Bölüm 4: Yığın

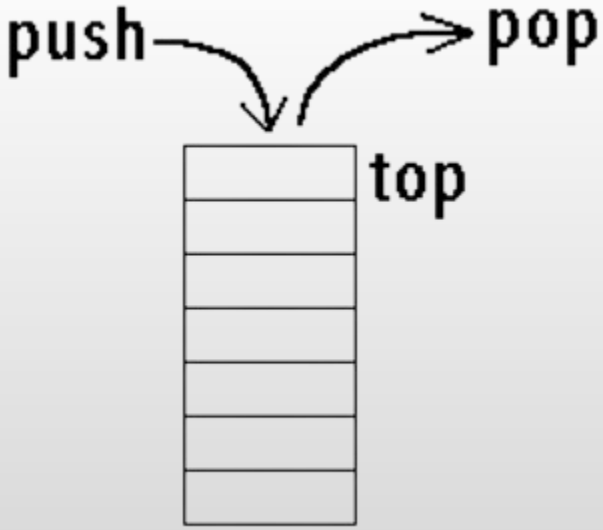
Veri Yapıları

Yığın (Stack)

- Yeni bir öğenin eklenmesi veya mevcut öğenin çıkarılması yığının en üstünden (top) gerçekleşir.
- Üst üste konulmuş kutular veya tabaklar gibi düşünülebilir.
- En alttaki öğeye ulaşmak için yığının üzerindeki öğeler tek tek çıkarılır.
- LIFO (Son Giren İlk Çıkar) veya FILO (İlk Giren Son Çıkar) mantığıyla çalışır.

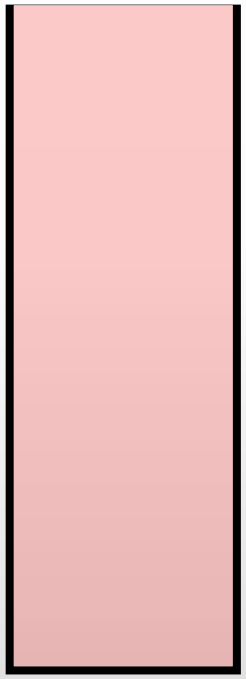
Temel İşlemler

- push(): Yığının üstüne yeni bir öğe ekler.
- pop(): Yığının üstündeki öğeyi çıkarır.
- top(): Yığının üstündeki öğeyi döndürür.
- isEmpty(): Yığının boş olup olmadığını kontrol eder.
- size(): Yığının boyutunu döndürür.



Gösterim

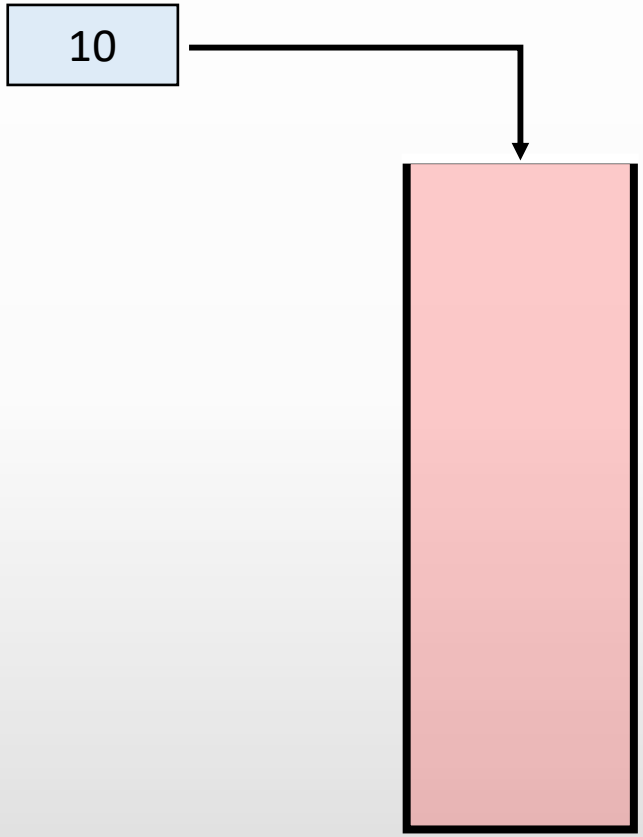
top ⟶ null



Yığıt

Ekleme İşlemi:

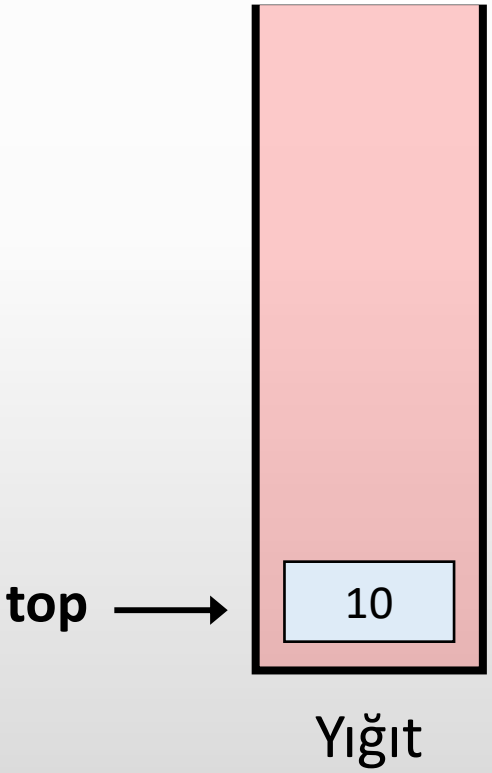
push(10)



top ⟶ null

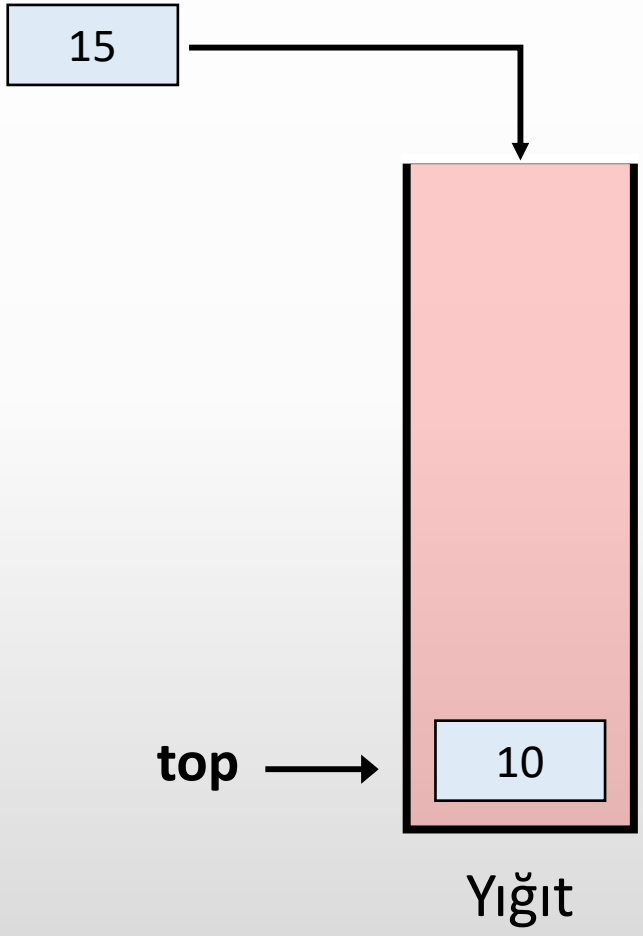
Ekleme İşlemi:

push(10)



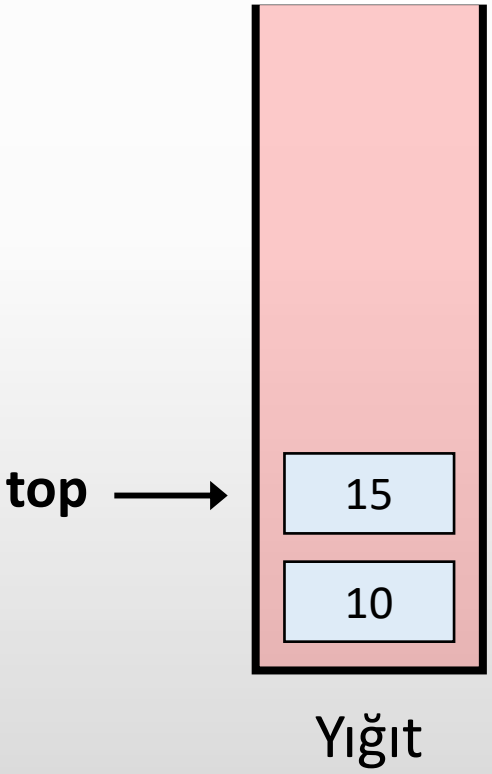
Ekleme İşlemi:

push(15)



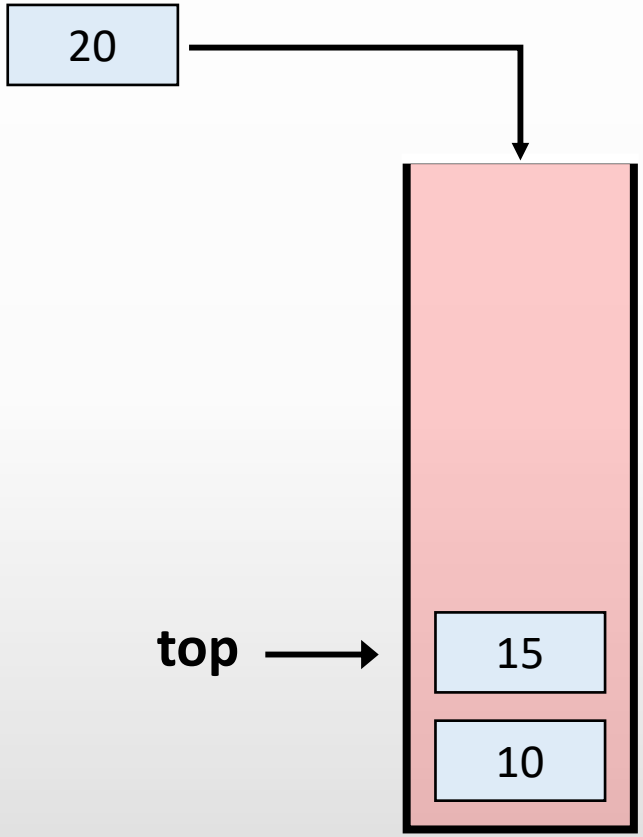
Ekleme İşlemi:

push(15)



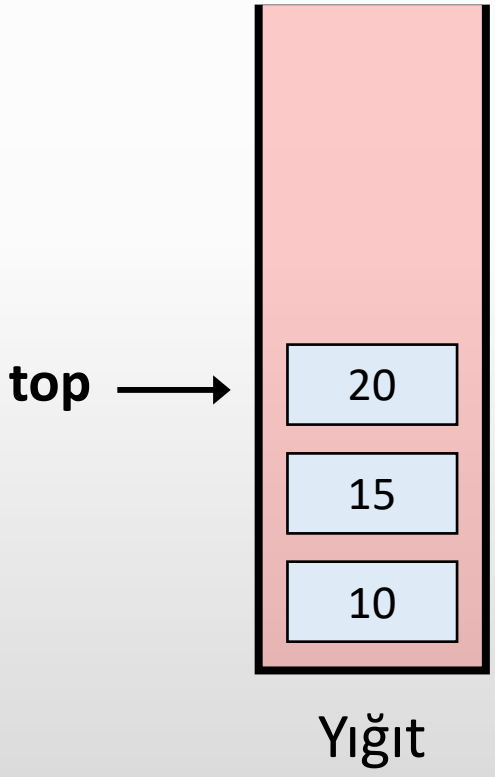
Ekleme İşlemi:

push(20)



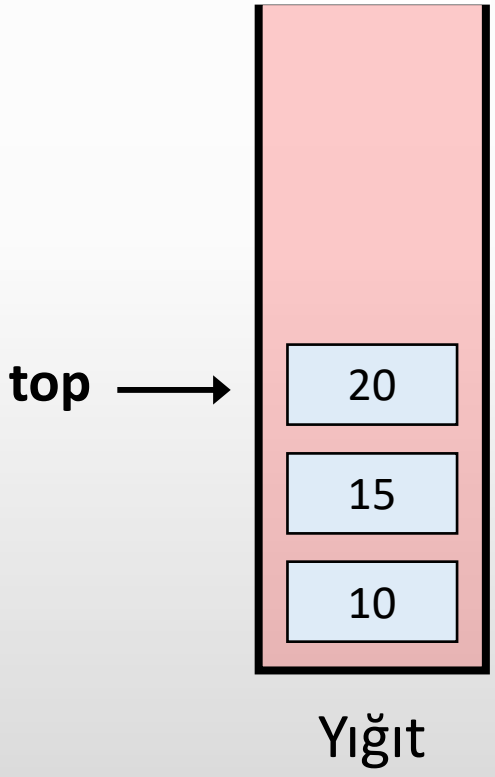
Ekleme İşlemi:

push(20)



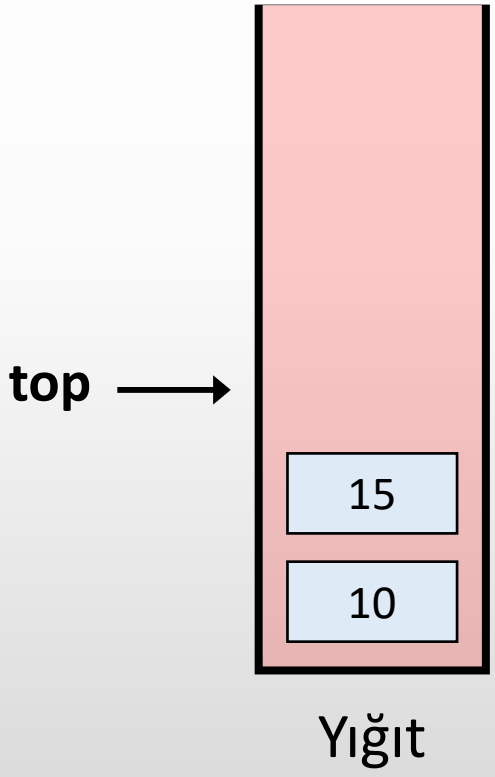
Silme İşlemi:

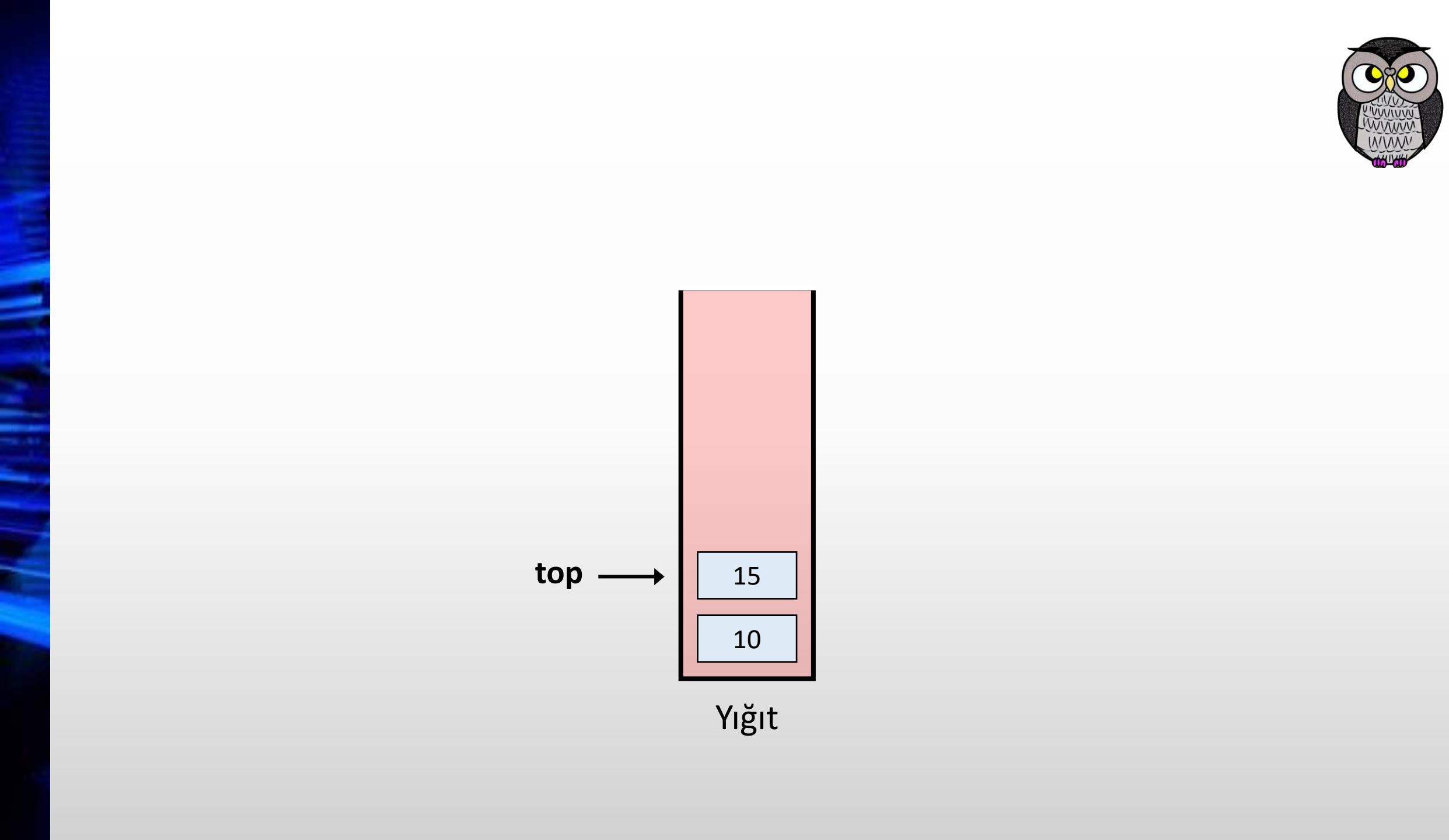
pop()



Silme İşlemi:

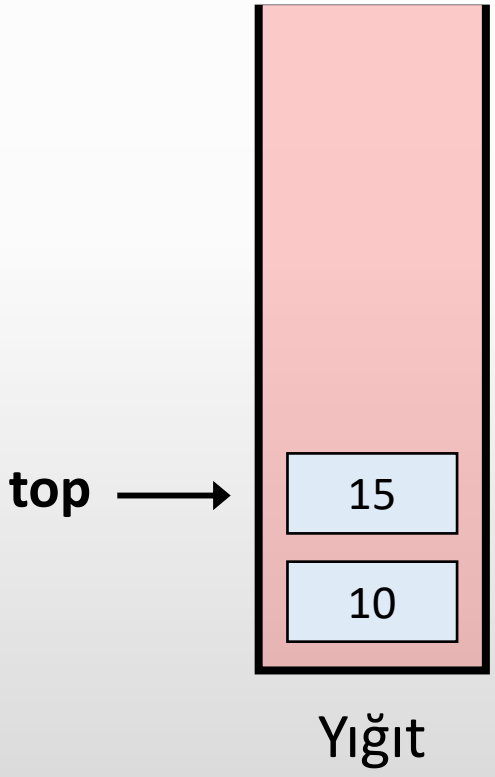
pop()




top
15
10
Yığıt

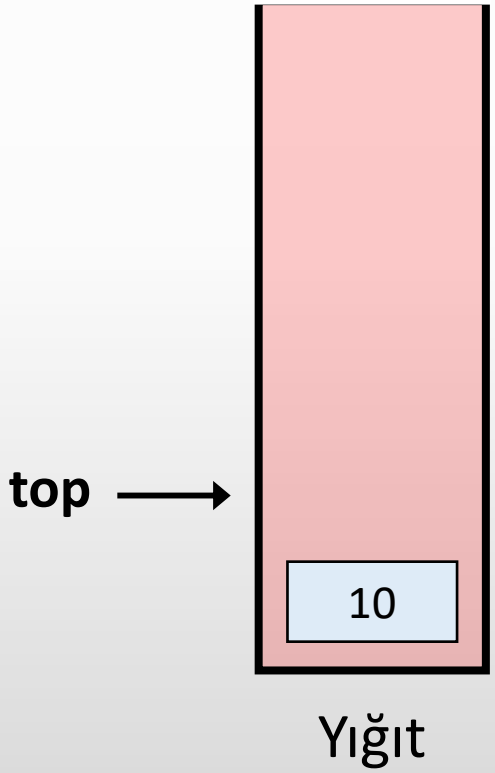
Silme İşlemi:

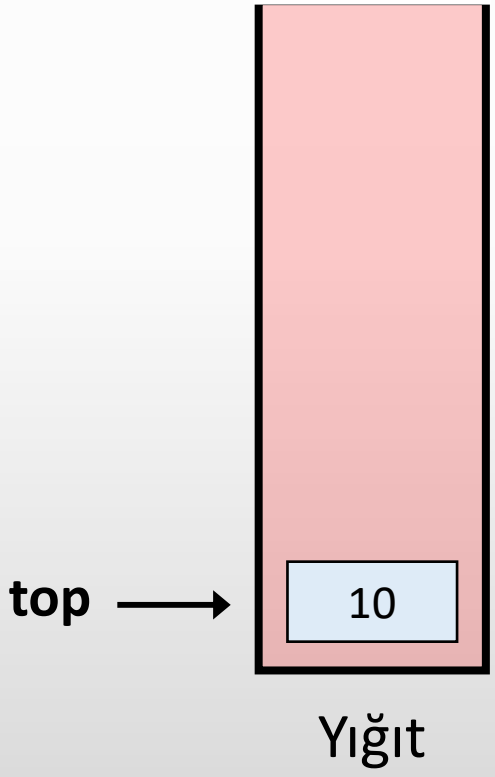
pop()



Silme İşlemi:

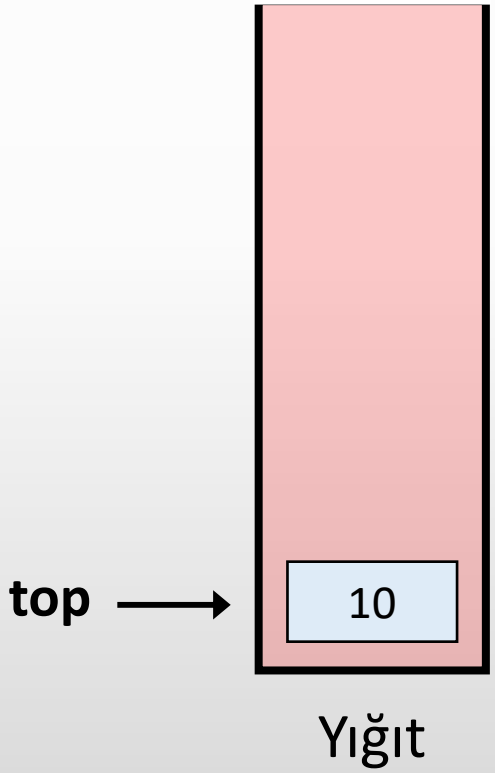
pop()




top
10
Yığıt

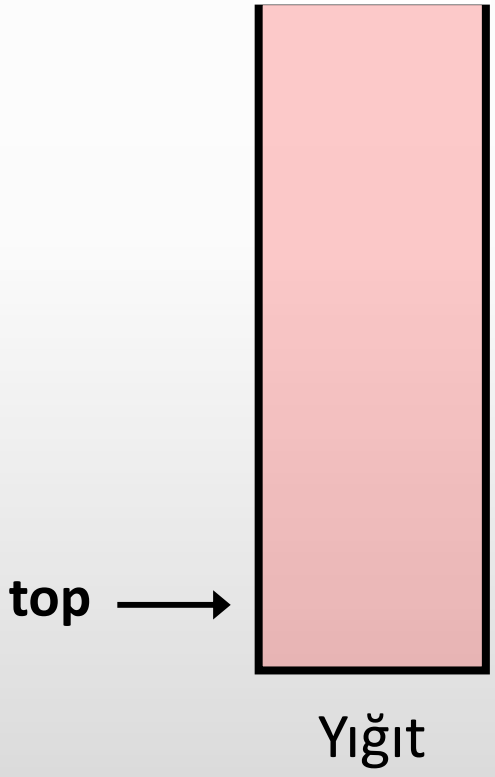
Silme İşlemi:

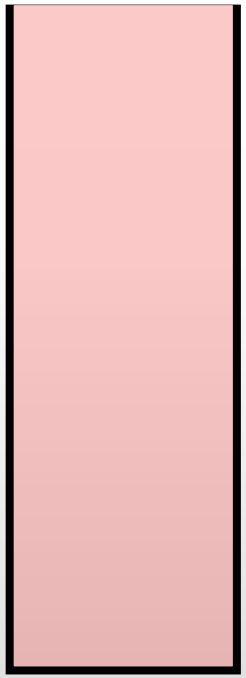
pop()



Silme İşlemi:

pop()





Yığıt

top ⟶ **null**

Push İşlemi

- Yığına bir öğe ekler.
- Yığın dolu ise, «Üst Taşma» (Overflow) hatası verir.

```
başla
  eğer yığın doluysa
    hata ver
  değilse
    top değerini artır
    yığın[top]'a değer ata
son
```

Pop İşlemi

- Yığının üstündeki öğeyi çıkarır.
- Yığın boş ise, «Alt Taşma» (Underflow) hatası verir.

```
başla
  eğer yığın boşsa
    hata ver
  değilse
    yığın[top] değerini sakla
    top değerini azalt
    saklanan değeri döndür
son
```

Top İşlemi

- Yığının üstündeki öğeyi döndür.

```
başla
  yığın[top] değerini sakla
  saklanan değeri döndür
son
```

isEmpty İşlemi

- Yığının boş olup olmadığını kontrol eder.
- Yığın boş ise true dön, değilse false dön.

```
başla
  eğer top değeri < 1 ise
    true döndür
  değilse
    false döndür
son
```

Zaman Karmaşıklığı

- Yığın üzerindeki işlemler, sabit zaman karmaşıklığına (O(1)) sahiptir.
- İşlemlerin süresi yığının boyutundan bağımsızdır.
- Yığın kullanımının veri saklama açısından ekstra alan maliyeti yoktur.

Yığın Türleri

- Sabit Boyutlu (Fixed Size Stack)
- Dinamik Boyutlu (Dynamic Size Stack)

Yığın Türleri

- **Sabit Boyutlu Yığın (Fixed Size Stack)**
  - Sabit boyuta sahiptir, dinamik olarak büyüyemez veya küçülemez.
  - Dolu yığına öğe eklenmeye çalışılırsa, taşma hatası meydana gelir.
  - Boş yığından öğe çıkarma istenirse, alt taşma hatası meydana gelir.
  - Arka planda dizi yapısını kullanır.
- Dinamik Boyutlu Yığın (Dynamic Size Stack)

Yığın Türleri

- Sabit Boyutlu Yığın (Fixed Size Stack)
- **Dinamik Boyutlu Yığın (Dynamic Size Stack)**
  - Dinamik olarak büyüyebilir veya küçülebilir.
  - Dolu yığına yeni öğe ekleneceğinde otomatik olarak boyutunu artırır.
  - Eleman sayısı boyutuna oranla az ise boyutunu azaltır.
  - Arka planda bağlı liste yapısını kullanır.

Uygulama Örnekleri

- Matematiksel ifadelerin (infix, prefix, postfix) dönüşümünde kullanılır.
- Metin düzenleme uygulamalarında geri al, yeniden uygula işlemlerinde.
- Web tarayıcılarının gezinme geçmişini yönetmek için kullanılır.
- Hanoi Kuleleri, ağaç üzerinde gezinme, hisse senedi sıçrama problemi, histogram problemleri gibi algoritmalarda kullanılır.
- Şövalye Turu, N-Vezir problemleri, labirent'de yol bulma gibi oyunlarda geri izleme (backtracking) için kullanılır.

Uygulama Örnekleri

- Topolojik Sıralama, Güçlü Bağlantılı Bileşenlerin bulunması gibi çizge algoritmalarında kullanılır.
- Bir programın çalışma sırasında bellek ve fonksiyon çağrıları yönetiminde kullanılır.
- Bir dizenin ters çevrilmesi gibi işlemlerde kullanılır. Dize karakterleri yığına birer birer eklenir ve ters sırada alınır.

Yığın Gerçekleştirimi (implementation)

- Dizi (array) veya bağlı liste (linked list) kullanılarak gerçekleştirilebilir.
- **Dizi Tabanlı:**
  - push(), üst öğenin indeksini artırarak, yeni öğeyi bu indekse atar.
  - pop(), üst öğenin indeksini azaltarak, indeksteki değeri döndürür.
- **Bağlı Liste Tabanlı:**
  - push(), yeni bir düğüm oluşturulur, listenin başına eklenir.
  - pop(), liste başındaki düğüm listeden çıkarılır, ve değeri döndürülür.

Dizi Tabanlı Yığın Gerçekleştirme

- Bu yaklaşıma "sınırlı yığın" denir.
- Yığının kapasitesi oluşturulma anında belirlenir ve değiştirilemez.
- Basit bir yöntem ve kolayca uygulanabilir.
- İşaretçiler kullanılmadığı için bellek tasarrufu sağlar.
- Ekleme yapmadan önce yığının dolu olmadığından emin olunmalıdır.
- Çıkarma işleminde, kullanılmayan yeri temizlemek için null atanmalıdır.
- Belleği doldurma amaçlı gelebilecek saldırılara karşı güvenlidir.
- Bellek kullanımı sıkı bir şekilde kontrol edilir.

Bağlı Liste Tabanlı Yığın Gerçekleştirme

- Elemanların bir zincir şeklinde birbirine bağlı olması, esnek boyut yönetimini mümkün kılar.
- Çalışma zamanında ihtiyaca göre boyutu değişir.
- İşaretçilerin kullanılması nedeniyle ekstra bellek alanı kullanır.
- Rastgele erişim mümkün değilken, üstteki elemana erişim hızlı ve basittir.
- Ekleme ve çıkarma işlemleri etkin bir şekilde gerçekleştirilir.

Uygulama

```
// Örnek değişkenleri
Dugum top;
int uzunluk;

public void push(int veri) {
  Dugum gecici = new Dugum(veri);
  gecici.sonraki = top;
  top = gecici;
  uzunluk++;
}
```

Uygulama - push

top ⟶ null

```
// Örnek değişkenleri
Dugum top;
int uzunluk;

public void push(int veri) {
  Dugum gecici = new Dugum(veri);
  gecici.sonraki = top;
  top = gecici;
  uzunluk++;
}
```

top ⟶ null

uzunluk = 0

```
// Örnek değişkenleri
Dugum top;
int uzunluk;

public void push(int veri) {
  Dugum gecici = new Dugum(veri);
  gecici.sonraki = top;
  top = gecici;
  uzunluk++;
}
```

top ⟶ null

uzunluk = 0

push(10)

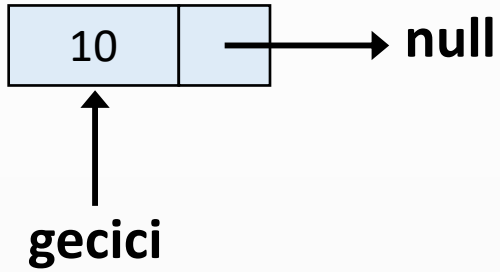
```
// Örnek değişkenleri
Dugum top;
int uzunluk;

public void push(int veri) {
  Dugum gecici = new Dugum(veri);
  gecici.sonraki = top;
  top = gecici;
  uzunluk++;
}
```

top ⟶ null

uzunluk = 0

push(10)

```
// Örnek değişkenleri
Dugum top;
int uzunluk;

public void push(int veri) {
  Dugum gecici = new Dugum(veri);
  gecici.sonraki = top;
  top = gecici;
  uzunluk++;
}
```

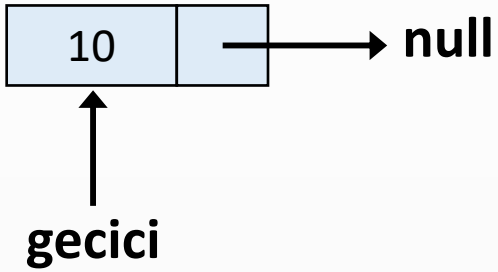


top ⟶ null

uzunluk = 0

push(10)

```
// Örnek değişkenleri
Dugum top;
int uzunluk;

public void push(int veri) {
  Dugum gecici = new Dugum(veri);
  gecici.sonraki = top;
  top = gecici;
  uzunluk++;
}
```

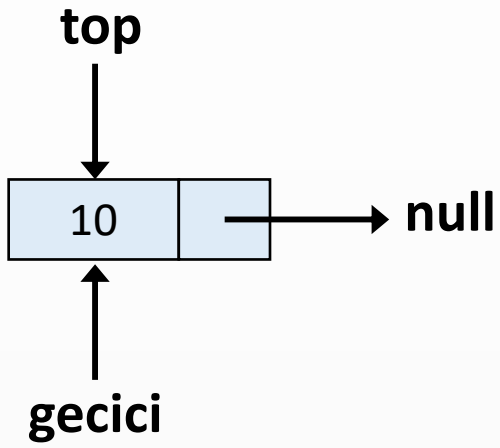


top ⟶ null

uzunluk = 0

push(10)

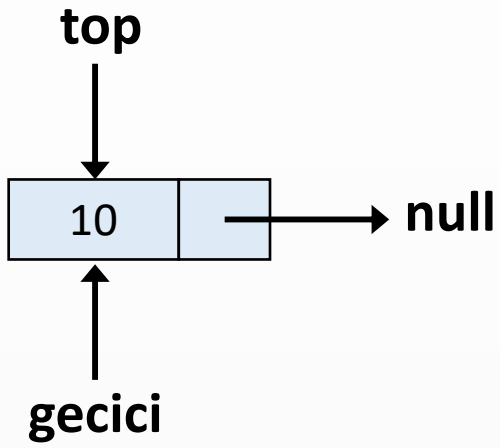
```
// Örnek değişkenleri
Dugum top;
int uzunluk;

public void push(int veri) {
  Dugum gecici = new Dugum(veri);
  gecici.sonraki = top;
  top = gecici;
  uzunluk++;
}
```



uzunluk = 0

push(10)

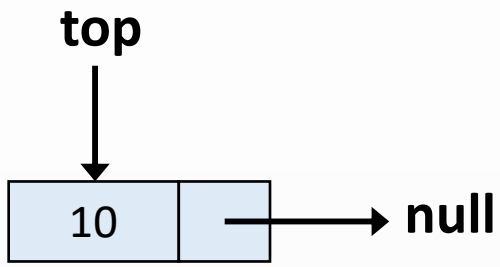
```
// Örnek değişkenleri
Dugum top;
int uzunluk;

public void push(int veri) {
  Dugum gecici = new Dugum(veri);
  gecici.sonraki = top;
  top = gecici;
  uzunluk++;
}
```



uzunluk = 1

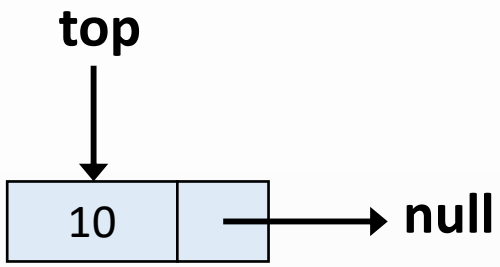
push(10)

```
// Örnek değişkenleri
Dugum top;
int uzunluk;

public void push(int veri) {
  Dugum gecici = new Dugum(veri);
  gecici.sonraki = top;
  top = gecici;
  uzunluk++;
}
```



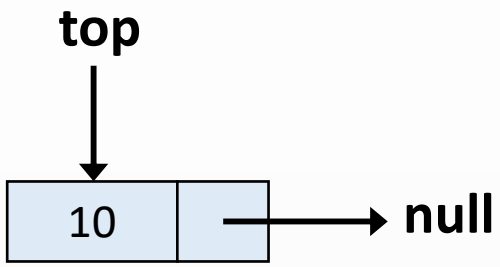
uzunluk = 1

```
// Örnek değişkenleri
Dugum top;
int uzunluk;

public void push(int veri) {
  Dugum gecici = new Dugum(veri);
  gecici.sonraki = top;
  top = gecici;
  uzunluk++;
}
```



uzunluk = 1

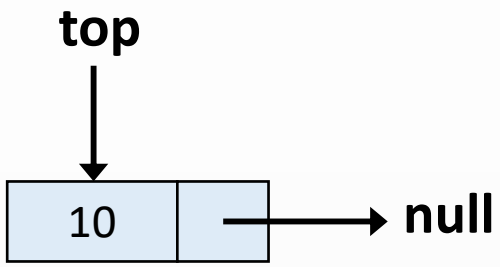
```
// Örnek değişkenleri
Dugum top;
int uzunluk;

public void push(int veri) {
  Dugum gecici = new Dugum(veri);
  gecici.sonraki = top;
  top = gecici;
  uzunluk++;
}
```



uzunluk = 1

push(15)

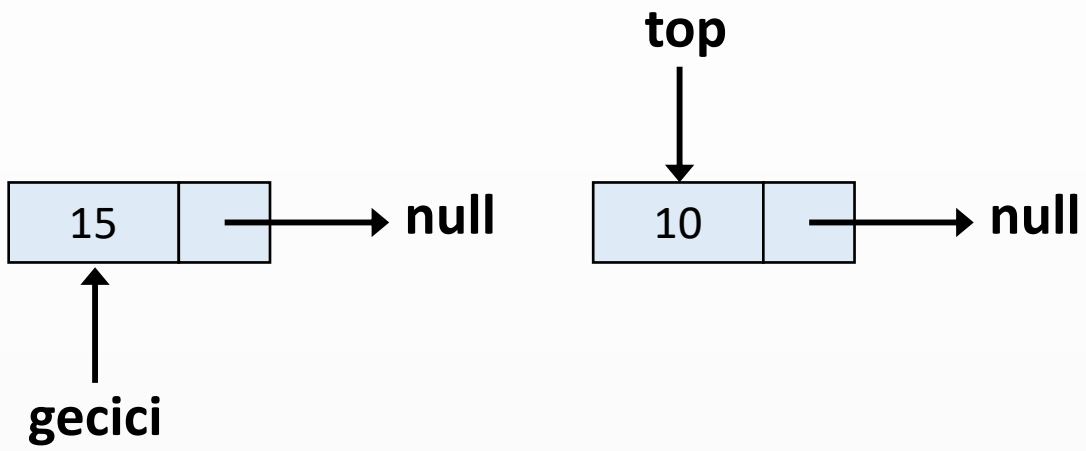
```
// Örnek değişkenleri
Dugum top;
int uzunluk;

public void push(int veri) {
  Dugum gecici = new Dugum(veri);
  gecici.sonraki = top;
  top = gecici;
  uzunluk++;
}
```



uzunluk = 1

push(15)

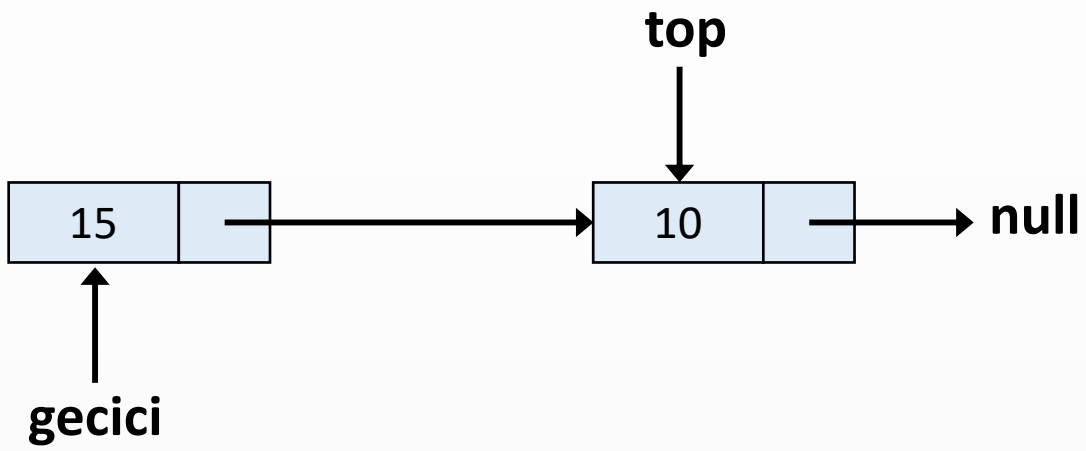
```
// Örnek değişkenleri
Dugum top;
int uzunluk;

public void push(int veri) {
  Dugum gecici = new Dugum(veri);
  gecici.sonraki = top;
  top = gecici;
  uzunluk++;
}
```



uzunluk = 1

push(15)

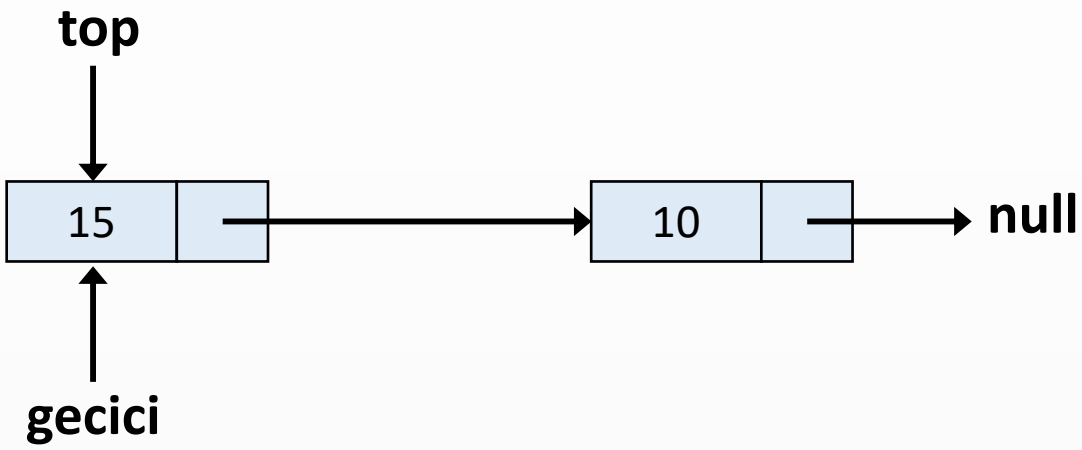
```
// Örnek değişkenleri
Dugum top;
int uzunluk;

public void push(int veri) {
  Dugum gecici = new Dugum(veri);
  gecici.sonraki = top;
  top = gecici;
  uzunluk++;
}
```



uzunluk = 1

push(15)

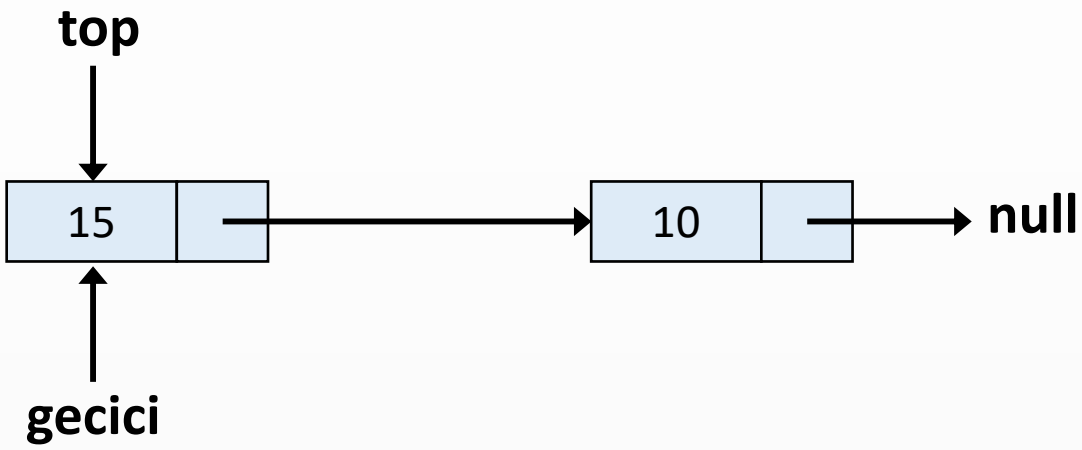
```
// Örnek değişkenleri
Dugum top;
int uzunluk;

public void push(int veri) {
  Dugum gecici = new Dugum(veri);
  gecici.sonraki = top;
  top = gecici;
  uzunluk++;
}
```



uzunluk = 1

push(15)

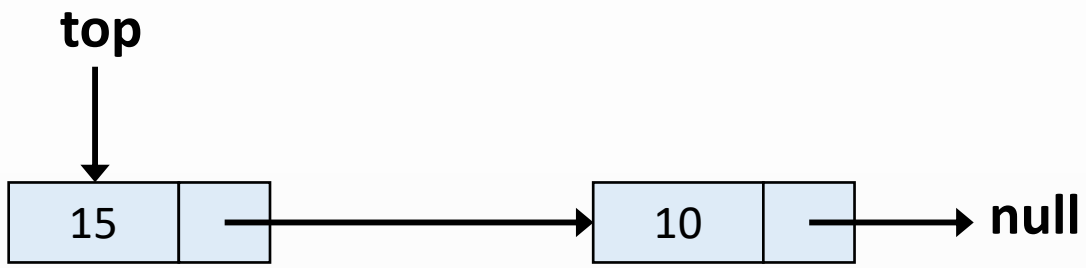
```
// Örnek değişkenleri
Dugum top;
int uzunluk;

public void push(int veri) {
  Dugum gecici = new Dugum(veri);
  gecici.sonraki = top;
  top = gecici;
  uzunluk++;
}
```



uzunluk = 2

push(15)

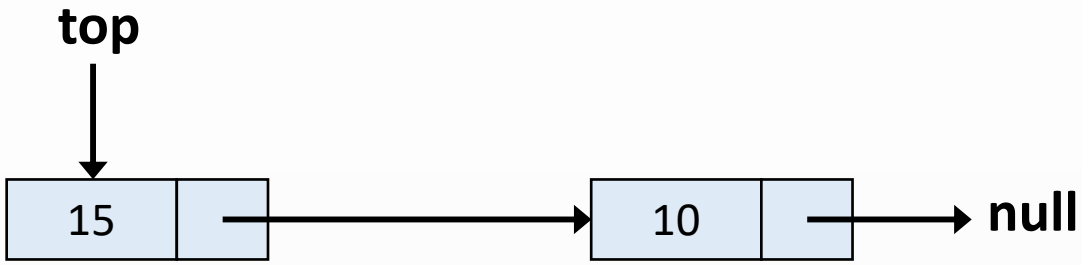
```
// Örnek değişkenleri
Dugum top;
int uzunluk;

public void push(int veri) {
  Dugum gecici = new Dugum(veri);
  gecici.sonraki = top;
  top = gecici;
  uzunluk++;
}
```



uzunluk = 2

```
// Örnek değişkenleri
Dugum top;
int uzunluk;

public void push(int veri) {
  Dugum gecici = new Dugum(veri);
  gecici.sonraki = top;
  top = gecici;
  uzunluk++;
}
```

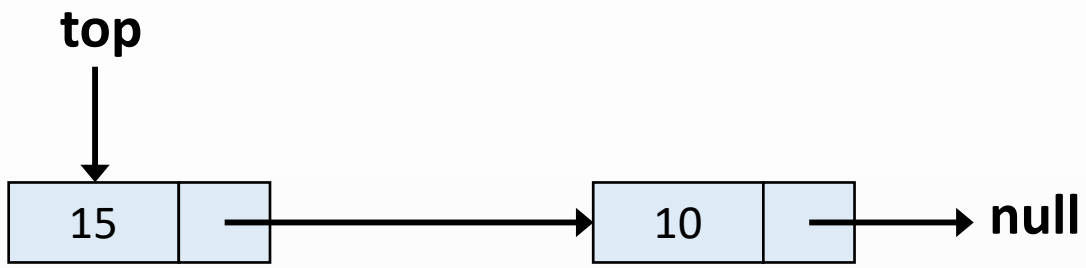
Uygulama - pop



uzunluk = 2

pop()

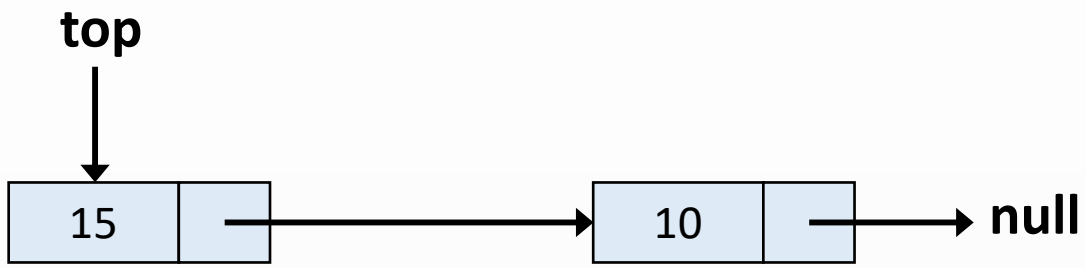
```
// Örnek değişkenleri
Dugum top;
int uzunluk;

public int pop() {
  int sonuc = top.veri;
  top = top.sonraki;
  uzunluk--;
  return sonuc;
}
```



uzunluk = 2

pop()

```
// Örnek değişkenleri
Dugum top;
int uzunluk;

public int pop() {
  int sonuc = top.veri;
  top = top.sonraki;
  uzunluk--;
  return sonuc;
}
```

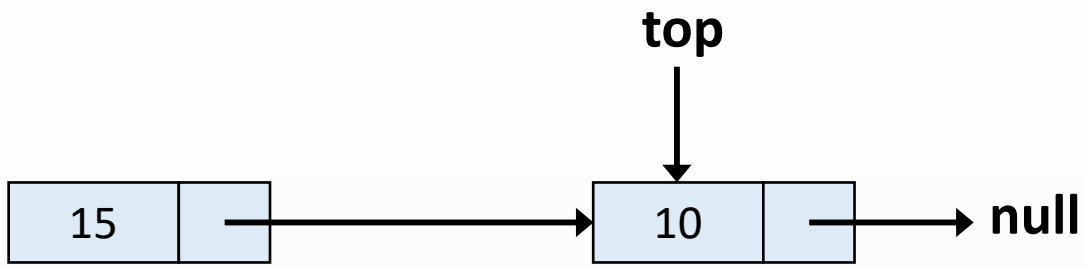


uzunluk = 2

sonuc = 15

pop()

```java
// Örnek değişkenleri
Dugum top;
int uzunluk;

public int pop() {
  int sonuc = top.veri;
  top = top.sonraki;
  uzunluk--;
  return sonuc;
}
```

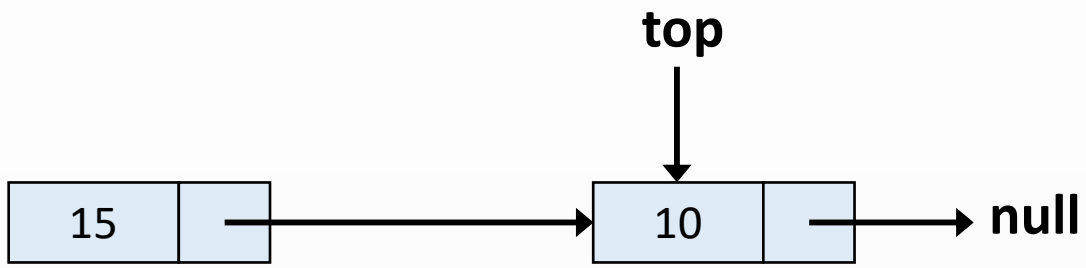


uzunluk = 2

sonuc = 15

pop()

```
// Örnek değişkenleri
Dugum top;
int uzunluk;

public int pop() {
  int sonuc = top.veri;
  top = top.sonraki;
  uzunluk--;
  return sonuc;
}
```

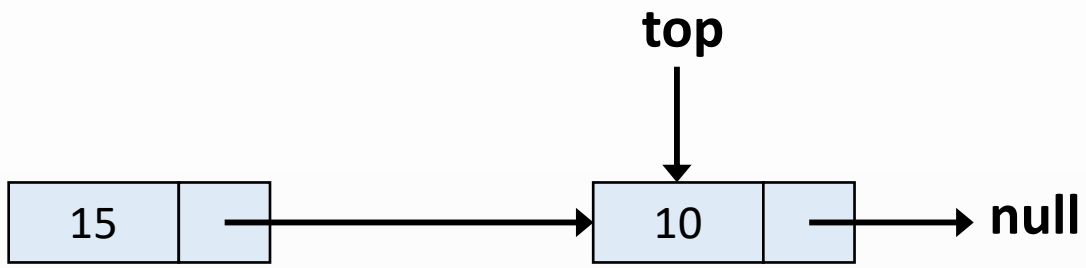


uzunluk = 1

sonuc = 15

pop()

```
// Örnek değişkenleri
Dugum top;
int uzunluk;

public int pop() {
  int sonuc = top.veri;
  top = top.sonraki;
  uzunluk--;
  return sonuc;
}
```

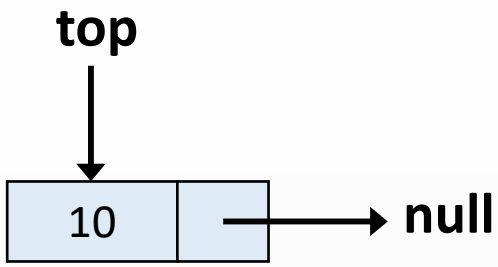


uzunluk = 1

sonuc = 15

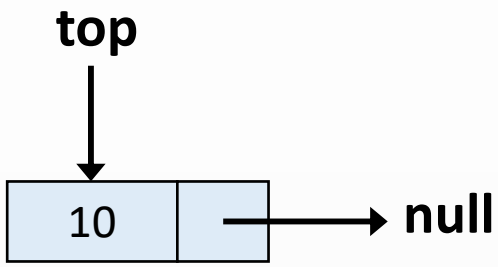
pop()

```
// Örnek değişkenleri
Dugum top;
int uzunluk;

public int pop() {
  int sonuc = top.veri;
  top = top.sonraki;
  uzunluk--;
  return sonuc;
}
```



uzunluk = 1

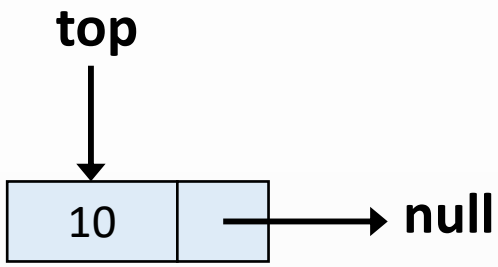
```
// Örnek değişkenleri
Dugum top;
int uzunluk;

public int pop() {
  int sonuc = top.veri;
  top = top.sonraki;
  uzunluk--;
  return sonuc;
}
```



uzunluk = 1

pop()

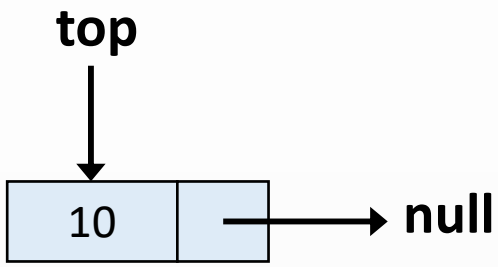
```
// Örnek değişkenleri
Dugum top;
int uzunluk;

public int pop() {
  int sonuc = top.veri;
  top = top.sonraki;
  uzunluk--;
  return sonuc;
}
```



uzunluk = 1

pop()

```
// Örnek değişkenleri
Dugum top;
int uzunluk;

public int pop() {
  int sonuc = top.veri;
  top = top.sonraki;
  uzunluk--;
  return sonuc;
}
```

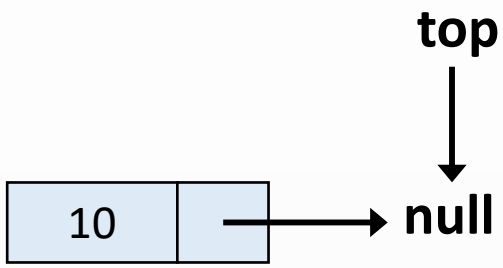


uzunluk = 1

sonuc = 10

pop()

```java
// Örnek değişkenleri
Dugum top;
int uzunluk;

public int pop() {
  int sonuc = top.veri;
  top = top.sonraki;
  uzunluk--;
  return sonuc;
}
```

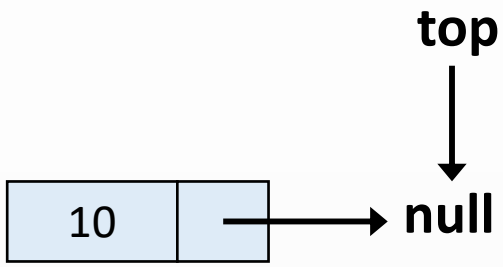


uzunluk = 1

sonuc = 10

pop()

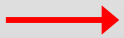
```
// Örnek değişkenleri
Dugum top;
int uzunluk;

public int pop() {
  int sonuc = top.veri;
  top = top.sonraki;
  uzunluk--;
  return sonuc;
}
```



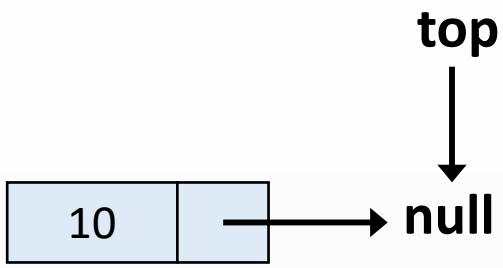
uzunluk = 0

sonuc = 10

pop()



```
// Örnek değişkenleri
Dugum top;
int uzunluk;

public int pop() {
  int sonuc = top.veri;
  top = top.sonraki;
  uzunluk--;
  return sonuc;
}
```

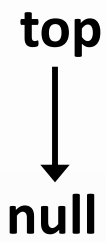


uzunluk = 0

sonuc = 10

pop()

```
// Örnek değişkenleri
Dugum top;
int uzunluk;

public int pop() {
  int sonuc = top.veri;
  top = top.sonraki;
  uzunluk--;
  return sonuc;
}
```



uzunluk = 0

```
// Örnek değişkenleri
Dugum top;
int uzunluk;

public int pop() {
  int sonuc = top.veri;
  top = top.sonraki;
  uzunluk--;
  return sonuc;
}
```

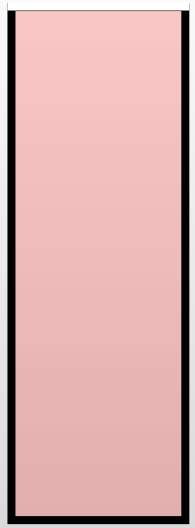
Karakter Dizisini (String) Tersine Çevirme

```
Stack<Character> yigit = new Stack<>();
char[] karakterler = str.toCharArray();
 for(char c : karakterler) {
    yigit.push(c);
 }
 for(int i = 0; i < str.length(); i++) {
    karakterler[i] = yigit.pop();
 }
 return new String(karakterler);
```

```
String str = "ABCD";
```

```
Stack<Character> yigit = new Stack<>();
char[] karakterler = str.toCharArray();
 for(char c : karakterler) {
    yigit.push(c);
 }
 for(int i = 0; i < str.length(); i++) {
    karakterler[i] = yigit.pop();
 }
 return new String(karakterler);
```

```
String str = "ABCD";
```



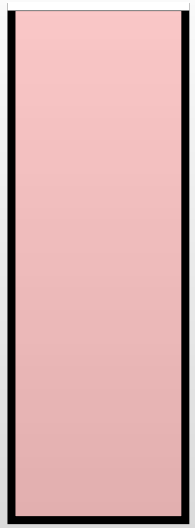
yigit

```
Stack<Character> yigit = new Stack<>();
char[] karakterler = str.toCharArray();
 for(char c : karakterler) {
    yigit.push(c);
 }
 for(int i = 0; i < str.length(); i++) {
    karakterler[i] = yigit.pop();
 }
 return new String(karakterler);
```

```
String str = "ABCD";
```

karakterler[]

A	B	C	D
0	1	2	3



yigit

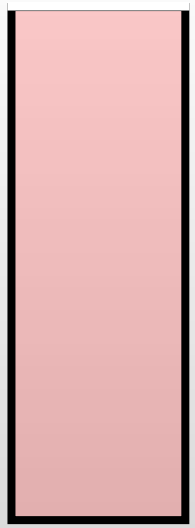
```
Stack<Character> yigit = new Stack<>();
char[] karakterler = str.toCharArray();
 for(char c : karakterler) {
    yigit.push(c);
 }
 for(int i = 0; i < str.length(); i++) {
    karakterler[i] = yigit.pop();
 }
 return new String(karakterler);
```

```
String str = "ABCD";
```

karakterler[]

A	B	C	D
0	1	2	3

```
char c = 'A'
```



yigit

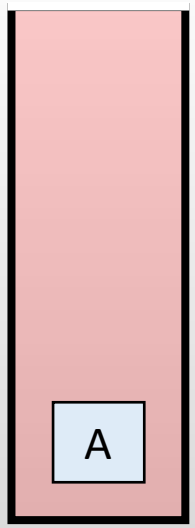
```
Stack<Character> yigit = new Stack<>();
char[] karakterler = str.toCharArray();
 for(char c : karakterler) {
    yigit.push(c);
 }
 for(int i = 0; i < str.length(); i++) {
    karakterler[i] = yigit.pop();
 }
 return new String(karakterler);
```

```
String str = "ABCD";
```

karakterler[]

A	B	C	D
0	1	2	3

```
char c = 'A'
```



yigit

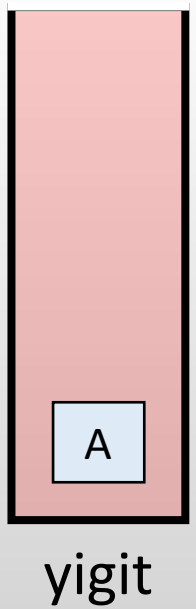
```
Stack<Character> yigit = new Stack<>();
char[] karakterler = str.toCharArray();
 for(char c : karakterler) {
    yigit.push(c);
 }
 for(int i = 0; i < str.length(); i++) {
    karakterler[i] = yigit.pop();
 }
 return new String(karakterler);
```

```
String str = "ABCD";
```

karakterler[]

A	B	C	D
0	1	2	3

```
char c = 'B'
```



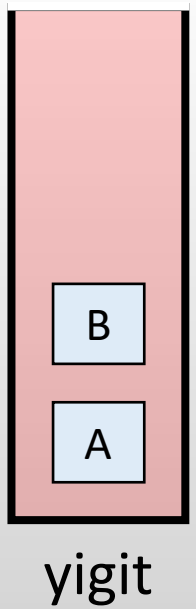

```
Stack<Character> yigit = new Stack<>();
char[] karakterler = str.toCharArray();
 for(char c : karakterler) {
    yigit.push(c);
 }
 for(int i = 0; i < str.length(); i++) {
    karakterler[i] = yigit.pop();
 }
 return new String(karakterler);
```

```
String str = "ABCD";
```

karakterler[]

A	B	C	D
0	1	2	3

```
char c = 'B'
```



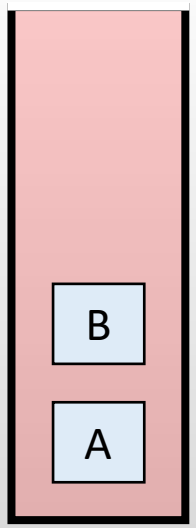

```
Stack<Character> yigit = new Stack<>();
char[] karakterler = str.toCharArray();
 for(char c : karakterler) {
    yigit.push(c);
 }
 for(int i = 0; i < str.length(); i++) {
    karakterler[i] = yigit.pop();
 }
 return new String(karakterler);
```

```
String str = "ABCD";
```

karakterler[]

A	B	C	D
0	1	2	3

```
char c = 'C'
```



yigit

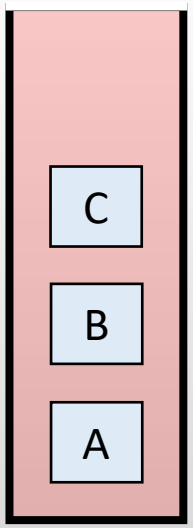
```
Stack<Character> yigit = new Stack<>();
char[] karakterler = str.toCharArray();
 for(char c : karakterler) {
    yigit.push(c);
 }
 for(int i = 0; i < str.length(); i++) {
    karakterler[i] = yigit.pop();
 }
 return new String(karakterler);
```

```
String str = "ABCD";
```

karakterler[]

A	B	C	D
0	1	2	3

```
char c = 'C'
```



yigit

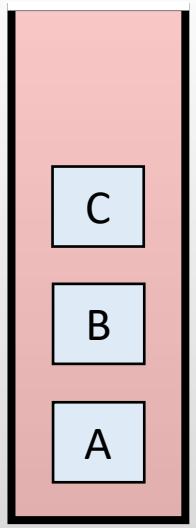
```
Stack<Character> yigit = new Stack<>();
char[] karakterler = str.toCharArray();
 for(char c : karakterler) {
    yigit.push(c);
 }
 for(int i = 0; i < str.length(); i++) {
    karakterler[i] = yigit.pop();
 }
 return new String(karakterler);
```

```
String str = "ABCD";
```

karakterler[]

A	B	C	D
0	1	2	3

```
char c = 'D'
```



yigit

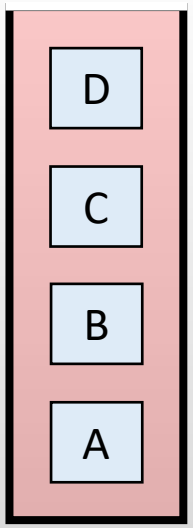
```
Stack<Character> yigit = new Stack<>();
char[] karakterler = str.toCharArray();
 for(char c : karakterler) {
    yigit.push(c);
 }
 for(int i = 0; i < str.length(); i++) {
    karakterler[i] = yigit.pop();
 }
 return new String(karakterler);
```

```
String str = "ABCD";
```

karakterler[]

A	B	C	D
0	1	2	3

```
char c = 'D'
```



yigit

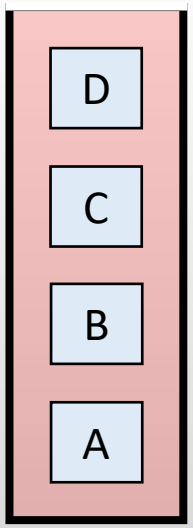
```
Stack<Character> yigit = new Stack<>();
char[] karakterler = str.toCharArray();
 for(char c : karakterler) {
    yigit.push(c);
 }
 for(int i = 0; i < str.length(); i++) {
    karakterler[i] = yigit.pop();
 }
 return new String(karakterler);
```

```
String str = "ABCD";
```

karakterler[]

A	B	C	D
0	1	2	3

```
char c = 'D'
```



yigit

```
Stack<Character> yigit = new Stack<>();
char[] karakterler = str.toCharArray();
 for(char c : karakterler) {
    yigit.push(c);
 }
 for(int i = 0; i < str.length(); i++) {
    karakterler[i] = yigit.pop();
 }
 return new String(karakterler);
```

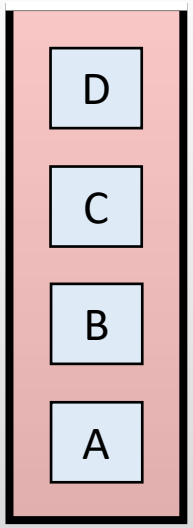
```
String str = "ABCD";
```

```
str.length() = 4
```

karakterler[]

A	B	C	D
0	1	2	3

```
i = 0
```



yigit

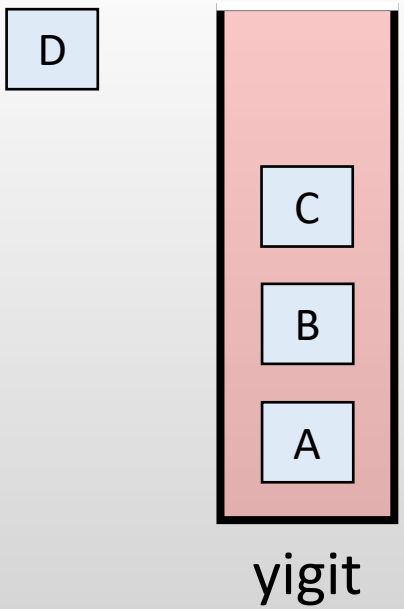
```
Stack<Character> yigit = new Stack<>();
char[] karakterler = str.toCharArray();
 for(char c : karakterler) {
    yigit.push(c);
 }
 for(int i = 0; i < str.length(); i++) {
    karakterler[i] = yigit.pop();
 }
 return new String(karakterler);
```

```
String str = "ABCD";
```

karakterler[]	A	B	C	D
	0	1	2	3

```
str.length() = 4
```

```
i = 0
```




```
Stack<Character> yigit = new Stack<>();
char[] karakterler = str.toCharArray();
 for(char c : karakterler) {
    yigit.push(c);
 }
 for(int i = 0; i < str.length(); i++) {
    karakterler[i] = yigit.pop();
 }
 return new String(karakterler);
```

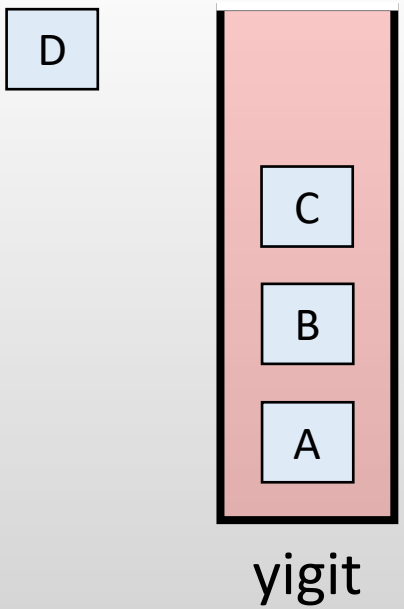
```
String str = "ABCD";
```

```
str.length() = 4
```

karakterler[]

D	B	C	D
0	1	2	3

```
i = 0
```




```
Stack<Character> yigit = new Stack<>();
char[] karakterler = str.toCharArray();
 for(char c : karakterler) {
    yigit.push(c);
 }
 for(int i = 0; i < str.length(); i++) {
    karakterler[i] = yigit.pop();
 }
 return new String(karakterler);
```

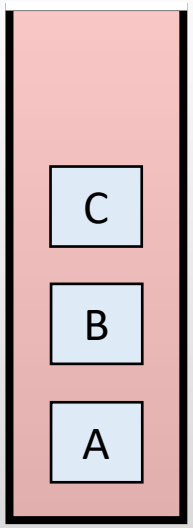
```
String str = "ABCD";
```

```
str.length() = 4
```

karakterler[]

D	B	C	D
0	1	2	3

```
i = 1
```



yigit

```
Stack<Character> yigit = new Stack<>();
char[] karakterler = str.toCharArray();
 for(char c : karakterler) {
    yigit.push(c);
 }
 for(int i = 0; i < str.length(); i++) {
    karakterler[i] = yigit.pop();
 }
 return new String(karakterler);
```

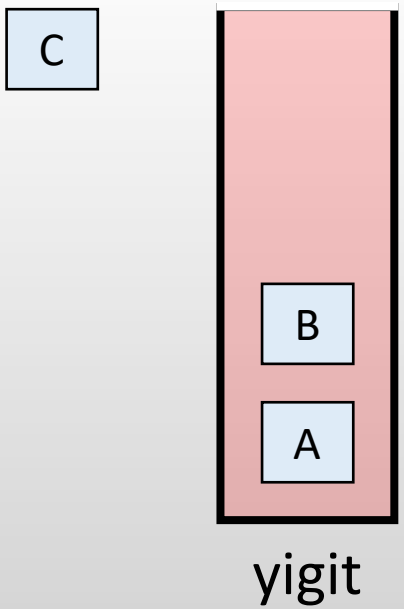
```
String str = "ABCD";
```

```
str.length() = 4
```

karakterler[]

D	B	C	D
0	1	2	3

```
i = 1
```




```
Stack<Character> yigit = new Stack<>();
char[] karakterler = str.toCharArray();
 for(char c : karakterler) {
    yigit.push(c);
 }
 for(int i = 0; i < str.length(); i++) {
    karakterler[i] = yigit.pop();
 }
 return new String(karakterler);
```

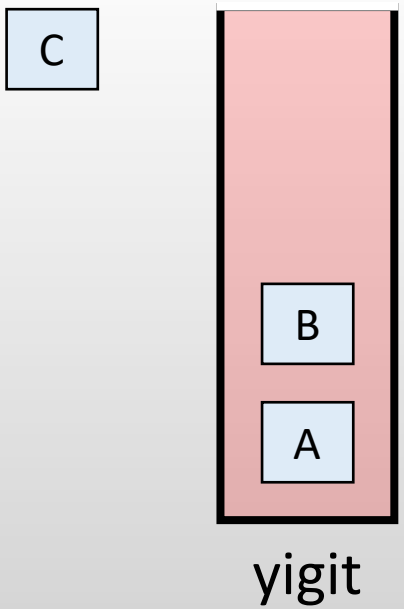
```
String str = "ABCD";
```

```
str.length() = 4
```

karakterler[]

D	C	C	D
0	1	2	3

```
i = 1
```




```
Stack<Character> yigit = new Stack<>();
char[] karakterler = str.toCharArray();
 for(char c : karakterler) {
    yigit.push(c);
 }
 for(int i = 0; i < str.length(); i++) {
    karakterler[i] = yigit.pop();
 }
 return new String(karakterler);
```

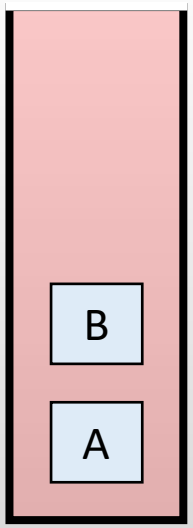
```
String str = "ABCD";
```

```
str.length() = 4
```

karakterler[]

D	C	C	D
0	1	2	3

```
i = 2
```



yigit

```
Stack<Character> yigit = new Stack<>();
char[] karakterler = str.toCharArray();
 for(char c : karakterler) {
    yigit.push(c);
 }
 for(int i = 0; i < str.length(); i++) {
    karakterler[i] = yigit.pop();
 }
 return new String(karakterler);
```

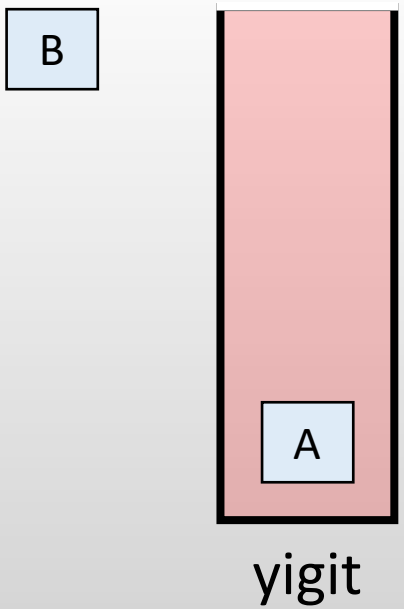
```
String str = "ABCD";
```

```
str.length() = 4
```

karakterler[]

D	C	C	D
0	1	2	3

```
i = 2
```




```
Stack<Character> yigit = new Stack<>();
char[] karakterler = str.toCharArray();
 for(char c : karakterler) {
    yigit.push(c);
 }
 for(int i = 0; i < str.length(); i++) {
    karakterler[i] = yigit.pop();
 }
 return new String(karakterler);
```

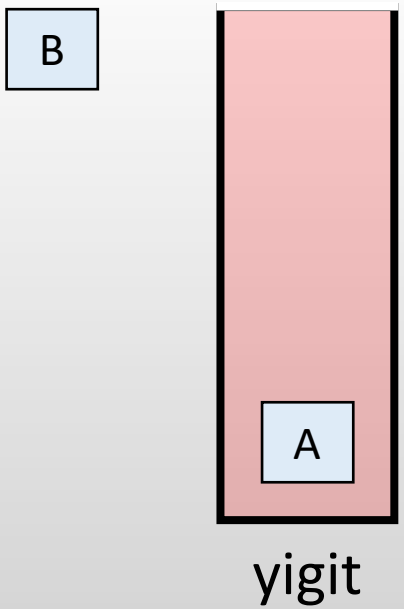
```
String str = "ABCD";
```

```
str.length() = 4
```

karakterler[]

D	C	B	D
0	1	2	3

```
i = 2
```




```
Stack<Character> yigit = new Stack<>();
char[] karakterler = str.toCharArray();
 for(char c : karakterler) {
    yigit.push(c);
 }
 for(int i = 0; i < str.length(); i++) {
    karakterler[i] = yigit.pop();
 }
 return new String(karakterler);
```

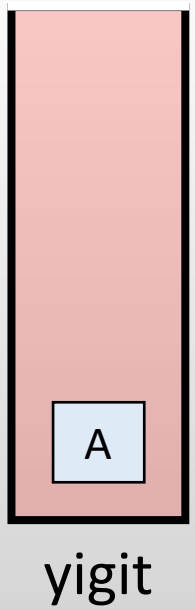
```
String str = "ABCD";
```

```
str.length() = 4
```

karakterler[]

D	C	B	D
0	1	2	3

```
i = 3
```



yigit

```
Stack<Character> yigit = new Stack<>();
char[] karakterler = str.toCharArray();
 for(char c : karakterler) {
    yigit.push(c);
 }
 for(int i = 0; i < str.length(); i++) {
    karakterler[i] = yigit.pop();
 }
 return new String(karakterler);
```

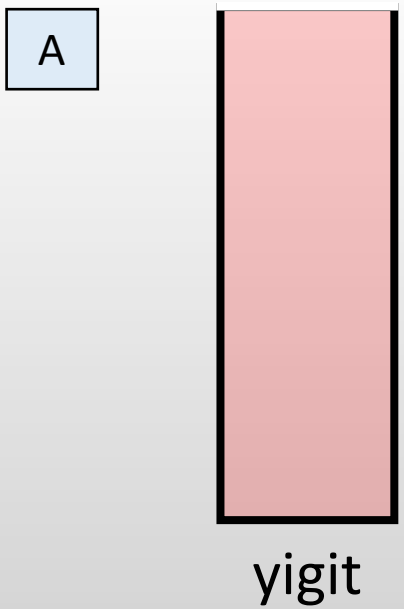
```
String str = "ABCD";
```

```
str.length() = 4
```

karakterler[]

D	C	B	D
0	1	2	3

```
i = 3
```



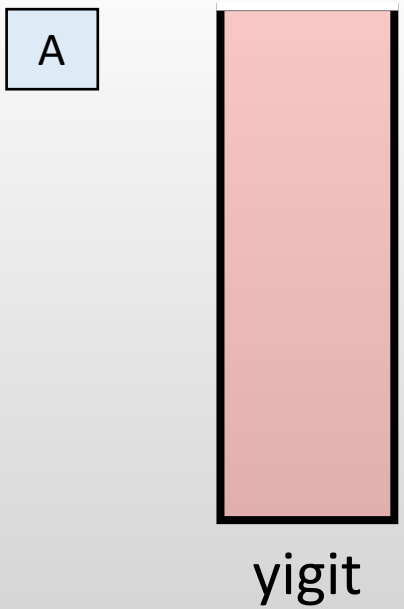

```
Stack<Character> yigit = new Stack<>();
char[] karakterler = str.toCharArray();
 for(char c : karakterler) {
    yigit.push(c);
 }
 for(int i = 0; i < str.length(); i++) {
    karakterler[i] = yigit.pop();
 }
 return new String(karakterler);
```

```
String str = "ABCD";
```

karakterler[]	D	C	B	A
	0	1	2	3

str.length() = 4

i = 3




```
Stack<Character> yigit = new Stack<>();
char[] karakterler = str.toCharArray();
 for(char c : karakterler) {
    yigit.push(c);
 }
 for(int i = 0; i < str.length(); i++) {
    karakterler[i] = yigit.pop();
 }
 return new String(karakterler);
```

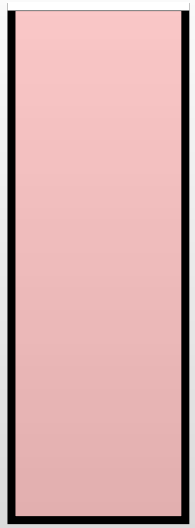
```
String str = "ABCD";
```

```
str.length() = 4
```

karakterler[]

D	C	B	A
0	1	2	3

```
i = 4
```



yigit

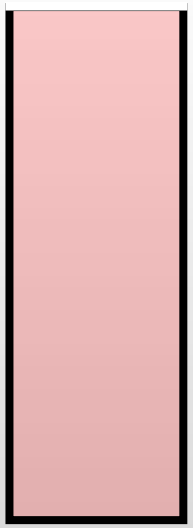
```
Stack<Character> yigit = new Stack<>();
char[] karakterler = str.toCharArray();
 for(char c : karakterler) {
    yigit.push(c);
 }
 for(int i = 0; i < str.length(); i++) {
    karakterler[i] = yigit.pop();
 }
 return new String(karakterler);
```

```
String str = "ABCD";
```

karakterler[]

D	C	B	A
0	1	2	3

```
str.length() = 4

i = 4

return → "DCBA"
```



yigit

```
Stack<Character> yigit = new Stack<>();
char[] karakterler = str.toCharArray();
 for(char c : karakterler) {
    yigit.push(c);
 }
 for(int i = 0; i < str.length(); i++) {
    karakterler[i] = yigit.pop();
 }
 return new String(karakterler);
```

Sonraki En Büyük Eleman Problemi

- Verilen bir dizide, her eleman için kendisinden daha büyük olan bir sonraki elemanı bulma.

- **Örnek**:
  - **Girdi**: `dizi  = {4, 7, 3, 4,  8,  1}`
  - **Çıktı**: `sonuc = {7, 8, 4, 8, -1, -1}`

Sonraki En Büyük Eleman Problemi

```
int[] sonrakiBuyukEleman(int[] dizi) {
   int[] sonuc = new int[dizi.length];
   Stack<Integer> yigit = new Stack<>();
   for(int i = dizi.length - 1; i >= 0; i--) {
      if(!yigit.isEmpty()) {
         while(!yigit.isEmpty() &&
               yigit.peek() <= dizi[i]) {
            yigit.pop();
         }
      }
      if(yigit.isEmpty()) {
         sonuc[i] = -1;
      }
      else {
         sonuc[i] = yigit.peek();
      }
      yigit.push(dizi[i]);
   }
   return sonuc;
}
```

Sonraki En Büyük Eleman Problemi

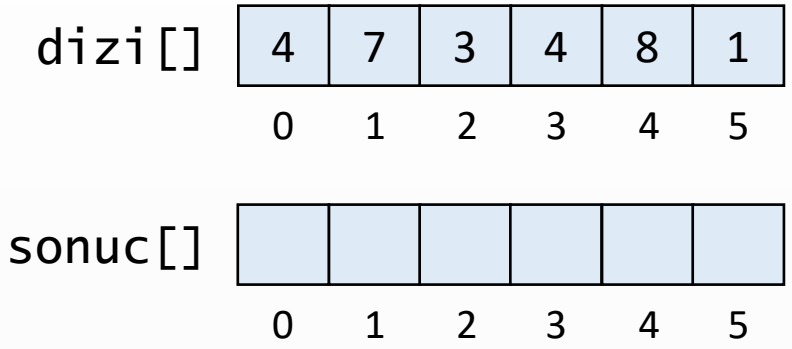
```
int[] sonrakiBuyukEleman(int[] dizi) {
   int[] sonuc = new int[dizi.length];
   Stack<Integer> yigit = new Stack<>();
   for(int i = dizi.length - 1; i >= 0; i--) {
      if(!yigit.isEmpty()) {
         while(!yigit.isEmpty() &&
               yigit.peek() <= dizi[i]) {
            yigit.pop();
         }
      }
      if(yigit.isEmpty()) {
         sonuc[i] = -1;
      }
      else {
         sonuc[i] = yigit.peek();
      }
      yigit.push(dizi[i]);
   }
   return sonuc;
}
```

```
sonrakiBuyukEleman(dizi);
```

dizi[]	4	7	3	4	8	1
	0	1	2	3	4	5

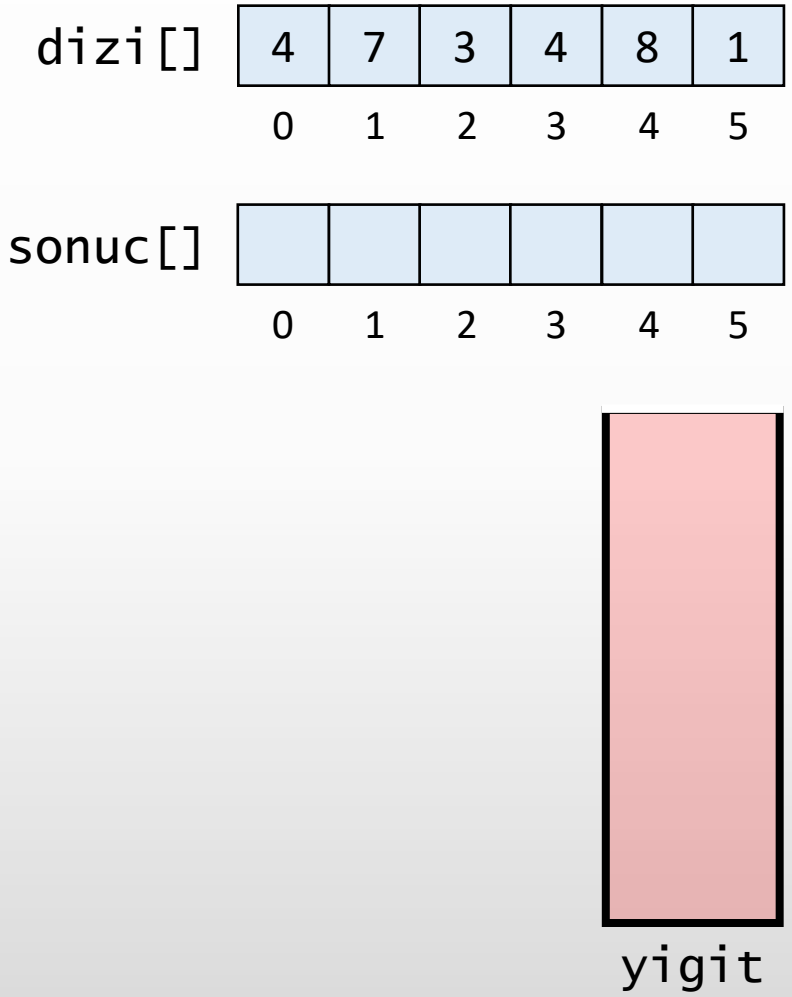
```
int[] sonrakiBuyukEleman(int[] dizi) {
   int[] sonuc = new int[dizi.length];
   Stack<Integer> yigit = new Stack<>();
   for(int i = dizi.length - 1; i >= 0; i--) {
      if(!yigit.isEmpty()) {
         while(!yigit.isEmpty() &&
               yigit.peek() <= dizi[i]) {
            yigit.pop();
         }
      }
      if(yigit.isEmpty()) {
         sonuc[i] = -1;
      }
      else {
         sonuc[i] = yigit.peek();
      }
      yigit.push(dizi[i]);
   }
   return sonuc;
}
```

```
sonrakiBuyukEleman(dizi);
```



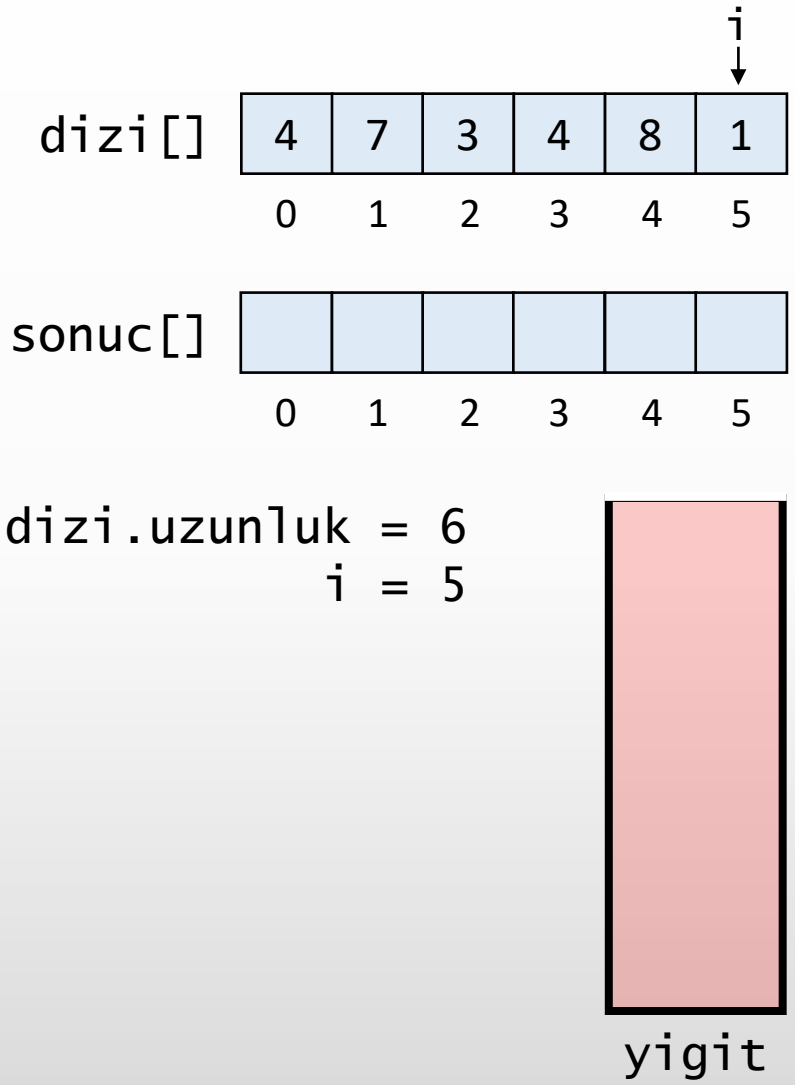

```
sonrakiBuyukEleman(dizi);
```

```
int[] sonrakiBuyukEleman(int[] dizi) {
   int[] sonuc = new int[dizi.length];
   Stack<Integer> yigit = new Stack<>();
   for(int i = dizi.length - 1; i >= 0; i--) {
      if(!yigit.isEmpty()) {
         while(!yigit.isEmpty() &&
               yigit.peek() <= dizi[i]) {
            yigit.pop();
         }
      }
      if(yigit.isEmpty()) {
         sonuc[i] = -1;
      }
      else {
         sonuc[i] = yigit.peek();
      }
      yigit.push(dizi[i]);
   }
   return sonuc;
}
```



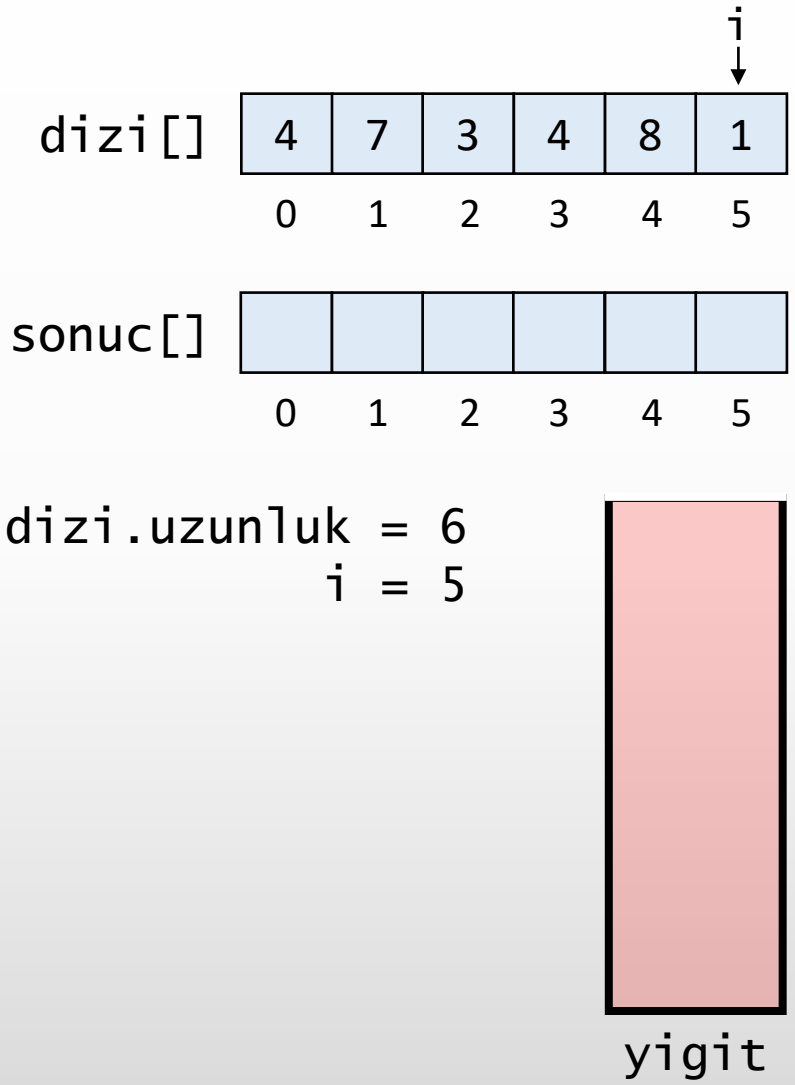

```
sonrakiBuyukEleman(dizi);
```

```java
int[] sonrakiBuyukEleman(int[] dizi) {
   int[] sonuc = new int[dizi.length];
   Stack<Integer> yigit = new Stack<>();
   for(int i = dizi.length - 1; i >= 0; i--) {
      if(!yigit.isEmpty()) {
         while(!yigit.isEmpty() &&
               yigit.peek() <= dizi[i]) {
            yigit.pop();
         }
      }
      if(yigit.isEmpty()) {
         sonuc[i] = -1;
      }
      else {
         sonuc[i] = yigit.peek();
      }
      yigit.push(dizi[i]);
   }
   return sonuc;
}
```



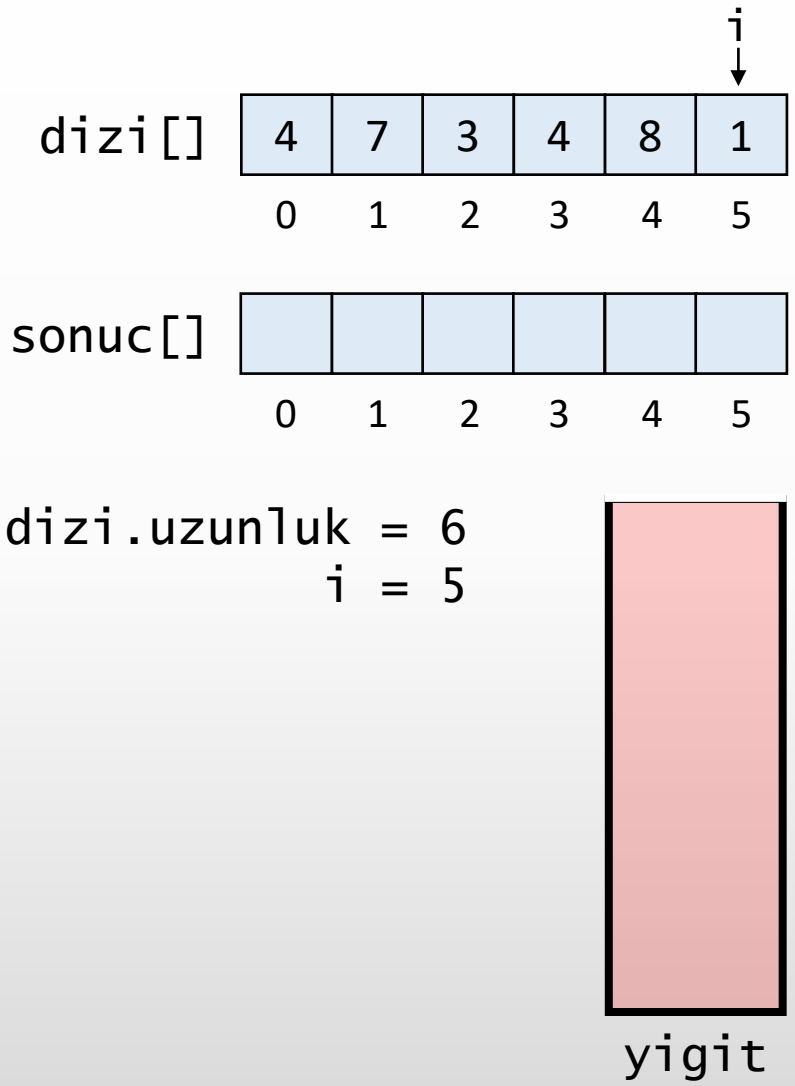

```
sonrakiBuyukEleman(dizi);
```

```java
int[] sonrakiBuyukEleman(int[] dizi) {
   int[] sonuc = new int[dizi.length];
   Stack<Integer> yigit = new Stack<>();
   for(int i = dizi.length - 1; i >= 0; i--) {
      if(!yigit.isEmpty()) {
         while(!yigit.isEmpty() &&
               yigit.peek() <= dizi[i]) {
            yigit.pop();
         }
      }
      if(yigit.isEmpty()) {
         sonuc[i] = -1;
      }
      else {
         sonuc[i] = yigit.peek();
      }
      yigit.push(dizi[i]);
   }
   return sonuc;
}
```



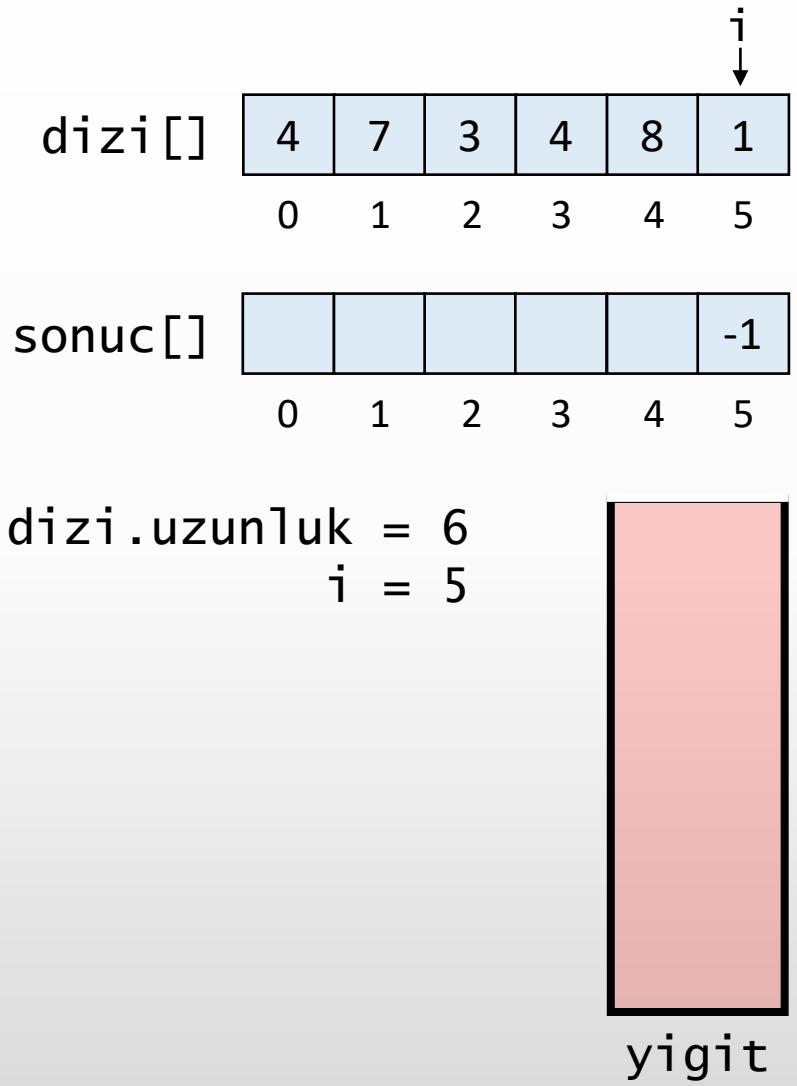

```
sonrakiBuyukEleman(dizi);
```

```
int[] sonrakiBuyukEleman(int[] dizi) {
   int[] sonuc = new int[dizi.length];
   Stack<Integer> yigit = new Stack<>();
   for(int i = dizi.length - 1; i >= 0; i--) {
      if(!yigit.isEmpty()) {
         while(!yigit.isEmpty() &&
               yigit.peek() <= dizi[i]) {
            yigit.pop();
         }
      }
      if(yigit.isEmpty()) {
         sonuc[i] = -1;
      }
      else {
         sonuc[i] = yigit.peek();
      }
      yigit.push(dizi[i]);
   }
   return sonuc;
}
```



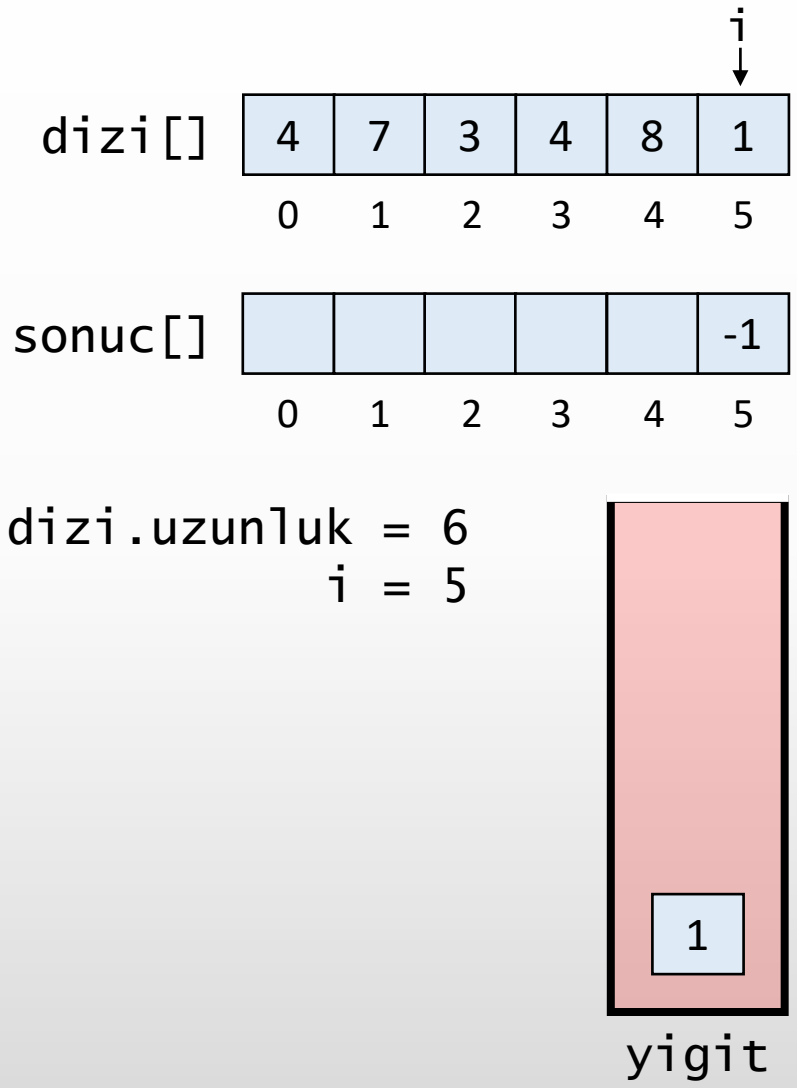

```
sonrakiBuyukEleman(dizi);
```

```java
int[] sonrakiBuyukEleman(int[] dizi) {
   int[] sonuc = new int[dizi.length];
   Stack<Integer> yigit = new Stack<>();
   for(int i = dizi.length - 1; i >= 0; i--) {
      if(!yigit.isEmpty()) {
         while(!yigit.isEmpty() &&
               yigit.peek() <= dizi[i]) {
            yigit.pop();
         }
      }
      if(yigit.isEmpty()) {
         sonuc[i] = -1;
      }
      else {
         sonuc[i] = yigit.peek();
      }
      yigit.push(dizi[i]);
   }
   return sonuc;
}
```



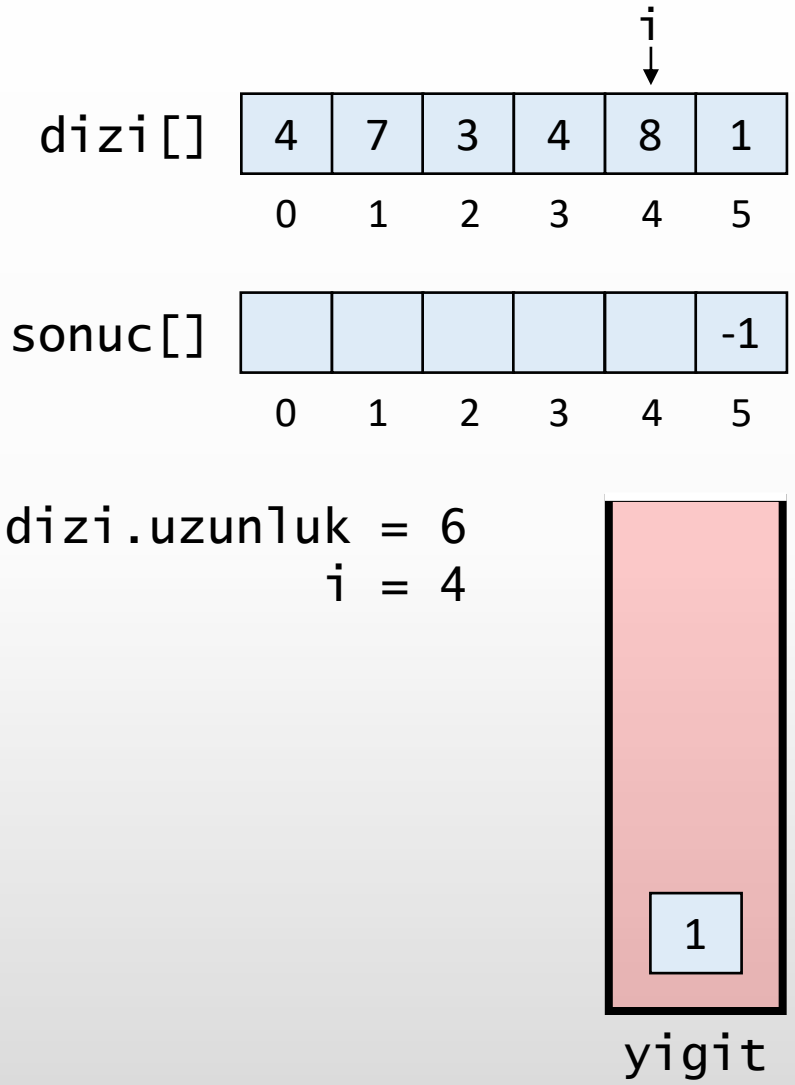
sonrakiBuyukEleman(dizi);

```
int[] sonrakiBuyukEleman(int[] dizi) {
   int[] sonuc = new int[dizi.length];
   Stack<Integer> yigit = new Stack<>();
   for(int i = dizi.length - 1; i >= 0; i--) {
      if(!yigit.isEmpty()) {
         while(!yigit.isEmpty() &&
               yigit.peek() <= dizi[i]) {
            yigit.pop();
         }
      }
      if(yigit.isEmpty()) {
         sonuc[i] = -1;
      }
      else {
         sonuc[i] = yigit.peek();
      }
      yigit.push(dizi[i]);
   }
   return sonuc;
}
```



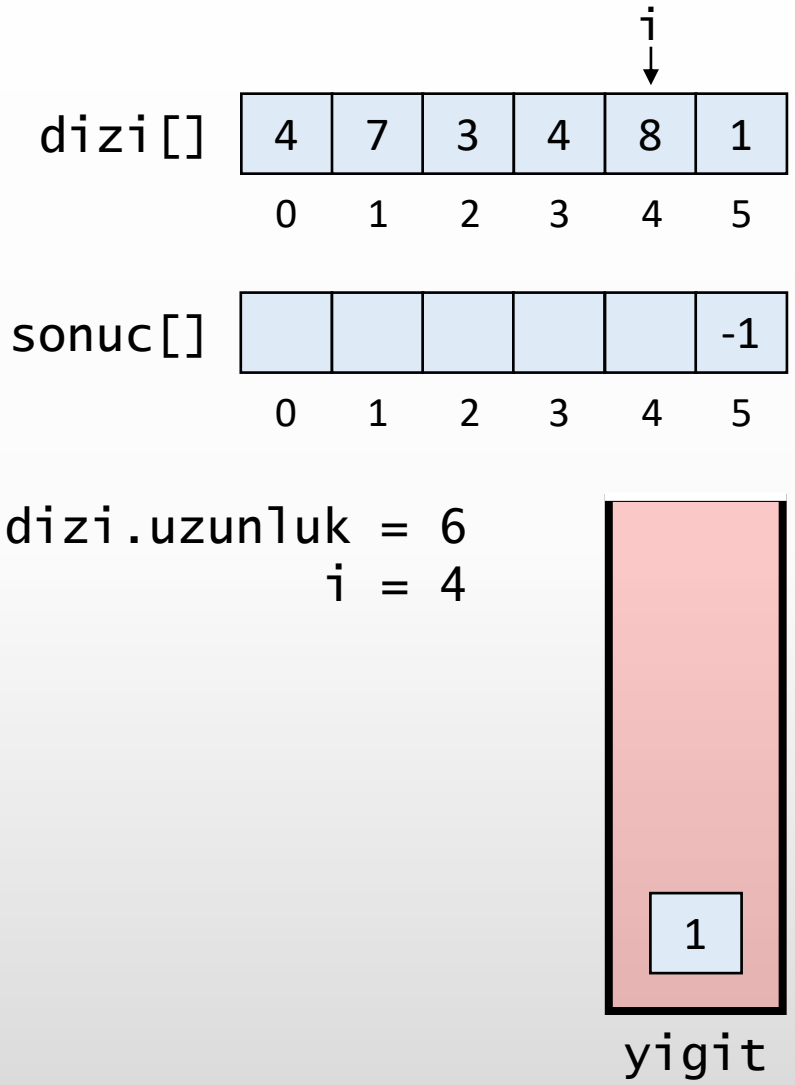
sonrakiBuyukEleman(dizi);

```
int[] sonrakiBuyukEleman(int[] dizi) {
   int[] sonuc = new int[dizi.length];
   Stack<Integer> yigit = new Stack<>();
   for(int i = dizi.length - 1; i >= 0; i--) {
      if(!yigit.isEmpty()) {
         while(!yigit.isEmpty() &&
               yigit.peek() <= dizi[i]) {
            yigit.pop();
         }
      }
      if(yigit.isEmpty()) {
         sonuc[i] = -1;
      }
      else {
         sonuc[i] = yigit.peek();
      }
      yigit.push(dizi[i]);
   }
   return sonuc;
}
```



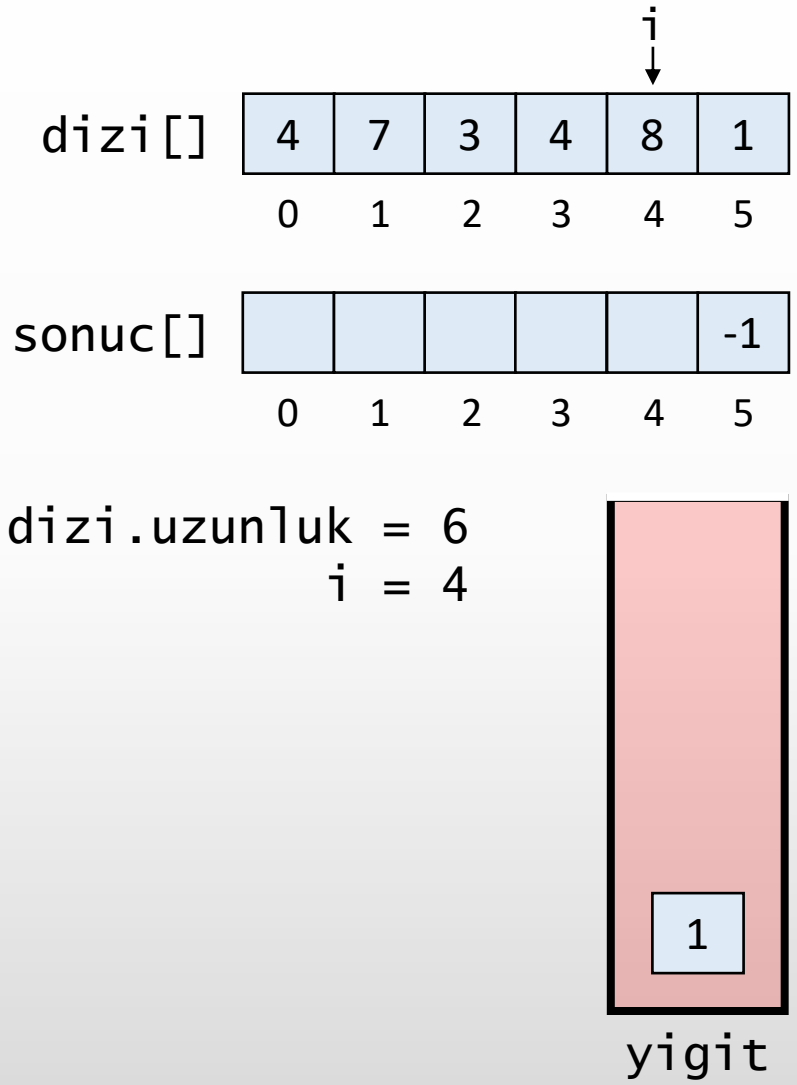

```
sonrakiBuyukEleman(dizi);
```

```
int[] sonrakiBuyukEleman(int[] dizi) {
   int[] sonuc = new int[dizi.length];
   Stack<Integer> yigit = new Stack<>();
   for(int i = dizi.length - 1; i >= 0; i--) {
      if(!yigit.isEmpty()) {
         while(!yigit.isEmpty() &&
               yigit.peek() <= dizi[i]) {
            yigit.pop();
         }
      }
      if(yigit.isEmpty()) {
         sonuc[i] = -1;
      }
      else {
         sonuc[i] = yigit.peek();
      }
      yigit.push(dizi[i]);
   }
   return sonuc;
}
```



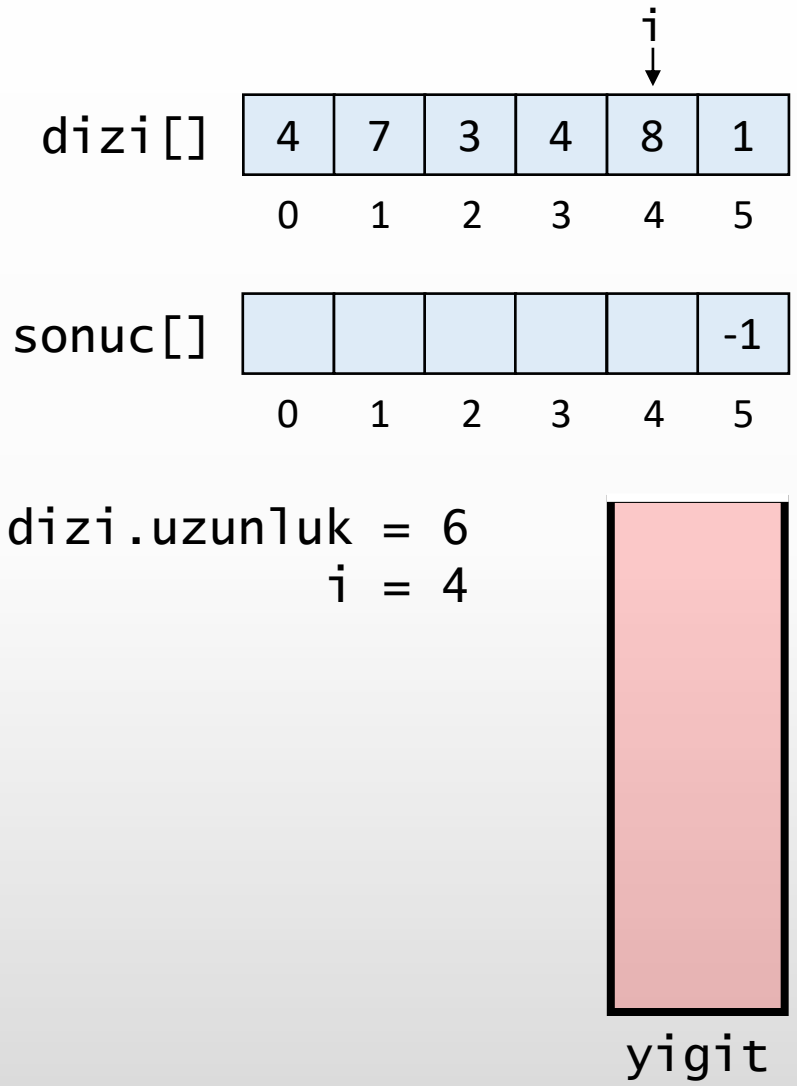

```
sonrakiBuyukEleman(dizi);
```

```
int[] sonrakiBuyukEleman(int[] dizi) {
   int[] sonuc = new int[dizi.length];
   Stack<Integer> yigit = new Stack<>();
   for(int i = dizi.length - 1; i >= 0; i--) {
      if(!yigit.isEmpty()) {
         while(!yigit.isEmpty() &&
               yigit.peek() <= dizi[i]) {
            yigit.pop();
         }
      }
      if(yigit.isEmpty()) {
         sonuc[i] = -1;
      }
      else {
         sonuc[i] = yigit.peek();
      }
      yigit.push(dizi[i]);
   }
   return sonuc;
}
```



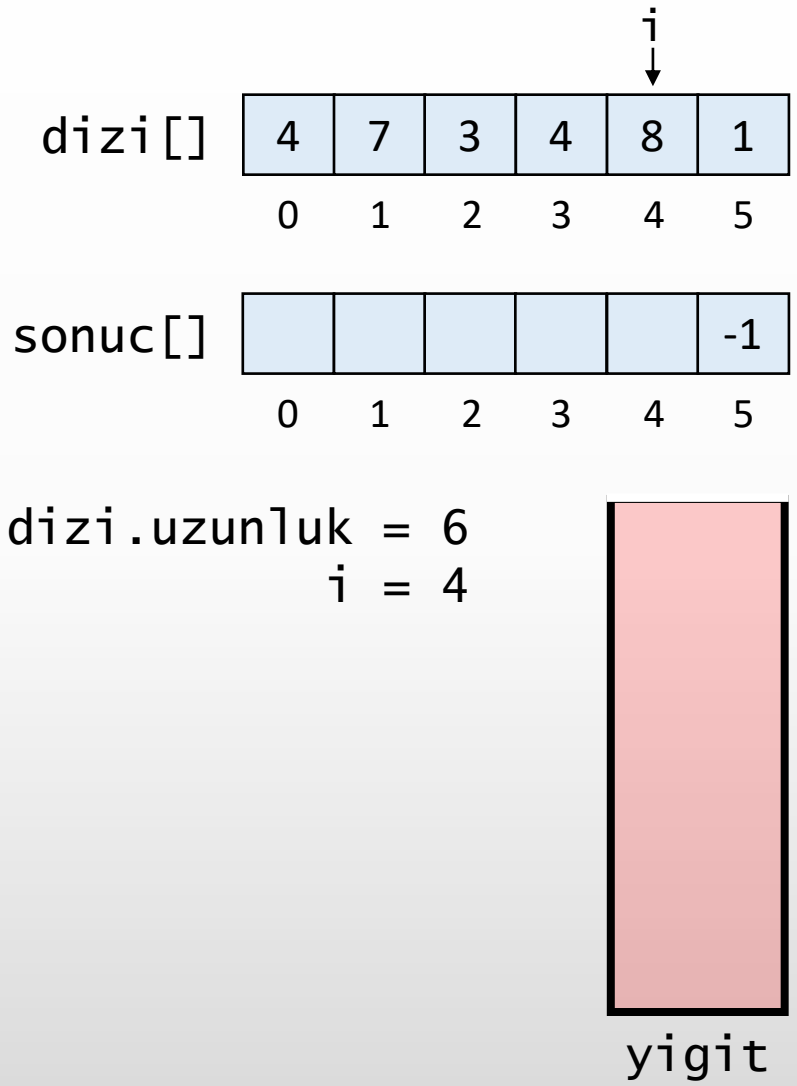

```
sonrakiBuyukEleman(dizi);
```

```
int[] sonrakiBuyukEleman(int[] dizi) {
   int[] sonuc = new int[dizi.length];
   Stack<Integer> yigit = new Stack<>();
   for(int i = dizi.length - 1; i >= 0; i--) {
      if(!yigit.isEmpty()) {
         while(!yigit.isEmpty() &&
               yigit.peek() <= dizi[i]) {
            yigit.pop();
         }
      }
      if(yigit.isEmpty()) {
         sonuc[i] = -1;
      }
      else {
         sonuc[i] = yigit.peek();
      }
      yigit.push(dizi[i]);
   }
   return sonuc;
}
```



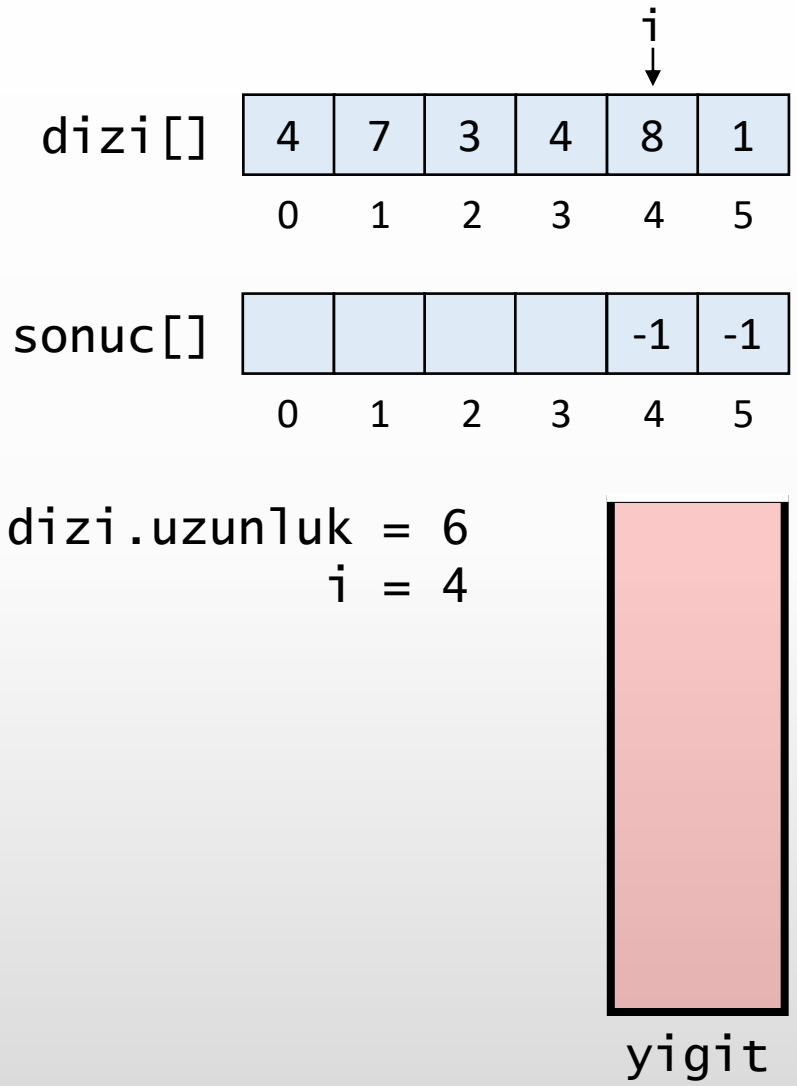

```
sonrakiBuyukEleman(dizi);
```

```
int[] sonrakiBuyukEleman(int[] dizi) {
   int[] sonuc = new int[dizi.length];
   Stack<Integer> yigit = new Stack<>();
   for(int i = dizi.length - 1; i >= 0; i--) {
      if(!yigit.isEmpty()) {
         while(!yigit.isEmpty() &&
               yigit.peek() <= dizi[i]) {
            yigit.pop();
         }
      }
      if(yigit.isEmpty()) {
         sonuc[i] = -1;
      }
      else {
         sonuc[i] = yigit.peek();
      }
      yigit.push(dizi[i]);
   }
   return sonuc;
}
```



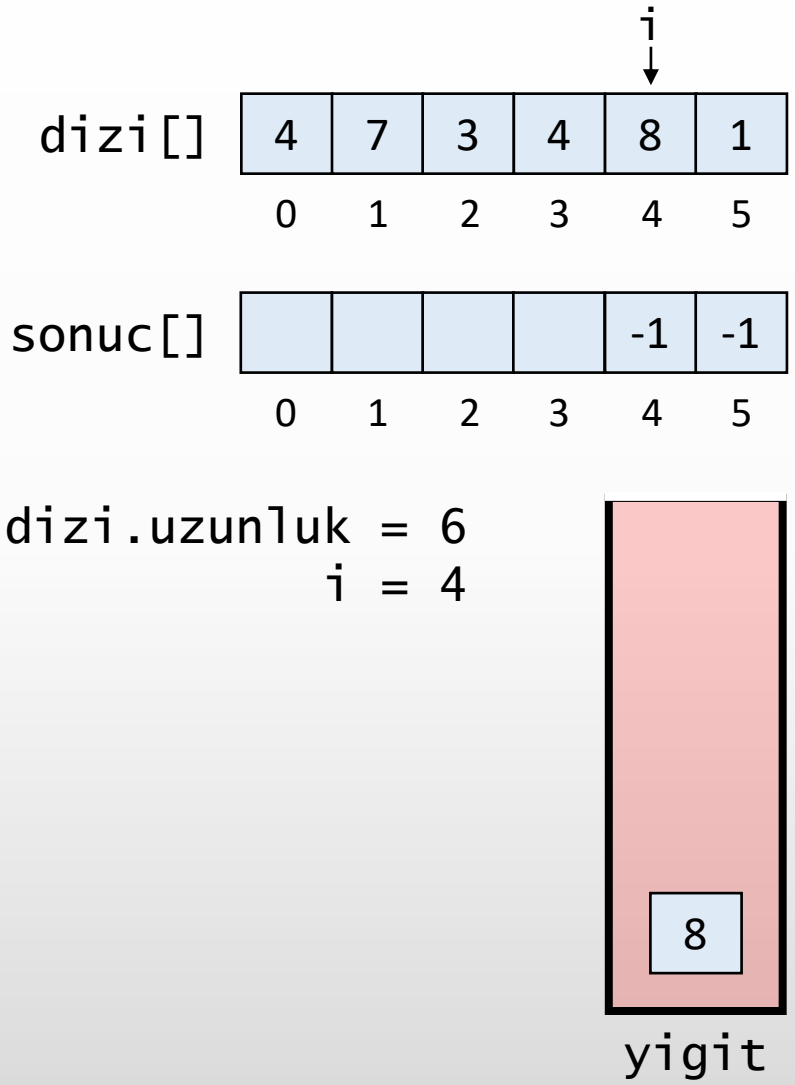

```
sonrakiBuyukEleman(dizi);
```

```java
int[] sonrakiBuyukEleman(int[] dizi) {
   int[] sonuc = new int[dizi.length];
   Stack<Integer> yigit = new Stack<>();
   for(int i = dizi.length - 1; i >= 0; i--) {
      if(!yigit.isEmpty()) {
         while(!yigit.isEmpty() &&
               yigit.peek() <= dizi[i]) {
            yigit.pop();
         }
      }
      if(yigit.isEmpty()) {
         sonuc[i] = -1;
      }
      else {
         sonuc[i] = yigit.peek();
      }
      yigit.push(dizi[i]);
   }
   return sonuc;
}
```



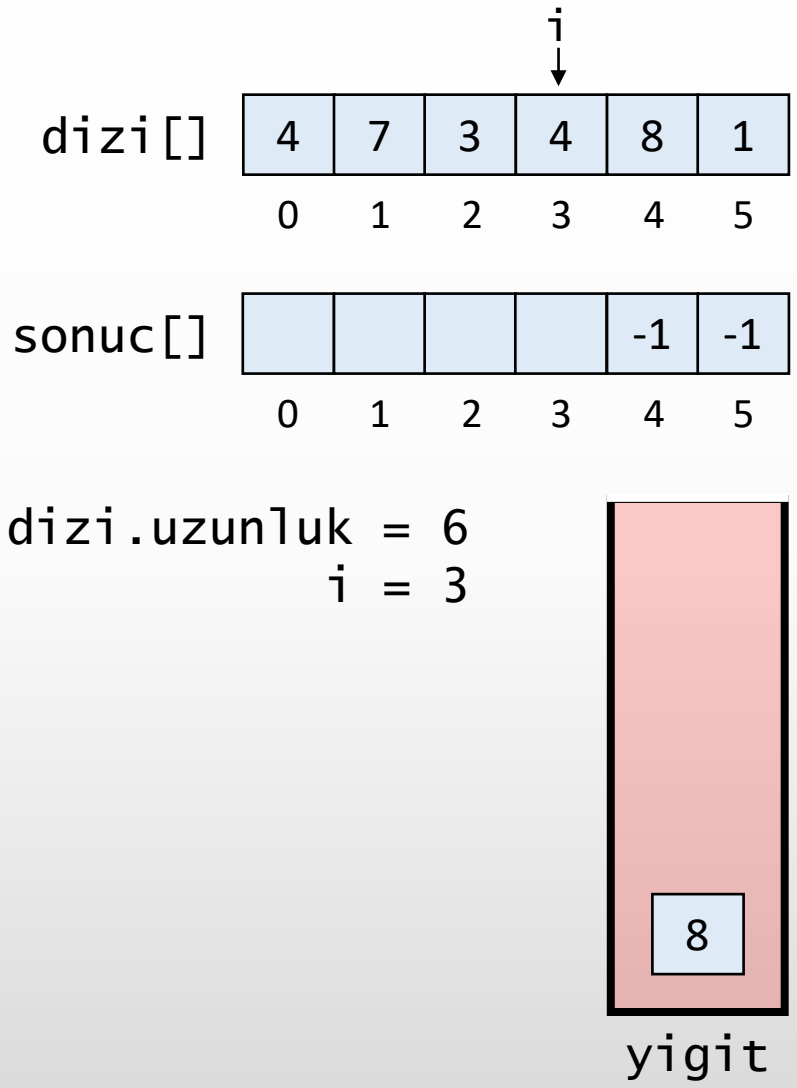

```
sonrakiBuyukEleman(dizi);
```

```java
int[] sonrakiBuyukEleman(int[] dizi) {
   int[] sonuc = new int[dizi.length];
   Stack<Integer> yigit = new Stack<>();
   for(int i = dizi.length - 1; i >= 0; i--) {
      if(!yigit.isEmpty()) {
         while(!yigit.isEmpty() &&
               yigit.peek() <= dizi[i]) {
            yigit.pop();
         }
      }
      if(yigit.isEmpty()) {
         sonuc[i] = -1;
      }
      else {
         sonuc[i] = yigit.peek();
      }
      yigit.push(dizi[i]);
   }
   return sonuc;
}
```



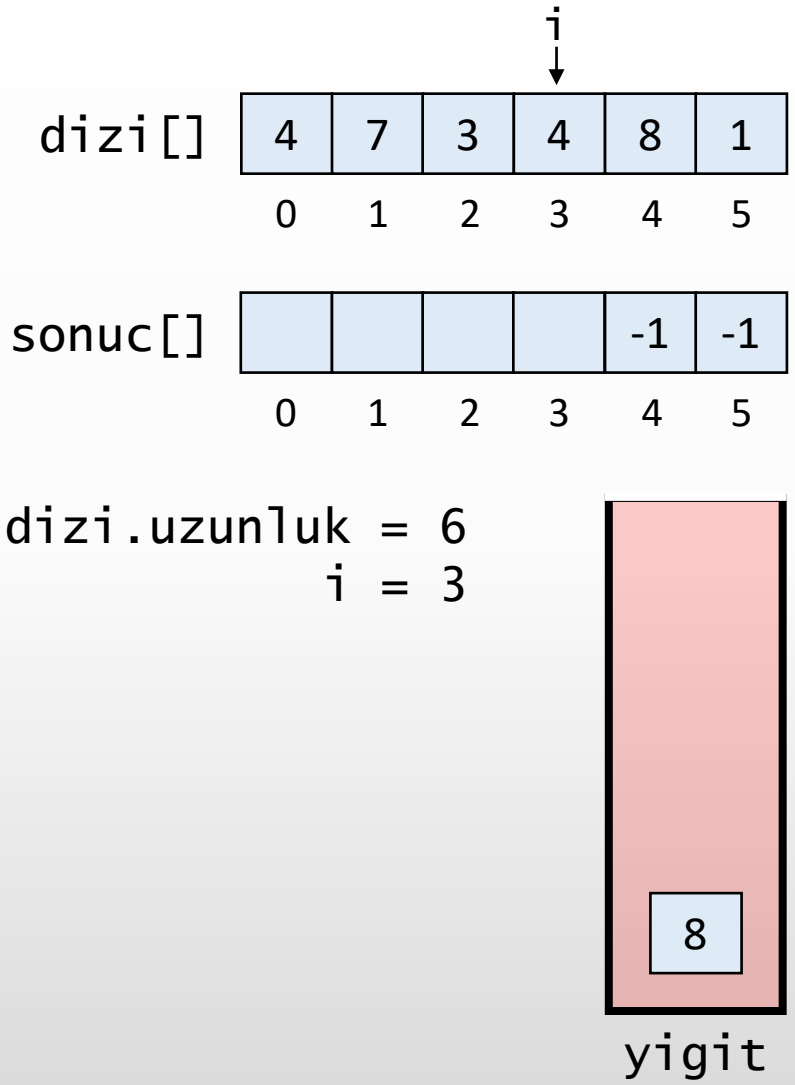

```
sonrakiBuyukEleman(dizi);
```

```
int[] sonrakiBuyukEleman(int[] dizi) {
   int[] sonuc = new int[dizi.length];
   Stack<Integer> yigit = new Stack<>();
   for(int i = dizi.length - 1; i >= 0; i--) {
      if(!yigit.isEmpty()) {
         while(!yigit.isEmpty() &&
               yigit.peek() <= dizi[i]) {
            yigit.pop();
         }
      }
      if(yigit.isEmpty()) {
         sonuc[i] = -1;
      }
      else {
         sonuc[i] = yigit.peek();
      }
      yigit.push(dizi[i]);
   }
   return sonuc;
}
```



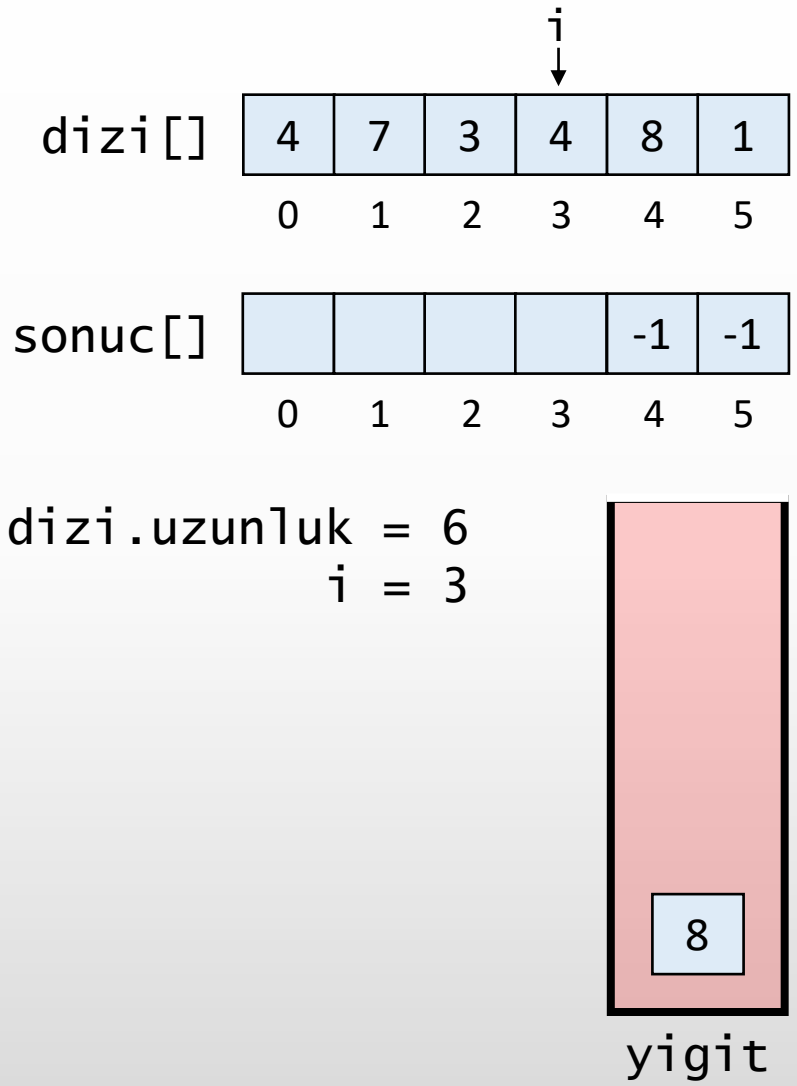

```
sonrakiBuyukEleman(dizi);
```

```java
int[] sonrakiBuyukEleman(int[] dizi) {
   int[] sonuc = new int[dizi.length];
   Stack<Integer> yigit = new Stack<>();
   for(int i = dizi.length - 1; i >= 0; i--) {
      if(!yigit.isEmpty()) {
         while(!yigit.isEmpty() &&
               yigit.peek() <= dizi[i]) {
            yigit.pop();
         }
      }
      if(yigit.isEmpty()) {
         sonuc[i] = -1;
      }
      else {
         sonuc[i] = yigit.peek();
      }
      yigit.push(dizi[i]);
   }
   return sonuc;
}
```



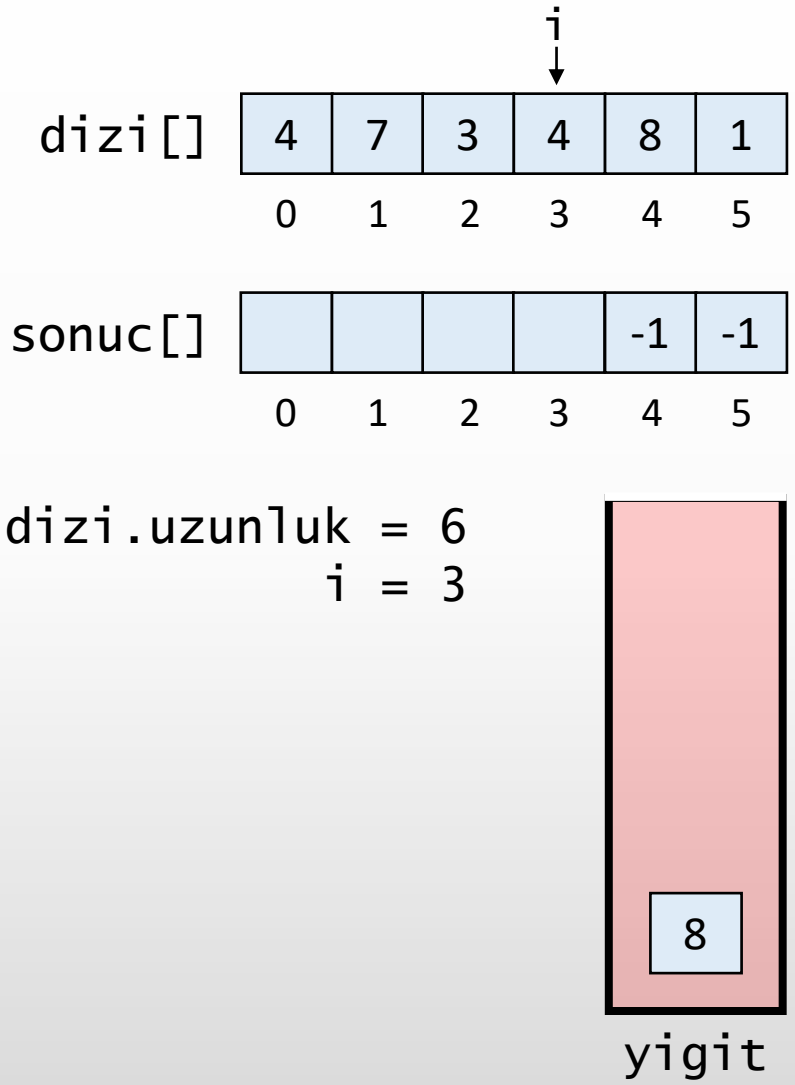

```
sonrakiBuyukEleman(dizi);
```

```
int[] sonrakiBuyukEleman(int[] dizi) {
   int[] sonuc = new int[dizi.length];
   Stack<Integer> yigit = new Stack<>();
   for(int i = dizi.length - 1; i >= 0; i--) {
      if(!yigit.isEmpty()) {
         while(!yigit.isEmpty() &&
               yigit.peek() <= dizi[i]) {
            yigit.pop();
         }
      }
      if(yigit.isEmpty()) {
         sonuc[i] = -1;
      }
      else {
         sonuc[i] = yigit.peek();
      }
      yigit.push(dizi[i]);
   }
   return sonuc;
}
```



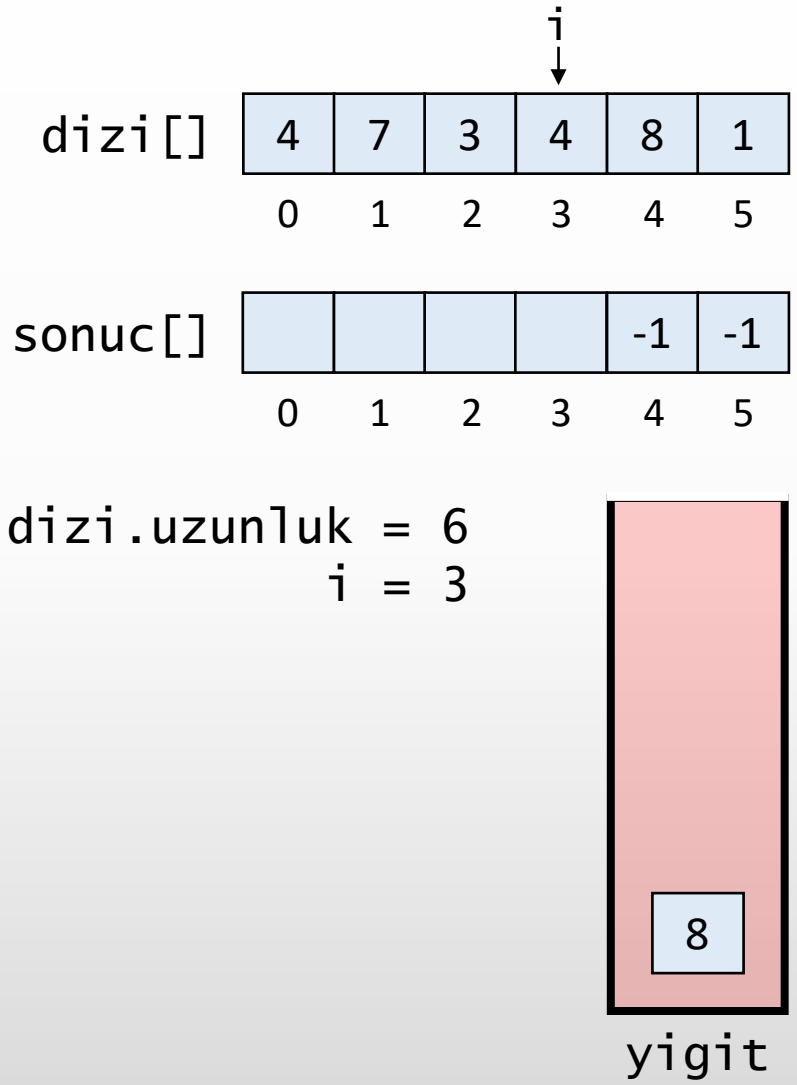

```
sonrakiBuyukEleman(dizi);
```

```
int[] sonrakiBuyukEleman(int[] dizi) {
   int[] sonuc = new int[dizi.length];
   Stack<Integer> yigit = new Stack<>();
   for(int i = dizi.length - 1; i >= 0; i--) {
      if(!yigit.isEmpty()) {
         while(!yigit.isEmpty() &&
               yigit.peek() <= dizi[i]) {
            yigit.pop();
         }
      }
      if(yigit.isEmpty()) {
         sonuc[i] = -1;
      }
      else {
         sonuc[i] = yigit.peek();
      }
      yigit.push(dizi[i]);
   }
   return sonuc;
}
```



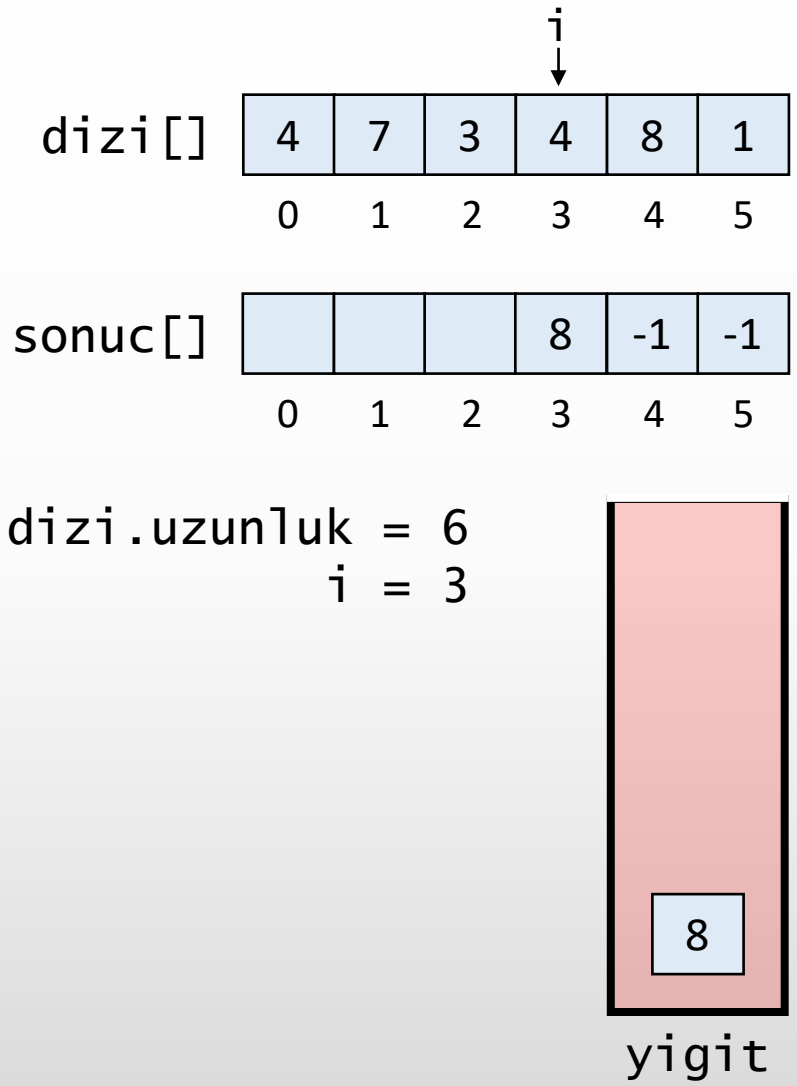
sonrakiBuyukEleman(dizi);

```
int[] sonrakiBuyukEleman(int[] dizi) {
   int[] sonuc = new int[dizi.length];
   Stack<Integer> yigit = new Stack<>();
   for(int i = dizi.length - 1; i >= 0; i--) {
      if(!yigit.isEmpty()) {
         while(!yigit.isEmpty() &&
               yigit.peek() <= dizi[i]) {
            yigit.pop();
         }
      }
      if(yigit.isEmpty()) {
         sonuc[i] = -1;
      }
      else {
         sonuc[i] = yigit.peek();
      }
      yigit.push(dizi[i]);
   }
   return sonuc;
}
```



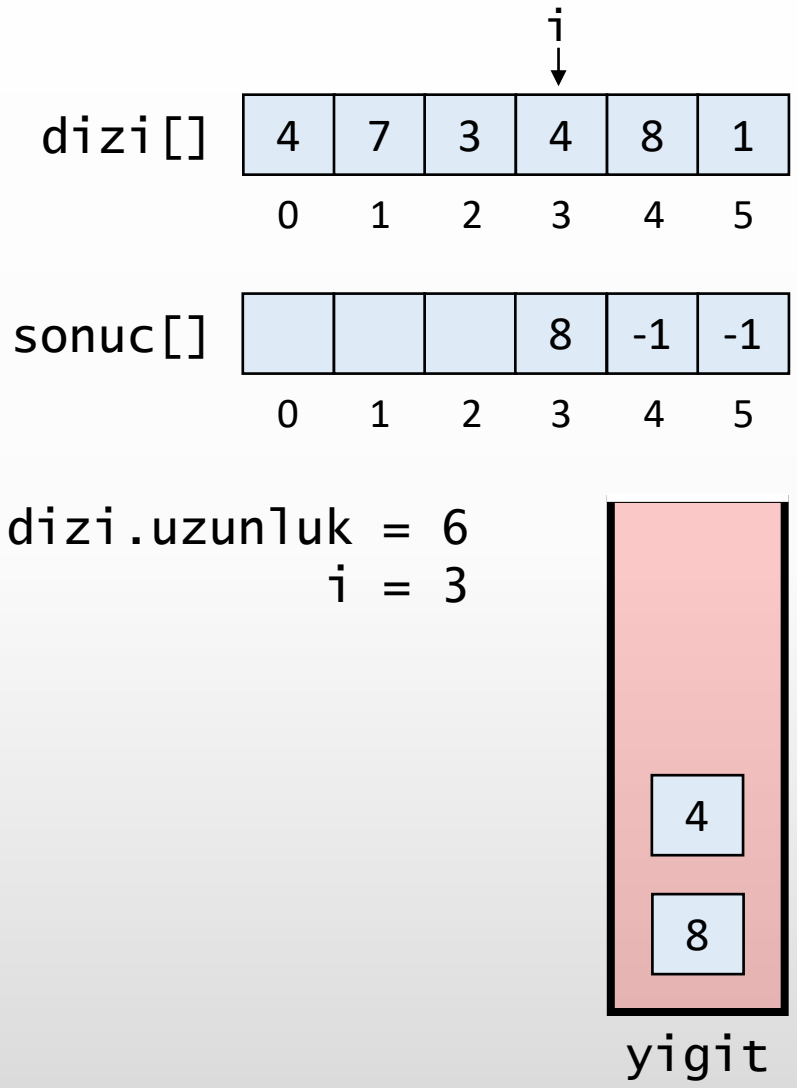

```
sonrakiBuyukEleman(dizi);
```

```java
int[] sonrakiBuyukEleman(int[] dizi) {
   int[] sonuc = new int[dizi.length];
   Stack<Integer> yigit = new Stack<>();
   for(int i = dizi.length - 1; i >= 0; i--) {
      if(!yigit.isEmpty()) {
         while(!yigit.isEmpty() &&
               yigit.peek() <= dizi[i]) {
            yigit.pop();
         }
      }
      if(yigit.isEmpty()) {
         sonuc[i] = -1;
      }
      else {
         sonuc[i] = yigit.peek();
      }
      yigit.push(dizi[i]);
   }
   return sonuc;
}
```



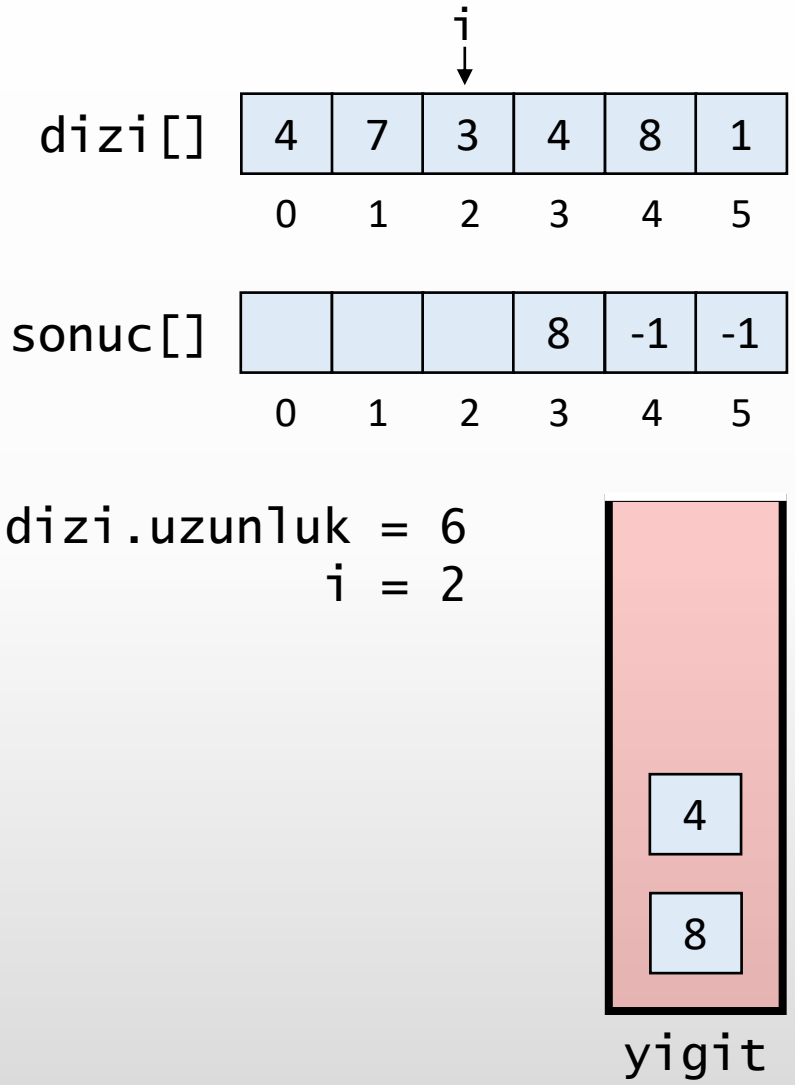
sonrakiBuyukEleman(dizi);

```
int[] sonrakiBuyukEleman(int[] dizi) {
   int[] sonuc = new int[dizi.length];
   Stack<Integer> yigit = new Stack<>();
   for(int i = dizi.length - 1; i >= 0; i--) {
      if(!yigit.isEmpty()) {
         while(!yigit.isEmpty() &&
               yigit.peek() <= dizi[i]) {
            yigit.pop();
         }
      }
      if(yigit.isEmpty()) {
         sonuc[i] = -1;
      }
      else {
         sonuc[i] = yigit.peek();
      }
      yigit.push(dizi[i]);
   }
   return sonuc;
}
```



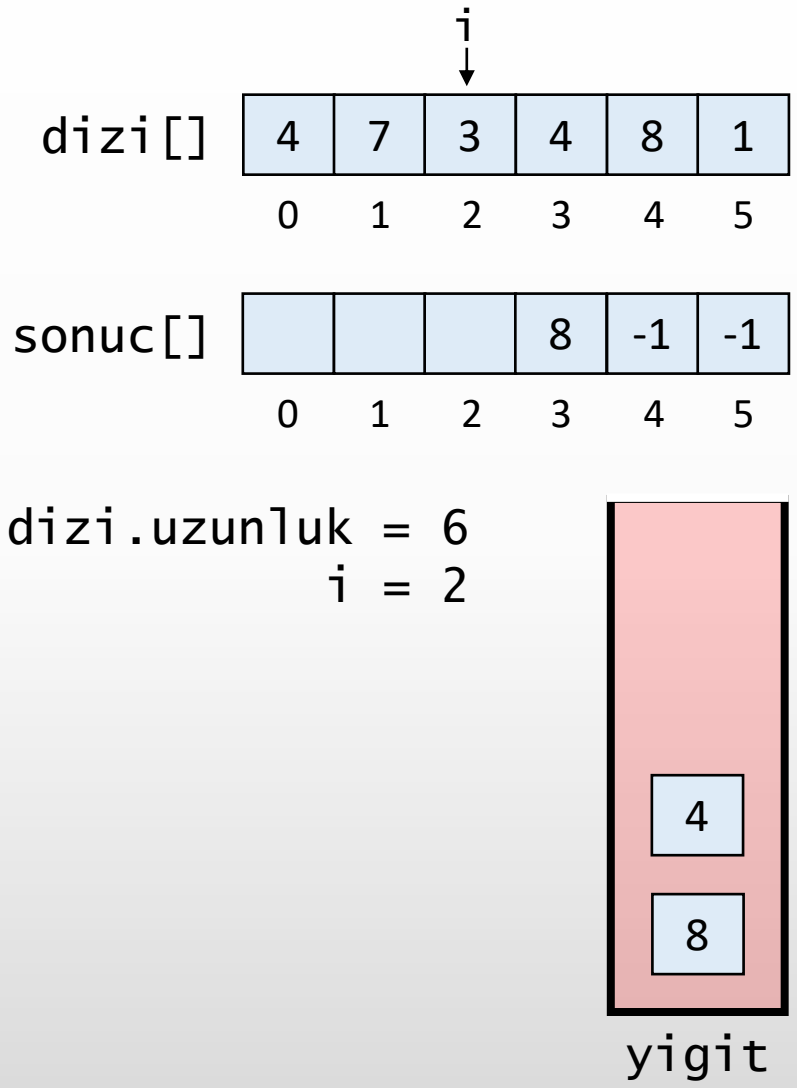
sonrakiBuyukEleman(dizi);

```
int[] sonrakiBuyukEleman(int[] dizi) {
   int[] sonuc = new int[dizi.length];
   Stack<Integer> yigit = new Stack<>();
   for(int i = dizi.length - 1; i >= 0; i--) {
      if(!yigit.isEmpty()) {
         while(!yigit.isEmpty() &&
               yigit.peek() <= dizi[i]) {
            yigit.pop();
         }
      }
      if(yigit.isEmpty()) {
         sonuc[i] = -1;
      }
      else {
         sonuc[i] = yigit.peek();
      }
      yigit.push(dizi[i]);
   }
   return sonuc;
}
```



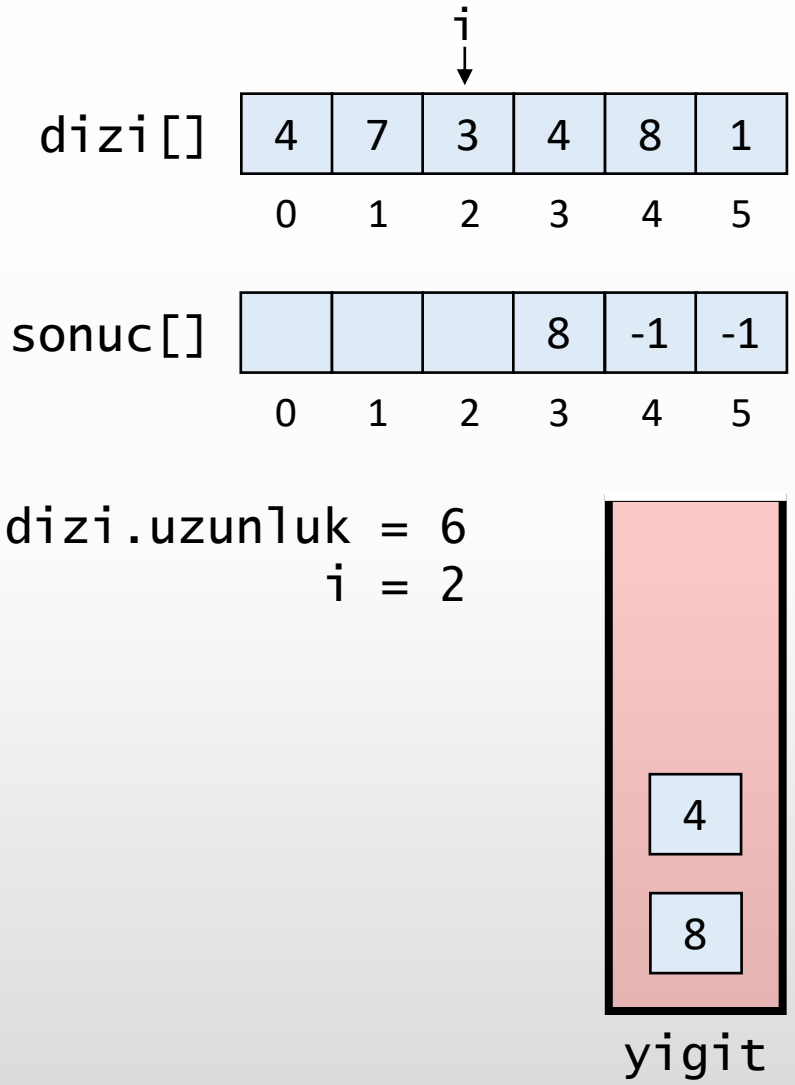

```
sonrakiBuyukEleman(dizi);
```

```
int[] sonrakiBuyukEleman(int[] dizi) {
   int[] sonuc = new int[dizi.length];
   Stack<Integer> yigit = new Stack<>();
   for(int i = dizi.length - 1; i >= 0; i--) {
      if(!yigit.isEmpty()) {
         while(!yigit.isEmpty() &&
               yigit.peek() <= dizi[i]) {
            yigit.pop();
         }
      }
      if(yigit.isEmpty()) {
         sonuc[i] = -1;
      }
      else {
         sonuc[i] = yigit.peek();
      }
      yigit.push(dizi[i]);
   }
   return sonuc;
}
```



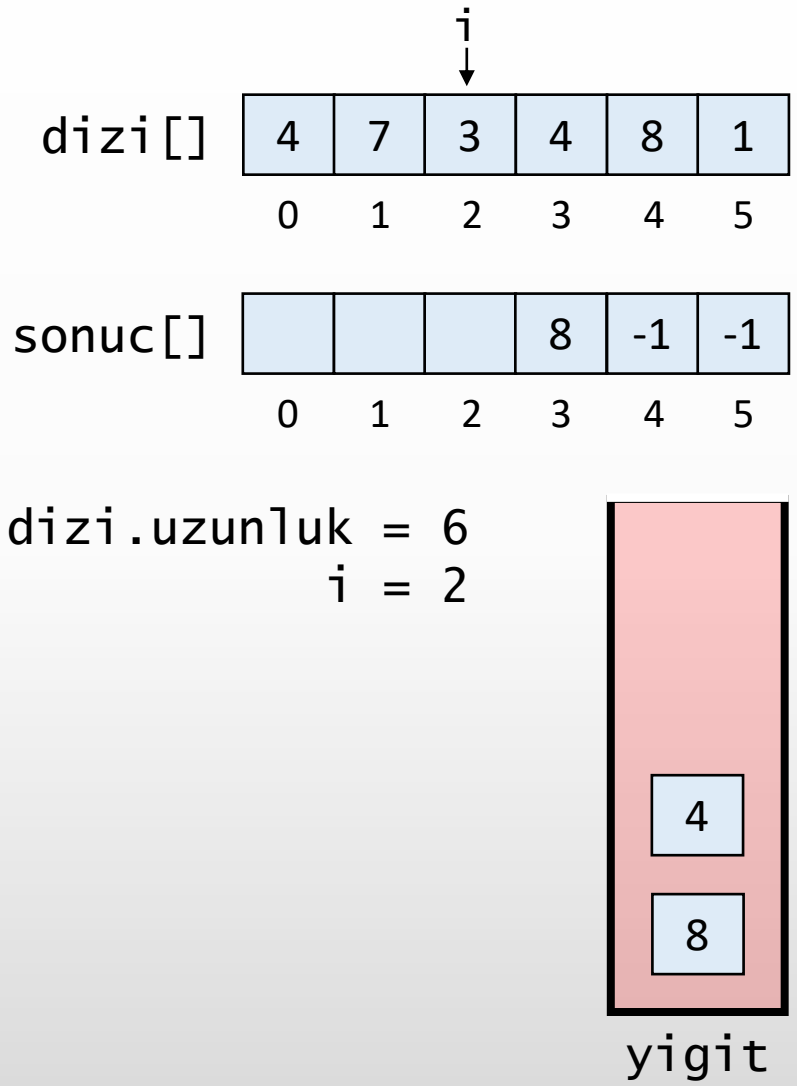

```
sonrakiBuyukEleman(dizi);
```

```java
int[] sonrakiBuyukEleman(int[] dizi) {
   int[] sonuc = new int[dizi.length];
   Stack<Integer> yigit = new Stack<>();
   for(int i = dizi.length - 1; i >= 0; i--) {
      if(!yigit.isEmpty()) {
         while(!yigit.isEmpty() &&
               yigit.peek() <= dizi[i]) {
            yigit.pop();
         }
      }
      if(yigit.isEmpty()) {
         sonuc[i] = -1;
      }
      else {
         sonuc[i] = yigit.peek();
      }
      yigit.push(dizi[i]);
   }
   return sonuc;
}
```



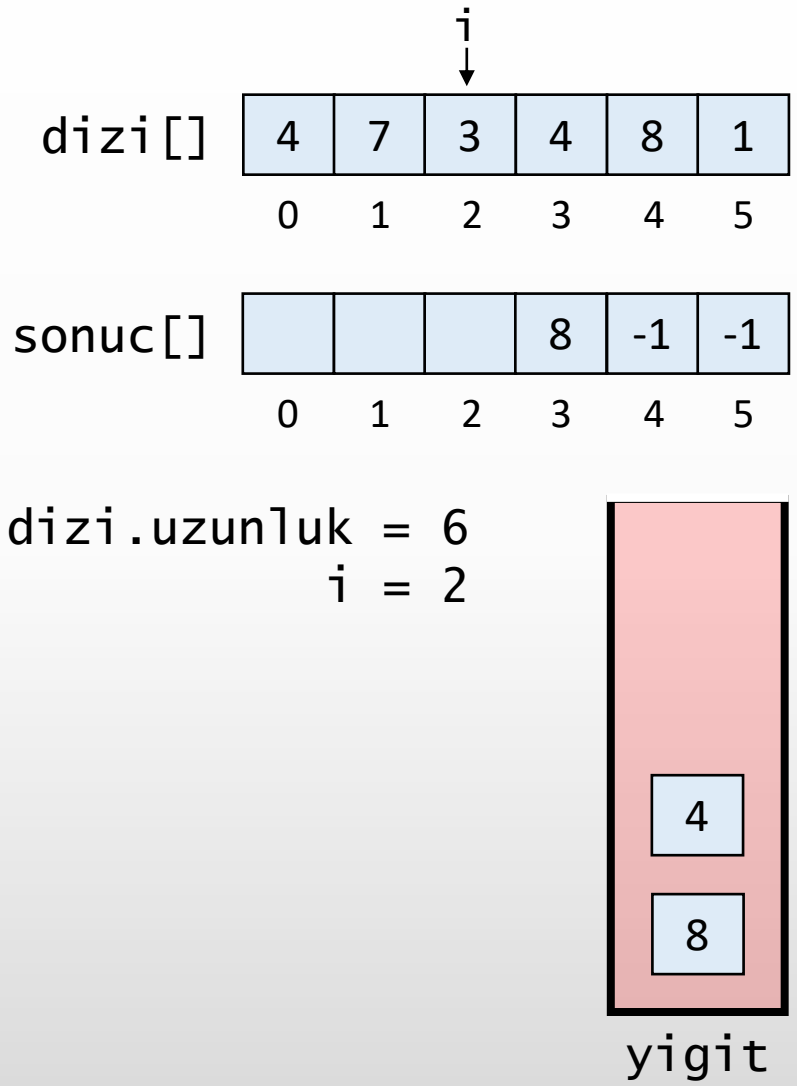

```
sonrakiBuyukEleman(dizi);
```

```
int[] sonrakiBuyukEleman(int[] dizi) {
   int[] sonuc = new int[dizi.length];
   Stack<Integer> yigit = new Stack<>();
   for(int i = dizi.length - 1; i >= 0; i--) {
      if(!yigit.isEmpty()) {
         while(!yigit.isEmpty() &&
               yigit.peek() <= dizi[i]) {
            yigit.pop();
         }
      }
      if(yigit.isEmpty()) {
         sonuc[i] = -1;
      }
      else {
         sonuc[i] = yigit.peek();
      }
      yigit.push(dizi[i]);
   }
   return sonuc;
}
```



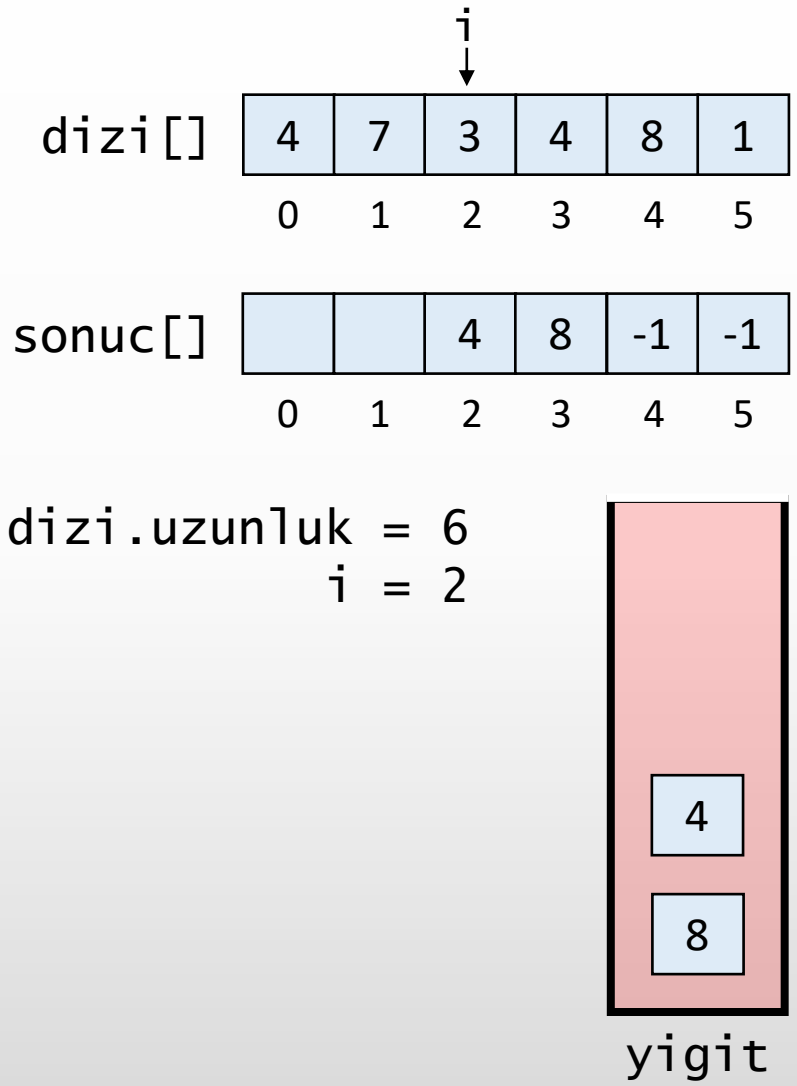
sonrakiBuyukEleman(dizi);

```
int[] sonrakiBuyukEleman(int[] dizi) {
   int[] sonuc = new int[dizi.length];
   Stack<Integer> yigit = new Stack<>();
   for(int i = dizi.length - 1; i >= 0; i--) {
      if(!yigit.isEmpty()) {
         while(!yigit.isEmpty() &&
               yigit.peek() <= dizi[i]) {
            yigit.pop();
         }
      }
      if(yigit.isEmpty()) {
         sonuc[i] = -1;
      }
      else {
         sonuc[i] = yigit.peek();
      }
      yigit.push(dizi[i]);
   }
   return sonuc;
}
```



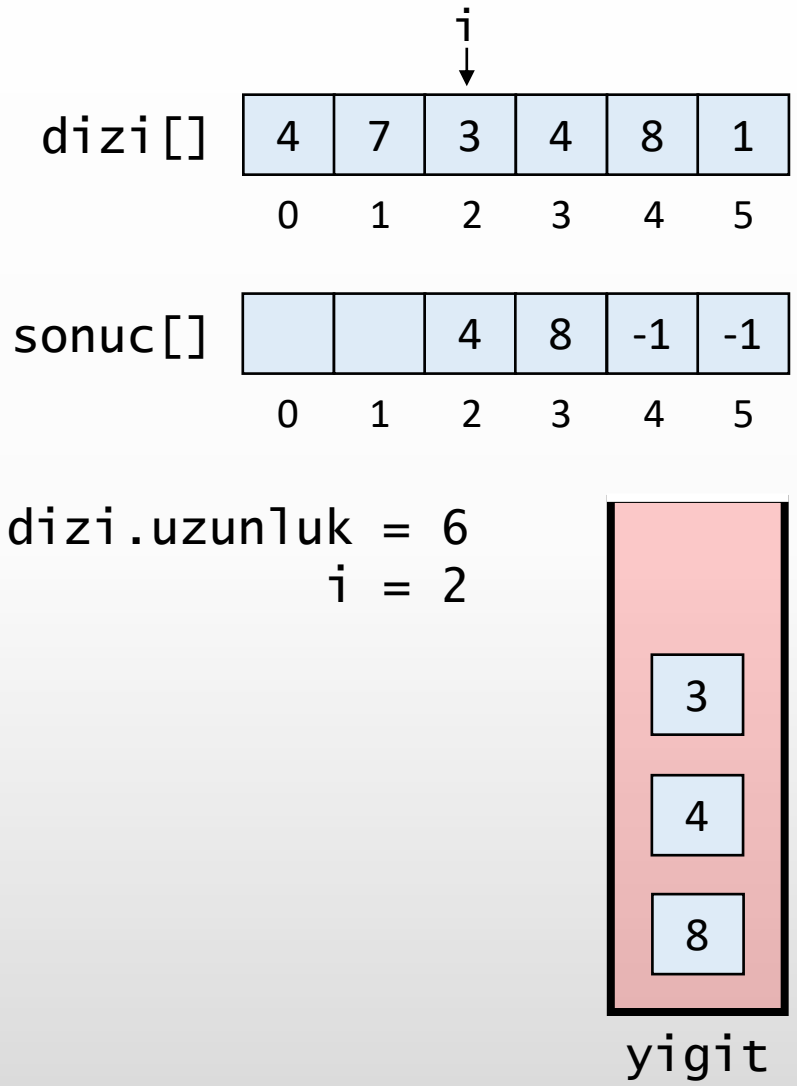
sonrakiBuyukEleman(dizi);

```
int[] sonrakiBuyukEleman(int[] dizi) {
   int[] sonuc = new int[dizi.length];
   Stack<Integer> yigit = new Stack<>();
   for(int i = dizi.length - 1; i >= 0; i--) {
      if(!yigit.isEmpty()) {
         while(!yigit.isEmpty() &&
               yigit.peek() <= dizi[i]) {
            yigit.pop();
         }
      }
      if(yigit.isEmpty()) {
         sonuc[i] = -1;
      }
      else {
         sonuc[i] = yigit.peek();
      }
      yigit.push(dizi[i]);
   }
   return sonuc;
}
```



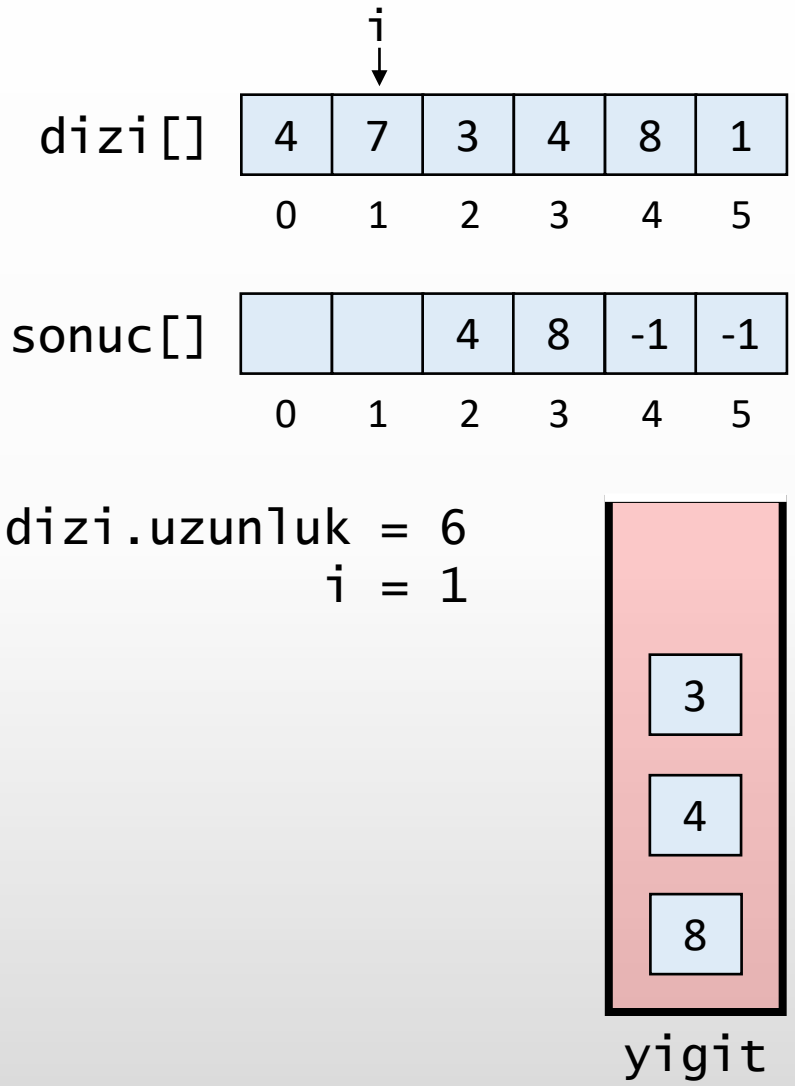

```
sonrakiBuyukEleman(dizi);
```

```
int[] sonrakiBuyukEleman(int[] dizi) {
   int[] sonuc = new int[dizi.length];
   Stack<Integer> yigit = new Stack<>();
   for(int i = dizi.length - 1; i >= 0; i--) {
      if(!yigit.isEmpty()) {
         while(!yigit.isEmpty() &&
               yigit.peek() <= dizi[i]) {
            yigit.pop();
         }
      }
      if(yigit.isEmpty()) {
         sonuc[i] = -1;
      }
      else {
         sonuc[i] = yigit.peek();
      }
      yigit.push(dizi[i]);
   }
   return sonuc;
}
```



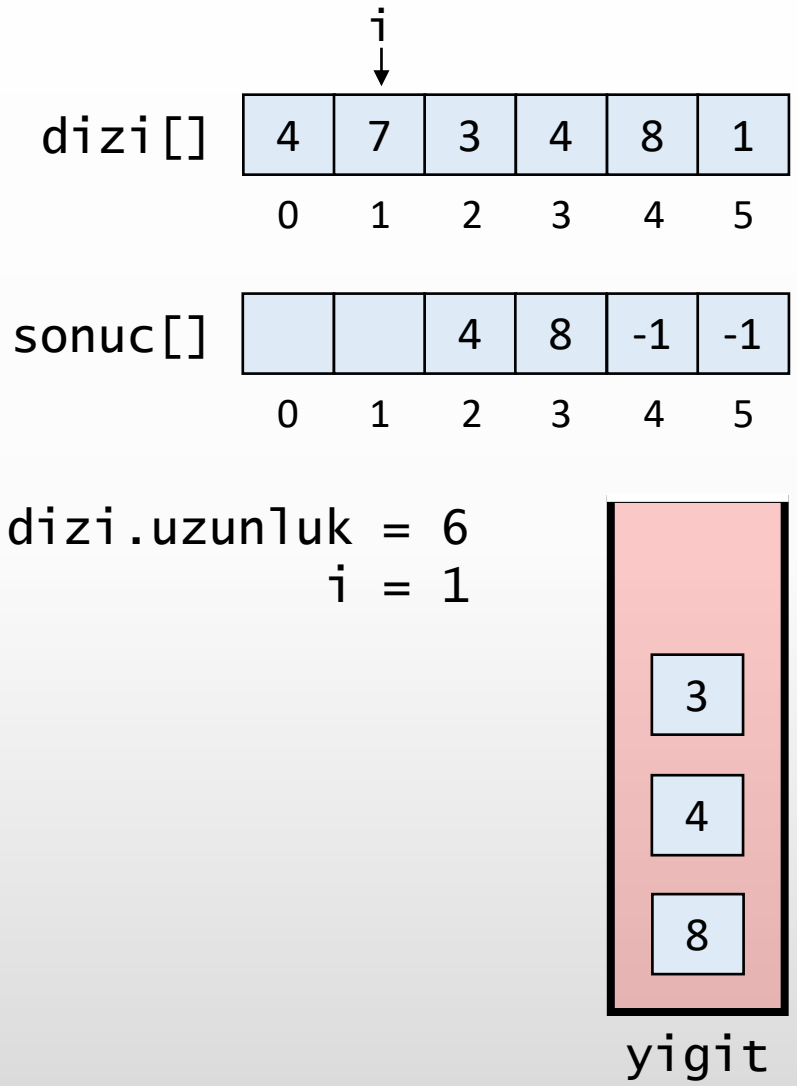
sonrakiBuyukEleman(dizi);

```
int[] sonrakiBuyukEleman(int[] dizi) {
   int[] sonuc = new int[dizi.length];
   Stack<Integer> yigit = new Stack<>();
   for(int i = dizi.length - 1; i >= 0; i--) {
      if(!yigit.isEmpty()) {
         while(!yigit.isEmpty() &&
               yigit.peek() <= dizi[i]) {
            yigit.pop();
         }
      }
      if(yigit.isEmpty()) {
         sonuc[i] = -1;
      }
      else {
         sonuc[i] = yigit.peek();
      }
      yigit.push(dizi[i]);
   }
   return sonuc;
}
```



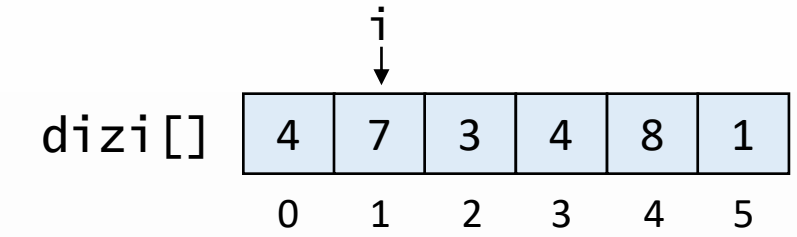

```
sonrakiBuyukEleman(dizi);
```

```java
int[] sonrakiBuyukEleman(int[] dizi) {
   int[] sonuc = new int[dizi.length];
   Stack<Integer> yigit = new Stack<>();
   for(int i = dizi.length - 1; i >= 0; i--) {
      if(!yigit.isEmpty()) {
         while(!yigit.isEmpty() &&
               yigit.peek() <= dizi[i]) {
            yigit.pop();
         }
      }
      if(yigit.isEmpty()) {
         sonuc[i] = -1;
      }
      else {
         sonuc[i] = yigit.peek();
      }
      yigit.push(dizi[i]);
   }
   return sonuc;
}
```



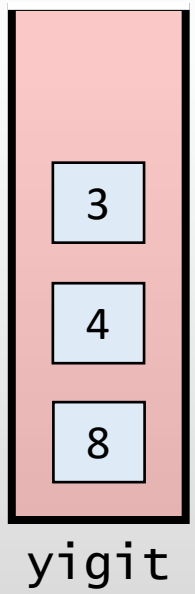

```
sonrakiBuyukEleman(dizi);
```

```java
int[] sonrakiBuyukEleman(int[] dizi) {
   int[] sonuc = new int[dizi.length];
   Stack<Integer> yigit = new Stack<>();
   for(int i = dizi.length - 1; i >= 0; i--) {
      if(!yigit.isEmpty()) {
         while(!yigit.isEmpty() &&
               yigit.peek() <= dizi[i]) {
            yigit.pop();
         }
      }
      if(yigit.isEmpty()) {
         sonuc[i] = -1;
      }
      else {
         sonuc[i] = yigit.peek();
      }
      yigit.push(dizi[i]);
   }
   return sonuc;
}
```



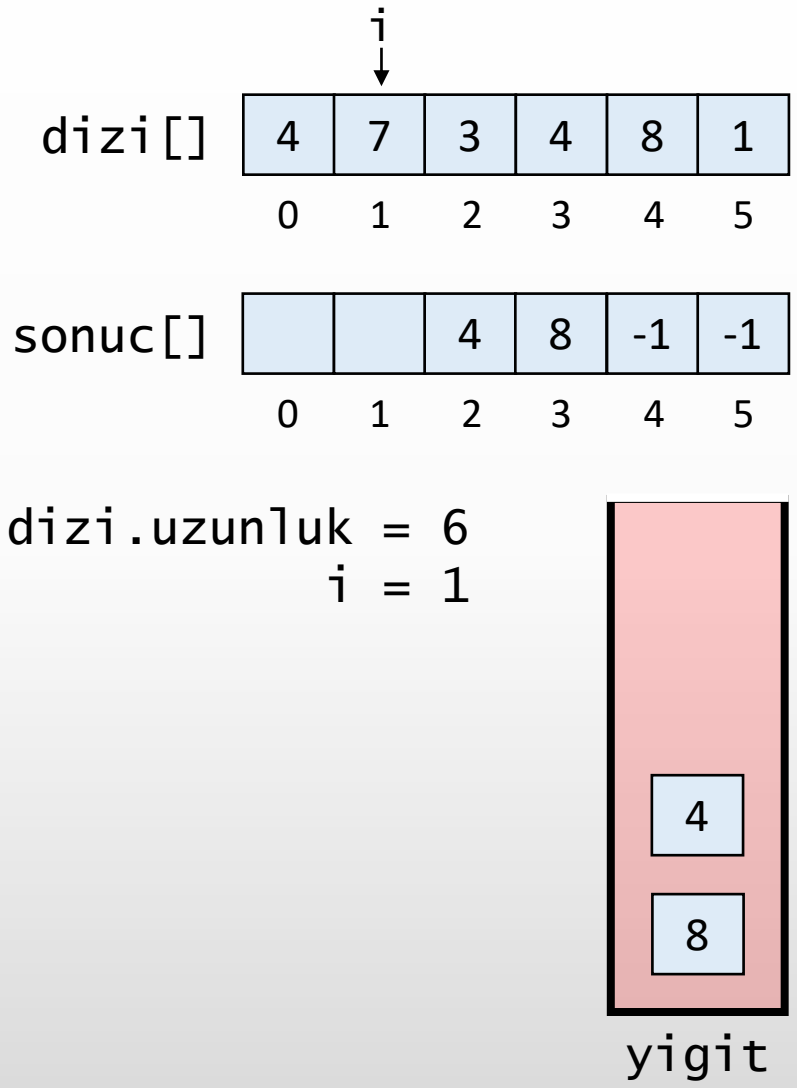
dizi.uzunluk = 6

i = 1



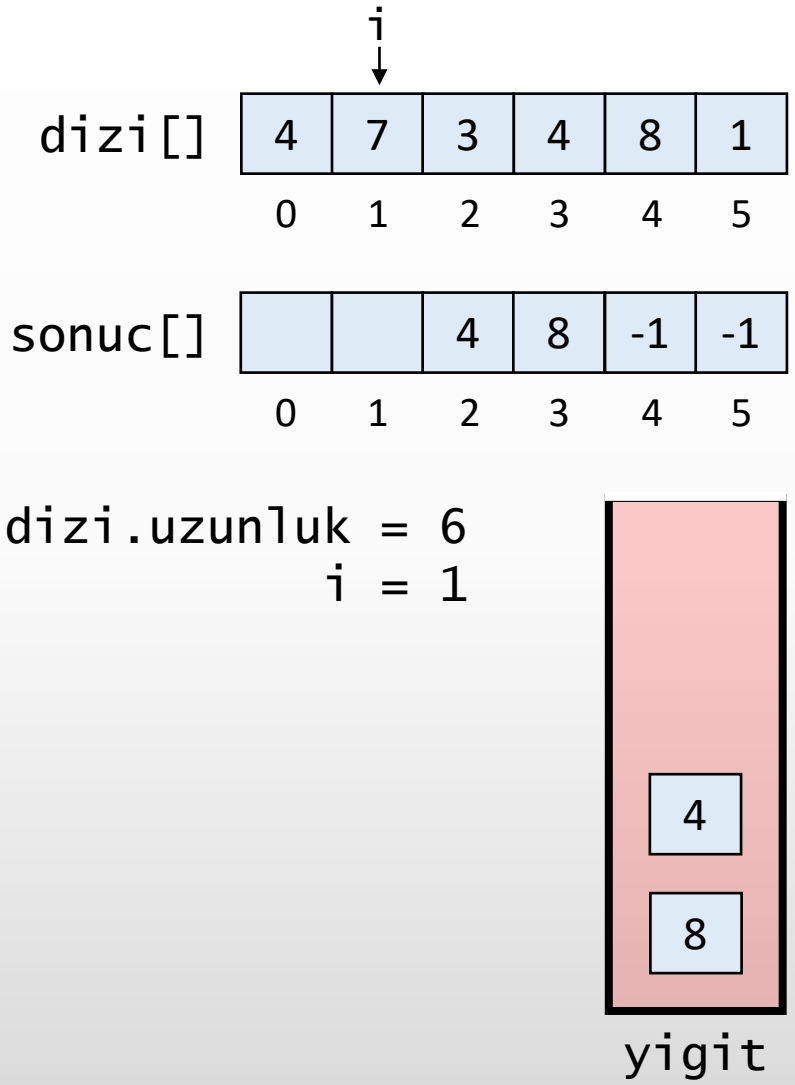
sonrakiBuyukEleman(dizi);

```
int[] sonrakiBuyukEleman(int[] dizi) {
   int[] sonuc = new int[dizi.length];
   Stack<Integer> yigit = new Stack<>();
   for(int i = dizi.length - 1; i >= 0; i--) {
      if(!yigit.isEmpty()) {
         while(!yigit.isEmpty() &&
               yigit.peek() <= dizi[i]) {
            yigit.pop();
         }
      }
      if(yigit.isEmpty()) {
         sonuc[i] = -1;
      }
      else {
         sonuc[i] = yigit.peek();
      }
      yigit.push(dizi[i]);
   }
   return sonuc;
}
```



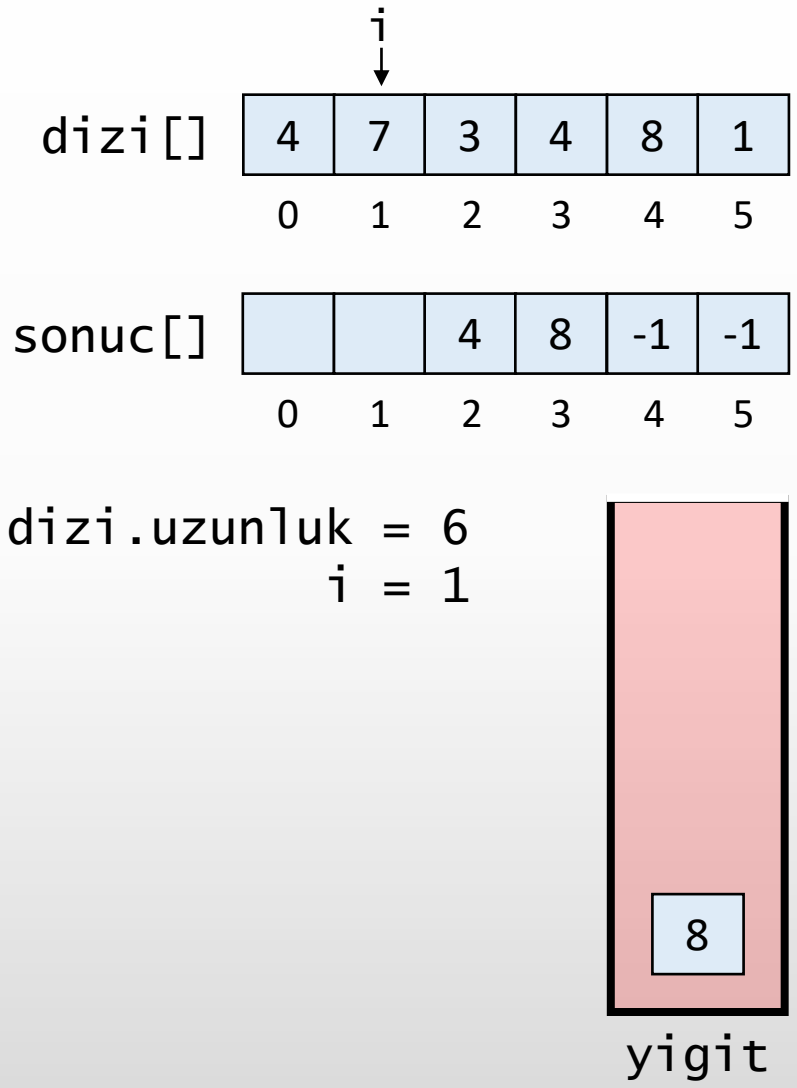

```
sonrakiBuyukEleman(dizi);
```

```java
int[] sonrakiBuyukEleman(int[] dizi) {
   int[] sonuc = new int[dizi.length];
   Stack<Integer> yigit = new Stack<>();
   for(int i = dizi.length - 1; i >= 0; i--) {
      if(!yigit.isEmpty()) {
         while(!yigit.isEmpty() &&
               yigit.peek() <= dizi[i]) {
            yigit.pop();
         }
      }
      if(yigit.isEmpty()) {
         sonuc[i] = -1;
      }
      else {
         sonuc[i] = yigit.peek();
      }
      yigit.push(dizi[i]);
   }
   return sonuc;
}
```



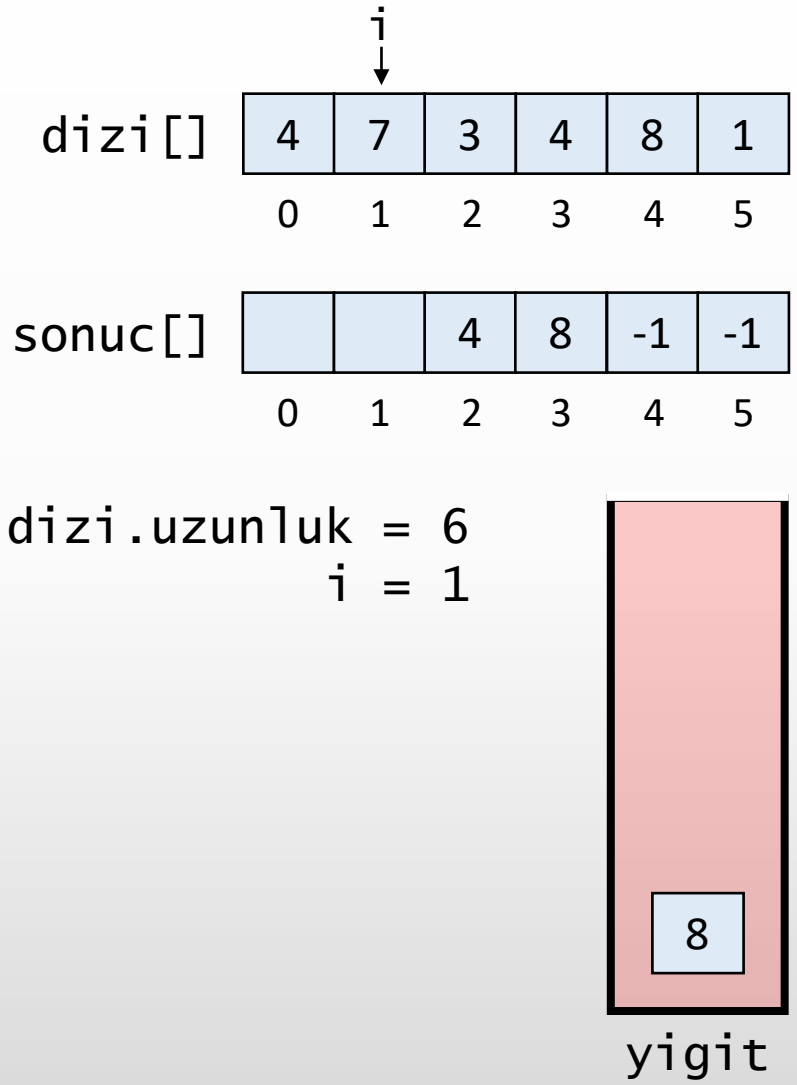
sonrakiBuyukEleman(dizi);

```
int[] sonrakiBuyukEleman(int[] dizi) {
   int[] sonuc = new int[dizi.length];
   Stack<Integer> yigit = new Stack<>();
   for(int i = dizi.length - 1; i >= 0; i--) {
      if(!yigit.isEmpty()) {
         while(!yigit.isEmpty() &&
               yigit.peek() <= dizi[i]) {
            yigit.pop();
         }
      }
      if(yigit.isEmpty()) {
         sonuc[i] = -1;
      }
      else {
         sonuc[i] = yigit.peek();
      }
      yigit.push(dizi[i]);
   }
   return sonuc;
}
```



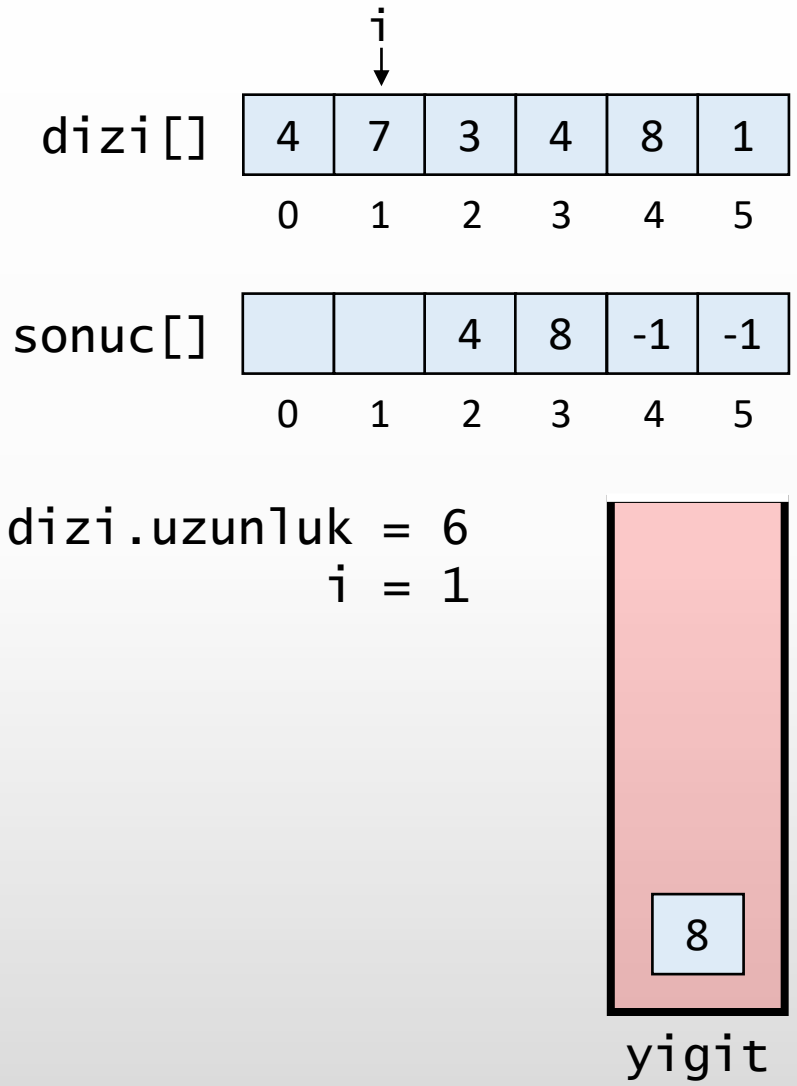

```
sonrakiBuyukEleman(dizi);
```

```
int[] sonrakiBuyukEleman(int[] dizi) {
   int[] sonuc = new int[dizi.length];
   Stack<Integer> yigit = new Stack<>();
   for(int i = dizi.length - 1; i >= 0; i--) {
      if(!yigit.isEmpty()) {
         while(!yigit.isEmpty() &&
               yigit.peek() <= dizi[i]) {
            yigit.pop();
         }
      }
      if(yigit.isEmpty()) {
         sonuc[i] = -1;
      }
      else {
         sonuc[i] = yigit.peek();
      }
      yigit.push(dizi[i]);
   }
   return sonuc;
}
```



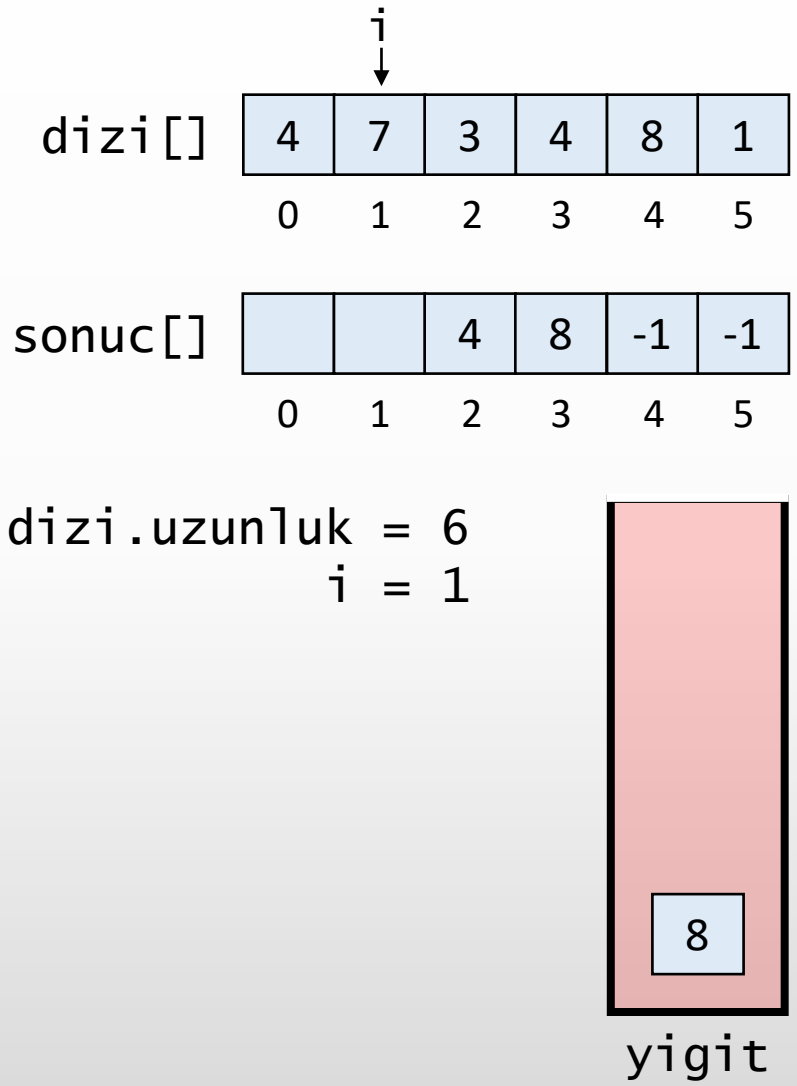
sonrakiBuyukEleman(dizi);

```
int[] sonrakiBuyukEleman(int[] dizi) {
   int[] sonuc = new int[dizi.length];
   Stack<Integer> yigit = new Stack<>();
   for(int i = dizi.length - 1; i >= 0; i--) {
      if(!yigit.isEmpty()) {
         while(!yigit.isEmpty() &&
               yigit.peek() <= dizi[i]) {
            yigit.pop();
         }
      }
      if(yigit.isEmpty()) {
         sonuc[i] = -1;
      }
      else {
         sonuc[i] = yigit.peek();
      }
      yigit.push(dizi[i]);
   }
   return sonuc;
}
```



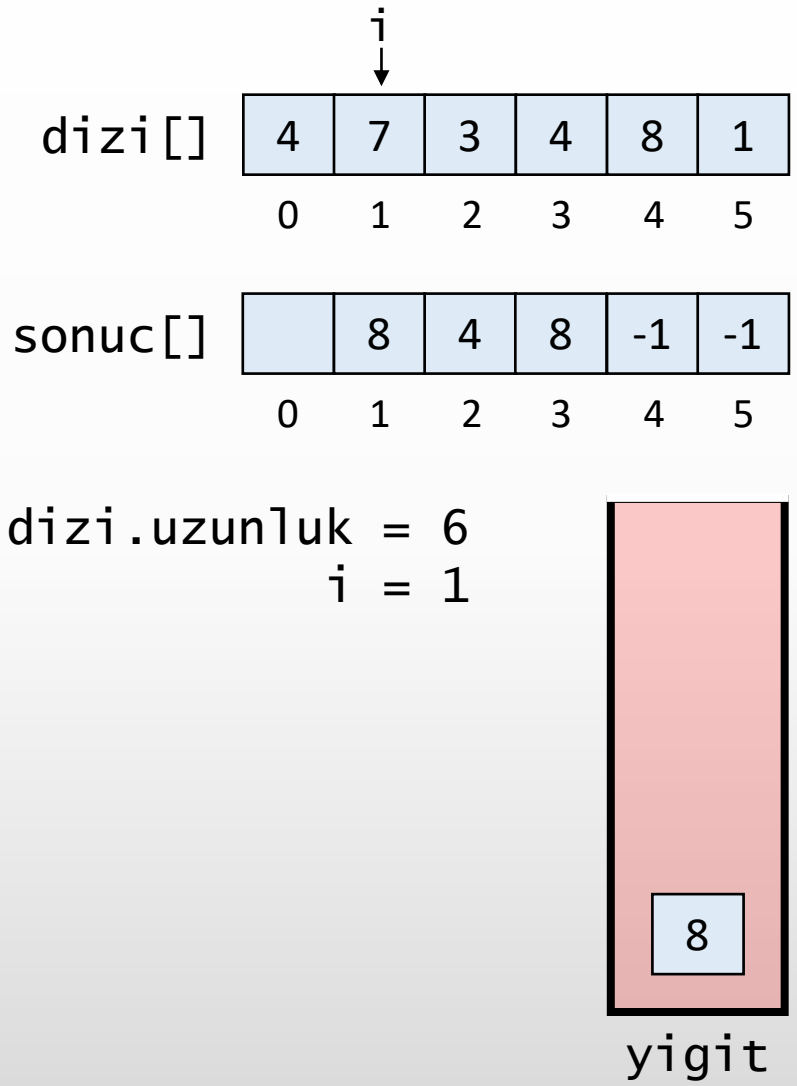

```
sonrakiBuyukEleman(dizi);
```

```
int[] sonrakiBuyukEleman(int[] dizi) {
   int[] sonuc = new int[dizi.length];
   Stack<Integer> yigit = new Stack<>();
   for(int i = dizi.length - 1; i >= 0; i--) {
      if(!yigit.isEmpty()) {
         while(!yigit.isEmpty() &&
               yigit.peek() <= dizi[i]) {
            yigit.pop();
         }
      }
      if(yigit.isEmpty()) {
         sonuc[i] = -1;
      }
      else {
         sonuc[i] = yigit.peek();
      }
      yigit.push(dizi[i]);
   }
   return sonuc;
}
```



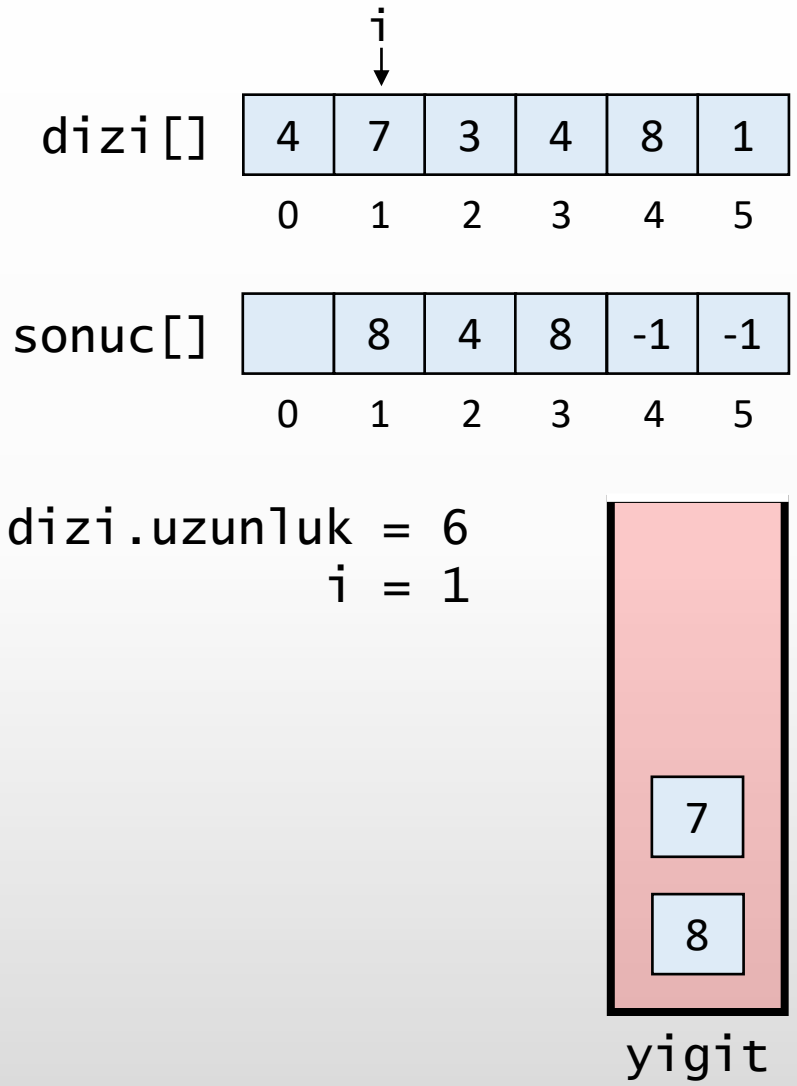

```
sonrakiBuyukEleman(dizi);
```

```
int[] sonrakiBuyukEleman(int[] dizi) {
   int[] sonuc = new int[dizi.length];
   Stack<Integer> yigit = new Stack<>();
   for(int i = dizi.length - 1; i >= 0; i--) {
      if(!yigit.isEmpty()) {
         while(!yigit.isEmpty() &&
               yigit.peek() <= dizi[i]) {
            yigit.pop();
         }
      }
      if(yigit.isEmpty()) {
         sonuc[i] = -1;
      }
      else {
         sonuc[i] = yigit.peek();
      }
      yigit.push(dizi[i]);
   }
   return sonuc;
}
```



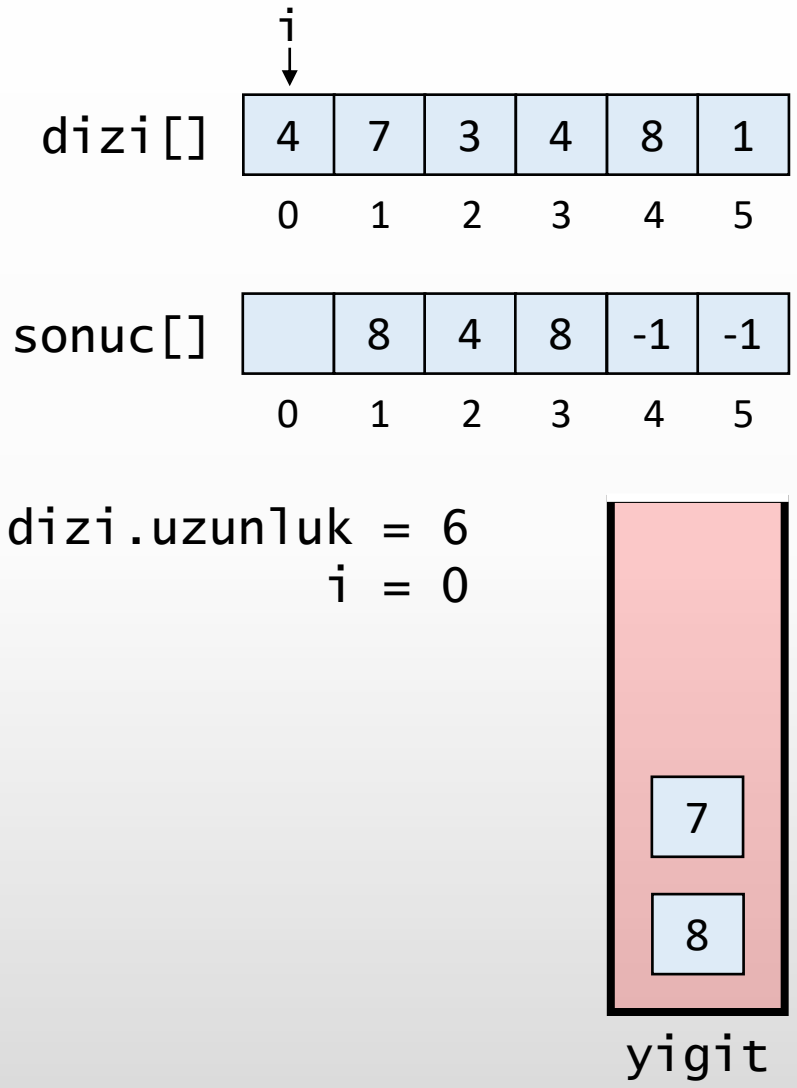

```
sonrakiBuyukEleman(dizi);
```

```
int[] sonrakiBuyukEleman(int[] dizi) {
   int[] sonuc = new int[dizi.length];
   Stack<Integer> yigit = new Stack<>();
   for(int i = dizi.length - 1; i >= 0; i--) {
      if(!yigit.isEmpty()) {
         while(!yigit.isEmpty() &&
               yigit.peek() <= dizi[i]) {
            yigit.pop();
         }
      }
      if(yigit.isEmpty()) {
         sonuc[i] = -1;
      }
      else {
         sonuc[i] = yigit.peek();
      }
      yigit.push(dizi[i]);
   }
   return sonuc;
}
```



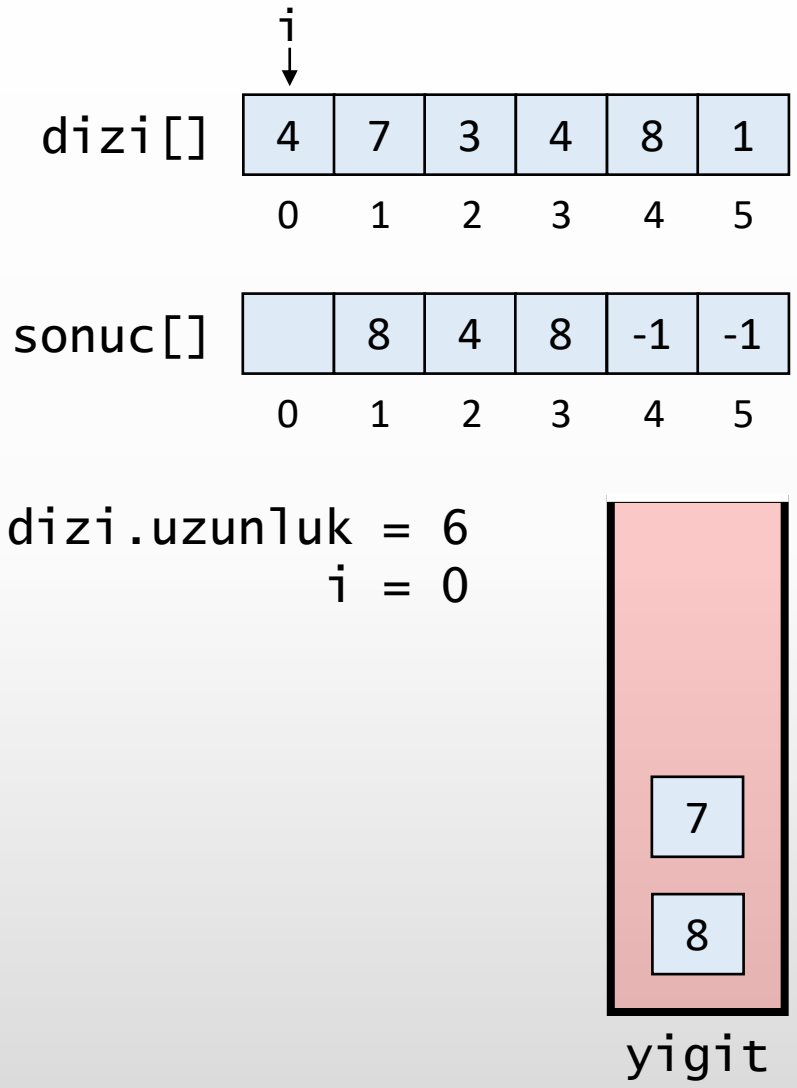
sonrakiBuyukEleman(dizi);

```
int[] sonrakiBuyukEleman(int[] dizi) {
   int[] sonuc = new int[dizi.length];
   Stack<Integer> yigit = new Stack<>();
   for(int i = dizi.length - 1; i >= 0; i--) {
      if(!yigit.isEmpty()) {
         while(!yigit.isEmpty() &&
               yigit.peek() <= dizi[i]) {
            yigit.pop();
         }
      }
      if(yigit.isEmpty()) {
         sonuc[i] = -1;
      }
      else {
         sonuc[i] = yigit.peek();
      }
      yigit.push(dizi[i]);
   }
   return sonuc;
}
```



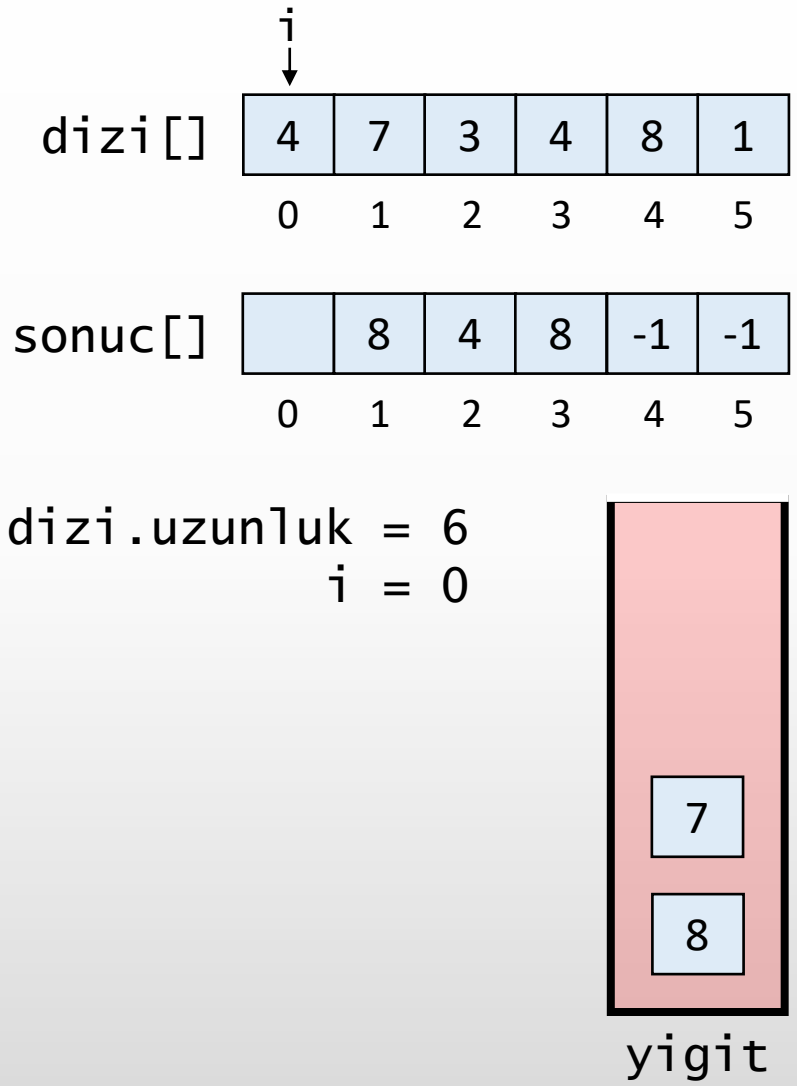

```
sonrakiBuyukEleman(dizi);
```

```
int[] sonrakiBuyukEleman(int[] dizi) {
   int[] sonuc = new int[dizi.length];
   Stack<Integer> yigit = new Stack<>();
   for(int i = dizi.length - 1; i >= 0; i--) {
      if(!yigit.isEmpty()) {
         while(!yigit.isEmpty() &&
               yigit.peek() <= dizi[i]) {
            yigit.pop();
         }
      }
      if(yigit.isEmpty()) {
         sonuc[i] = -1;
      }
      else {
         sonuc[i] = yigit.peek();
      }
      yigit.push(dizi[i]);
   }
   return sonuc;
}
```



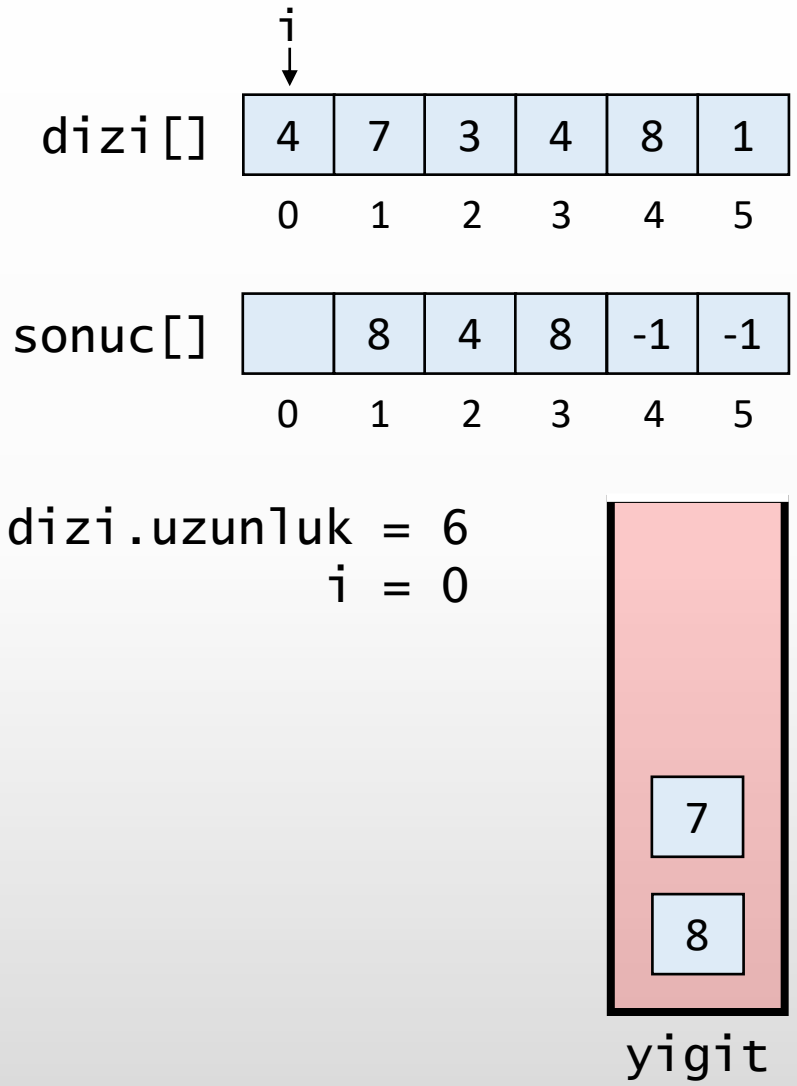

```
sonrakiBuyukEleman(dizi);
```

```
int[] sonrakiBuyukEleman(int[] dizi) {
   int[] sonuc = new int[dizi.length];
   Stack<Integer> yigit = new Stack<>();
   for(int i = dizi.length - 1; i >= 0; i--) {
      if(!yigit.isEmpty()) {
         while(!yigit.isEmpty() &&
               yigit.peek() <= dizi[i]) {
            yigit.pop();
         }
      }
      if(yigit.isEmpty()) {
         sonuc[i] = -1;
      }
      else {
         sonuc[i] = yigit.peek();
      }
      yigit.push(dizi[i]);
   }
   return sonuc;
}
```



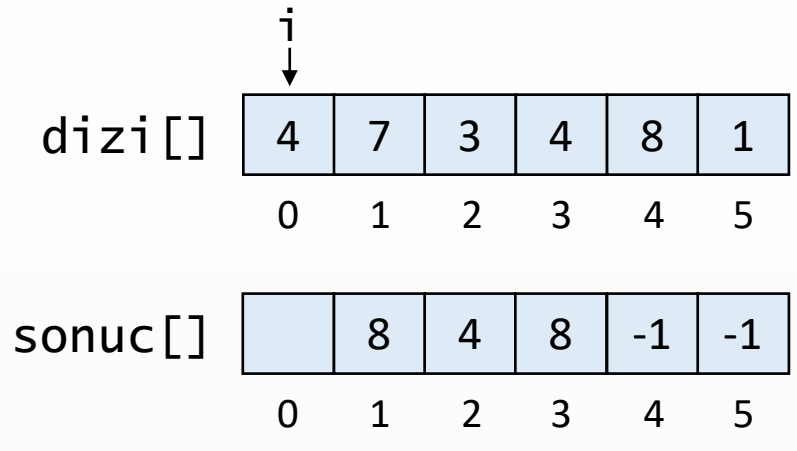

```
sonrakiBuyukEleman(dizi);
```

```
int[] sonrakiBuyukEleman(int[] dizi) {
   int[] sonuc = new int[dizi.length];
   Stack<Integer> yigit = new Stack<>();
   for(int i = dizi.length - 1; i >= 0; i--) {
      if(!yigit.isEmpty()) {
         while(!yigit.isEmpty() &&
               yigit.peek() <= dizi[i]) {
            yigit.pop();
         }
      }
      if(yigit.isEmpty()) {
         sonuc[i] = -1;
      }
      else {
         sonuc[i] = yigit.peek();
      }
      yigit.push(dizi[i]);
   }
   return sonuc;
}
```



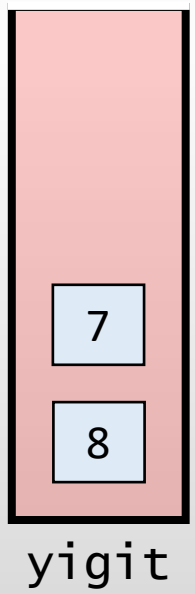

```
sonrakiBuyukEleman(dizi);
```

```java
int[] sonrakiBuyukEleman(int[] dizi) {
   int[] sonuc = new int[dizi.length];
   Stack<Integer> yigit = new Stack<>();
   for(int i = dizi.length - 1; i >= 0; i--) {
      if(!yigit.isEmpty()) {
         while(!yigit.isEmpty() &&
               yigit.peek() <= dizi[i]) {
            yigit.pop();
         }
      }
      if(yigit.isEmpty()) {
         sonuc[i] = -1;
      }
      else {
         sonuc[i] = yigit.peek();
      }
      yigit.push(dizi[i]);
   }
   return sonuc;
}
```



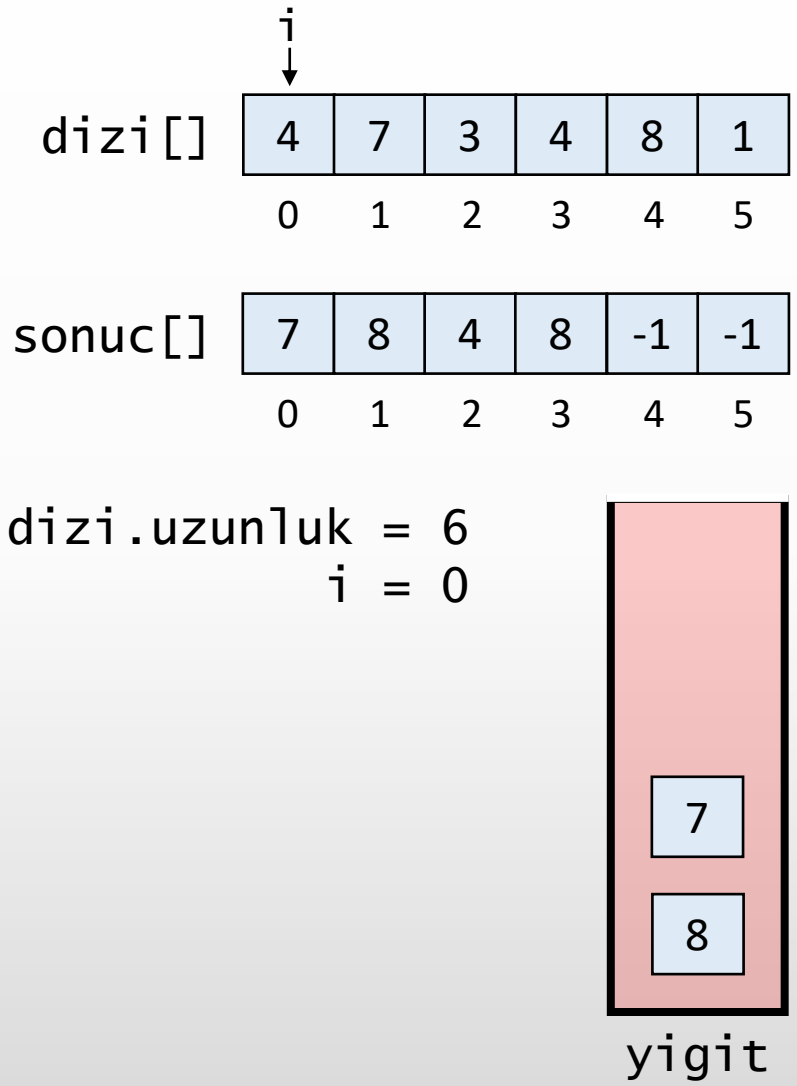
dizi.uzunluk = 6

i = 0



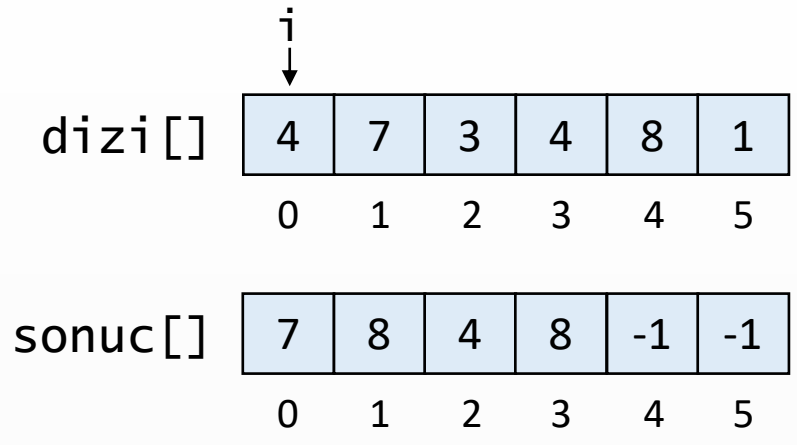
sonrakiBuyukEleman(dizi);

```
int[] sonrakiBuyukEleman(int[] dizi) {
   int[] sonuc = new int[dizi.length];
   Stack<Integer> yigit = new Stack<>();
   for(int i = dizi.length - 1; i >= 0; i--) {
      if(!yigit.isEmpty()) {
         while(!yigit.isEmpty() &&
               yigit.peek() <= dizi[i]) {
            yigit.pop();
         }
      }
      if(yigit.isEmpty()) {
         sonuc[i] = -1;
      }
      else {
         sonuc[i] = yigit.peek();
      }
      yigit.push(dizi[i]);
   }
   return sonuc;
}
```

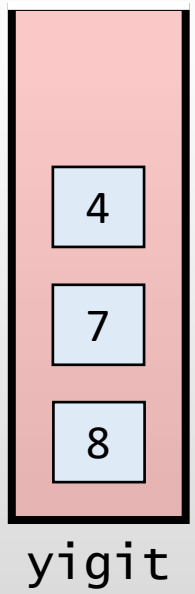



```
sonrakiBuyukEleman(dizi);
```

```
int[] sonrakiBuyukEleman(int[] dizi) {
   int[] sonuc = new int[dizi.length];
   Stack<Integer> yigit = new Stack<>();
   for(int i = dizi.length - 1; i >= 0; i--) {
      if(!yigit.isEmpty()) {
         while(!yigit.isEmpty() &&
               yigit.peek() <= dizi[i]) {
            yigit.pop();
         }
      }
      if(yigit.isEmpty()) {
         sonuc[i] = -1;
      }
      else {
         sonuc[i] = yigit.peek();
      }
      yigit.push(dizi[i]);
   }
   return sonuc;
}
```



dizi.uzunluk = 6
i = 0



sonrakiBuyukEleman(dizi);

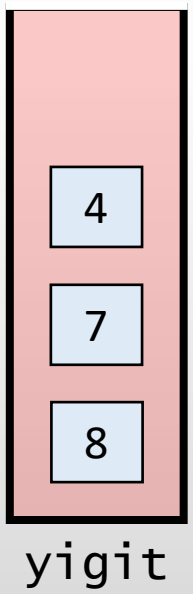
```
int[] sonrakiBuyukEleman(int[] dizi) {
   int[] sonuc = new int[dizi.length];
   Stack<Integer> yigit = new Stack<>();
   for(int i = dizi.length - 1; i >= 0; i--) {
      if(!yigit.isEmpty()) {
         while(!yigit.isEmpty() &&
               yigit.peek() <= dizi[i]) {
            yigit.pop();
         }
      }
      if(yigit.isEmpty()) {
         sonuc[i] = -1;
      }
      else {
         sonuc[i] = yigit.peek();
      }
      yigit.push(dizi[i]);
   }
   return sonuc;
}
```

dizi[]	4	7	3	4	8	1
	0	1	2	3	4	5

sonuc[]	7	8	4	8	-1	-1
	0	1	2	3	4	5

dizi.uzunluk = 6

i = -1




```
sonrakiBuyukEleman(dizi);
```

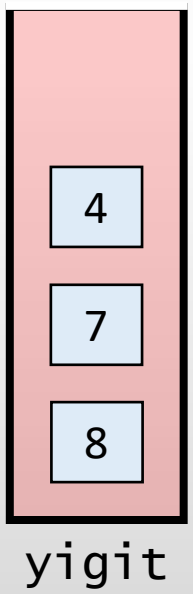
```
int[] sonrakiBuyukEleman(int[] dizi) {
   int[] sonuc = new int[dizi.length];
   Stack<Integer> yigit = new Stack<>();
   for(int i = dizi.length - 1; i >= 0; i--) {
      if(!yigit.isEmpty()) {
         while(!yigit.isEmpty() &&
               yigit.peek() <= dizi[i]) {
            yigit.pop();
         }
      }
      if(yigit.isEmpty()) {
         sonuc[i] = -1;
      }
      else {
         sonuc[i] = yigit.peek();
      }
      yigit.push(dizi[i]);
   }
   return sonuc;
}
```

dizi[]	4	7	3	4	8	1
	0	1	2	3	4	5

sonuc[]	7	8	4	8	-1	-1
	0	1	2	3	4	5

dizi.uzunluk = 6

i = -1




```
sonrakiBuyukEleman(dizi);
```

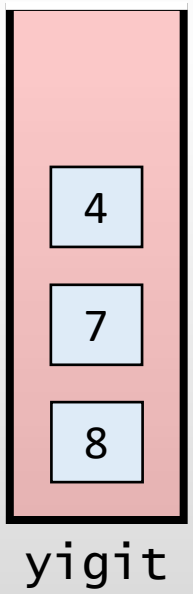
```
int[] sonrakiBuyukEleman(int[] dizi) {
   int[] sonuc = new int[dizi.length];
   Stack<Integer> yigit = new Stack<>();
   for(int i = dizi.length - 1; i >= 0; i--) {
      if(!yigit.isEmpty()) {
         while(!yigit.isEmpty() &&
               yigit.peek() <= dizi[i]) {
            yigit.pop();
         }
      }
      if(yigit.isEmpty()) {
         sonuc[i] = -1;
      }
      else {
         sonuc[i] = yigit.peek();
      }
      yigit.push(dizi[i]);
   }
   return sonuc;
}
```

dizi[]	4	7	3	4	8	1
	0	1	2	3	4	5

sonuc[]	7	8	4	8	-1	-1
	0	1	2	3	4	5

dizi.uzunluk = 6

i = -1



sonrakiBuyukEleman(dizi);

```
int[] sonrakiBuyukEleman(int[] dizi) {
   int[] sonuc = new int[dizi.length];
   Stack<Integer> yigit = new Stack<>();
   for(int i = dizi.length - 1; i >= 0; i--) {
      if(!yigit.isEmpty()) {
         while(!yigit.isEmpty() &&
               yigit.peek() <= dizi[i]) {
            yigit.pop();
         }
      }
      if(yigit.isEmpty()) {
         sonuc[i] = -1;
      }
      else {
         sonuc[i] = yigit.peek();
      }
      yigit.push(dizi[i]);
   }
   return sonuc;
}
```

Örnek

Parantez Kontrolü

- S, '(', ')', '{', '}', '[' ve ']' karakterlerinden oluşan bir karakter dizisidir.
- S dizisi; (1) aç parantezler aynı tip parantez ile kapatılmış ise ve (2) parantezler doğru sırada kapatılmış ise geçerlidir.
- Problemi çözmek için, eşleşmemiş açık parantezler takip edilmelidir.
- S dizisi baştan sona gezilir.
  - Karakter açık parantez ise, yığına eklenir.
  - Karakter kapalı parantez ise, yığından çıkarılır.
  - Karakter ile yığından çıkan aynı tipte değilse S dizisi **geçersizdir**.
- S dizisi sonuna gelindiğinde yığın boş ise, S dizisi **geçerlidir**.

Parantez Kontrolü

Girdi	İşlem	Yığın
{	push	{
{	push	{ {
[	push	{ { [
]	pop	{ {
}	pop	{
(	push	{ (
)	pop	{
}	pop	
[	push	[
]	pop	

Parantez Kontrolü

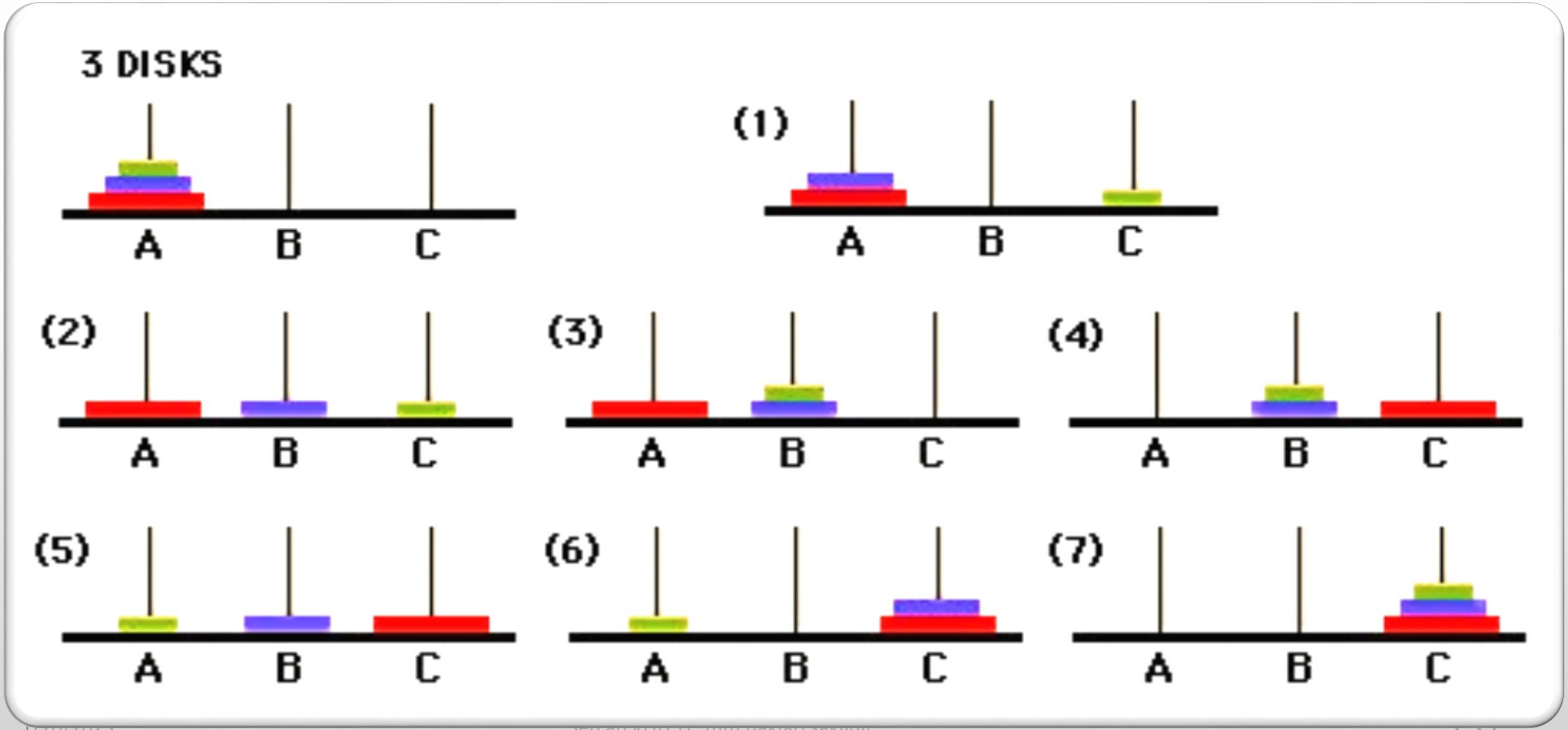
```
var stack = new Stack<String>();
  for (var c : s.toCharArray()) {
    if (c == '(' || c == '[' || c == '{') {
      stack.push(String.valueOf(c));
    } else if ((c == '}' && !stack.pop().equals("{")) ||
               (c == ']' && !stack.pop().equals("[")) ||
               (c == ')' && !stack.pop().equals("("))) {
      return false;
    }
  }
}
return stack.isEmpty();
```

Örnek

Hanoi Kuleleri

- Hanoi Kuleleri, bilgisayar bilimleri ve matematikte klasik bir problemdir.
- Aşağıdaki kurallara uyarak bir disk yığını bir çividen diğerine taşınır.
  - Aynı anda bir disk taşınabilir.
  - Bir disk daha büyük bir diskin veya boş çivinin üzerine yerleştirilebilir.
  - Amaç, gerektiğinde yardımcı bir çivi kullanarak tüm disk yığınını kaynaktan hedefe taşımaktır.

Hanoi Kuleleri



Hanoi Kuleleri

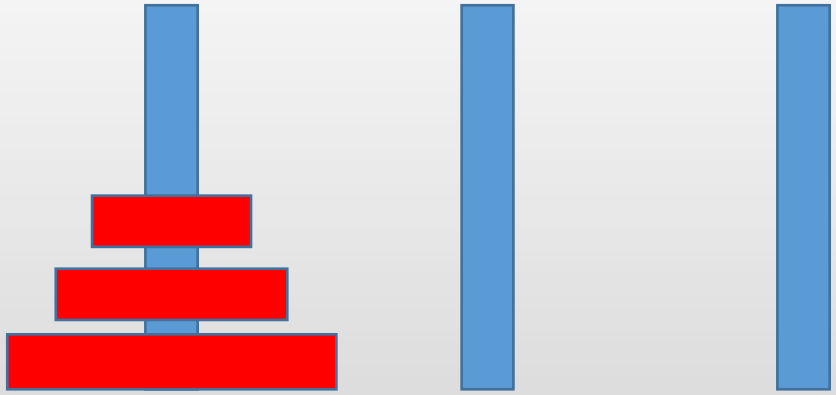
```
void hanoi(int diskSayisi) {
  Stack<Disk> yigin = new Stack<>();
  yigin.push(new Disk(diskSayisi, 'A', 'B', 'C')); // Başlangıç durumu
  while (!yigin.isEmpty()) {
    Disk disk = yigin.pop();
    if (disk.boyut == 1) {
      System.out.println("Diski" + disk.kaynak + "tan" + disk.hedef + "e taşı.");
    } else {
      // Yardımcı çubuğu kullanarak diskleri geçici olarak taşı
      yigin.push(new Disk(disk.boyut-1, disk.yard, disk.kaynak, disk.hedef));
      yigin.push(new Disk(1, disk.kaynak, disk.yardimci, disk.hedef));
      yigin.push(new Disk(disk.boyut-1, disk.kaynak, disk.hedef, disk.yard));
    }
  }
```

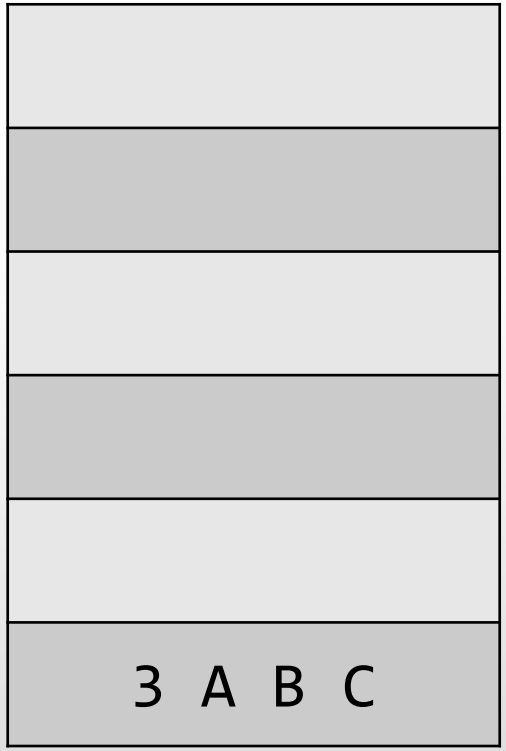
Hanoi Kuleleri

- Döngünün içinde, yığından bir Disk nesnesi çıkarılır, boyutu kontrol edilir.
  - **Boyut 1 ise**, disk doğrudan hedefe taşınır, hareketi belirten bir mesaj yazdırılır.
  - **Değilse,** yığına üç yeni Disk nesnesi yerleştirilerek, problemi daha küçük alt problemlere böler.
    - İlk nesne, hedefi yardımcı olarak kullanarak, disk.boyut - 1 adet diski kaynaktan yardımcıya taşımayı temsil eder.
    - 2. nesne, 1 adet diskin kaynaktan hedefe taşınmasını temsil eder.
    - 3. nesne, kaynağı yardımcı olarak kullanarak, disk.boyut - 1 adet diski yardımcıdan hedefe taşımayı temsil eder.

Hanoi Kuleleri – Yığın İşlemleri

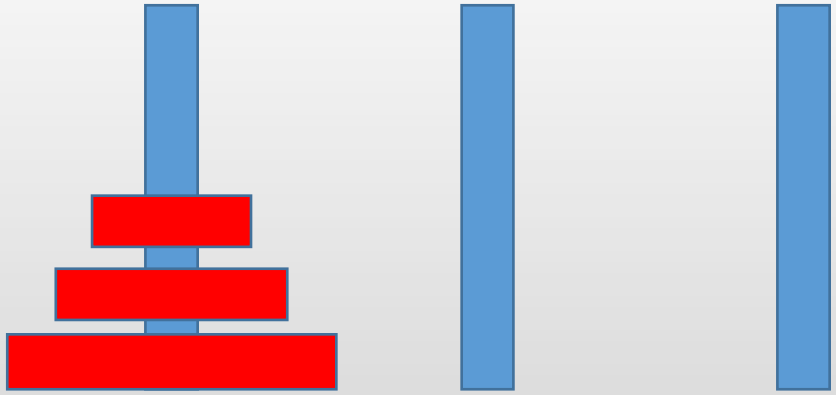
- `Yığına ekle 3 A B C`

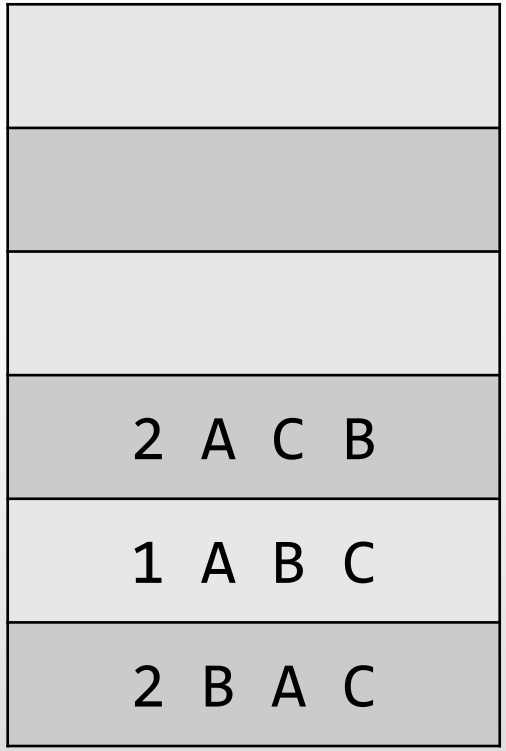




Hanoi Kuleleri – Yığın İşlemleri

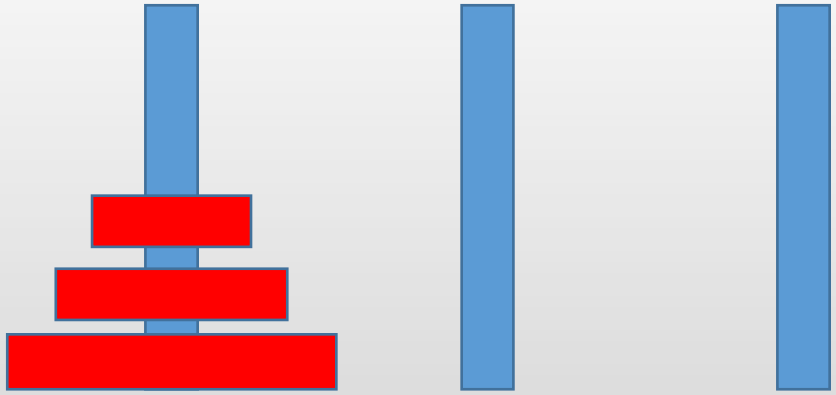
- Yığından al 3 A B C
- Yığına ekle 2 B A C
- Yığına ekle 1 A B C
- Yığına ekle 2 A C B





Hanoi Kuleleri – Yığın İşlemleri

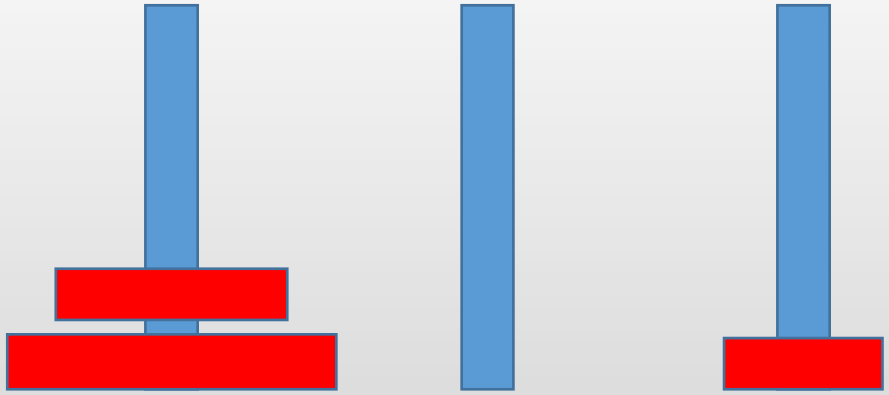
- Yığından al 2 A C B
- Yığına ekle 1 C A B
- Yığına ekle 1 A C B
- Yığına ekle 1 A B C



1 A B C
1 A C B
1 C A B
1 A B C
2 B A C

Hanoi Kuleleri – Yığın İşlemleri

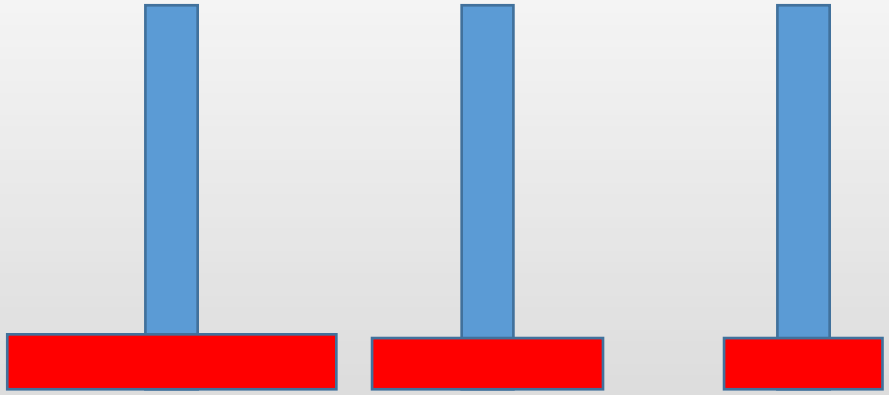
- Yığından al 1 A B C
- Diski A çubuğundan C çubuğuna taşı.



1 A C B
1 C A B
1 A B C
2 B A C

Hanoi Kuleleri – Yığın İşlemleri

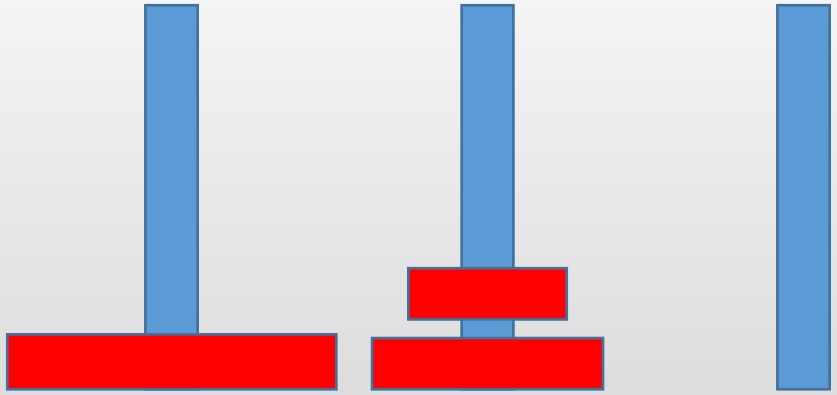
- Yığından al 1 A C B
- Diski A çubuğundan B çubuğuna taşı.

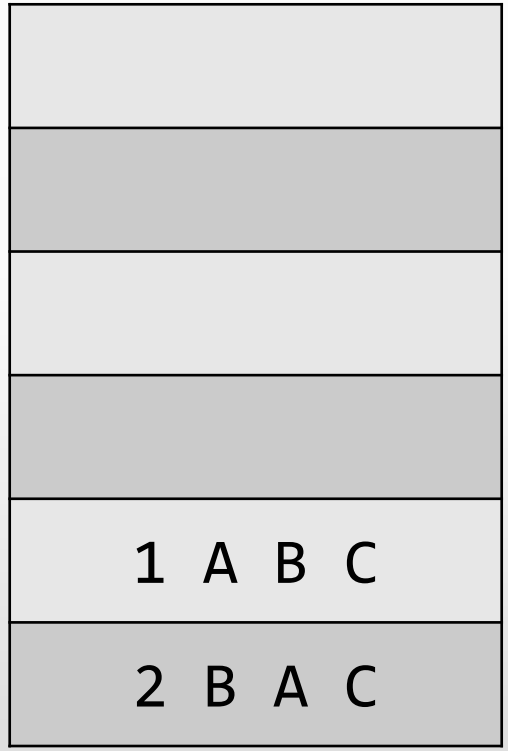


1 C A B
1 A B C
2 B A C

Hanoi Kuleleri – Yığın İşlemleri

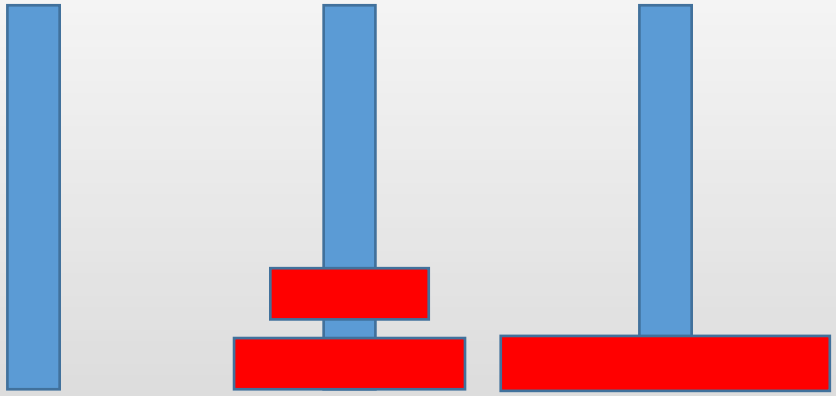
- Yığından al 1 C A B
- Diski C çubuğundan B çubuğuna taşı.

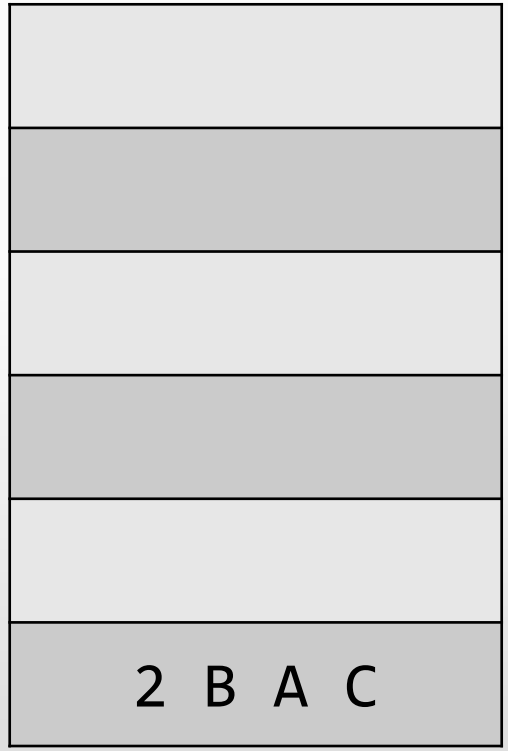




Hanoi Kuleleri – Yığın İşlemleri

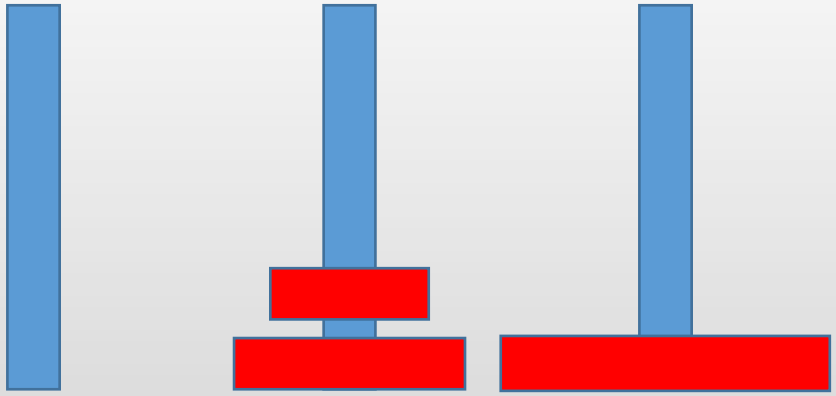
- `Yığından al 1 A B C`
- `Diski A çubuğundan C çubuğuna taşı.`

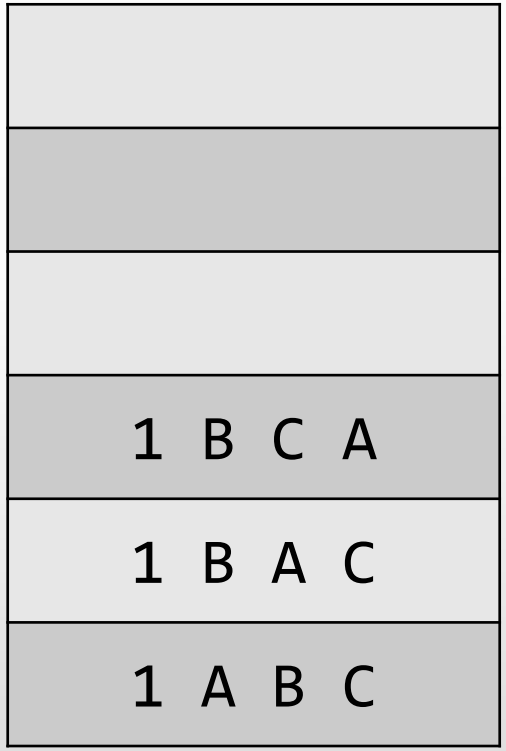




Hanoi Kuleleri – Yığın İşlemleri

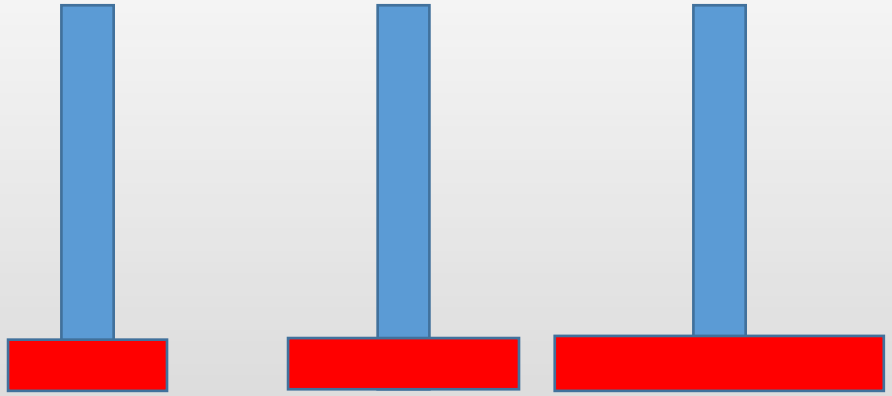
- Yığından al 2 B A C
- Yığına ekle 1 A B C
- Yığına ekle 1 B A C
- Yığına ekle 1 B C A

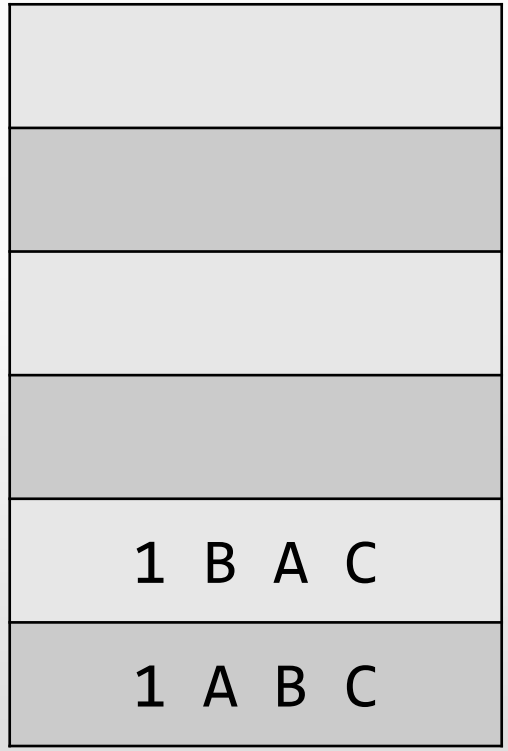




Hanoi Kuleleri – Yığın İşlemleri

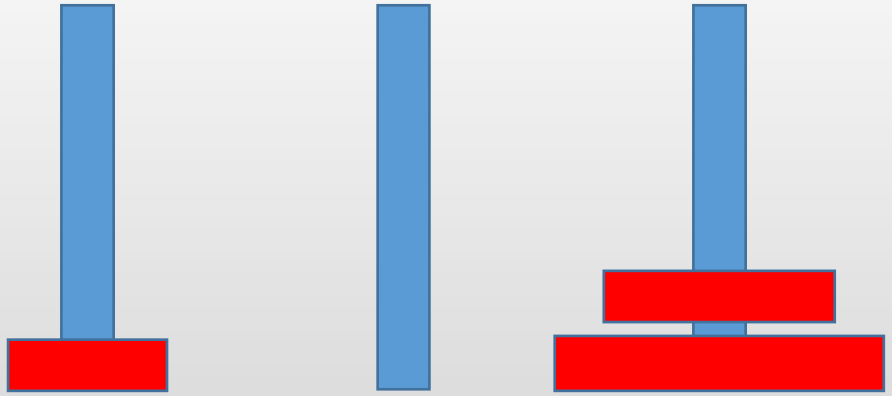
- Yığından al 1 B C A
- Diski B çubuğundan A çubuğuna taşı.

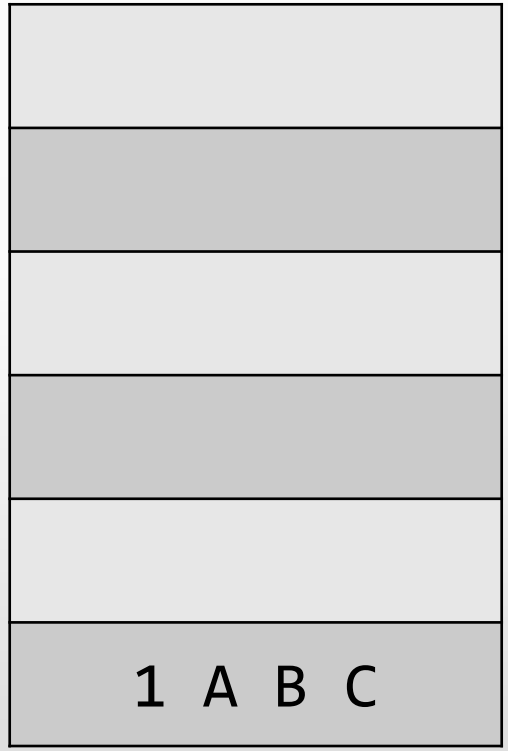




Hanoi Kuleleri – Yığın İşlemleri

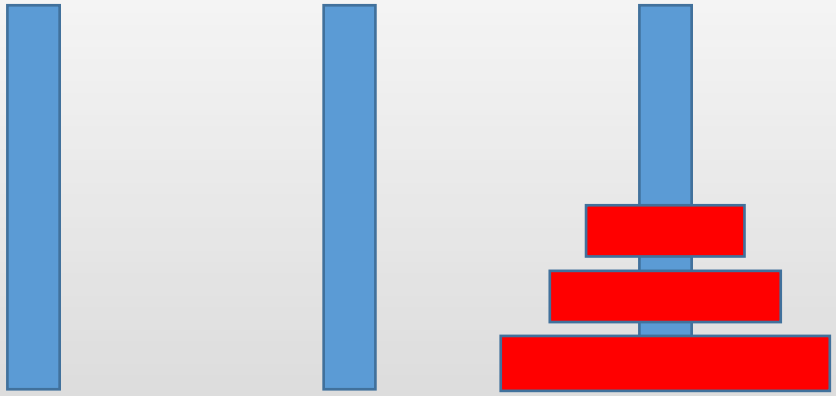
- Yığından al 1 B A C
- Diski B çubuğundan C çubuğuna taşı.





Hanoi Kuleleri – Yığın İşlemleri

- Yığından al 1 A B C
- Diski A çubuğundan C çubuğuna taşı.



Örnek

Stock Span Problemi

- "Hisse Senedi Sıçrama Problemi," hisse senedi veya finansal veri analizinde kullanılan bir problemdir.
- Bir zaman aralığındaki hisse senedi fiyatlarını inceler.
- Her bir gün için, o güne kadar kaç ardışık gün kapanış fiyatının arttığını hesaplar.

Stock Span Problemi

- Verilen bir dizi hisse senedi fiyatı var.
- Her günün kapanış fiyatı sırasıyla dizi içinde bulunuyor.
- Her gün için, o güne kadar olan en uzun ardışık gün sayısının (bu günün fiyatından yüksek önceki günlerin sayısı) hesaplanması gereklidir.

- **Örnek**:
  - Hisse senedi fiyatları aşağıdaki gibi olsun:
  - [100, 80, 60, 70, 60, 75, 85]

Stock Span Problemi

- **Örnek**: [100, 80, 60, 70, 60, 75, 85]

- Bu verilere göre, her gün için ardışık gün sayısı:
- 1. 100 : 1 (ilk gün olduğu için)
- 2. 80 : 1 (80 > 100 değil)
- 3. 60 : 1 (60 > 80 değil)
- 4. 70 : 2 (70 > 60)
- 5. 60 : 1 (60 > 70 değil)
- 6. 75 : 4 (75 > {60, 70, 60})
- 7. 85 : 6 (85 > {75, 60, 70, 60, 80})

Stock Span Problemi

```
int[] sicrama = new int[fiyatlar.length];
Stack<Integer> yigin = new Stack<>();

yigin.push(0);
sicrama[0] = 1;

for (int i = 1; i < fiyatlar.length; i++) {
  while (!yigin.isEmpty() && fiyatlar[i] >= fiyatlar[yigin.peek()]) {
    yigin.pop();
  }
  sicrama[i] = yigin.isEmpty() ? (i + 1) : (i - yigin.peek());
  yigin.push(i);
}
```

Stock Span Problemi

- [100, 80, 60, 70, 60, 75, 85]
- Yığına ekle 0

0	**100**

Stock Span Problemi

- [100, 80, 60, 70, 60, 75, 85]
- Yığına ekle 1, sicrama[1] 1

1	**80**
0	**100**

Stock Span Problemi

- [100, 80, 60, 70, 60, 75, 85]
- Yığına ekle 2, sicrama[2] 1

2	**60**
1	**80**
0	**100**

Stock Span Problemi

- [100, 80, 60, 70, 60, 75, 85]

- Yığından al 2
- Yığına ekle 3, sicrama[3] 2

3	**70**
1	**80**
0	**100**

Stock Span Problemi

- [100, 80, 60, 70, 60, 75, 85]
- Yığına ekle 4, sicrama[4] 1

4	60
3	70
1	80
0	100

Stock Span Problemi

- [100, 80, 60, 70, 60, 75, 85]

- Yığından al 4
- Yığından al 3
- Yığına ekle 5, sicrama[5] 4

5	**75**
1	**80**
0	**100**

Stock Span Problemi

- [100, 80, 60, 70, 60, 75, 85]

- Yığından al 5
- Yığından al 1
- Yığına ekle 6, sicrama[6] 6

6	**85**
0	**100**

Örnek

Infix, Prefix, Postfix İfade Dönüşümleri

- Matematiksel ve mantıksal ifadelerle çalışırken kullanılır.
- İfadelerin basit bir şekilde değerlendirilmesini sağlar.
- İşlem öncelikleri ve parantez kontrollerini ortadan kaldırır.
- Programlama dilleri, ifadeleri öntakı gösterimle temsil eder.
- Bilgisayar ve hesap makinelerinde aritmetik işlemlerde kullanılır.
- Matematiksel denklemleri basitleştirme, sembolik cebir veya karmaşık problemleri çözme gibi işlemler sırasında faydalanılır.

Infix, Prefix, Postfix İfade Dönüşümleri

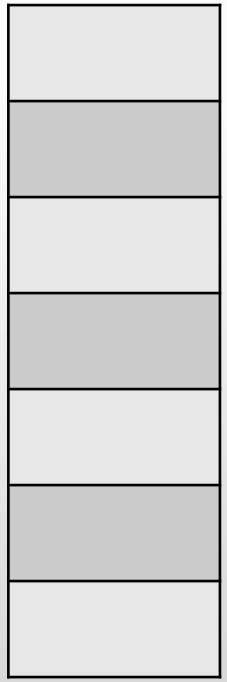
- İçtakı İfade: A * (B + C) / D
- Öntakı İfade: * A / + B C D
- Arttakı İfade: A B C + * D /

- İşlem öncelikleri ve parantezler kayboldu!

Infix İfadeyi Postfix İfadeye Dönüştürme

- Infix İfade: A*(B+C)/D

Postfix:

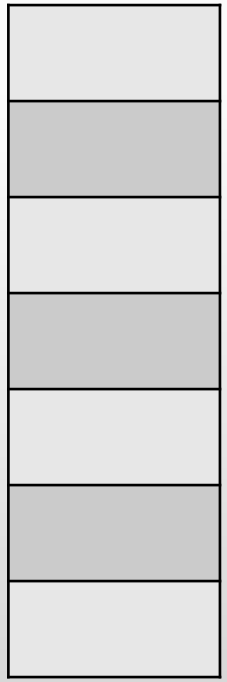


Infix İfadeyi Postfix İfadeye Dönüştürme

- Infix İfade: A*(B+C)/D

- Adım: A
- postfix: A

Postfix: A

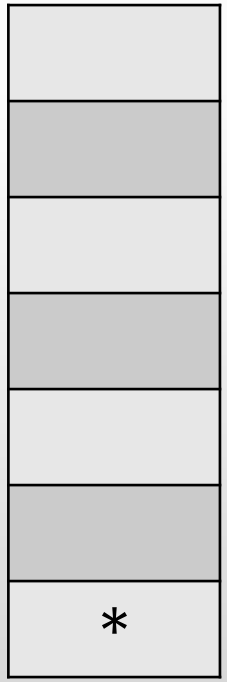


Infix İfadeyi Postfix İfadeye Dönüştürme

- Infix İfade: A*(B+C)/D

- Adım: *
- Yığına koy *

Postfix: A

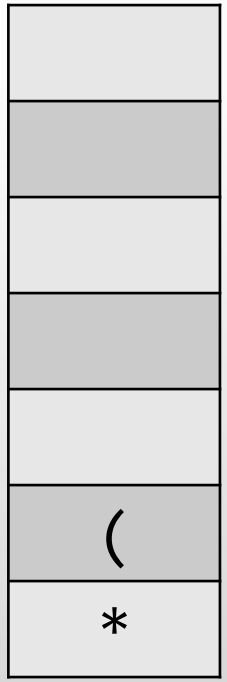


Infix İfadeyi Postfix İfadeye Dönüştürme

- Infix İfade: A*(B+C)/D

- Adım: (
- Yığına koy (

Postfix: A

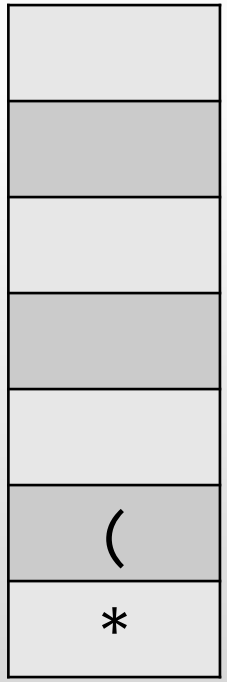


Infix İfadeyi Postfix İfadeye Dönüştürme

- Infix İfade: A*(B+C)/D

- Adım: B
- postfix: AB

Postfix: AB

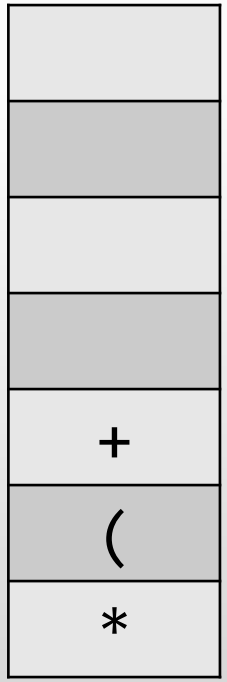


Infix İfadeyi Postfix İfadeye Dönüştürme

- Infix İfade: A*(B+C)/D

- Adım: +
- Yığına koy +

Postfix: AB

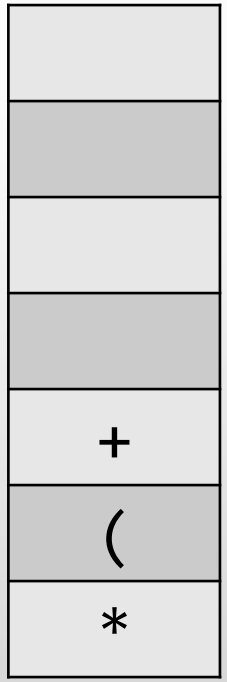


Infix İfadeyi Postfix İfadeye Dönüştürme

- Infix İfade: A*(B+C)/D

- Adım: C
- postfix: ABC

Postfix: ABC

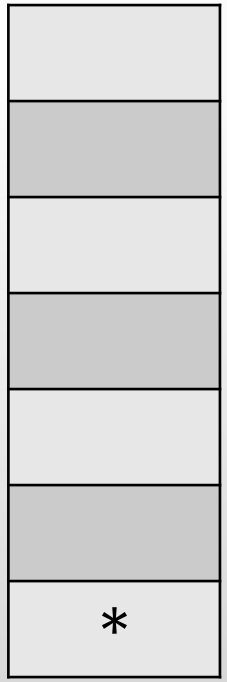


Infix İfadeyi Postfix İfadeye Dönüştürme

- Infix İfade: A*(B+C)/D

- Adım:)
- Yığından al +
- postfix: ABC+
- Yığından al (

Postfix: ABC+

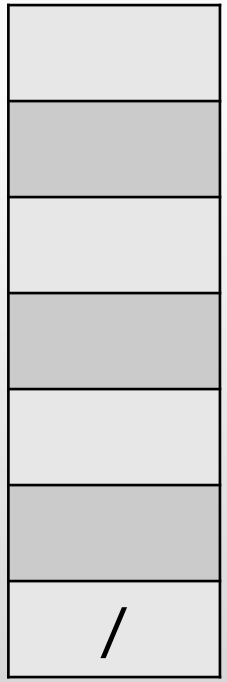


Infix İfadeyi Postfix İfadeye Dönüştürme

- Infix İfade: A*(B+C)/D

- Adım: /
- Yığından al *
- postfix: ABC+*
- Yığına koy /

Postfix: ABC+*

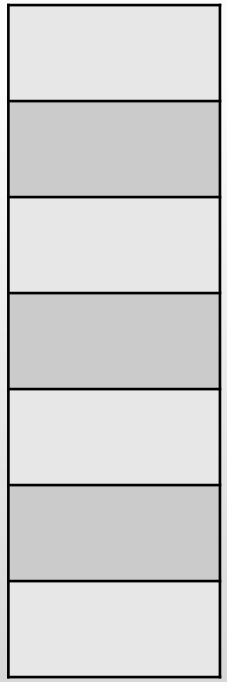


Infix İfadeyi Postfix İfadeye Dönüştürme

- Infix İfade: A*(B+C)/D

- Adım: D
- postfix: ABC+*D
- Yığından al /
- postfix: ABC+*D/

Postfix: ABC+*D/



SON