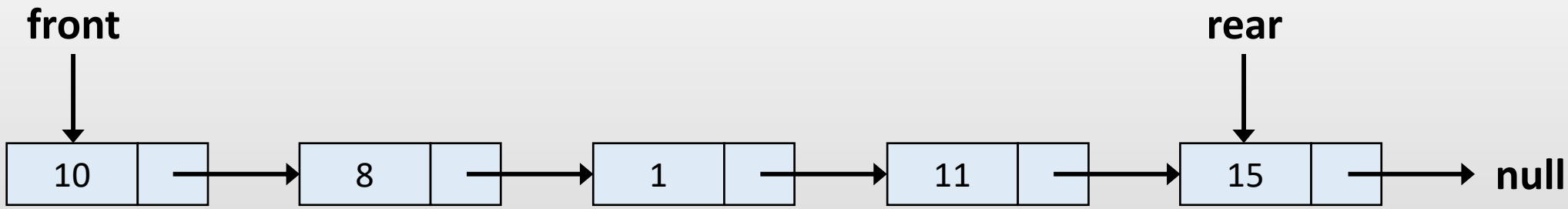
Bölüm 5: Kuyruk

Veri Yapıları

Kuyruk (Queue)

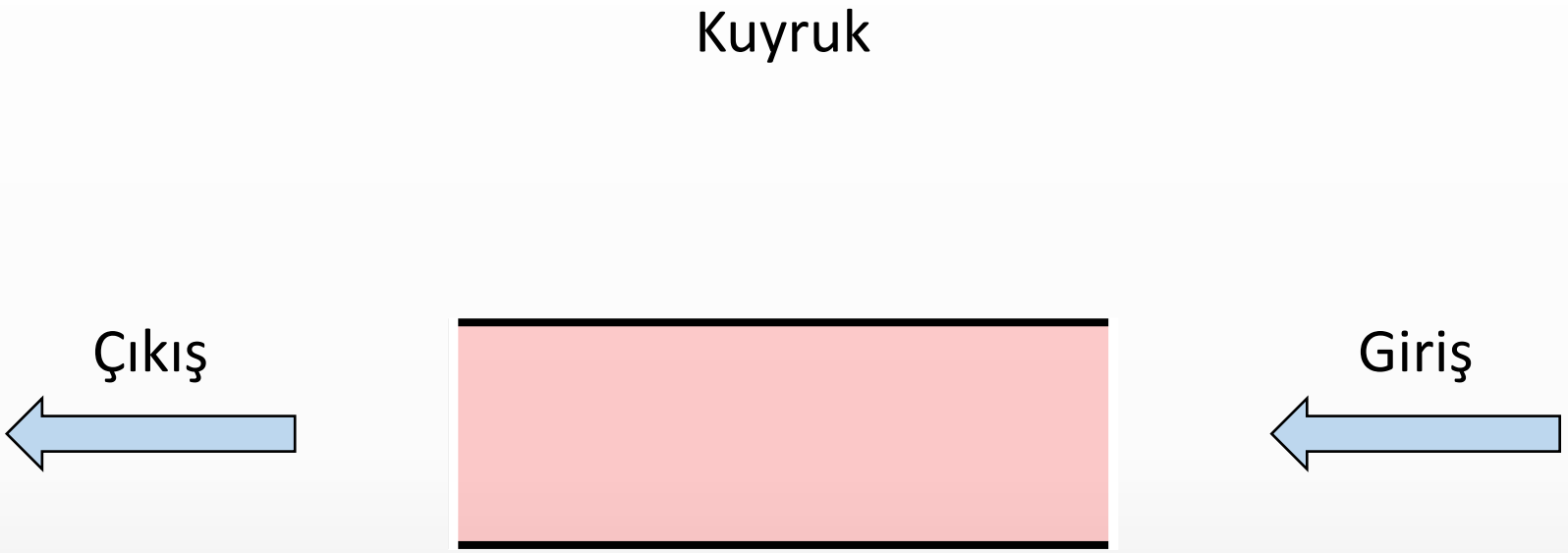
- Her iki ucu açık doğrusal bir veri yapısıdır.
- İşlemler, İlk Giren İlk Çıkar (FIFO) sırasına göre gerçekleştirilir.
- Listeye eklemeler bir uçtan, çıkarmalar diğer uçtan gerçekleştirilir.



Temel Kuyruk İşlemleri

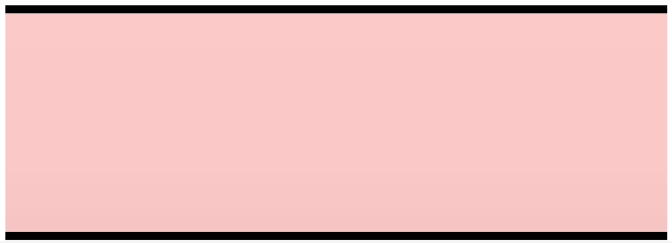
- Ekleme (Enqueue):
  - Kuyruğun sonuna yeni bir öğe ekler.
  - Zaman karmaşıklığı: O(1)
- Çıkarma (Dequeue):
  - Kuyruğun başındaki öğeyi kuyruktan çıkarır.
  - İlk giren öğeyi çıkarır (FIFO ilkesi).
  - Zaman karmaşıklığı: O(1)

Gösterim



Kuyruk

Çıkış



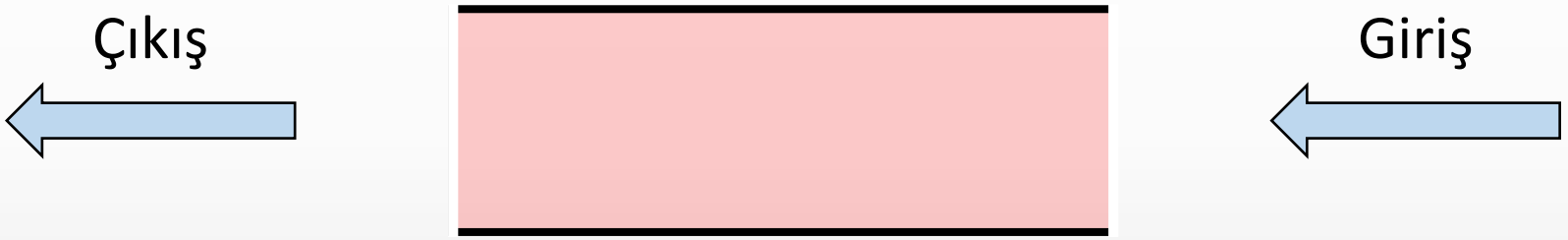
Giriş

front ⟶ **null**

rear ⟶ **null**

enqueue(20)

Kuyruk

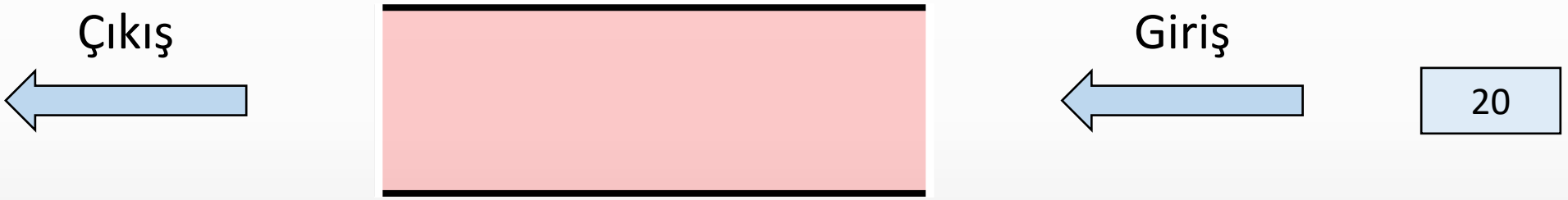


front ⟶ null

rear ⟶ null

enqueue(20)

Kuyruk

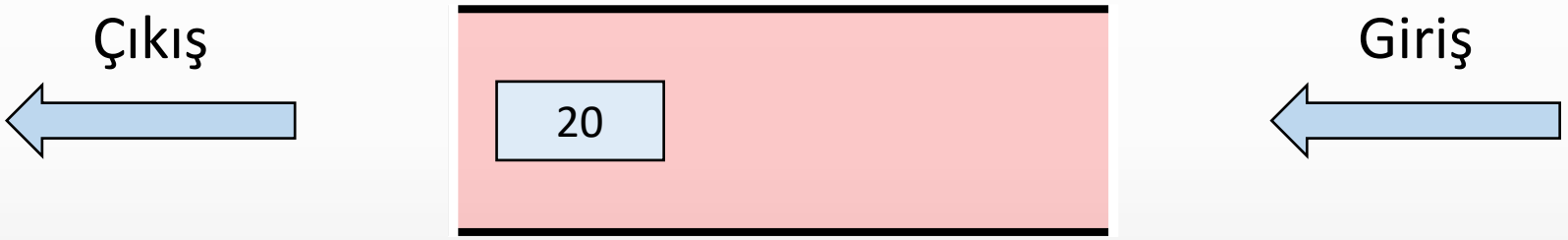


front ⟶ null

rear ⟶ null

enqueue(20)

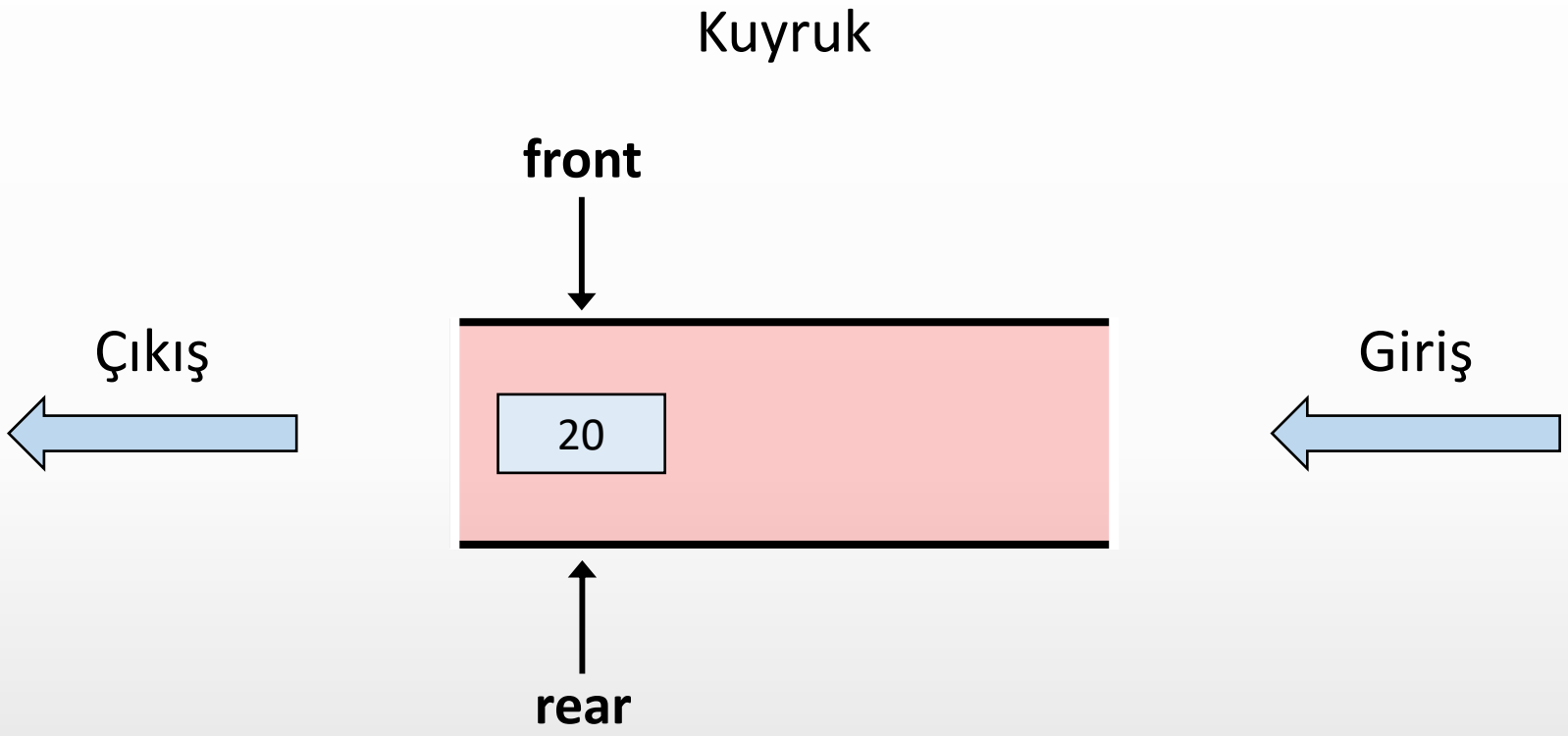
Kuyruk



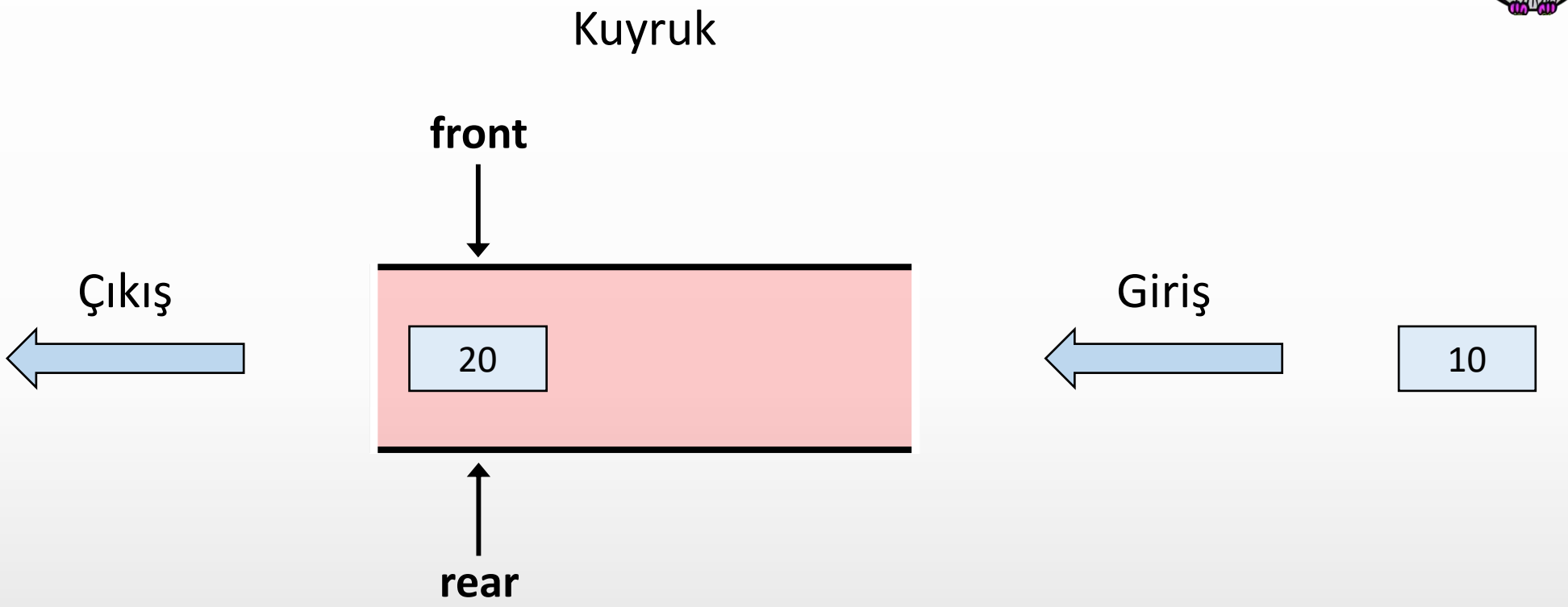
front ⟶ null

rear ⟶ null

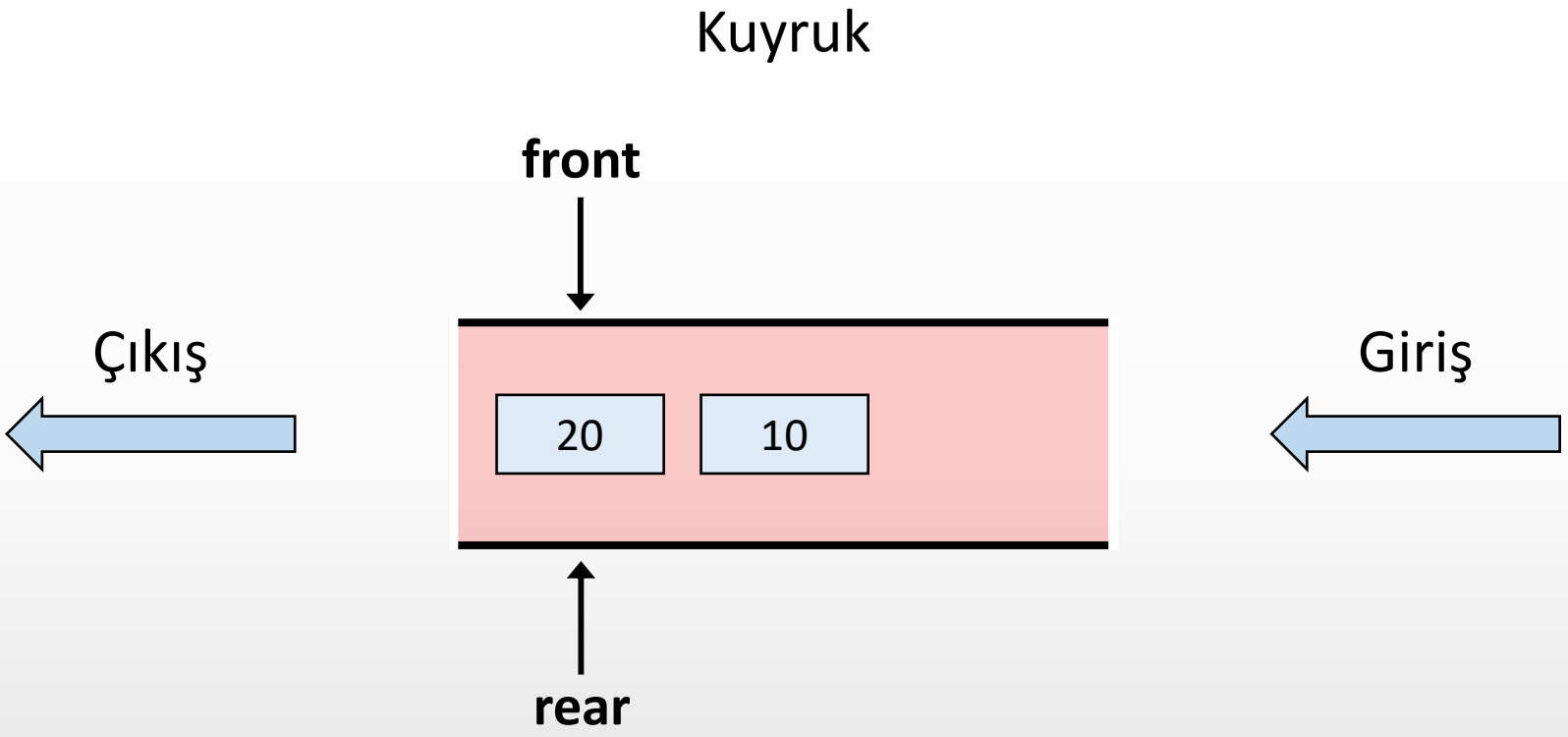
enqueue(20)



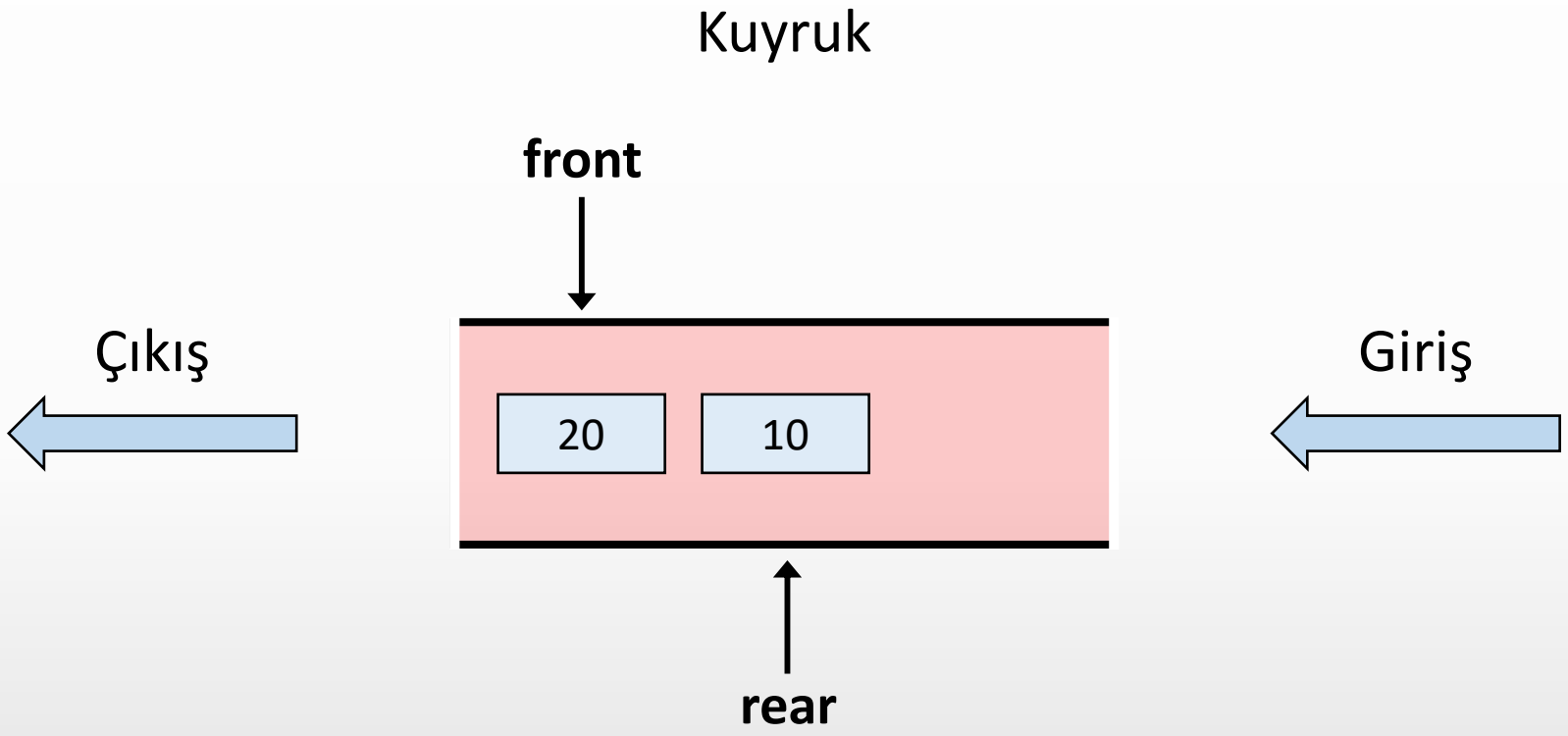
enqueue(10)



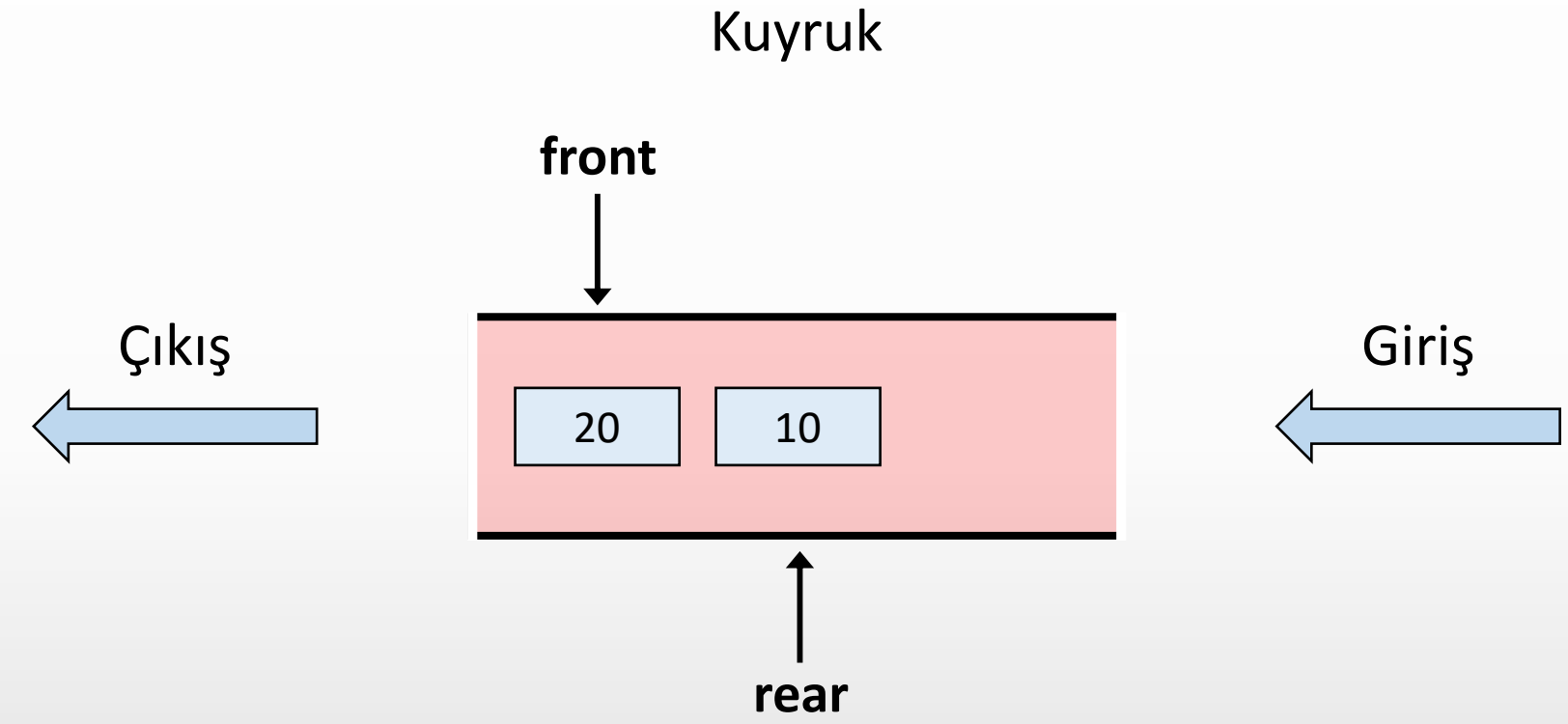
enqueue(10)



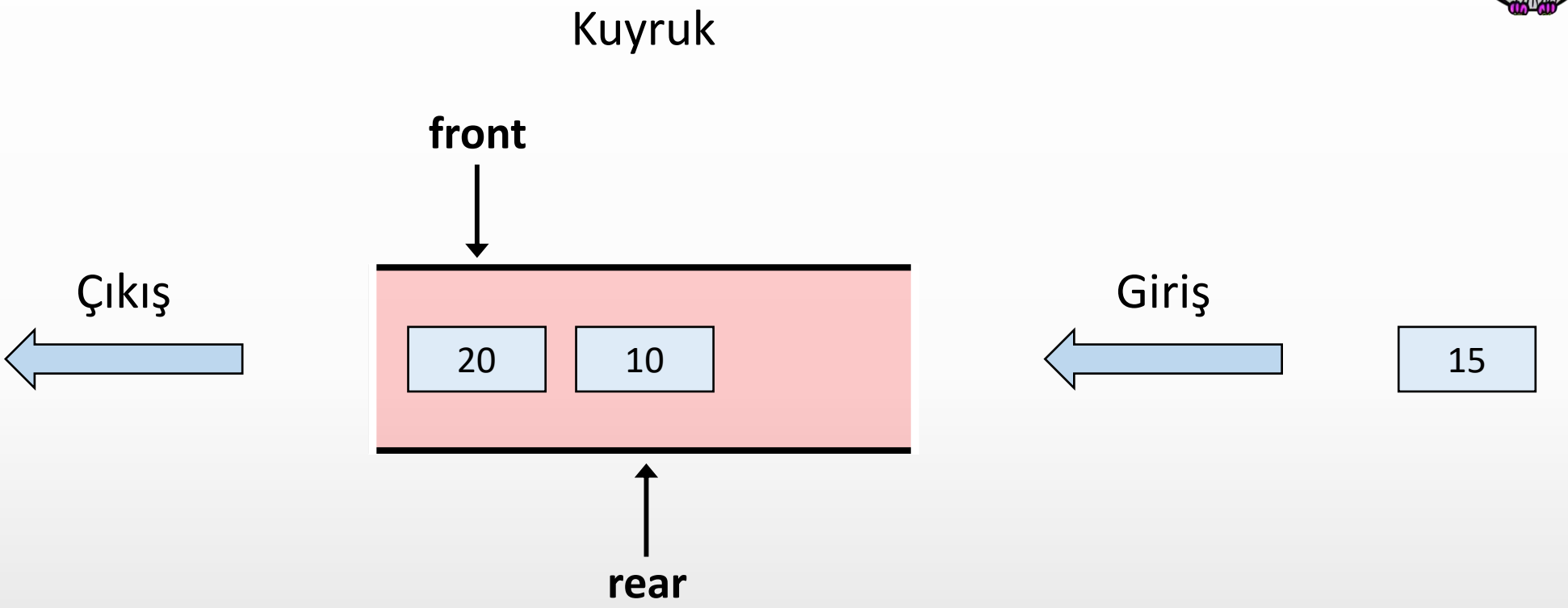
enqueue(10)



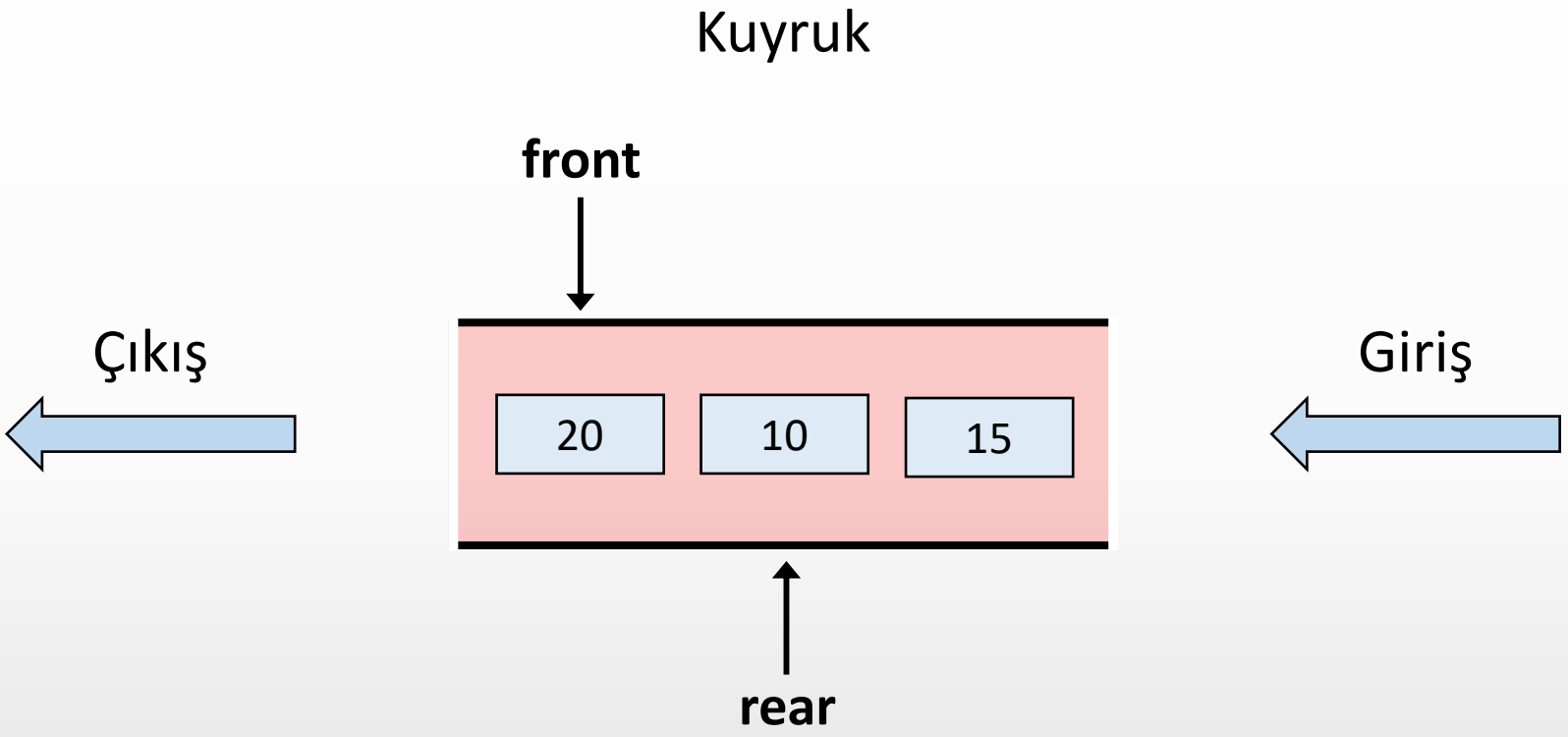
enqueue(15)



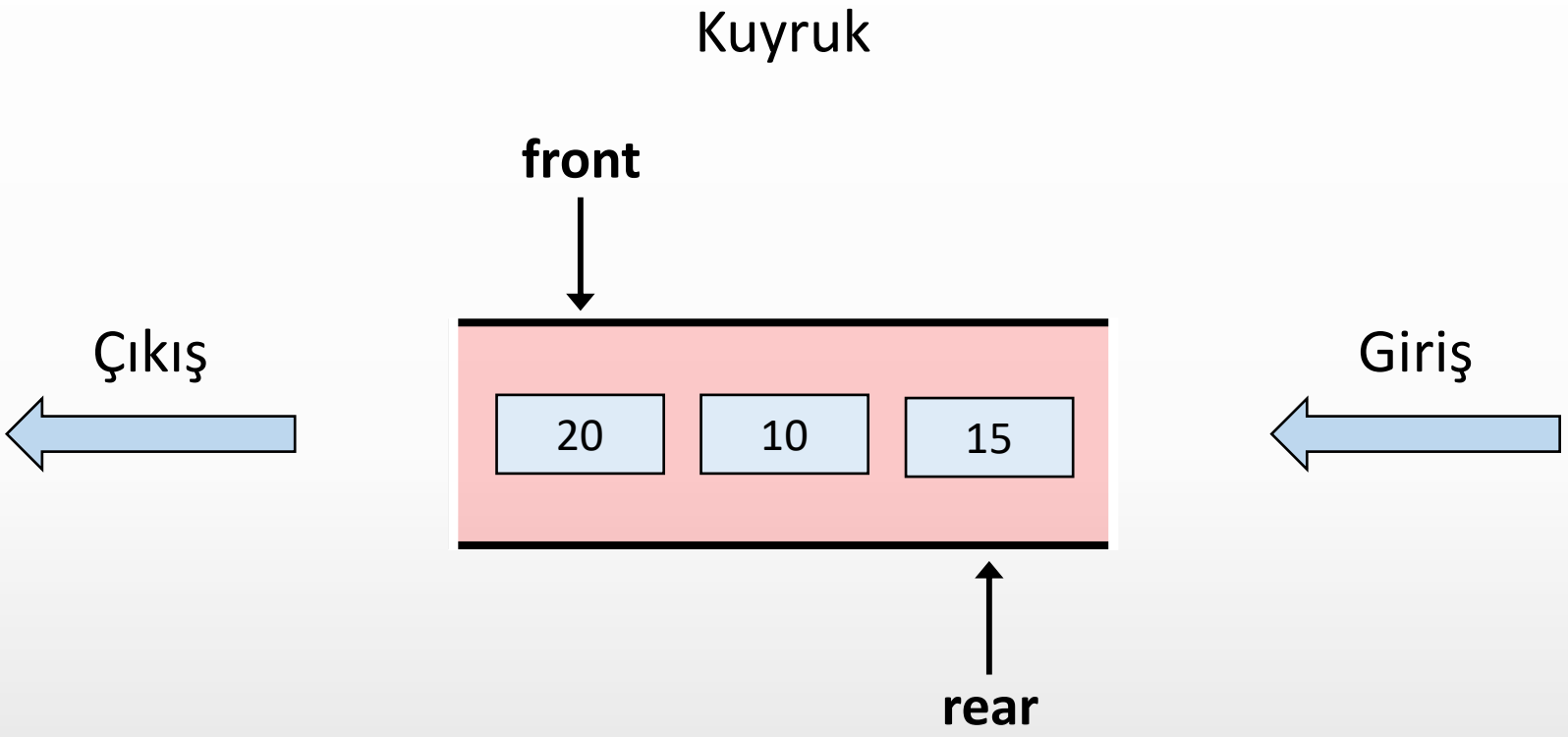
enqueue(15)



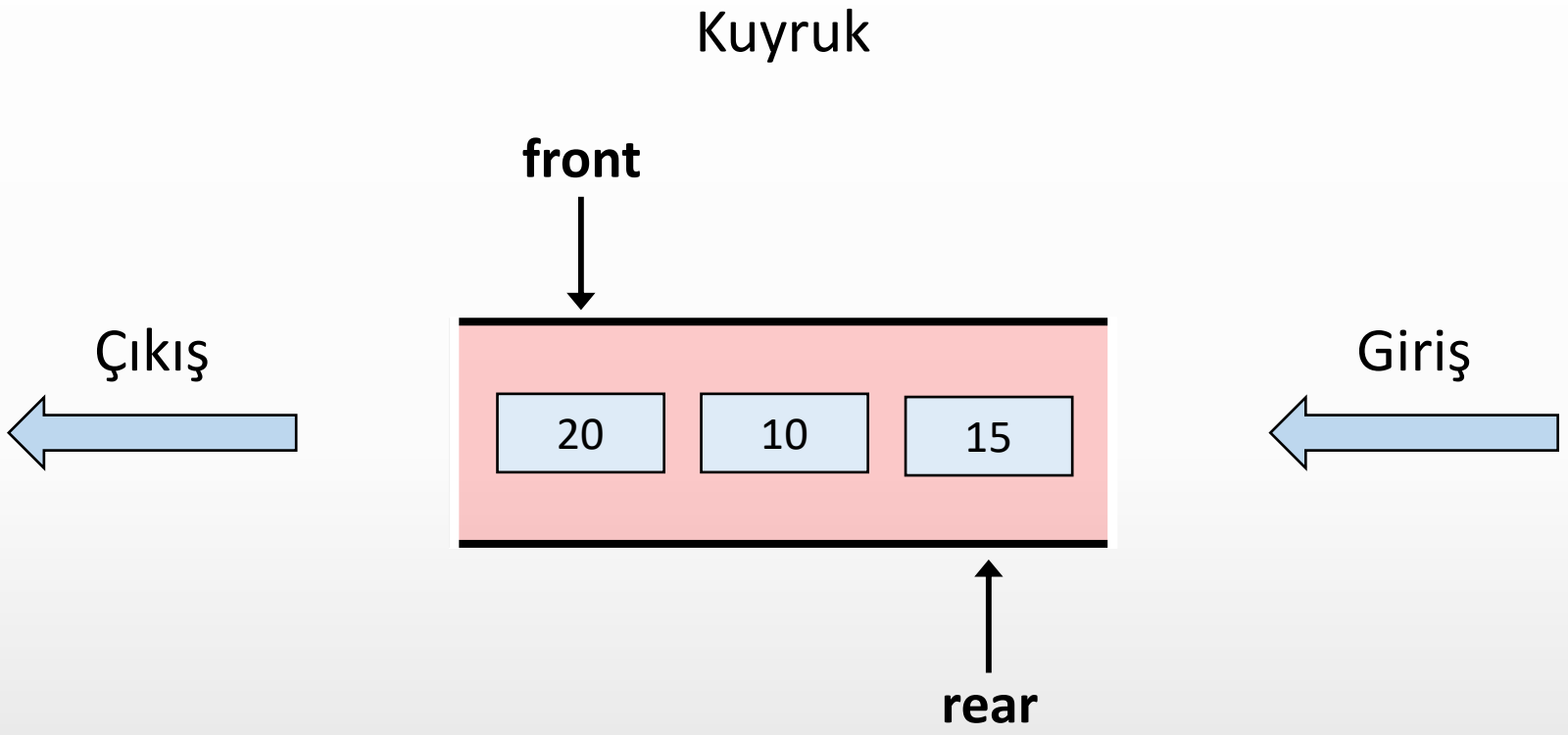
enqueue(15)



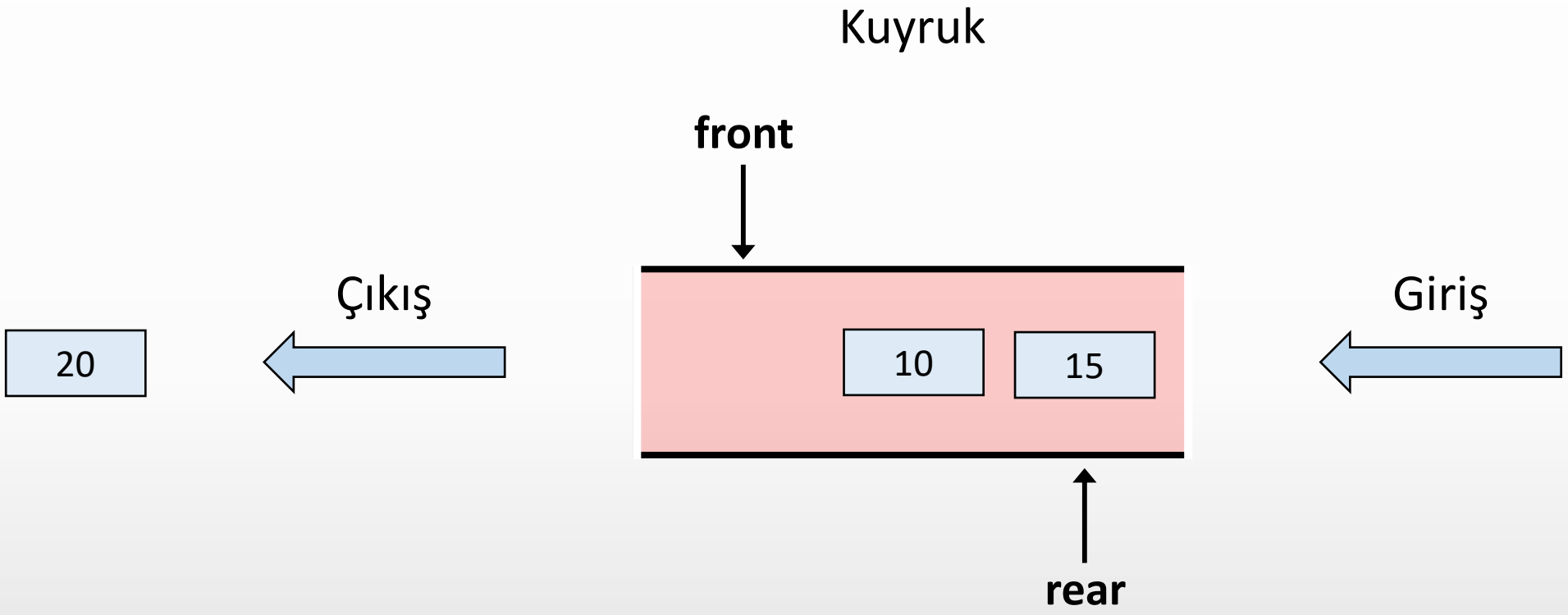
enqueue(15)



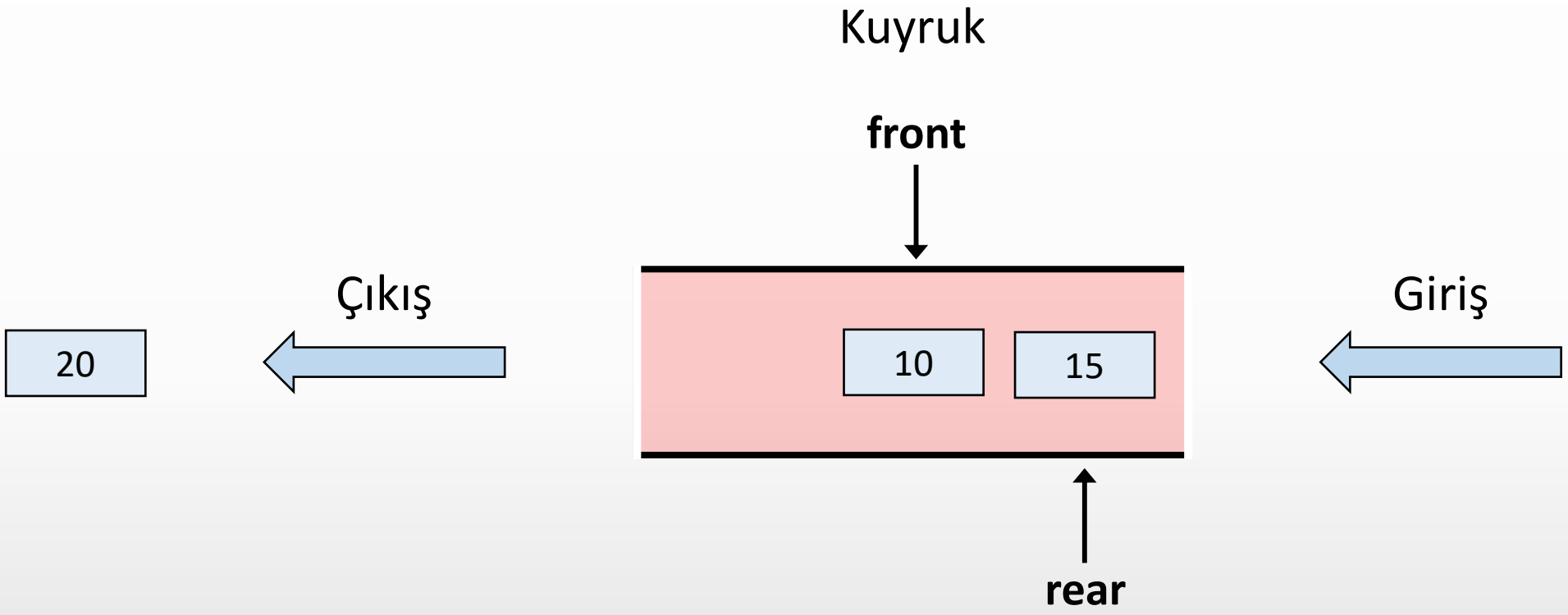
dequeue()

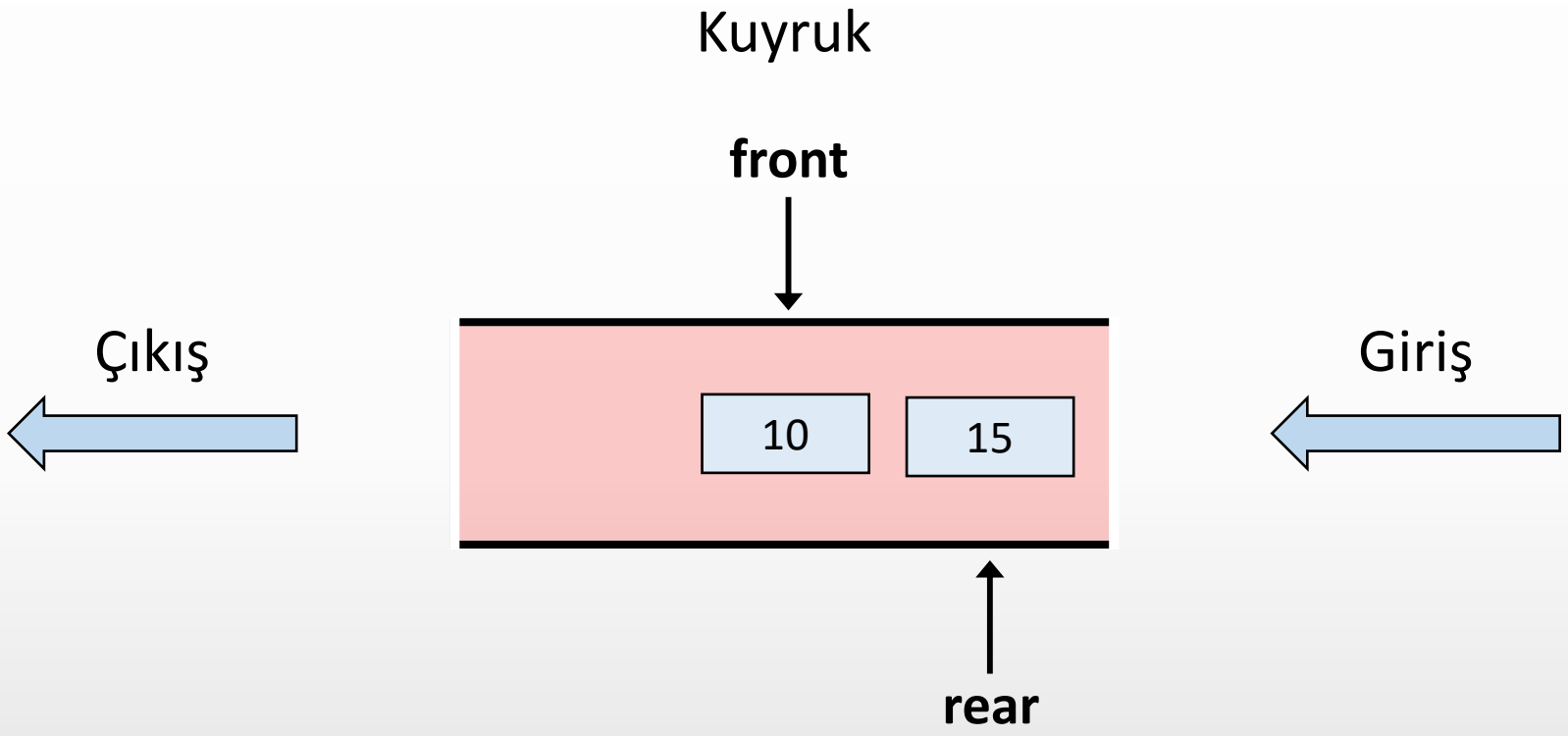


dequeue()

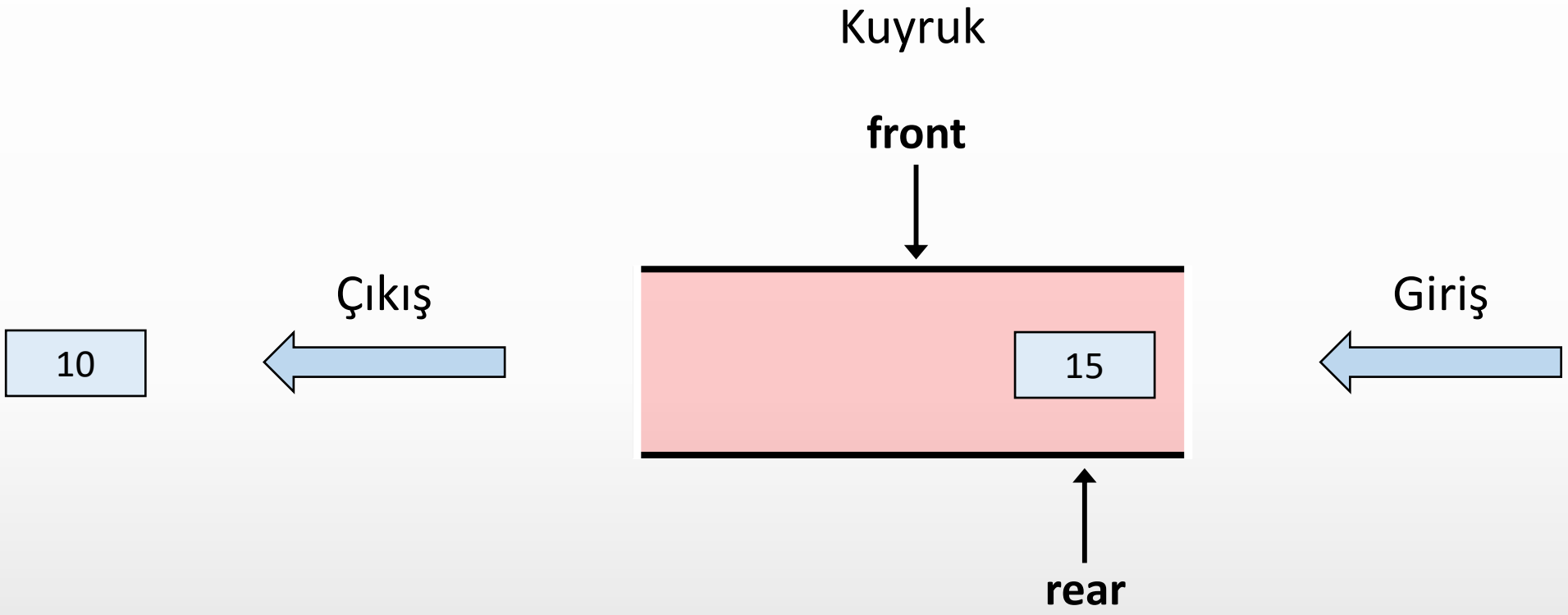


dequeue()

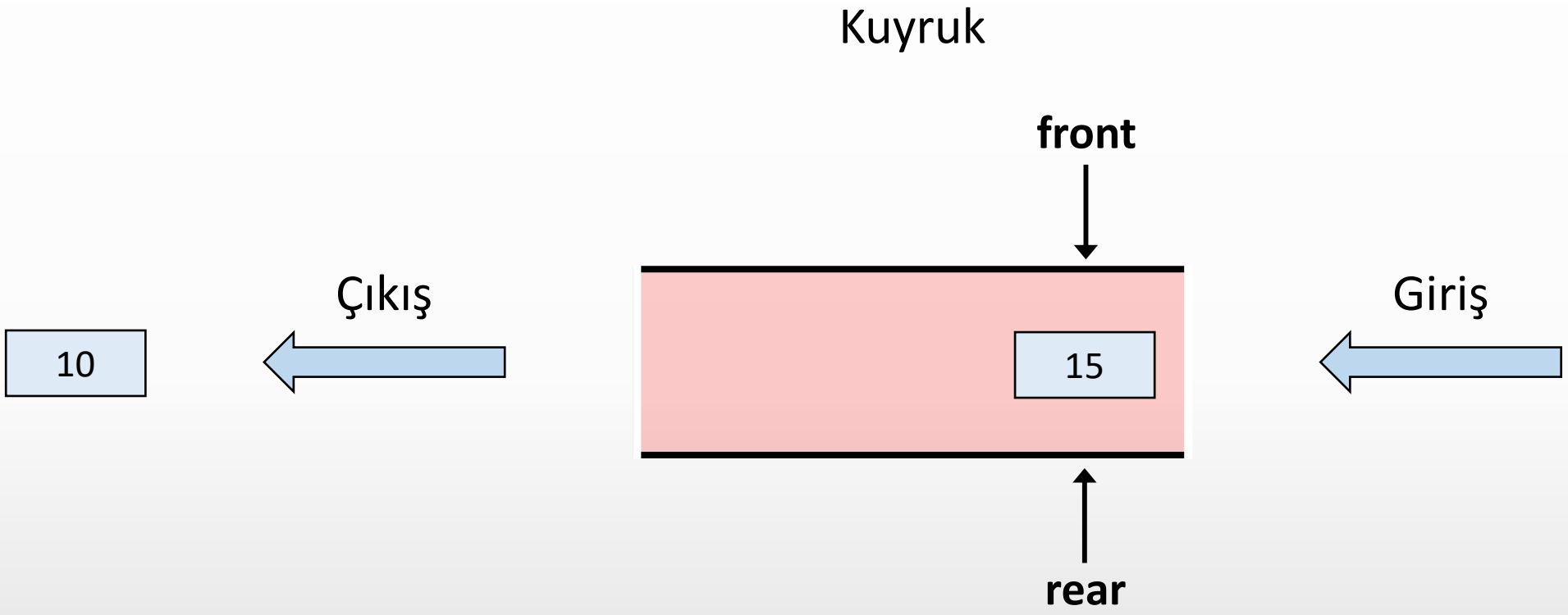


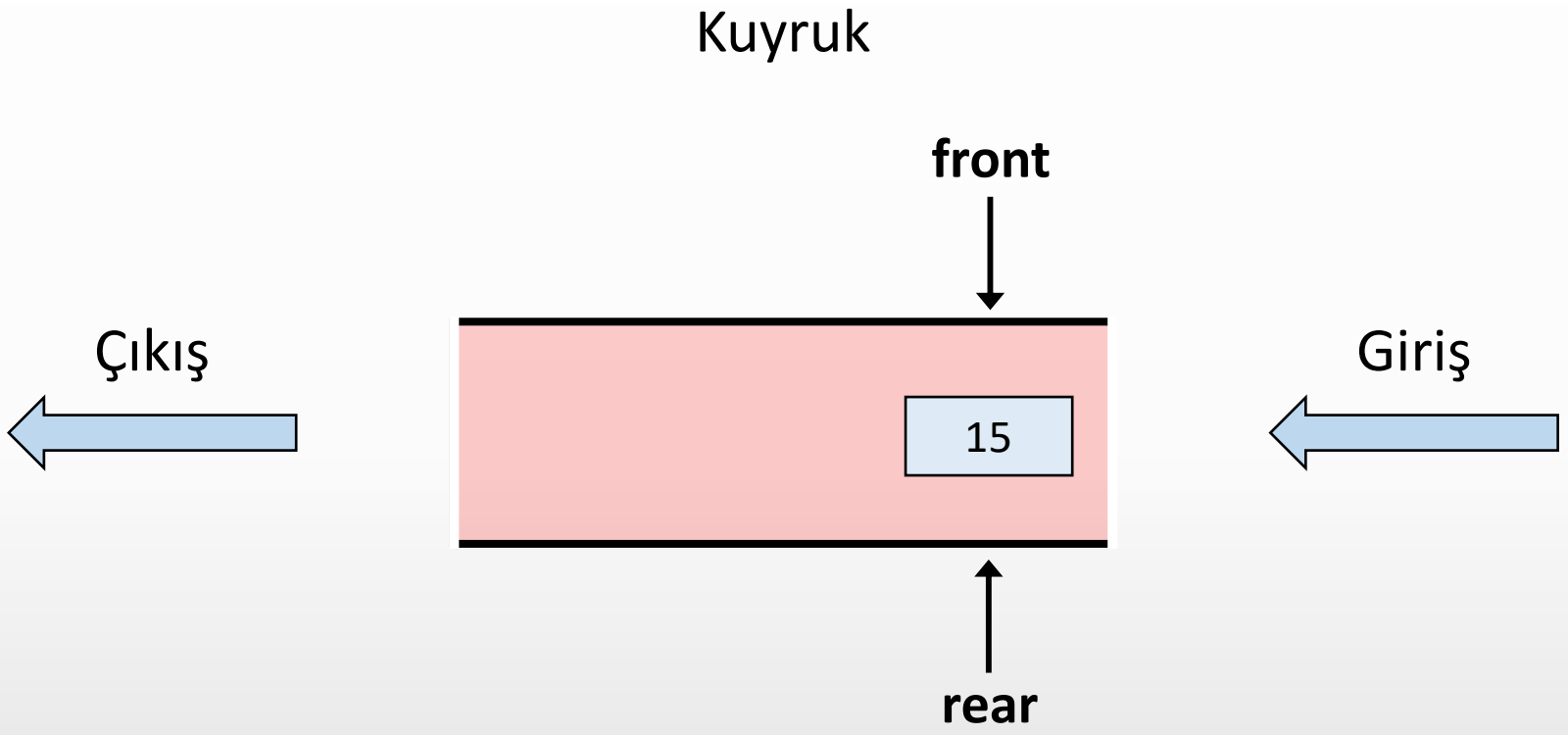

Kuyruk
front
Çıkış
10
15
Giriş
rear

dequeue()

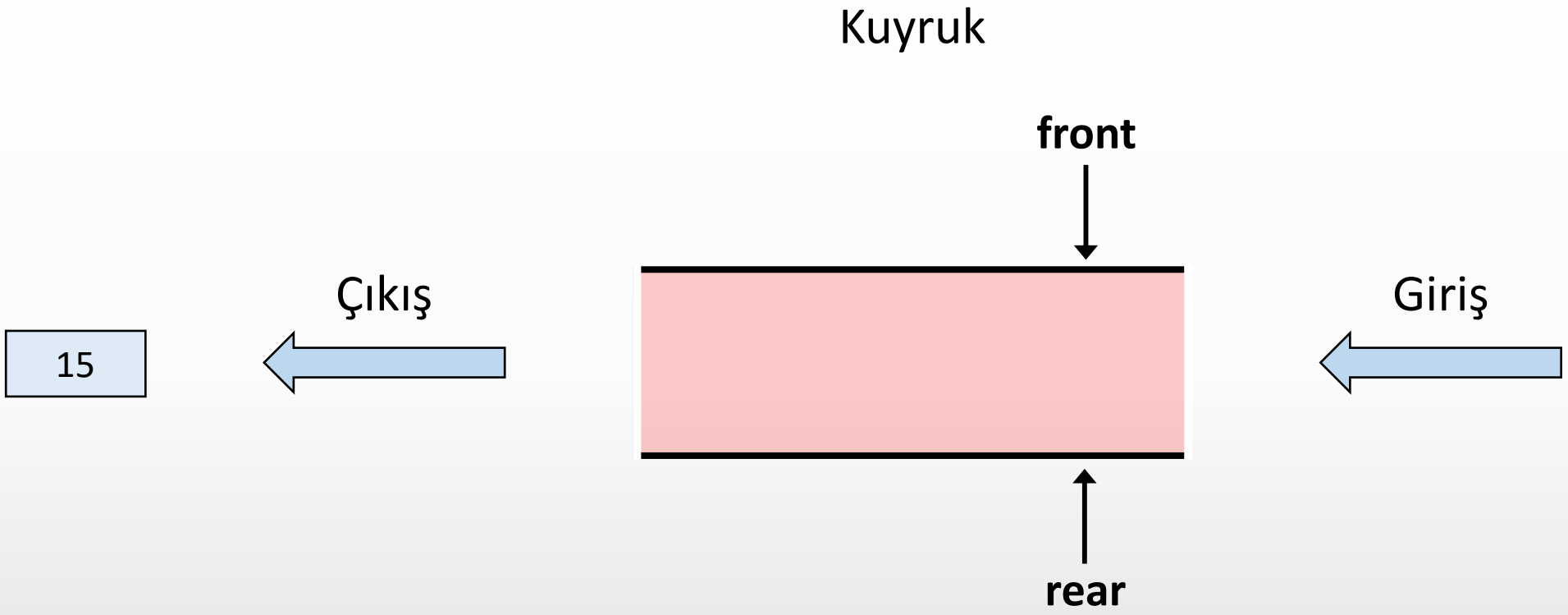


dequeue()




Kuyruk
front
Çıkış
15
Giriş
rear

dequeue()



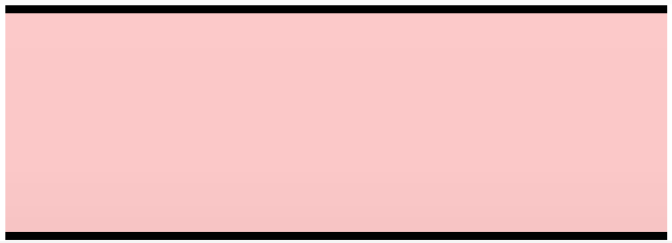
dequeue()

front ⟶ null

rear ⟶ null

Kuyruk

Çıkış



Giriş

front ⟶ **null**

rear ⟶ **null**

Dizi Temsili

- Kuyruk, dizi kullanılarak temsil edilebilir.
  - **Kuyruk:** Öğeleri saklayan dizinin adı.
  - **Ön (Front):** Kuyruğu temsil eden dizideki ilk öğeyi gösteren indis.
  - **Arka (Rear):** Kuyruğu temsil eden dizideki son öğeyi gösteren indis.
- Ön (Front) ve Arka (Rear) indisleri, kuyruğun başını ve sonunu işaret eder.
- Kuyruğa öğe eklerken Rear artar, öğe çıkartılırken Front artar.
- Dizi temsili basit ve hızlıdır, ancak sabit boyuta sahiptir.

Bağlı Liste Temsili

- Kuyruk bağlı listeler kullanılarak temsil edilebilir.
- Kuyruğu temsil etmek için aşağıdaki yapılar:
  - Bağlı liste
  - Ön ve Arka işaretçileri
  - Nesneler
- Her bir kuyruk öğesi, bir bağlı liste düğümüdür.
- Öğeleri dinamik olarak saklar, boyutu otomatik olarak değişir.
- Dizi temsiline göre daha karmaşıktır ve bellek yönetimi gerektirir.

Kuyruk Türleri

- Giriş Sınırlı Kuyruk (Input Restricted Queue)
- Çıkış Sınırlı Kuyruk (Output Restricted Queue)
- Dairesel Kuyruk (Circular Queue)
- Çift Uçlu Kuyruk (Double-Ended Queue veya Dequeue)
- Öncelikli Kuyruk (Priority Queue)

Kuyruk Türleri

- **Giriş Sınırlı Kuyruk (Input Restricted Queue)**
  - Basit bir kuyruktur.
  - Giriş sadece bir uçtan, çıkış işlemi her iki uçtan yapılabilir.
- Çıkış Sınırlı Kuyruk (Output Restricted Queue)
- Dairesel Kuyruk (Circular Queue)
- Çift Uçlu Kuyruk (Double-Ended Queue veya Dequeue)
- Öncelikli Kuyruk (Priority Queue)

Kuyruk Türleri

- Giriş Sınırlı Kuyruk (Input Restricted Queue)
- **Çıkış Sınırlı Kuyruk (Output Restricted Queue)**
  - Basit bir kuyruktur.
  - Giriş her iki uçtan, çıkış sadece bir uçtan yapılabilir.
- Dairesel Kuyruk (Circular Queue)
- Çift Uçlu Kuyruk (Double-Ended Queue veya Dequeue)
- Öncelikli Kuyruk (Priority Queue)

Kuyruk Türleri

- Giriş Sınırlı Kuyruk (Input Restricted Queue)
- Çıkış Sınırlı Kuyruk (Output Restricted Queue)
- **Dairesel Kuyruk (Circular Queue)**
  - Özel bir kuyruk türüdür.
  - Son öğenin, ilk öğeye bağlandığı bir döngü oluşturur.
  - İşlemler FIFO (İlk Giren, İlk Çıkar) düzeninde gerçekleştirilir.
- Çift Uçlu Kuyruk (Double-Ended Queue veya Dequeue)
- Öncelikli Kuyruk (Priority Queue)

Kuyruk Türleri

- Giriş Sınırlı Kuyruk (Input Restricted Queue)
- Çıkış Sınırlı Kuyruk (Output Restricted Queue)
- Dairesel Kuyruk (Circular Queue)
- **Çift Uçlu Kuyruk (Double-Ended Queue veya Dequeue)**
  - Giriş ve çıkış işlemleri her iki uçtan da yapılabilir.
- Öncelikli Kuyruk (Priority Queue)

Kuyruk Türleri

- Giriş Sınırlı Kuyruk (Input Restricted Queue)
- Çıkış Sınırlı Kuyruk (Output Restricted Queue)
- Dairesel Kuyruk (Circular Queue)
- Çift Uçlu Kuyruk (Double-Ended Queue veya Dequeue)
- **Öncelikli Kuyruk (Priority Queue)**
  - Öğelere atanan önceliğe göre erişilen özel bir kuyruk türüdür.
  - Öğelerin önceliği, erişim sırasını belirler.

Kuyrukta Temel İşlemler

- enqueue(), kuyruğun sonuna yeni bir öğe ekler.
- dequeue(), kuyruğun başındaki öğeyi kuyruktan çıkarır.
- peek(), front(), kuyruğun başındaki öğeyi döndürür, kuyruktan çıkarmaz.
- rear(), kuyruğun sonundaki öğeyi döndürür, kuyruktan çıkarmaz.
- isFull(), kuyruğun dolu olup olmadığını kontrol eder.
- isNull(), kuyruğun boş olup olmadığını kontrol eder.

Enqueue()

- İlk adımda, kuyruğun dolu olup olmadığı kontrol edilir.
- Kuyruk doluysa, taşma (overflow) hatası döndürülür ve işlem sonlandırılır.
- Dolu değilse, arka işaretçi tarafından işaret edilen konuma öğe eklenir.
- Arka işaretçi sonraki boş alana işaret etmek için artırılır.
- İşlem sonucu "başarılı" olarak döndürülür.

Enqueue İşlemi

```
public void enqueue(int veri) {
  Dugum gecici = new Dugum(veri);
  if(bosMu()) {
    front = gecici;
  }
  else {
    rear.sonraki = gecici;
  }
  rear = gecici;
  uzunluk++;
}
```

front ⟶ null

rear ⟶ null

uzunluk = 0

enqueue(10)

```
public void enqueue(int veri) {
  Dugum gecici = new Dugum(veri);
  if(bosMu()) {
    front = gecici;
  }
  else {
    rear.sonraki = gecici;
  }
  rear = gecici;
  uzunluk++;
}
```

front ⟶ null

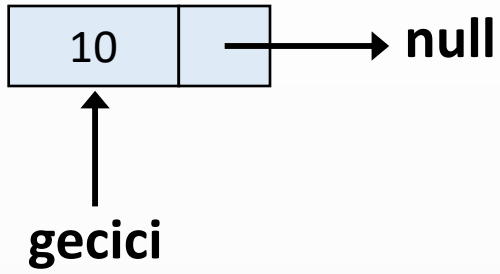
rear ⟶ null

uzunluk = 0

veri = 10

enqueue(10)

```
public void enqueue(int veri) {
  Dugum gecici = new Dugum(veri);
  if(bosMu()) {
    front = gecici;
  }
  else {
    rear.sonraki = gecici;
  }
  rear = gecici;
  uzunluk++;
}
```



front ⟶ **null**

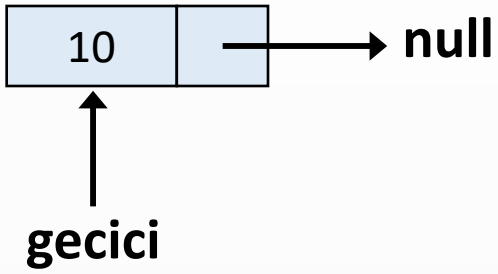
rear ⟶ **null**

uzunluk = 0

veri = 10

enqueue(10)

```
public void enqueue(int veri) {
  Dugum gecici = new Dugum(veri);
  if(bosMu()) {
    front = gecici;
  }
  else {
    rear.sonraki = gecici;
  }
  rear = gecici;
  uzunluk++;
}
```



front ⟶ **null**

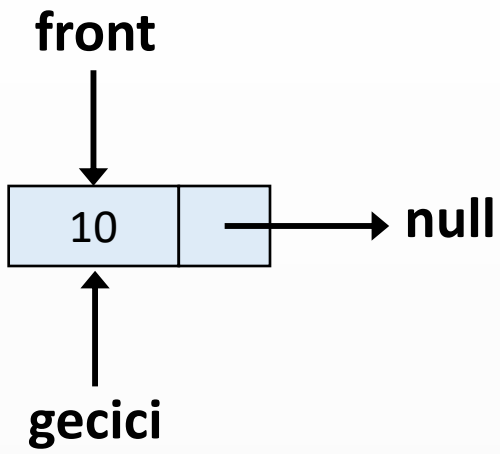
rear ⟶ **null**

uzunluk = 0

veri = 10

enqueue(10)

```
public void enqueue(int veri) {
  Dugum gecici = new Dugum(veri);
  if(bosMu()) {
    front = gecici;
  }
  else {
    rear.sonraki = gecici;
  }
  rear = gecici;
  uzunluk++;
}
```



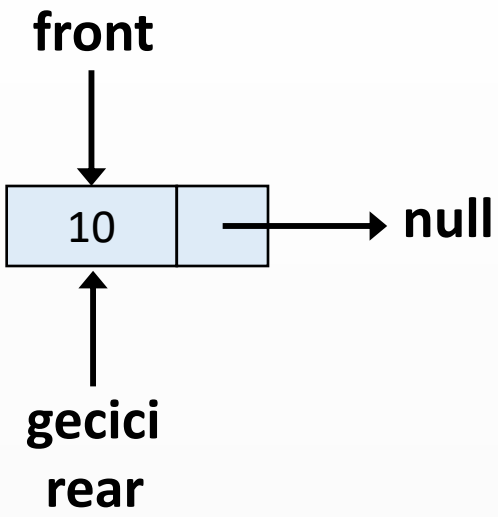
rear ⟶ null

uzunluk = 0

veri = 10

enqueue(10)

```
public void enqueue(int veri) {
  Dugum gecici = new Dugum(veri);
  if(bosMu()) {
    front = gecici;
  }
  else {
    rear.sonraki = gecici;
  }
  rear = gecici;
  uzunluk++;
}
```

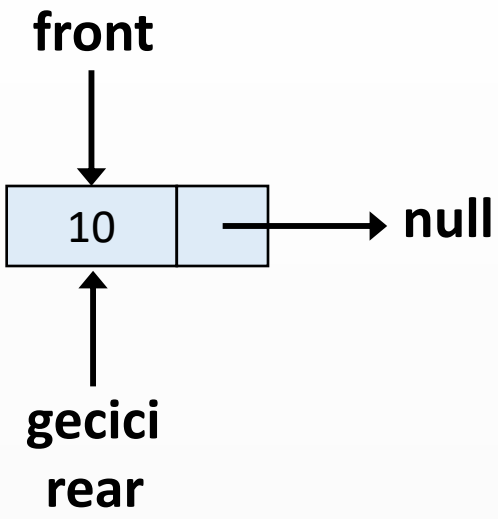


uzunluk = 0

veri = 10

enqueue(10)

```
public void enqueue(int veri) {
  Dugum gecici = new Dugum(veri);
  if(bosMu()) {
    front = gecici;
  }
  else {
    rear.sonraki = gecici;
  }
  rear = gecici;
  uzunluk++;
}
```

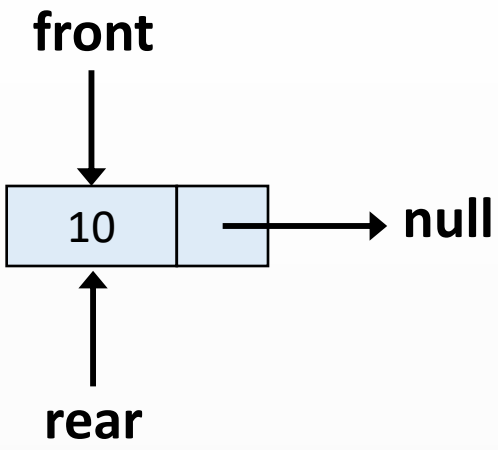


uzunluk = 1

veri = 10

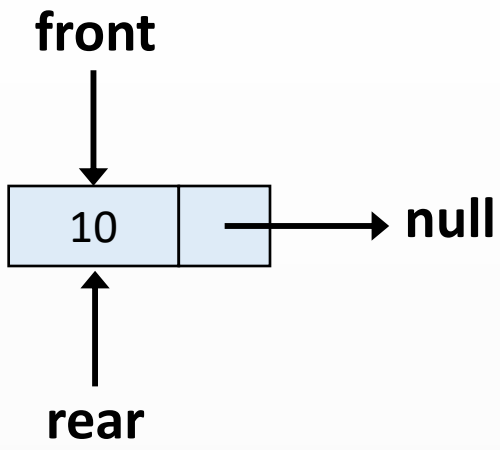
enqueue(10)


```
public void enqueue(int veri) {
  Dugum gecici = new Dugum(veri);
  if(bosMu()) {
    front = gecici;
  }
  else {
    rear.sonraki = gecici;
  }
  rear = gecici;
  uzunluk++;
}
```



uzunluk = 1

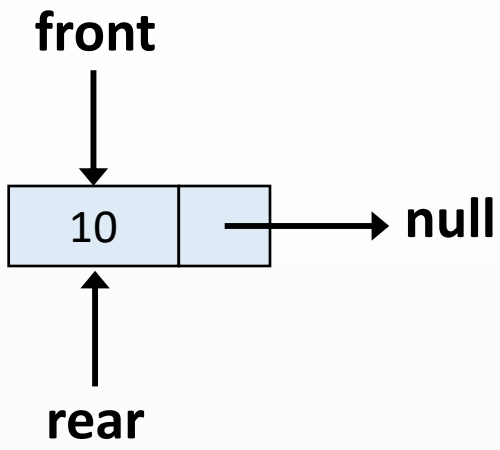
```
public void enqueue(int veri) {
  Dugum gecici = new Dugum(veri);
  if(bosMu()) {
    front = gecici;
  }
  else {
    rear.sonraki = gecici;
  }
  rear = gecici;
  uzunluk++;
}
```



uzunluk = 1

enqueue(15)

```
public void enqueue(int veri) {
  Dugum gecici = new Dugum(veri);
  if(bosMu()) {
    front = gecici;
  }
  else {
    rear.sonraki = gecici;
  }
  rear = gecici;
  uzunluk++;
}
```

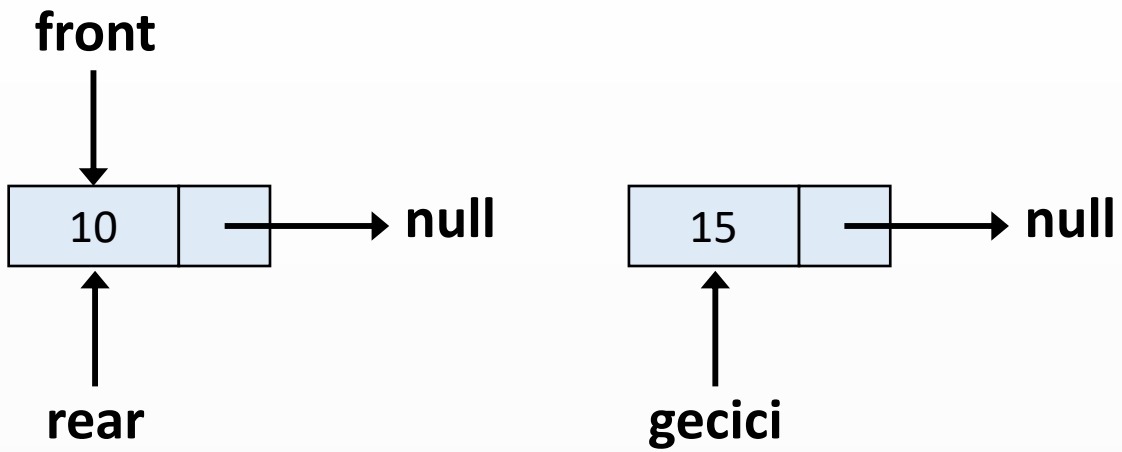


uzunluk = 1

veri = 15

enqueue(15)

```
public void enqueue(int veri) {
  Dugum gecici = new Dugum(veri);
  if(bosMu()) {
    front = gecici;
  }
  else {
    rear.sonraki = gecici;
  }
  rear = gecici;
  uzunluk++;
}
```

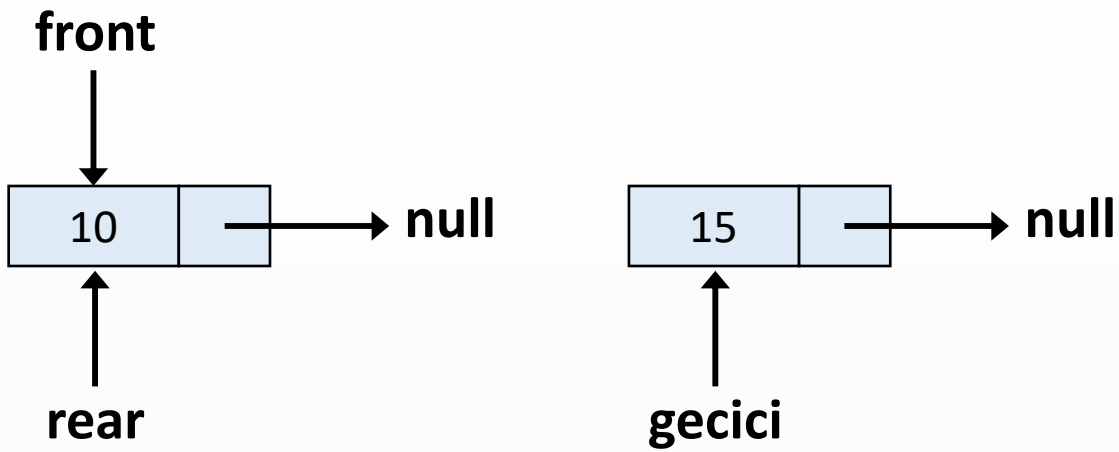


uzunluk = 1

veri = 15

enqueue(15)

```
public void enqueue(int veri) {
  Dugum gecici = new Dugum(veri);
  if(bosMu()) {
    front = gecici;
  }
  else {
    rear.sonraki = gecici;
  }
  rear = gecici;
  uzunluk++;
}
```

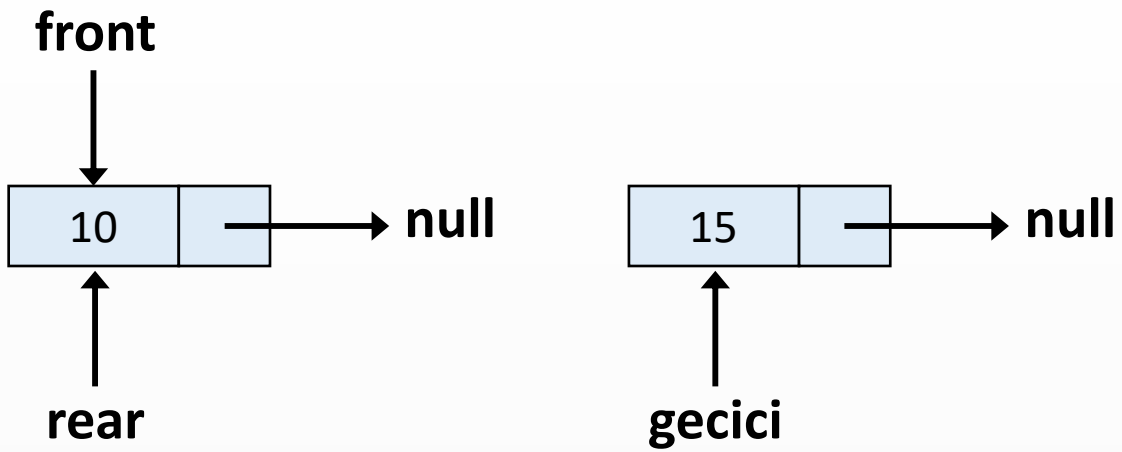


uzunluk = 1

veri = 15

enqueue(15)

```
public void enqueue(int veri) {
  Dugum gecici = new Dugum(veri);
  if(bosMu()) {
    front = gecici;
  }
  else {
    rear.sonraki = gecici;
  }
  rear = gecici;
  uzunluk++;
}
```

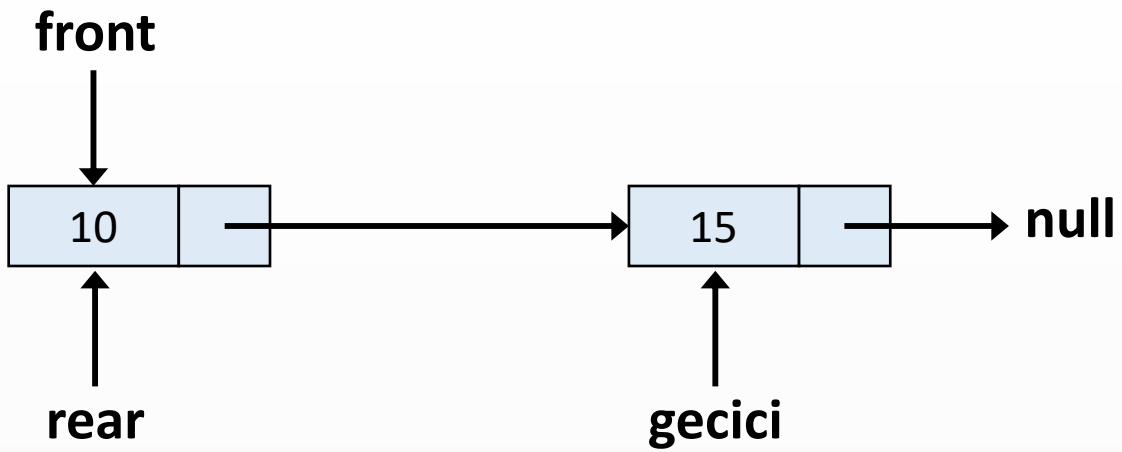


uzunluk = 1

veri = 15

enqueue(15)

```
public void enqueue(int veri) {
  Dugum gecici = new Dugum(veri);
  if(bosMu()) {
    front = gecici;
  }
  else {
    rear.sonraki = gecici;
  }
  rear = gecici;
  uzunluk++;
}
```

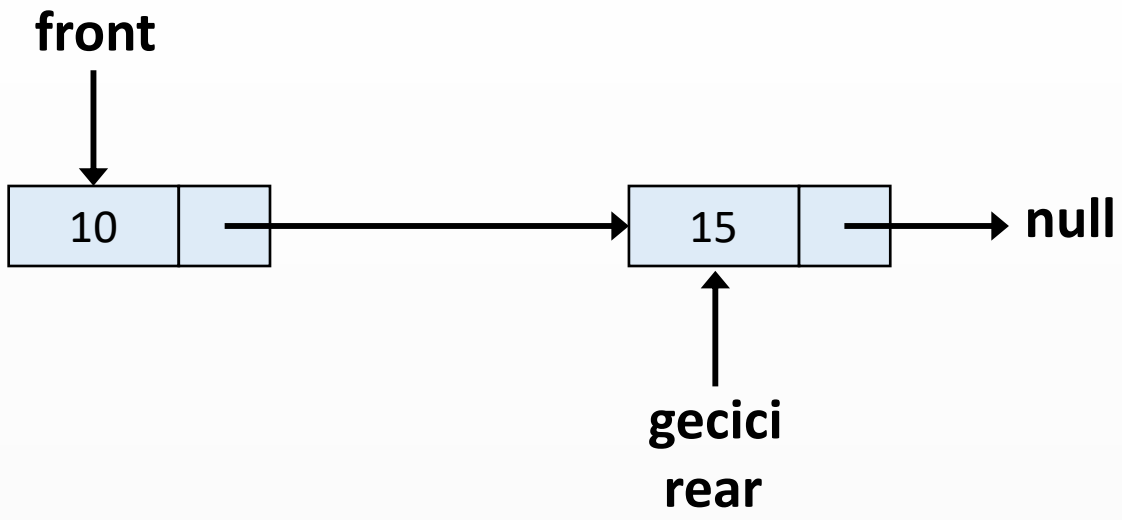


uzunluk = 1

veri = 15

enqueue(15)

```
public void enqueue(int veri) {
  Dugum gecici = new Dugum(veri);
  if(bosMu()) {
    front = gecici;
  }
  else {
    rear.sonraki = gecici;
  }
  rear = gecici;
  uzunluk++;
}
```

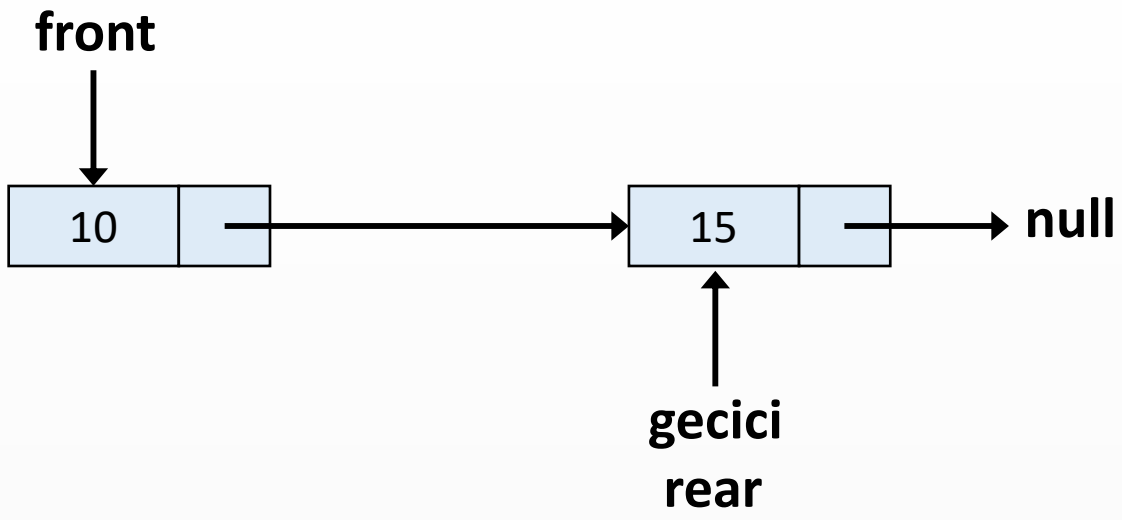


uzunluk = 1

veri = 15

enqueue(15)

```
public void enqueue(int veri) {
  Dugum gecici = new Dugum(veri);
  if(bosMu()) {
    front = gecici;
  }
  else {
    rear.sonraki = gecici;
  }
  rear = gecici;
  uzunluk++;
}
```

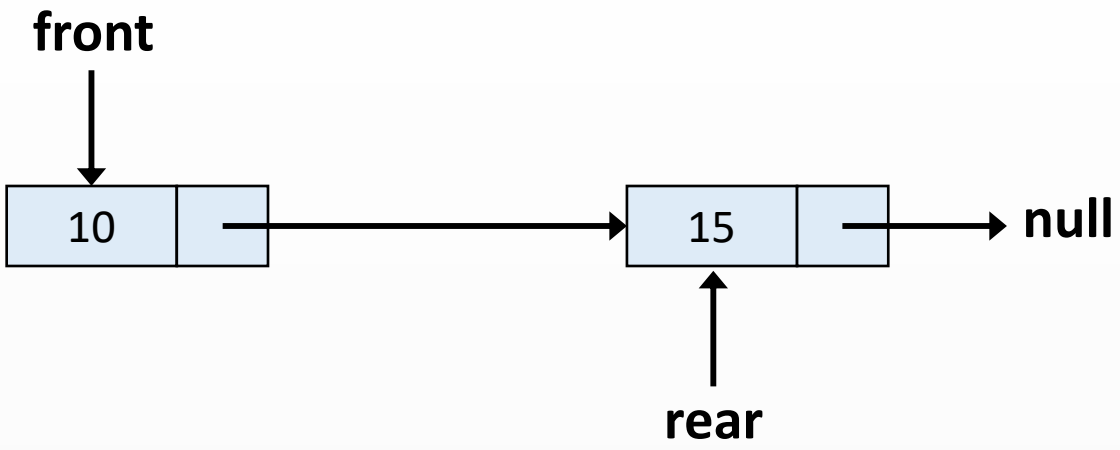


uzunluk = 2

veri = 15

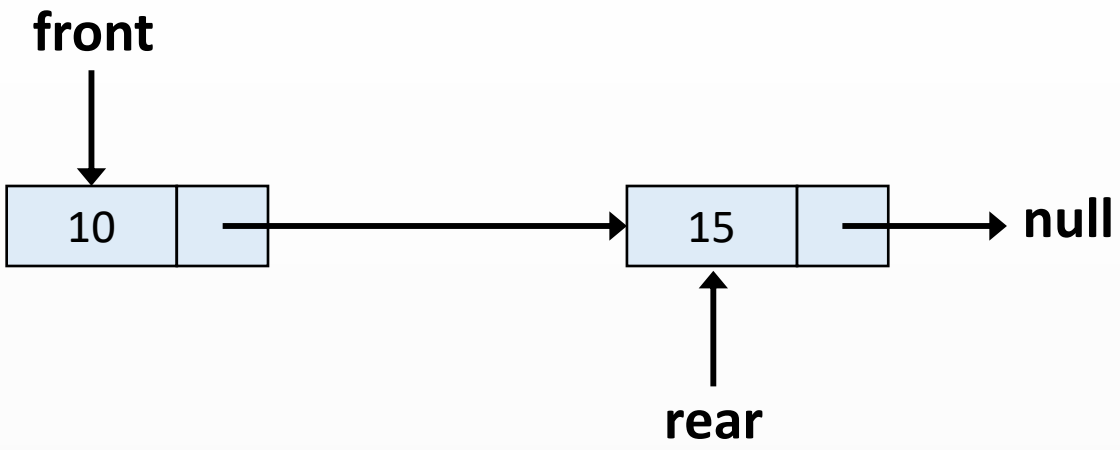
enqueue(15)

```
public void enqueue(int veri) {
  Dugum gecici = new Dugum(veri);
  if(bosMu()) {
    front = gecici;
  }
  else {
    rear.sonraki = gecici;
  }
  rear = gecici;
  uzunluk++;
}
```



uzunluk = 2

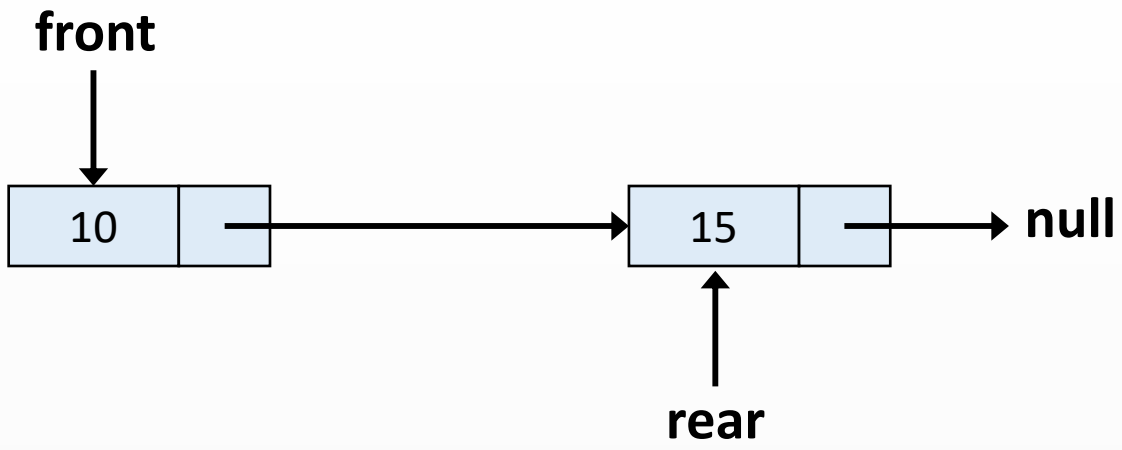
```
public void enqueue(int veri) {
  Dugum gecici = new Dugum(veri);
  if(bosMu()) {
    front = gecici;
  }
  else {
    rear.sonraki = gecici;
  }
  rear = gecici;
  uzunluk++;
}
```



uzunluk = 2

enqueue(20)

```
public void enqueue(int veri) {
  Dugum gecici = new Dugum(veri);
  if(bosMu()) {
    front = gecici;
  }
  else {
    rear.sonraki = gecici;
  }
  rear = gecici;
  uzunluk++;
}
```

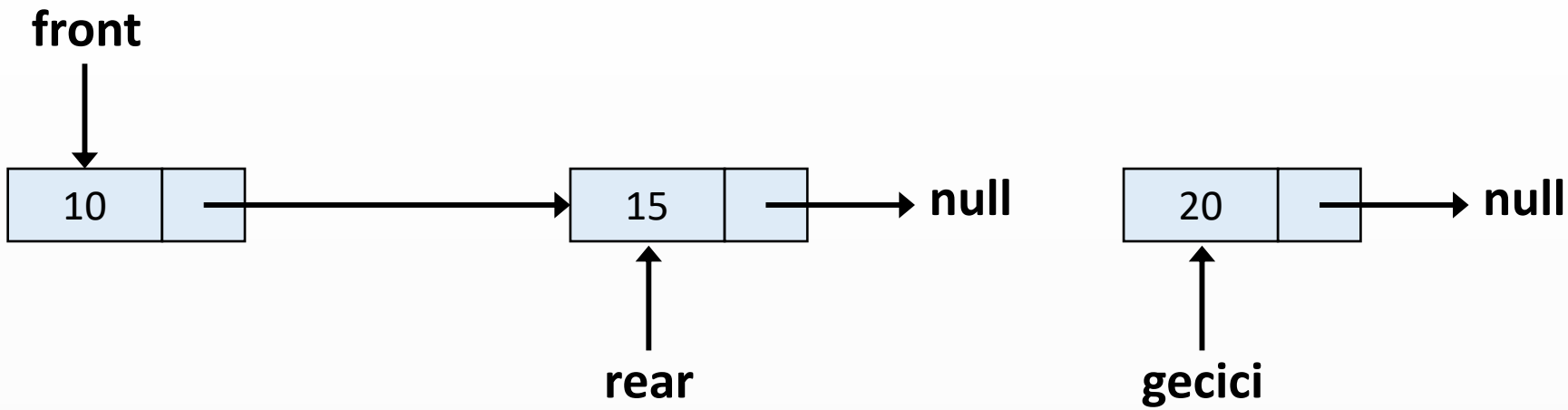


uzunluk = 2

veri = 20

enqueue(20)

```
public void enqueue(int veri) {
  Dugum gecici = new Dugum(veri);
  if(bosMu()) {
    front = gecici;
  }
  else {
    rear.sonraki = gecici;
  }
  rear = gecici;
  uzunluk++;
}
```

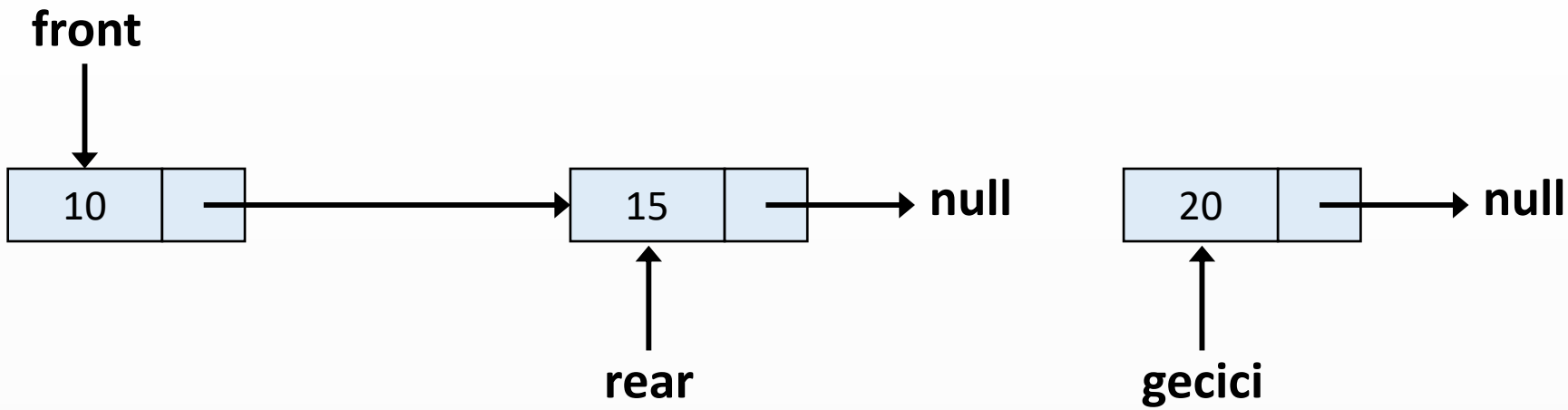


uzunluk = 2

veri = 20

enqueue(20)

```
public void enqueue(int veri) {
  Dugum gecici = new Dugum(veri);
  if(bosMu()) {
    front = gecici;
  }
  else {
    rear.sonraki = gecici;
  }
  rear = gecici;
  uzunluk++;
}
```

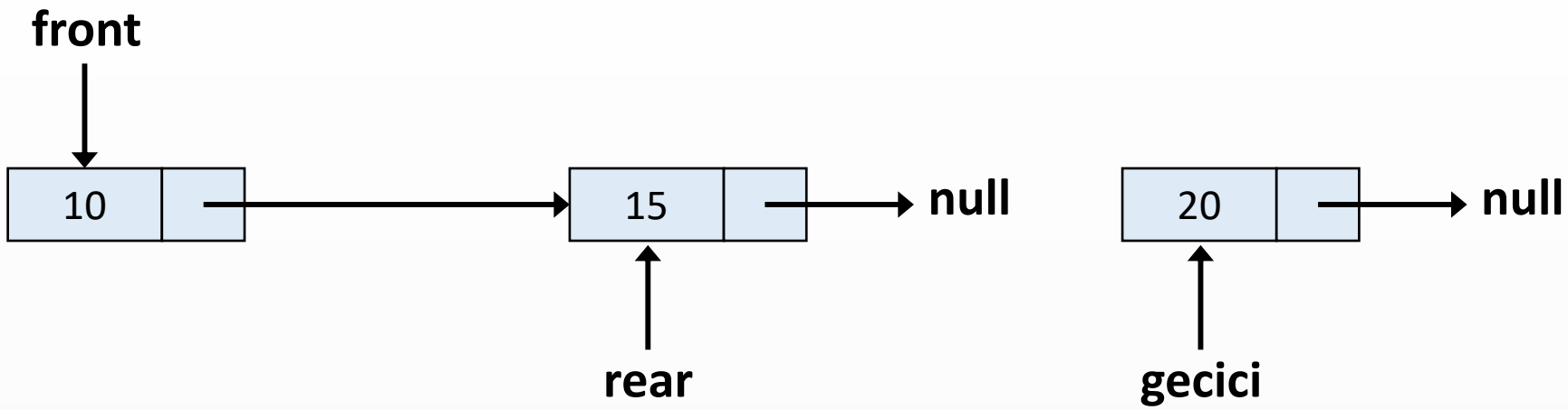


uzunluk = 2

veri = 20

enqueue(20)

```
public void enqueue(int veri) {
  Dugum gecici = new Dugum(veri);
  if(bosMu()) {
    front = gecici;
  }
  else {
    rear.sonraki = gecici;
  }
  rear = gecici;
  uzunluk++;
}
```

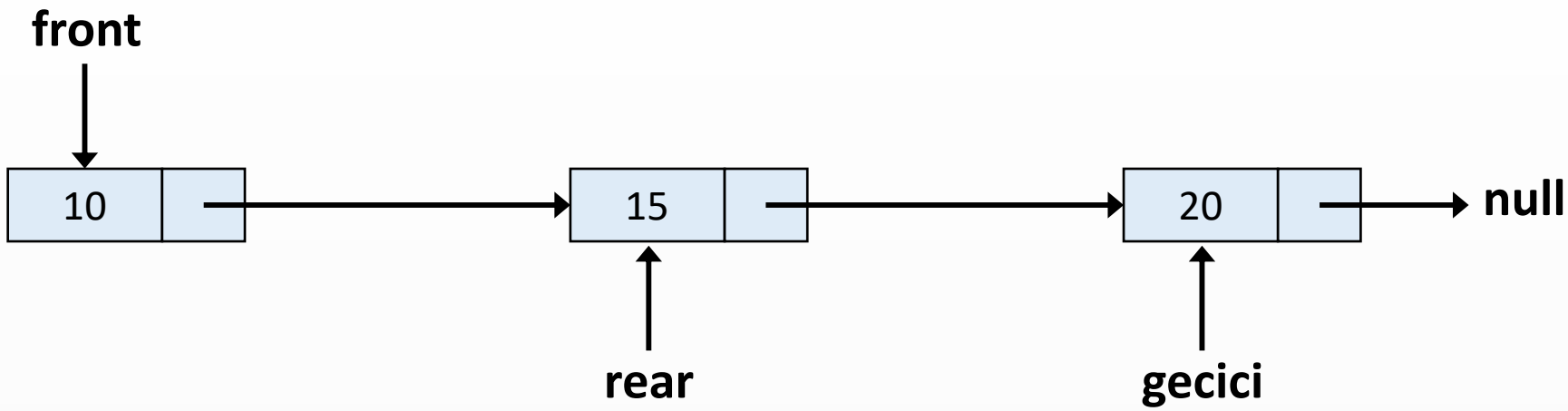


uzunluk = 2

veri = 20

enqueue(20)

```
public void enqueue(int veri) {
  Dugum gecici = new Dugum(veri);
  if(bosMu()) {
    front = gecici;
  }
  else {
    rear.sonraki = gecici;
  }
  rear = gecici;
  uzunluk++;
}
```

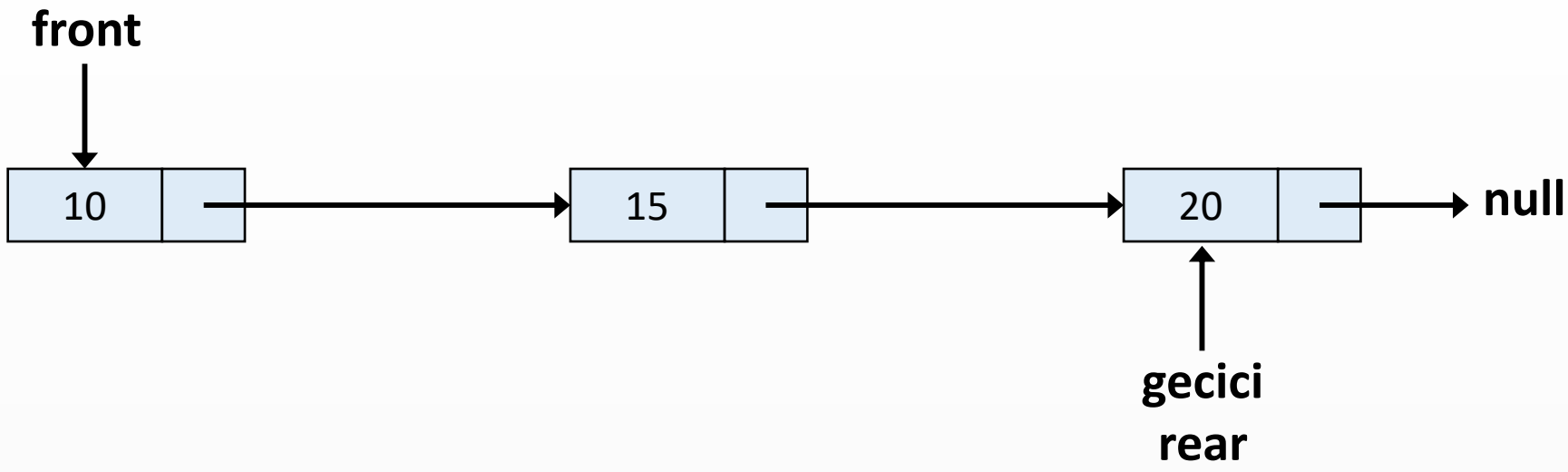


uzunluk = 2

veri = 20

enqueue(20)

```
public void enqueue(int veri) {
  Dugum gecici = new Dugum(veri);
  if(bosMu()) {
    front = gecici;
  }
  else {
    rear.sonraki = gecici;
  }
  rear = gecici;
  uzunluk++;
}
```

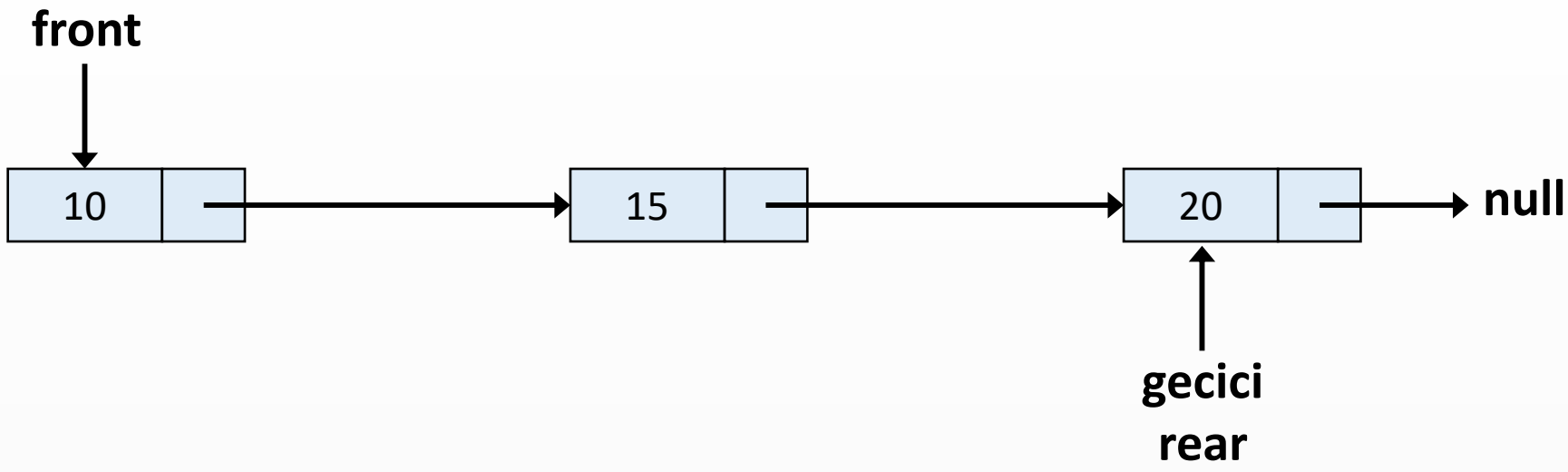


uzunluk = 2

veri = 20

enqueue(20)

```
public void enqueue(int veri) {
  Dugum gecici = new Dugum(veri);
  if(bosMu()) {
    front = gecici;
  }
  else {
    rear.sonraki = gecici;
  }
  rear = gecici;
  uzunluk++;
}
```

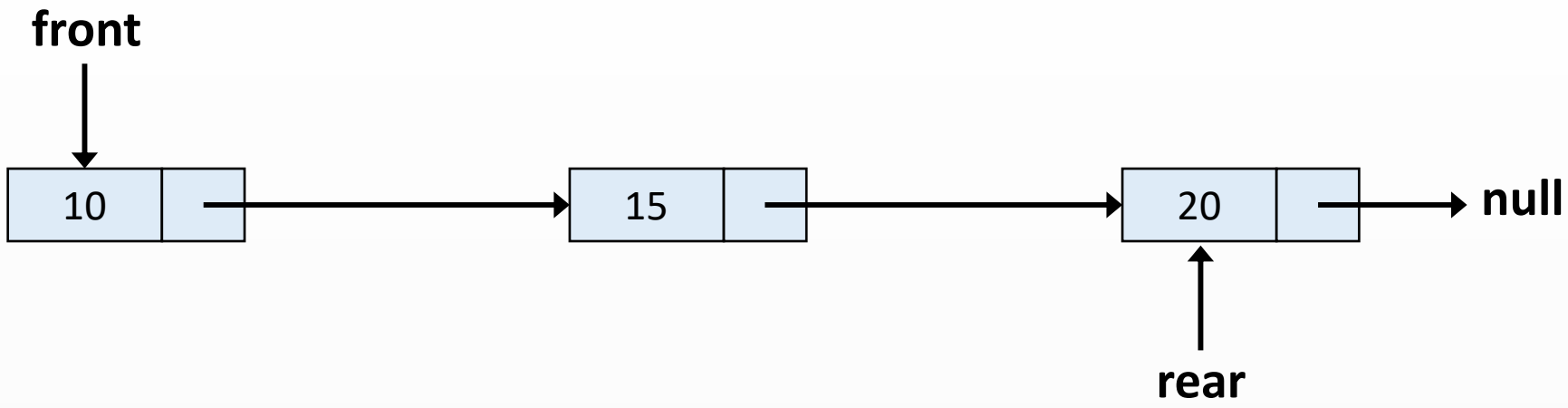


uzunluk = 3

veri = 20

enqueue(20)

```
public void enqueue(int veri) {
  Dugum gecici = new Dugum(veri);
  if(bosMu()) {
    front = gecici;
  }
  else {
    rear.sonraki = gecici;
  }
  rear = gecici;
  uzunluk++;
}
```



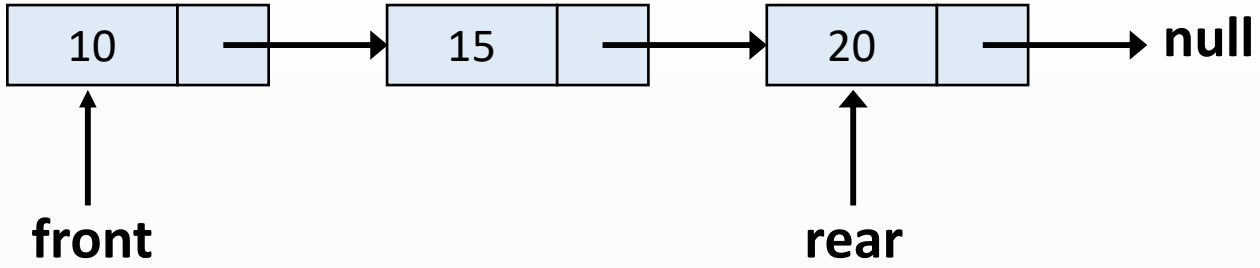
uzunluk = 3

```
public void enqueue(int veri) {
  Dugum gecici = new Dugum(veri);
  if(bosMu()) {
    front = gecici;
  }
  else {
    rear.sonraki = gecici;
  }
  rear = gecici;
  uzunluk++;
}
```

Dequeue()

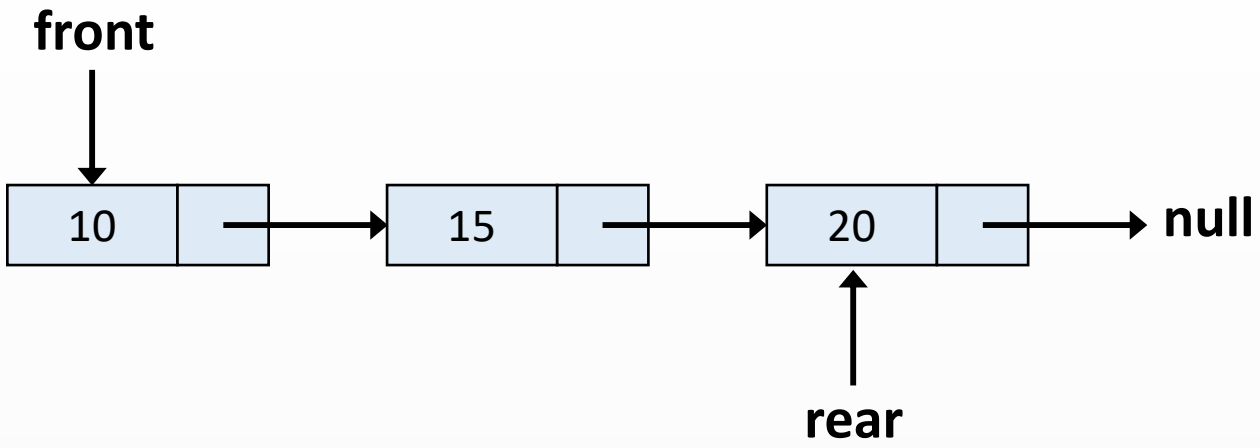
- İlk adımda, kuyruğun boş olup olmadığı kontrol edilir.
- Kuyruk boşsa, taşma (underflow) hatası döndürülür ve işlem sonlandırılır.
- Boş değilse, ön işaretçisi tarafından işaret edilen öğeye erişilir.
- Ön işaretçisi bir sonraki öğeye işaret etmesi için artırılır.
- İşlem sonucu "başarılı" olarak döndürülür.

Dequeue İşlemi



uzunluk = 3

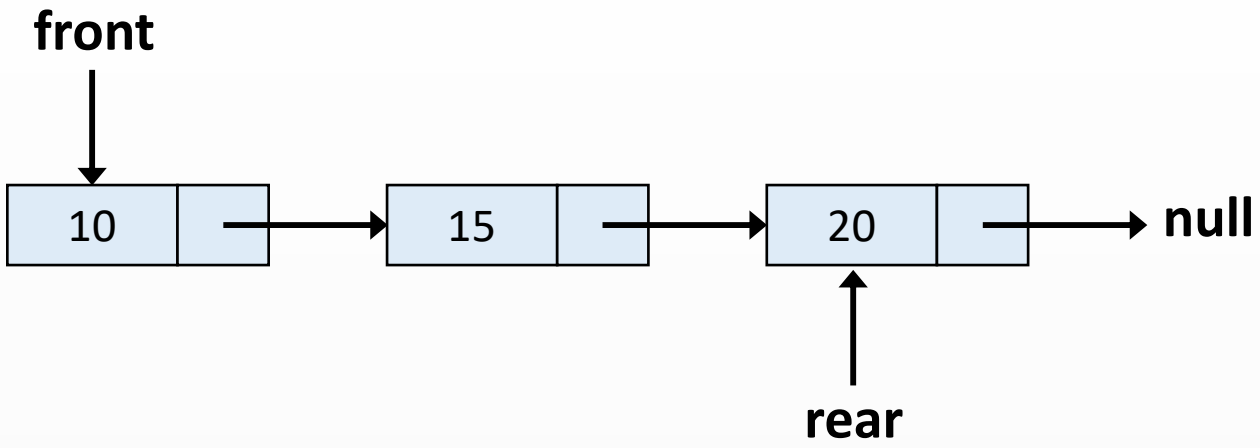
```
public int dequeue() {
  if(bosMu()) {
     throw new NoSuchElementException();
  }
  int sonuc = front.veri;
  front = front.sonraki;
  if(front == null) {
     rear = null;
  }
  uzunluk--;
  return sonuc;
}
```



uzunluk = 3

dequeue()

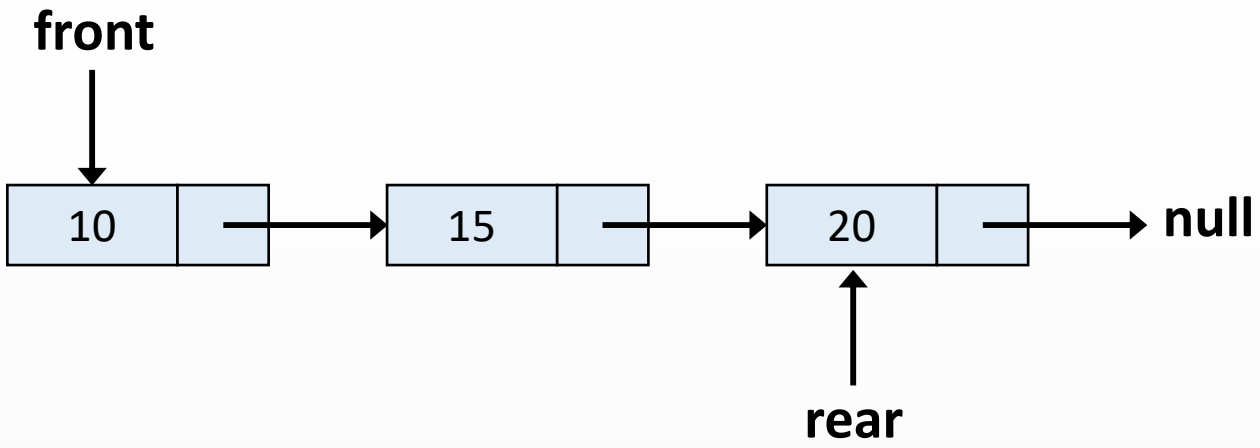
```
public int dequeue() {
  if(bosMu()) {
     throw new NoSuchElementException();
  }
  int sonuc = front.veri;
  front = front.sonraki;
  if(front == null) {
     rear = null;
  }
  uzunluk--;
  return sonuc;
}
```



uzunluk = 3

dequeue()

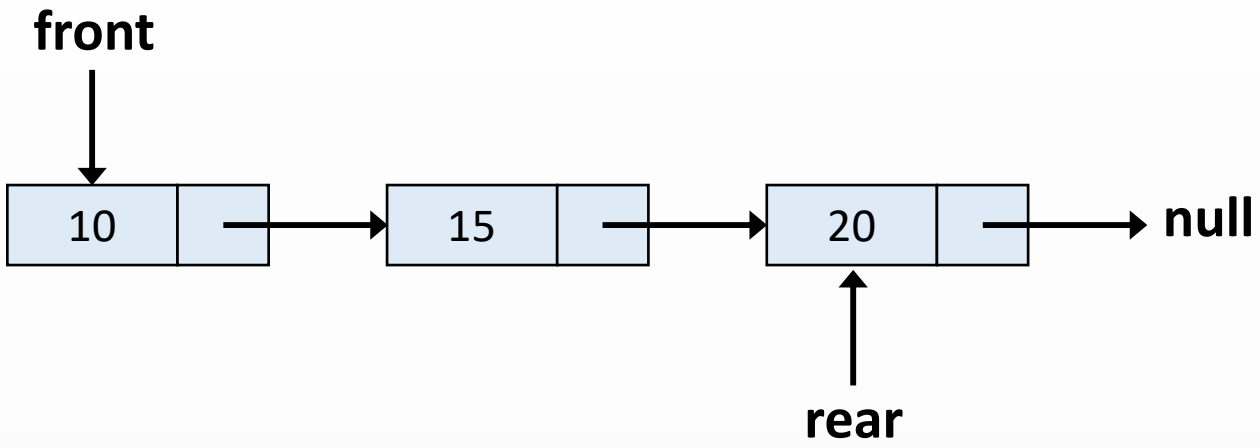
```
public int dequeue() {
  if(bosMu()) {
     throw new NoSuchElementException();
  }
  int sonuc = front.veri;
  front = front.sonraki;
  if(front == null) {
     rear = null;
  }
  uzunluk--;
  return sonuc;
}
```



uzunluk = 3

dequeue()

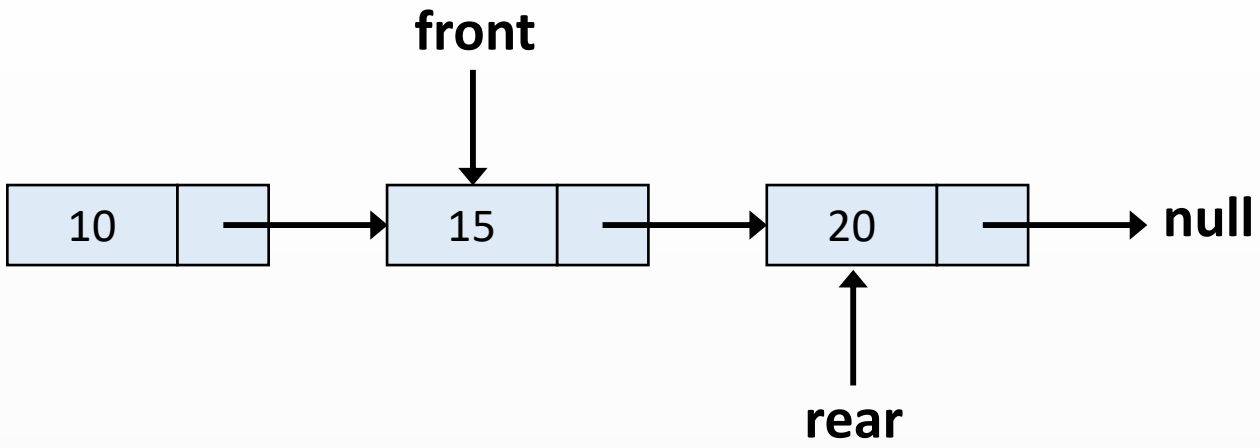
```
public int dequeue() {
  if(bosMu()) {
     throw new NoSuchElementException();
  }
  int sonuc = front.veri;
  front = front.sonraki;
  if(front == null) {
     rear = null;
  }
  uzunluk--;
  return sonuc;
}
```



uzunluk = 3
sonuc = 10

dequeue()

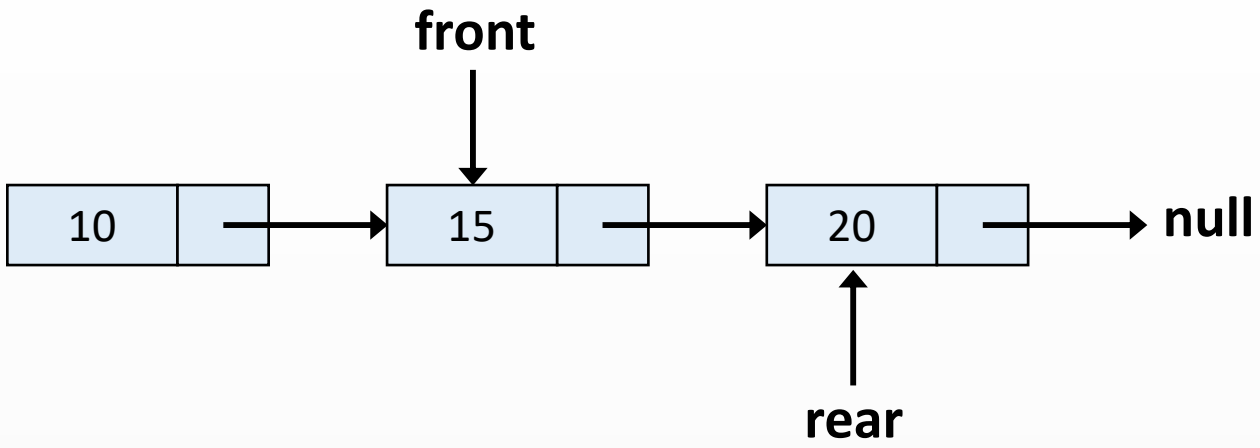
```
public int dequeue() {
  if(bosMu()) {
     throw new NoSuchElementException();
  }
  int sonuc = front.veri;
  front = front.sonraki;
  if(front == null) {
     rear = null;
  }
  uzunluk--;
  return sonuc;
}
```



uzunluk = 3
sonuc = 10

dequeue()

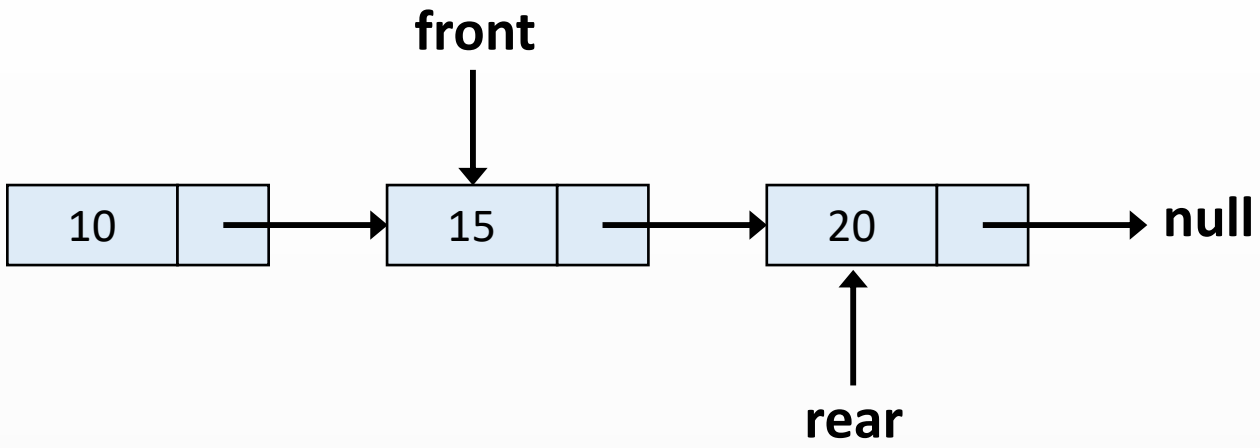
```
public int dequeue() {
  if(bosMu()) {
     throw new NoSuchElementException();
  }
  int sonuc = front.veri;
  front = front.sonraki;
  if(front == null) {
     rear = null;
  }
  uzunluk--;
  return sonuc;
}
```



uzunluk = 3
sonuc = 10

dequeue()

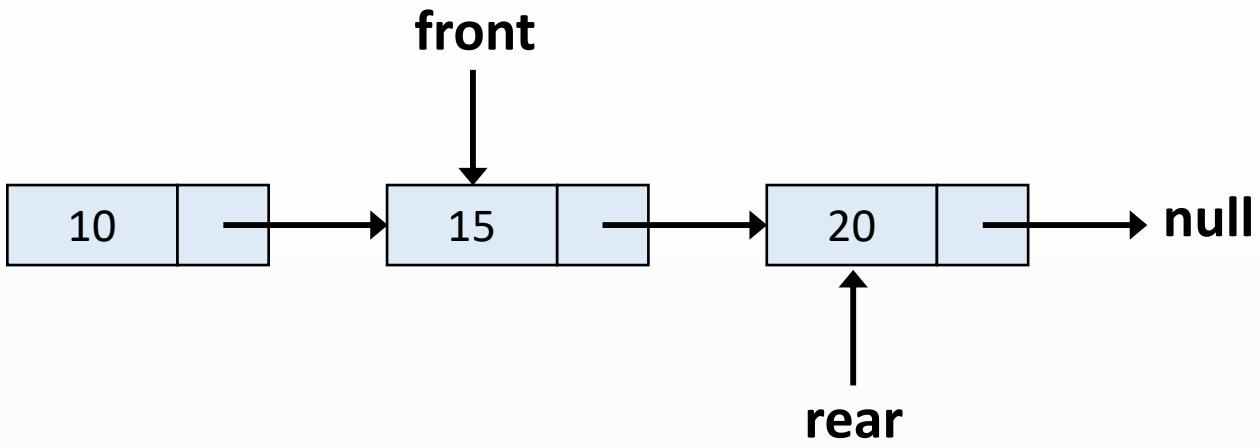
```
public int dequeue() {
  if(bosMu()) {
     throw new NoSuchElementException();
  }
  int sonuc = front.veri;
  front = front.sonraki;
  if(front == null) {
     rear = null;
  }
  uzunluk--;
  return sonuc;
}
```



uzunluk = 2
sonuc = 10

dequeue()

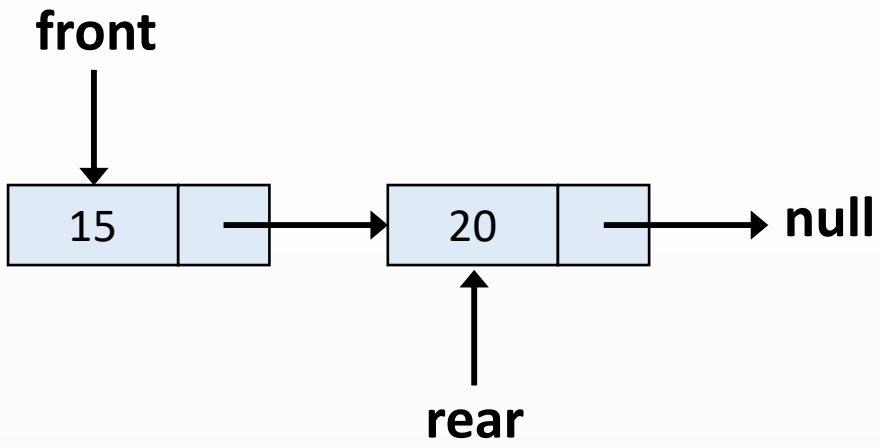
```
public int dequeue() {
  if(bosMu()) {
     throw new NoSuchElementException();
  }
  int sonuc = front.veri;
  front = front.sonraki;
  if(front == null) {
     rear = null;
  }
  uzunluk--;
  return sonuc;
}
```



uzunluk = 2
sonuc = 10

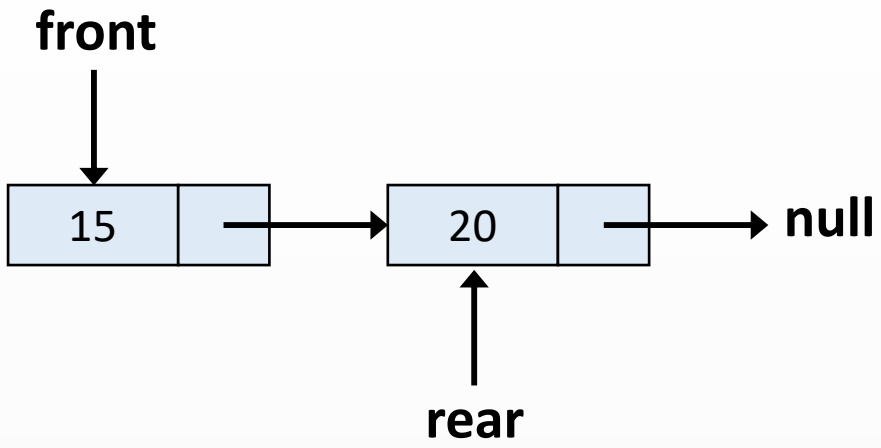
dequeue()

```
public int dequeue() {
  if(bosMu()) {
     throw new NoSuchElementException();
  }
  int sonuc = front.veri;
  front = front.sonraki;
  if(front == null) {
     rear = null;
  }
  uzunluk--;
  return sonuc;
}
```



uzunluk = 2

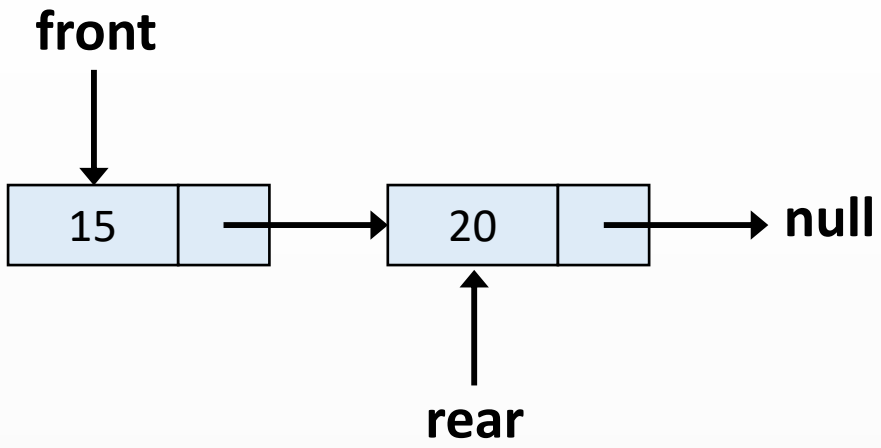
```
public int dequeue() {
  if(bosMu()) {
     throw new NoSuchElementException();
  }
  int sonuc = front.veri;
  front = front.sonraki;
  if(front == null) {
     rear = null;
  }
  uzunluk--;
  return sonuc;
}
```



uzunluk = 2

dequeue()

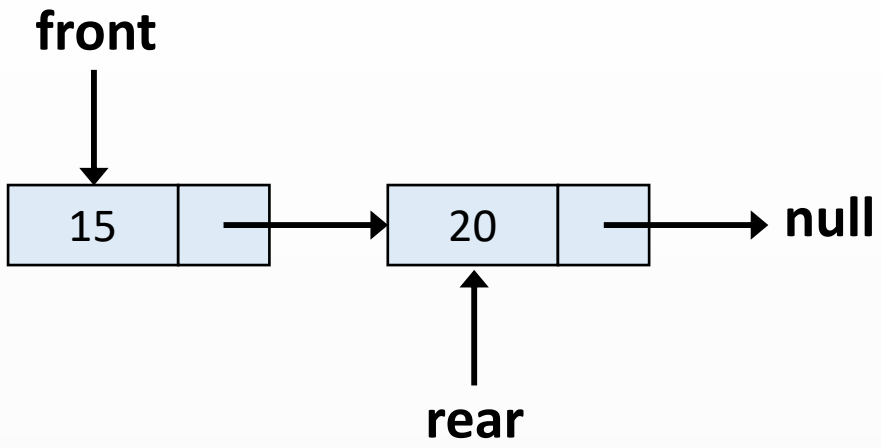
```
public int dequeue() {
  if(bosMu()) {
     throw new NoSuchElementException();
  }
  int sonuc = front.veri;
  front = front.sonraki;
  if(front == null) {
     rear = null;
  }
  uzunluk--;
  return sonuc;
}
```



uzunluk = 2

dequeue()

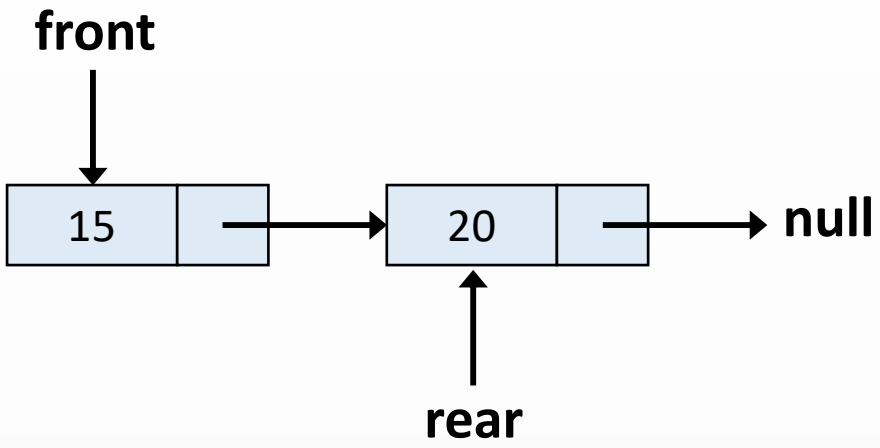
```
public int dequeue() {
  if(bosMu()) {
     throw new NoSuchElementException();
  }
  int sonuc = front.veri;
  front = front.sonraki;
  if(front == null) {
     rear = null;
  }
  uzunluk--;
  return sonuc;
}
```



uzunluk = 2

dequeue()

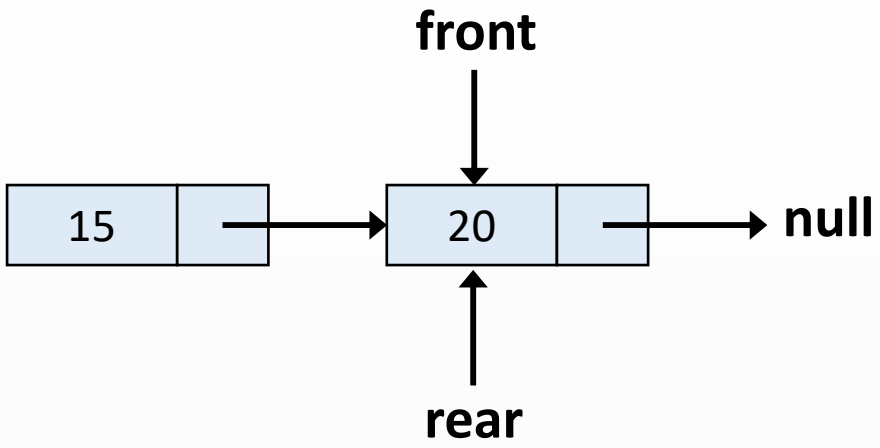
```
public int dequeue() {
  if(bosMu()) {
     throw new NoSuchElementException();
  }
  int sonuc = front.veri;
  front = front.sonraki;
  if(front == null) {
     rear = null;
  }
  uzunluk--;
  return sonuc;
}
```



uzunluk = 2
sonuc = 15

dequeue()

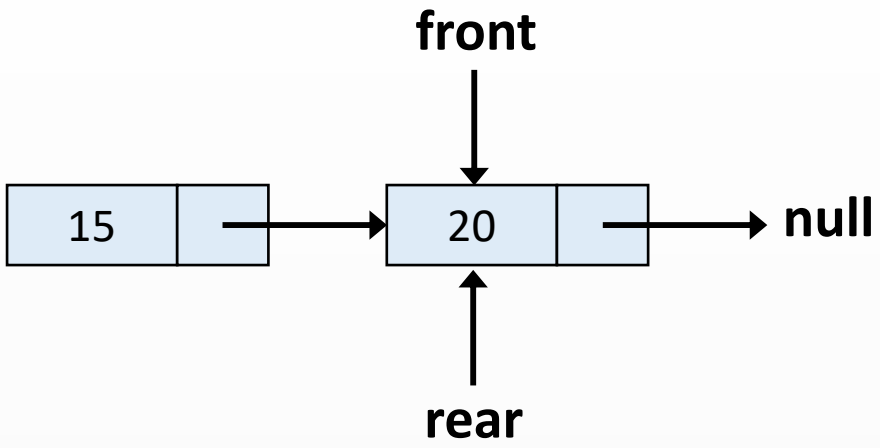
```
public int dequeue() {
  if(bosMu()) {
     throw new NoSuchElementException();
  }
  int sonuc = front.veri;
  front = front.sonraki;
  if(front == null) {
     rear = null;
  }
  uzunluk--;
  return sonuc;
}
```



uzunluk = 2
sonuc = 15

dequeue()

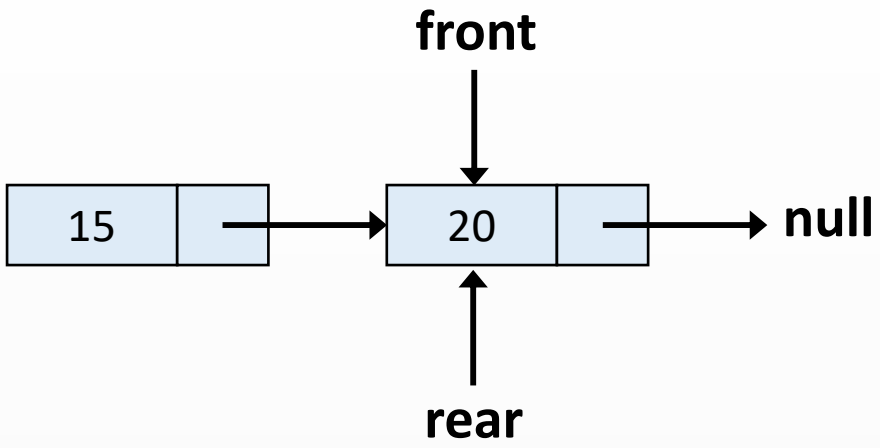
```
public int dequeue() {
  if(bosMu()) {
     throw new NoSuchElementException();
  }
  int sonuc = front.veri;
  front = front.sonraki;
  if(front == null) {
     rear = null;
  }
  uzunluk--;
  return sonuc;
}
```



uzunluk = 2
sonuc = 15

dequeue()

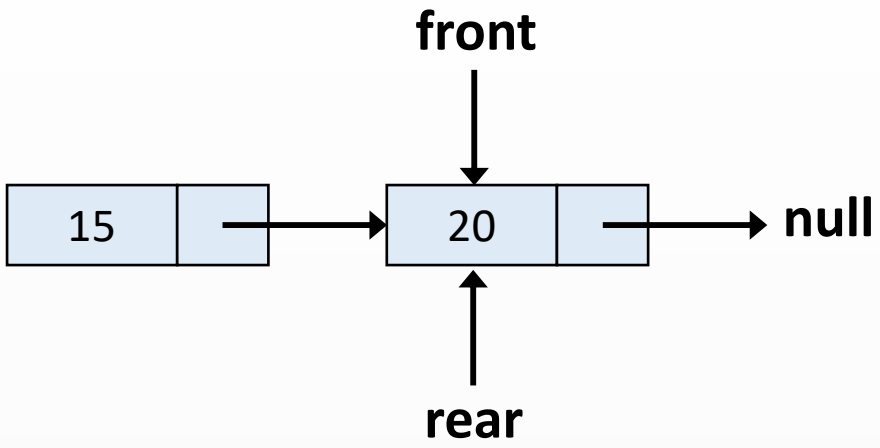
```
public int dequeue() {
  if(bosMu()) {
     throw new NoSuchElementException();
  }
  int sonuc = front.veri;
  front = front.sonraki;
  if(front == null) {
     rear = null;
  }
  uzunluk--;
  return sonuc;
}
```



uzunluk = 1
sonuc = 15

dequeue()

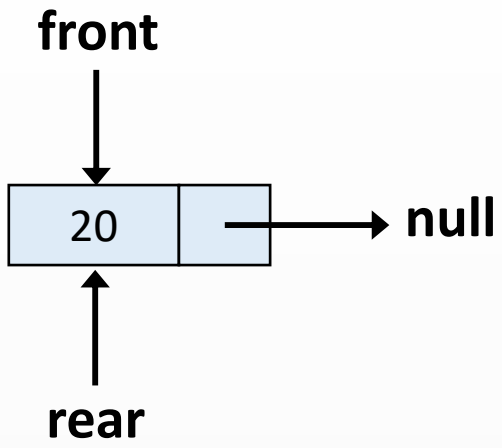
```
public int dequeue() {
  if(bosMu()) {
     throw new NoSuchElementException();
  }
  int sonuc = front.veri;
  front = front.sonraki;
  if(front == null) {
     rear = null;
  }
  uzunluk--;
  return sonuc;
}
```



uzunluk = 1
sonuc = 15

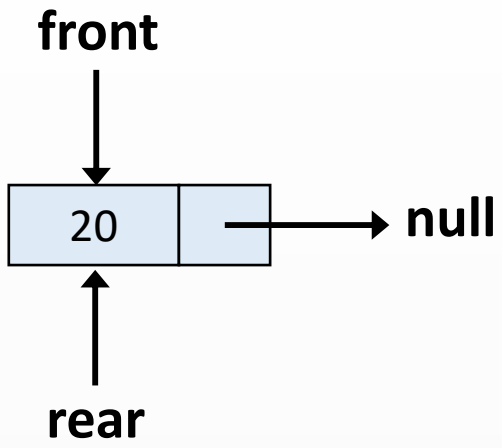
dequeue()

```
public int dequeue() {
  if(bosMu()) {
     throw new NoSuchElementException();
  }
  int sonuc = front.veri;
  front = front.sonraki;
  if(front == null) {
     rear = null;
  }
  uzunluk--;
  return sonuc;
}
```



uzunluk = 1

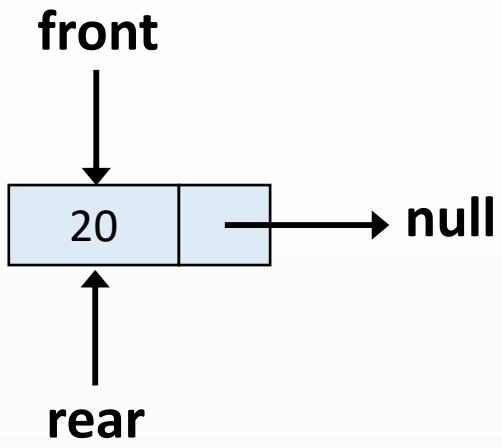
```
public int dequeue() {
  if(bosMu()) {
     throw new NoSuchElementException();
  }
  int sonuc = front.veri;
  front = front.sonraki;
  if(front == null) {
     rear = null;
  }
  uzunluk--;
  return sonuc;
}
```



uzunluk = 1

dequeue()

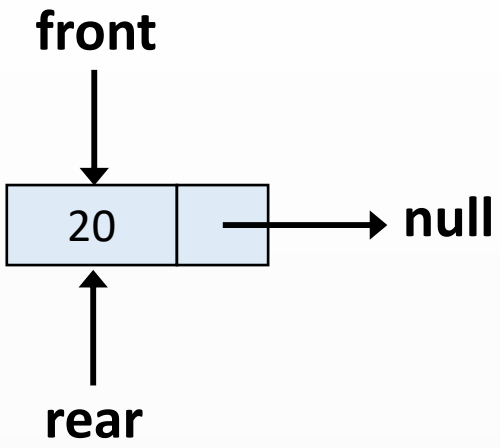
```
public int dequeue() {
  if(bosMu()) {
     throw new NoSuchElementException();
  }
  int sonuc = front.veri;
  front = front.sonraki;
  if(front == null) {
     rear = null;
  }
  uzunluk--;
  return sonuc;
}
```



```
public int dequeue() {
  if(bosMu()) {
     throw new NoSuchElementException();
  }
  int sonuc = front.veri;
  front = front.sonraki;
  if(front == null) {
     rear = null;
  }
  uzunluk--;
  return sonuc;
}
```

uzunluk = 1

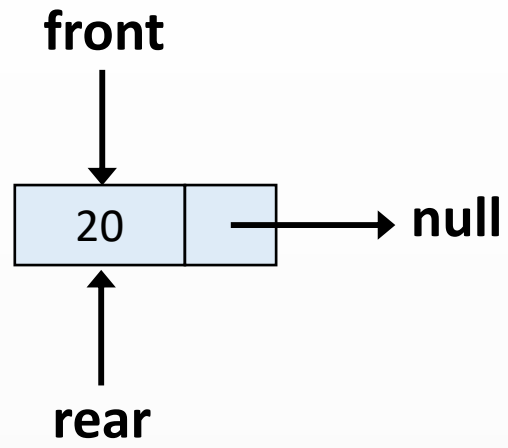
dequeue()



```
public int dequeue() {
  if(bosMu()) {
     throw new NoSuchElementException();
  }
  int sonuc = front.veri;
  front = front.sonraki;
  if(front == null) {
     rear = null;
  }
  uzunluk--;
  return sonuc;
}
```

uzunluk = 1

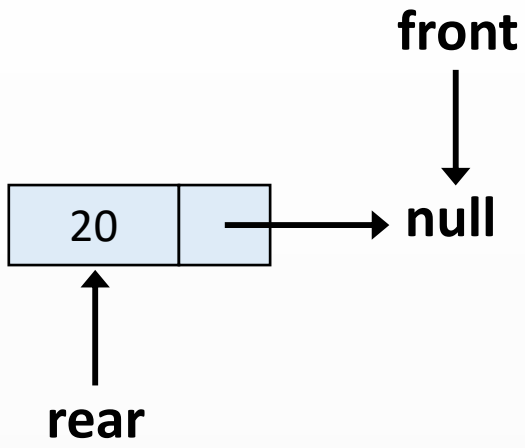
dequeue()



uzunluk = 1
sonuc = 20

dequeue()

```
public int dequeue() {
  if(bosMu()) {
     throw new NoSuchElementException();
  }
  int sonuc = front.veri;
  front = front.sonraki;
  if(front == null) {
     rear = null;
  }
  uzunluk--;
  return sonuc;
}
```

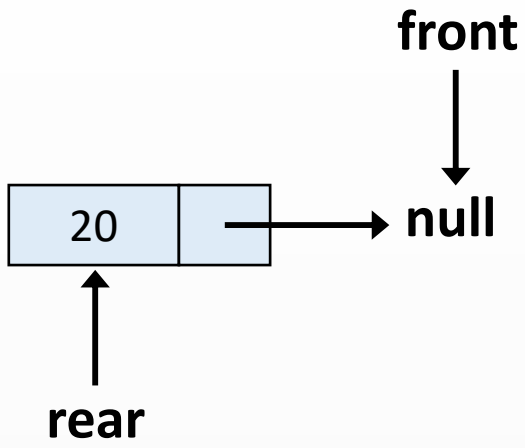


uzunluk = 1

sonuc = 20

dequeue()

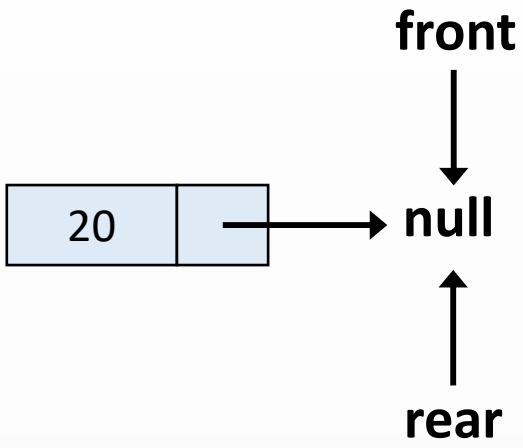
```
public int dequeue() {
  if(bosMu()) {
     throw new NoSuchElementException();
  }
  int sonuc = front.veri;
  front = front.sonraki;
  if(front == null) {
     rear = null;
  }
  uzunluk--;
  return sonuc;
}
```



uzunluk = 1
sonuc = 20

dequeue()

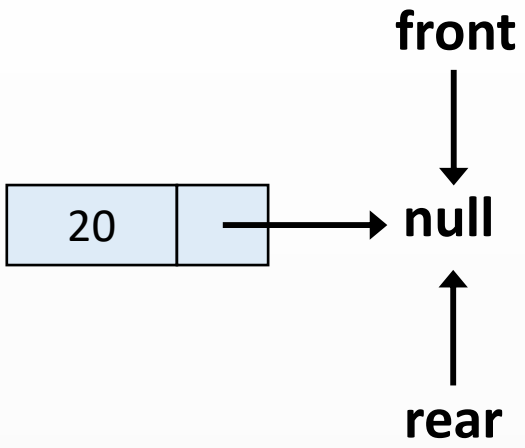
```
public int dequeue() {
  if(bosMu()) {
     throw new NoSuchElementException();
  }
  int sonuc = front.veri;
  front = front.sonraki;
  if(front == null) {
     rear = null;
  }
  uzunluk--;
  return sonuc;
}
```



uzunluk = 1
sonuc = 20

dequeue()

```
public int dequeue() {
  if(bosMu()) {
     throw new NoSuchElementException();
  }
  int sonuc = front.veri;
  front = front.sonraki;
  if(front == null) {
     rear = null;
  }
  uzunluk--;
  return sonuc;
}
```

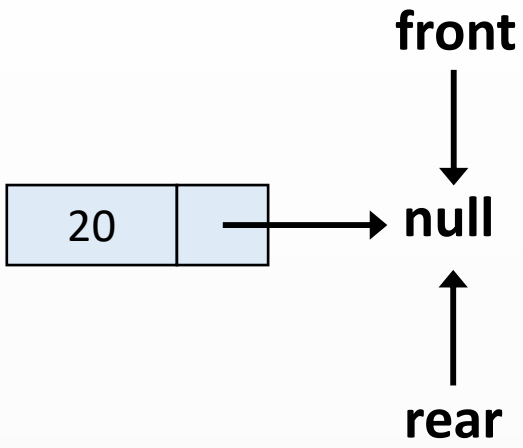


uzunluk = 0

sonuc = 20

dequeue()

```
public int dequeue() {
  if(bosMu()) {
     throw new NoSuchElementException();
  }
  int sonuc = front.veri;
  front = front.sonraki;
  if(front == null) {
     rear = null;
  }
  uzunluk--;
  return sonuc;
}
```

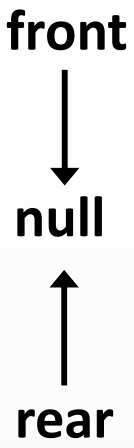


uzunluk = 0

sonuc = 20

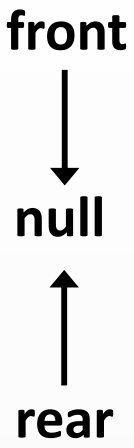
dequeue()

```
public int dequeue() {
  if(bosMu()) {
     throw new NoSuchElementException();
  }
  int sonuc = front.veri;
  front = front.sonraki;
  if(front == null) {
     rear = null;
  }
  uzunluk--;
  return sonuc;
}
```



uzunluk = 0

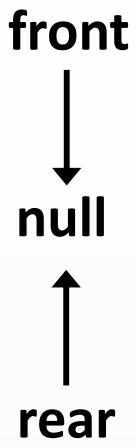
```
public int dequeue() {
  if(bosMu()) {
     throw new NoSuchElementException();
  }
  int sonuc = front.veri;
  front = front.sonraki;
  if(front == null) {
     rear = null;
  }
  uzunluk--;
  return sonuc;
}
```



uzunluk = 0

dequeue()

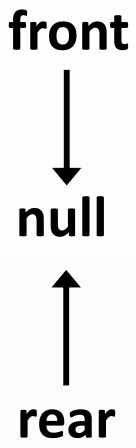
```
public int dequeue() {
  if(bosMu()) {
     throw new NoSuchElementException();
  }
  int sonuc = front.veri;
  front = front.sonraki;
  if(front == null) {
     rear = null;
  }
  uzunluk--;
  return sonuc;
}
```



uzunluk = 0

dequeue()

```
public int dequeue() {
  if(bosMu()) {
     throw new NoSuchElementException();
  }
  int sonuc = front.veri;
  front = front.sonraki;
  if(front == null) {
     rear = null;
  }
  uzunluk--;
  return sonuc;
}
```

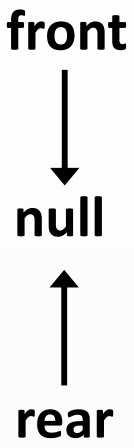


uzunluk = 0

dequeue()

```
public int dequeue() {
  if(bosMu()) {
     throw new NoSuchElementException();
  }
  int sonuc = front.veri;
  front = front.sonraki;
  if(front == null) {
     rear = null;
  }
  uzunluk--;
  return sonuc;
}
```

Böyle Bir Eleman Yok İstisnası

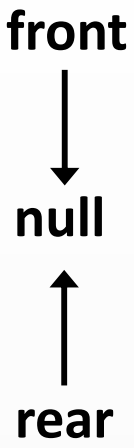


uzunluk = 0

dequeue()

```
public int dequeue() {
  if(bosMu()) {
     throw new NoSuchElementException();
  }
  int sonuc = front.veri;
  front = front.sonraki;
  if(front == null) {
     rear = null;
  }
  uzunluk--;
  return sonuc;
}
```

Böyle Bir Eleman Yok İstisnası

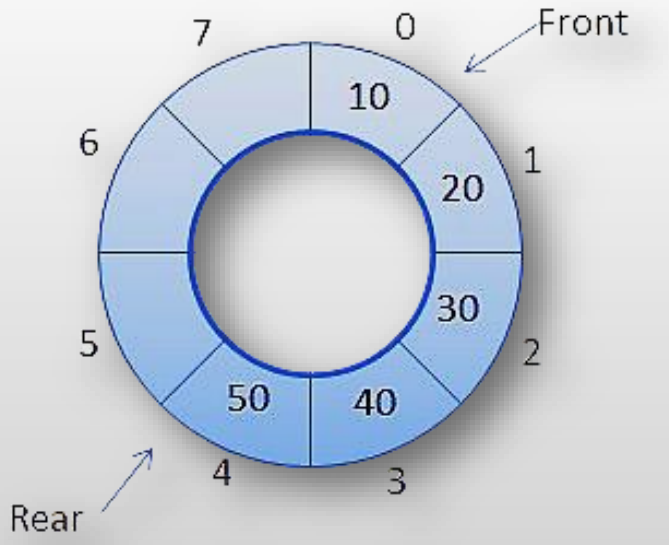


uzunluk = 0

```
public int dequeue() {
  if(bosMu()) {
     throw new NoSuchElementException();
  }
  int sonuc = front.veri;
  front = front.sonraki;
  if(front == null) {
     rear = null;
  }
  uzunluk--;
  return sonuc;
}
```

Dairesel Kuyruk (Circular Queue)

- Normal kuyruk gibi çalışır, son öğe null yerine ilk öğeyi işaret eder.
- Baştaki ve sondaki öğe bir döngü oluşturur.
- Dizi tabanlı temsilde, öğelerin döngüsel hareketi için modulo kullanılır.
- Modulo işlemi, kuyruğun kapasitesi aşılmadan öğelerin eklenmesini ve çıkarılmasını sağlar.
- Ring buffer olarak da bilinir.



Dairesel Kuyruk (Circular Queue)

```
public class DaireselKuyruk<E> {
  E[] dizi;      // Öğeleri saklar
  int bas;       // Kuyruğun ilk öğesini gösterir
  int son;       // Kuyruğun son öğesini gösterir
  int boyut;     // Kuyruktaki öğe sayısı
  int kapasite; // Kuyruğun kapasitesi

  public DaireselKuyruk(int kapasite) {
    this.kapasite = kapasite;
    dizi = (E[]) new Object[kapasite];
    bas = son = boyut = 0;  // Kuyruk boş
  }
}
```

Dairesel Kuyruk (Circular Queue)

```
public void ekle(E oge) {

  if (boyut == kapasite) {
    throw new IllegalStateException("Kuyruk dolu.");
  }

  dizi[son] = oge; // Öğeyi kuyruğa ekle
  boyut++; // Kuyruktaki öğe sayısını artır
  son = (son + 1) % kapasite; // Modulo işlemi
}
```

Dairesel Kuyruk (Circular Queue)

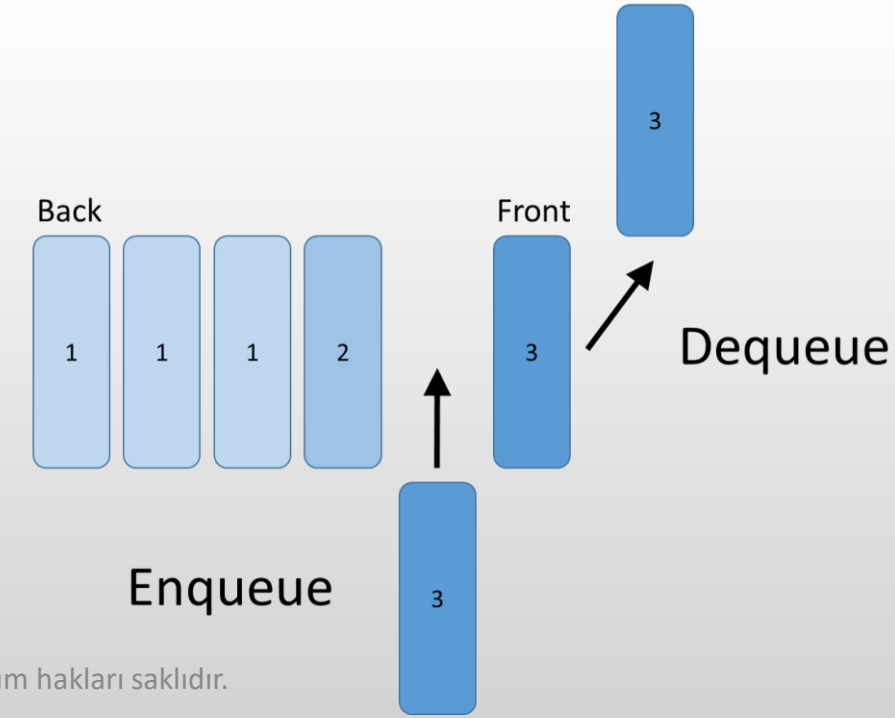
```
public E cikar() {

  if (bosMu()) {
    throw new IllegalStateException("Kuyruk boş.");
  }

  E oge = (E) dizi[bas]; // Baştaki öğeyi al
  bas = (bas + 1) % kapasite; // Modulo işlemi
  boyut--; // Kuyruktaki öğe sayısını azalt
  return oge; // Çıkarılan öğeyi döndür
}
```

Öncelikli Kuyruk (Priority Queue)

- Öğeleri öncelik düzenine göre sıralar.
- Öncelik, yüksek değere sahip öğenin önce işlenmesini sağlar.
- Zaman duyarlı sistemler, iş parçacığı yönetimi, veri sıkıştırma algoritmaları gibi alanlarda kullanılır.



Öncelikli Kuyruk (Priority Queue)

```
public class Eleman<T> {

  T veri;
  int oncelik;

  Eleman(T veri, int oncelik) {
    this.veri = veri;
    this.oncelik = oncelik;
  }
}
```

Öncelikli Kuyruk (Priority Queue)

```
public class OncelikliKuyruk<E> {

  private Object[] dizi;
  private int boyut;
  int BASLANGIC_KAPASITESI = 10;

  public OncelikliKuyruk() {
    dizi = new Object[BASLANGIC_KAPASITESI];
    boyut = 0;
  }
}
```

Öncelikli Kuyruk (Priority Queue)

```
public void ekle(E veri, int öncelik) {
  if (boyut == dizi.length) {
    diziGenislet();
  }
  Eleman<E> yeniEleman = new Eleman<E>(veri, öncelik);
  int indeks = boyut;
  // Öncelik değerlerine göre doğru konumu bul
  while (indeks > 0 && öncelik < dizi[indeks - 1].oncelik) {
    dizi[indeks] = dizi[indeks - 1];
    indeks--;
  }
  dizi[indeks] = yeniEleman;
  boyut++;
}
```

Öncelikli Kuyruk (Priority Queue)

```
public E cikar() {
  if (bosMu()) {
    throw new IllegalStateException("Öncelikli kuyruk boş");
  }
  E cikarilan = (E) ((Eleman) dizi[0]).veri;
  // Elemanlar bir önceki elemanın yerine kaydırılır.
  for (int i = 0; i < boyut - 1; i++) {
    dizi[i] = dizi[i + 1];
  }
  dizi[boyut - 1] = null;
  boyut--;
  return cikarilan;
}
```

Deque (Double Ended Queue)

- Çift uçlu kuyruk olarak da bilinir.
- Hem ön hem de arka tarafından öğe eklenebilir veya çıkarılabilir.
- FIFO kuralını ihlal edebilir.

Deque (Double Ended Queue)

```
void basaEkle(E veri) {

  CiftYonluDugum<E> yeniDugum = new CiftYonluDugum<E>(veri);

  if (bosMu()) {
    bas = yeniDugum;
    son = yeniDugum;
  } else {
    yeniDugum.sonraki = bas;
    bas.onceki = yeniDugum;
    bas = yeniDugum;
  }
}
```

Deque (Double Ended Queue)

```
public void sonaEkle(E veri) {

  CiftYonluDugum<E> yeniDugum = new CiftYonluDugum<E>(veri);

  if (bosMu()) {
    bas = yeniDugum;
    son = yeniDugum;
  } else {
    yeniDugum.onceki = son;
    son.sonraki = yeniDugum;
    son = yeniDugum;
  }
}
```

Deque (Double Ended Queue)

```
public E bastanCikar() {

  if (bosMu()) {
    throw new IllegalStateException("Dequeue boş");
  }

  E veri = bas.veri;
  bas = bas.sonraki;
  if (bas != null) {
    bas.onceki = null;
  }
  return veri;
}
```

Deque (Double Ended Queue)

```
public E sondanCikar() {

  if (bosMu()) {
    throw new IllegalStateException("Dequeue boş");
  }

  E veri = son.veri;
  son = son.onceki;
  if (son != null) {
    son.sonraki = null;
  }
  return veri;
}
```

Örnek

1'den n'e Kadar İkilik Sayıları Üretme

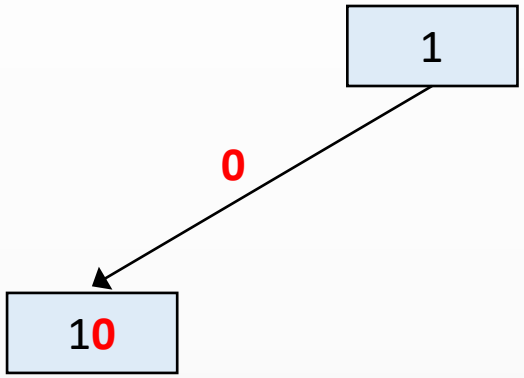
- Kuyruk veri yapısı kullanılarak 1'den n'ye kadar ikilik sayılar nasıl üretilir?
- **Örnek 1:**
  - **Girdi**: n = 3
  - **Çıktı**: sonuc = {"1","10","11"}
- **Örnek 2:**
  - **Girdi**: n = 5
  - **Çıktı**: sonuc = {"1","10","11","100","101"}

1'den n'e Kadar İkilik Sayıları Üretme

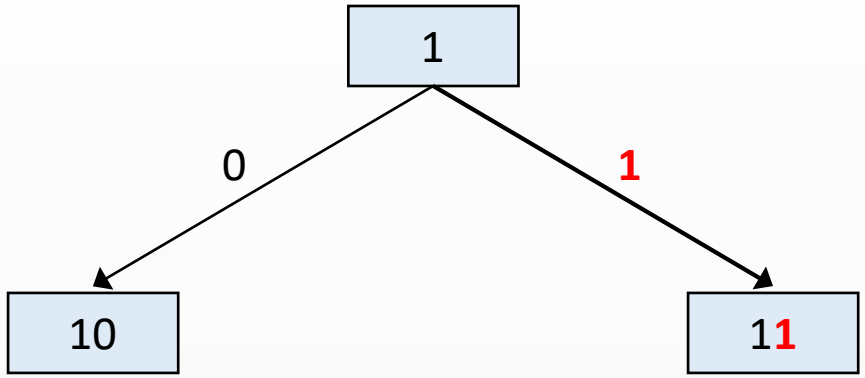
1

İkilik	Onluk
1	1

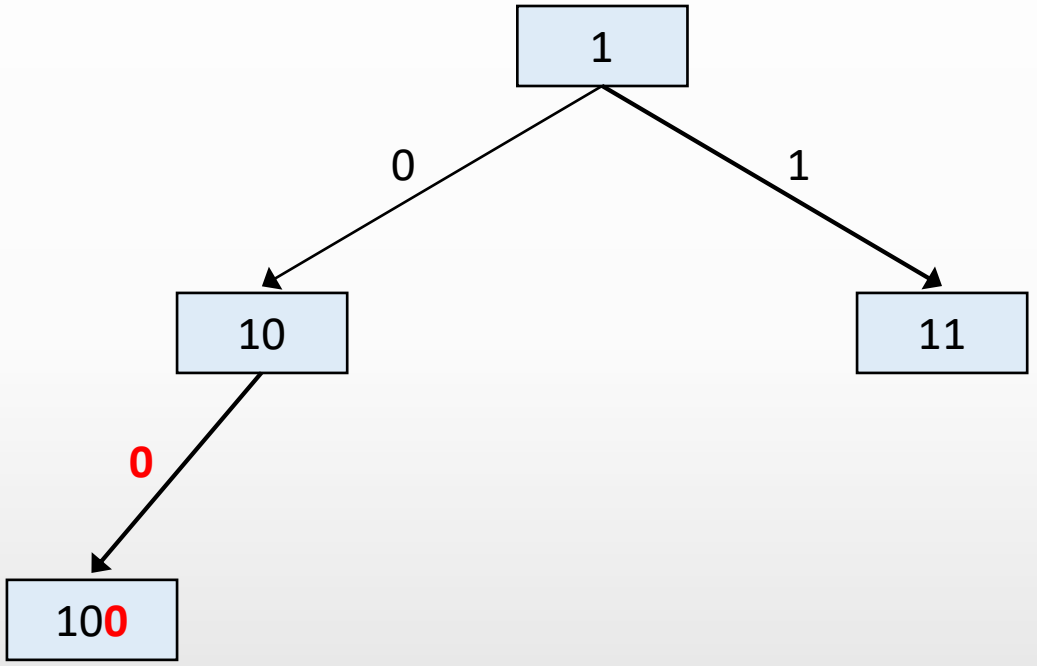
1'den n'e Kadar İkilik Sayıları Üretme



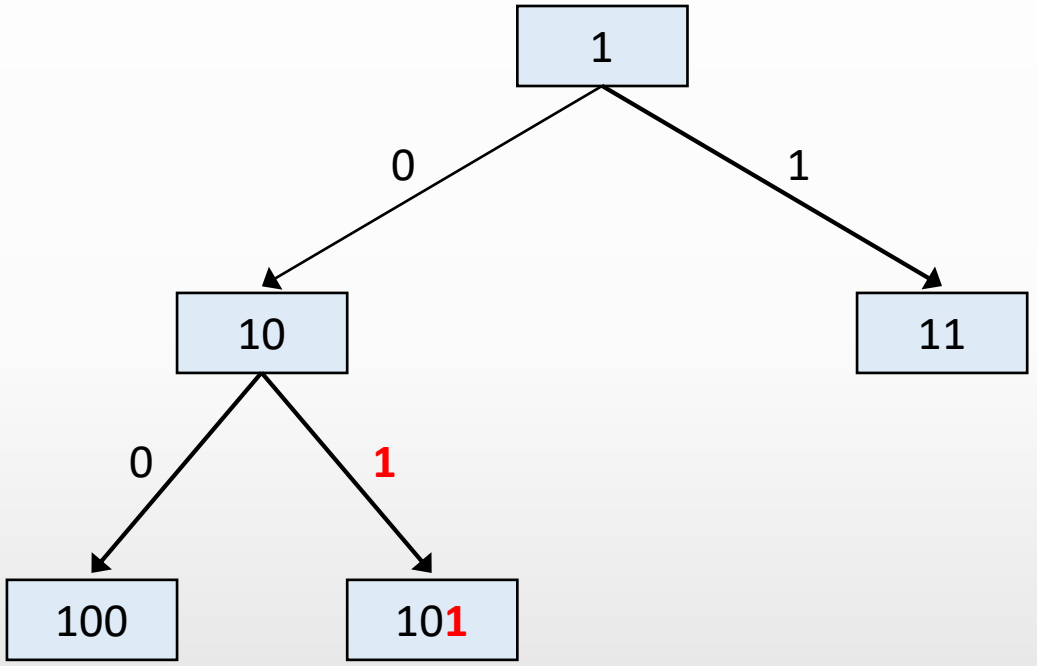
İkilik	Onluk
1	1
10	2



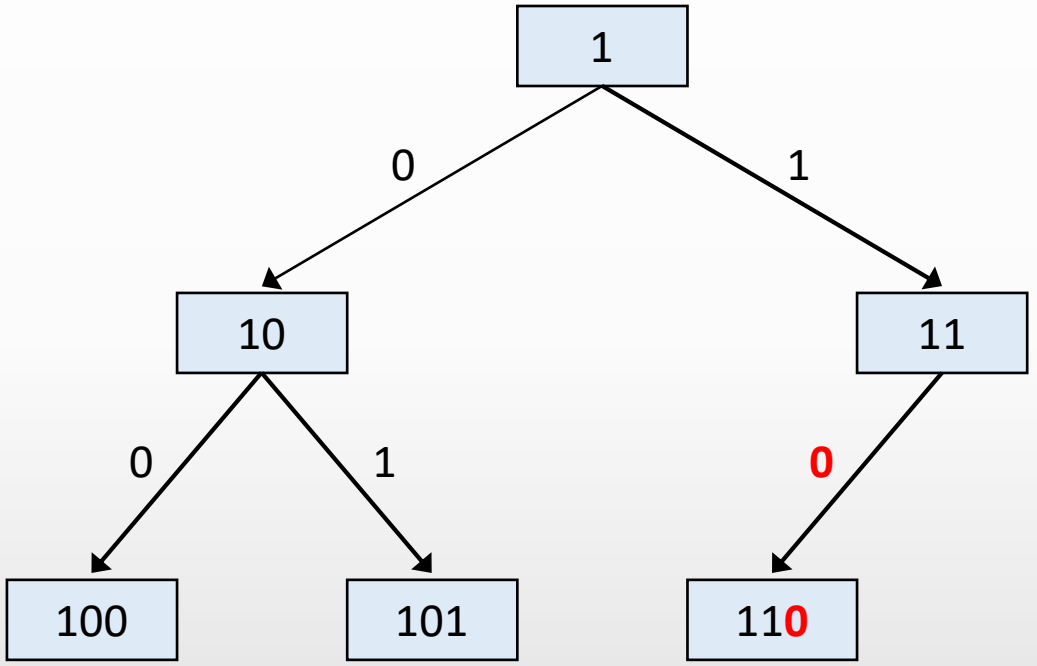
İkilik	Onluk
1	1
10	2
11	3



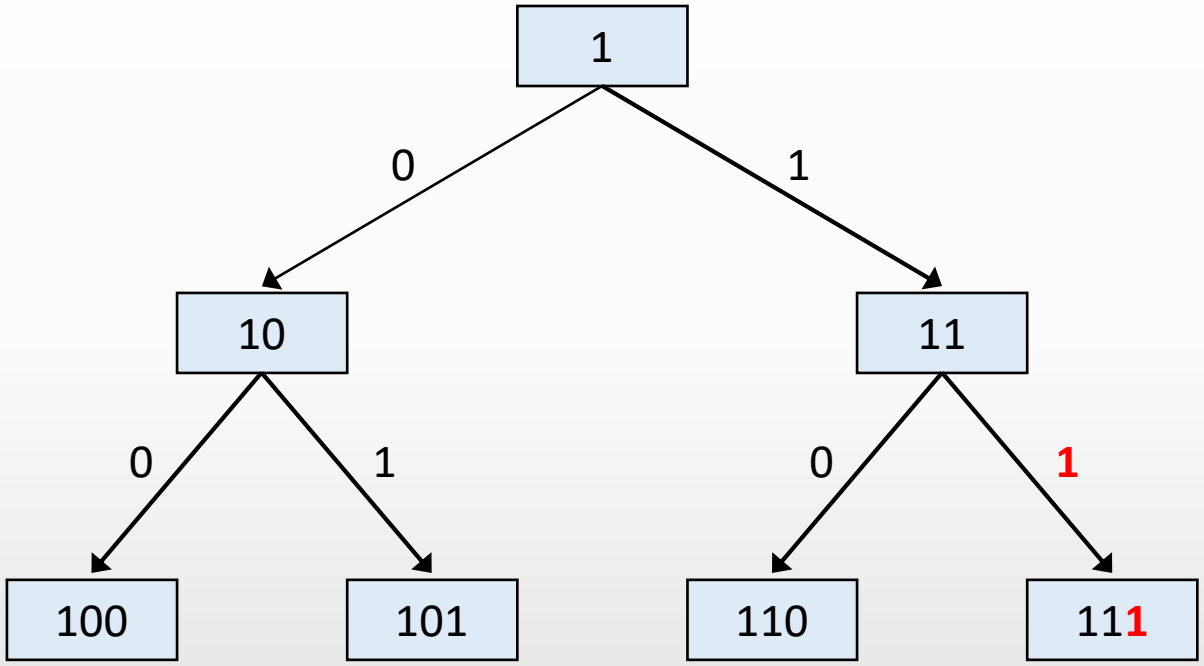
İkilik	Onluk
1	1
10	2
11	3
100	4



İkilik	Onluk
1	1
10	2
11	3
100	4
101	5



İkilik	Onluk
1	1
10	2
11	3
100	4
101	5
110	6



İkilik	Onluk
1	1
10	2
11	3
100	4
101	5
110	6
111	7

```
String[] ikilikSayiUret(int n) {
  String[] sonuc = new String[n];
  Queue<String> q = new LinkedList<>();
  q.offer("1");
  for(int i = 0; i < n; i++) {
     sonuc[i] = q.poll();
     String n1 = sonuc[i] + "0";
     String n2 = sonuc[i] + "1";
     q.offer(n1);
     q.offer(n2);
  }
  return sonuc;
}
```

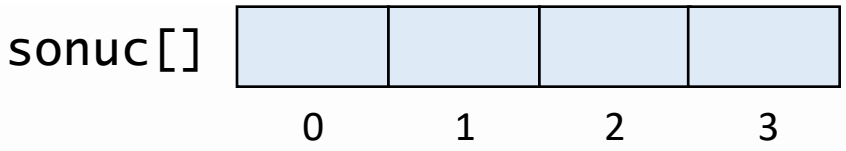
```
ikilikSayiUret(4);
```

```
String[] ikilikSayiUret(int n) {
  String[] sonuc = new String[n];
  Queue<String> q = new LinkedList<>();
  q.offer("1");
  for(int i = 0; i < n; i++) {
     sonuc[i] = q.poll();
     String n1 = sonuc[i] + "0";
     String n2 = sonuc[i] + "1";
     q.offer(n1);
     q.offer(n2);
  }
  return sonuc;
}
```

n = 4

ikilikSayiUret(4);

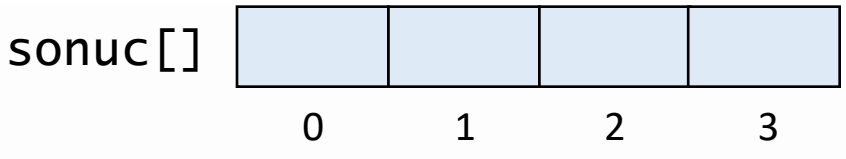
```
String[] ikilikSayiUret(int n) {
  String[] sonuc = new String[n];
  Queue<String> q = new LinkedList<>();
  q.offer("1");
  for(int i = 0; i < n; i++) {
     sonuc[i] = q.poll();
     String n1 = sonuc[i] + "0";
     String n2 = sonuc[i] + "1";
     q.offer(n1);
     q.offer(n2);
  }
  return sonuc;
}
```

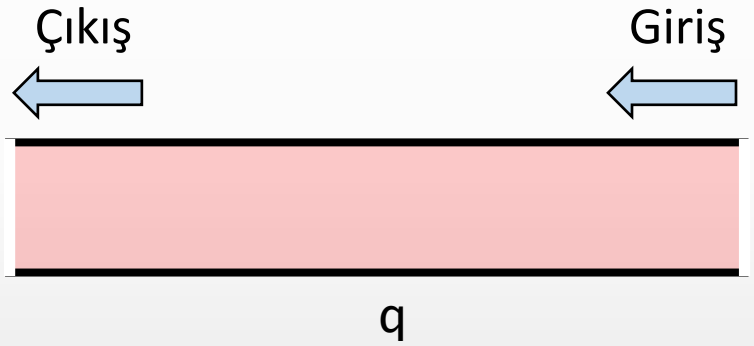


n = 4

ikilikSayiUret(4);

```
String[] ikilikSayiUret(int n) {
  String[] sonuc = new String[n];
  Queue<String> q = new LinkedList<>();
  q.offer("1");
  for(int i = 0; i < n; i++) {
     sonuc[i] = q.poll();
     String n1 = sonuc[i] + "0";
     String n2 = sonuc[i] + "1";
     q.offer(n1);
     q.offer(n2);
  }
  return sonuc;
}
```

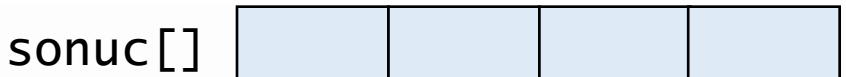


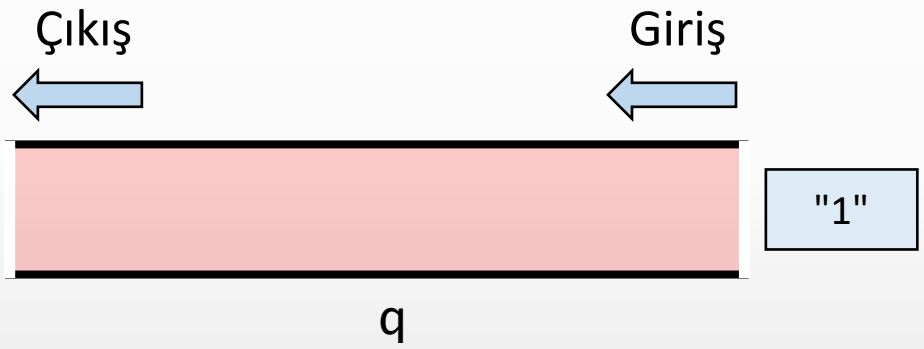


n = 4

ikilikSayiUret(4);


```
String[] ikilikSayiUret(int n) {
  String[] sonuc = new String[n];
  Queue<String> q = new LinkedList<>();
  q.offer("1");
  for(int i = 0; i < n; i++) {
     sonuc[i] = q.poll();
     String n1 = sonuc[i] + "0";
     String n2 = sonuc[i] + "1";
     q.offer(n1);
     q.offer(n2);
  }
  return sonuc;
}
```

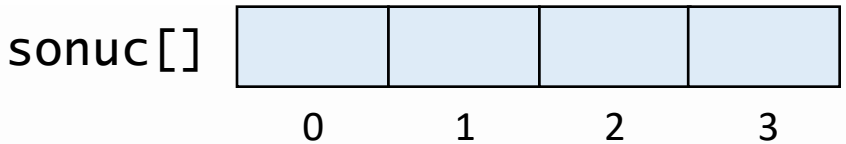


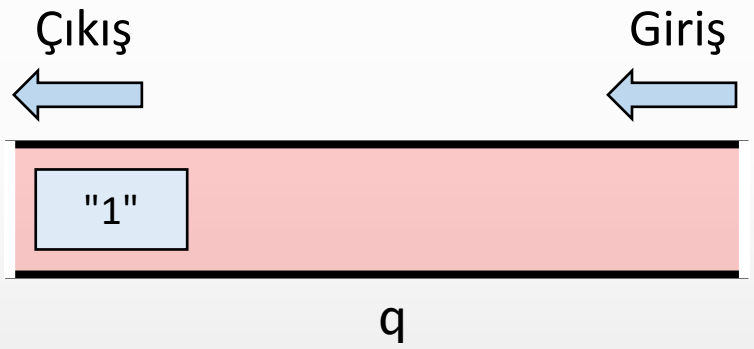


n = 4

ikilikSayiUret(4);


```
String[] ikilikSayiUret(int n) {
  String[] sonuc = new String[n];
  Queue<String> q = new LinkedList<>();
  q.offer("1");
  for(int i = 0; i < n; i++) {
     sonuc[i] = q.poll();
     String n1 = sonuc[i] + "0";
     String n2 = sonuc[i] + "1";
     q.offer(n1);
     q.offer(n2);
  }
  return sonuc;
}
```

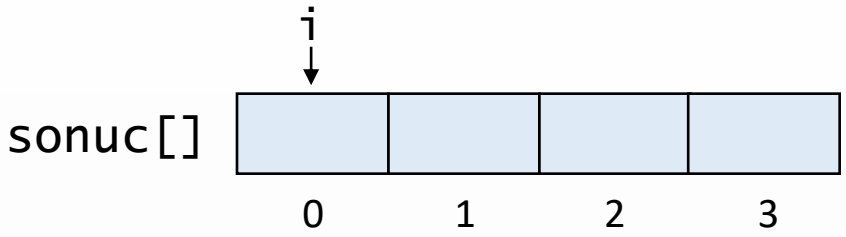


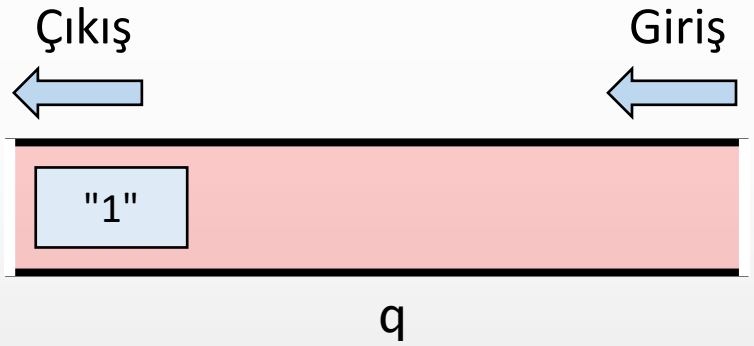


n = 4

ikilikSayiUret(4);


```
String[] ikilikSayiUret(int n) {
  String[] sonuc = new String[n];
  Queue<String> q = new LinkedList<>();
  q.offer("1");
  for(int i = 0; i < n; i++) {
     sonuc[i] = q.poll();
     String n1 = sonuc[i] + "0";
     String n2 = sonuc[i] + "1";
     q.offer(n1);
     q.offer(n2);
  }
  return sonuc;
}
```



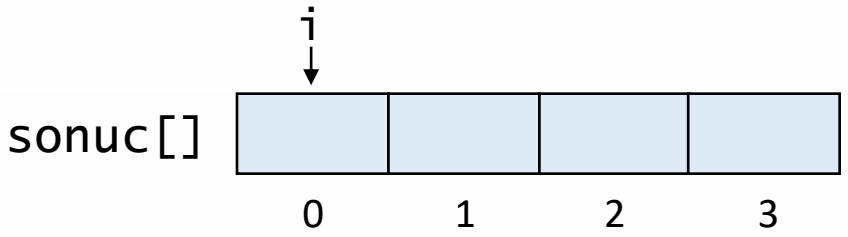


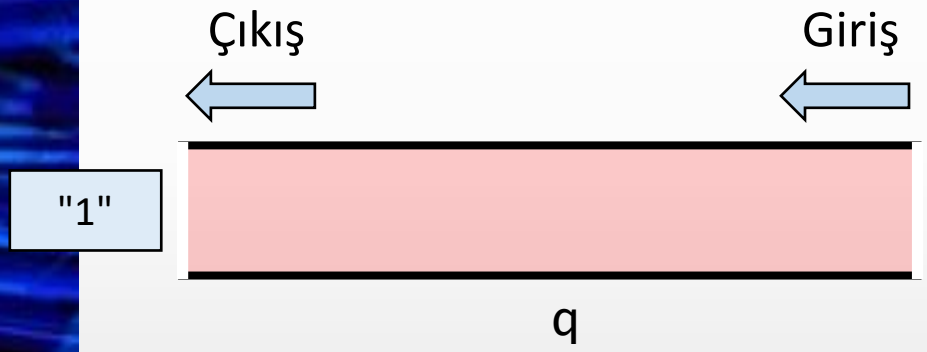
i = 0

n = 4

ikilikSayiUret(4);

```
String[] ikilikSayiUret(int n) {
  String[] sonuc = new String[n];
  Queue<String> q = new LinkedList<>();
  q.offer("1");
  for(int i = 0; i < n; i++) {
     sonuc[i] = q.poll();
     String n1 = sonuc[i] + "0";
     String n2 = sonuc[i] + "1";
     q.offer(n1);
     q.offer(n2);
  }
  return sonuc;
}
```



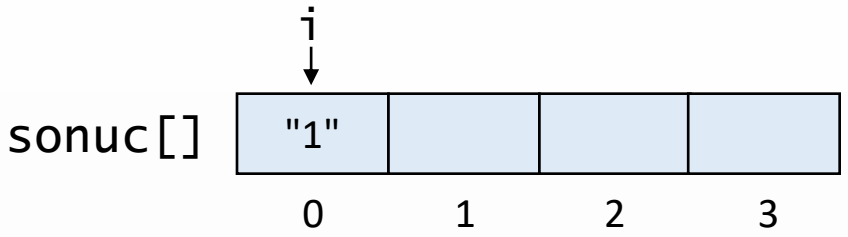


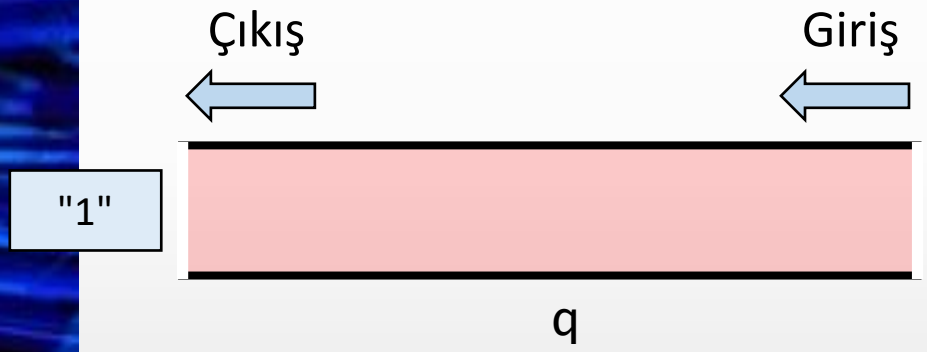
i = 0

n = 4

ikilikSayiUret(4);

```
String[] ikilikSayiUret(int n) {
  String[] sonuc = new String[n];
  Queue<String> q = new LinkedList<>();
  q.offer("1");
  for(int i = 0; i < n; i++) {
     sonuc[i] = q.poll();
     String n1 = sonuc[i] + "0";
     String n2 = sonuc[i] + "1";
     q.offer(n1);
     q.offer(n2);
  }
  return sonuc;
}
```



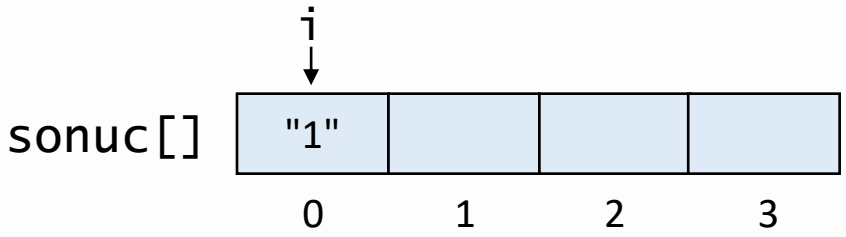


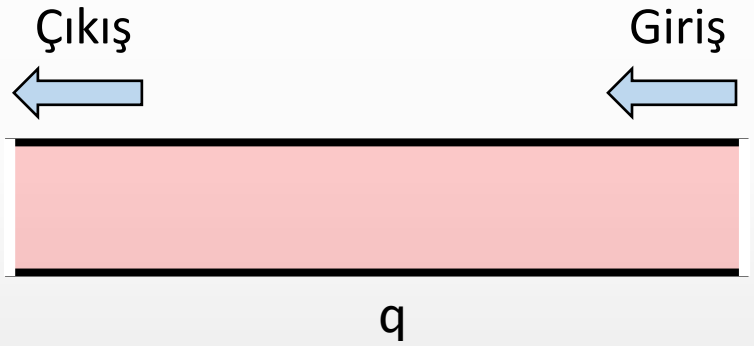
i = 0

n = 4

ikilikSayiUret(4);

```
String[] ikilikSayiUret(int n) {
  String[] sonuc = new String[n];
  Queue<String> q = new LinkedList<>();
  q.offer("1");
  for(int i = 0; i < n; i++) {
     sonuc[i] = q.poll();
     String n1 = sonuc[i] + "0";
     String n2 = sonuc[i] + "1";
     q.offer(n1);
     q.offer(n2);
  }
  return sonuc;
}
```



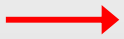


n1 = "10"

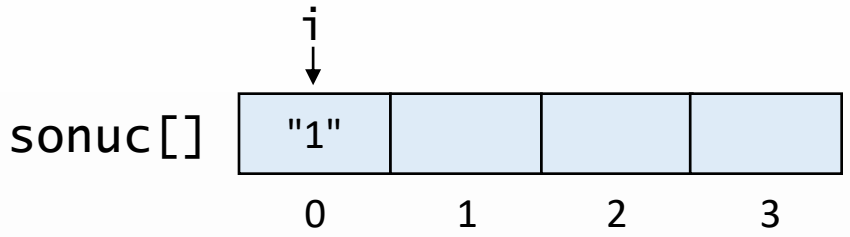
i = 0

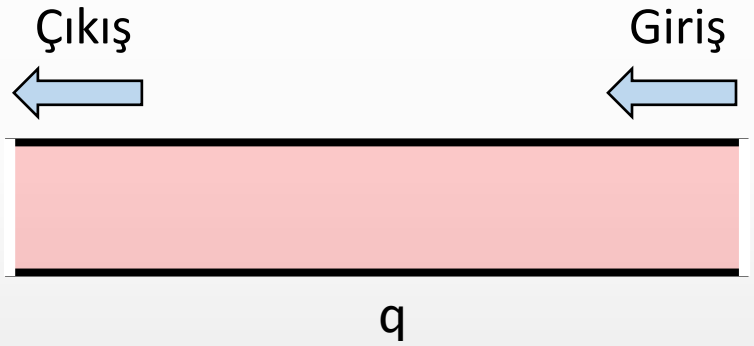
n = 4

ikilikSayiUret(4);



```
String[] ikilikSayiUret(int n) {
  String[] sonuc = new String[n];
  Queue<String> q = new LinkedList<>();
  q.offer("1");
  for(int i = 0; i < n; i++) {
     sonuc[i] = q.poll();
     String n1 = sonuc[i] + "0";
     String n2 = sonuc[i] + "1";
     q.offer(n1);
     q.offer(n2);
  }
  return sonuc;
}
```





n2 = "11"

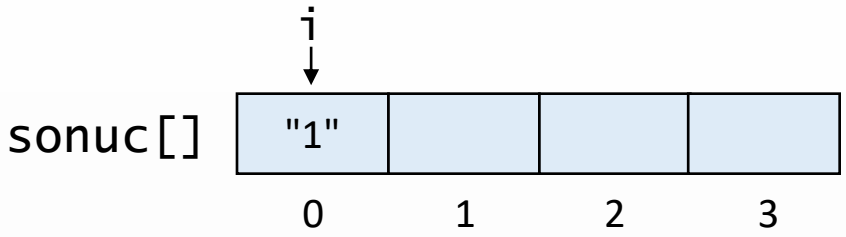
n1 = "10"

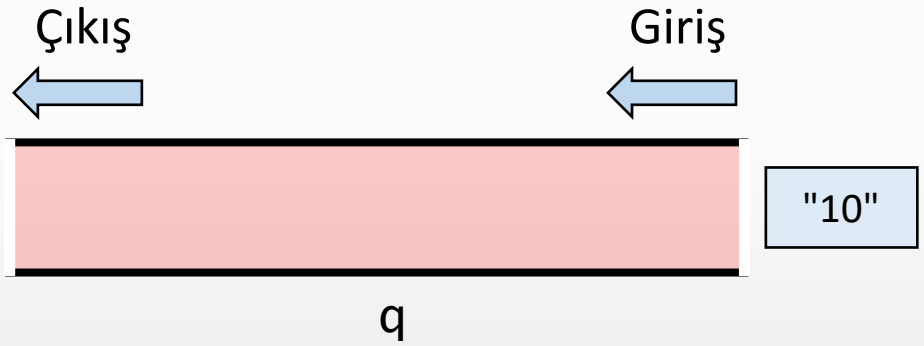
i = 0

n = 4

ikilikSayiUret(4);

```
String[] ikilikSayiUret(int n) {
  String[] sonuc = new String[n];
  Queue<String> q = new LinkedList<>();
  q.offer("1");
  for(int i = 0; i < n; i++) {
     sonuc[i] = q.poll();
     String n1 = sonuc[i] + "0";
     String n2 = sonuc[i] + "1";
     q.offer(n1);
     q.offer(n2);
  }
  return sonuc;
}
```





n2 = "11"

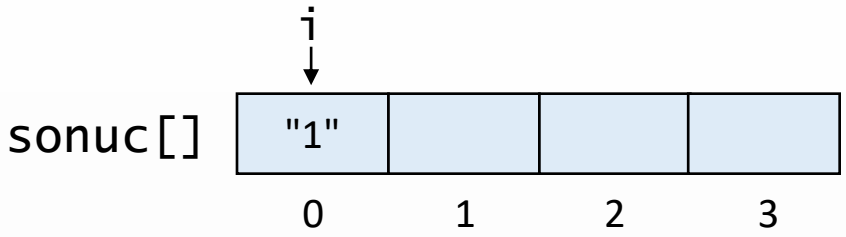
n1 = "10"

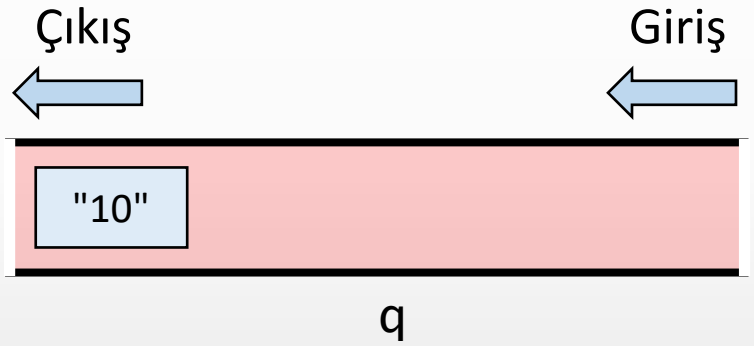
i = 0

n = 4

ikilikSayiUret(4);

```
String[] ikilikSayiUret(int n) {
  String[] sonuc = new String[n];
  Queue<String> q = new LinkedList<>();
  q.offer("1");
  for(int i = 0; i < n; i++) {
     sonuc[i] = q.poll();
     String n1 = sonuc[i] + "0";
     String n2 = sonuc[i] + "1";
     q.offer(n1);
     q.offer(n2);
  }
  return sonuc;
}
```





n2 = "11"

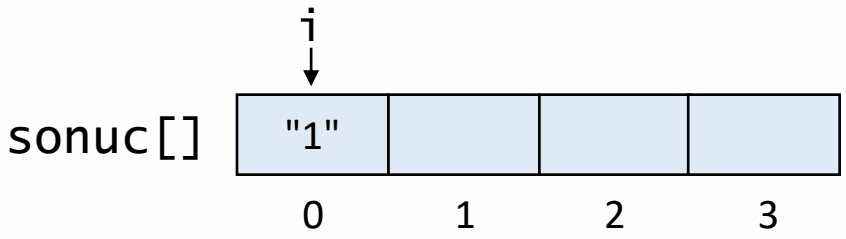
n1 = "10"

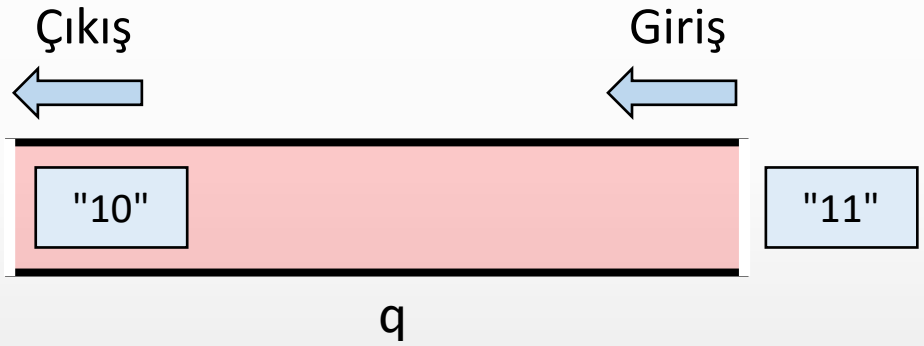
i = 0

n = 4

ikilikSayiUret(4);

```
String[] ikilikSayiUret(int n) {
  String[] sonuc = new String[n];
  Queue<String> q = new LinkedList<>();
  q.offer("1");
  for(int i = 0; i < n; i++) {
     sonuc[i] = q.poll();
     String n1 = sonuc[i] + "0";
     String n2 = sonuc[i] + "1";
     q.offer(n1);
     q.offer(n2);
  }
  return sonuc;
}
```





n2 = "11"

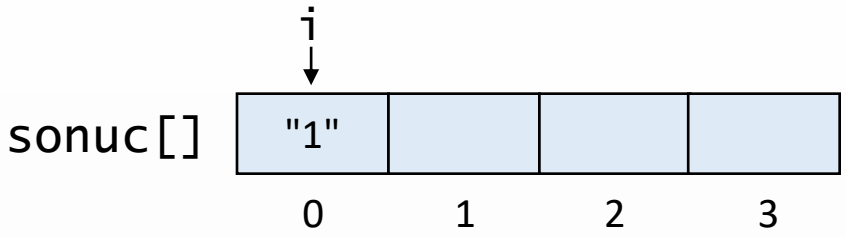
n1 = "10"

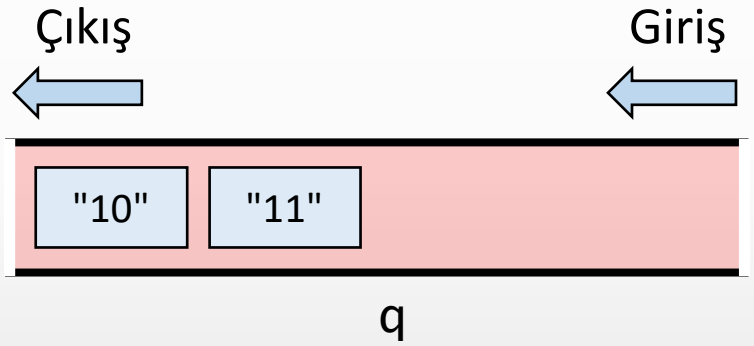
i = 0

n = 4

ikilikSayiUret(4);

```
String[] ikilikSayiUret(int n) {
  String[] sonuc = new String[n];
  Queue<String> q = new LinkedList<>();
  q.offer("1");
  for(int i = 0; i < n; i++) {
     sonuc[i] = q.poll();
     String n1 = sonuc[i] + "0";
     String n2 = sonuc[i] + "1";
     q.offer(n1);
     q.offer(n2);
  }
  return sonuc;
}
```





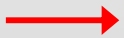
n2 = "11"

n1 = "10"

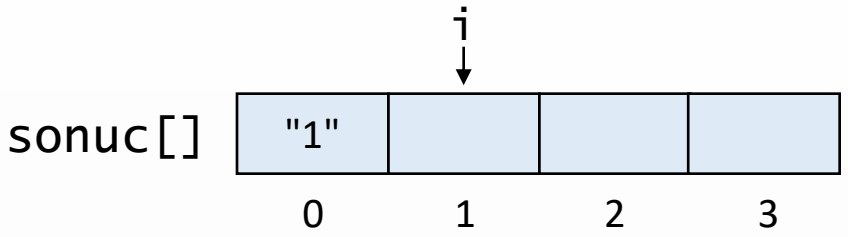
i = 0

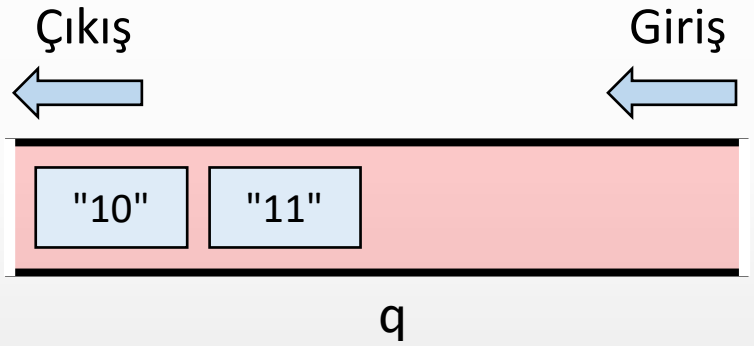
n = 4

ikilikSayiUret(4);



```
String[] ikilikSayiUret(int n) {
  String[] sonuc = new String[n];
  Queue<String> q = new LinkedList<>();
  q.offer("1");
  for(int i = 0; i < n; i++) {
     sonuc[i] = q.poll();
     String n1 = sonuc[i] + "0";
     String n2 = sonuc[i] + "1";
     q.offer(n1);
     q.offer(n2);
  }
  return sonuc;
}
```



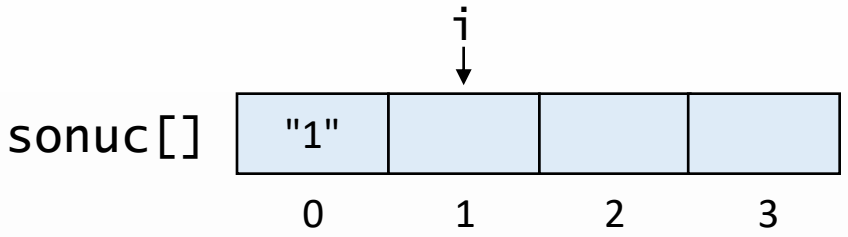


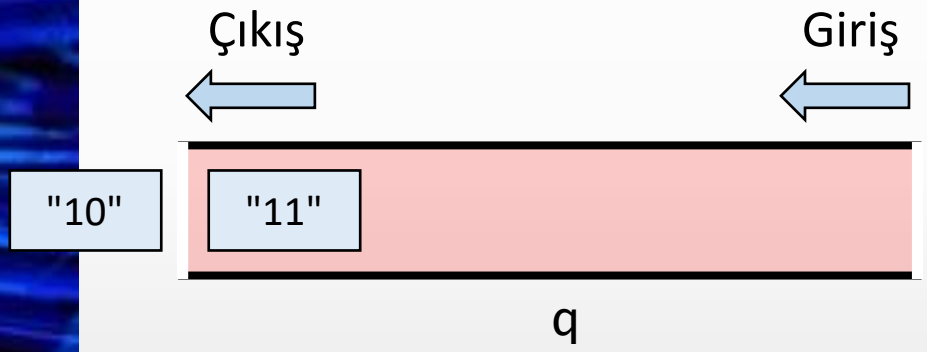
i = 1

n = 4

ikilikSayiUret(4);

```
String[] ikilikSayiUret(int n) {
  String[] sonuc = new String[n];
  Queue<String> q = new LinkedList<>();
  q.offer("1");
  for(int i = 0; i < n; i++) {
     sonuc[i] = q.poll();
     String n1 = sonuc[i] + "0";
     String n2 = sonuc[i] + "1";
     q.offer(n1);
     q.offer(n2);
  }
  return sonuc;
}
```



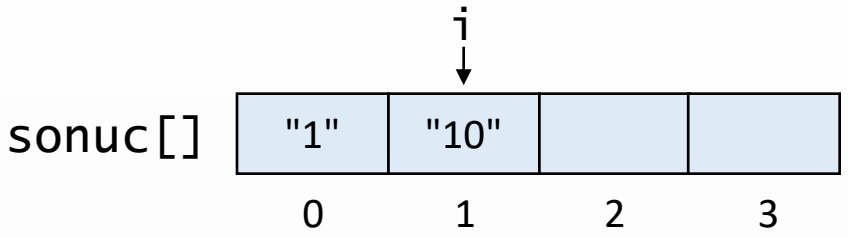


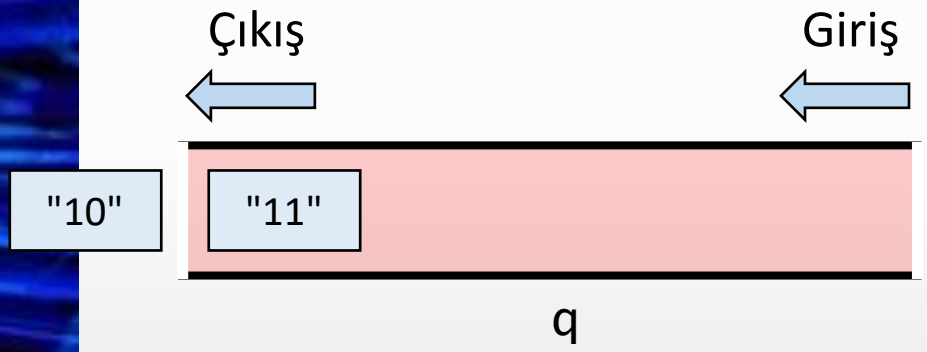
i = 1

n = 4

ikilikSayiUret(4);

```
String[] ikilikSayiUret(int n) {
  String[] sonuc = new String[n];
  Queue<String> q = new LinkedList<>();
  q.offer("1");
  for(int i = 0; i < n; i++) {
     sonuc[i] = q.poll();
     String n1 = sonuc[i] + "0";
     String n2 = sonuc[i] + "1";
     q.offer(n1);
     q.offer(n2);
  }
  return sonuc;
}
```



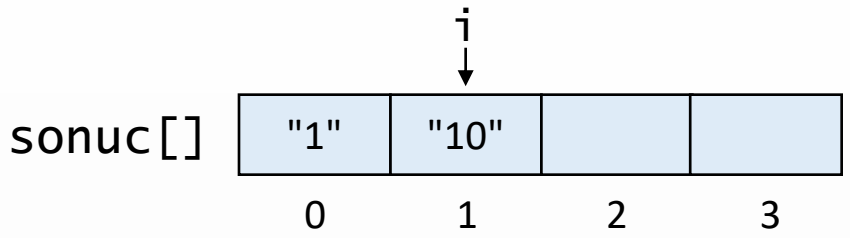


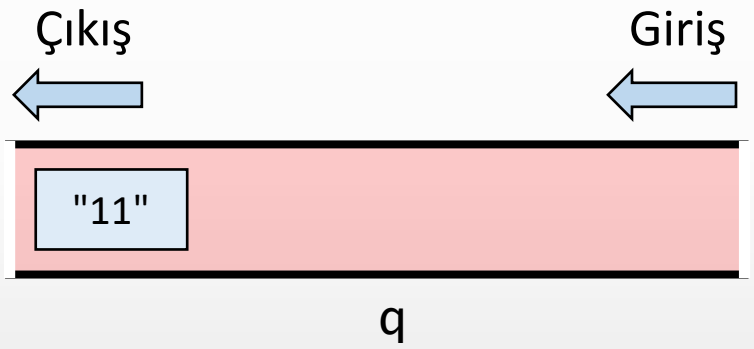
i = 1

n = 4

ikilikSayiUret(4);

```
String[] ikilikSayiUret(int n) {
  String[] sonuc = new String[n];
  Queue<String> q = new LinkedList<>();
  q.offer("1");
  for(int i = 0; i < n; i++) {
     sonuc[i] = q.poll();
     String n1 = sonuc[i] + "0";
     String n2 = sonuc[i] + "1";
     q.offer(n1);
     q.offer(n2);
  }
  return sonuc;
}
```



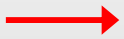


n1 = "100"

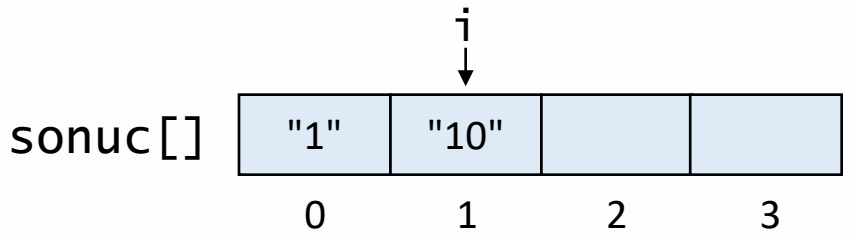
i = 1

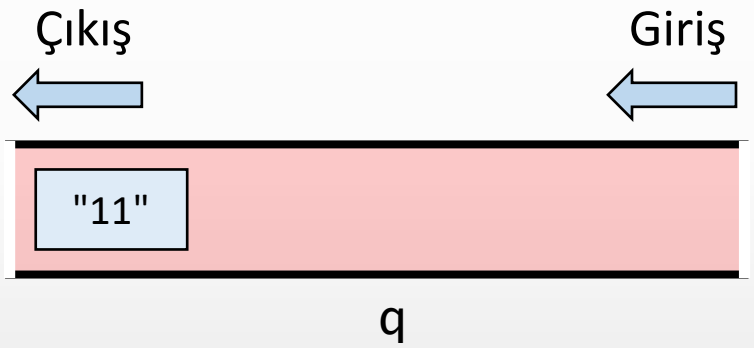
n = 4

ikilikSayiUret(4);



```
String[] ikilikSayiUret(int n) {
  String[] sonuc = new String[n];
  Queue<String> q = new LinkedList<>();
  q.offer("1");
  for(int i = 0; i < n; i++) {
     sonuc[i] = q.poll();
     String n1 = sonuc[i] + "0";
     String n2 = sonuc[i] + "1";
     q.offer(n1);
     q.offer(n2);
  }
  return sonuc;
}
```





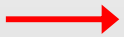
n2 = "101"

n1 = "100"

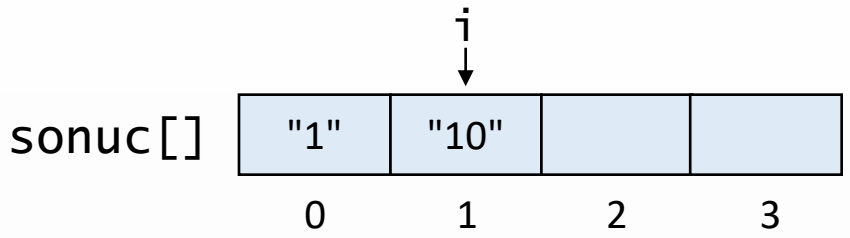
i = 1

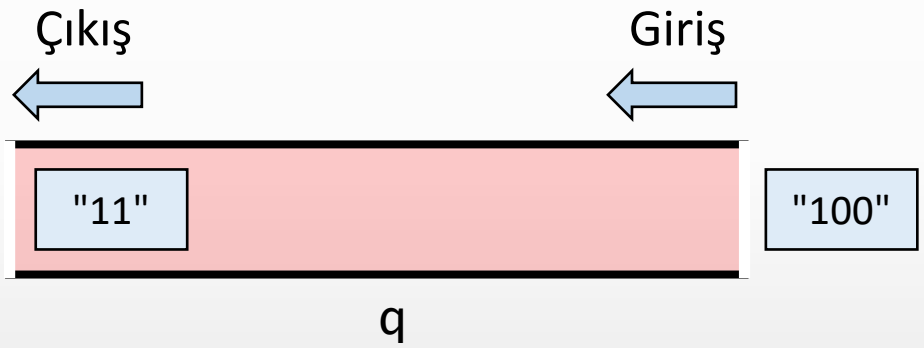
n = 4

ikilikSayiUret(4);



```
String[] ikilikSayiUret(int n) {
  String[] sonuc = new String[n];
  Queue<String> q = new LinkedList<>();
  q.offer("1");
  for(int i = 0; i < n; i++) {
     sonuc[i] = q.poll();
     String n1 = sonuc[i] + "0";
     String n2 = sonuc[i] + "1";
     q.offer(n1);
     q.offer(n2);
  }
  return sonuc;
}
```





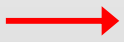
n2 = "101"

n1 = "100"

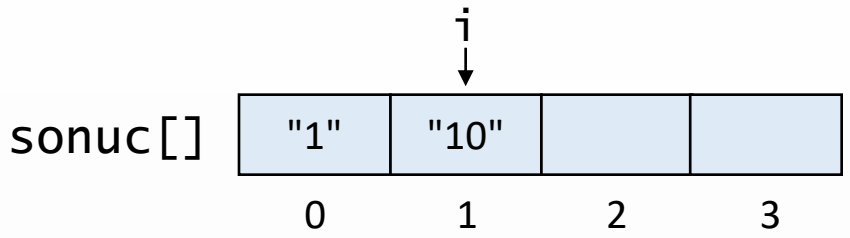
i = 1

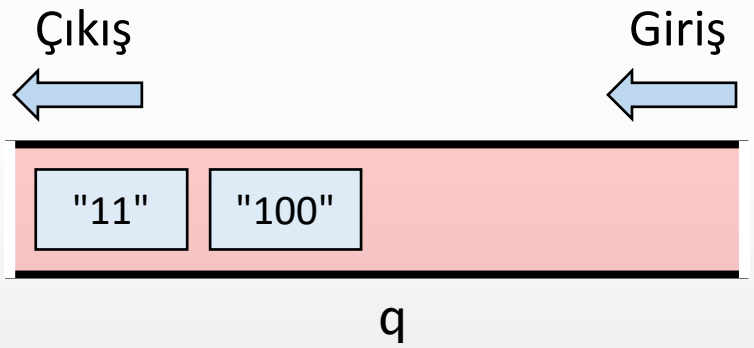
n = 4

ikilikSayiUret(4);



```
String[] ikilikSayiUret(int n) {
  String[] sonuc = new String[n];
  Queue<String> q = new LinkedList<>();
  q.offer("1");
  for(int i = 0; i < n; i++) {
     sonuc[i] = q.poll();
     String n1 = sonuc[i] + "0";
     String n2 = sonuc[i] + "1";
     q.offer(n1);
     q.offer(n2);
  }
  return sonuc;
}
```





n2 = "101"

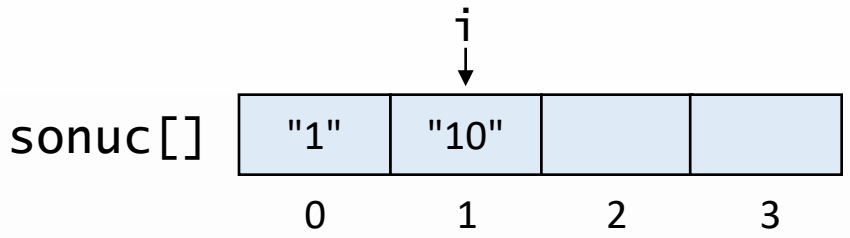
n1 = "100"

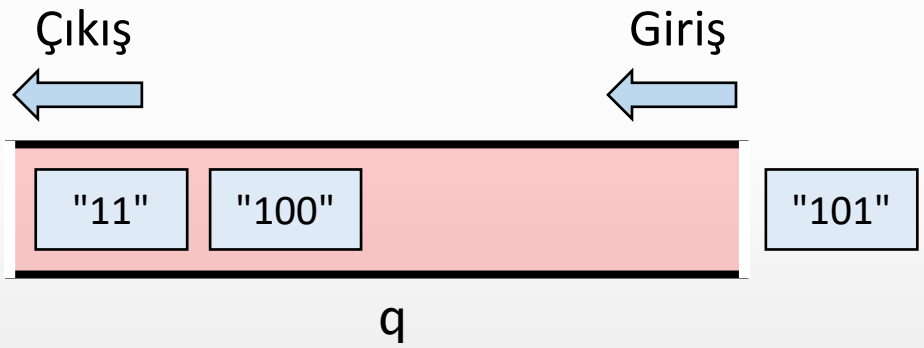
i = 1

n = 4

ikilikSayiUret(4);

```
String[] ikilikSayiUret(int n) {
  String[] sonuc = new String[n];
  Queue<String> q = new LinkedList<>();
  q.offer("1");
  for(int i = 0; i < n; i++) {
     sonuc[i] = q.poll();
     String n1 = sonuc[i] + "0";
     String n2 = sonuc[i] + "1";
     q.offer(n1);
     q.offer(n2);
  }
  return sonuc;
}
```





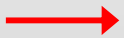
n2 = "101"

n1 = "100"

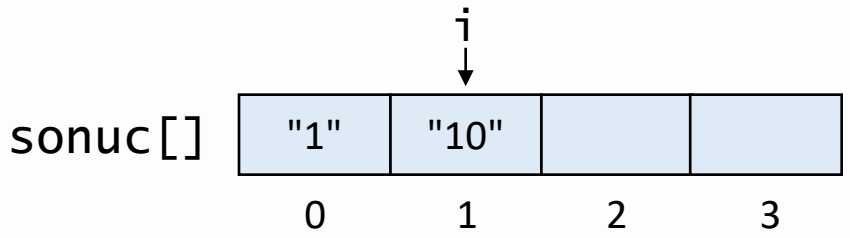
i = 1

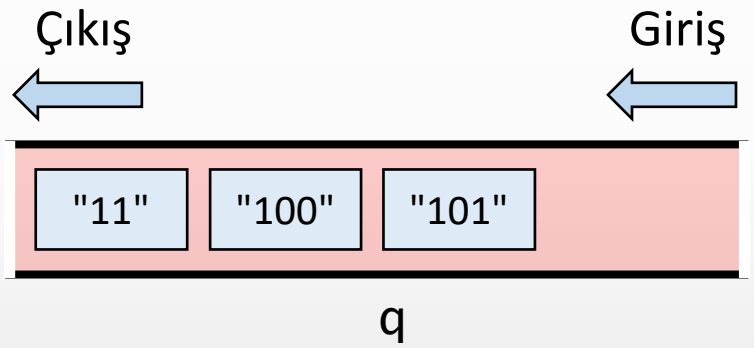
n = 4

ikilikSayiUret(4);



```
String[] ikilikSayiUret(int n) {
  String[] sonuc = new String[n];
  Queue<String> q = new LinkedList<>();
  q.offer("1");
  for(int i = 0; i < n; i++) {
     sonuc[i] = q.poll();
     String n1 = sonuc[i] + "0";
     String n2 = sonuc[i] + "1";
     q.offer(n1);
     q.offer(n2);
  }
  return sonuc;
}
```





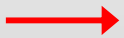
n2 = "101"

n1 = "100"

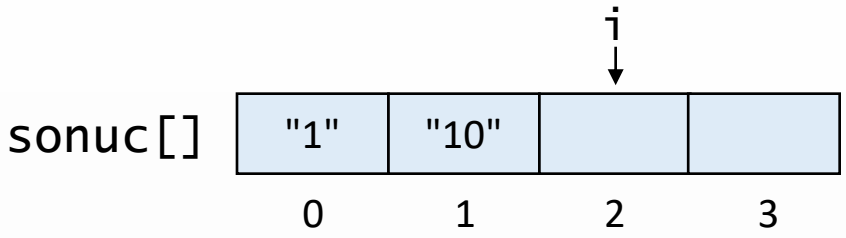
i = 1

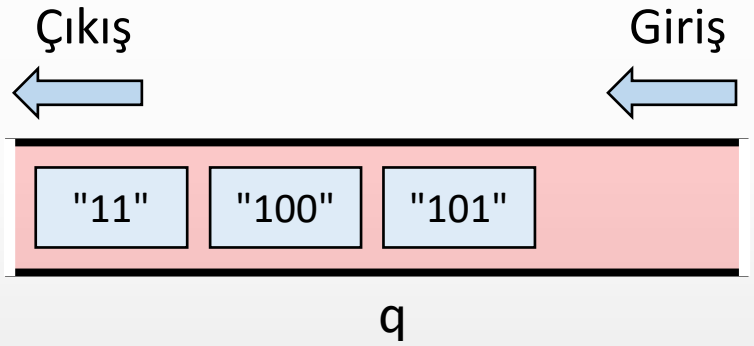
n = 4

ikilikSayiUret(4);



```
String[] ikilikSayiUret(int n) {
  String[] sonuc = new String[n];
  Queue<String> q = new LinkedList<>();
  q.offer("1");
  for(int i = 0; i < n; i++) {
     sonuc[i] = q.poll();
     String n1 = sonuc[i] + "0";
     String n2 = sonuc[i] + "1";
     q.offer(n1);
     q.offer(n2);
  }
  return sonuc;
}
```



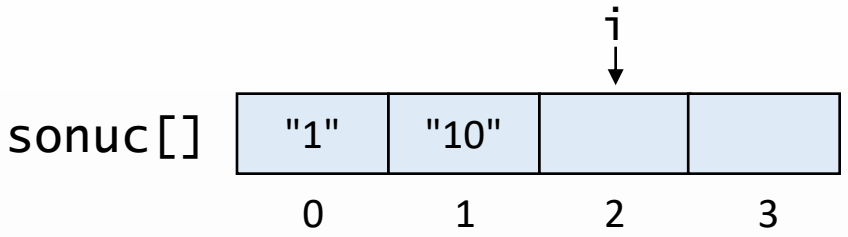


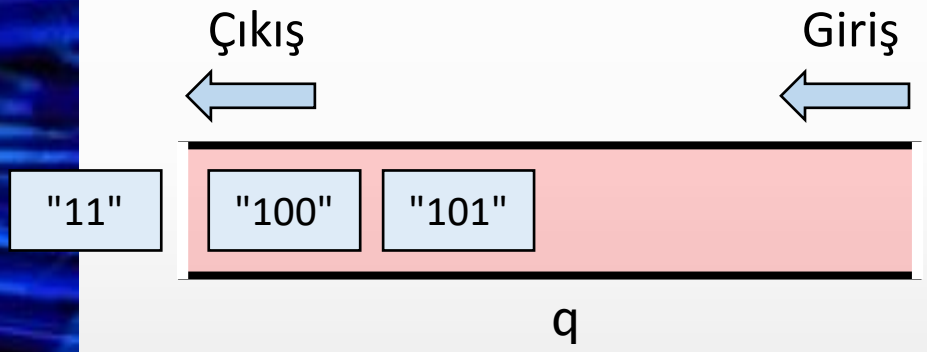
i = 2

n = 4

ikilikSayiUret(4);

```
String[] ikilikSayiUret(int n) {
  String[] sonuc = new String[n];
  Queue<String> q = new LinkedList<>();
  q.offer("1");
  for(int i = 0; i < n; i++) {
     sonuc[i] = q.poll();
     String n1 = sonuc[i] + "0";
     String n2 = sonuc[i] + "1";
     q.offer(n1);
     q.offer(n2);
  }
  return sonuc;
}
```



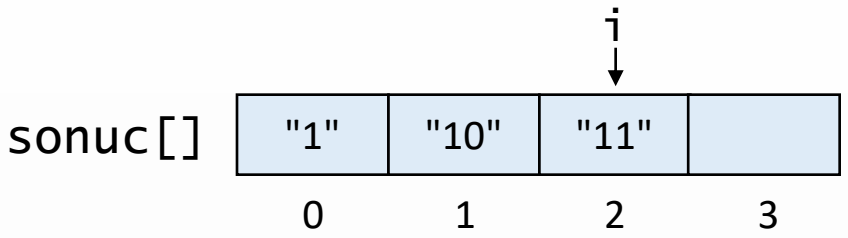


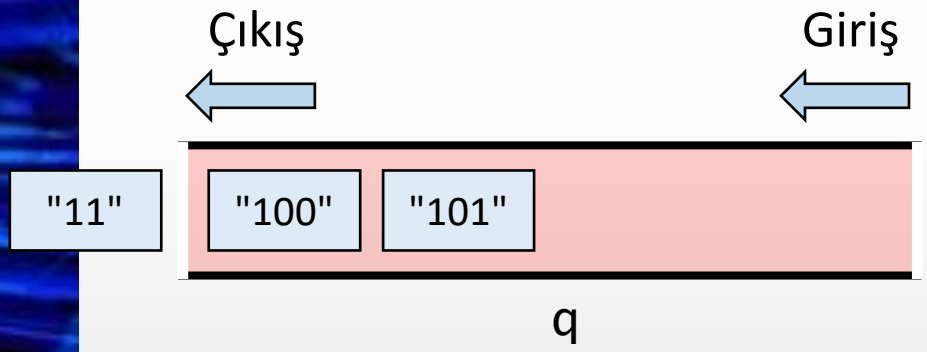
i = 2

n = 4

ikilikSayiUret(4);

```
String[] ikilikSayiUret(int n) {
  String[] sonuc = new String[n];
  Queue<String> q = new LinkedList<>();
  q.offer("1");
  for(int i = 0; i < n; i++) {
     sonuc[i] = q.poll();
     String n1 = sonuc[i] + "0";
     String n2 = sonuc[i] + "1";
     q.offer(n1);
     q.offer(n2);
  }
  return sonuc;
}
```

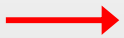




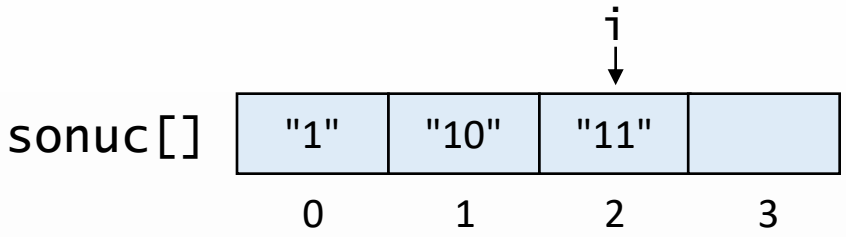
i = 2

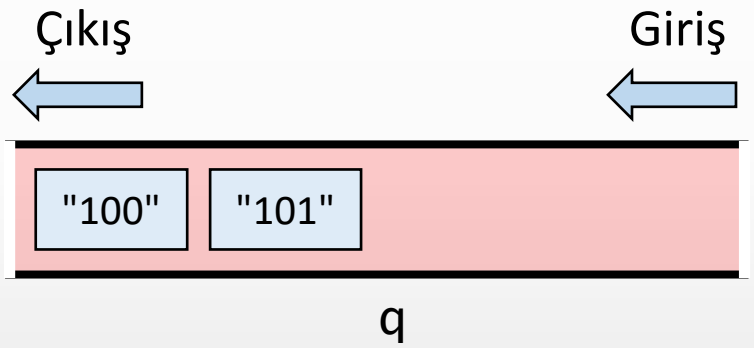
n = 4

ikilikSayiUret(4);



```
String[] ikilikSayiUret(int n) {
  String[] sonuc = new String[n];
  Queue<String> q = new LinkedList<>();
  q.offer("1");
  for(int i = 0; i < n; i++) {
     sonuc[i] = q.poll();
     String n1 = sonuc[i] + "0";
     String n2 = sonuc[i] + "1";
     q.offer(n1);
     q.offer(n2);
  }
  return sonuc;
}
```



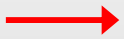


n1 = "110"

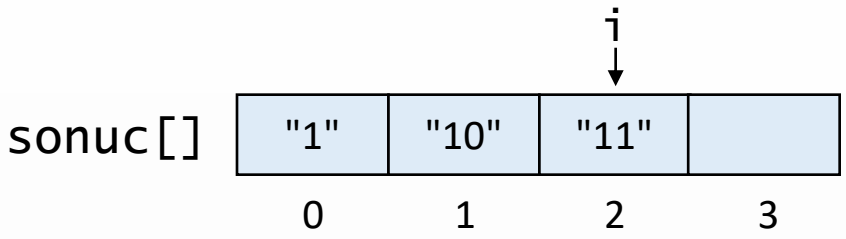
i = 2

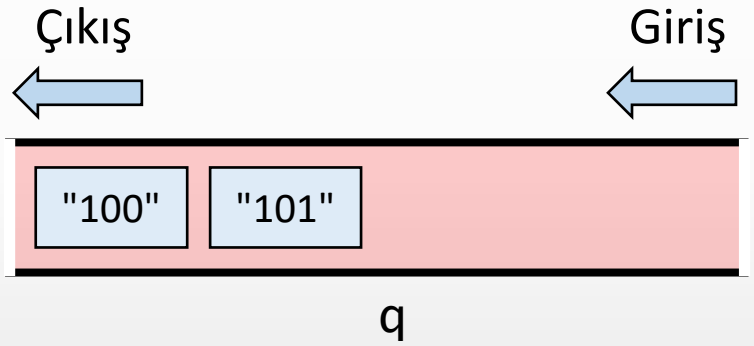
n = 4

ikilikSayiUret(4);



```
String[] ikilikSayiUret(int n) {
  String[] sonuc = new String[n];
  Queue<String> q = new LinkedList<>();
  q.offer("1");
  for(int i = 0; i < n; i++) {
     sonuc[i] = q.poll();
     String n1 = sonuc[i] + "0";
     String n2 = sonuc[i] + "1";
     q.offer(n1);
     q.offer(n2);
  }
  return sonuc;
}
```





n2 = "111"

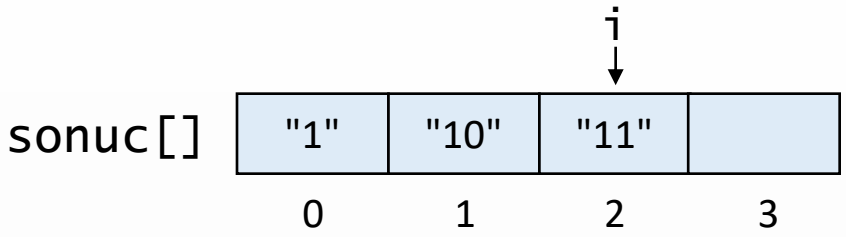
n1 = "110"

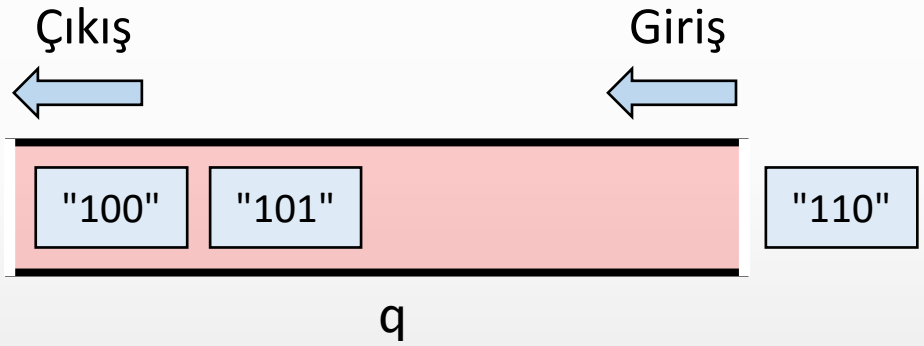
i = 2

n = 4

ikilikSayiUret(4);

```
String[] ikilikSayiUret(int n) {
  String[] sonuc = new String[n];
  Queue<String> q = new LinkedList<>();
  q.offer("1");
  for(int i = 0; i < n; i++) {
     sonuc[i] = q.poll();
     String n1 = sonuc[i] + "0";
     String n2 = sonuc[i] + "1";
     q.offer(n1);
     q.offer(n2);
  }
  return sonuc;
}
```





n2 = "111"

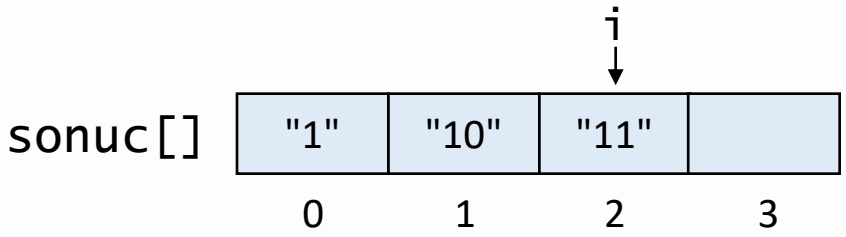
n1 = "110"

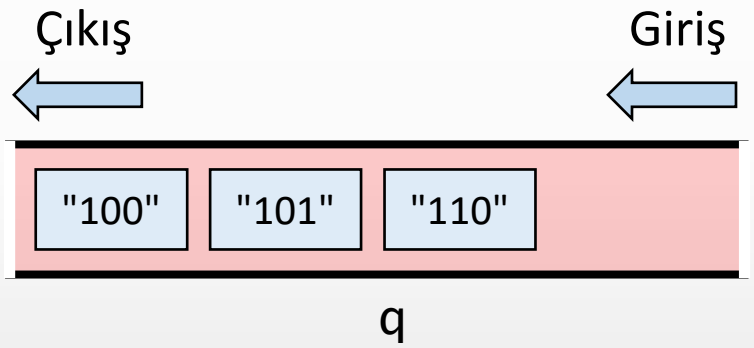
i = 2

n = 4

ikilikSayiUret(4);

```
String[] ikilikSayiUret(int n) {
  String[] sonuc = new String[n];
  Queue<String> q = new LinkedList<>();
  q.offer("1");
  for(int i = 0; i < n; i++) {
     sonuc[i] = q.poll();
     String n1 = sonuc[i] + "0";
     String n2 = sonuc[i] + "1";
     q.offer(n1);
     q.offer(n2);
  }
  return sonuc;
}
```





n2 = "111"

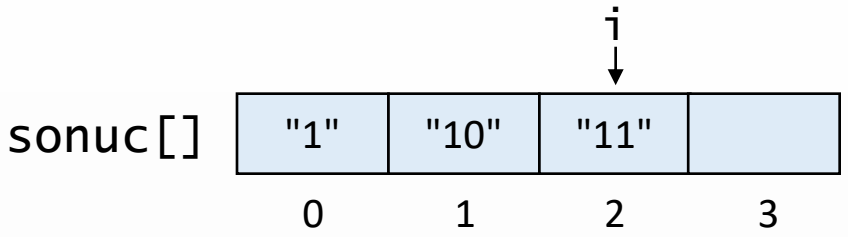
n1 = "110"

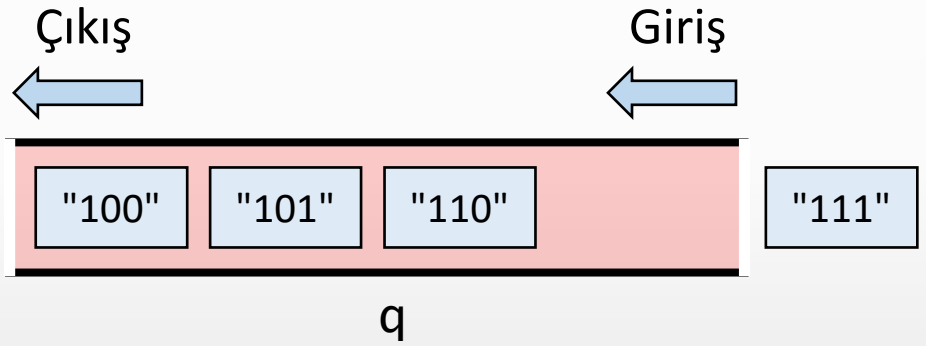
i = 2

n = 4

ikilikSayiUret(4);

```
String[] ikilikSayiUret(int n) {
  String[] sonuc = new String[n];
  Queue<String> q = new LinkedList<>();
  q.offer("1");
  for(int i = 0; i < n; i++) {
     sonuc[i] = q.poll();
     String n1 = sonuc[i] + "0";
     String n2 = sonuc[i] + "1";
     q.offer(n1);
     q.offer(n2);
  }
  return sonuc;
}
```





n2 = "111"

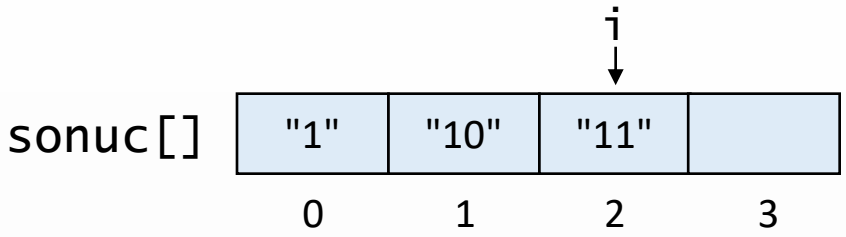
n1 = "110"

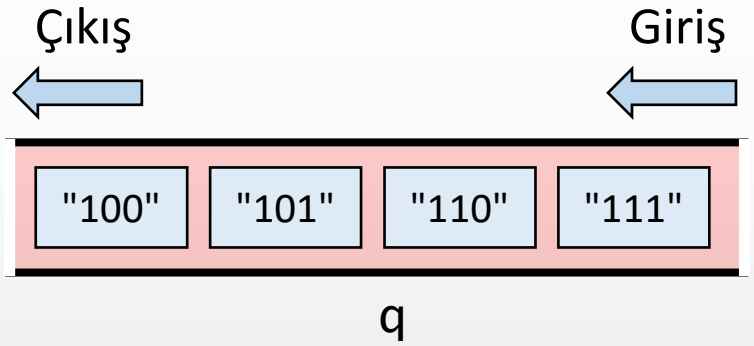
i = 2

n = 4

ikilikSayiUret(4);

```
String[] ikilikSayiUret(int n) {
  String[] sonuc = new String[n];
  Queue<String> q = new LinkedList<>();
  q.offer("1");
  for(int i = 0; i < n; i++) {
     sonuc[i] = q.poll();
     String n1 = sonuc[i] + "0";
     String n2 = sonuc[i] + "1";
     q.offer(n1);
     q.offer(n2);
  }
  return sonuc;
}
```





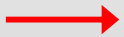
n2 = "111"

n1 = "110"

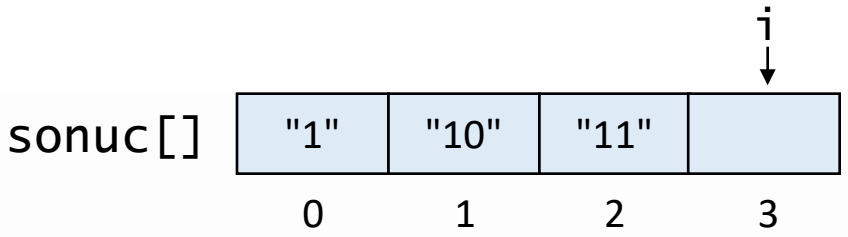
i = 2

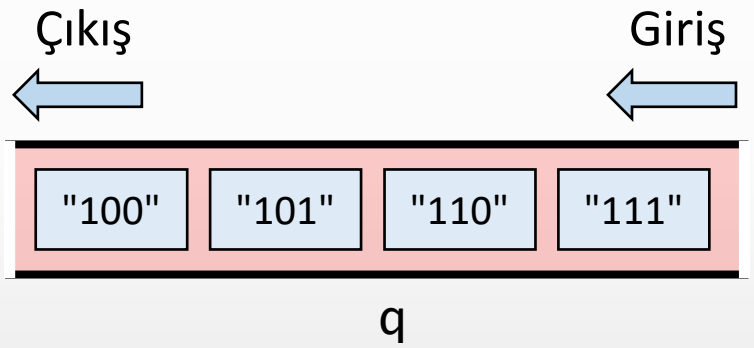
n = 4

ikilikSayiUret(4);



```
String[] ikilikSayiUret(int n) {
  String[] sonuc = new String[n];
  Queue<String> q = new LinkedList<>();
  q.offer("1");
  for(int i = 0; i < n; i++) {
     sonuc[i] = q.poll();
     String n1 = sonuc[i] + "0";
     String n2 = sonuc[i] + "1";
     q.offer(n1);
     q.offer(n2);
  }
  return sonuc;
}
```



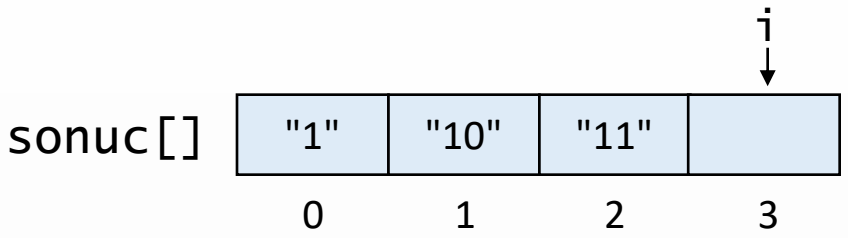


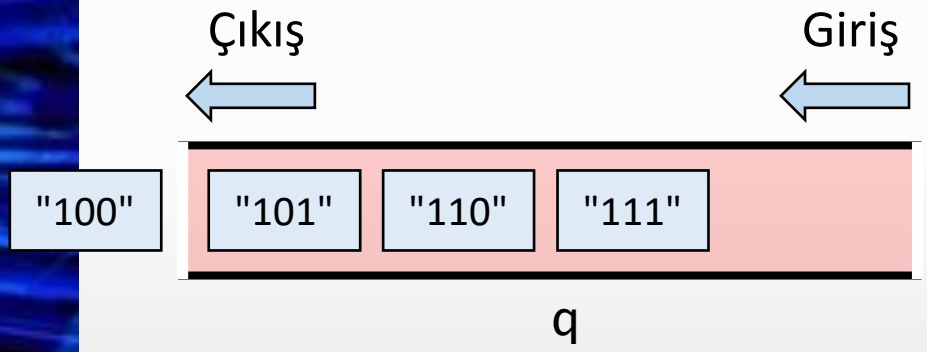
i = 3

n = 4

ikilikSayiUret(4);


```
String[] ikilikSayiUret(int n) {
  String[] sonuc = new String[n];
  Queue<String> q = new LinkedList<>();
  q.offer("1");
  for(int i = 0; i < n; i++) {
     sonuc[i] = q.poll();
     String n1 = sonuc[i] + "0";
     String n2 = sonuc[i] + "1";
     q.offer(n1);
     q.offer(n2);
  }
  return sonuc;
}
```



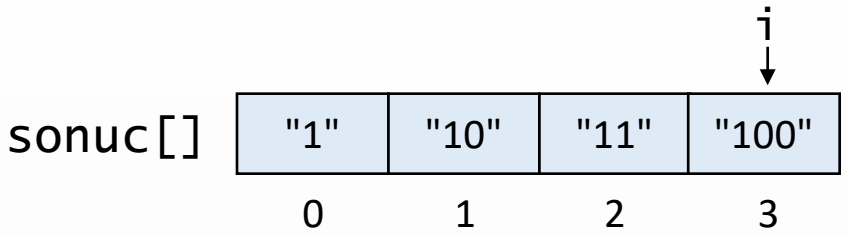


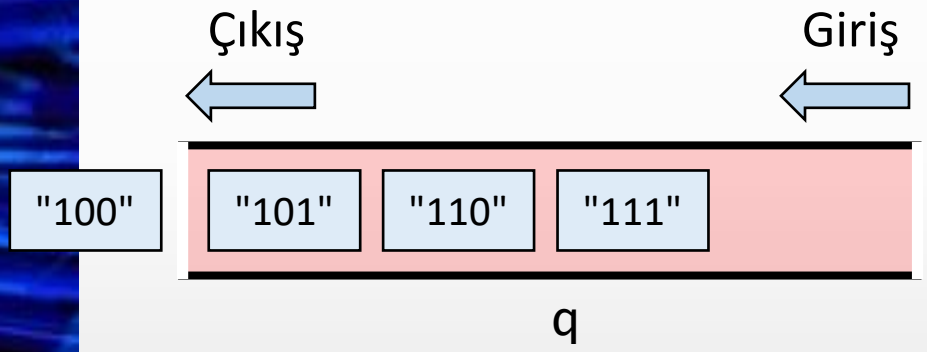
i = 3

n = 4

ikilikSayiUret(4);

```
String[] ikilikSayiUret(int n) {
  String[] sonuc = new String[n];
  Queue<String> q = new LinkedList<>();
  q.offer("1");
  for(int i = 0; i < n; i++) {
     sonuc[i] = q.poll();
     String n1 = sonuc[i] + "0";
     String n2 = sonuc[i] + "1";
     q.offer(n1);
     q.offer(n2);
  }
  return sonuc;
}
```



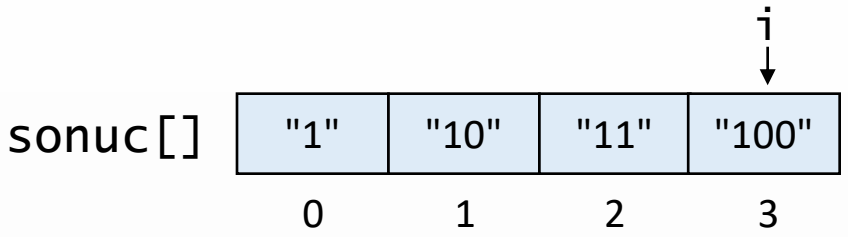


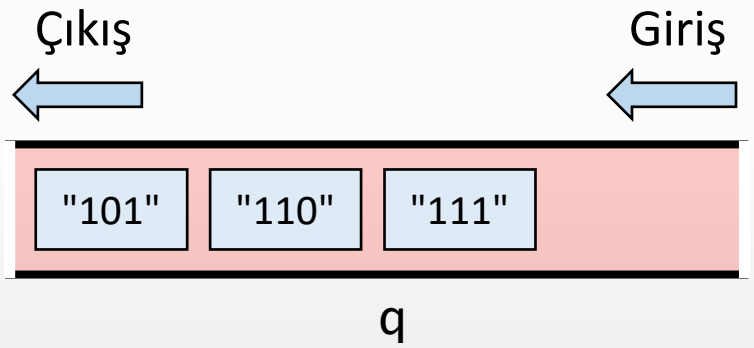
i = 3

n = 4

ikilikSayiUret(4);

```
String[] ikilikSayiUret(int n) {
  String[] sonuc = new String[n];
  Queue<String> q = new LinkedList<>();
  q.offer("1");
  for(int i = 0; i < n; i++) {
     sonuc[i] = q.poll();
     String n1 = sonuc[i] + "0";
     String n2 = sonuc[i] + "1";
     q.offer(n1);
     q.offer(n2);
  }
  return sonuc;
}
```



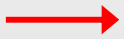


n1 = "1000"

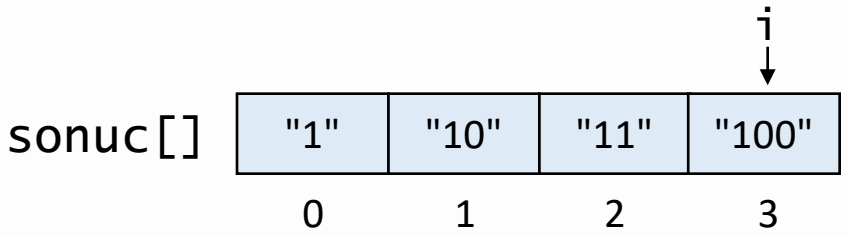
i = 3

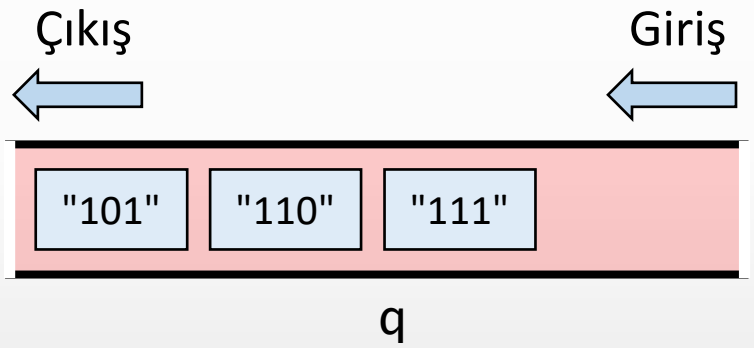
n = 4

ikilikSayiUret(4);



```
String[] ikilikSayiUret(int n) {
  String[] sonuc = new String[n];
  Queue<String> q = new LinkedList<>();
  q.offer("1");
  for(int i = 0; i < n; i++) {
     sonuc[i] = q.poll();
     String n1 = sonuc[i] + "0";
     String n2 = sonuc[i] + "1";
     q.offer(n1);
     q.offer(n2);
  }
  return sonuc;
}
```





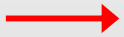
n2 = "1001"

n1 = "1000"

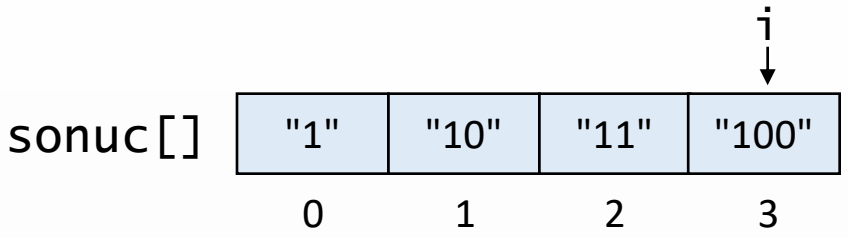
i = 3

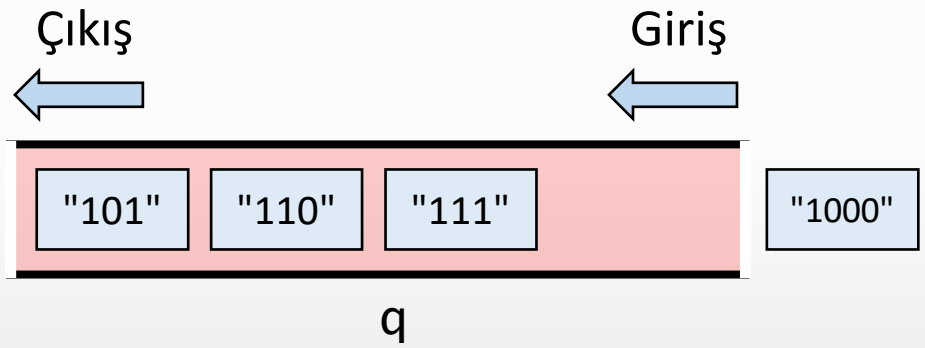
n = 4

ikilikSayiUret(4);



```
String[] ikilikSayiUret(int n) {
  String[] sonuc = new String[n];
  Queue<String> q = new LinkedList<>();
  q.offer("1");
  for(int i = 0; i < n; i++) {
     sonuc[i] = q.poll();
     String n1 = sonuc[i] + "0";
     String n2 = sonuc[i] + "1";
     q.offer(n1);
     q.offer(n2);
  }
  return sonuc;
}
```





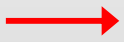
n2 = "1001"

n1 = "1000"

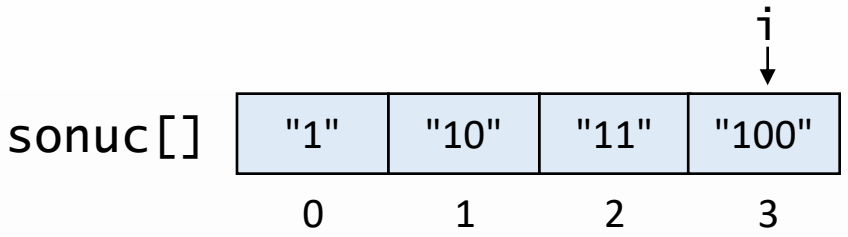
i = 3

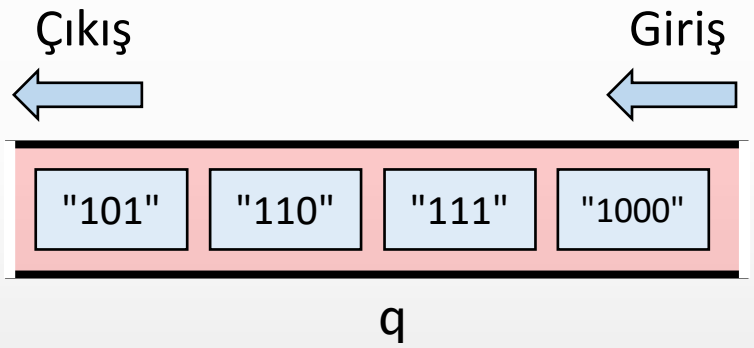
n = 4

ikilikSayiUret(4);



```
String[] ikilikSayiUret(int n) {
  String[] sonuc = new String[n];
  Queue<String> q = new LinkedList<>();
  q.offer("1");
  for(int i = 0; i < n; i++) {
     sonuc[i] = q.poll();
     String n1 = sonuc[i] + "0";
     String n2 = sonuc[i] + "1";
     q.offer(n1);
     q.offer(n2);
  }
  return sonuc;
}
```





n2 = "1001"

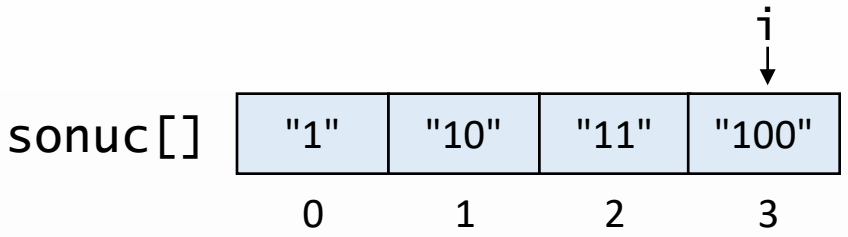
n1 = "1000"

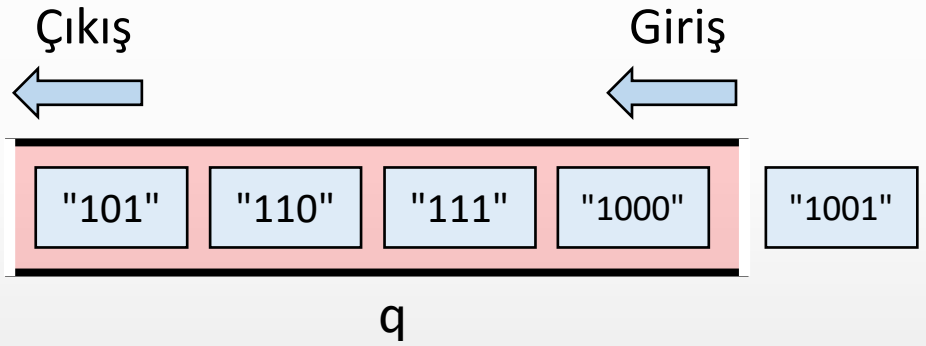
i = 3

n = 4

ikilikSayiUret(4);

```
String[] ikilikSayiUret(int n) {
  String[] sonuc = new String[n];
  Queue<String> q = new LinkedList<>();
  q.offer("1");
  for(int i = 0; i < n; i++) {
     sonuc[i] = q.poll();
     String n1 = sonuc[i] + "0";
     String n2 = sonuc[i] + "1";
     q.offer(n1);
     q.offer(n2);
  }
  return sonuc;
}
```





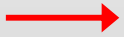
n2 = "1001"

n1 = "1000"

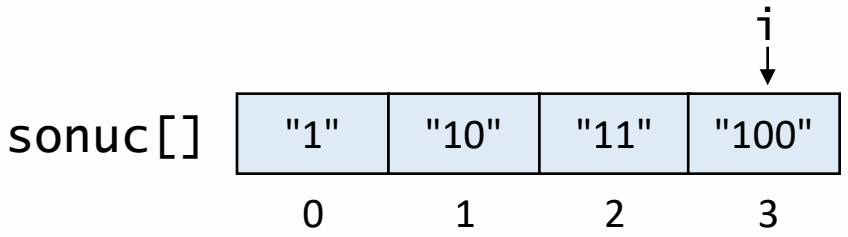
i = 3

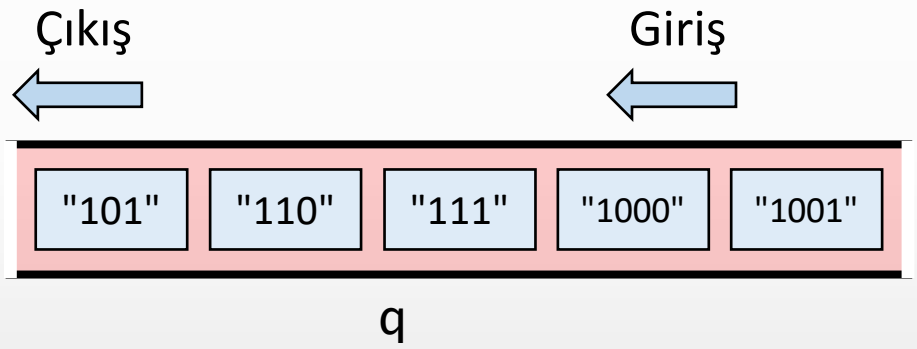
n = 4

ikilikSayiUret(4);



```
String[] ikilikSayiUret(int n) {
  String[] sonuc = new String[n];
  Queue<String> q = new LinkedList<>();
  q.offer("1");
  for(int i = 0; i < n; i++) {
     sonuc[i] = q.poll();
     String n1 = sonuc[i] + "0";
     String n2 = sonuc[i] + "1";
     q.offer(n1);
     q.offer(n2);
  }
  return sonuc;
}
```





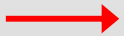
n2 = "1001"

n1 = "1000"

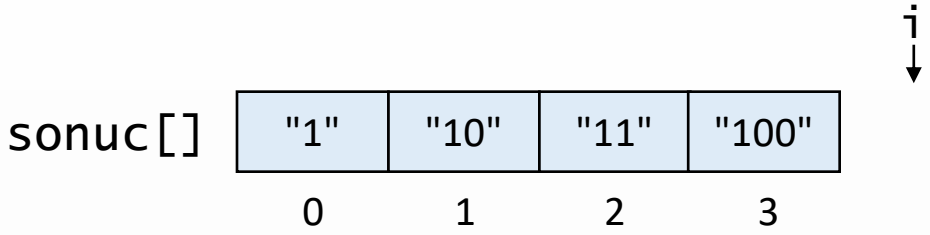
i = 3

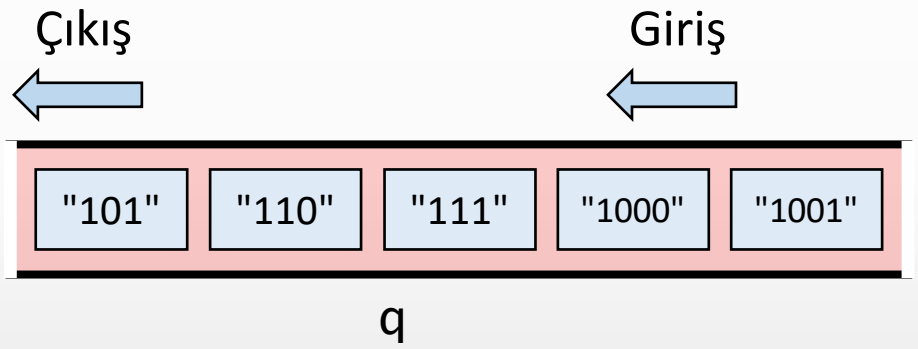
n = 4

ikilikSayiUret(4);



```
String[] ikilikSayiUret(int n) {
  String[] sonuc = new String[n];
  Queue<String> q = new LinkedList<>();
  q.offer("1");
  for(int i = 0; i < n; i++) {
     sonuc[i] = q.poll();
     String n1 = sonuc[i] + "0";
     String n2 = sonuc[i] + "1";
     q.offer(n1);
     q.offer(n2);
  }
  return sonuc;
}
```



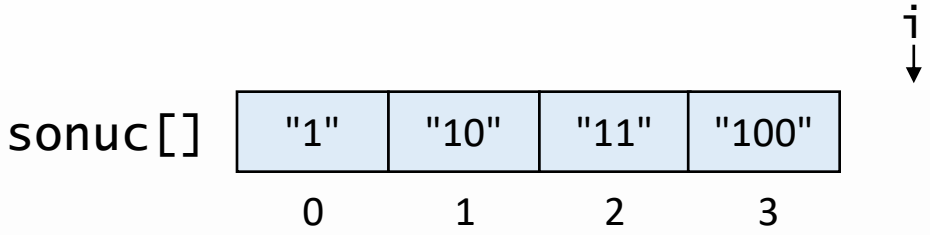


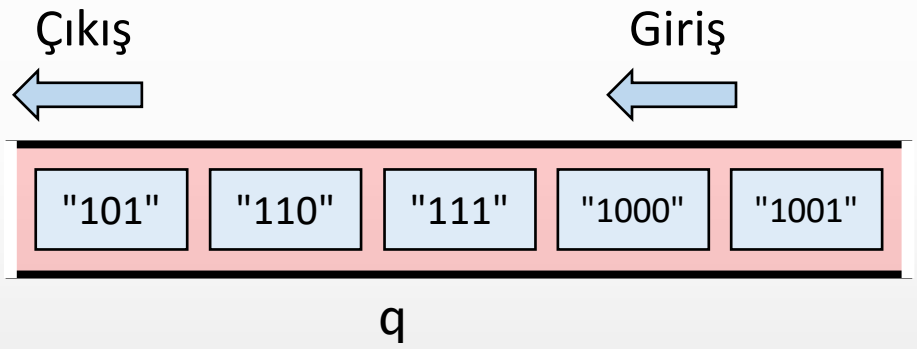
i = 4

n = 4

ikilikSayiUret(4);


```
String[] ikilikSayiUret(int n) {
  String[] sonuc = new String[n];
  Queue<String> q = new LinkedList<>();
  q.offer("1");
  for(int i = 0; i < n; i++) {
     sonuc[i] = q.poll();
     String n1 = sonuc[i] + "0";
     String n2 = sonuc[i] + "1";
     q.offer(n1);
     q.offer(n2);
  }
  return sonuc;
}
```

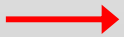




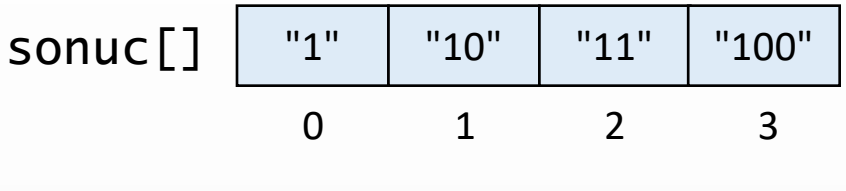
i = 4

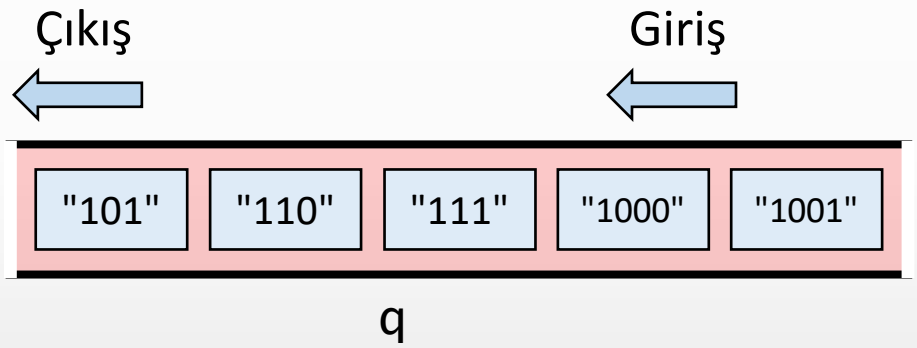
n = 4

ikilikSayiUret(4);



```
String[] ikilikSayiUret(int n) {
  String[] sonuc = new String[n];
  Queue<String> q = new LinkedList<>();
  q.offer("1");
  for(int i = 0; i < n; i++) {
     sonuc[i] = q.poll();
     String n1 = sonuc[i] + "0";
     String n2 = sonuc[i] + "1";
     q.offer(n1);
     q.offer(n2);
  }
  return sonuc;
}
```






```
ikilikSayiUret(4);
```

```
String[] ikilikSayiUret(int n) {
  String[] sonuc = new String[n];
  Queue<String> q = new LinkedList<>();
  q.offer("1");
  for(int i = 0; i < n; i++) {
     sonuc[i] = q.poll();
     String n1 = sonuc[i] + "0";
     String n2 = sonuc[i] + "1";
     q.offer(n1);
     q.offer(n2);
  }
  return sonuc;
}
```

SON