdır. İki anlamlılığa (ambiguity) izin verilemez.
2. Sonluluk (Finiteness): Algoritma sonsuz döngüye girmemeli; her türlü girdi için belirli ve sonlu sayıda adımdan sonra tamamlanmalıdır.
3. Girdi (Input): Algoritmanın dışarıdan aldığı sıfır veya daha fazla sayıda tanımlı girdi verisi bulunabilir.
4. Çıktı (Output): Algoritma en az bir veya daha fazla çıktı/sonuç üretmelidir. Girdiler ile çıktılar arasında tanımlı bir ilişki bulunur.
5. Etkinlik / Yapılabilirlik (Effectiveness): Algoritmadaki her adım temel, uygulanabilir ve pratik olmalıdır. İnsan tarafından kağıt-kalemle ya da bilgisayar tarafından temel komutlarla yürütülebilir olmalıdır.
6. Genellik (Generality): Bir algoritma sadece tek bir özel değer için değil, tanımlandığı problem sınıfındaki tüm geçerli girdiler için doğru çalışmalıdır.

Slayt 2: Algoritmaların İfade Edilme Türleri

1. Slayt Sorusu:

• Algoritmaların ifade edilme türlerinden üç tanesini örneklerle açıklayınız.

2. Detaylı Açıklama ve Cevap:

Algoritmalar genel olarak üç temel yapıda ifade edilir:

1. Doğal Dil (Metinsel İfade): Algoritmanın günlük konuşma dili (Türkçe, İngilizce vb.) kullanılarak adım adım yazılmasıdır.
Örnek (İki sayının toplamını bulan algoritma):
Adım 1: Başla.
Adım 2: Birinci sayıyı (A) oku.
Adım 3: İkinci sayıyı (B) oku.
Adım 4: Toplam = A + B işlemini yap.
Adım 5: Toplam değerini ekrana yazdır.
Adım 6: Bitir.

2. Sözde Kod (Pseudocode): Herhangi bir programlama dilinin katı sözdizimi (syntax) kurallarına bağlı kalmadan, yarı programlama yarı doğal dil kullanarak algoritmanın mantıksal akışını gösteren ifadedir.
Örnek:
```
BAŞLA
OKU A, B
C = A + B
YAZ C
BİTİR
```

3. Akış Şemaları (Flowcharts): Algoritma adımlarının standart geometrik şekiller (elips: başla/bitir, paralelkenar: bilgi girişi, dikdörtgen: işlem, eşkenar dörtgen: karar/koşul) ve oklar kullanılarak görselleştirilmesidir.

Slayt 3: Akış Şeması Türleri

1. Slayt Sorusu:

• Sıralı, Mantıksal ve Döngülü akış şemalarını örneklerle açıklayınız.

2. Detaylı Açıklama ve Cevap:

1. Sıralı (Sequential) Akış Şemaları:
İşlem adımlarının yukarıdan aşağıya doğru hiçbir dallanma (koşul) veya döngü içermeden, düz bir sıra halinde yürütüldüğü yapılardır.

Örnek: Kullanıcıdan alınan yarıçap (r) ile dairenin alanını hesaplama: Başla -> r Oku -> Alan = 3.14 * r * r -> Alan Yaz -> Bitir.

2. Mantıksal / Dallanmalı (Conditional/Decision) Akış Şemaları:
Bir koşula/şarta bağlı olarak akışın farklı yollara ayrıldığı (Eğer / Değilse yapısı) şemalardır.
Örnek: Girilen sayının pozitif mi negatif mi olduğunu kontrol etme: Başla -> Sayı Oku -> Eğer Sayı >= 0 ise 'Pozitif' Yaz, Değilse 'Negatif' Yaz -> Bitir.

3. Döngülü (Iterative/Looping) Akış Şemaları:
Belirli bir işlem bloğunun bir koşul sağlandığı sürece tekrar tekrar çalıştırıldığı yapılardır.
Örnek: 1'den 10'a kadar olan sayıları ekrana yazırma: Başla -> i = 1 -> Döngü (i <= 10): Yaz(i), i = i + 1 -> Bitir.

Slayt 4: Algoritmanın Doğruluğu

1. Slayt Sorusu:

Algoritmanın doğruluğu ne demektir? Hangi tür bir durumda Algoritma yanlıştır denir?

2. Detaylı Açıklama ve Cevap:

Algoritmanın Doğruluğu (Correctness): Bir algoritmanın, problem tanımına uygun olan her türlü geçerli girdi (valid input) için beklenen ve doğru olan çıktıyı sonlu bir süre içerisinde üretmesi durumuna algoritmanın doğruluğu denir. Doğruluk iki ana bileşenden oluşur:
• *Kısmı Doğruluk (Partial Correctness):* Algoritma durursa doğru sonucu verir.
• *Tam Doğruluk (Total Correctness):* Algoritma kesinlikle durur ve doğru sonucu verir.

Algoritmanın Yanlış Sayıldığı Durumlar:
1. Yanlış Çıktı Üretmesi: Geçerli bir girdi için beklenen matematiksel/mantıksal sonuçtan farklı veya hatalı bir değer üretmesi.
2. Sonlanmaması (Sonsuz Döngü / Infinite Loop): Algoritmanın durma koşulunun gerçekleşmemesi sebebiyle sonsuza kadar çalışmaya devam etmesi.
3. İstisna / Tanımsız Durum Hataları: Sıfıra bölme, dizi sınırlarını aşma (Out of Bounds) gibi durumlarda algoritmanın çökmesi veya kilitlenmesi.

Slayt 5: Algoritma Analizi

1. Slayt Sorusu:

Algoritma analizi ne demektir? Algoritma Analizi hangi durumları ele alır?

2. Detaylı Açıklama ve Cevap:

Algoritma Analizi: Bir algoritmanın çalışması için ihtiyaç duyduğu bilgisayar kaynaklarının (zaman ve hafıza) girdi boyutuna (n) bağlı olarak teorik ve matematiksel biçimde incelenmesidir.

Algoritma Analizinin Ele Aldığı Durumlar:
1. Zaman Karmaşıklığı (Time Complexity): Algoritmanın girdinin boyutu büyüdükçe kaç birim işlem/adım gerçekleştirdiğinin analizi.
2. Alan / Hafıza Karmaşıklığı (Space Complexity): Algoritmanın yürütülmesi sırasında bellekte (RAM/Call Stack) ne kadar yer kapladığının analizi.
3. Senaryo Analizleri (Asymptotic Notations):

- En Kötü Durum (Worst-Case / Big-O - O): Algoritmanın verebileceği en yavaş çalışma süresi sınırı.
- En İyi Durum (Best-Case / Big-Omega - Ω): Algoritmanın en ideal şartlardaki çalışma süresi.
- Ortalama Durum (Average-Case / Big-Theta - Θ): Rastgele girdiler karşısındaki beklenen performans.

Slayt 6: İki Sayının Toplamı Algoritması

1. Slayt Algoritması:

1. Başla
2. A ← 5
3. B ← 8
4. C ← A + B
5. Yaz(C)
6. Bitir

2. Ne İş Yapar ve Adım Adım Analiz:

Bu algoritma, sabit iki sayıyı (5 ve 8) toplayan ve sonucunu ekrana yazdıran sıralı (sequential) temel bir aritmetik algoritmadır.

Adım Adım İzleme (Execution Trace):

- Adım 2: A değişkenine 5 değeri atanır (A = 5).
- Adım 3: B değişkenine 8 değeri atanır (B = 8).
- Adım 4: A ve B toplanır (5 + 8 = 13) ve C değişkenine yazılır (C = 13).
- Adım 5: Ekrana 13 değeri yazdırılır.

Slayt 7: Üç Sayı Arasından En Küçüğü Bulma Algoritması

1. Slayt Algoritması:

1. Başla
2. A ← 7
3. B ← 3

4. C ← 10
5. Eğer A < B ve A < C ise EnKüçük ← A
6. Değilse eğer B < A ve B < C ise EnKüçük ← B
7. Aksi halde EnKüçük ← C
8. Yaz(EnKüçük)
9. Bitir

2. Ne İş Yapar ve Adım Adım Analiz:

Bu algoritma, verilen A, B ve C sayılarını (7, 3, 10) birbiriyle karşılaştırarak aralarındaki **en küçük sayıyı tespit eden ve ekrana yazdıran** mantıksal (dallanmalı) bir algoritmadır.

Adım Adım İzleme (Execution Trace):
• A = 7, B = 3, C = 10 değerleri tanımlanır.
• Adım 5 Kontrolü: (7 < 3 ve 7 < 10) → (Yanlış ve Doğru) = Yanlış. Bu adım atlanır.
• Adım 6 Kontrolü: (3 < 7 ve 3 < 10) → (Doğru ve Doğru) = Doğru! EnKüçük ← 3 atanır.
• Adım 8: Ekrana EnKüçük değeri olan 3 yazdırılır.

Slayt 8: Çift / Tek Sayı Kontrolü Algoritması

1. Slayt Algoritması (Eksik Metin Tamamlama ile):

1. Başla
2. N ← 11
3. Eğer N mod 2 = 0 ise
   Yaz("Çift Sayıdır")
  Değilse
   Yaz("Tek Sayıdır")
4. Bitir

2. Ne İş Yapar ve Slayttaki Eksikliklerin Tamamlanması:

Bu algoritma, verilen bir N tamsayısının (11) 2'ye bölümünden kalanı (mod) kontrol ederek sayının **Tek mi Çift mi** olduğunu belirler.

Slayttaki Noktalı Yerlerin Tamamlanması:
• N mod 2 = 0 ise Yaz("Çift Sayıdır") (Noktalı yer: Çift Sayıdır)
• Değilse Yaz("Tek Sayıdır") (Noktalı yer: Tek Sayıdır)

Çalışma Analizi: 11 mod 2 = 1'dir (0'a eşit değildir). Bu nedenle Değilse bloğu çalışır ve ekrana 'Tek Sayıdır' yazdırılır.

Slayt 9: Faktöriyel Hesaplama Algoritması

1. Slayt Algoritması:

1. Başla
2. N ← 4
3. F ← 1
4. i ← 1
5. Döngü i ≤ N olduğu sürece
   F ← F * i
   i ← i + 1
6. Döngü bittiğinde Yaz(F)
7. Bitir

2. Ne İş Yapar ve Adım Adım Analiz:

Bu algoritma, verilen bir N sayısının (4) faktöriyel değerini (N! = 1 * 2 * ... * N) döngüsel (iteratif) yöntemle hesaplar.

Adım Adım İzleme Table (Trace Table):

Adım / Adım İterasyon	i Değeri	F (F faktör)	Koşul (i <= 4)
Başlangıç	1	1	1 <= 4 (Doğru)
1. İterasyon	2 (1+1)	1 * 1 = 1	2 <= 4 (Doğru)
2. İterasyon	3 (2+1)	1 * 2 = 2	3 <= 4 (Doğru)
3. İterasyon	4 (3+1)	2 * 3 = 6	4 <= 4 (Doğru)

4. İterasyon sonunda F = 6 * 4 = 24 olur, i = 5 olur. 5 <= 4 (Yanlış) olduğu için döngü biter ve ekrana **24** yazdırılır.

Slayt 10: 1'den N'e Kadar Olan Sayıların Toplamı Algoritması

1. Slayt Algoritması:

1. Başla
2. N ← 5
3. Toplam ← 0
4. i ← 1

5. Döngü i ≤ N olduğu sürece
   Toplam ← Toplam + i
   i ← i + 1
6. Yaz(Toplam)
7. Bitir

2. Ne İş Yapar ve Adım Adım Analiz:

Bu algoritma, 1'den N'e (5) kadar olan ardışık tamsayıların toplamını (1 + 2 + 3 + 4 + 5) hesaplayan döngüsel bir algoritmadır.

İzleme:

• i = 1 → Toplam = 0 + 1 = 1
• i = 2 → Toplam = 1 + 2 = 3
• i = 3 → Toplam = 3 + 3 = 6
• i = 4 → Toplam = 6 + 4 = 10
• i = 5 → Toplam = 10 + 5 = 15
• i = 6 → Döngü sonlanır. Ekrana **15** yazdırılır.

Slayt 11: Sayıyı Tersine Çevirme (Reverse Number) Algoritması

1. Slayt Algoritması:

1. Başla
2. N ← 1234
3. T ← 0
4. Döngü N > 0 olduğu sürece
   K ← N mod 10
   T ← T * 10 + K
   N ← N / 10 (tamsayı bölme)
5. Yaz(T)
6. Bitir

2. Ne İş Yapar ve Adım Adım Analiz:

Bu algoritma, verilen bir N sayısının (1234) basamaklarını ters çevirerek (4321) T değişkenine oluşturan algoritmadır.

İterasyon	N (Girdi)	K (N mod 10)	T (T * 10 + K)	Yeni N (N / 10)
1. İterasyon	1234	4	0*10 + 4 = 4	123

2. İterasyon	123	3	4*10 + 3 = 43	12
3. İterasyon	12	2	43*10 + 2 = 432	1
4. İterasyon	1	1	432*10 + 1 = 4321	0

N = 0 olduğunda döngü sonlanır ve ekrana **4321** yazdırılır.

Slayt 12: İki Değişkenin Değerini Yer Değiştirme (Swapping)

1. Slayt Sorusu ve Kod Satırları:

Z = X
X = Y
Y = Z

Yukarıdaki satırlarla ne işlem yapılmıştır? Açıklayınız.

2. Detaylı Açıklama ve Cevap:

Bu işlem, iki değişkenin (X ve Y) değerlerini geçici bir üçüncü değişken (Z - temp) kullanarak birbirleriyle takas etme (SWAP / Yer Değiştirme) işlemidir.

Çalışma Mantığı (Örnek: X = 5, Y = 9 olsun):
1. Z = X → Z değişkenine X'in değeri (5) kopyalanır. (Geçici hafıza)
2. X = Y → X değişkenine Y'nin değeri (9) atanır. (X artık 9 oldu)
3. Y = Z → Y değişkenine Z'de saklanan ilk X değeri (5) atanır. (Y artık 5 oldu)
Sonuç: X = 9 ve Y = 5 haline gelmiştir.

Slayt 13: Koşullu İfade İle Değişken Analizi

1. Slayt Sorusu:

a = 3 olarak değerlendirildiğinde a, b ve c değerleri sonuçta ne olur?

```
if (a > 5) then
  b = 2 * a
  c = b * b
else
  b = a + 3
  c = b * b
```

2. Detaylı Çözüm ve Cevap:

• Başlangıçta a = 3 olarak verilmiştir.
• Koşul Kontrolü: (a > 5) yani (3 > 5) kontrol edilir. Bu ifade **YANLIŞ (FALSE)** sonucunu verir.
• Akış 'else' (aksi halde) bloğuna geçer.
• b = a + 3 → b = 3 + 3 = 6 atanır.
• c = b * b → c = 6 * 6 = 36 atanır.

Sonuç Değerleri:

- a = 3
- b = 6
- c = 36

Slayt 14: Kapsamlı Teorik Sorular ve Java İteratif vs Rekürsif Analizi

Soru 1: Algoritmanın Giriş Verisi Neler Olabilir? Örneklerle Açıklayınız.

Algoritma girdileri problemin türüne göre çok çeşitli olabilir:
• Sayısal Veriler: Tam sayılar, ondalıklı sayılar (Örn: Yaş, maaş, matris boyutları).
• Metinsel / Karakter Verileri: Karakter dizileri (Örn: Kullanıcı adı, arama terimi).
• Mantıksal (Boolean) Veriler: True/False bayrakları (Örn: Kullanıcı aktif mi?).
• Veri Yapıları ve Nesneler: Diziler, bağlı listeler, ağaçlar, veritabanı kayıtları.
• Dosya ve Medya Verileri: Resim (görüntü işleme algoritmaları için pixel matrisi), ses dosyası, PDF dokümanı.

Soru 2: Algoritmanın Çıktısı Neler Olabilir? Örneklerle Açıklayınız.

Algoritma çıktıları şunlar olabilir:
• Sayısal / Metinsel Sonuç: Ekrana yazdırılan bir hesaplama sonucu (Örn: Faktöriyel = 120).
• Mantıksal Dönüş Değeri: Bir elemanın dizide bulunup bulunmadığı bilgisi (True/False).
• İşlenmiş / Sıralanmış Veri Yapısı: Sıralama algoritması sonucu elde edilen sıralı dizi.
• Dosya / Grafik Çıktısı: Oluşturulan bir grafik, işlenmiş bir fotoğraf dosyası veya rapor.
• Fiziksel / Gömülü Sistem Eylemi: Bir robot kolunun hareket etmesi veya uyarı sesinin çalması.

Soru 3: İteratif ve Rekürsif Ne Demektir?

• **İteratif (Tekrarlamalı):** Problemin çözümünün döngü yapıları (for, while, do-while) kullanılarak adım adım gerçekleştirilmesidir. Bellekte ekstra yığın çerçevesi (stack frame) oluşturmaz.
• **Rekürsif (Öz-Yinelemeli):** Bir fonksiyonun kendi içerisinden kendisini daha küçük parametrelerle tekrar çağırması esasına dayanan yöntemdir. Mutlaka sonlanmayı sağlayan bir **Temel Durum (Base Case)** içermelidir.

Soru 4: Yapı Bir ve Yapı İki İncelemesi & Java Kıyaslaması

• **YAPI BİR (Rekürsif Faktöriyel):** Metot kendi içerisinden F(n-1) olarak kendisini çağırmaktadır. n <= 1 durumu temel durumdur (base case).
• **YAPI İKİ (İteratif Faktöriyel):** Hesaplama işlemi herhangi bir öz-yineleme olmaksızın klasik for döngüsü ile yürütülmektedir.

Java Programı (Her İki Yöntemin Uygulaması ve Testi):

Java Test Kod Bloğu

```
public class FaktoriyelTest {
   // YAPI BİR: Rekürsif (Öz-Yinelemeli)
   public static long faktoriyelRekursif(int n) {
      if (n <= 1)
         return 1;
      return n * faktoriyelRekursif(n - 1);
   }

   // YAPI İKİ: İteratif (Döngüsel)
   public static long faktoriyelIteratif(int n) {
      long sonuc = 1;
      for (int i = 1; i <= n; i++) {
         sonuc *= i;
      }
      return sonuc;
   }

   public static void main(String[] args) {
      int n = 20; // Test parametresi

      // Rekürsif Performans Ölçümü
      long baslangicRek = System.nanoTime();
      long sonucRek = faktoriyelRekursif(n);
      long bitisRek = System.nanoTime();
      long sureRek = bitisRek - baslangicRek;

      // İteratif Performans Ölçümü
      long baslangicIter = System.nanoTime();
      long sonucIter = faktoriyelIteratif(n);
      long bitisIter = System.nanoTime();
      long sureIter = bitisIter - baslangicIter;

      System.out.println(n + "! Rekürsif Sonuç: " + sonucRek + " | Süre: " + sureRek + " ns");
```

```
        System.out.println(n + "! İteratif Sonuç: " + sonucIter + " | Süre: " + sureIter + " ns");
    }
}
```

ZAMAN ve HAFIZA (Zaman ve Alan Karmaşıklığı) Bakımından Kıyaslama:

Kriter / Metrik	YAPI BİR (Rekürsif)	YAPI İKİ (İteratif)
Zaman Karmaşıklığı (Time)	O(n) - n adet fonksiyon çağrısı	O(n) - n adet döngü adımı
Hafıza Karmaşıklığı (Space)	O(n) - Çağrı Yığını (Call Stack)	O(1) - Sabit Hafıza
Overhead (Ek Yük)	Yüksek (Stack Frame oluşturma maliyeti)	Yok (Yalnızca değişken güncelleme)
Risk Faktörü	StackOverflowError (Büyük n değerlerinde)	Yok (Güvenli bellek kullanımı)

Detaylı Kıyaslama Yorumu:
1. Zaman (Time) Bakımından: Teorik olarak her iki yöntem de n adım sürer O(n). Ancak pratikte Rekürsif yöntem, her öz-yinelemeli çağrıda işlemcinin adres saklaması ve bellek açması nedeniyle (Call Stack Overhead) İteratif yönteme göre nanosaniye düzeyinde daha yavaş çalışır.
2. Hafıza (Memory) Bakımından: İteratif yöntem bellekte sadece 'sonuc' ve 'i' değişkenlerini tuttuğu için O(1) alan kaplar. Rekürsif yöntem ise n adet fonksiyon çağrısını belleğin 'Call Stack' bölgesinde üst üste yığdığı için O(n) hafıza kaplar. n değeri çok büyüdüğünde rekürsif yöntem 'java.lang.StackOverflowError' hatası vererek çöker.