

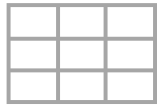
BİL3014

Algoritma Analizi

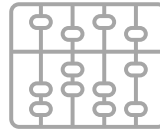
Giriş



Bilgisayar Biliminde Algoritma Analizinin Yeri



Veri Yapıları



Ayrık Matematik



Algoritma Analizi

Neler var neler?

01 Giriş

02 Asimptotik Notasyon ve Analizi

03 Böl Yönet Yaklaşımı ve Algoritmaları

04 Dinamik Programlama

05 Greedy Algoritmalar

06 Rekürsif Algoritmalar

07 Sıralama ve Arama Algoritmaları

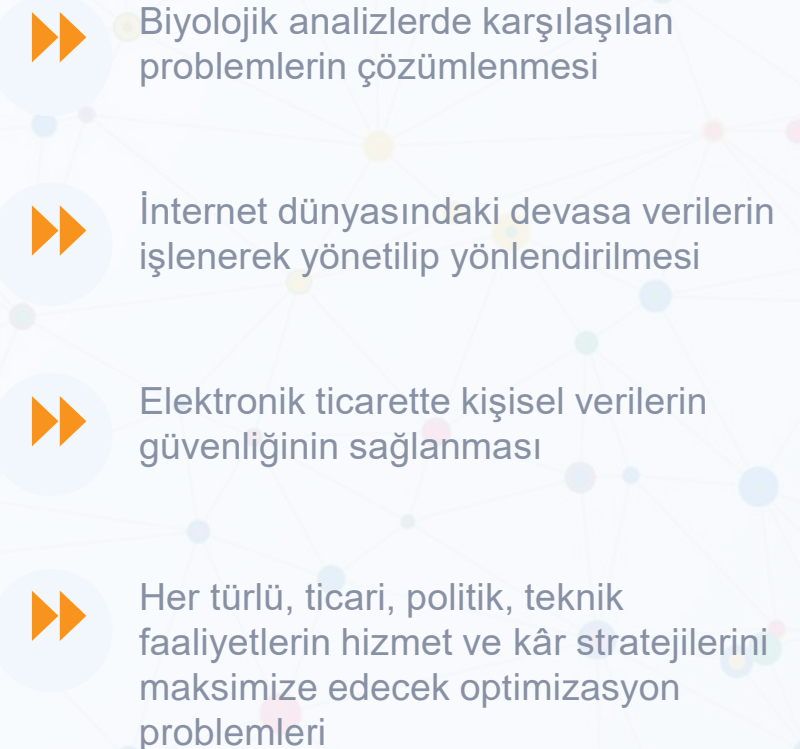
08 Graf Algoritmaları

09 Gelişmiş Veri Yapıları

10 Algoritma Dizayn Teknikleri

Algoritma Nedir?

Ne Tür Problemler Algoritma ile Çözülür?

- 
- ▶▶ Biyolojik analizlerde karşılaşılan problemlerin çözümlenmesi
 - ▶▶ İnternet dünyasındaki devasa verilerin işlenerek yönetilip yönlendirilmesi
 - ▶▶ Elektronik ticarete kişisel verilerin güvenliğinin sağlanması
 - ▶▶ Her türlü, ticari, politik, teknik faaliyetlerin hizmet ve kâr stratejilerini maksimize edecek optimizasyon problemleri

Birtakım Kavramlar

Algoritmalarla ilgili zaman zaman duyabileceğimiz kavramlar vardır. Kısaca Bakalım.



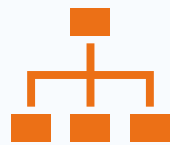
Veri

- Bilgisayar sistemlerinde saklanan ve işlenen bilgilerdir. Veriler, farklı formatlarda saklanabilir (örneğin, metin, ses, video, görüntü) ve farklı amaçlar için kullanılabilir (örneğin, analitik, saklama veya görüntüleme).



Veri Yapısı

- Verilerin saklanması ve işlenmesi için belirli bir yapı kullanılmasıdır. Veri yapıları, verilerin düzenli ve hızlı bir şekilde erişilmesini, işlenmesini ve depolanmasını sağlar. Örneğin, linked list, tree, stack, queue gibi.



Veri Modeli

- Verilerin nasıl saklanması ve işlenmesi gerektiğini belirleyen bir yapıdır. Veri modeli, verilerin ilişkilerini ve veriler arasındaki bağlantıları tanımlar. Örneğin, relational database model, object-oriented model, NoSQL model gibi.



Neden Veri Yapıları?

- Veri yapıları, büyük veri miktarlarının işlenmesini ve saklanmasını kolaylaştırır.
- Ayrıca, verilerin aranması, sıralanması veya değiştirilmesi gibi işlemlerin hızlı ve verimli bir şekilde gerçekleştirilmesini sağlar.



Neden Veri Yapıları?

Örnek

- Her biri satır başına ortalama 10 kelimeden ve yine ortalama 20 satırdan oluşan 3000 metin koleksiyonu olduğunu düşünelim.
 - →600,000 kelime
 - Bu metinler içinde “dünya” kelimesi ile eşleşecek bütün kelimeleri bulmak isteyelim
 - Doğru eşleştirme için yapılacak karşılaştırmanın 1 sn. sürdüğünü varsayalım.
-
- Doğrusal Arama:
 - 1 sn. X 600.000 Kelime = 166 Saat
 - İkili Arama:
 - Toplam Adım Sayısı $\log_2 N = \log_2 600000$ yaklaşık 20 iterasyon
 - 20 karşılaştırma = 20 sn
 - 20 sn. **»»»** 166 sa.



Neden Veri Modeli?

- *Veri modeli*, verilerin hızlı, verimli ve doğru bir şekilde işlenmesini ve saklanmasını sağlar.
- Ayrıca, verilerin anlaşılır ve kolayca kullanılabilir hale gelmesini ve veri entegrasyonu veya veri değişimi gibi fonksiyonların gerçekleştirilmesini kolaylaştırır.
- *Veri modeli*, verilerin uygun bir şekilde yapılandırılmasını ve anlaşılmasını sağlar, bu da veri analitik veya veri madenciliği gibi uygulamalar için gerekli verilerin daha hızlı ve doğru bir şekilde elde edilmesini sağlar.

Birtakım Kavramlar

Algoritmalarla ilgili zaman zaman duyabileceğimiz kavramlar vardır. Kısaca Bakalım.

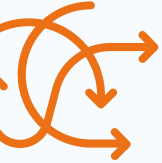
Program

- Bir problemin, çeşitli makine dilleri kullanılarak bilgisayar ve bilgisayar mantığında çalışan cihazlara, onların anlayabileceği şekilde işlenmesine **programlama**, bunun sonucunda oluşan belirli bir çalışma algoritmalarına sahip kodlara ise **program** denir.



Verimlilik ve Zor Problemler

- Algoritma analizinin büyük bölümü **verimli** algoritmalarla ilgilidir. Verimliliğin genel ölçüsü **hız**dır, yani sonucu üretmek için gereken süredir. Bununla birlikte verimli çözümü bilinmeyen problemler de vardır.
- **NP-tam** olarak bilinen problemler için verimli bir algoritma hiç bulunmamasına rağmen böyle bir algoritma bulunamayacağı da kanıtlanamamıştır.



Birtakım Kavramlar

Algoritmalarla ilgili zaman zaman duyabileceğimiz kavramlar vardır. Kısaca Bakalım.

Algoritmik Çözüm

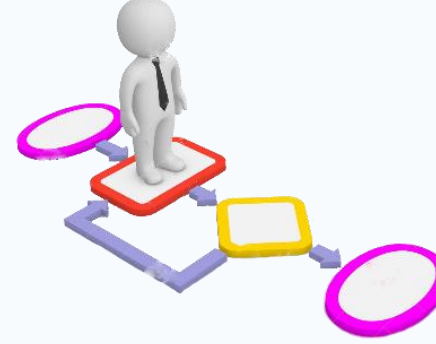


- Algoritmik çözüm, bir problemin belirli bir yöntemle veya algoritma kullanarak çözülmesidir.
- Algoritmik çözüm, problemi parçalara bölerek, çözümü kolaylaştıran adımlar oluşturarak gerçekleştirilir.

Birtakım Kavramlar

Algoritmalarla ilgili
zaman zaman
duyabileceğimiz
kavramlar vardır.
Kısaca Bakalım.

Algoritmik Çözüm



- Algoritmik çözüm sürecinde, öncelikle problem tanımlanır ve analiz edilir.
- Ardından, en uygun algoritma seçilir ve uygulanır.
- Uygulamanın sonunda, algoritmanın doğruluğu ve verimliliği test edilir.
- Algoritmik çözüm, birçok farklı problem türünü çözebilir, örneğin matematiksel, veri yapıları, görüntü işleme, sosyal ağ analizi gibi.

Birtakım Kavramlar

Algoritmalarla ilgili
zaman zaman
duyabileceğimiz
kavramlar vardır.
Kısaca Bakalım.

Pseudo Kod



- Pseudo Kod: Algoritmaların veya programların tanımlanması için kullanılan yapısal bir dildir.
- Programlama dillerine benzer, ancak sınırlamalar veya detaylı sözdizimi yoktur.
- Algoritma veya programın nasıl çalıştığını tanımlamak için basit ve anlaşılır bir dildir.

Birtakım Kavramlar

Algoritmalarla ilgili zaman zaman duyabileceğimiz kavramlar vardır. Kısaca Bakalım.

Pseudo Kod

```
ALGORİTMA: En Büyük Sayı
```

```
BEGIN
```

```
  Tanımla: liste, sayı, enbüyük  
  enbüyük = 0
```

```
  FOR i IN 0 to liste.size-1 DO
```

```
    sayı = liste[i]
```

```
    IF sayı > enbüyük THEN
```

```
      enbüyük = sayı
```

```
    END IF
```

```
  END FOR
```

```
  PRINT "En büyük sayı:", enbüyük
```

```
END
```

“

Algoritma Analizi Nedir?

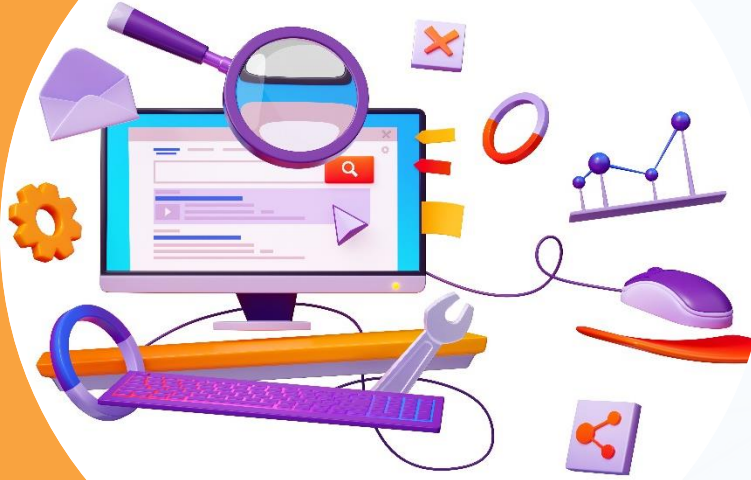
Algoritma Analizi,
Algoritmaların veya
programların
verimliliklerinin
ölçülmesi ve
değerlendirilmesi
olarak tanımlanır.



- Algoritmaların bellek ve zaman gereksinimleri, performansları ve scalability gibi özelliklerini değerlendirir.
- Burada temel hesap birimi seçilir ve programın görevini yerine getirebilmesi için bu işlemde kaç adet yapılması gerektiğini bulmaya yarayan bir bağıntı hesaplanır.
- Eğer bu bağıntı zamanla ilgiliyse çalışma hızını, bellek gereksinimiyle ilgiliyse bellek gereksinimi ortaya koyar.

“

Algoritma Analizi Nedir?



- Algoritma Analizi bilgisayar programlarının performans ve kaynak kullanımı üzerine teorik çalışmadır.
- Algoritma analizinde öncelikle ve özellikle ***performans*** üzerinde durulur.
- Bilgisayar Programlarında işlerin nasıl daha hızlı gerçekleştirilebileceği ele alınır.

Algoritmadan Beklenenler

- **Correctness (doğruluk):** Algoritmanın belirli bir girdi için doğru çıktı vermesi beklenir.
- **Efficiency (verimlilik):** Algoritmanın girdi boyutuna göre kabul edilebilir bir zaman veya alan karmaşıklığına sahip olması beklenir.
- **Scalability (ölçeklenebilirlik):** Algoritmanın büyük girdilerle bile çalışabilmesi beklenir.
- **Generality (genellik):** Algoritmanın farklı girdiler için çalışabilmesi beklenir.
- **Simplicity (basitlik):** Algoritmanın anlaşılır ve kolay uygulanabilir olması beklenir.
- **Robustness (dayanıklılık):** Algoritmanın hatalar veya beklenmedik durumlar karşısında stabil çalışması beklenir.
- **Reusability (tekrar kullanılabilirlik):** Algoritmanın farklı projelerde kullanılabilmesi beklenir.

Algoritmadan Beklenenler



Performans Nerede?

Algoritmanın belirli bir girdi için doğru çıktı vermesi

Algoritmanın girdi boyutuna göre kabul edilebilir bir

çalışma süresi (Zaman Karmaşıklığı): Algoritmanın büyük girdilerle bile çalışabilmesi

Algoritmanın çalışabilmesi beklenir.

Algoritmanın anlaşılır ve kolay uygulanabilir olması

Algoritmanın hatalar veya beklenmedik durumlar

durumlarında çalışabilmesi (Genel Kullanılabilirlik): Algoritmanın farklı projelerde

kullanılabilmesi beklenir.



Performans

- Algoritma analizinde ***performans***, programların veya algoritmaların etkililiğinin ve verimliliğinin ölçülmesi ve değerlendirilmesidir.
- Performans, bir programın veya algoritmanın işletim sistemi, bellek, disk veya diğer kaynaklar gibi çevresel faktörlere göre hızını veya verimliliğini ifade eder.





Performans Neden Önemli?

- Programlar veya algoritmalar genellikle büyük veri kümeleri veya zaman kısıtlı problemler gibi zorlu uygulamalar için tasarlandıklarında yavaş çalışabilir veya yanıt vermeyebilir.
- Performans, kullanıcıların beklentilerini karşılamak, sistem güvenliğini ve verimliliğini artırmak, süreçlerin verimliliğini artırmak veya işletmelerin maliyetlerini azaltmak gibi çeşitli amaçlar için kritik öneme sahiptir.
- Algoritma analizi, performansı iyileştirmek için gereken değişikliklerin veya optimizasyonların belirlenmesine yardımcı olabilir.



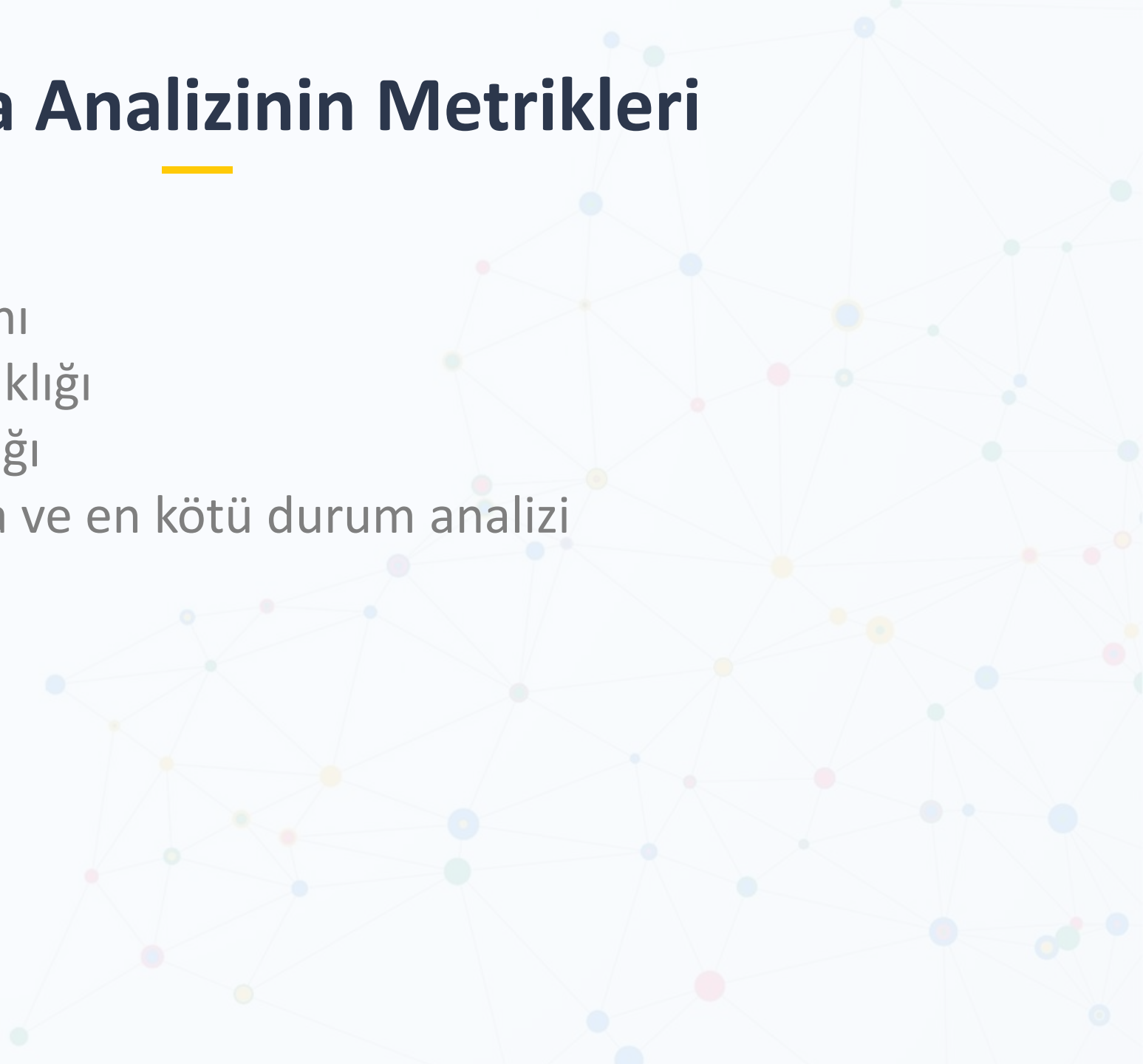
Bir algoritmayı neden analiz ederiz ?

- Algoritmanın verimliliğini ve performansını ölçmek ve değerlendirmek için,
- Algoritmanın bellek, disk, işlem gücü ve diğer kaynaklar gibi çevresel faktörlere göre hızını ve verimliliğini ölçmek ve iyileştirmek için,
- Problem ve algoritmaları zorluk derecesine göre sınıflandırmak için,
- Performansı tahmin etmek, algoritmaları karşılaştırmak ve parametrelerini ayarlamak için.



Algoritma Analizinin Metrikleri

- Yürütme Zamanı
- Zaman Karmaşıklığı
- Alan Karmaşıklığı
- En iyi, ortalama ve en kötü durum analizi



Yürütme Zamanı

- *Yürütme zamanı*, bir algoritmanın bellekte veya bir işletim sisteminde çalışması sırasında harcadığı zamanı ifade eder.
- Algoritmanın girdi boyutu, hızı, sıklığı gibi birçok faktöre bağlı olarak değişebilir.
- Örneğin; iki farklı veri yapısı için yapılan arama işlemlerinin yürütme zamanları farklı olabilir. Aynı veri boyutunda, bir **dizi** içinde yapılan lineer arama, bir başka veri yapısı olan **AVL tree** içinde yapılırsa daha hızlı sonuç alınabilir.
- Başka bir örnek olarak, bir algoritmanın veri boyutu arttıkça yürütme zamanı da artar. Bir algoritmanın veri boyutu 10 olduğunda yürütme zamanı 1 saniye olsa, veri boyutu 100 olduğunda yürütme zamanı 10 saniye olabilir.



Zaman Karmaşıklığı

- Bir algoritmanın çalışması için gereken zamanın bir fonksiyonu olarak ifade edilir.
- Bir algoritmanın zaman karmaşıklığı, algoritmanın girdisinin boyutu (n) ile değiştiğinde, algoritmanın yürütme süresinin ne kadar artacağını belirler.
- Zaman karmaşıklığı, genellikle Big-O gösterimi kullanılarak ifade edilir.
- Big-O gösterimi, algoritmanın en kötü durum zaman karmaşıklığını ifade eder.
- Bu, algoritmanın en kötü durumda, girdinin ne kadar büyük olursa olsun, ne kadar sürede çalışacağını gösterir.
- Örneğin, bir algoritmanın zaman karmaşıklığı $O(n^2)$ ise, girdinin boyutu (n) iki katına çıktığında, algoritmanın çalışma süresi dört katına çıkacaktır.



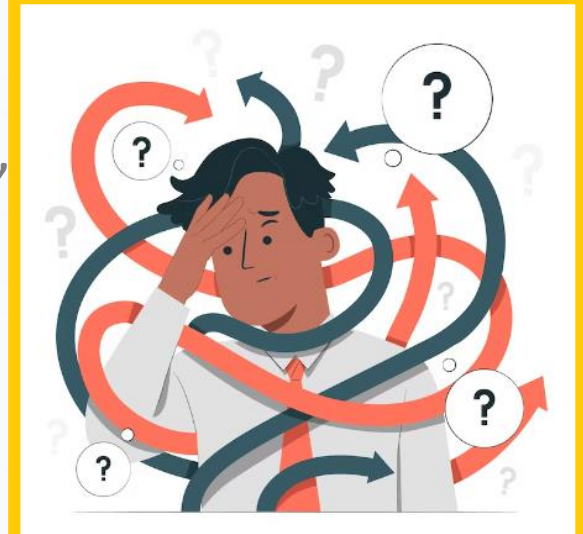
Zaman Karmaşıklığı

- Zaman karmaşıklığı, algoritmanın performansını değerlendirmek için önemlidir.
- Bir algoritmanın zaman karmaşıklığı ne kadar düşük olursa, o kadar hızlı çalışacaktır.
- Örneğin, bir dizideki tüm elemanların toplamını bulan bir algoritma düşünelim. Bu algoritmanın bir döngü içinde tüm elemanları toplaması gerekiyor. Bu durumda, algoritmanın zaman karmaşıklığı $O(n)$ olacaktır, çünkü dizinin boyutu arttıkça, algoritmanın çalışma süresi de doğru orantılı olarak artacaktır.



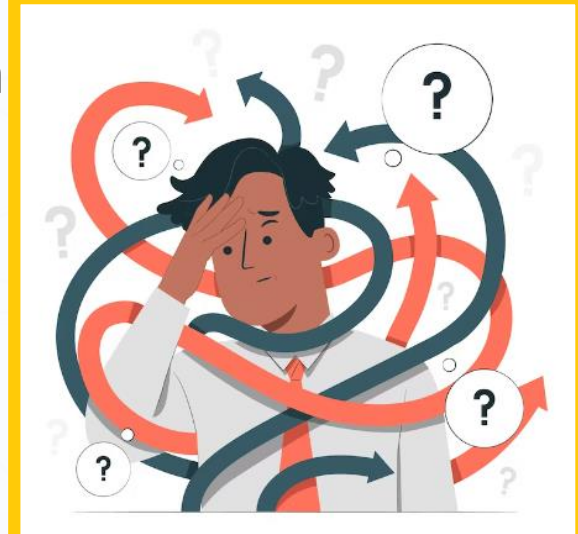
Alan Karmaşıklığı

- Alan karmaşıklığı (Space complexity), bir algoritmanın çalışması için gereken bellek alanının miktarını ölçen bir performans metriğidir.
- Bir algoritmanın alan karmaşıklığı, çoğu zaman problem boyutuna bağlıdır ve genellikle problem boyutu arttıkça artar.
- Alan karmaşıklığı, bazı algoritmaların sınırlarını belirlemek için kullanılabilir.
- Örneğin, bir algoritmanın alan karmaşıklığı sabit kalırken, zaman karmaşıklığı problem boyutuyla artabilir.
- Bu, bazı durumlarda alan karmaşıklığı ile zaman karmaşıklığı arasında bir denge sağlanması gerektiği anlamına gelir.



Alan Karmaşıklığı

- Örneğin, bir dizideki elemanları ters çevirme işlemini yapan bir algoritma düşünelim. Bu algoritmanın alan maliyeti, sadece dizi boyutu ile ilgilidir ve $O(1)$ (sabit) dir. Ancak, zaman karmaşıklığı dizi boyutu ile doğrusal olarak artar ve $O(n)$ (lineer) dir.
- Başka bir örnek olarak, merge sort algoritmasını ele alalım. Bu algoritmanın alan maliyeti, n elemanlı bir dizinin sıralanması için $O(n)$ bellek kullanır. Ancak, merge sort'un zaman karmaşıklığı $O(n \log n)$ olduğundan, alan maliyeti ile zaman karmaşıklığı arasında bir denge sağlanması gerekir.
- Bu nedenle, algoritma analizinde zaman karmaşıklığı ve alan maliyeti gibi metriklerin her ikisi de önemlidir.



Alan Maliyeti

- Alan maliyeti, bir algoritmanın çalışması için gereken bellek alanının miktarını ifade eder.
- Bu bellek alanı, verilerin depolanması, geçici değişkenlerin saklanması ve programın yürütülmesi sırasında işletim sistemi tarafından tahsis edilen gerçek bellek miktarıdır.
- Örneğin, bir algoritmanın *alan karmaşıklığı* $O(n)$ olsun. Bu, en kötü durumda, algoritmanın girdi boyutunun (n) doğrusal bir fonksiyonu kadar bellek kullanacağı anlamına gelir.
- Ancak, gerçek zamanlı "alan maliyeti" bu tahminin tam olarak üstünde veya altında olabilir.
- Bu, işletim sistemi ve donanım kaynaklarına, bellek yönetimine, derleme optimizasyonlarına vb. bağlıdır.



En iyi-Ortalama-En kötü Durumlar

- Bir algoritmanın performansı, verilen girdi boyutuna bağlı olarak farklı şekillerde değişebilir.
- Bu farklı senaryolara *en iyi durum (best case)*, *ortalama durum (average case)* ve *en kötü durum (worst case)* olarak adlandırılır.



En iyi-Ortalama-En kötü Durumlar

- **En iyi durum:** Algoritmanın en iyi durumda çalışması, en az sayıda işlem yapacağı durumdur.
- En iyi durum, algoritmanın olay sayısını veya eleman sayısını doğru tahmin edebildiği veya en iyi durum için özel olarak optimize edildiği durumlarda gerçekleşebilir.
- En iyi durumda çalışan bir algoritmanın zaman veya alan karmaşıklığı genellikle diğer senaryolara göre daha azdır.



En iyi-Ortalama-En kötü Durumlar

- **Ortalama durum:** Ortalama durum, verilerin rastgele bir şekilde dağıldığı senaryoyu ifade eder.
- Bu senaryo, bir algoritmanın gerçek dünya senaryolarında karşılaşılabileceği duruma en yakın senaryodur.
- Genellikle, bir algoritmanın ortalama durum performansı, en iyi durum performansından daha kötüdür, ancak en kötü durum performansından daha iyidir.



En iyi-Ortalama-En kötü Durumlar

- **En kötü durum** senaryosu, bir algoritmanın en yavaş şekilde çalıştığı durumdur.
- Bu durum genellikle, verilerin en kötü şekilde organize edildiği senaryolarda ortaya çıkar.
- Örneğin, bir dizi tamamen ters sıralandığında, sıralama algoritması için en kötü durum senaryosudur.



En iyi-Ortalama-En kötü Durumlar

- Bir algoritmanın en iyi, ortalama ve en kötü durum performansı, algoritmanın tasarımında ve analizinde önemli bir rol oynar.
- Algoritmanın en iyi performansı, bazı durumlarda gereksiz bir performans artışına neden olabilirken, en kötü performans ise kullanılabilirliği sınırlandırabilir.
- Bu nedenle, bir algoritmanın tasarımı ve analizi, tüm durumlar göz önünde bulundurularak yapılmalıdır.

