Bölüm 4: Yığın

Veri Yapıları

Yığın (Stack)

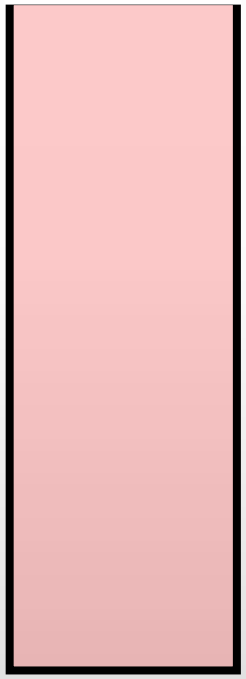
- Yığın (stack), yeni öğelerin eklenmesi ve mevcut öğelerin kaldırılmasının yığının en üstünden (top) gerçekleştiği bir lineer veri yapısıdır.
- Yığın, üst üste konulmuş kutular veya tabaklar gibi düşünülebilir, en üstteki elemana ulaşmak için yığının üzerindeki öğeleri tek tek çıkarmak gerekir.
- İşlemlerin belirli bir sırayı takip ettiği önemli bir veri yapısıdır. Bu sıra, LIFO (Son Giren İlk Çıkar) veya FILO (İlk Giren Son Çıkar) şeklinde olabilir.
  - LIFO, yığına son eklenen öğenin ilk çıkacağı anlamına gelir.
  - FILO ise yığına ilk eklenen öğenin en son çıkacağı anlamına gelir.

Temel İşlemler

- push() İşlemi: Yığına yeni bir öğe eklemek için kullanılır. Eklenen öğe yığının en üstüne eklenir.
- pop() İşlemi: Yığından bir öğeyi çıkarmak için kullanılır. Çıkarılan öğe yığının en üstündeki öğedir.
- top() İşlemi: Yığının en üstündeki öğeyi döndürmek için kullanılır.
- isEmpty() İşlemi: Yığının boş olup olmadığını kontrol etmek için kullanılır.
- size() İşlemi: Yığının boyutunu döndürmek için kullanılır.

Gösterim

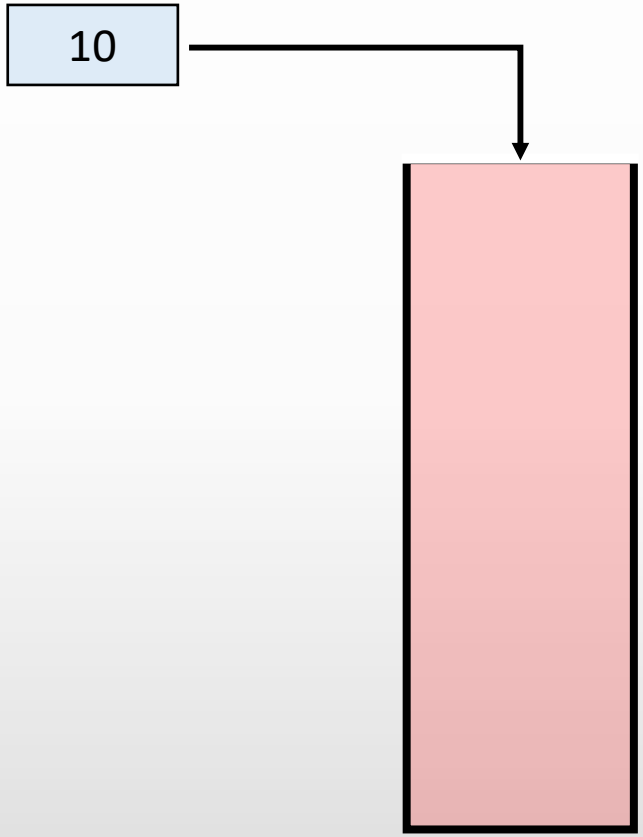
top ⟶ null



Yığıt

Ekleme İşlemi:

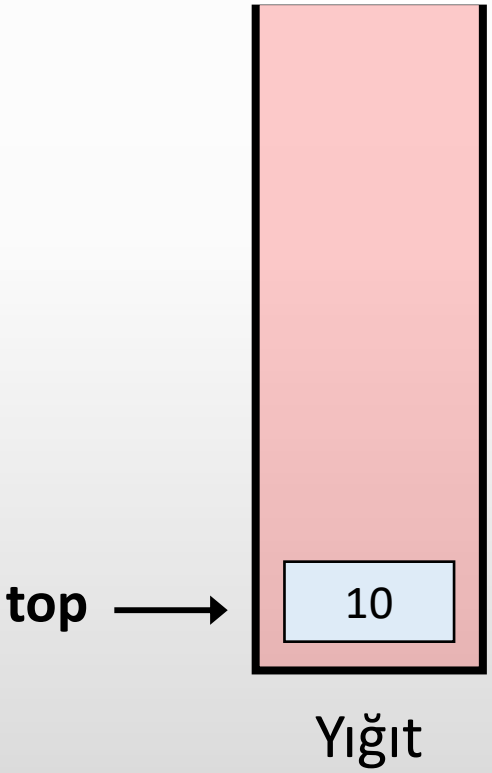
push(10)



top ⟶ null

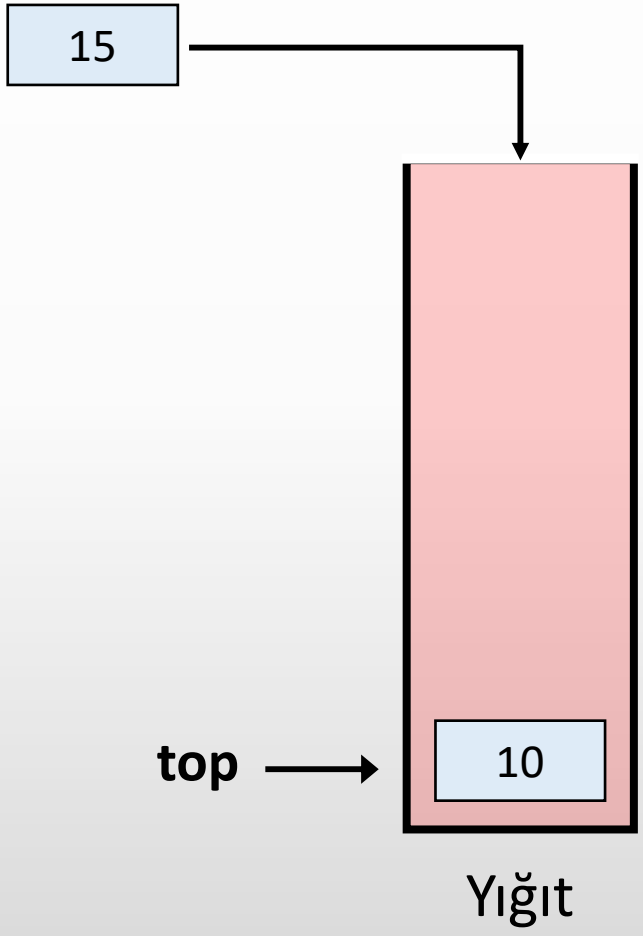
Ekleme İşlemi:

push(10)



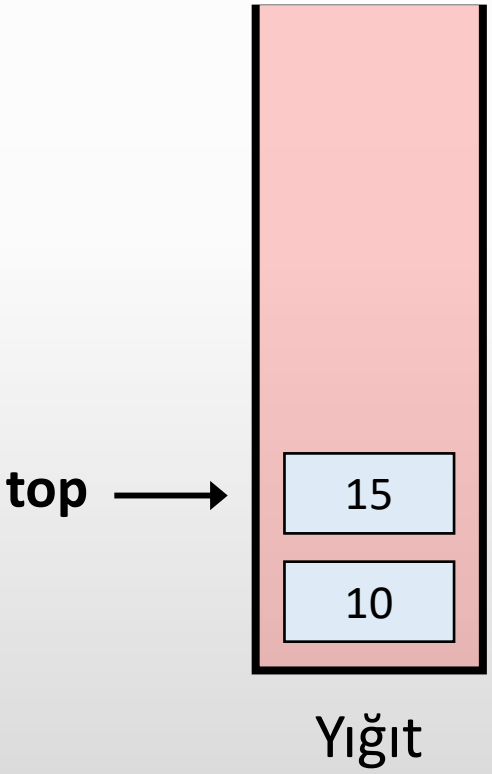
Ekleme İşlemi:

push(15)



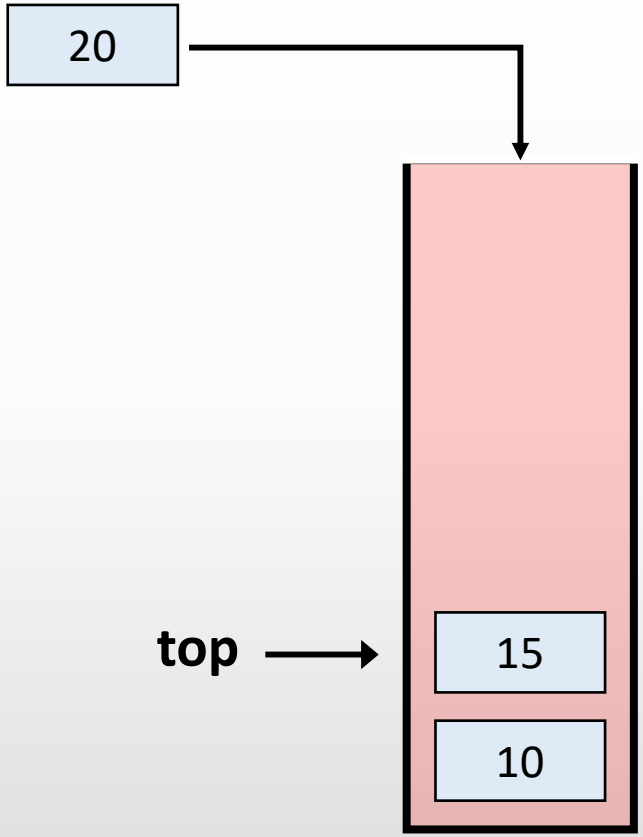
Ekleme İşlemi:

push(15)



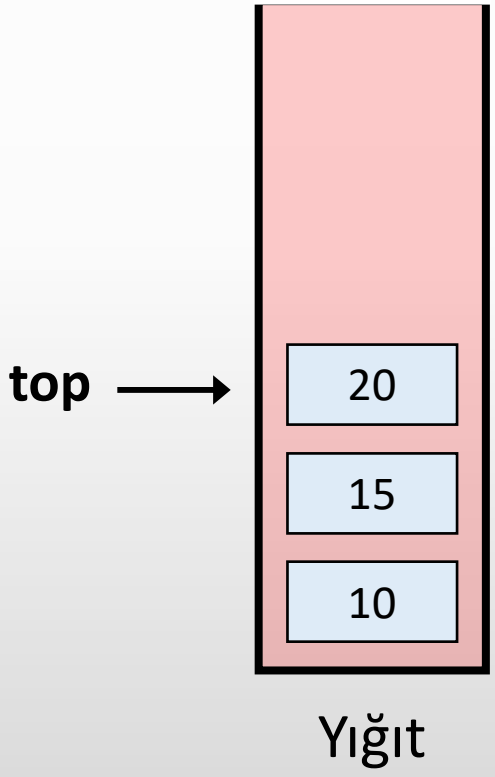
Ekleme İşlemi:

push(20)



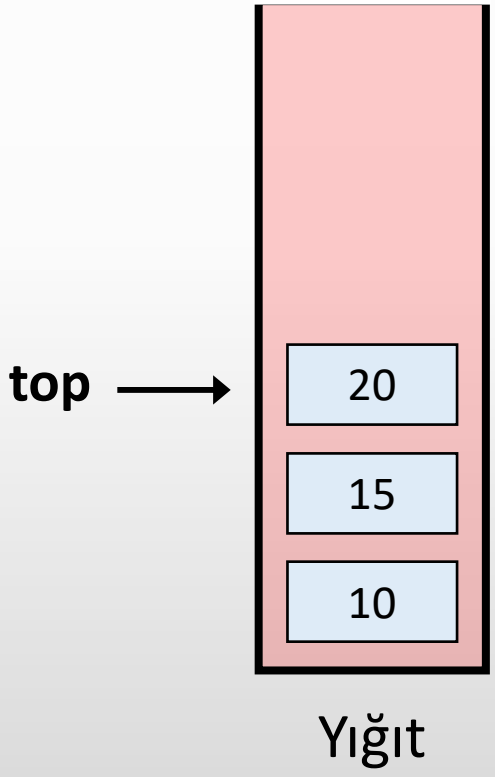
Ekleme İşlemi:

push(20)



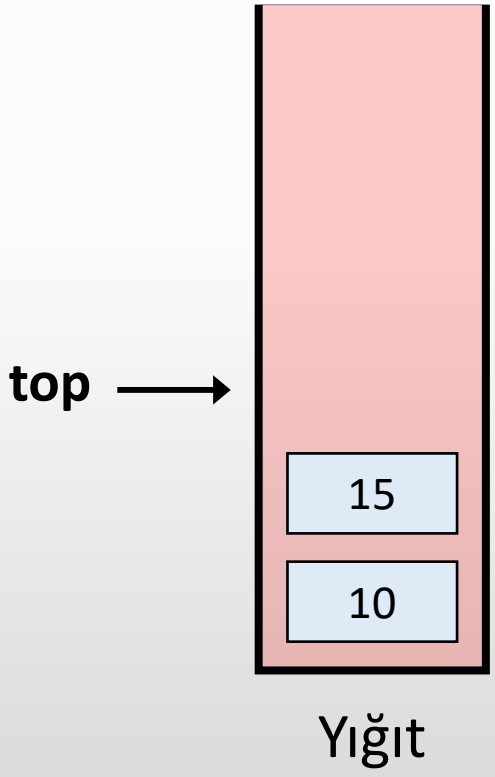
Silme İşlemi:

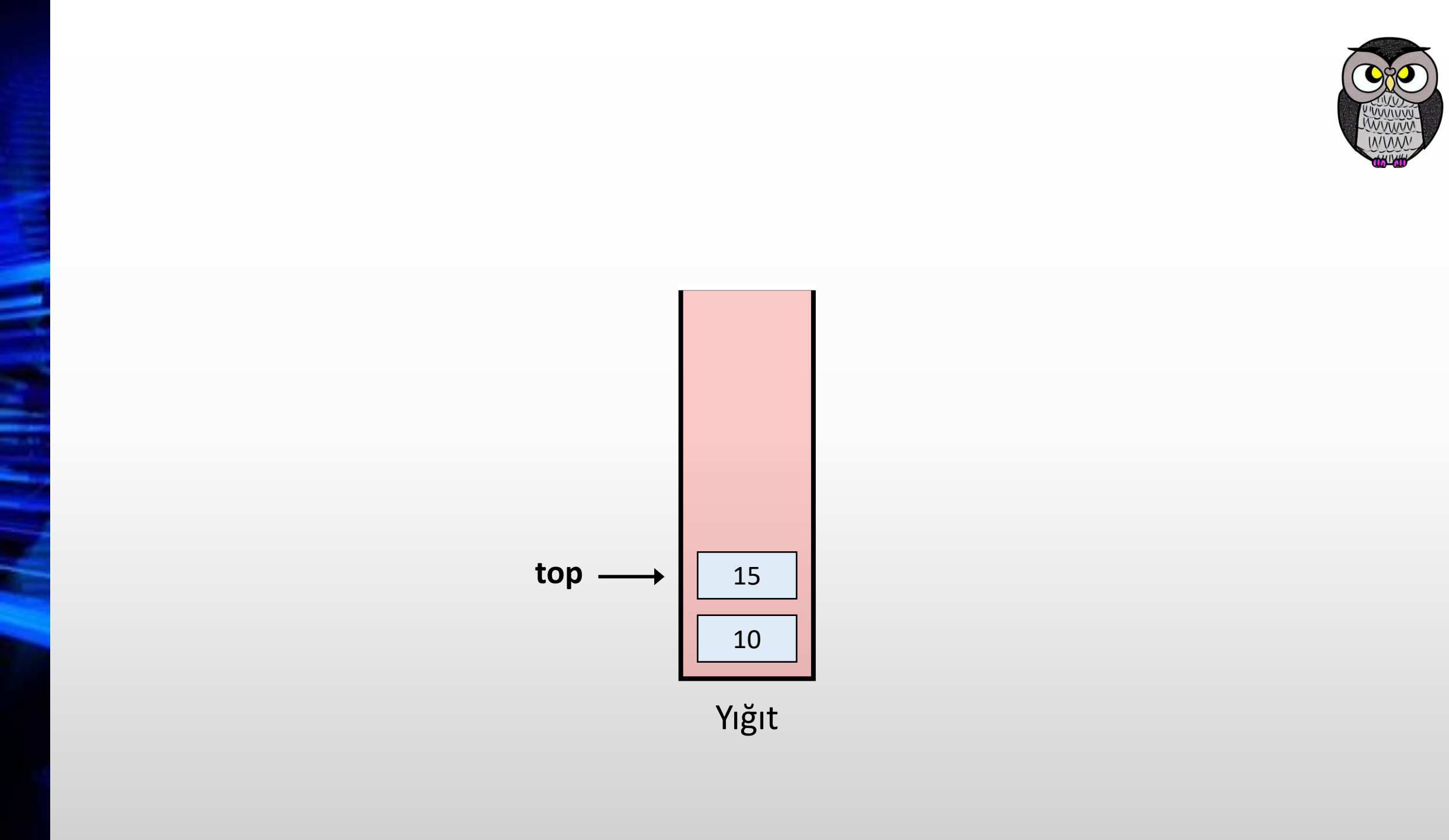
pop()



Silme İşlemi:

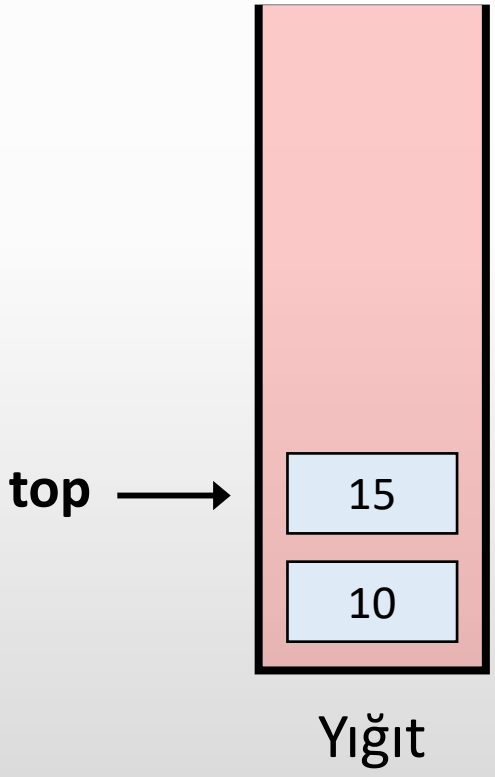
pop()




top
15
10
Yığıt

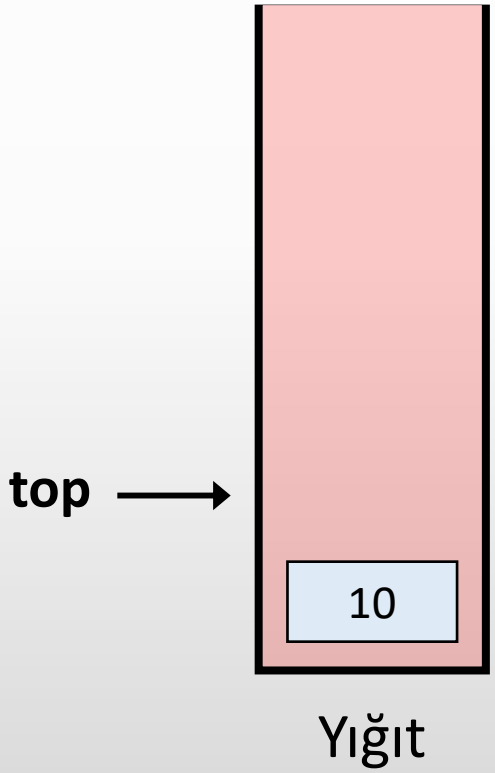
Silme İşlemi:

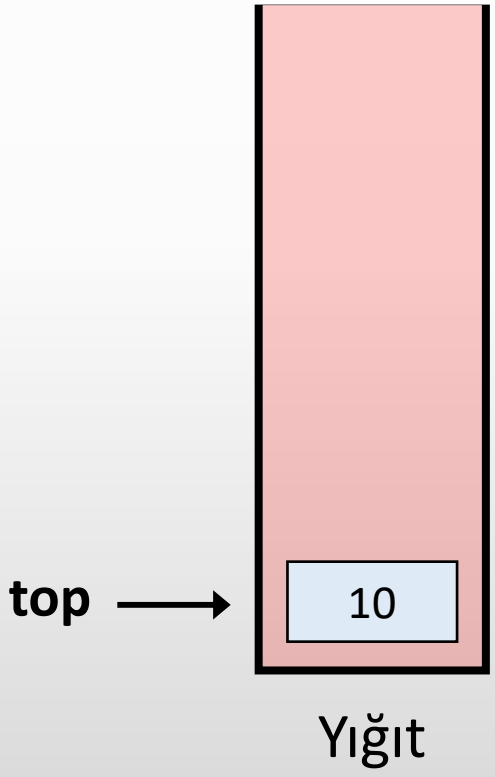
pop()



Silme İşlemi:

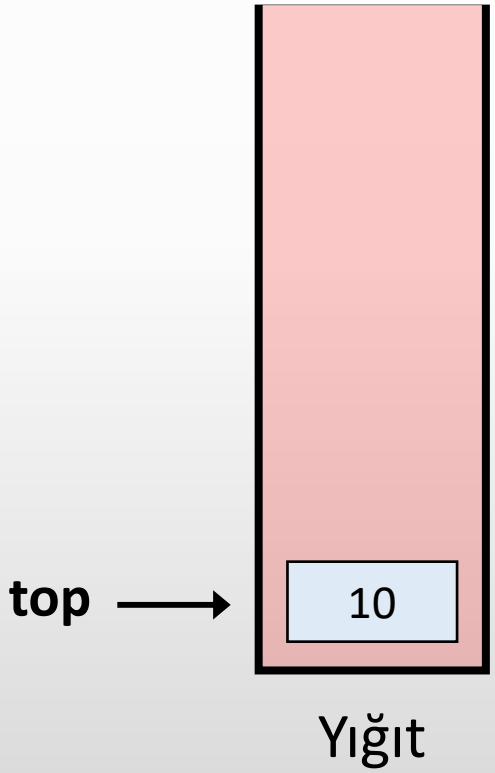
pop()




top
10
Yığıt

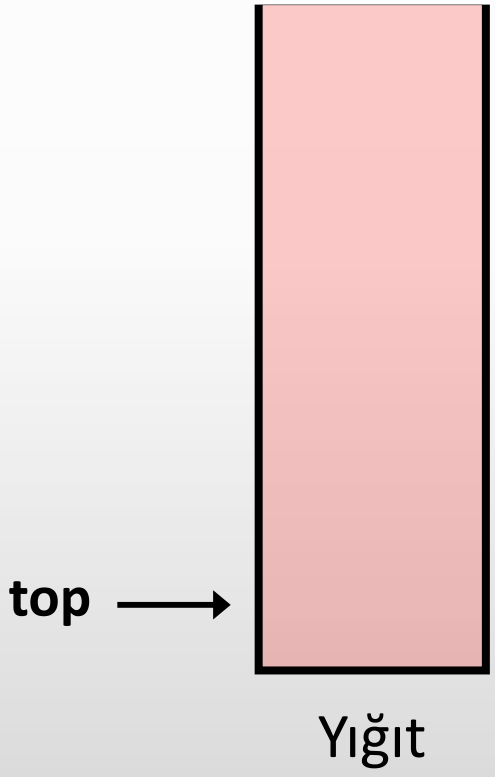
Silme İşlemi:

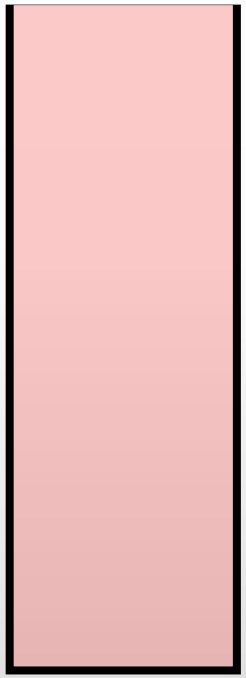
pop()



Silme İşlemi:

pop()





Yığıt

top ⟶ **null**

Push İşlemi Algoritması

- Push işlemi, yığına bir öğe eklemek için kullanılır ve eğer yığın dolu ise, bu durum bir «Üst Taşma» (Overflow) durumu olarak adlandırılır.

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

Pop İşlemi Algoritması

- Pop işlemi, yığından bir öğe çıkarmak için kullanılır ve çıkartılan öğeler, eklenme sırasının tersine (son eklenen ilk çıkarılan) sırayla çıkartılır. Eğer yığın boş ise, bu durum bir «Alt Taşma» (Underflow) durumu olarak adlandırılır.

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

Top İşlemi Algoritması

- Top işlemi, yığının en üstündeki öğeyi döndürmek için kullanılır.

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

isEmpty İşlemi Algoritması

- isEmpty işlemi, yığının boş olup olmadığını kontrol etmek için kullanılır. Boşsa true, doluysa false döndürür.

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

İşlemler ve Zaman Karmaşıklığı

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

Yığın Türleri

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

Yığın Türleri

- **Sabit Boyutlu Yığın (Fixed Size Stack)**
  - Adından da anlaşılacağı gibi belirli bir sabit boyuta sahiptir ve dinamik olarak büyüyemez veya küçülemez.
  - **Taşma (Overflow) Durumu**: Eğer sabit boyutlu yığın doluysa ve yeni bir öğe eklemeye çalışılırsa, bir taşma hatası meydana gelir.
  - **Alt Taşma (Underflow) Durumu**: Eğer sabit boyutlu yığın boşsa ve bir öğe çıkarmaya çalışılırsa, bir alt taşma hatası meydana gelir.
- 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. Yığın dolu olduğunda, yeni öğeyi barındırmak için otomatik olarak boyutunu artırır ve boş olduğunda boyutunu azaltır.
  - Bağlı liste kullanılarak uygulanır çünkü yığının boyutunu kolayca değiştirmeye izin verir.

Uygulama Örnekleri

- **İnfix'ten Postfix/Prefix Dönüşümü**: Matematiksel ifadelerin dönüşümünde kullanılır.
- **Yeniden Yapma ve Geri Alma (redo-undo)**: Metin düzenleyicilerden Photoshop'a kadar birçok yerde kullanılır.
- **İleri ve Geri Hareket Etme**: Web tarayıcılarının gezinme geçmişini yönetmek için kullanılır.
- **Algoritmalar**: Hanoi Kuleleri, ağaç veri yapısında gezinme, hisse senedi sıçrama problemi, histogram problemleri gibi algoritma uygulamalarında kullanılır.
- **Geri İzleme (Backtracking)**: Şövalye Turu, N-Vezir problemleri, labirentlerde yol bulma ve satranç gibi oyunlarda kullanılır.

Uygulama Örnekleri

- **Çizge Algoritmaları**: Topolojik Sıralama ve Güçlü Bağlantılı Bileşenlerin bulunmasında kullanılır.
- **Bellek Yönetimi**: Her bir programın çalışma sırasında kullandığı bellek tahsisleri için kullanılır.
- **Dize Ters Çevirme**: Bir dizenin ters çevrilmesinde kullanılır. Dize karakterleri yığına birer birer eklenir ve ters sırada alınır.
- **Fonksiyon Çağrıları**: Bilgisayar sistemlerinde işlev çağrılarını uygulamak için kullanılır. Son çağrılan işlev her zaman önce tamamlanır.

Yığın Gerçekleştirme (implementation)

- Dizi (array) veya bağlı liste (linked list) kullanılarak gerçekleştirilebilir.
- **Dizi Tabanlı:**
  - push() işlemi, üst öğenin endeksini artırarak ve yeni öğeyi bu endekse atayarak uygulanır.
  - pop() işlemi, üst öğenin endeksini azaltarak ve bu endeksteki değeri döndürerek uygulanır.
- **Bağlı Liste Tabanlı:**
  - push() işlemi, yeni öğe ile yeni bir düğüm oluşturarak ve mevcut üst düğümün sonraki işaretçisini yeni düğüme ayarlayarak uygulanır.
  - pop() işlemi, üst düğümün sonraki işaretçisini bir sonraki düğüme ayarlayarak ve mevcut üst düğümün değerini döndürerek uygulanır.

Dizi Tabanlı Yığın Gerçekleştirme

- Kolay uygulanabilir.
- İşaretçiler kullanılmadığı için bellek tasarrufu sağlar.
- Dinamik değildir, yani çalışma zamanındaki ihtiyaca göre büyüme ve küçülme yapamaz.
- Yığının toplam boyutu önceden tanımlanmalıdır.

Bağlı Liste Tabanlı Yığın Gerçekleştirme

- Çalışma zamanındaki ihtiyaca göre büyüme ve küçülme yapabilir.
- JVM gibi birçok sanal makinede kullanılır.
- İşaretçilerin kullanılması nedeniyle ekstra bellek gerektirir.
- Yığında rastgele erişim mümkün değildir.

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)

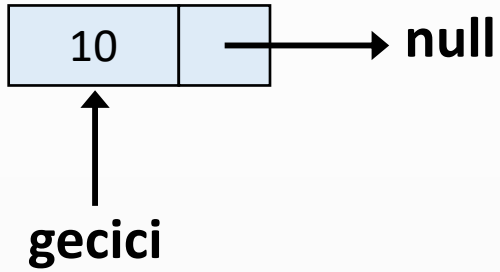
```
// Örnek değişkenleri
Dugum top;
int uzunluk;

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++;
}
```

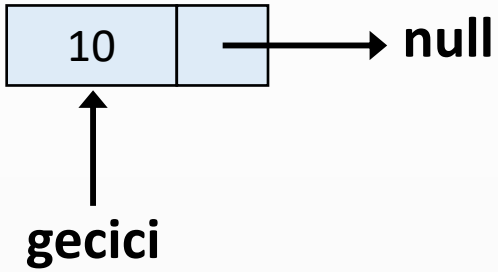


top ⟶ null

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++;
}
```

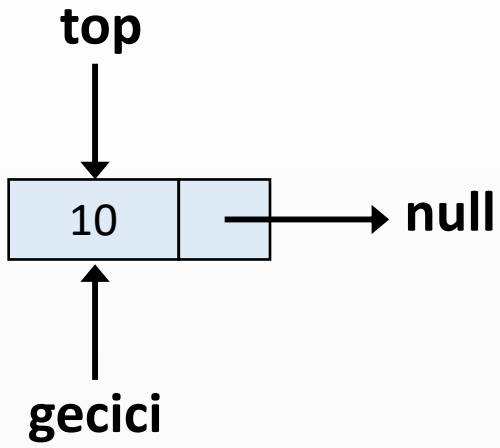


top ⟶ null

uzunluk = 0

push(10)

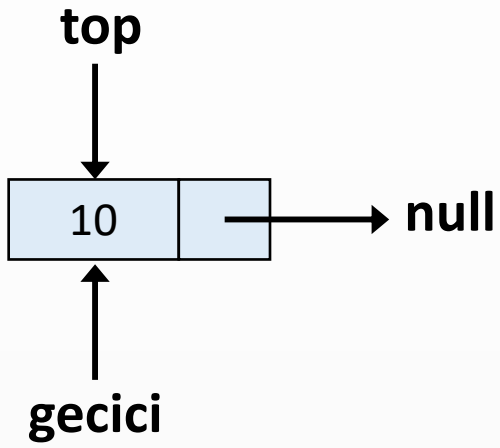
```
// Örnek değişkenleri
Dugum top;
int uzunluk;

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



uzunluk = 0

push(10)

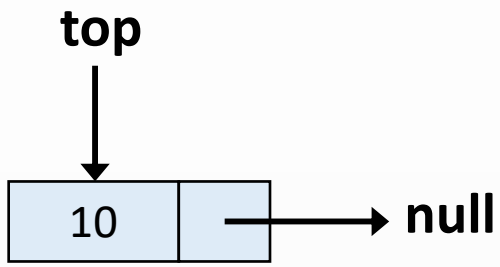
```
// Örnek değişkenleri
Dugum top;
int uzunluk;

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



uzunluk = 1

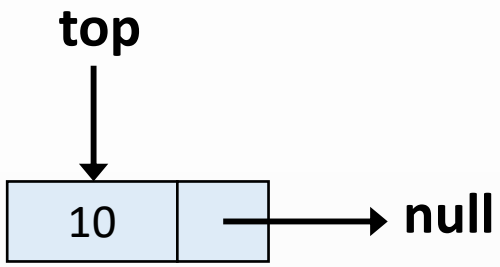
push(10)

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

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



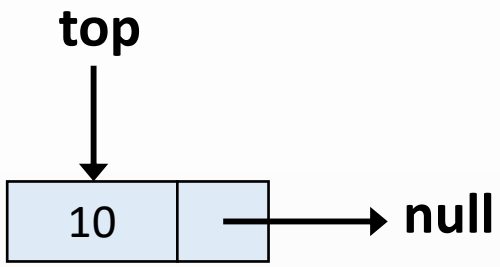
uzunluk = 1

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

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



uzunluk = 1

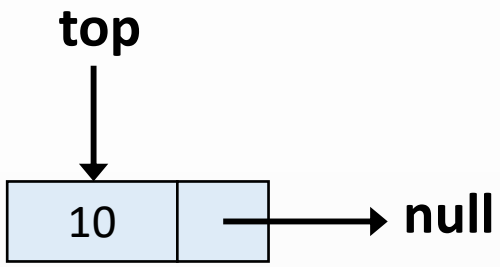
```
// Örnek değişkenleri
Dugum top;
int uzunluk;

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



uzunluk = 1

push(15)

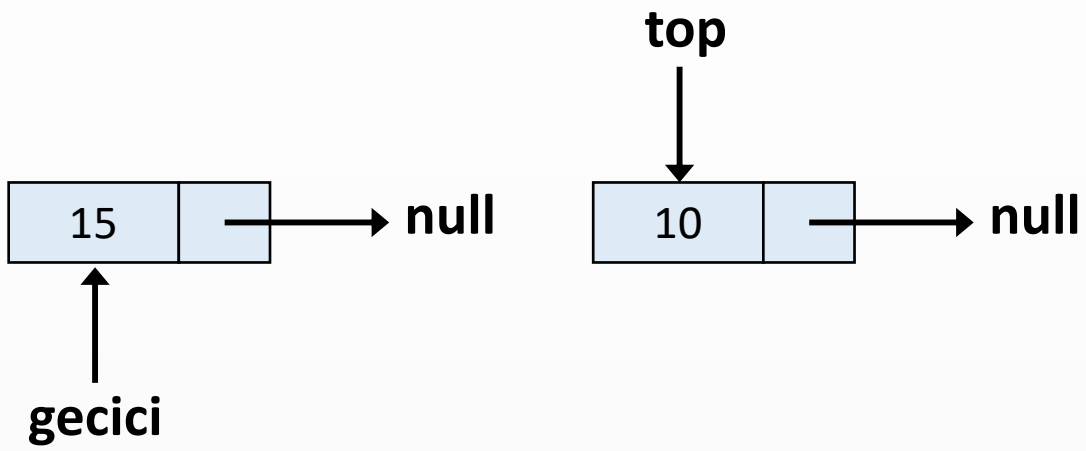
```
// Örnek değişkenleri
Dugum top;
int uzunluk;

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



uzunluk = 1

push(15)

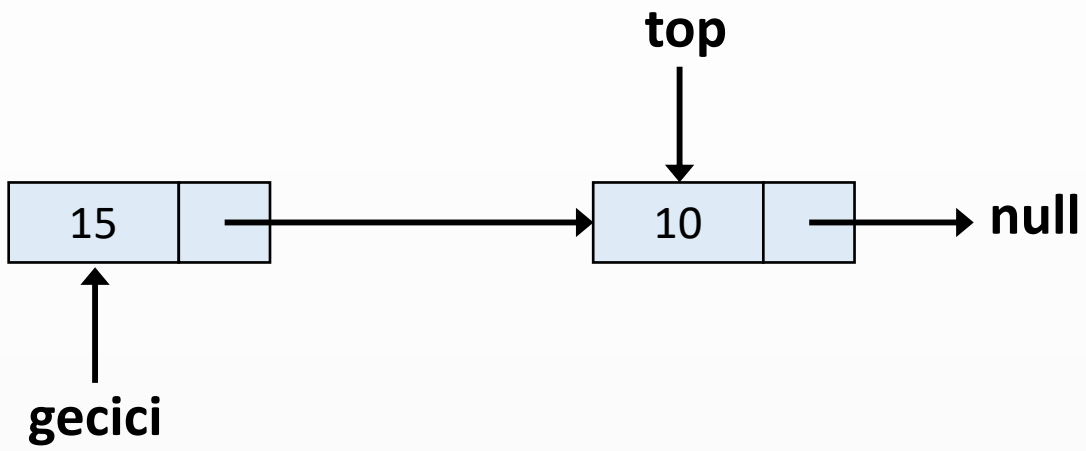
```
// Örnek değişkenleri
Dugum top;
int uzunluk;

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



uzunluk = 1

push(15)

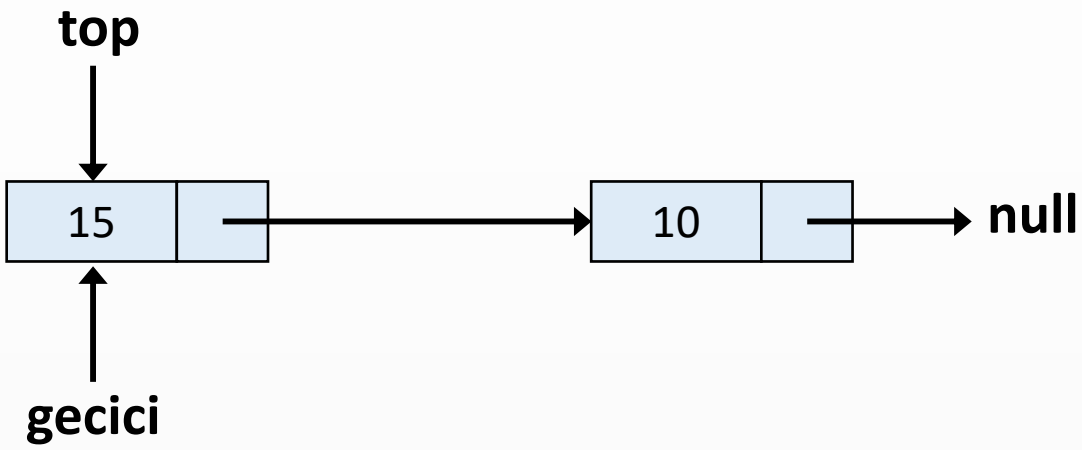
```
// Örnek değişkenleri
Dugum top;
int uzunluk;

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



uzunluk = 1

push(15)

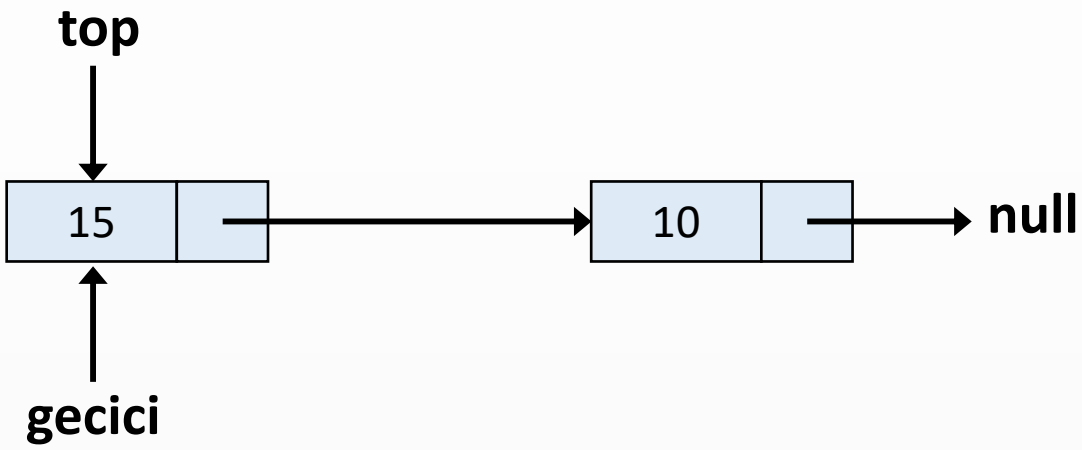
```
// Örnek değişkenleri
Dugum top;
int uzunluk;

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



uzunluk = 1

push(15)

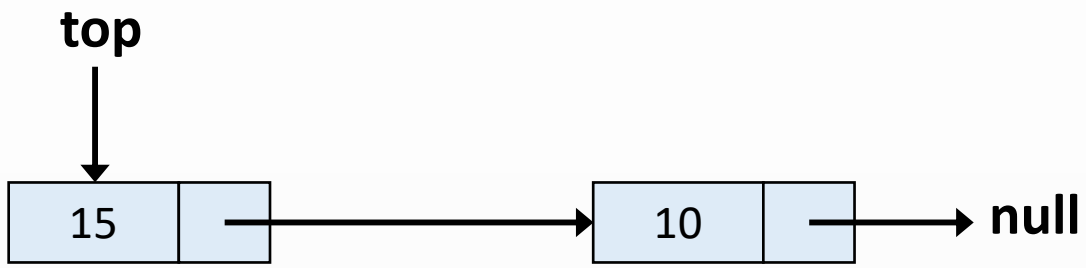
```
// Örnek değişkenleri
Dugum top;
int uzunluk;

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



uzunluk = 2

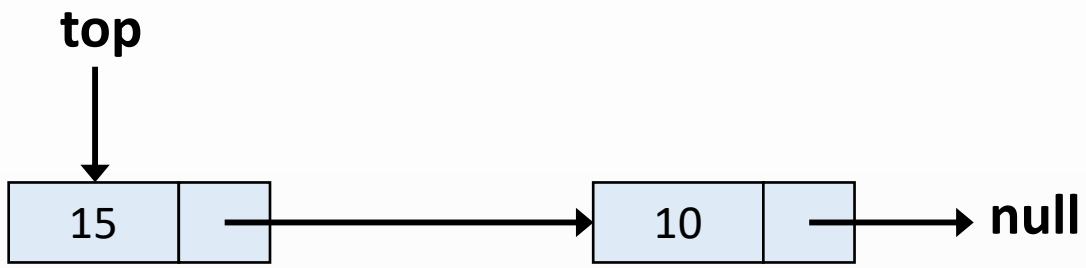
push(15)

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

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



uzunluk = 2

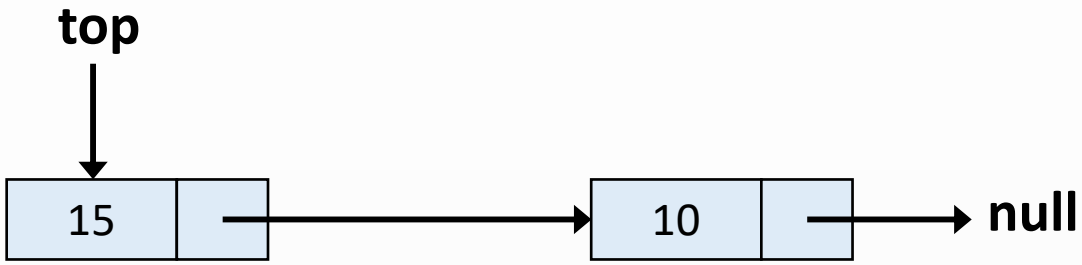
```
// Örnek değişkenleri
Dugum top;
int uzunluk;

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



uzunluk = 2

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

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

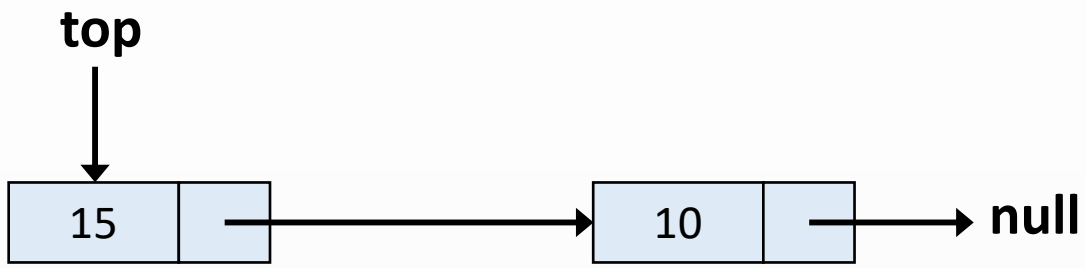
Uygulama - pop



uzunluk = 2

pop()

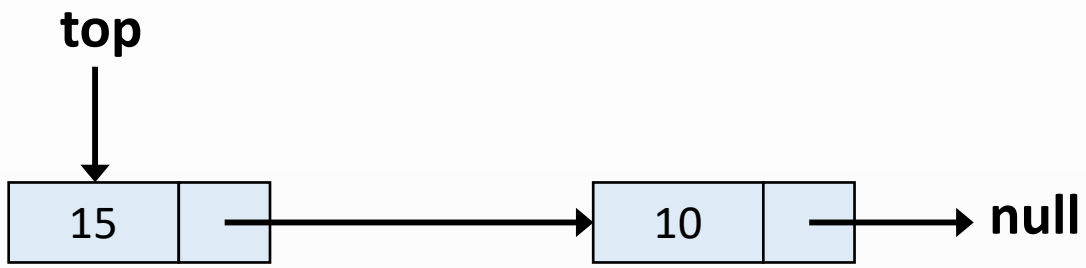
```
// Örnek değişkenleri
Dugum top;
int uzunluk;

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;
}
```

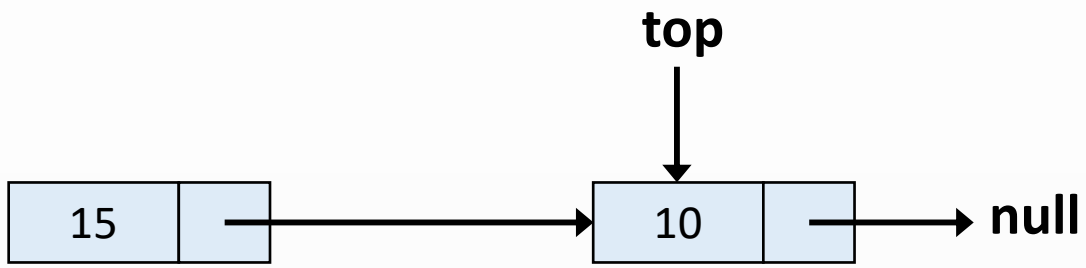


uzunluk = 2

sonuc = 15

pop()

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

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

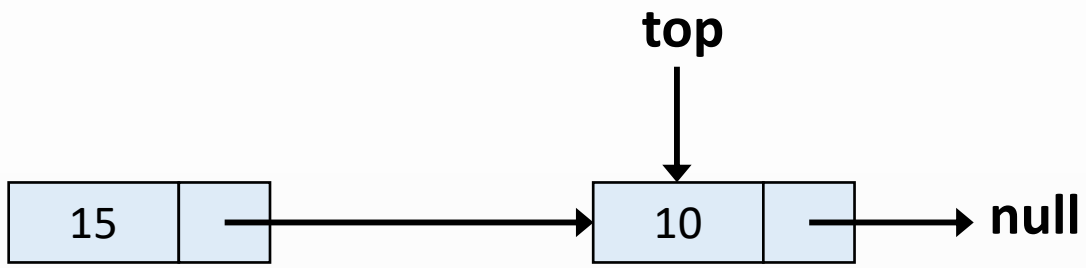


uzunluk = 2

sonuc = 15

pop()

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

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

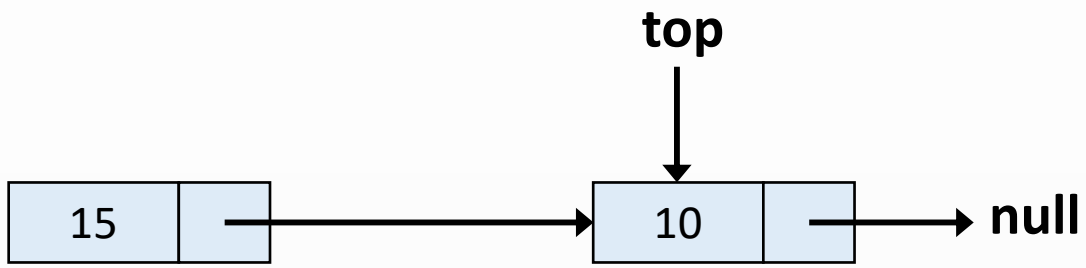


uzunluk = 1

sonuc = 15

pop()

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

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

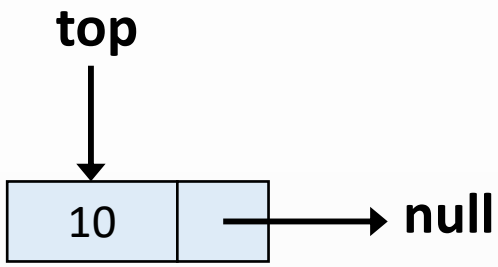


uzunluk = 1

sonuc = 15

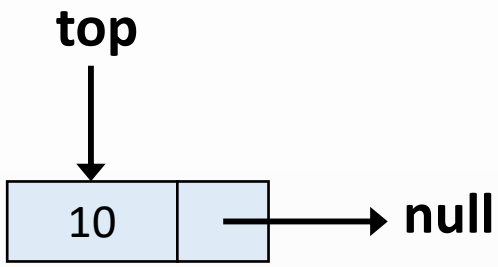
pop()

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

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



uzunluk = 1

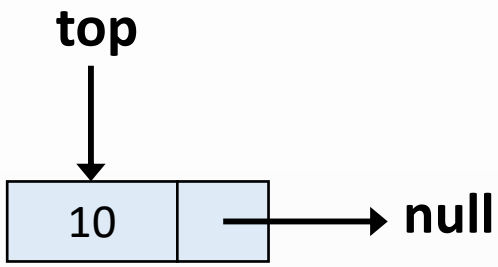
```
// Örnek değişkenleri
Dugum top;
int uzunluk;

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



uzunluk = 1

pop()

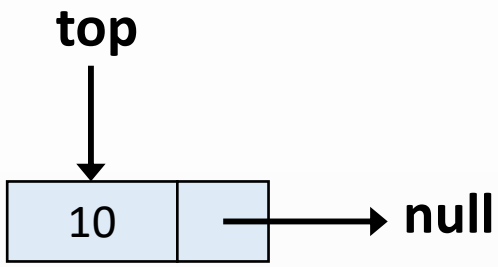
```
// Örnek değişkenleri
Dugum top;
int uzunluk;

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;
}
```

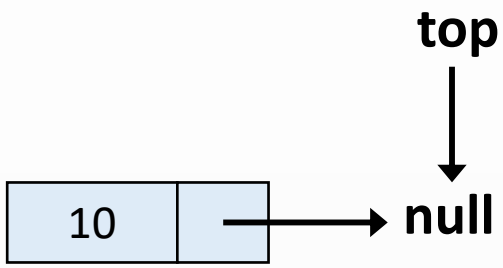


uzunluk = 1

sonuc = 10

pop()

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

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

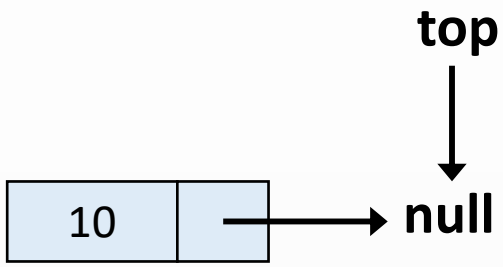


uzunluk = 1

sonuc = 10

pop()

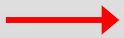
```
// Örnek değişkenleri
Dugum top;
int uzunluk;

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



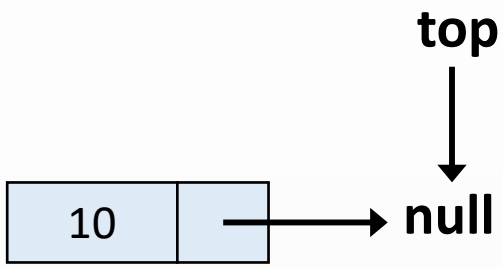
uzunluk = 0

sonuc = 10

pop()



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

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

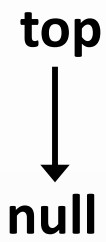


uzunluk = 0

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;
}
```

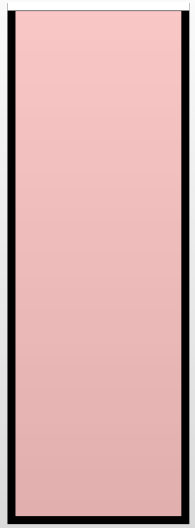
Karakter Dizisini (String) Tersine Çevirme

```
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";
```



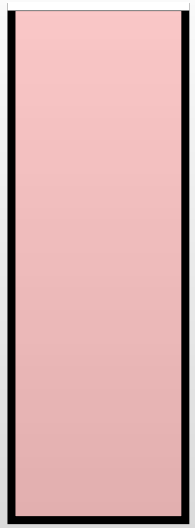
yigit

```
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

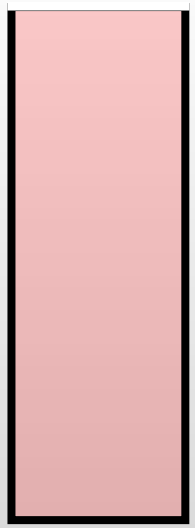
```
Stack<Character> yigit = new Stack<>();
char[] karakterler = str.toCharArray();
 for(char c : karakterler) {
    yigit.push(c);
 }
 for(int i = 0; i < str.length(); i++) {
    karakterler[i] = yigit.pop();
 }
 return new String(karakterler);
```

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

karakterler[]

A	B	C	D
0	1	2	3

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



yigit

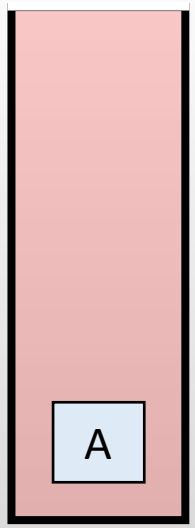
```
Stack<Character> yigit = new Stack<>();
char[] karakterler = str.toCharArray();
 for(char c : karakterler) {
    yigit.push(c);
 }
 for(int i = 0; i < str.length(); i++) {
    karakterler[i] = yigit.pop();
 }
 return new String(karakterler);
```

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

karakterler[]

A	B	C	D
0	1	2	3

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



yigit

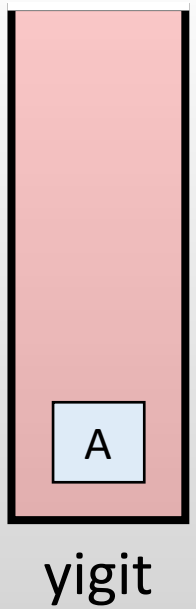
```
Stack<Character> yigit = new Stack<>();
char[] karakterler = str.toCharArray();
 for(char c : karakterler) {
    yigit.push(c);
 }
 for(int i = 0; i < str.length(); i++) {
    karakterler[i] = yigit.pop();
 }
 return new String(karakterler);
```

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

karakterler[]

A	B	C	D
0	1	2	3

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



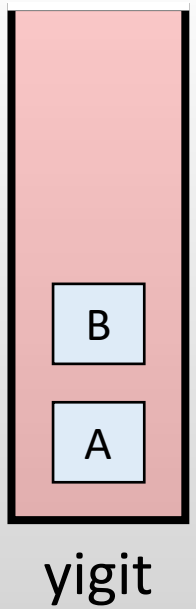

```
Stack<Character> yigit = new Stack<>();
char[] karakterler = str.toCharArray();
 for(char c : karakterler) {
    yigit.push(c);
 }
 for(int i = 0; i < str.length(); i++) {
    karakterler[i] = yigit.pop();
 }
 return new String(karakterler);
```

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

karakterler[]

A	B	C	D
0	1	2	3

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



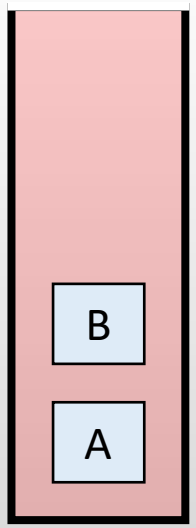

```
Stack<Character> yigit = new Stack<>();
char[] karakterler = str.toCharArray();
 for(char c : karakterler) {
    yigit.push(c);
 }
 for(int i = 0; i < str.length(); i++) {
    karakterler[i] = yigit.pop();
 }
 return new String(karakterler);
```

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

karakterler[]

A	B	C	D
0	1	2	3

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



yigit

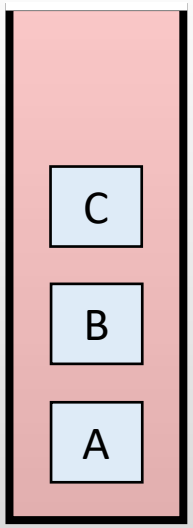
```
Stack<Character> yigit = new Stack<>();
char[] karakterler = str.toCharArray();
 for(char c : karakterler) {
    yigit.push(c);
 }
 for(int i = 0; i < str.length(); i++) {
    karakterler[i] = yigit.pop();
 }
 return new String(karakterler);
```

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

karakterler[]

A	B	C	D
0	1	2	3

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



yigit

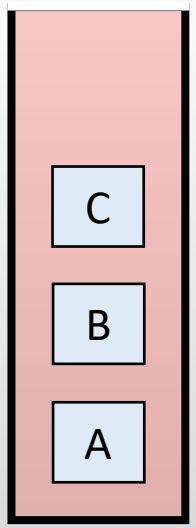
```
Stack<Character> yigit = new Stack<>();
char[] karakterler = str.toCharArray();
 for(char c : karakterler) {
    yigit.push(c);
 }
 for(int i = 0; i < str.length(); i++) {
    karakterler[i] = yigit.pop();
 }
 return new String(karakterler);
```

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

karakterler[]

A	B	C	D
0	1	2	3

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



yigit

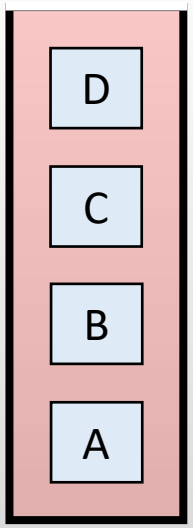
```
Stack<Character> yigit = new Stack<>();
char[] karakterler = str.toCharArray();
 for(char c : karakterler) {
    yigit.push(c);
 }
 for(int i = 0; i < str.length(); i++) {
    karakterler[i] = yigit.pop();
 }
 return new String(karakterler);
```

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

karakterler[]

A	B	C	D
0	1	2	3

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



yigit

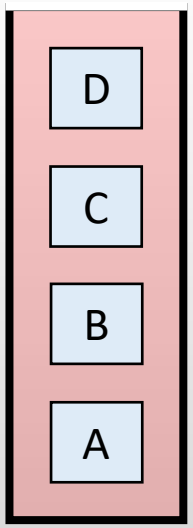
```
Stack<Character> yigit = new Stack<>();
char[] karakterler = str.toCharArray();
 for(char c : karakterler) {
    yigit.push(c);
 }
 for(int i = 0; i < str.length(); i++) {
    karakterler[i] = yigit.pop();
 }
 return new String(karakterler);
```

```
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);
```

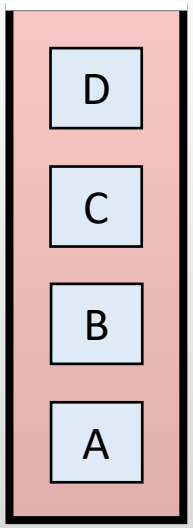
```
String str = "ABCD";
```

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

karakterler[]

A	B	C	D
0	1	2	3

```
i = 0
```



yigit

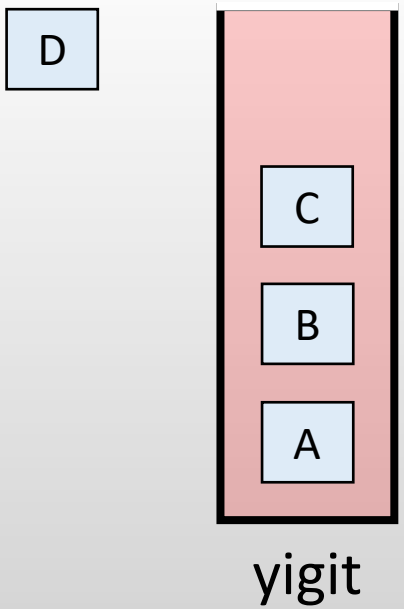
```
Stack<Character> yigit = new Stack<>();
char[] karakterler = str.toCharArray();
 for(char c : karakterler) {
    yigit.push(c);
 }
 for(int i = 0; i < str.length(); i++) {
    karakterler[i] = yigit.pop();
 }
 return new String(karakterler);
```

```
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);
```

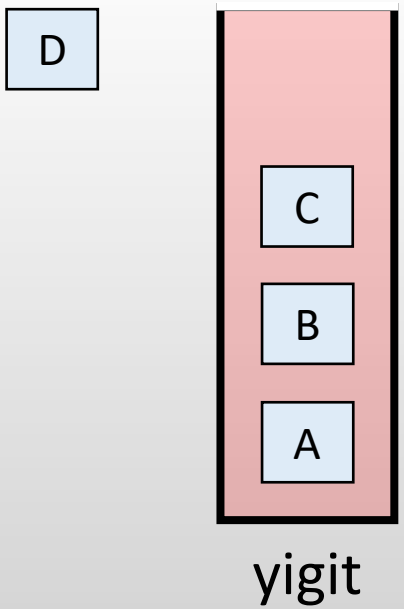
```
String str = "ABCD";
```

```
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);
```

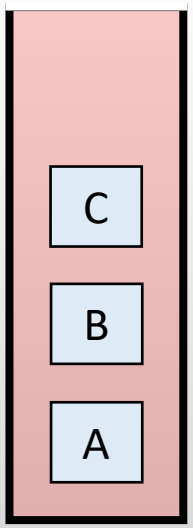
```
String str = "ABCD";
```

```
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);
```

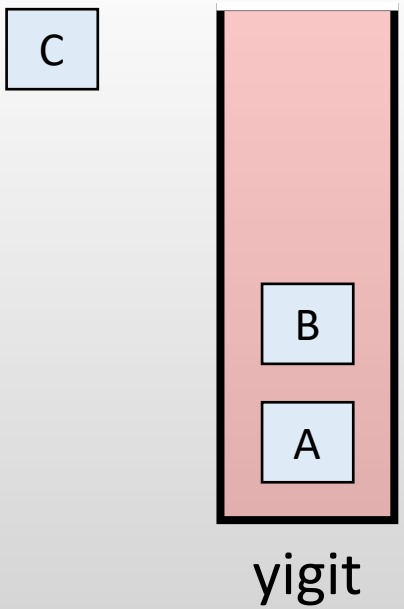
```
String str = "ABCD";
```

```
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);
```

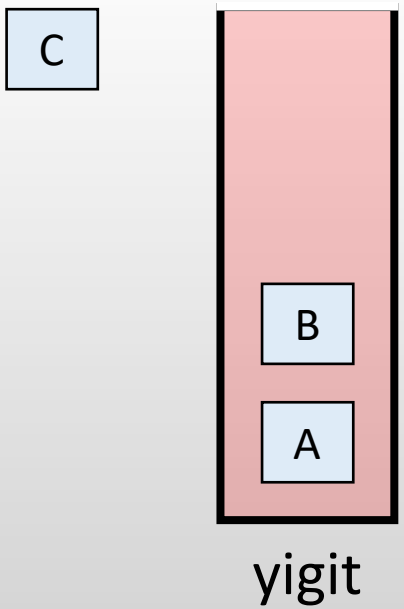
```
String str = "ABCD";
```

```
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);
```

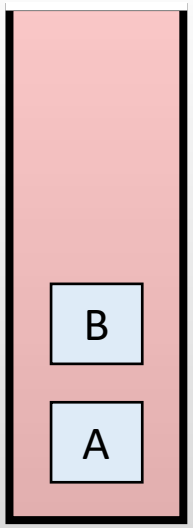
```
String str = "ABCD";
```

```
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);
```

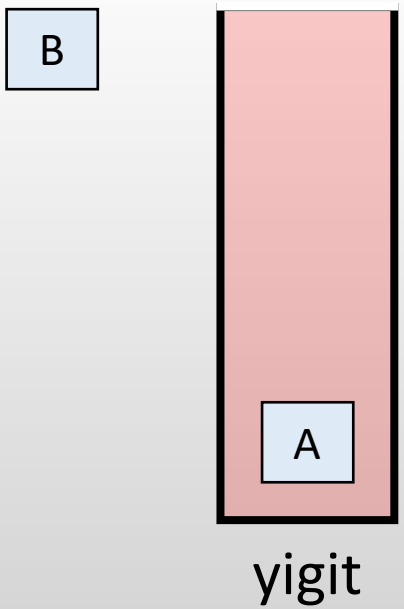
```
String str = "ABCD";
```

```
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);
```

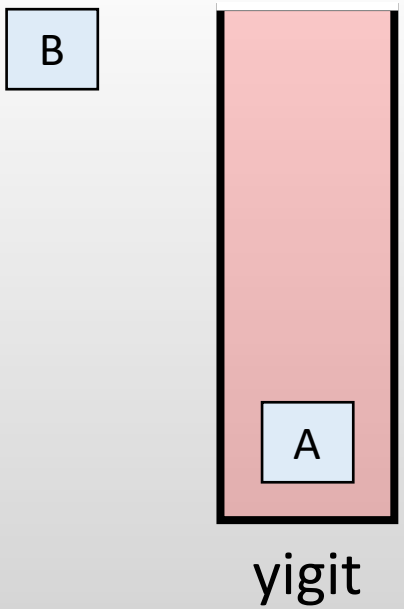
```
String str = "ABCD";
```

```
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);
```

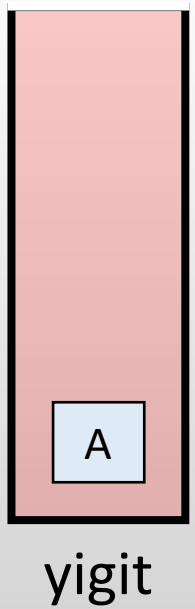
```
String str = "ABCD";
```

```
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);
```

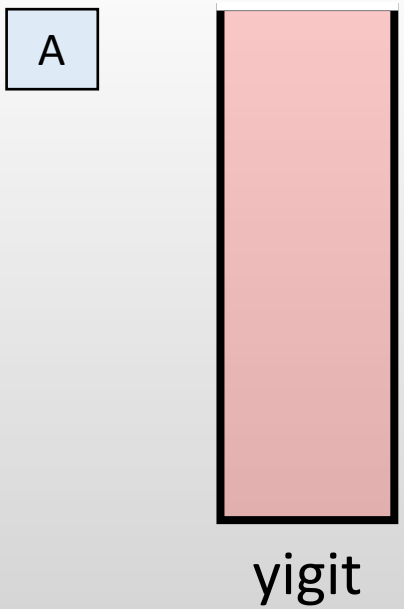
```
String str = "ABCD";
```

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

karakterler[]

D	C	B	D
0	1	2	3

```
i = 3
```



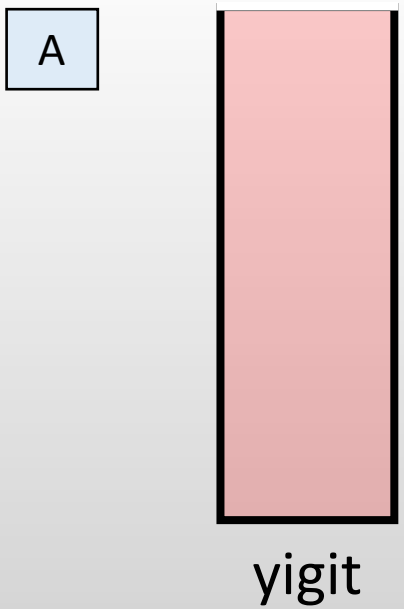

```
Stack<Character> yigit = new Stack<>();
char[] karakterler = str.toCharArray();
 for(char c : karakterler) {
    yigit.push(c);
 }
 for(int i = 0; i < str.length(); i++) {
    karakterler[i] = yigit.pop();
 }
 return new String(karakterler);
```

```
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);
```

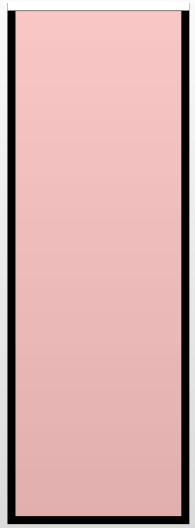
```
String str = "ABCD";
```

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

karakterler[]

D	C	B	A
0	1	2	3

```
i = 4
```



yigit

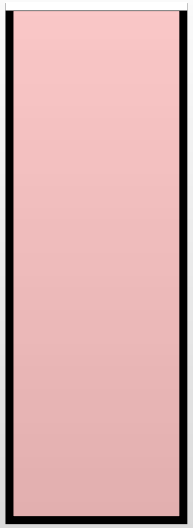
```
Stack<Character> yigit = new Stack<>();
char[] karakterler = str.toCharArray();
 for(char c : karakterler) {
    yigit.push(c);
 }
 for(int i = 0; i < str.length(); i++) {
    karakterler[i] = yigit.pop();
 }
 return new String(karakterler);
```

```
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 dizinin her elemanı için dizinin içinde bu elemandan daha büyük olan bir sonraki elemanı bulma problemini ifade eder.
- Bu problemin temel amacı, her elemanın bir sonraki büyük elemanını belirlemektir.
- **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

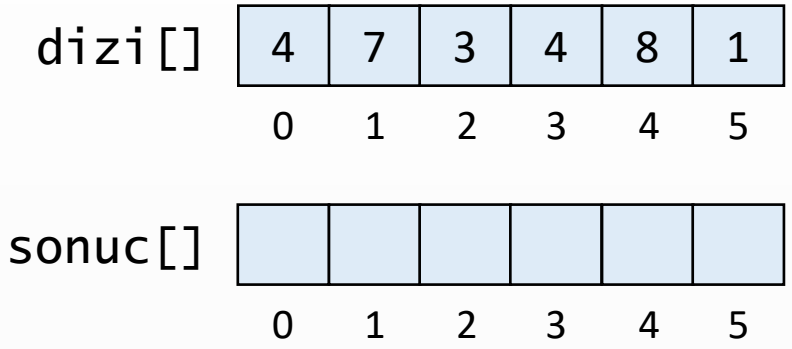
```
int[] sonrakiBuyukEleman(int[] dizi) {
   int[] sonuc = new int[dizi.length];
   Stack<Integer> yigit = new Stack<>();
   for(int i = dizi.length - 1; i >= 0; i--) {
      if(!yigit.isEmpty()) {
         while(!yigit.isEmpty() &&
               yigit.peek() <= dizi[i]) {
            yigit.pop();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

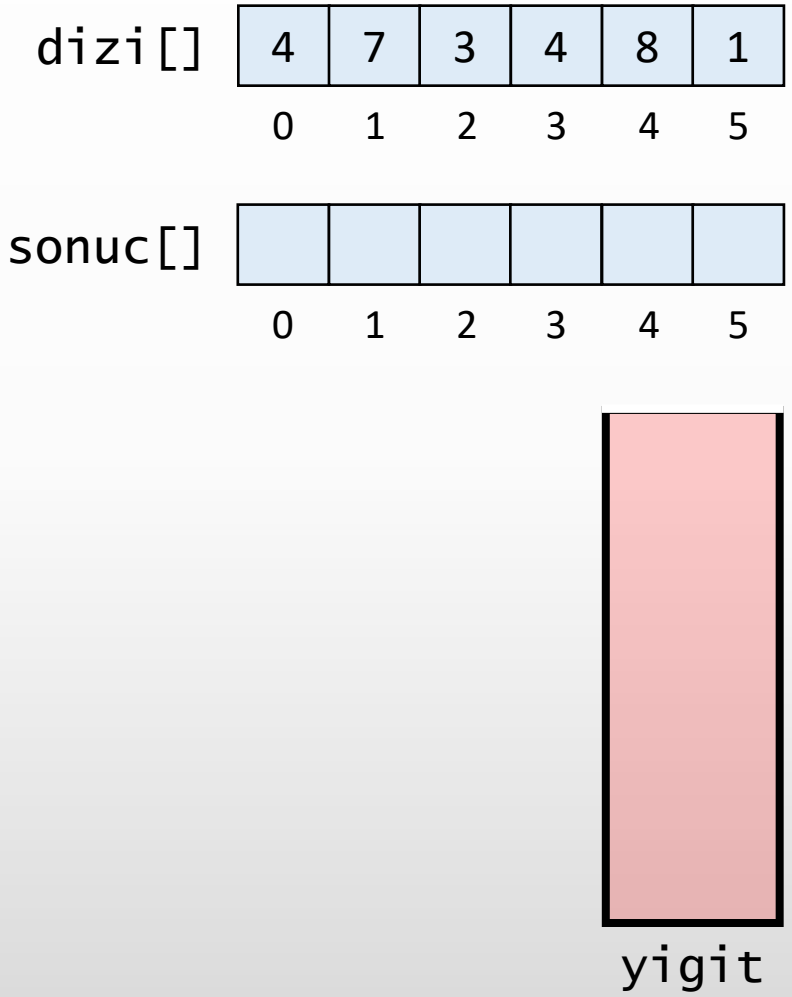
```
int[] sonrakiBuyukEleman(int[] dizi) {
   int[] sonuc = new int[dizi.length];
   Stack<Integer> yigit = new Stack<>();
   for(int i = dizi.length - 1; i >= 0; i--) {
      if(!yigit.isEmpty()) {
         while(!yigit.isEmpty() &&
               yigit.peek() <= dizi[i]) {
            yigit.pop();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

```
sonrakiBuyukEleman(dizi);
```



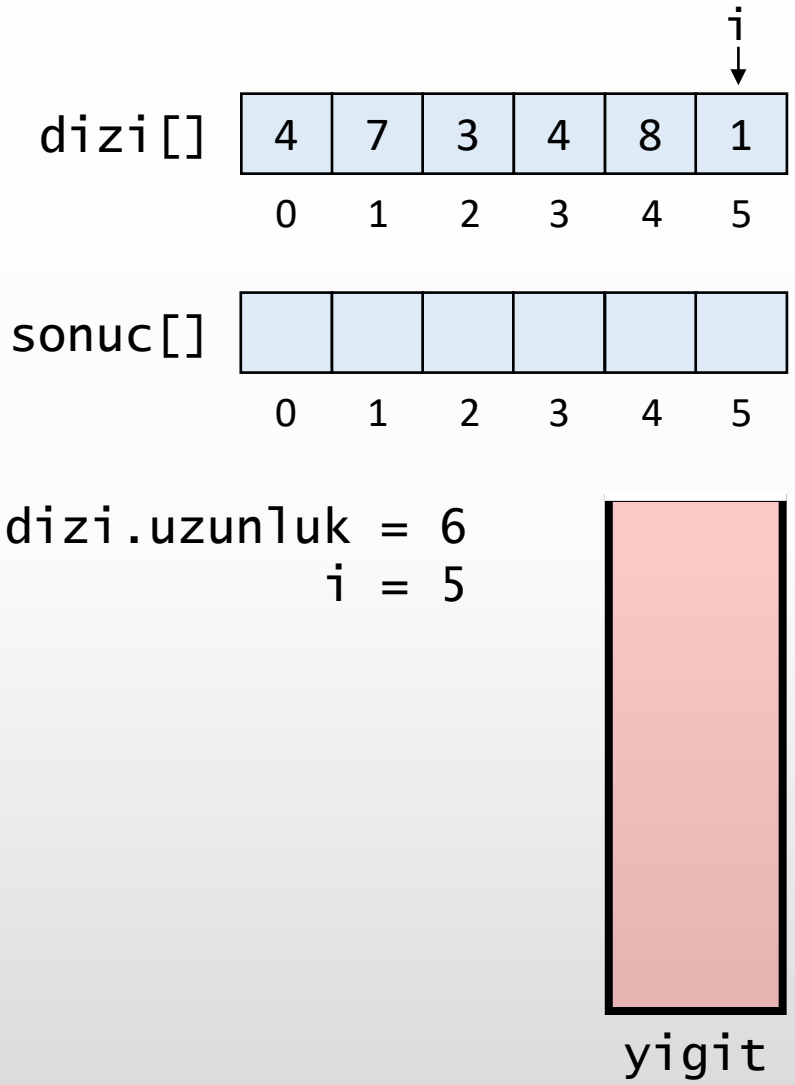

```
sonrakiBuyukEleman(dizi);
```

```
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;
}
```



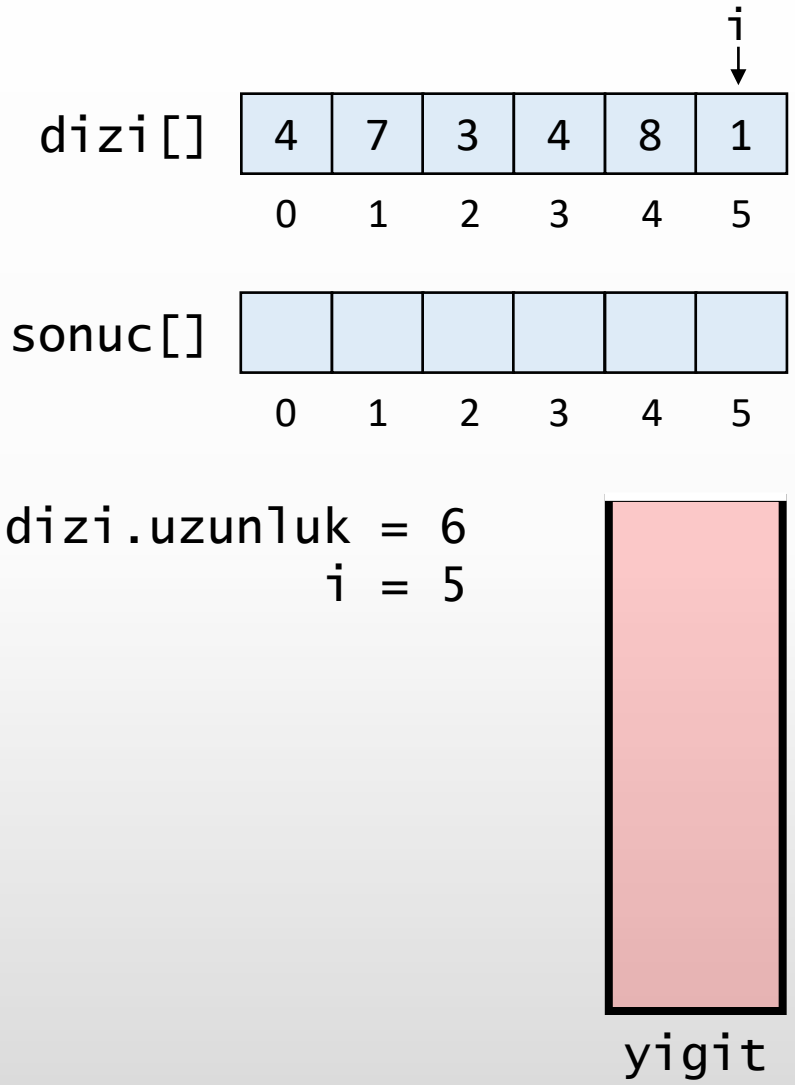

```
sonrakiBuyukEleman(dizi);
```

```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;
}
```



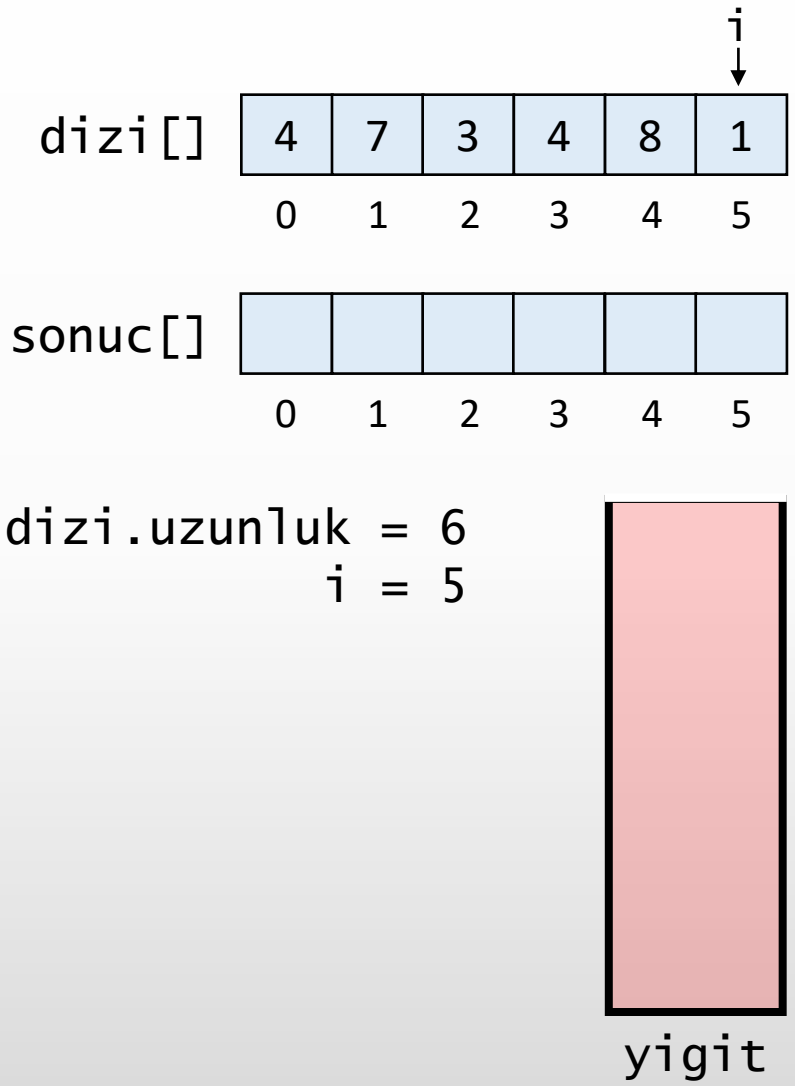

```
sonrakiBuyukEleman(dizi);
```

```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;
}
```



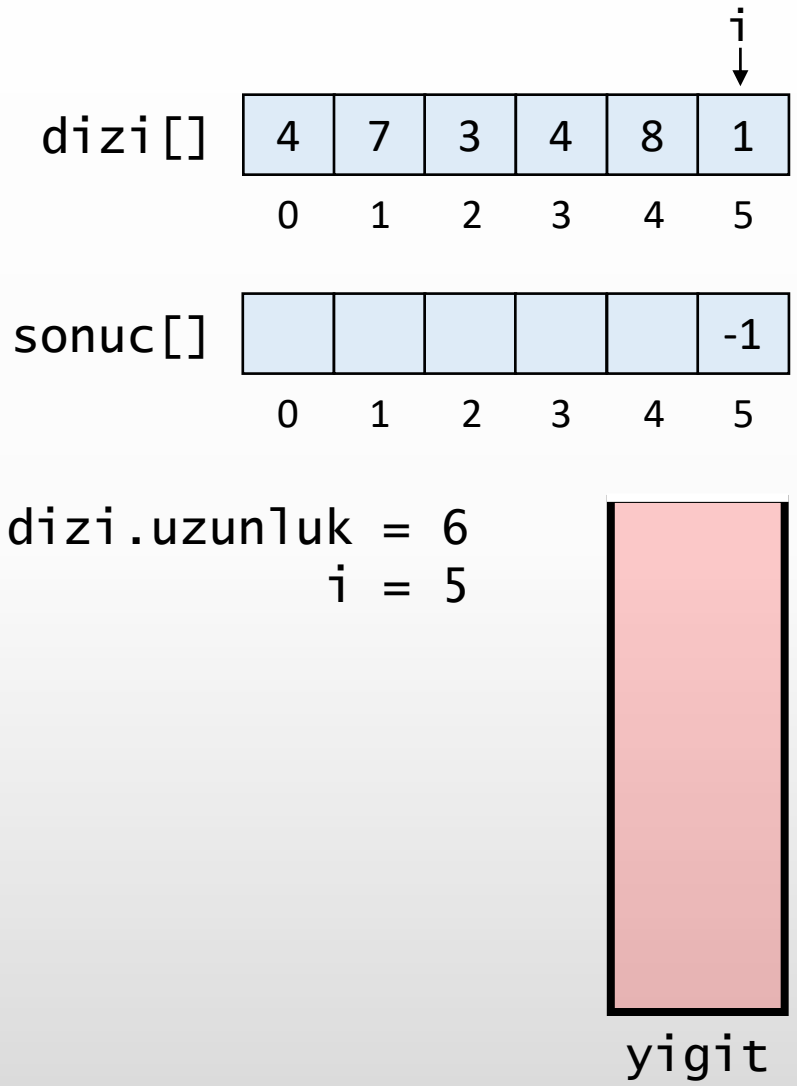

```
sonrakiBuyukEleman(dizi);
```

```
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;
}
```



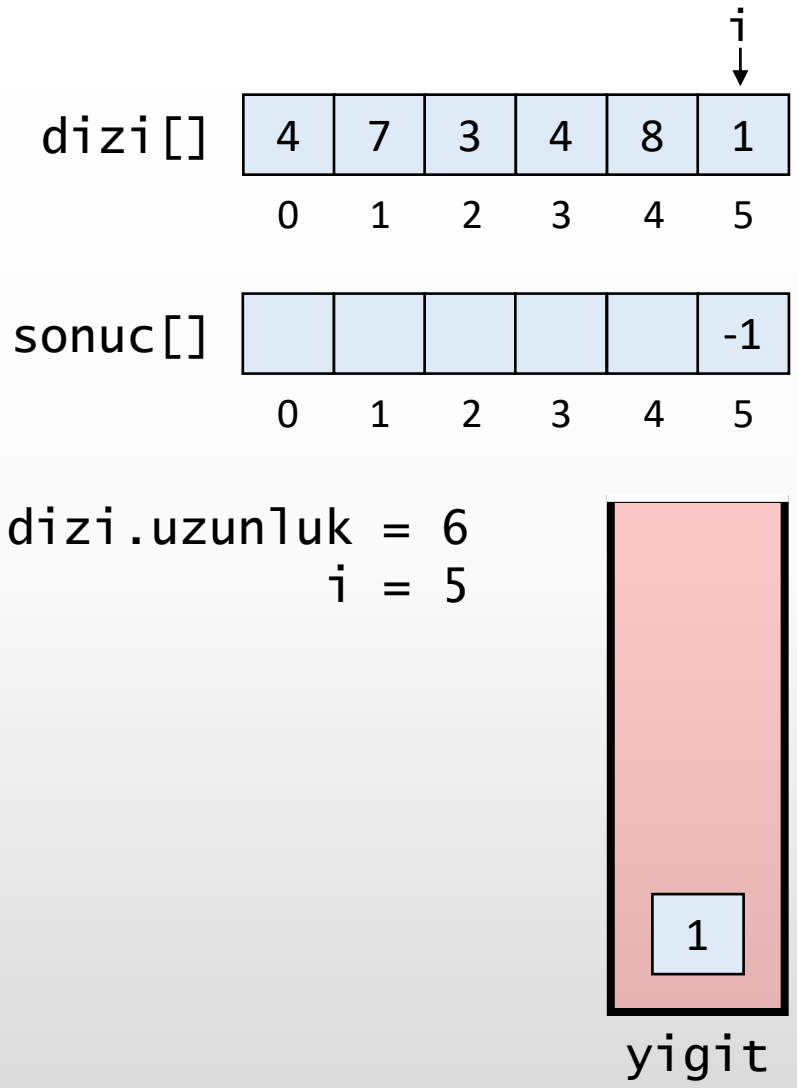

```
sonrakiBuyukEleman(dizi);
```

```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;
}
```



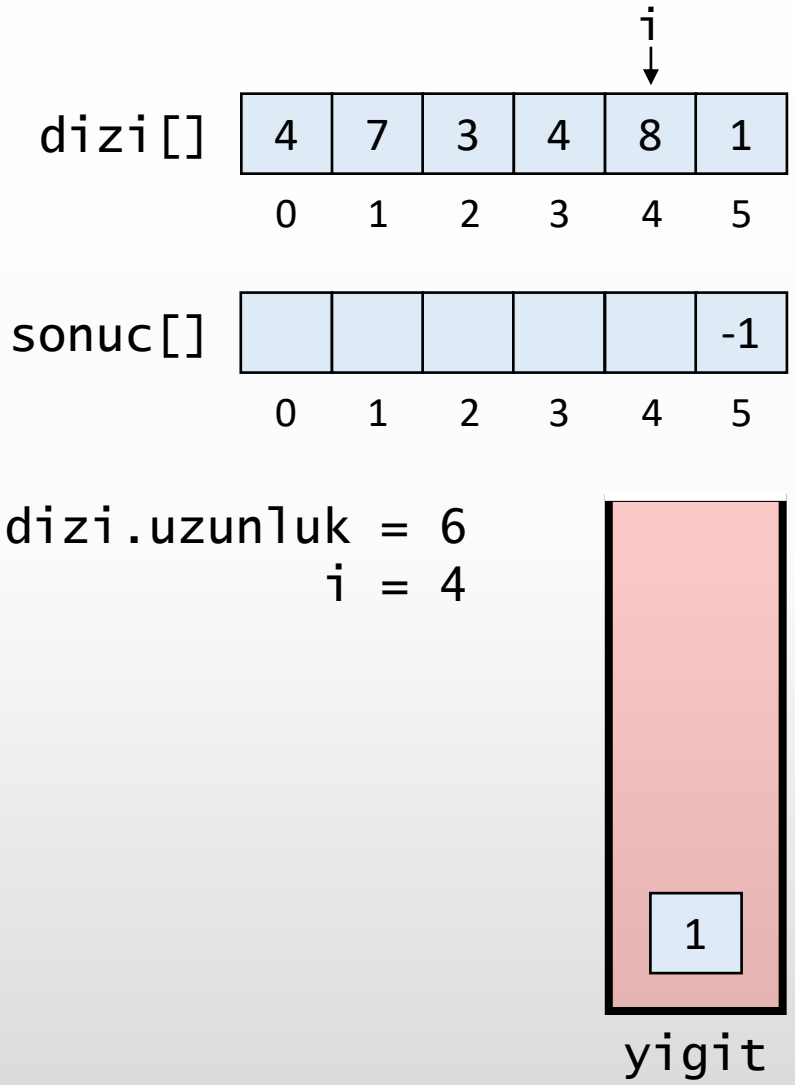
sonrakiBuyukEleman(dizi);

```
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;
}
```



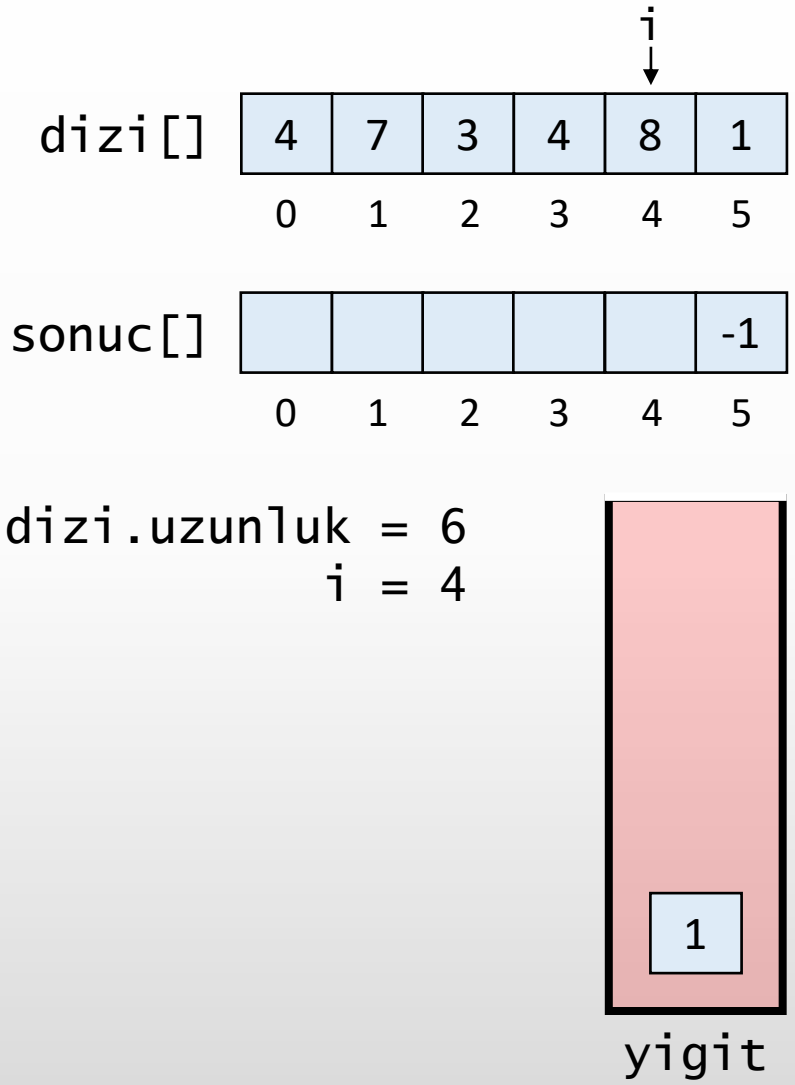
sonrakiBuyukEleman(dizi);

```
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;
}
```



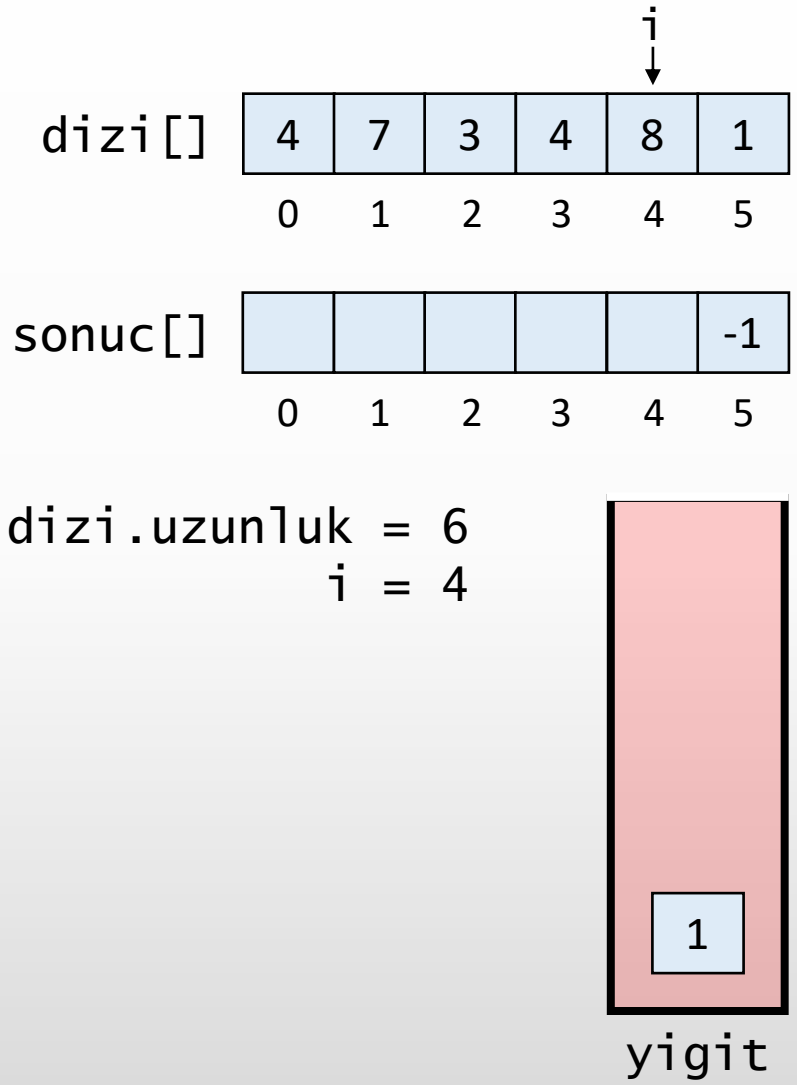

```
sonrakiBuyukEleman(dizi);
```

```
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;
}
```



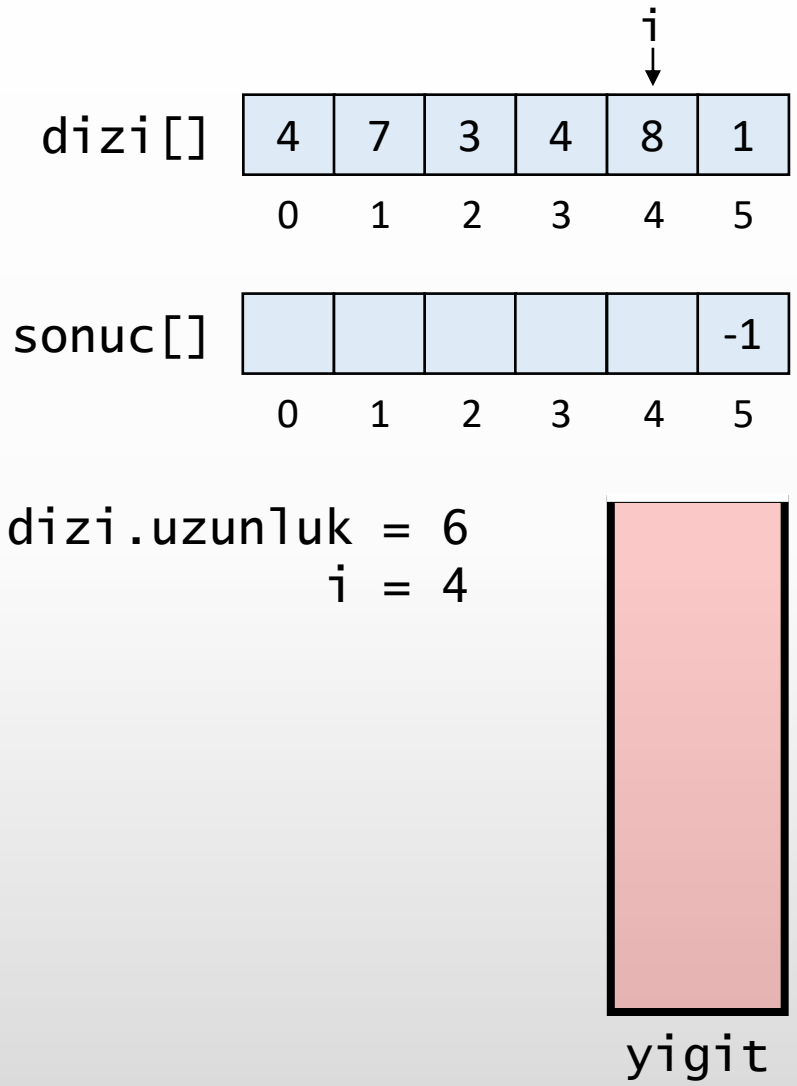

```
sonrakiBuyukEleman(dizi);
```

```
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;
}
```



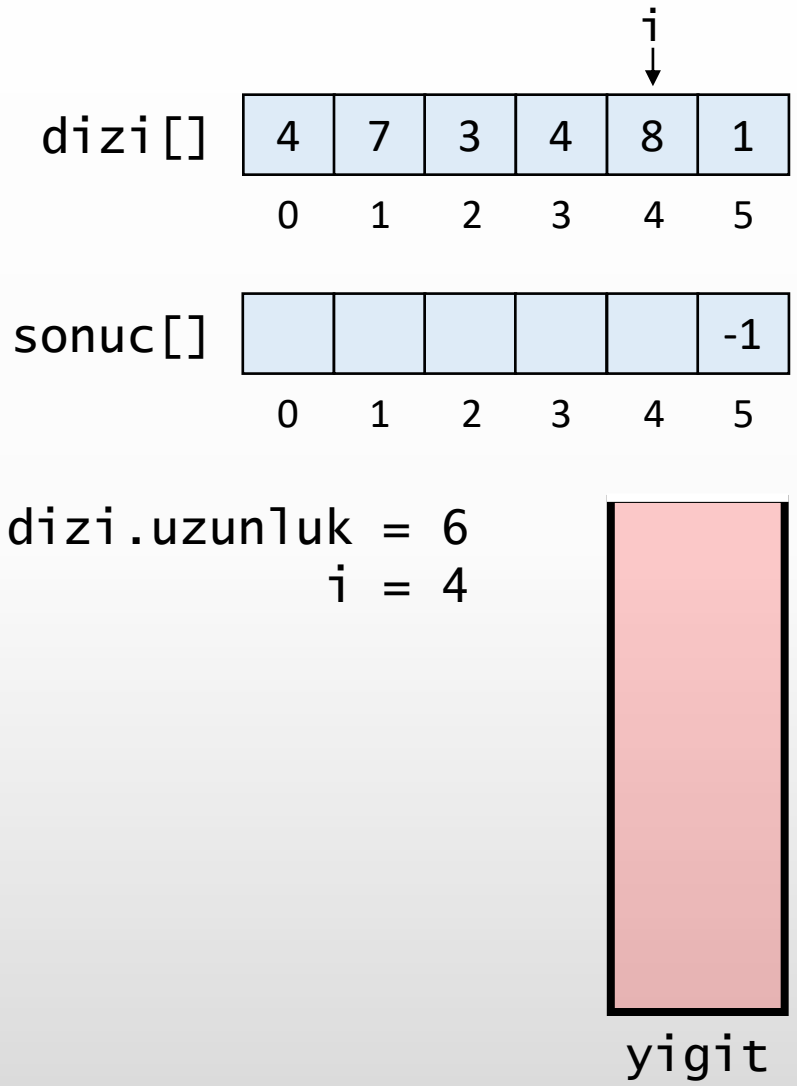

```
sonrakiBuyukEleman(dizi);
```

```
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;
}
```



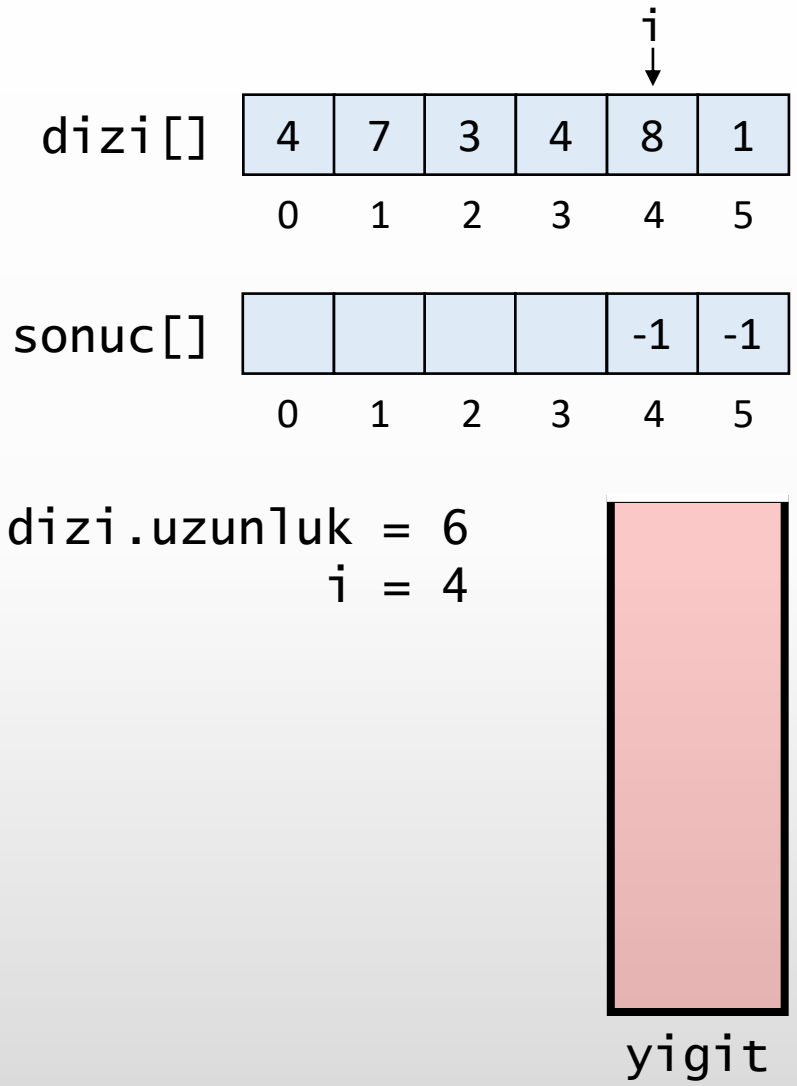

```
sonrakiBuyukEleman(dizi);
```

```
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;
}
```



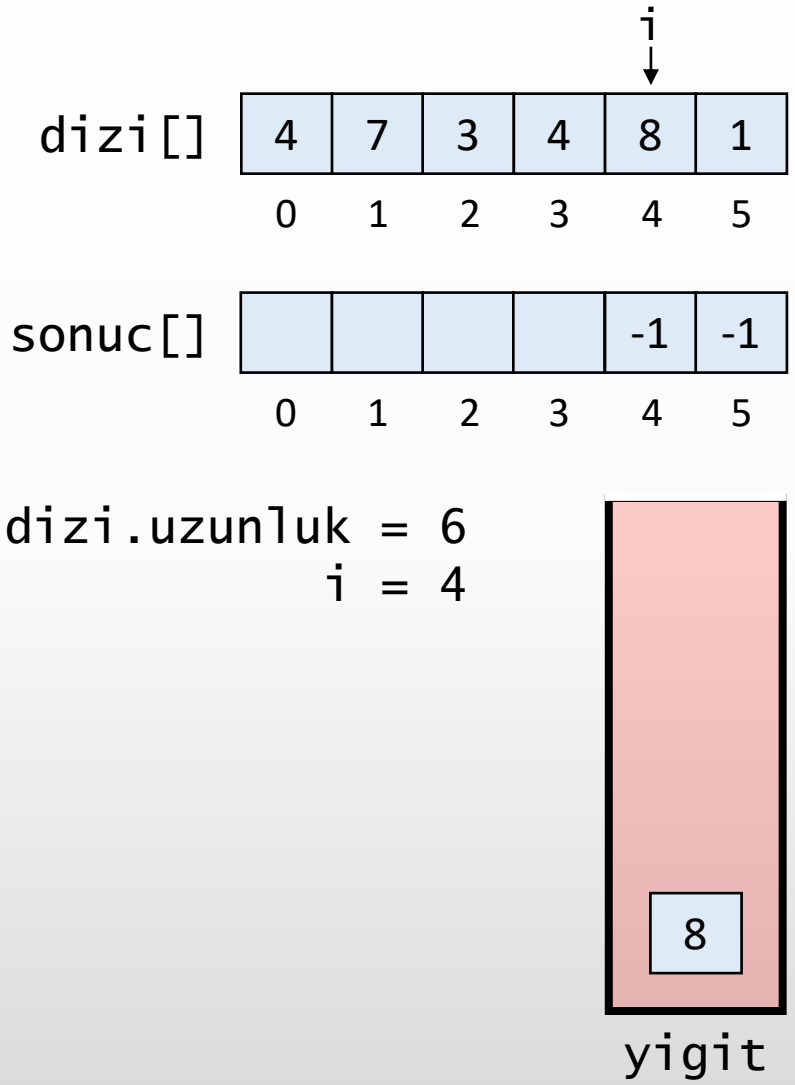

```
sonrakiBuyukEleman(dizi);
```

```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;
}
```



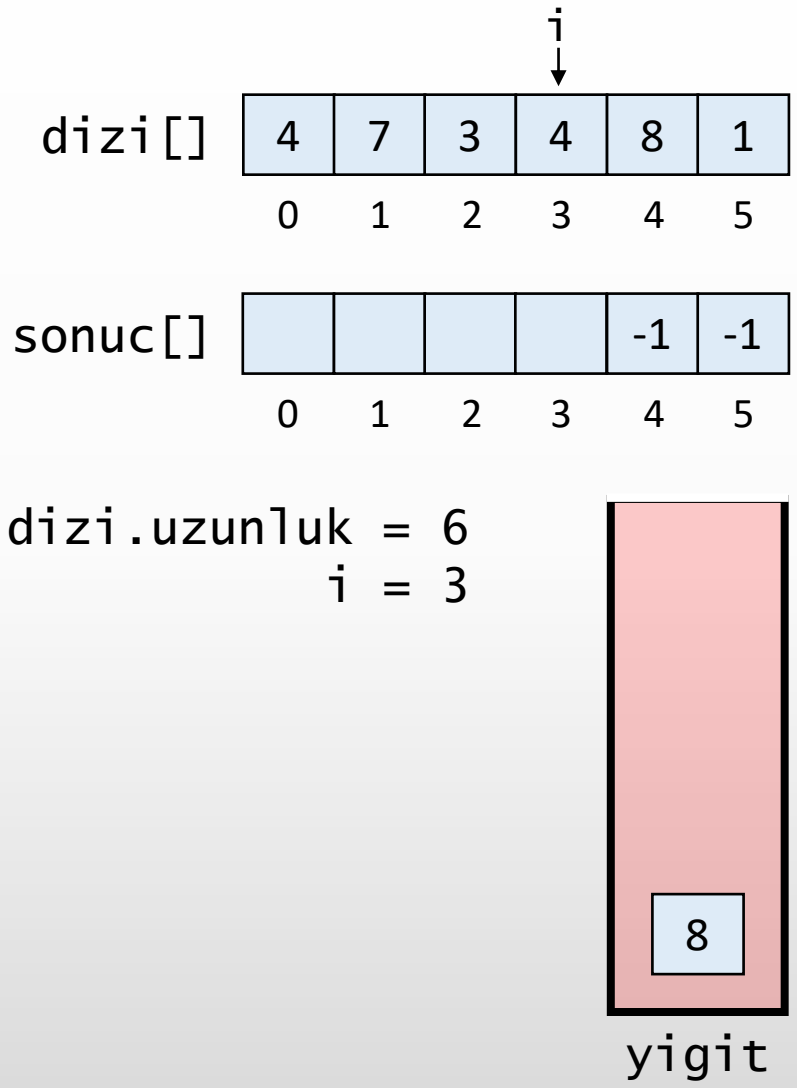

```
sonrakiBuyukEleman(dizi);
```

```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;
}
```



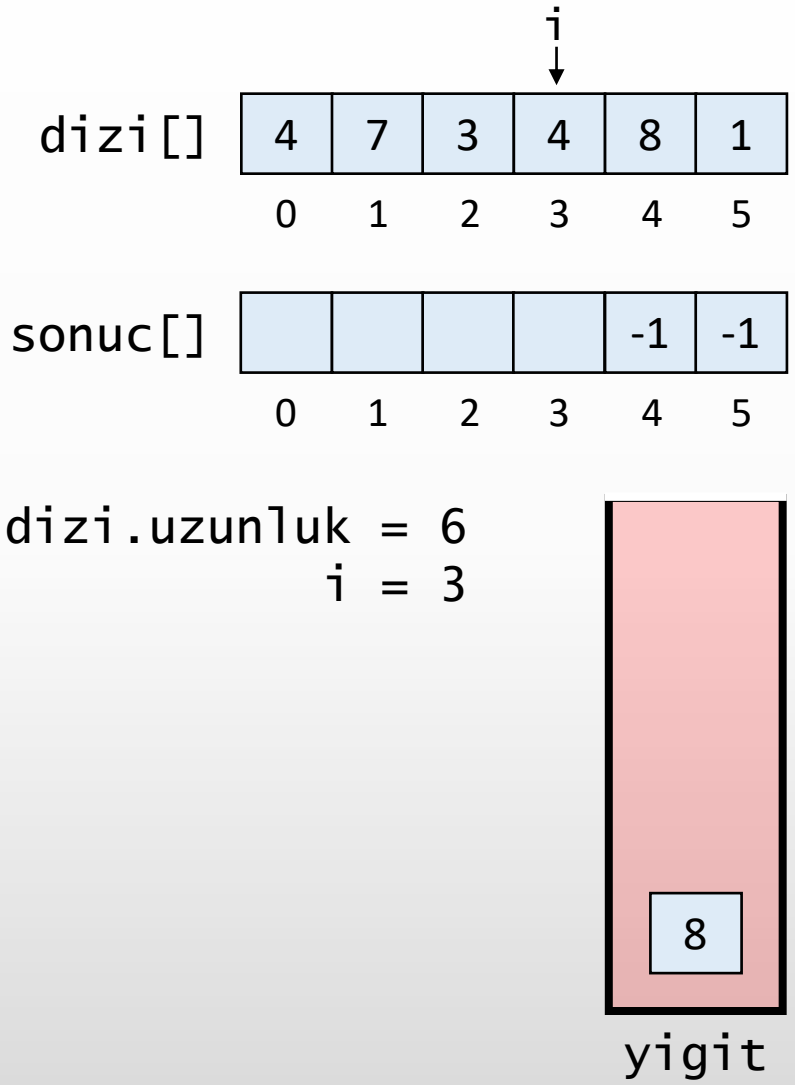

```
sonrakiBuyukEleman(dizi);
```

```
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;
}
```



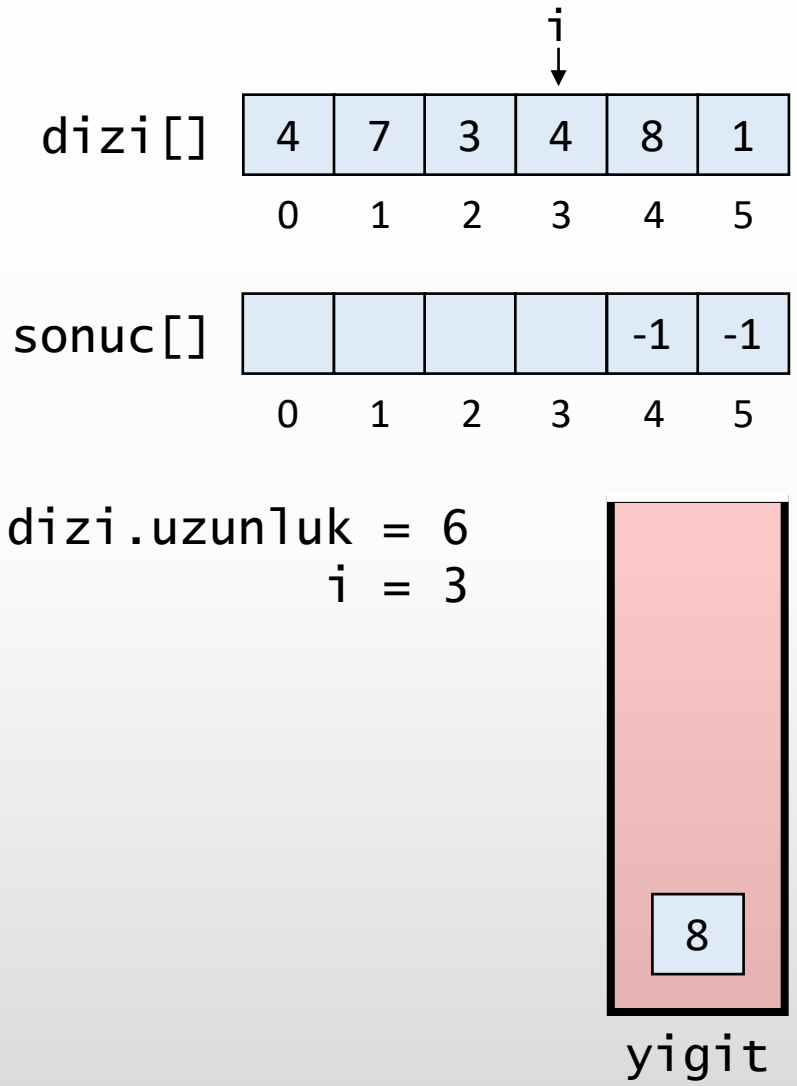

```
sonrakiBuyukEleman(dizi);
```

```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;
}
```



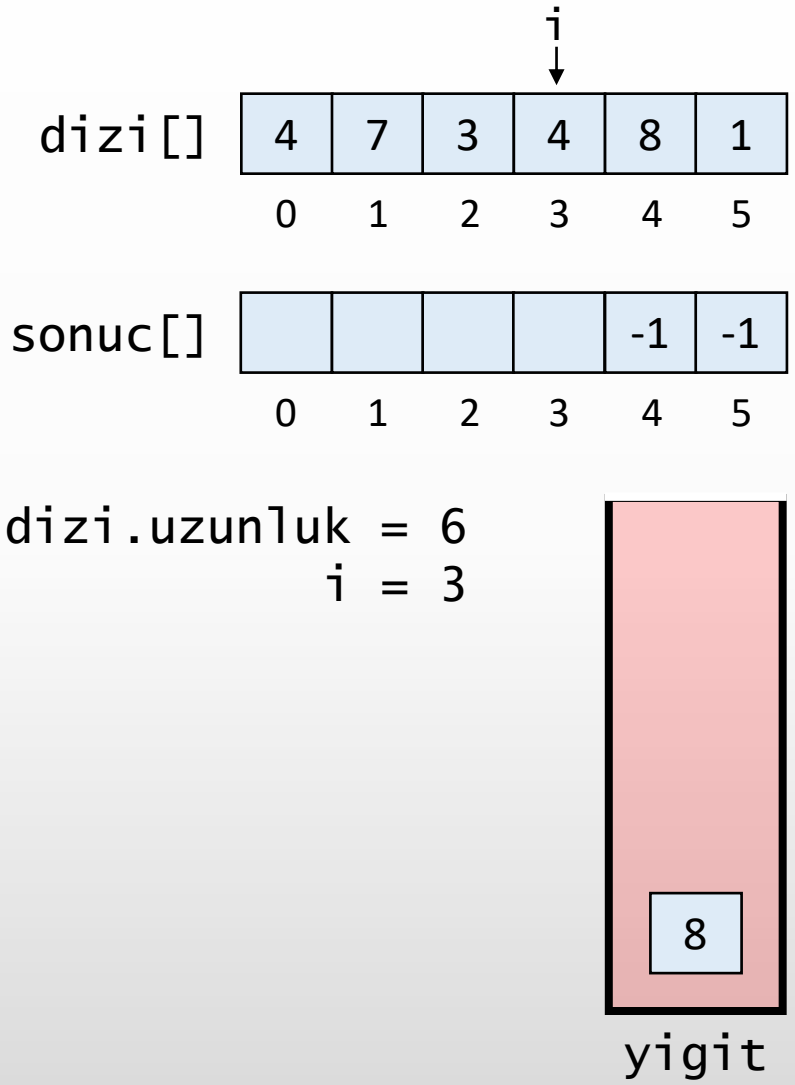

```
sonrakiBuyukEleman(dizi);
```

```
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;
}
```



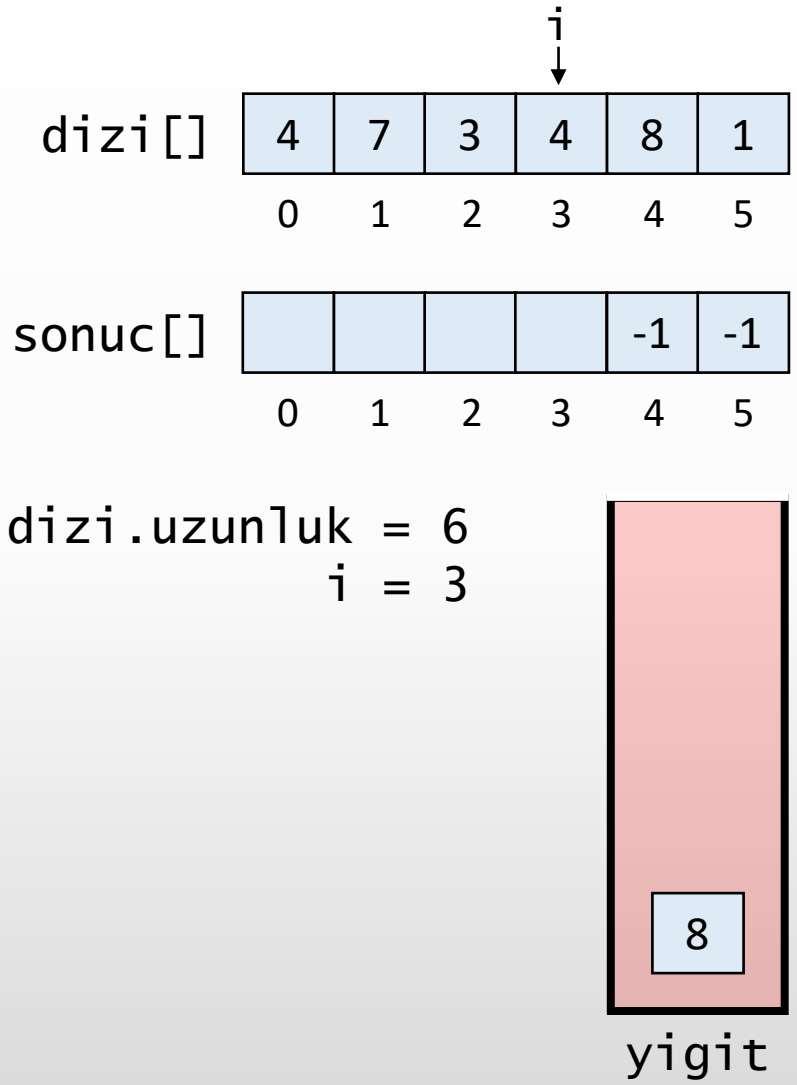

```
sonrakiBuyukEleman(dizi);
```

```
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;
}
```



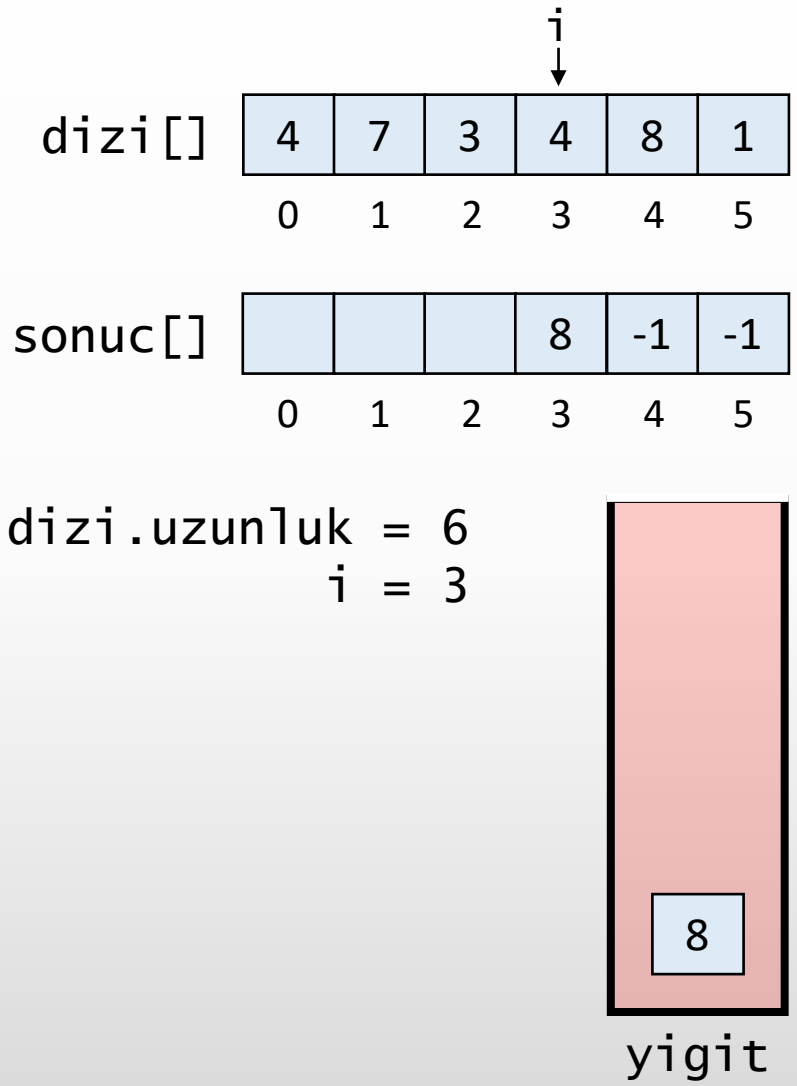
sonrakiBuyukEleman(dizi);

```
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;
}
```



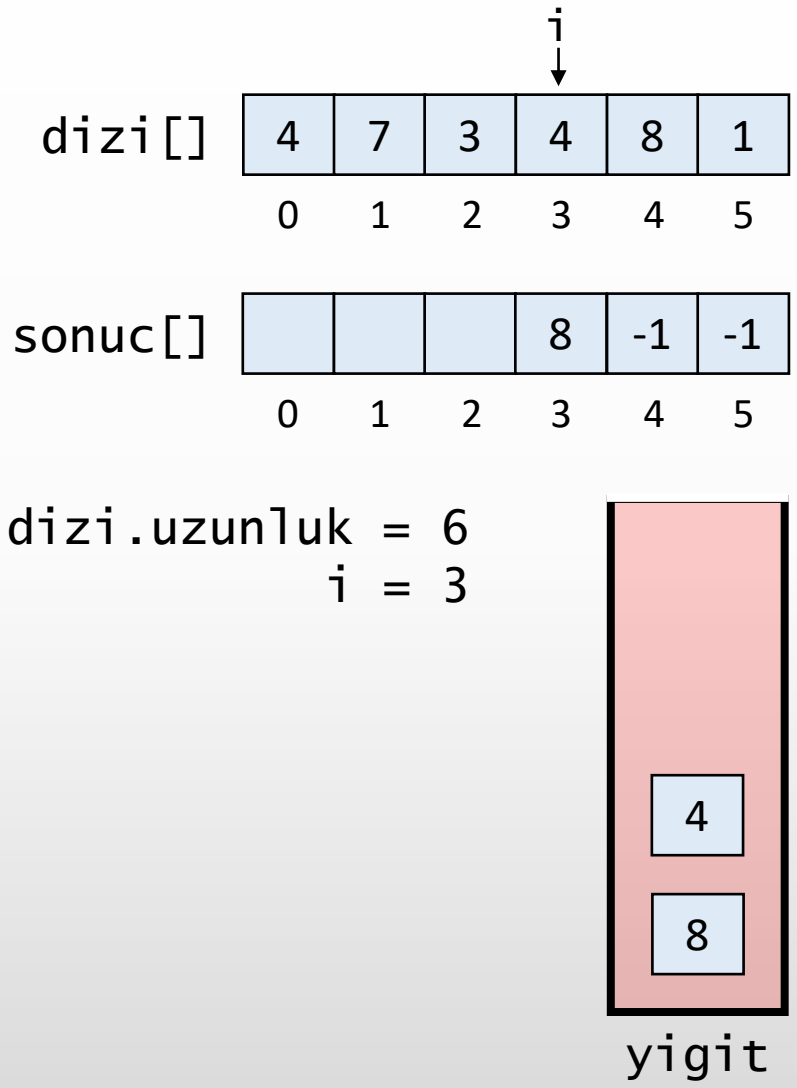

```
sonrakiBuyukEleman(dizi);
```

```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;
}
```



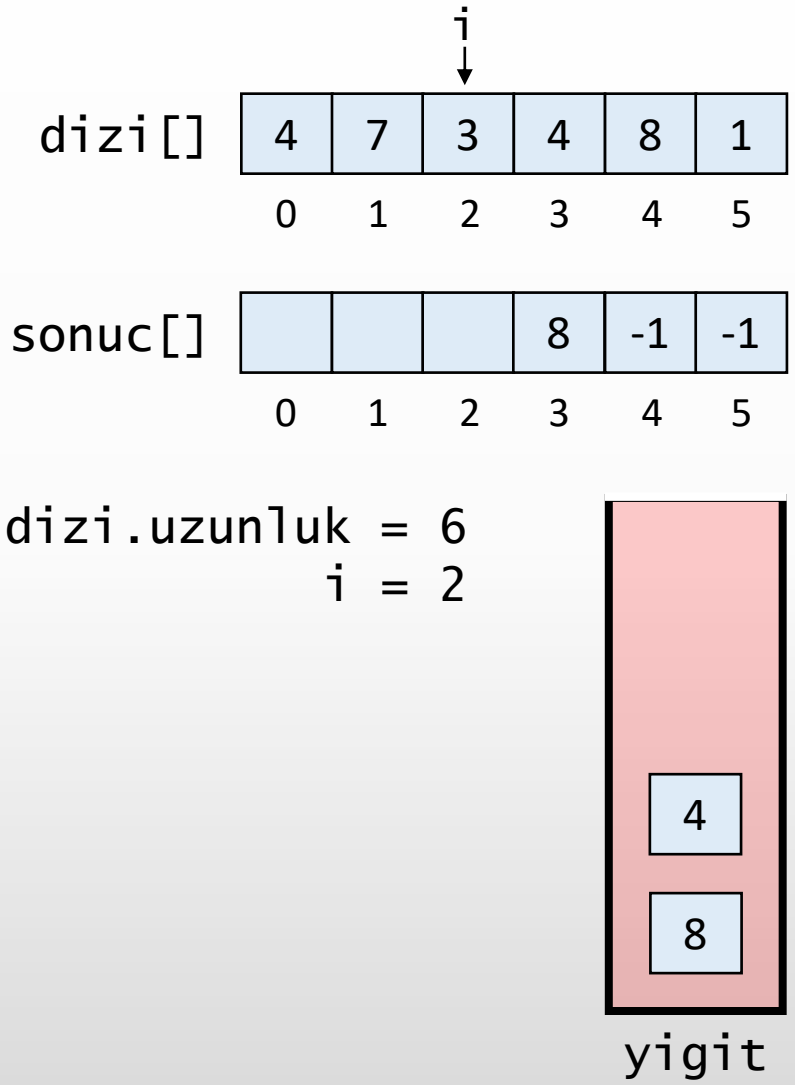
sonrakiBuyukEleman(dizi);

```
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;
}
```



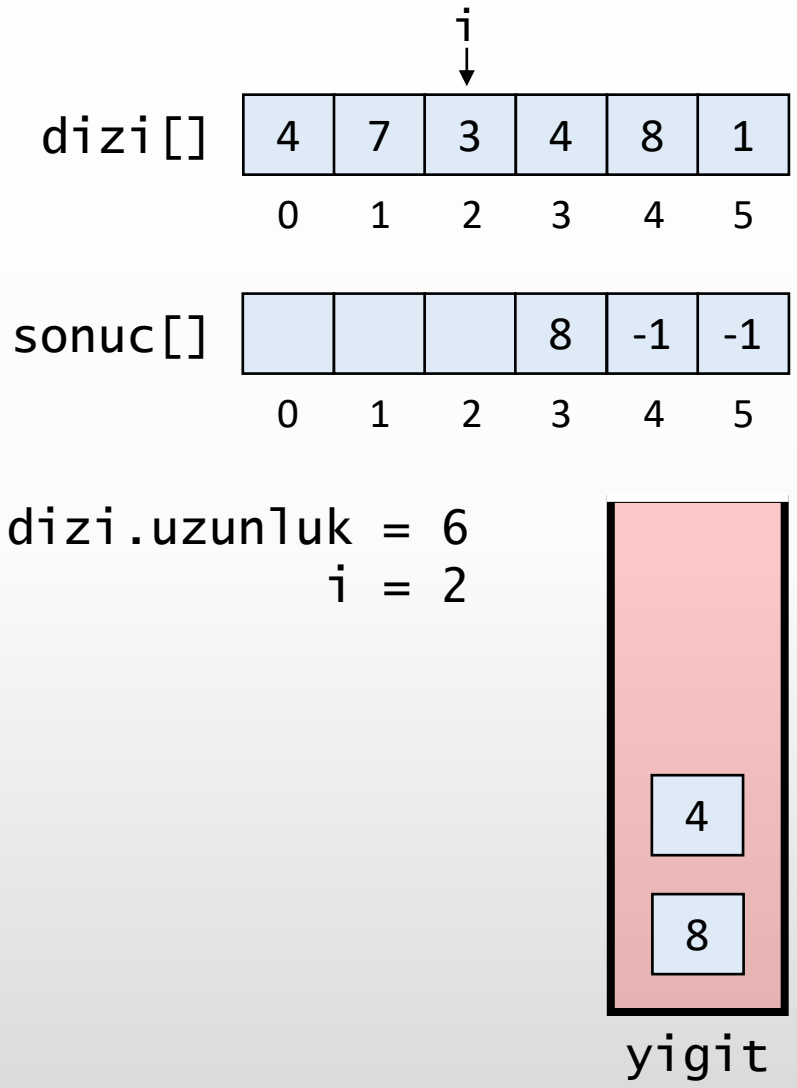
sonrakiBuyukEleman(dizi);

```
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;
}
```



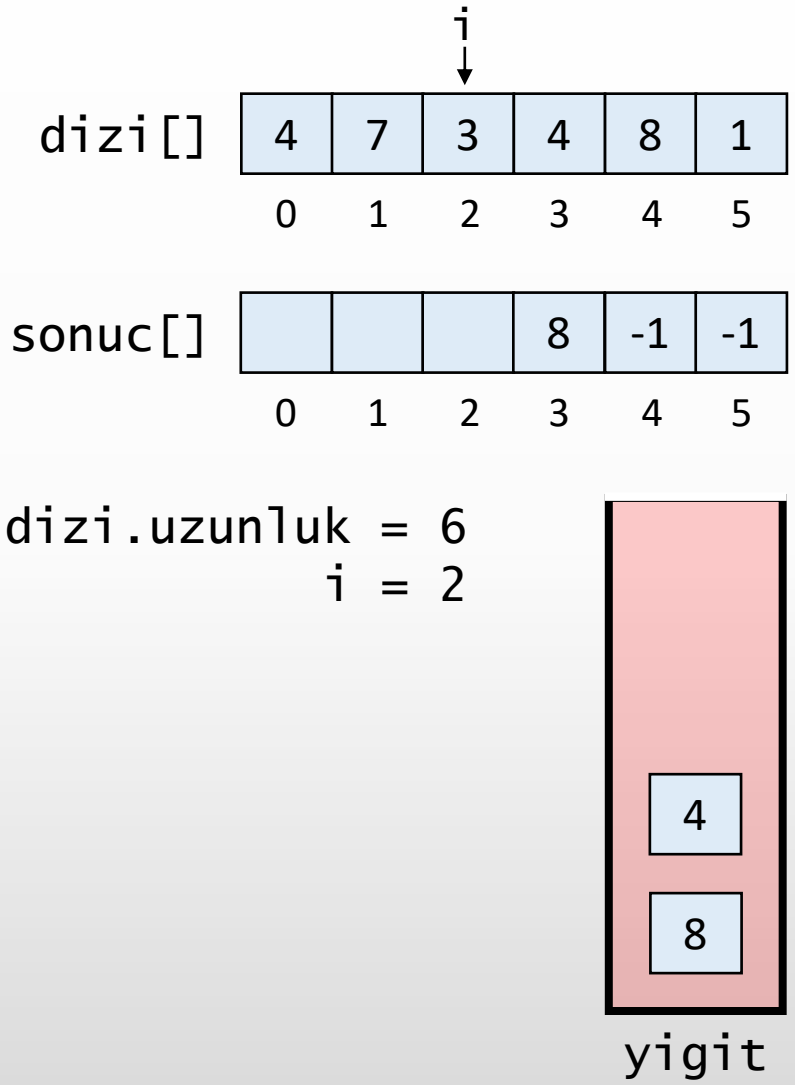

```
sonrakiBuyukEleman(dizi);
```

```
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;
}
```



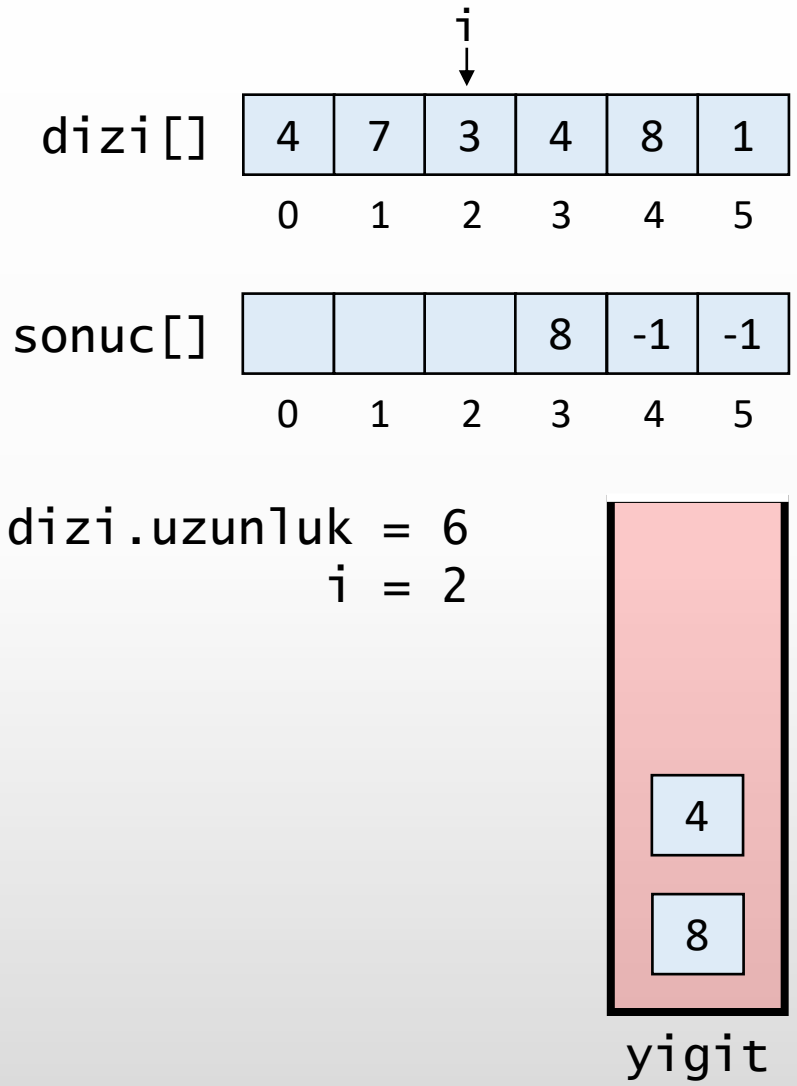

```
sonrakiBuyukEleman(dizi);
```

```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;
}
```



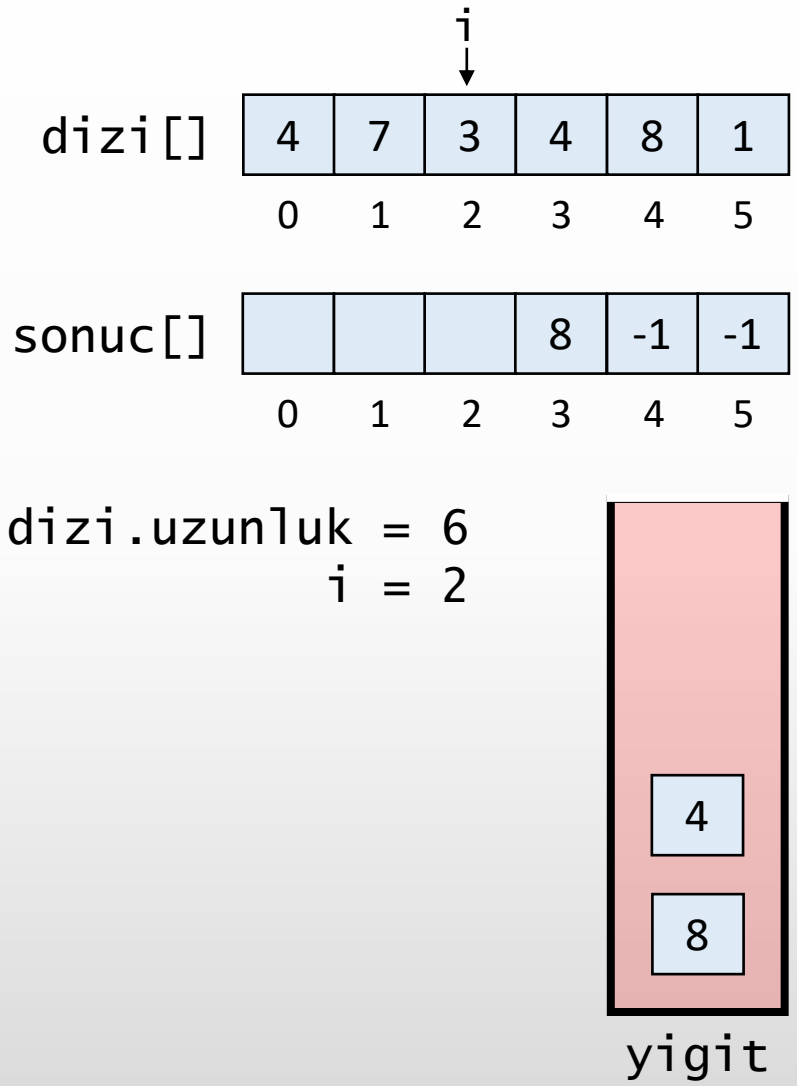

```
sonrakiBuyukEleman(dizi);
```

```
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;
}
```



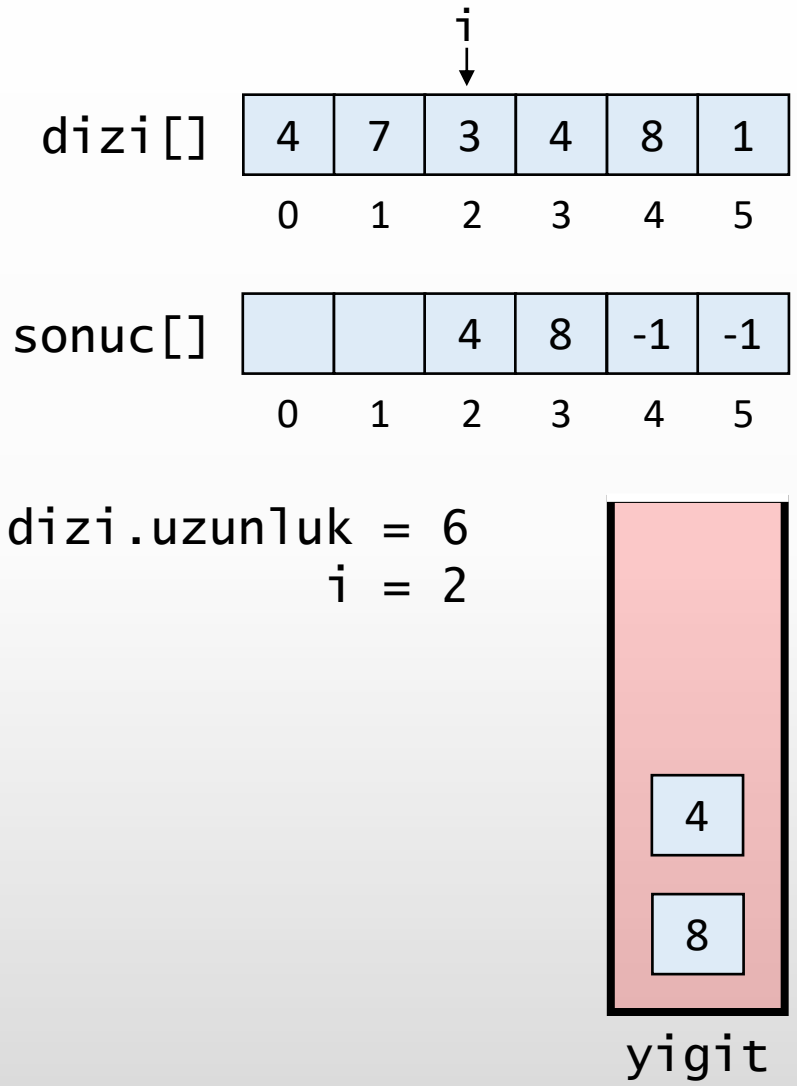
sonrakiBuyukEleman(dizi);

```
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;
}
```



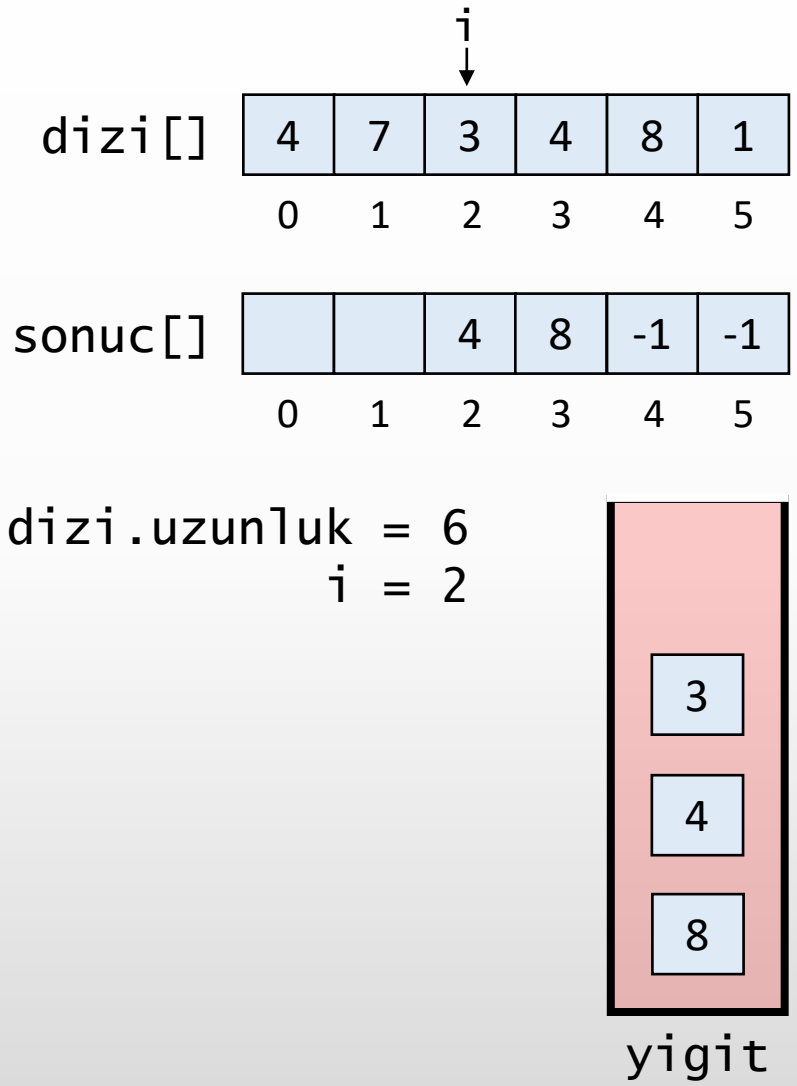
sonrakiBuyukEleman(dizi);

```
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;
}
```



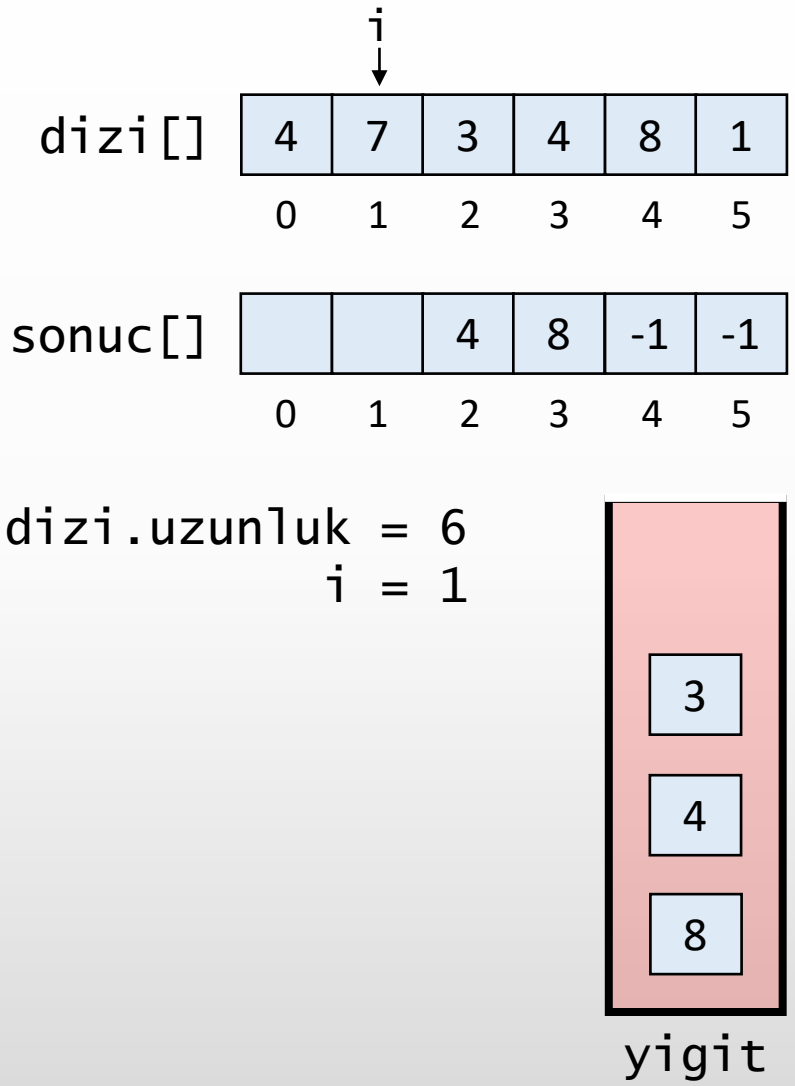

```
sonrakiBuyukEleman(dizi);
```

```
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;
}
```



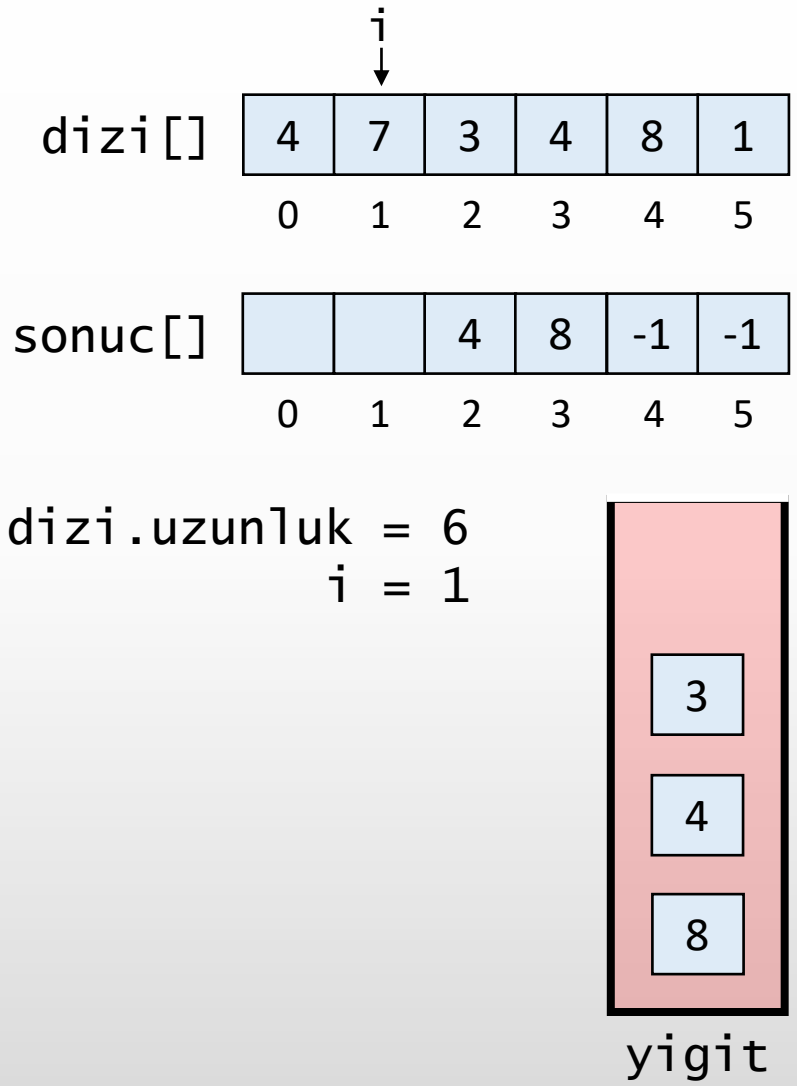
sonrakiBuyukEleman(dizi);

```
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;
}
```



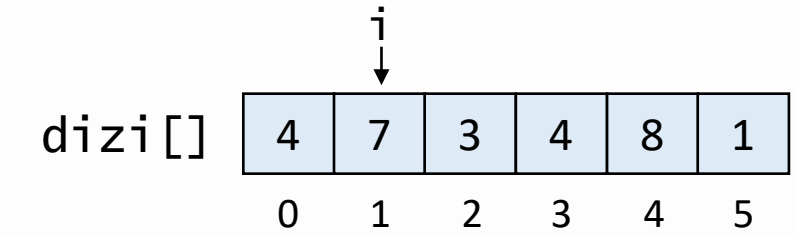

```
sonrakiBuyukEleman(dizi);
```

```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;
}
```



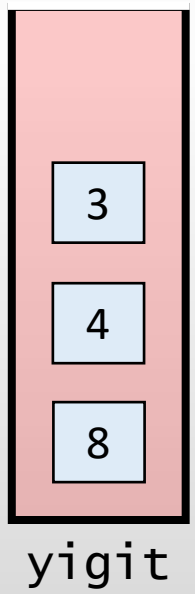

```
sonrakiBuyukEleman(dizi);
```

```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;
}
```



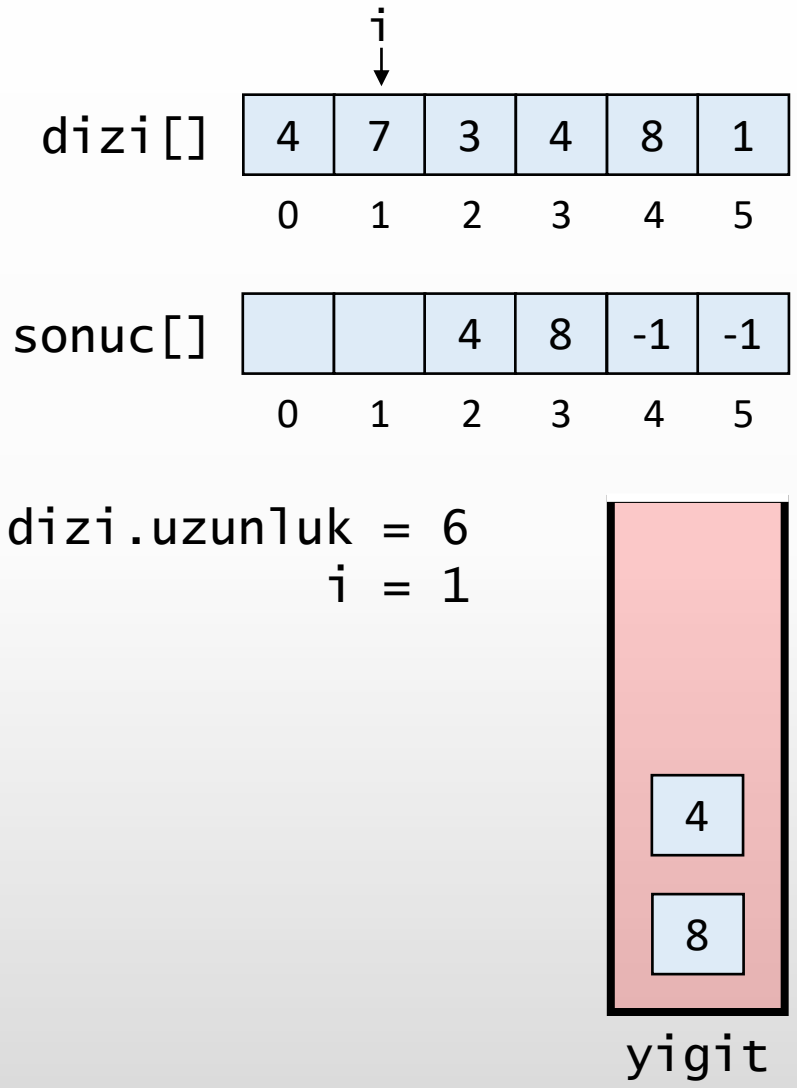
dizi.uzunluk = 6

i = 1



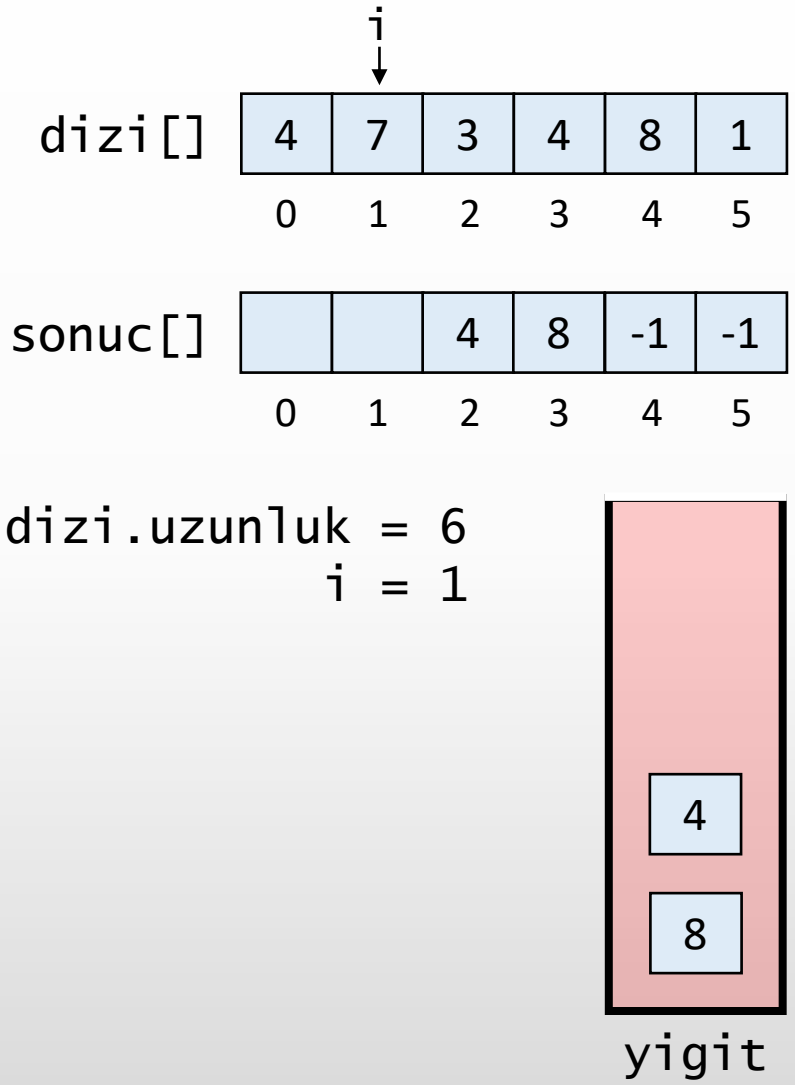
sonrakiBuyukEleman(dizi);

```
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;
}
```



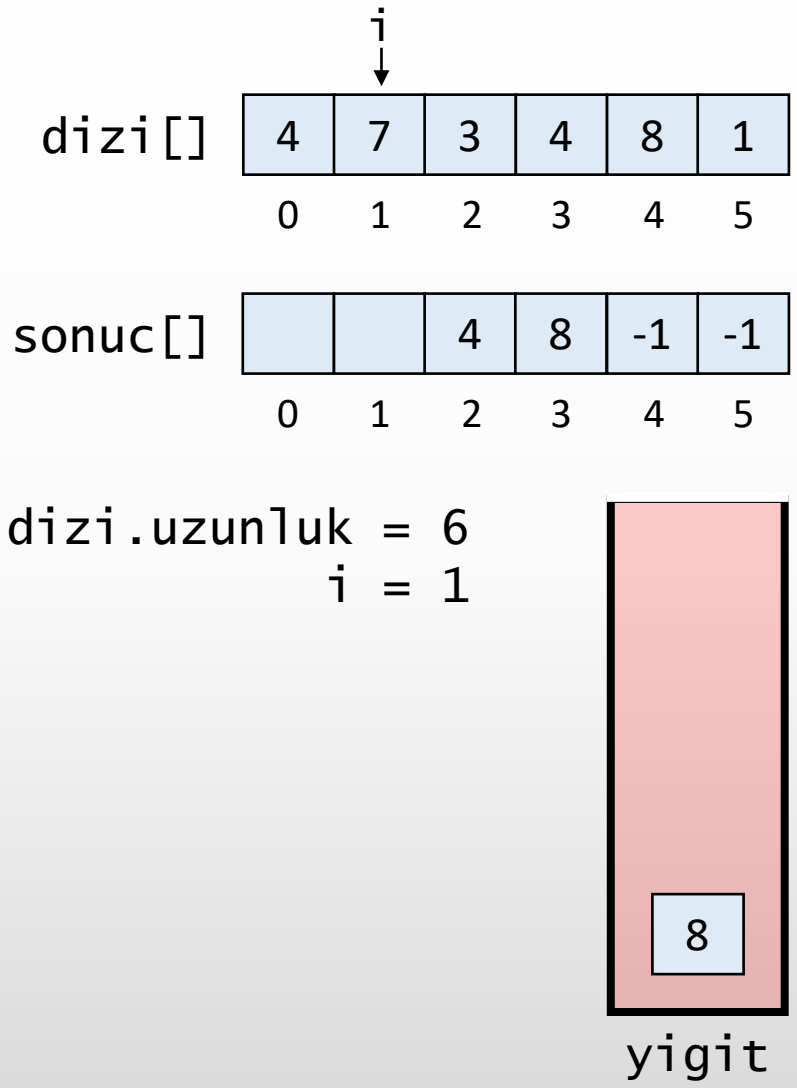

```
sonrakiBuyukEleman(dizi);
```

```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;
}
```



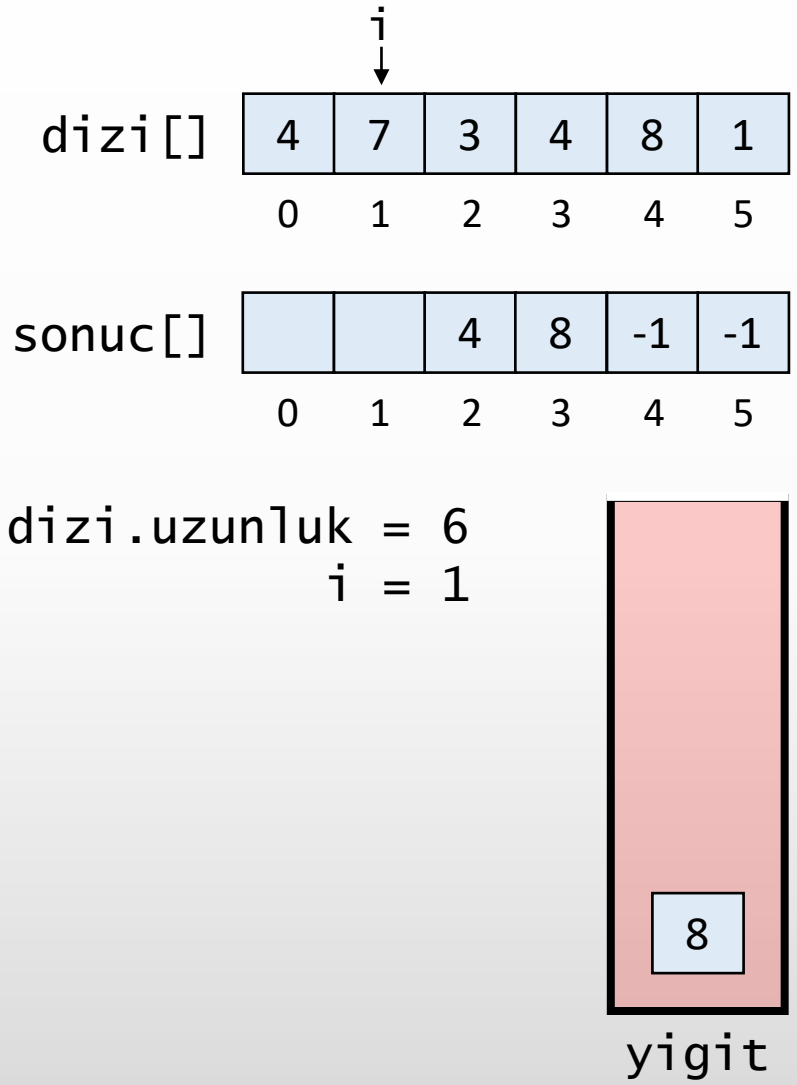
sonrakiBuyukEleman(dizi);

```
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;
}
```



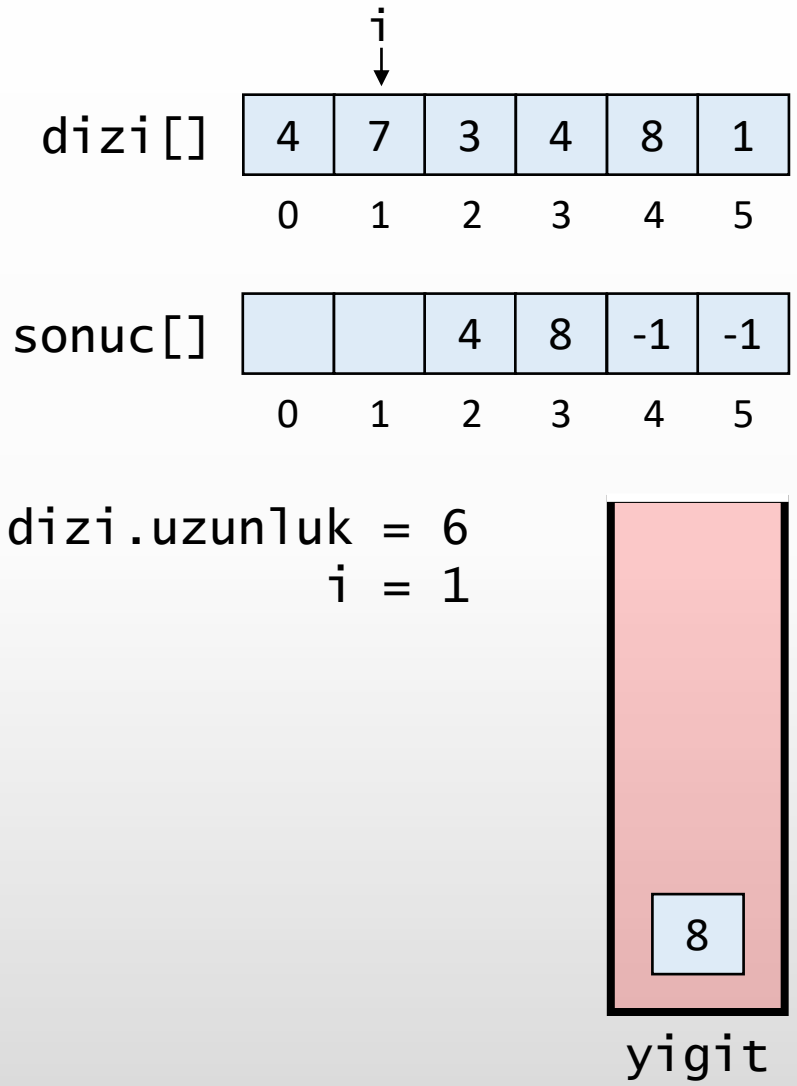

```
sonrakiBuyukEleman(dizi);
```

```
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;
}
```



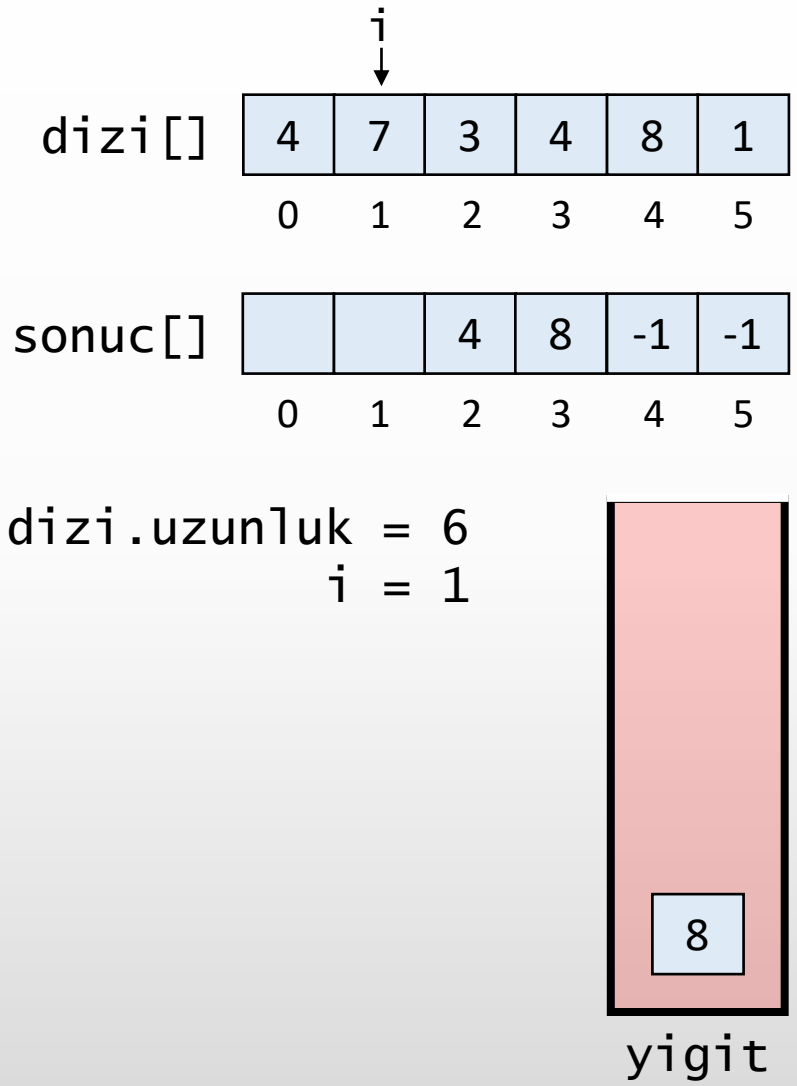
sonrakiBuyukEleman(dizi);

```
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;
}
```



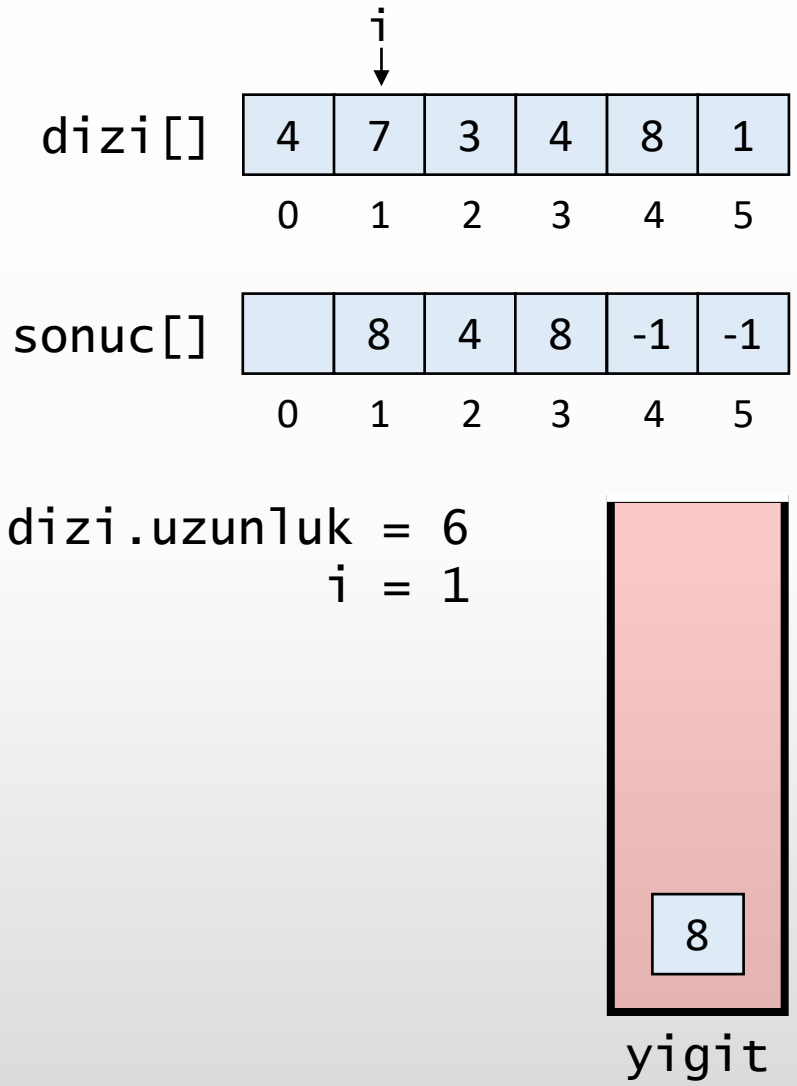

```
sonrakiBuyukEleman(dizi);
```

```
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;
}
```



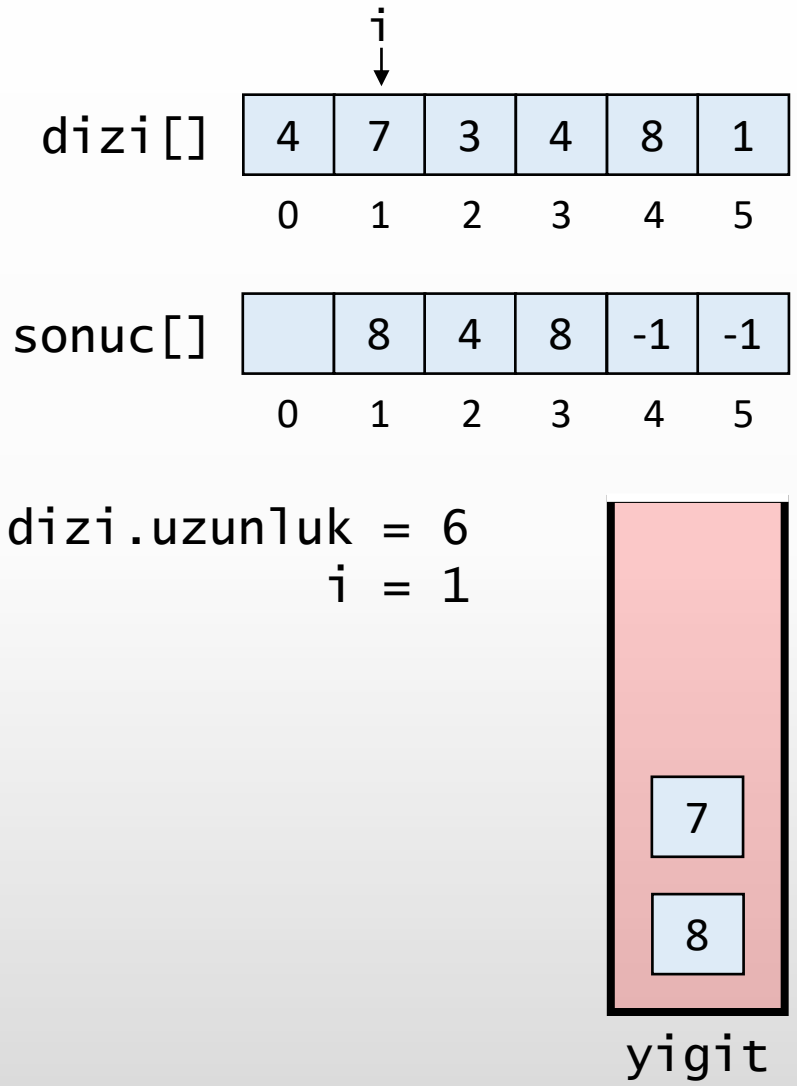

```
sonrakiBuyukEleman(dizi);
```

```
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;
}
```



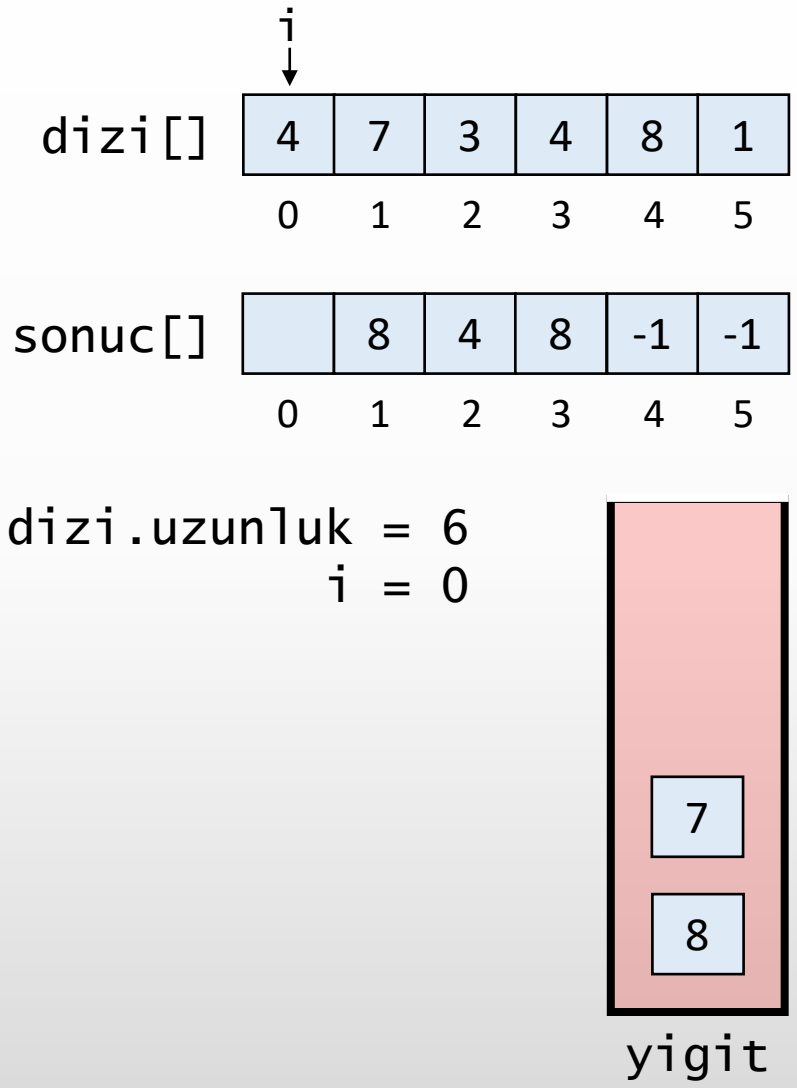

```
sonrakiBuyukEleman(dizi);
```

```
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;
}
```



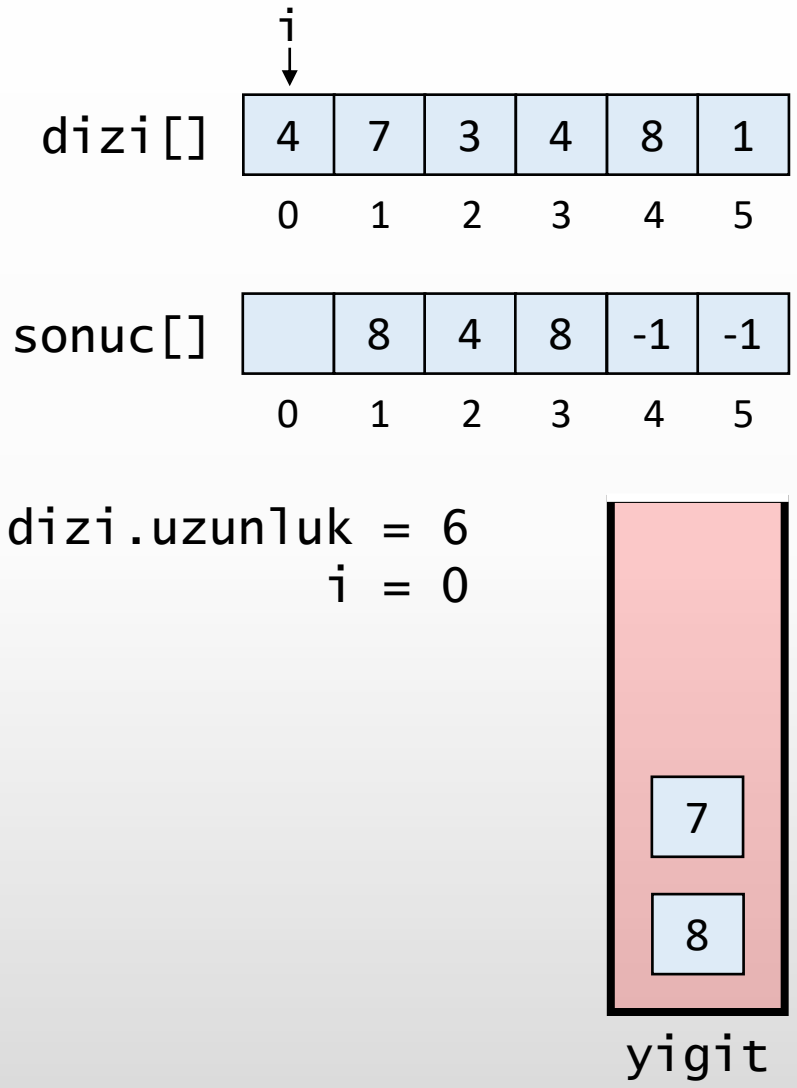
sonrakiBuyukEleman(dizi);

```
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;
}
```



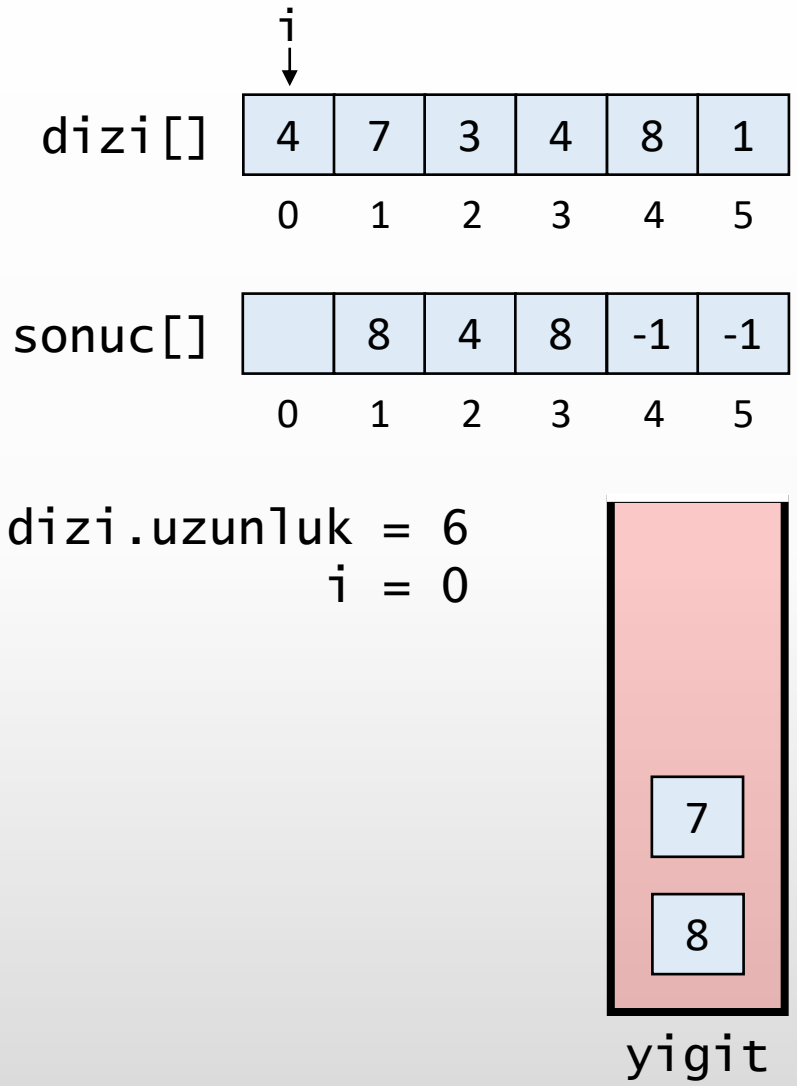

```
sonrakiBuyukEleman(dizi);
```

```
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;
}
```



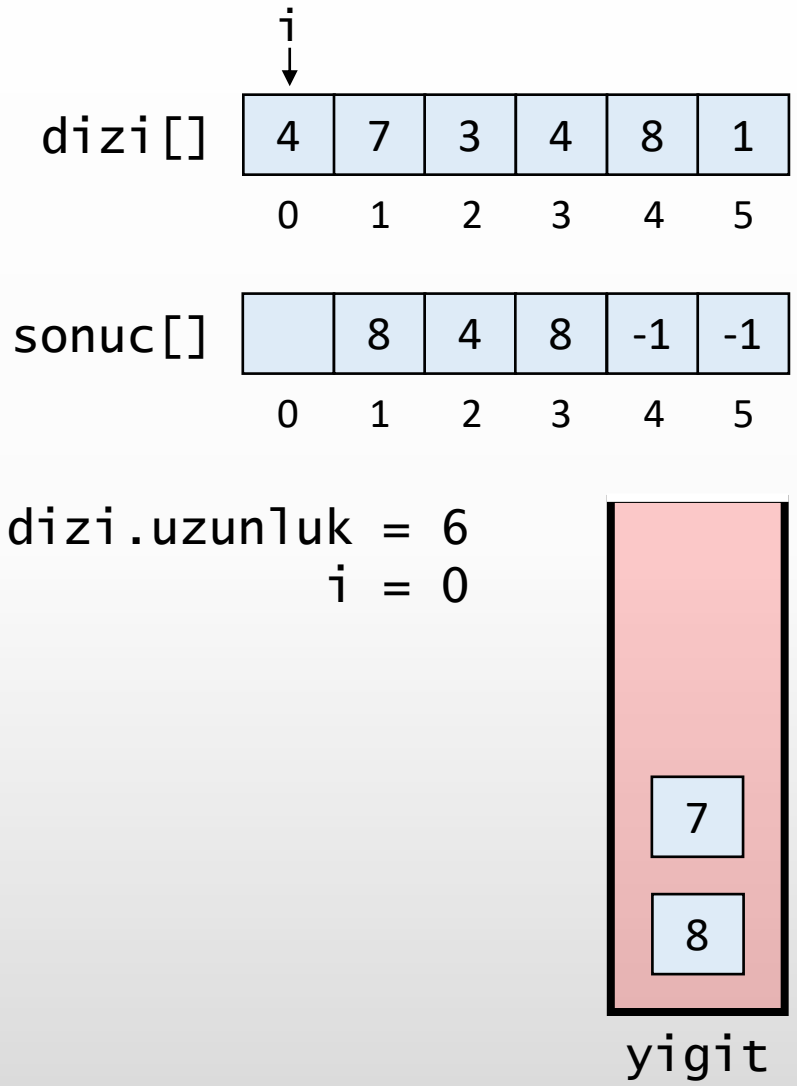

```
sonrakiBuyukEleman(dizi);
```

```
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;
}
```



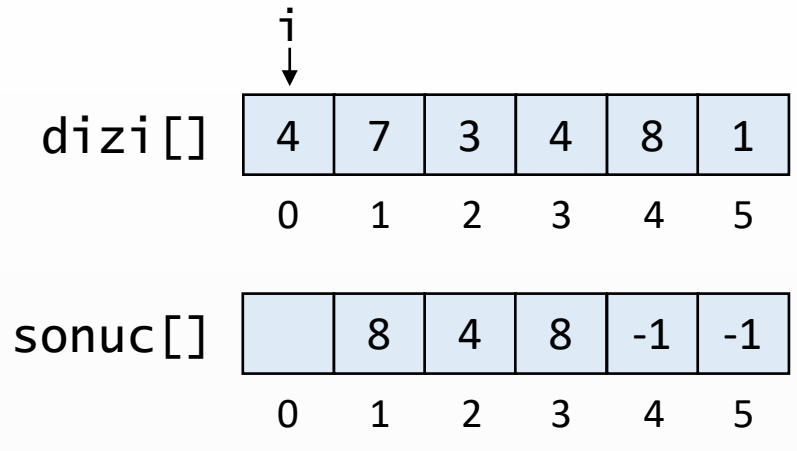

```
sonrakiBuyukEleman(dizi);
```

```
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;
}
```



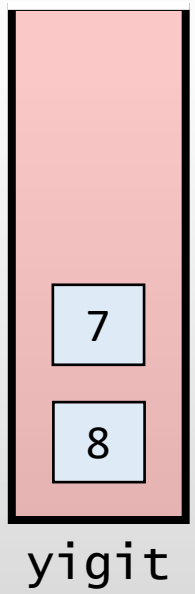

```
sonrakiBuyukEleman(dizi);
```

```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;
}
```



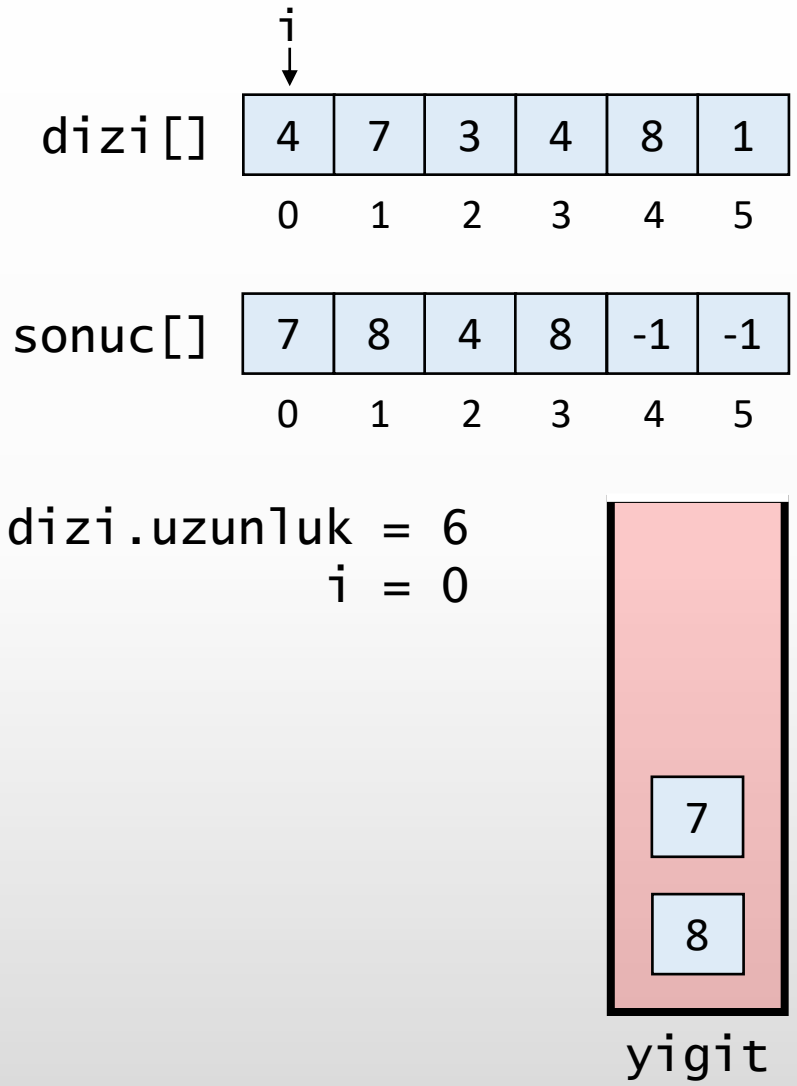
dizi.uzunluk = 6

i = 0



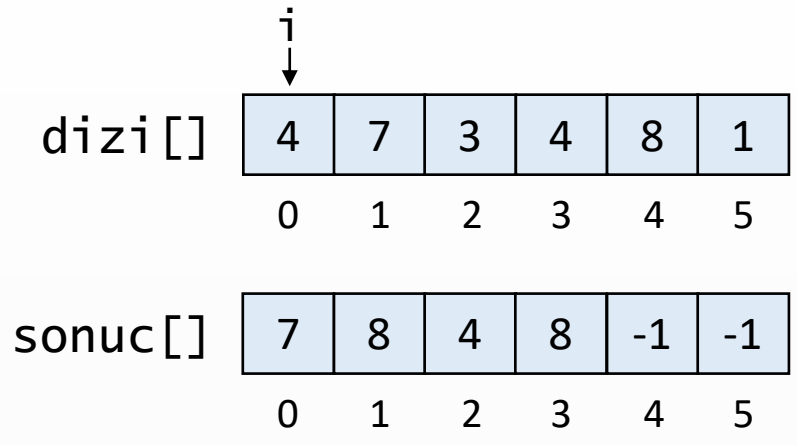
sonrakiBuyukEleman(dizi);

```
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;
}
```

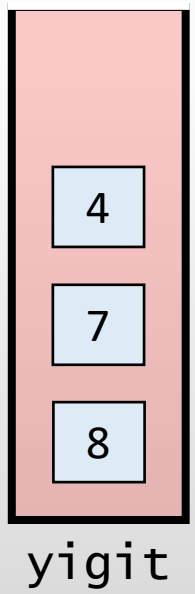



```
sonrakiBuyukEleman(dizi);
```

```
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);

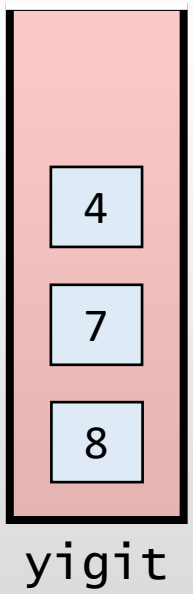
```
int[] sonrakiBuyukEleman(int[] dizi) {
   int[] sonuc = new int[dizi.length];
   Stack<Integer> yigit = new Stack<>();
   for(int i = dizi.length - 1; i >= 0; i--) {
      if(!yigit.isEmpty()) {
         while(!yigit.isEmpty() &&
               yigit.peek() <= dizi[i]) {
            yigit.pop();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

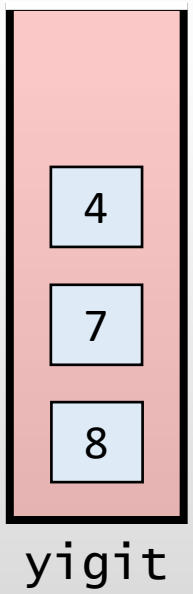
```
int[] sonrakiBuyukEleman(int[] dizi) {
   int[] sonuc = new int[dizi.length];
   Stack<Integer> yigit = new Stack<>();
   for(int i = dizi.length - 1; i >= 0; i--) {
      if(!yigit.isEmpty()) {
         while(!yigit.isEmpty() &&
               yigit.peek() <= dizi[i]) {
            yigit.pop();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

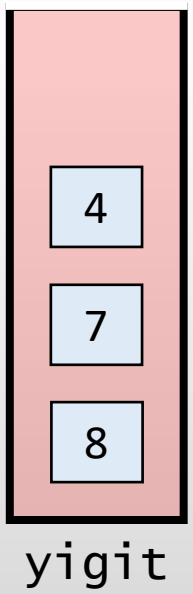
```
int[] sonrakiBuyukEleman(int[] dizi) {
   int[] sonuc = new int[dizi.length];
   Stack<Integer> yigit = new Stack<>();
   for(int i = dizi.length - 1; i >= 0; i--) {
      if(!yigit.isEmpty()) {
         while(!yigit.isEmpty() &&
               yigit.peek() <= dizi[i]) {
            yigit.pop();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



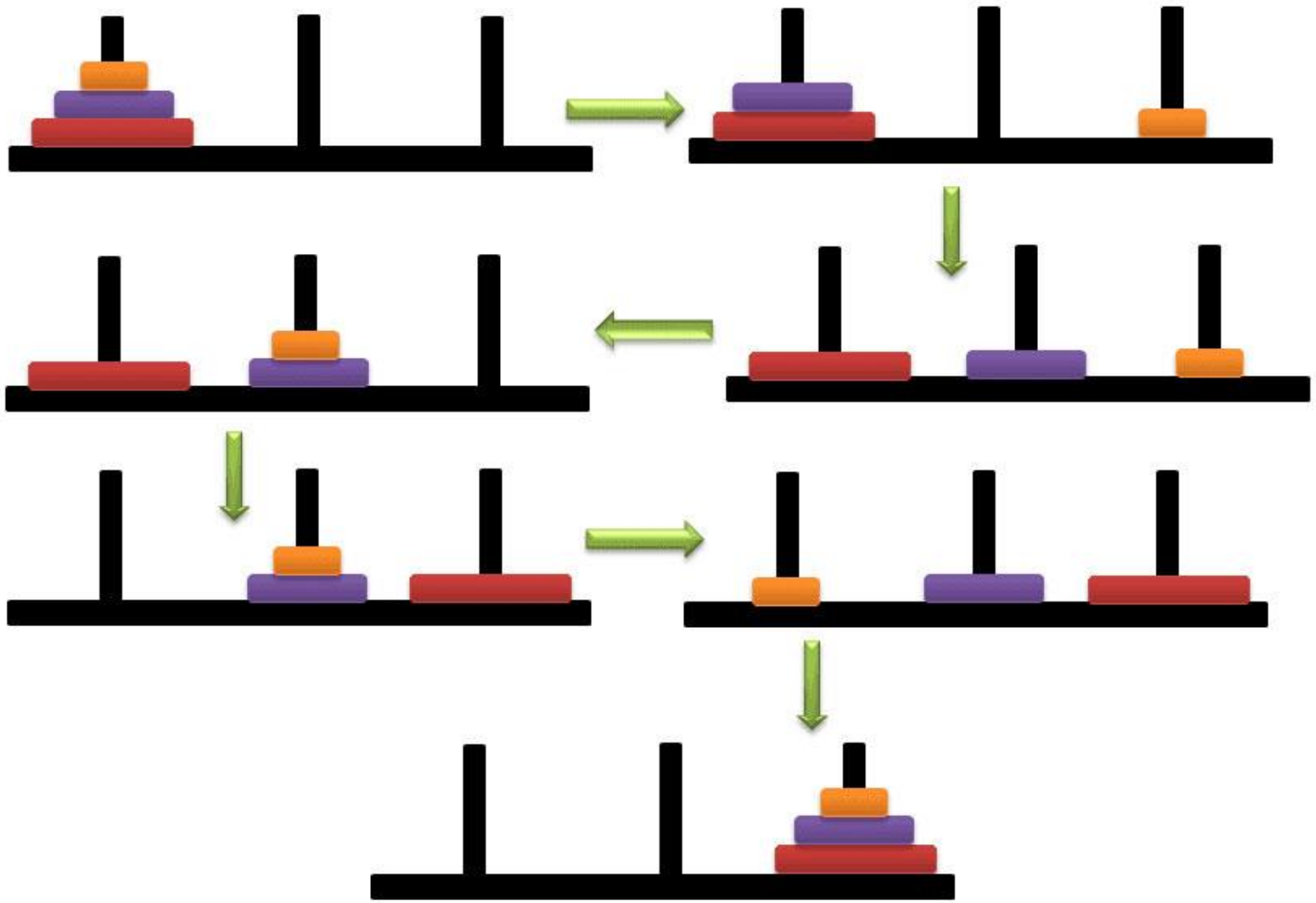
sonrakiBuyukEleman(dizi);

```
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;
}
```

Hanoi Kuleleri

- Hanoi Kuleleri, bilgisayar bilimleri ve matematikte klasik bir problemdir; burada amaç, belirli kurallara uyarak bir disk yığınını bir çividen diğerine taşımaktır:
  - Aynı anda yalnızca bir disk taşınabilir.
  - Bir disk yalnızca daha büyük bir diskin veya boş bir çivinin üzerine yerleştirilebilir.
  - Amaç, gerektiğinde yedek (yardımcı) bir sabitleyici kullanarak tüm disk yığınını kaynak sabitleyiciden hedef sabitleyiciye taşımaktır.

Hanoi Kuleleri



Hanoi Kuleleri

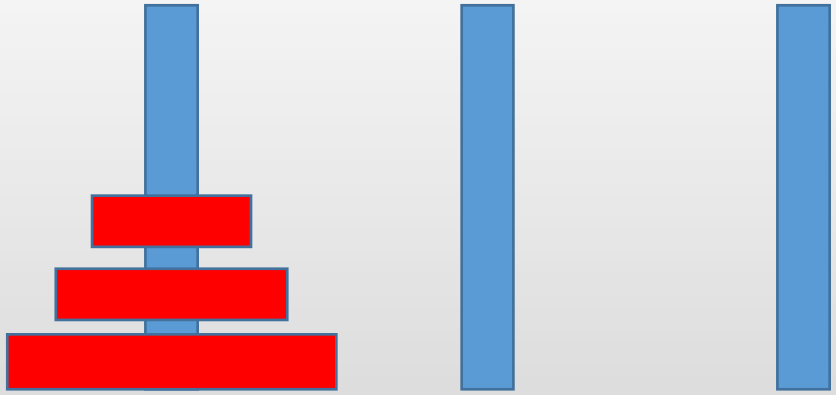
```
static void hanoi(int diskSayisi) {
  Stack<Disk> yigin = new Stack<>();
  yigin.push(new Disk(diskSayisi, 'A', 'B', 'C')); // Başlangıç durumu
  while (!yigin.isEmpty()) {
    Disk disk = yigin.pop();
    if (disk.boyut == 1) {
      System.out.println("Diski" + disk.kaynak + "tan" + disk.hedef + "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));
    }
  }
```

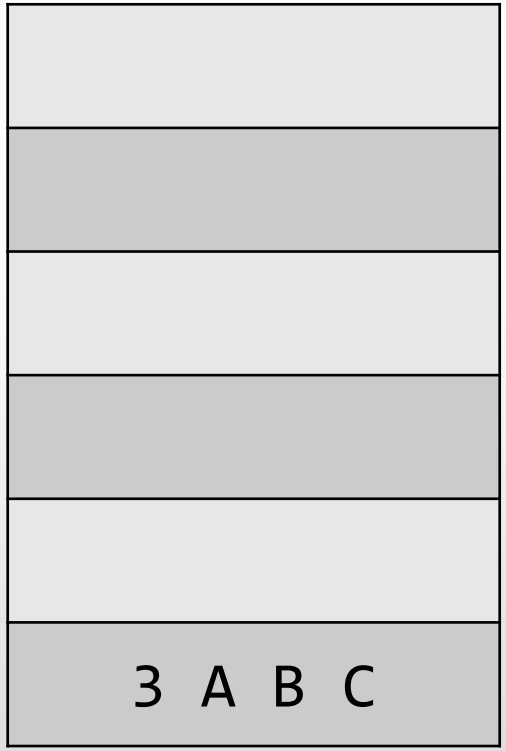
Hanoi Kuleleri

- Döngünün içinde, yığından bir Disk nesnesi çıkarır ve boyutunu kontrol eder.
  - **Boyut 1 ise**, bu, doğrudan hedefe taşınabilecek tek bir disk olduğu anlamına gelir. Kod daha sonra hareketi belirten bir mesaj yazdırır.
  - **Değilse,** Hanoi Kuleleri problemine yinelemeli yaklaşımı simüle eder. Yığına üç yeni Disk nesnesi yerleştirerek sorunu daha küçük alt problemlere böler.
    - İlk nesne, hedefi yardımcı olarak kullanarak üst disk.boyut - 1 disklerini kaynaktan yardımcıya taşımayı temsil eder.
    - 2. nesne, en büyük diskin (boyut 1) kaynaktan hedefe taşınmasını temsil eder.
    - 3. nesne, kaynağı yardımcı olarak kullanarak disk.boyut - 1 daha küçük diskin yardımcıdan hedefe taşınmasını temsil eder.

Hanoi Kuleleri – Yığın İşlemleri

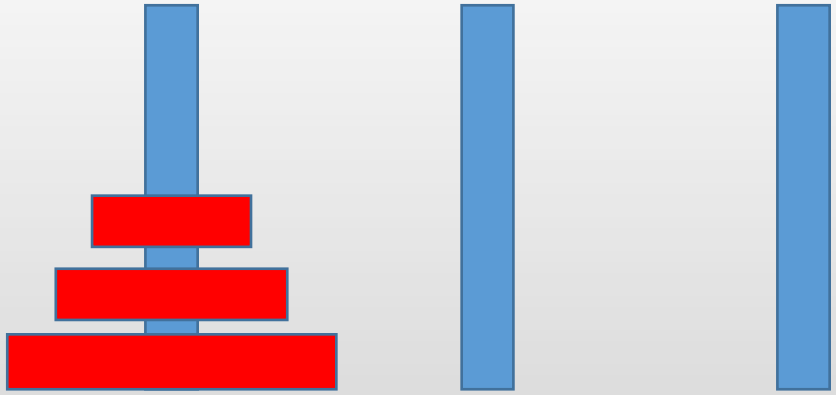
- `Yığına ekle 3 A B C`

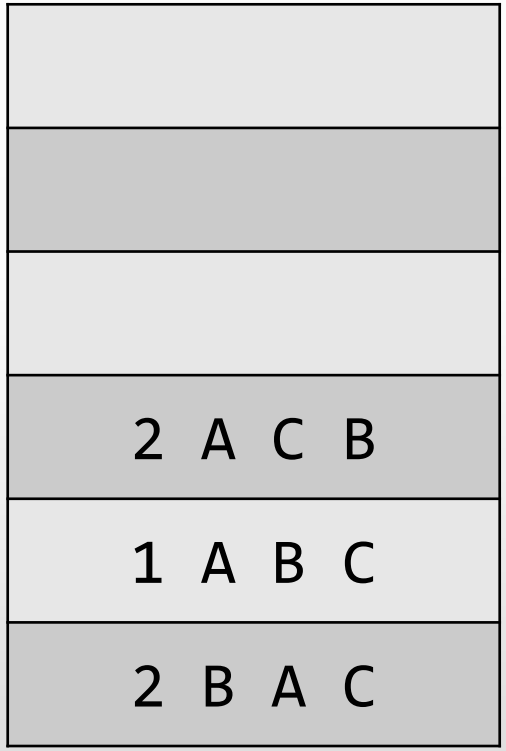




Hanoi Kuleleri – Yığın İşlemleri

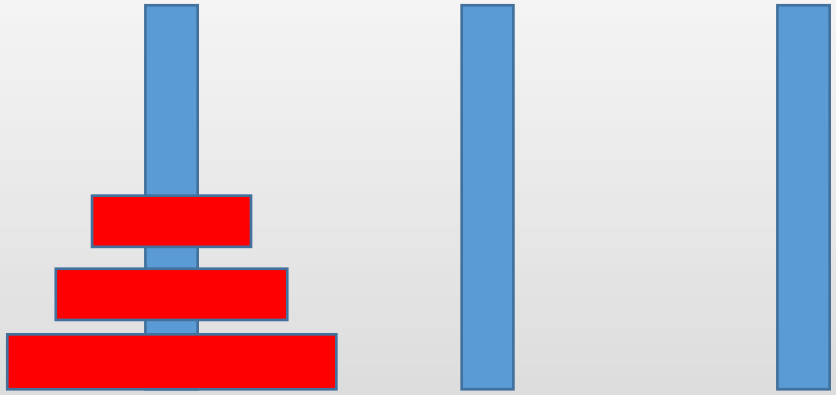
- Yığından al 3 A B C
- Yığına ekle 2 B A C
- Yığına ekle 1 A B C
- Yığına ekle 2 A C B





Hanoi Kuleleri – Yığın İşlemleri

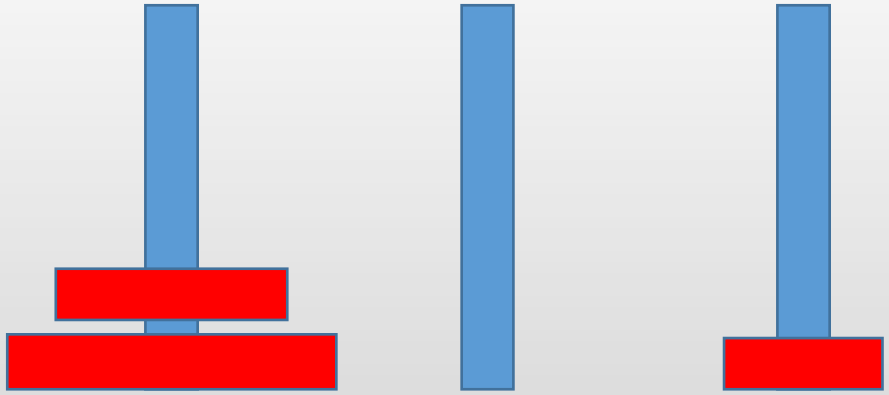
- Yığından al 2 A C B
- Yığına ekle 1 C A B
- Yığına ekle 1 A C B
- Yığına ekle 1 A B C



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

Hanoi Kuleleri – Yığın İşlemleri

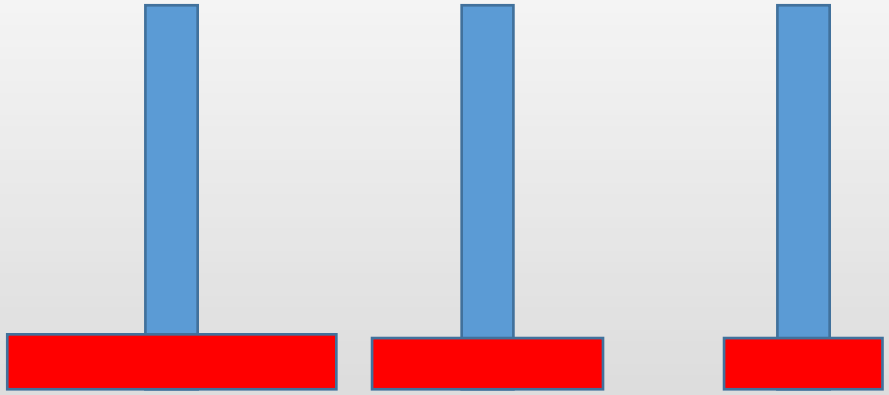
- Yığından al 1 A B C
- Diski A çubuğundan C çubuğuna taşı.



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

Hanoi Kuleleri – Yığın İşlemleri

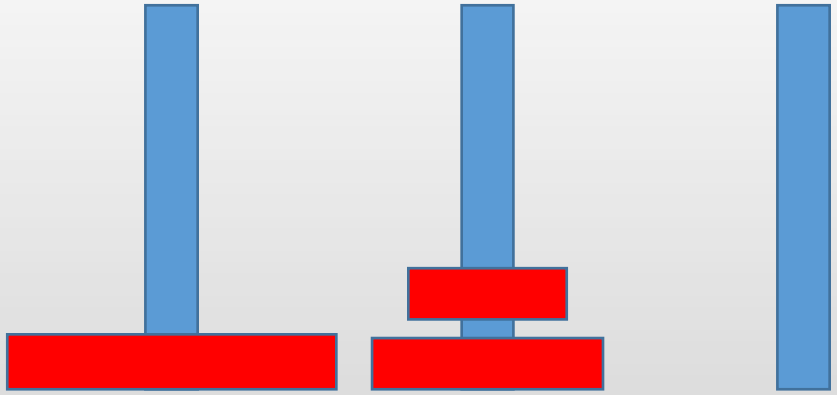
- Yığından al 1 A C B
- Diski A çubuğundan B çubuğuna taşı.

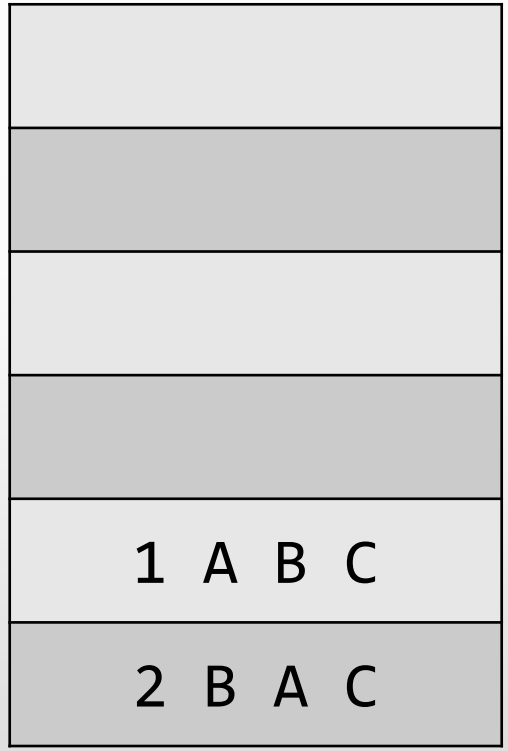


1 C A B
1 A B C
2 B A C

Hanoi Kuleleri – Yığın İşlemleri

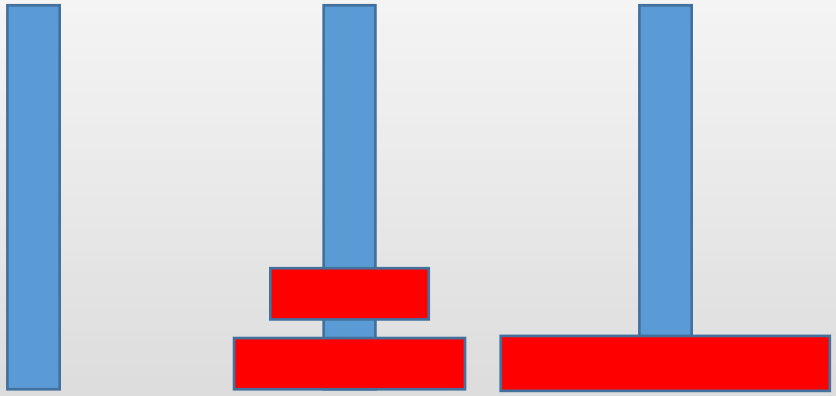
- Yığından al 1 C A B
- Diski C çubuğundan B çubuğuna taşı.

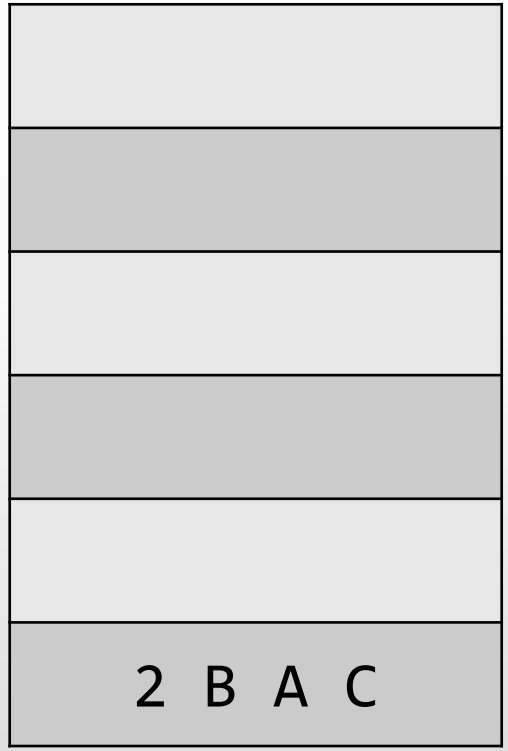




Hanoi Kuleleri – Yığın İşlemleri

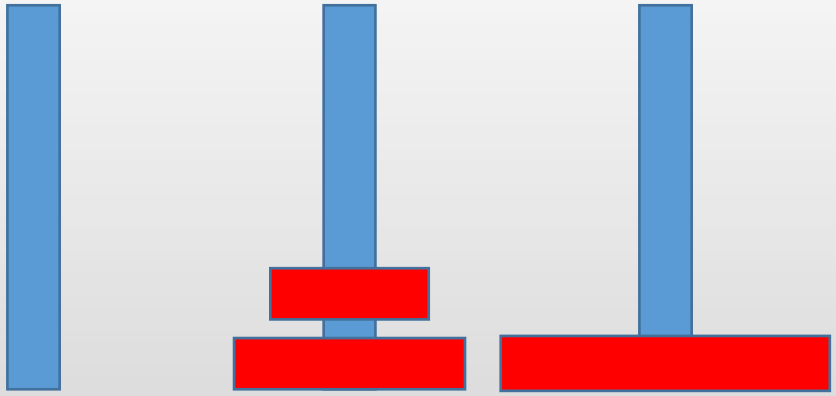
- Yığından al 1 A B C
- Diski A çubuğundan C çubuğuna taşı.

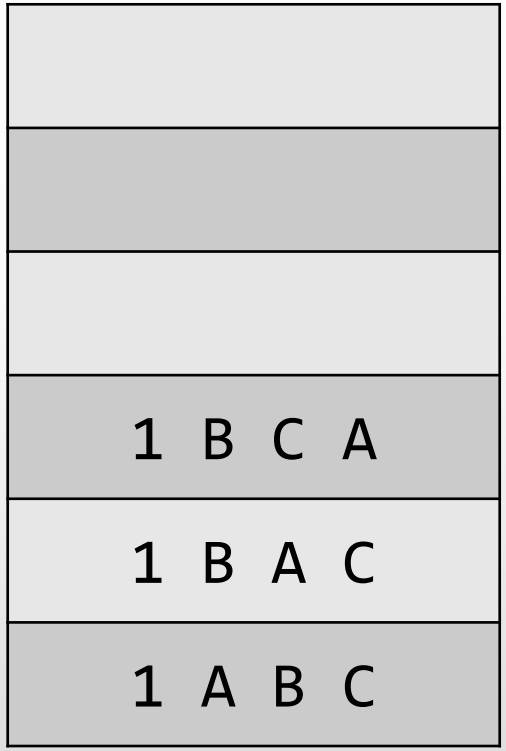




Hanoi Kuleleri – Yığın İşlemleri

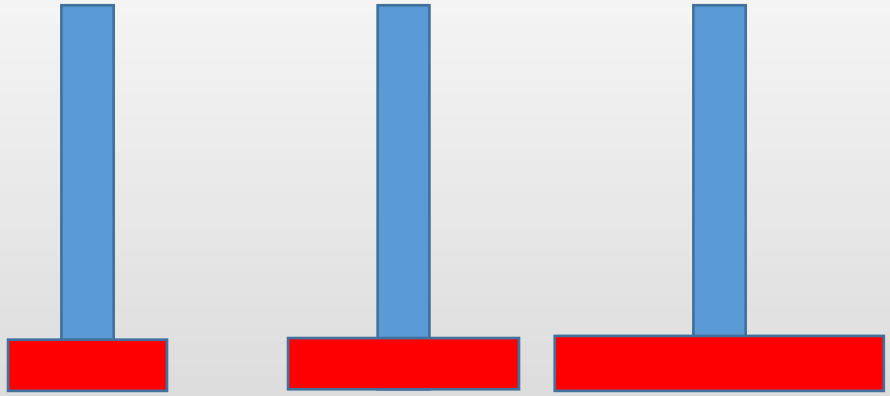
- Yığından al 2 B A C
- Yığına ekle 1 A B C
- Yığına ekle 1 B A C
- Yığına ekle 1 B C A

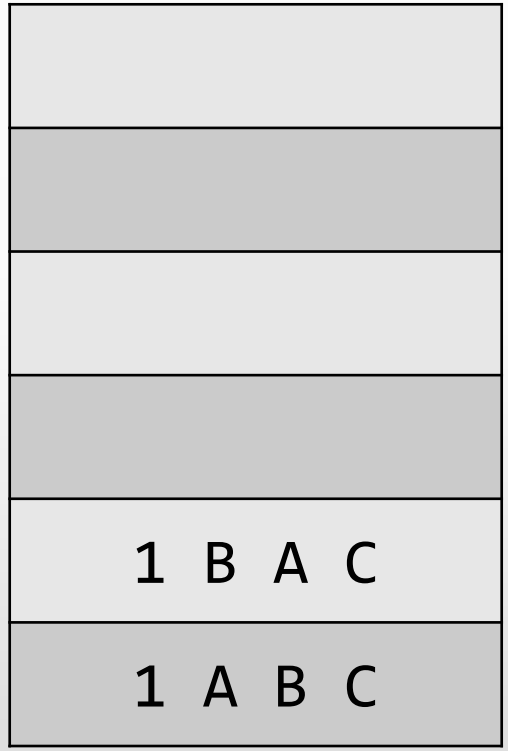




Hanoi Kuleleri – Yığın İşlemleri

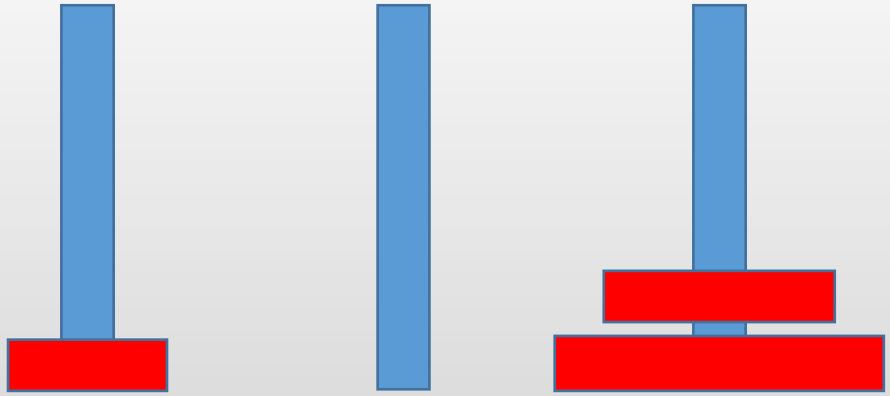
- Yığından al 1 B C A
- Diski B çubuğundan A çubuğuna taşı.

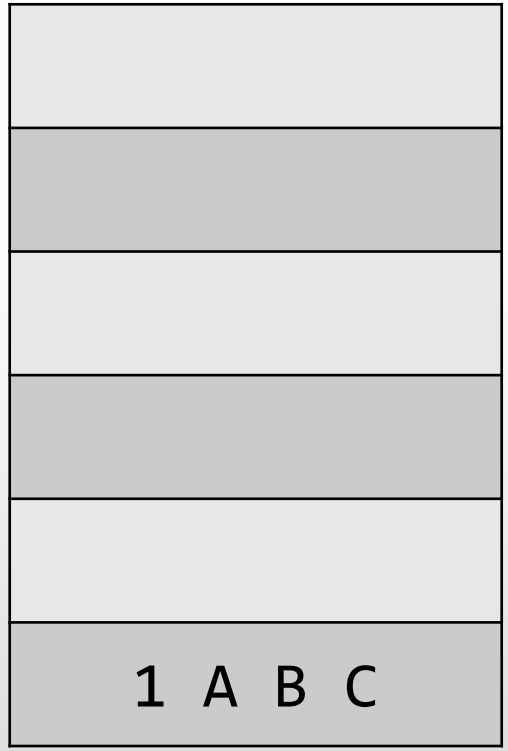




Hanoi Kuleleri – Yığın İşlemleri

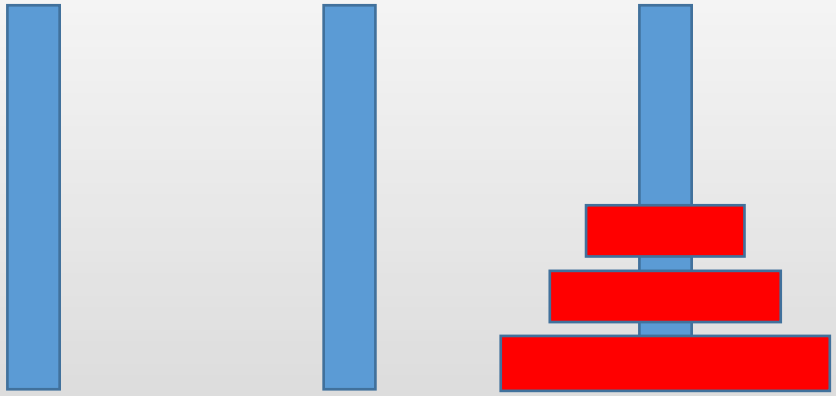
- `Yığından al 1 B A C`
- `Diski B çubuğundan C çubuğuna taşı.`

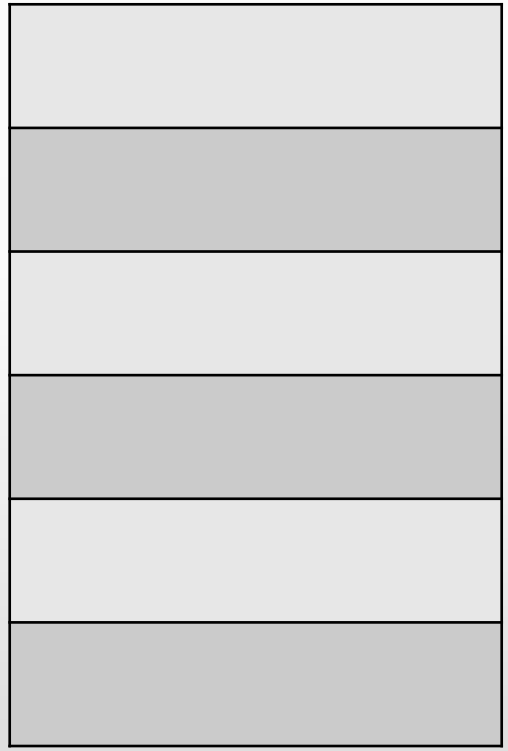




Hanoi Kuleleri – Yığın İşlemleri

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





Stock Span Problemi

- "Hisse Senedi Sıçrama Problemi," hisse senedi veya finansal veri analizinde kullanılan bir problemdir.
- Belirli bir süre içerisindeki hisse senedi fiyatlarını inceleyerek, her bir günün hangi günlerden daha yüksek bir kapanış fiyatına sahip olduğunu bulmayı amaçlar.
- Yani, her günün o ana kadar kaç ardışık gün boyunca 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 sayısını (bu günün fiyatından yüksek önceki günlerin sayısı) hesaplamamız gerekiyor,.
- **Örnek**:
  - Diyelim ki elimizde aşağıdaki hisse senedi fiyatları bulunuyor:
  - [100, 80, 60, 70, 60, 75, 85]

Stock Span Problemi

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

- Bu verilere göre, her günün 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**

İçtakı (Infix) ifadeyi Öntakı (Prefix) Dönüştürme

- Infix ifadeyi prefix ifadeye dönüştürme, matematiksel ve mantıksal ifadelerle çalışırken kullanılan bir işlemdir.
  - **İfadeleri Değerlendirme**: Matematiksel ve mantıksal ifadelerin daha basit bir şekilde değerlendirilmesine olanak tanır. İşlem önceliklerini ve parantezleri ortadan kaldırır.
  - **Derleyici ve Yorumlayıcı Tasarımı**: Birçok programlama dili, ifadeleri prefix notasyonla temsil eder. Derleyici yorumlayıcı tasarlarken, kaynak kodundan alınan infix ifadeleri analiz etme ve yürütme için gerekebilir.
  - **Aritmetik İşlemler**: Özellikle bilgisayar cebir sistemleri ve hesap makinelerinde aritmetik işlemlerde yaygın olarak kullanılır.
  - **İfade Manipülasyonu**: İfadelerin manipülasyonunu basitleştirir. Bu, matematiksel denklemleri basitleştirmek, sembolik cebir yapmak veya karmaşık problemleri çözmek gibi işlemler sırasında faydalı olabilir.

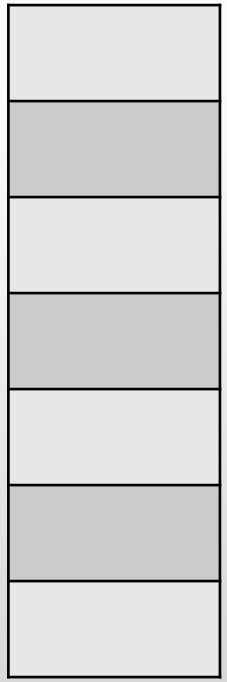
İçtakı (Infix) ifadeyi Öntakı (Prefix) Dönüştürme

- Infix İfade: A*(B+C)/D
- Postfix İfade: ABC+*D/
- Prefix İfade: */A+BCD

İçtakı (Infix) ifadeyi Öntakı (Postfix) Dönüştürme

- Infix İfade: A*(B+C)/D

Postfix:

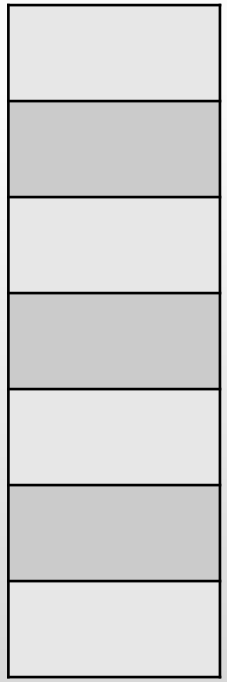


İçtakı (Infix) ifadeyi Öntakı (Postfix) Dönüştürme

- Infix İfade: A*(B+C)/D

- Adım: A
- postfix: A

Postfix: A

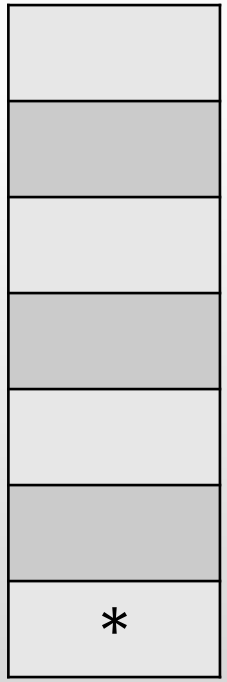


İçtakı (Infix) ifadeyi Öntakı (Postfix) Dönüştürme

- Infix İfade: A*(B+C)/D

- Adım: *
- Yığına koy *

Postfix: A

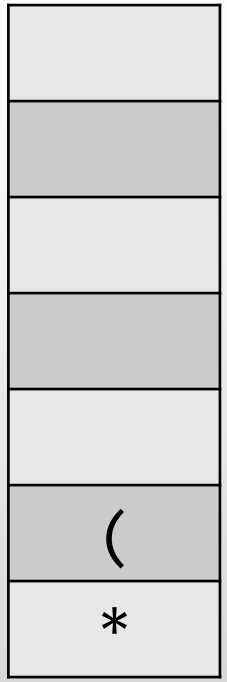


İçtakı (Infix) ifadeyi Öntakı (Postfix) Dönüştürme

- Infix İfade: A*(B+C)/D

- Adım: (
- Yığına koy (

Postfix: A

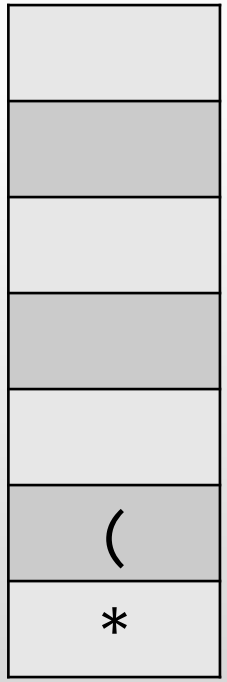


İçtakı (Infix) ifadeyi Öntakı (Postfix) Dönüştürme

- Infix İfade: A*(B+C)/D

- Adım: B
- postfix: AB

Postfix: AB

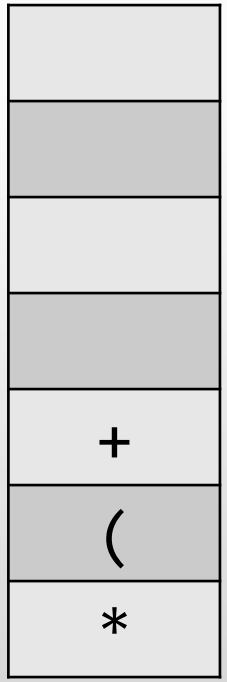


İçtakı (Infix) ifadeyi Öntakı (Postfix) Dönüştürme

- Infix İfade: A*(B+C)/D

- Adım: +
- Yığına koy +

Postfix: AB

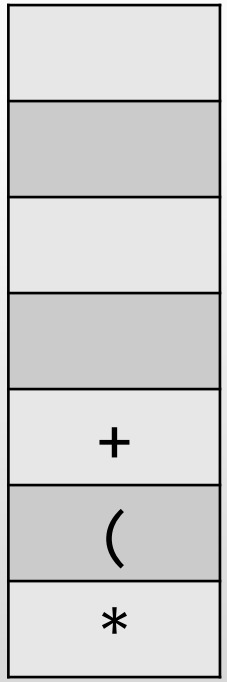


İçtakı (Infix) ifadeyi Öntakı (Postfix) Dönüştürme

- Infix İfade: A*(B+C)/D

- Adım: C
- postfix: ABC

Postfix: ABC

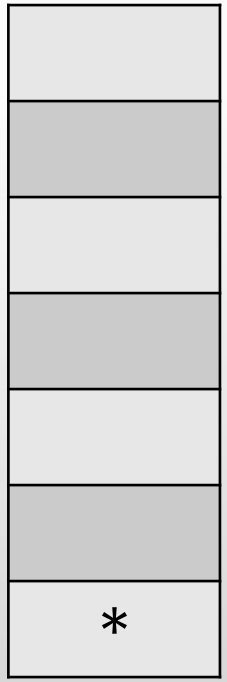


İçtakı (Infix) ifadeyi Öntakı (Postfix) Dönüştürme

- Infix İfade: A*(B+C)/D

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

Postfix: ABC+

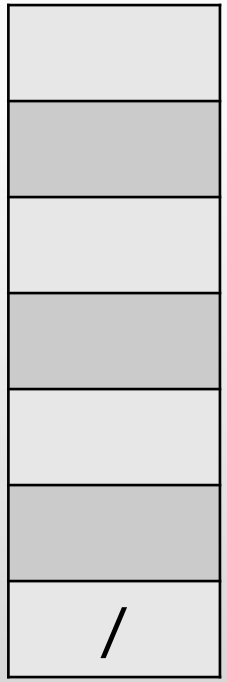


İçtakı (Infix) ifadeyi Öntakı (Postfix) Dönüştürme

- Infix İfade: A*(B+C)/D

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

Postfix: ABC+*

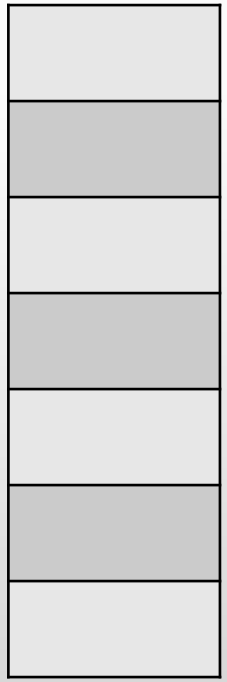


İçtakı (Infix) ifadeyi Öntakı (Postfix) Dönüştürme

- Infix İfade: A*(B+C)/D

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

Postfix: ABC+*D/



Ödev

- Verilen bir metnin Palindrome olup olmadığını Stack yapısını kullanarak O(n) karmaşıklığında bulan kod parçasını yazınız. Algoritmanızın çalışma mantığını kısaca anlatınız.
- sercan.kulcu@giresun.edu.tr
- Son tarih: 2 Kasım 2023 saat 23:59'a kadar
- Konu: «BİLM-201 Ödev 2»

Merhaba Hocam,

Ben xxxx numaralı …. …. 'yım. Ekte ödevimi gönderiyorum.

Saygılarımla,

İyi çalışmalar dilerim.

Ekler: odev2.java

Ödev

- Ödevler isteğe bağlı değil, **zorunlu**.
- ChatGPT gibi araçlara yazdırabilirsiniz, ancak **ne gönderdiğinizden** sorumlusunuz.
- Ödevi **bireysel** olarak yapmalısınız.
- Mail **konu** ve metin kısmına dikkat ediniz.
- Ödevinizi **tam** olarak teslim ediniz. (koda müdahale gerekmemeli)

SON