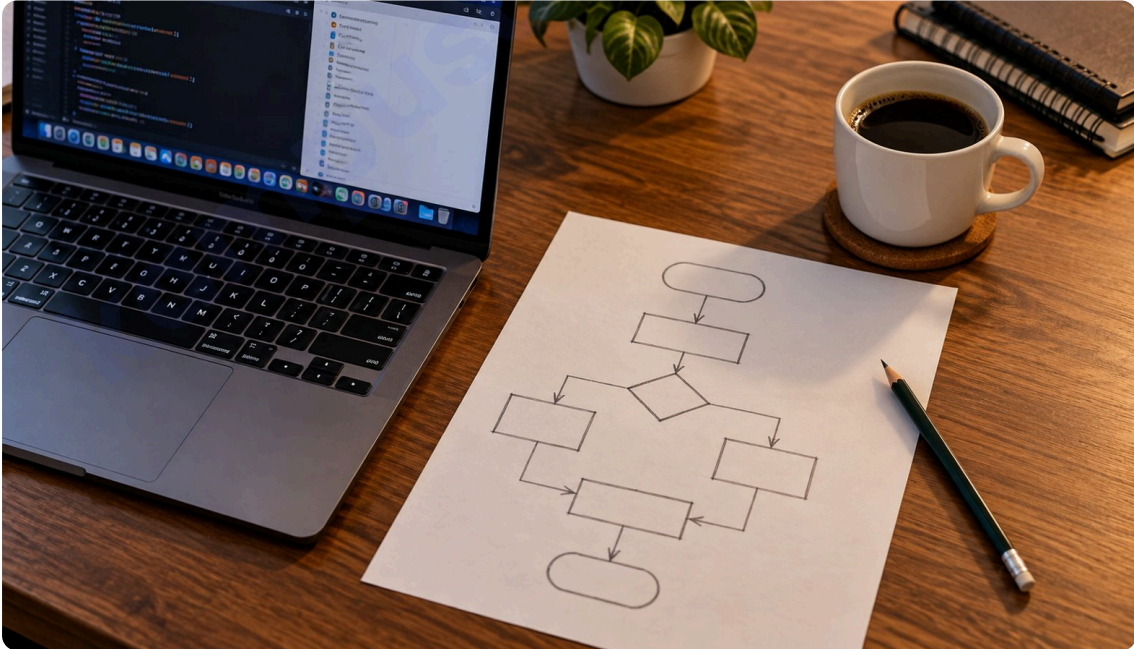


B Ö L Ü M

8

Veri Yapıları ve Algoritmalar



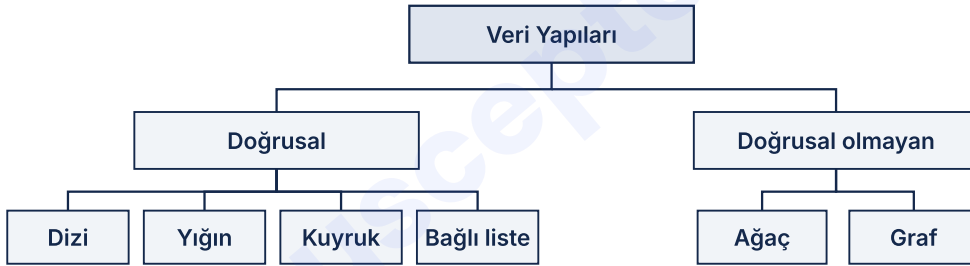
8.1 Veri Yapısı Nedir?

TANIM

Veri yapısı, verilerin bilgisayar belleğinde nasıl düzenleneceğini, saklanacağını ve onlara nasıl erişileceğini belirleyen yapıdır.

Kütüphanedeki kitapların raflara konu ve yazar sırasıyla dizilmesi, aranan kitabı kolayca bulmayı sağlar. Veri yapıları da verileri belirli bir düzende tutarak eklemeyi, silmeyi, aramayı ve sıralamayı kolaylaştırır. Doğru veri yapısını seçmek, programın hızını ve bellek kullanımını doğrudan etkiler.

Veri yapıları elemanların dizilişine göre ikiye ayrılır: elemanların bir sıra hâlinde ard arda dizildiği **doğrusal** yapılar ve elemanların dallanarak ya da ağ şeklinde bağlandığı **doğrusal olmayan** yapılar.



Temel veri yapılarının sınıflandırılması

8.2 Dizi (Array)

TANIM

Dizi, aynı türden verilerin bellekte ard arda saklandığı ve her elemana bir **indis** (sıra numarası) ile erişilen veri yapısıdır.

Beş öğrencinin notlarını `not1`, `not2` ... gibi ayrı ayrı değişkenlerde tutmak yerine, hepsi tek bir dizide tutulur. Elemanlara köşeli parantez içindeki indis ile ulaşılır. Programlama dillerinin çoğunda **ilk elemanın indisi 0'dır**; bu yüzden n elemanlı bir dizinin son elemanının indisi $n - 1$ olur.

		notlar[2] = 60			
notlar	70	85	60	90	75
	[0]	[1]	[2]	[3]	[4]

indis (sıra numarası) 0'dan başlar

Beş elemanlı notlar dizisi

Avantajı İstenen elemana indisiyle doğrudan ve çok hızlı ulaşılır: notlar[3] = 90.

Dezavantajı Boyutu genellikle baştan belirlenir. Araya eleman eklemek ya da aradan eleman silmek için diğer elemanların kaydırılması gerekir.

8.3 Yığın (Stack)

TANIM

Yığın, elemanların yalnızca bir uçtan (tepeden) eklenip çıkarıldığı veri yapısıdır. **Son giren ilk çıkar** (LIFO: Last In, First Out).

Üst üste konmuş tabakları düşünün: en son konan tabak en üsttedir ve ilk o alınır. Altteki bir tabağa ulaşmak için önce üsttekilerin alınması gerekir.

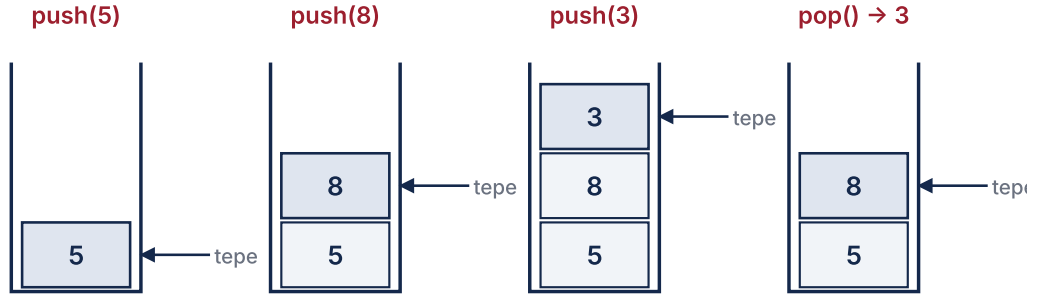


En son konan tabak ilk alınır: son giren ilk çıkar (LIFO)

push Yığının tepesine eleman ekler.

pop Tepedeki elemanı çıkarır ve döndürür.

top (peek) Tepedeki elemanı çıkarmadan gösterir.



Boş bir yığına 5, 8 ve 3 eklenir; pop() işlemi en son giren 3'ü çıkarır

Kullanım alanları: metin düzenleyicilerdeki **geri al** (Ctrl+Z) komutu, web tarayıcısındaki **geri** düğmesi, parantezlerin doğru açılıp kapandığının denetlenmesi.

8.4 Kuyruk (Queue)

TANIM

Kuyruk, elemanların bir uçtan (arkadan) eklenip diğer uçtan (önden) çıkarıldığı veri yapısıdır. **İlk giren ilk çıkar** (FIFO: First In, First Out).

Banka ya da market kasasındaki sırayı düşünün: sıraya ilk giren ilk hizmet alır, yeni gelen sıranın sonuna geçer.



Sıraya ilk giren ilk hizmet alır: ilk giren ilk çıkar (FIFO)



Kuyruğa A, B, C, D sırasıyla eklenmiştir; ilk çıkacak olan A'dır

Kullanım alanları: yazıcıya gönderilen belgelerin sırayla basılması, çağrı merkezinde bekleyen aramalar, işletim sisteminin işlemciyi programlara sırayla vermesi.

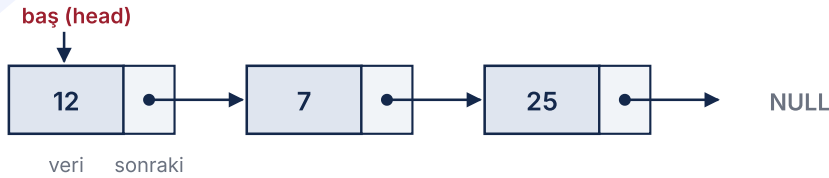
FARK

Yığın ile kuyruk arasındaki tek fark elemanların çıkış sırasıdır. A, B, C sırasıyla eklenen elemanlar yığından C, B, A; kuyruktan A, B, C sırasıyla çıkar.

8.5 Bağlı Liste (Linked List)

TANIM

Bağlı liste, her elemanı (düğümü) hem veriyi hem de bir sonraki düğümün adresini tutan veri yapısıdır. Son düğüm hiçbir yeri göstermez (NULL).



Üç düğümlü bir bağlı liste

Dizide elemanlar bellekte yan yana durur; bağlı listede ise düğümler bellekte dağınık olabilir ve birbirine adreslerle bağlanır. Bir trenin vagonlarına benzer: her vagon bir sonrakine bağlıdır ve araya yeni bir vagon eklemek için yalnızca iki bağlantının değiştirilmesi yeterlidir.



Her vagon yalnızca bir sonrakine bağlıdır; bağlı listedeki düğümler de böyledir

	Dizi	Bağlı liste
Elemana erişim	İndisle doğrudan ve hızlı	Baştan başlayarak tek tek ilerlenir
Araya ekleme / silme	Zor; elemanlar kaydırılır	Kolay; yalnızca bağlantılar değişir
Boyut	Genellikle sabit	Gerektikçe büyüyüp küçülür

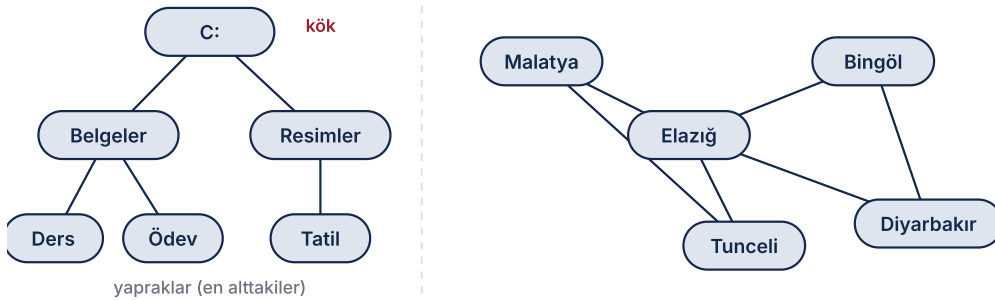
8.6 Ağaç ve Graf

Ağaç (Tree)

Elemanların hiyerarşik (üst-alt) ilişkisiyle bağlandığı yapıdır. En üstteki düğüme **kök**, alt düğümü olmayanlara **yaprak** denir. Bilgisayardaki klasör yapısı, bir şirketin organizasyon şeması ve soy ağacı ağaç yapısına örnektir.

Graf (Graph)

Düğümün (köşelerin) birbirine **kenarlarla** serbestçe bağlandığı yapıdır; ağaçtan farklı olarak bir düğüme birden çok yoldan ulaşılabilir. Şehirler ve aralarındaki yollar, sosyal ağlardaki arkadaşlıklar ve bilgisayar ağları graf ile gösterilir.



Solda klasör yapısı (ağaç), sağda şehirler ve aralarındaki yollar (graf)



Ağaç: kökten alt klasörlere dallanan hiyerarşi



Graf: şehirler (düğümler) ve yollar (kenarlar)

Özet

Veri yapısı	Türü	Temel kural	Günlük hayattan örnek
Dizi	Doğrusal	Elemanlara indisle erişilir	Numaralı dolaplar
Yığın	Doğrusal	Son giren ilk çıkar (LIFO)	Üst üste tabaklar
Kuyruk	Doğrusal	İlk giren ilk çıkar (FIFO)	Banka sırası
Bağlı liste	Doğrusal	Her düğüm bir sonrakini gösterir	Tren vagonları
Ağaç	Doğrusal olmayan	Kökten dallara hiyerarşi	Klasör yapısı
Graf	Doğrusal olmayan	Düğüm kenarlarla serbestçe bağlı	Şehirler ve yollar

8.7 Algoritma Nedir?

TANIM

Algoritma, bir problemi çözmek ya da bir işi yapmak için izlenecek, sırası belli ve sonlu sayıdaki adımlar dizisidir.

Algoritma kavramı yalnızca bilgisayara özgü değildir; bir yemek tarifi ya da bir cihazın kurulum talimatı da birer algoritmadır. Örneğin çay demleme algoritması:

- ◆ 1. Çaydanlığın altına su koy ve ocağı yak.
- ◆ 2. Demliğe çay koy.
- ◆ 3. Su kaynayınca bir kısmını demliğe dök.
- ◆ 4. Demliği çaydanlığın üstüne koy, 10–15 dakika demlenmesini bekle.
- ◆ 5. Bardağa çay ve sıcak su koyarak servis et.



Çay demlemek de sırası belli, sonlu adımlardan oluşan bir algoritmadır

İyi Bir Algoritmanın Özellikleri

Giriş	Dışarıdan sıfır ya da daha fazla veri alır.
Çıkış	En az bir sonuç üretir.
Açıklık (belirlilik)	Her adım net ve tek anlamlı olmalıdır; “biraz bekle” gibi belirsiz ifadeler bulunmamalıdır.
Sonluluk	Belirli sayıda adımdan sonra mutlaka sona ermelidir; sonsuza kadar çalışmamalıdır.
Etkinlik	Her adım yapılabilir olmalı ve gereksiz adım içermemelidir.

Algoritmanın Gösterim Biçimleri

Sözel anlatım	Adımlar günlük dille, numaralandırılarak yazılır.
Sözde kod (pseudo code)	Programlama diline benzeyen ama belirli bir dile bağlı olmayan kısa ifadelerle yazılır.
Akış şeması	Adımlar standart şekillerle çizilir ve oklarla birbirine bağlanır.

8.8 Akış Şeması Sembolleri

Akış şemaları algoritmayı görsel hâle getirir; her şeklin belirli bir anlamı vardır. Akış yukarıdan aşağıya doğru okunur.

Sembol	Adı	Ne için kullanılır?
	Başla / Bitir	Algoritmanın başladığı ve bittiği yer
	Giriş / Çıkış	Veri okuma (girme) ya da sonucu yazdırma
	İşlem	Hesaplama, atama gibi işlemler
	Karar	Koşul sınanır; Evet / Hayır yollarına ayrılır
	Akış yönü	Adımların hangi sırayla yapılacağını gösterir

Akış şemalarında üç temel yapı kullanılır: adımların sırayla yapıldığı **sıralı yapı**, bir koşula göre farklı yollardan birinin seçildiği **karar (seçim) yapısı** ve bazı adımların bir koşul sağlandığı sürece tekrarlandığı **döngü (tekrar) yapısı**.

8.9 Temel Algoritma Örnekleri

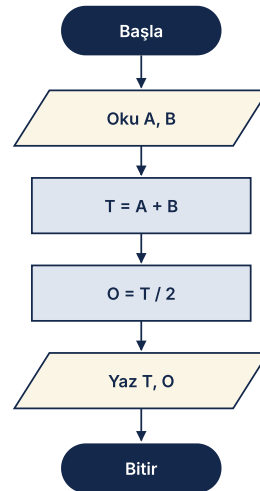
ÖRNEK 1

Klavyeden girilen iki sayının toplamını ve ortalamasını bulan algoritmayı yazınız; akış şemasını çiziniz. (Sıralı yapı)

ÇÖZÜM

1. Başla
2. A ve B sayılarını oku
3. $T = A + B$
4. $O = T / 2$
5. T ve O'yu yaz
6. Bitir

Deneme: A = 8, B = 12 girilirse T = 20, O = 10 yazılır.

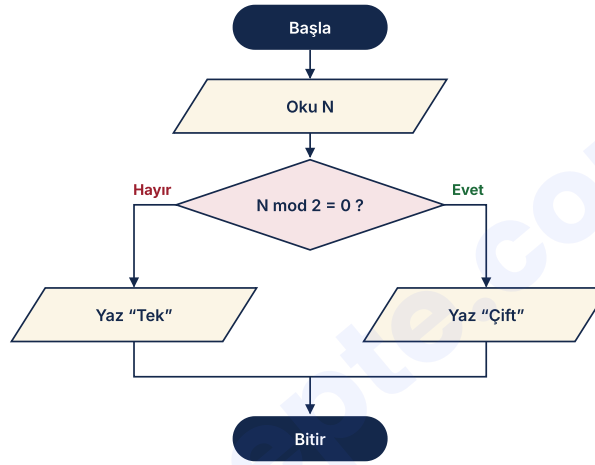


ÖRNEK 2

Girilen bir tam sayının tek mi çift mi olduğunu bulan algoritmanın akış şemasını çiziniz. (Karar yapısı)

ÇÖZÜM

Bir sayı 2'ye bölündüğünde kalan 0 ise çifttir. Kalan, **mod** işlemiyle bulunur: $7 \bmod 2 = 1$, $10 \bmod 2 = 0$.



Tek-çift bulma algoritması

ÖRNEK 3

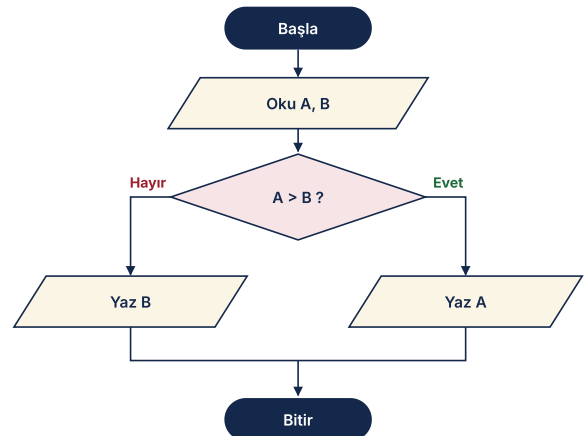
Girilen iki sayıdan büyük olanı yazdıran algoritmayı sözde kodla yazınız ve akış şemasını çiziniz.

ÇÖZÜM

```

BAŞLA
OKU A, B
EĞER A > B İSE
    YAZ A
DEĞİLSE
    YAZ B
BİTİR
  
```

Sayılar eşitse koşul sağlanmaz ve B yazılır; iki sayı aynı olduğu için sonuç yine doğrudur.

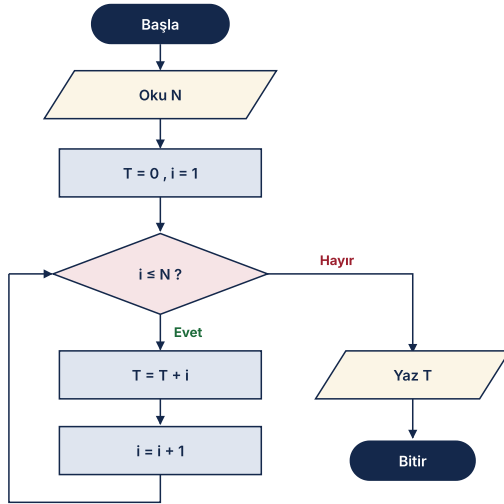


ÖRNEK 4

1'den N'e kadar olan sayıların toplamını bulan algoritmanın akış şemasını çiziniz. N = 5 için adım adım izleyiniz. (Döngü yapısı)

ÇÖZÜM

Toplamı tutan **T** başlangıçta 0, sayacı tutan **i** başlangıçta 1 yapılır. i, N'i geçene kadar T'ye eklenir ve her turda 1 artırılır.



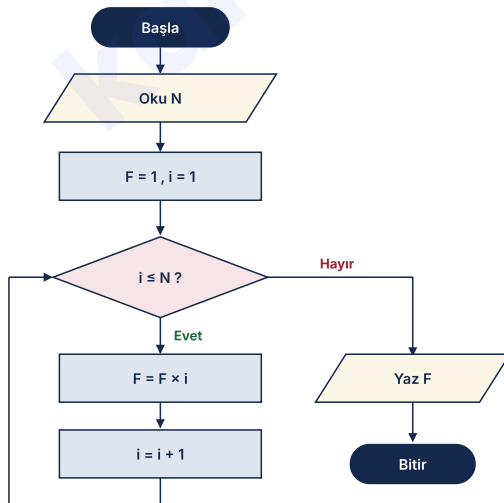
Tur	i	i ≤ 5?	T = T + i
1	1	Evet	0 + 1 = 1
2	2	Evet	1 + 2 = 3
3	3	Evet	3 + 3 = 6
4	4	Evet	6 + 4 = 10
5	5	Evet	10 + 5 = 15
6	Hayır		15 yazılır

ÖRNEK 5

Girilen N sayısının faktöriyelini ($N! = 1 \times 2 \times \dots \times N$) bulan algoritmanın akış şemasını çiziniz. N = 4 için izleyiniz.

ÇÖZÜM

Çarpım bulunacağı için başlangıç değeri 0 değil **1** olmalıdır; aksi hâlde sonuç hep 0 çıkar.



Tur	i	F = F × i
1	1	1 × 1 = 1
2	2	1 × 2 = 2
3	3	2 × 3 = 6
4	4	6 × 4 = 24
5		24 yazılır

8.10 Dizilerle Algoritmalar: Arama ve Sıralama

Veri yapıları ile algoritmalar birlikte kullanılır: veri yapısı veriyi düzenli tutar, algoritma bu veri üzerinde işlem yapar. En sık yapılan işlemler bir dizide en büyük elemanı bulmak, bir elemanı aramak ve diziyi sıralamaktır.

Dizideki En Büyük Elemanı Bulma

İlk eleman geçici olarak en büyük kabul edilir; sonra diğer elemanlar tek tek bununla karşılaştırılır ve daha büyüğü bulunursa en büyük güncellenir.

```
EnB = A[0]
DÖNGÜ i = 1'den n - 1'e kadar
    EĞER A[i] > EnB İSE EnB = A[i]
YAZ EnB
```

i	A[i]	A[i] > EnB ?	EnB
—	—	—	12 (başlangıç)
1	7	Hayır	12
2	25	Evet	25
3	3	Hayır	25
4	18	Hayır	25

A = [12, 7, 25, 3, 18] dizisi için sonuç **25**'tir.

Doğrusal (Sıralı) Arama

Aranan değer dizinin başından sonuna kadar elemanlarla tek tek karşılaştırılır.

Bulunursa indisi verilir; dizinin sonuna gelinip bulunamazsa “bulunamadı” denir. Dizinin sıralı olması gerekmez.

Aranan	Karşılaştırılan elemanlar	Sonuç
3	12 → 7 → 25 → 3	4. karşılaştırmada bulundu, indis 3
9	12 → 7 → 25 → 3 → 18	5 karşılaştırma yapıldı, bulunamadı

İkili (Binary) Arama

Yalnızca **sıralı** dizilerde kullanılır. Aranılan değer dizinin ortasındaki elemanla karşılaştırılır: eşitse bulunmuştur; küçükse sol yarıda, büyükse sağ yarıda aramaya devam edilir. Her adımda aranacak alan yarıya indiği için doğrusal aramadan çok daha hızlıdır. Sözlükte kelime ararken de benzer biçimde ortadan açıp ileri ya da geri gideriz.



Sözlükte kelime aramak ikili aramaya benzer: ortadan açılır, aranılan kelimeye göre ileri ya da geri gidilir

Sıralı dizi: [3, 8, 12, 17, 25, 31, 40]. Aranılan değer: 25.

Adım	Aralık (indis)	Orta indis	Ortadaki eleman	Karar
1	0 – 6	3	17	$25 > 17 \rightarrow$ sağ yarıya geç
2	4 – 6	5	31	$25 < 31 \rightarrow$ sol yarıya geç
3	4 – 4	4	25	Eşit $\rightarrow$ bulundu

İkili arama 25'i **3 adımda** bulmuştur; aynı değer doğrusal aramayla 5 karşılaştırmada bulunurdu. Dizi büyüdükçe fark çok artar: 1000 elemanlı sıralı bir dizide ikili arama en fazla 10 adımda sonuca ulaşır.

Kabarcık (Bubble) Sıralama

Dizi baştan sona taranır ve **yan yana iki eleman** karşılaştırılır; sıraları yanlışsa yer değiştirilir. Her geçişin sonunda en büyük eleman, suyun yüzeyine çıkan bir kabarcık gibi dizinin sonuna yerleşir. Hiç yer değiştirme olmayana kadar geçişler tekrarlanır.



Sıralama: karışık elemanlar küçükten büyüğe dizilir

$A = [5, 3, 8, 1]$ dizisini küçükten büyüğe sıralayalım:

Geçiş	Karşılaştırmalar	Geçiş sonunda dizi
1	5-3 değiş $\rightarrow$ 3, 5, 8, 1 · 5-8 kalır · 8-1 değiş	[3, 5, 1, 8]
2	3-5 kalır · 5-1 değiş	[3, 1, 5, 8]
3	3-1 değiş	[1, 3, 5, 8]

8.11 Veri Yapısı ve Algoritma Bir Arada

ÖRNEK 6

Yığın kullanarak “KİTAP” kelimesini tersine çeviren algoritmayı açıklayınız.

ÇÖZÜM

Kelimenin harfleri sırayla yığına eklenir (push), sonra yığın boşalana kadar çıkarılır (pop). Yığın LIFO çalıştığı için harfler ters sırayla çıkar.

İşlem	Yığın (alt $\rightarrow$ tepe)	Çıktı
push K, İ, T, A, P	K İ T A P	—
pop $\times$ 5	boş	P A T İ K

Sonuç: **PATİK**.

ÖRNEK 7

Bir ifadedeki parantezlerin dengeli olup olmadığı yığınla nasıl denetlenir? $(a + b) \times (c - (d + e))$ ifadesini inceleyiniz.

ÇÖZÜM

- ◆ Her "(" görüldüğünde yığına eklenir (push).
- ◆ Her ")" görüldüğünde yığından bir eleman çıkarılır (pop). Yığın boşken ")" gelirse ifade **hatalıdır**.
- ◆ İfadenin sonunda yığın boşsa parantezler **dengeli**dir.

İfadede üç "(" ve üç ")" vardır; her kapanış bir açılışla eşleşir ve sonunda yığın boş kalır: ifade **dengeli**dir. $(a + b))$ ifadesinde ise ikinci ")" geldiğinde yığın boştur: ifade **hatalıdır**.

ÖRNEK 8

Bir yazıcıya sırasıyla A (3 sayfa), B (1 sayfa) ve C (2 sayfa) belgeleri gönderiliyor. Belgeler hangi sırayla basılır? Yığın kullanılsaydı sıra ne olurdu?

ÇÖZÜM

Yazıcılar **kuyruk** kullanır: ilk gönderilen ilk basılır. Basım sırası **A, B, C**'dir.

Yığın kullanılsaydı en son gönderilen ilk basılırdı: **C, B, A**. Bu durumda ilk gönderen kişi en uzun süre beklerdi; bu yüzden yazıcı için doğru veri yapısı kuyruktur.