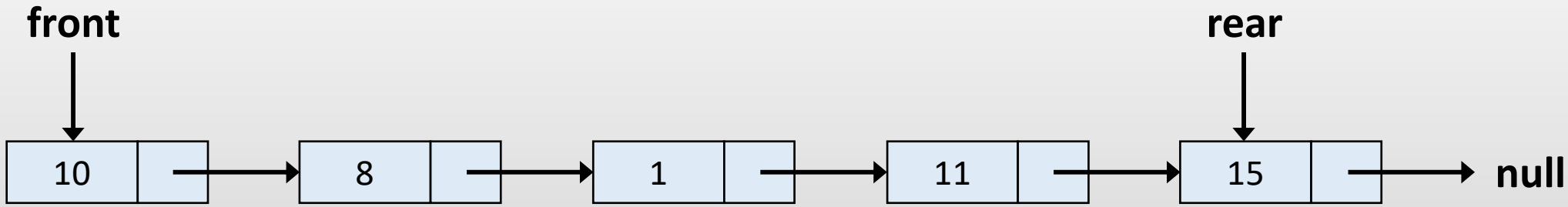
Bölüm 5: Kuyruk

Veri Yapıları

Kuyruk (Queue)

- Kuyruk, her iki ucu açık bir lineer veri yapısıdır.
- İşlemler, İlk Giren İlk Çıkar (FIFO) sırasına göre gerçekleştirilir.
- Listeye eklemeler bir uçtan, çıkarmalar diğer uçtan gerçekleştirilir.
- İlk eklenen öğe, ilk çıkarma işlemine tabi tutulan öğedir.



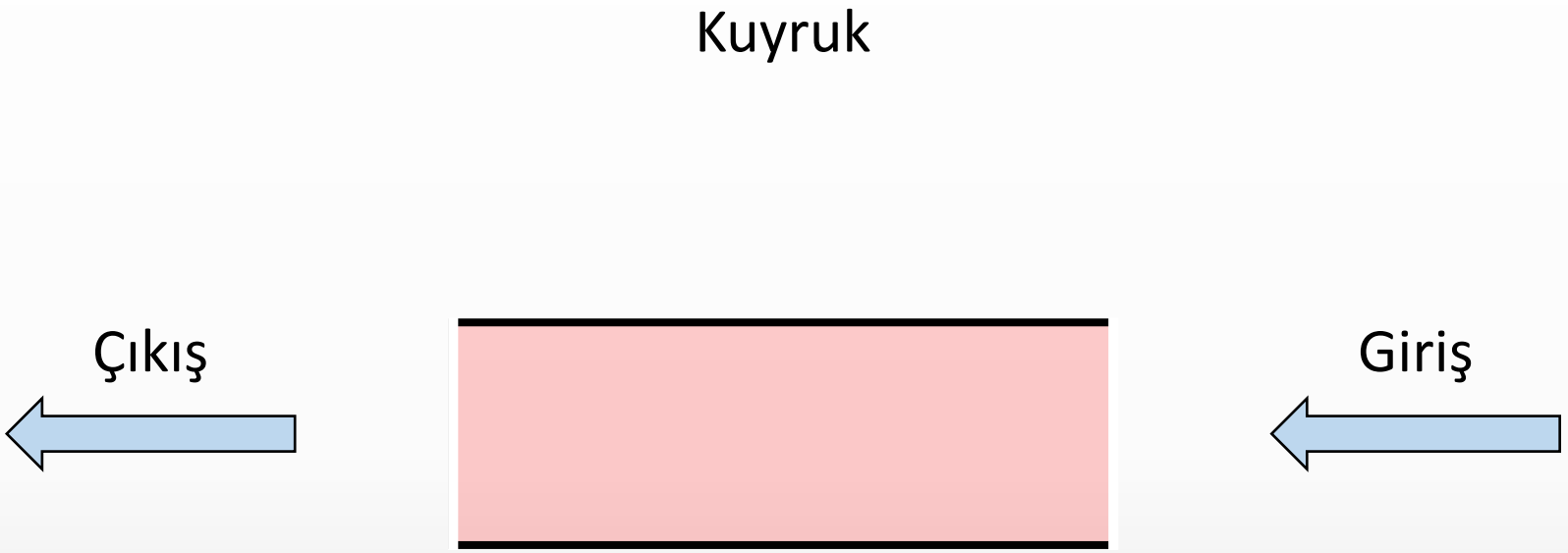
Temel Kuyruk İşlemleri

- Ekleme (Enqueue):
  - Kuyruğa yeni bir öğe ekler.
  - Yeni eklenen öğe kuyruğun sonunda yer alır.
  - Time complexity: O(1)
- Çıkarma (Dequeue):
  - Kuyruğun başındaki öğeyi kuyruktan çıkarır.
  - İlk giren öğeyi çıkarır (FIFO ilkesi).
  - Time complexity: O(1)

Kuyruk Giriş ve Çıkış Noktaları

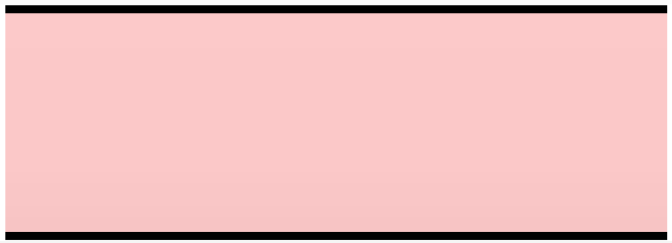
- Kuyruğa yeni öğeler eklerken ve öğeleri çıkartırken, her iki uç kullanılır.
- Kuyruğun ön tarafına eklenen öğe, kuyruğun başında bekleyen ilk öğedir. Buna "kuyruğun önü" (bazen "kuyruğun başı") denir.
- Kuyruğun son tarafına eklenen öğe, en son eklenen öğedir. Buna "kuyruğun sonu" (veya "kuyruğun kuyruğu") denir.
- Kuyruğun önü (başı), ilk hizmet alacak öğeyi temsil eder.
- Kuyruğun sonu (kuyruğu), en son eklenen öğeyi temsil eder.

Gösterim



Kuyruk

Çıkış



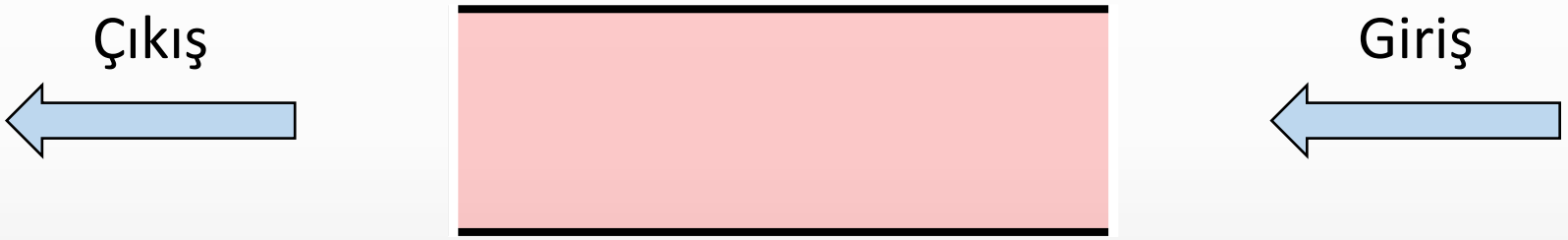
Giriş

front ⟶ **null**

rear ⟶ **null**

enqueue(20)

Kuyruk

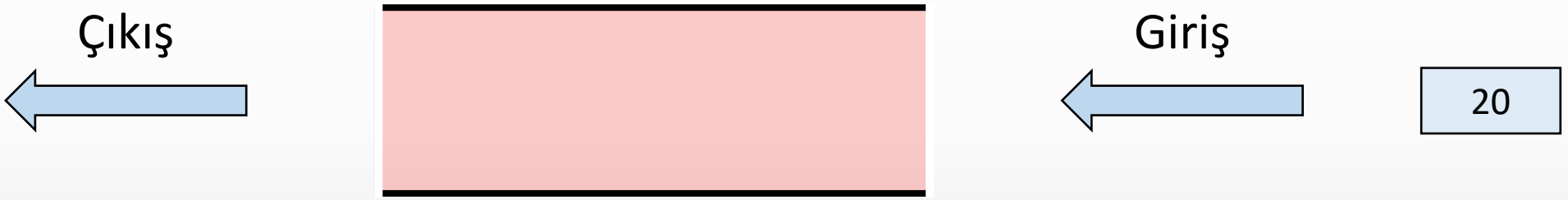


front ⟶ null

rear ⟶ null

enqueue(20)

Kuyruk

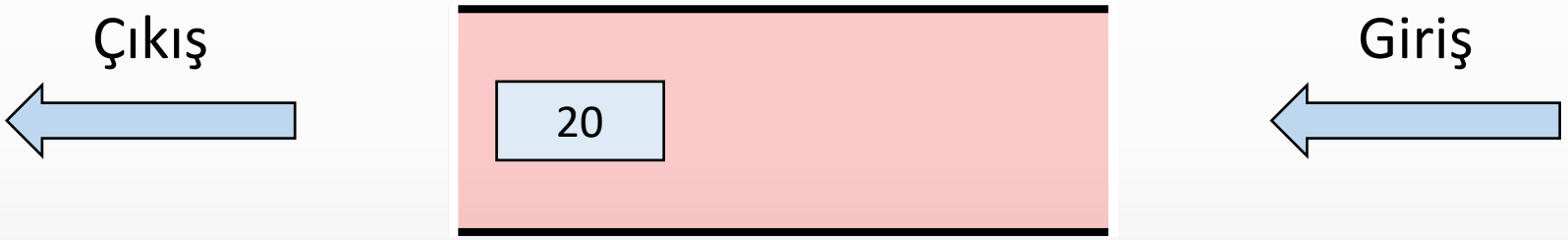


front ⟶ null

rear ⟶ null

enqueue(20)

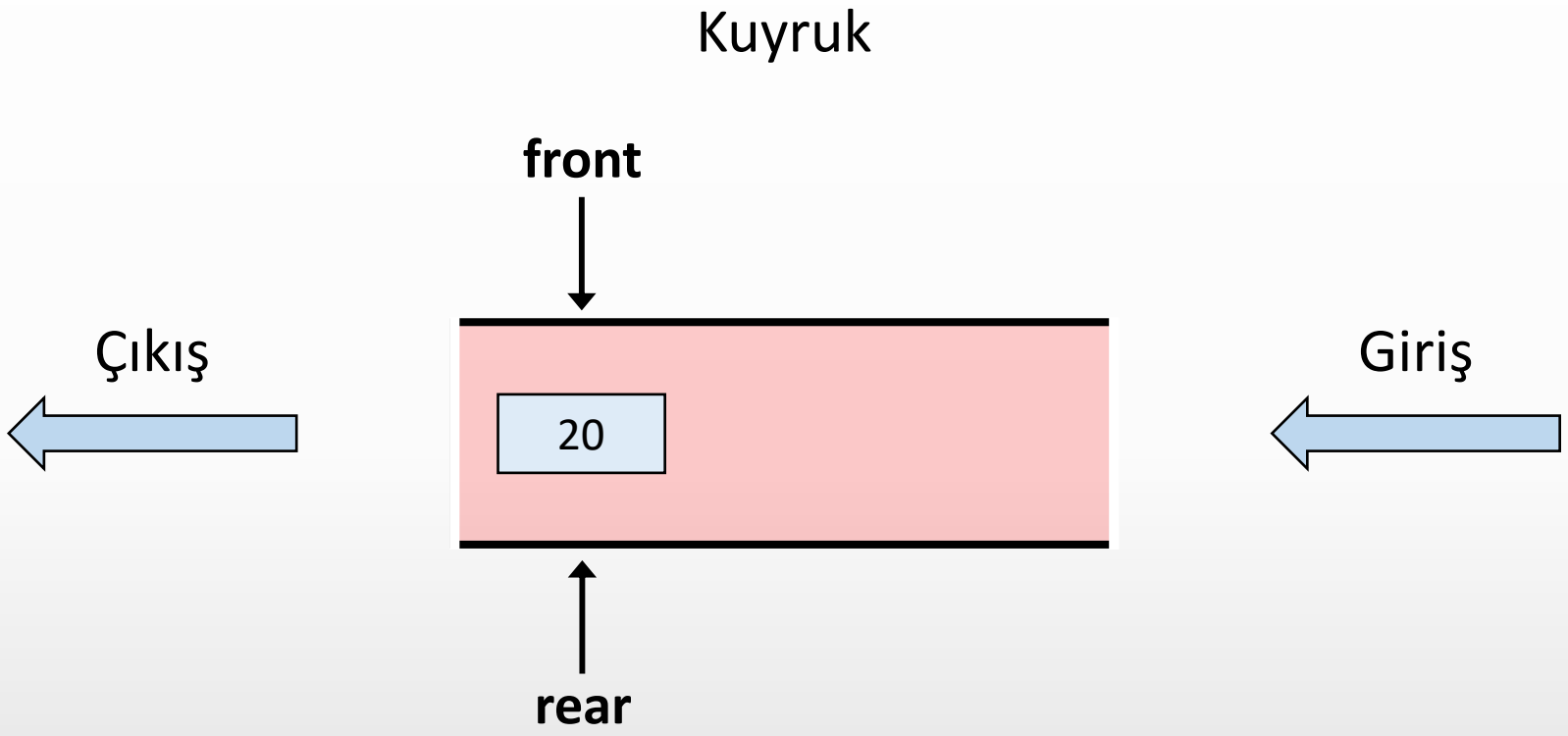
Kuyruk



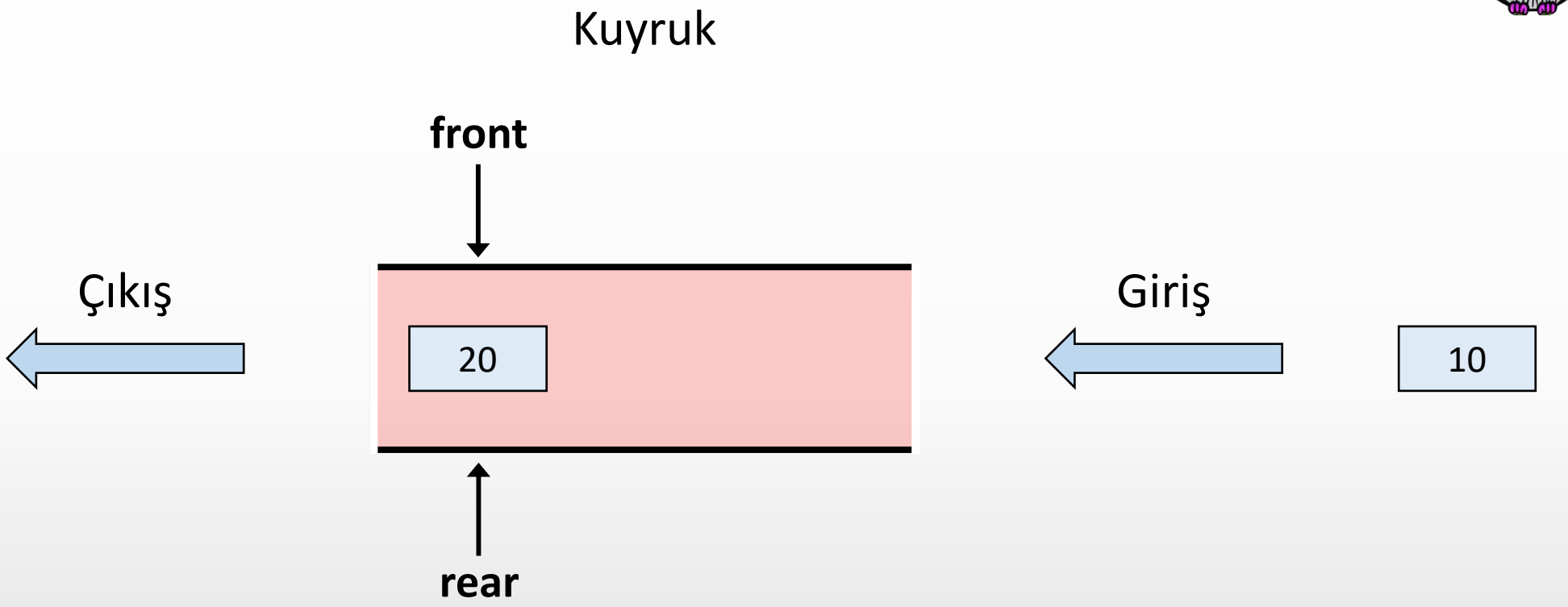
front ⟶ null

rear ⟶ null

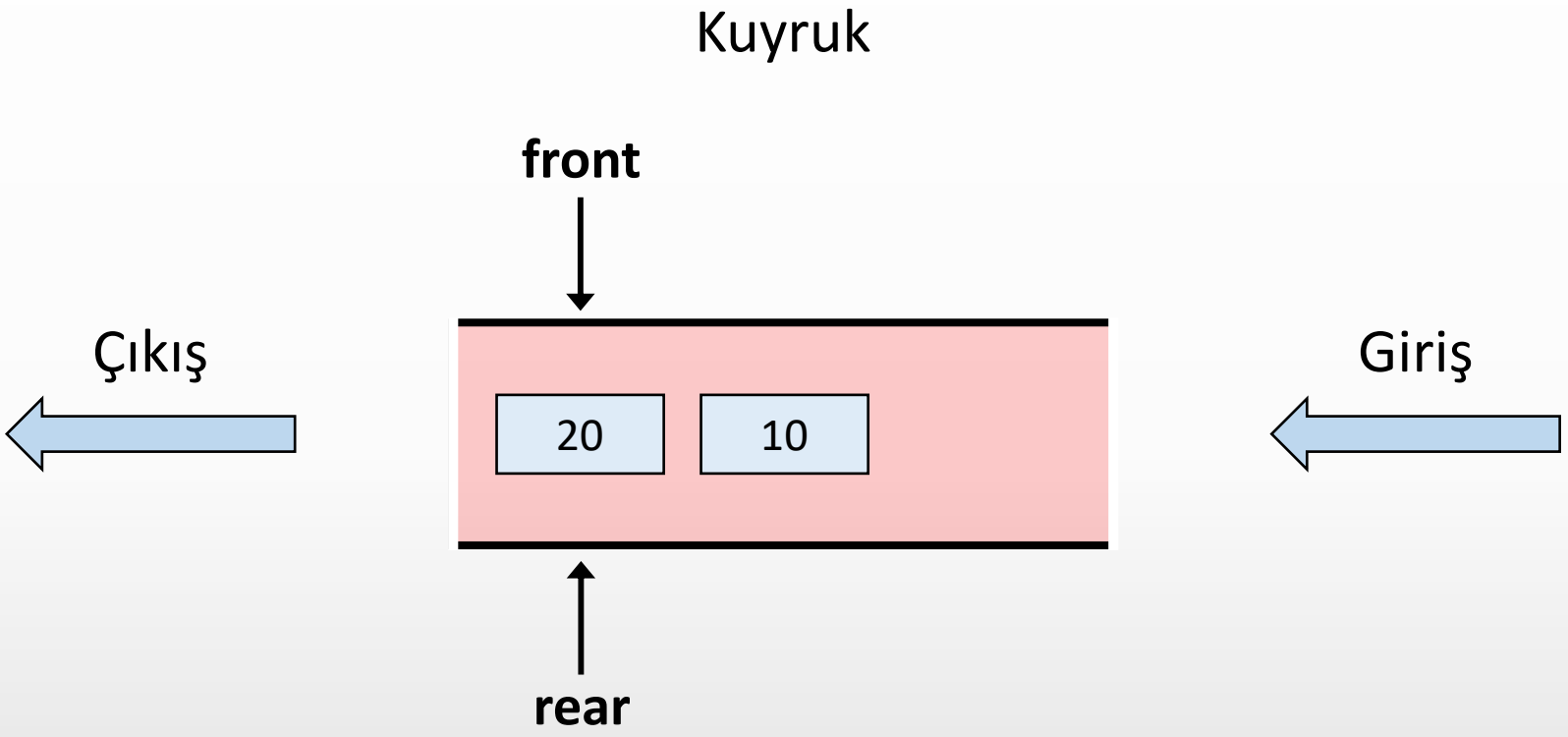
enqueue(20)



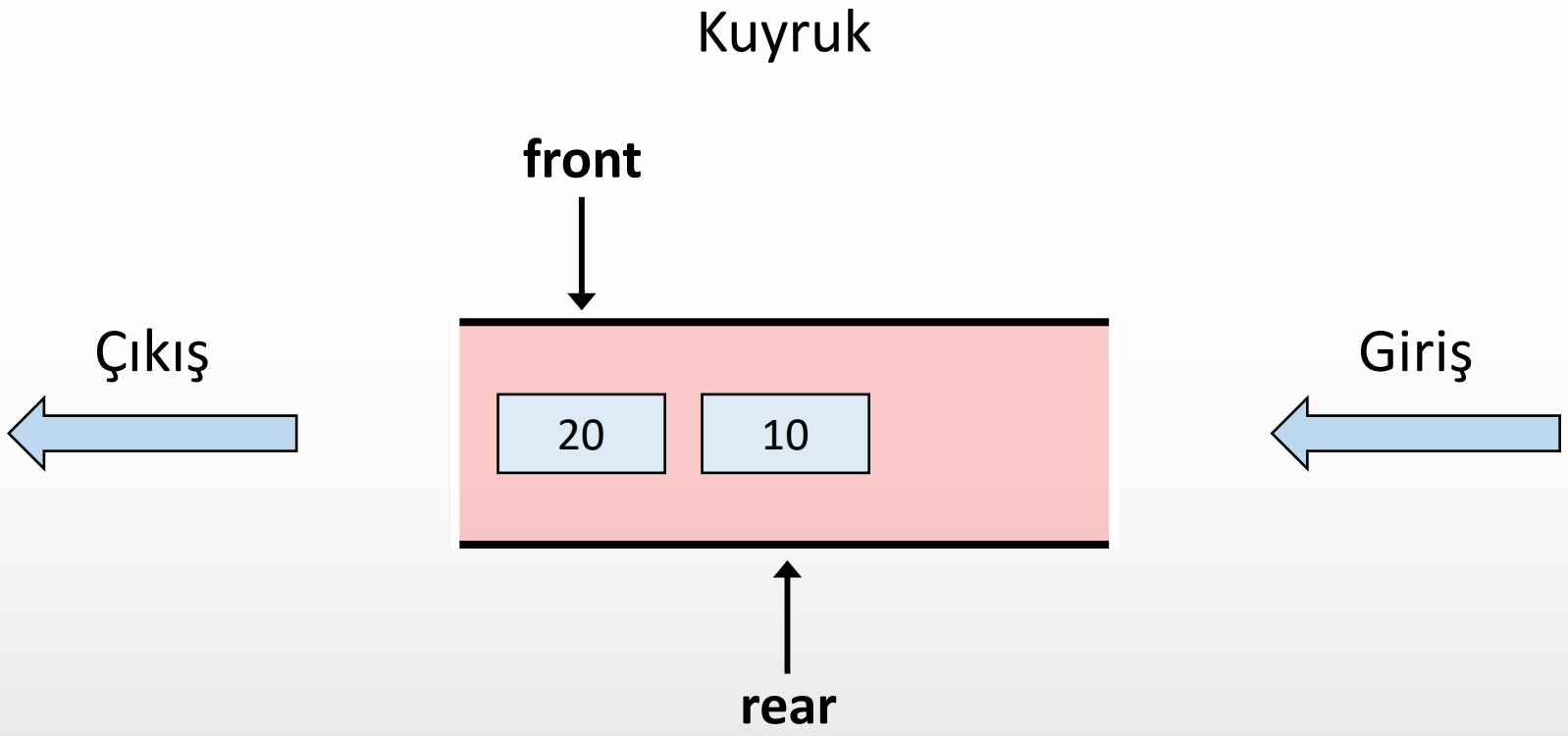
enqueue(10)



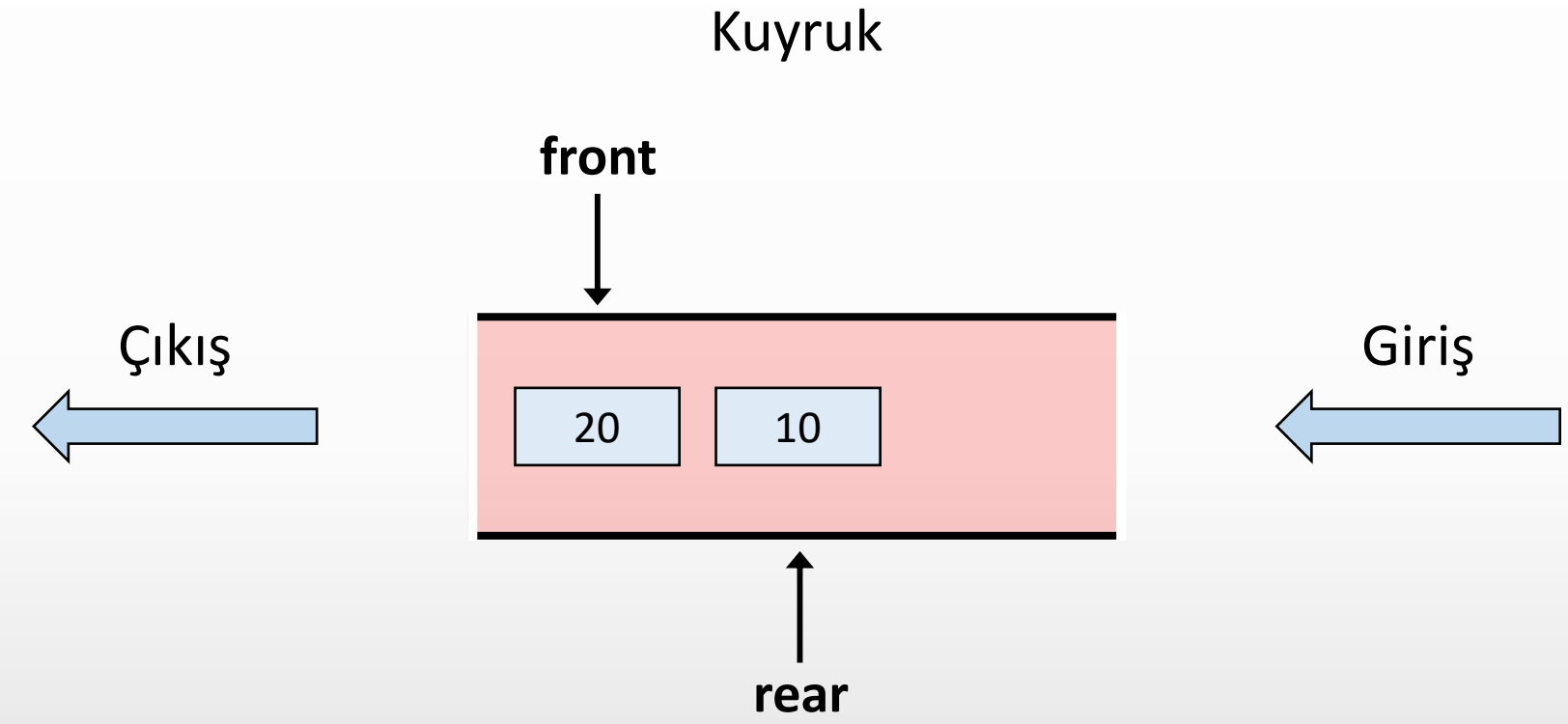
enqueue(10)



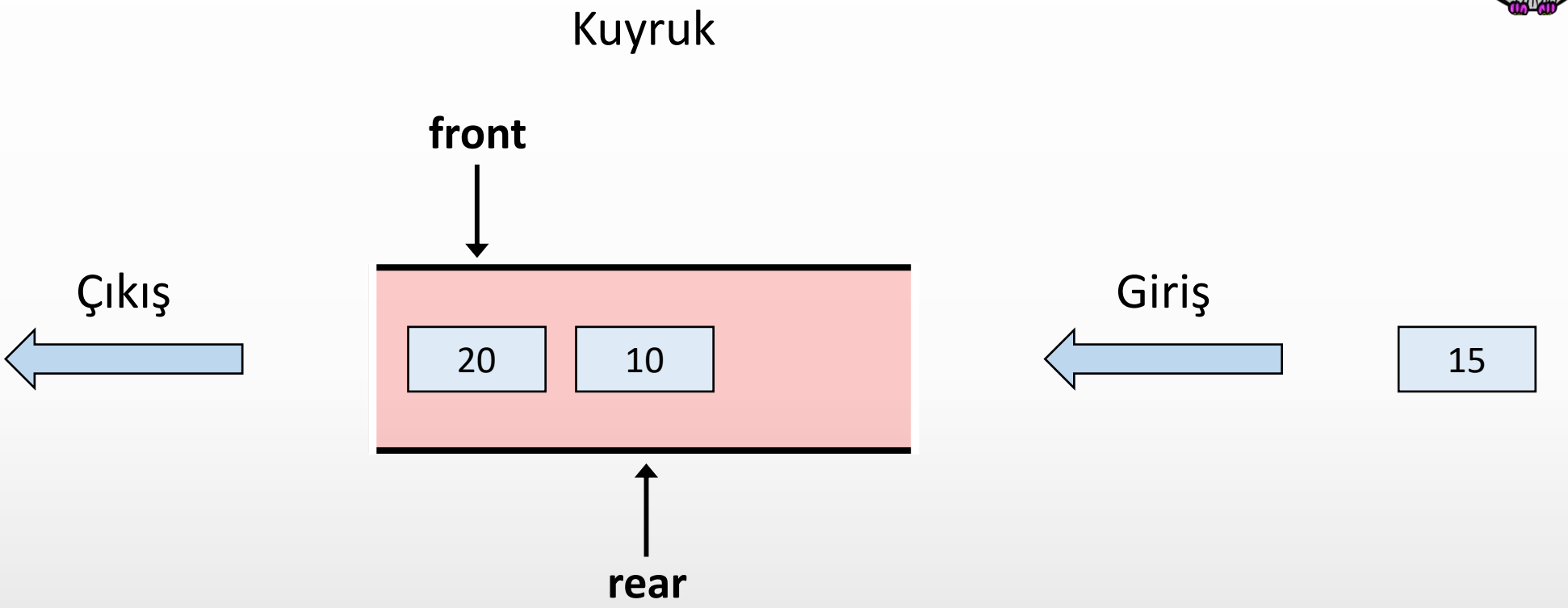
enqueue(10)



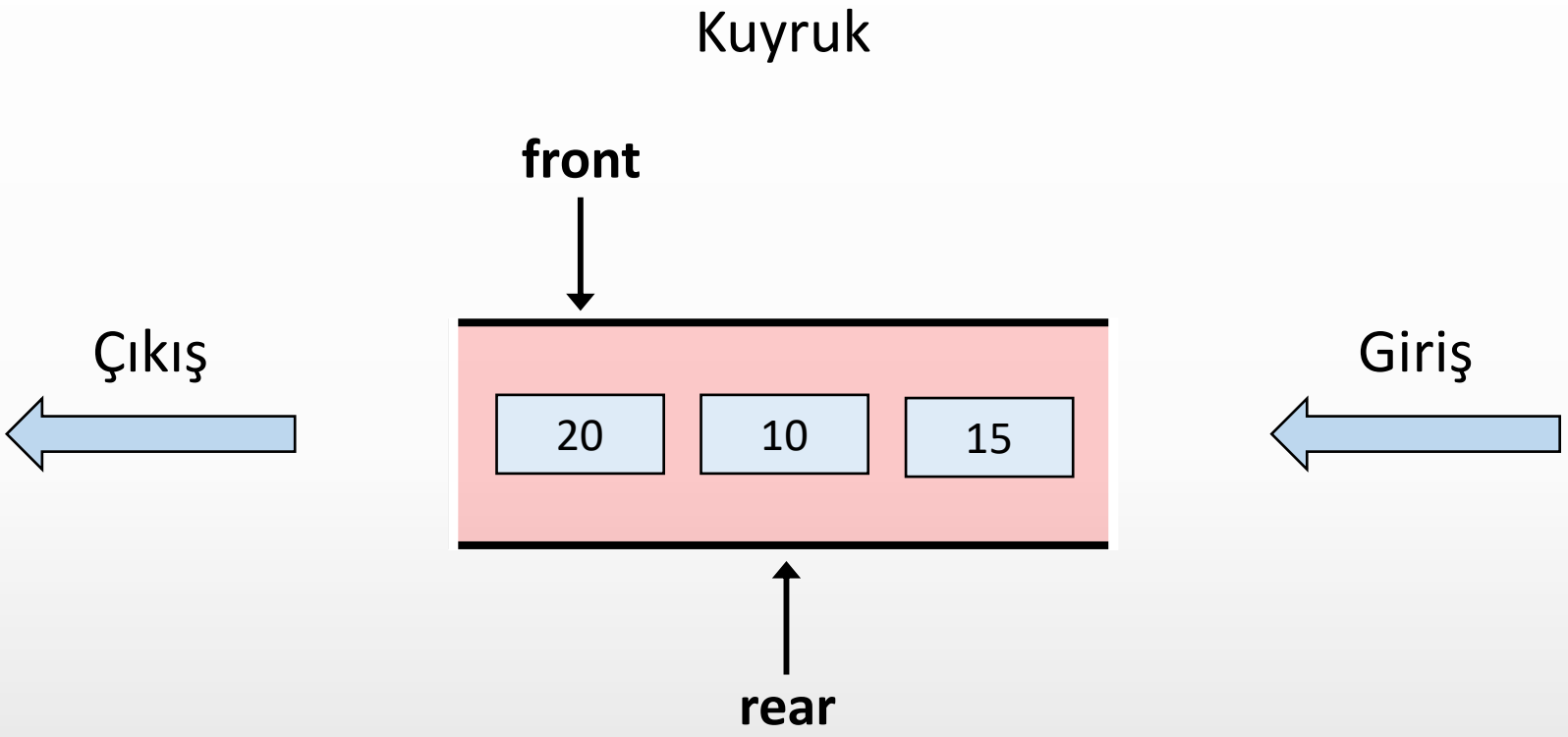
enqueue(15)



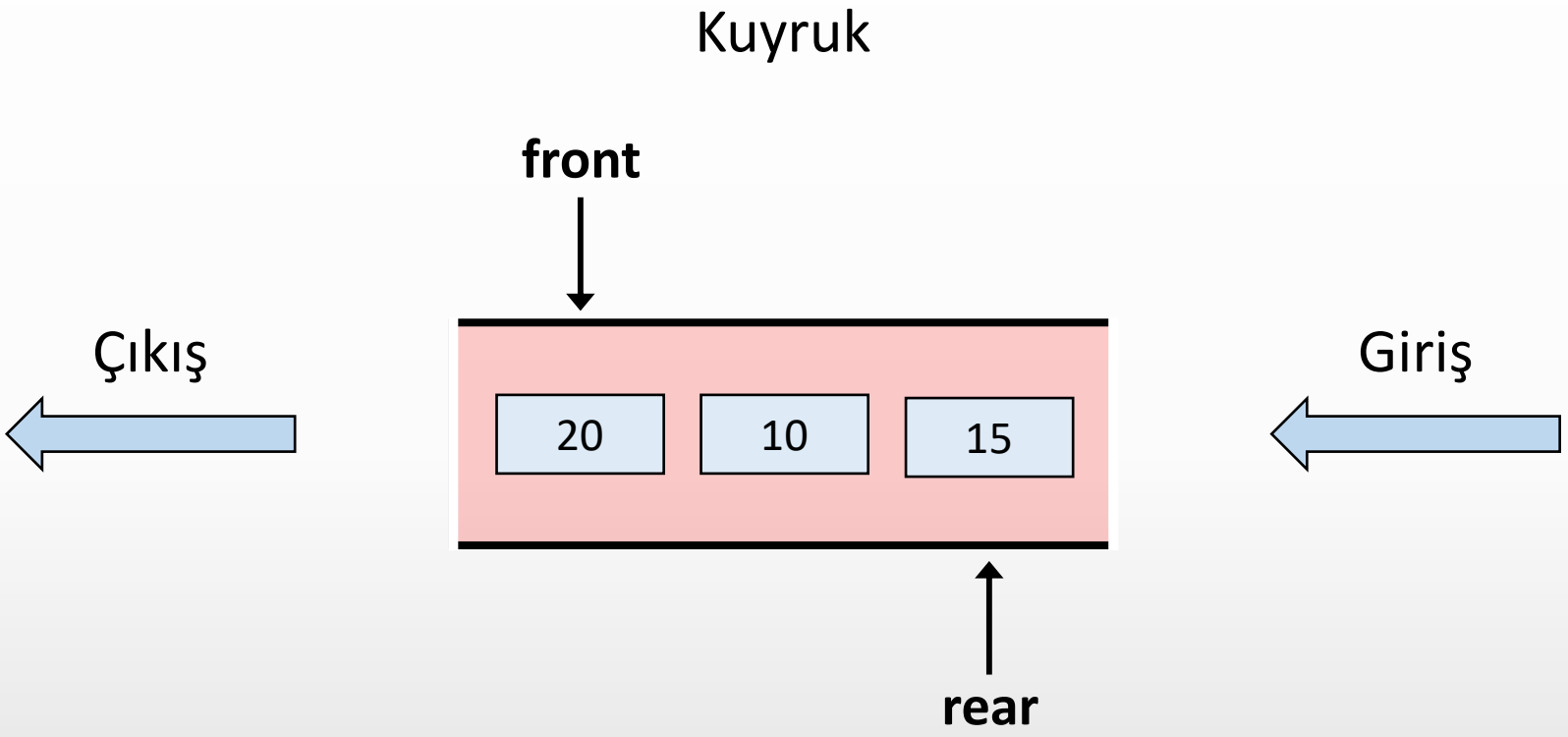
enqueue(15)



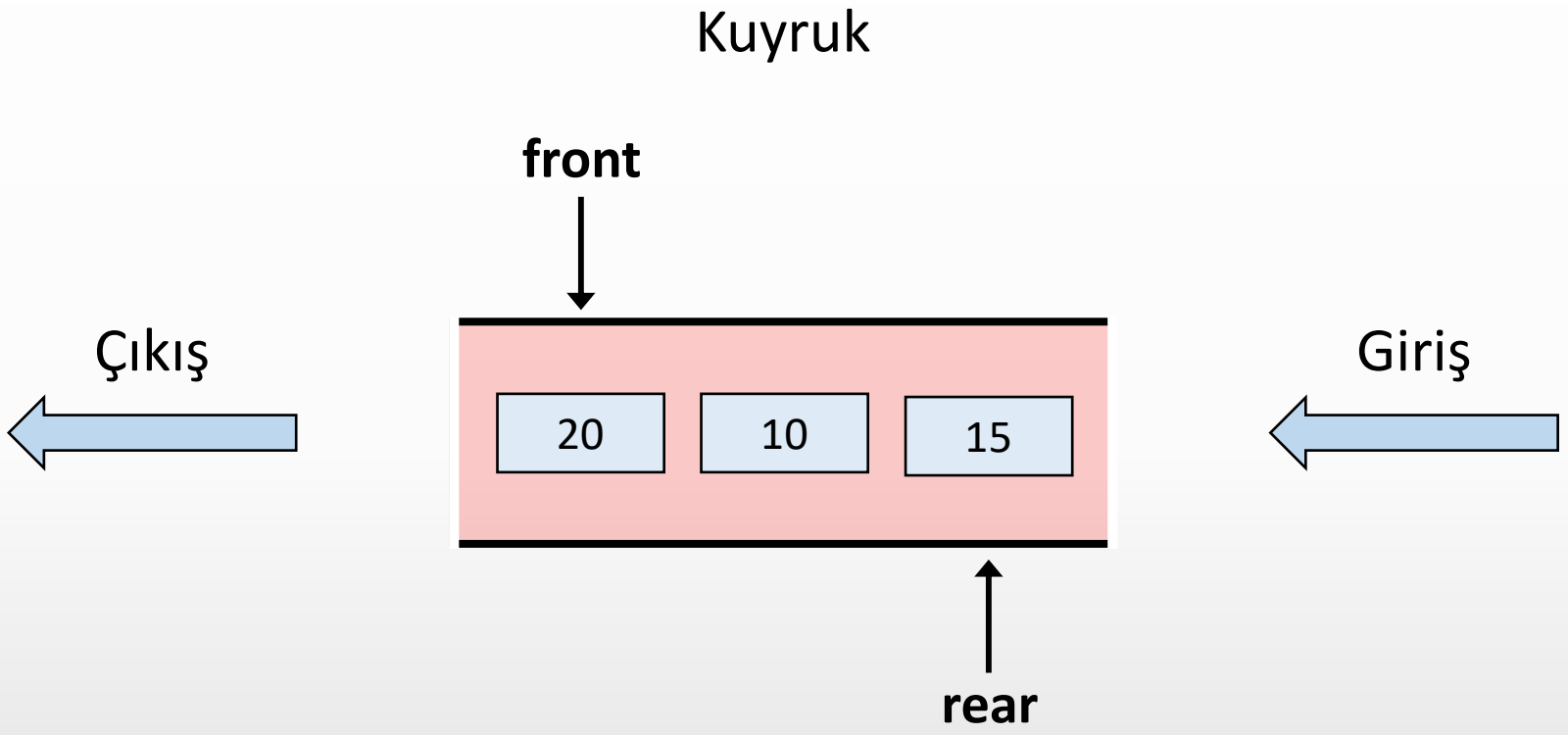
enqueue(15)



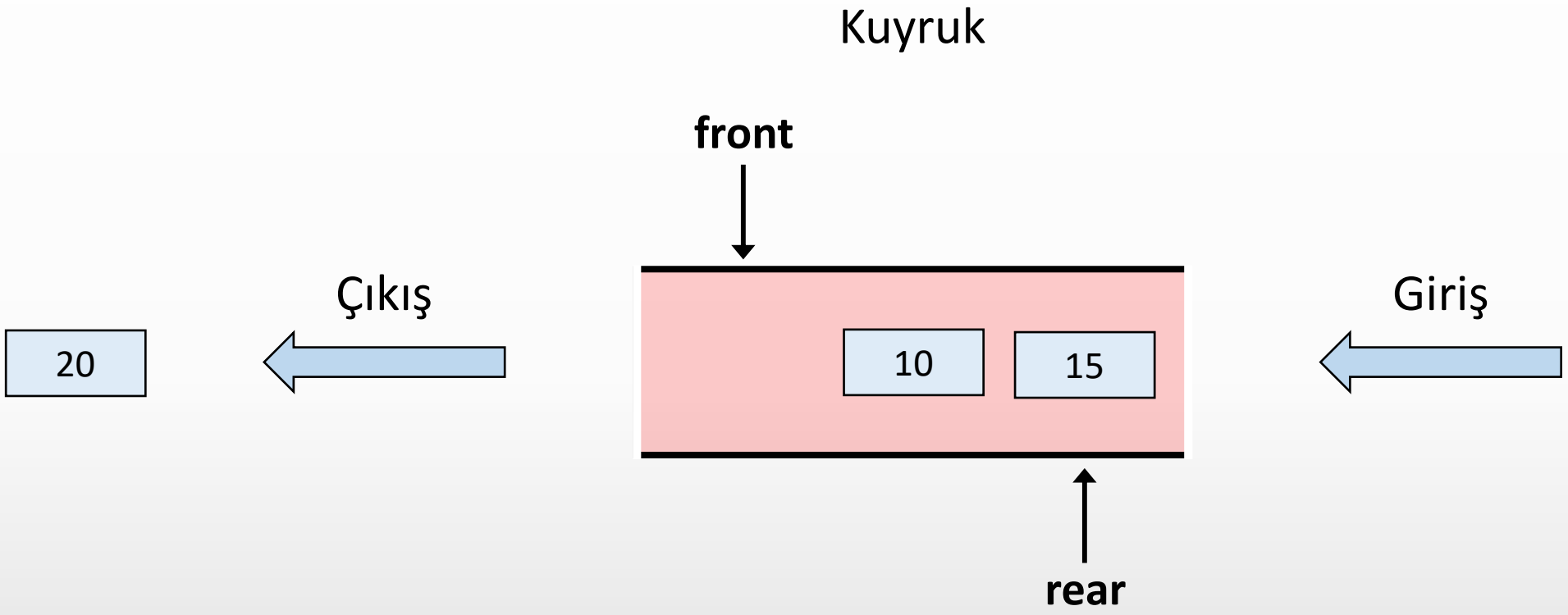
enqueue(15)



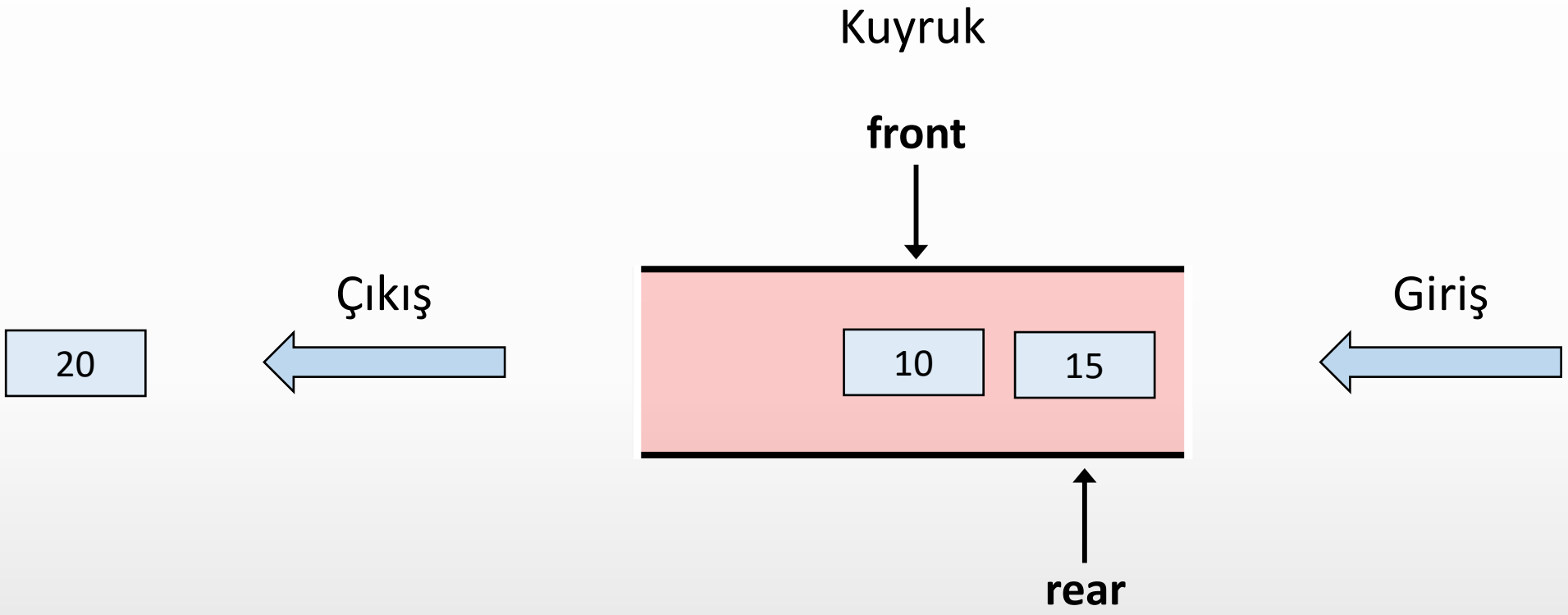
dequeue()

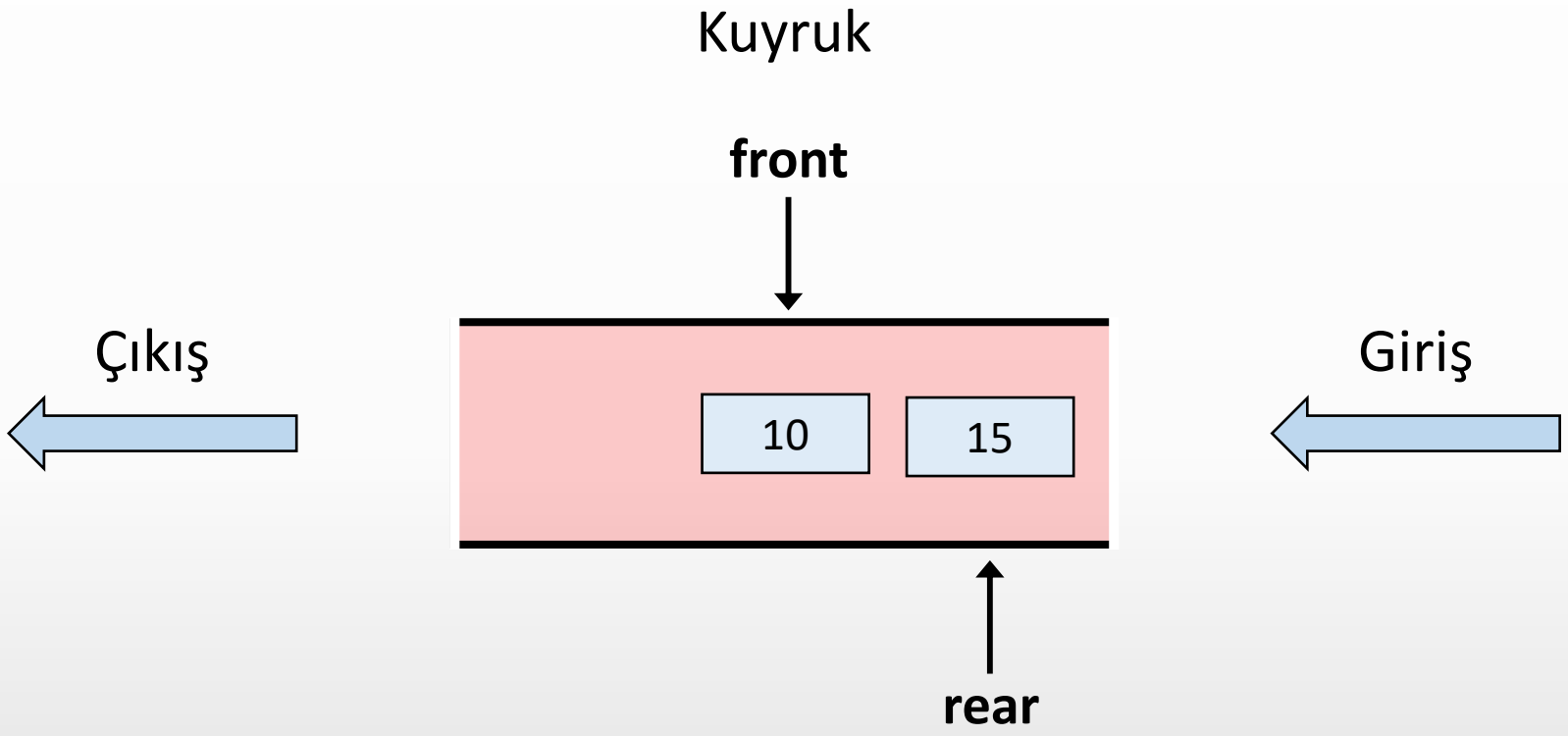


dequeue()

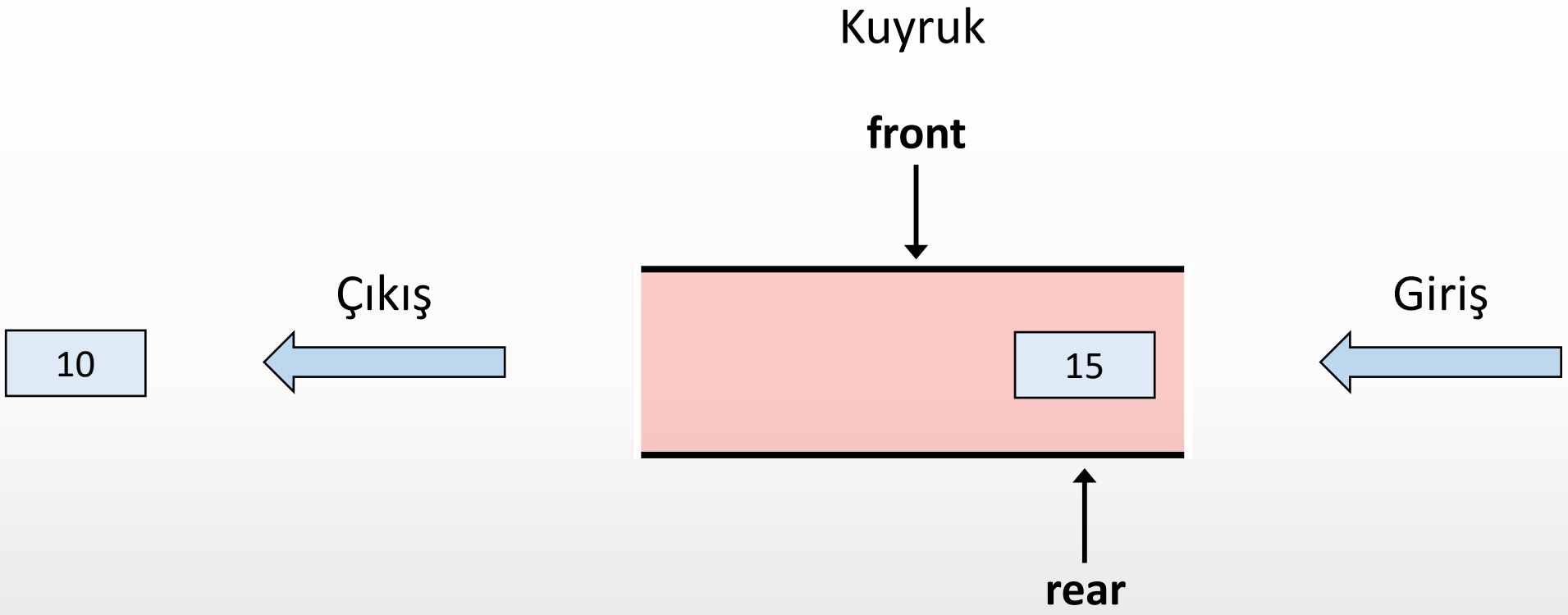


dequeue()

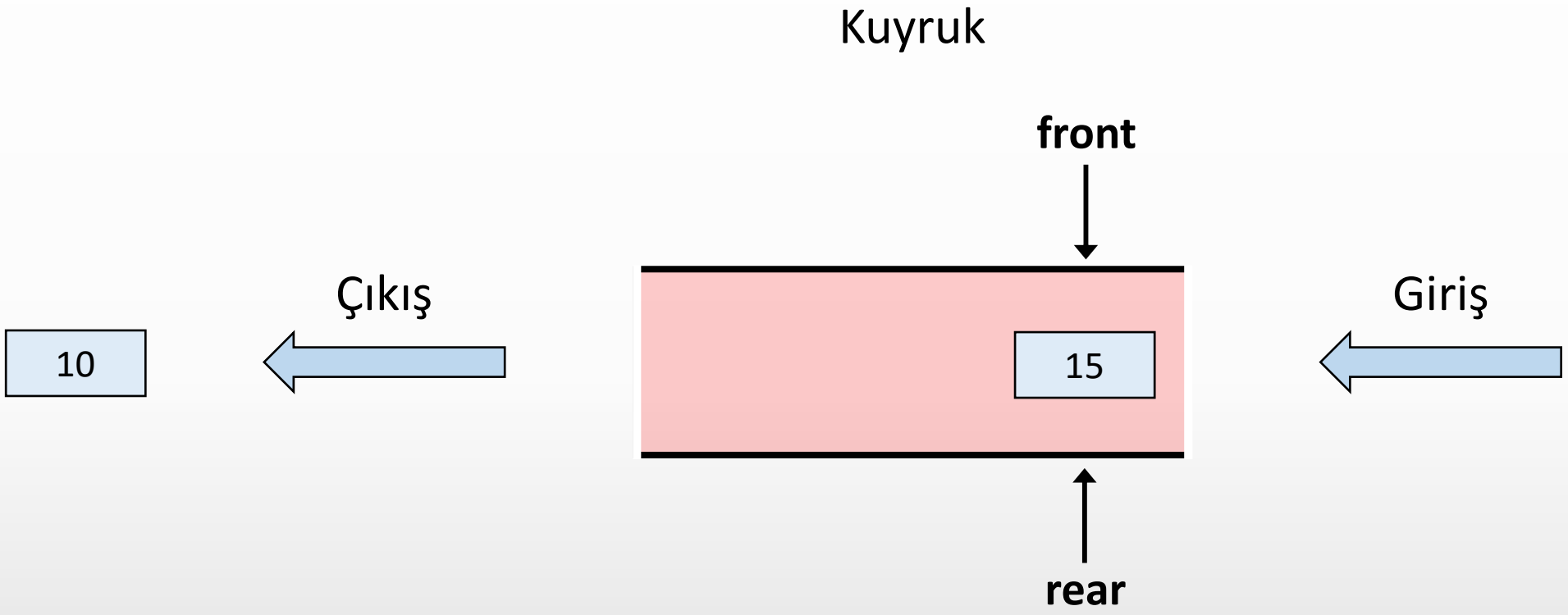


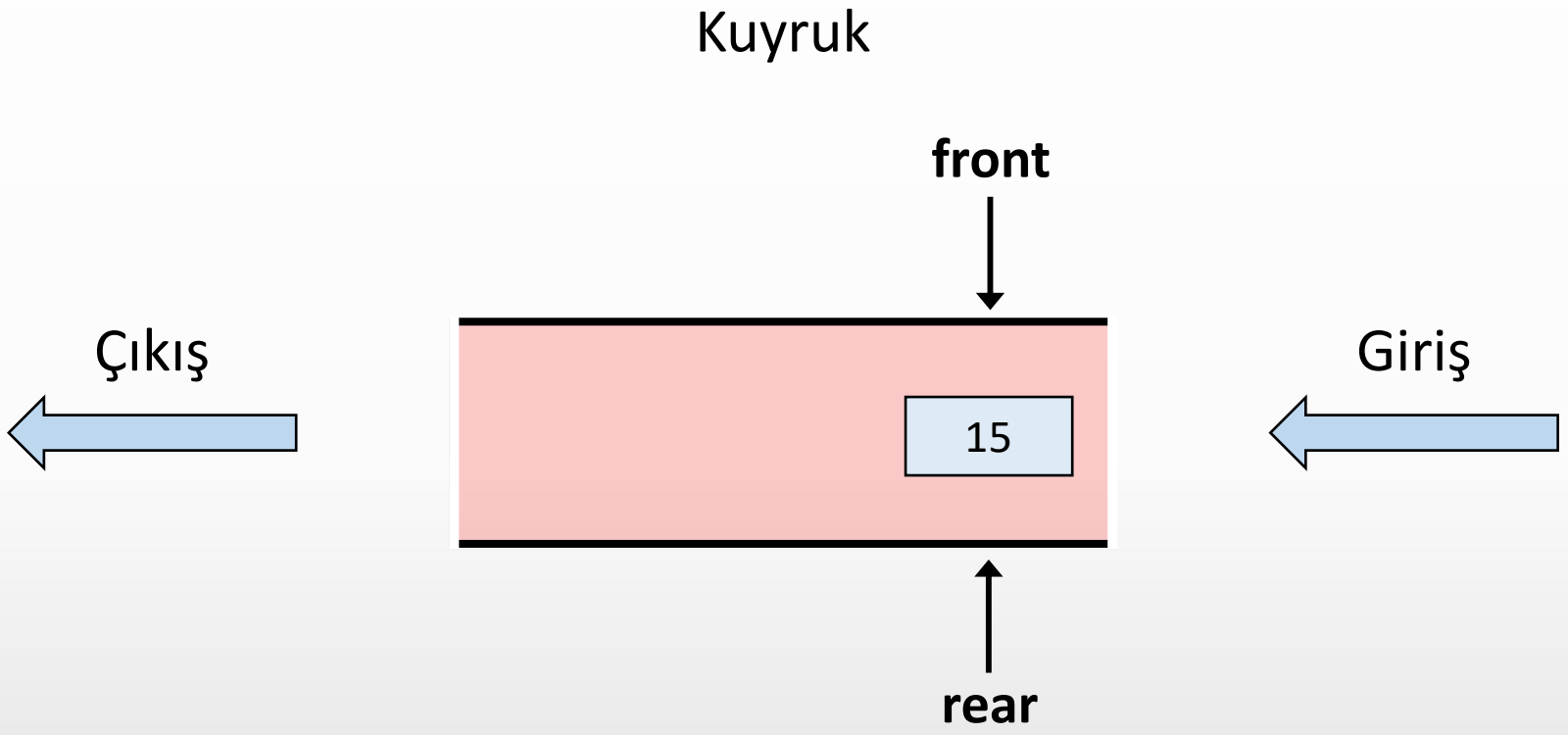

Kuyruk
front
Çıkış
10
15
Giriş
rear

dequeue()

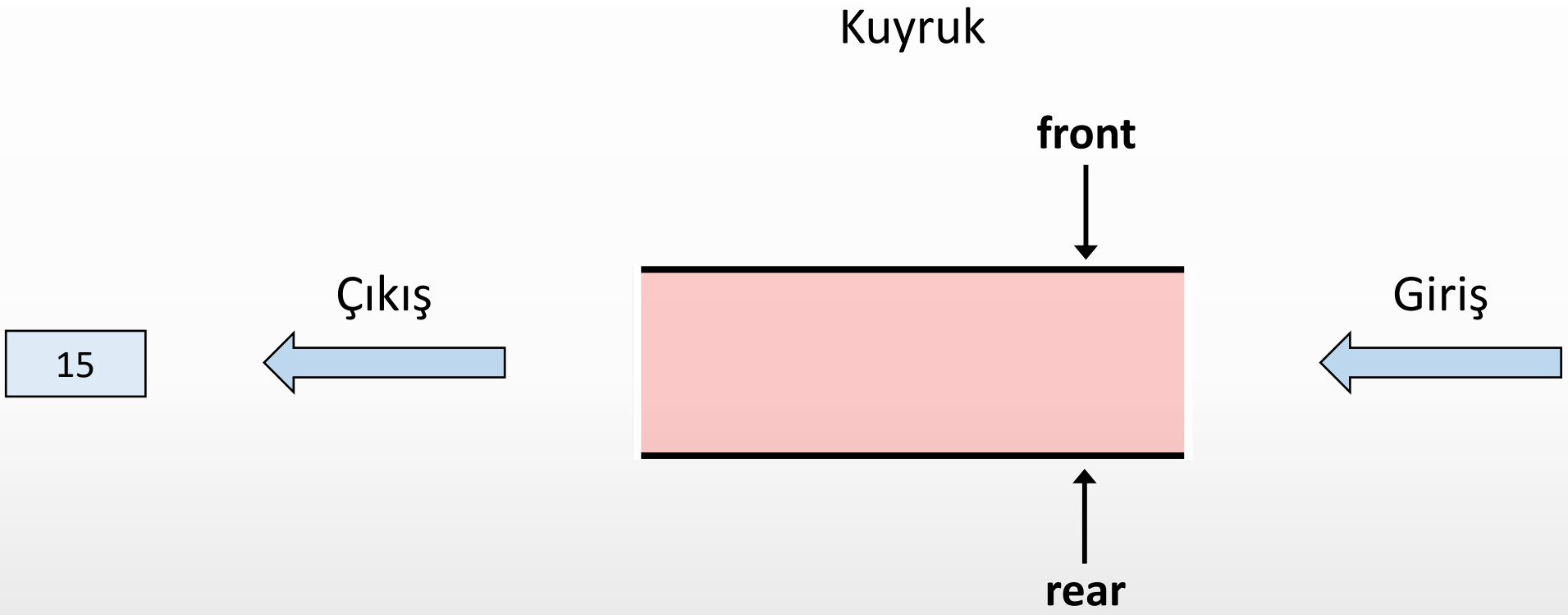


dequeue()




Kuyruk
front
Çıkış
15
Giriş
rear

dequeue()



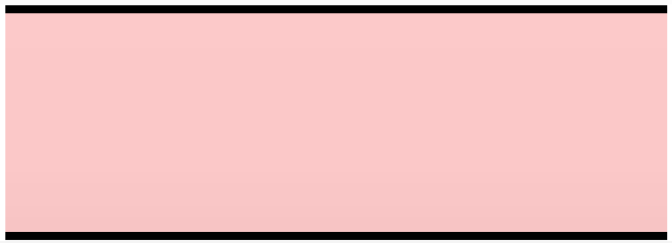
dequeue()

front ⟶ null

rear ⟶ null

Kuyruk

Çıkış



Giriş

front ⟶ **null**

rear ⟶ **null**

Hızlı ve Esnek

- Verilerin eklenmesi (Enqueue) ve çıkarılması (Dequeue) işlemleri genellikle sabit bir zaman karmaşıklığına sahiptir (O(1)).
- Bu, kuyrukların hızlı ve etkili bir şekilde kullanılmasını sağlar.
- Kuyruk veri yapısı, çoklu veri işleme için idealdir.
- Hem başından hem de sonundan veriye erişim sağlar.
- Hızlı ve esnek bir veri yapısıdır, işlem yapma açısından etkili bir seçenektir.

Dizi Temsili

- Kuyruk, dizi kullanılarak temsil edilebilir.
  - Kuyruk: Kuyruk öğelerini depolayan 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 bir boyuta sahiptir.

Bağlı Liste Temsili

- Kuyruk aynı zamanda bağlı listeler kullanılarak temsil edilebilir.
- Kuyruğu temsil etmek için aşağıdaki yapılar ve gösterge (pointer) kullanılır:
  - Bağlı Listeler (Linked-lists)
  - Gösterge (Pointer)
  - Yapılar (Structures)
- Bağlı liste temsilinde, her kuyruk öğesi bir bağlı liste düğümüdür.
- Bağlı listeler öğeleri dinamik olarak depolamaya izin verir, bu nedenle boyutları değiştirilebilir.
- Kuyruk işlemleri bu bağlı listeler üzerinde gerçekleştirilir.
- Dizi temsiline göre biraz 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)**
  - Bu, basit bir kuyruktur.
  - Bu tür bir kuyrukta, giriş sadece bir uçtan yapılabilir, ancak silme işlemi her iki uçtan da gerçekleştirilebilir.
- 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)**
  - Bu da basit bir kuyruktur.
  - Bu tür bir kuyrukta, giriş her iki uçtan yapılabilir, ancak silme işlemi sadece bir uçtan gerçekleştirilebilir.
- 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)**
  - Bu, özel bir kuyruk türüdür.
  - Son pozisyonun ilk pozisyona bağlandığı bir döngü oluşturur.
  - İşlemler yine 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)**
  - Hem ekleme hem de silme işlemlerinin her iki uçtan da gerçekleştirilebildiği özel bir kuyruktur.
- 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 kuyruktur.
  - Öğelerin önceliği, erişim sırasını belirler.

Kuyrukta Temel İşlemler

- Enqueue(), kuyruğun sonuna yeni bir öğe ekler.
- Dequeue(), kuyruğun başındaki öğeyi kuyruktan çıkarır (ilk giren öğeyi alır).
- Peek() veya front(), kuyruğun başındaki öğeyi alır, ancak onu kuyruktan silmez.
- Rear(), kuyruğun sonundaki öğeyi alır, ancak onu kuyruktan silmez.
- isFull(), kuyruğun dolu olup olmadığını doğrular.
- isNull(), kuyruğun boş olup olmadığını doğrular.

Enqueue()

- İlk adımda, kuyruğun dolu olup olmadığı kontrol edilir.
- Eğer kuyruk doluysa, taşma (overflow) hatası döndürülür ve işlem sonlandırılır.
- Eğer kuyruk dolu değilse, arka gösterge (rear pointer) bir sonraki boş alana işaret etmek için artırılır.
- Arka gösterge (rear pointer) tarafından işaret edilen konuma veri öğesi eklenir.
- İşlem başarıyla tamamlandığında "başarı" döndürülür.

Enqueue İşlemi

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

Enqueue İşlemi

front ⟶ null

rear ⟶ null

uzunluk = 0

```
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

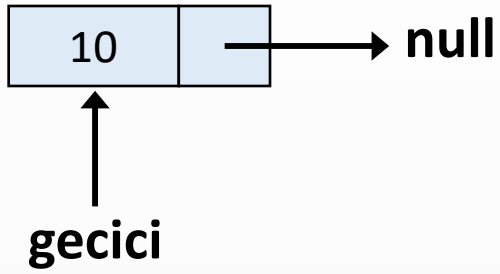
rear ⟶ null

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**

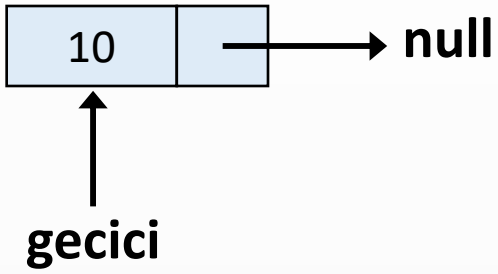
rear ⟶ **null**

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**

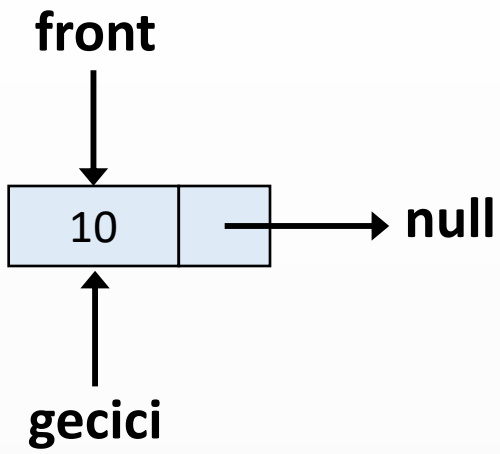
rear ⟶ **null**

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++;
}
```



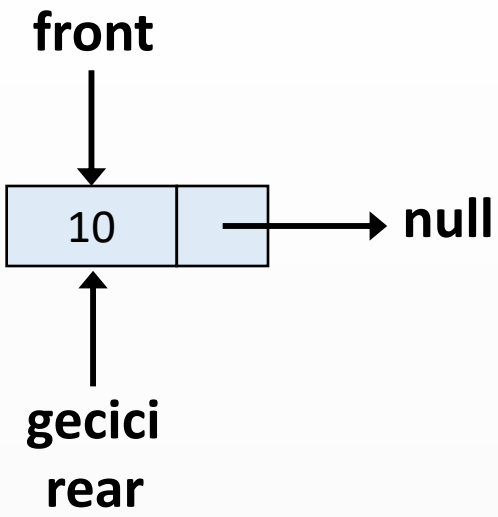
rear ⟶ null

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++;
}
```

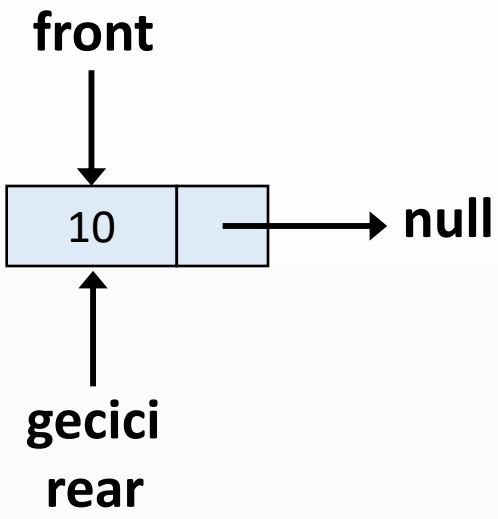


uzunluk = 0

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++;
}
```

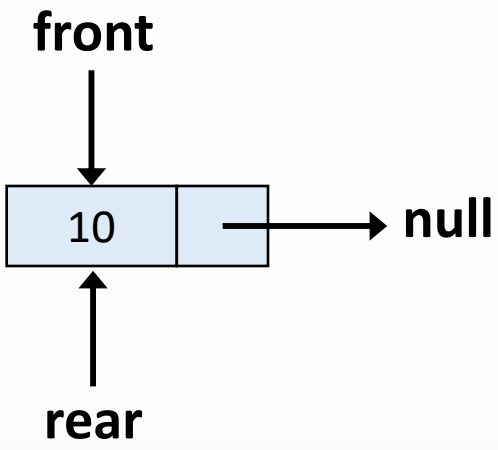


uzunluk = 1

veri = 10

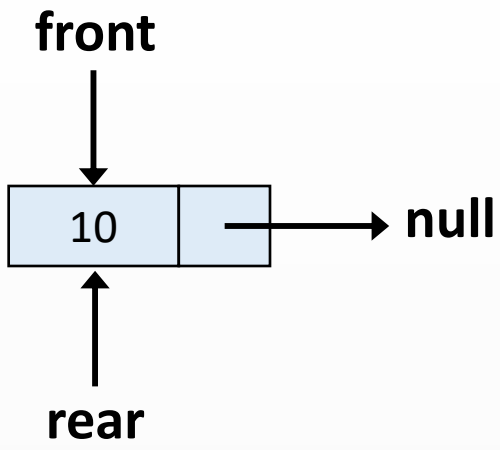
enqueue(10)


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



uzunluk = 1

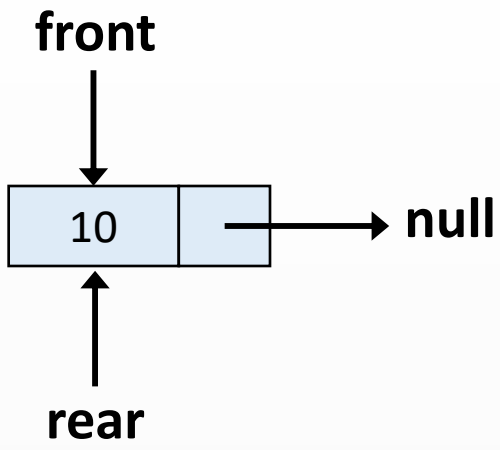
```
public void enqueue(int veri) {
  Dugum gecici = new Dugum(veri);
  if(bosMu()) {
    front = gecici;
  }
  else {
    rear.sonraki = gecici;
  }
  rear = gecici;
  uzunluk++;
}
```



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++;
}
```

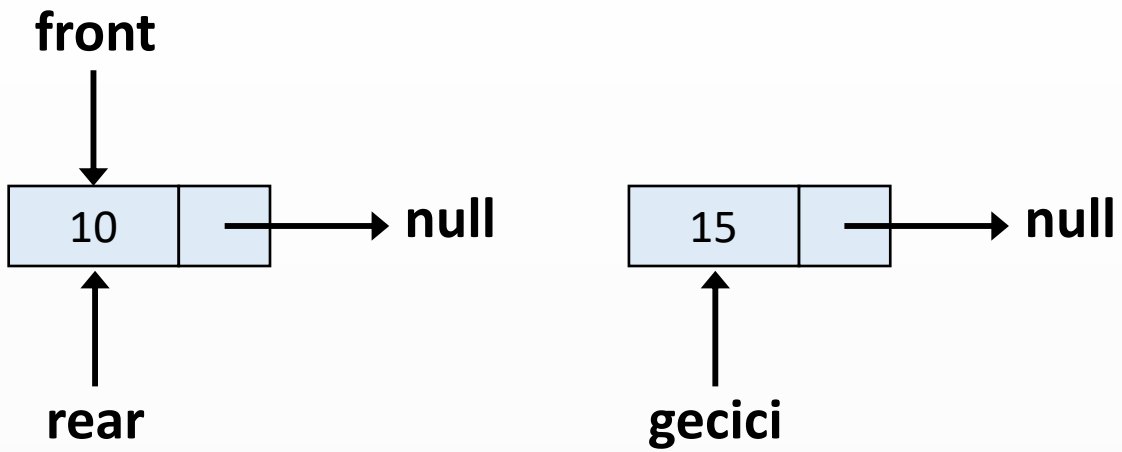


uzunluk = 1

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++;
}
```

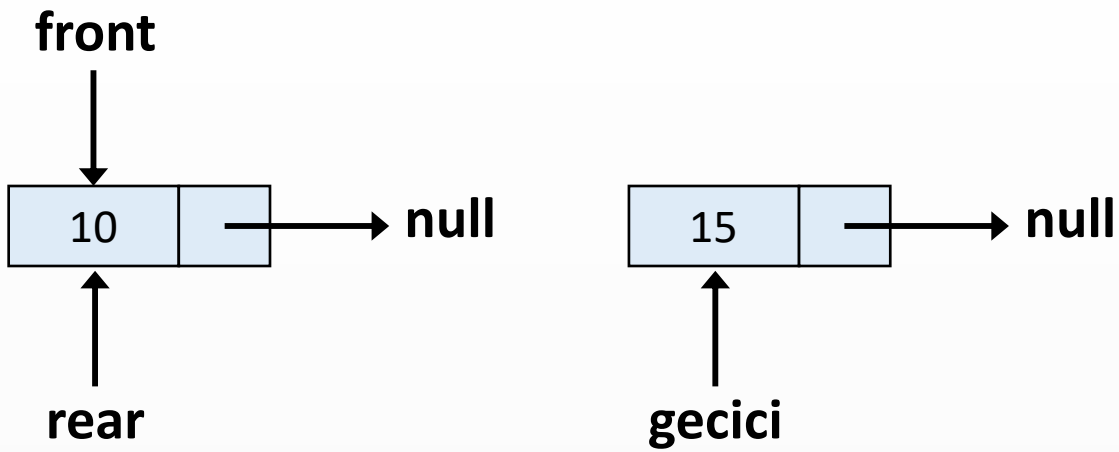


uzunluk = 1

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++;
}
```

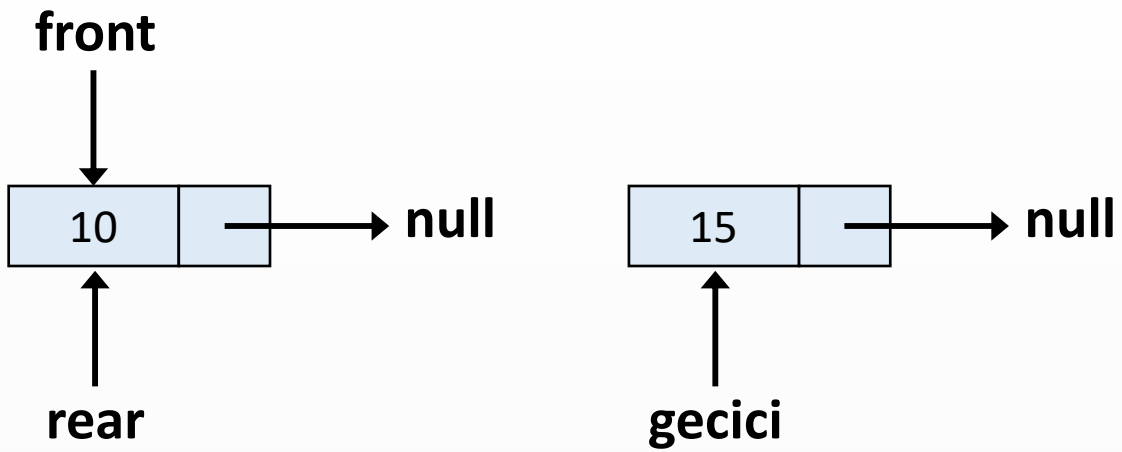


uzunluk = 1

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++;
}
```

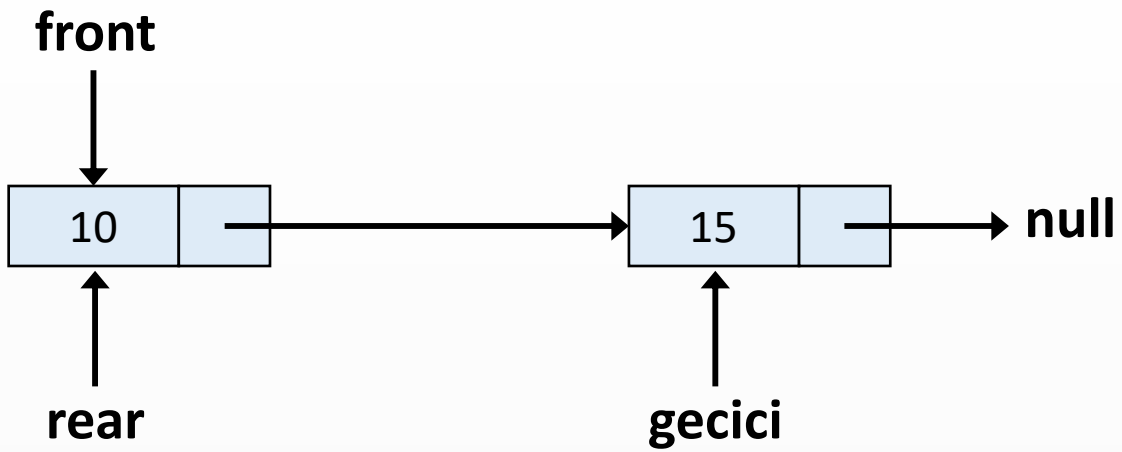


uzunluk = 1

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++;
}
```

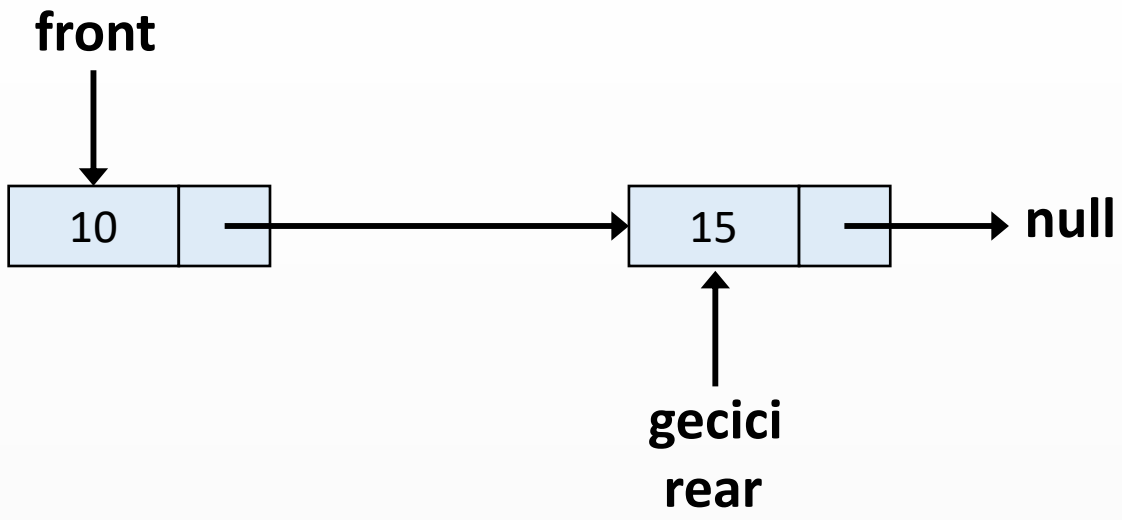


uzunluk = 1

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++;
}
```

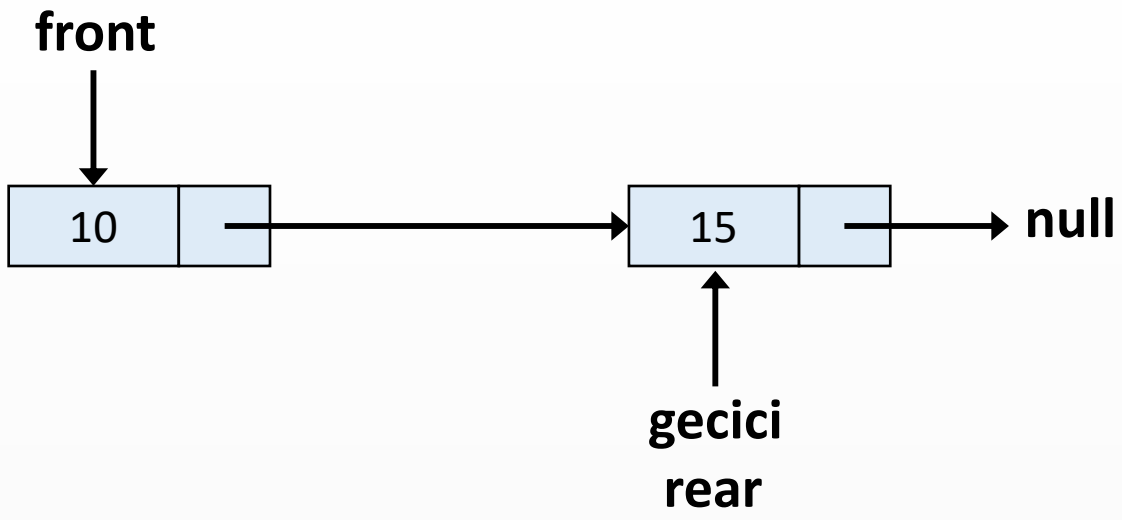


uzunluk = 1

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++;
}
```

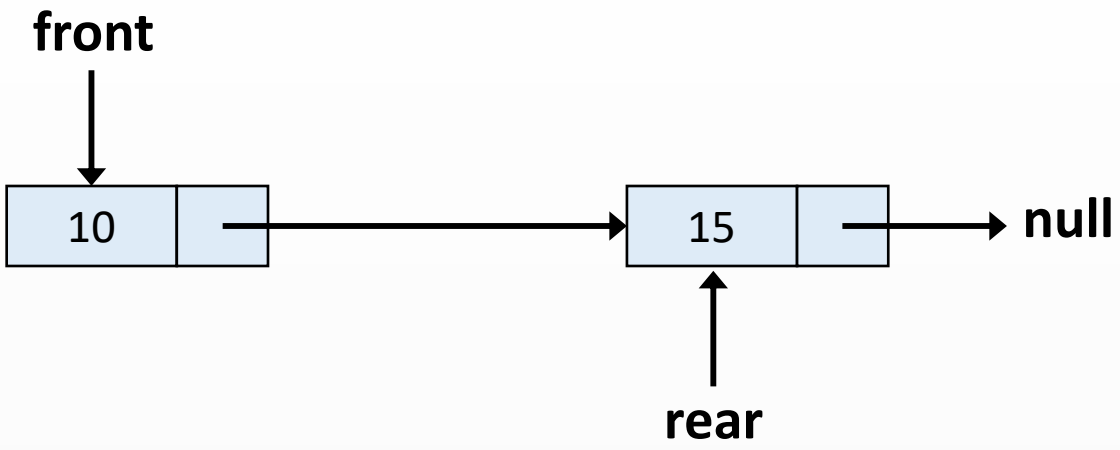


uzunluk = 2

veri = 15

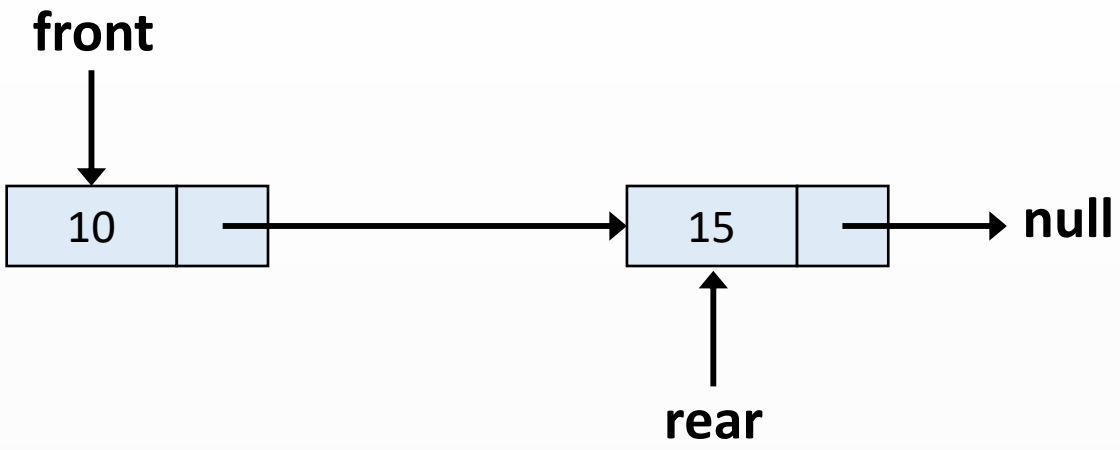
enqueue(15)

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



uzunluk = 2

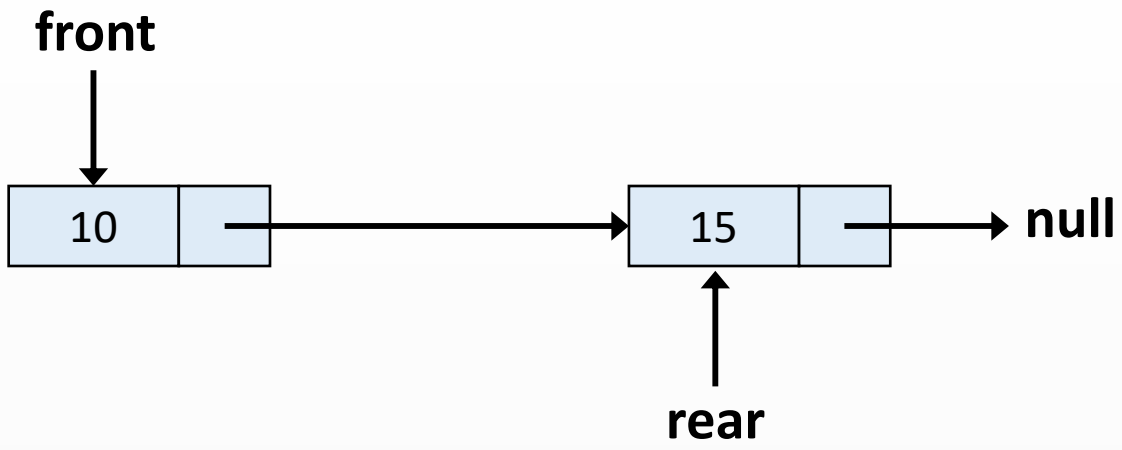
```
public void enqueue(int veri) {
  Dugum gecici = new Dugum(veri);
  if(bosMu()) {
    front = gecici;
  }
  else {
    rear.sonraki = gecici;
  }
  rear = gecici;
  uzunluk++;
}
```



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++;
}
```

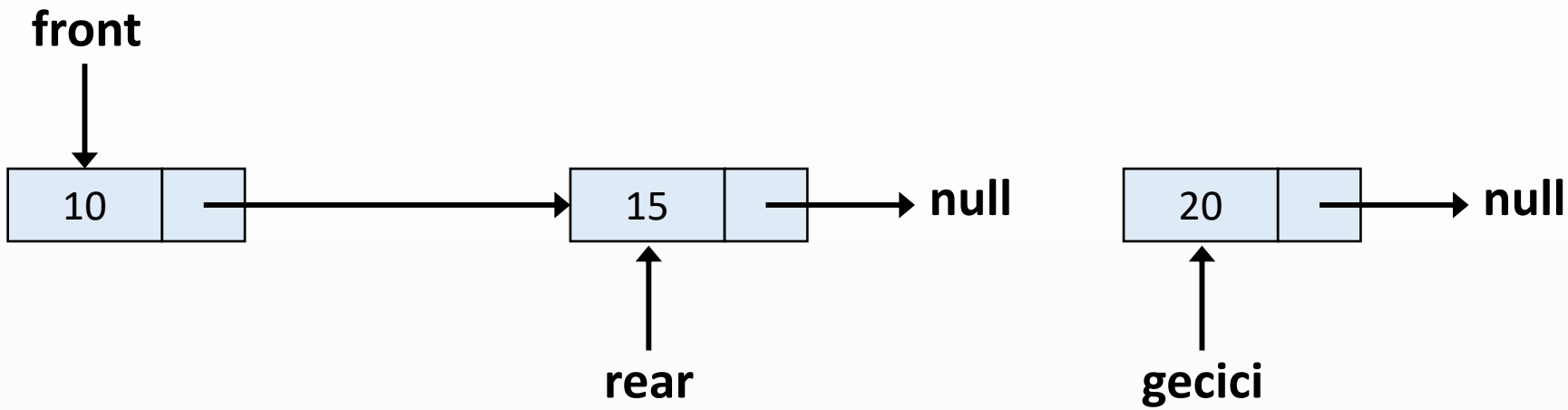


uzunluk = 2

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++;
}
```

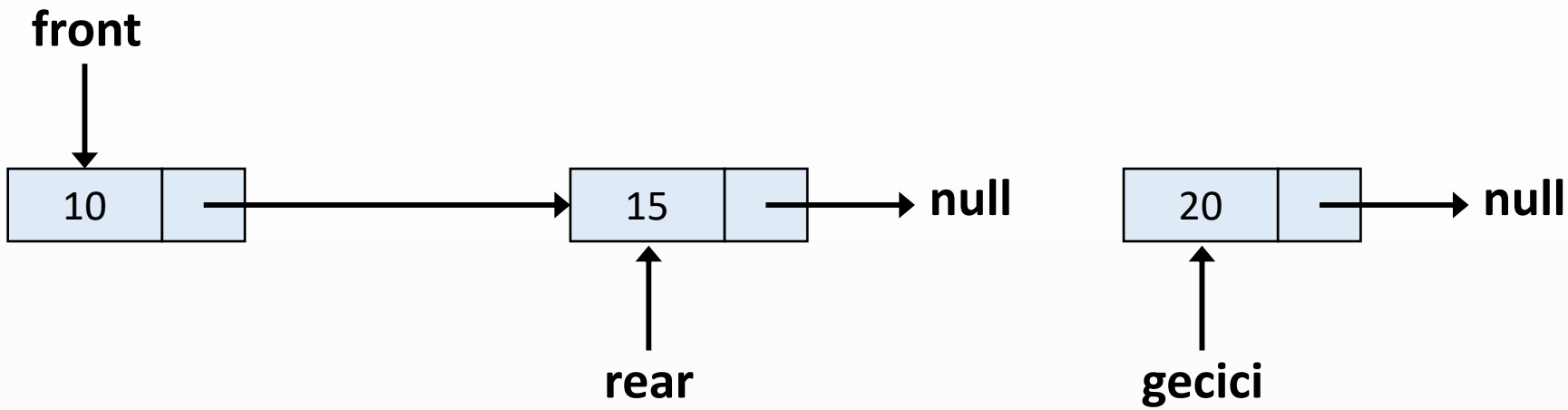


uzunluk = 2

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++;
}
```

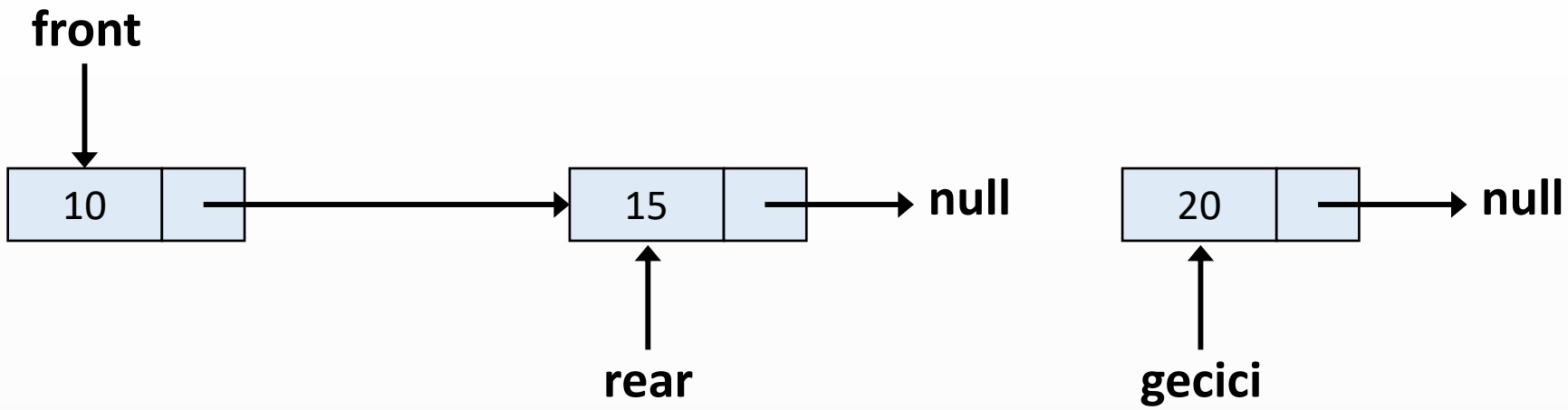


uzunluk = 2

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++;
}
```

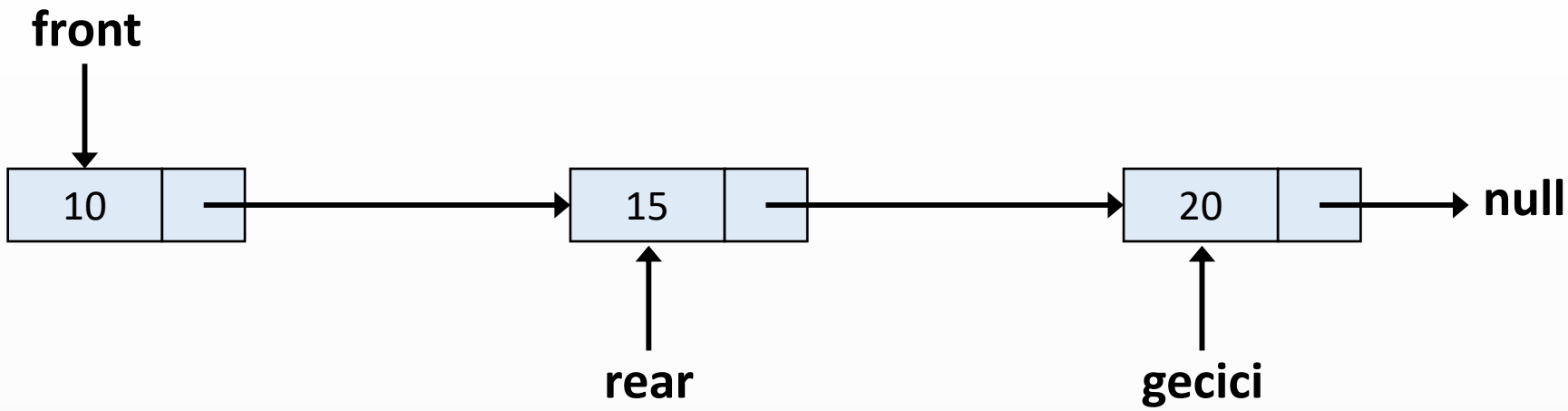


uzunluk = 2

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++;
}
```

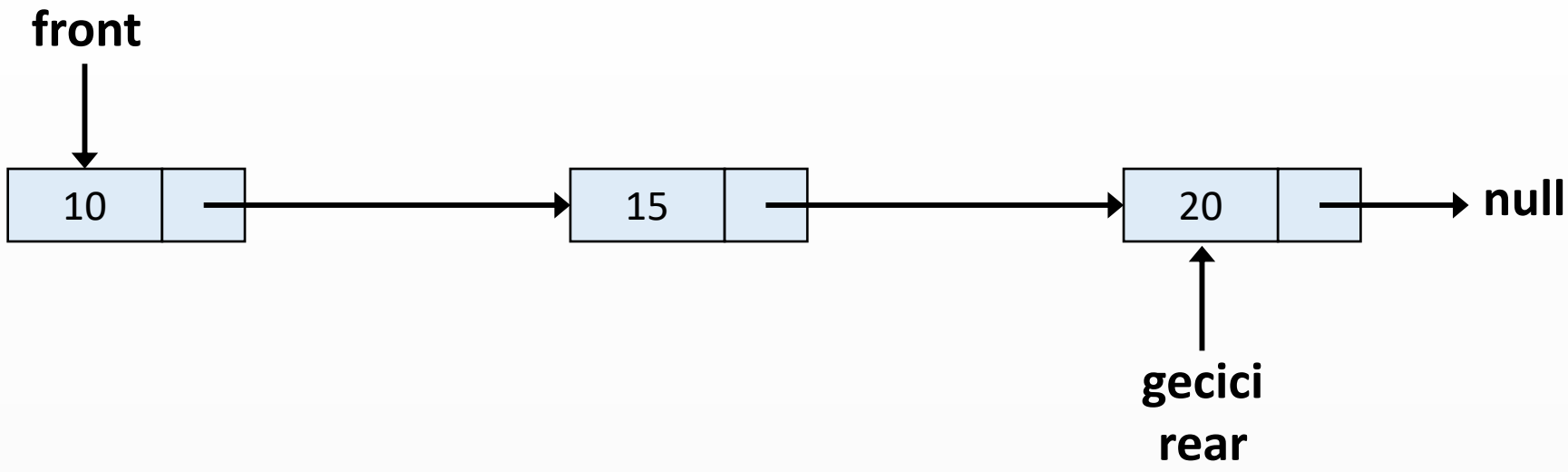


uzunluk = 2

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++;
}
```

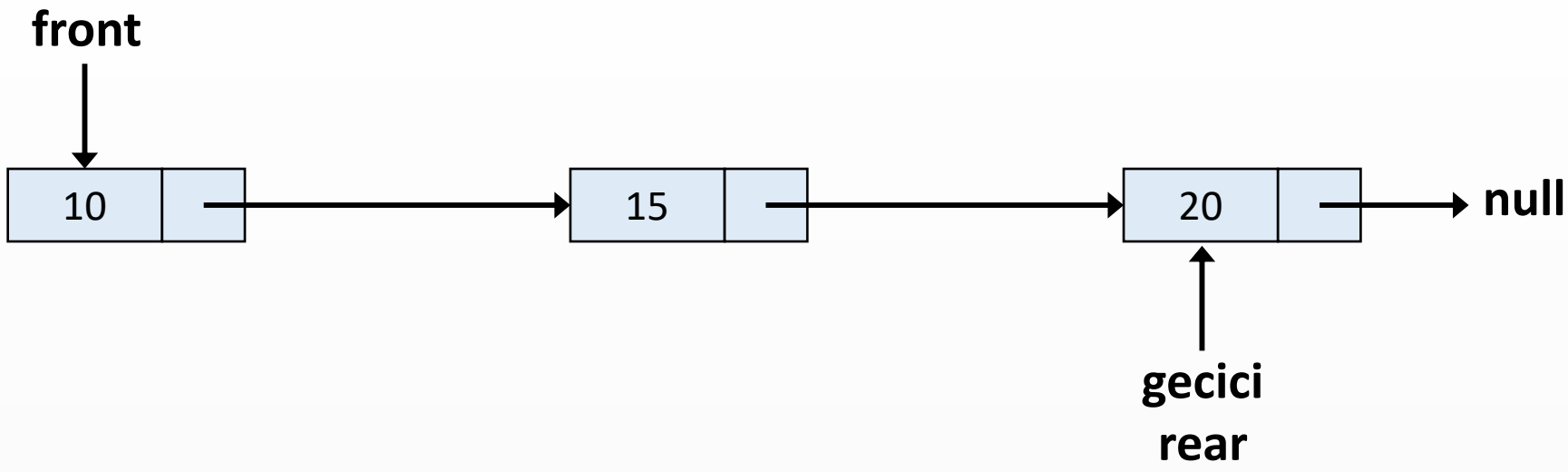


uzunluk = 2

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++;
}
```

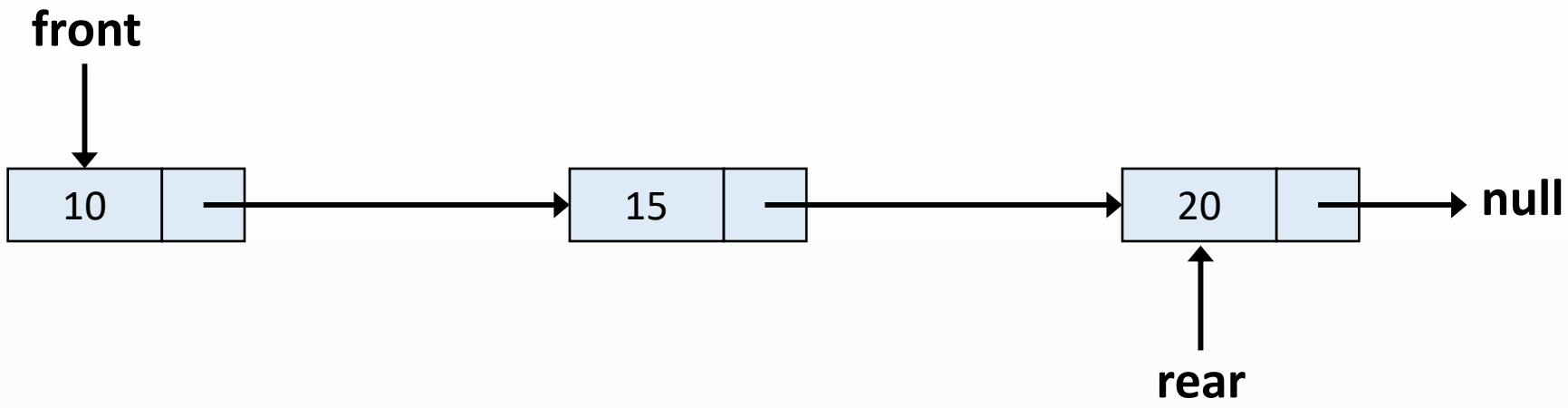


uzunluk = 3

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++;
}
```



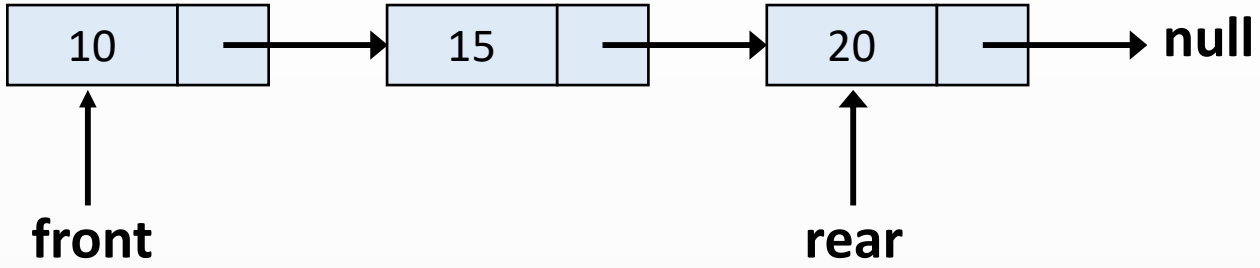
uzunluk = 3

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

Dequeue()

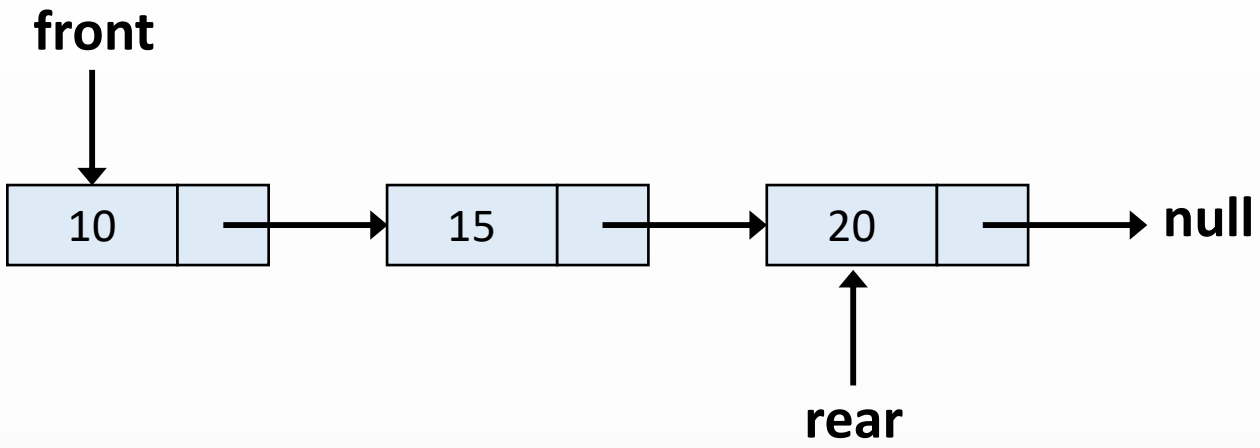
- İlk adımda, kuyruğun boş olup olmadığı kontrol edilir.
- Eğer kuyruk boşsa, alt taşma (underflow) hatası döndürülür ve işlem sonlandırılır.
- Eğer kuyruk boş değilse, ön gösterge (front pointer) tarafından işaret edilen veriye erişilir.
- Veriye erişildikten sonra, ön gösterge (front pointer) bir sonraki kullanılabilir veri öğesine işaret etmek için artırılır.
- İşlem başarıyla tamamlandığında "başarı" döndürülür.

Dequeue İşlemi



uzunluk = 3

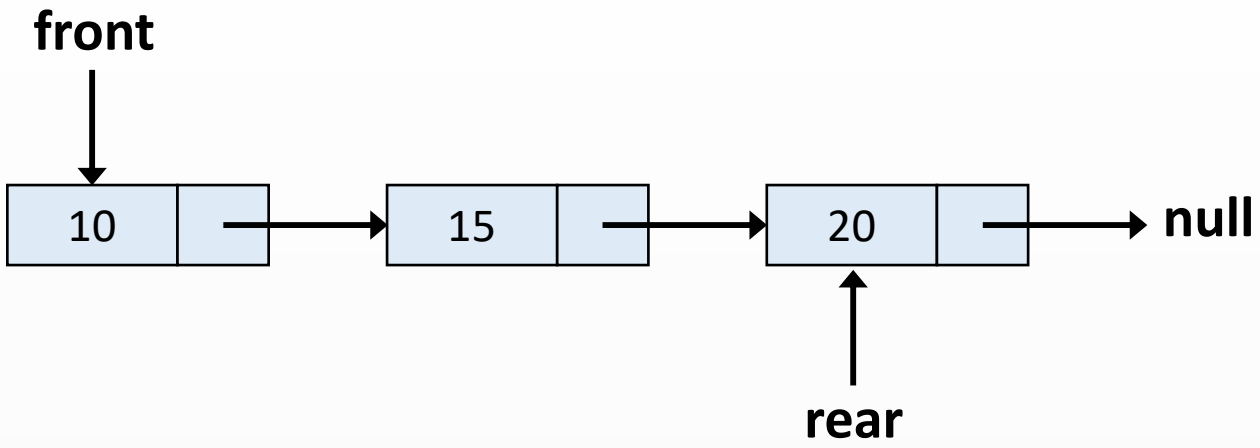
```
public int dequeue() {
  if(bosMu()) {
     throw new NoSuchElementException();
  }
  int sonuc = front.veri;
  front = front.sonraki;
  if(front == null) {
     rear = null;
  }
  uzunluk--;
  return sonuc;
}
```



uzunluk = 3

dequeue()

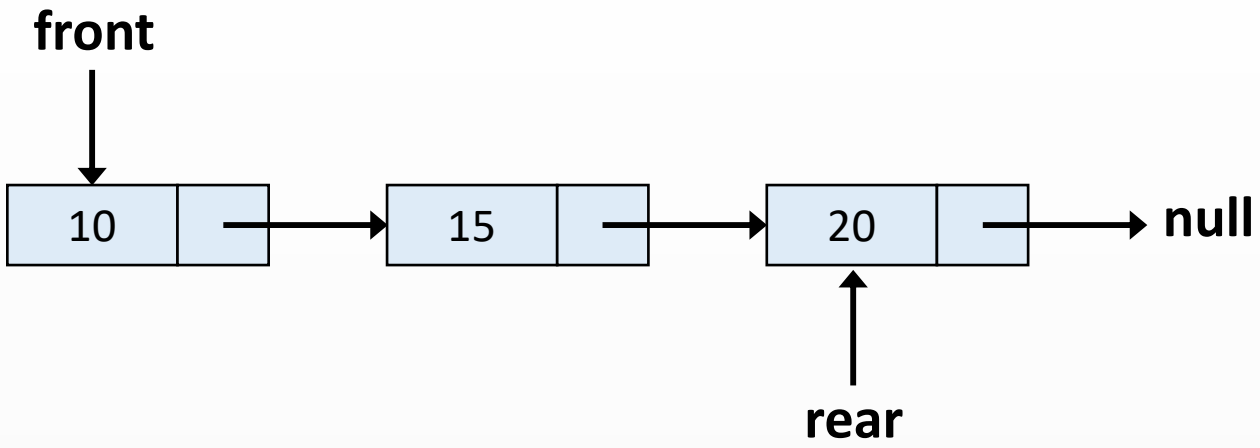
```
public int dequeue() {
  if(bosMu()) {
     throw new NoSuchElementException();
  }
  int sonuc = front.veri;
  front = front.sonraki;
  if(front == null) {
     rear = null;
  }
  uzunluk--;
  return sonuc;
}
```



uzunluk = 3

dequeue()

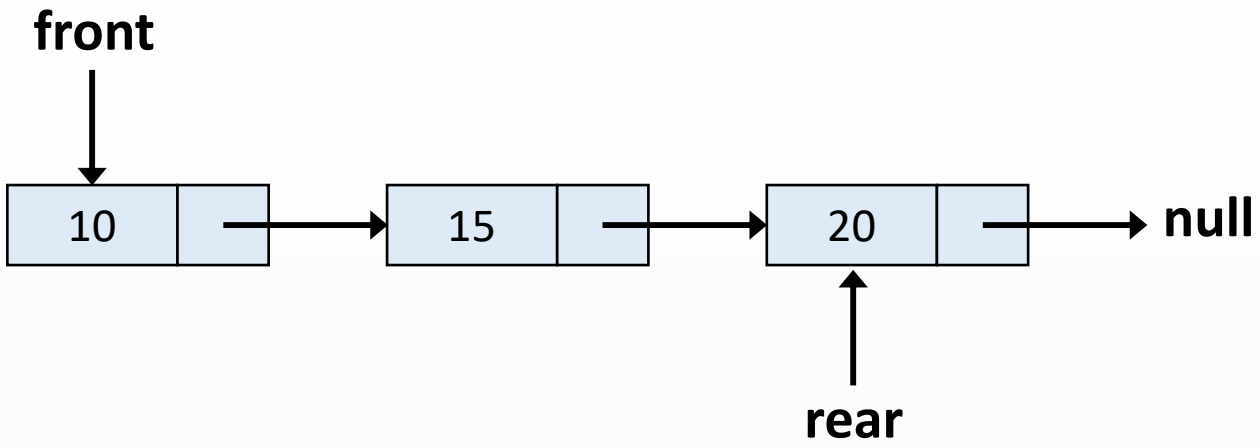
```
public int dequeue() {
  if(bosMu()) {
     throw new NoSuchElementException();
  }
  int sonuc = front.veri;
  front = front.sonraki;
  if(front == null) {
     rear = null;
  }
  uzunluk--;
  return sonuc;
}
```



uzunluk = 3

dequeue()

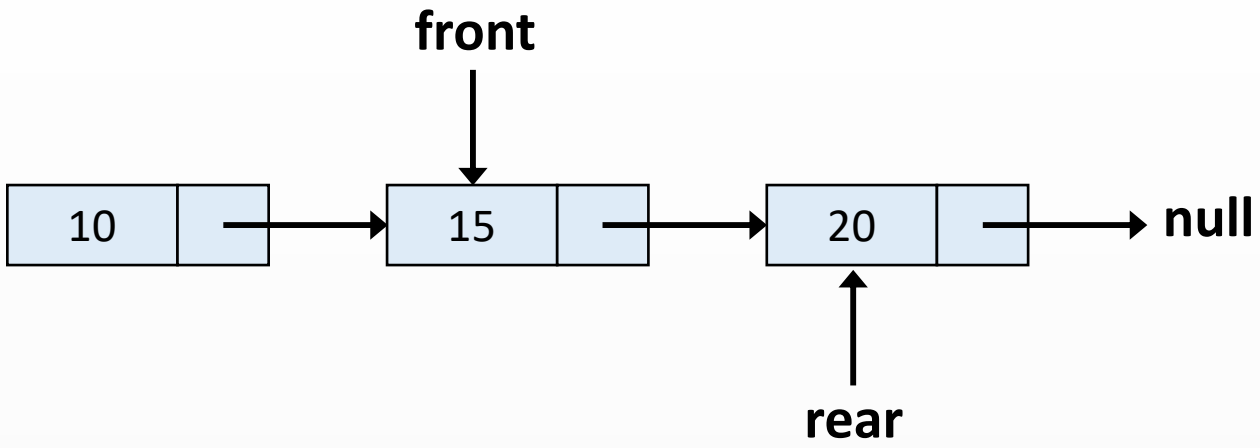
```
public int dequeue() {
  if(bosMu()) {
     throw new NoSuchElementException();
  }
  int sonuc = front.veri;
  front = front.sonraki;
  if(front == null) {
     rear = null;
  }
  uzunluk--;
  return sonuc;
}
```



uzunluk = 3
sonuc = 10

dequeue()

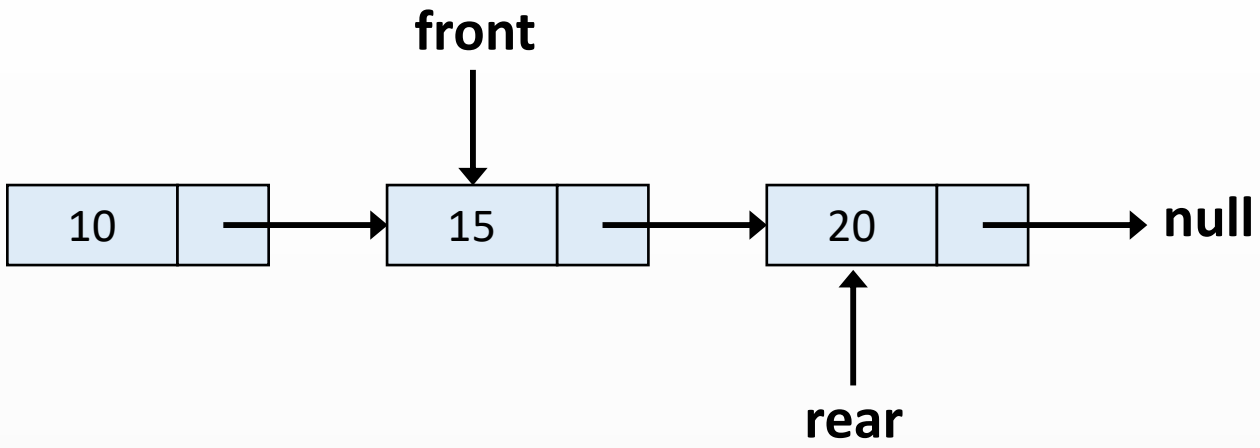
```
public int dequeue() {
  if(bosMu()) {
     throw new NoSuchElementException();
  }
  int sonuc = front.veri;
  front = front.sonraki;
  if(front == null) {
     rear = null;
  }
  uzunluk--;
  return sonuc;
}
```



uzunluk = 3
sonuc = 10

dequeue()

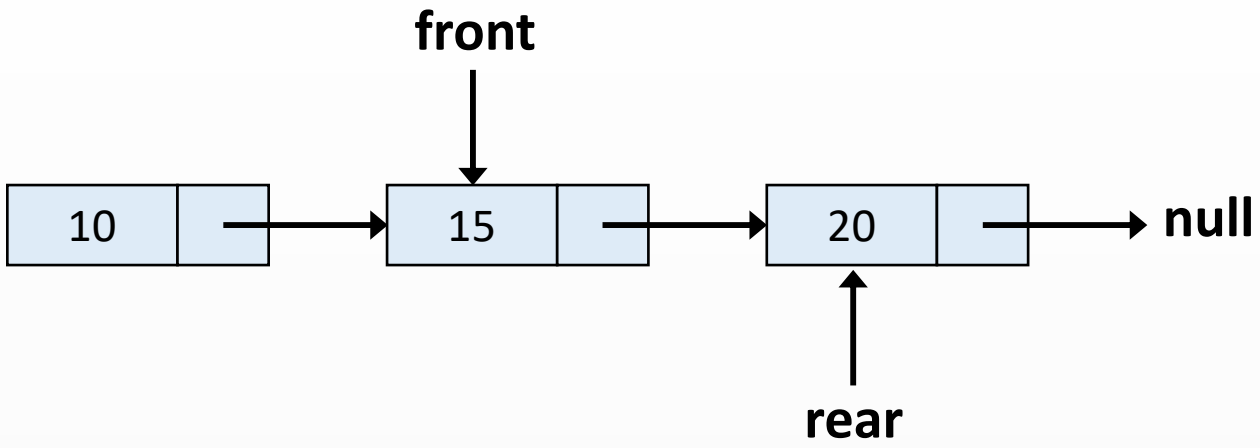
```
public int dequeue() {
  if(bosMu()) {
     throw new NoSuchElementException();
  }
  int sonuc = front.veri;
  front = front.sonraki;
  if(front == null) {
     rear = null;
  }
  uzunluk--;
  return sonuc;
}
```



uzunluk = 3
sonuc = 10

dequeue()

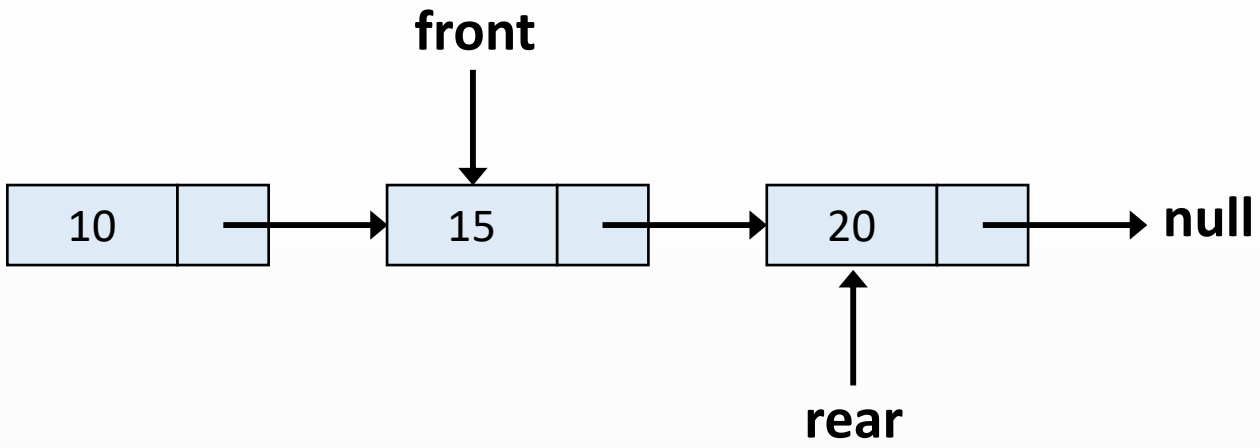
```
public int dequeue() {
  if(bosMu()) {
     throw new NoSuchElementException();
  }
  int sonuc = front.veri;
  front = front.sonraki;
  if(front == null) {
     rear = null;
  }
  uzunluk--;
  return sonuc;
}
```



uzunluk = 2
sonuc = 10

dequeue()

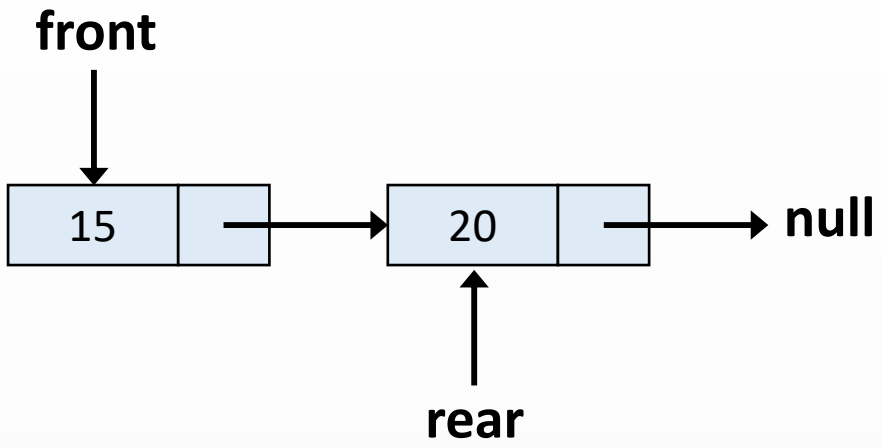
```
public int dequeue() {
  if(bosMu()) {
     throw new NoSuchElementException();
  }
  int sonuc = front.veri;
  front = front.sonraki;
  if(front == null) {
     rear = null;
  }
  uzunluk--;
  return sonuc;
}
```



uzunluk = 2
sonuc = 10

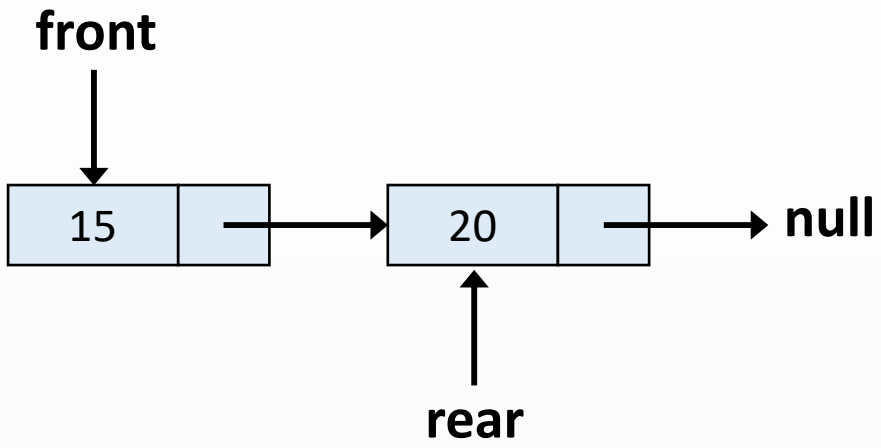
dequeue()

```
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

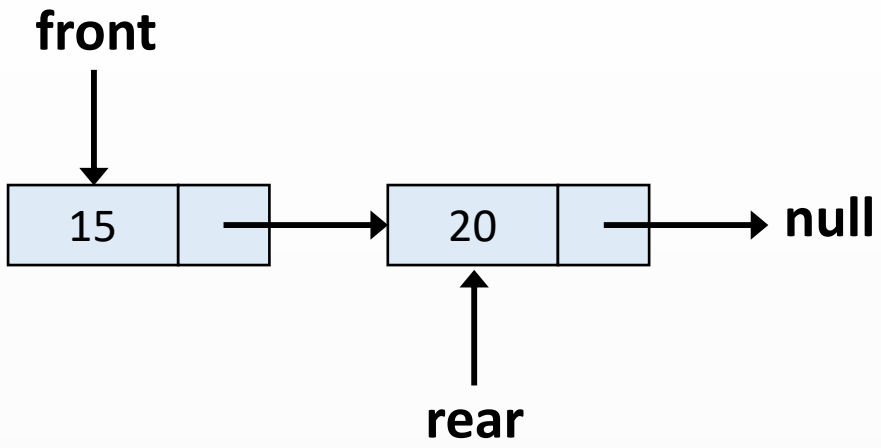
```
public int dequeue() {
  if(bosMu()) {
     throw new NoSuchElementException();
  }
  int sonuc = front.veri;
  front = front.sonraki;
  if(front == null) {
     rear = null;
  }
  uzunluk--;
  return sonuc;
}
```



uzunluk = 2

dequeue()

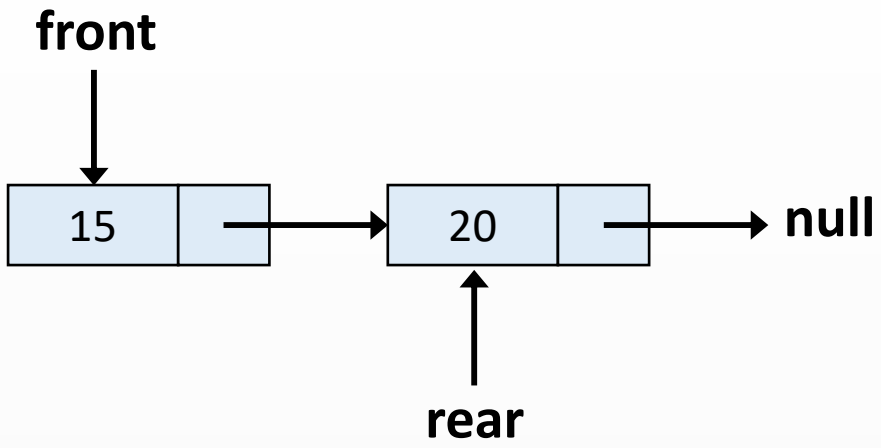
```
public int dequeue() {
  if(bosMu()) {
     throw new NoSuchElementException();
  }
  int sonuc = front.veri;
  front = front.sonraki;
  if(front == null) {
     rear = null;
  }
  uzunluk--;
  return sonuc;
}
```



uzunluk = 2

dequeue()

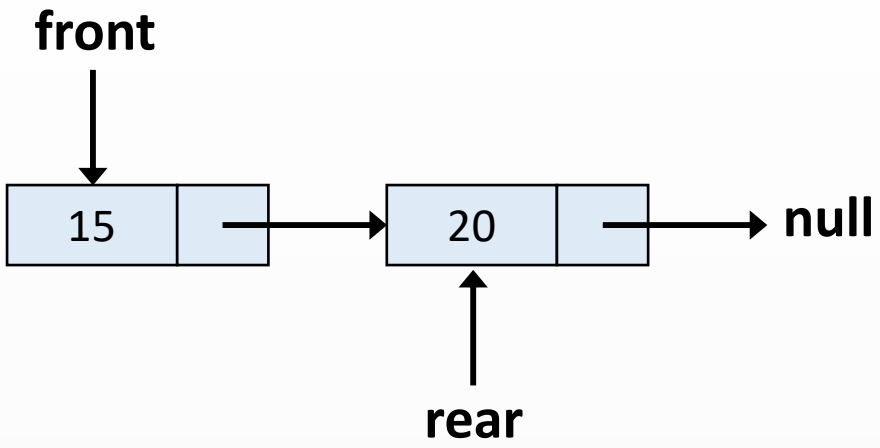
```
public int dequeue() {
  if(bosMu()) {
     throw new NoSuchElementException();
  }
  int sonuc = front.veri;
  front = front.sonraki;
  if(front == null) {
     rear = null;
  }
  uzunluk--;
  return sonuc;
}
```



uzunluk = 2

dequeue()

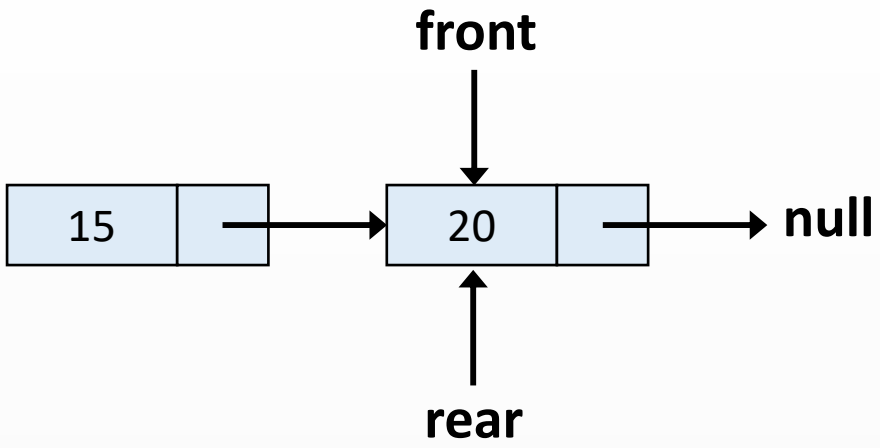
```
public int dequeue() {
  if(bosMu()) {
     throw new NoSuchElementException();
  }
  int sonuc = front.veri;
  front = front.sonraki;
  if(front == null) {
     rear = null;
  }
  uzunluk--;
  return sonuc;
}
```



uzunluk = 2
sonuc = 15

dequeue()

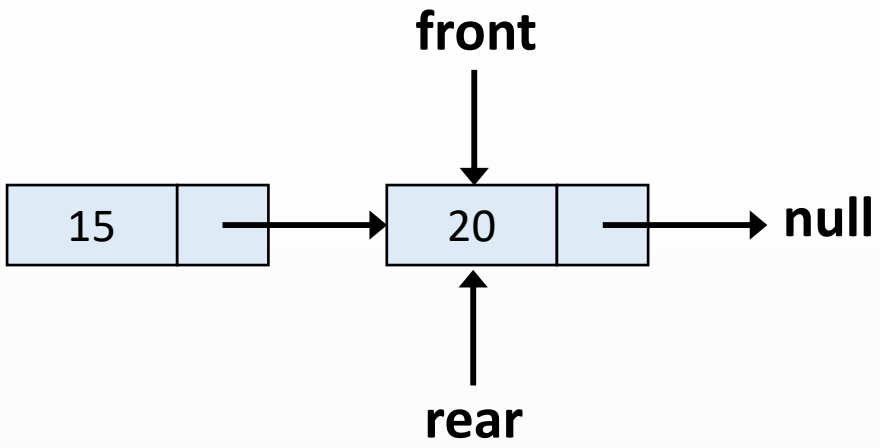
```
public int dequeue() {
  if(bosMu()) {
     throw new NoSuchElementException();
  }
  int sonuc = front.veri;
  front = front.sonraki;
  if(front == null) {
     rear = null;
  }
  uzunluk--;
  return sonuc;
}
```



uzunluk = 2
sonuc = 15

dequeue()

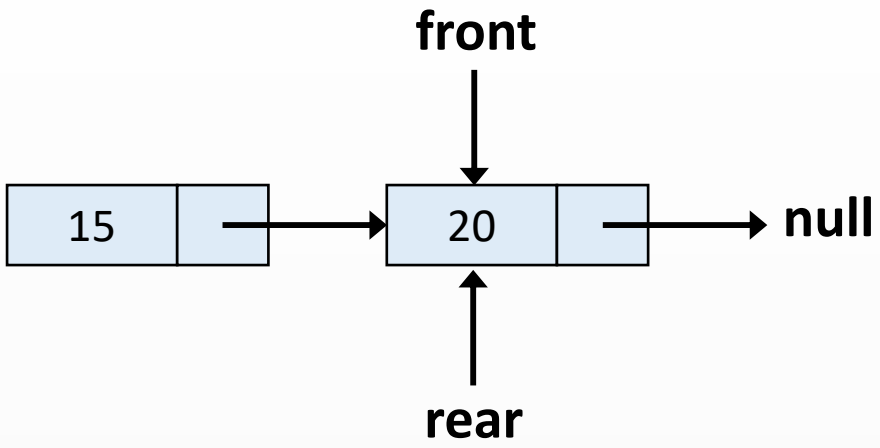
```
public int dequeue() {
  if(bosMu()) {
     throw new NoSuchElementException();
  }
  int sonuc = front.veri;
  front = front.sonraki;
  if(front == null) {
     rear = null;
  }
  uzunluk--;
  return sonuc;
}
```



uzunluk = 2
sonuc = 15

dequeue()

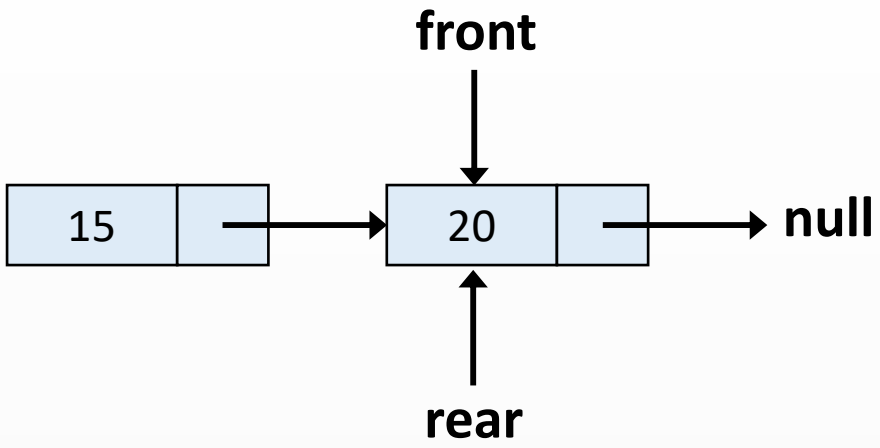
```
public int dequeue() {
  if(bosMu()) {
     throw new NoSuchElementException();
  }
  int sonuc = front.veri;
  front = front.sonraki;
  if(front == null) {
     rear = null;
  }
  uzunluk--;
  return sonuc;
}
```



uzunluk = 1
sonuc = 15

dequeue()

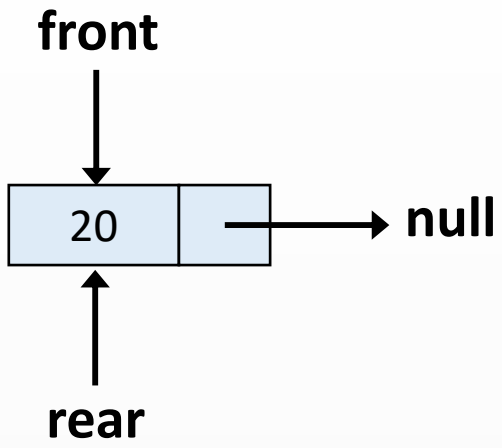
```
public int dequeue() {
  if(bosMu()) {
     throw new NoSuchElementException();
  }
  int sonuc = front.veri;
  front = front.sonraki;
  if(front == null) {
     rear = null;
  }
  uzunluk--;
  return sonuc;
}
```



uzunluk = 1
sonuc = 15

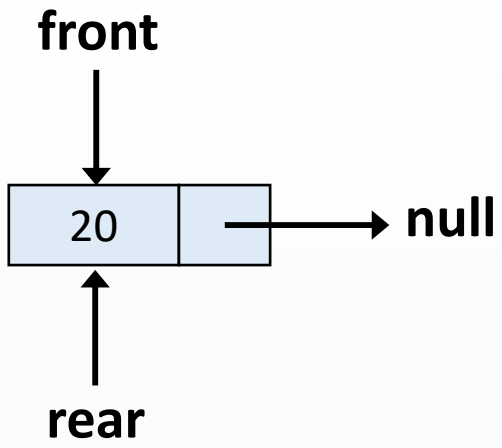
dequeue()

```
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

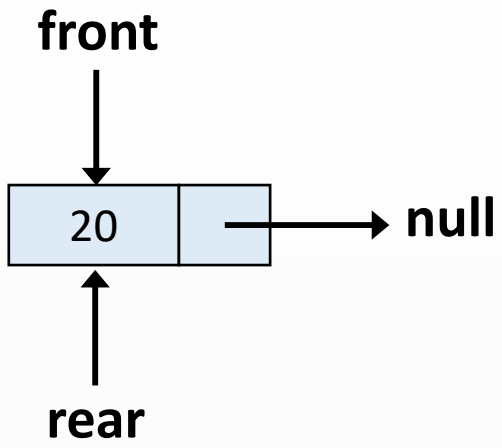
```
public int dequeue() {
  if(bosMu()) {
     throw new NoSuchElementException();
  }
  int sonuc = front.veri;
  front = front.sonraki;
  if(front == null) {
     rear = null;
  }
  uzunluk--;
  return sonuc;
}
```



uzunluk = 1

dequeue()

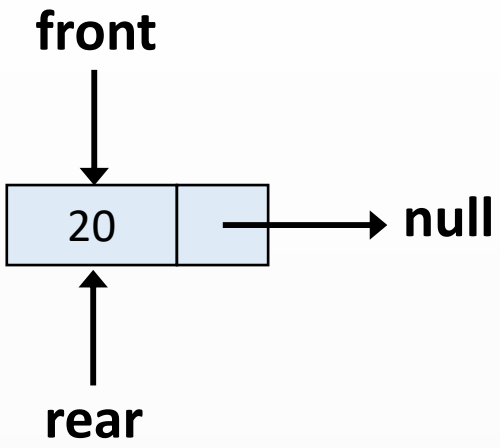
```
public int dequeue() {
  if(bosMu()) {
     throw new NoSuchElementException();
  }
  int sonuc = front.veri;
  front = front.sonraki;
  if(front == null) {
     rear = null;
  }
  uzunluk--;
  return sonuc;
}
```



```
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

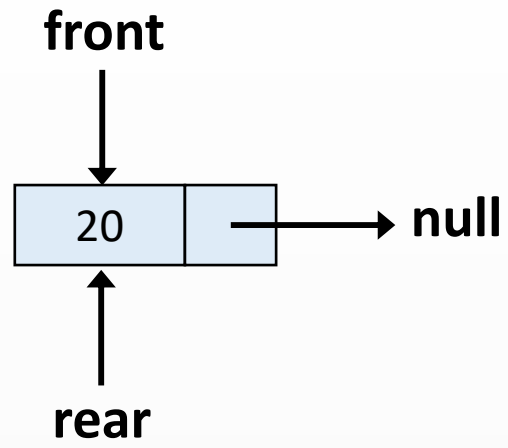
dequeue()



```
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

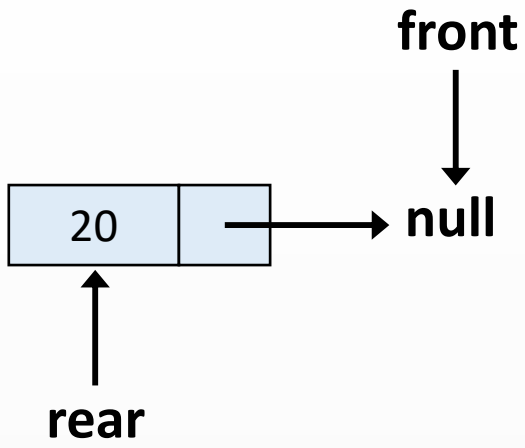
dequeue()



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;
}
```

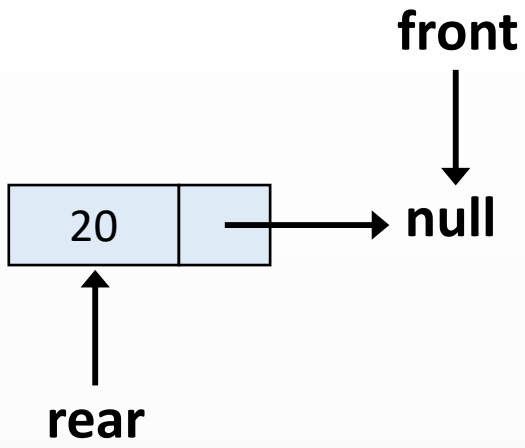


uzunluk = 1

sonuc = 20

dequeue()

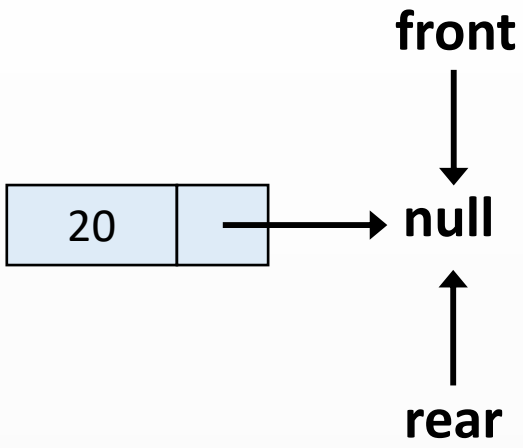
```
public int dequeue() {
  if(bosMu()) {
     throw new NoSuchElementException();
  }
  int sonuc = front.veri;
  front = front.sonraki;
  if(front == null) {
     rear = null;
  }
  uzunluk--;
  return sonuc;
}
```



uzunluk = 1
sonuc = 20

dequeue()

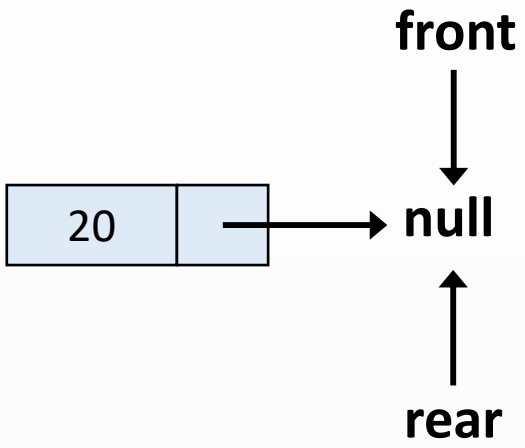
```
public int dequeue() {
  if(bosMu()) {
     throw new NoSuchElementException();
  }
  int sonuc = front.veri;
  front = front.sonraki;
  if(front == null) {
     rear = null;
  }
  uzunluk--;
  return sonuc;
}
```



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;
}
```

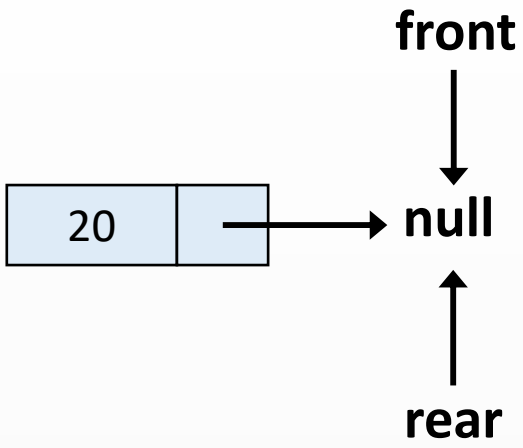


uzunluk = 0

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;
}
```

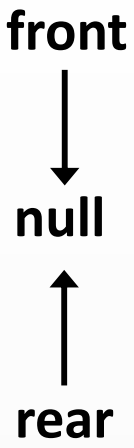


uzunluk = 0

sonuc = 20

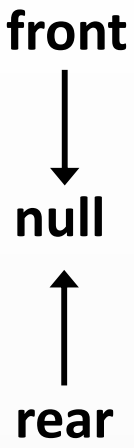
dequeue()

```
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

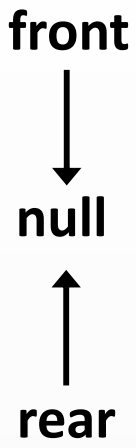
```
public int dequeue() {
  if(bosMu()) {
     throw new NoSuchElementException();
  }
  int sonuc = front.veri;
  front = front.sonraki;
  if(front == null) {
     rear = null;
  }
  uzunluk--;
  return sonuc;
}
```



uzunluk = 0

dequeue()

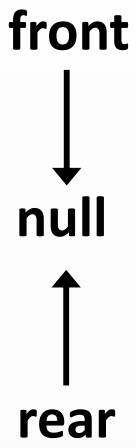
```
public int dequeue() {
  if(bosMu()) {
     throw new NoSuchElementException();
  }
  int sonuc = front.veri;
  front = front.sonraki;
  if(front == null) {
     rear = null;
  }
  uzunluk--;
  return sonuc;
}
```



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;
}
```

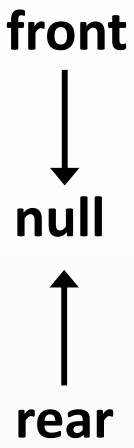


uzunluk = 0

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

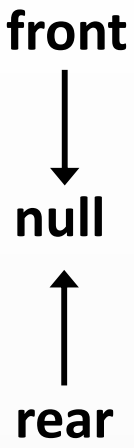


uzunluk = 0

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)

- Dairesel kuyruğun çalışma prensibi, lineer kuyruğa benzer, ancak son öğe ilk öğe ile bağlantılıdır.
- İlk giren öğe (front), son giren öğe (rear) ile bir döngü oluşturur.
- Öğeler kuyruğa sırayla eklenir ve kuyruğun sonundan başına geri döner.
- Dairesel kuyrukta, öğelerin döngüsel hareketini hesaplamak için modulo işlemi kullanılır.
- Modulo işlemi, kuyruğun kapasitesi aşılmadan öğelerin eklenmesini ve kaldırılmasını sağlar.

Öncelikli Kuyruk (Priority Queue)

- Öncelikli kuyruk, öğeleri belirli bir öncelik düzenine göre sıralayan özel bir kuyruk türüdür.
- Öncelik, öğelerin sırasını belirler. Öncelik yüksek değere sahip öğenin önce işlenmesini veya düşük değere sahip öğenin önce işlenmesini sağlayabilir.
- En yüksek değere sahip öğeye öncelik vererek azalan sıralı bir kuyruk oluşturulabilir.
- En düşük değere sahip öğeye öncelik vererek artan sıralı bir kuyruk oluşturulabilir.
- Öncelikli kuyruklar, zaman duyarlı işlemleri, iş parçacığı yönetimini, veri sıkıştırma algoritmalarını ve daha birçok alanda kullanılır.

Dequeue (Double Ended Queue)

- Çift Uçlu Kuyruk olarak da bilinir. İsmi çift uçlu olmasından gelir.
- Bu, kuyruğun hem ön hem de arka tarafından öğe eklenebilir veya çıkarılabilir, diğer kuyruklardan farklıdır.
- Dequeue, FIFO kuralını ihlal edebilir. Çünkü her iki uçtan da eklemeler ve çıkarmalar yapılabilir.
- Dequeue, iş parçacığı yönetimi, arama algoritmaları ve daha birçok alanda kullanılabilir.

1'den n'e Kadar İkilik Sayıları Üretme

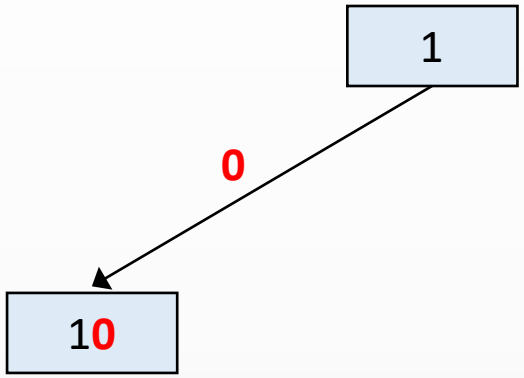
- Kuyruk veri yapısı kullanılarak 1'den n'ye kadar ikilik sayılar nasıl üretilir?
- **Örnek 1:**
  - **Girdi**: n = 3
  - **Çıktı**: sonuc = {"1","10","11"}
- **Örnek 2:**
  - **Girdi**: n = 5
  - **Çıktı**: sonuc = {"1","10","11","100","101"}

1'den n'e Kadar İkilik Sayıları Üretme

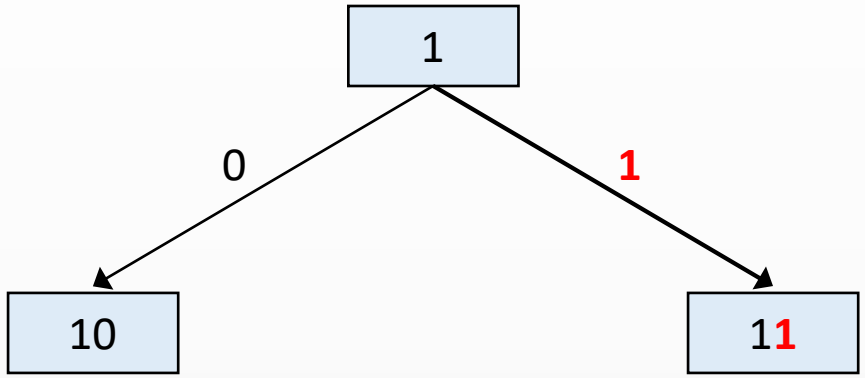
1

İkilik	Onluk
1	1

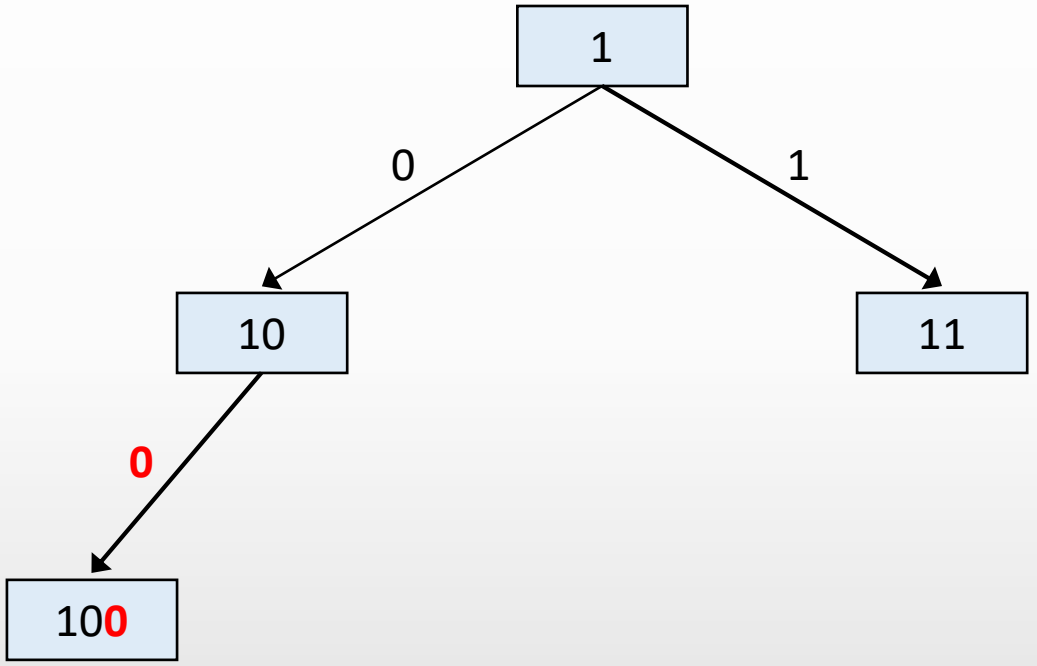
1'den n'e Kadar İkilik Sayıları Üretme



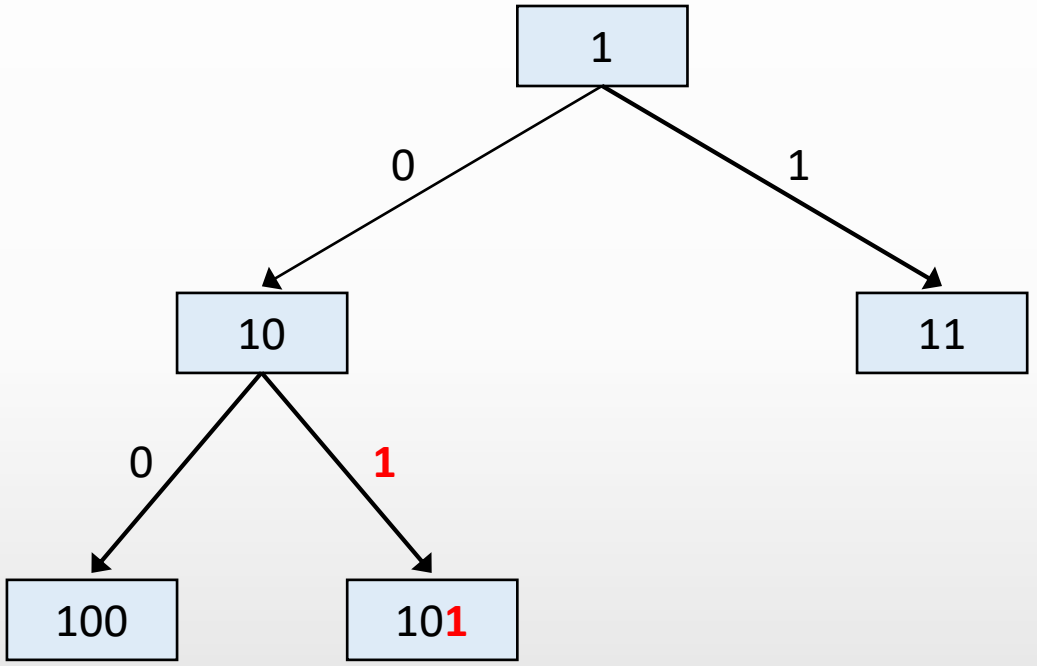
İkilik	Onluk
1	1
10	2



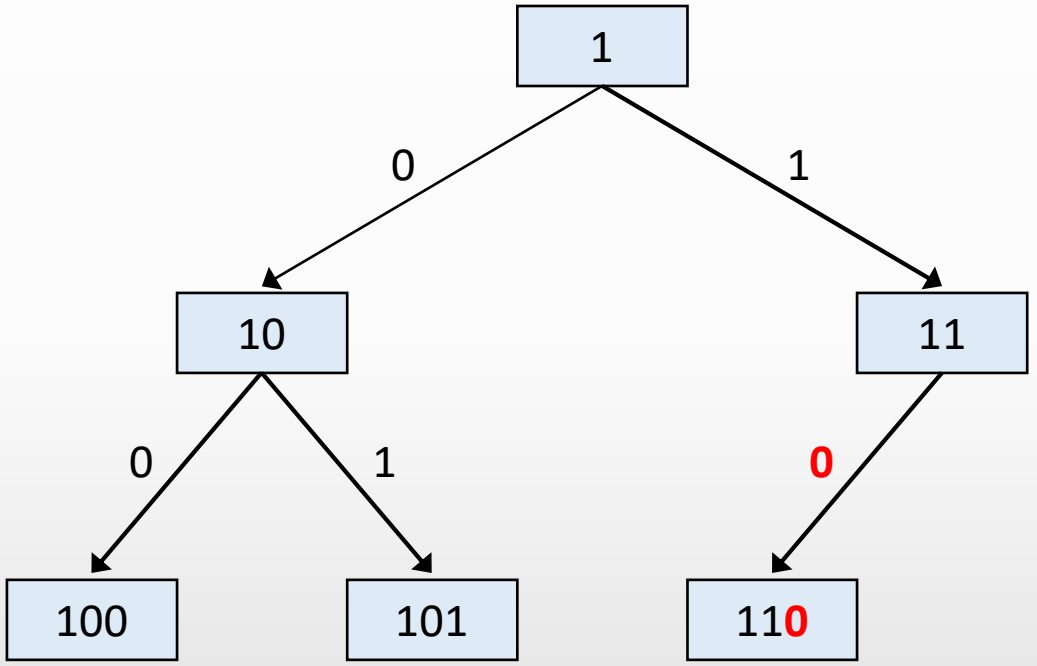
İkilik	Onluk
1	1
10	2
11	3



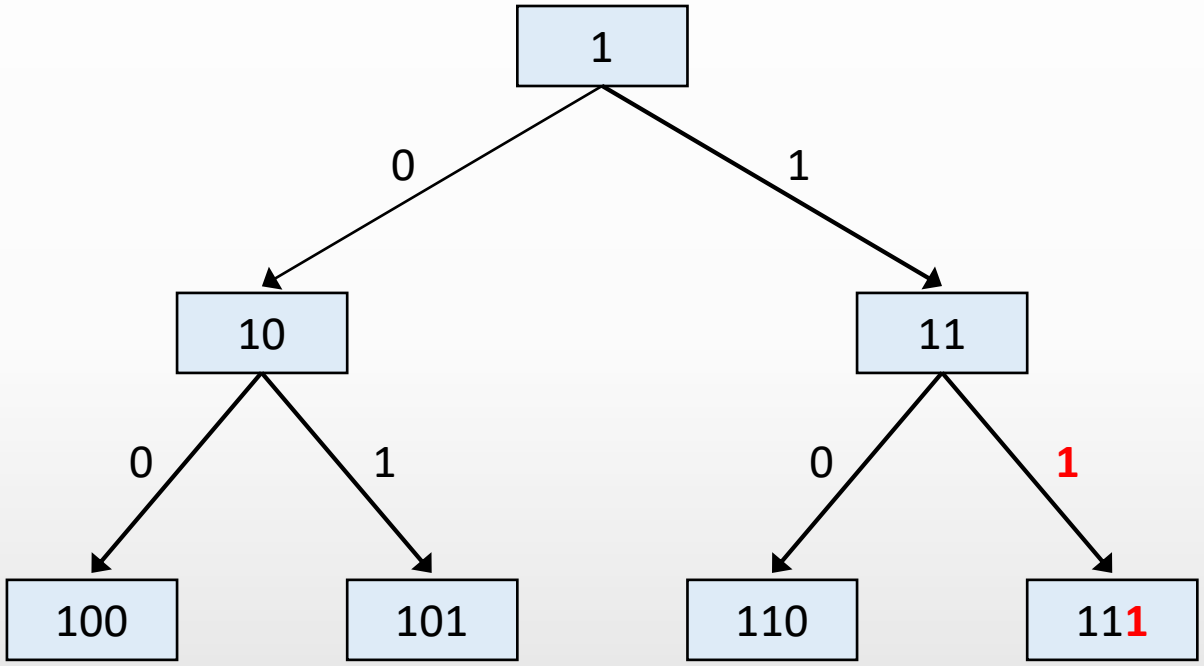
İkilik	Onluk
1	1
10	2
11	3
100	4



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;
}
```

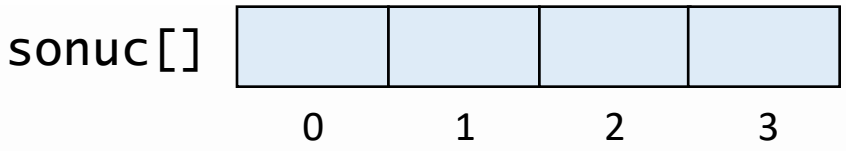
```
ikilikSayiUret(4);
```

```
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);

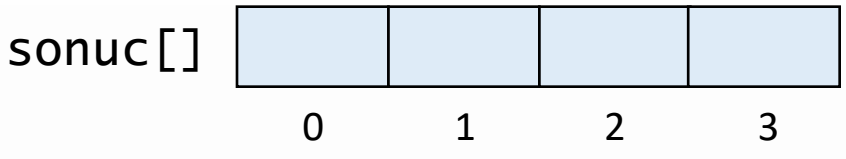
```
String[] ikilikSayiUret(int n) {
  String[] sonuc = new String[n];
  Queue<String> q = new LinkedList<>();
  q.offer("1");
  for(int i = 0; i < n; i++) {
     sonuc[i] = q.poll();
     String n1 = sonuc[i] + "0";
     String n2 = sonuc[i] + "1";
     q.offer(n1);
     q.offer(n2);
  }
  return sonuc;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

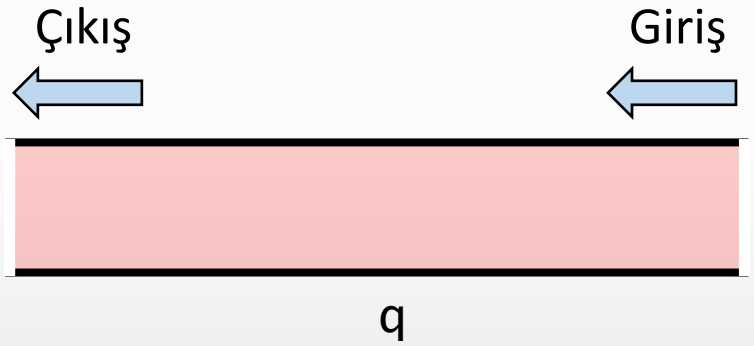


n = 4

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;
}
```

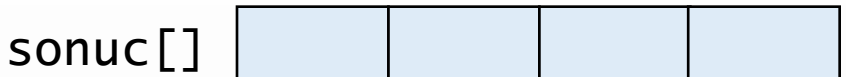


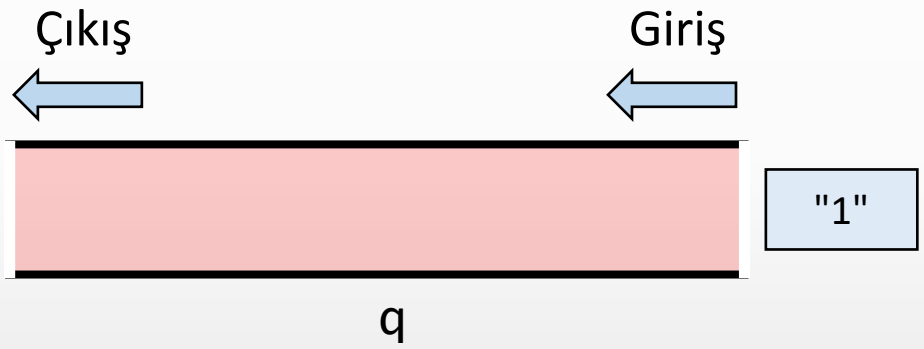


n = 4

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;
}
```

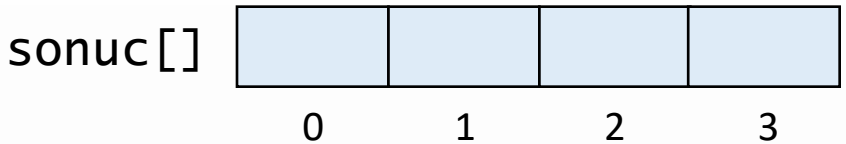


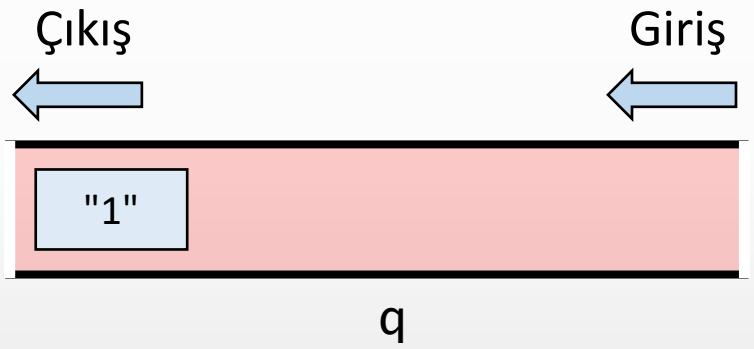


n = 4

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;
}
```

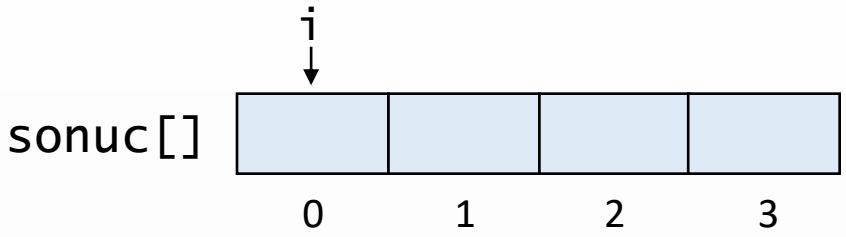


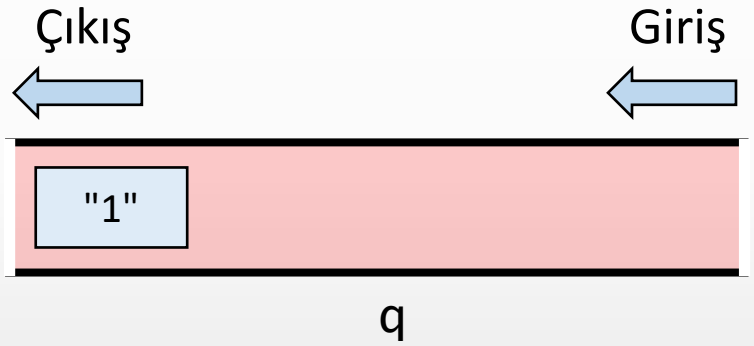


n = 4

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;
}
```



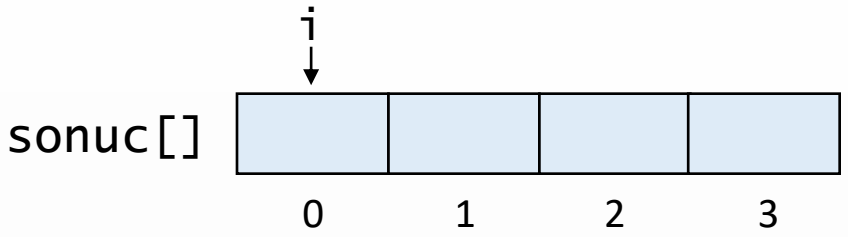


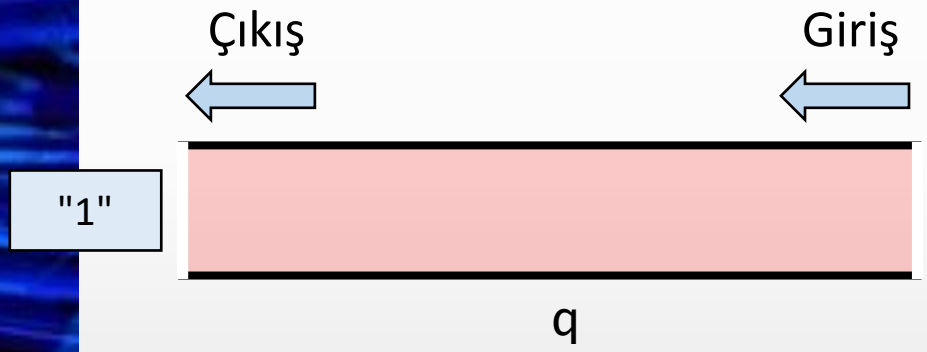
i = 0

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;
}
```



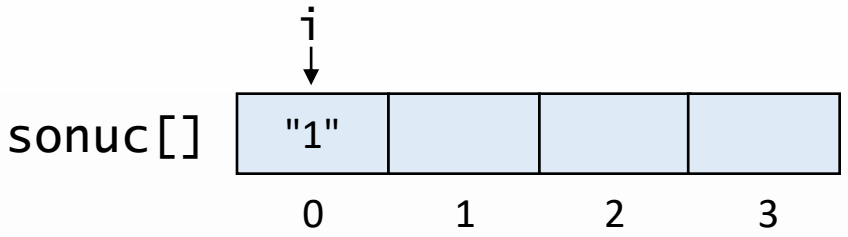


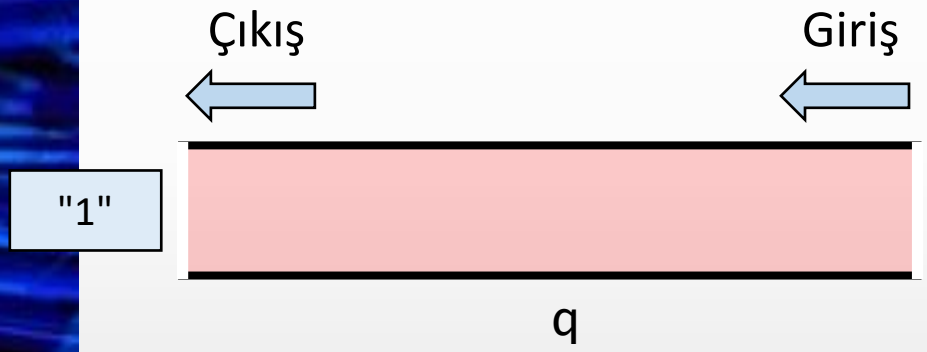
i = 0

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;
}
```



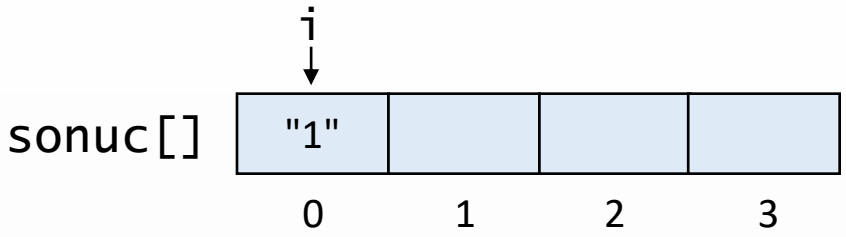


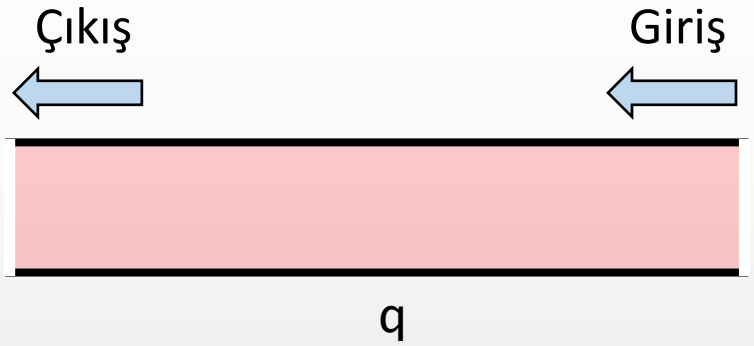
i = 0

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;
}
```



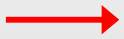


n1 = "10"

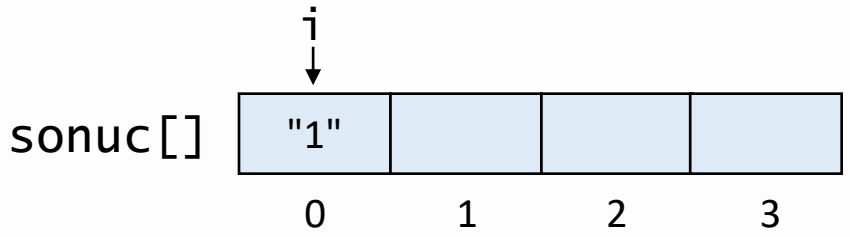
i = 0

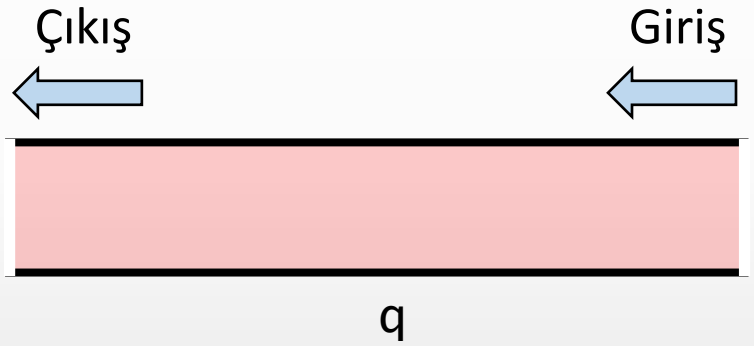
n = 4

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"

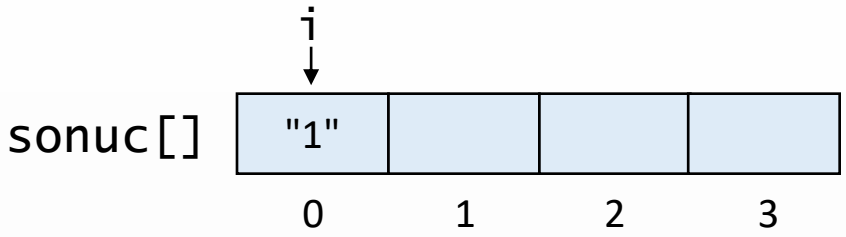
n1 = "10"

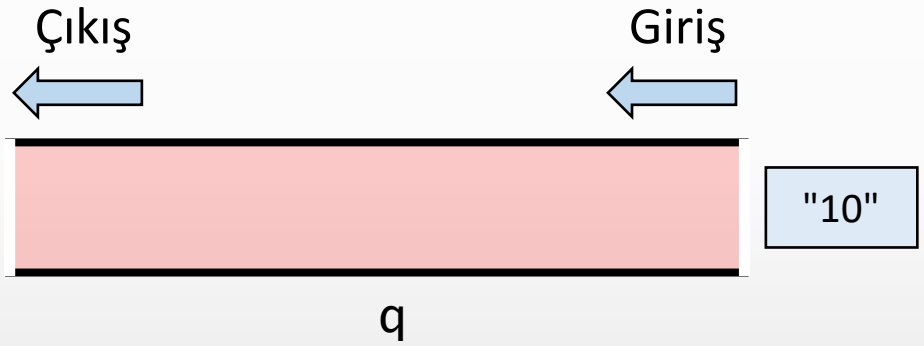
i = 0

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"

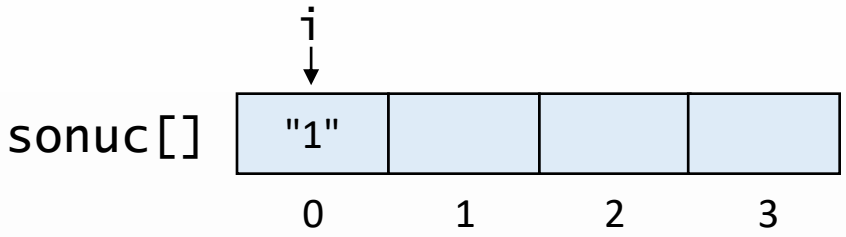
n1 = "10"

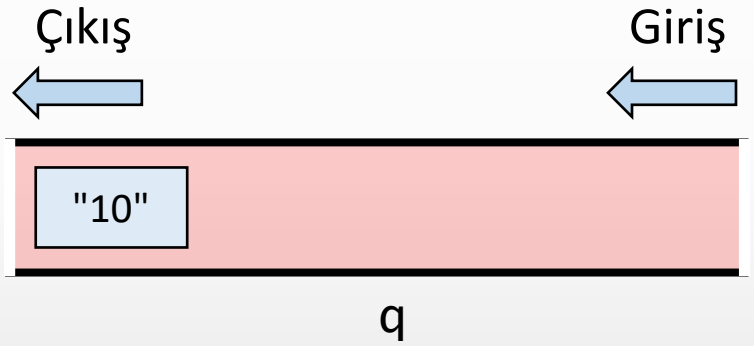
i = 0

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"

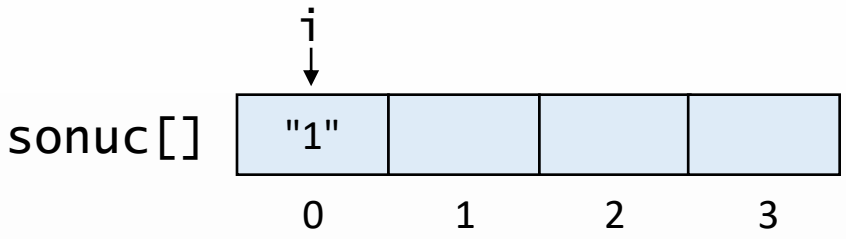
n1 = "10"

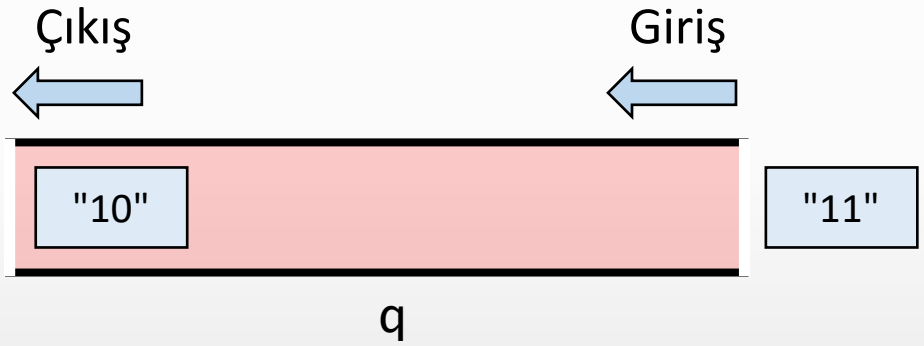
i = 0

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"

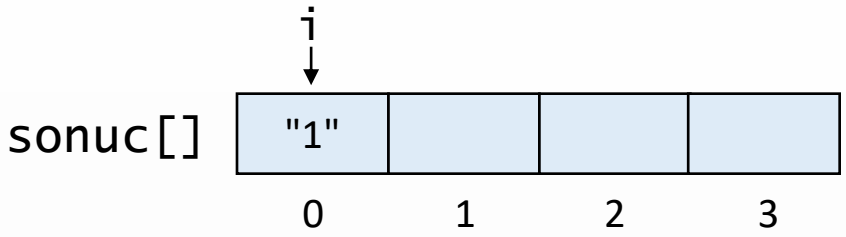
n1 = "10"

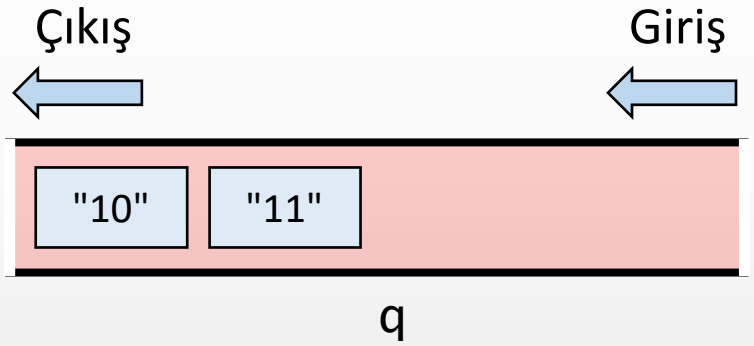
i = 0

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;
}
```





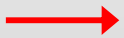
n2 = "11"

n1 = "10"

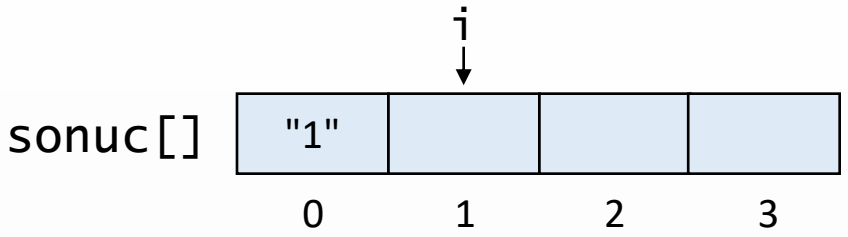
i = 0

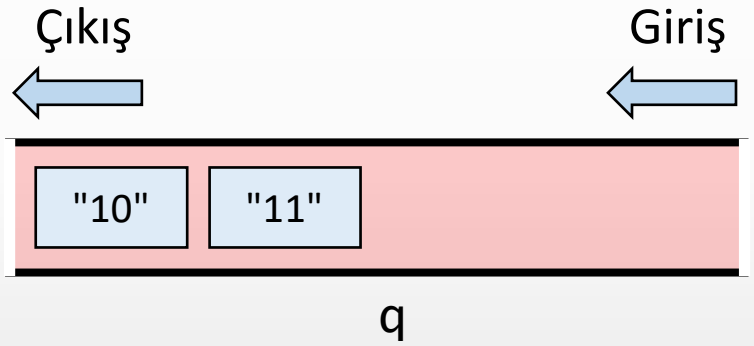
n = 4

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;
}
```



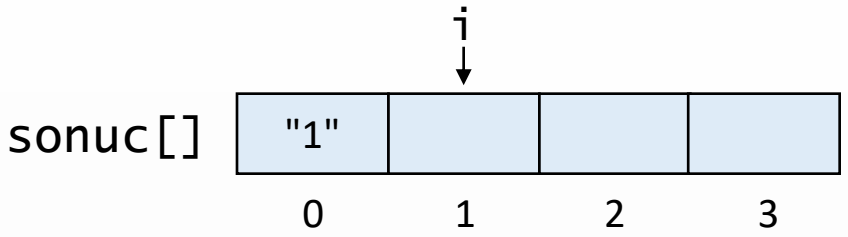


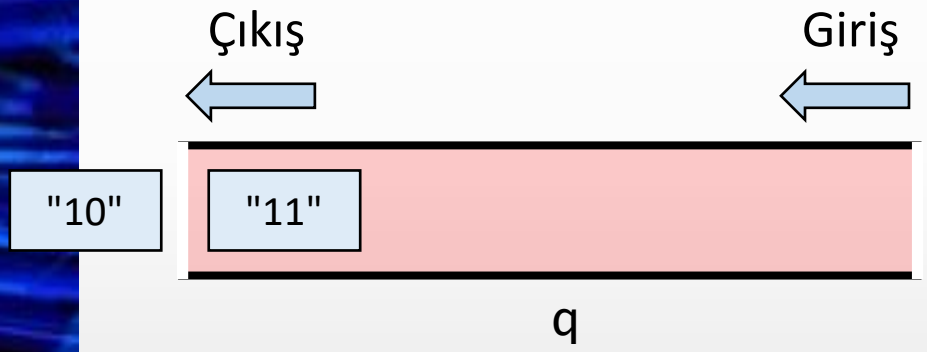
i = 1

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;
}
```



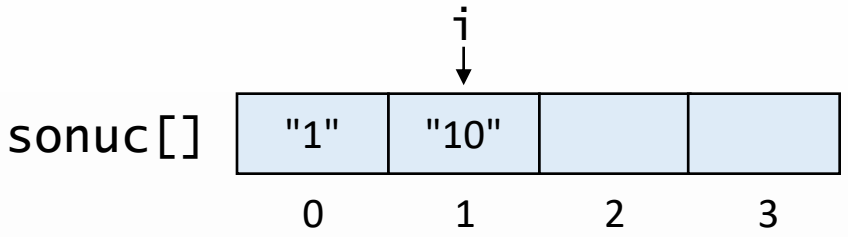


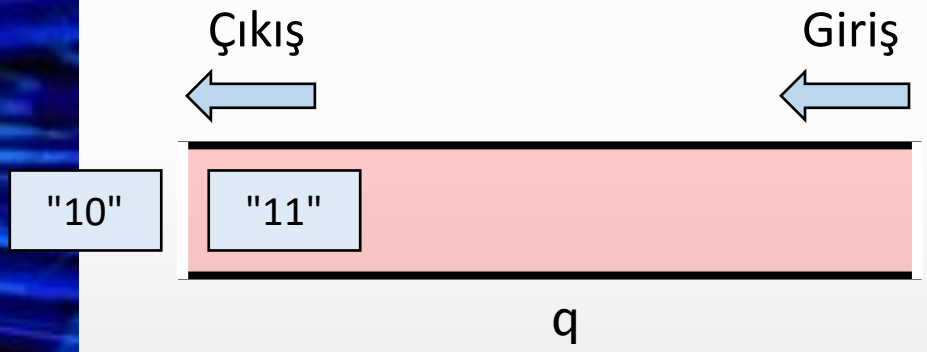
i = 1

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;
}
```



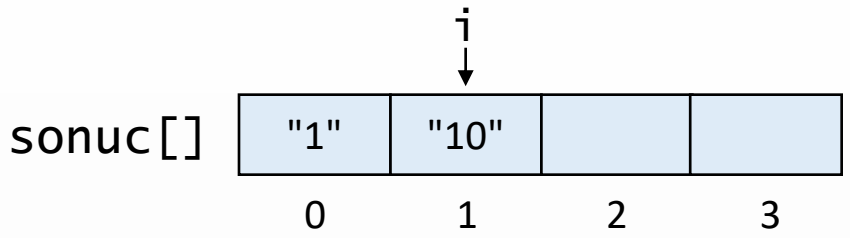


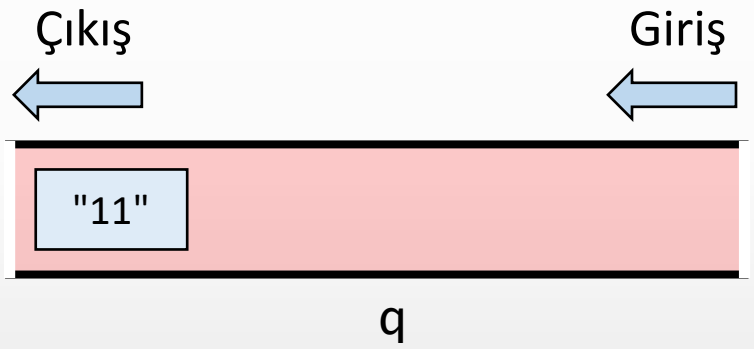
i = 1

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;
}
```



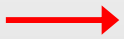


n1 = "100"

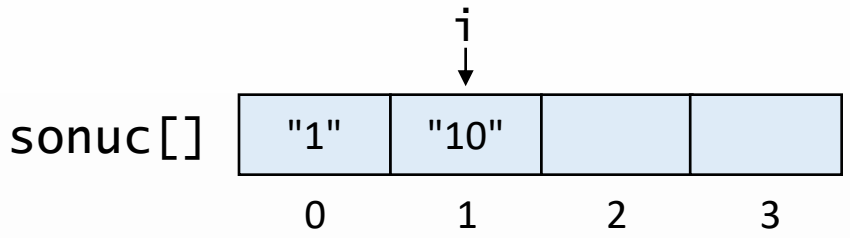
i = 1

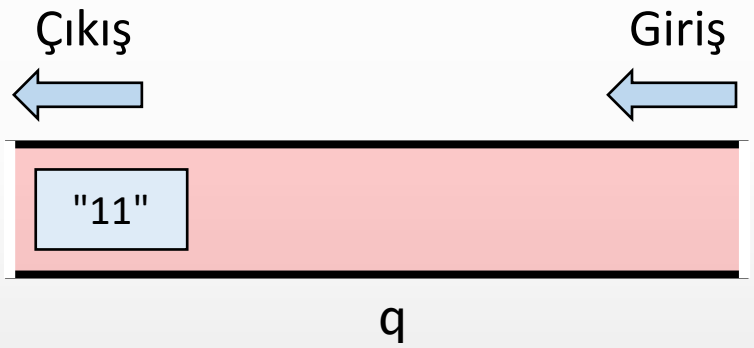
n = 4

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;
}
```





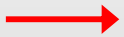
n2 = "101"

n1 = "100"

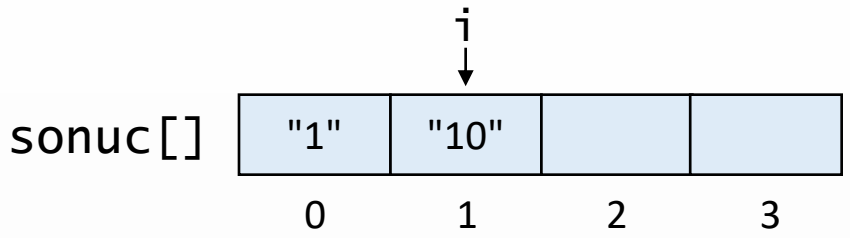
i = 1

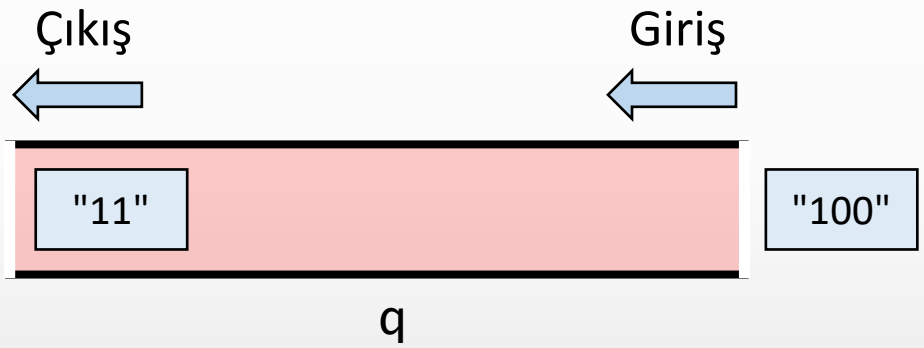
n = 4

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;
}
```





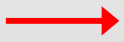
n2 = "101"

n1 = "100"

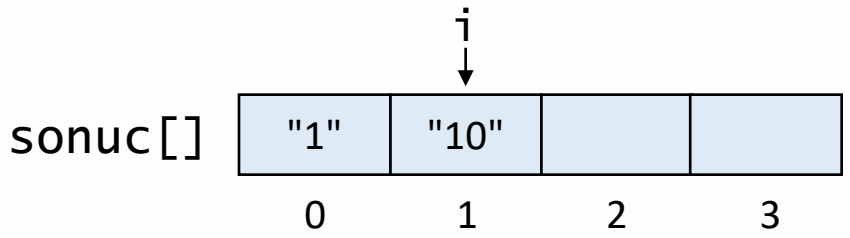
i = 1

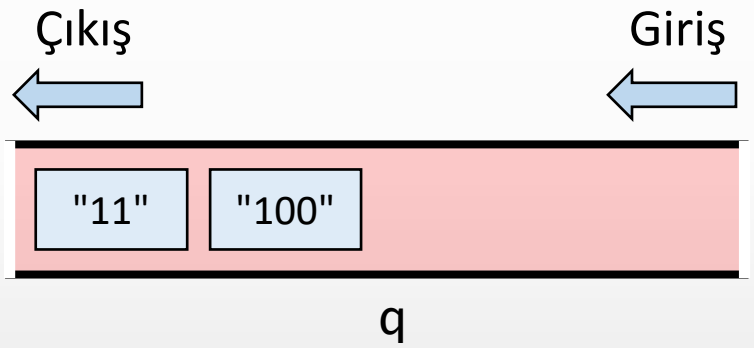
n = 4

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;
}
```





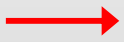
n2 = "101"

n1 = "100"

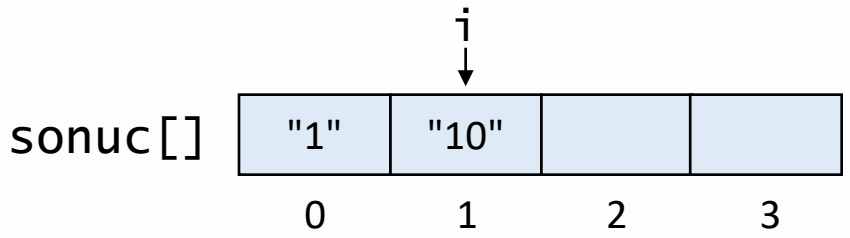
i = 1

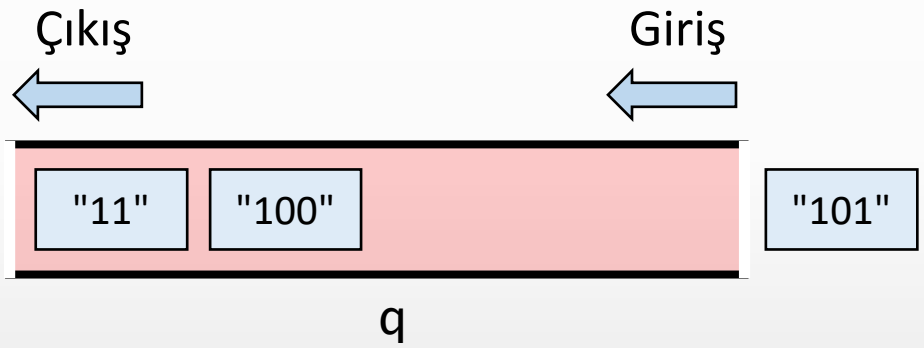
n = 4

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;
}
```





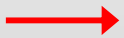
n2 = "101"

n1 = "100"

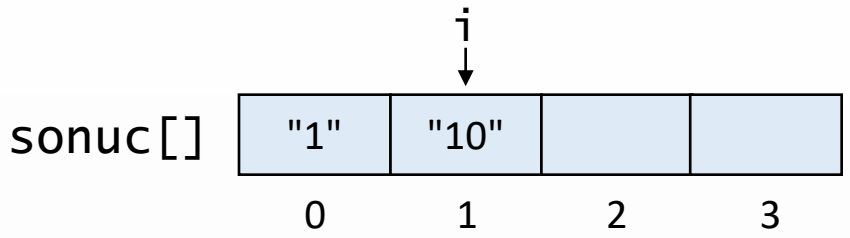
i = 1

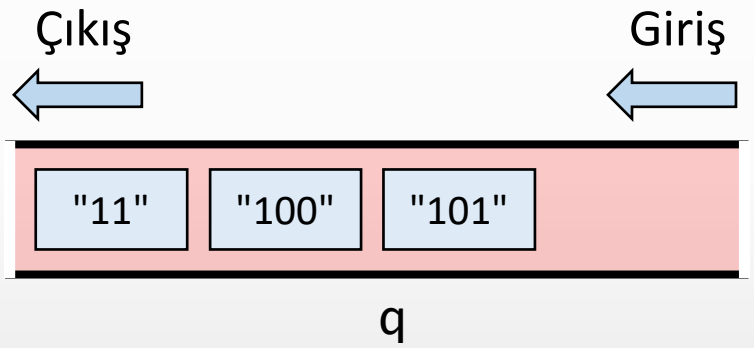
n = 4

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;
}
```





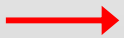
n2 = "101"

n1 = "100"

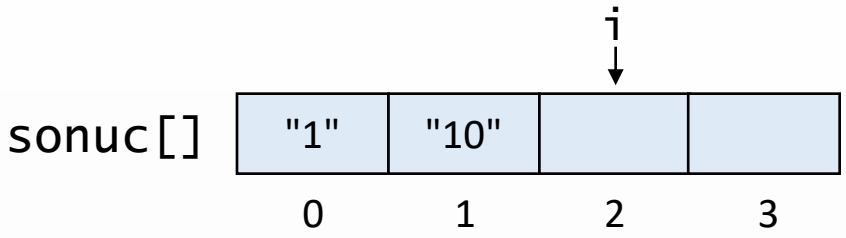
i = 1

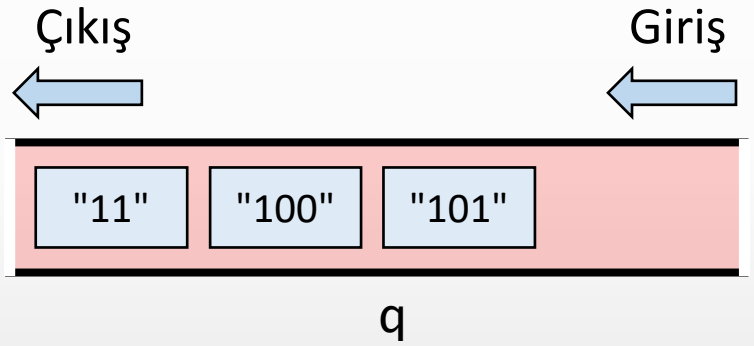
n = 4

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;
}
```



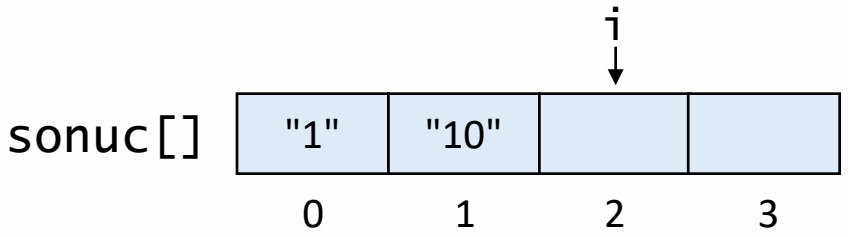


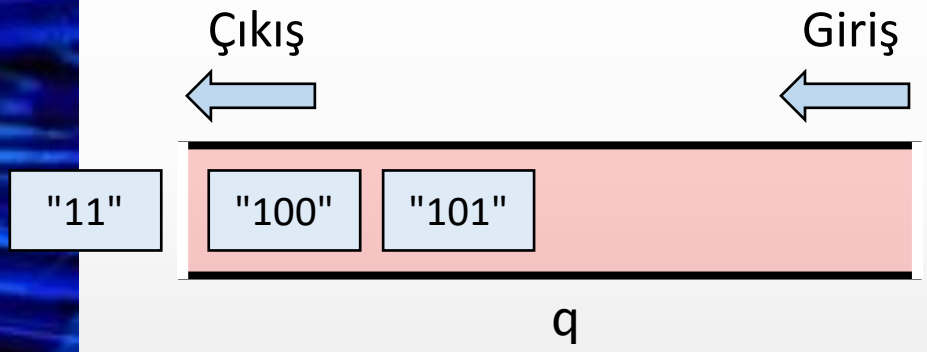
i = 2

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;
}
```



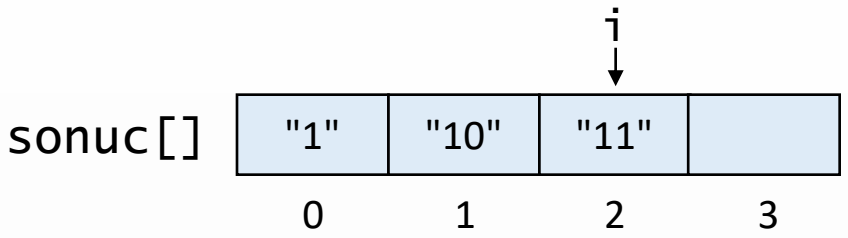


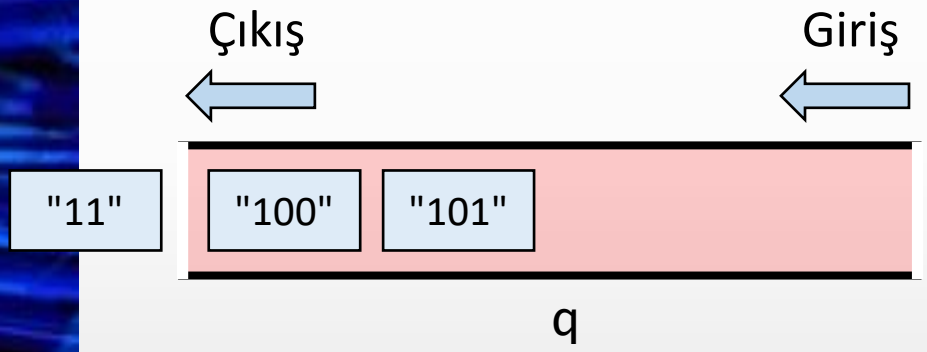
i = 2

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;
}
```

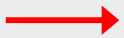




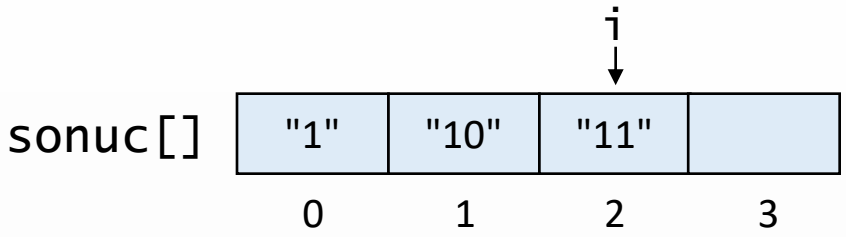
i = 2

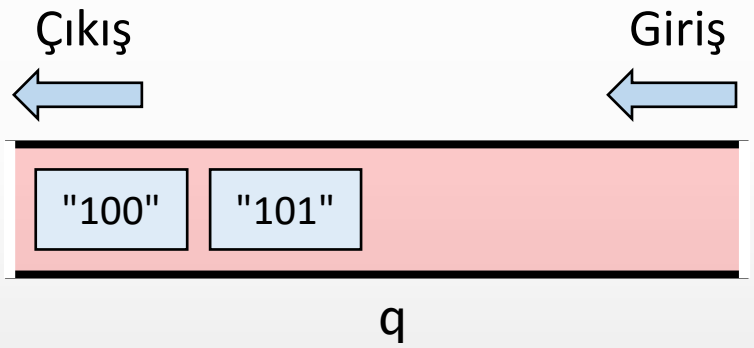
n = 4

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;
}
```



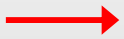


n1 = "110"

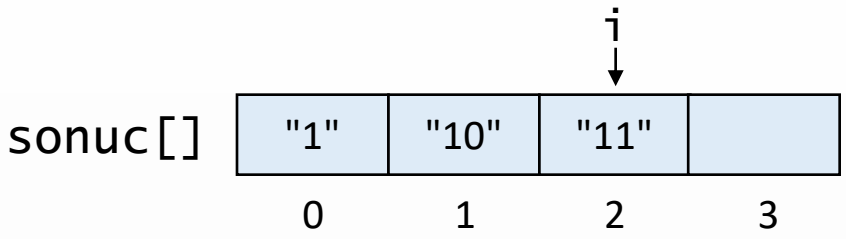
i = 2

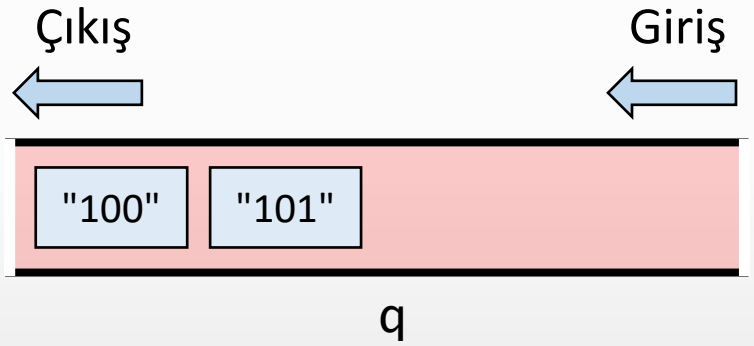
n = 4

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"

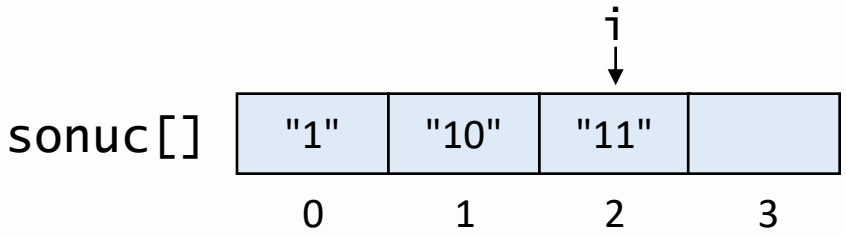
n1 = "110"

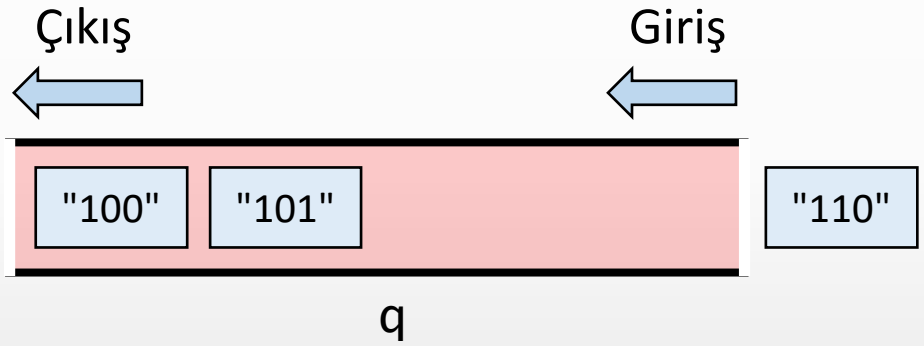
i = 2

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"

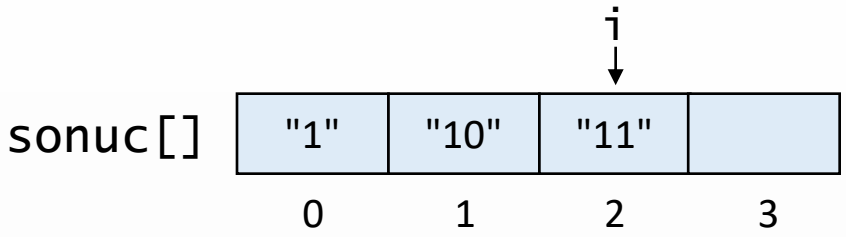
n1 = "110"

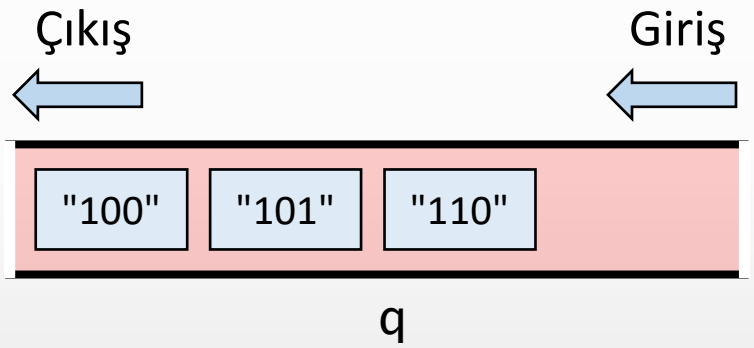
i = 2

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"

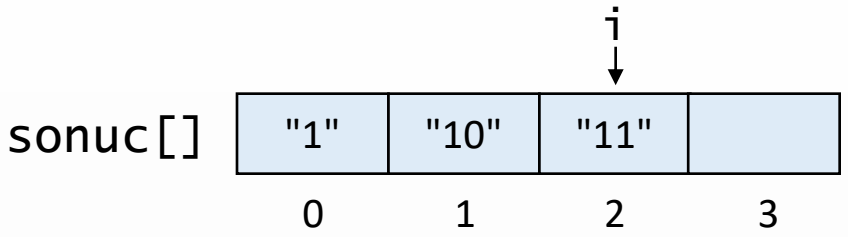
n1 = "110"

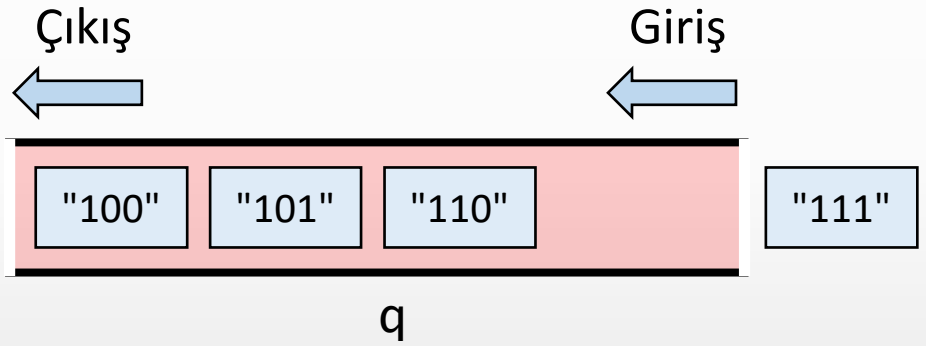
i = 2

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"

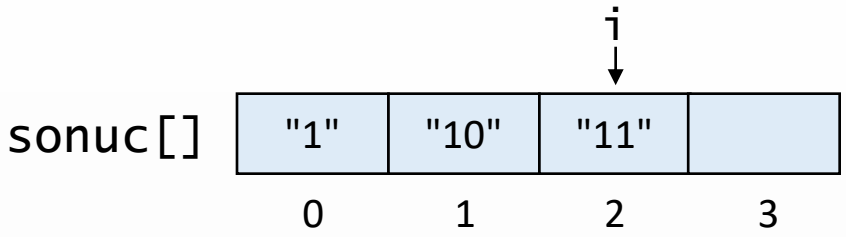
n1 = "110"

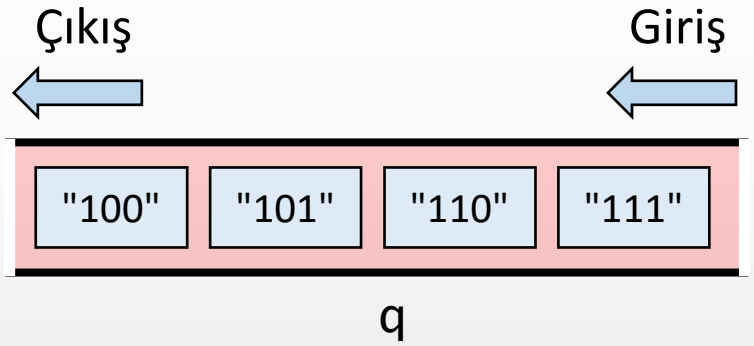
i = 2

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;
}
```





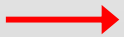
n2 = "111"

n1 = "110"

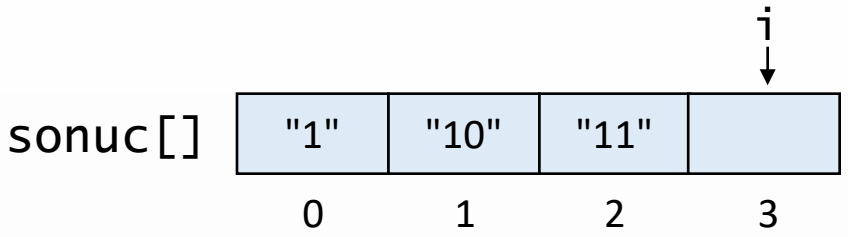
i = 2

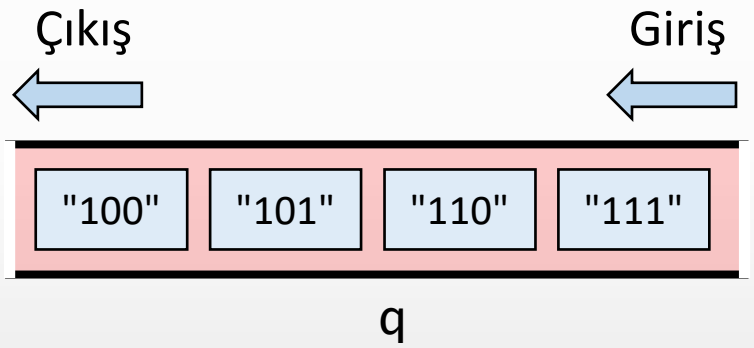
n = 4

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;
}
```



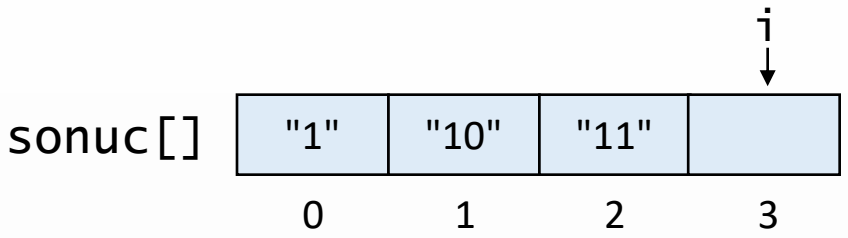


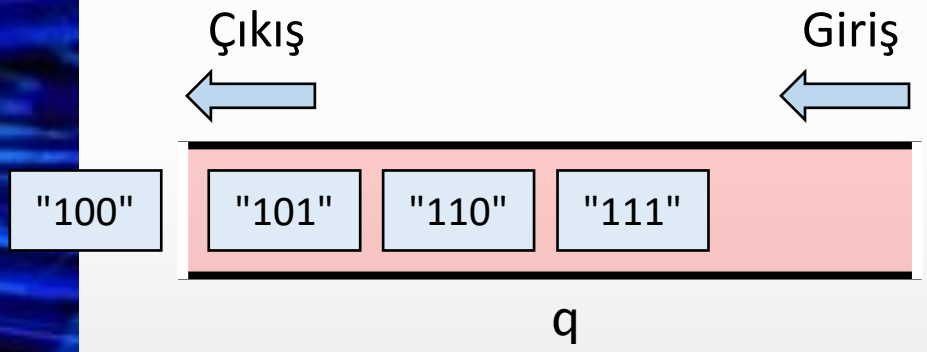
i = 3

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;
}
```



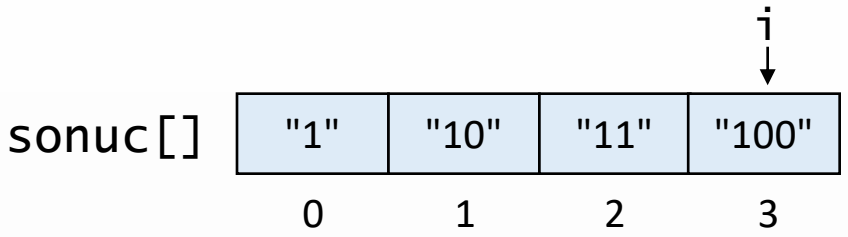


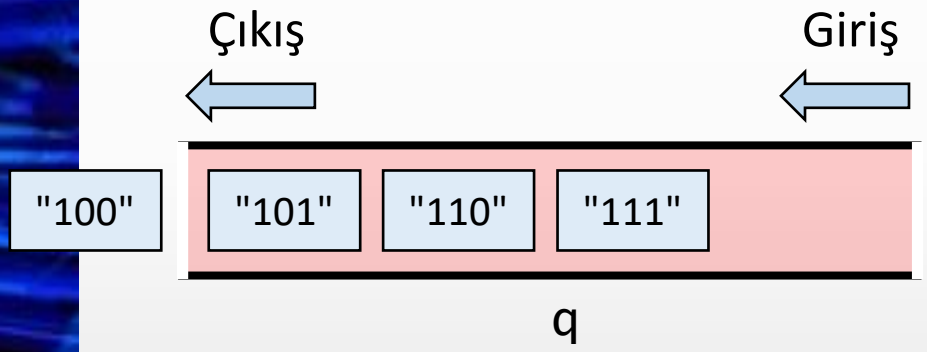
i = 3

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;
}
```



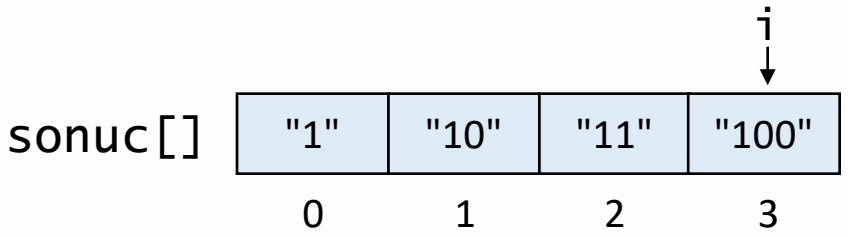


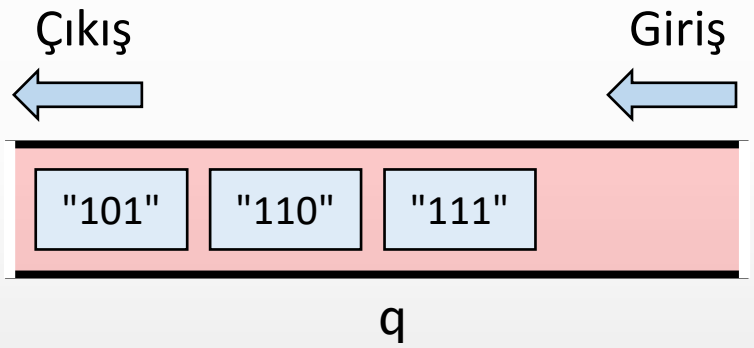
i = 3

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;
}
```



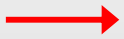


n1 = "1000"

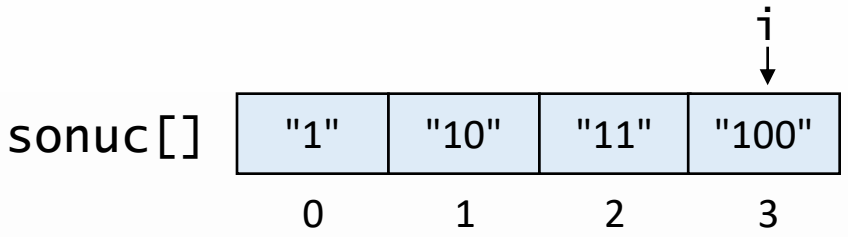
i = 3

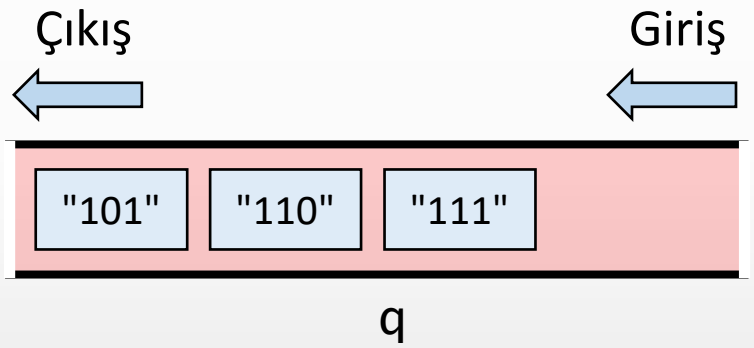
n = 4

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;
}
```





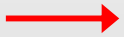
n2 = "1001"

n1 = "1000"

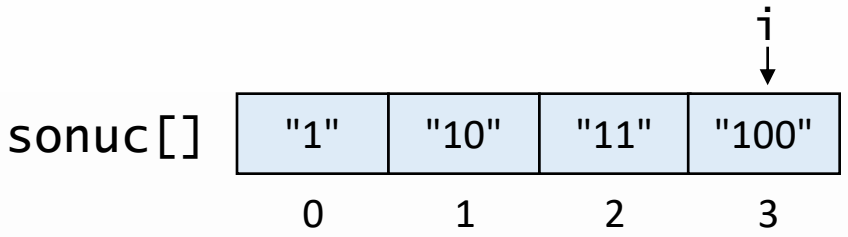
i = 3

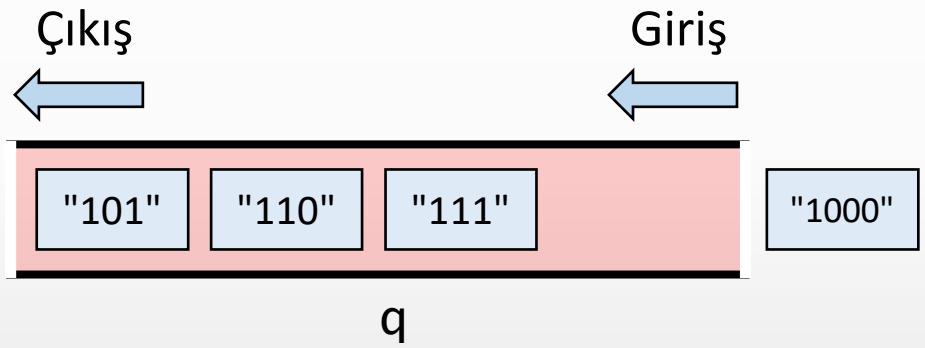
n = 4

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;
}
```





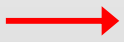
n2 = "1001"

n1 = "1000"

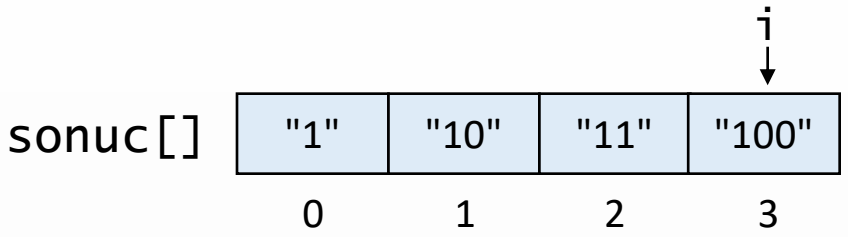
i = 3

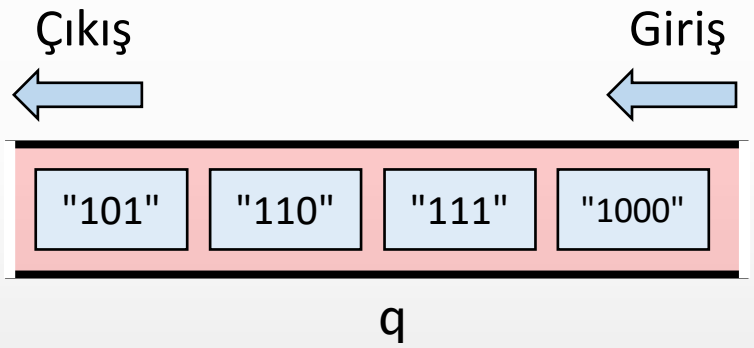
n = 4

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"

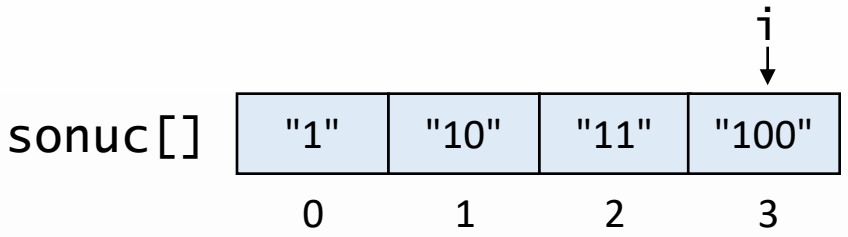
n1 = "1000"

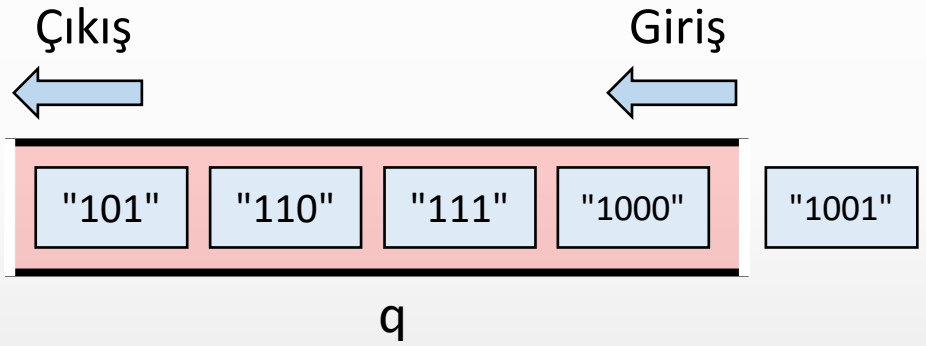
i = 3

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;
}
```





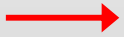
n2 = "1001"

n1 = "1000"

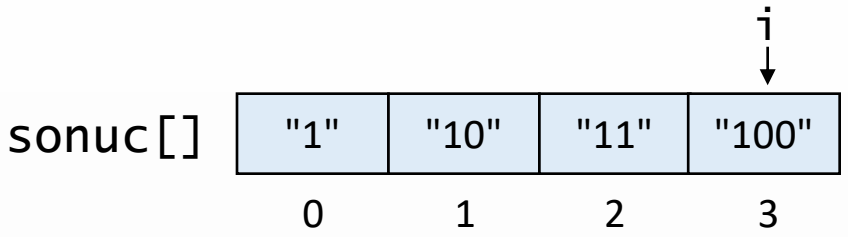
i = 3

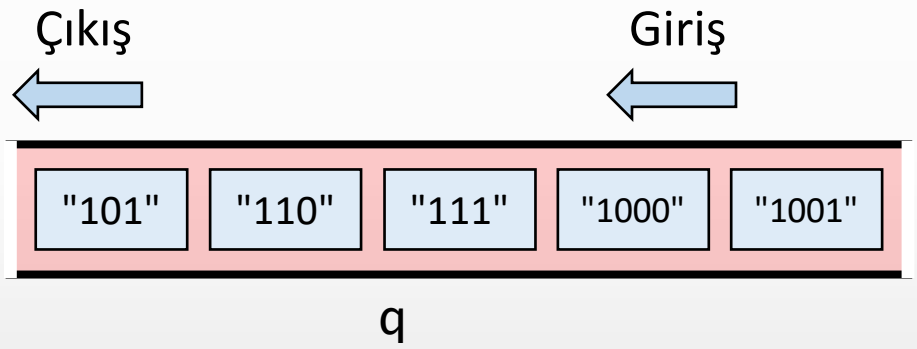
n = 4

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;
}
```





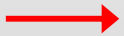
n2 = "1001"

n1 = "1000"

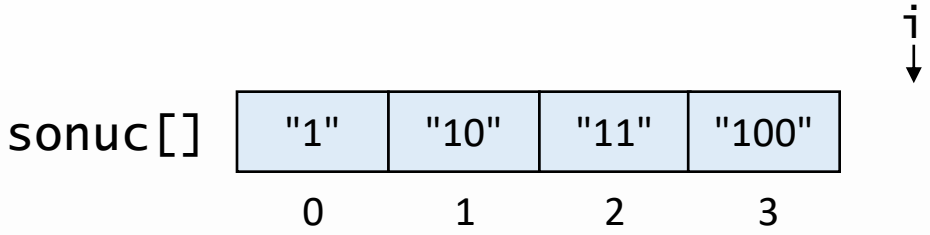
i = 3

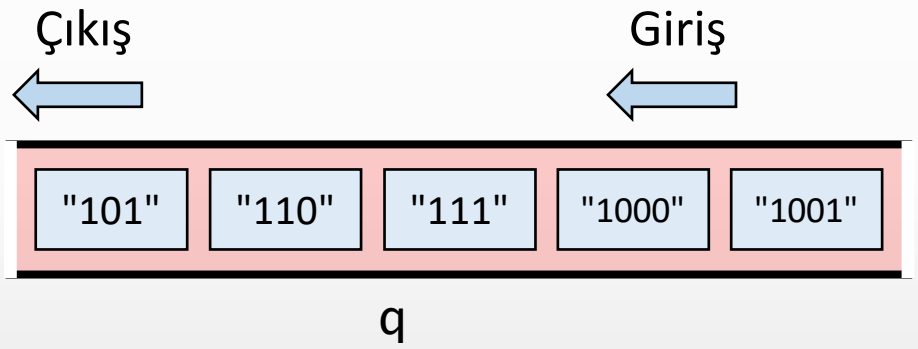
n = 4

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;
}
```



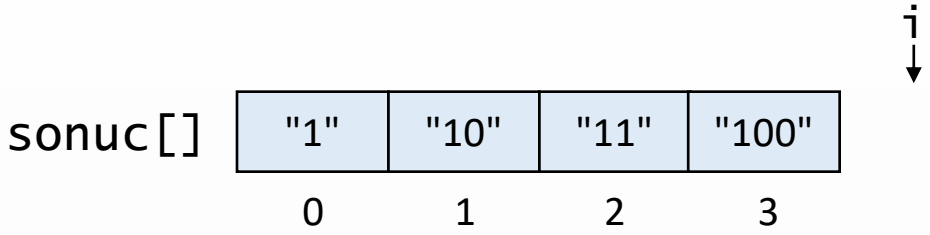


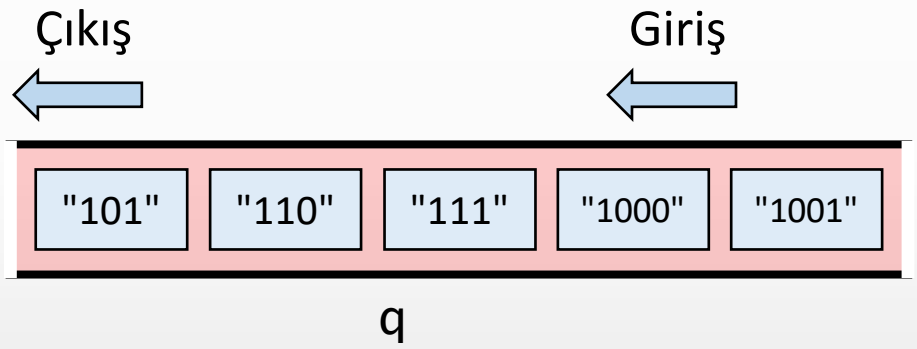
i = 4

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;
}
```

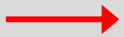




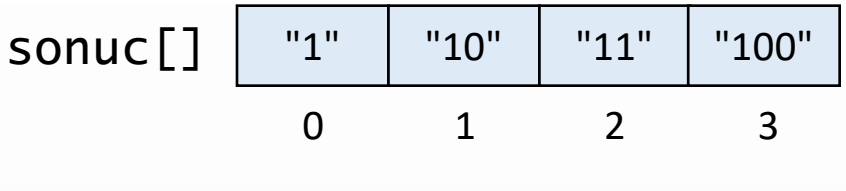
i = 4

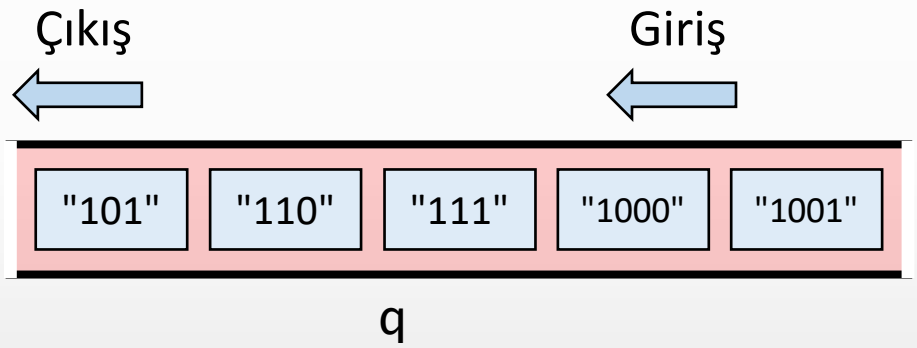
n = 4

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