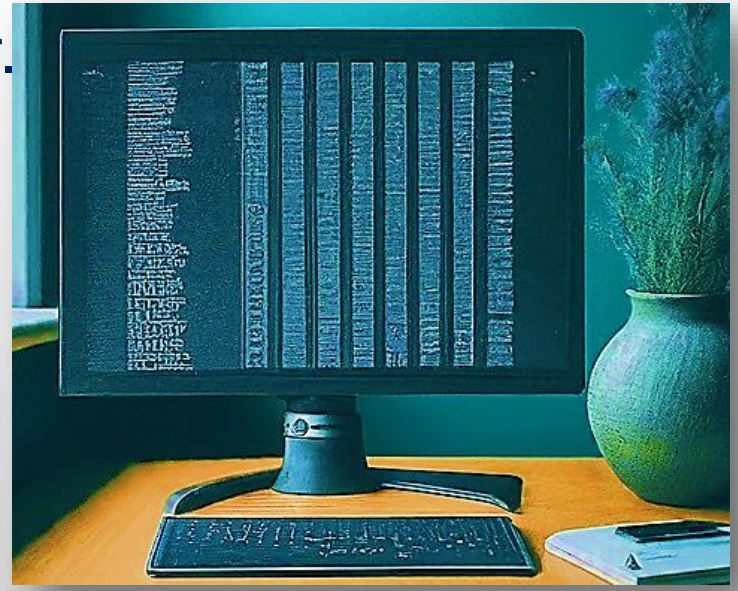
Bölüm 2: Sıralama Algoritmaları

Algoritmalar

Sıralama Algoritmaları

- Alışveriş listelerinden, sosyal medya gönderilerine kadar
  - dijital dünyada sürekli veriyle karşılaşıyoruz.
- Bu verileri etkin bir şekilde kullanabilmek için, düzenli tutmak gerekir.
- Sıralama algoritmaları,
  - Bir listedeki öğeleri belirli bir kritere göre düzenler.
  - Bu kriter,
    - sayısal değer,
    - alfabetik sıra veya
    - tarih olabilir.



Sıralama Algoritmalarının Çeşitleri

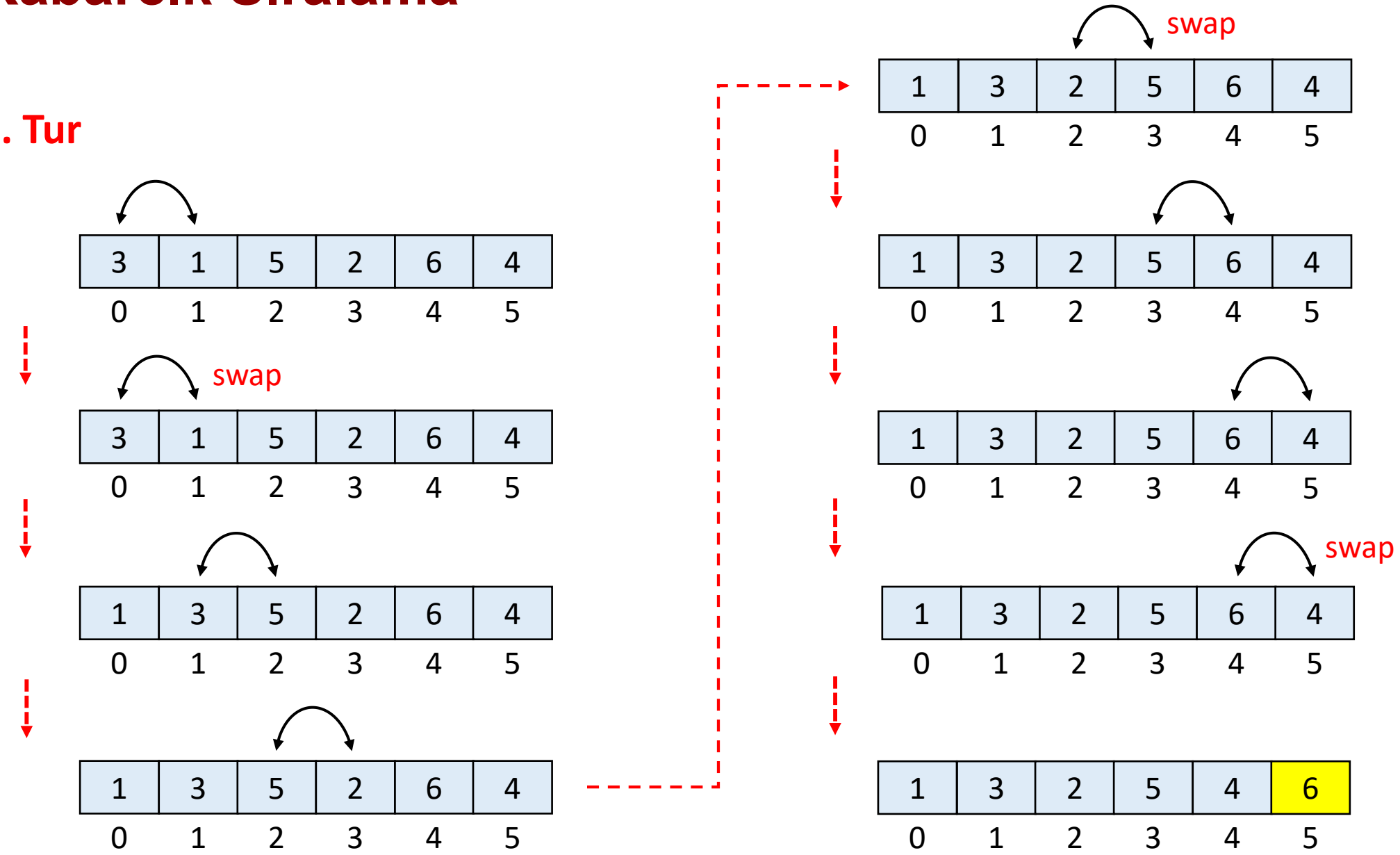
- Farklı sıralama algoritmaları, farklı çalışma prensiplerine sahiptir.
- Kabarcık Sıralama (*Bubble Sort*):
  - Verileri yan yana karşılaştırarak sıralar.
- Seçerek Sıralama (*Selection Sort*):
  - En küçük/büyük öğeyi bulup, sona/başa yerleştirir, sonra kalanı sıralar.
- Araya Ekleyerek Sıralama (*Insertion Sort*):
  - Elemanları doğru sıraya yerleştiriyormuş gibi sıralar.
- Birleştirerek Sıralama (*Merge Sort*):
  - Listeyi yarıya bölüp sıralar, sonra birleştirir.

Kabarcık Sıralama (Bubble Sort)

- Her bir adımda dizideki büyük elemanlar dizinin sonuna doğru kaydırılır.
  - Komşu elemanlar ikili olarak birbiriyle karşılaştırılır.
  - Sonuca göre elemanların yerleri değiştirilir.

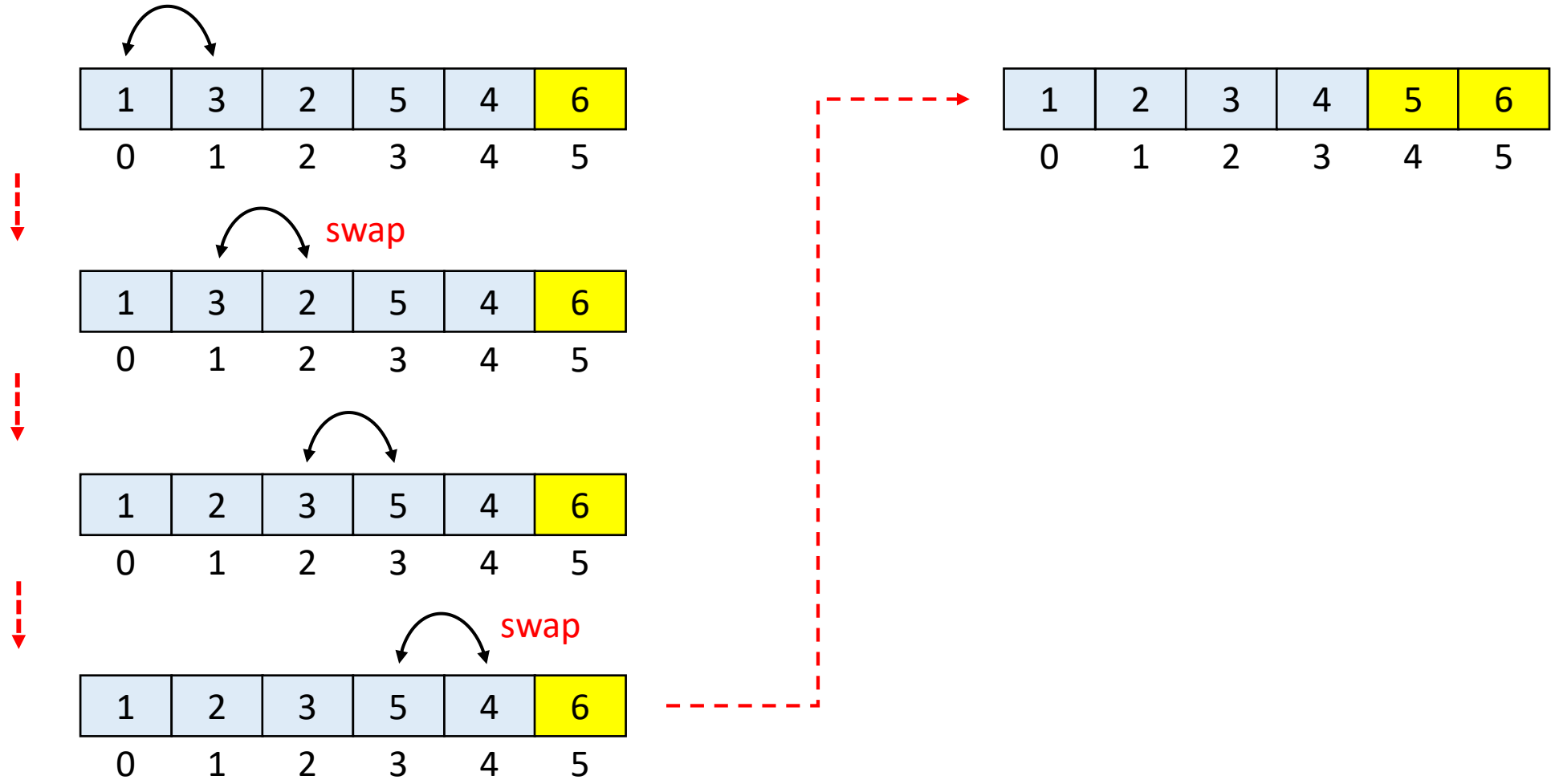
Kabarcık Sıralama

1. Tur



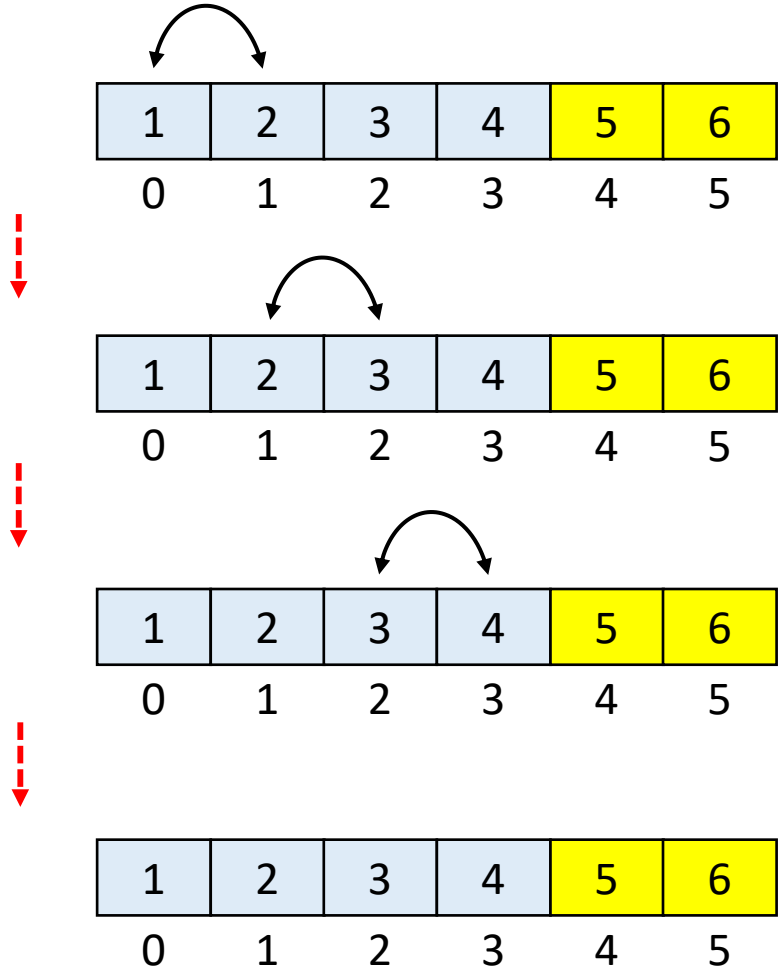
Kabarcık Sıralama

2. Tur



Kabarcık Sıralama

3. Tur



Kabarcık Sıralama

dizi[]	5	1	9	2	10
	0	1	2	3	4

```
boolean yerDegistiMi;
for(int i = 0; i < n - 1; i++) {
   yerDegistiMi = false;
   for(int j = 0; j < n - 1 - i; j++) {
      if(dizi[j] > dizi[j+1]) {
         int gecici = dizi[j];
         dizi[j] = dizi[j+1];
         dizi[j+1] = gecici;
         yerDegistiMi = true;
      }
   }
   if(yerDegistiMi == false) break;
}
```

Kabarcık Sıralama

dizi[]	5	1	9	2	10
	0	1	2	3	4

```
boolean yerDegistiMi;
for(int i = 0; i < n - 1; i++) {
   yerDegistiMi = false;
   for(int j = 0; j < n - 1 - i; j++) {
      if(dizi[j] > dizi[j+1]) {
         int gecici = dizi[j];
         dizi[j] = dizi[j+1];
         dizi[j+1] = gecici;
         yerDegistiMi = true;
      }
   }
   if(yerDegistiMi == false) break;
}
```

n = 5

Kabarcık Sıralama

dizi[]	5	1	9	2	10
	0	1	2	3	4

yerDegistiMi

n = 5

```
boolean yerDegistiMi;
for(int i = 0; i < n - 1; i++) {
   yerDegistiMi = false;
   for(int j = 0; j < n - 1 - i; j++) {
      if(dizi[j] > dizi[j+1]) {
         int gecici = dizi[j];
         dizi[j] = dizi[j+1];
         dizi[j+1] = gecici;
         yerDegistiMi = true;
      }
   }
   if(yerDegistiMi == false) break;
}
```

Kabarcık Sıralama

dizi[]	5	1	9	2	10
	0	1	2	3	4

yerDegistiMi
i = 0

n = 5

```
boolean yerDegistiMi;
for(int i = 0; i < n - 1; i++) {
   yerDegistiMi = false;
   for(int j = 0; j < n - 1 - i; j++) {
      if(dizi[j] > dizi[j+1]) {
         int gecici = dizi[j];
         dizi[j] = dizi[j+1];
         dizi[j+1] = gecici;
         yerDegistiMi = true;
      }
   }
   if(yerDegistiMi == false) break;
}
```

Kabarcık Sıralama

dizi[]	5	1	9	2	10
	0	1	2	3	4

```
yerDegistiMi = false
i = 0
```

```
n = 5
```

```
boolean yerDegistiMi;
for(int i = 0; i < n - 1; i++) {
   yerDegistiMi = false;
   for(int j = 0; j < n - 1 - i; j++) {
      if(dizi[j] > dizi[j+1]) {
         int gecici = dizi[j];
         dizi[j] = dizi[j+1];
         dizi[j+1] = gecici;
         yerDegistiMi = true;
      }
   }
   if(yerDegistiMi == false) break;
}
```

Kabarcık Sıralama

dizi[]	5	1	9	2	10
	0	1	2	3	4

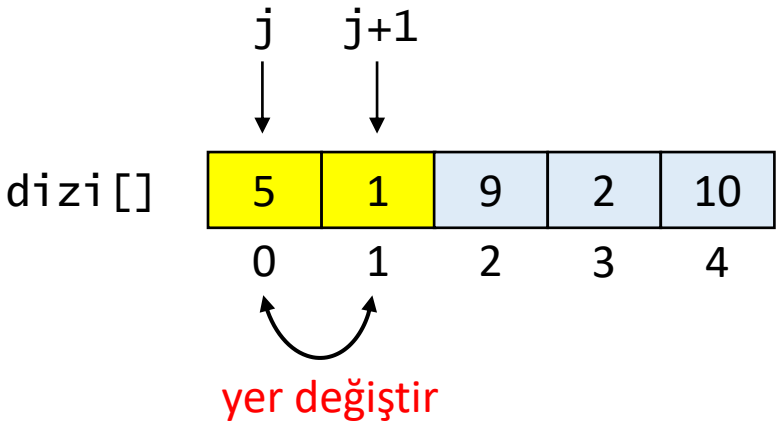
```
yerDegistiMi = false
i = 0
j = 0
```

```
n = 5
```



```
boolean yerDegistiMi;
for(int i = 0; i < n - 1; i++) {
   yerDegistiMi = false;
   for(int j = 0; j < n - 1 - i; j++) {
      if(dizi[j] > dizi[j+1]) {
         int gecici = dizi[j];
         dizi[j] = dizi[j+1];
         dizi[j+1] = gecici;
         yerDegistiMi = true;
      }
   }
   if(yerDegistiMi == false) break;
}
```

Kabarcık Sıralama

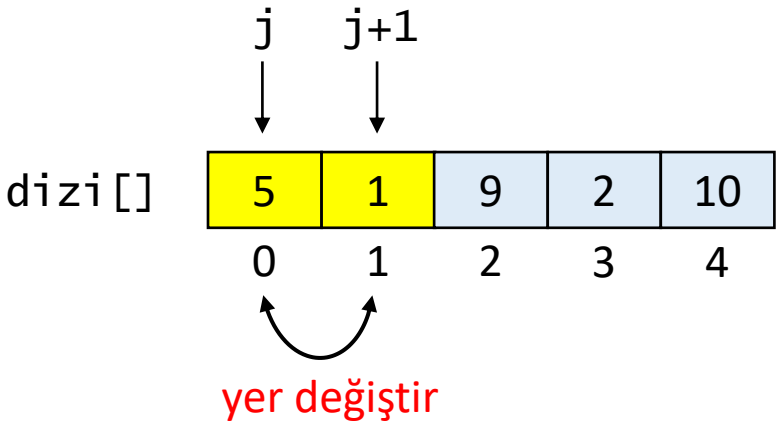


```
yerDegistiMi = false
i = 0
j = 0




n = 5
```

→

```
boolean yerDegistiMi;
for(int i = 0; i < n - 1; i++) {
   yerDegistiMi = false;
   for(int j = 0; j < n - 1 - i; j++) {
      if(dizi[j] > dizi[j+1]) {
         int gecici = dizi[j];
         dizi[j] = dizi[j+1];
         dizi[j+1] = gecici;
         yerDegistiMi = true;
      }
   }
   if(yerDegistiMi == false) break;
}
```

Kabarcık Sıralama



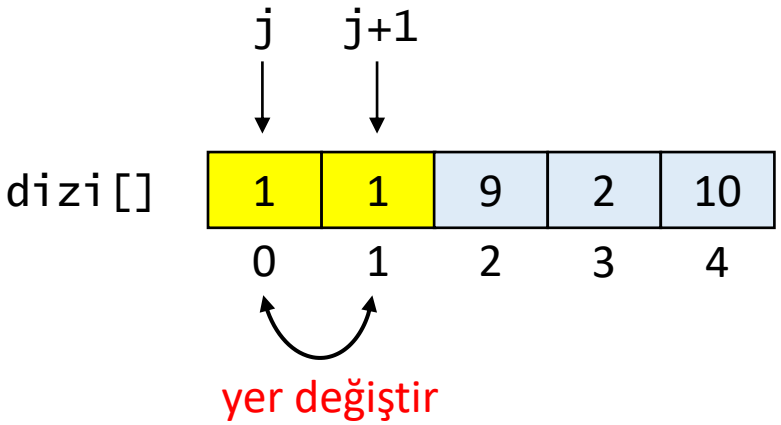
```
yerDegistiMi = false
i = 0
j = 0
gecici = 5
```

```
n = 5
```

→

```
boolean yerDegistiMi;
for(int i = 0; i < n - 1; i++) {
   yerDegistiMi = false;
   for(int j = 0; j < n - 1 - i; j++) {
      if(dizi[j] > dizi[j+1]) {
         int gecici = dizi[j];
         dizi[j] = dizi[j+1];
         dizi[j+1] = gecici;
         yerDegistiMi = true;
      }
   }
   if(yerDegistiMi == false) break;
}
```

Kabarcık Sıralama

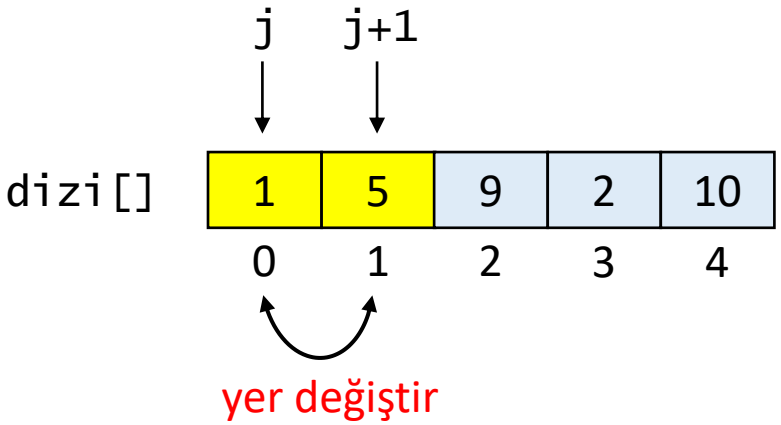


```
yerDegistiMi = false
i = 0
j = 0
gecici = 5
```

```
n = 5
```

```
boolean yerDegistiMi;
for(int i = 0; i < n - 1; i++) {
   yerDegistiMi = false;
   for(int j = 0; j < n - 1 - i; j++) {
      if(dizi[j] > dizi[j+1]) {
         int gecici = dizi[j];
         dizi[j] = dizi[j+1];
         dizi[j+1] = gecici;
         yerDegistiMi = true;
      }
   }
   if(yerDegistiMi == false) break;
}
```

Kabarcık Sıralama



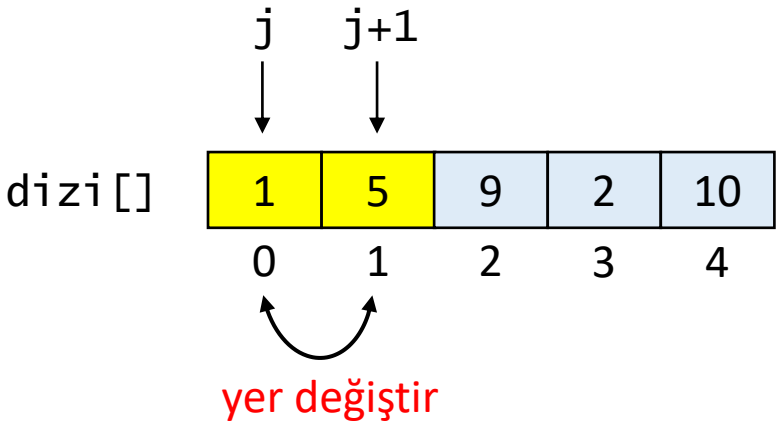
```
yerDegistiMi = false
i = 0
j = 0
gecici = 5
```

```
n = 5
```

→

```
boolean yerDegistiMi;
for(int i = 0; i < n - 1; i++) {
   yerDegistiMi = false;
   for(int j = 0; j < n - 1 - i; j++) {
      if(dizi[j] > dizi[j+1]) {
         int gecici = dizi[j];
         dizi[j] = dizi[j+1];
         dizi[j+1] = gecici;
         yerDegistiMi = true;
      }
   }
   if(yerDegistiMi == false) break;
}
```

Kabarcık Sıralama

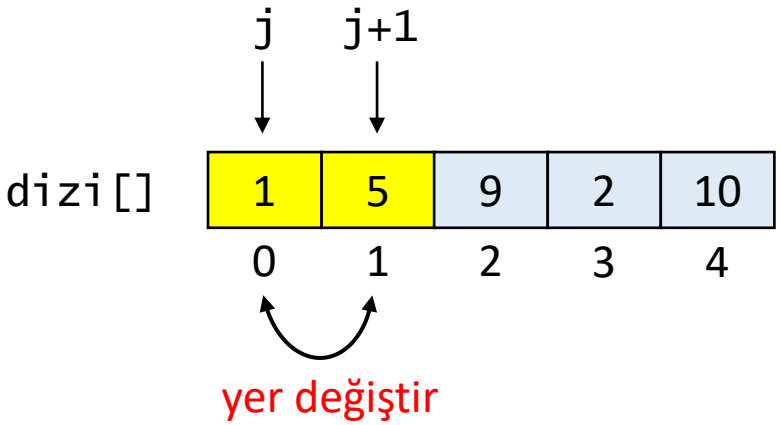


```
yerDegistiMi = true
i = 0
j = 0
gecici = 5
```

```
n = 5
```

```
boolean yerDegistiMi;
for(int i = 0; i < n - 1; i++) {
   yerDegistiMi = false;
   for(int j = 0; j < n - 1 - i; j++) {
      if(dizi[j] > dizi[j+1]) {
         int gecici = dizi[j];
         dizi[j] = dizi[j+1];
         dizi[j+1] = gecici;
         yerDegistiMi = true;
      }
   }
   if(yerDegistiMi == false) break;
}
```

Kabarcık Sıralama



```
yerDegistiMi = true
i = 0
j = 0
gecici = 5
```

```
n = 5
```

```
boolean yerDegistiMi;
for(int i = 0; i < n - 1; i++) {
   yerDegistiMi = false;
   for(int j = 0; j < n - 1 - i; j++) {
      if(dizi[j] > dizi[j+1]) {
         int gecici = dizi[j];
         dizi[j] = dizi[j+1];
         dizi[j+1] = gecici;
         yerDegistiMi = true;
      }
   }
   if(yerDegistiMi == false) break;
}
```

Kabarcık Sıralama

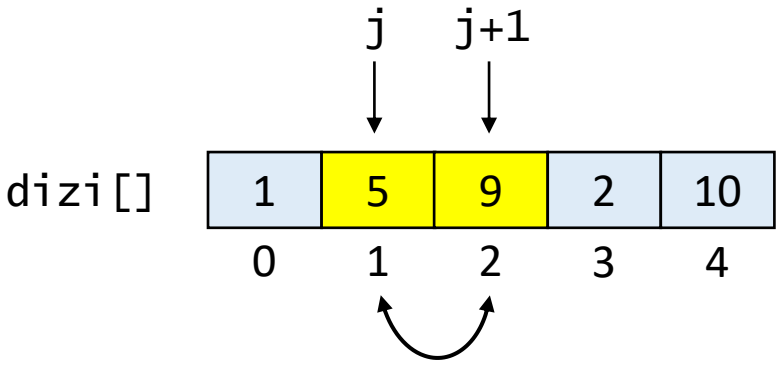
dizi[]	1	5	9	2	10
	0	1	2	3	4

```
yerDegistiMi = true
i = 0
j = 1
```

```
n = 5
```

```
boolean yerDegistiMi;
for(int i = 0; i < n - 1; i++) {
   yerDegistiMi = false;
   for(int j = 0; j < n - 1 - i; j++) {
      if(dizi[j] > dizi[j+1]) {
         int gecici = dizi[j];
         dizi[j] = dizi[j+1];
         dizi[j+1] = gecici;
         yerDegistiMi = true;
      }
   }
   if(yerDegistiMi == false) break;
}
```

Kabarcık Sıralama

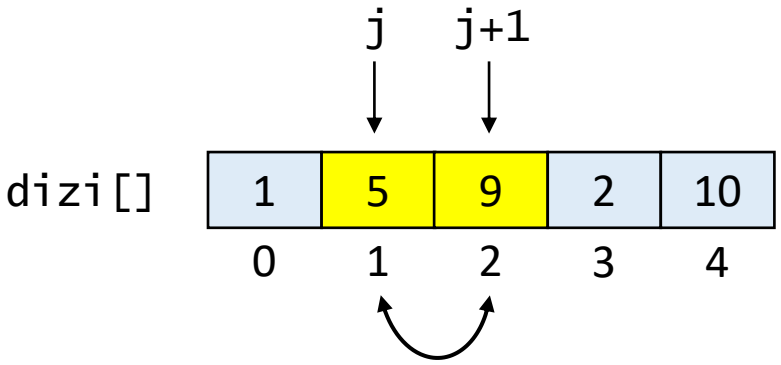


```
yerDegistiMi = true
i = 0
j = 1
```

```
n = 5
```

```
boolean yerDegistiMi;
for(int i = 0; i < n - 1; i++) {
   yerDegistiMi = false;
   for(int j = 0; j < n - 1 - i; j++) {
      if(dizi[j] > dizi[j+1]) {
         int gecici = dizi[j];
         dizi[j] = dizi[j+1];
         dizi[j+1] = gecici;
         yerDegistiMi = true;
      }
   }
   if(yerDegistiMi == false) break;
}
```

Kabarcık Sıralama



```
yerDegistiMi = true
i = 0
j = 1
```

```
n = 5
```

```
boolean yerDegistiMi;
for(int i = 0; i < n - 1; i++) {
   yerDegistiMi = false;
   for(int j = 0; j < n - 1 - i; j++) {
      if(dizi[j] > dizi[j+1]) {
         int gecici = dizi[j];
         dizi[j] = dizi[j+1];
         dizi[j+1] = gecici;
         yerDegistiMi = true;
      }
   }
   if(yerDegistiMi == false) break;
}
```

Kabarcık Sıralama

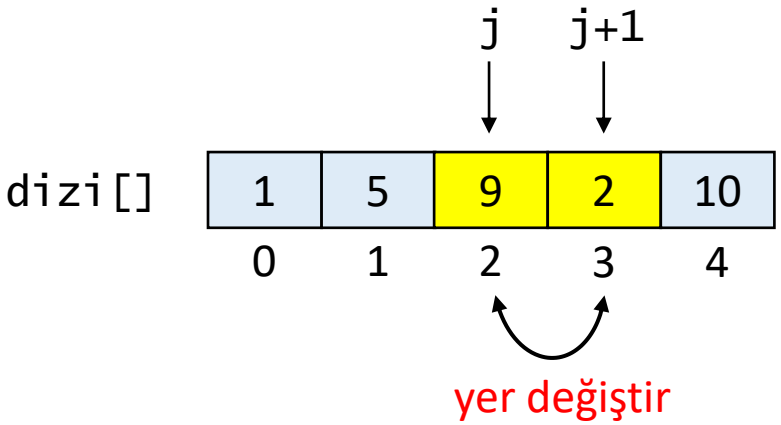
dizi[]	1	5	9	2	10
	0	1	2	3	4

```
yerDegistiMi = true
i = 0
j = 2
```

```
n = 5
```

```
boolean yerDegistiMi;
for(int i = 0; i < n - 1; i++) {
   yerDegistiMi = false;
   for(int j = 0; j < n - 1 - i; j++) {
      if(dizi[j] > dizi[j+1]) {
         int gecici = dizi[j];
         dizi[j] = dizi[j+1];
         dizi[j+1] = gecici;
         yerDegistiMi = true;
      }
   }
   if(yerDegistiMi == false) break;
}
```

Kabarcık Sıralama

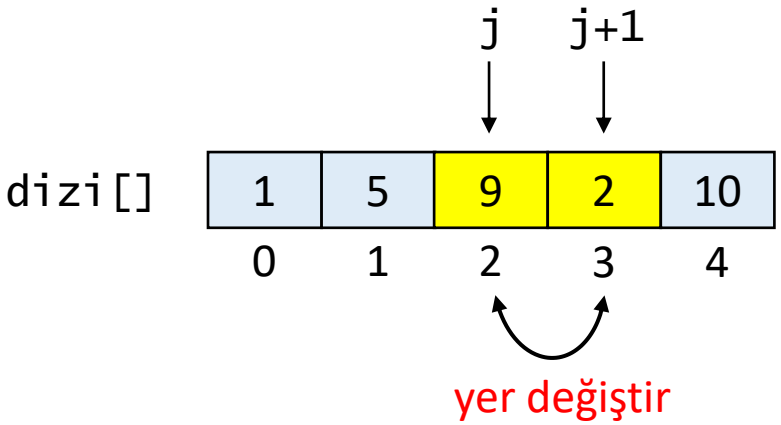


```
yerDegistiMi = true
i = 0
j = 2
```

```
n = 5
```

```
boolean yerDegistiMi;
for(int i = 0; i < n - 1; i++) {
   yerDegistiMi = false;
   for(int j = 0; j < n - 1 - i; j++) {
      if(dizi[j] > dizi[j+1]) {
         int gecici = dizi[j];
         dizi[j] = dizi[j+1];
         dizi[j+1] = gecici;
         yerDegistiMi = true;
      }
   }
   if(yerDegistiMi == false) break;
}
```

Kabarcık Sıralama

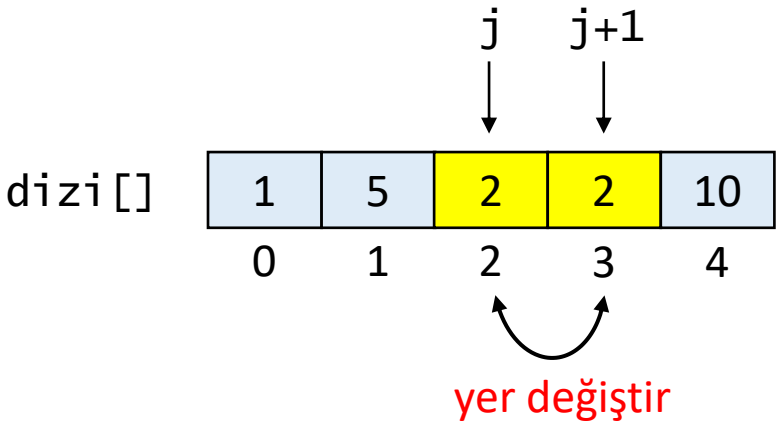


```
yerDegistiMi = true
i = 0
j = 2
gecici = 9
```

```
n = 5
```

```
boolean yerDegistiMi;
for(int i = 0; i < n - 1; i++) {
   yerDegistiMi = false;
   for(int j = 0; j < n - 1 - i; j++) {
      if(dizi[j] > dizi[j+1]) {
         int gecici = dizi[j];
         dizi[j] = dizi[j+1];
         dizi[j+1] = gecici;
         yerDegistiMi = true;
      }
   }
   if(yerDegistiMi == false) break;
}
```

Kabarcık Sıralama

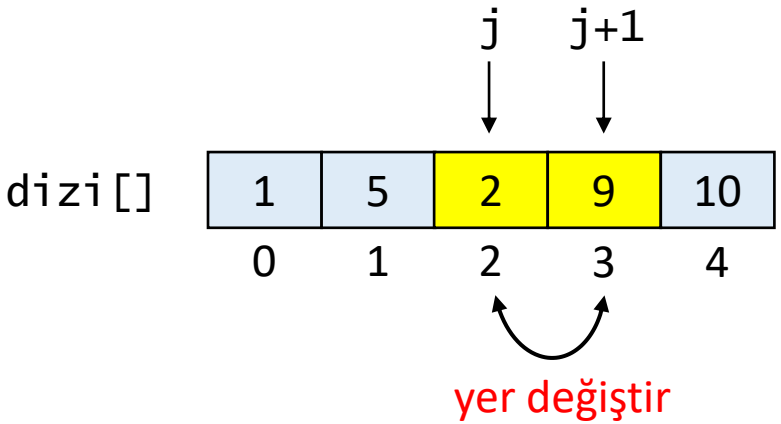


```
yerDegistiMi = true
i = 0
j = 2
gecici = 9
```

```
n = 5
```

```
boolean yerDegistiMi;
for(int i = 0; i < n - 1; i++) {
   yerDegistiMi = false;
   for(int j = 0; j < n - 1 - i; j++) {
      if(dizi[j] > dizi[j+1]) {
         int gecici = dizi[j];
         dizi[j] = dizi[j+1];
         dizi[j+1] = gecici;
         yerDegistiMi = true;
      }
   }
   if(yerDegistiMi == false) break;
}
```

Kabarcık Sıralama

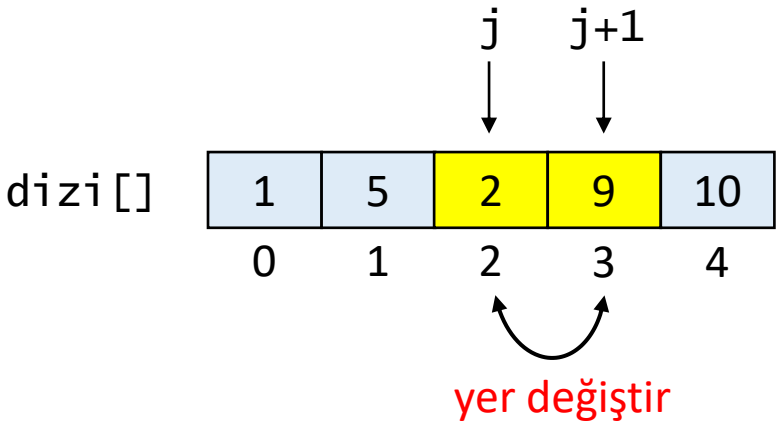


```
yerDegistiMi = true
i = 0
j = 2
gecici = 9
```

```
n = 5
```

```
boolean yerDegistiMi;
for(int i = 0; i < n - 1; i++) {
   yerDegistiMi = false;
   for(int j = 0; j < n - 1 - i; j++) {
      if(dizi[j] > dizi[j+1]) {
         int gecici = dizi[j];
         dizi[j] = dizi[j+1];
         dizi[j+1] = gecici;
         yerDegistiMi = true;
      }
   }
   if(yerDegistiMi == false) break;
}
```

Kabarcık Sıralama

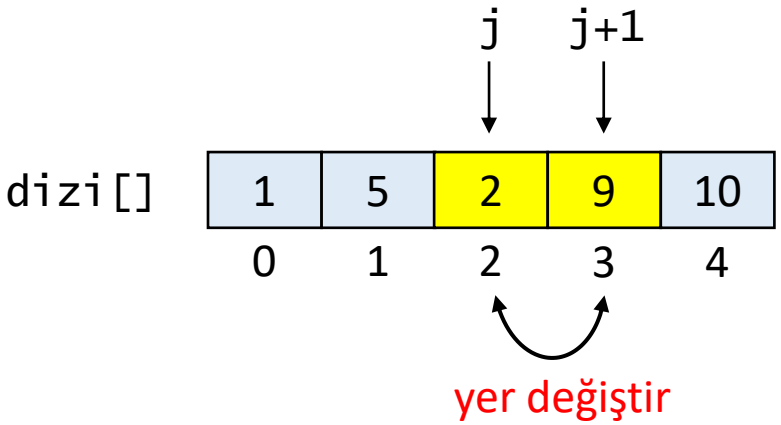


```
yerDegistiMi = true
i = 0
j = 2
gecici = 9
```

```
n = 5
```

```
boolean yerDegistiMi;
for(int i = 0; i < n - 1; i++) {
   yerDegistiMi = false;
   for(int j = 0; j < n - 1 - i; j++) {
      if(dizi[j] > dizi[j+1]) {
         int gecici = dizi[j];
         dizi[j] = dizi[j+1];
         dizi[j+1] = gecici;
         yerDegistiMi = true;
      }
   }
   if(yerDegistiMi == false) break;
}
```

Kabarcık Sıralama



```
yerDegistiMi = true
i = 0
j = 2
gecici = 9
```

```
n = 5
```

```
boolean yerDegistiMi;
for(int i = 0; i < n - 1; i++) {
   yerDegistiMi = false;
   for(int j = 0; j < n - 1 - i; j++) {
      if(dizi[j] > dizi[j+1]) {
         int gecici = dizi[j];
         dizi[j] = dizi[j+1];
         dizi[j+1] = gecici;
         yerDegistiMi = true;
      }
   }
   if(yerDegistiMi == false) break;
}
```

Kabarcık Sıralama

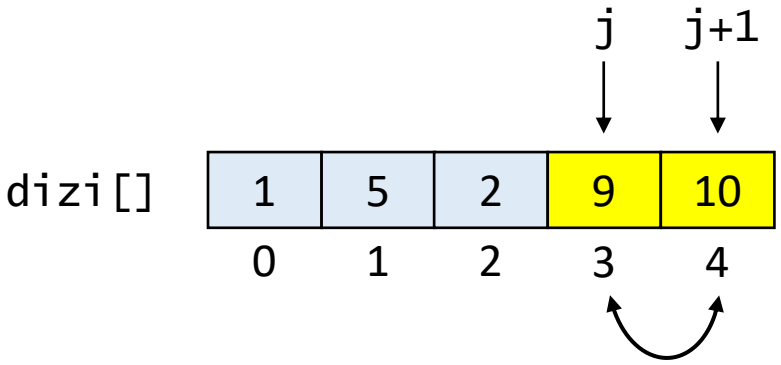
dizi[]	1	5	2	9	10
	0	1	2	3	4

```
yerDegistiMi = true
i = 0
j = 3
```

```
n = 5
```

```
boolean yerDegistiMi;
for(int i = 0; i < n - 1; i++) {
   yerDegistiMi = false;
   for(int j = 0; j < n - 1 - i; j++) {
      if(dizi[j] > dizi[j+1]) {
         int gecici = dizi[j];
         dizi[j] = dizi[j+1];
         dizi[j+1] = gecici;
         yerDegistiMi = true;
      }
   }
   if(yerDegistiMi == false) break;
}
```

Kabarcık Sıralama

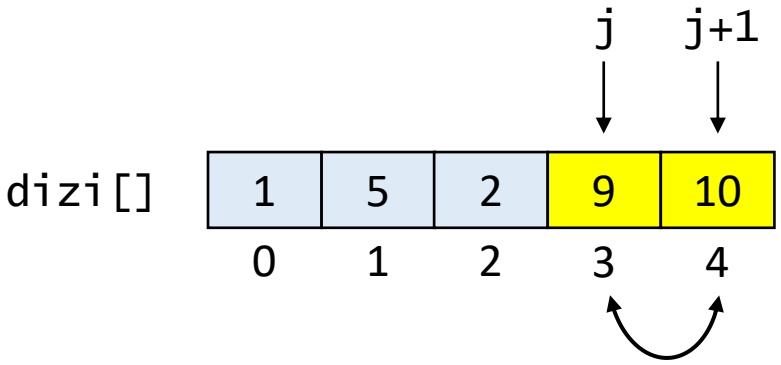


```
yerDegistiMi = true
i = 0
j = 3
```

```
n = 5
```

```
boolean yerDegistiMi;
for(int i = 0; i < n - 1; i++) {
   yerDegistiMi = false;
   for(int j = 0; j < n - 1 - i; j++) {
      if(dizi[j] > dizi[j+1]) {
         int gecici = dizi[j];
         dizi[j] = dizi[j+1];
         dizi[j+1] = gecici;
         yerDegistiMi = true;
      }
   }
   if(yerDegistiMi == false) break;
}
```

Kabarcık Sıralama



```
yerDegistiMi = true
i = 0
j = 3
```

```
n = 5
```

```
boolean yerDegistiMi;
for(int i = 0; i < n - 1; i++) {
   yerDegistiMi = false;
   for(int j = 0; j < n - 1 - i; j++) {
      if(dizi[j] > dizi[j+1]) {
         int gecici = dizi[j];
         dizi[j] = dizi[j+1];
         dizi[j+1] = gecici;
         yerDegistiMi = true;
      }
   }
   if(yerDegistiMi == false) break;
}
```

Kabarcık Sıralama

dizi[]	1	5	2	9	10
	0	1	2	3	4

```
yerDegistiMi = true
i = 0
j = 4
```

```
n = 5
```

```
boolean yerDegistiMi;
for(int i = 0; i < n - 1; i++) {
   yerDegistiMi = false;
   for(int j = 0; j < n - 1 - i; j++) {
      if(dizi[j] > dizi[j+1]) {
         int gecici = dizi[j];
         dizi[j] = dizi[j+1];
         dizi[j+1] = gecici;
         yerDegistiMi = true;
      }
   }
   if(yerDegistiMi == false) break;
}
```

Kabarcık Sıralama

dizi[]	1	5	2	9	10
	0	1	2	3	4

yerDegistiMi = true
i = 0

n = 5

```
boolean yerDegistiMi;
for(int i = 0; i < n - 1; i++) {
   yerDegistiMi = false;
   for(int j = 0; j < n - 1 - i; j++) {
      if(dizi[j] > dizi[j+1]) {
         int gecici = dizi[j];
         dizi[j] = dizi[j+1];
         dizi[j+1] = gecici;
         yerDegistiMi = true;
      }
   }
   if(yerDegistiMi == false) break;
}
```

Kabarcık Sıralama

dizi[]	1	5	2	9	10
	0	1	2	3	4

```
yerDegistiMi = true
i = 1
```

```
n = 5
```

```
boolean yerDegistiMi;
for(int i = 0; i < n - 1; i++) {
   yerDegistiMi = false;
   for(int j = 0; j < n - 1 - i; j++) {
      if(dizi[j] > dizi[j+1]) {
         int gecici = dizi[j];
         dizi[j] = dizi[j+1];
         dizi[j+1] = gecici;
         yerDegistiMi = true;
      }
   }
   if(yerDegistiMi == false) break;
}
```

Kabarcık Sıralama

dizi[]	1	5	2	9	10
	0	1	2	3	4

```
yerDegistiMi = false
i = 1
```

```
n = 5
```

```
boolean yerDegistiMi;
for(int i = 0; i < n - 1; i++) {
   yerDegistiMi = false;
   for(int j = 0; j < n - 1 - i; j++) {
      if(dizi[j] > dizi[j+1]) {
         int gecici = dizi[j];
         dizi[j] = dizi[j+1];
         dizi[j+1] = gecici;
         yerDegistiMi = true;
      }
   }
   if(yerDegistiMi == false) break;
}
```

Kabarcık Sıralama

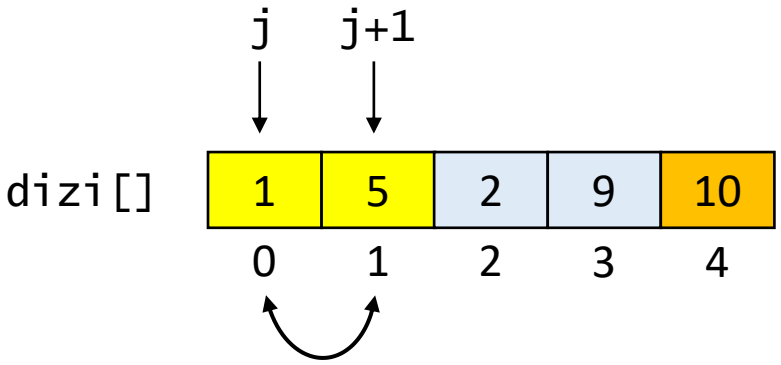
dizi[]	1	5	2	9	10
	0	1	2	3	4

```
yerDegistiMi = false
i = 1
j = 0
```

```
n = 5
```

```
boolean yerDegistiMi;
for(int i = 0; i < n - 1; i++) {
   yerDegistiMi = false;
   for(int j = 0; j < n - 1 - i; j++) {
      if(dizi[j] > dizi[j+1]) {
         int gecici = dizi[j];
         dizi[j] = dizi[j+1];
         dizi[j+1] = gecici;
         yerDegistiMi = true;
      }
   }
   if(yerDegistiMi == false) break;
}
```

Kabarcık Sıralama

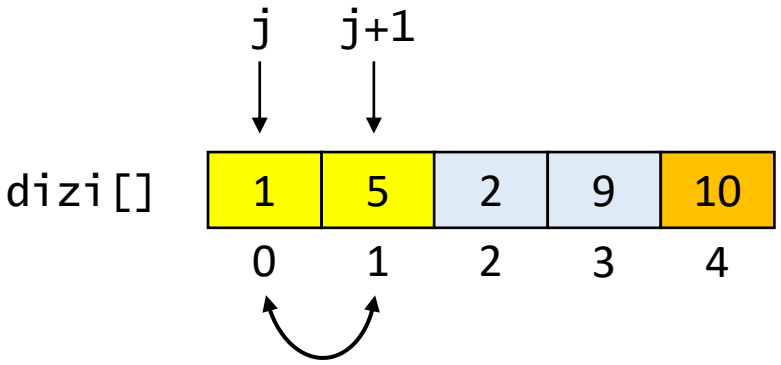


```
yerDegistiMi = false
i = 1
j = 0
```

```
n = 5
```

```
boolean yerDegistiMi;
for(int i = 0; i < n - 1; i++) {
   yerDegistiMi = false;
   for(int j = 0; j < n - 1 - i; j++) {
      if(dizi[j] > dizi[j+1]) {
         int gecici = dizi[j];
         dizi[j] = dizi[j+1];
         dizi[j+1] = gecici;
         yerDegistiMi = true;
      }
   }
   if(yerDegistiMi == false) break;
}
```

Kabarcık Sıralama



```
yerDegistiMi = false
i = 1
j = 0
```

```
n = 5
```

```
boolean yerDegistiMi;
for(int i = 0; i < n - 1; i++) {
   yerDegistiMi = false;
   for(int j = 0; j < n - 1 - i; j++) {
      if(dizi[j] > dizi[j+1]) {
         int gecici = dizi[j];
         dizi[j] = dizi[j+1];
         dizi[j+1] = gecici;
         yerDegistiMi = true;
      }
   }
   if(yerDegistiMi == false) break;
}
```

Kabarcık Sıralama

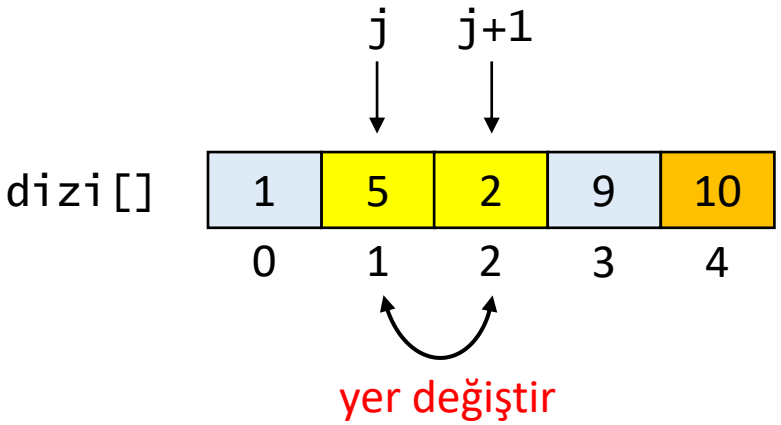
dizi[]	1	5	2	9	10
	0	1	2	3	4

```
yerDegistiMi = false
i = 1
j = 1
```

```
n = 5
```

```
boolean yerDegistiMi;
for(int i = 0; i < n - 1; i++) {
   yerDegistiMi = false;
   for(int j = 0; j < n - 1 - i; j++) {
      if(dizi[j] > dizi[j+1]) {
         int gecici = dizi[j];
         dizi[j] = dizi[j+1];
         dizi[j+1] = gecici;
         yerDegistiMi = true;
      }
   }
   if(yerDegistiMi == false) break;
}
```

Kabarcık Sıralama

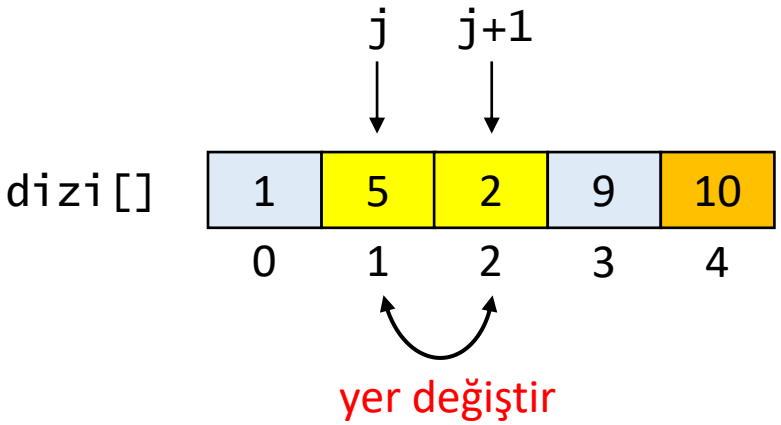


```
yerDegistiMi = false
i = 1
j = 1
```

```
n = 5
```

```
boolean yerDegistiMi;
for(int i = 0; i < n - 1; i++) {
   yerDegistiMi = false;
   for(int j = 0; j < n - 1 - i; j++) {
      if(dizi[j] > dizi[j+1]) {
         int gecici = dizi[j];
         dizi[j] = dizi[j+1];
         dizi[j+1] = gecici;
         yerDegistiMi = true;
      }
   }
   if(yerDegistiMi == false) break;
}
```

Kabarcık Sıralama

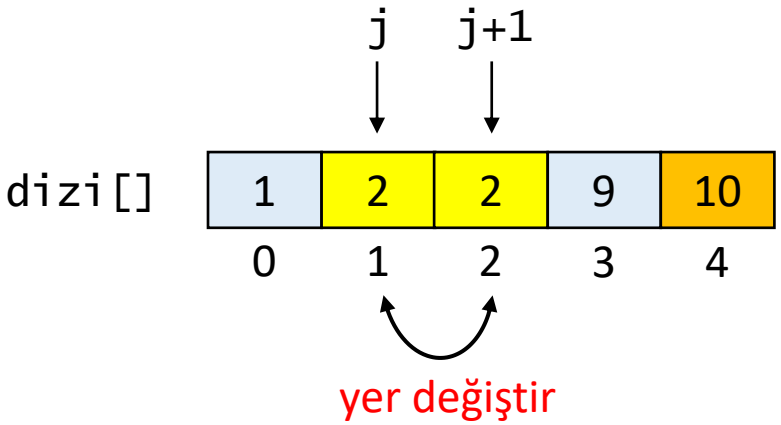


```
yerDegistiMi = false
i = 1
j = 1
gecici = 5
```

```
n = 5
```

```
boolean yerDegistiMi;
for(int i = 0; i < n - 1; i++) {
   yerDegistiMi = false;
   for(int j = 0; j < n - 1 - i; j++) {
      if(dizi[j] > dizi[j+1]) {
         int gecici = dizi[j];
         dizi[j] = dizi[j+1];
         dizi[j+1] = gecici;
         yerDegistiMi = true;
      }
   }
   if(yerDegistiMi == false) break;
}
```

Kabarcık Sıralama

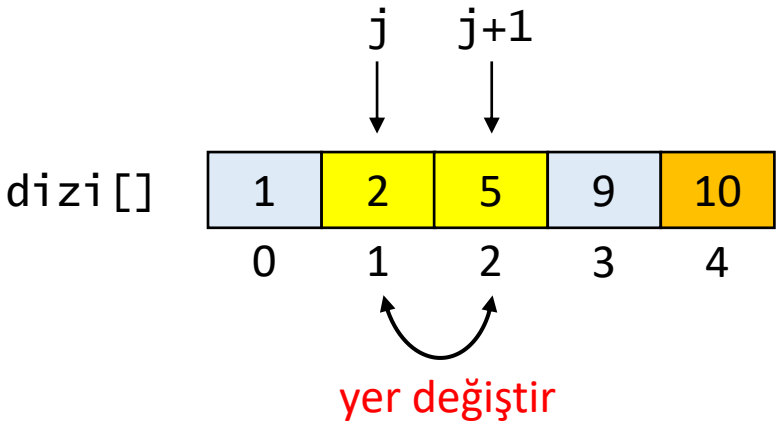


```
yerDegistiMi = false
i = 1
j = 1
gecici = 5
```

```
n = 5
```

```
boolean yerDegistiMi;
for(int i = 0; i < n - 1; i++) {
   yerDegistiMi = false;
   for(int j = 0; j < n - 1 - i; j++) {
      if(dizi[j] > dizi[j+1]) {
         int gecici = dizi[j];
         dizi[j] = dizi[j+1];
         dizi[j+1] = gecici;
         yerDegistiMi = true;
      }
   }
   if(yerDegistiMi == false) break;
}
```

Kabarcık Sıralama

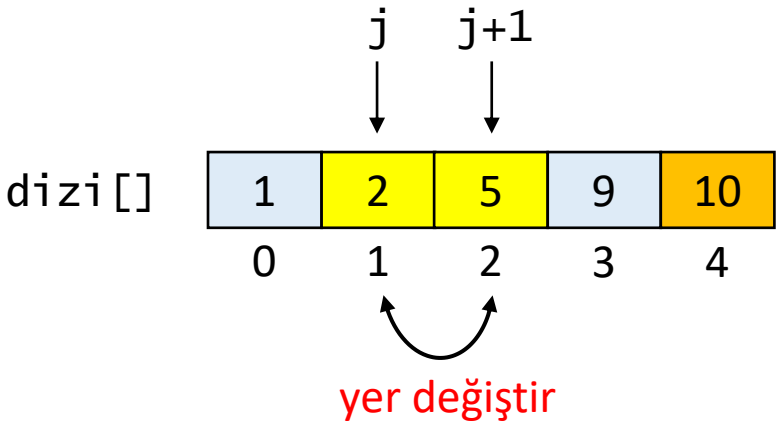


```
yerDegistiMi = false
i = 1
j = 1
gecici = 5
```

```
n = 5
```

```
boolean yerDegistiMi;
for(int i = 0; i < n - 1; i++) {
   yerDegistiMi = false;
   for(int j = 0; j < n - 1 - i; j++) {
      if(dizi[j] > dizi[j+1]) {
         int gecici = dizi[j];
         dizi[j] = dizi[j+1];
         dizi[j+1] = gecici;
         yerDegistiMi = true;
      }
   }
   if(yerDegistiMi == false) break;
}
```

Kabarcık Sıralama

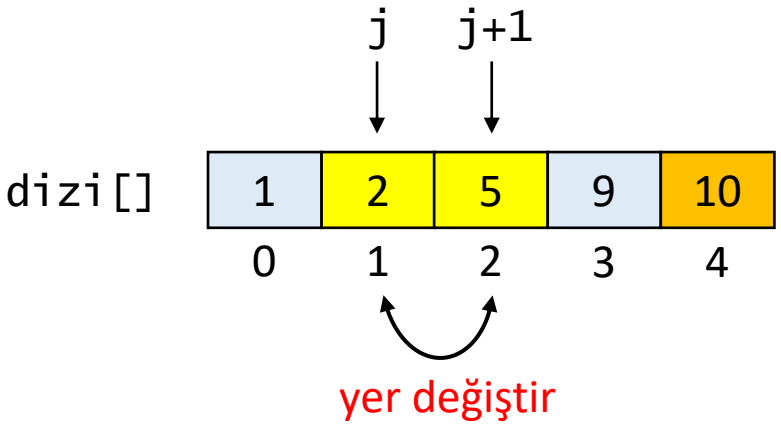


```
yerDegistiMi = true
i = 1
j = 1
gecici = 5
```

```
n = 5
```

```
boolean yerDegistiMi;
for(int i = 0; i < n - 1; i++) {
   yerDegistiMi = false;
   for(int j = 0; j < n - 1 - i; j++) {
      if(dizi[j] > dizi[j+1]) {
         int gecici = dizi[j];
         dizi[j] = dizi[j+1];
         dizi[j+1] = gecici;
         yerDegistiMi = true;
      }
   }
   if(yerDegistiMi == false) break;
}
```

Kabarcık Sıralama



```
yerDegistiMi = true
i = 1
j = 1
gecici = 5
```

```
n = 5
```

```
boolean yerDegistiMi;
for(int i = 0; i < n - 1; i++) {
   yerDegistiMi = false;
   for(int j = 0; j < n - 1 - i; j++) {
      if(dizi[j] > dizi[j+1]) {
         int gecici = dizi[j];
         dizi[j] = dizi[j+1];
         dizi[j+1] = gecici;
         yerDegistiMi = true;
      }
   }
   if(yerDegistiMi == false) break;
}
```

Kabarcık Sıralama

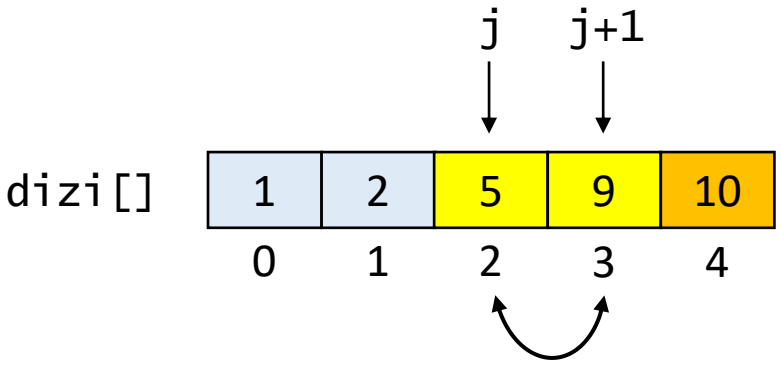
dizi[]	1	2	5	9	10
	0	1	2	3	4

```
yerDegistiMi = true
i = 1
j = 2
```

```
n = 5
```

```
boolean yerDegistiMi;
for(int i = 0; i < n - 1; i++) {
   yerDegistiMi = false;
   for(int j = 0; j < n - 1 - i; j++) {
      if(dizi[j] > dizi[j+1]) {
         int gecici = dizi[j];
         dizi[j] = dizi[j+1];
         dizi[j+1] = gecici;
         yerDegistiMi = true;
      }
   }
   if(yerDegistiMi == false) break;
}
```

Kabarcık Sıralama

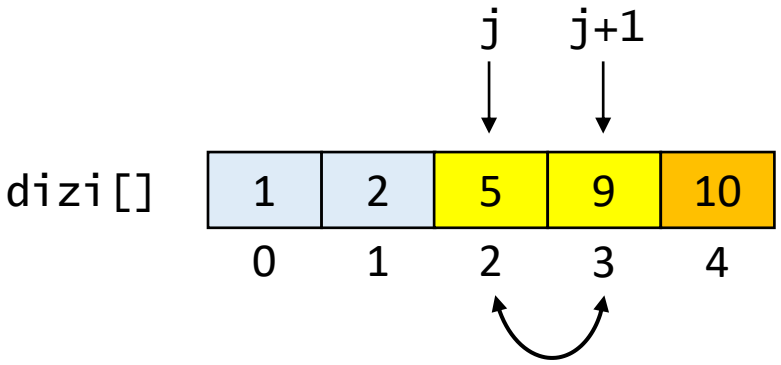


```
yerDegistiMi = true
i = 1
j = 2
```

```
n = 5
```

```
boolean yerDegistiMi;
for(int i = 0; i < n - 1; i++) {
   yerDegistiMi = false;
   for(int j = 0; j < n - 1 - i; j++) {
      if(dizi[j] > dizi[j+1]) {
         int gecici = dizi[j];
         dizi[j] = dizi[j+1];
         dizi[j+1] = gecici;
         yerDegistiMi = true;
      }
   }
   if(yerDegistiMi == false) break;
}
```

Kabarcık Sıralama



```
yerDegistiMi = true
i = 1
j = 2
```

```
n = 5
```

```
boolean yerDegistiMi;
for(int i = 0; i < n - 1; i++) {
   yerDegistiMi = false;
   for(int j = 0; j < n - 1 - i; j++) {
      if(dizi[j] > dizi[j+1]) {
         int gecici = dizi[j];
         dizi[j] = dizi[j+1];
         dizi[j+1] = gecici;
         yerDegistiMi = true;
      }
   }
   if(yerDegistiMi == false) break;
}
```

Kabarcık Sıralama

dizi[]	1	2	5	9	10
	0	1	2	3	4

```
yerDegistiMi = true
i = 1
j = 3
```

```
n = 5
```

```
boolean yerDegistiMi;
for(int i = 0; i < n - 1; i++) {
   yerDegistiMi = false;
   for(int j = 0; j < n - 1 - i; j++) {
      if(dizi[j] > dizi[j+1]) {
         int gecici = dizi[j];
         dizi[j] = dizi[j+1];
         dizi[j+1] = gecici;
         yerDegistiMi = true;
      }
   }
   if(yerDegistiMi == false) break;
}
```

Kabarcık Sıralama

dizi[]	1	2	5	9	10
	0	1	2	3	4

yerDegistiMi = true
i = 1

n = 5

```
boolean yerDegistiMi;
for(int i = 0; i < n - 1; i++) {
   yerDegistiMi = false;
   for(int j = 0; j < n - 1 - i; j++) {
      if(dizi[j] > dizi[j+1]) {
         int gecici = dizi[j];
         dizi[j] = dizi[j+1];
         dizi[j+1] = gecici;
         yerDegistiMi = true;
      }
   }
   if(yerDegistiMi == false) break;
}
```

Kabarcık Sıralama

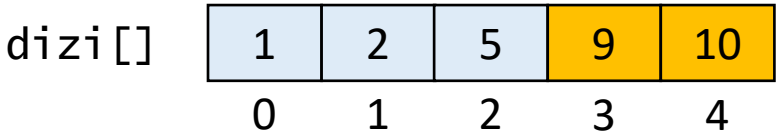
dizi[]	1	2	5	9	10
	0	1	2	3	4

```
yerDegistiMi = true
i = 2
```

```
n = 5
```

```
boolean yerDegistiMi;
for(int i = 0; i < n - 1; i++) {
   yerDegistiMi = false;
   for(int j = 0; j < n - 1 - i; j++) {
      if(dizi[j] > dizi[j+1]) {
         int gecici = dizi[j];
         dizi[j] = dizi[j+1];
         dizi[j+1] = gecici;
         yerDegistiMi = true;
      }
   }
   if(yerDegistiMi == false) break;
}
```

Kabarcık Sıralama




```
yerDegistiMi = false
i = 2
```

```
n = 5
```

```
boolean yerDegistiMi;
for(int i = 0; i < n - 1; i++) {
   yerDegistiMi = false;
   for(int j = 0; j < n - 1 - i; j++) {
      if(dizi[j] > dizi[j+1]) {
         int gecici = dizi[j];
         dizi[j] = dizi[j+1];
         dizi[j+1] = gecici;
         yerDegistiMi = true;
      }
   }
   if(yerDegistiMi == false) break;
}
```

Kabarcık Sıralama

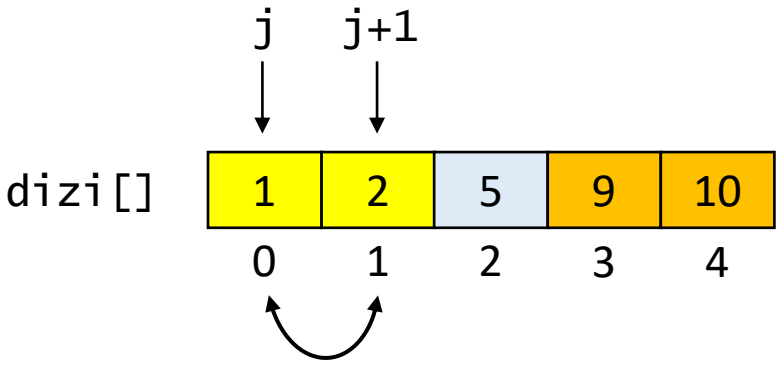
dizi[]	1	2	5	9	10
	0	1	2	3	4

```
yerDegistiMi = false
i = 2
j = 0
```

```
n = 5
```

```
boolean yerDegistiMi;
for(int i = 0; i < n - 1; i++) {
   yerDegistiMi = false;
   for(int j = 0; j < n - 1 - i; j++) {
      if(dizi[j] > dizi[j+1]) {
         int gecici = dizi[j];
         dizi[j] = dizi[j+1];
         dizi[j+1] = gecici;
         yerDegistiMi = true;
      }
   }
   if(yerDegistiMi == false) break;
}
```

Kabarcık Sıralama

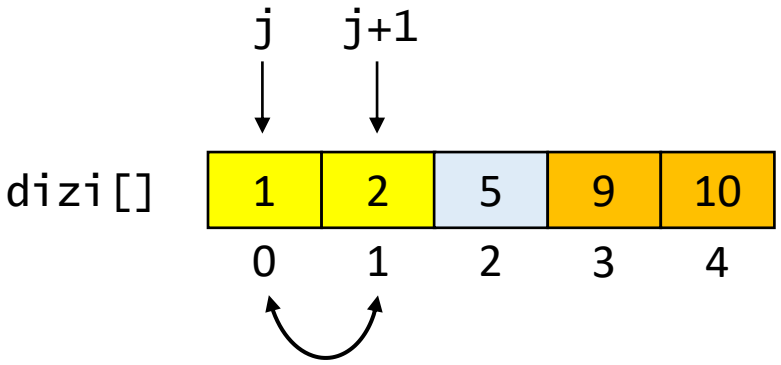


```
yerDegistiMi = false
i = 2
j = 0
```

```
n = 5
```

```
boolean yerDegistiMi;
for(int i = 0; i < n - 1; i++) {
   yerDegistiMi = false;
   for(int j = 0; j < n - 1 - i; j++) {
      if(dizi[j] > dizi[j+1]) {
         int gecici = dizi[j];
         dizi[j] = dizi[j+1];
         dizi[j+1] = gecici;
         yerDegistiMi = true;
      }
   }
   if(yerDegistiMi == false) break;
}
```

Kabarcık Sıralama



```
yerDegistiMi = false
i = 2
j = 0
```

```
n = 5
```

```
boolean yerDegistiMi;
for(int i = 0; i < n - 1; i++) {
   yerDegistiMi = false;
   for(int j = 0; j < n - 1 - i; j++) {
      if(dizi[j] > dizi[j+1]) {
         int gecici = dizi[j];
         dizi[j] = dizi[j+1];
         dizi[j+1] = gecici;
         yerDegistiMi = true;
      }
   }
   if(yerDegistiMi == false) break;
}
```

Kabarcık Sıralama

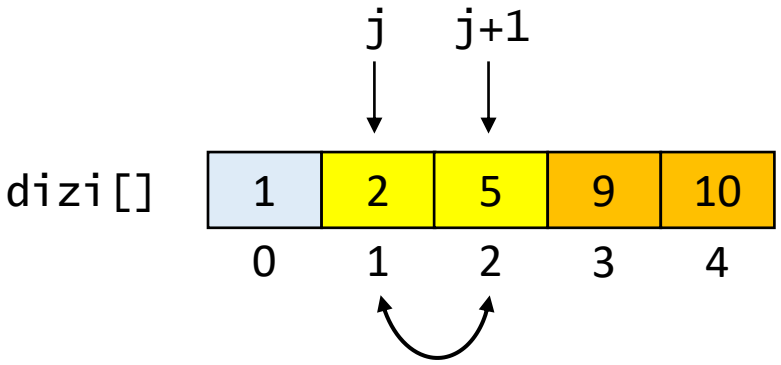
dizi[]	1	2	5	9	10
	0	1	2	3	4

```
yerDegistiMi = false
i = 2
j = 1
```

```
n = 5
```

```
boolean yerDegistiMi;
for(int i = 0; i < n - 1; i++) {
   yerDegistiMi = false;
   for(int j = 0; j < n - 1 - i; j++) {
      if(dizi[j] > dizi[j+1]) {
         int gecici = dizi[j];
         dizi[j] = dizi[j+1];
         dizi[j+1] = gecici;
         yerDegistiMi = true;
      }
   }
   if(yerDegistiMi == false) break;
}
```

Kabarcık Sıralama

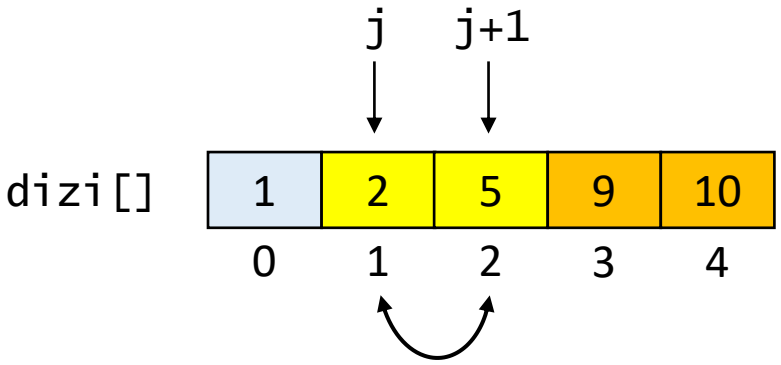


```
yerDegistiMi = false
i = 2
j = 1
```

```
n = 5
```

```
boolean yerDegistiMi;
for(int i = 0; i < n - 1; i++) {
   yerDegistiMi = false;
   for(int j = 0; j < n - 1 - i; j++) {
      if(dizi[j] > dizi[j+1]) {
         int gecici = dizi[j];
         dizi[j] = dizi[j+1];
         dizi[j+1] = gecici;
         yerDegistiMi = true;
      }
   }
   if(yerDegistiMi == false) break;
}
```

Kabarcık Sıralama



```
yerDegistiMi = false
i = 2
j = 1
```

```
n = 5
```

```
boolean yerDegistiMi;
for(int i = 0; i < n - 1; i++) {
   yerDegistiMi = false;
   for(int j = 0; j < n - 1 - i; j++) {
      if(dizi[j] > dizi[j+1]) {
         int gecici = dizi[j];
         dizi[j] = dizi[j+1];
         dizi[j+1] = gecici;
         yerDegistiMi = true;
      }
   }
   if(yerDegistiMi == false) break;
}
```

Kabarcık Sıralama

dizi[]	1	2	5	9	10
	0	1	2	3	4

```
yerDegistiMi = false
i = 2
j = 2
```

```
n = 5
```

```
boolean yerDegistiMi;
for(int i = 0; i < n - 1; i++) {
   yerDegistiMi = false;
   for(int j = 0; j < n - 1 - i; j++) {
      if(dizi[j] > dizi[j+1]) {
         int gecici = dizi[j];
         dizi[j] = dizi[j+1];
         dizi[j+1] = gecici;
         yerDegistiMi = true;
      }
   }
   if(yerDegistiMi == false) break;
}
```

Kabarcık Sıralama

dizi[]	1	2	5	9	10
	0	1	2	3	4

```
yerDegistiMi = false
i = 2
```

```
n = 5
```

```
boolean yerDegistiMi;
for(int i = 0; i < n - 1; i++) {
   yerDegistiMi = false;
   for(int j = 0; j < n - 1 - i; j++) {
      if(dizi[j] > dizi[j+1]) {
         int gecici = dizi[j];
         dizi[j] = dizi[j+1];
         dizi[j+1] = gecici;
         yerDegistiMi = true;
      }
   }
   if(yerDegistiMi == false) break;
}
```

Kabarcık Sıralama

dizi[]	1	2	5	9	10
	0	1	2	3	4

```
boolean yerDegistiMi;
for(int i = 0; i < n - 1; i++) {
   yerDegistiMi = false;
   for(int j = 0; j < n - 1 - i; j++) {
      if(dizi[j] > dizi[j+1]) {
         int gecici = dizi[j];
         dizi[j] = dizi[j+1];
         dizi[j+1] = gecici;
         yerDegistiMi = true;
      }
   }
   if(yerDegistiMi == false) break;
}
```

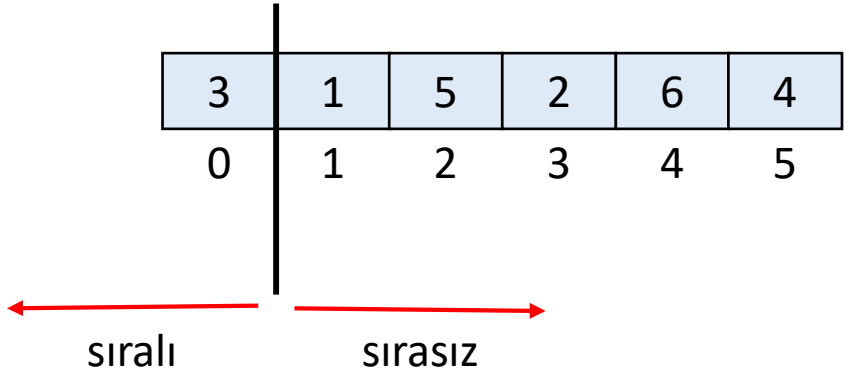
Eklemeli Sıralama (Insertion Sort)

- Eldeki oyun kartlarını sıralamaya benzer.
- Verilen dizi iki parçaya ayrılır:
  - Sıralı kısım
  - Sırasız kısım
- Sırasız kısımdan ilk eleman, sıralı kısımda doğru konuma yerleştirilir.
- Sırasız kısımda bulunan diğer elemanların kaydırılmasını gerektirir.

Eklemeli Sıralama

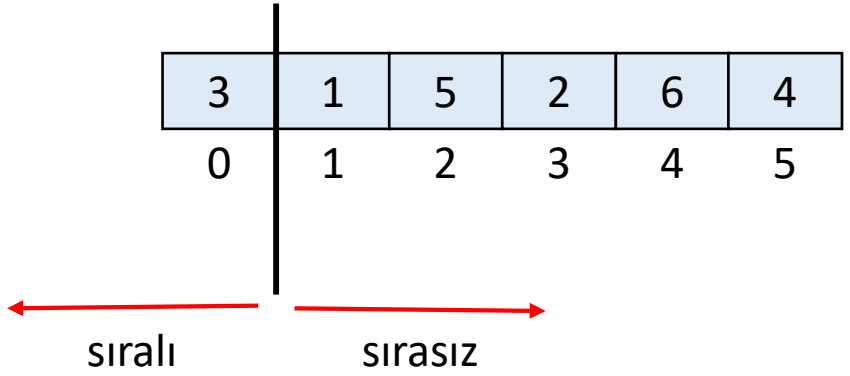
3	1	5	2	6	4
0	1	2	3	4	5

Eklemeli Sıralama



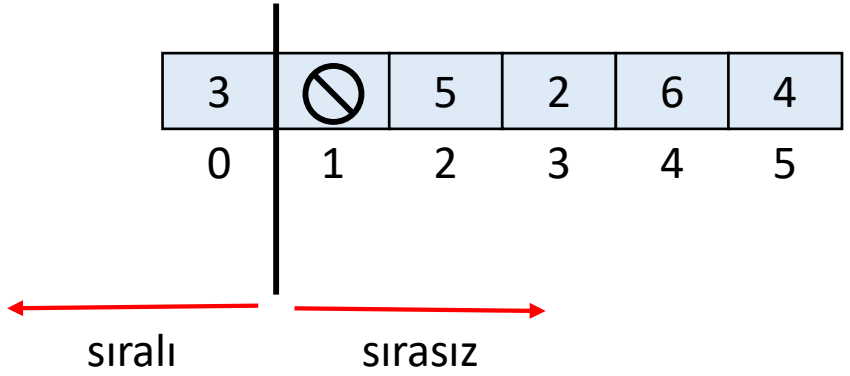
Eklemeli Sıralama

```
gecici = 1
```

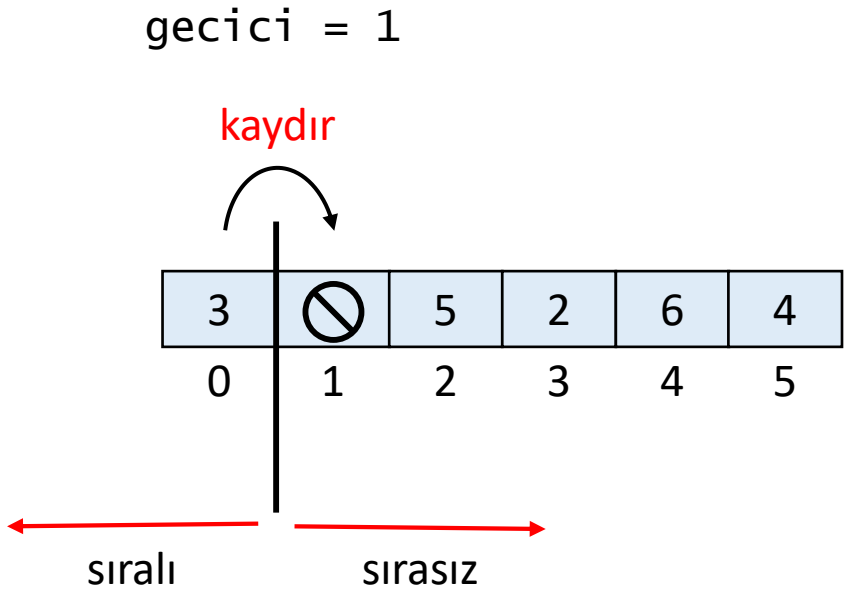


Eklemeli Sıralama

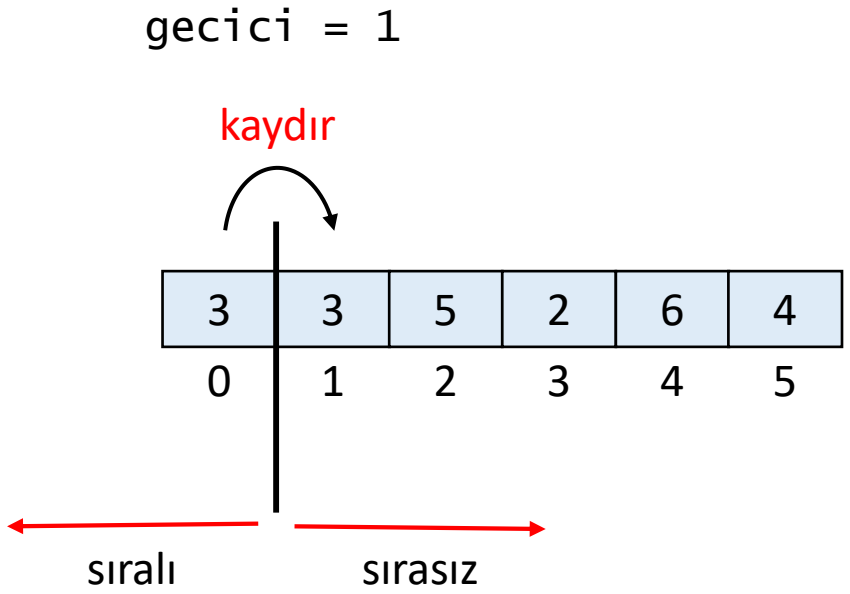
```
gecici = 1
```



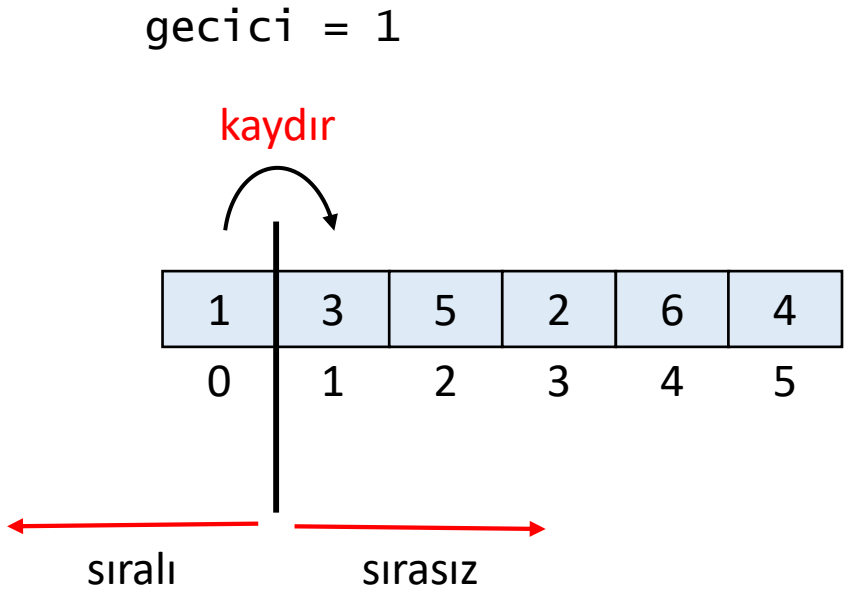
Eklemeli Sıralama



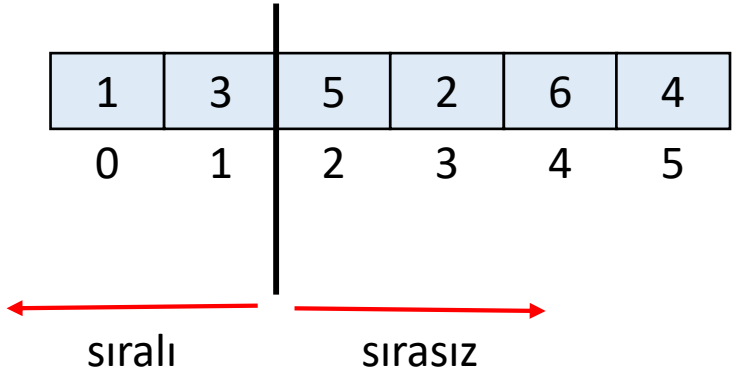
Eklemeli Sıralama



Eklemeli Sıralama

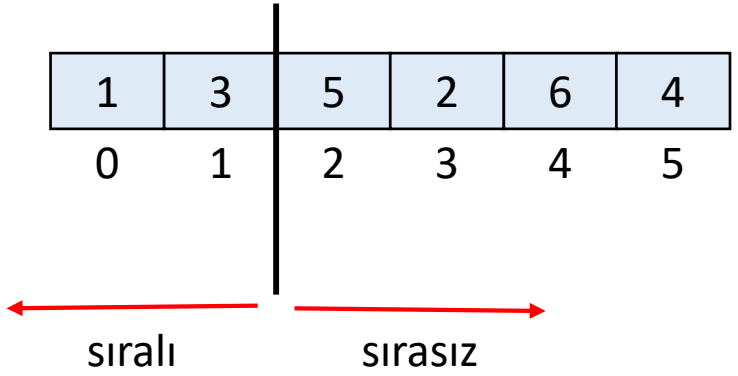


Eklemeli Sıralama



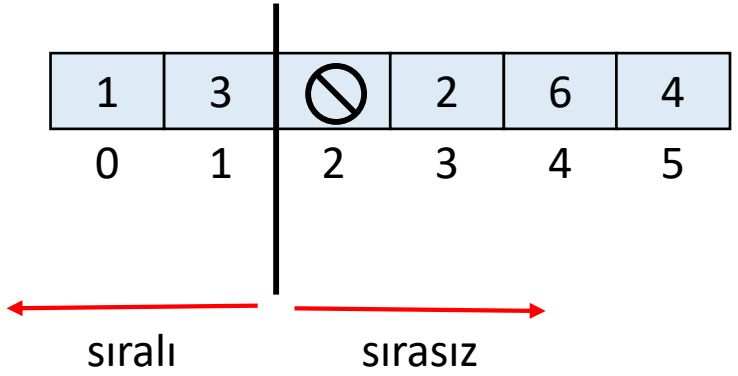
Eklemeli Sıralama

```
gecici = 5
```



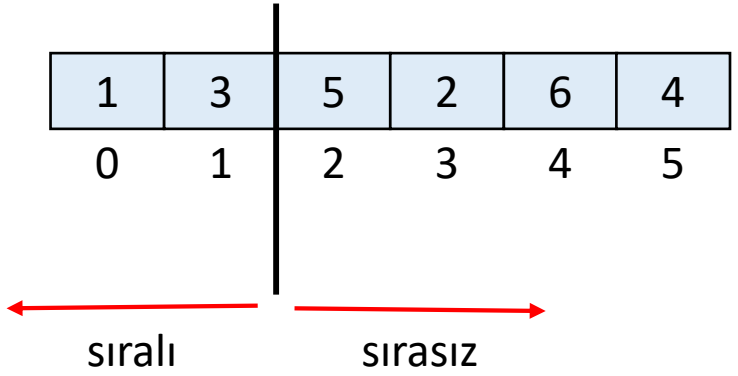
Eklemeli Sıralama

```
gecici = 5
```

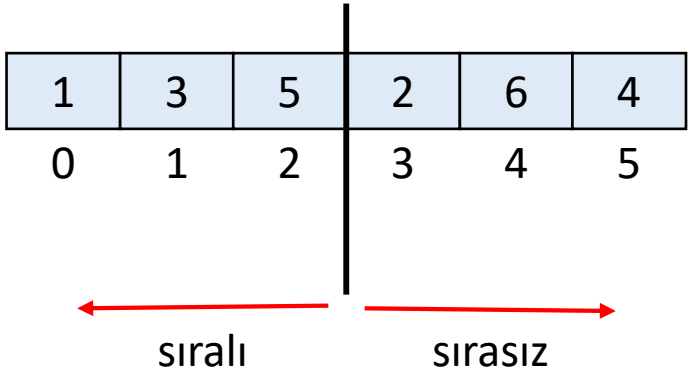


Eklemeli Sıralama

```
gecici = 5
```

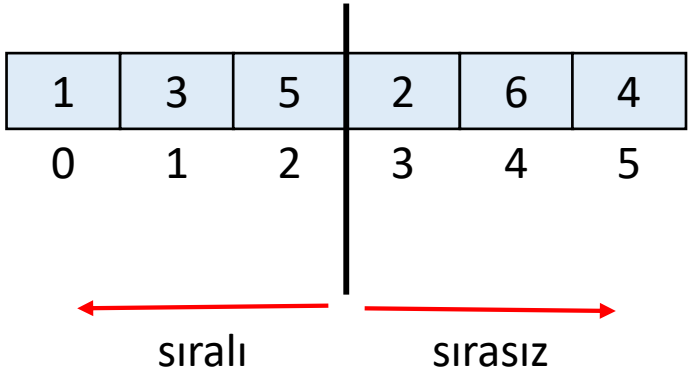


Eklemeli Sıralama



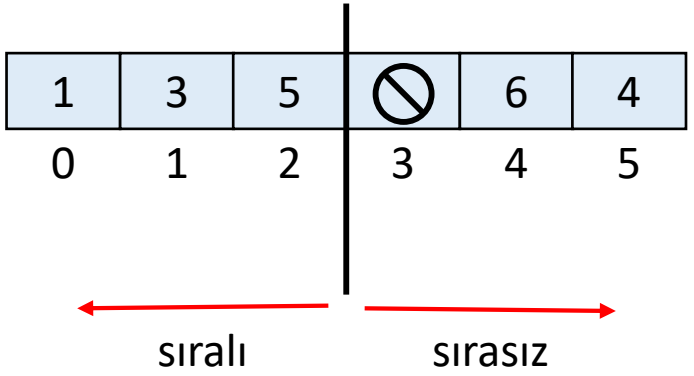
Eklemeli Sıralama

```
gecici = 2
```

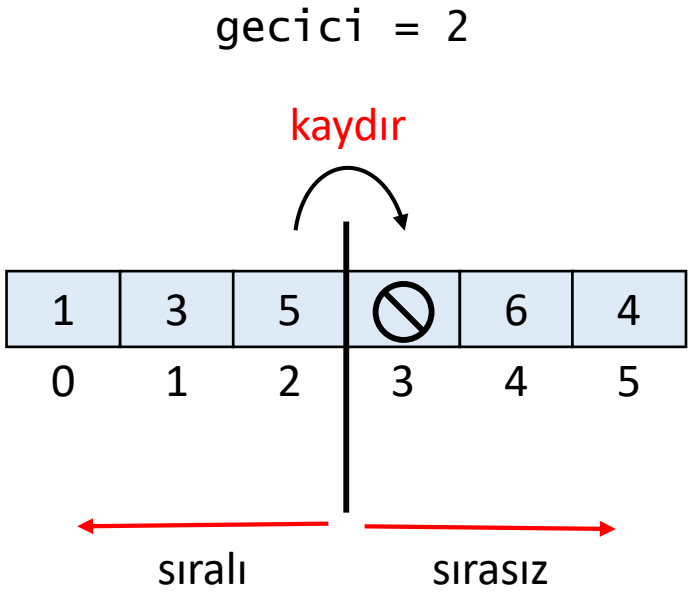


Eklemeli Sıralama

```
gecici = 2
```

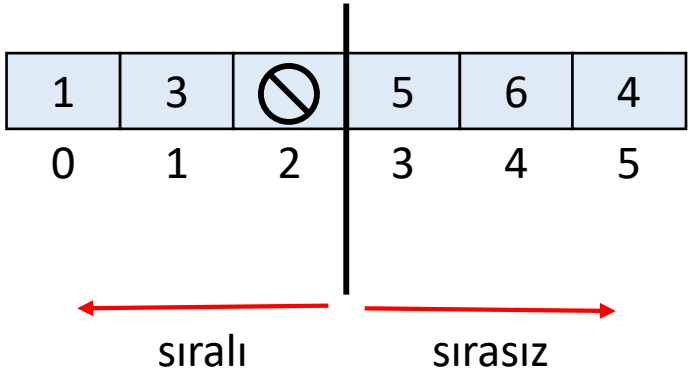


Eklemeli Sıralama

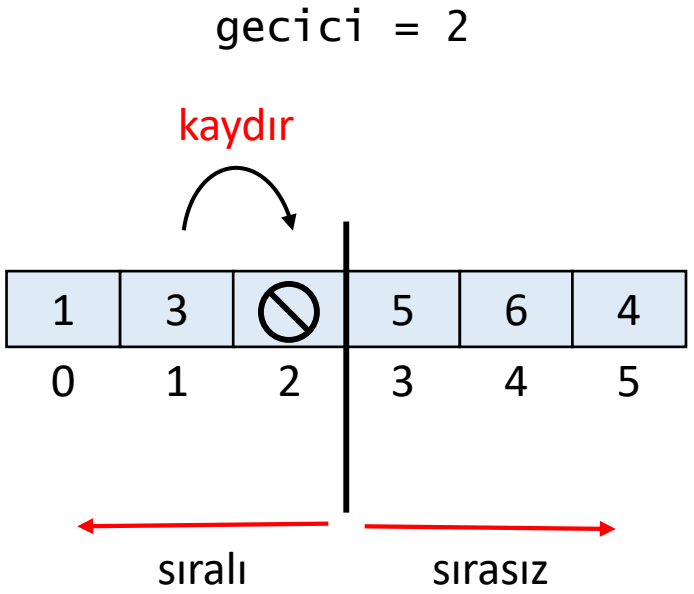


Eklemeli Sıralama

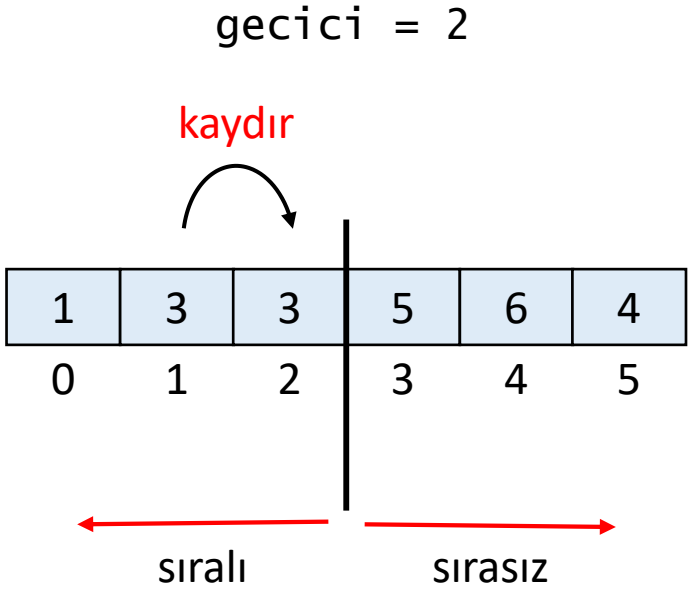
```
gecici = 2
```



Eklemeli Sıralama

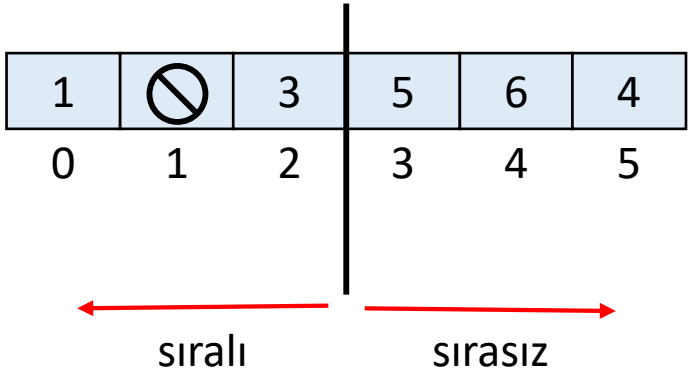


Eklemeli Sıralama



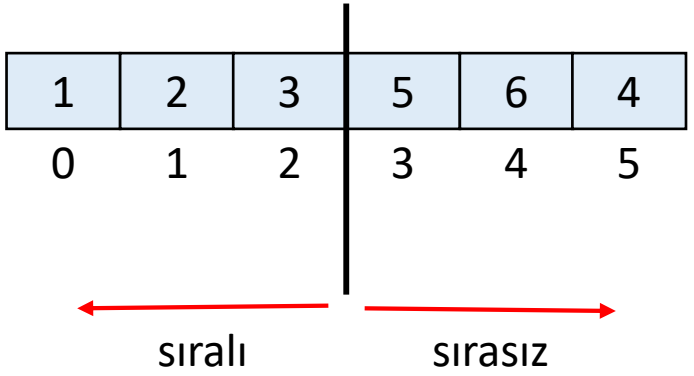
Eklemeli Sıralama

```
gecici = 2
```

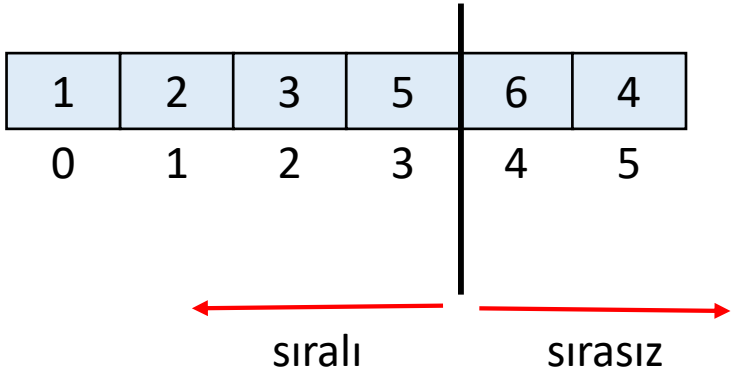


Eklemeli Sıralama

```
gecici = 2
```

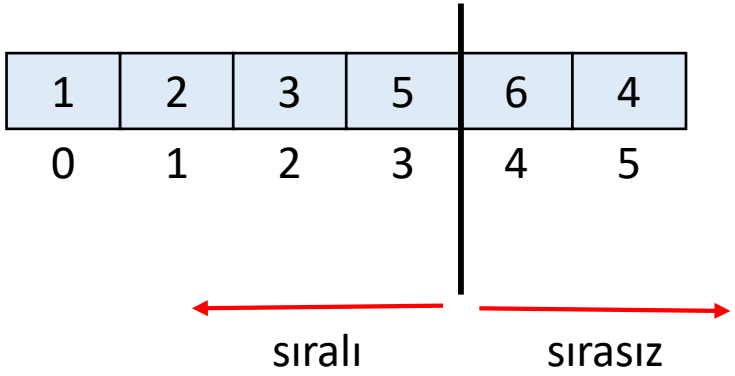


Eklemeli Sıralama



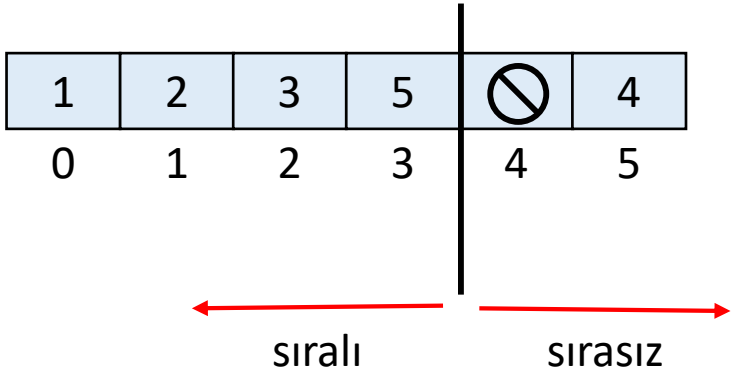
Eklemeli Sıralama

```
gecici = 6
```



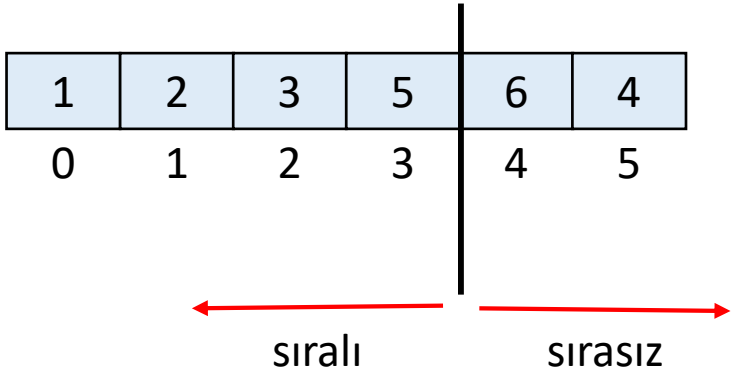
Eklemeli Sıralama

```
gecici = 6
```

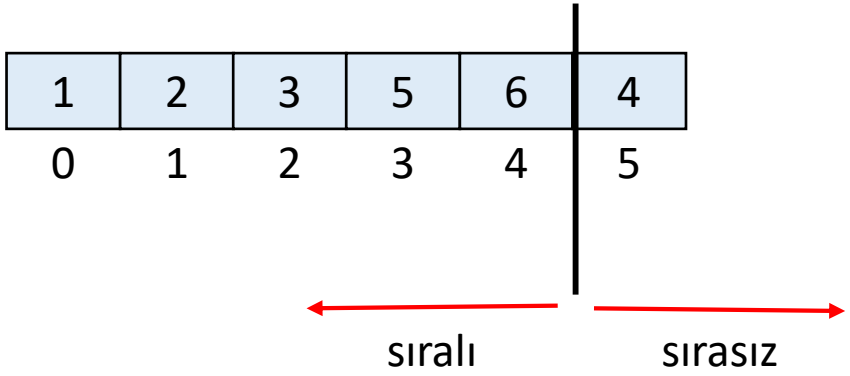


Eklemeli Sıralama

```
gecici = 6
```

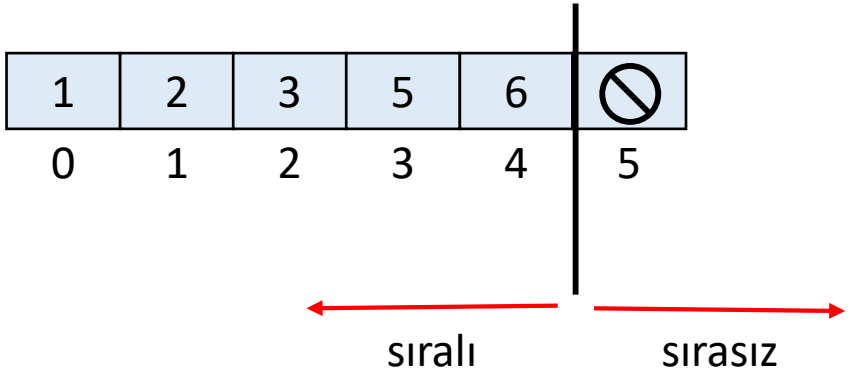


Eklemeli Sıralama

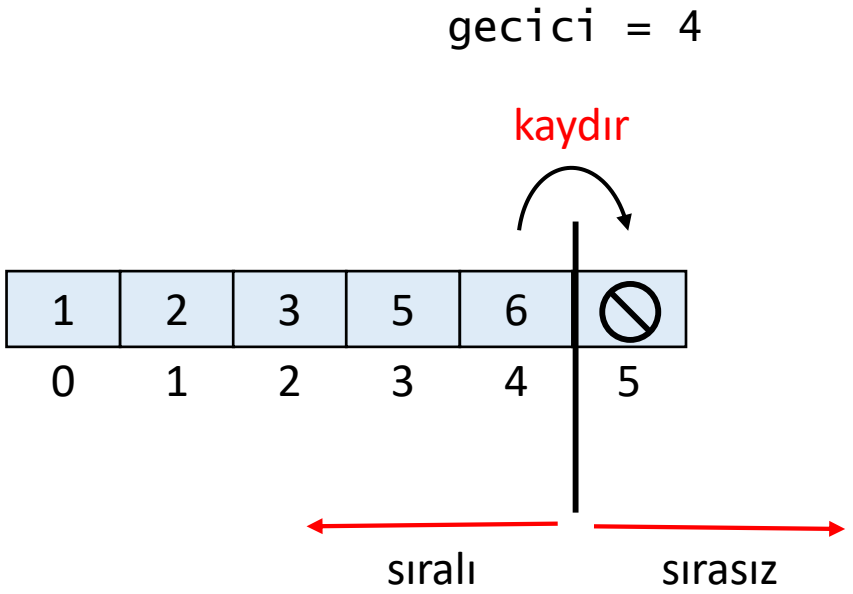


Eklemeli Sıralama

```
gecici = 4
```

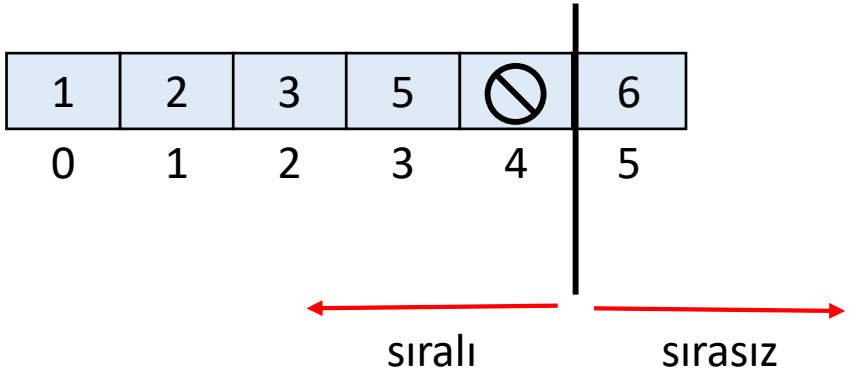


Eklemeli Sıralama

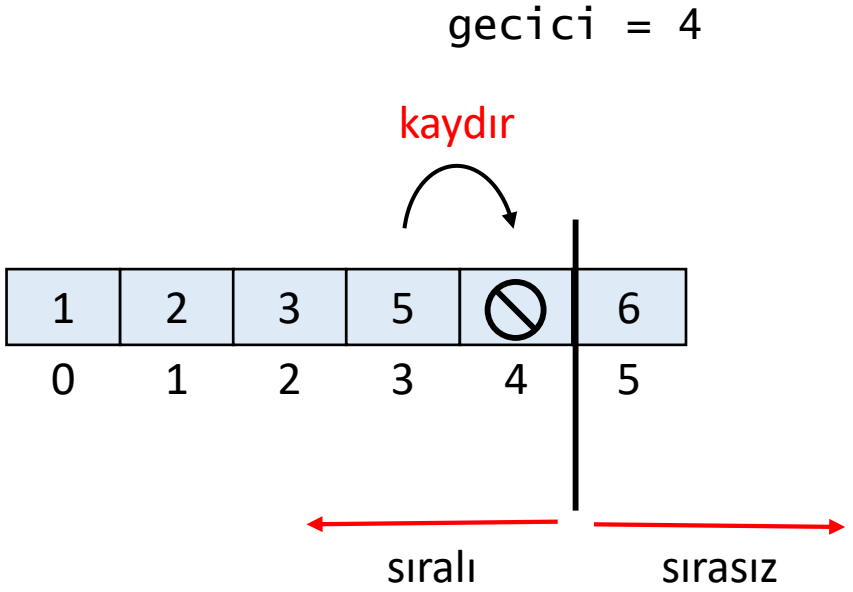


Eklemeli Sıralama

```
gecici = 4
```

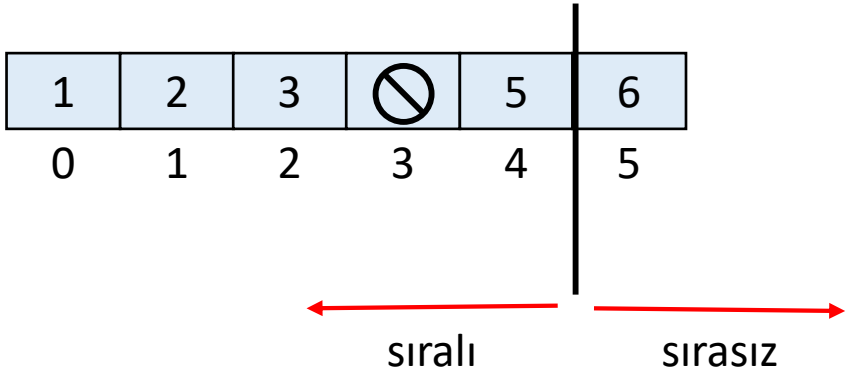


Eklemeli Sıralama



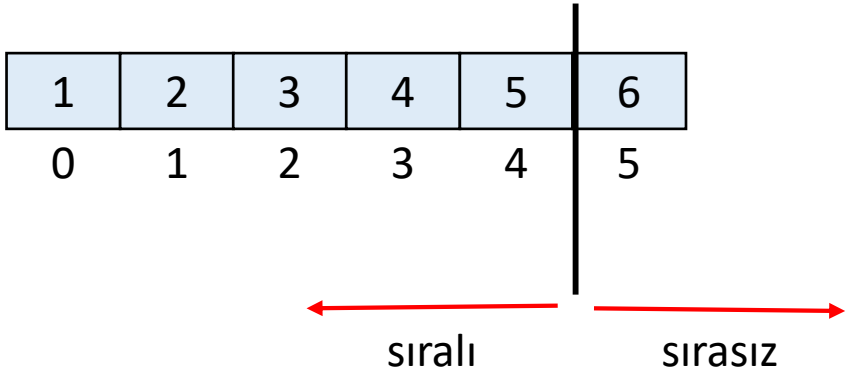
Eklemeli Sıralama

```
gecici = 4
```



Eklemeli Sıralama

```
gecici = 4
```



Eklemeli Sıralama

1	2	3	4	5	6
0	1	2	3	4	5

Eklemeli Sıralama

dizi[]	5	1	9	2	10
	0	1	2	3	4

```
for(int i = 1; i < n; i++) {
   int gecici = dizi[i];
   int j = i - 1;
   while(j >= 0 && dizi[j] > gecici) {
      dizi[j+1] = dizi[j];
      j = j - 1;
   }
   dizi[j+1] = gecici;
}
```

Eklemeli Sıralama

dizi[]	5	1	9	2	10
	0	1	2	3	4

```
for(int i = 1; i < n; i++) {
   int gecici = dizi[i];
   int j = i - 1;
   while(j >= 0 && dizi[j] > gecici) {
      dizi[j+1] = dizi[j];
      j = j - 1;
   }
   dizi[j+1] = gecici;
}
```

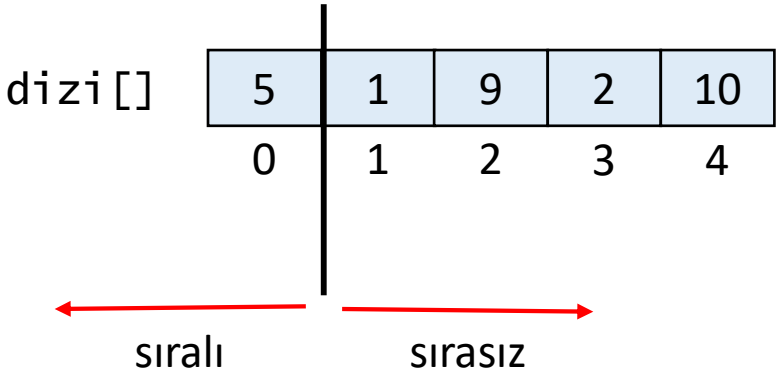
Eklemeli Sıralama

dizi[]	5	1	9	2	10
	0	1	2	3	4

n = 5

```
for(int i = 1; i < n; i++) {
   int gecici = dizi[i];
   int j = i - 1;
   while(j >= 0 && dizi[j] > gecici) {
      dizi[j+1] = dizi[j];
      j = j - 1;
   }
   dizi[j+1] = gecici;
}
```

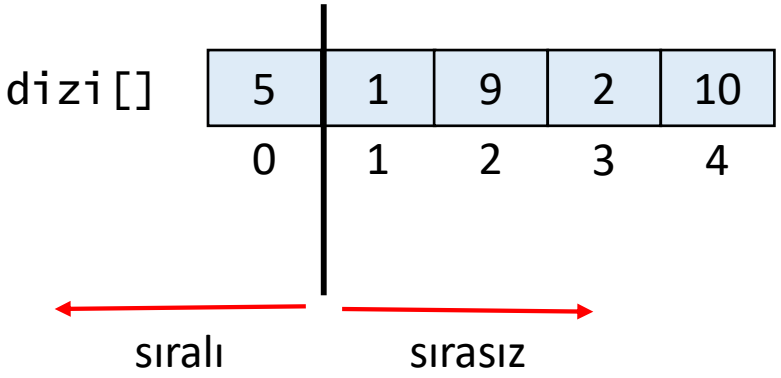
Eklemeli Sıralama




```
for(int i = 1; i < n; i++) {
   int gecici = dizi[i];
   int j = i - 1;
   while(j >= 0 && dizi[j] > gecici) {
      dizi[j+1] = dizi[j];
      j = j - 1;
   }
   dizi[j+1] = gecici;
}
```

n = 5

Eklemeli Sıralama

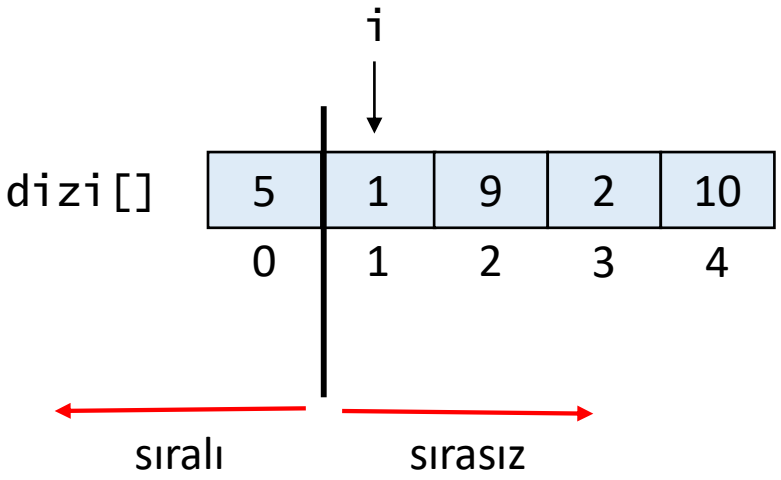


i = 1

n = 5

```
for(int i = 1; i < n; i++) {
   int gecici = dizi[i];
   int j = i - 1;
   while(j >= 0 && dizi[j] > gecici) {
      dizi[j+1] = dizi[j];
      j = j - 1;
   }
   dizi[j+1] = gecici;
}
```

Eklemeli Sıralama

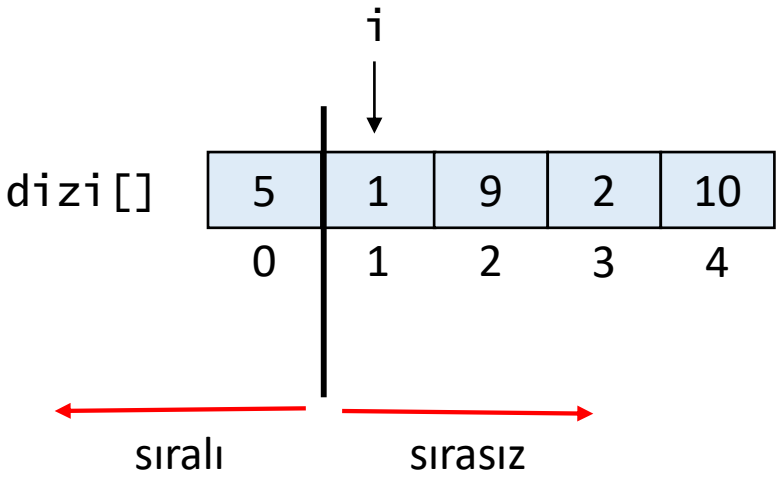


i = 1

n = 5

```
for(int i = 1; i < n; i++) {
   int gecici = dizi[i];
   int j = i - 1;
   while(j >= 0 && dizi[j] > gecici) {
      dizi[j+1] = dizi[j];
      j = j - 1;
   }
   dizi[j+1] = gecici;
}
```

Eklemeli Sıralama

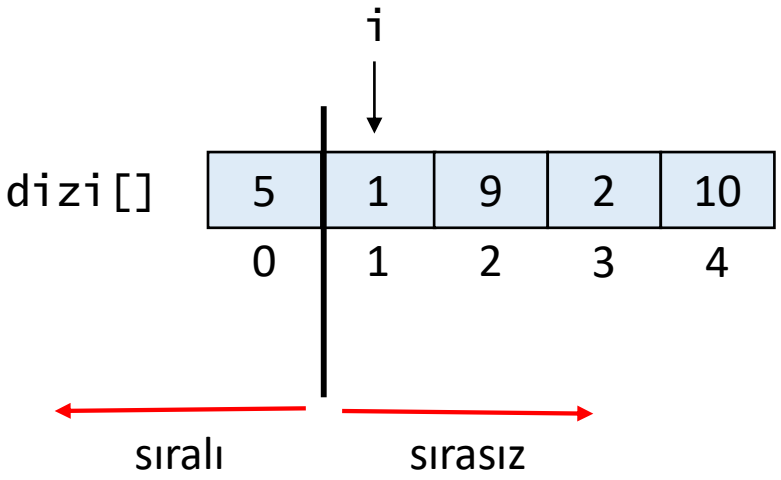


```
i = 1
gecici = 1
```

```
n = 5
```

```
for(int i = 1; i < n; i++) {
   int gecici = dizi[i];
   int j = i - 1;
   while(j >= 0 && dizi[j] > gecici) {
      dizi[j+1] = dizi[j];
      j = j - 1;
   }
   dizi[j+1] = gecici;
}
```

Eklemeli Sıralama

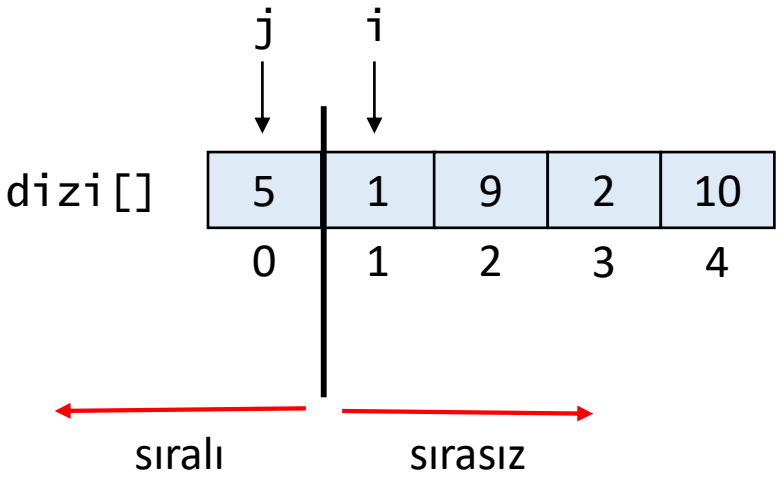



```
i = 1
gecici = 1
j = 0
```

```
n = 5
```

```
for(int i = 1; i < n; i++) {
   int gecici = dizi[i];
   int j = i - 1;
   while(j >= 0 && dizi[j] > gecici) {
      dizi[j+1] = dizi[j];
      j = j - 1;
   }
   dizi[j+1] = gecici;
}
```

Eklemeli Sıralama

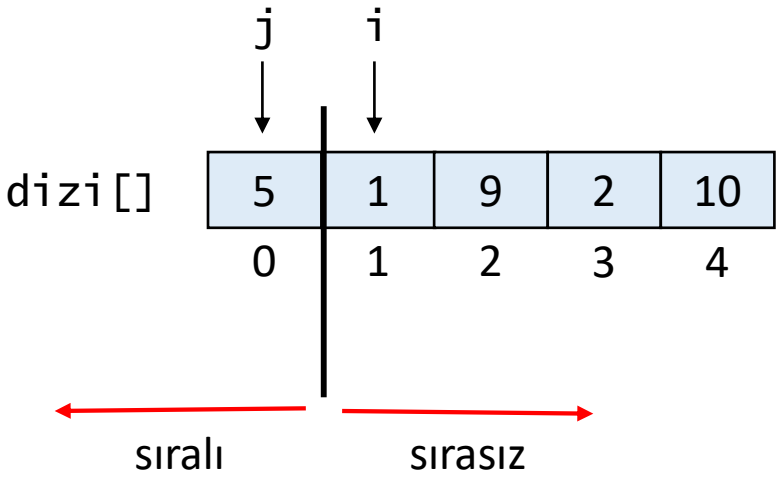


```
i = 1
gecici = 1
j = 0
```

```
n = 5
```

```
for(int i = 1; i < n; i++) {
   int gecici = dizi[i];
   int j = i - 1;
   while(j >= 0 && dizi[j] > gecici) {
      dizi[j+1] = dizi[j];
      j = j - 1;
   }
   dizi[j+1] = gecici;
}
```

Eklemeli Sıralama

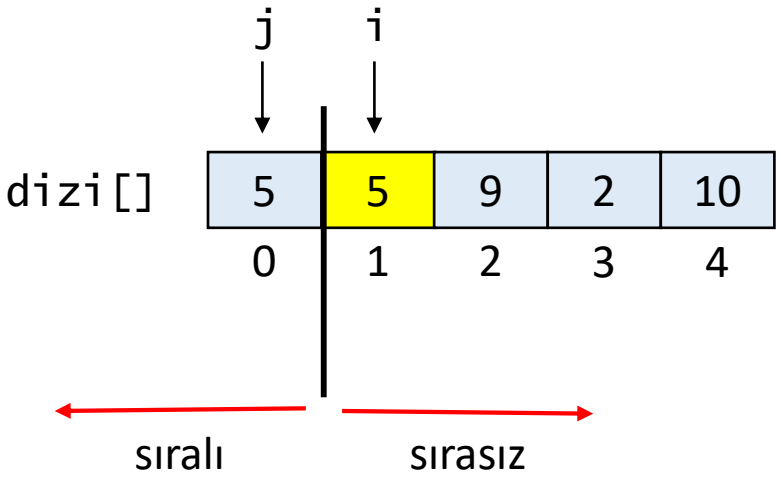



```
i = 1
gecici = 1
j = 0
```

```
n = 5
```

```
for(int i = 1; i < n; i++) {
   int gecici = dizi[i];
   int j = i - 1;
   while(j >= 0 && dizi[j] > gecici) {
      dizi[j+1] = dizi[j];
      j = j - 1;
   }
   dizi[j+1] = gecici;
}
```

Eklemeli Sıralama

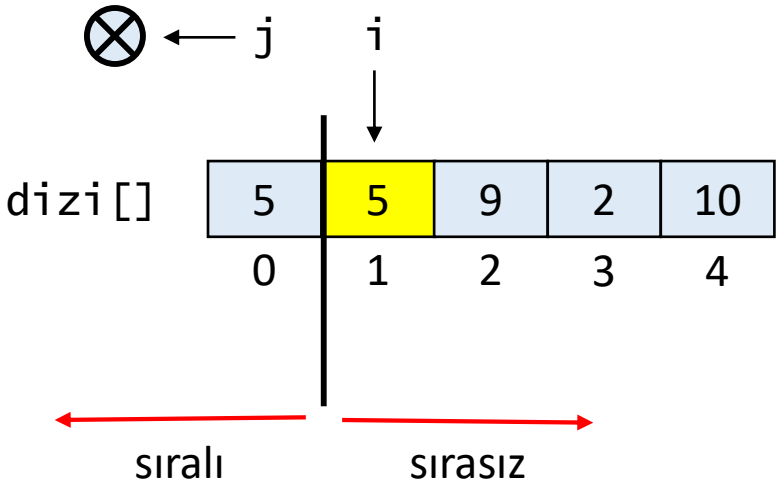


```
i = 1
gecici = 1
j = 0
```

```
n = 5
```

```
for(int i = 1; i < n; i++) {
   int gecici = dizi[i];
   int j = i - 1;
   while(j >= 0 && dizi[j] > gecici) {
      dizi[j+1] = dizi[j];
      j = j - 1;
   }
   dizi[j+1] = gecici;
}
```

Eklemeli Sıralama

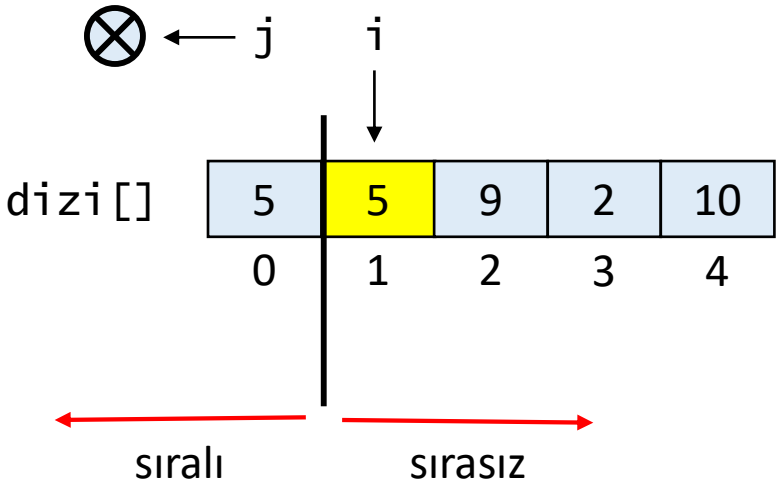


```
i = 1
gecici = 1
j = -1
```

```
n = 5
```

```
for(int i = 1; i < n; i++) {
   int gecici = dizi[i];
   int j = i - 1;
   while(j >= 0 && dizi[j] > gecici) {
      dizi[j+1] = dizi[j];
      j = j - 1;
   }
   dizi[j+1] = gecici;
}
```

Eklemeli Sıralama

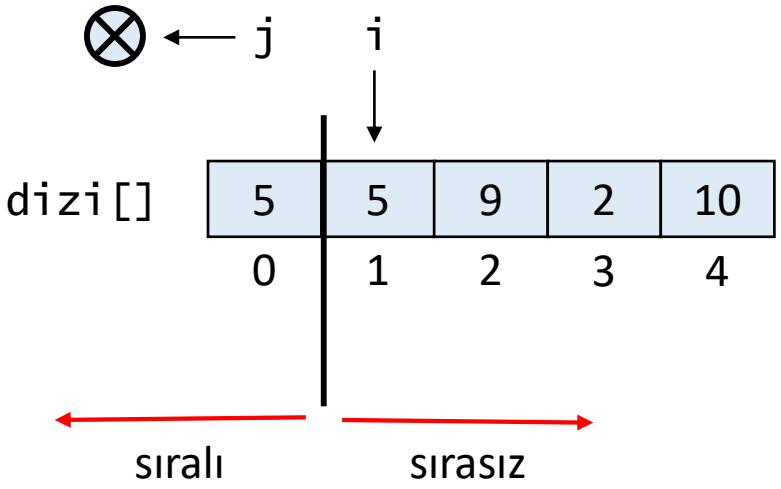


```
i = 1
gecici = 1
j = -1
```

```
n = 5
```

```
for(int i = 1; i < n; i++) {
   int gecici = dizi[i];
   int j = i - 1;
   while(j >= 0 && dizi[j] > gecici) {
      dizi[j+1] = dizi[j];
      j = j - 1;
   }
   dizi[j+1] = gecici;
}
```

Eklemeli Sıralama

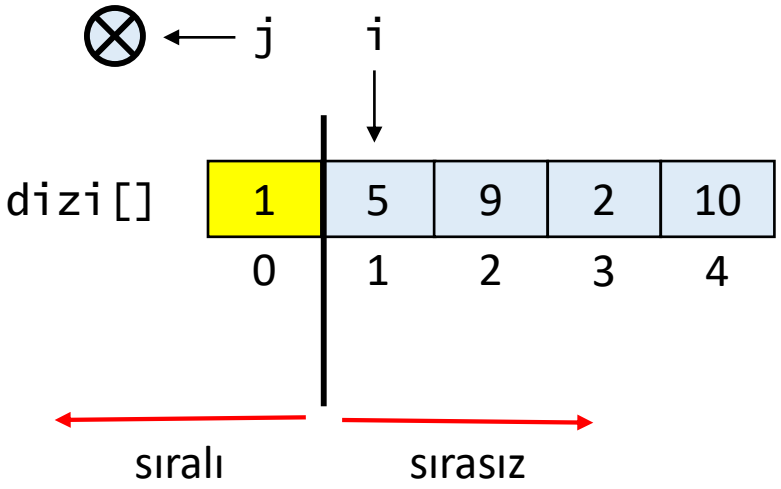


```
i = 1
gecici = 1
j = -1
```

```
n = 5
```

```
for(int i = 1; i < n; i++) {
   int gecici = dizi[i];
   int j = i - 1;
   while(j >= 0 && dizi[j] > gecici) {
      dizi[j+1] = dizi[j];
      j = j - 1;
   }
   dizi[j+1] = gecici;
}
```

Eklemeli Sıralama

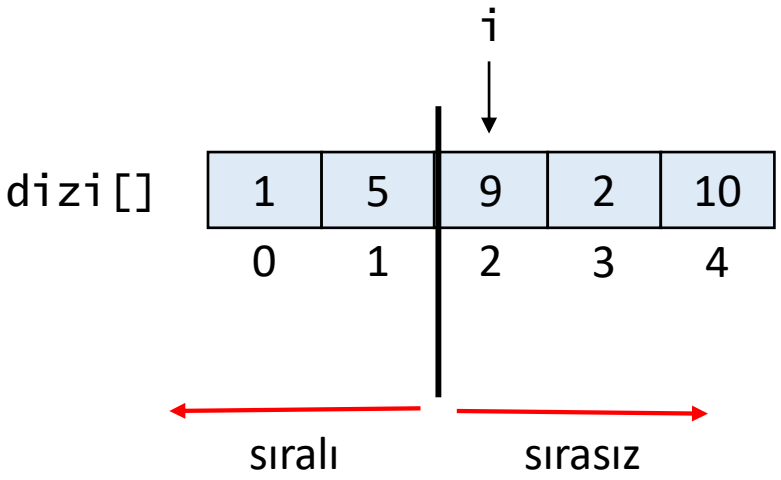


```
i = 1
gecici = 1
j = -1
```

```
n = 5
```

```
for(int i = 1; i < n; i++) {
   int gecici = dizi[i];
   int j = i - 1;
   while(j >= 0 && dizi[j] > gecici) {
      dizi[j+1] = dizi[j];
      j = j - 1;
   }
   dizi[j+1] = gecici;
}
```

Eklemeli Sıralama

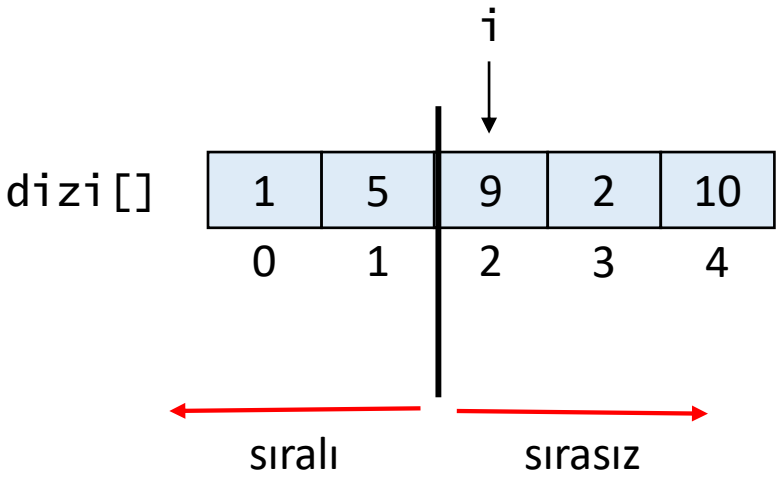


i = 2

n = 5

```
for(int i = 1; i < n; i++) {
   int gecici = dizi[i];
   int j = i - 1;
   while(j >= 0 && dizi[j] > gecici) {
      dizi[j+1] = dizi[j];
      j = j - 1;
   }
   dizi[j+1] = gecici;
}
```

Eklemeli Sıralama

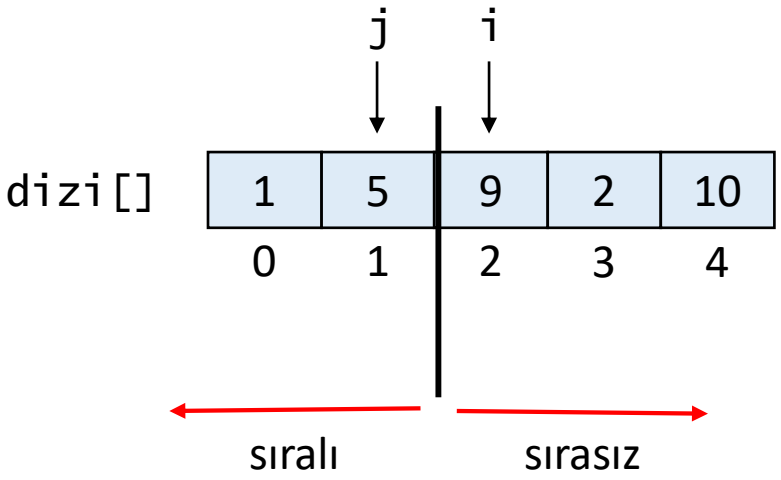



```
i = 2
gecici = 9
```

```
n = 5
```

```
for(int i = 1; i < n; i++) {
   int gecici = dizi[i];
   int j = i - 1;
   while(j >= 0 && dizi[j] > gecici) {
      dizi[j+1] = dizi[j];
      j = j - 1;
   }
   dizi[j+1] = gecici;
}
```

Eklemeli Sıralama

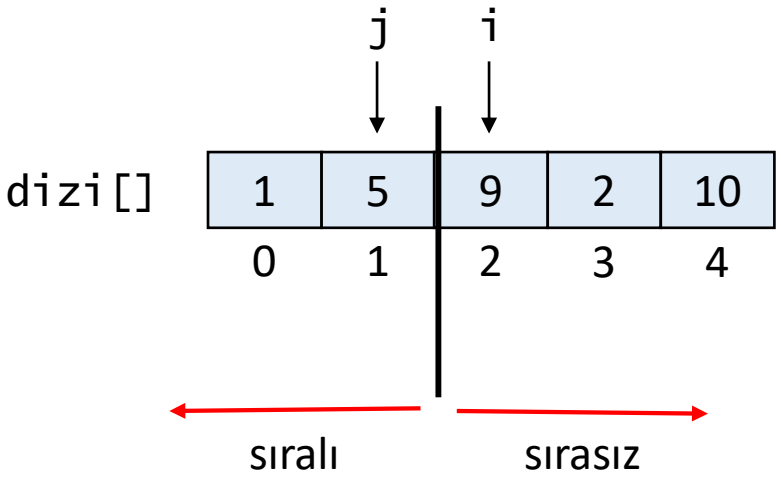



```
i = 2
gecici = 9
j = 1
```

```
n = 5
```

```
for(int i = 1; i < n; i++) {
   int gecici = dizi[i];
   int j = i - 1;
   while(j >= 0 && dizi[j] > gecici) {
      dizi[j+1] = dizi[j];
      j = j - 1;
   }
   dizi[j+1] = gecici;
}
```

Eklemeli Sıralama

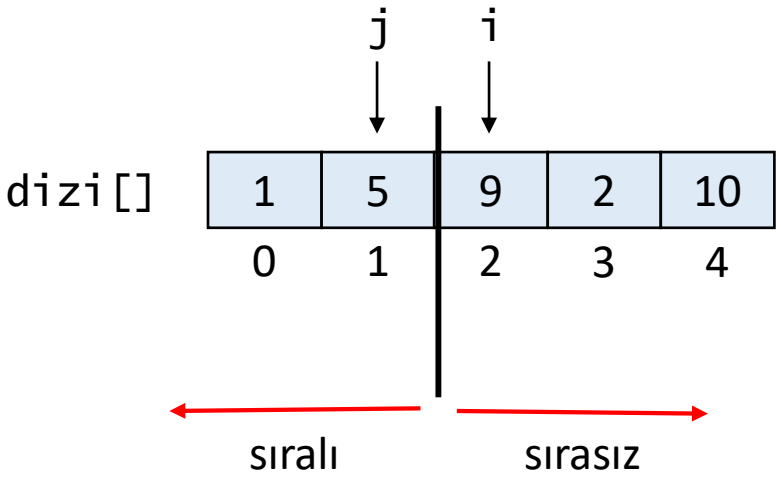



```
i = 2
gecici = 9
j = 1
```

```
n = 5
```

```
for(int i = 1; i < n; i++) {
   int gecici = dizi[i];
   int j = i - 1;
   while(j >= 0 && dizi[j] > gecici) {
      dizi[j+1] = dizi[j];
      j = j - 1;
   }
   dizi[j+1] = gecici;
}
```

Eklemeli Sıralama

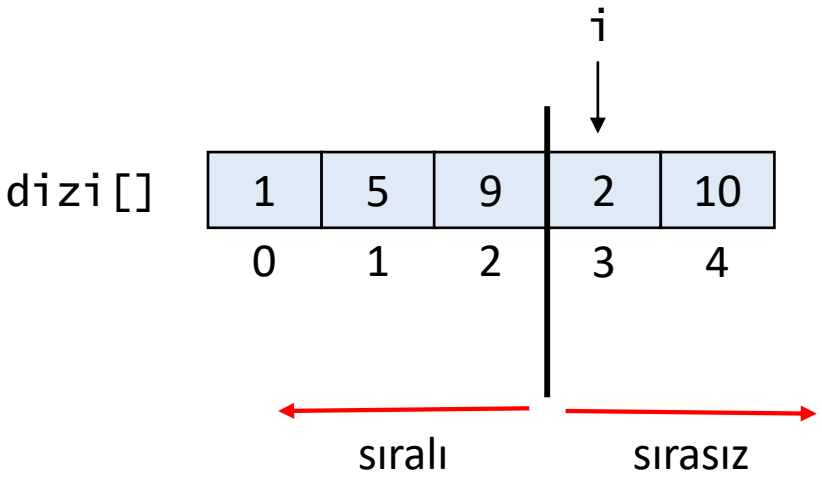



```
i = 2
gecici = 9
j = 1
```

```
n = 5
```

```
for(int i = 1; i < n; i++) {
   int gecici = dizi[i];
   int j = i - 1;
   while(j >= 0 && dizi[j] > gecici) {
      dizi[j+1] = dizi[j];
      j = j - 1;
   }
   dizi[j+1] = gecici;
}
```

Eklemeli Sıralama

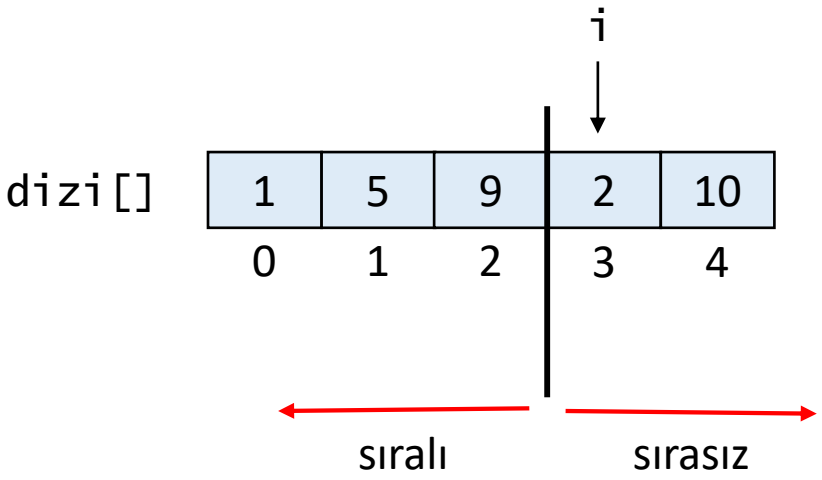


i = 3

n = 5

```
for(int i = 1; i < n; i++) {
   int gecici = dizi[i];
   int j = i - 1;
   while(j >= 0 && dizi[j] > gecici) {
      dizi[j+1] = dizi[j];
      j = j - 1;
   }
   dizi[j+1] = gecici;
}
```

Eklemeli Sıralama

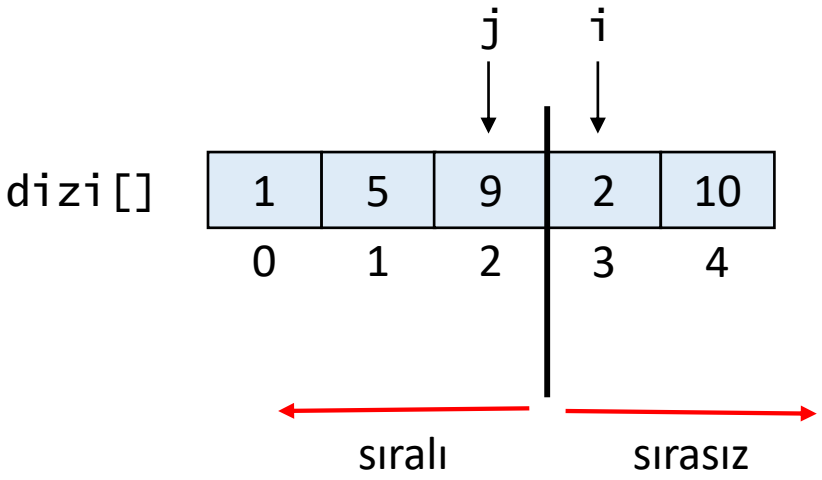



```
i = 3
gecici = 2
```

```
n = 5
```

```
for(int i = 1; i < n; i++) {
   int gecici = dizi[i];
   int j = i - 1;
   while(j >= 0 && dizi[j] > gecici) {
      dizi[j+1] = dizi[j];
      j = j - 1;
   }
   dizi[j+1] = gecici;
}
```

Eklemeli Sıralama

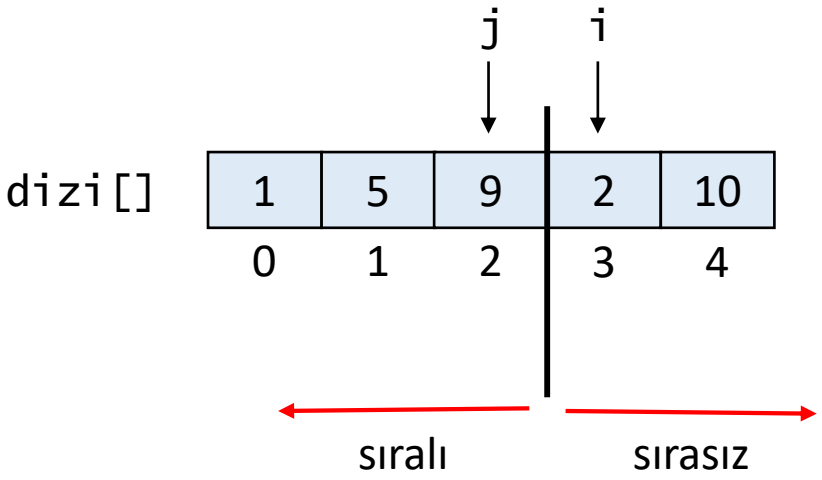



```
i = 3
gecici = 2
j = 2
```

```
n = 5
```

```
for(int i = 1; i < n; i++) {
   int gecici = dizi[i];
   int j = i - 1;
   while(j >= 0 && dizi[j] > gecici) {
      dizi[j+1] = dizi[j];
      j = j - 1;
   }
   dizi[j+1] = gecici;
}
```

Eklemeli Sıralama

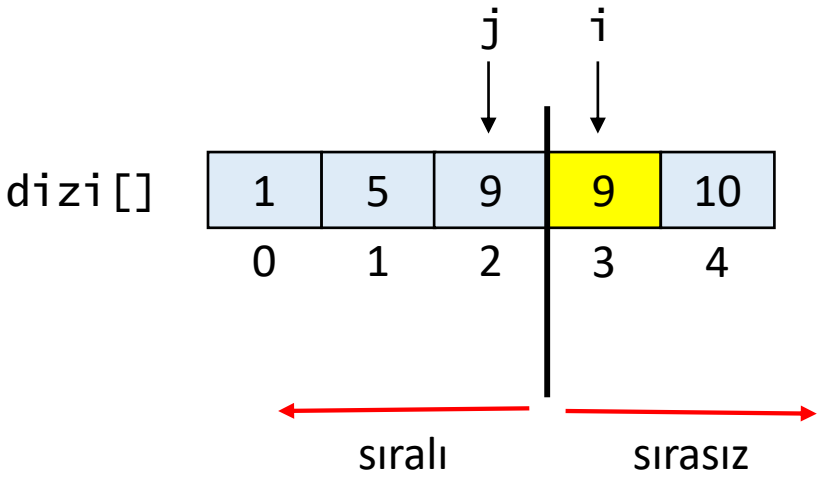



```
i = 3
gecici = 2
j = 2
```

```
n = 5
```

```
for(int i = 1; i < n; i++) {
   int gecici = dizi[i];
   int j = i - 1;
   while(j >= 0 && dizi[j] > gecici) {
      dizi[j+1] = dizi[j];
      j = j - 1;
   }
   dizi[j+1] = gecici;
}
```

Eklemeli Sıralama

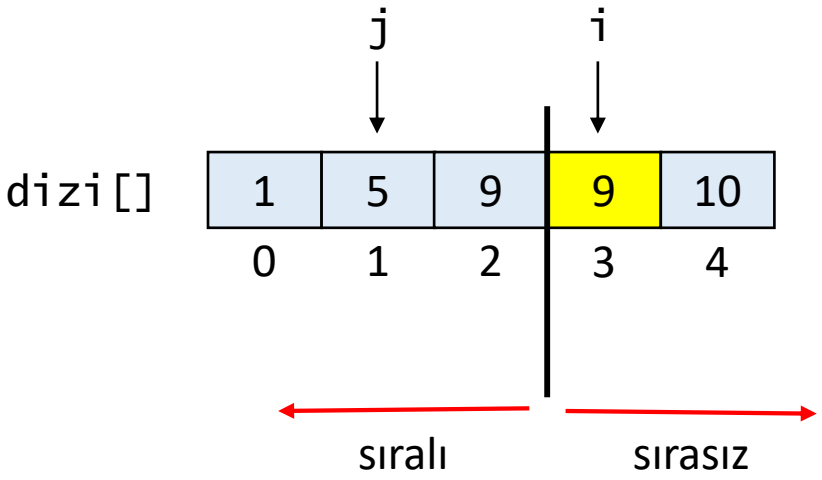



```
i = 3
gecici = 2
j = 2
```

```
n = 5
```

```
for(int i = 1; i < n; i++) {
   int gecici = dizi[i];
   int j = i - 1;
   while(j >= 0 && dizi[j] > gecici) {
      dizi[j+1] = dizi[j];
      j = j - 1;
   }
   dizi[j+1] = gecici;
}
```

Eklemeli Sıralama

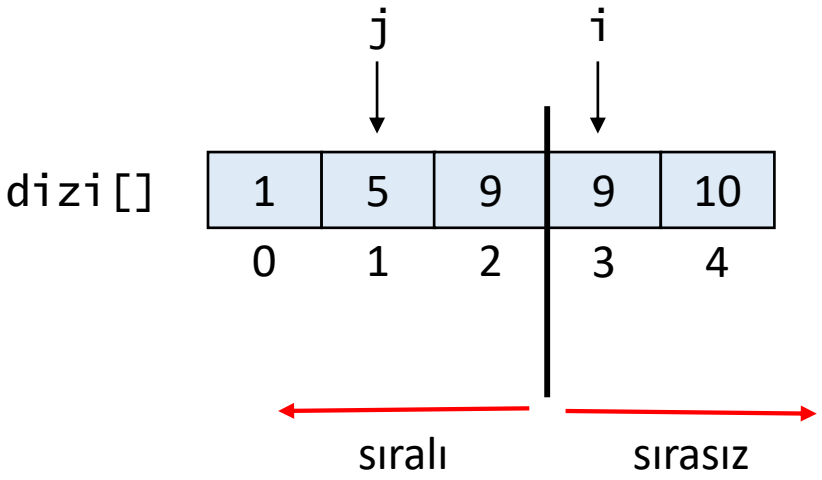



```
i = 3
gecici = 2
j = 1
```

```
n = 5
```

```
for(int i = 1; i < n; i++) {
   int gecici = dizi[i];
   int j = i - 1;
   while(j >= 0 && dizi[j] > gecici) {
      dizi[j+1] = dizi[j];
      j = j - 1;
   }
   dizi[j+1] = gecici;
}
```

Eklemeli Sıralama

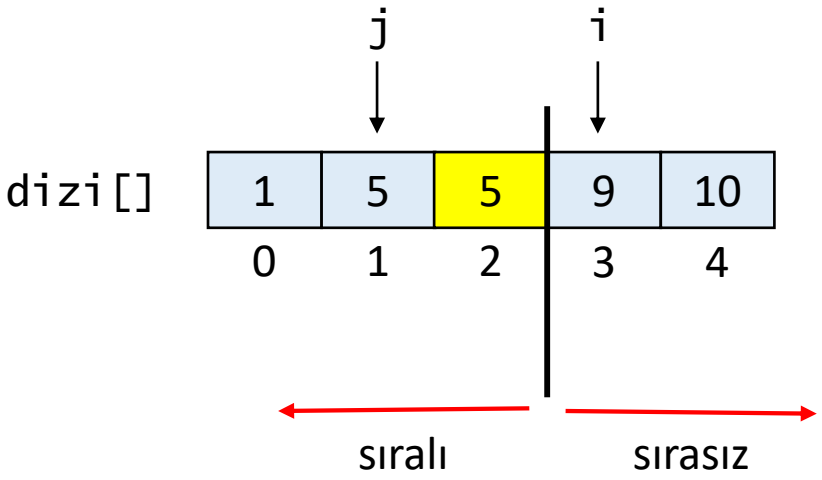



```
i = 3
gecici = 2
j = 1
```

```
n = 5
```

```
for(int i = 1; i < n; i++) {
   int gecici = dizi[i];
   int j = i - 1;
   while(j >= 0 && dizi[j] > gecici) {
      dizi[j+1] = dizi[j];
      j = j - 1;
   }
   dizi[j+1] = gecici;
}
```

Eklemeli Sıralama

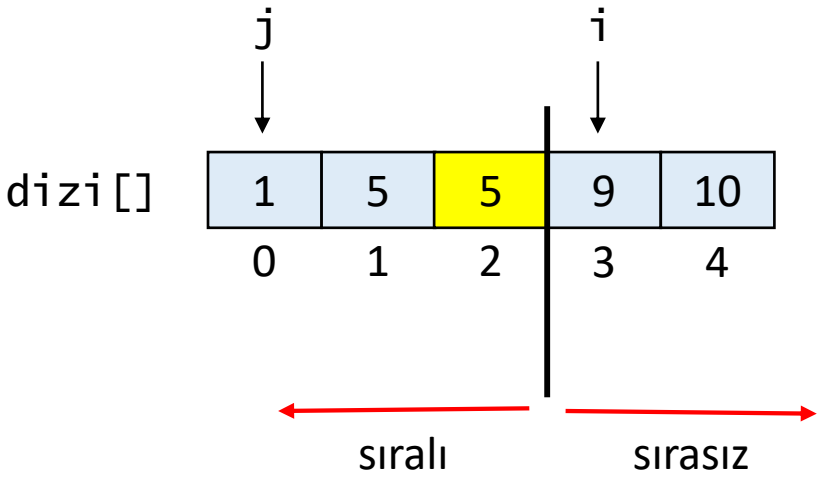



```
i = 3
gecici = 2
j = 1
```

```
n = 5
```

```
for(int i = 1; i < n; i++) {
   int gecici = dizi[i];
   int j = i - 1;
   while(j >= 0 && dizi[j] > gecici) {
      dizi[j+1] = dizi[j];
      j = j - 1;
   }
   dizi[j+1] = gecici;
}
```

Eklemeli Sıralama

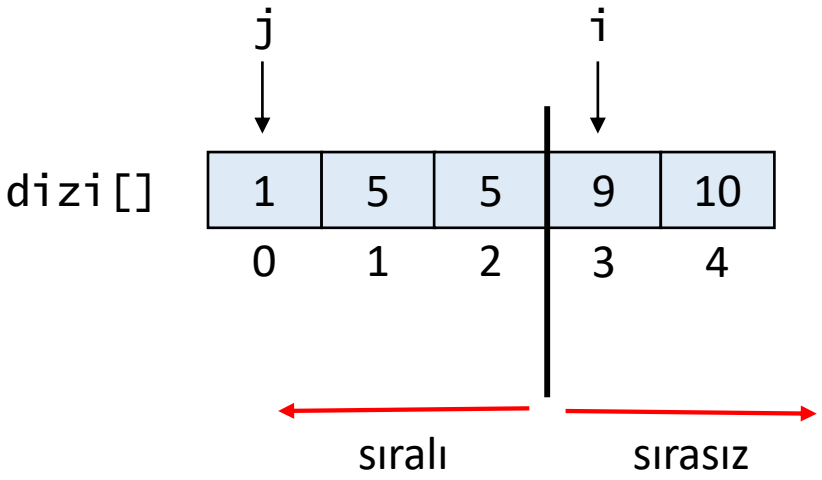



```
i = 3
gecici = 2
j = 0
```

```
n = 5
```

```
for(int i = 1; i < n; i++) {
   int gecici = dizi[i];
   int j = i - 1;
   while(j >= 0 && dizi[j] > gecici) {
      dizi[j+1] = dizi[j];
      j = j - 1;
   }
   dizi[j+1] = gecici;
}
```

Eklemeli Sıralama

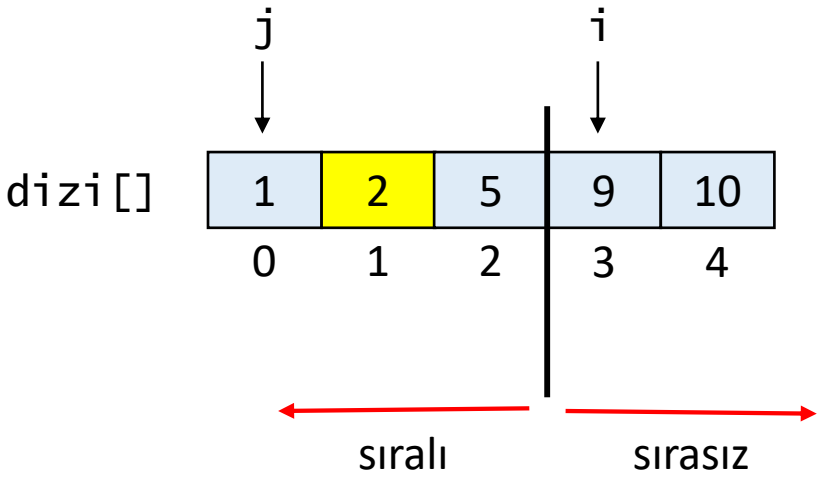



```
i = 3
gecici = 2
j = 0
```

```
n = 5
```

```
for(int i = 1; i < n; i++) {
   int gecici = dizi[i];
   int j = i - 1;
   while(j >= 0 && dizi[j] > gecici) {
      dizi[j+1] = dizi[j];
      j = j - 1;
   }
   dizi[j+1] = gecici;
}
```

Eklemeli Sıralama

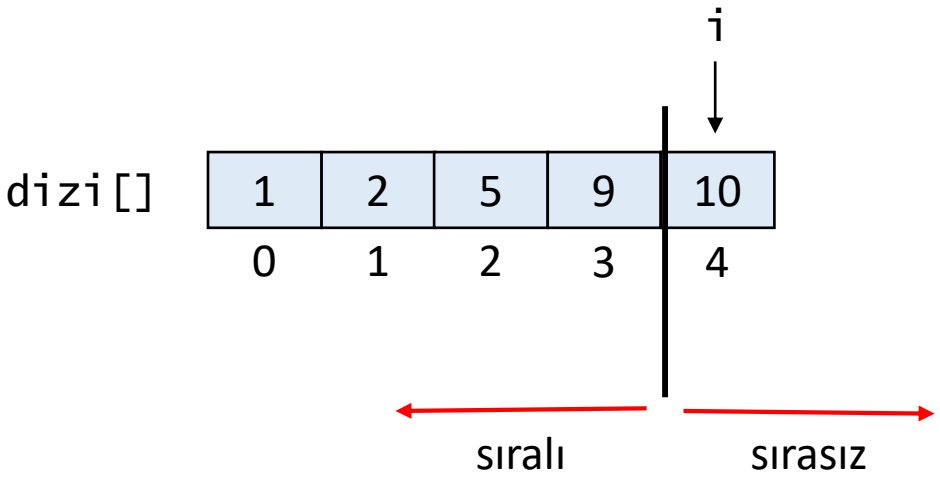


```
i = 3
gecici = 2
j = 0
```

```
n = 5
```

```
for(int i = 1; i < n; i++) {
   int gecici = dizi[i];
   int j = i - 1;
   while(j >= 0 && dizi[j] > gecici) {
      dizi[j+1] = dizi[j];
      j = j - 1;
   }
   dizi[j+1] = gecici;
}
```

Eklemeli Sıralama

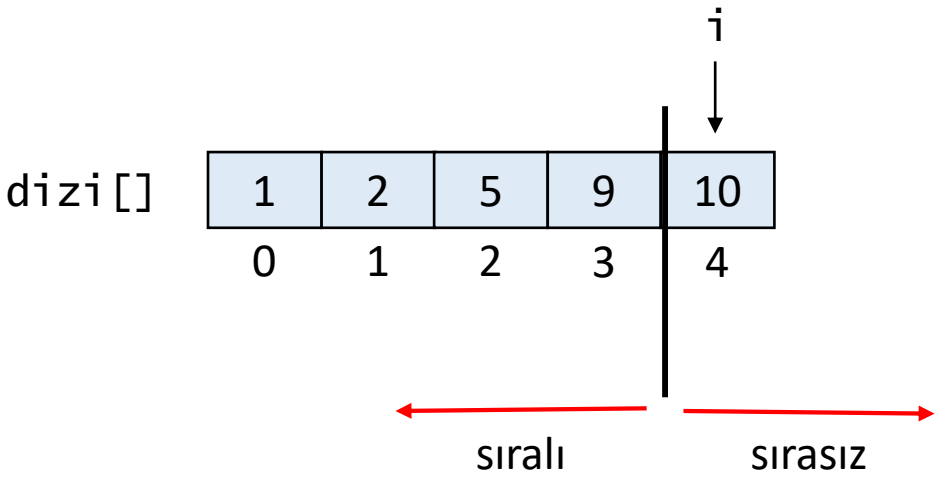


i = 4

n = 5

```
for(int i = 1; i < n; i++) {
   int gecici = dizi[i];
   int j = i - 1;
   while(j >= 0 && dizi[j] > gecici) {
      dizi[j+1] = dizi[j];
      j = j - 1;
   }
   dizi[j+1] = gecici;
}
```

Eklemeli Sıralama

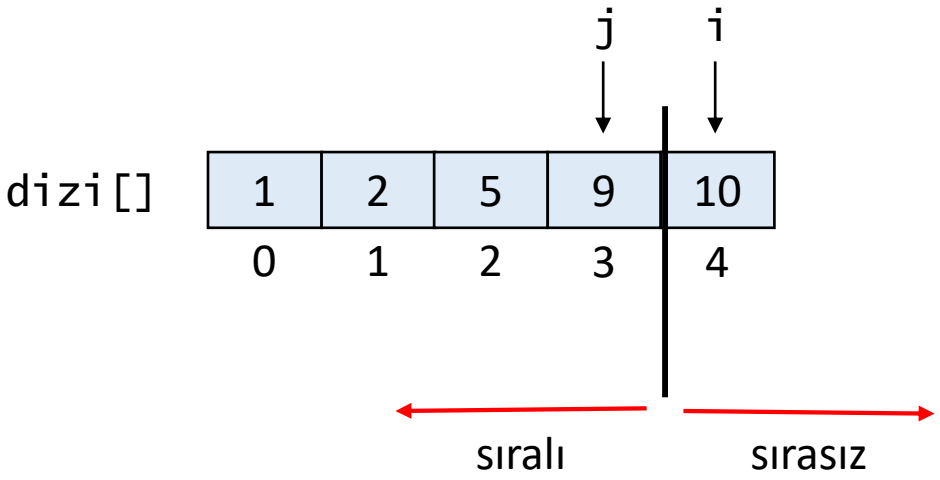



```
i = 4
gecici = 10
```

```
n = 5
```

```
for(int i = 1; i < n; i++) {
   int gecici = dizi[i];
   int j = i - 1;
   while(j >= 0 && dizi[j] > gecici) {
      dizi[j+1] = dizi[j];
      j = j - 1;
   }
   dizi[j+1] = gecici;
}
```

Eklemeli Sıralama

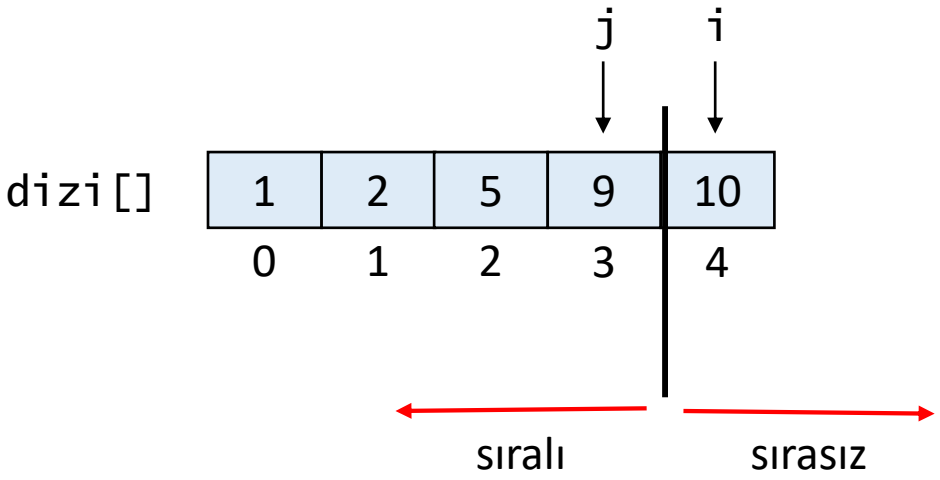



```
i = 4
gecici = 10
j = 3
```

```
n = 5
```

```
for(int i = 1; i < n; i++) {
   int gecici = dizi[i];
   int j = i - 1;
   while(j >= 0 && dizi[j] > gecici) {
      dizi[j+1] = dizi[j];
      j = j - 1;
   }
   dizi[j+1] = gecici;
}
```

Eklemeli Sıralama

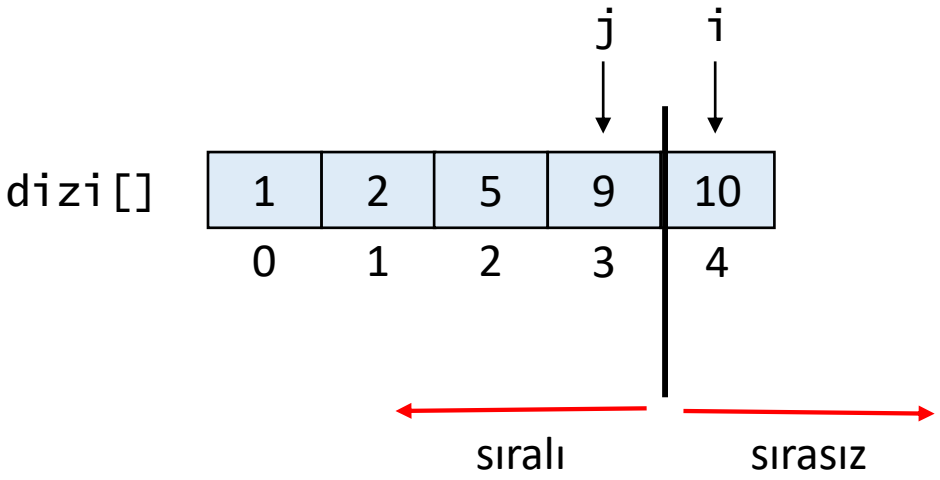



```
i = 4
gecici = 10
j = 3
```

```
n = 5
```

```
for(int i = 1; i < n; i++) {
   int gecici = dizi[i];
   int j = i - 1;
   while(j >= 0 && dizi[j] > gecici) {
      dizi[j+1] = dizi[j];
      j = j - 1;
   }
   dizi[j+1] = gecici;
}
```

Eklemeli Sıralama




```
i = 4
gecici = 10
j = 3
```

```
n = 5
```

```
for(int i = 1; i < n; i++) {
   int gecici = dizi[i];
   int j = i - 1;
   while(j >= 0 && dizi[j] > gecici) {
      dizi[j+1] = dizi[j];
      j = j - 1;
   }
   dizi[j+1] = gecici;
}
```

Eklemeli Sıralama

dizi[]	1	2	5	9	10
	0	1	2	3	4

i = 5

n = 5

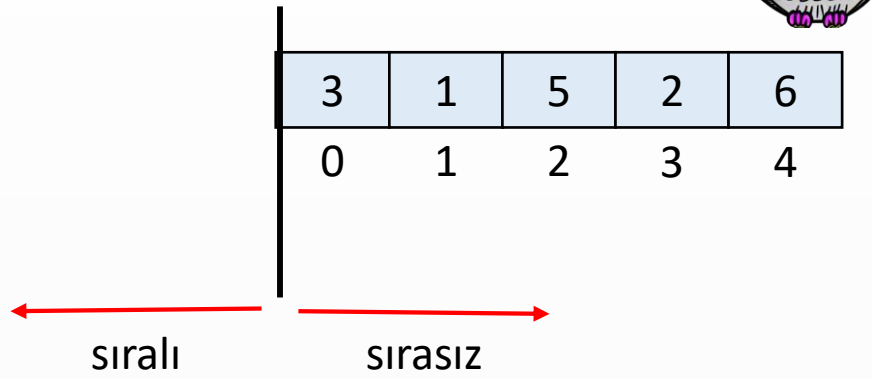
```
for(int i = 1; i < n; i++) {
   int gecici = dizi[i];
   int j = i - 1;
   while(j >= 0 && dizi[j] > gecici) {
      dizi[j+1] = dizi[j];
      j = j - 1;
   }
   dizi[j+1] = gecici;
}
```

Eklemeli Sıralama

dizi[]	1	2	5	9	10
	0	1	2	3	4

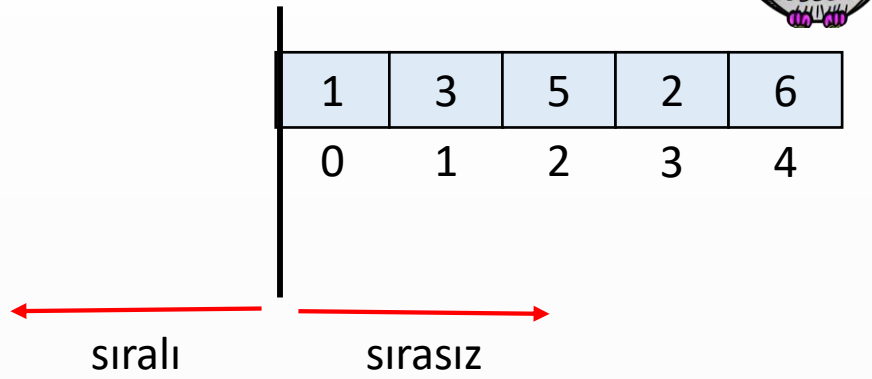
```
for(int i = 1; i < n; i++) {
   int gecici = dizi[i];
   int j = i - 1;
   while(j >= 0 && dizi[j] > gecici) {
      dizi[j+1] = dizi[j];
      j = j - 1;
   }
   dizi[j+1] = gecici;
}
```

Seçmeli Sıralama (Selection Sort)



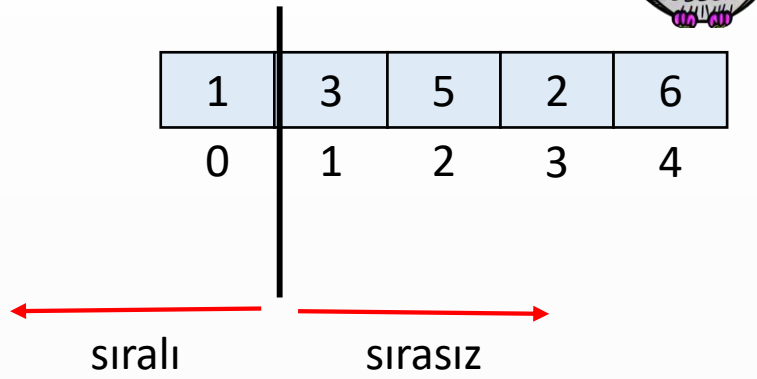
- Verilen dizi iki bölüme ayrılır:
  - Sıralı
  - Sırasız
- Her bir adımda,
  - Sırasız bölümden en küçük eleman bulunur.
  - Sırasız bölümün başındaki eleman ile yer değiştirilir.
  - Yer değiştirmenin ardından sıralı bölümün parçası olur.

Seçmeli Sıralama (Selection Sort)



- Verilen dizi iki bölüme ayrılır:
  - Sıralı
  - Sırasız
- Her bir adımda,
  - Sırasız bölümden en küçük eleman bulunur.
  - Sırasız bölümün başındaki eleman ile yer değiştirilir.
  - Yer değiştirmenin ardından sıralı bölümün parçası olur.

Seçmeli Sıralama (Selection Sort)



- Verilen dizi iki bölüme ayrılır:
  - Sıralı
  - Sırasız
- Her bir adımda,
  - Sırasız bölümden en küçük eleman bulunur.
  - Sırasız bölümün başındaki eleman ile yer değiştirilir.
  - Yer değiştirmenin ardından sıralı bölümün parçası olur.

Seçmeli Sıralama

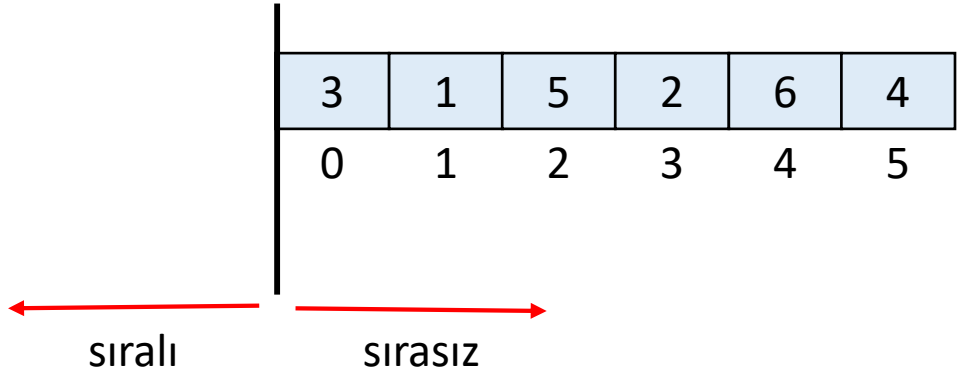
3	1	5	2	6	4
0	1	2	3	4	5

Seçmeli Sıralama

3	1	5	2	6	4
0	1	2	3	4	5

```
uzunluk = 6
```

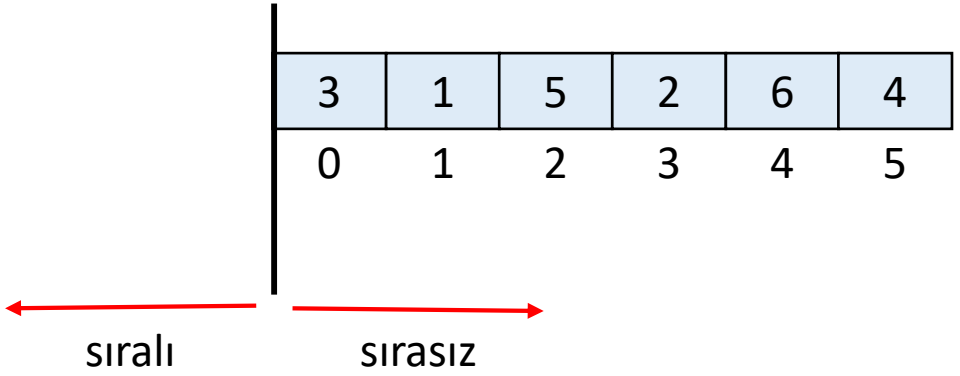
Seçmeli Sıralama




```
uzunluk = 6
```

Seçmeli Sıralama

1. Tur

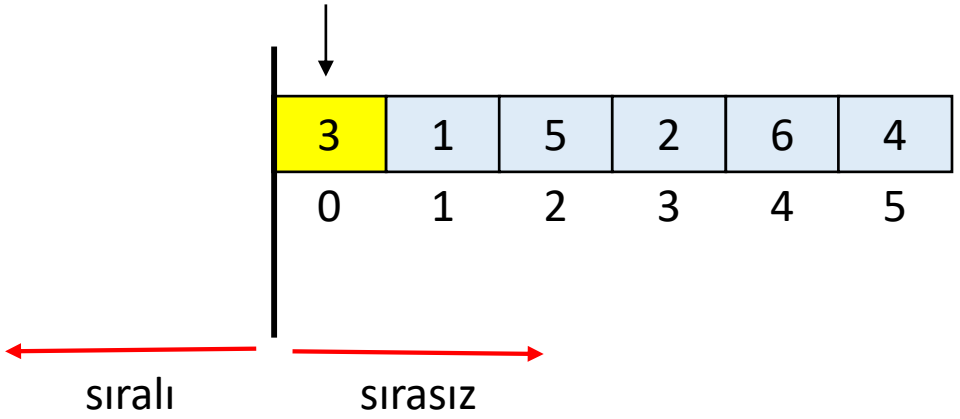



```
uzunluk = 6
```

Seçmeli Sıralama

1. Tur

```
min = 3
```

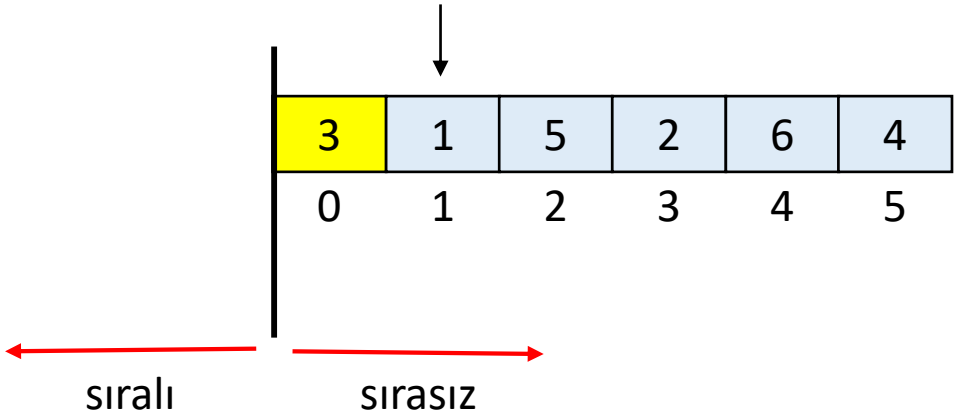



```
uzunluk = 6
```

Seçmeli Sıralama

1. Tur

```
min = 3
1 < min ?
```

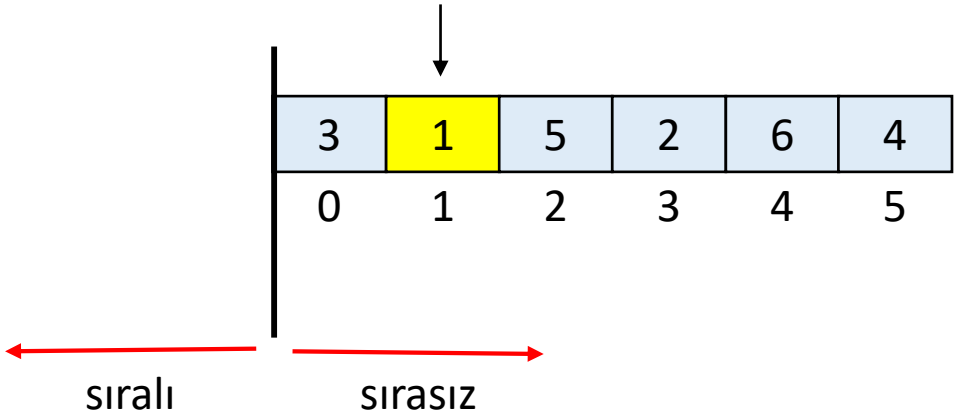



```
uzunluk = 6
```

Seçmeli Sıralama

1. Tur

```
min = 1
```

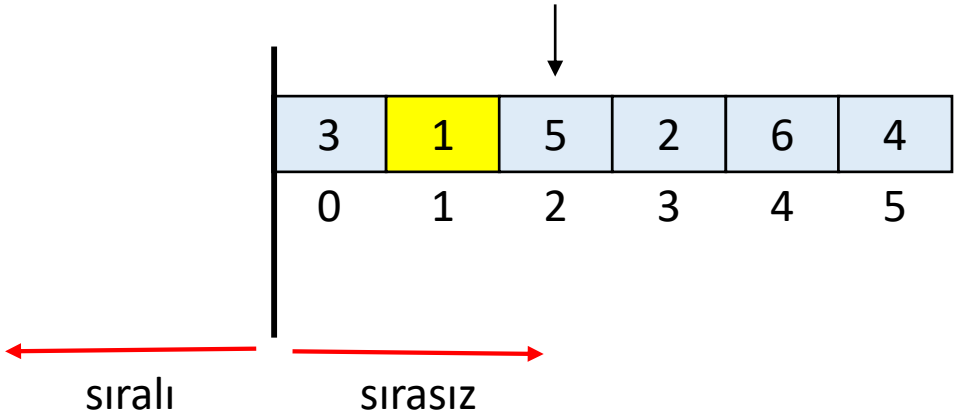



```
uzunluk = 6
```

Seçmeli Sıralama

1. Tur

```
min = 1
5 < min ?
```

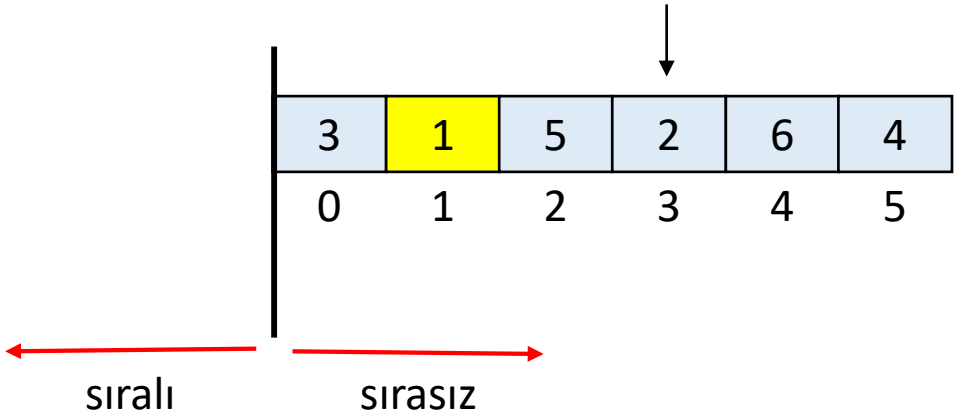



```
uzunluk = 6
```

Seçmeli Sıralama

1. Tur

```
min = 1
2 < min ?
```

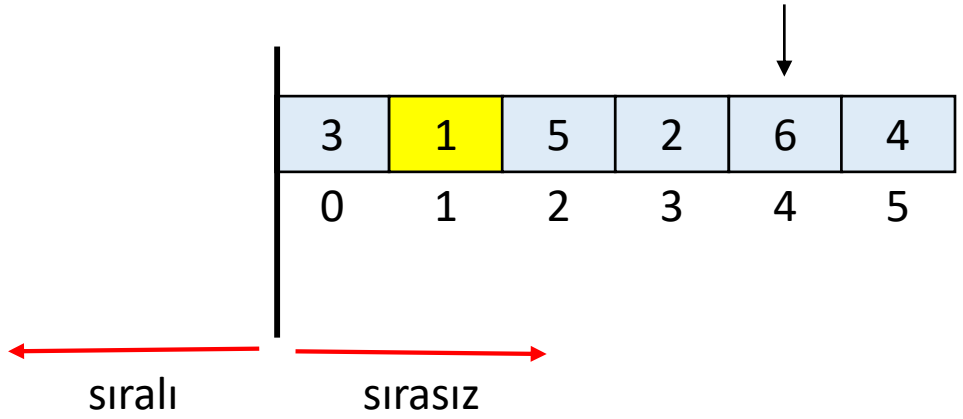



```
uzunluk = 6
```

Seçmeli Sıralama

1. Tur

```
min = 1
6 < min ?
```

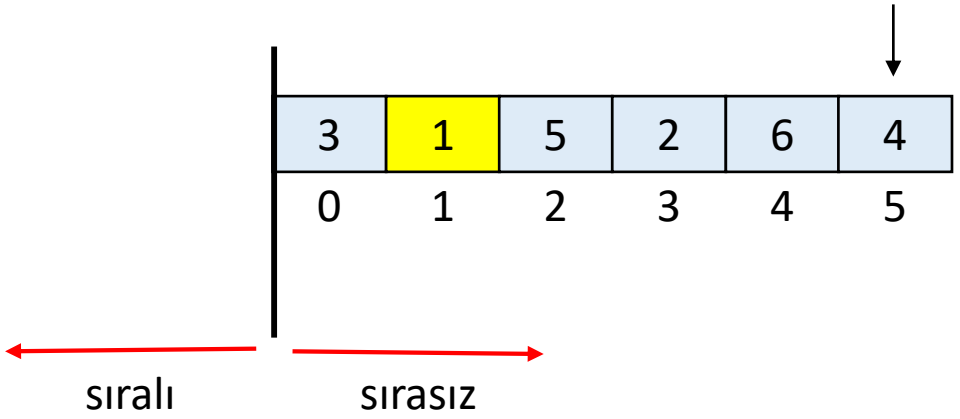



```
uzunluk = 6
```

Seçmeli Sıralama

1. Tur

```
min = 1
4 < min ?
```

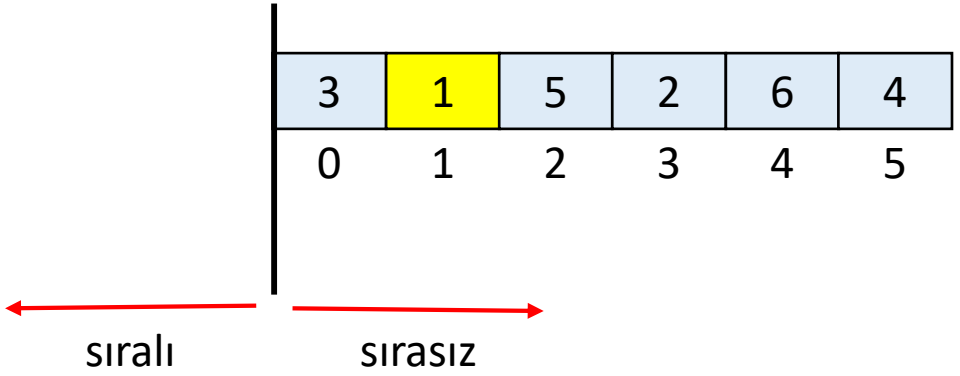



```
uzunluk = 6
```

Seçmeli Sıralama

1. Tur

```
min = 1
```

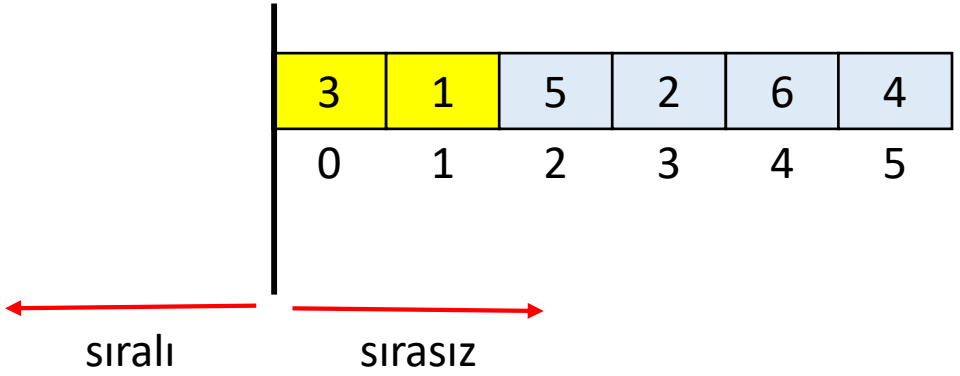



```
uzunluk = 6
```

Seçmeli Sıralama

1. Tur

```
min = 1
```

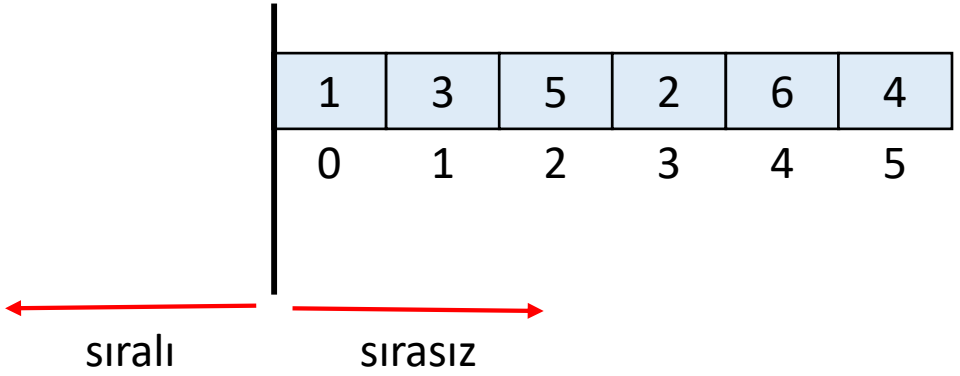



```
uzunluk = 6
```

Seçmeli Sıralama

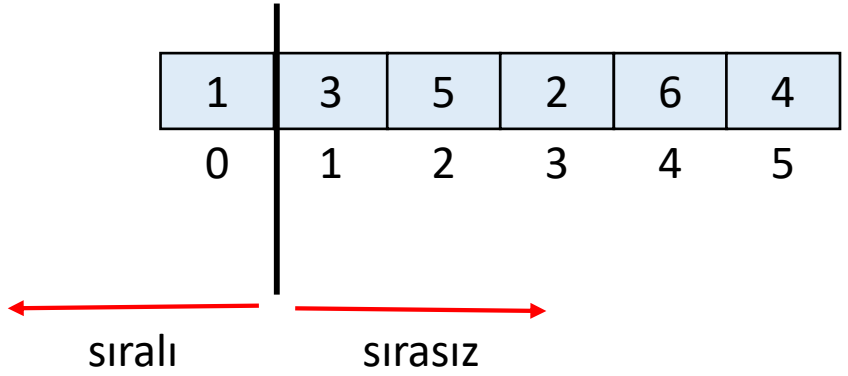
1. Tur

```
min = 1
```




```
uzunluk = 6
```

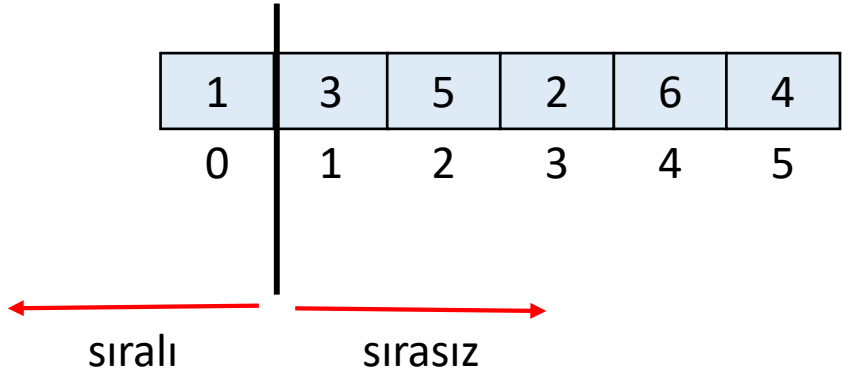
Seçmeli Sıralama




```
uzunluk = 6
```

Seçmeli Sıralama

2. Tur

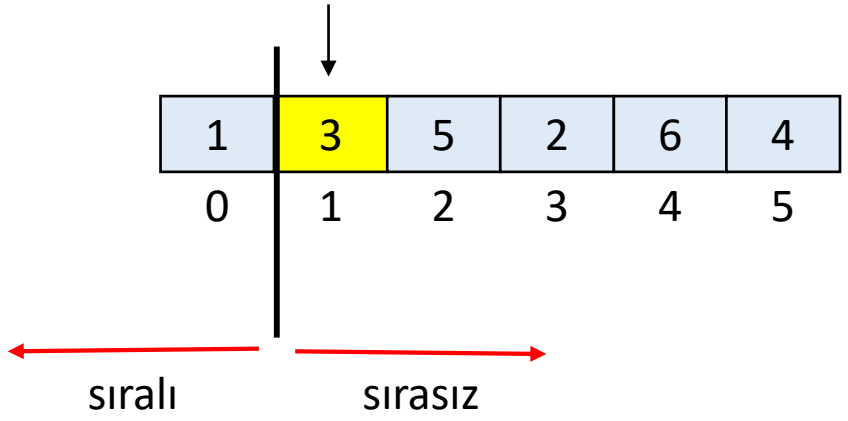



```
uzunluk = 6
```

Seçmeli Sıralama

2. Tur

```
min = 3
```

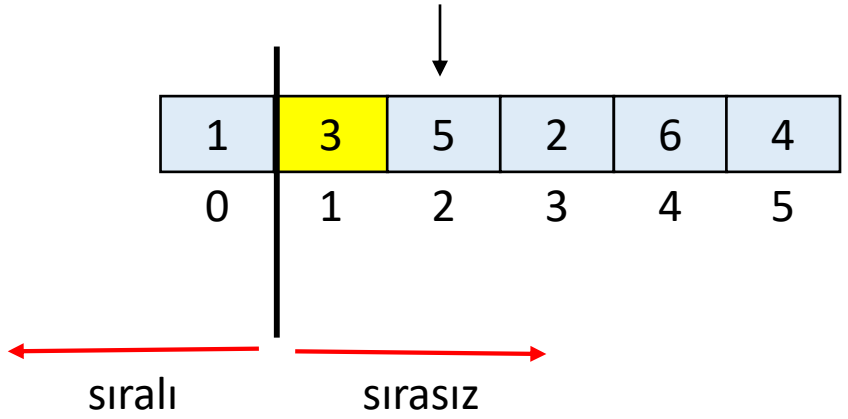



```
uzunluk = 6
```

Seçmeli Sıralama

2. Tur

```
min = 3
5 < min ?
```

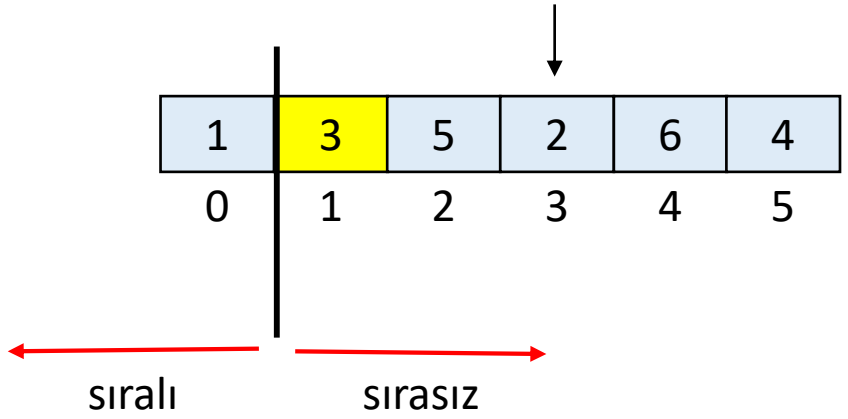



```
uzunluk = 6
```

Seçmeli Sıralama

2. Tur

```
min = 3
2 < min ?
```

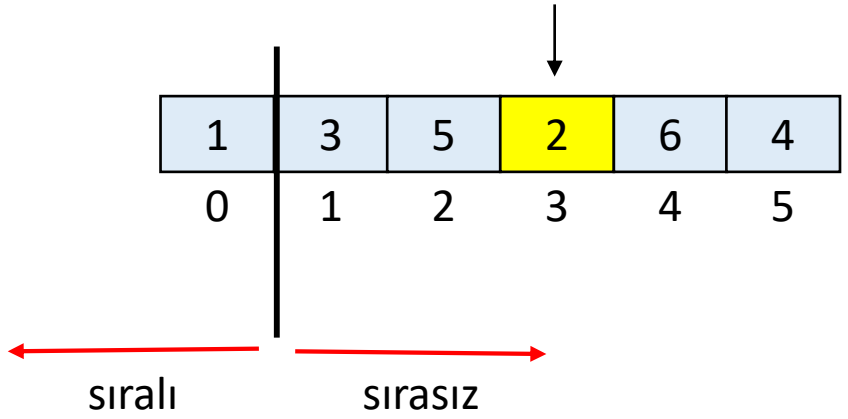



```
uzunluk = 6
```

Seçmeli Sıralama

2. Tur

```
min = 2
```

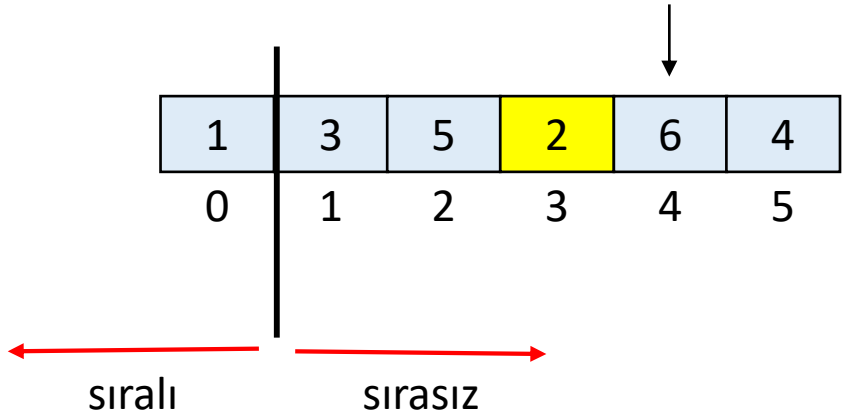



```
uzunluk = 6
```

Seçmeli Sıralama

2. Tur

```
min = 2
6 < min ?
```

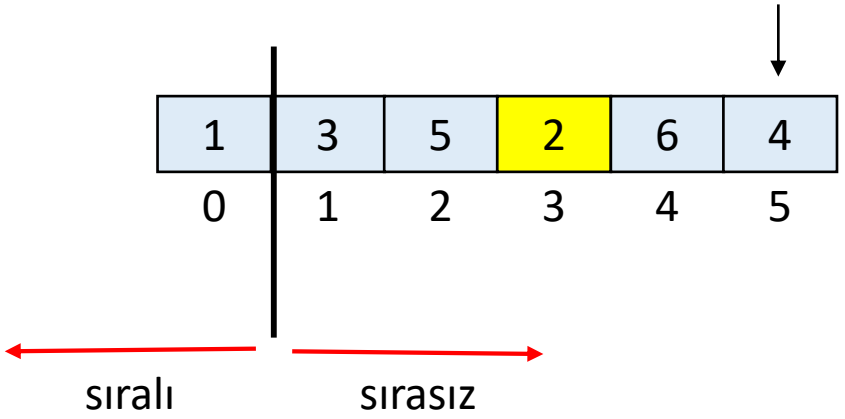



```
uzunluk = 6
```

Seçmeli Sıralama

2. Tur

```
min = 2
4 < min ?
```

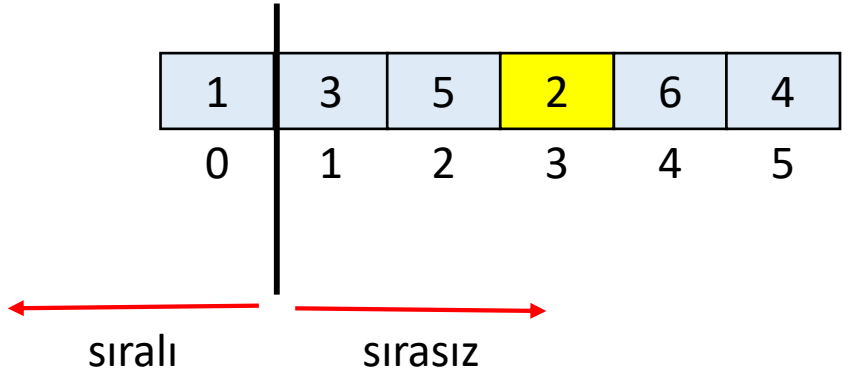



```
uzunluk = 6
```

Seçmeli Sıralama

2. Tur

```
min = 2
```

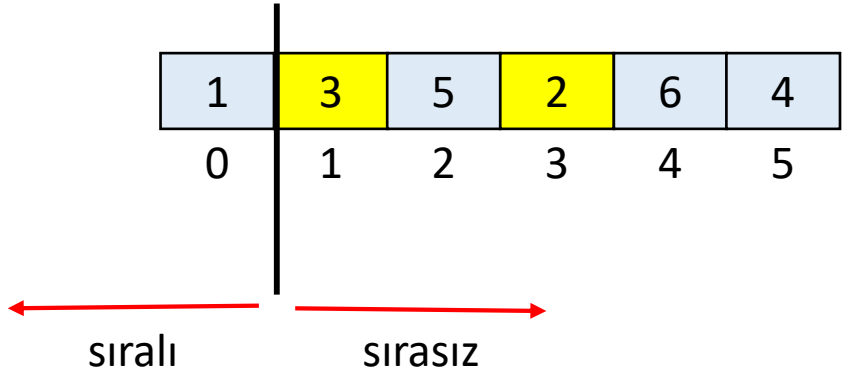



```
uzunluk = 6
```

Seçmeli Sıralama

2. Tur

```
min = 2
```

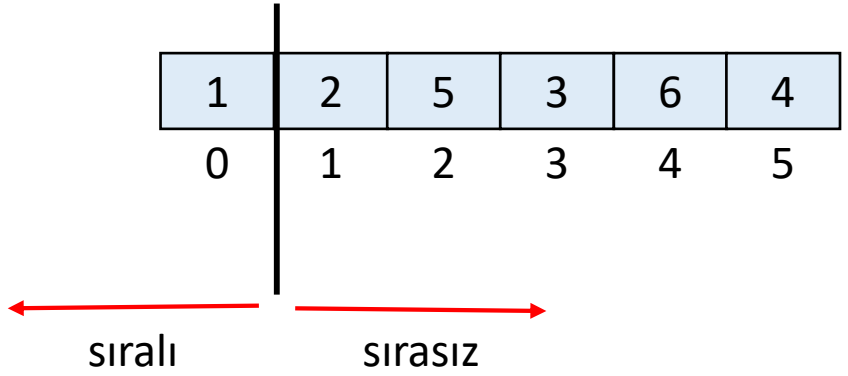



```
uzunluk = 6
```

Seçmeli Sıralama

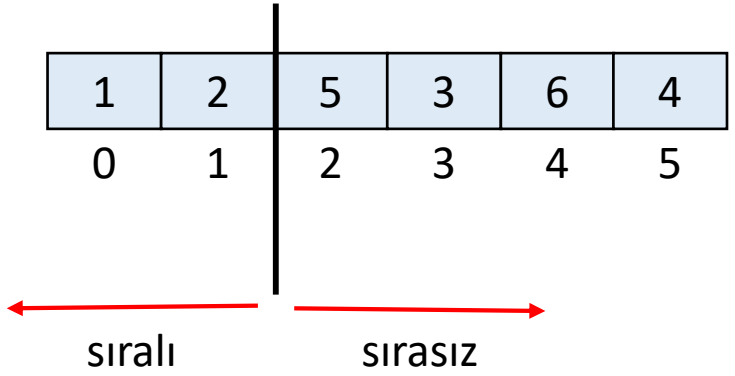
2. Tur

```
min = 2
```




```
uzunluk = 6
```

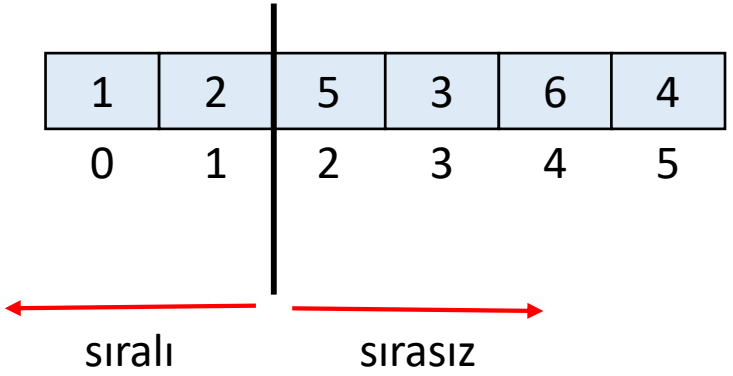
Seçmeli Sıralama




```
uzunluk = 6
```

Seçmeli Sıralama

3. Tur

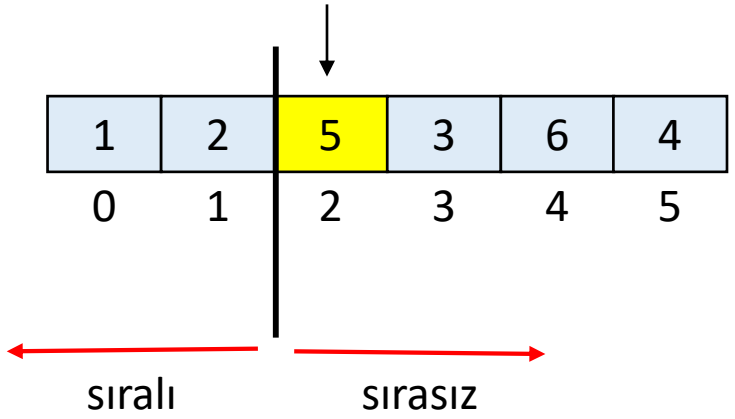



```
uzunluk = 6
```

Seçmeli Sıralama

3. Tur

```
min = 5
```

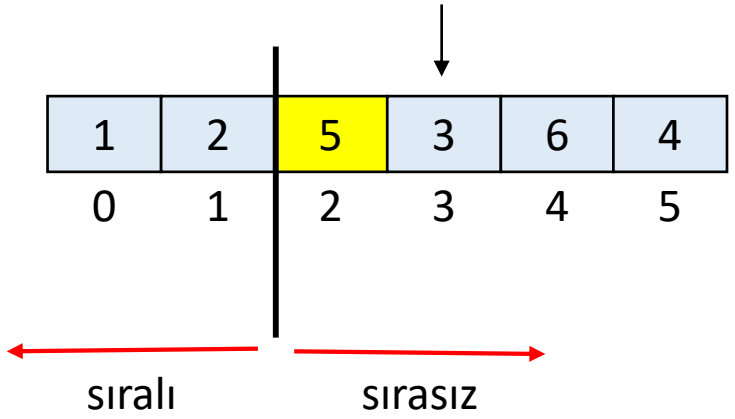



```
uzunluk = 6
```

Seçmeli Sıralama

3. Tur

```
min = 5
3 < min ?
```

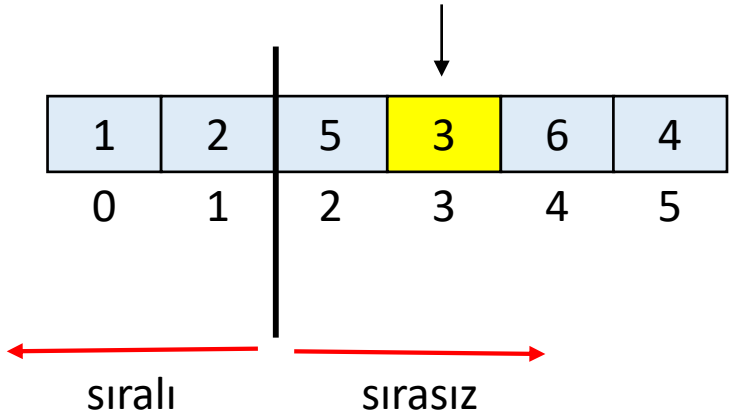



```
uzunluk = 6
```

Seçmeli Sıralama

3. Tur

```
min = 3
```

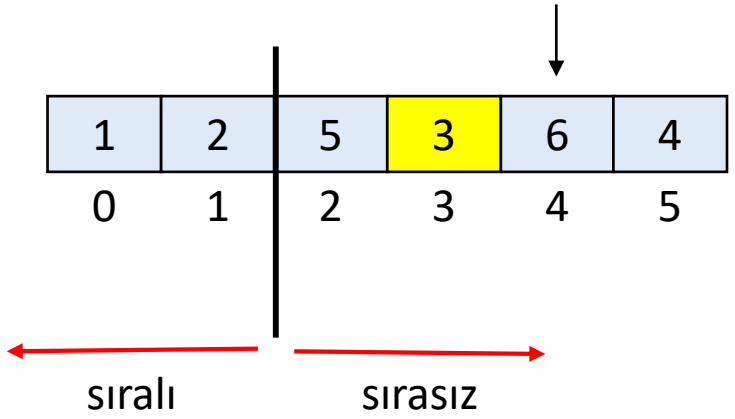



```
uzunluk = 6
```

Seçmeli Sıralama

3. Tur

```
min = 3
6 < min ?
```

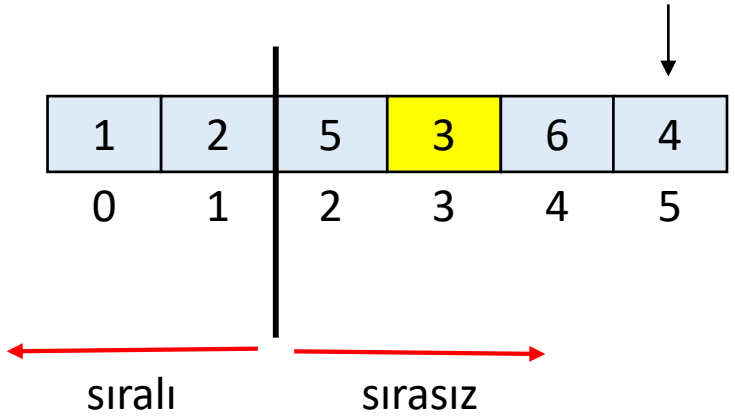



```
uzunluk = 6
```

Seçmeli Sıralama

3. Tur

```
min = 3
4 < min ?
```

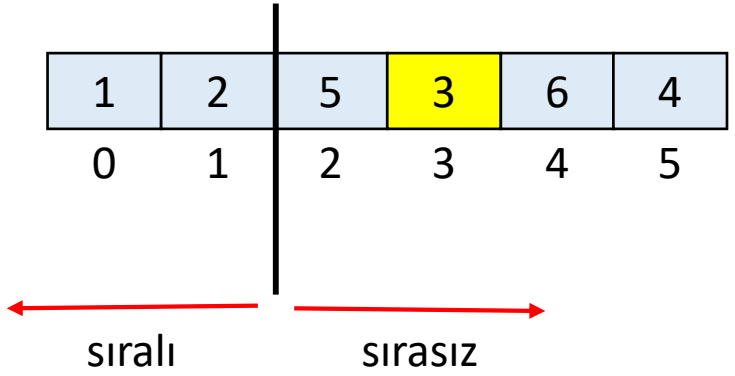



```
uzunluk = 6
```

Seçmeli Sıralama

3. Tur

```
min = 3
```

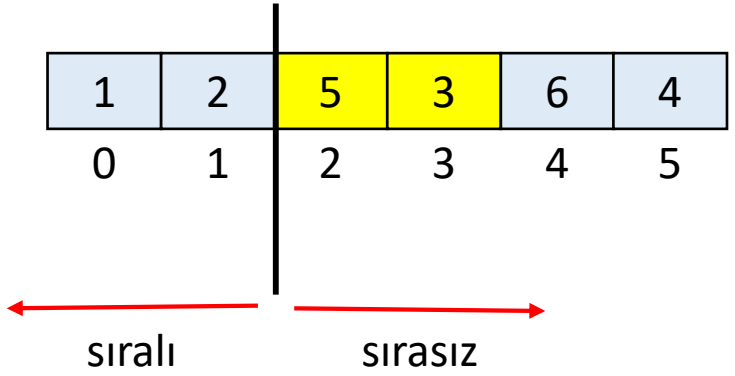



```
uzunluk = 6
```

Seçmeli Sıralama

3. Tur

```
min = 3
```

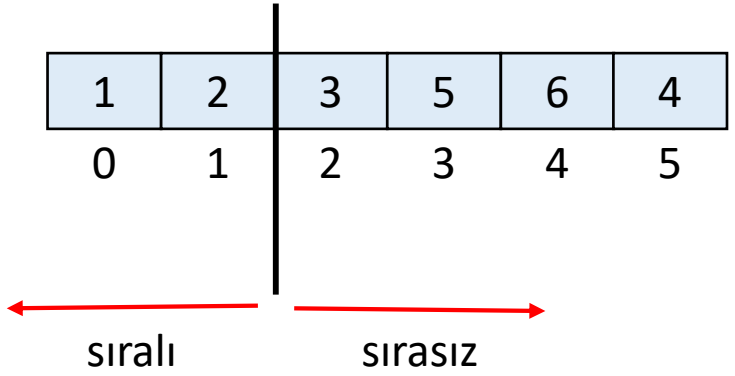



```
uzunluk = 6
```

Seçmeli Sıralama

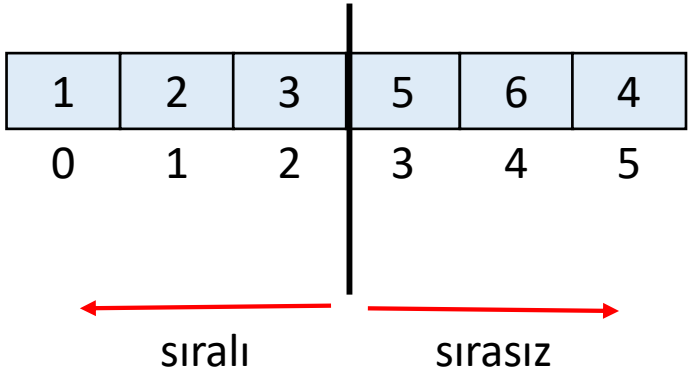
3. Tur

```
min = 3
```




```
uzunluk = 6
```

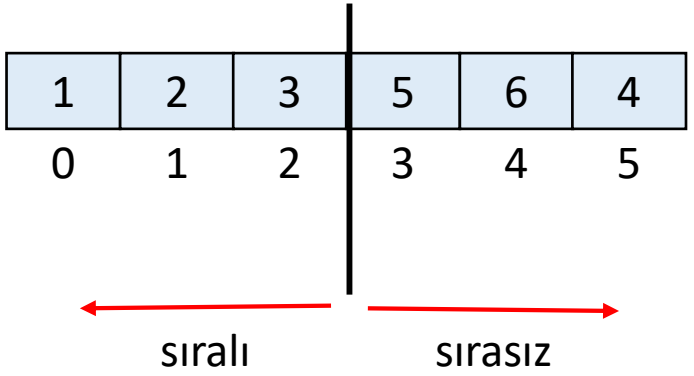
Seçmeli Sıralama




```
uzunluk = 6
```

Seçmeli Sıralama

4. Tur

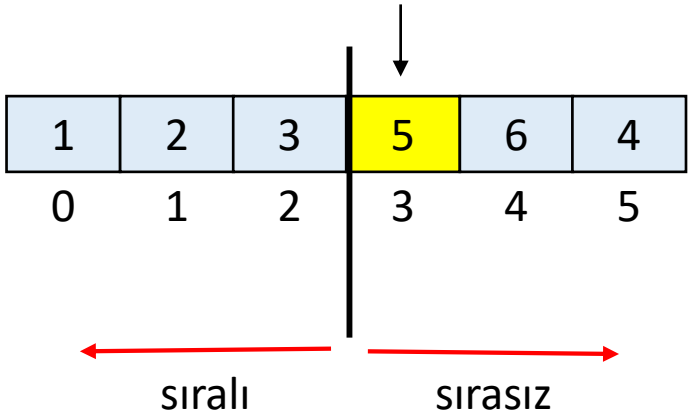



```
uzunluk = 6
```

Seçmeli Sıralama

4. Tur

```
min = 5
```

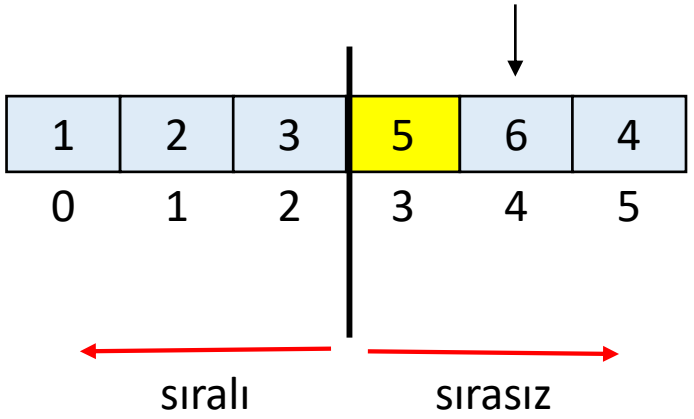



```
uzunluk = 6
```

Seçmeli Sıralama

4. Tur

```
min = 5
6 < min ?
```

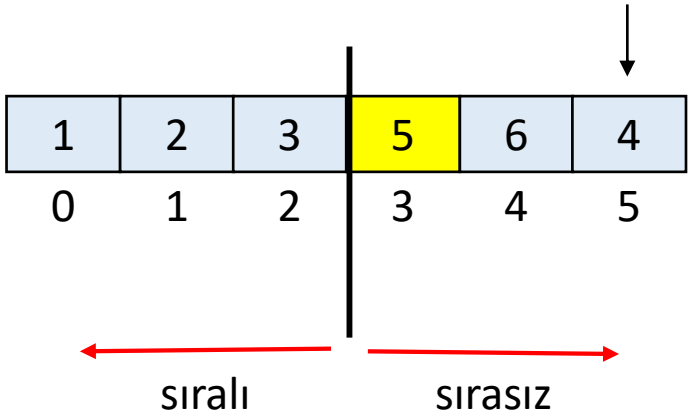



```
uzunluk = 6
```

Seçmeli Sıralama

4. Tur

```
min = 5
4 < min ?
```

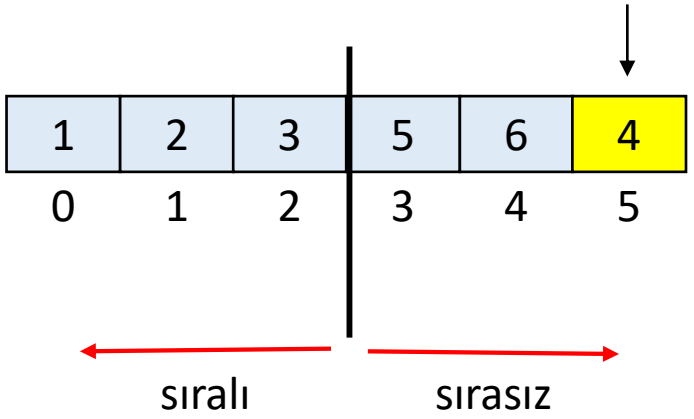



```
uzunluk = 6
```

Seçmeli Sıralama

4. Tur

```
min = 4
```

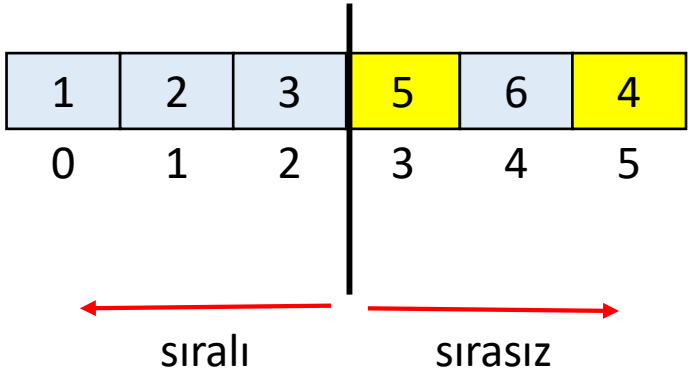



```
uzunluk = 6
```

Seçmeli Sıralama

4. Tur

```
min = 4
```

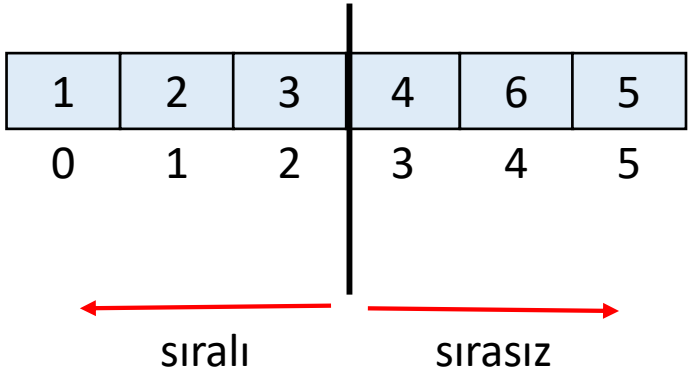



```
uzunluk = 6
```

Seçmeli Sıralama

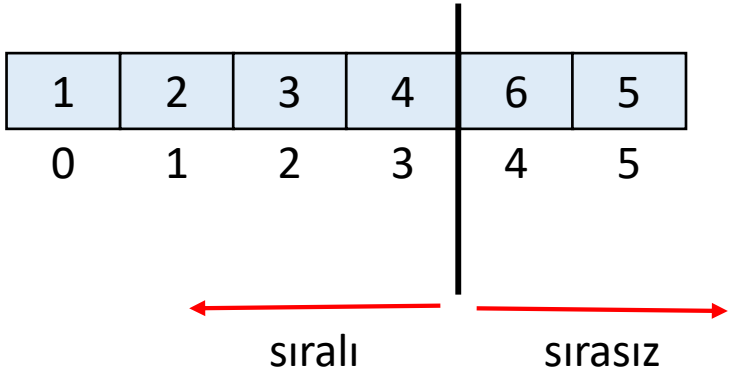
4. Tur

```
min = 4
```




```
uzunluk = 6
```

Seçmeli Sıralama

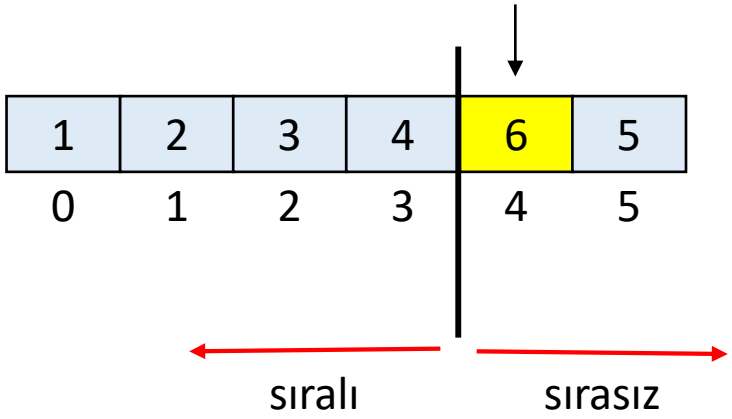



```
uzunluk = 6
```

Seçmeli Sıralama

5. Tur

```
min = 6
```

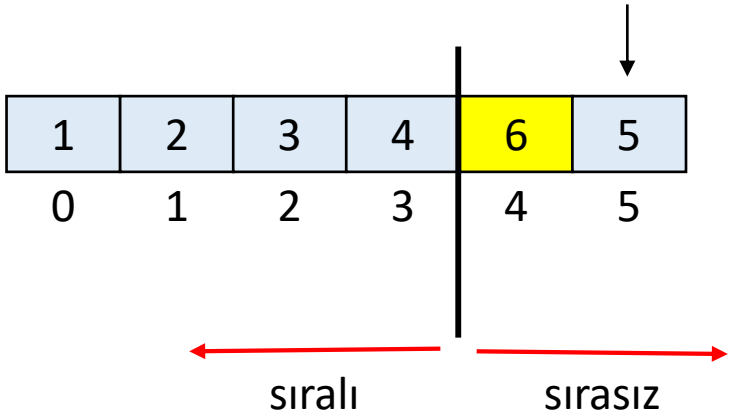



```
uzunluk = 6
```

Seçmeli Sıralama

5. Tur

```
min = 6
5 < min ?
```

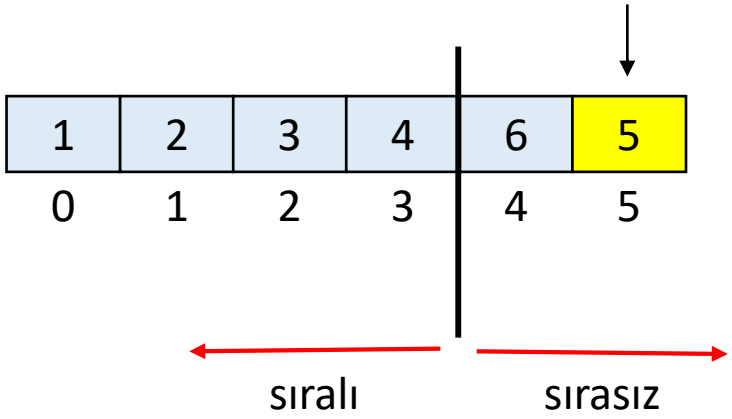



```
uzunluk = 6
```

Seçmeli Sıralama

5. Tur

```
min = 5
```

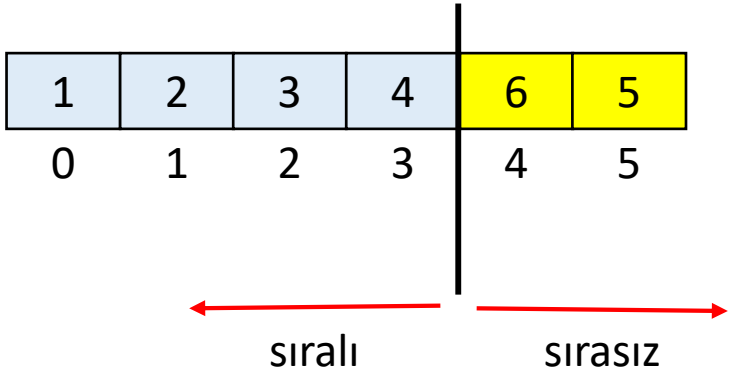



```
uzunluk = 6
```

Seçmeli Sıralama

5. Tur

```
min = 5
```

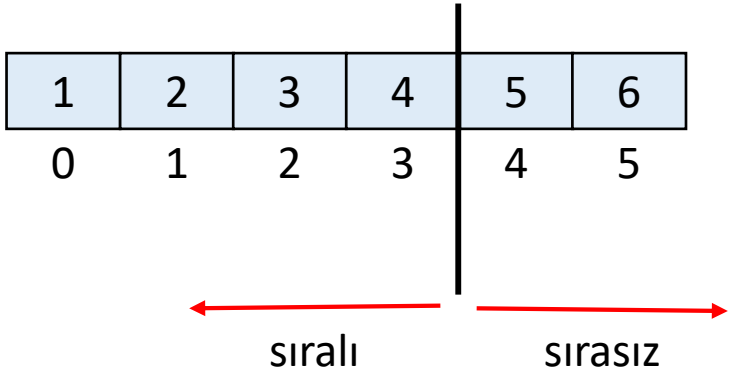



```
uzunluk = 6
```

Seçmeli Sıralama

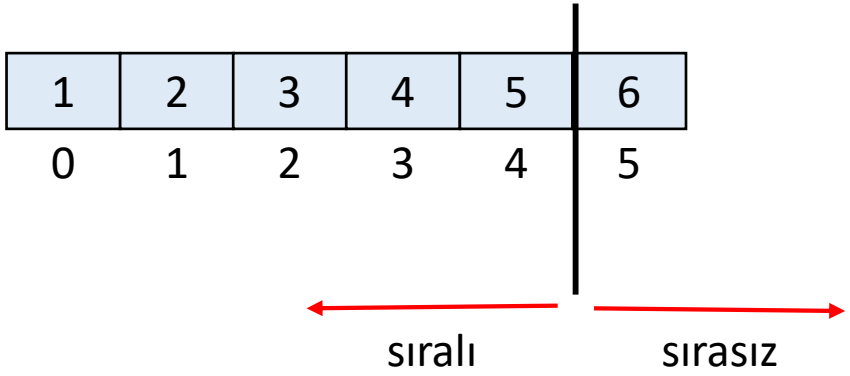
5. Tur

```
min = 5
```




```
uzunluk = 6
```

Seçmeli Sıralama




```
uzunluk = 6
```

Seçmeli Sıralama

1	2	3	4	5	6
0	1	2	3	4	5

```
uzunluk = 6
```

Seçmeli Sıralama

```
public void sort(int[] dizi) {
   int n = dizi.length;
   for(int i = 0; i < n - 1; i++) {
      int min = i;
      for(int j = i + 1; j < n; j++) {
         if(dizi[j] < dizi[min]) {
            min = j;
         }
      }
      int gecici = dizi[min];
      dizi[min] = dizi[i];
      dizi[i] = gecici;
   }
}
```

Seçmeli Sıralama

dizi[]	5	1	10	2	9
	0	1	2	3	4

```
public void sort(int[] dizi) {
   int n = dizi.length;
   for(int i = 0; i < n - 1; i++) {
      int min = i;
      for(int j = i + 1; j < n; j++) {
         if(dizi[j] < dizi[min]) {
            min = j;
         }
      }
      int gecici = dizi[min];
      dizi[min] = dizi[i];
      dizi[i] = gecici;
   }
}
```

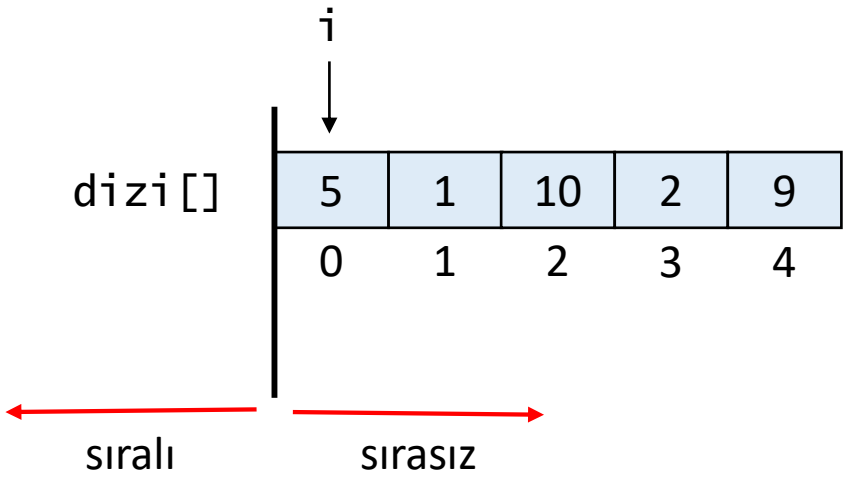
Seçmeli Sıralama

dizi[]	5	1	10	2	9
	0	1	2	3	4

```
public void sort(int[] dizi) {
   int n = dizi.length;
   for(int i = 0; i < n - 1; i++) {
      int min = i;
      for(int j = i + 1; j < n; j++) {
         if(dizi[j] < dizi[min]) {
            min = j;
         }
      }
      int gecici = dizi[min];
      dizi[min] = dizi[i];
      dizi[i] = gecici;
   }
}
```

n = 5

Seçmeli Sıralama

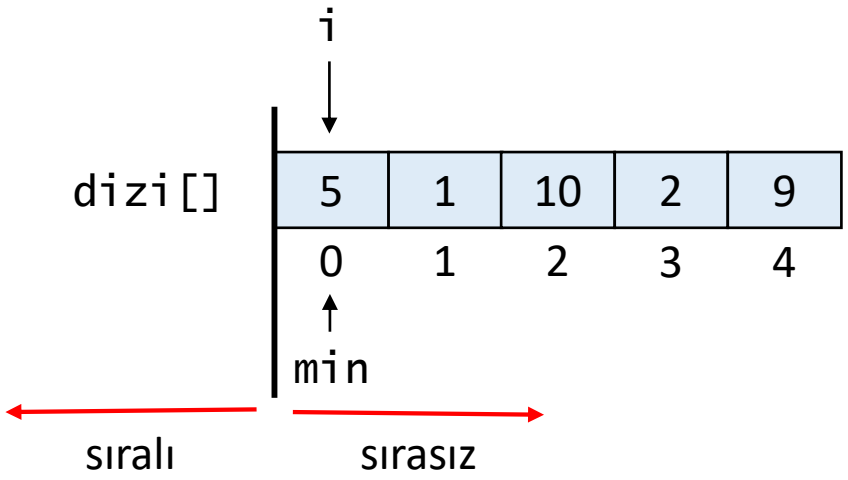


i = 0

n = 5

```
public void sort(int[] dizi) {
   int n = dizi.length;
   for(int i = 0; i < n - 1; i++) {
      int min = i;
      for(int j = i + 1; j < n; j++) {
         if(dizi[j] < dizi[min]) {
            min = j;
         }
      }
      int gecici = dizi[min];
      dizi[min] = dizi[i];
      dizi[i] = gecici;
   }
}
```

Seçmeli Sıralama

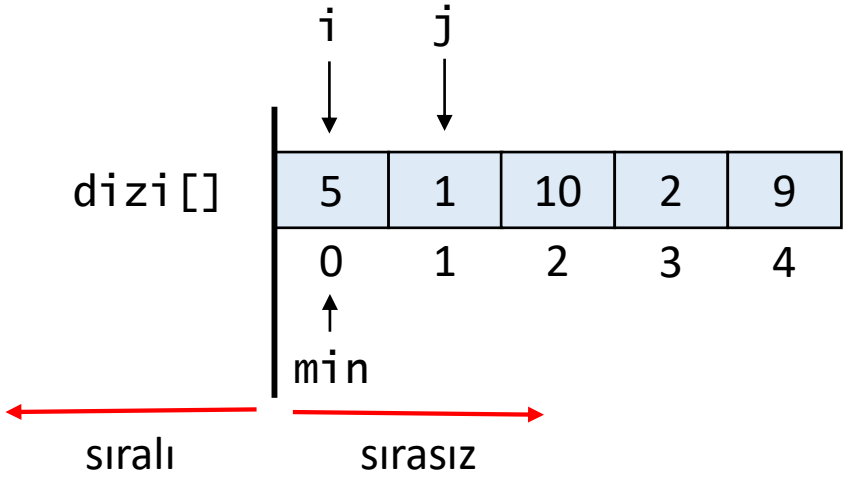


i = 0
min = 0

n = 5

```
public void sort(int[] dizi) {
   int n = dizi.length;
   for(int i = 0; i < n - 1; i++) {
      int min = i;
      for(int j = i + 1; j < n; j++) {
         if(dizi[j] < dizi[min]) {
            min = j;
         }
      }
      int gecici = dizi[min];
      dizi[min] = dizi[i];
      dizi[i] = gecici;
   }
}
```

Seçmeli Sıralama

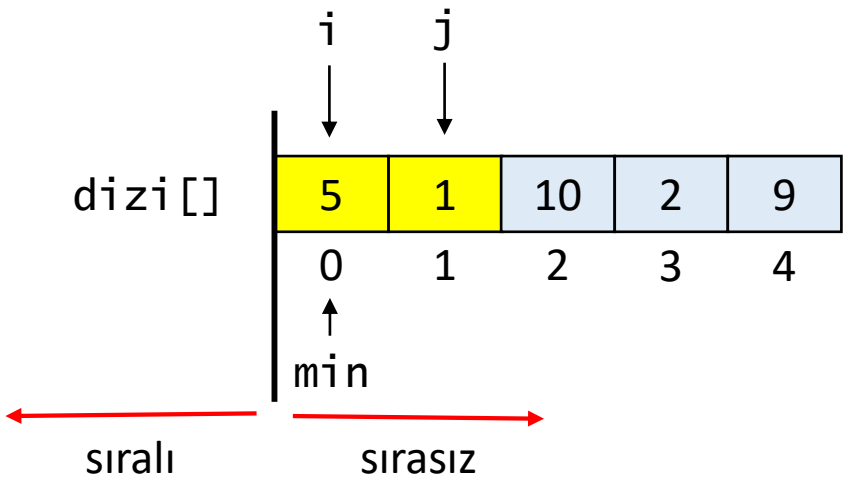


```
i = 0
min = 0
j = 1
```

```
n = 5
```

```
public void sort(int[] dizi) {
   int n = dizi.length;
   for(int i = 0; i < n - 1; i++) {
      int min = i;
      for(int j = i + 1; j < n; j++) {
         if(dizi[j] < dizi[min]) {
            min = j;
         }
      }
      int gecici = dizi[min];
      dizi[min] = dizi[i];
      dizi[i] = gecici;
   }
}
```

Seçmeli Sıralama

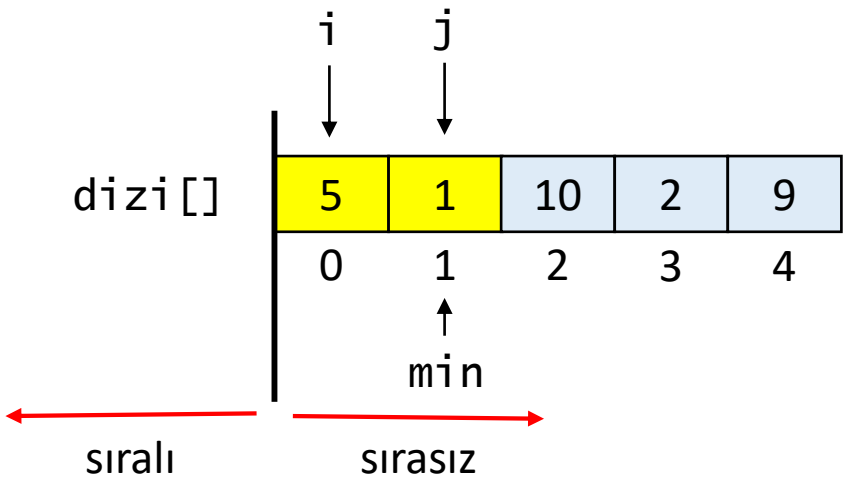


```
i = 0
min = 0
j = 1
```

```
n = 5
```

```
public void sort(int[] dizi) {
   int n = dizi.length;
   for(int i = 0; i < n - 1; i++) {
      int min = i;
      for(int j = i + 1; j < n; j++) {
         if(dizi[j] < dizi[min]) {
            min = j;
         }
      }
      int gecici = dizi[min];
      dizi[min] = dizi[i];
      dizi[i] = gecici;
   }
}
```

Seçmeli Sıralama

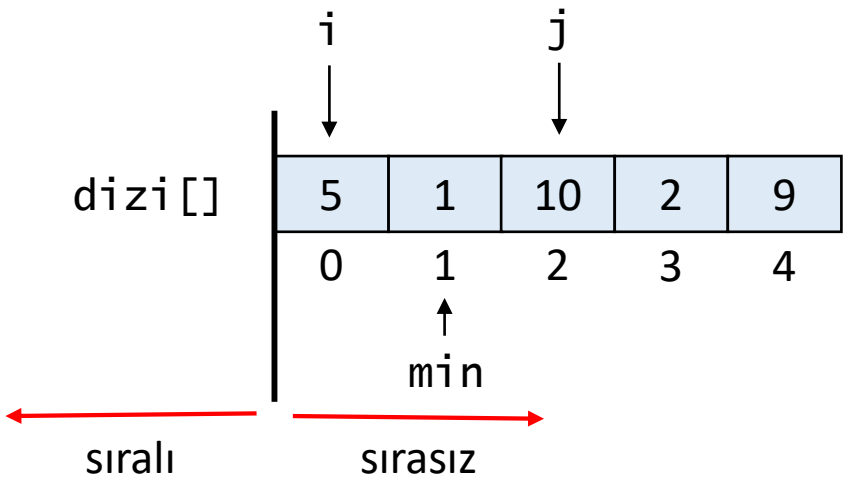


```
i = 0
min = 1
j = 1
```

```
n = 5
```

```
public void sort(int[] dizi) {
   int n = dizi.length;
   for(int i = 0; i < n - 1; i++) {
      int min = i;
      for(int j = i + 1; j < n; j++) {
         if(dizi[j] < dizi[min]) {
            min = j;
         }
      }
      int gecici = dizi[min];
      dizi[min] = dizi[i];
      dizi[i] = gecici;
   }
}
```

Seçmeli Sıralama

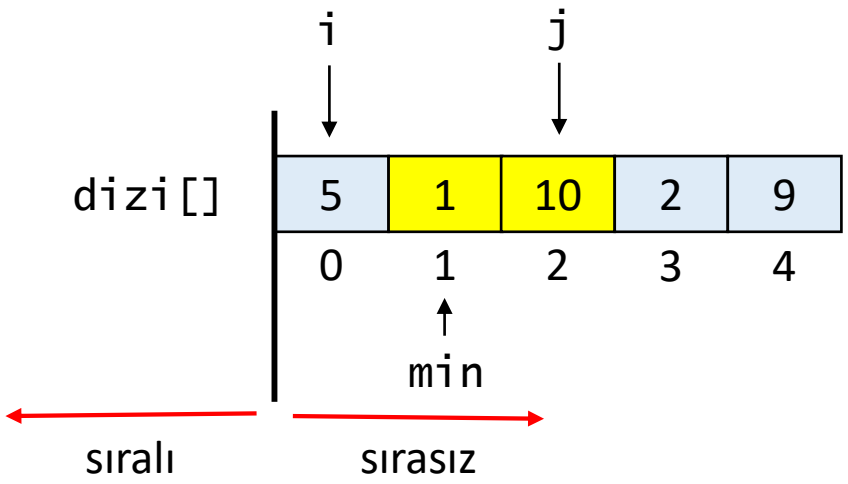


```
i = 0
min = 1
j = 2
```

```
n = 5
```

```
public void sort(int[] dizi) {
   int n = dizi.length;
   for(int i = 0; i < n - 1; i++) {
      int min = i;
      for(int j = i + 1; j < n; j++) {
         if(dizi[j] < dizi[min]) {
            min = j;
         }
      }
      int gecici = dizi[min];
      dizi[min] = dizi[i];
      dizi[i] = gecici;
   }
}
```

Seçmeli Sıralama

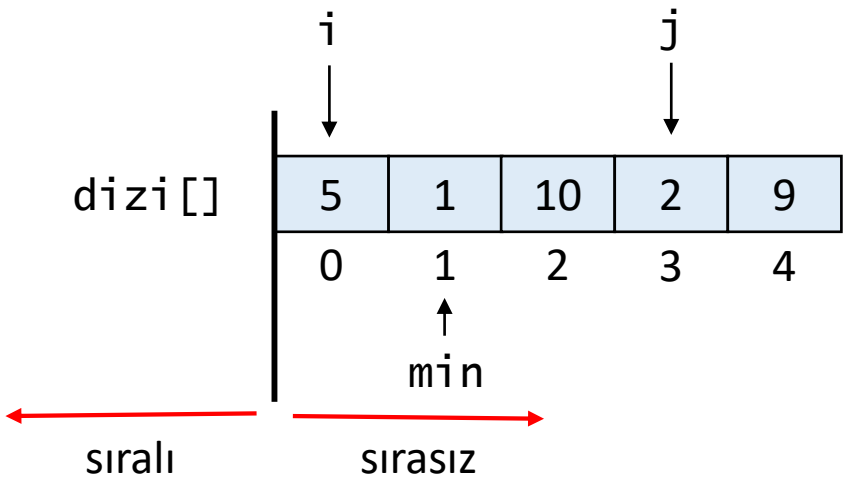


```
i = 0
min = 1
j = 2
```

```
n = 5
```

```
public void sort(int[] dizi) {
   int n = dizi.length;
   for(int i = 0; i < n - 1; i++) {
      int min = i;
      for(int j = i + 1; j < n; j++) {
         if(dizi[j] < dizi[min]) {
            min = j;
         }
      }
      int gecici = dizi[min];
      dizi[min] = dizi[i];
      dizi[i] = gecici;
   }
}
```

Seçmeli Sıralama

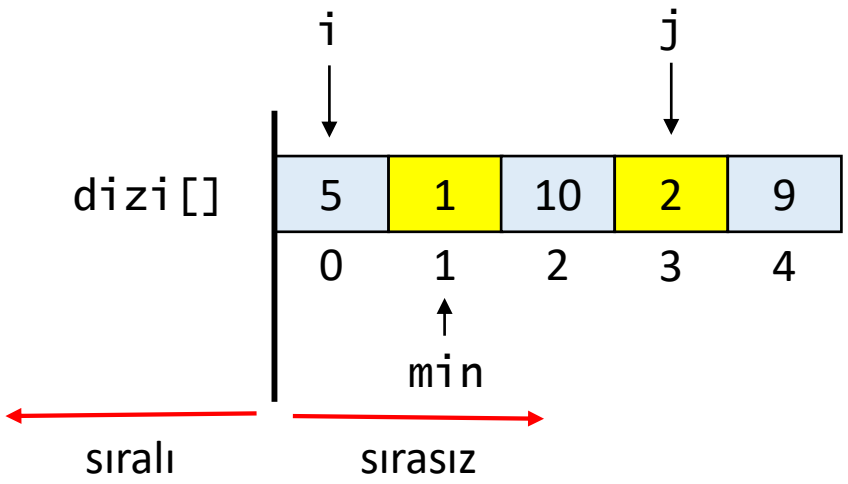


```
i = 0
min = 1
j = 3
```

```
n = 5
```

```
public void sort(int[] dizi) {
   int n = dizi.length;
   for(int i = 0; i < n - 1; i++) {
      int min = i;
      for(int j = i + 1; j < n; j++) {
         if(dizi[j] < dizi[min]) {
            min = j;
         }
      }
      int gecici = dizi[min];
      dizi[min] = dizi[i];
      dizi[i] = gecici;
   }
}
```

Seçmeli Sıralama

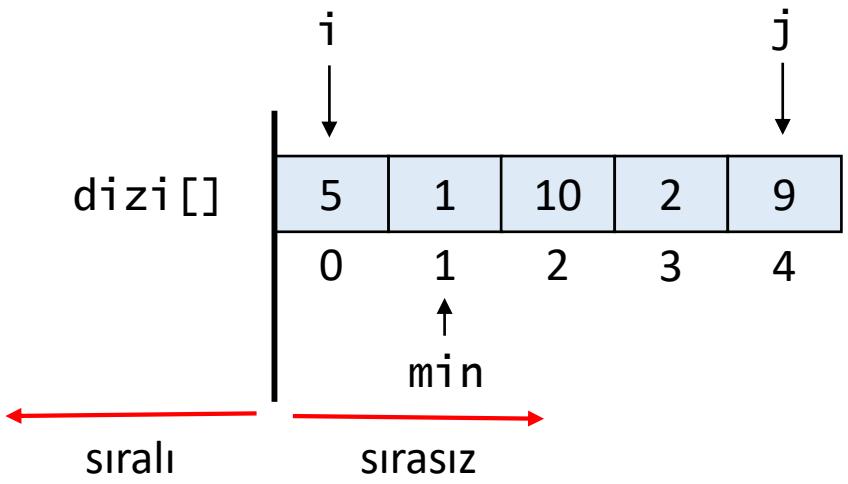


```
i = 0
min = 1
j = 3
```

```
n = 5
```

```
public void sort(int[] dizi) {
   int n = dizi.length;
   for(int i = 0; i < n - 1; i++) {
      int min = i;
      for(int j = i + 1; j < n; j++) {
         if(dizi[j] < dizi[min]) {
            min = j;
         }
      }
      int gecici = dizi[min];
      dizi[min] = dizi[i];
      dizi[i] = gecici;
   }
}
```

Seçmeli Sıralama

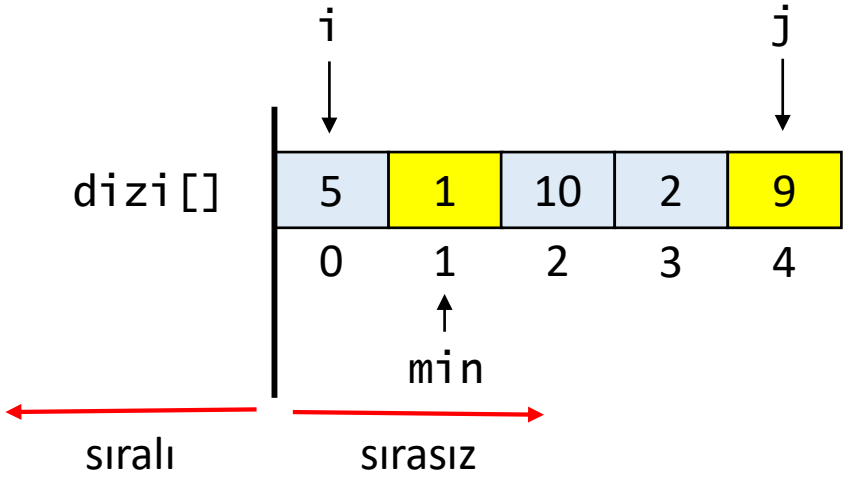


```
i = 0
min = 1
j = 4
```

```
n = 5
```

```
public void sort(int[] dizi) {
   int n = dizi.length;
   for(int i = 0; i < n - 1; i++) {
      int min = i;
      for(int j = i + 1; j < n; j++) {
         if(dizi[j] < dizi[min]) {
            min = j;
         }
      }
      int gecici = dizi[min];
      dizi[min] = dizi[i];
      dizi[i] = gecici;
   }
}
```

Seçmeli Sıralama

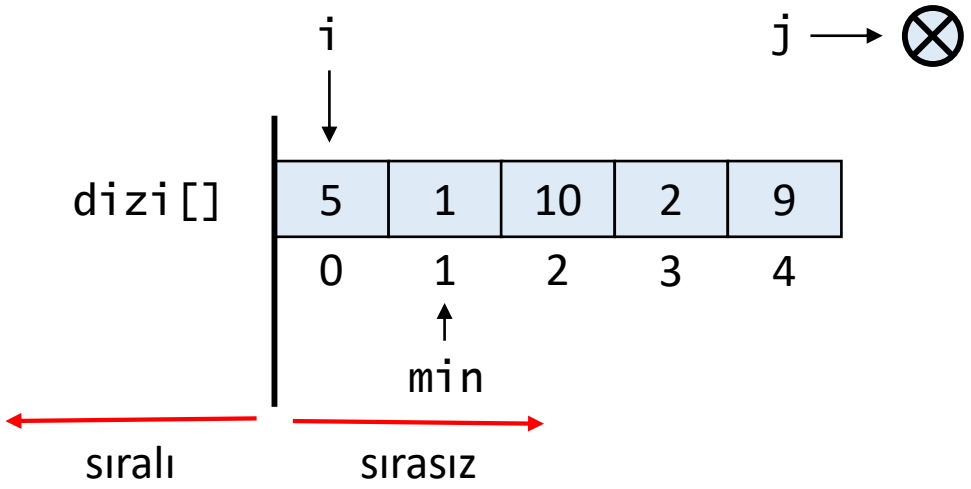


```
i = 0
min = 1
j = 4
```

```
n = 5
```

```
public void sort(int[] dizi) {
   int n = dizi.length;
   for(int i = 0; i < n - 1; i++) {
      int min = i;
      for(int j = i + 1; j < n; j++) {
         if(dizi[j] < dizi[min]) {
            min = j;
         }
      }
      int gecici = dizi[min];
      dizi[min] = dizi[i];
      dizi[i] = gecici;
   }
}
```

Seçmeli Sıralama

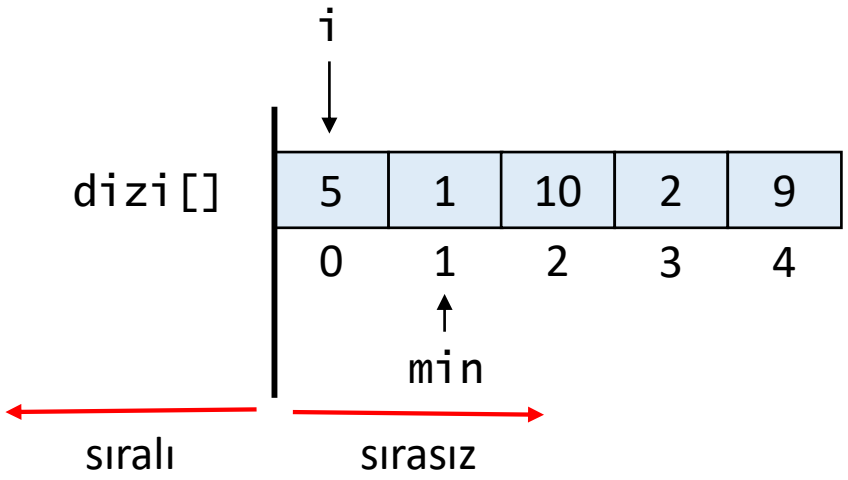


```
i = 0
min = 1
j = 5
```

```
n = 5
```

```
public void sort(int[] dizi) {
   int n = dizi.length;
   for(int i = 0; i < n - 1; i++) {
      int min = i;
      for(int j = i + 1; j < n; j++) {
         if(dizi[j] < dizi[min]) {
            min = j;
         }
      }
      int gecici = dizi[min];
      dizi[min] = dizi[i];
      dizi[i] = gecici;
   }
}
```

Seçmeli Sıralama

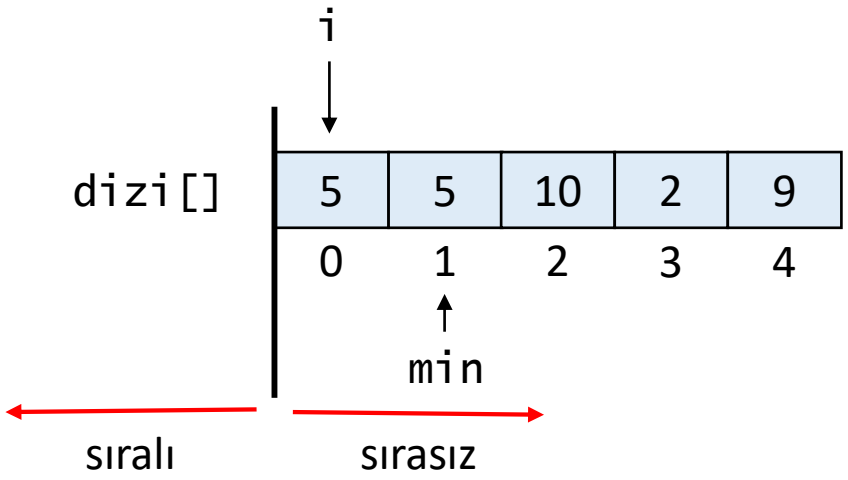


```
i = 0
min = 1
gecici = 1
```

```
n = 5
```

```
public void sort(int[] dizi) {
   int n = dizi.length;
   for(int i = 0; i < n - 1; i++) {
      int min = i;
      for(int j = i + 1; j < n; j++) {
         if(dizi[j] < dizi[min]) {
            min = j;
         }
      }
      int gecici = dizi[min];
      dizi[min] = dizi[i];
      dizi[i] = gecici;
   }
}
```

Seçmeli Sıralama

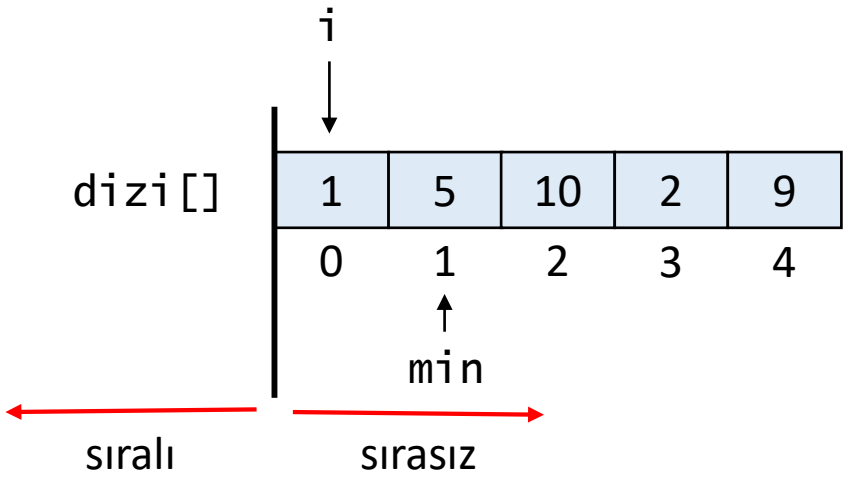


```
i = 0
min = 1
gecici = 1
```

```
n = 5
```

```
public void sort(int[] dizi) {
   int n = dizi.length;
   for(int i = 0; i < n - 1; i++) {
      int min = i;
      for(int j = i + 1; j < n; j++) {
         if(dizi[j] < dizi[min]) {
            min = j;
         }
      }
      int gecici = dizi[min];
      dizi[min] = dizi[i];
      dizi[i] = gecici;
   }
}
```

Seçmeli Sıralama

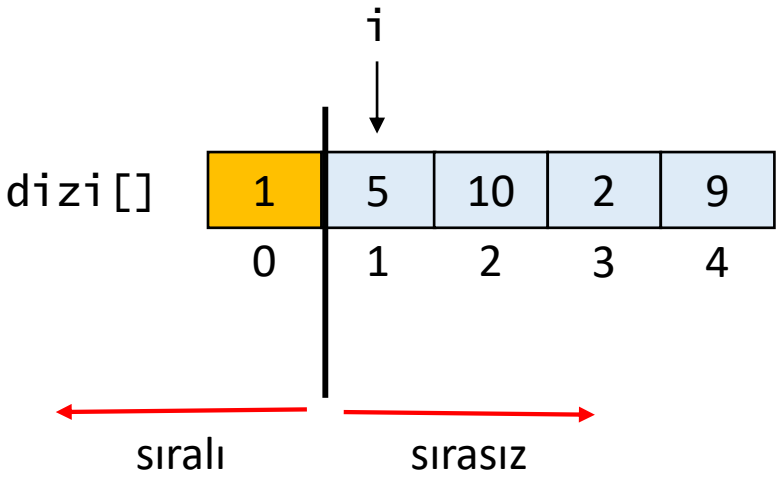


```
i = 0
min = 1
gecici = 1
```

```
n = 5
```

```
public void sort(int[] dizi) {
   int n = dizi.length;
   for(int i = 0; i < n - 1; i++) {
      int min = i;
      for(int j = i + 1; j < n; j++) {
         if(dizi[j] < dizi[min]) {
            min = j;
         }
      }
      int gecici = dizi[min];
      dizi[min] = dizi[i];
      dizi[i] = gecici;
   }
}
```

Seçmeli Sıralama

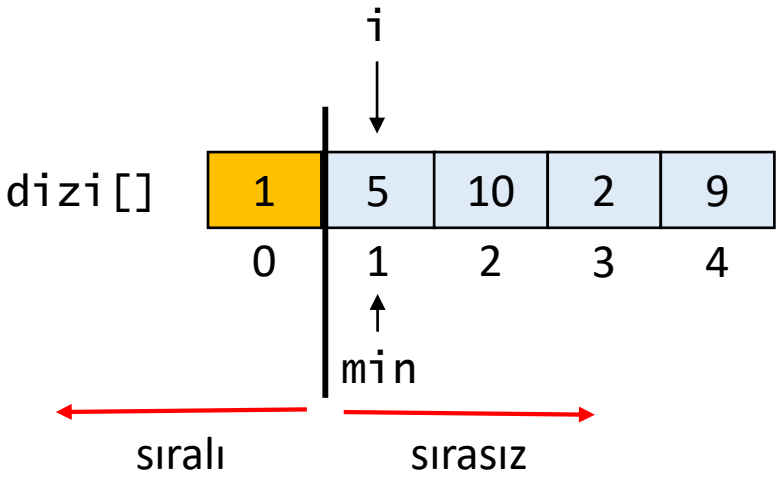


i = 1

n = 5

```
public void sort(int[] dizi) {
   int n = dizi.length;
   for(int i = 0; i < n - 1; i++) {
      int min = i;
      for(int j = i + 1; j < n; j++) {
         if(dizi[j] < dizi[min]) {
            min = j;
         }
      }
      int gecici = dizi[min];
      dizi[min] = dizi[i];
      dizi[i] = gecici;
   }
}
```

Seçmeli Sıralama

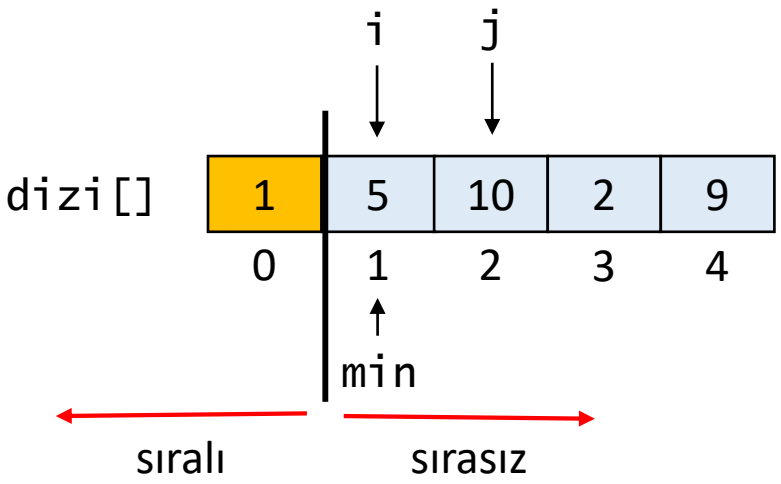


```
i = 1
min = 1
```

```
n = 5
```

```
public void sort(int[] dizi) {
   int n = dizi.length;
   for(int i = 0; i < n - 1; i++) {
      int min = i;
      for(int j = i + 1; j < n; j++) {
         if(dizi[j] < dizi[min]) {
            min = j;
         }
      }
      int gecici = dizi[min];
      dizi[min] = dizi[i];
      dizi[i] = gecici;
   }
}
```

Seçmeli Sıralama

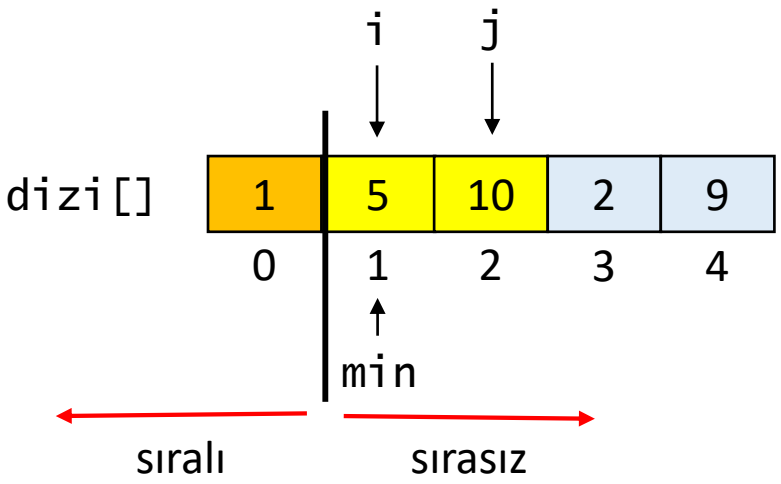


```
i = 1
min = 1
j = 2
```

```
n = 5
```

```
public void sort(int[] dizi) {
   int n = dizi.length;
   for(int i = 0; i < n - 1; i++) {
      int min = i;
      for(int j = i + 1; j < n; j++) {
         if(dizi[j] < dizi[min]) {
            min = j;
         }
      }
      int gecici = dizi[min];
      dizi[min] = dizi[i];
      dizi[i] = gecici;
   }
}
```

Seçmeli Sıralama

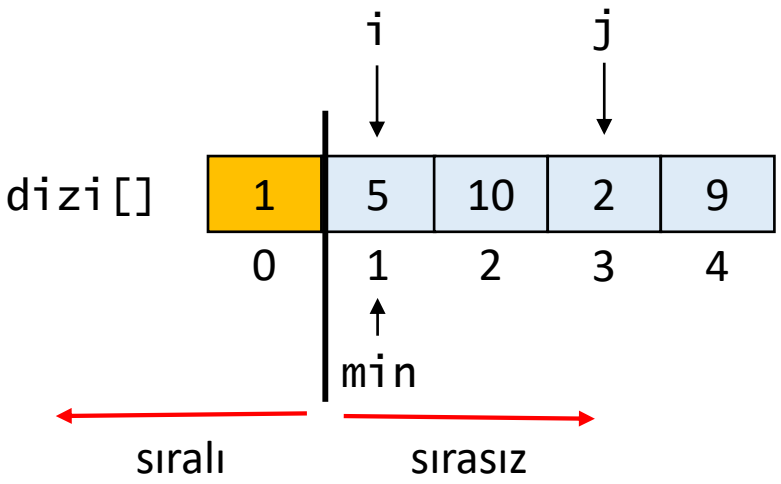


```
i = 1
min = 1
j = 2
```

```
n = 5
```

```
public void sort(int[] dizi) {
   int n = dizi.length;
   for(int i = 0; i < n - 1; i++) {
      int min = i;
      for(int j = i + 1; j < n; j++) {
         if(dizi[j] < dizi[min]) {
            min = j;
         }
      }
      int gecici = dizi[min];
      dizi[min] = dizi[i];
      dizi[i] = gecici;
   }
}
```

Seçmeli Sıralama

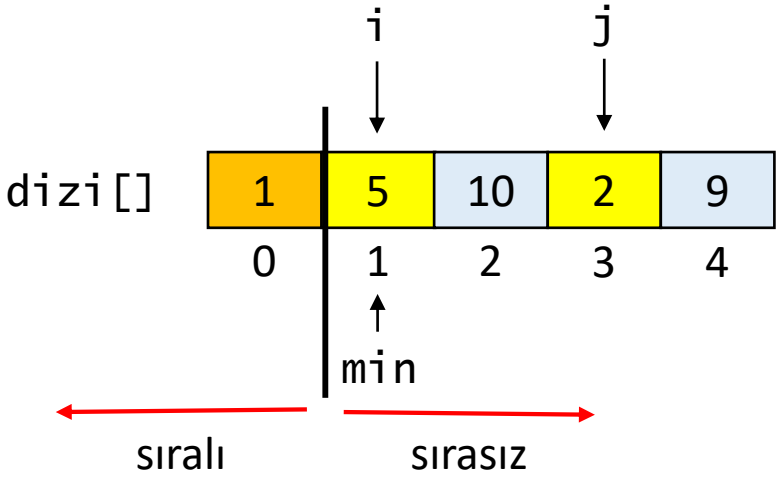


```
i = 1
min = 1
j = 3
```

```
n = 5
```

```
public void sort(int[] dizi) {
   int n = dizi.length;
   for(int i = 0; i < n - 1; i++) {
      int min = i;
      for(int j = i + 1; j < n; j++) {
         if(dizi[j] < dizi[min]) {
            min = j;
         }
      }
      int gecici = dizi[min];
      dizi[min] = dizi[i];
      dizi[i] = gecici;
   }
}
```

Seçmeli Sıralama

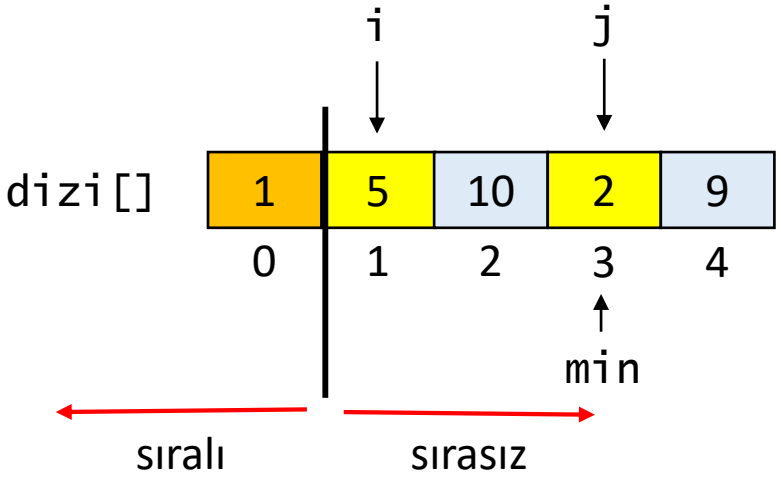


```
i = 1
min = 1
j = 3
```

```
n = 5
```

```
public void sort(int[] dizi) {
   int n = dizi.length;
   for(int i = 0; i < n - 1; i++) {
      int min = i;
      for(int j = i + 1; j < n; j++) {
         if(dizi[j] < dizi[min]) {
            min = j;
         }
      }
      int gecici = dizi[min];
      dizi[min] = dizi[i];
      dizi[i] = gecici;
   }
}
```

Seçmeli Sıralama

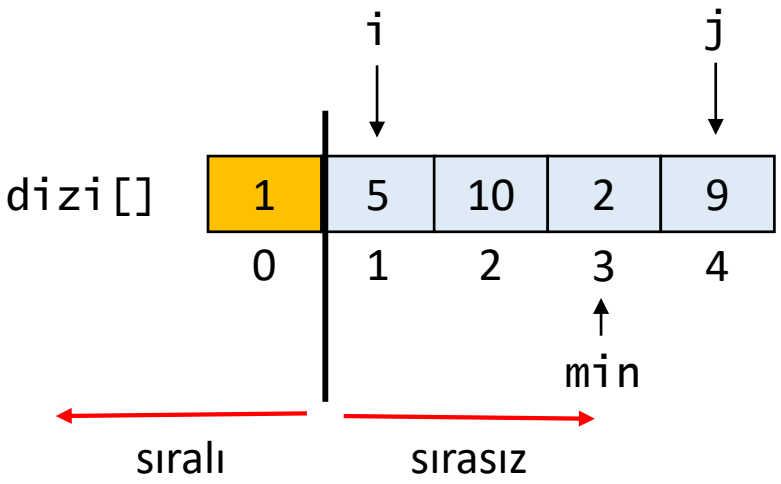


```
i = 1
min = 3
j = 3
```

```
n = 5
```

```
public void sort(int[] dizi) {
   int n = dizi.length;
   for(int i = 0; i < n - 1; i++) {
      int min = i;
      for(int j = i + 1; j < n; j++) {
         if(dizi[j] < dizi[min]) {
            min = j;
         }
      }
      int gecici = dizi[min];
      dizi[min] = dizi[i];
      dizi[i] = gecici;
   }
}
```

Seçmeli Sıralama

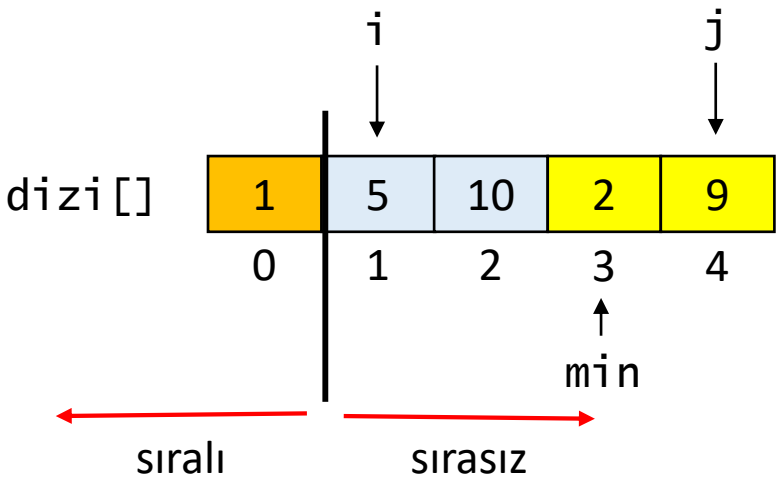


```
i = 1
min = 3
j = 4
```

```
n = 5
```

```
public void sort(int[] dizi) {
   int n = dizi.length;
   for(int i = 0; i < n - 1; i++) {
      int min = i;
      for(int j = i + 1; j < n; j++) {
         if(dizi[j] < dizi[min]) {
            min = j;
         }
      }
      int gecici = dizi[min];
      dizi[min] = dizi[i];
      dizi[i] = gecici;
   }
}
```

Seçmeli Sıralama

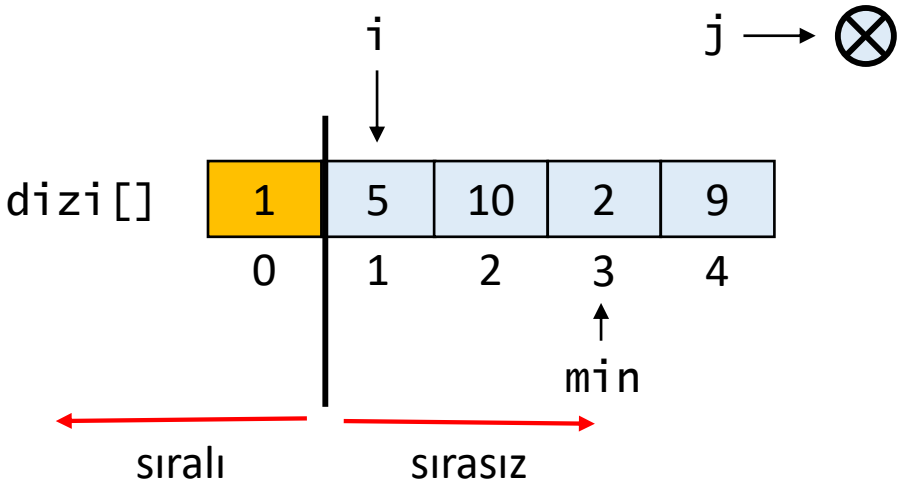


```
i = 1
min = 3
j = 4
```

```
n = 5
```

```
public void sort(int[] dizi) {
   int n = dizi.length;
   for(int i = 0; i < n - 1; i++) {
      int min = i;
      for(int j = i + 1; j < n; j++) {
         if(dizi[j] < dizi[min]) {
            min = j;
         }
      }
      int gecici = dizi[min];
      dizi[min] = dizi[i];
      dizi[i] = gecici;
   }
}
```

Seçmeli Sıralama

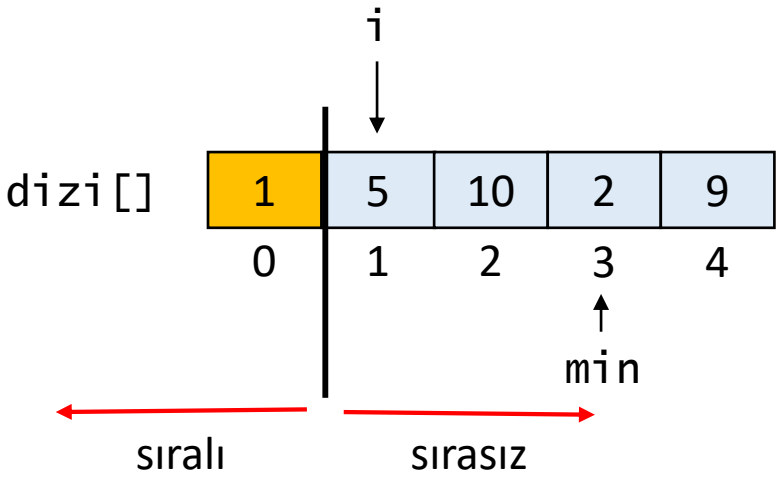


```
i = 1
min = 3
j = 5
```

```
n = 5
```

```
public void sort(int[] dizi) {
   int n = dizi.length;
   for(int i = 0; i < n - 1; i++) {
      int min = i;
      for(int j = i + 1; j < n; j++) {
         if(dizi[j] < dizi[min]) {
            min = j;
         }
      }
      int gecici = dizi[min];
      dizi[min] = dizi[i];
      dizi[i] = gecici;
   }
}
```

Seçmeli Sıralama

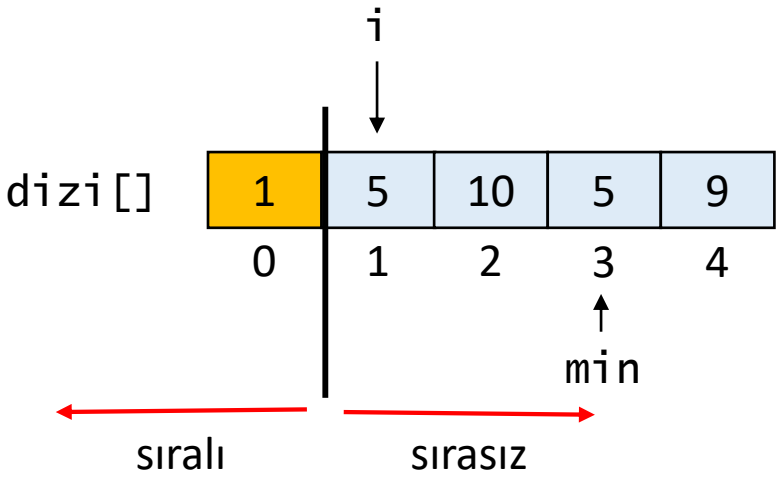


```
i = 1
min = 3
gecici = 2
```

```
n = 5
```

```
public void sort(int[] dizi) {
   int n = dizi.length;
   for(int i = 0; i < n - 1; i++) {
      int min = i;
      for(int j = i + 1; j < n; j++) {
         if(dizi[j] < dizi[min]) {
            min = j;
         }
      }
      int gecici = dizi[min];
      dizi[min] = dizi[i];
      dizi[i] = gecici;
   }
}
```

Seçmeli Sıralama

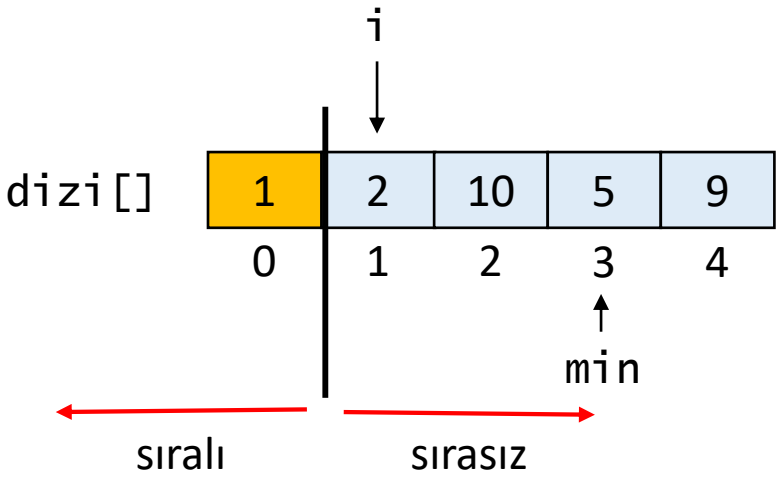


```
i = 1
min = 3
gecici = 2
```

```
n = 5
```

```
public void sort(int[] dizi) {
   int n = dizi.length;
   for(int i = 0; i < n - 1; i++) {
      int min = i;
      for(int j = i + 1; j < n; j++) {
         if(dizi[j] < dizi[min]) {
            min = j;
         }
      }
      int gecici = dizi[min];
      dizi[min] = dizi[i];
      dizi[i] = gecici;
   }
}
```

Seçmeli Sıralama

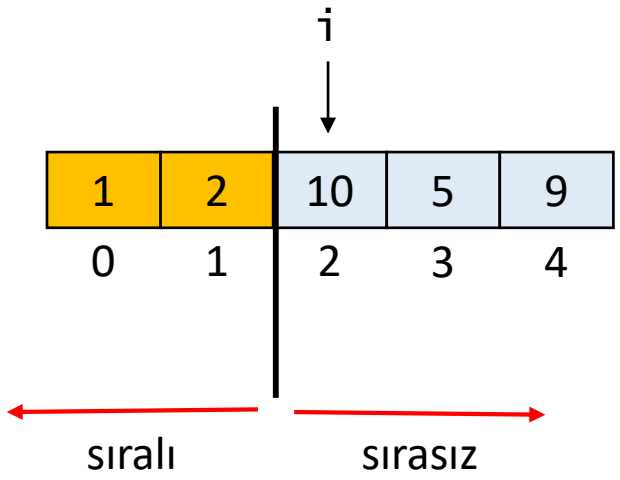


```
i = 1
min = 3
gecici = 2
```

```
n = 5
```

```
public void sort(int[] dizi) {
   int n = dizi.length;
   for(int i = 0; i < n - 1; i++) {
      int min = i;
      for(int j = i + 1; j < n; j++) {
         if(dizi[j] < dizi[min]) {
            min = j;
         }
      }
      int gecici = dizi[min];
      dizi[min] = dizi[i];
      dizi[i] = gecici;
   }
}
```

Seçmeli Sıralama

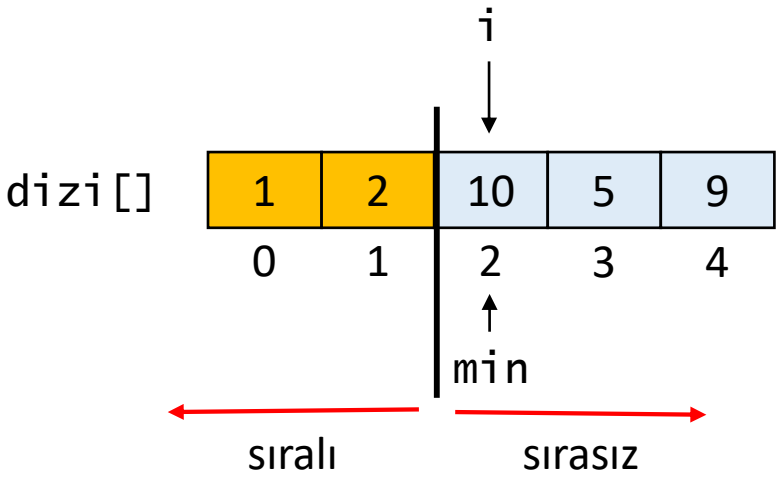


i = 2

n = 5

```
public void sort(int[] dizi) {
   int n = dizi.length;
   for(int i = 0; i < n - 1; i++) {
      int min = i;
      for(int j = i + 1; j < n; j++) {
         if(dizi[j] < dizi[min]) {
            min = j;
         }
      }
      int gecici = dizi[min];
      dizi[min] = dizi[i];
      dizi[i] = gecici;
   }
}
```

Seçmeli Sıralama

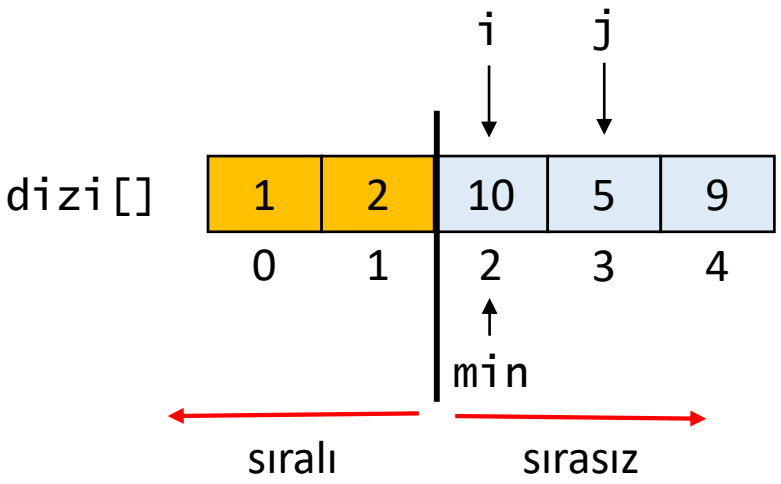


```
i = 2
min = 2
```

```
n = 5
```

```
public void sort(int[] dizi) {
   int n = dizi.length;
   for(int i = 0; i < n - 1; i++) {
      int min = i;
      for(int j = i + 1; j < n; j++) {
         if(dizi[j] < dizi[min]) {
            min = j;
         }
      }
      int gecici = dizi[min];
      dizi[min] = dizi[i];
      dizi[i] = gecici;
   }
}
```

Seçmeli Sıralama

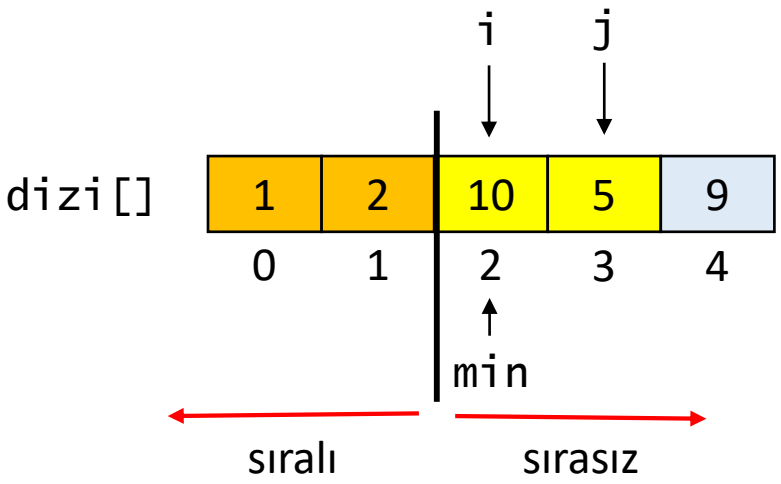


```
i = 2
min = 2
j = 3
```

```
n = 5
```

```
public void sort(int[] dizi) {
   int n = dizi.length;
   for(int i = 0; i < n - 1; i++) {
      int min = i;
      for(int j = i + 1; j < n; j++) {
         if(dizi[j] < dizi[min]) {
            min = j;
         }
      }
      int gecici = dizi[min];
      dizi[min] = dizi[i];
      dizi[i] = gecici;
   }
}
```

Seçmeli Sıralama



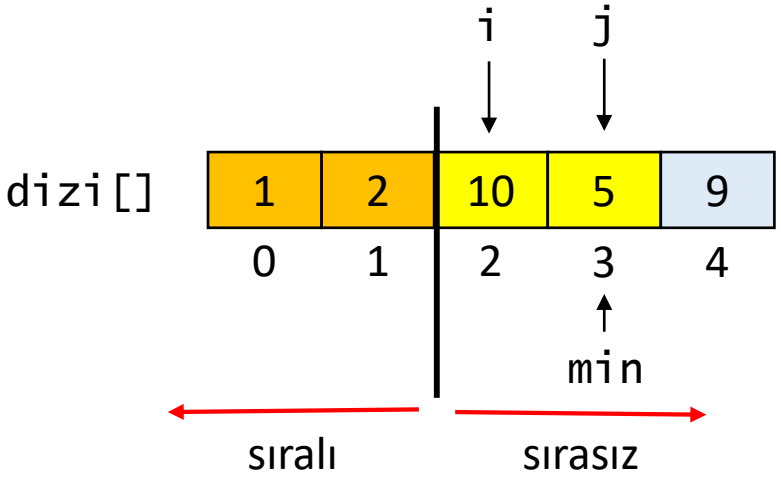
```
i = 2
min = 2
j = 3
```

```
n = 5
```

→

```
public void sort(int[] dizi) {
   int n = dizi.length;
   for(int i = 0; i < n - 1; i++) {
      int min = i;
      for(int j = i + 1; j < n; j++) {
         if(dizi[j] < dizi[min]) {
            min = j;
         }
      }
      int gecici = dizi[min];
      dizi[min] = dizi[i];
      dizi[i] = gecici;
   }
}
```

Seçmeli Sıralama

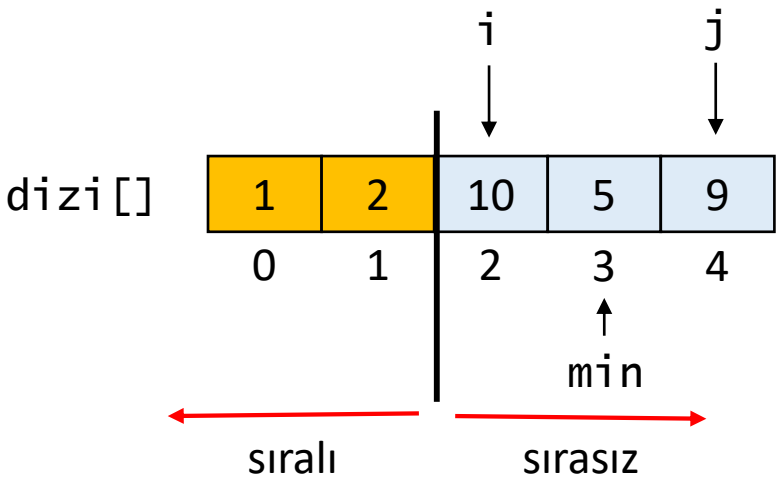


```
i = 2
min = 3
j = 3
```

```
n = 5
```

```
public void sort(int[] dizi) {
   int n = dizi.length;
   for(int i = 0; i < n - 1; i++) {
      int min = i;
      for(int j = i + 1; j < n; j++) {
         if(dizi[j] < dizi[min]) {
            min = j;
         }
      }
      int gecici = dizi[min];
      dizi[min] = dizi[i];
      dizi[i] = gecici;
   }
}
```

Seçmeli Sıralama

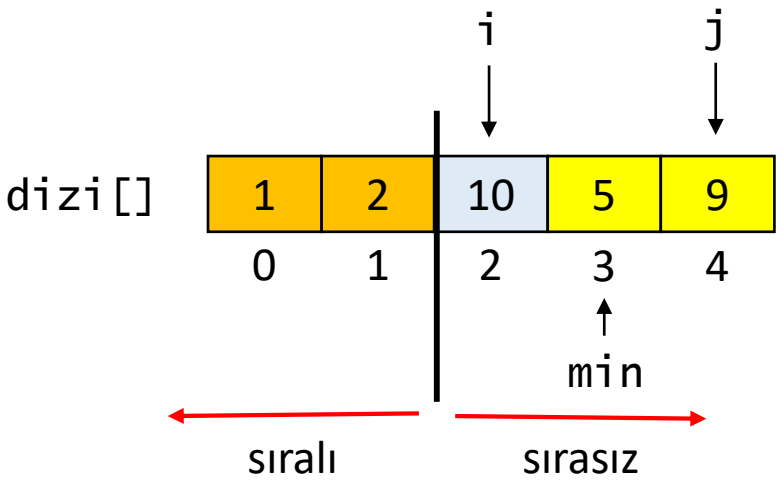


```
i = 2
min = 3
j = 4
```

```
n = 5
```

```
public void sort(int[] dizi) {
   int n = dizi.length;
   for(int i = 0; i < n - 1; i++) {
      int min = i;
      for(int j = i + 1; j < n; j++) {
         if(dizi[j] < dizi[min]) {
            min = j;
         }
      }
      int gecici = dizi[min];
      dizi[min] = dizi[i];
      dizi[i] = gecici;
   }
}
```

Seçmeli Sıralama

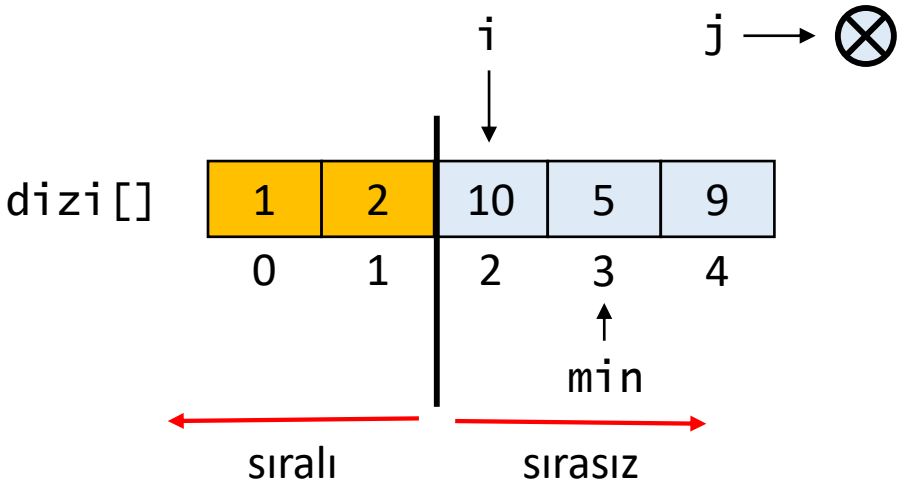


```
i = 2
min = 3
j = 4
```

```
n = 5
```

```
public void sort(int[] dizi) {
   int n = dizi.length;
   for(int i = 0; i < n - 1; i++) {
      int min = i;
      for(int j = i + 1; j < n; j++) {
         if(dizi[j] < dizi[min]) {
            min = j;
         }
      }
      int gecici = dizi[min];
      dizi[min] = dizi[i];
      dizi[i] = gecici;
   }
}
```

Seçmeli Sıralama

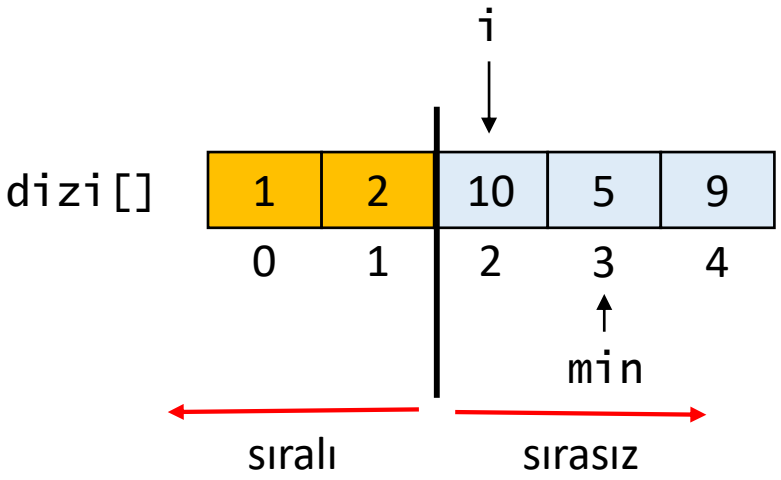


```
i = 2
min = 3
j = 5
```

```
n = 5
```

```
public void sort(int[] dizi) {
   int n = dizi.length;
   for(int i = 0; i < n - 1; i++) {
      int min = i;
      for(int j = i + 1; j < n; j++) {
         if(dizi[j] < dizi[min]) {
            min = j;
         }
      }
      int gecici = dizi[min];
      dizi[min] = dizi[i];
      dizi[i] = gecici;
   }
}
```

Seçmeli Sıralama

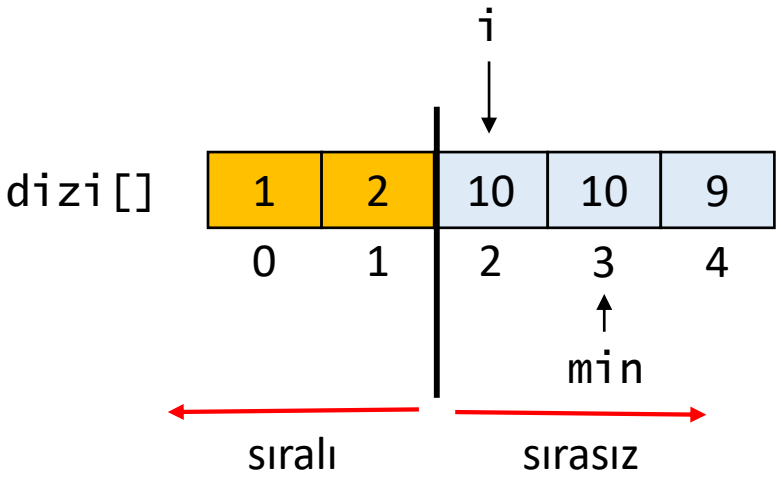


```
i = 2
min = 3
gecici = 5
```

```
n = 5
```

```
public void sort(int[] dizi) {
   int n = dizi.length;
   for(int i = 0; i < n - 1; i++) {
      int min = i;
      for(int j = i + 1; j < n; j++) {
         if(dizi[j] < dizi[min]) {
            min = j;
         }
      }
      int gecici = dizi[min];
      dizi[min] = dizi[i];
      dizi[i] = gecici;
   }
}
```

Seçmeli Sıralama

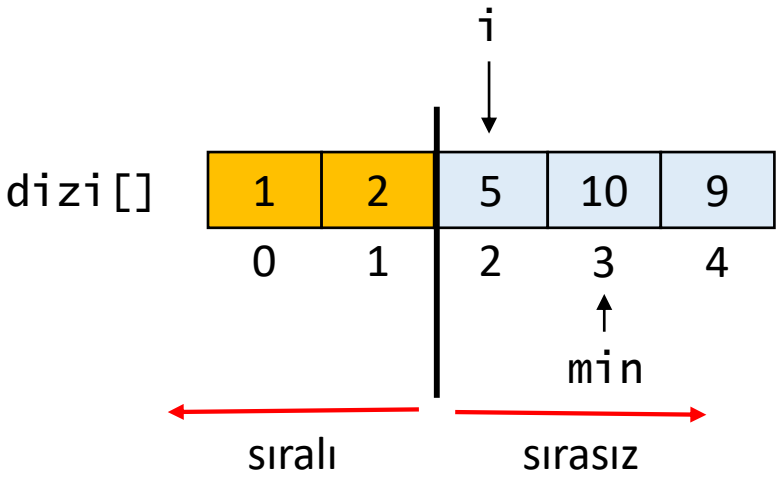


```
i = 2
min = 3
gecici = 5
```

```
n = 5
```

```
public void sort(int[] dizi) {
   int n = dizi.length;
   for(int i = 0; i < n - 1; i++) {
      int min = i;
      for(int j = i + 1; j < n; j++) {
         if(dizi[j] < dizi[min]) {
            min = j;
         }
      }
      int gecici = dizi[min];
      dizi[min] = dizi[i];
      dizi[i] = gecici;
   }
}
```

Seçmeli Sıralama

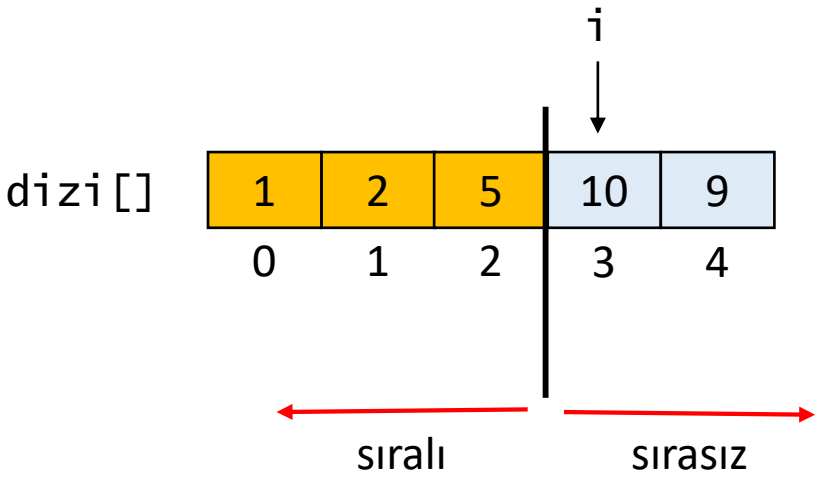


```
i = 2
min = 3
gecici = 5
```

```
n = 5
```

```
public void sort(int[] dizi) {
   int n = dizi.length;
   for(int i = 0; i < n - 1; i++) {
      int min = i;
      for(int j = i + 1; j < n; j++) {
         if(dizi[j] < dizi[min]) {
            min = j;
         }
      }
      int gecici = dizi[min];
      dizi[min] = dizi[i];
      dizi[i] = gecici;
   }
}
```

Seçmeli Sıralama

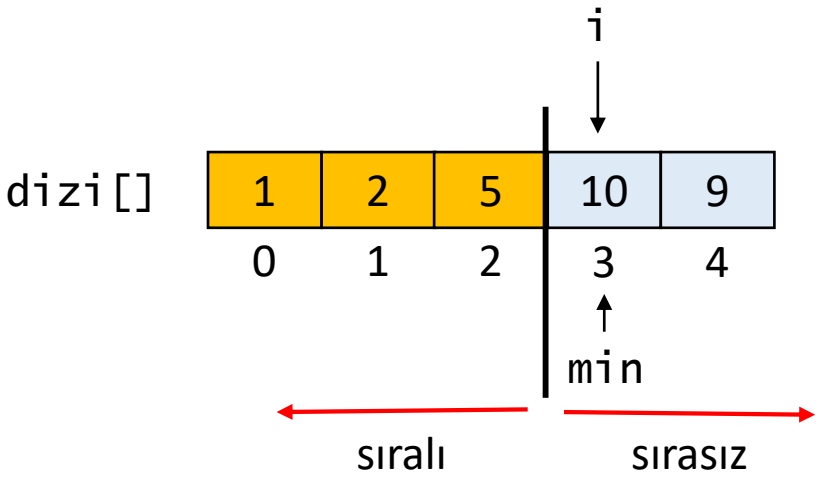


i = 3

n = 5

```
public void sort(int[] dizi) {
   int n = dizi.length;
   for(int i = 0; i < n - 1; i++) {
      int min = i;
      for(int j = i + 1; j < n; j++) {
         if(dizi[j] < dizi[min]) {
            min = j;
         }
      }
      int gecici = dizi[min];
      dizi[min] = dizi[i];
      dizi[i] = gecici;
   }
}
```

Seçmeli Sıralama

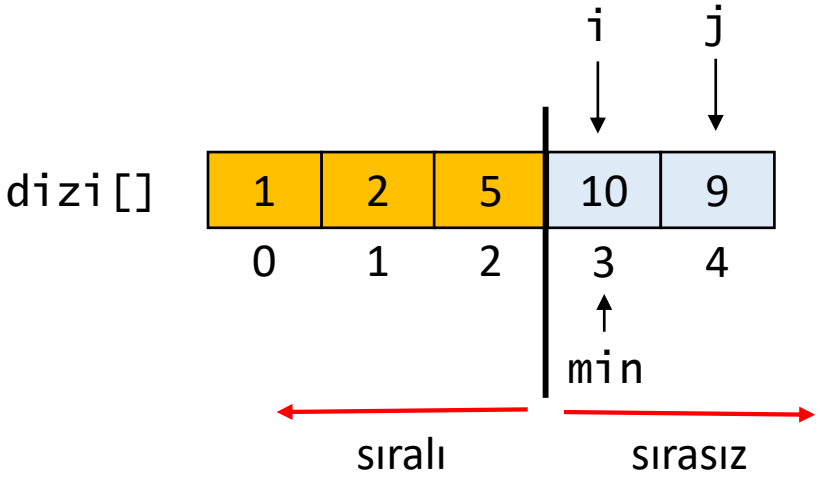


```
i = 3
min = 3
```

```
n = 5
```

```
public void sort(int[] dizi) {
   int n = dizi.length;
   for(int i = 0; i < n - 1; i++) {
      int min = i;
      for(int j = i + 1; j < n; j++) {
         if(dizi[j] < dizi[min]) {
            min = j;
         }
      }
      int gecici = dizi[min];
      dizi[min] = dizi[i];
      dizi[i] = gecici;
   }
}
```

Seçmeli Sıralama

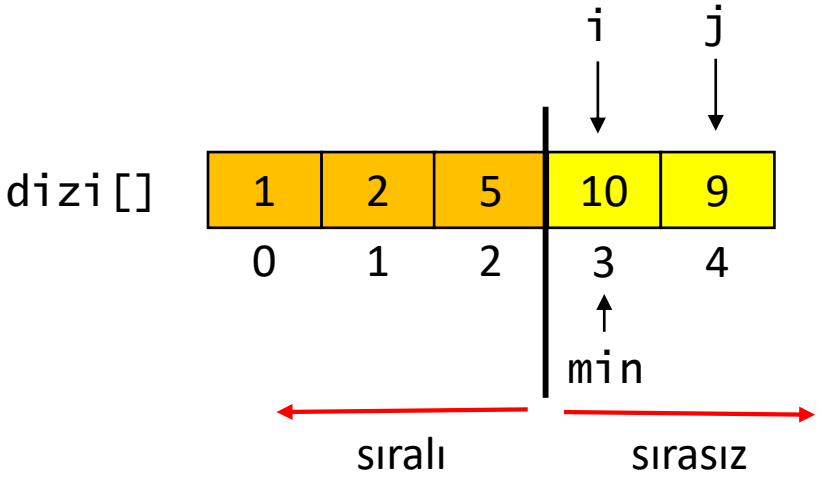


```
i = 3
min = 3
j = 4



n = 5
```

→

```
public void sort(int[] dizi) {
   int n = dizi.length;
   for(int i = 0; i < n - 1; i++) {
      int min = i;
      for(int j = i + 1; j < n; j++) {
         if(dizi[j] < dizi[min]) {
            min = j;
         }
      }
      int gecici = dizi[min];
      dizi[min] = dizi[i];
      dizi[i] = gecici;
   }
}
```

Seçmeli Sıralama

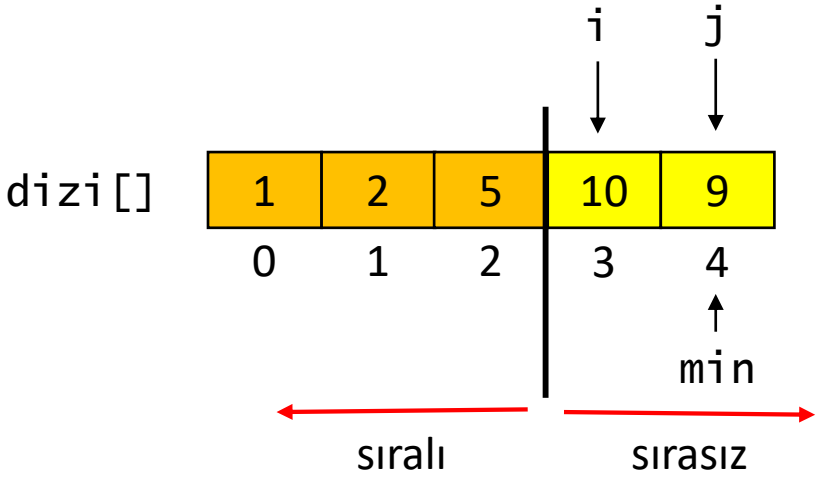


```
i = 3
min = 3
j = 4



n = 5
```

→

```
public void sort(int[] dizi) {
   int n = dizi.length;
   for(int i = 0; i < n - 1; i++) {
      int min = i;
      for(int j = i + 1; j < n; j++) {
         if(dizi[j] < dizi[min]) {
            min = j;
         }
      }
      int gecici = dizi[min];
      dizi[min] = dizi[i];
      dizi[i] = gecici;
   }
}
```

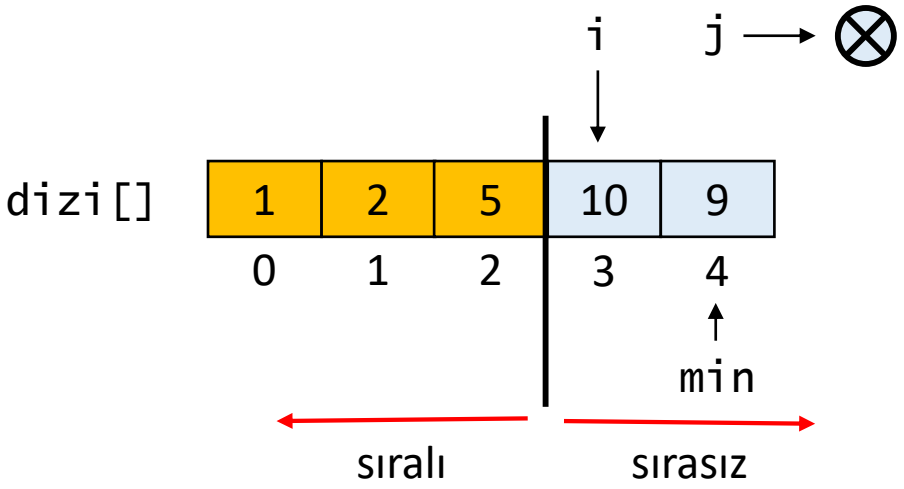
Seçmeli Sıralama



```
i = 3
min = 4
j = 4


n = 5
```

```
public void sort(int[] dizi) {
   int n = dizi.length;
   for(int i = 0; i < n - 1; i++) {
      int min = i;
      for(int j = i + 1; j < n; j++) {
         if(dizi[j] < dizi[min]) {
            min = j;
         }
      }
      int gecici = dizi[min];
      dizi[min] = dizi[i];
      dizi[i] = gecici;
   }
}
```

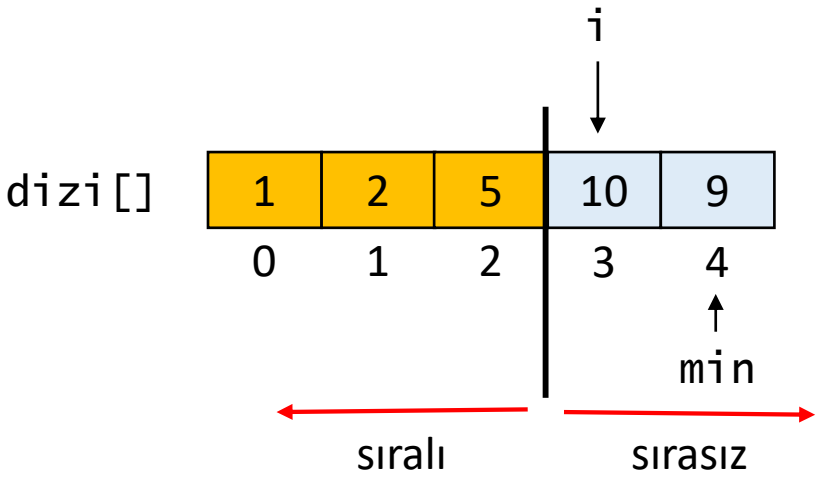
Seçmeli Sıralama



```
i = 3
min = 4
j = 5


n = 5
```

```
public void sort(int[] dizi) {
   int n = dizi.length;
   for(int i = 0; i < n - 1; i++) {
      int min = i;
      for(int j = i + 1; j < n; j++) {
         if(dizi[j] < dizi[min]) {
            min = j;
         }
      }
      int gecici = dizi[min];
      dizi[min] = dizi[i];
      dizi[i] = gecici;
   }
}
```

Seçmeli Sıralama

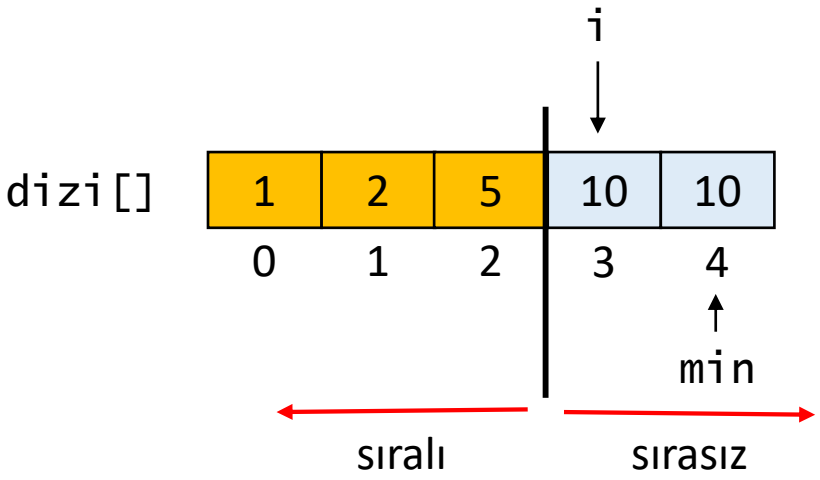


```
i = 3
min = 4
gecici = 9
```

```
n = 5
```

```
public void sort(int[] dizi) {
   int n = dizi.length;
   for(int i = 0; i < n - 1; i++) {
      int min = i;
      for(int j = i + 1; j < n; j++) {
         if(dizi[j] < dizi[min]) {
            min = j;
         }
      }
      int gecici = dizi[min];
      dizi[min] = dizi[i];
      dizi[i] = gecici;
   }
}
```

Seçmeli Sıralama

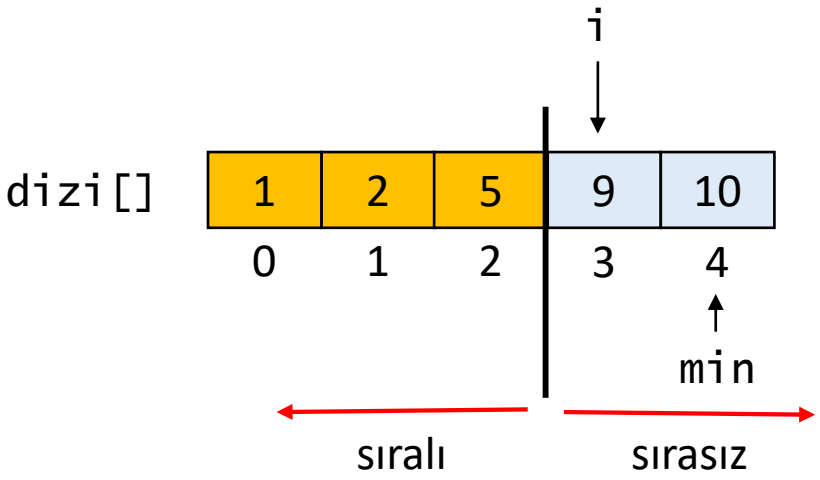


```
i = 3
min = 4
gecici = 9
```

```
n = 5
```

```
public void sort(int[] dizi) {
   int n = dizi.length;
   for(int i = 0; i < n - 1; i++) {
      int min = i;
      for(int j = i + 1; j < n; j++) {
         if(dizi[j] < dizi[min]) {
            min = j;
         }
      }
      int gecici = dizi[min];
      dizi[min] = dizi[i];
      dizi[i] = gecici;
   }
}
```

Seçmeli Sıralama

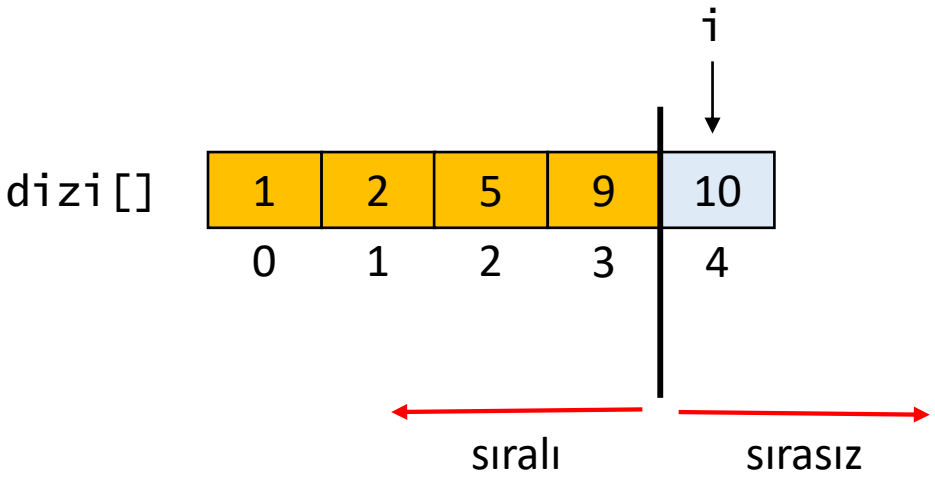


```
i = 3
min = 4
gecici = 9
```

```
n = 5
```

```
public void sort(int[] dizi) {
   int n = dizi.length;
   for(int i = 0; i < n - 1; i++) {
      int min = i;
      for(int j = i + 1; j < n; j++) {
         if(dizi[j] < dizi[min]) {
            min = j;
         }
      }
      int gecici = dizi[min];
      dizi[min] = dizi[i];
      dizi[i] = gecici;
   }
}
```

Seçmeli Sıralama

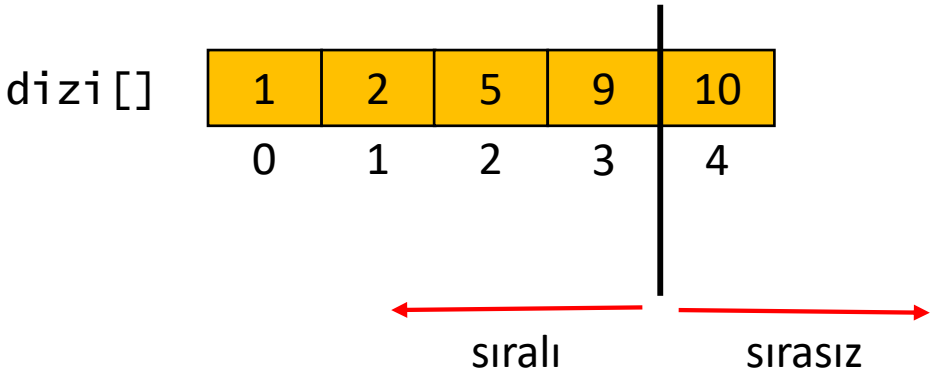


i = 4

n = 5

```
public void sort(int[] dizi) {
   int n = dizi.length;
   for(int i = 0; i < n - 1; i++) {
      int min = i;
      for(int j = i + 1; j < n; j++) {
         if(dizi[j] < dizi[min]) {
            min = j;
         }
      }
      int gecici = dizi[min];
      dizi[min] = dizi[i];
      dizi[i] = gecici;
   }
}
```

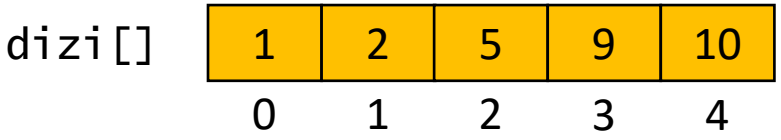
Seçmeli Sıralama



n = 5

```
public void sort(int[] dizi) {
   int n = dizi.length;
   for(int i = 0; i < n - 1; i++) {
      int min = i;
      for(int j = i + 1; j < n; j++) {
         if(dizi[j] < dizi[min]) {
            min = j;
         }
      }
      int gecici = dizi[min];
      dizi[min] = dizi[i];
      dizi[i] = gecici;
   }
}
```

Seçmeli Sıralama




```
public void sort(int[] dizi) {
   int n = dizi.length;
   for(int i = 0; i < n - 1; i++) {
      int min = i;
      for(int j = i + 1; j < n; j++) {
         if(dizi[j] < dizi[min]) {
            min = j;
         }
      }
      int gecici = dizi[min];
      dizi[min] = dizi[i];
      dizi[i] = gecici;
   }
}
```

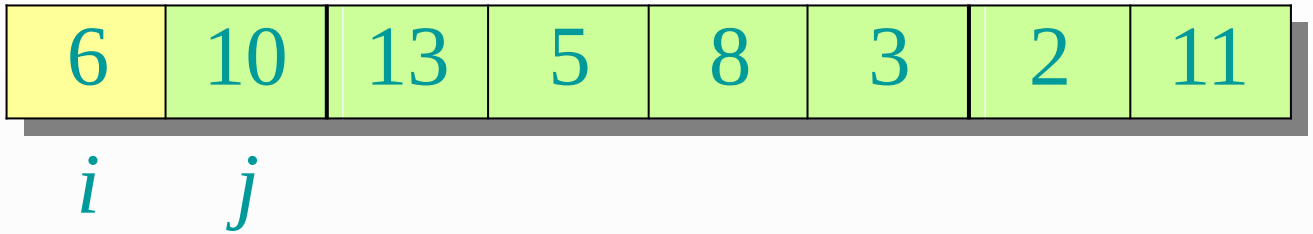
Çabuk Sıralama (QuickSort)

- Bir pivot nokta seçilir.
- Pivot noktadan dizi iki parçaya bölünür.
- İki alt parça özyinelemeli olarak sıralanır.

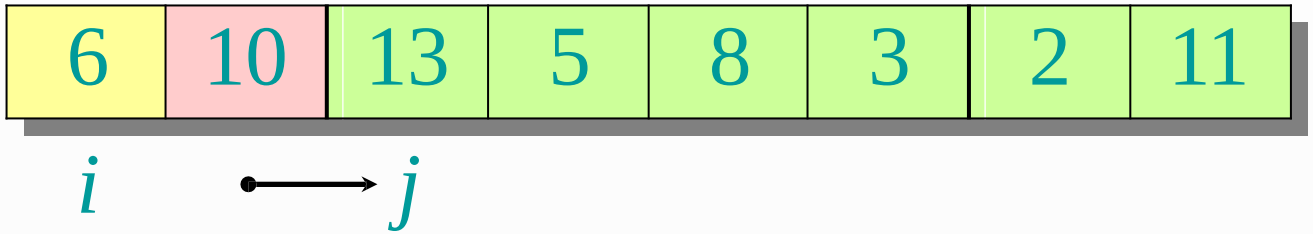
Parçalama (Partitioning) Prosedürü

```
function partition(A, p, q):
    pivot = A[q]  // pivot seç
    i = p - 1         // en küçük eleman indisi
    for j from p to q - 1:
        if A[j] <= pivot:
            i = i + 1
            swap(A[i], A[j])
    swap(A[i + 1], A[q])  // pivot ve elemanı yer değiş
    return i + 1  // parçalamadan sonra pivotun indisi
```

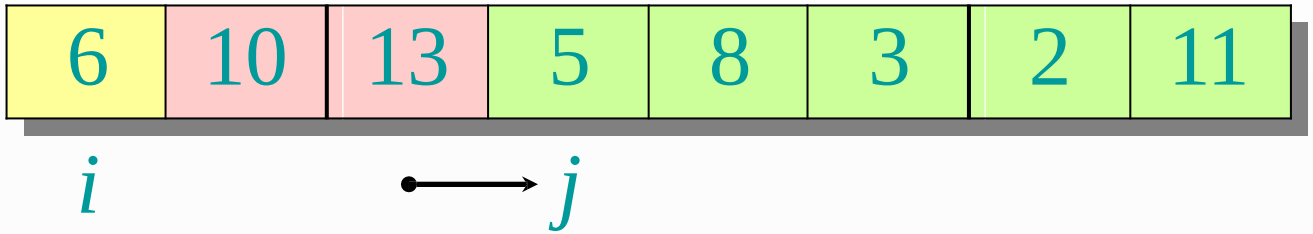
Parçalara Ayırma (Partitioning)



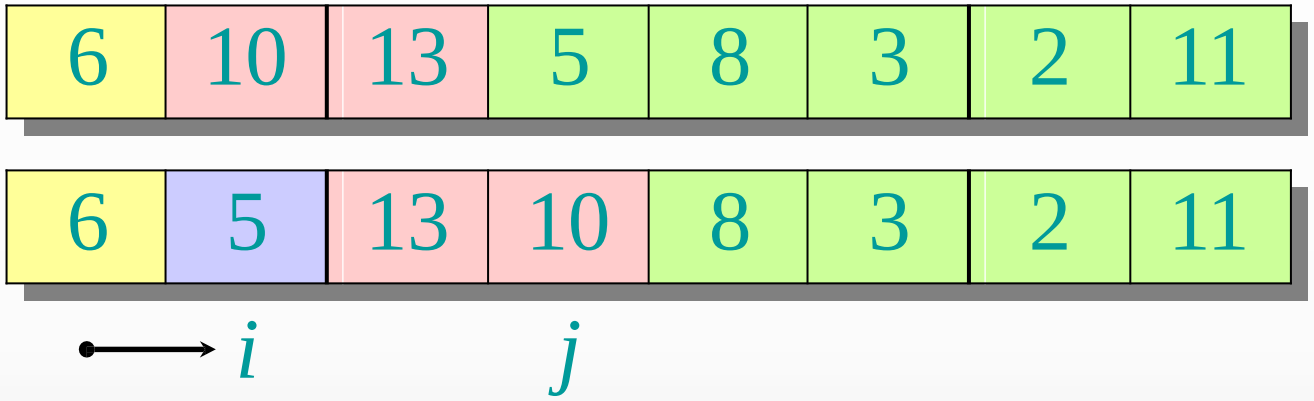
Parçalara Ayırma (Partitioning)



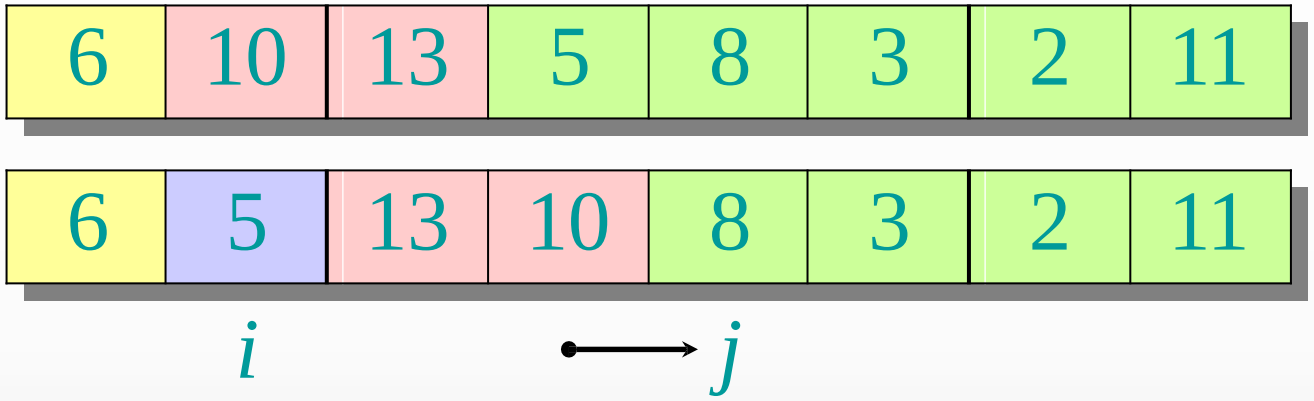
Parçalara Ayırma (Partitioning)



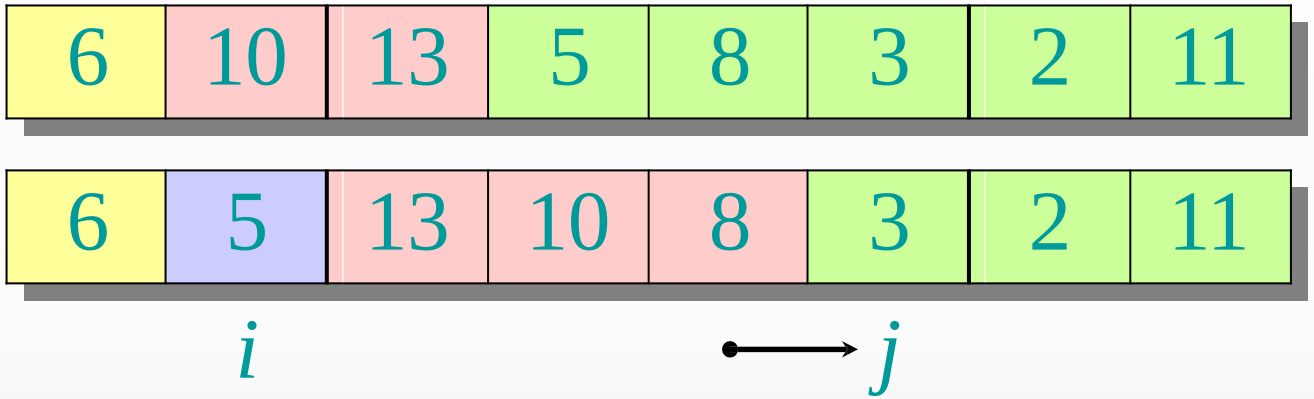
Parçalara Ayırma (Partitioning)



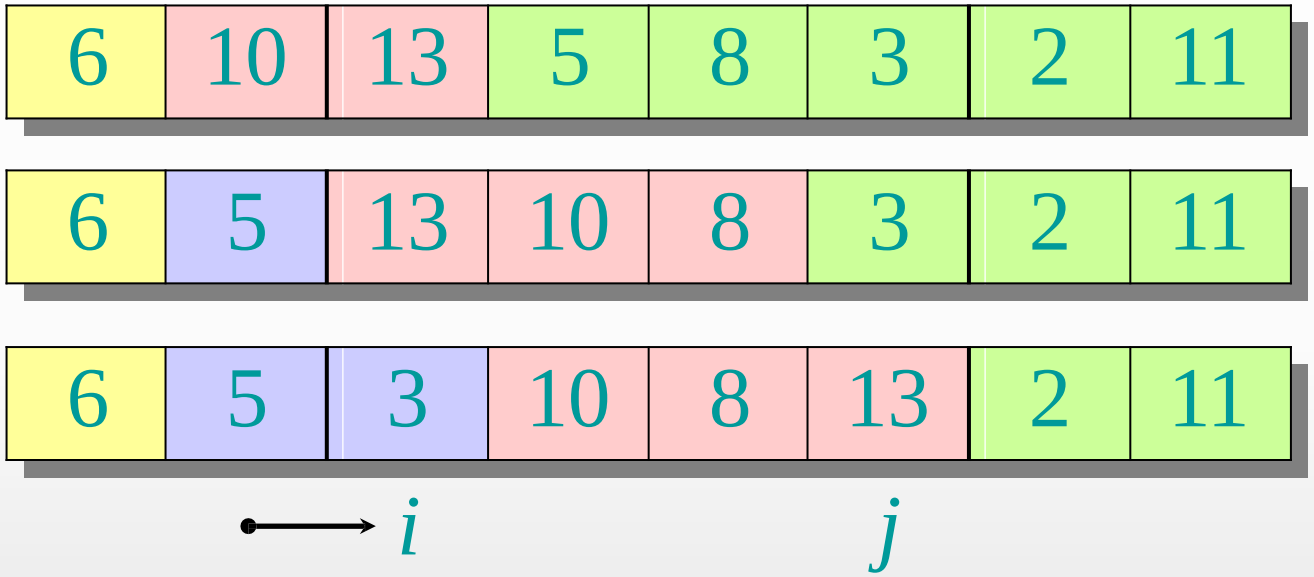
Parçalara Ayırma (Partitioning)



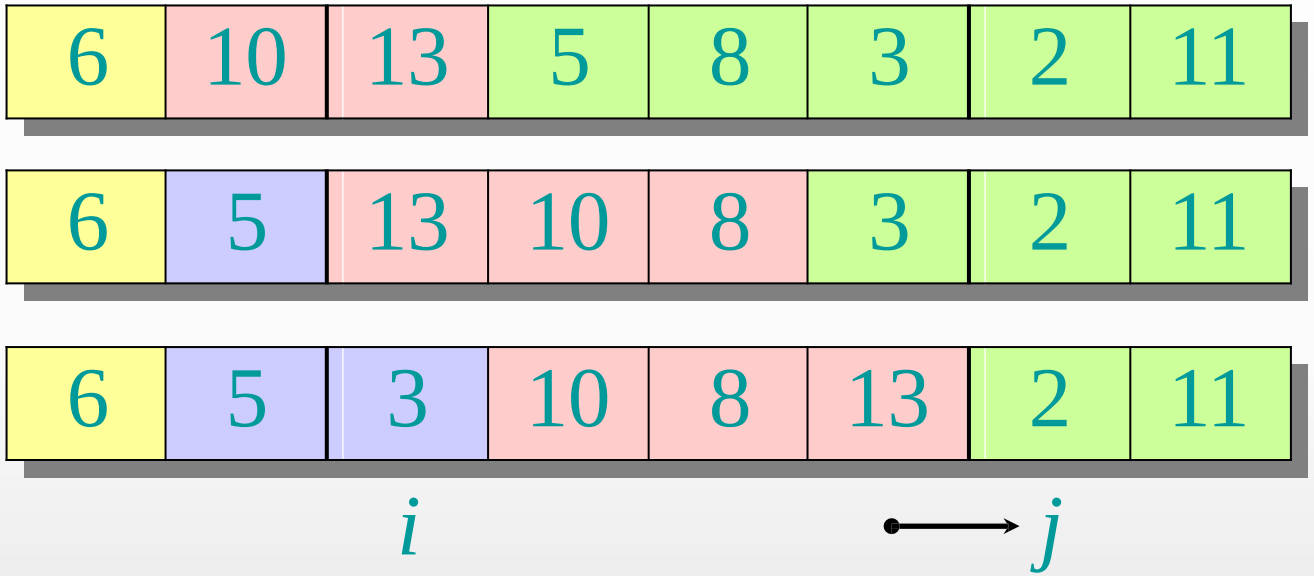
Parçalara Ayırma (Partitioning)



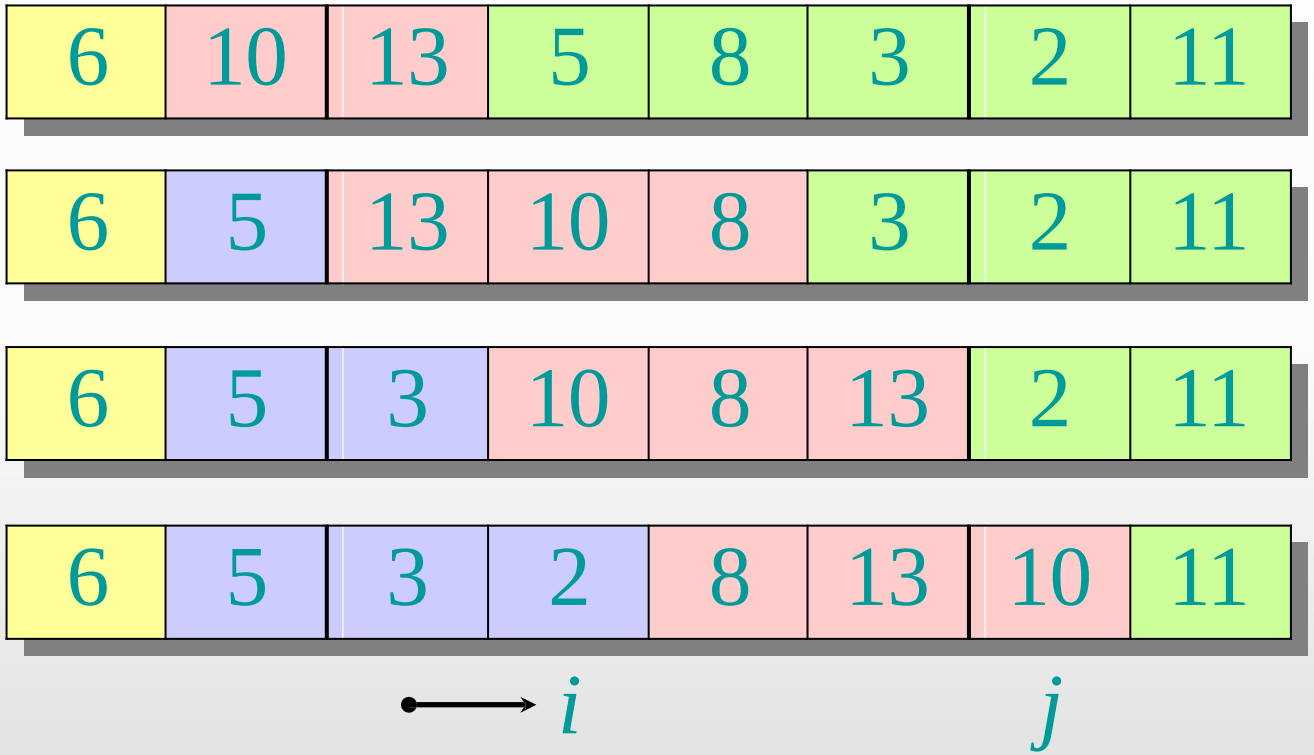
Parçalara Ayırma (Partitioning)



Parçalara Ayırma (Partitioning)

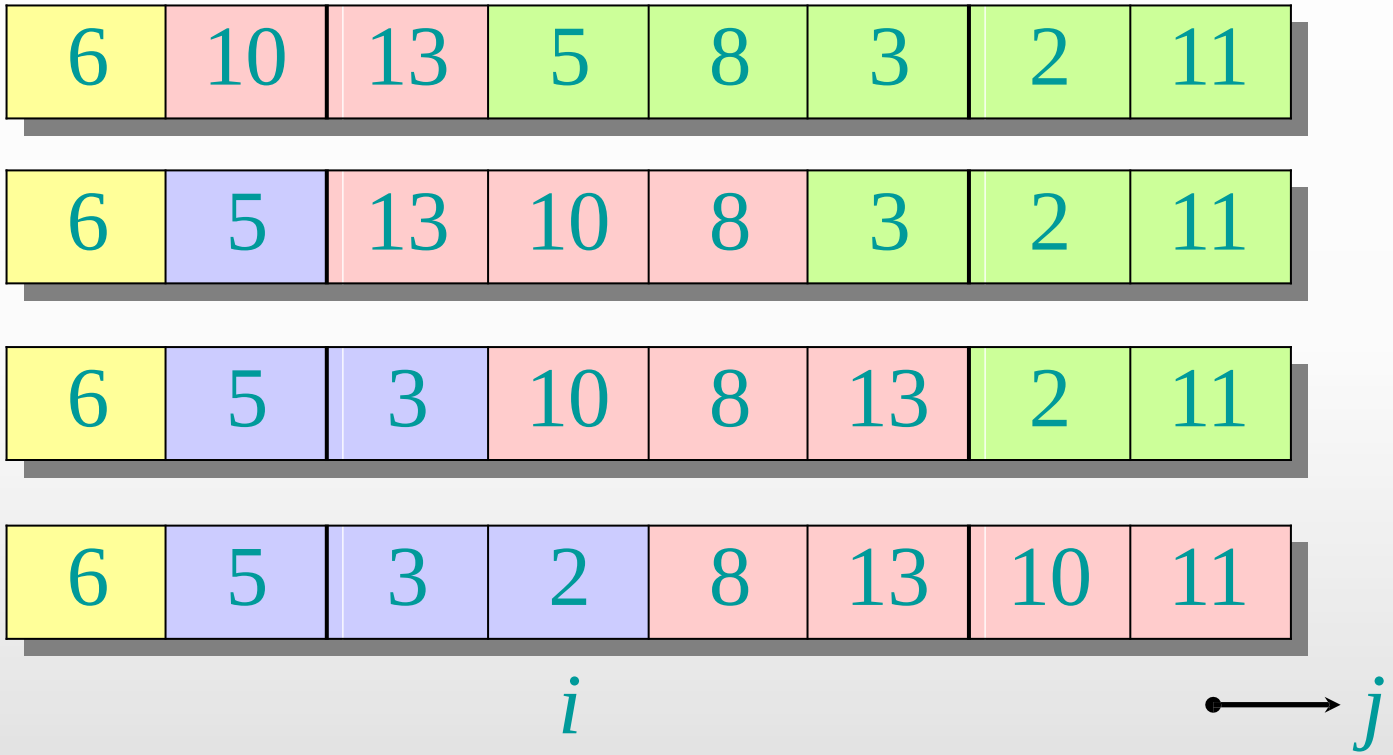


Parçalara Ayırma (Partitioning)

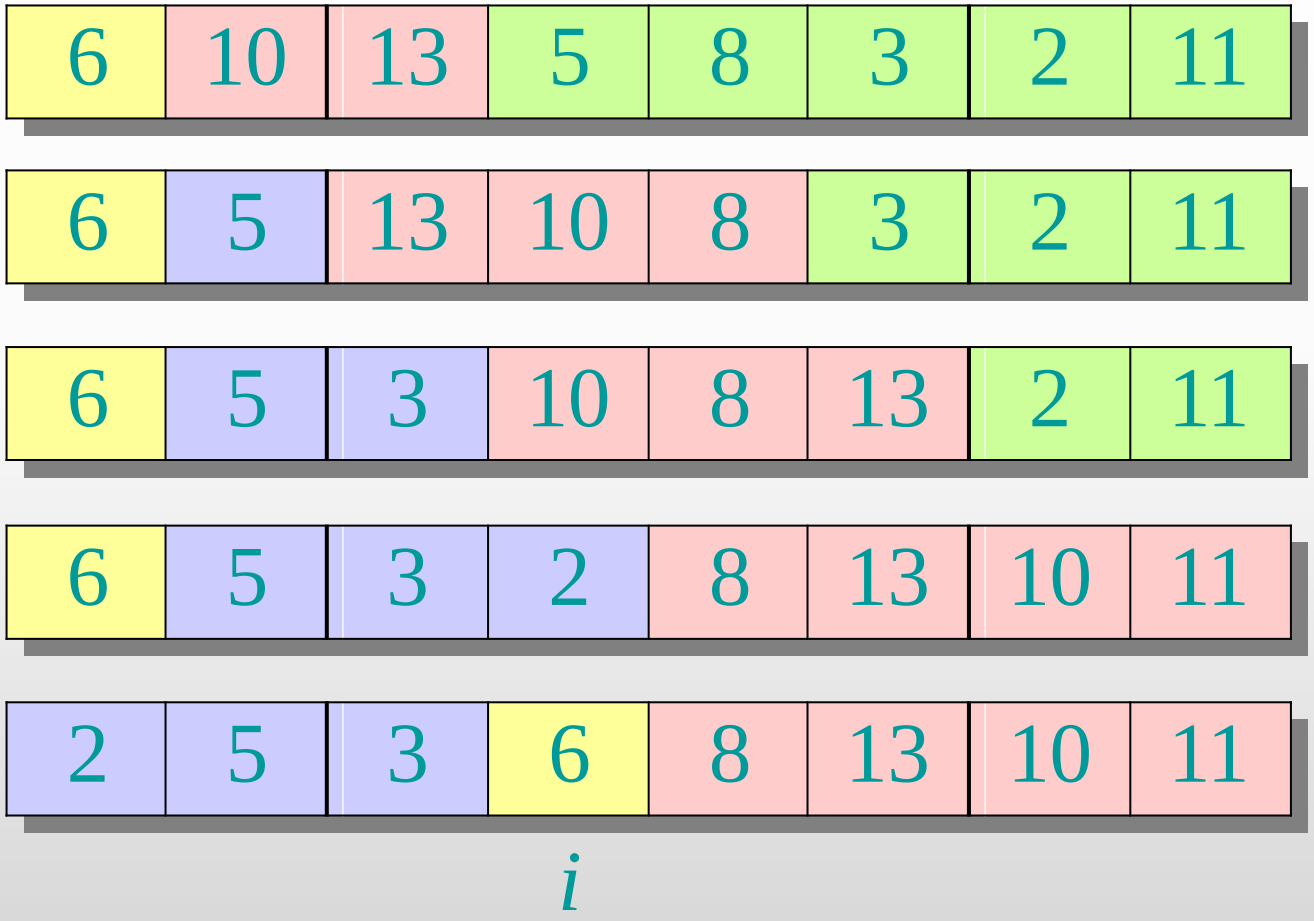


Parçalara Ayırma (Partitioning)

Parçalara Ayırma (Partitioning)



Parçalara Ayırma (Partitioning)



QuickSort

```
QUICKSORT(A, p, r)
  if p < r
    then q = PARTITION(A, p, r)
      QUICKSORT(A, p, q-1)
      QUICKSORT(A, q+1, r)
```

İlk çağrı: QUICKSORT(A, 1, n)

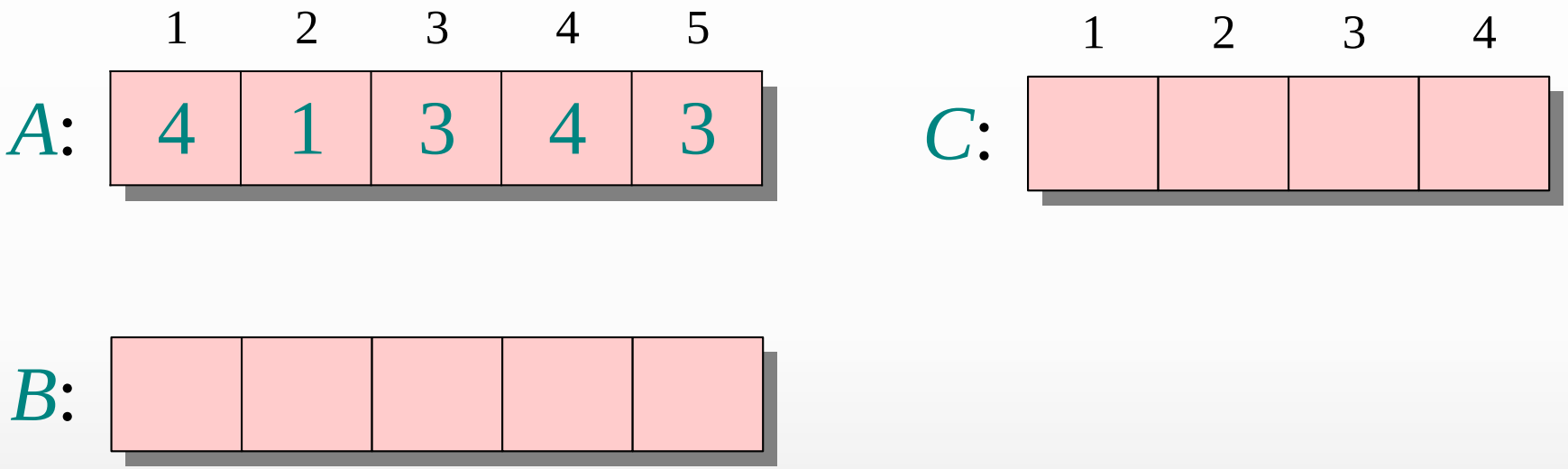
Saymalı Sıralama (Counting sort)

- Elemanlar birbiriyle karşılaştırılmaz.
- **Girdi**:
  - A[1 . . n],
  - A[j] $\in$ {1, 2, …, k} .
- **Çıktı**:
  - B[1 . . n], sıralı.
- **Geçici**:
  - C[1 . . k].

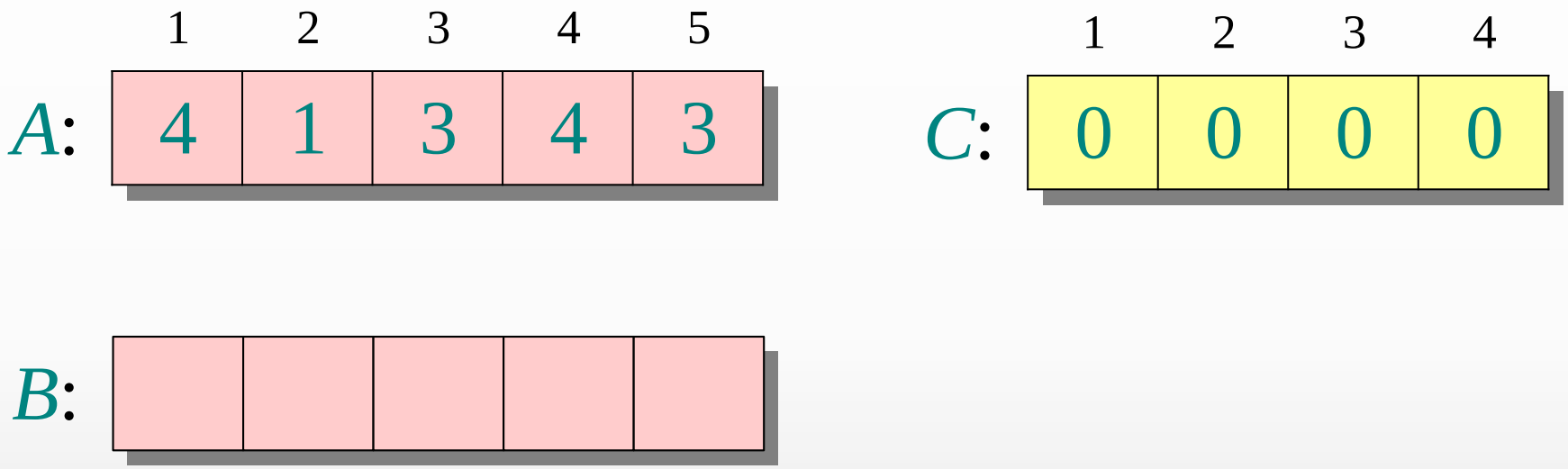
Saymalı Sıralama (Counting sort)

```
for i = 1 to k
  do C[i] = 0
for j = 1 to n
  do C[A[j]] = C[A[j]] + 1          // C[i] = |{key = i}|
for i = 2 to k
  do C[i] = C[i] + C[i-1]           // C[i] = |{key <= i}|
for j = n downto 1
  do B[C[A[j]]] = A[j]
    C[A[j]] = C[A[j]] - 1
```

Saymalı Sıralama (Counting sort)

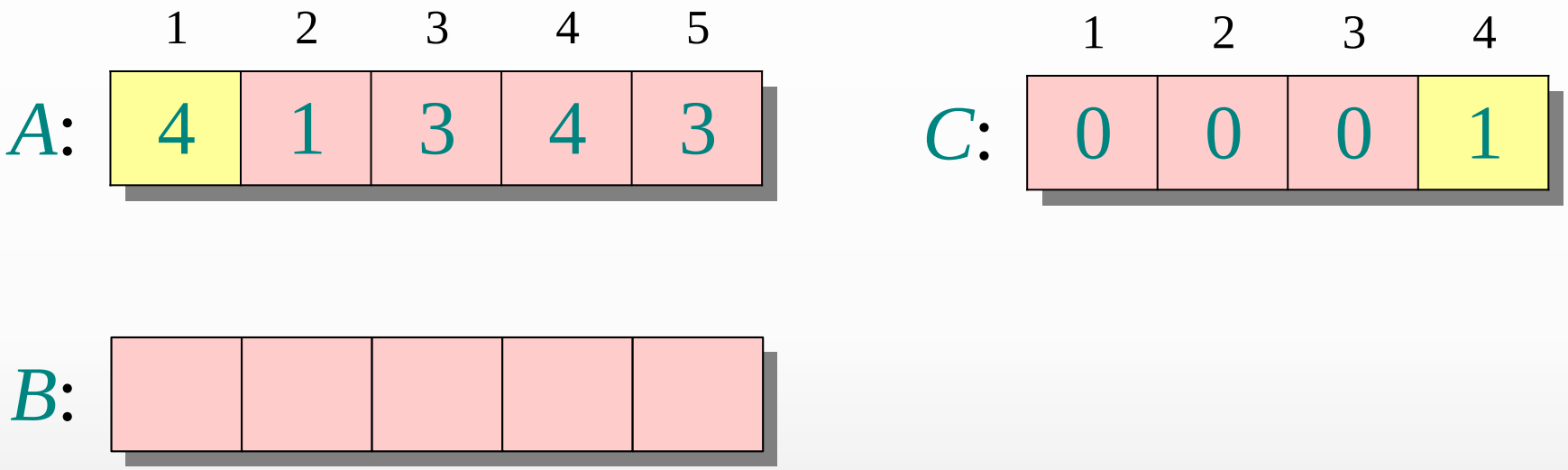


Döngü 1



for $i \leftarrow 1$ **to** k
do $C[i] \leftarrow 0$

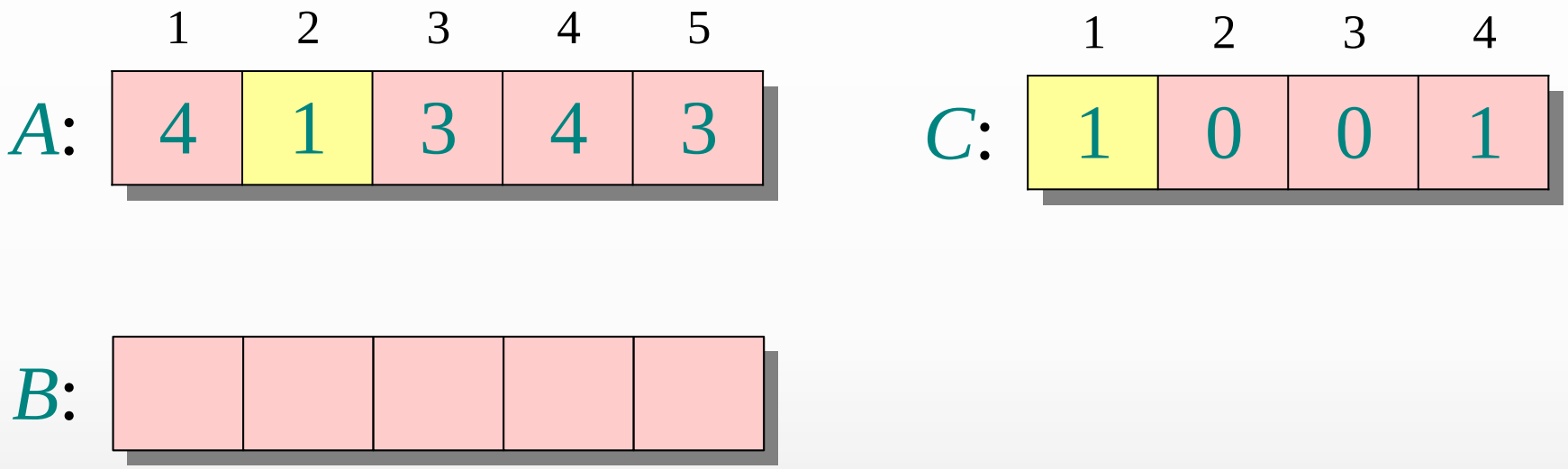
Döngü 2



for $j \leftarrow 1$ **to** n

do $C[A[j]] \leftarrow C[A[j]] + 1$ ▷ $C[i] = |\{\text{key} = i\}|$

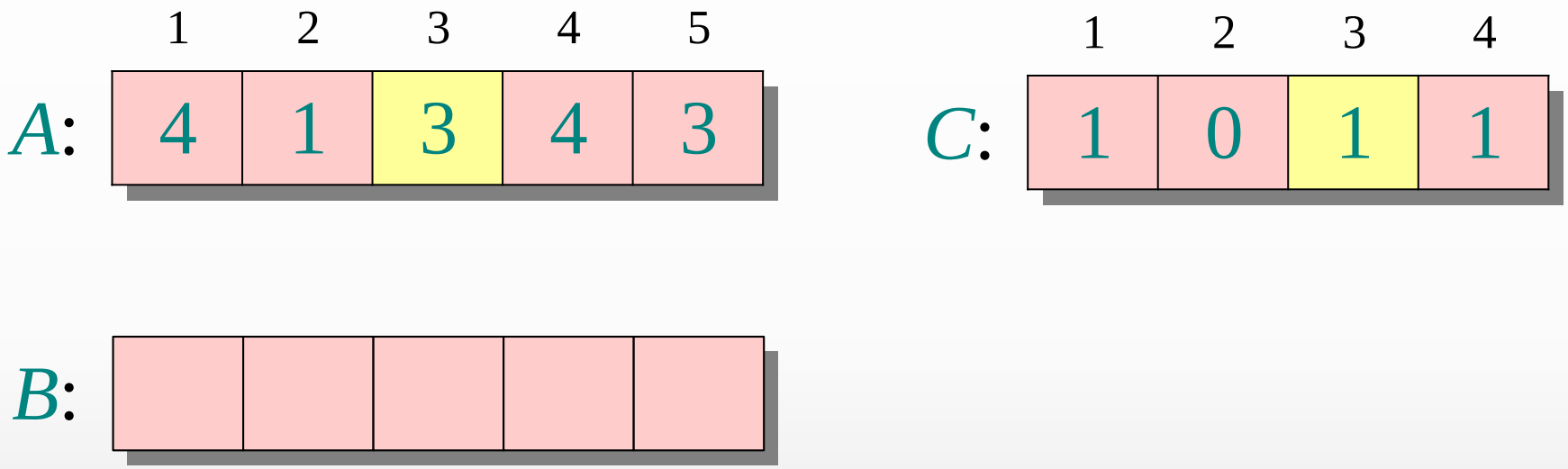
Döngü 2



for $j \leftarrow 1$ **to** n

do $C[A[j]] \leftarrow C[A[j]] + 1$ ▷ $C[i] = |\{\text{key} = i\}|$

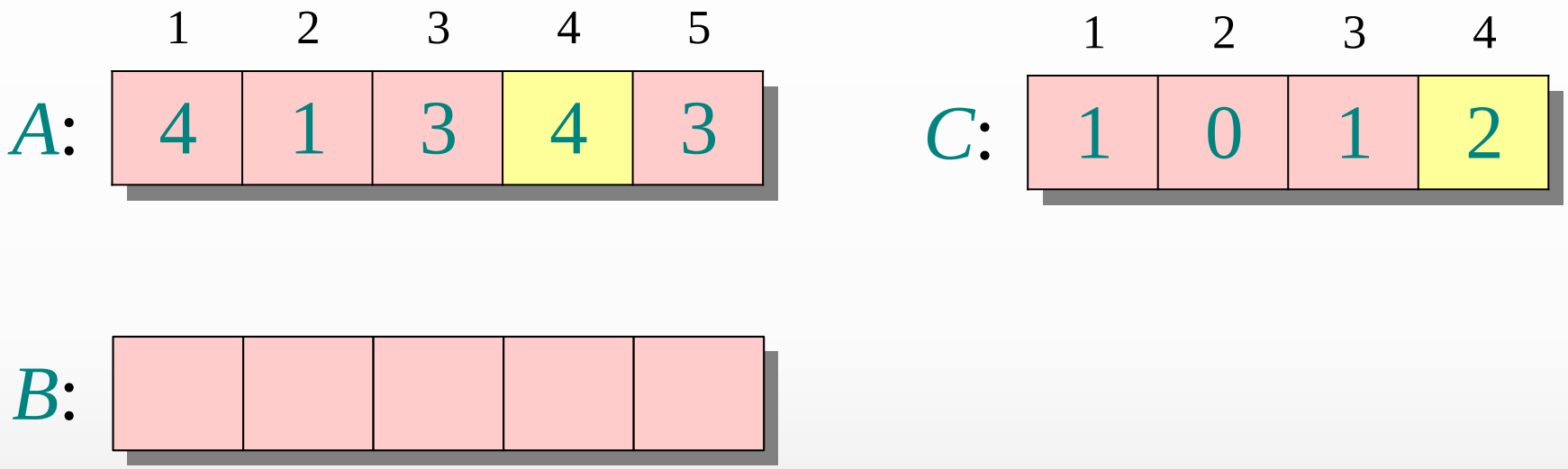
Döngü 2



for $j \leftarrow 1$ **to** n

do $C[A[j]] \leftarrow C[A[j]] + 1$ ▷ $C[i] = |\{\text{key} = i\}|$

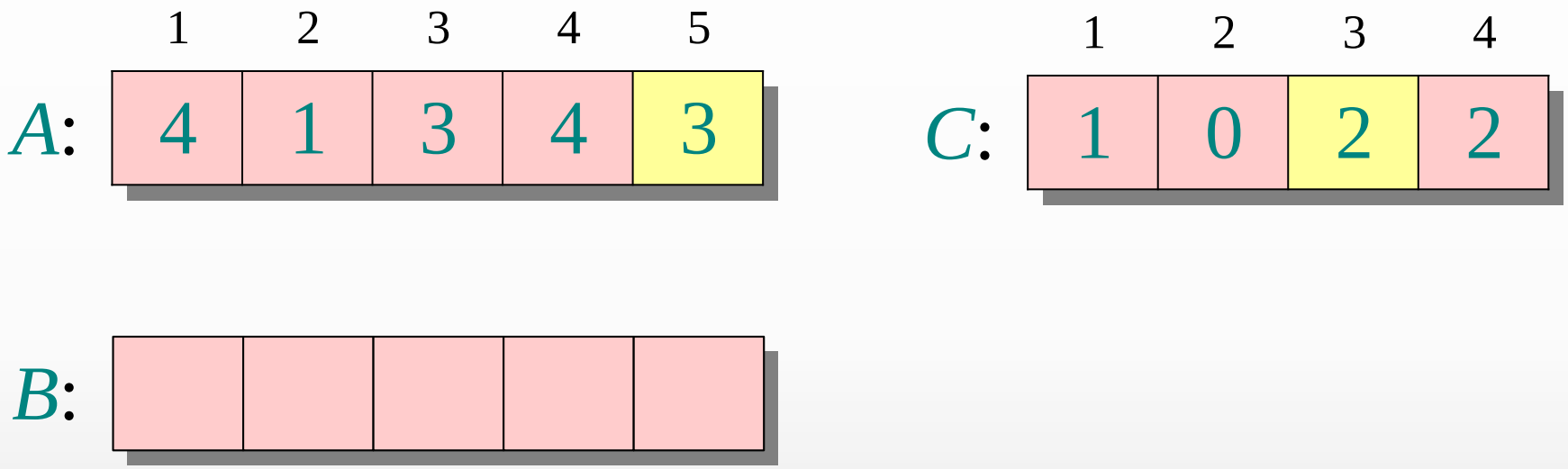
Döngü 2



for $j \leftarrow 1$ **to** n

do $C[A[j]] \leftarrow C[A[j]] + 1$ ▷ $C[i] = |\{\text{key} = i\}|$

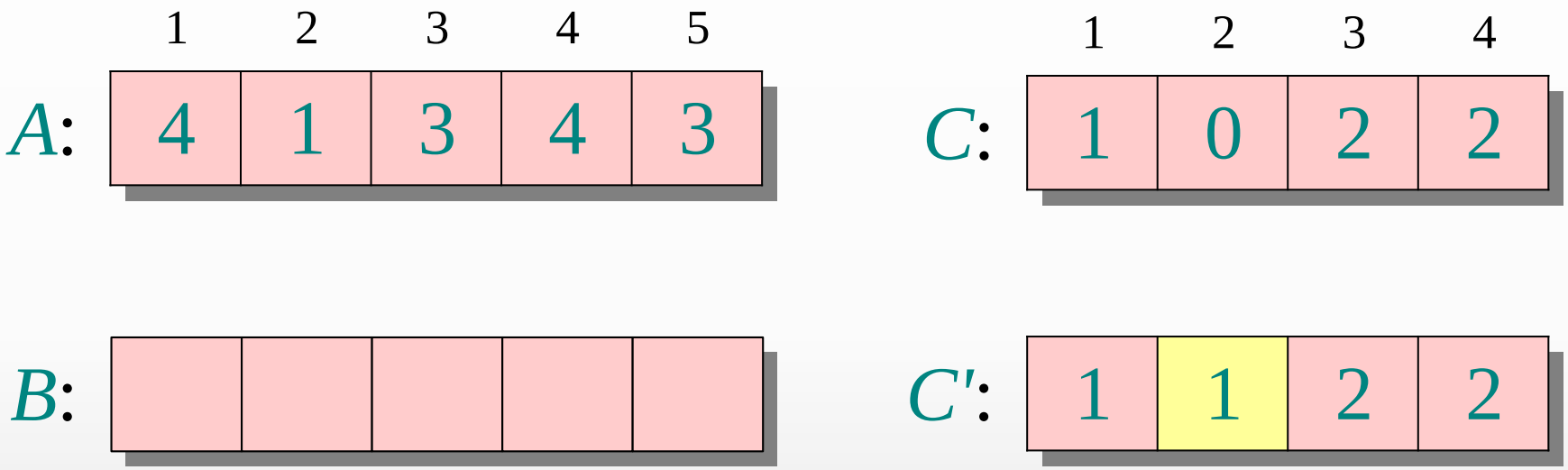
Döngü 2



for $j \leftarrow 1$ **to** n

do $C[A[j]] \leftarrow C[A[j]] + 1$ ▷ $C[i] = |\{\text{key} = i\}|$

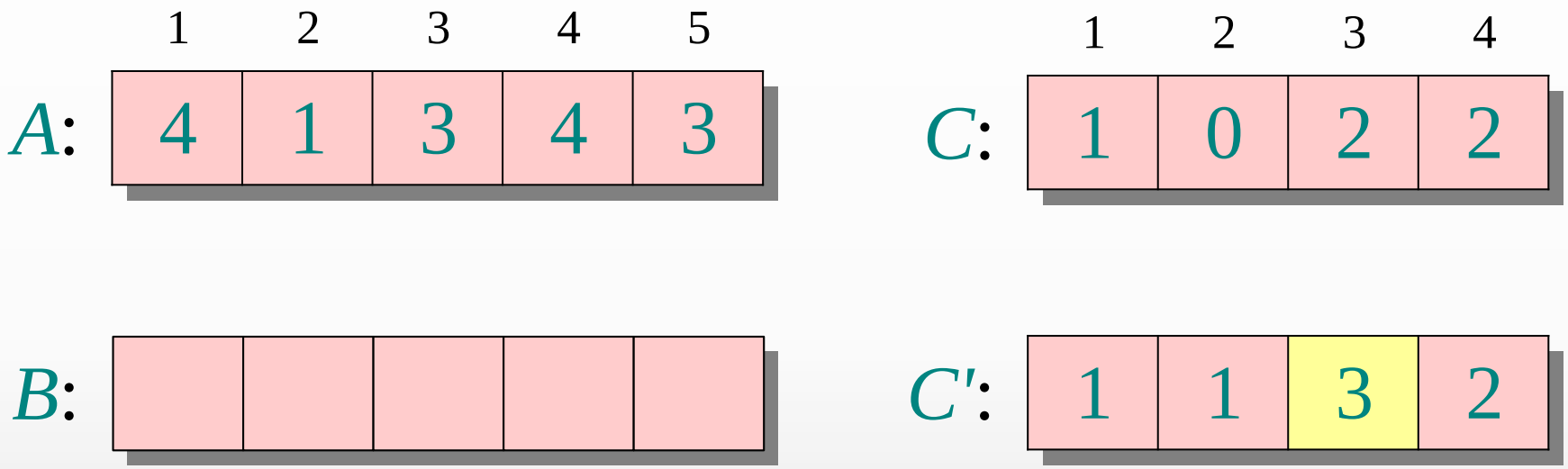
Döngü 3



for $i \leftarrow 2$ **to** k

do $C[i] \leftarrow C[i] + C[i-1]$ ▷ $C[i] = |\{\text{key} \leq i\}|$

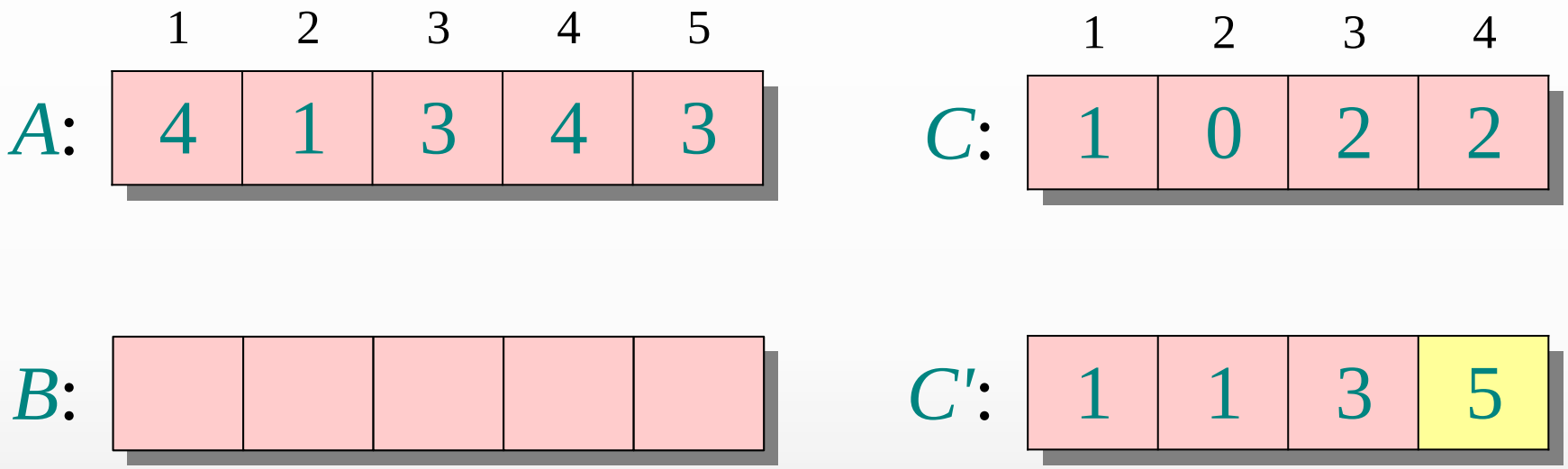
Döngü 3



for $i \leftarrow 2$ **to** k

do $C[i] \leftarrow C[i] + C[i-1]$ ▷ $C[i] = |\{\text{key} \leq i\}|$

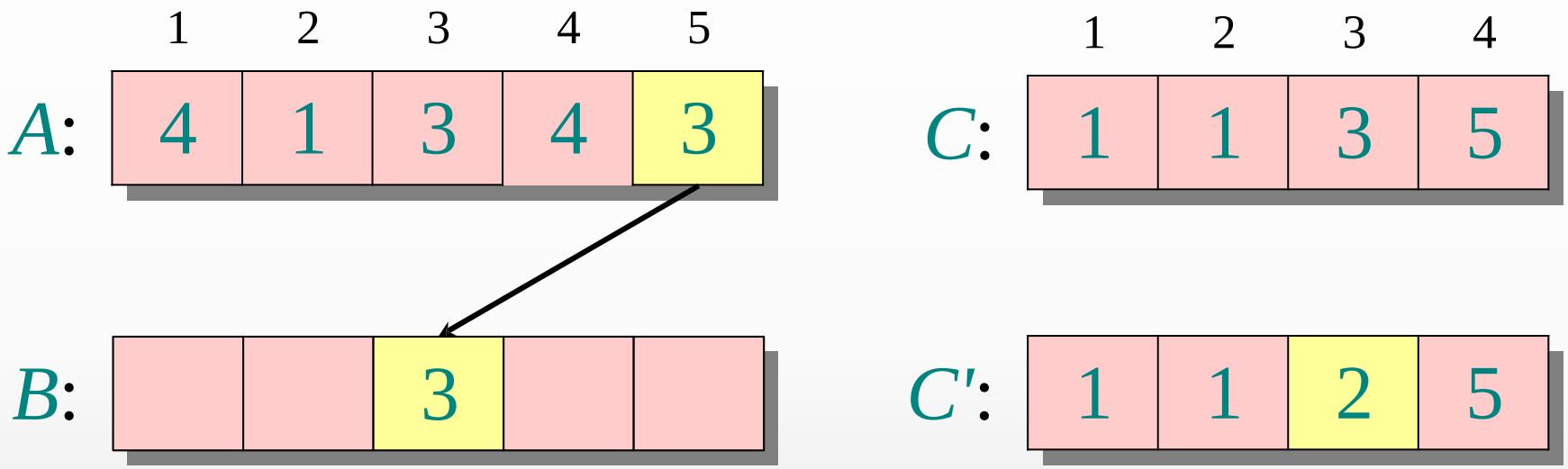
Döngü 3



for $i \leftarrow 2$ **to** k

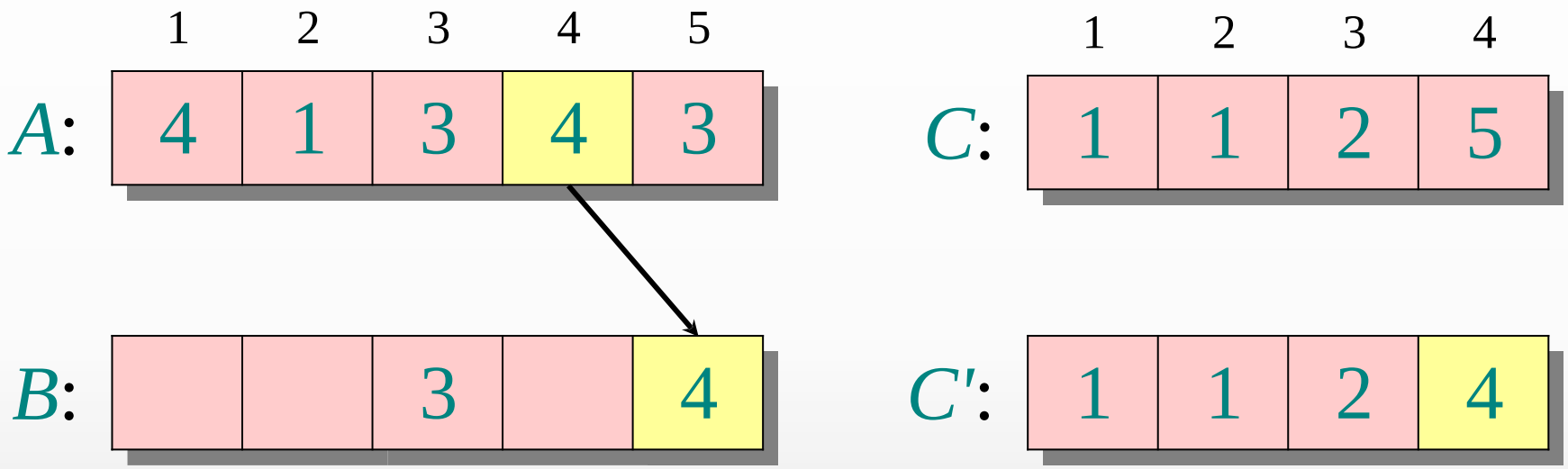
do $C[i] \leftarrow C[i] + C[i-1]$ ▷ $C[i] = |\{\text{key} \leq i\}|$

Döngü 4



for $j \leftarrow n$ **downto** 1
do $B[C[A[j]]] \leftarrow A[j]$
$C[A[j]] \leftarrow C[A[j]] - 1$

Döngü 4

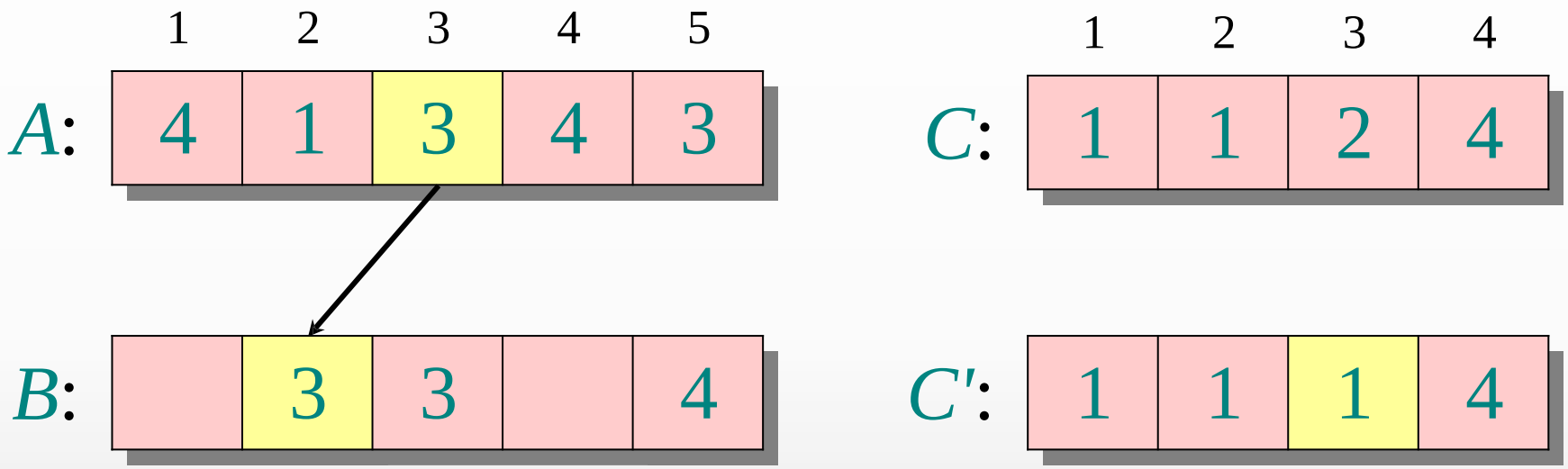


for $j \leftarrow n$ **downto** 1

do $B[C[A[j]]] \leftarrow A[j]$

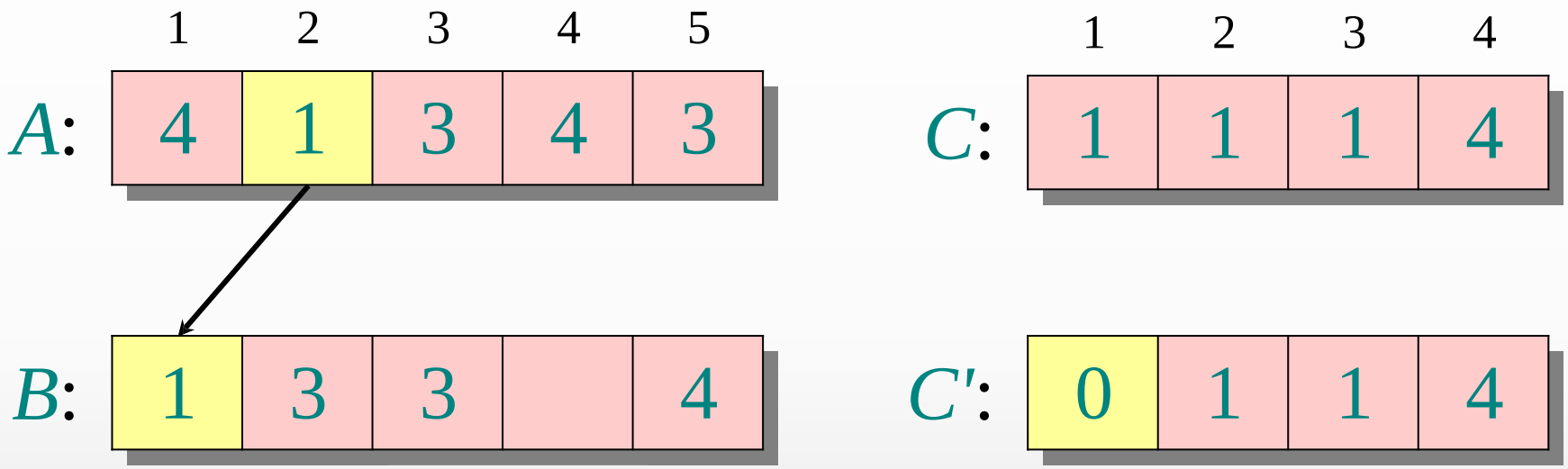
$C[A[j]] \leftarrow C[A[j]] - 1$

Döngü 4



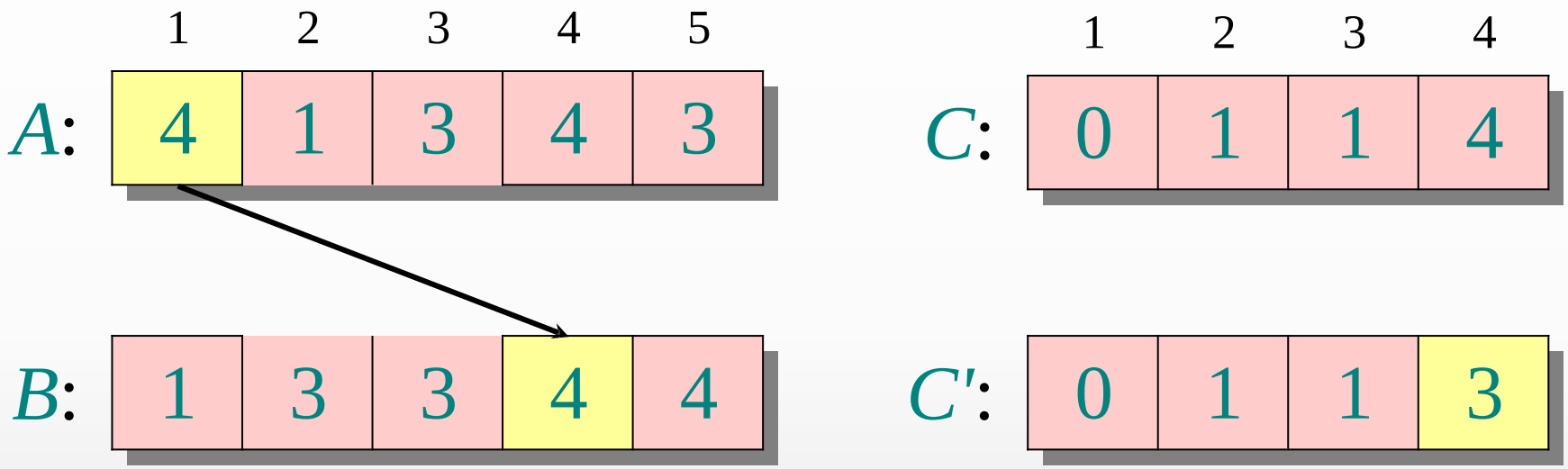
for $j \leftarrow n$ **downto** 1
do $B[C[A[j]]] \leftarrow A[j]$
$C[A[j]] \leftarrow C[A[j]] - 1$

Döngü 4



for $j \leftarrow n$ **downto** 1
do $B[C[A[j]]] \leftarrow A[j]$
$C[A[j]] \leftarrow C[A[j]] - 1$

Döngü 4



for $j \leftarrow n$ **downto** 1
do $B[C[A[j]]] \leftarrow A[j]$
$C[A[j]] \leftarrow C[A[j]] - 1$

Algoritma Karmaşıklığı

$\Theta(k)$ { **for** $i \leftarrow 1$ **to** k
  do $C[i] \leftarrow 0$

$\Theta(n)$ { **for** $j \leftarrow 1$ **to** n
  do $C[A[j]] \leftarrow C[A[j]] + 1$

$\Theta(k)$ { **for** $i \leftarrow 2$ **to** k
  do $C[i] \leftarrow C[i] + C[i–1]$

$\Theta(n)$ { **for** $j \leftarrow n$ **downto** 1
  do $B[C[A[j]]] \leftarrow A[j]$
  $C[A[j]] \leftarrow C[A[j]] – 1$

$\Theta(n + k)$

Birleştirmeli Sıralama

- Böl ve Fethet (*divide and conquer*) yaklaşımını kullanır.
- Etkili bir sıralama algoritmasıdır.
- 1945 yılında John von Neumann tarafından geliştirilmiştir.
- En kötü durumda bile O(n log n) karmaşıklığına sahiptir.
- Birleştirme işlemi için ek bellek kullanımı gerektirir.

Nasıl Çalışır?

- Bölme (Divide):
  - Liste iki alt listeye bölünür.
  - Bu alt listeler, tek bir eleman kalana kadar tekrar tekrar bölünür.
- Birleştirme (Conquer):
  - Alt listeler sıralanır ve birleştirilir.
  - Sıralı alt listeler, sıralı tek bir liste oluşturana kadar birleştirilir.

Algoritma Adımları

- Girdi listesini ikiye böl.
- Bölünen her listeyi tekrar merge sort ile sırala.
- Sıralanmış listeleri birleştir.

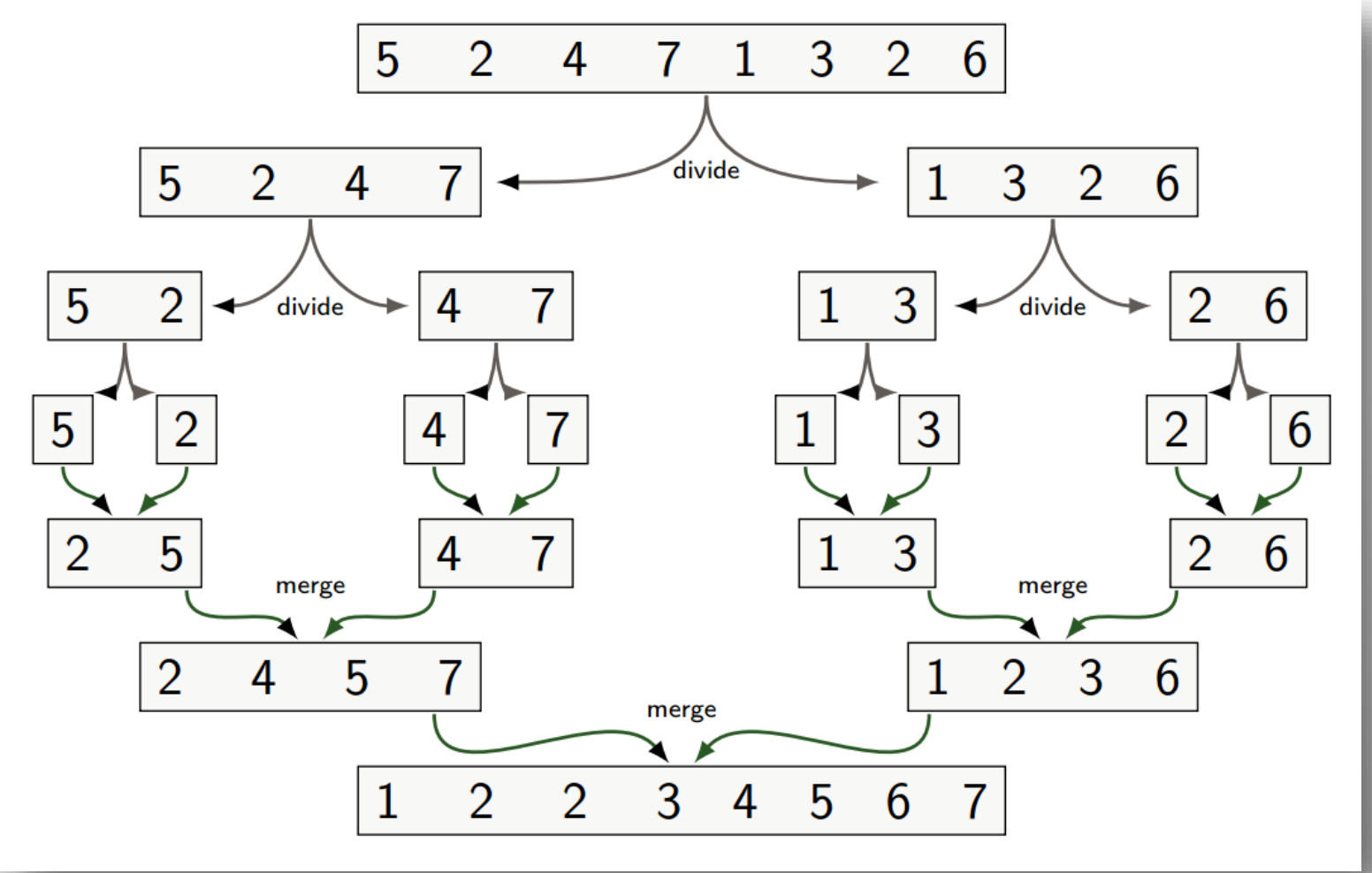
Örnek: Merge Sort

- Girdi: [38, 27, 43, 3, 9, 82, 10]
- Bölme: [38, 27, 43, 3][9, 82, 10]
- Tekrar Bölme: [38, 27][43, 3][9, 82][10]
- Tekrar Bölme: [38][27][43][3][9][82][10]
- Birleştirme: [27, 38][3, 43][9, 10, 82]
- Birleştirme: [3, 9, 10, 27, 38, 43, 82]

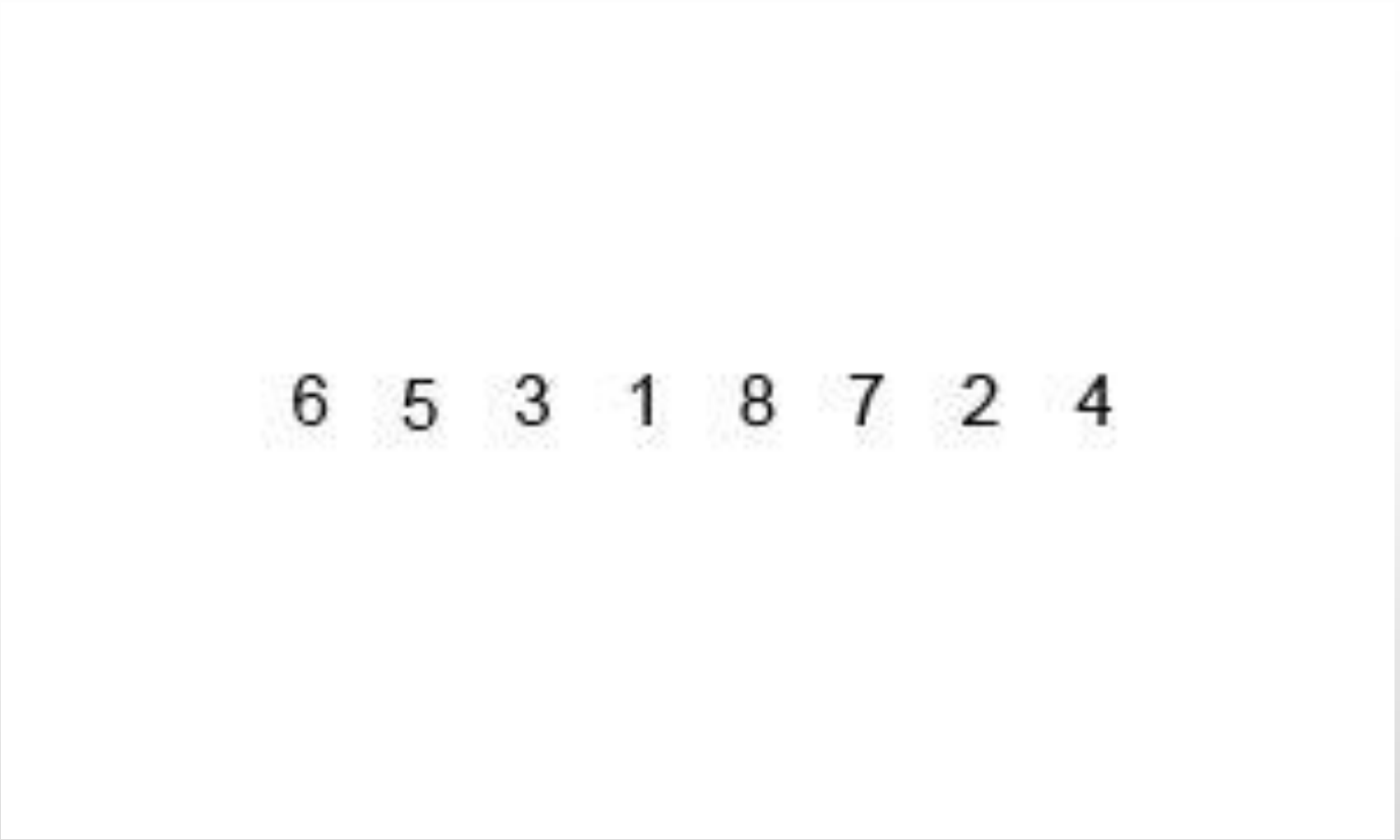
Zaman Karmaşıklığı

- En İyi Durum: *O(nlogn)*
- Ortalama Durum: *O(nlogn)*
- En Kötü Durum: *O(nlogn)*

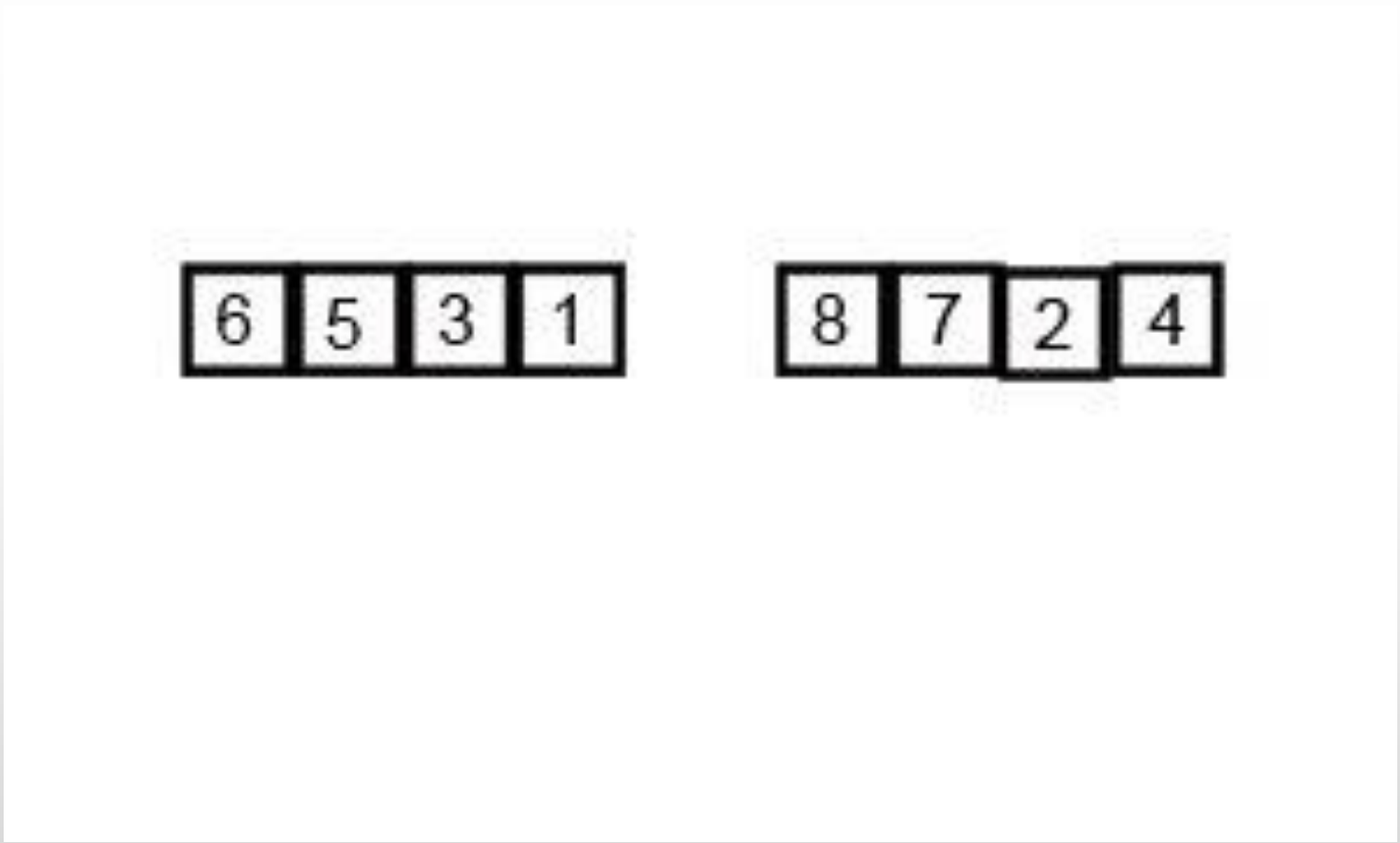
Merge Sort



Merge Sort

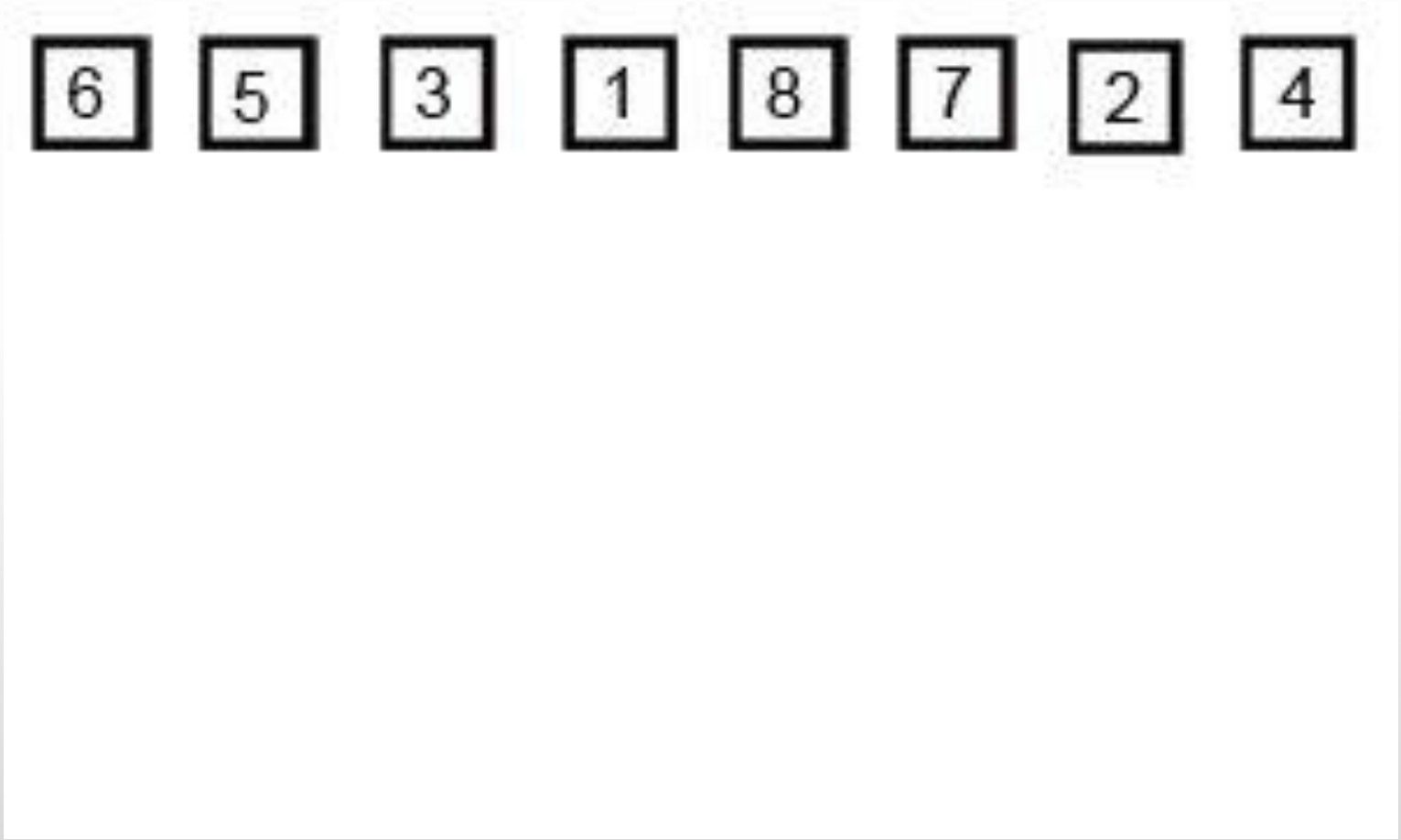


Merge Sort

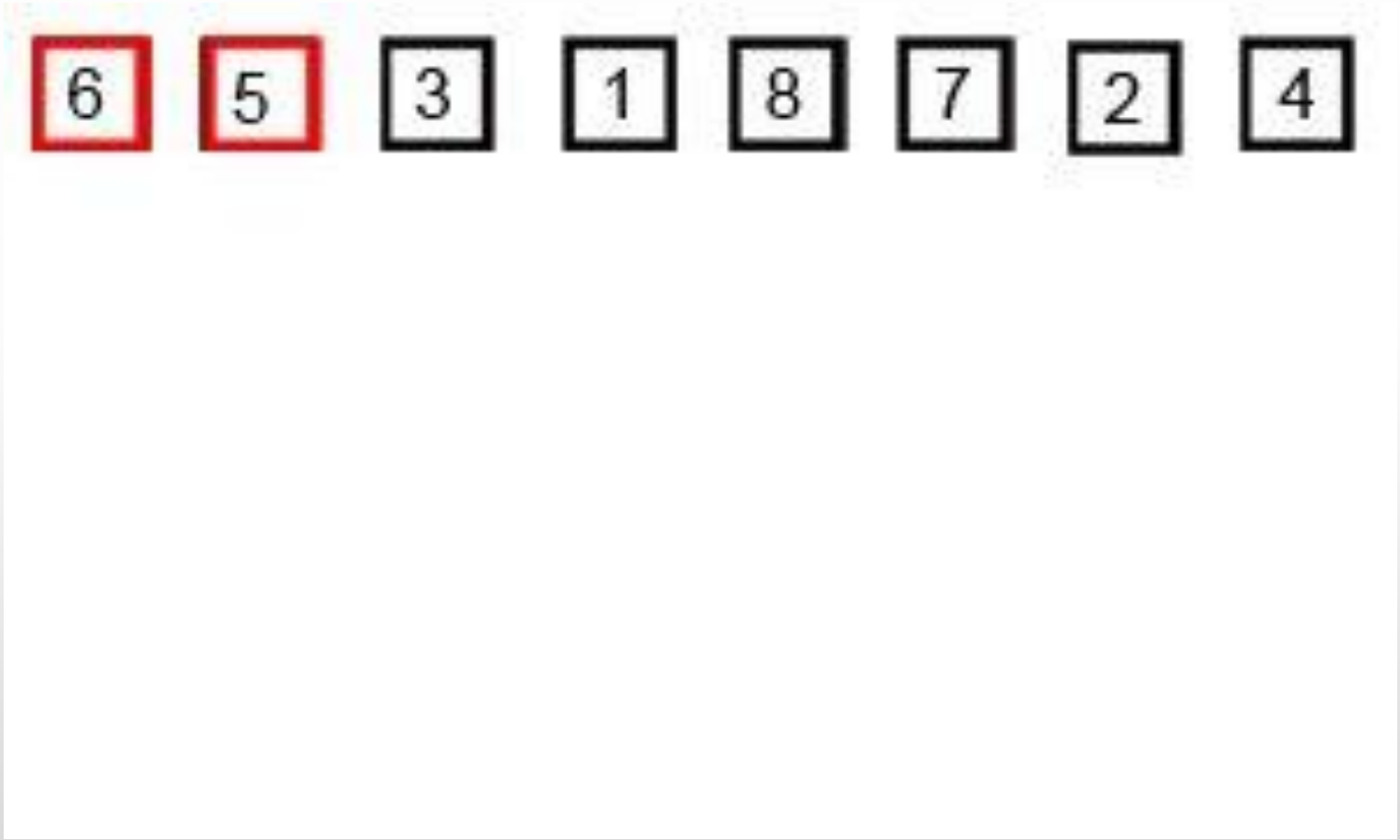


Merge Sort

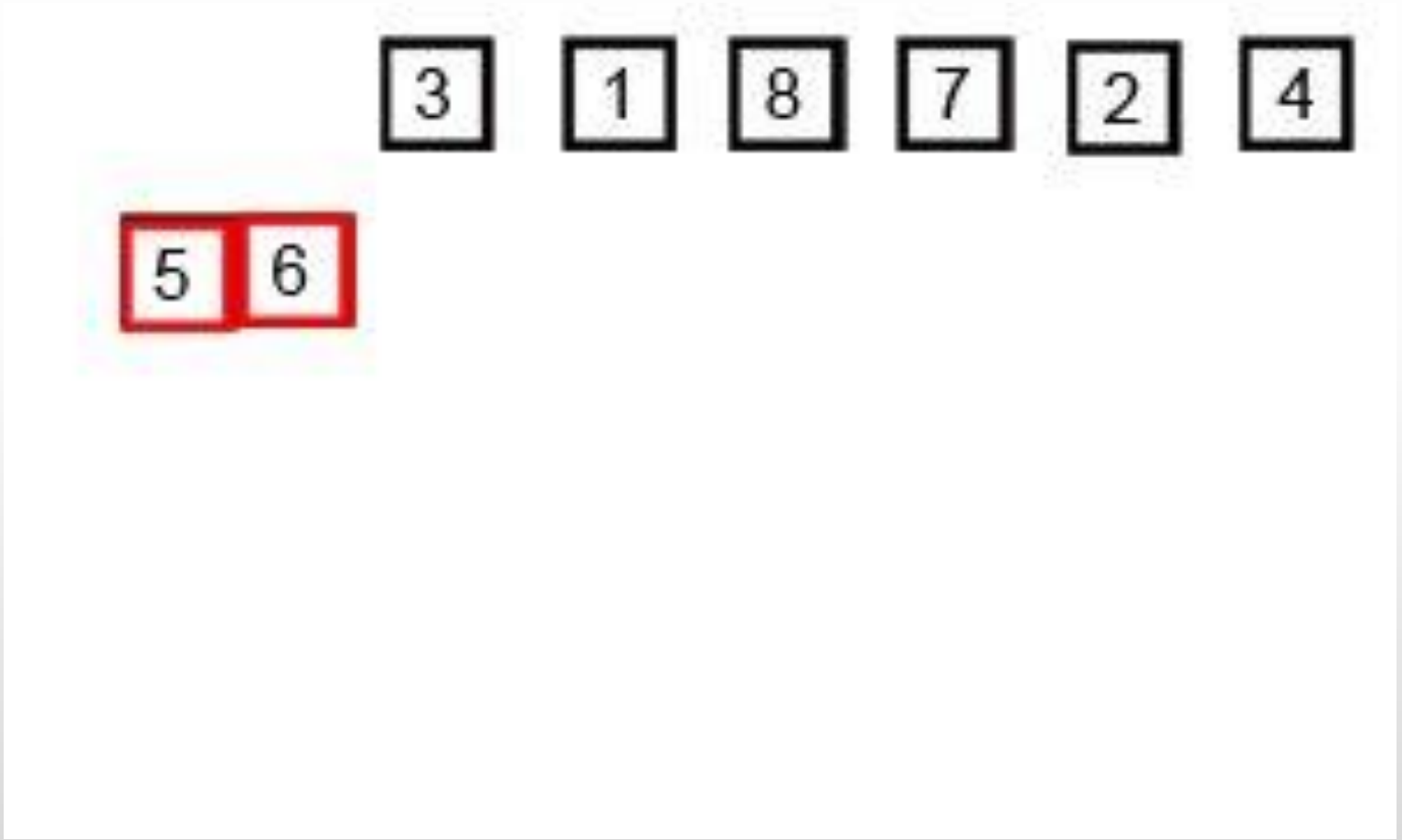
Merge Sort



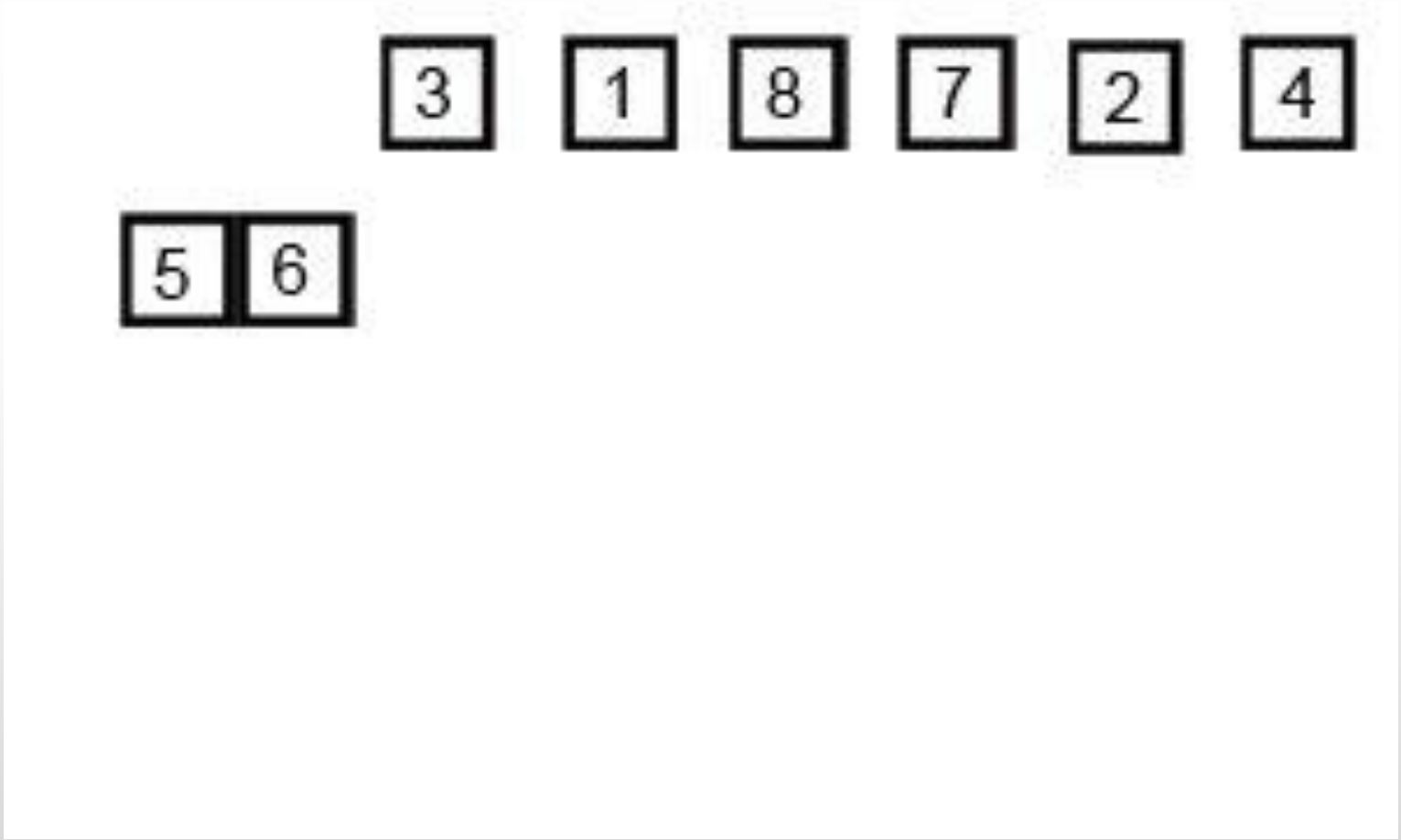
Merge Sort



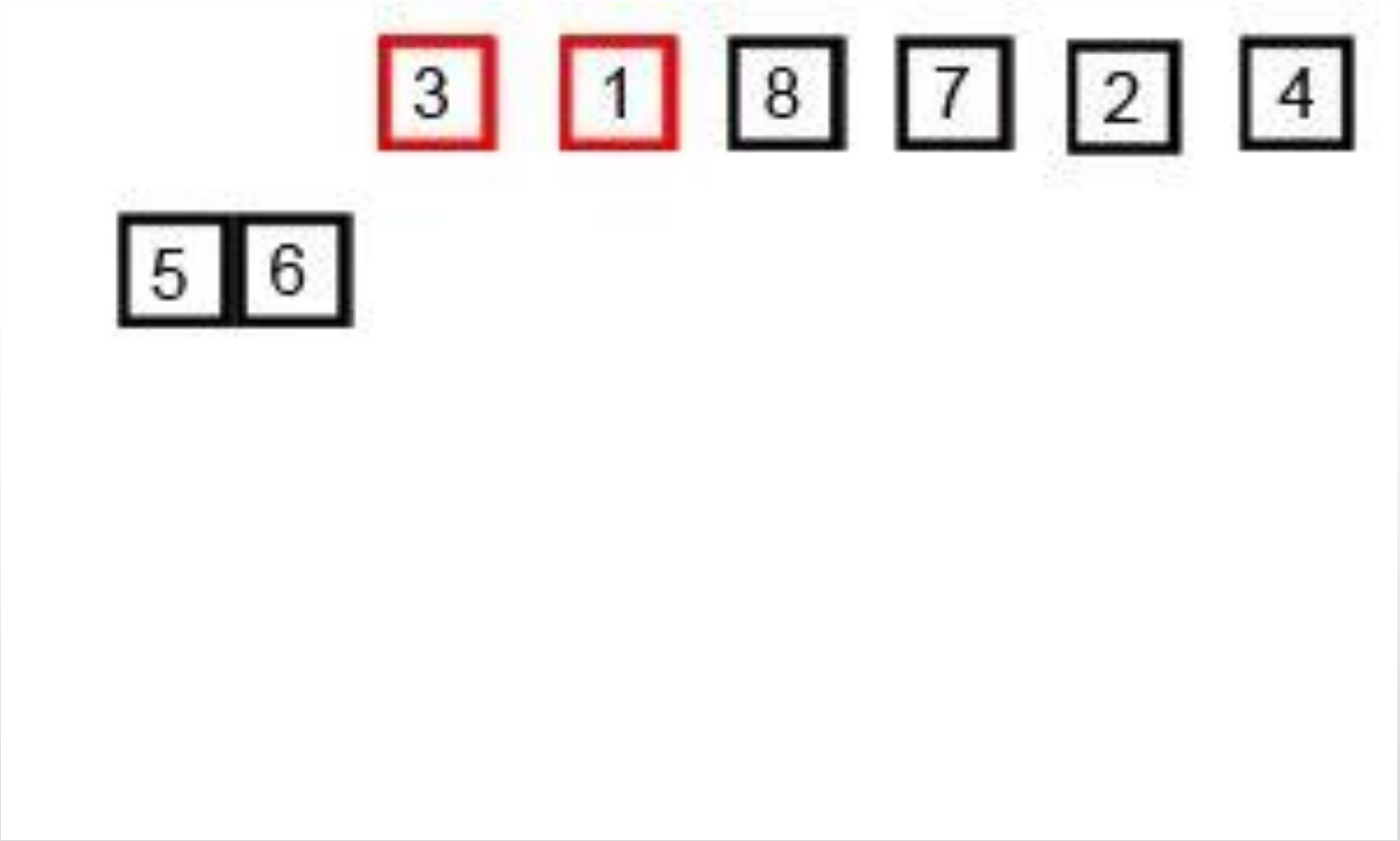
Merge Sort



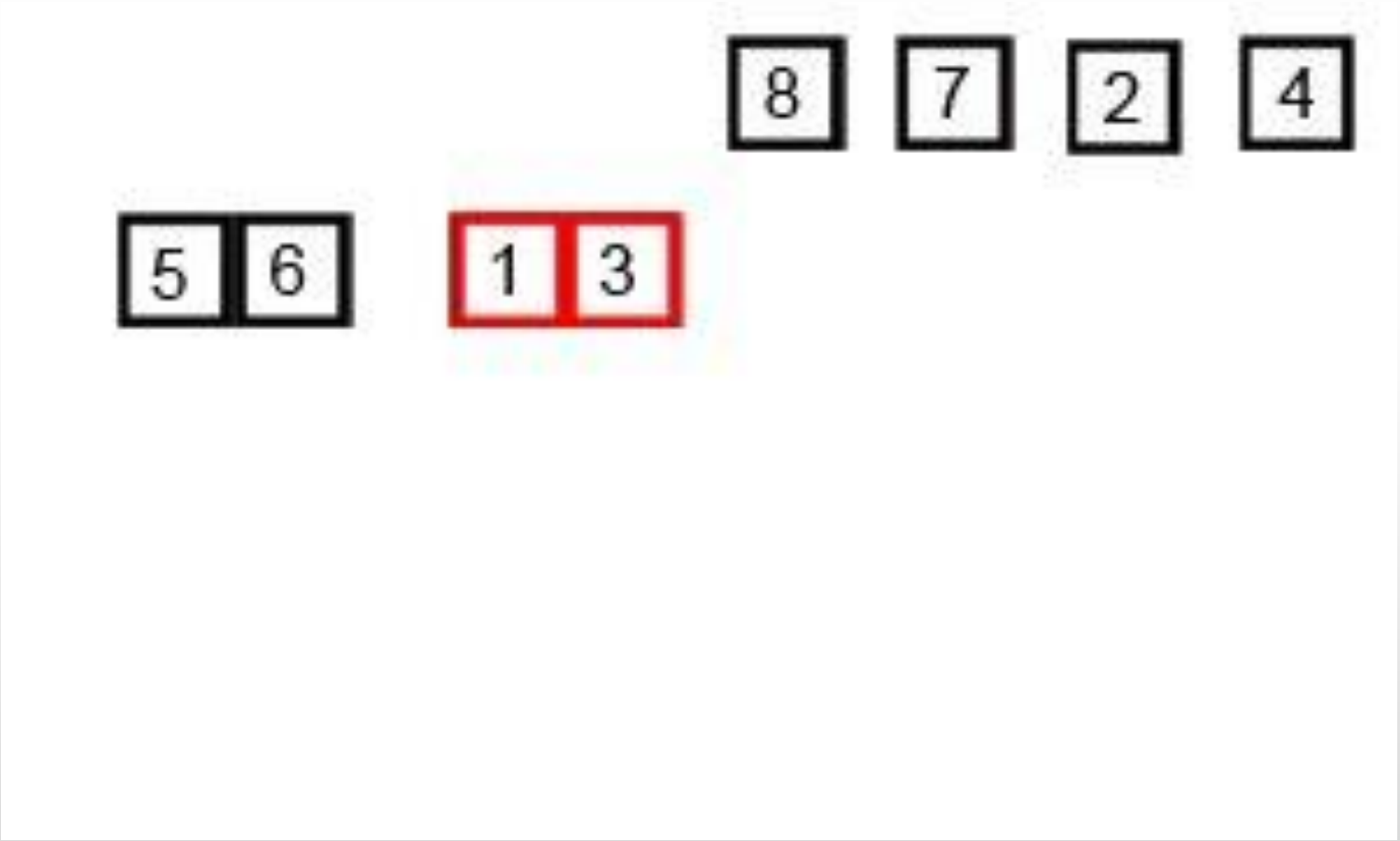
Merge Sort



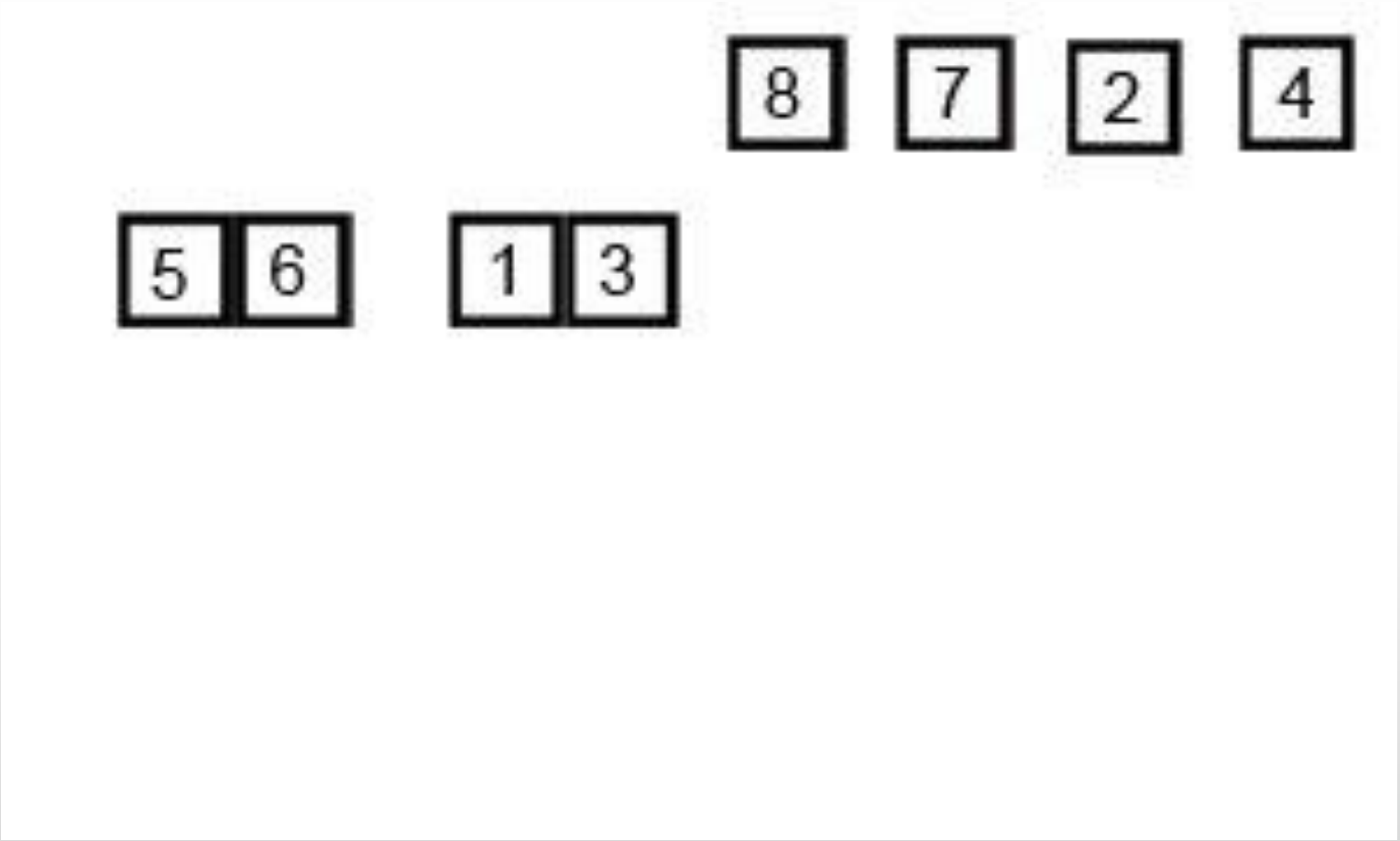
Merge Sort



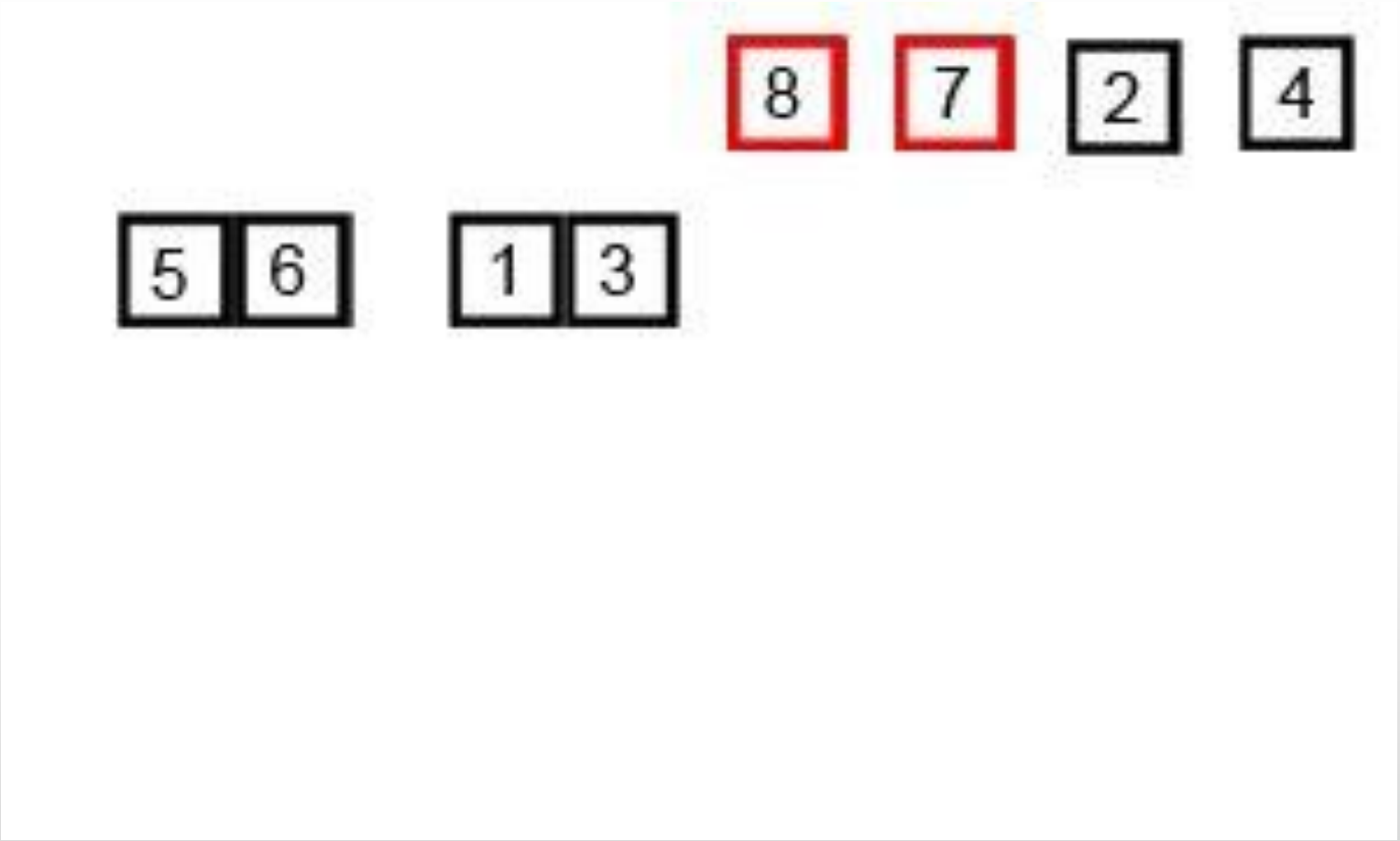
Merge Sort



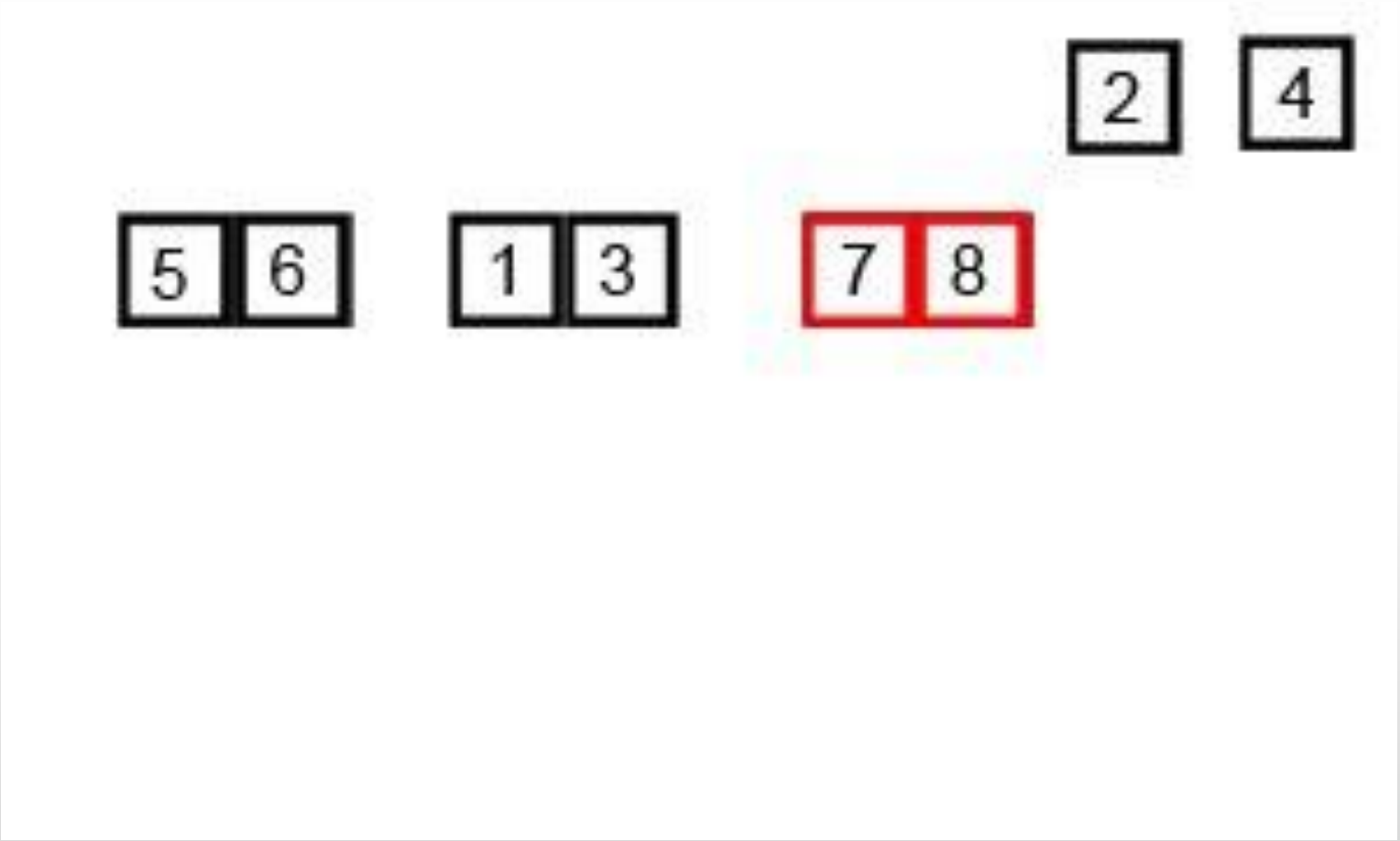
Merge Sort



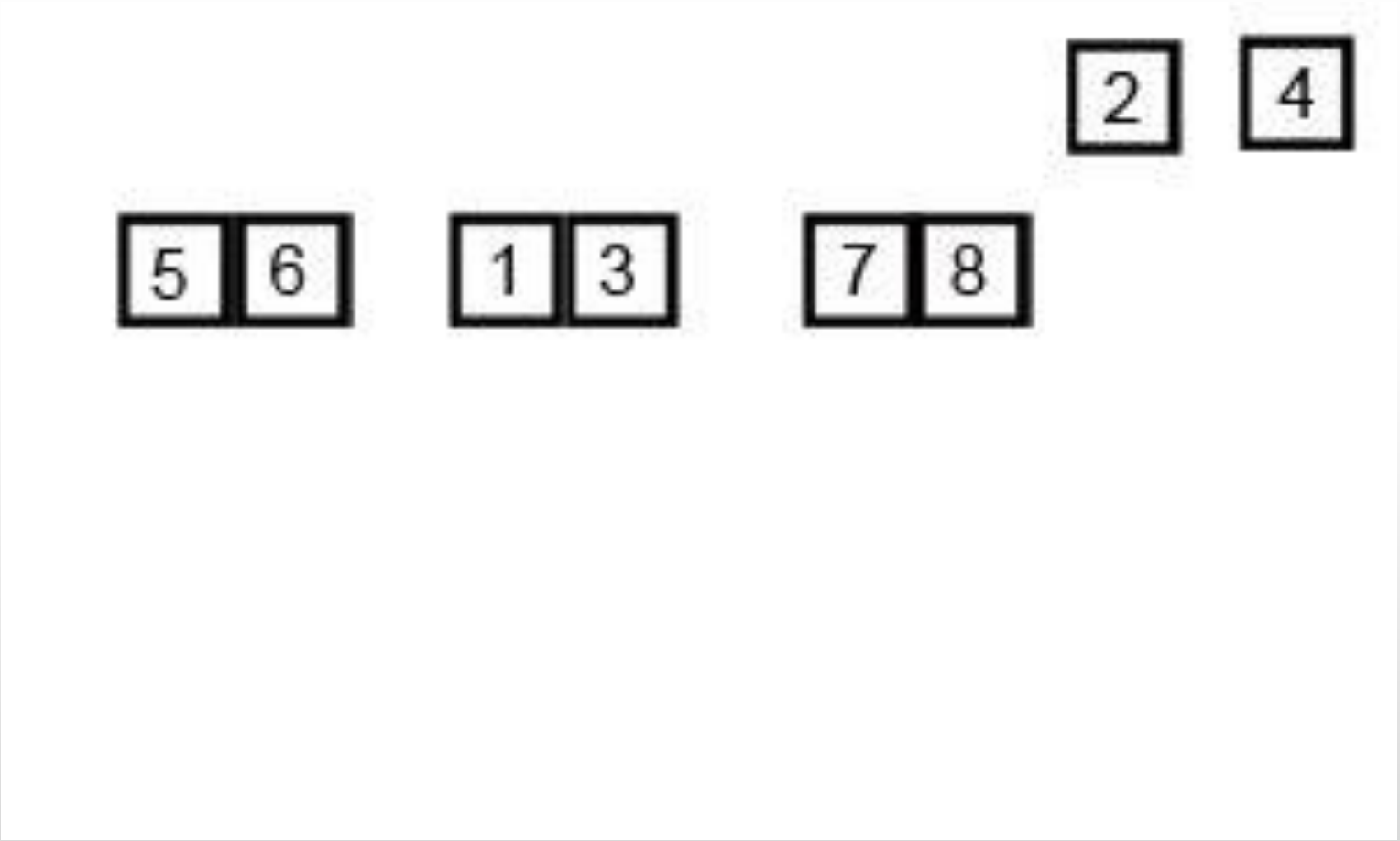
Merge Sort



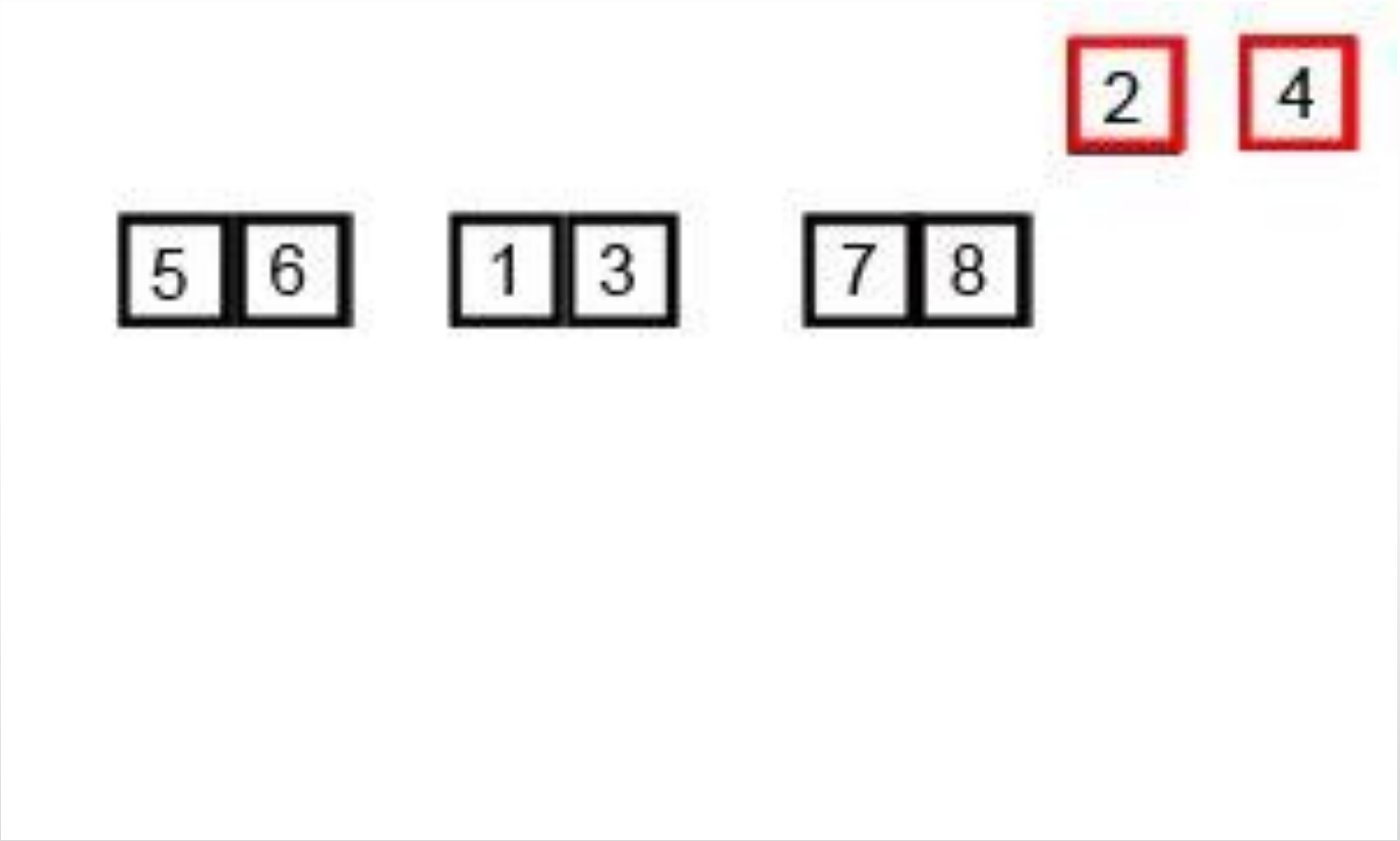
Merge Sort



Merge Sort



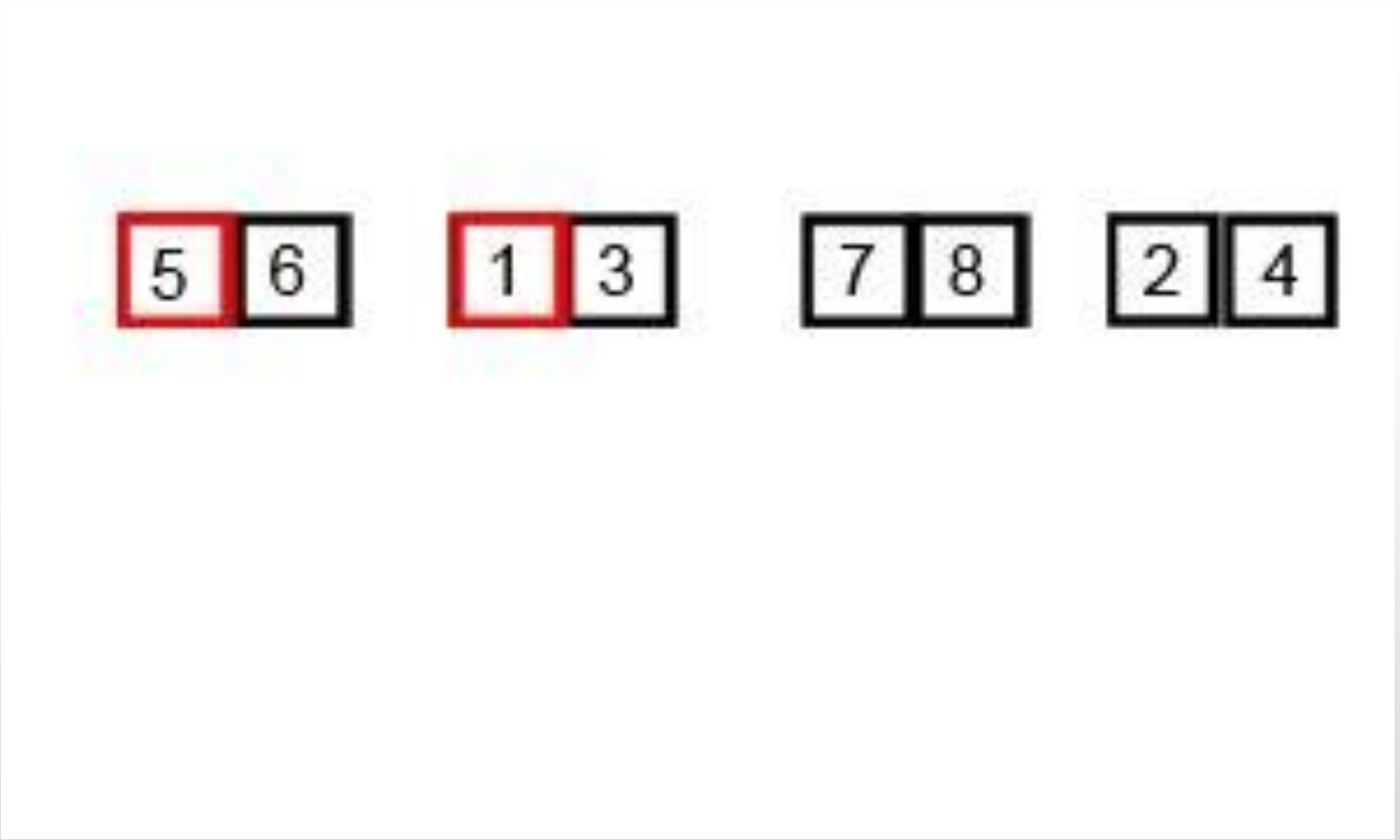
Merge Sort



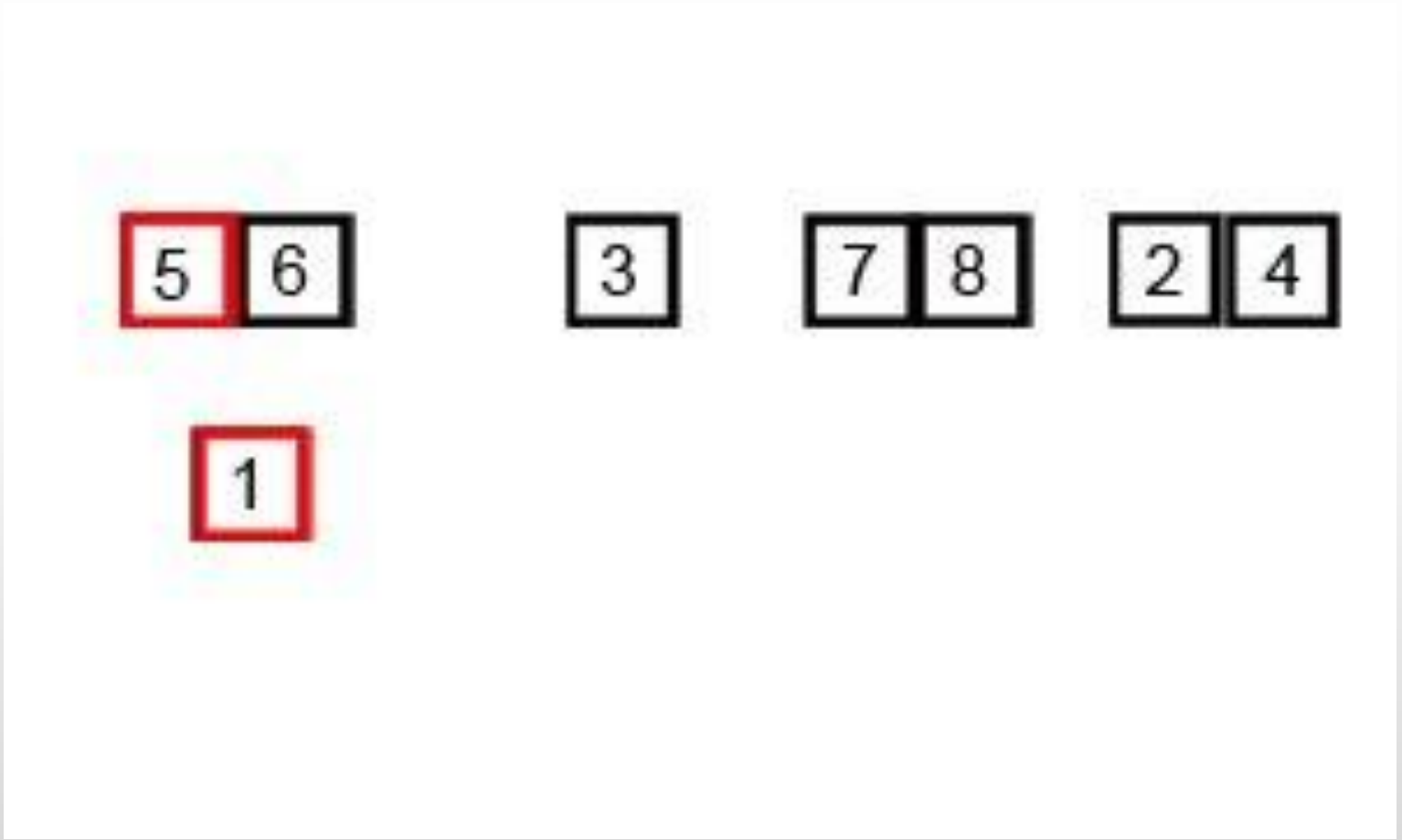
Merge Sort

Merge Sort

Merge Sort

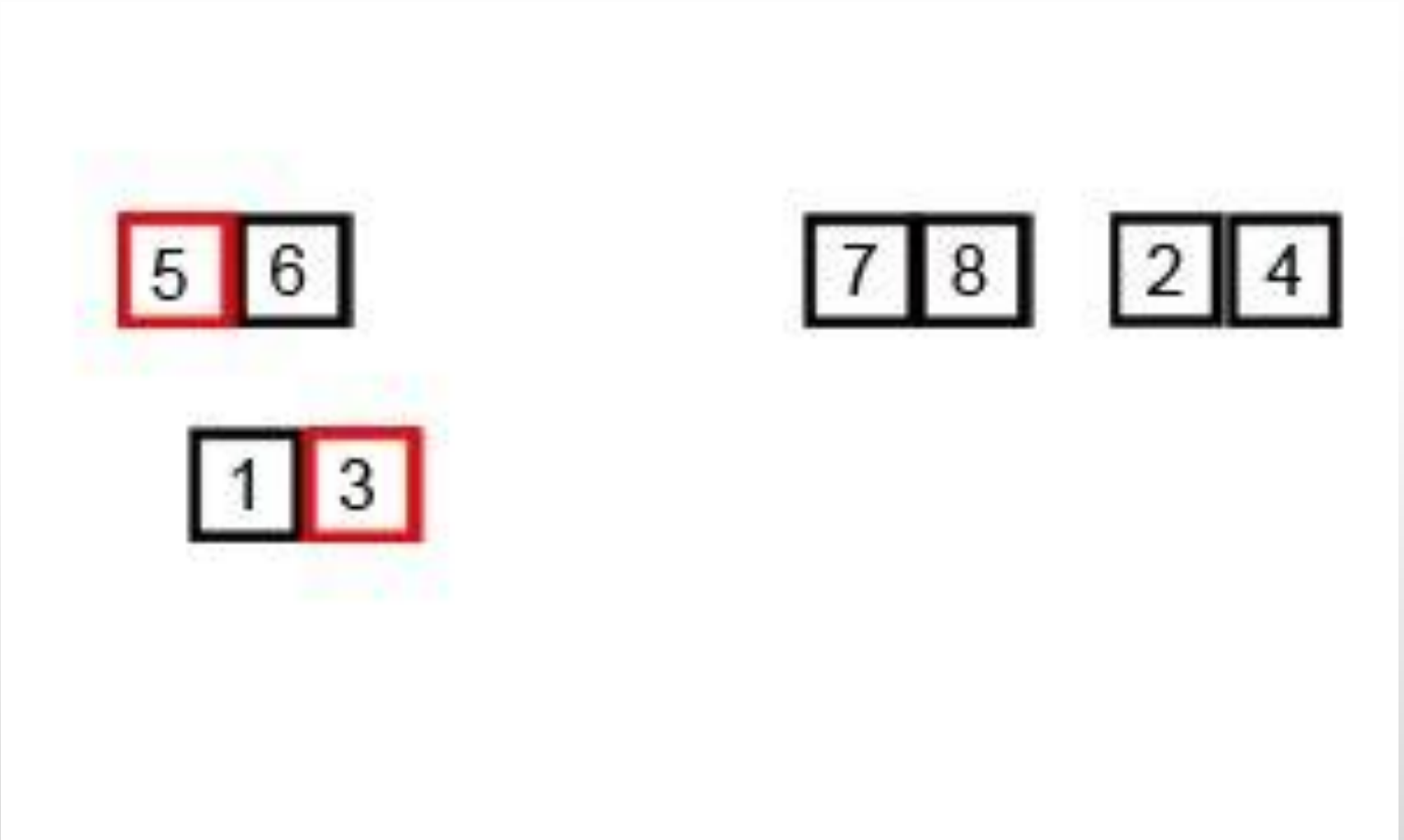


Merge Sort

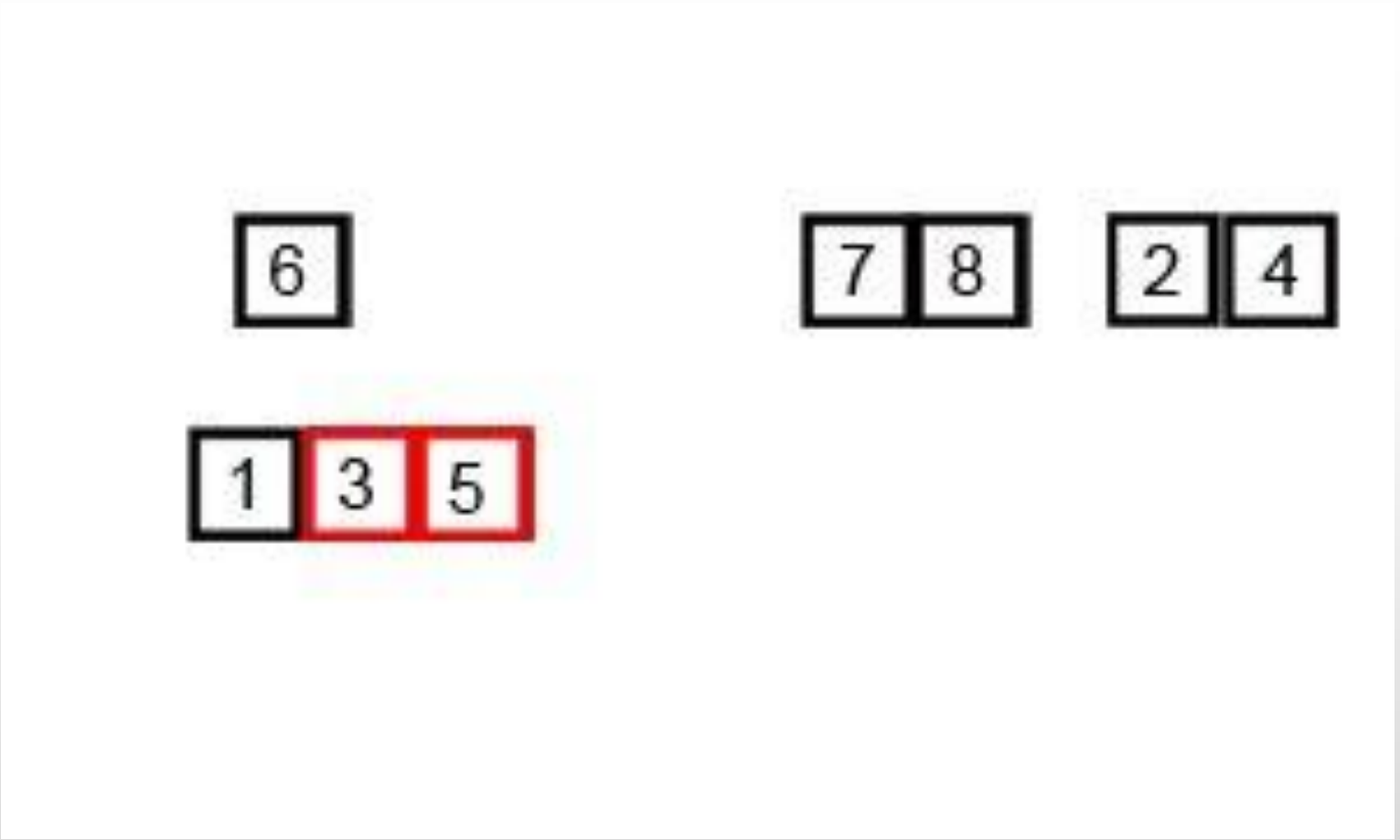


Merge Sort

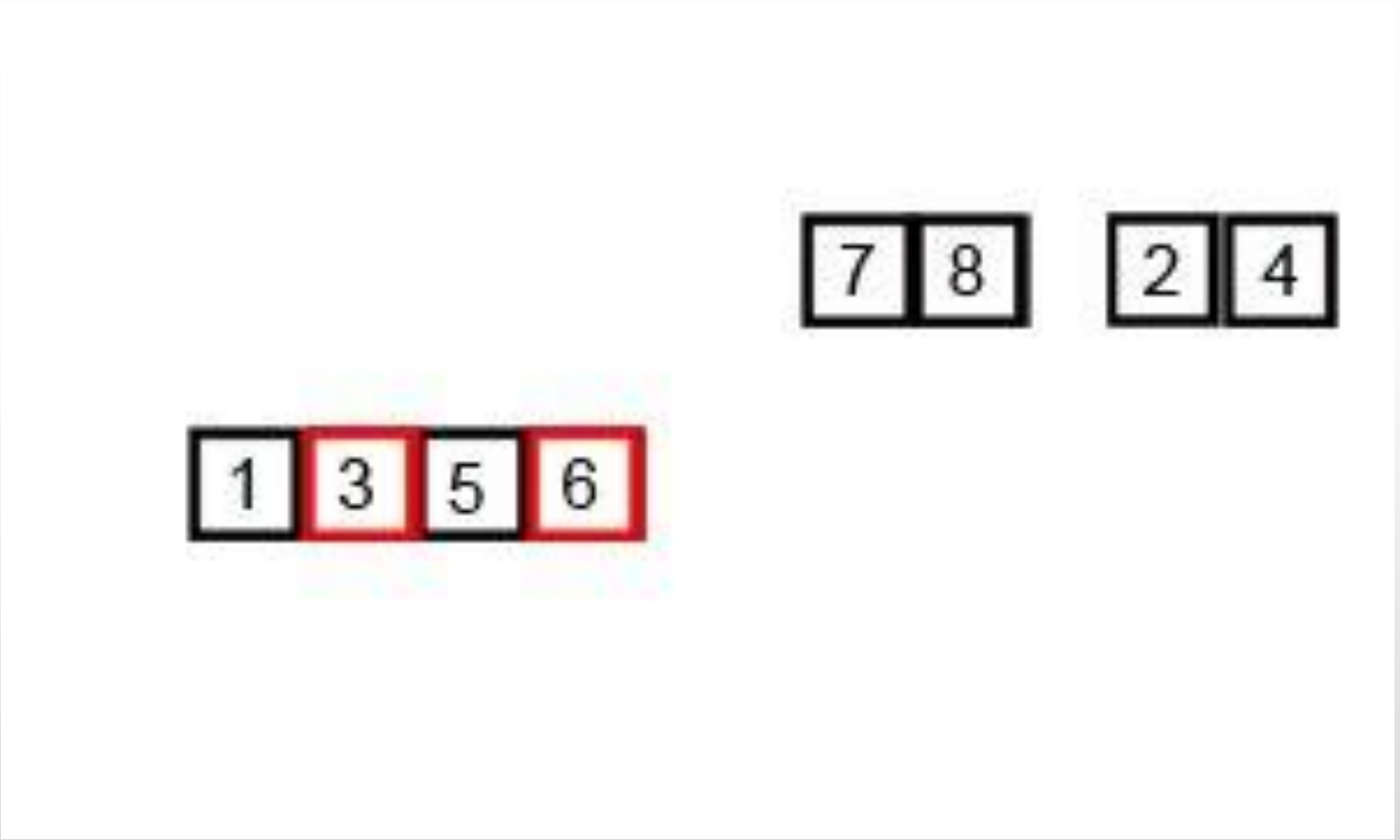
Merge Sort



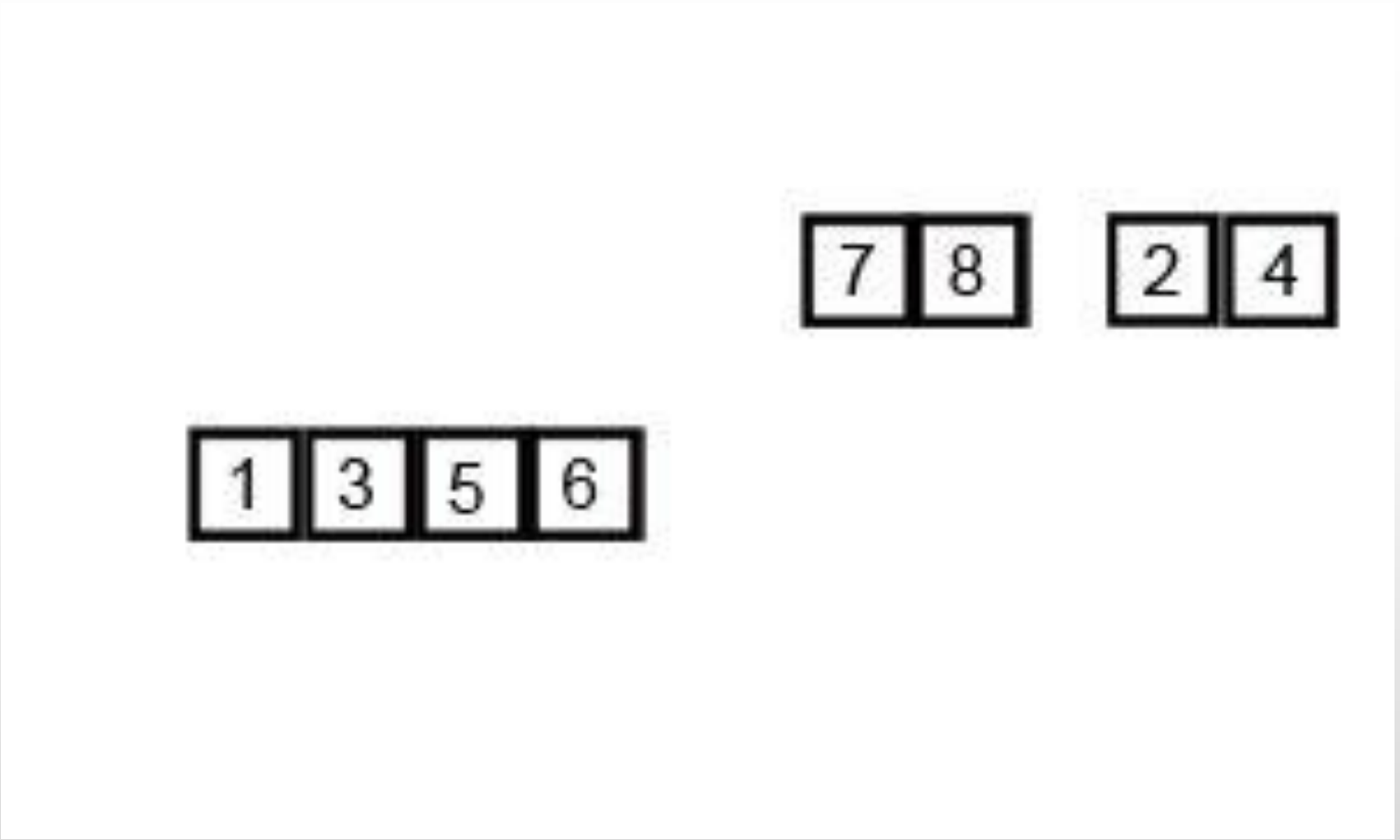
Merge Sort



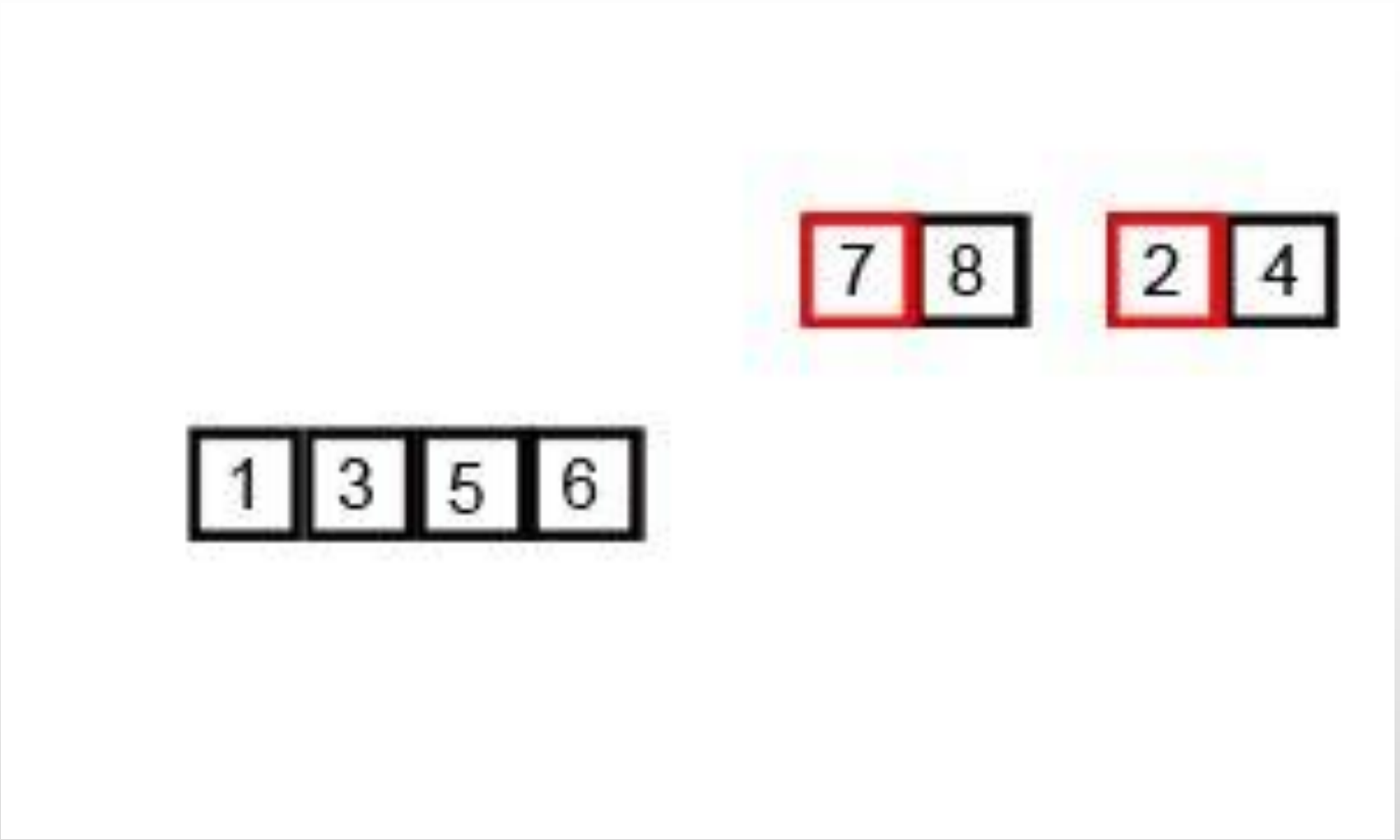
Merge Sort



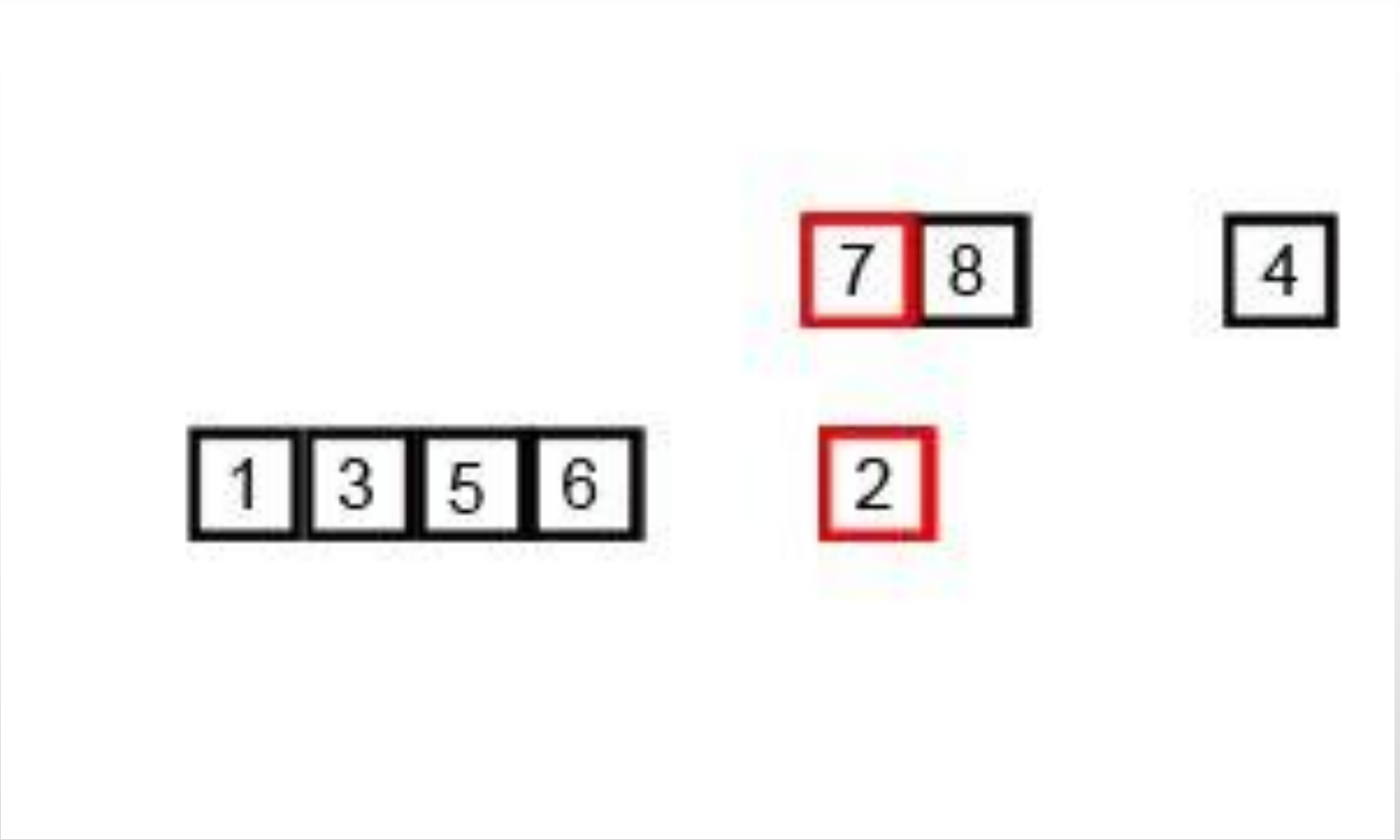
Merge Sort



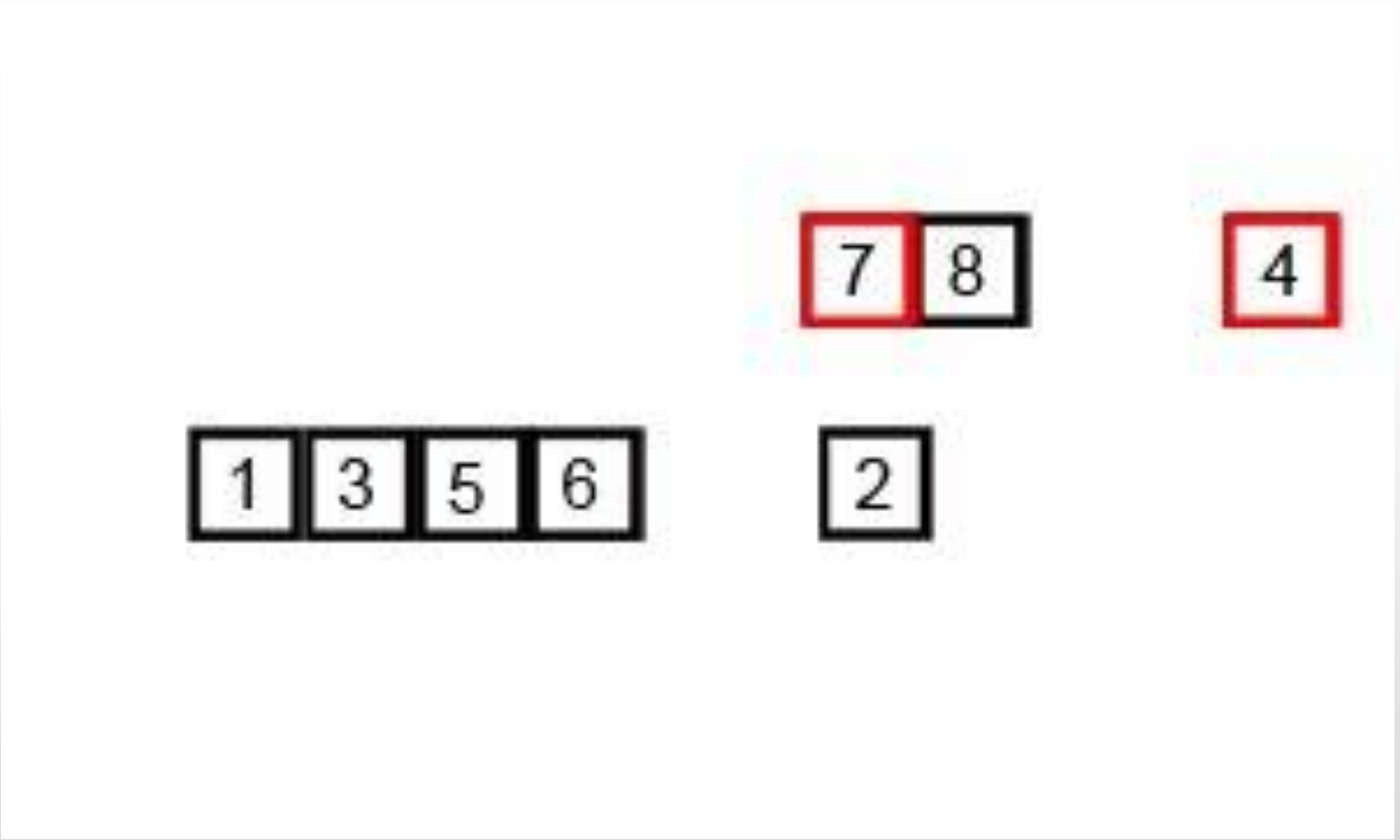
Merge Sort



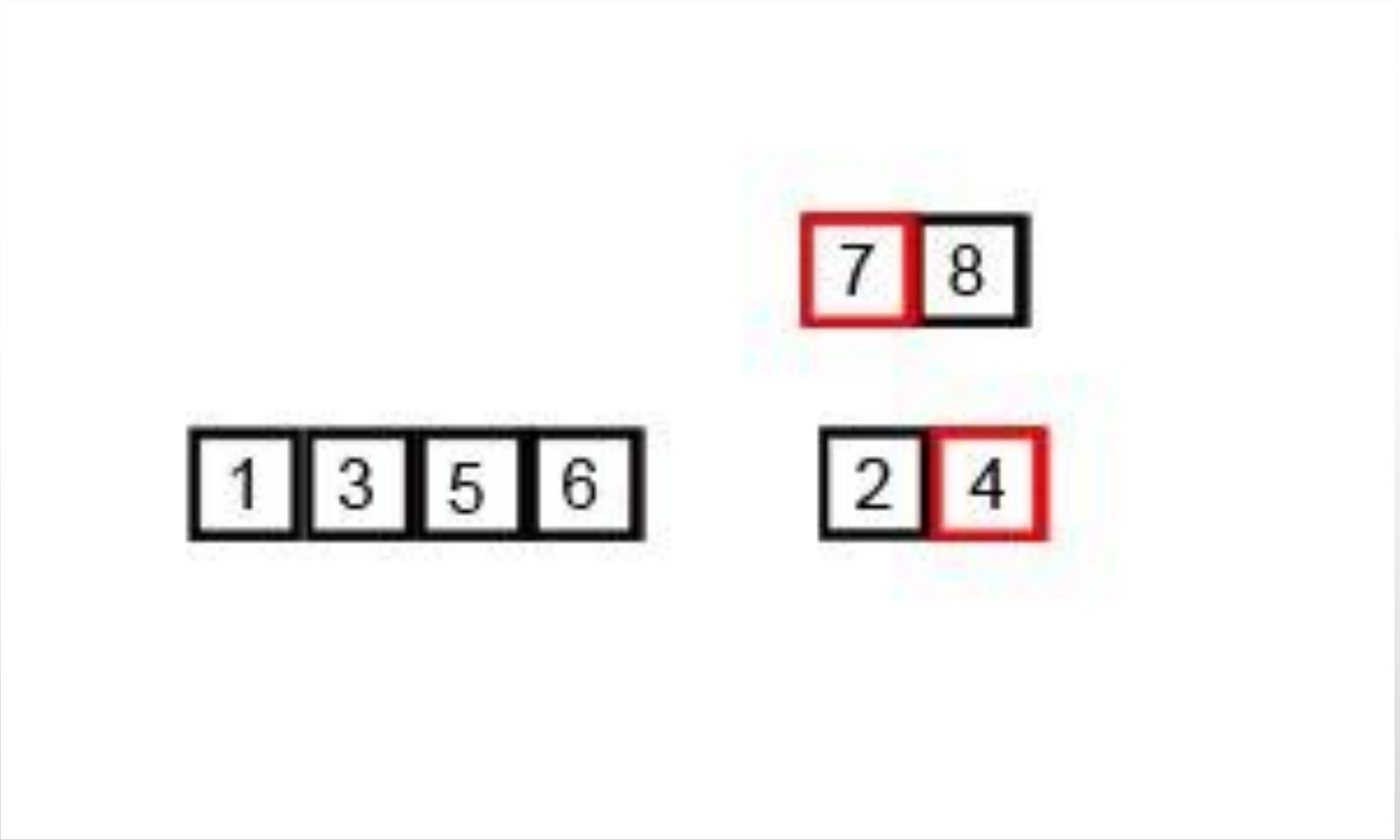
Merge Sort



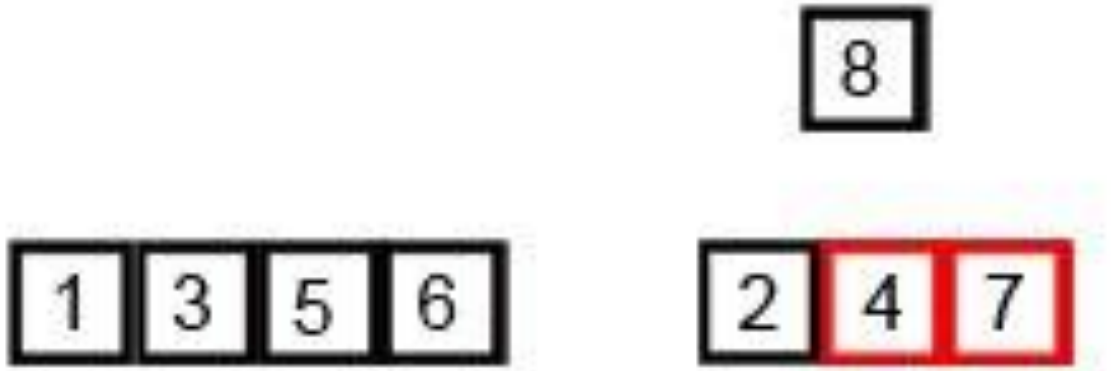
Merge Sort



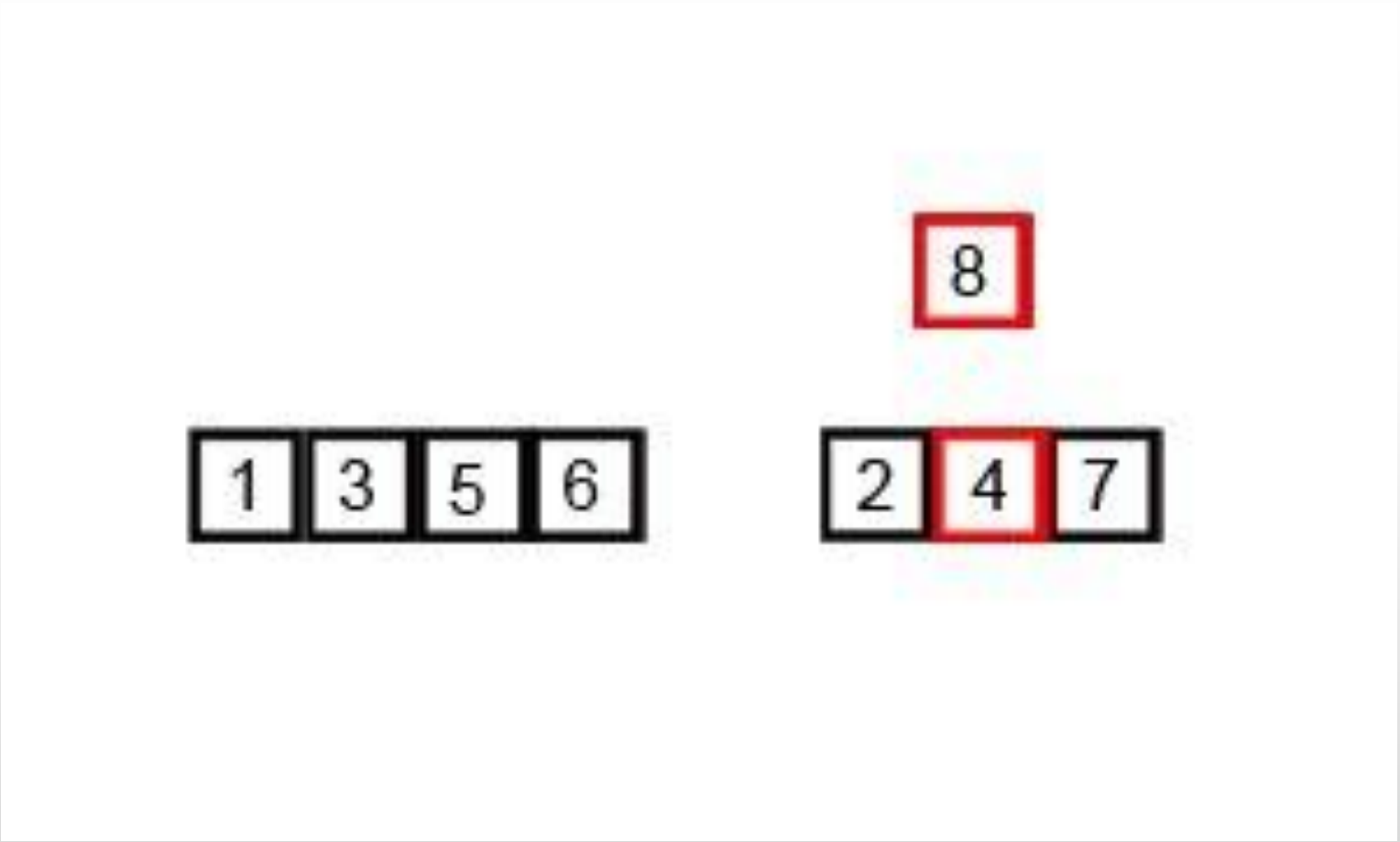
Merge Sort



Merge Sort



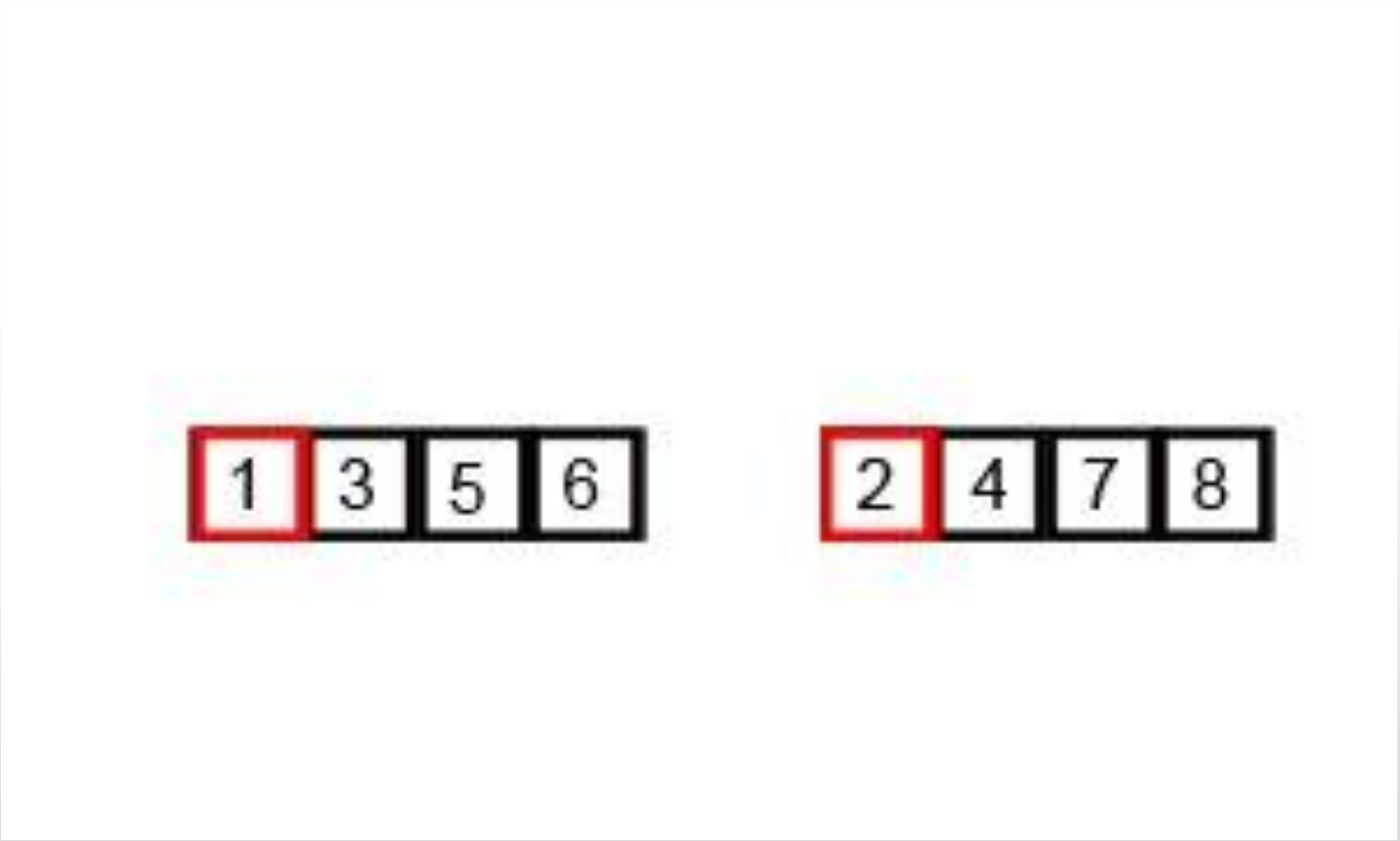
Merge Sort



Merge Sort

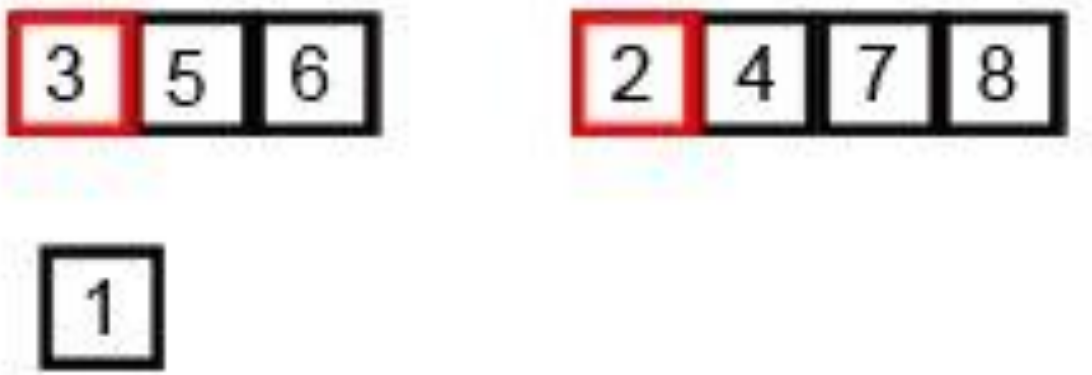
Merge Sort

Merge Sort

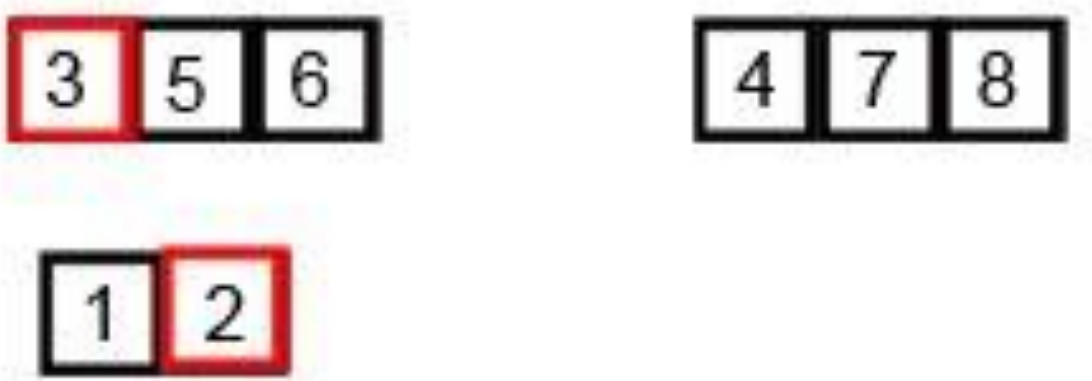


Merge Sort

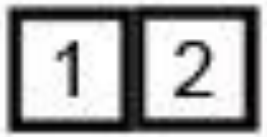
Merge Sort



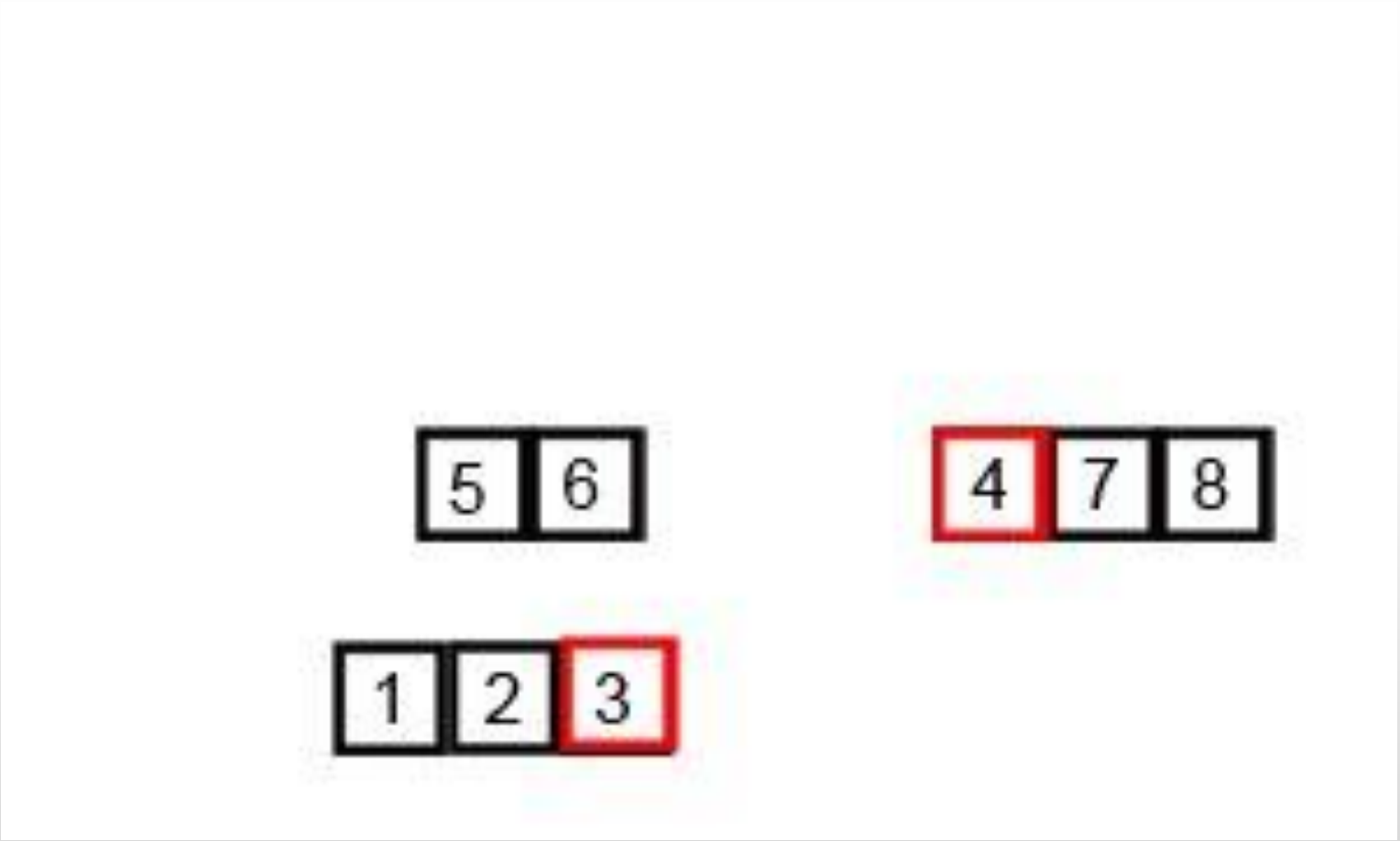
Merge Sort



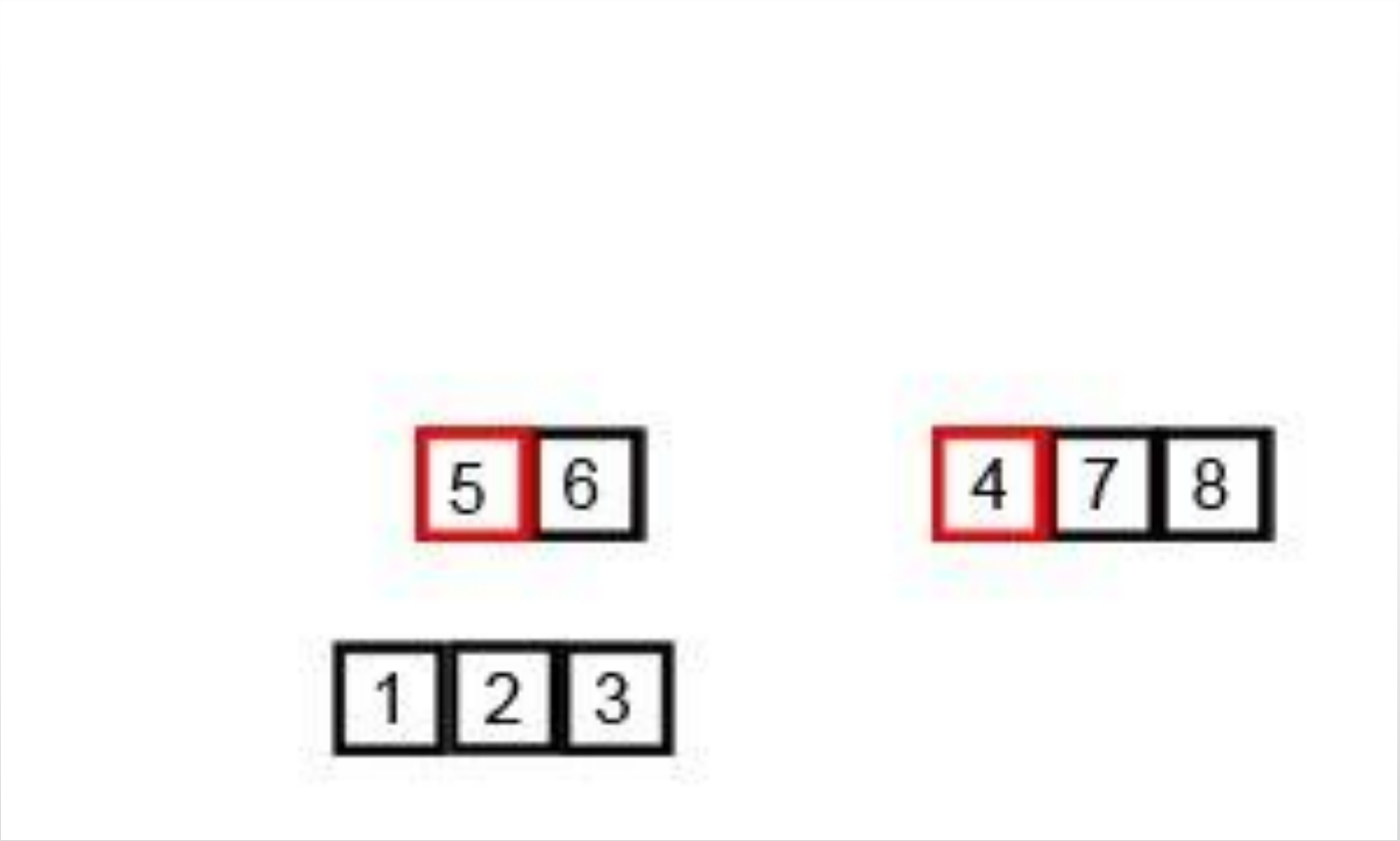
Merge Sort



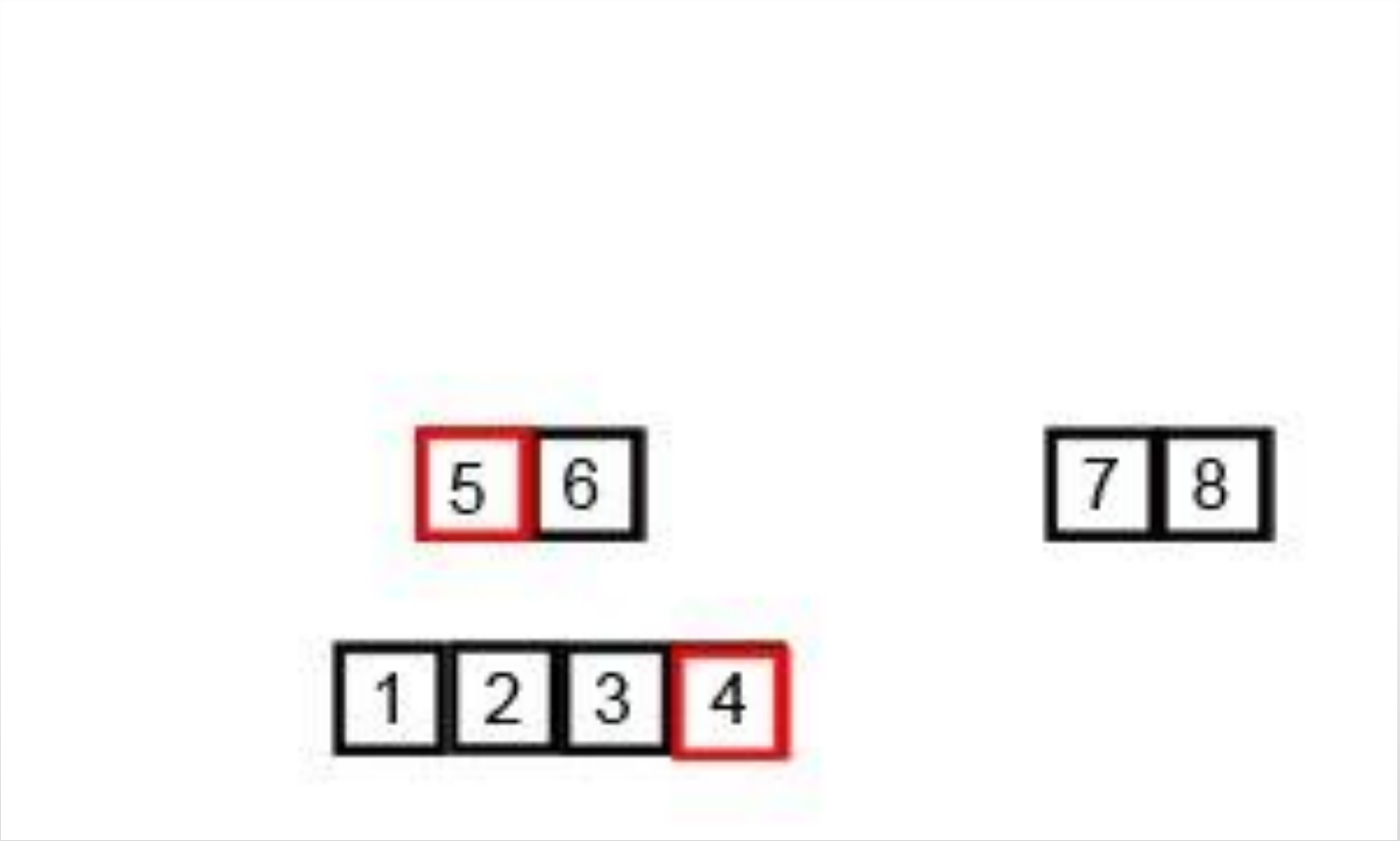
Merge Sort



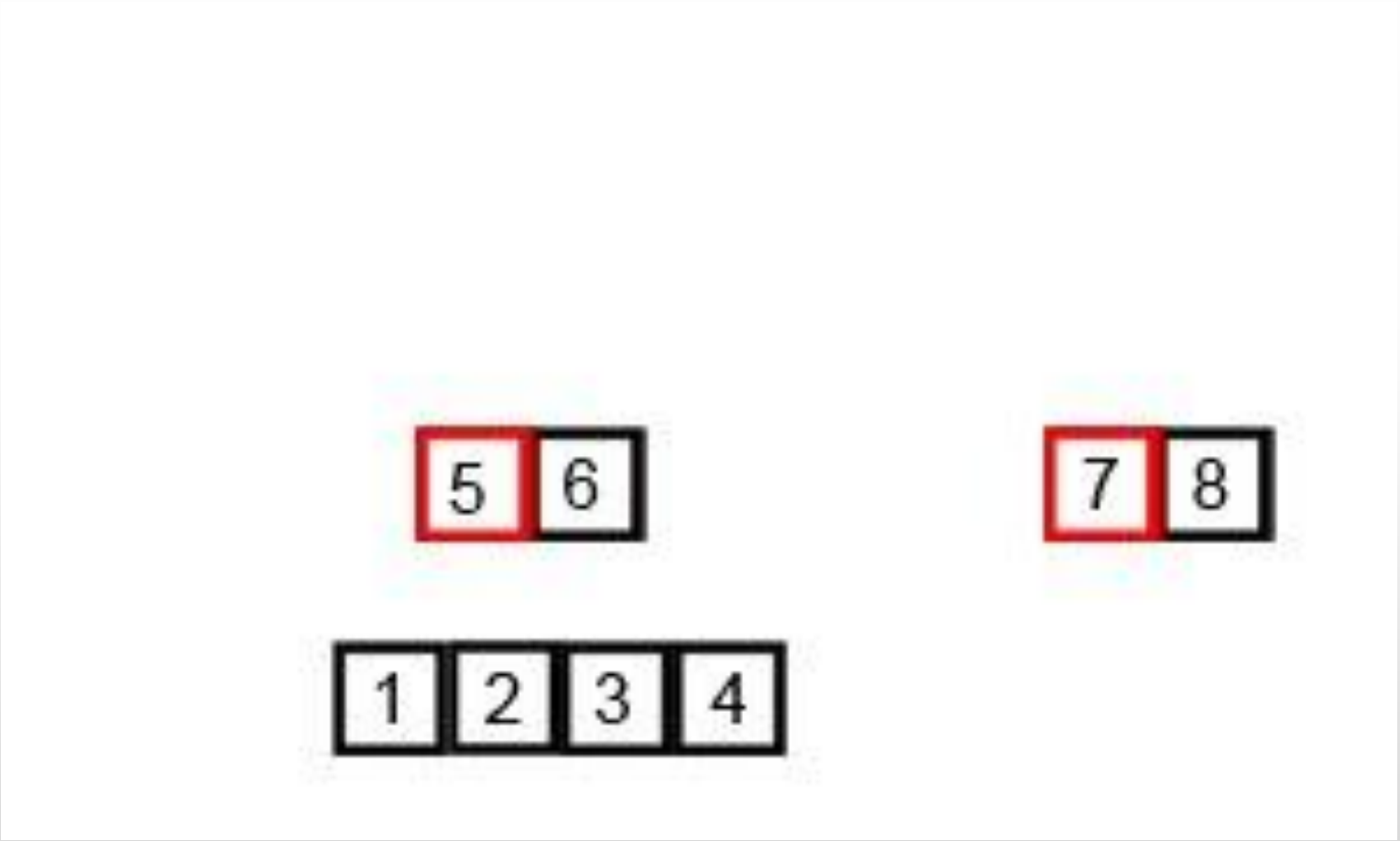
Merge Sort



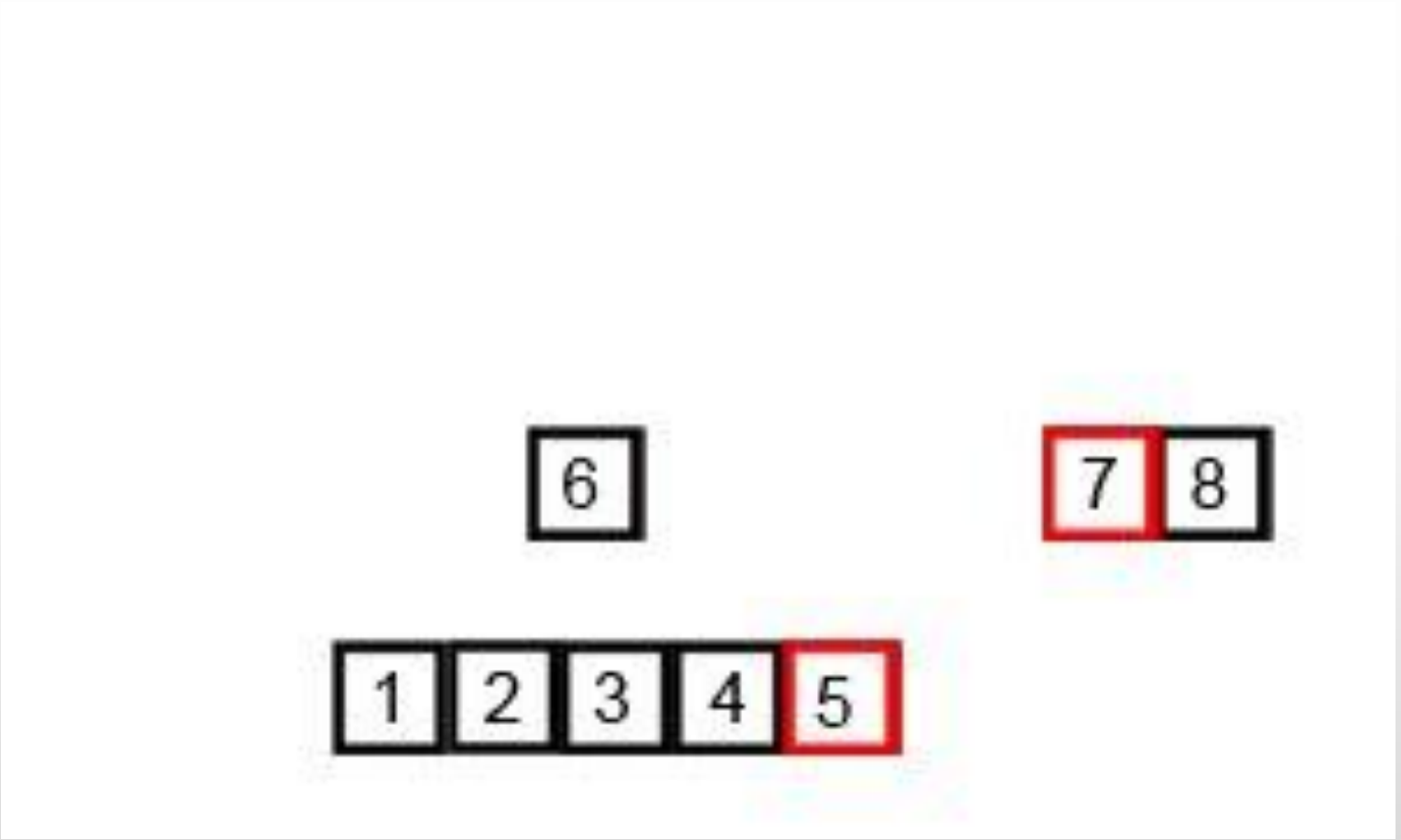
Merge Sort



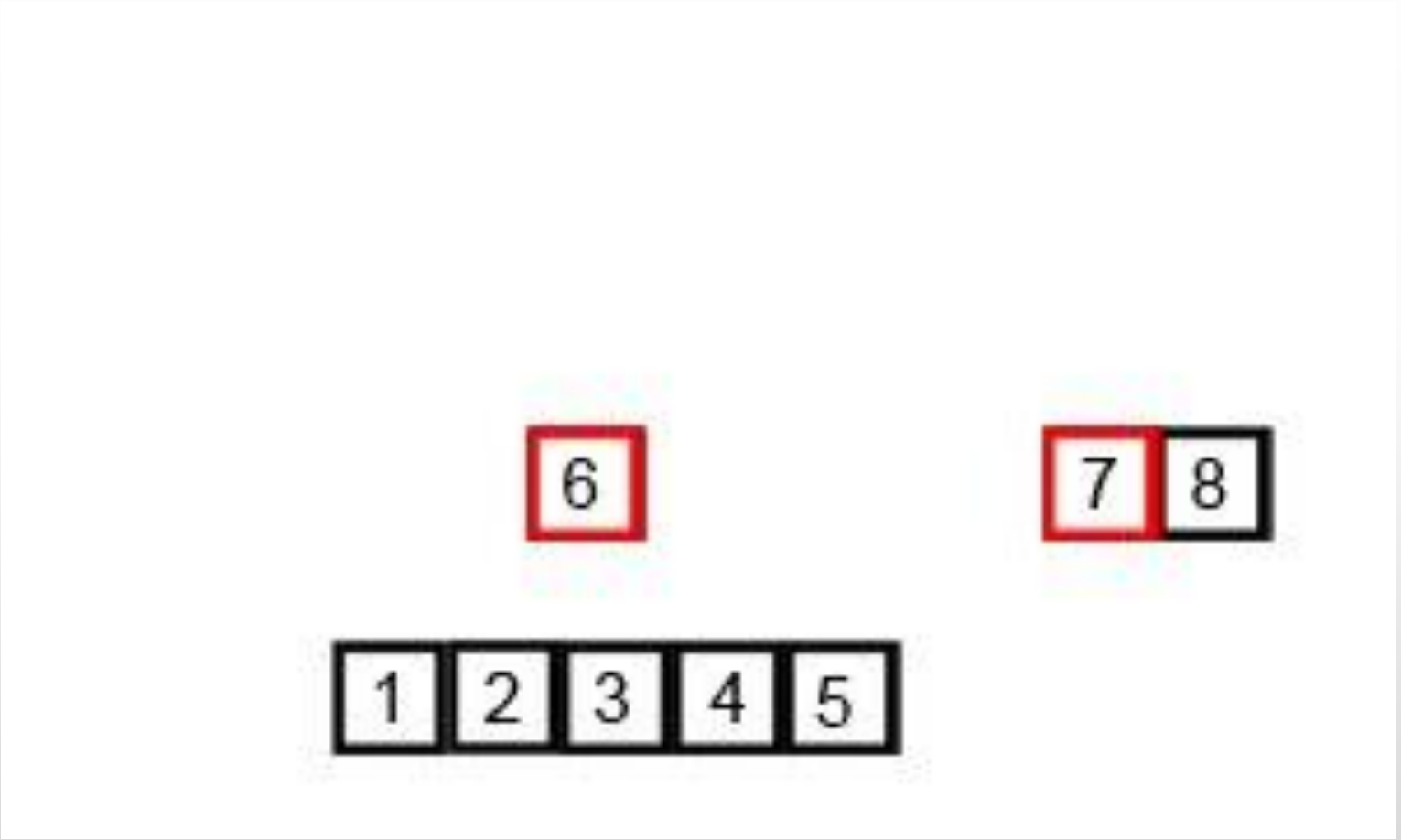
Merge Sort



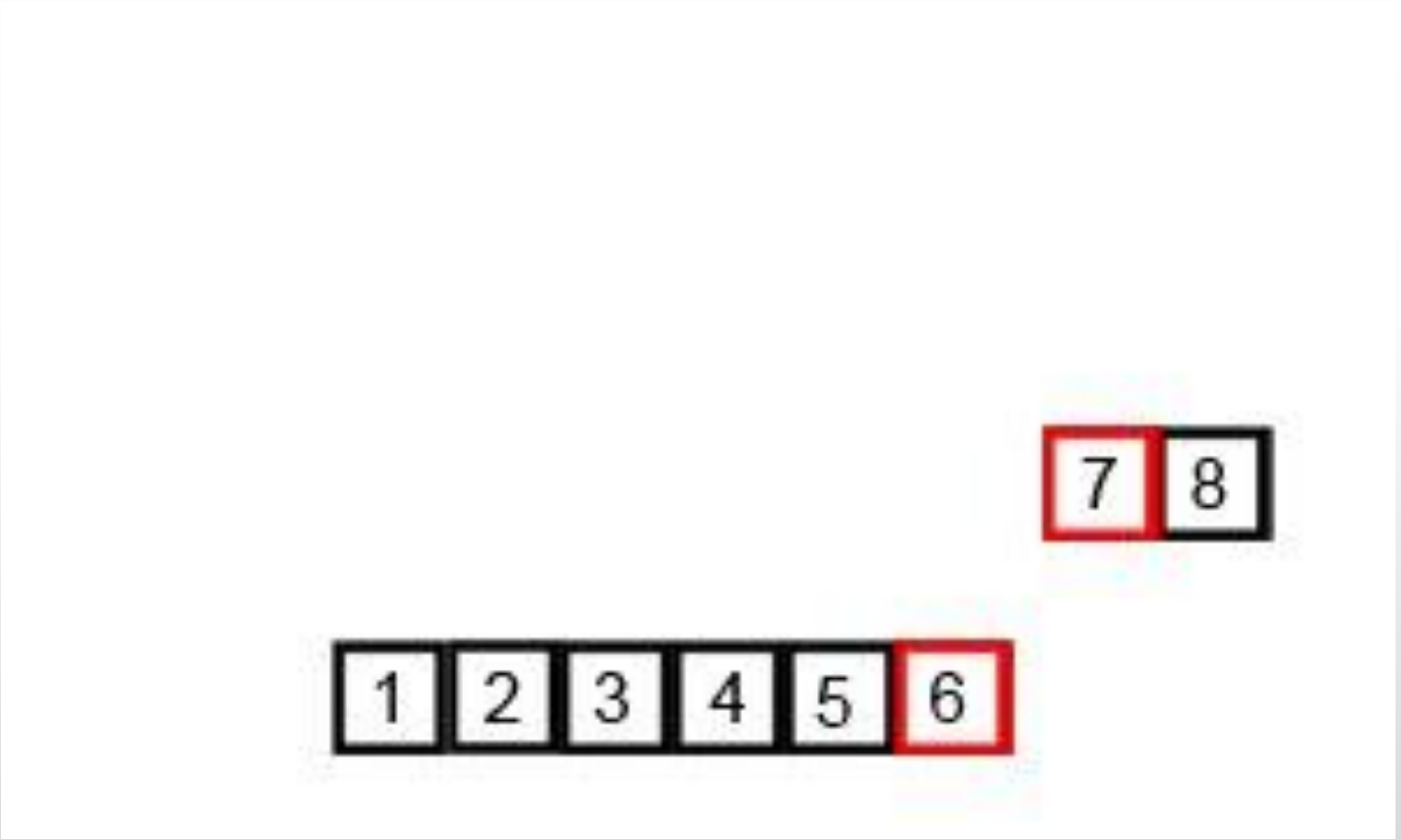
Merge Sort



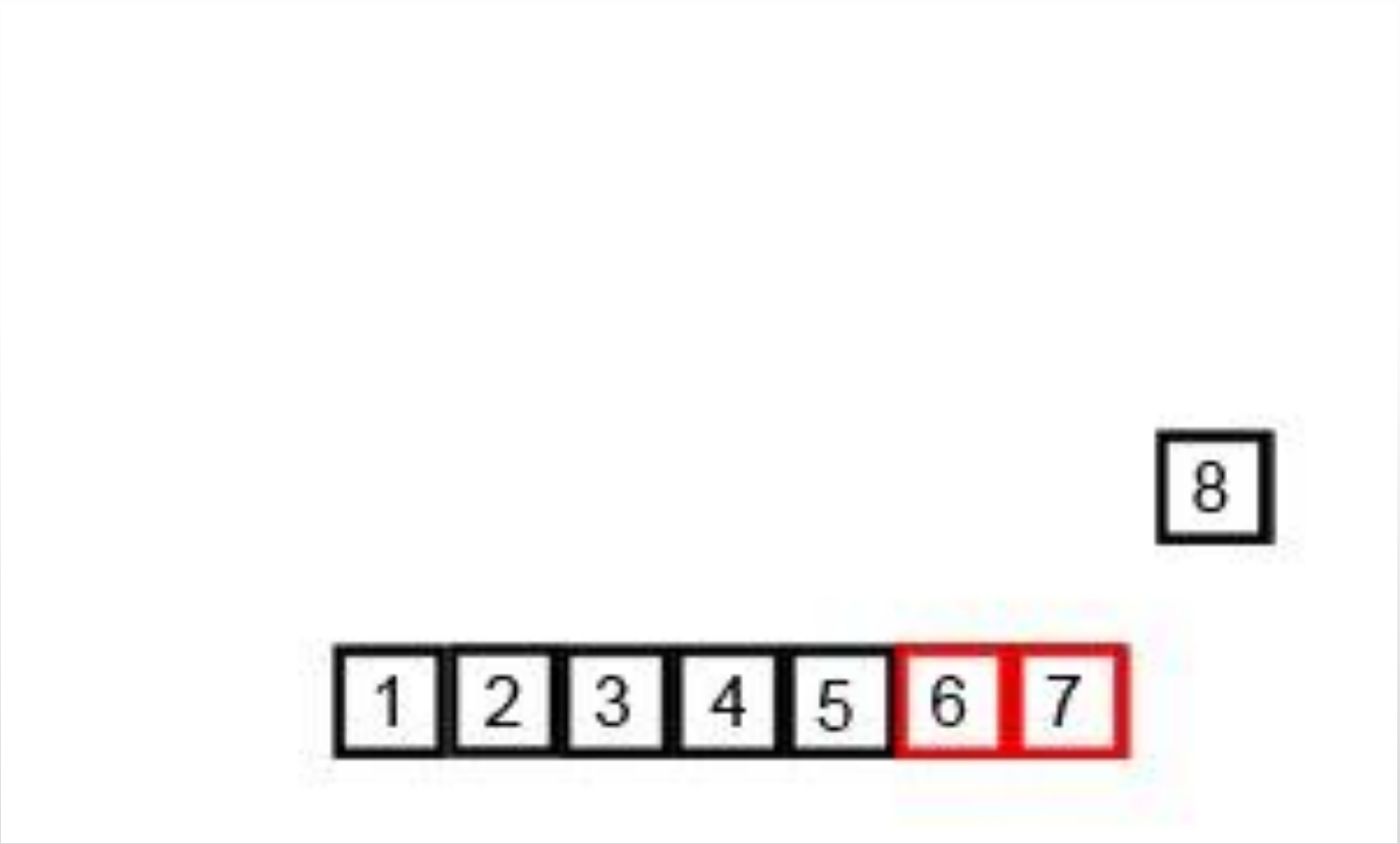
Merge Sort



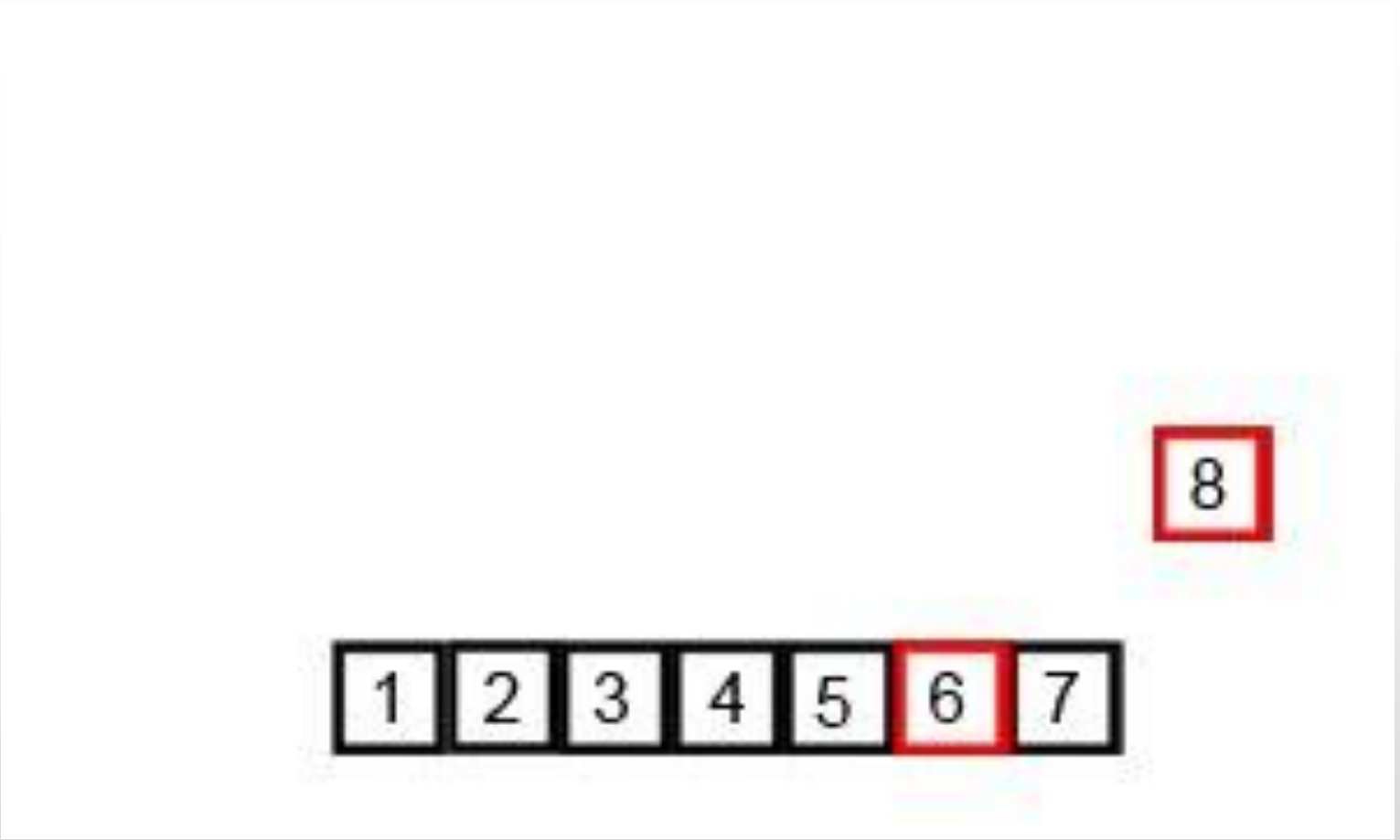
Merge Sort



Merge Sort



Merge Sort



Merge Sort

Merge Sort

Heap Sort Algoritması

- Elemanları sıralamak için kullanılan bir algoritma.
- Yığın (*Heapify*) veri yapısını kullanarak çalışır.
- Ortalama ve en kötü durum zaman karmaşıklığı *O(nlogn)*.

İşleyiş

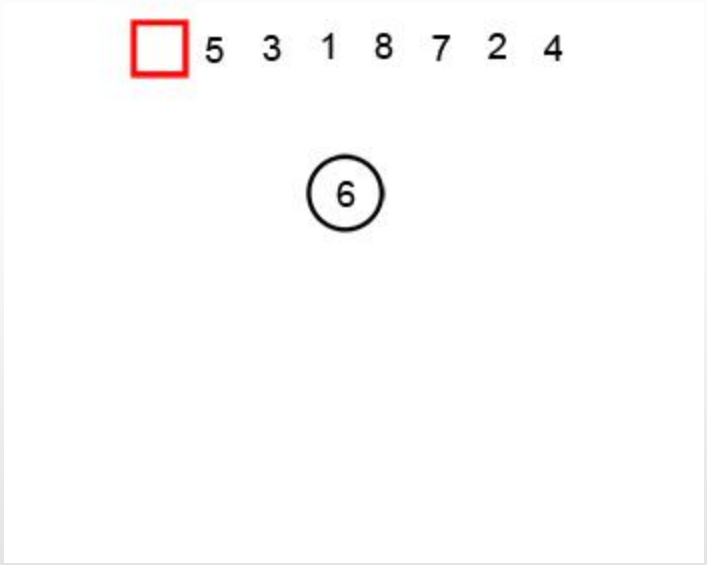
- Verilen dizi bir Max Heap yapısına dönüştürülür.
- Max Heap yapısından en büyük eleman alınıp dizinin sonuna yerleştirilir.
- Kök elemanı hariç kalan dizi tekrar Max Heap yapısına dönüştürülür.
- Tüm elemanlar sıralanana kadar adımlar tekrar edilir.

Örnek

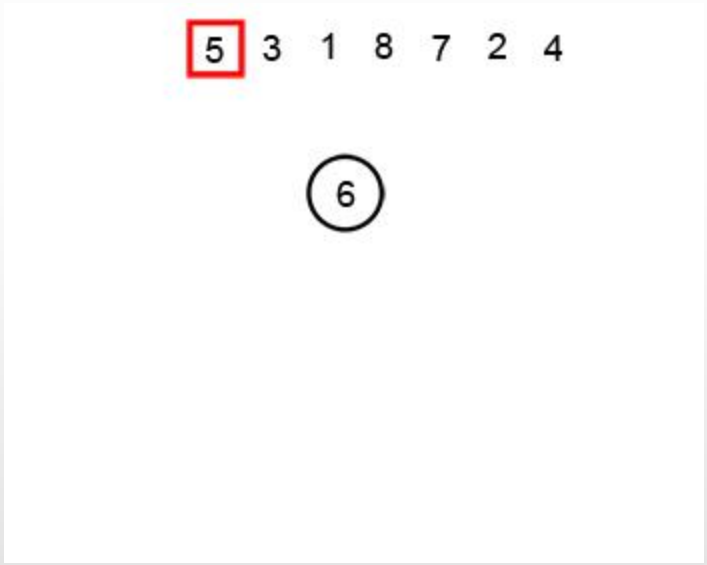
- Dizi:
  - [4, 10, 3, 5, 1]
- İşleyiş:
  - Max Heap Oluştur: [10, 5, 3, 4, 1]
  - Kök Elemanı Değiştir: [1, 5, 3, 4, 10]
  - Yeniden Heapify: [5, 4, 3, 1], [10]
  - Tekrarla: [4, 3, 1], [5],

Heap Sort

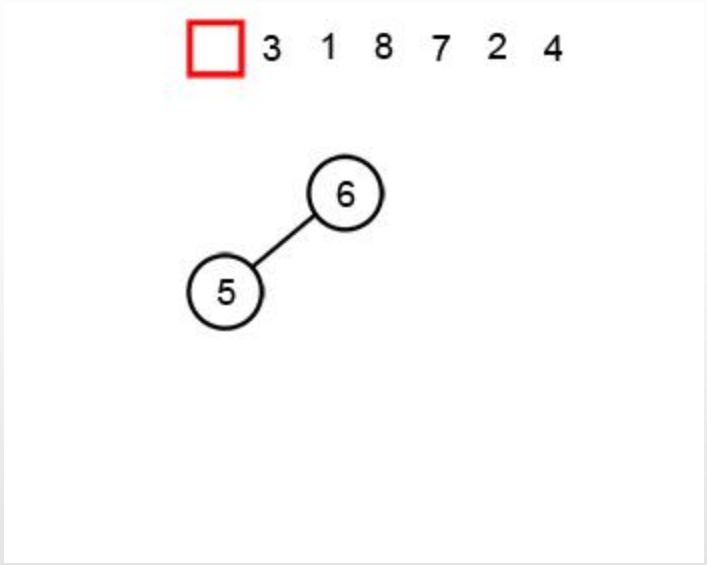
Heap Sort



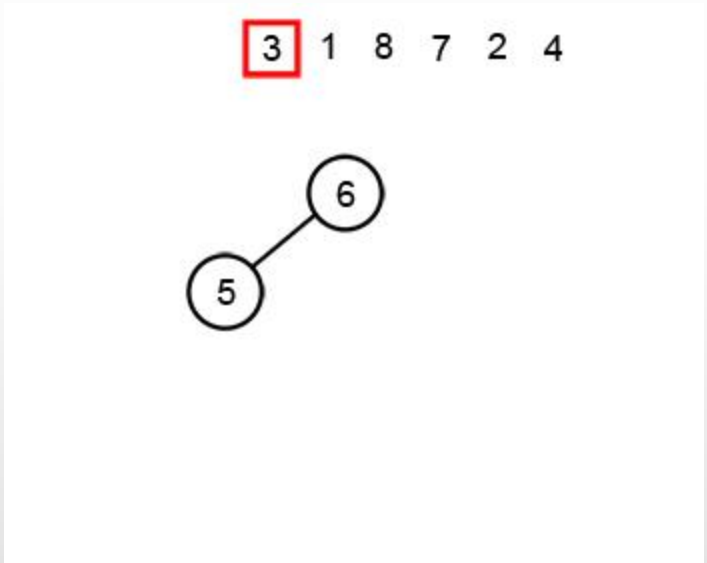
Heap Sort



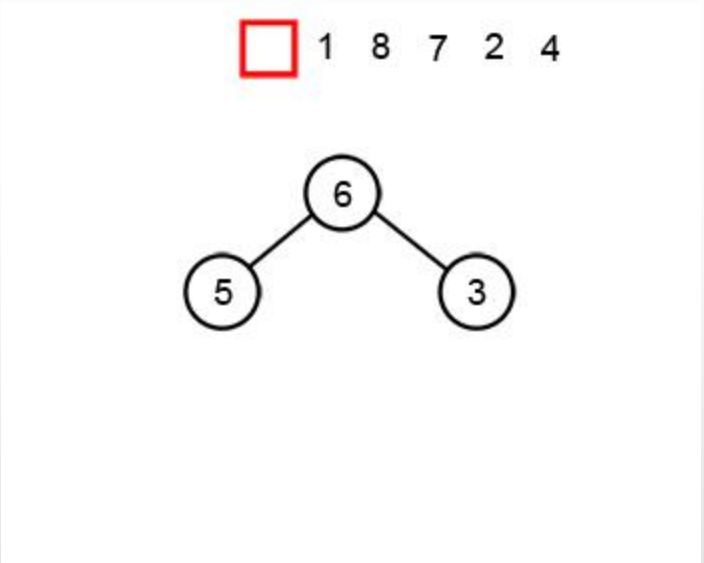
Heap Sort



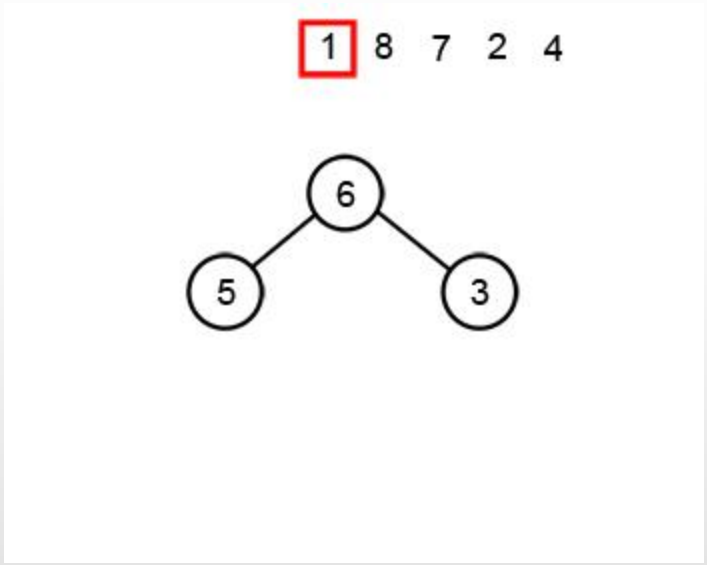
Heap Sort



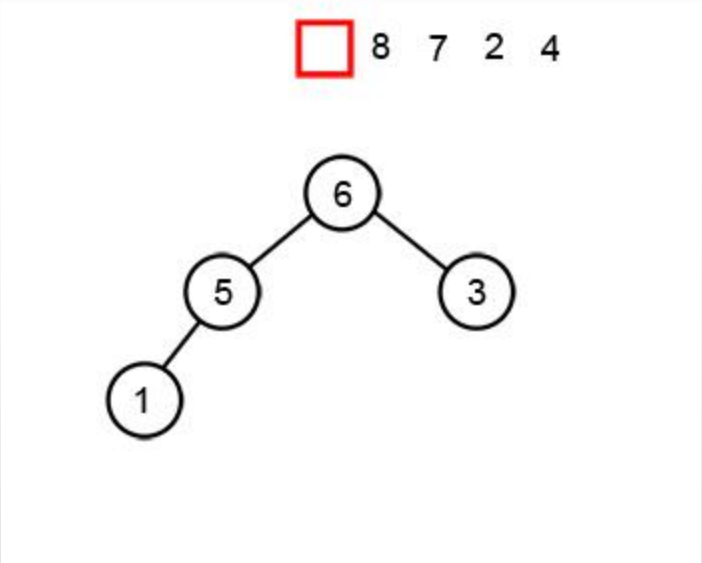
Heap Sort



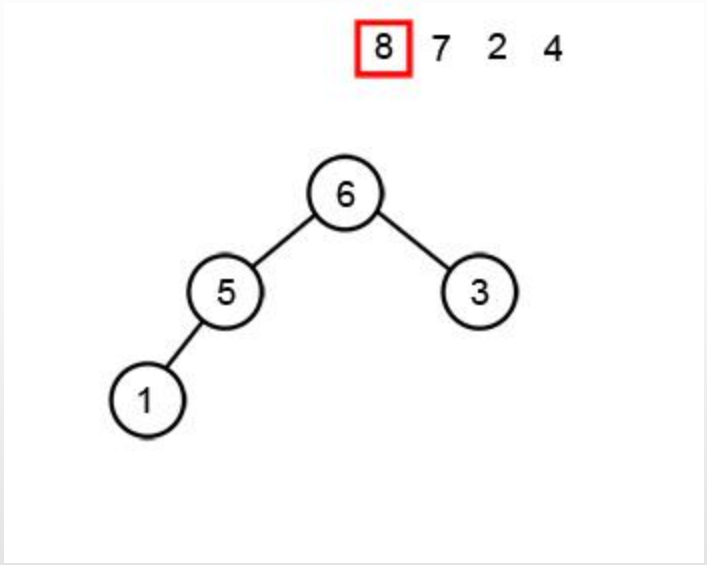
Heap Sort



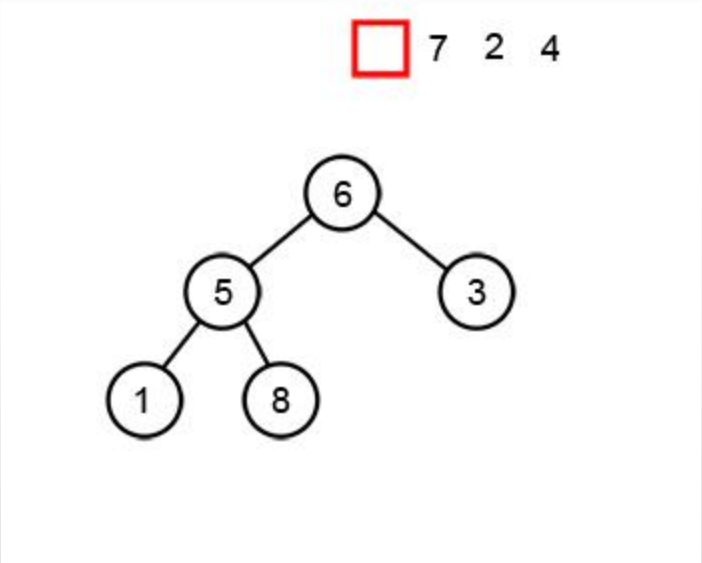
Heap Sort



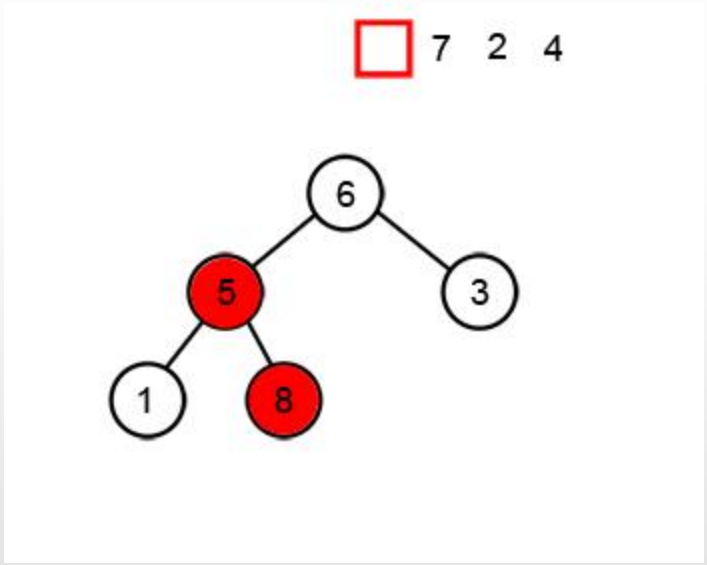
Heap Sort



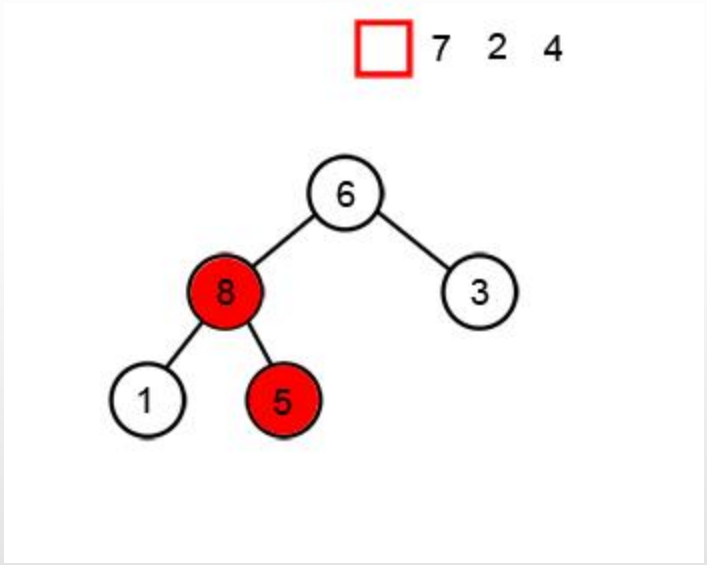
Heap Sort



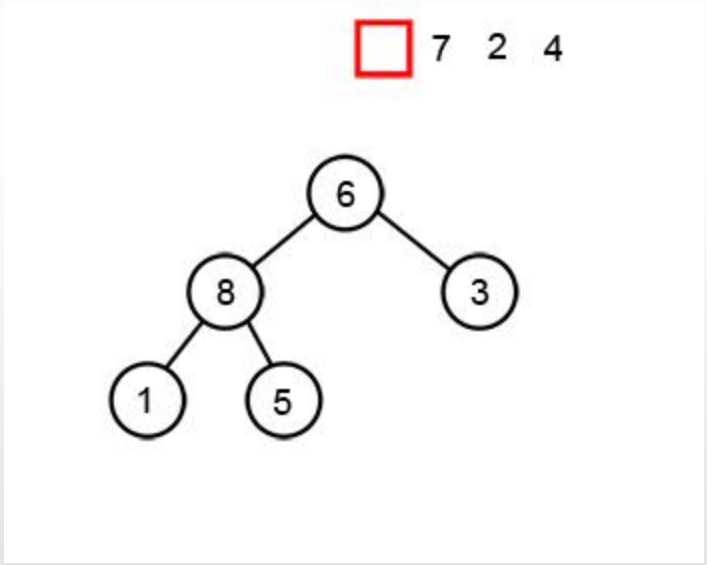
Heap Sort



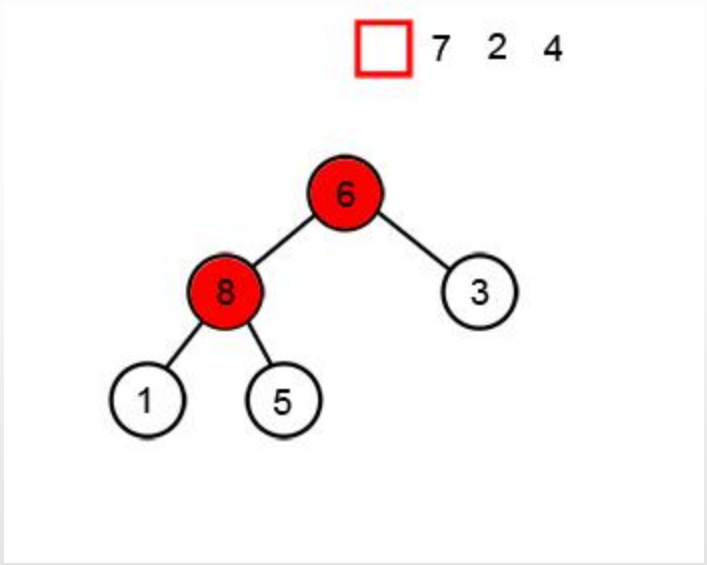
Heap Sort



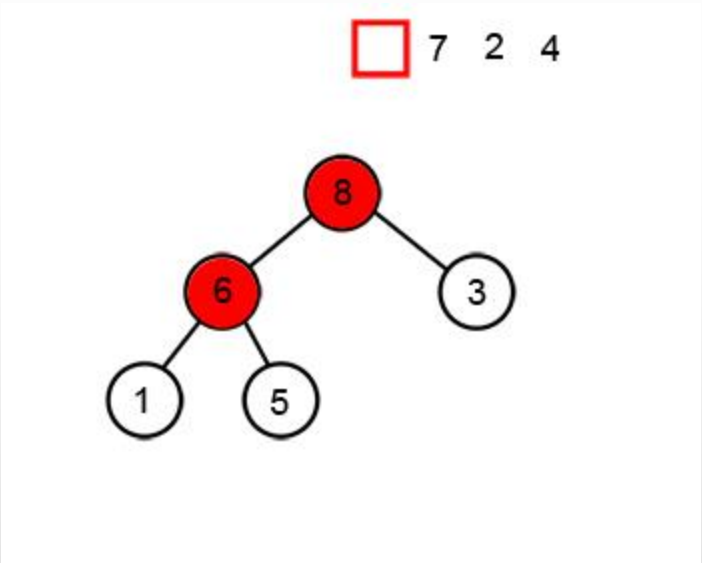
Heap Sort



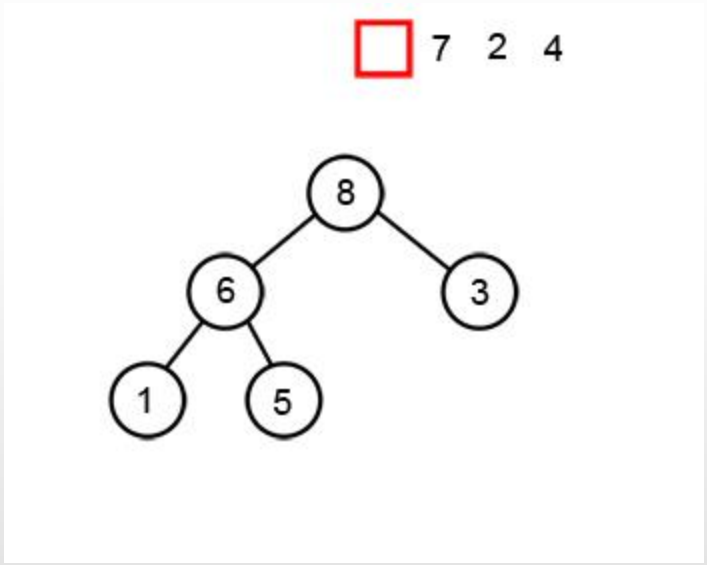
Heap Sort



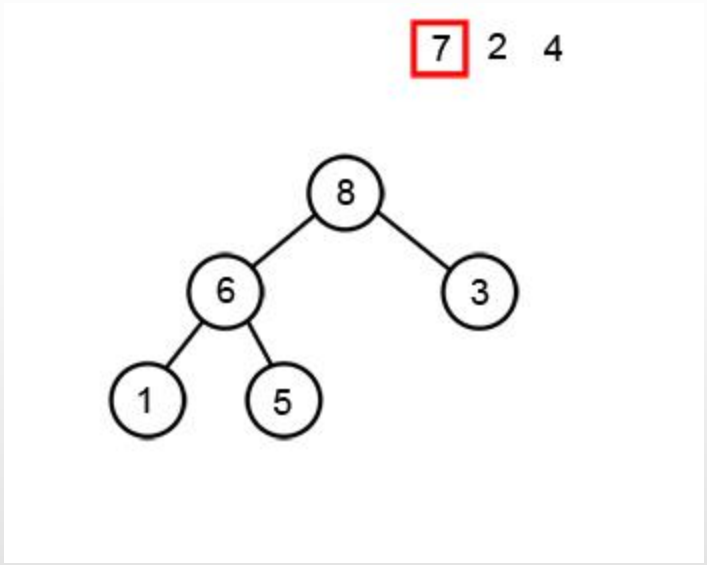
Heap Sort



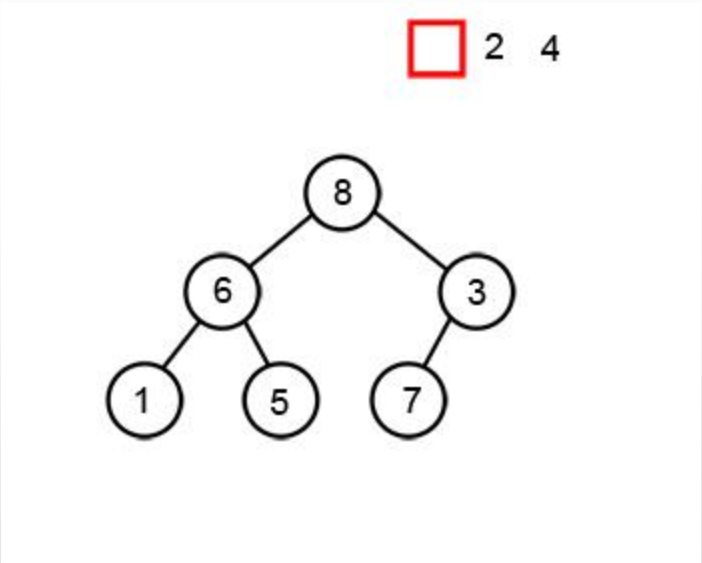
Heap Sort



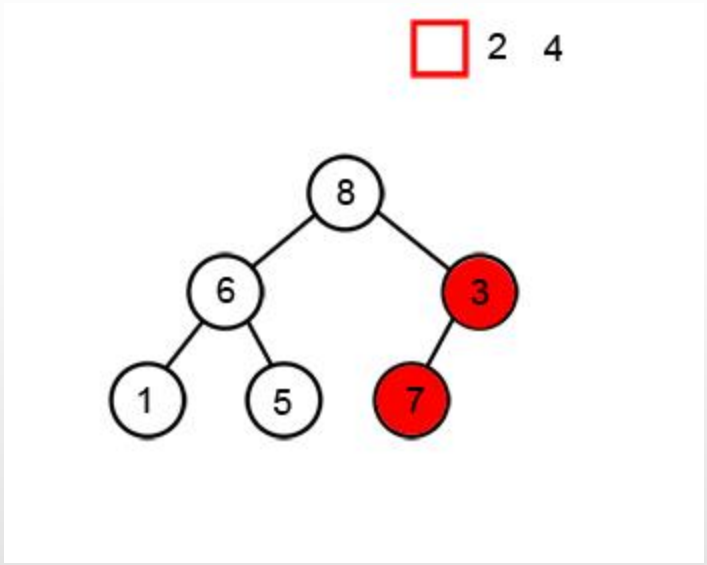
Heap Sort



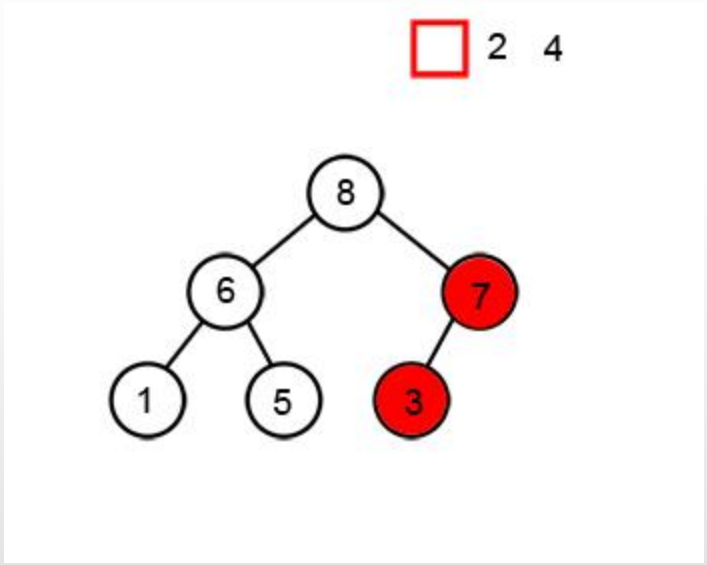
Heap Sort



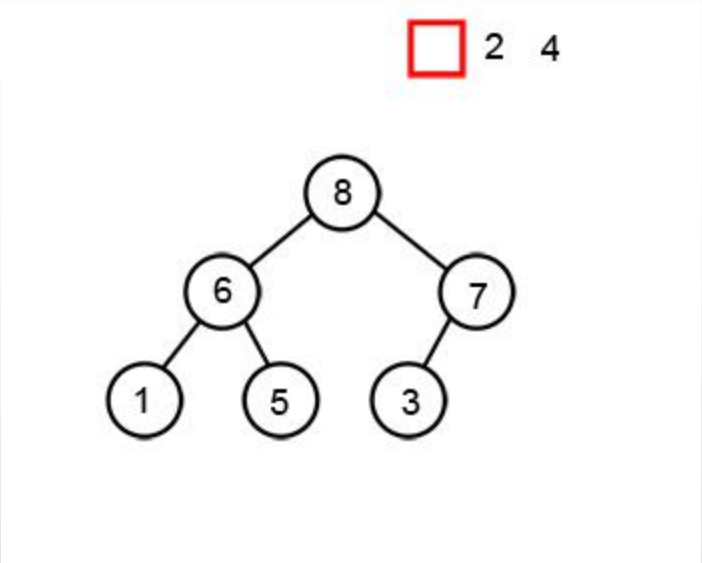
Heap Sort



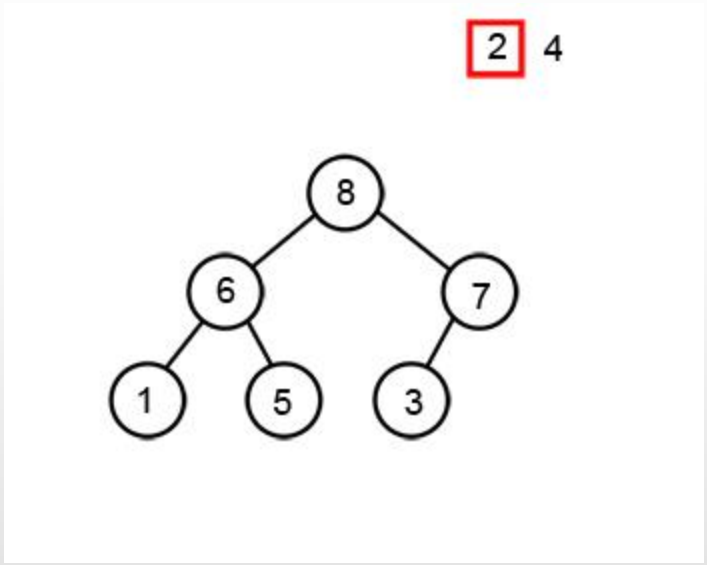
Heap Sort



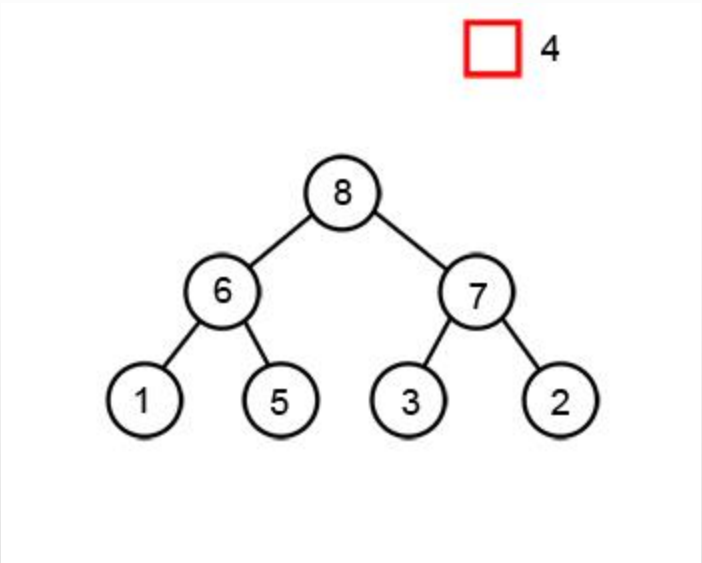
Heap Sort



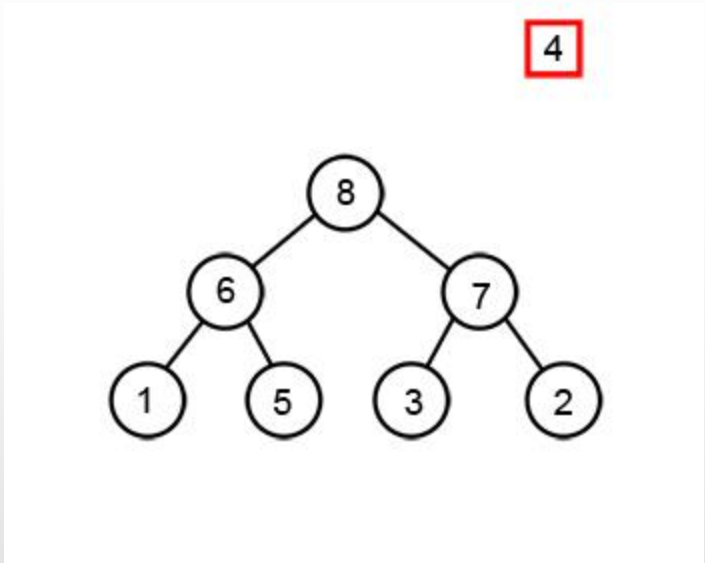
Heap Sort



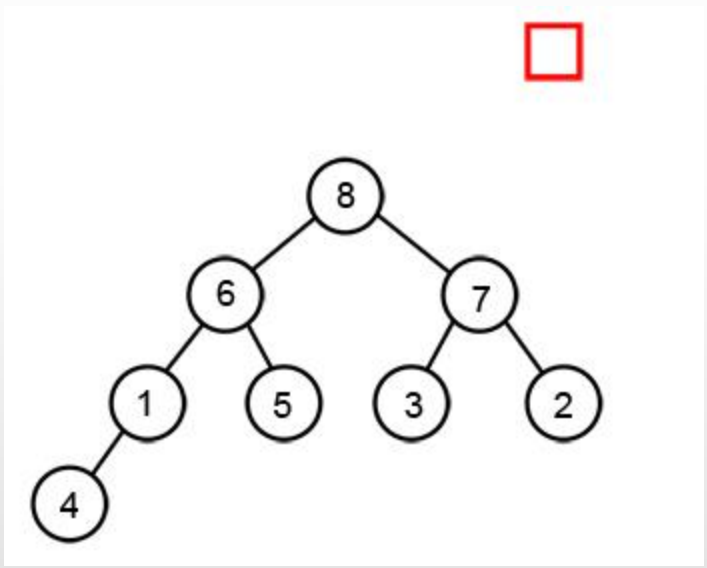
Heap Sort



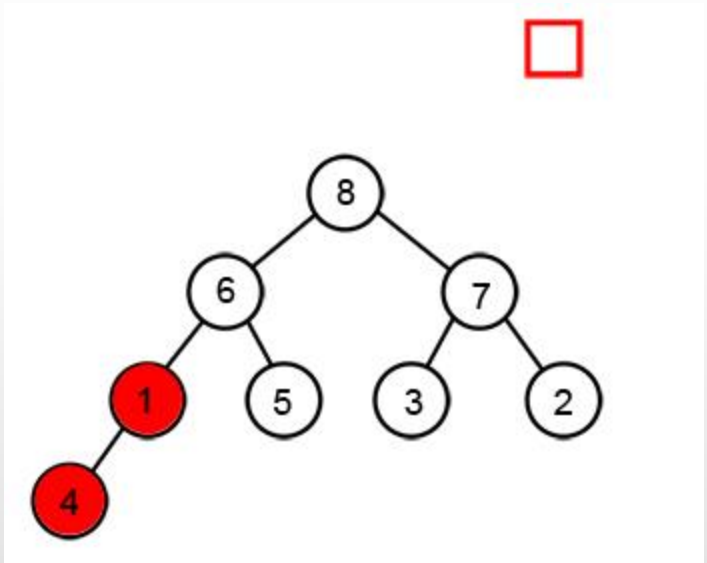
Heap Sort



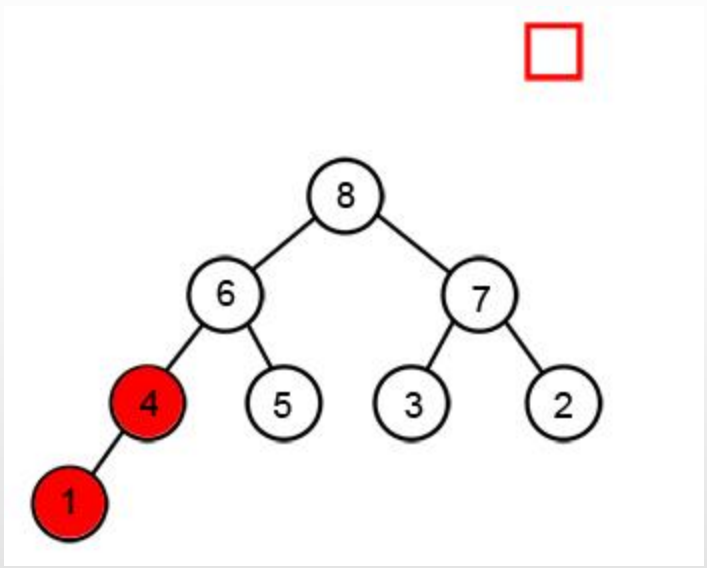
Heap Sort



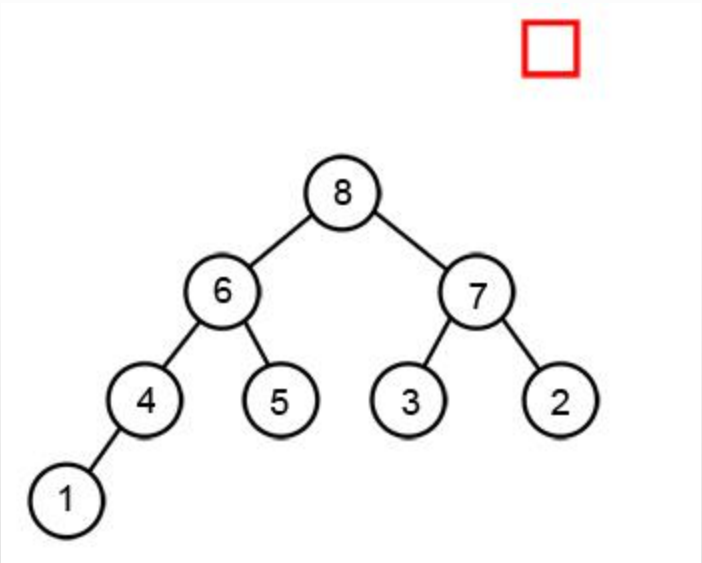
Heap Sort



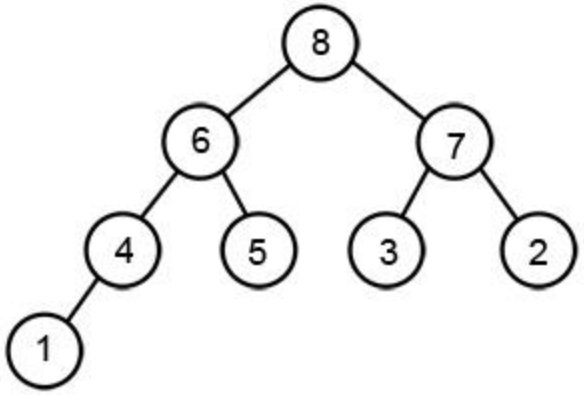
Heap Sort



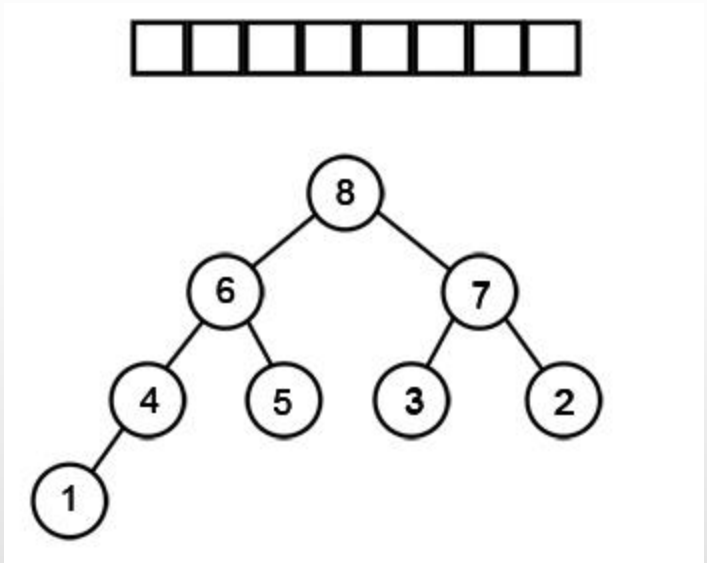
Heap Sort



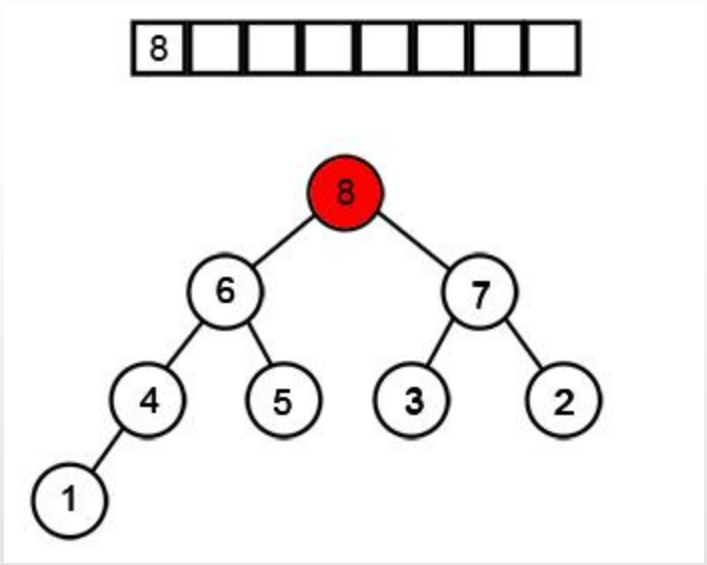
Heap Sort



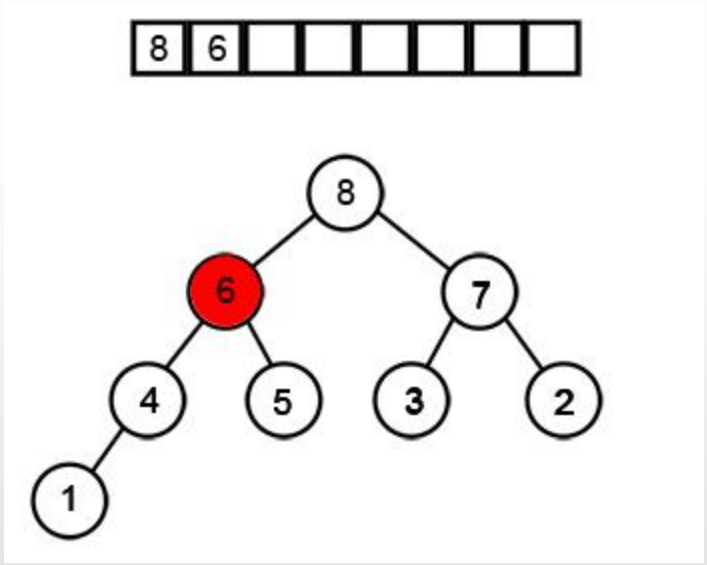
Heap Sort



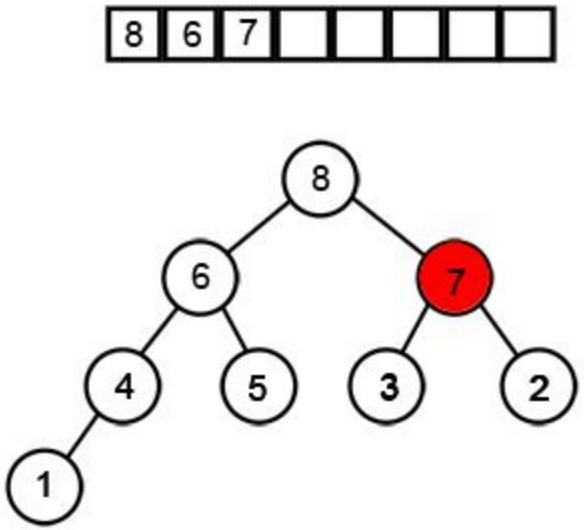
Heap Sort



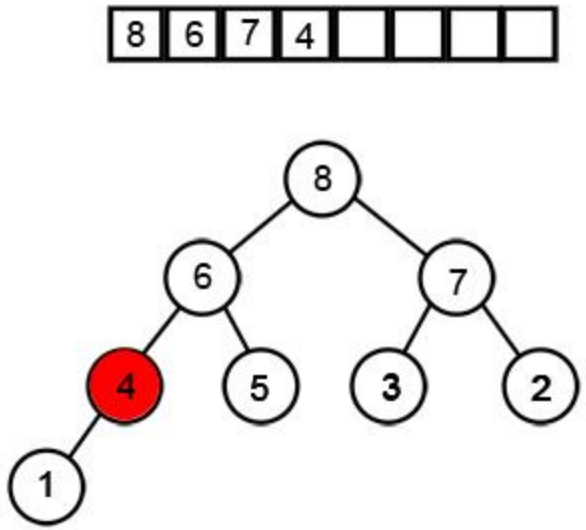
Heap Sort



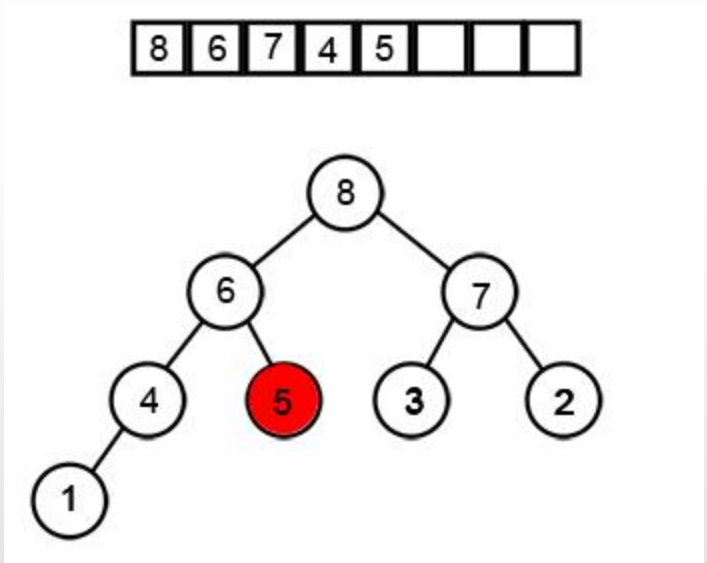
Heap Sort



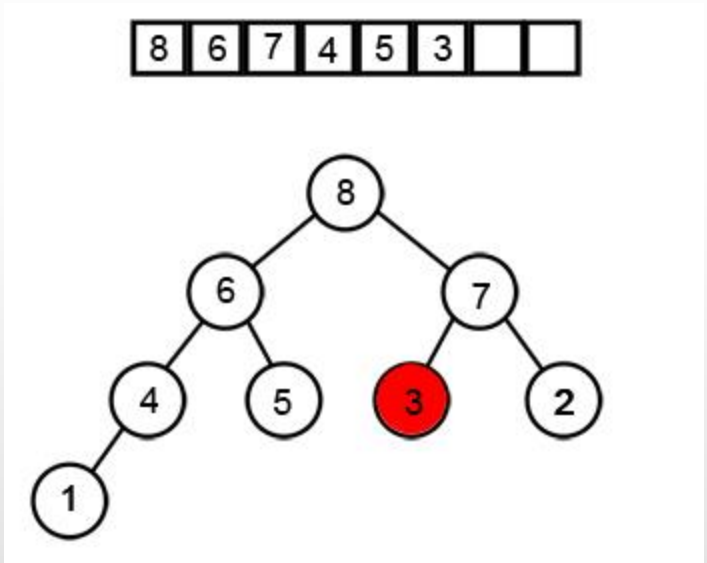
Heap Sort



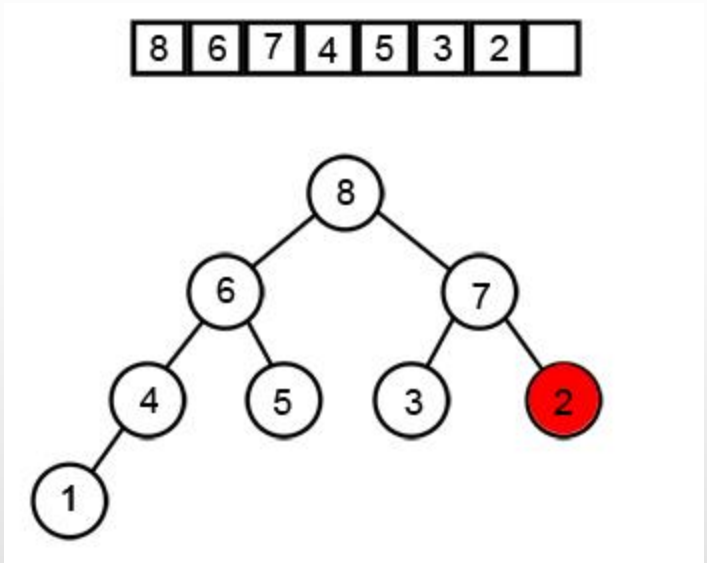
Heap Sort



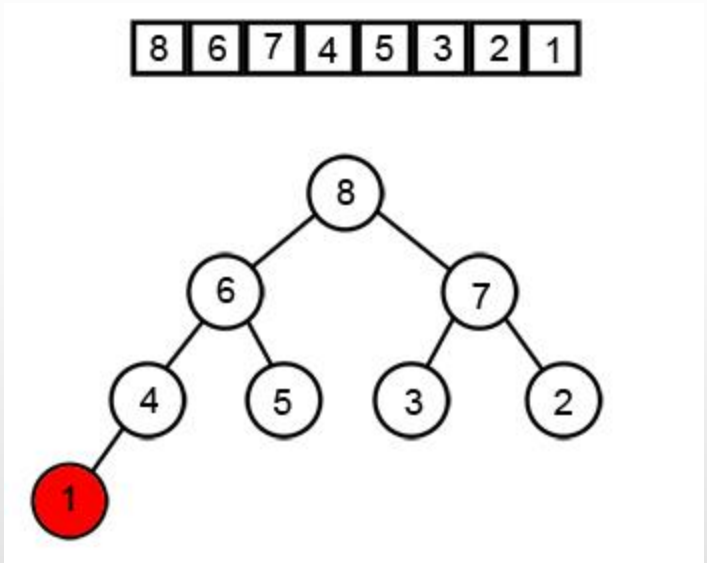
Heap Sort



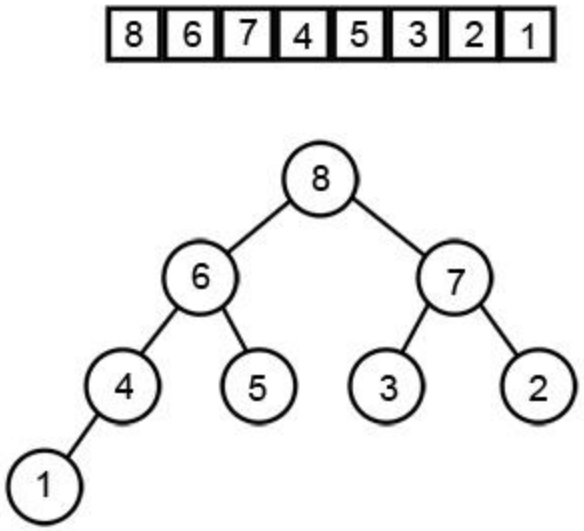
Heap Sort



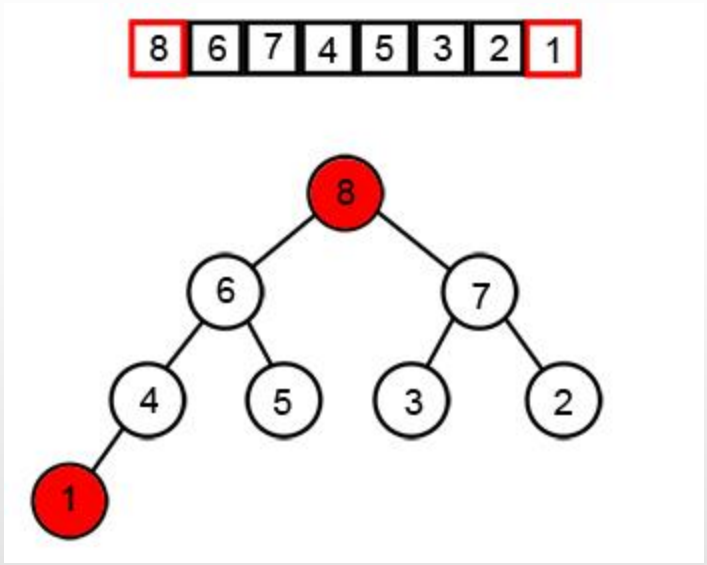
Heap Sort



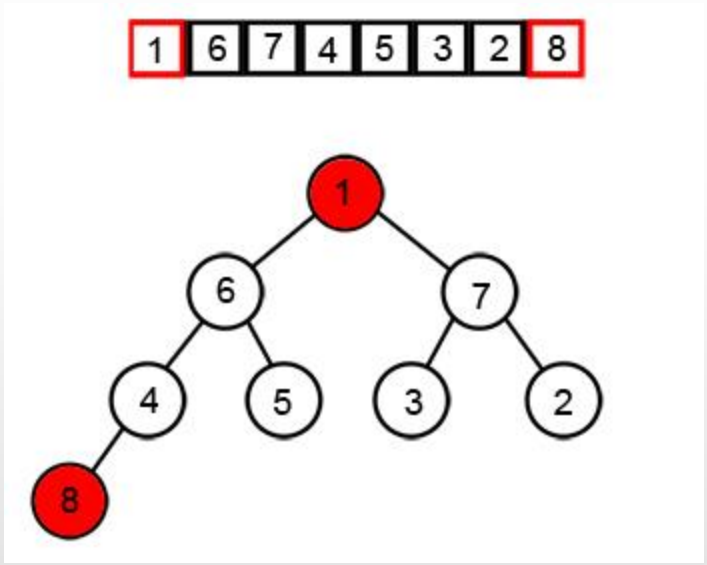
Heap Sort



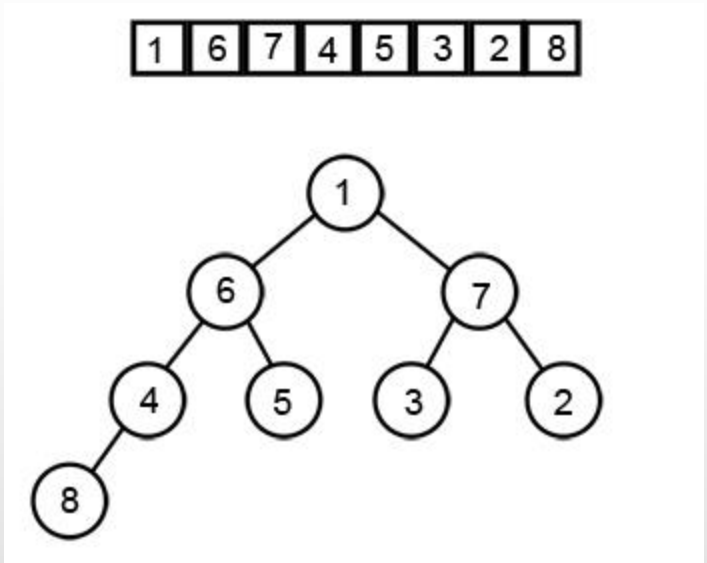
Heap Sort



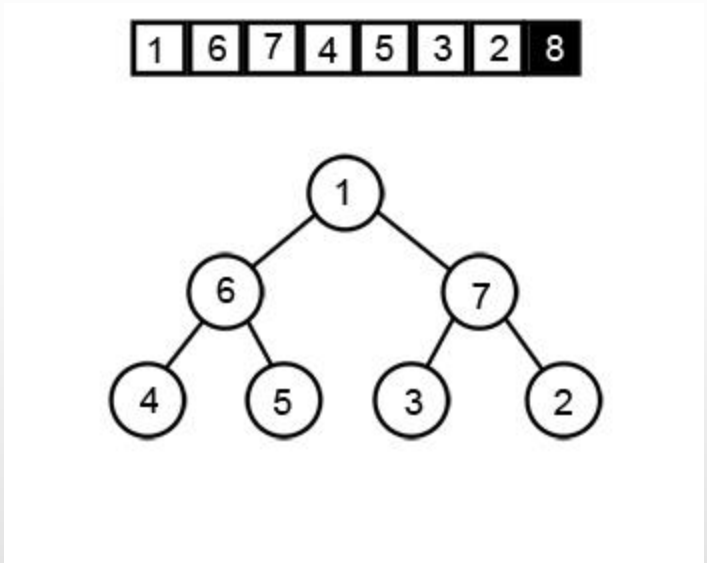
Heap Sort



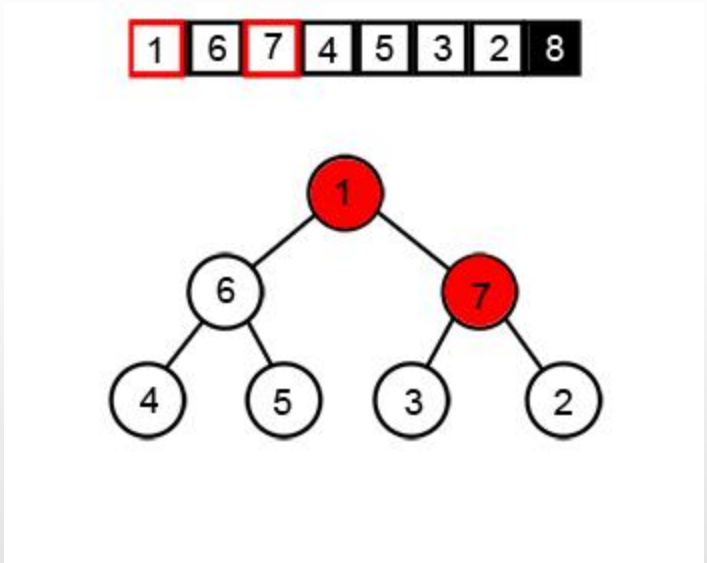
Heap Sort



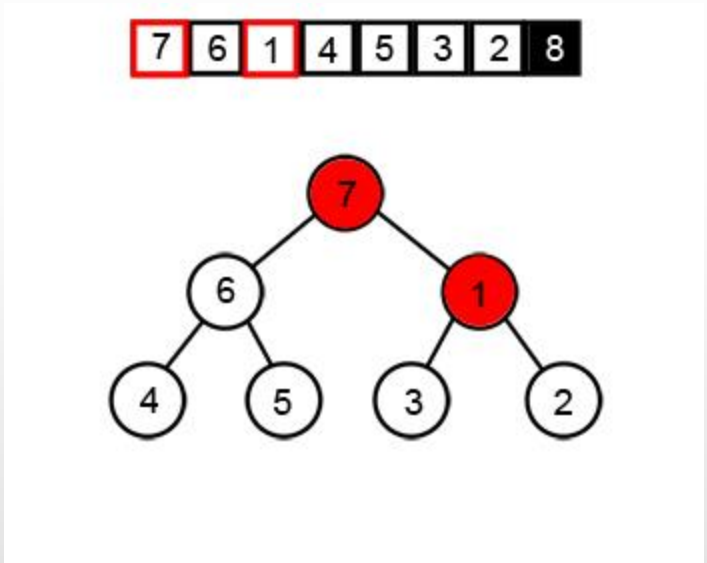
Heap Sort



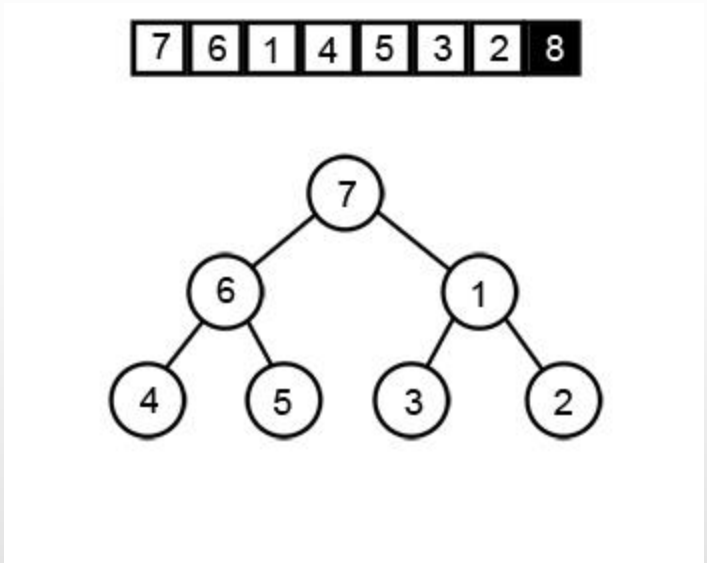
Heap Sort



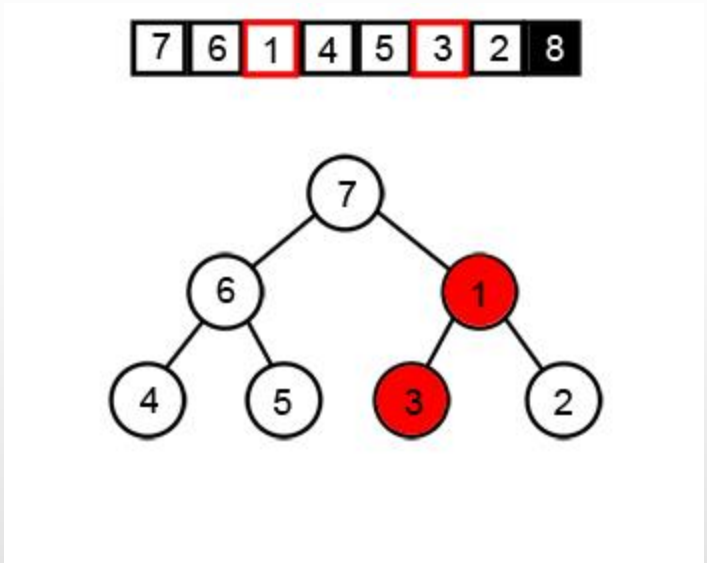
Heap Sort



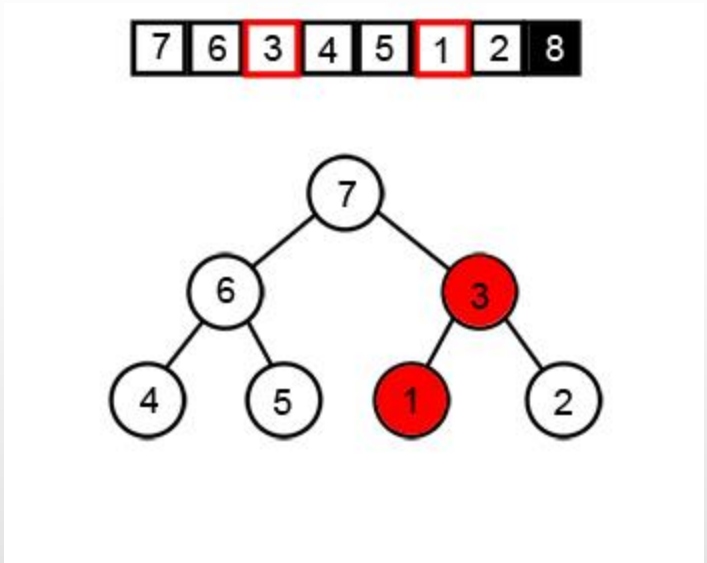
Heap Sort



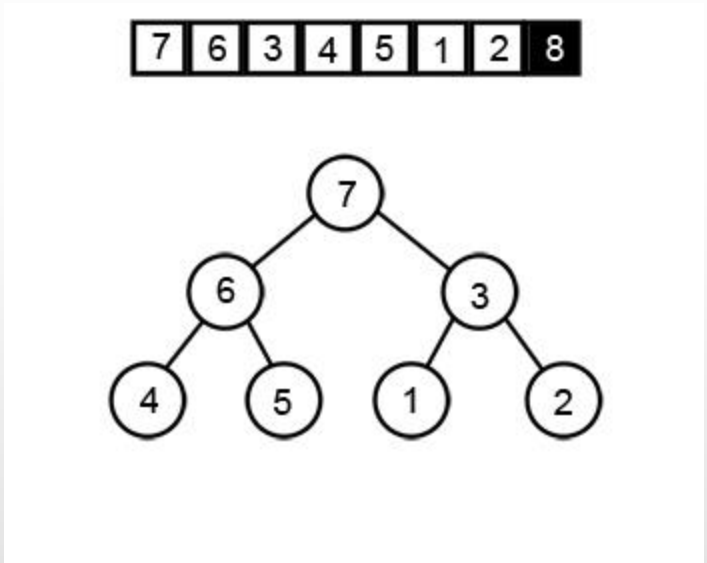
Heap Sort



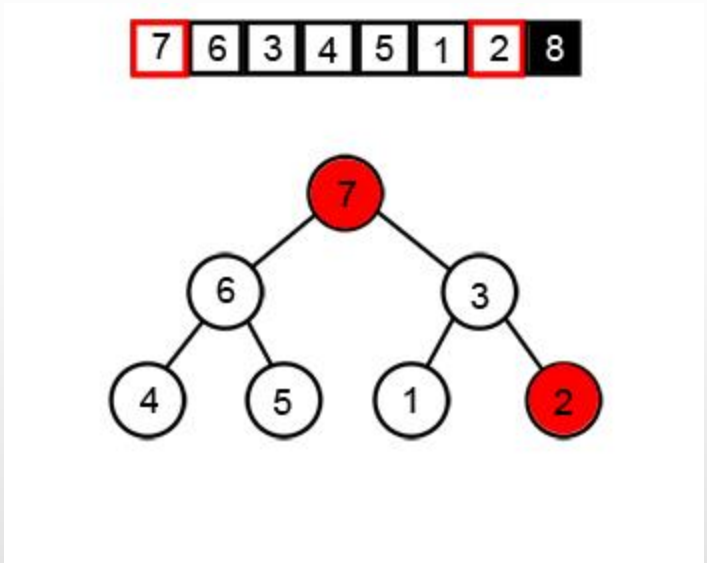
Heap Sort



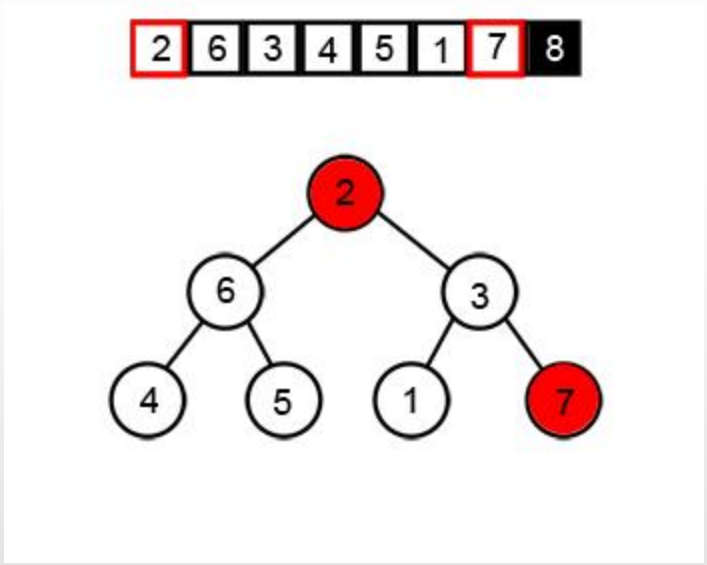
Heap Sort



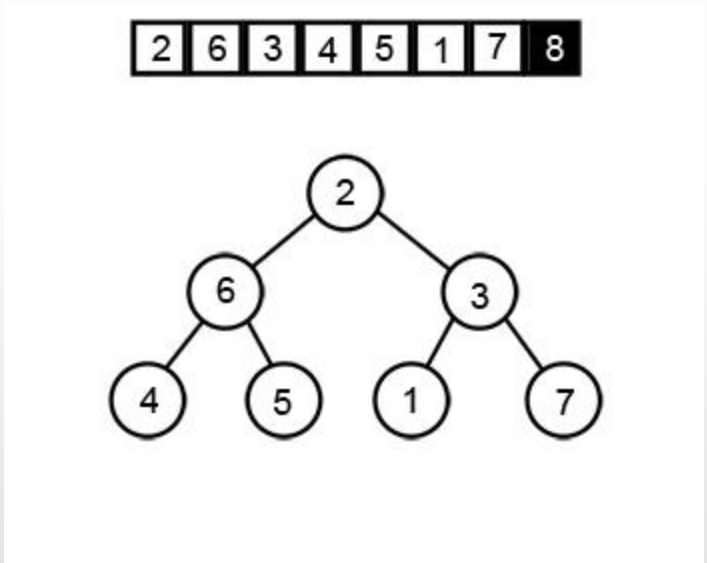
Heap Sort



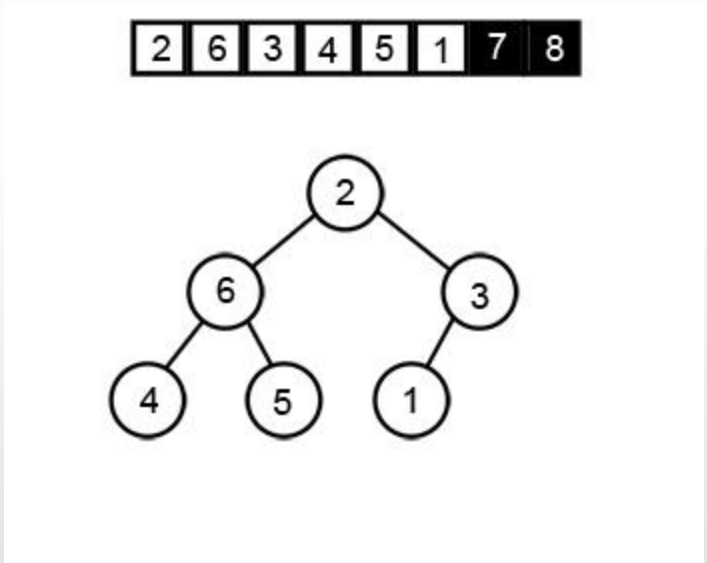
Heap Sort



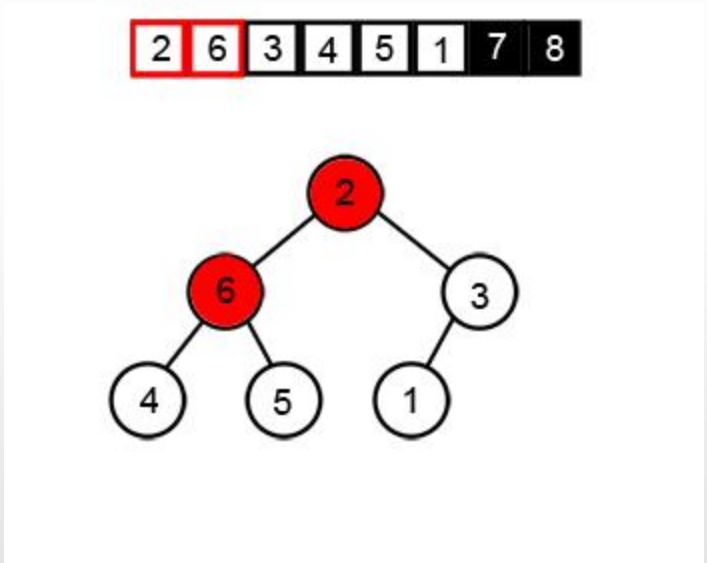
Heap Sort



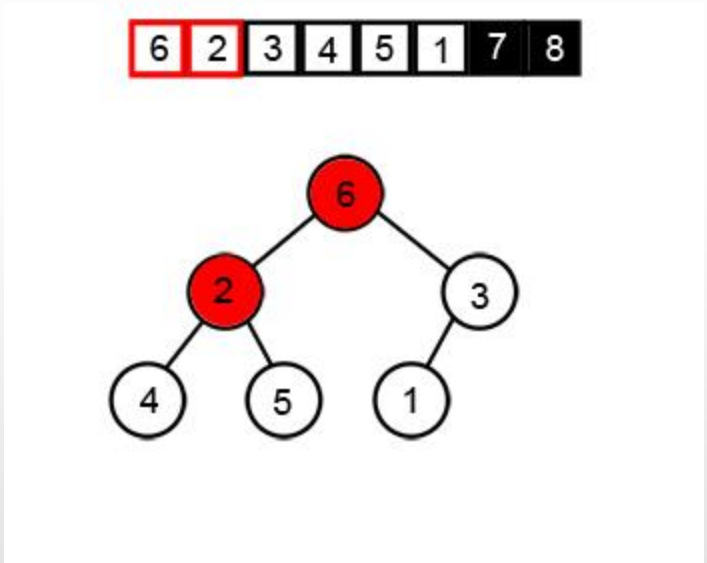
Heap Sort



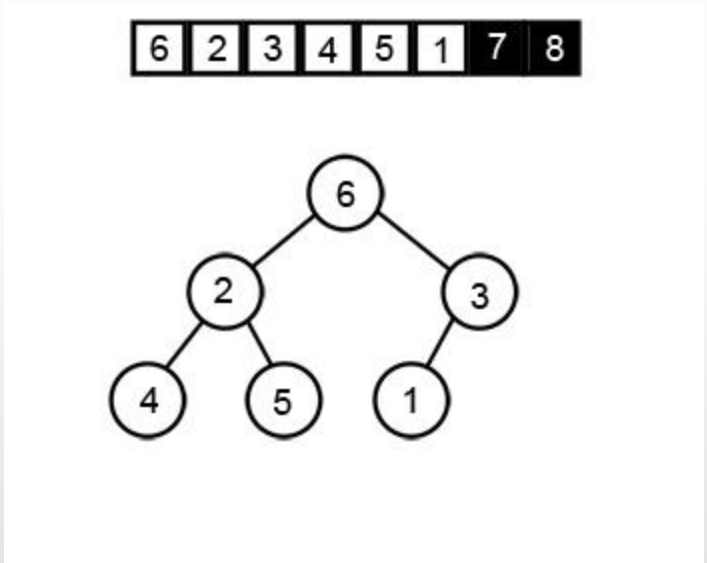
Heap Sort



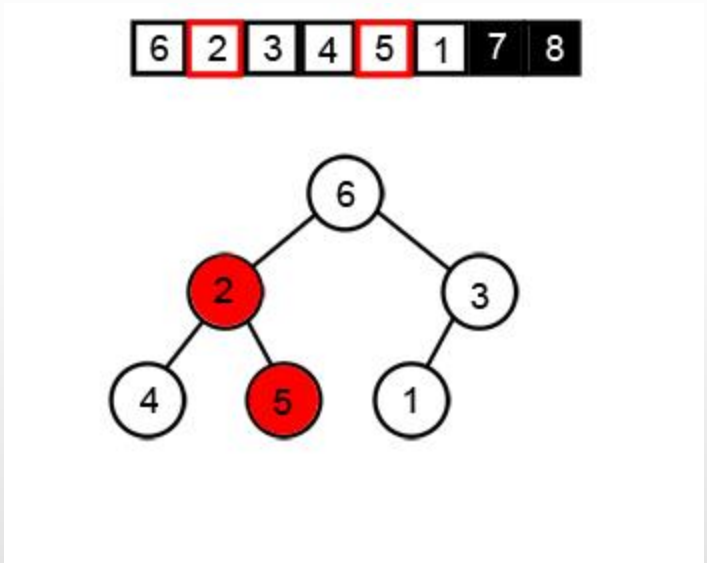
Heap Sort



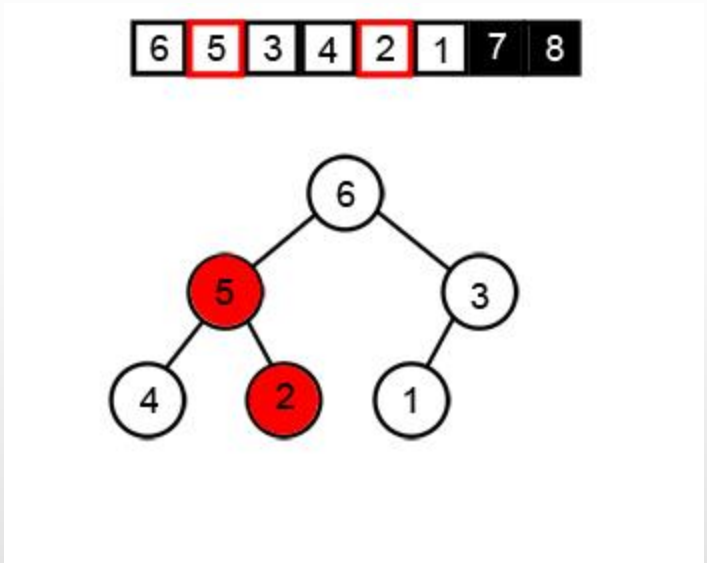
Heap Sort



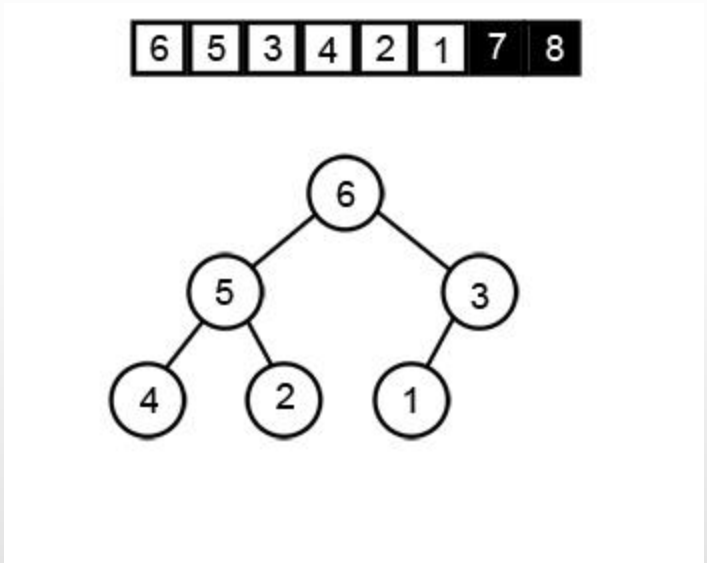
Heap Sort



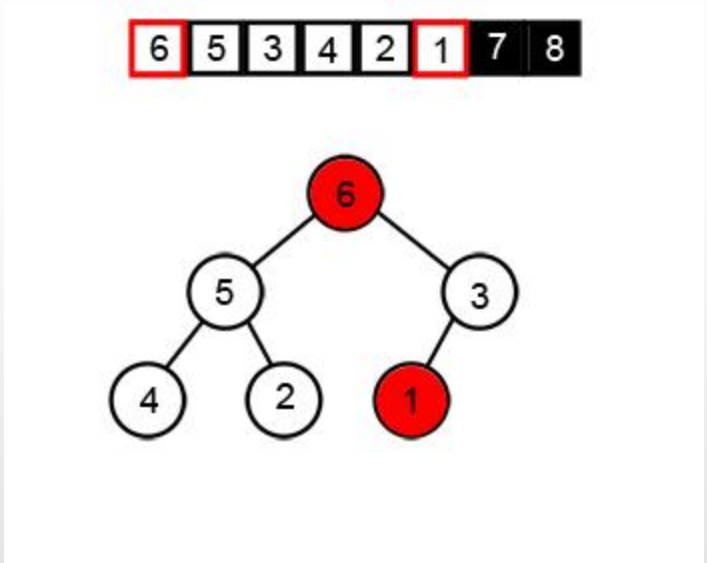
Heap Sort



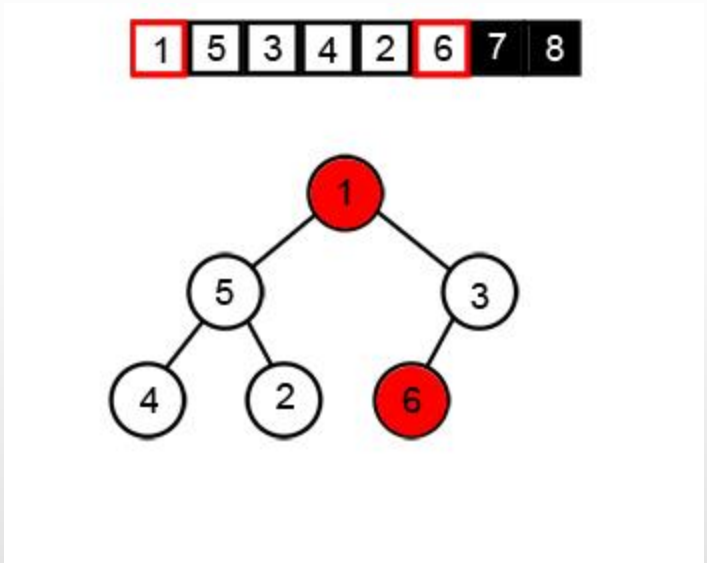
Heap Sort



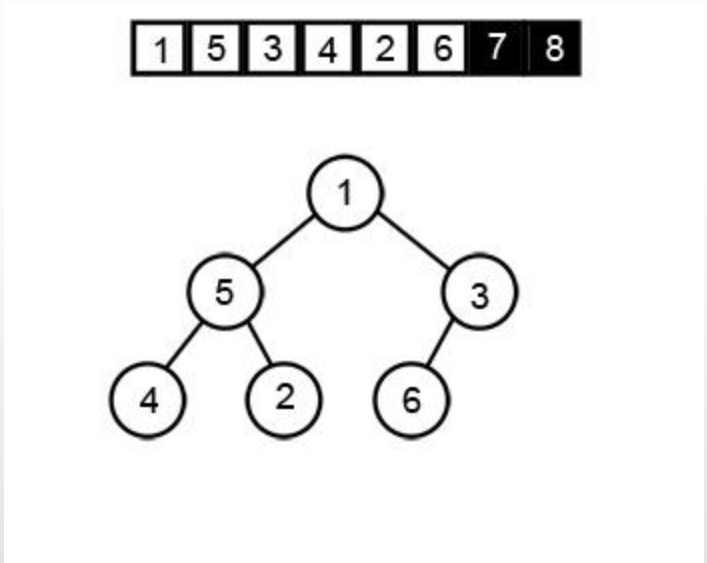
Heap Sort



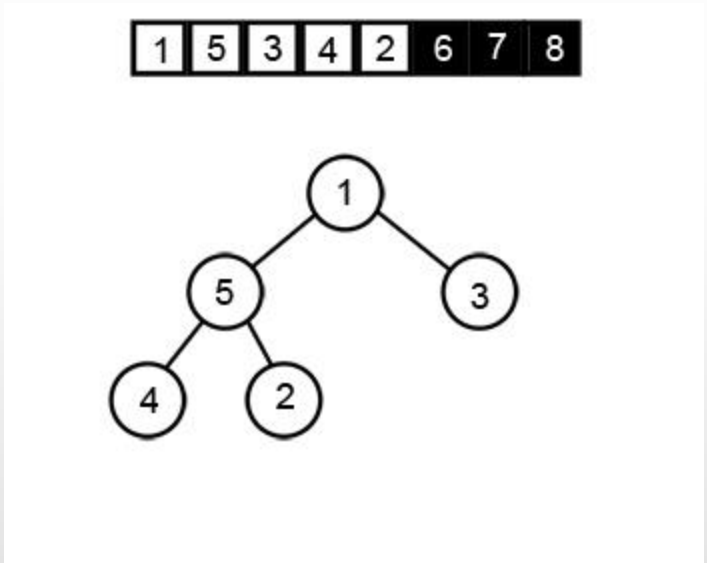
Heap Sort



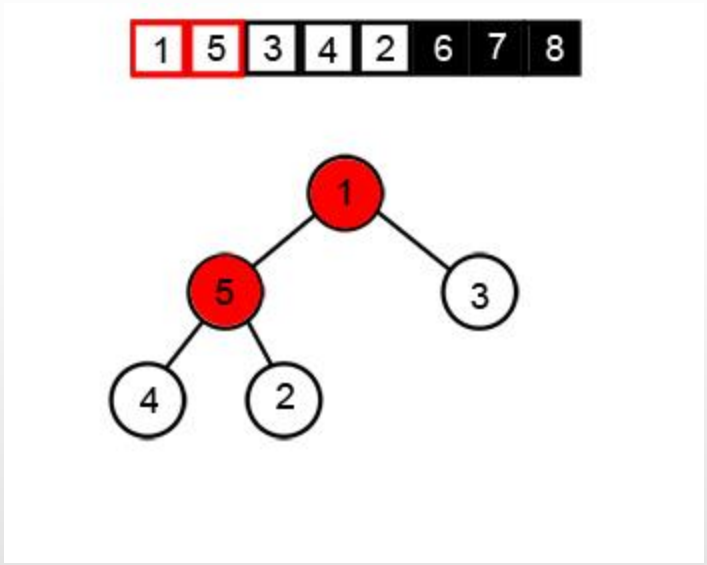
Heap Sort



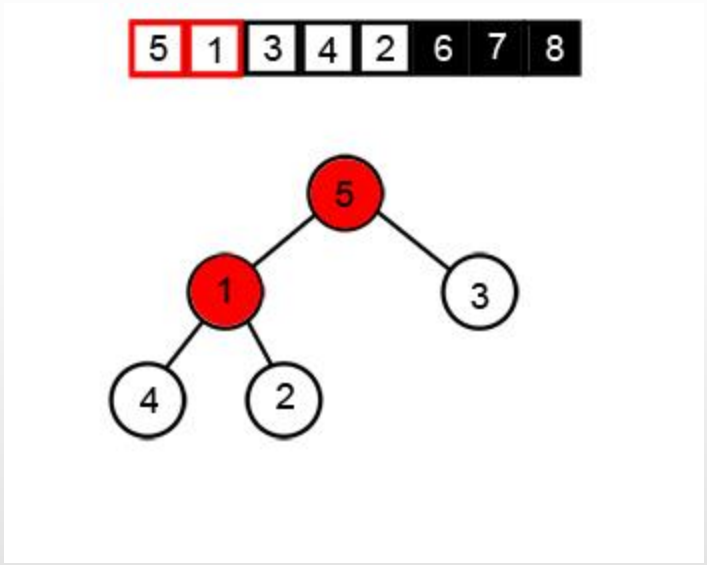
Heap Sort



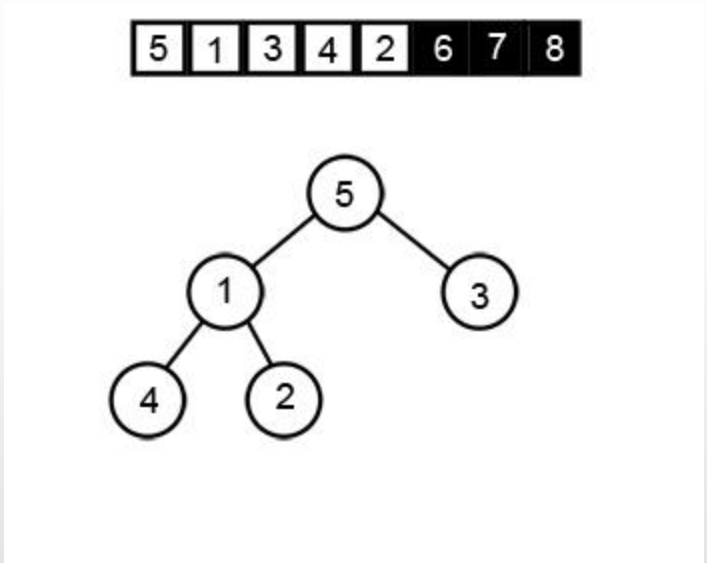
Heap Sort



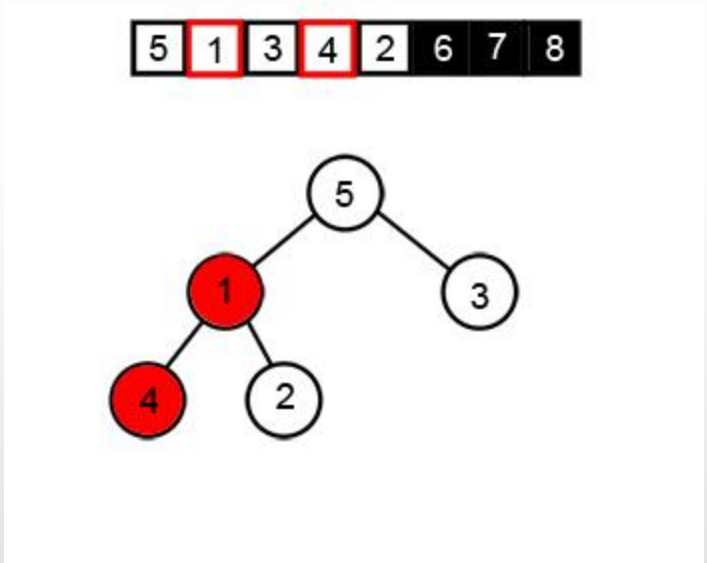
Heap Sort



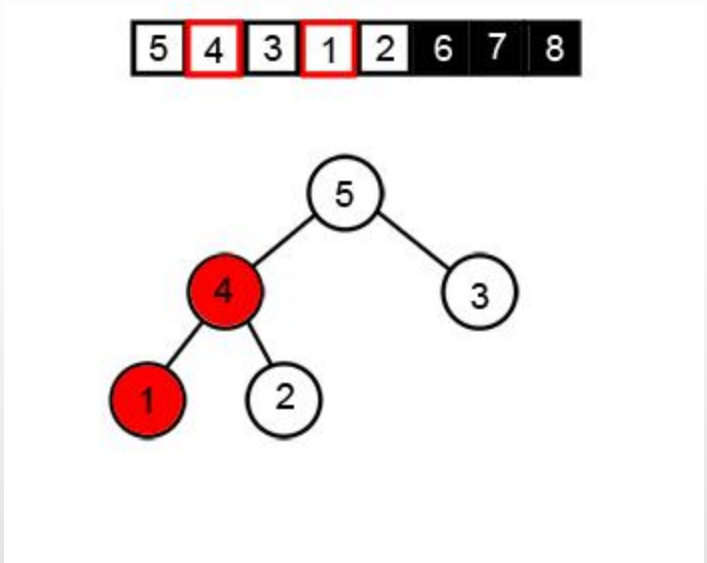
Heap Sort



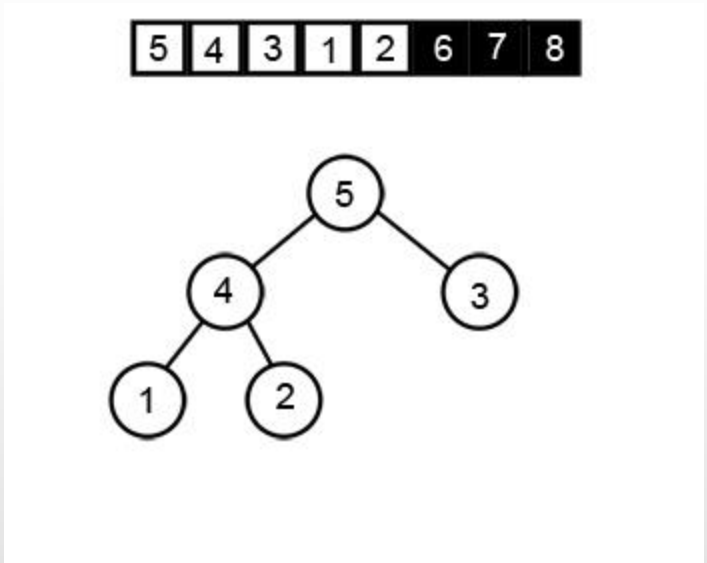
Heap Sort



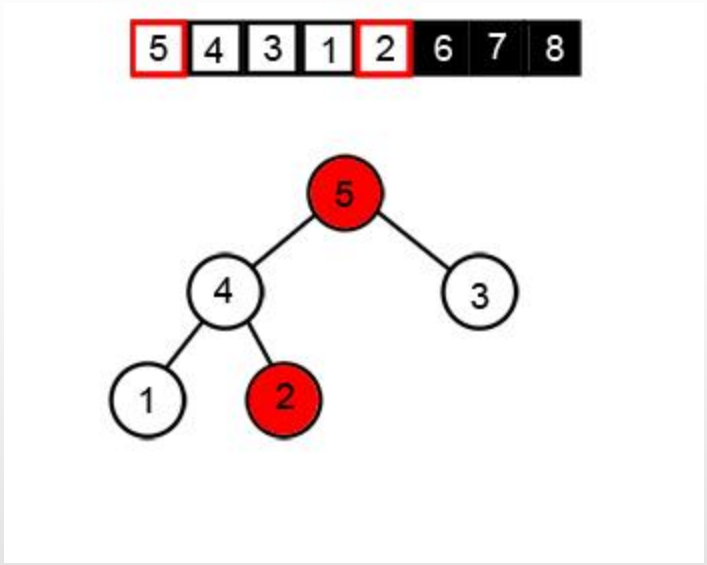
Heap Sort



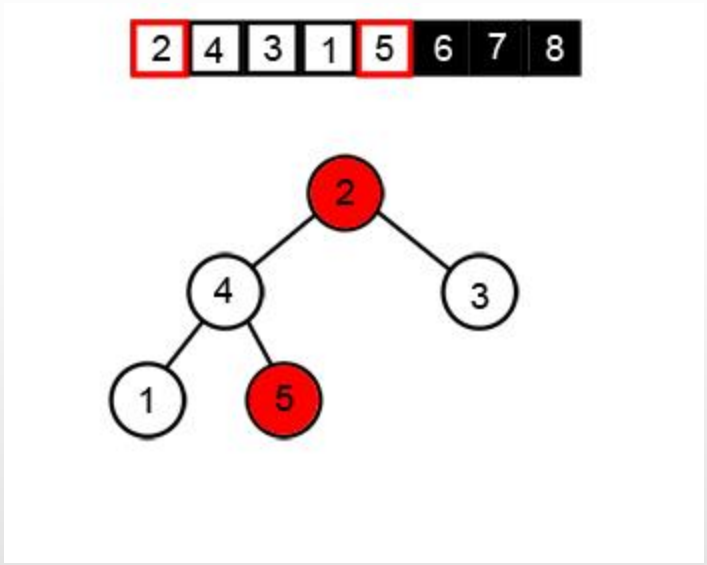
Heap Sort



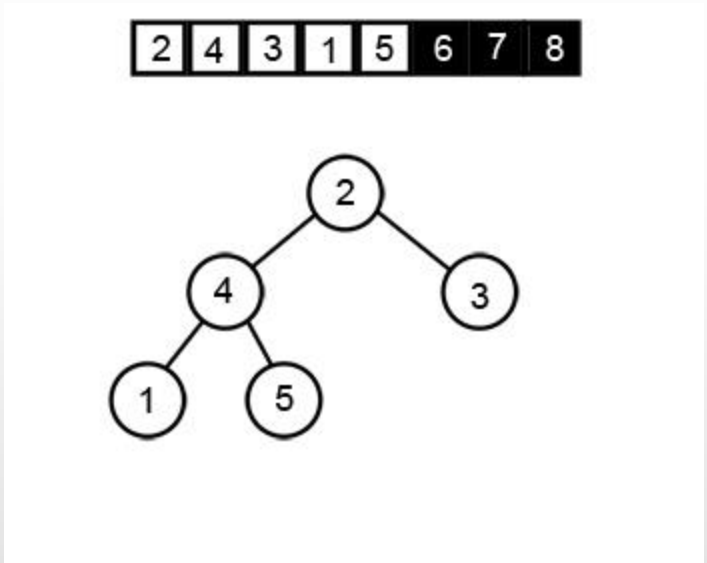
Heap Sort



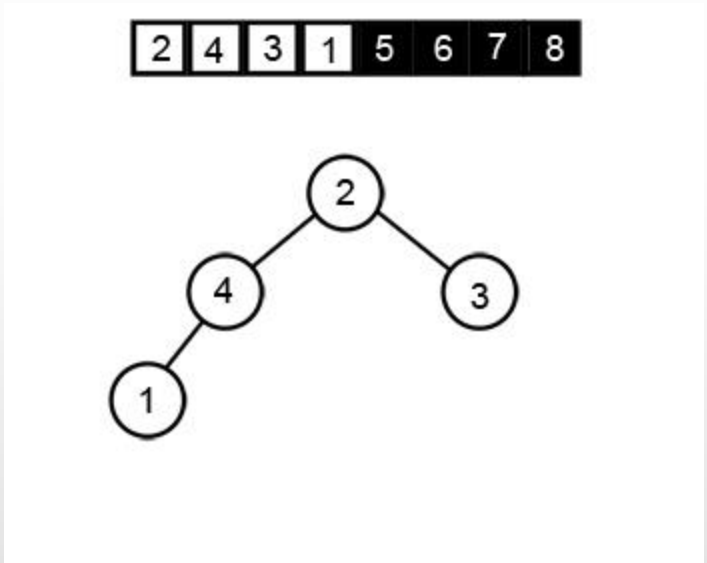
Heap Sort



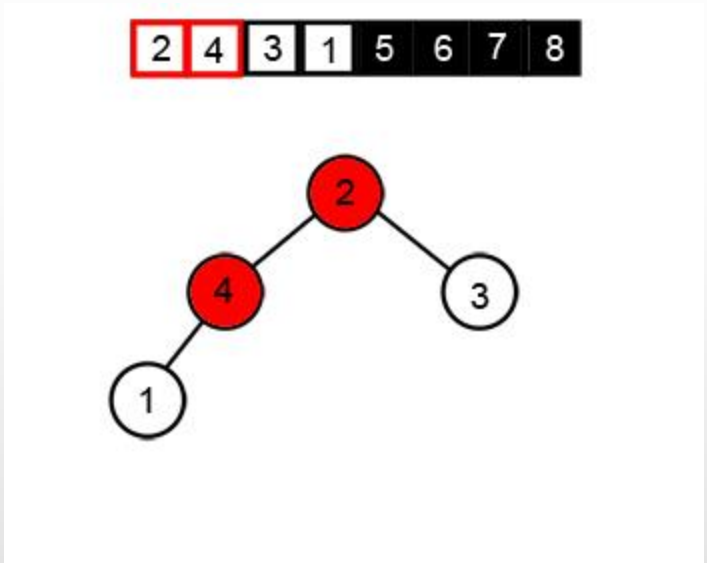
Heap Sort



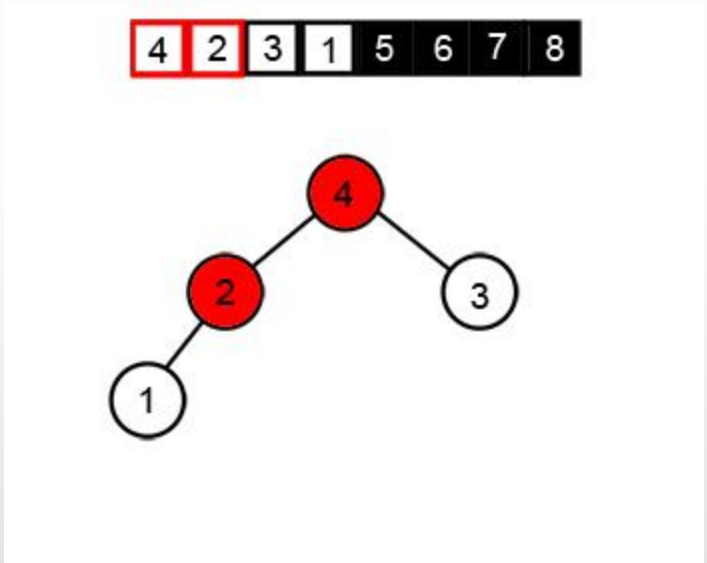
Heap Sort



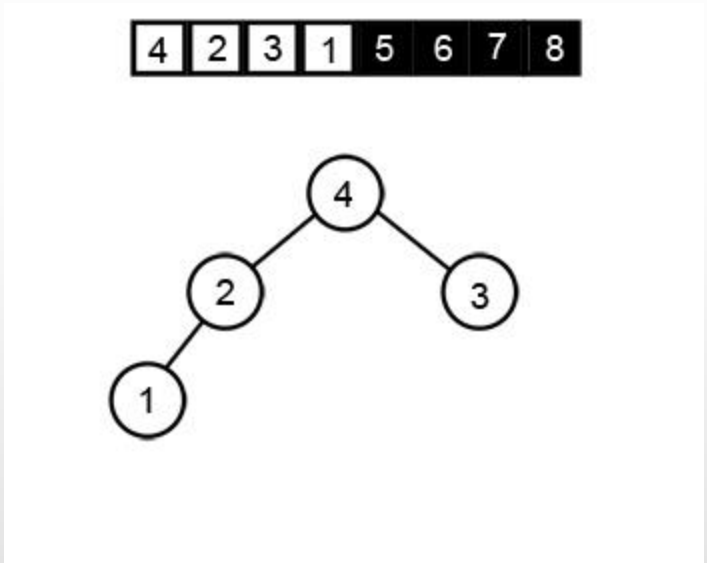
Heap Sort



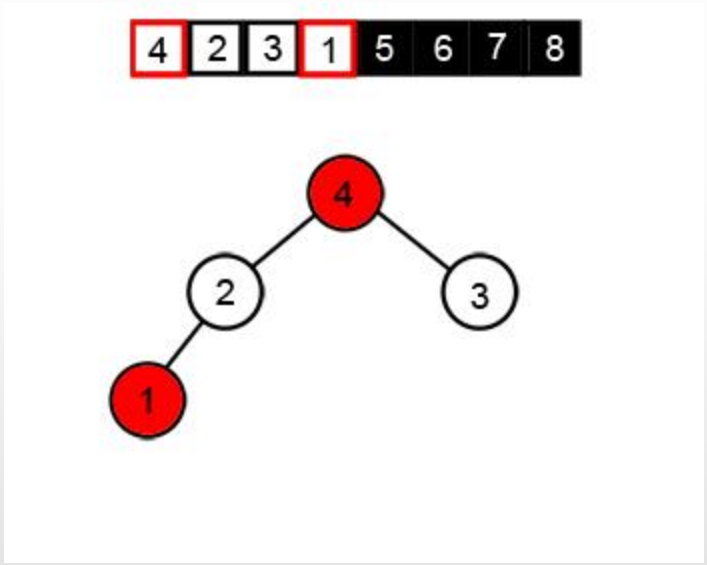
Heap Sort



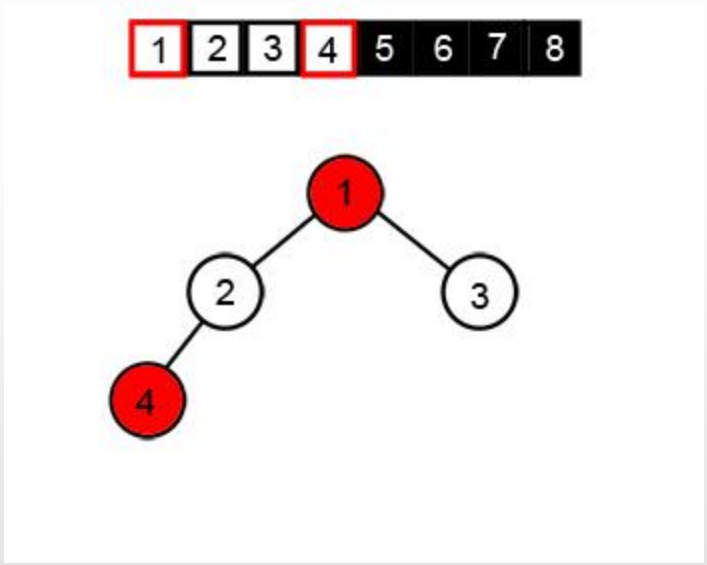
Heap Sort



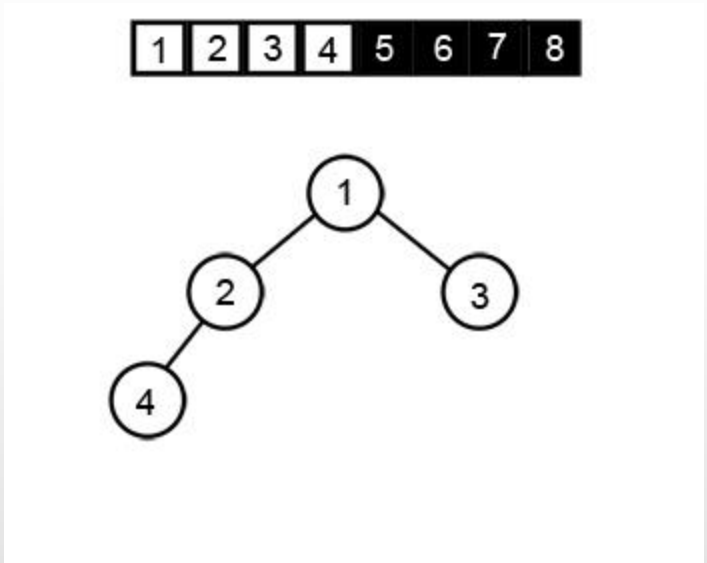
Heap Sort



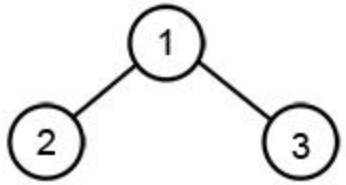
Heap Sort



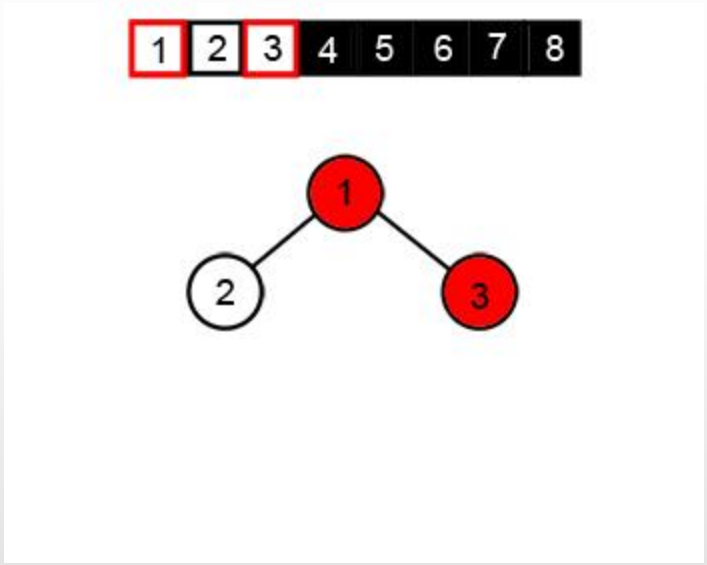
Heap Sort



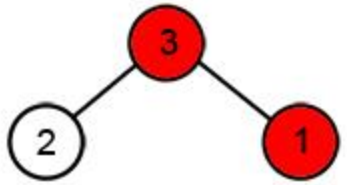
Heap Sort



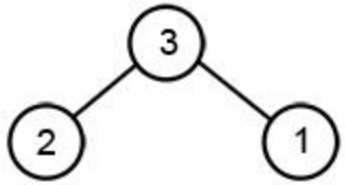
Heap Sort



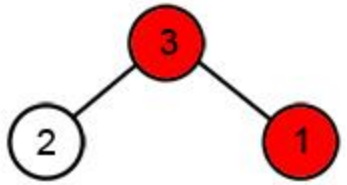
Heap Sort



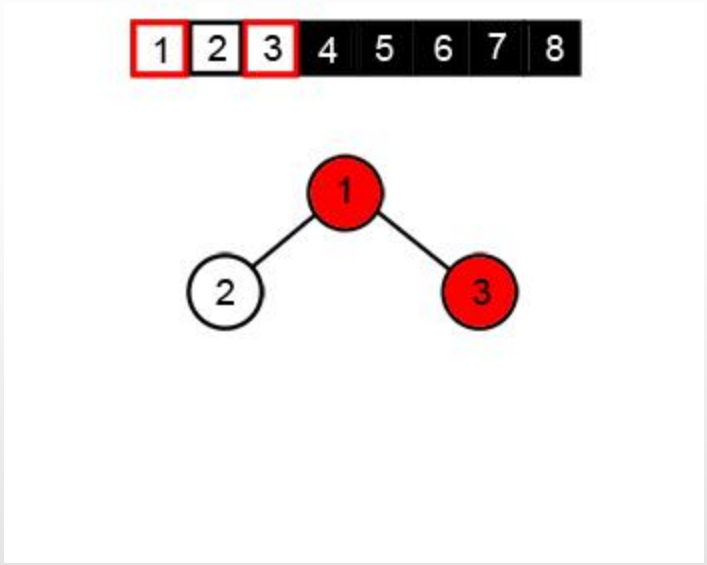
Heap Sort



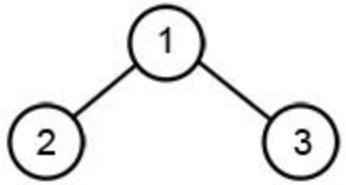
Heap Sort



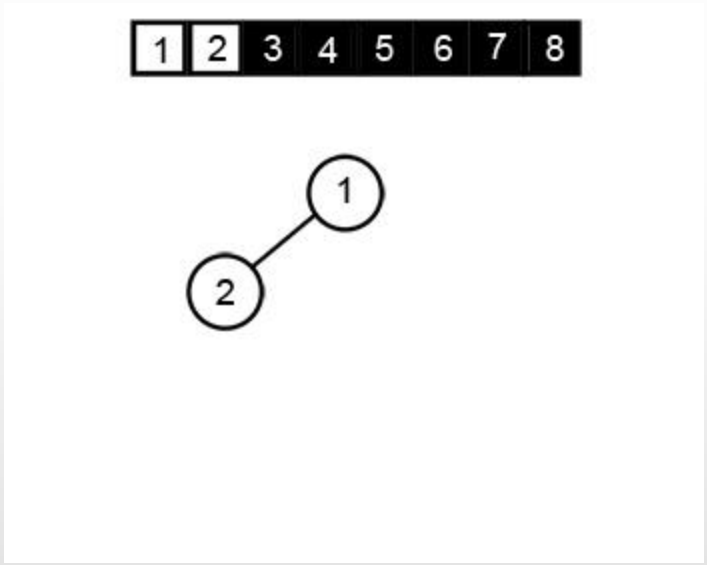
Heap Sort



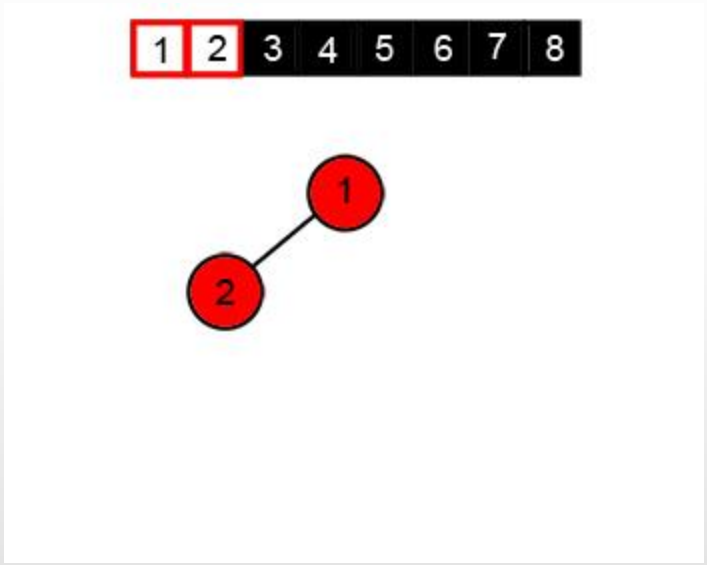
Heap Sort



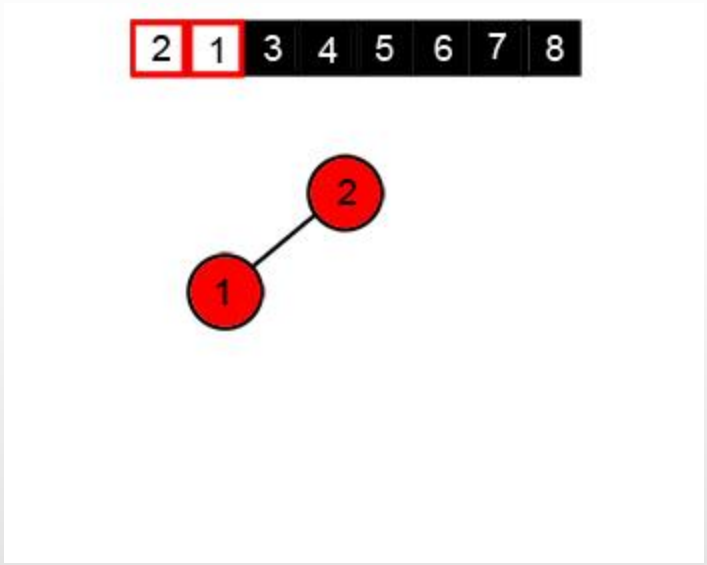
Heap Sort



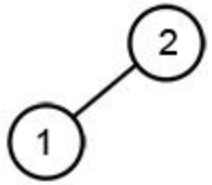
Heap Sort



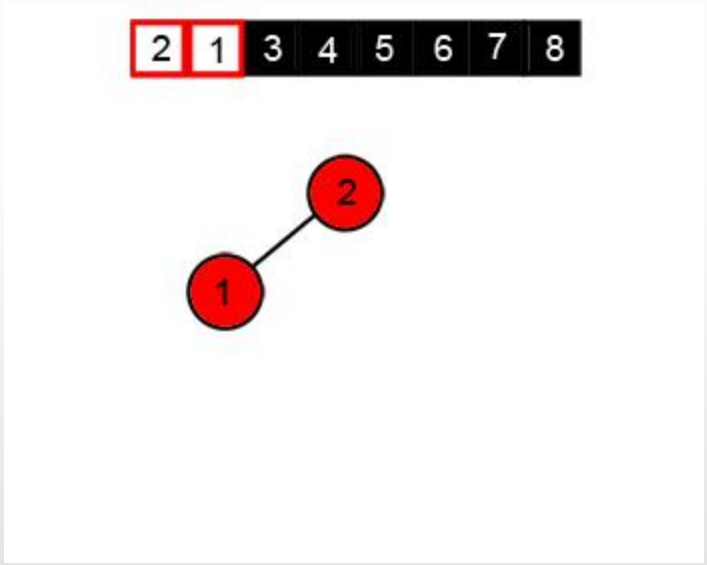
Heap Sort



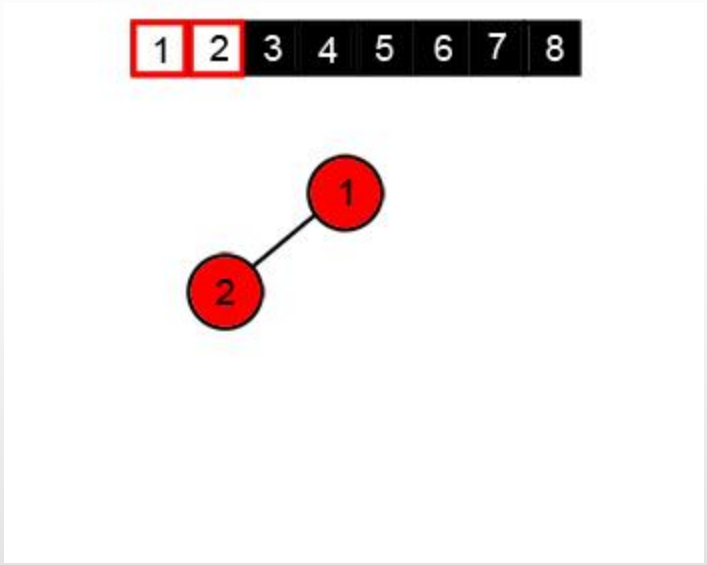
Heap Sort



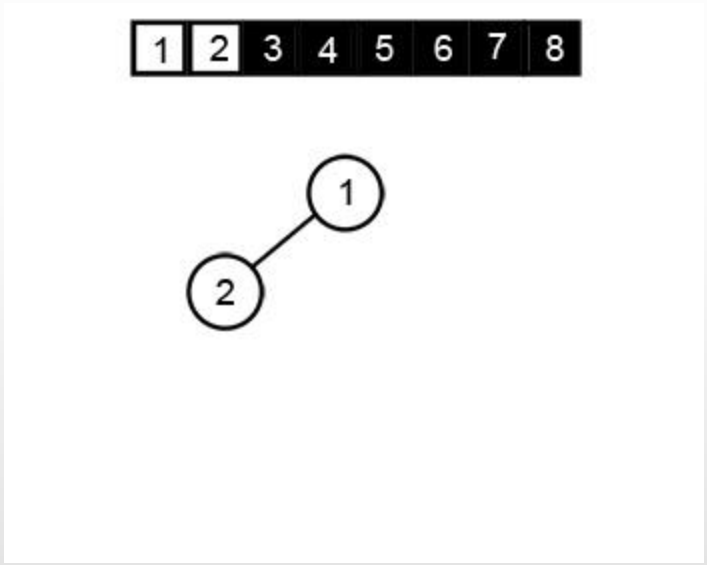
Heap Sort



Heap Sort



Heap Sort



Heap Sort

SON