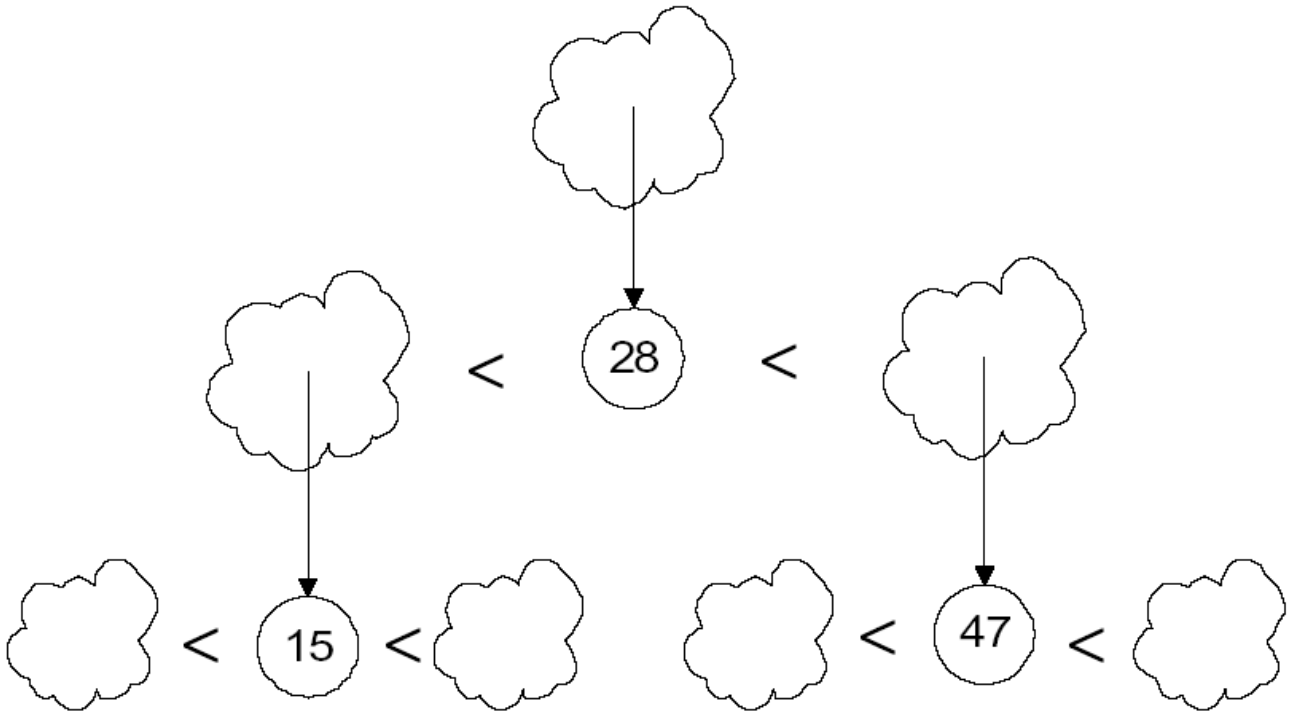
Hızlı Sıralama (Quick-Sort)

- Bu algoritma Böl-Yönet yaklaşımına dayanmaktadır.

Hızlı Sıralama



Bir pivot eleman seçilir. Pivottan küçük olanlar ve büyük olanlar olarak dizi ikiye bölünür.
Herbir taraf özyineli olarak sıralanır.

Hızlı sıralama (Quicksort) Algoritması

Tamsayılardan oluşan n elemanlı bir dizi verilmiş olsun.

- If {dizi sadece bir elemandan oluşuyorsa} , return
- Else
  - *pivot* olarak kullanmak için bir eleman seç
  - Diziyi aşağıdaki gibi iki alt dizi halinde böl:
    - *Pivot*'a eşit yada pivottan küçük olanlar
    - *Pivot*tan büyük olanlar
  - İki alt diziyi Hızlı sırala
  - Return sonuç dizi

Örnek

Verilen diziyi algoritmaya göre sıralayalım

40	20	10	80	60	50	7	30	100

Pivot Eleman seç

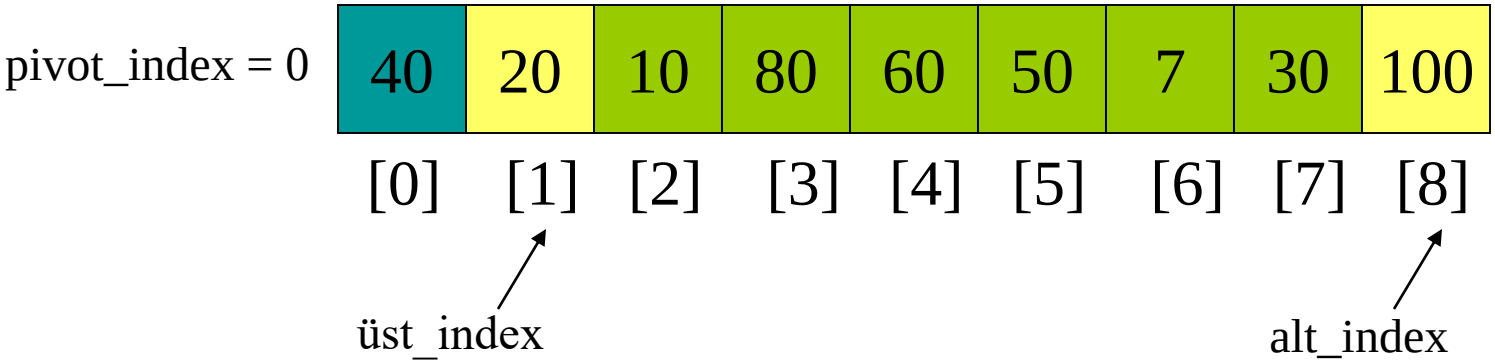
- Pivot eleman değişik şekillerde seçilebilir. Bu örnekte ilk eleman pivot olarak seçilecektir.

40	20	10	80	60	50	7	30	100

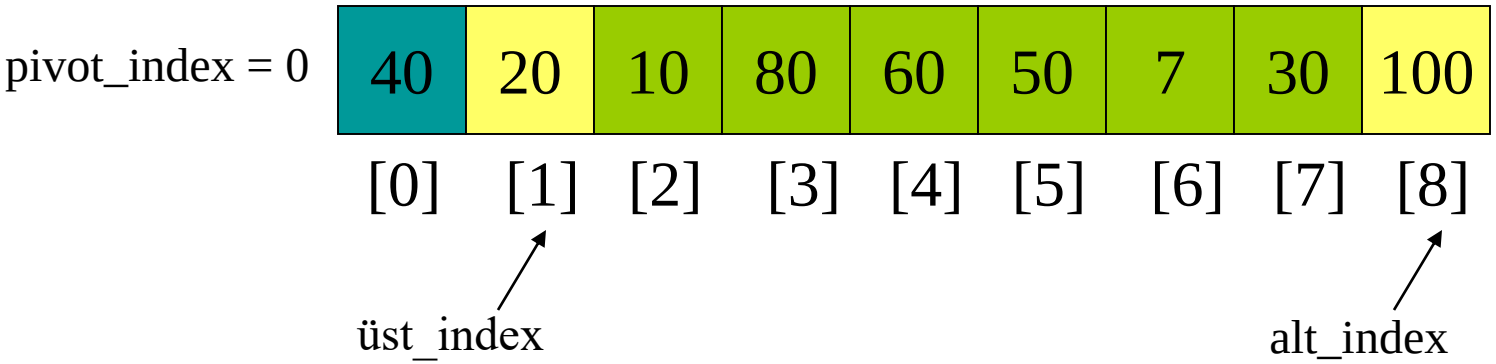
Diziyi bölme

- Pivottan küçük yada eşit olanlardan oluşan bir altdizi(<= pivot)
- Pivottan büyük olan elemanlardan oluşan bir dizi(> pivot)

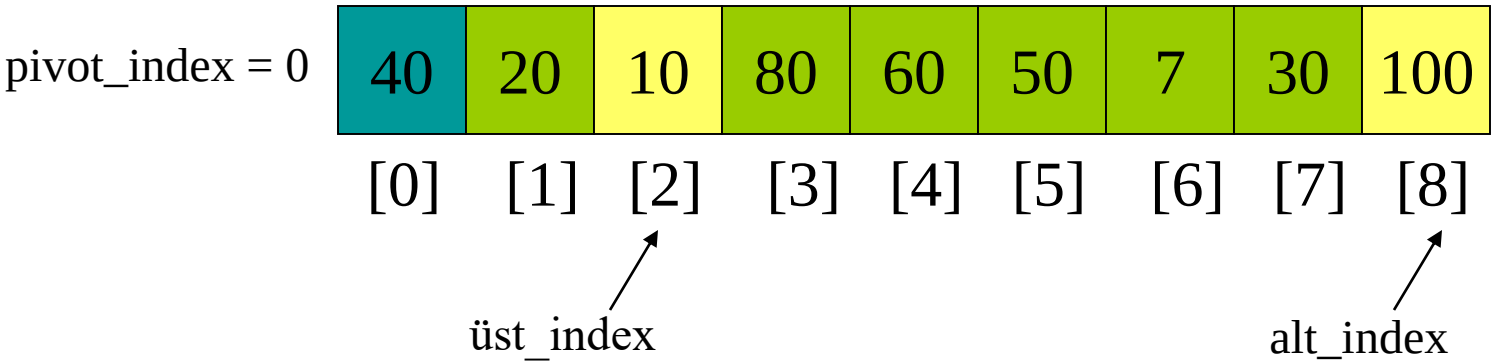
- **Altdiziler orijinal dizide tutulur.**
- **Bölme işlemini diziler parçalanamayacak hale gelene kadar rekürsif olarak tekrarla**


pivot_index = 0
40
20
10
80
60
50
7
30
100
[0]
[1]
[2]
[3]
[4]
[5]
[6]
[7]
[8]
üst_index
alt_index

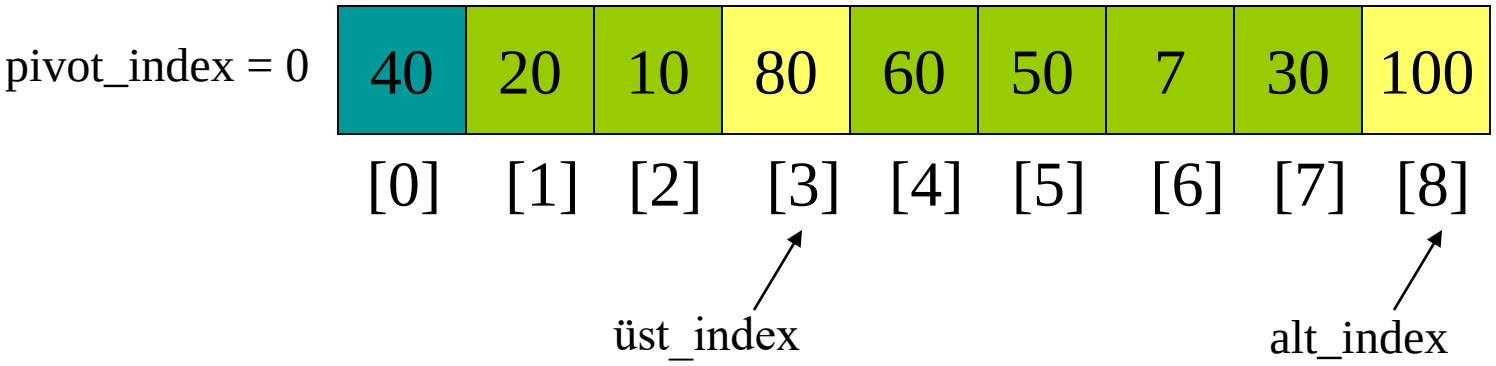
1. While data[üst_index] <= data[pivot]
   ++üst_index



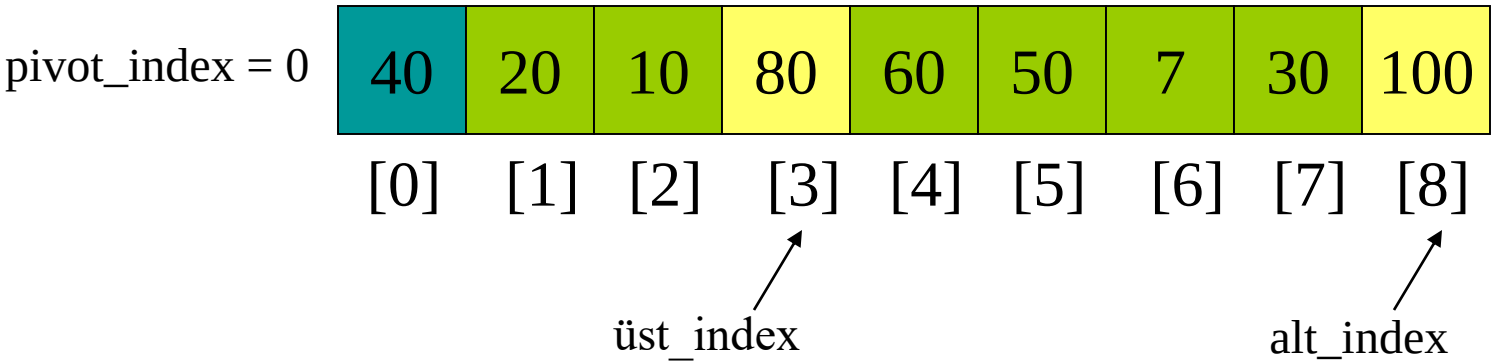
1. While data[üst_index] <= data[pivot]
   ++üst_index



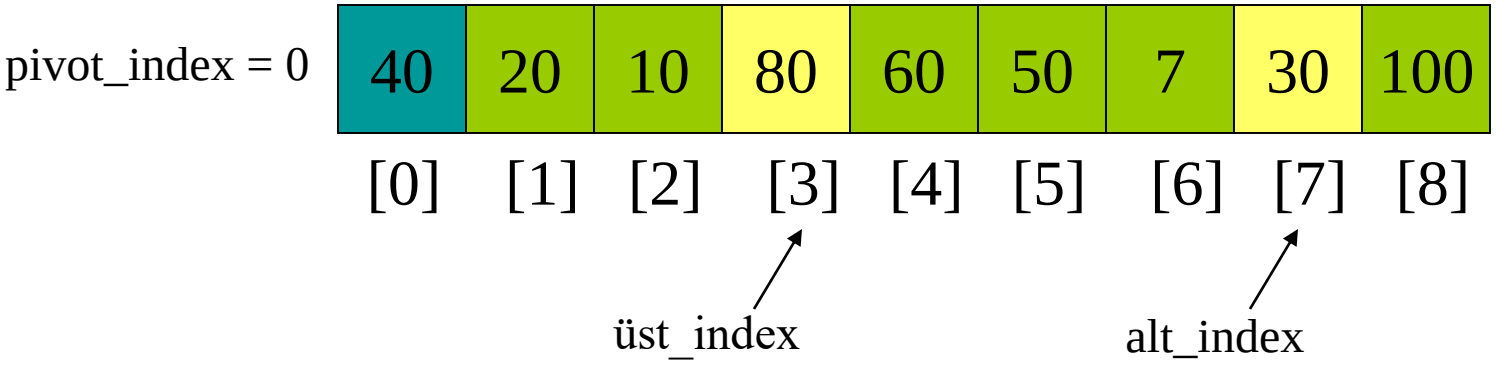
1. While data[üst_index] <= data[pivot]
   ++üst_index



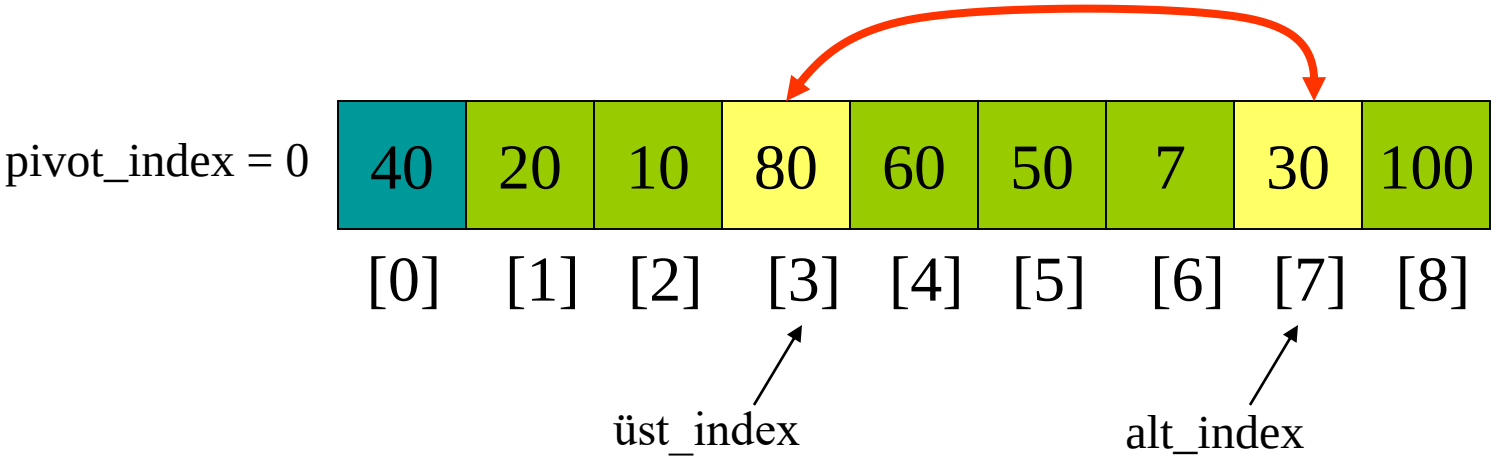
1. While data[üst_index] <= data[pivot]
   ++üst_index
2. While data[alt_index] > data[pivot]
   --alt_index



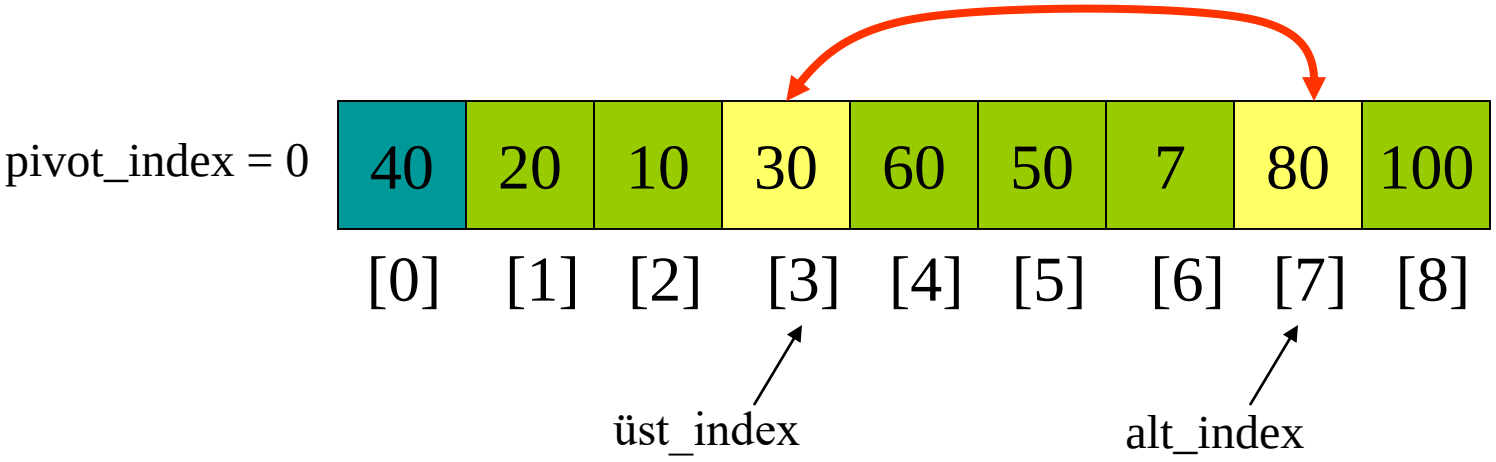
1. While data[üst_index] <= data[pivot]
   ++üst_index
2. While data[alt_index] > data[pivot]
   --alt_index



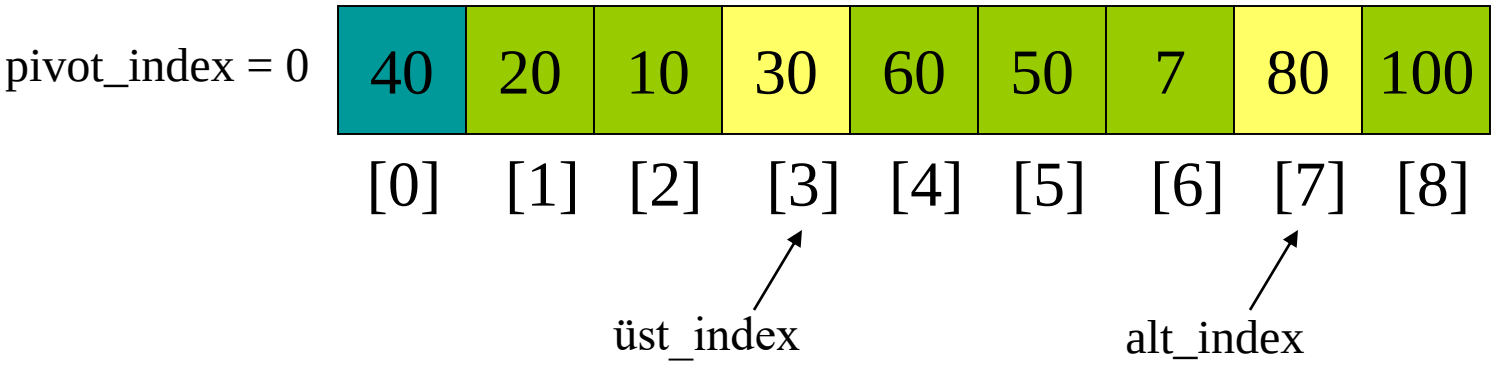
1. While data[üst_index] <= data[pivot]
   ++üst_index
2. While data[alt_index] > data[pivot]
   --alt_index
3. If üst_index < alt_index
   swap data[üst_index] and data[alt_index]



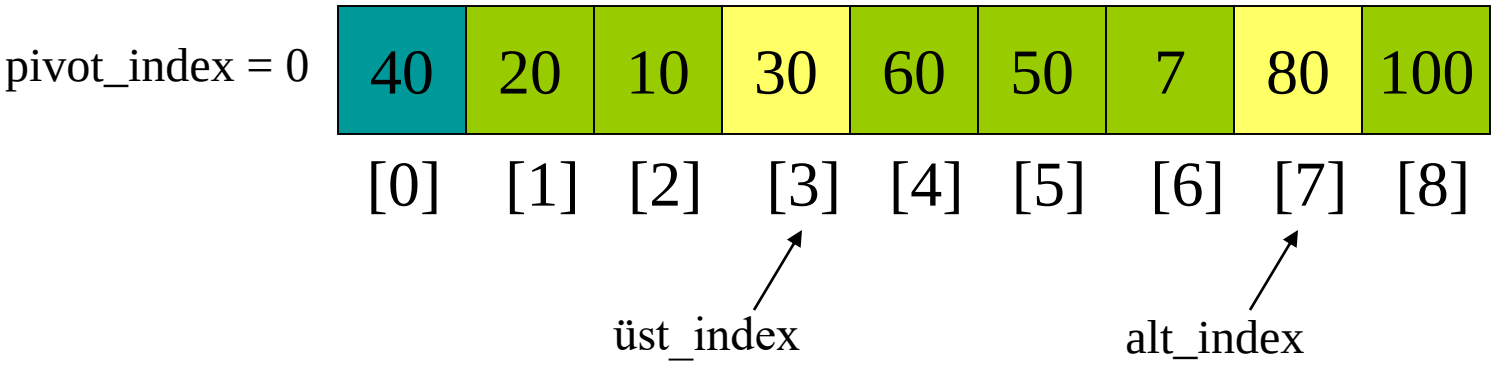
1. While data[üst_index] <= data[pivot]
   ++üst_index
2. While data[alt_index] > data[pivot]
   --alt_index
3. If üst_index < alt_index
   swap data[üst_index] and data[alt_index]



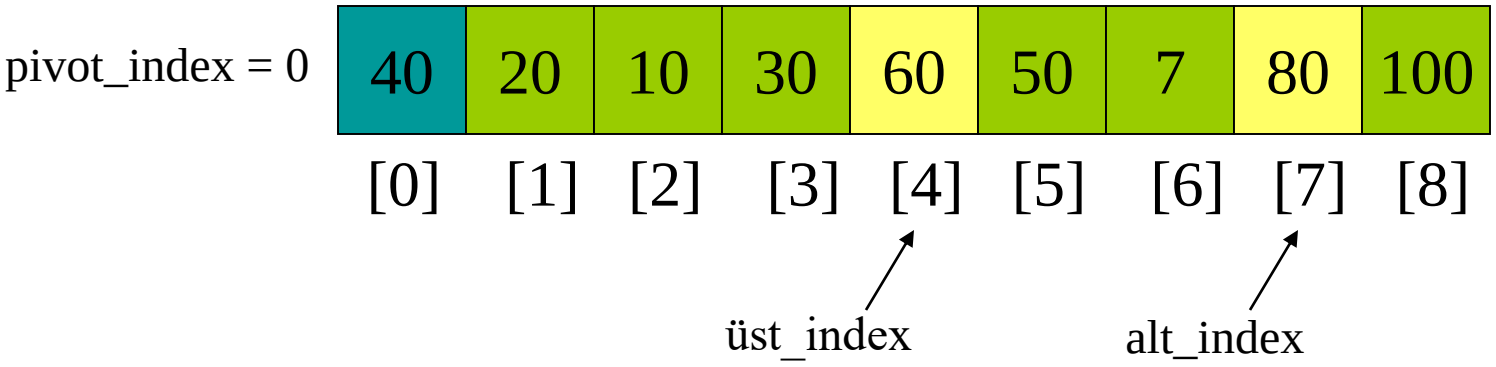
1. While data[üst_index] <= data[pivot]
   ++üst_index
2. While data[alt_index] > data[pivot]
   --alt_index
3. If üst_index < alt_index
   swap data[üst_index] and data[alt_index]
4. While alt_index > üst_index, go to 1.



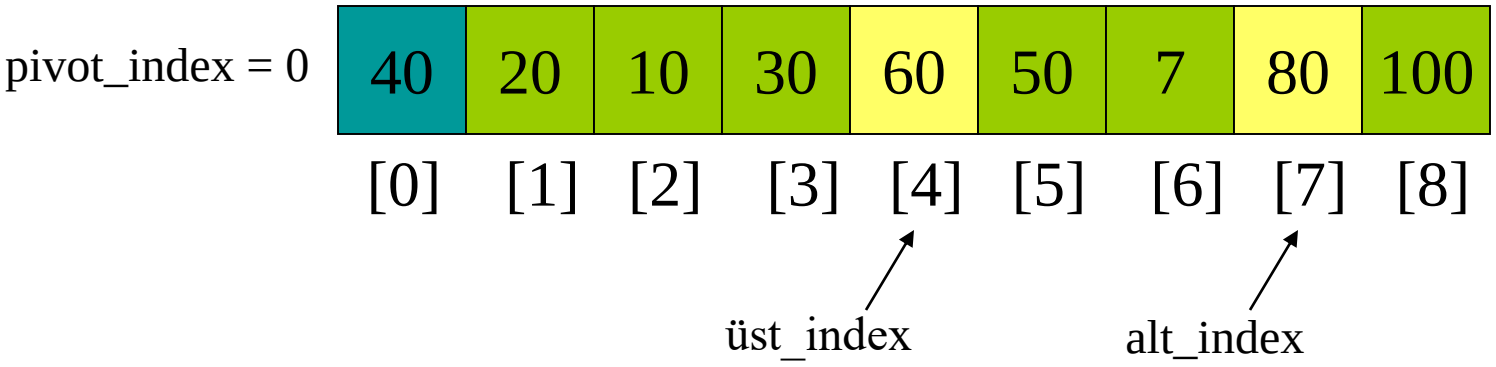
1. While data[üst_index] <= data[pivot]
   ++üst_index
2. While data[alt_index] > data[pivot]
   --alt_index
3. If üst_index < alt_index
   swap data[üst_index] and data[alt_index]
4. While alt_index > üst_index, go to 1.



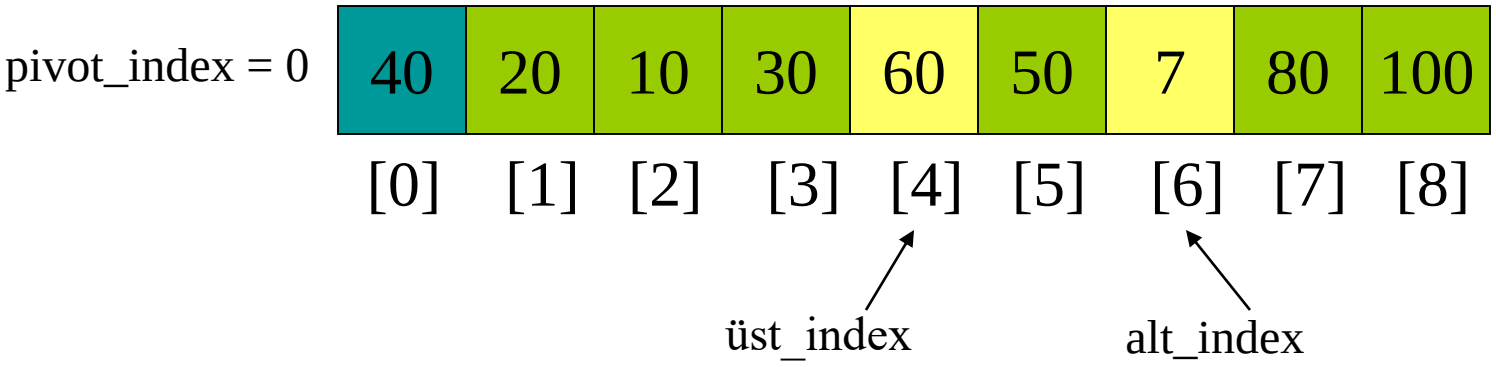
1. While data[üst_index] <= data[pivot]
   ++üst_index
2. While data[alt_index] > data[pivot]
   --alt_index
3. If üst_index < alt_index
   swap data[üst_index] and data[alt_index]
4. While alt_index > üst_index, go to 1.



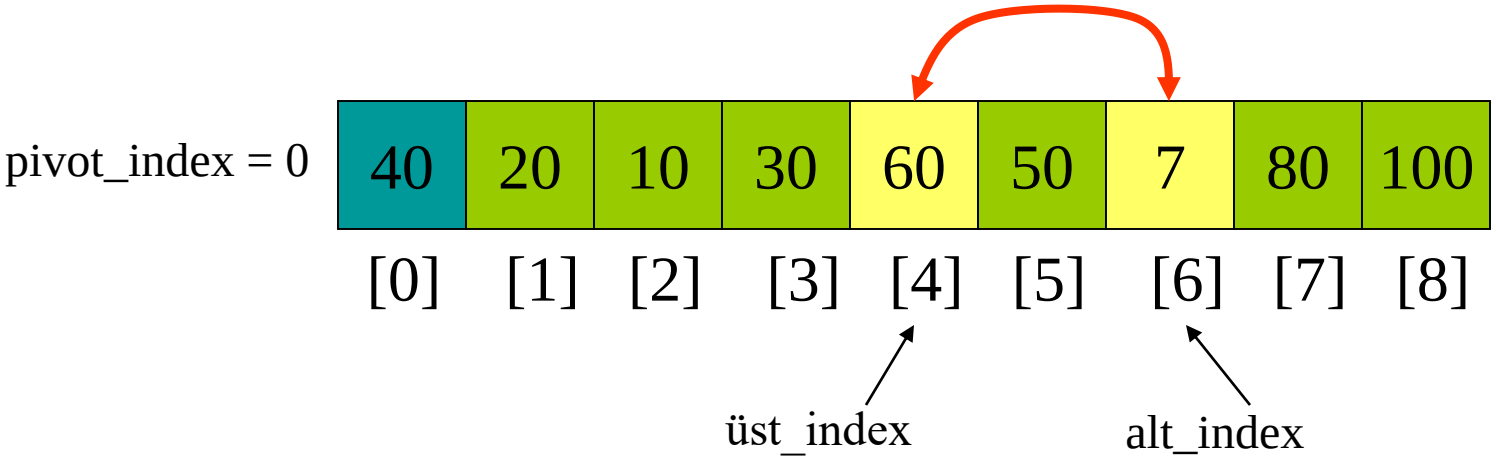
1. While data[üst_index] <= data[pivot]
   ++üst_index
2. While data[alt_index] > data[pivot]
   --alt_index
3. If üst_index < alt_index
   swap data[üst_index] and data[alt_index]
4. While alt_index > üst_index, go to 1.



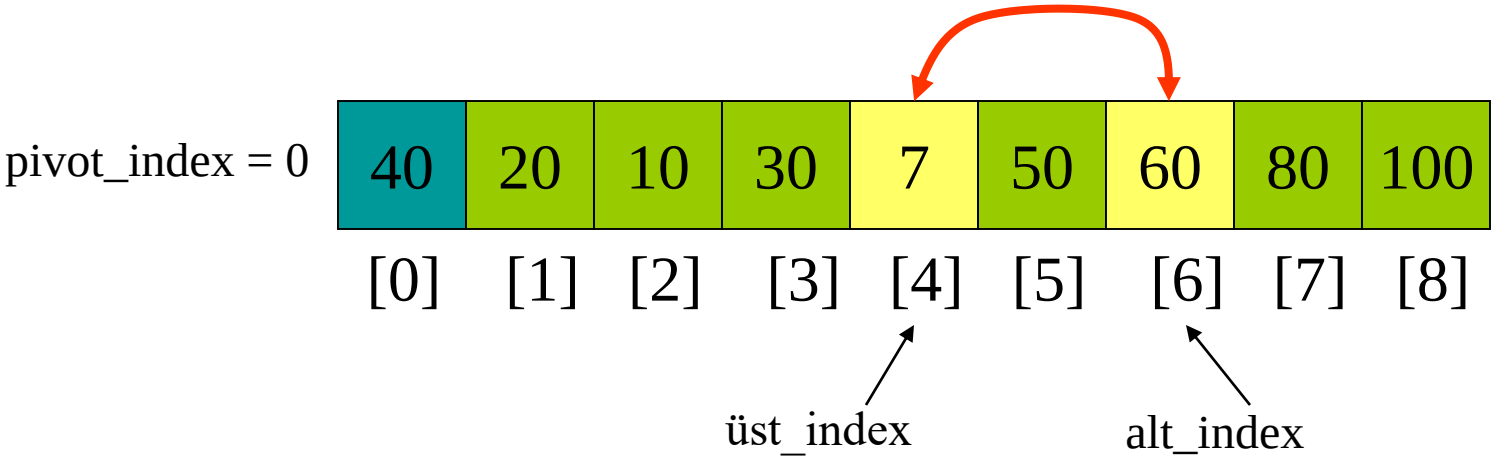
1. While data[üst_index] <= data[pivot]
   ++üst_index
2. While data[alt_index] > data[pivot]
   --alt_index
3. If üst_index < alt_index
   swap data[üst_index] and data[alt_index]
4. While alt_index > üst_index, go to 1.



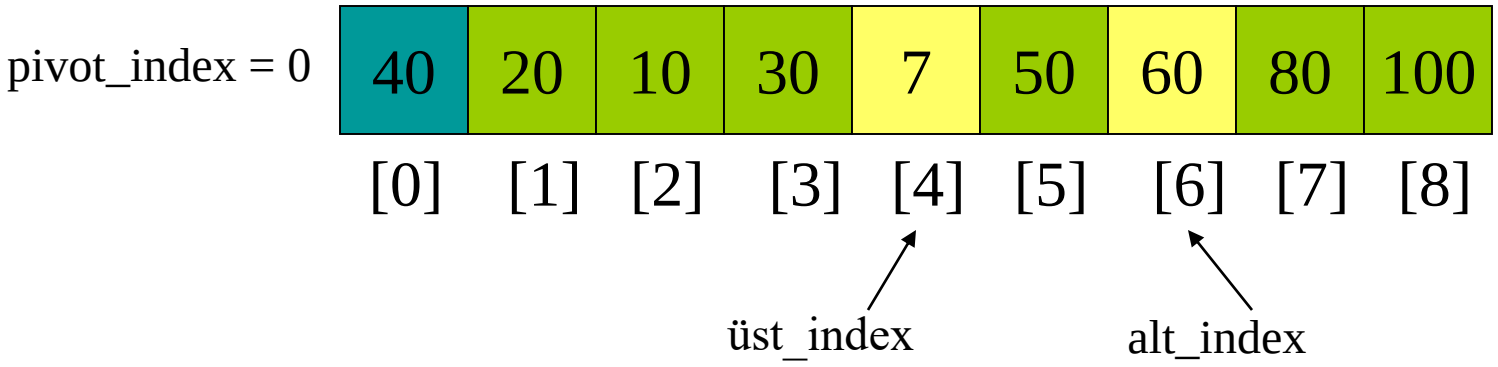
1. While data[üst_index] <= data[pivot]
   ++üst_index
2. While data[alt_index] > data[pivot]
   --alt_index
3. If üst_index < alt_index
   swap data[üst_index] and data[alt_index]
4. While alt_index > üst_index, go to 1.



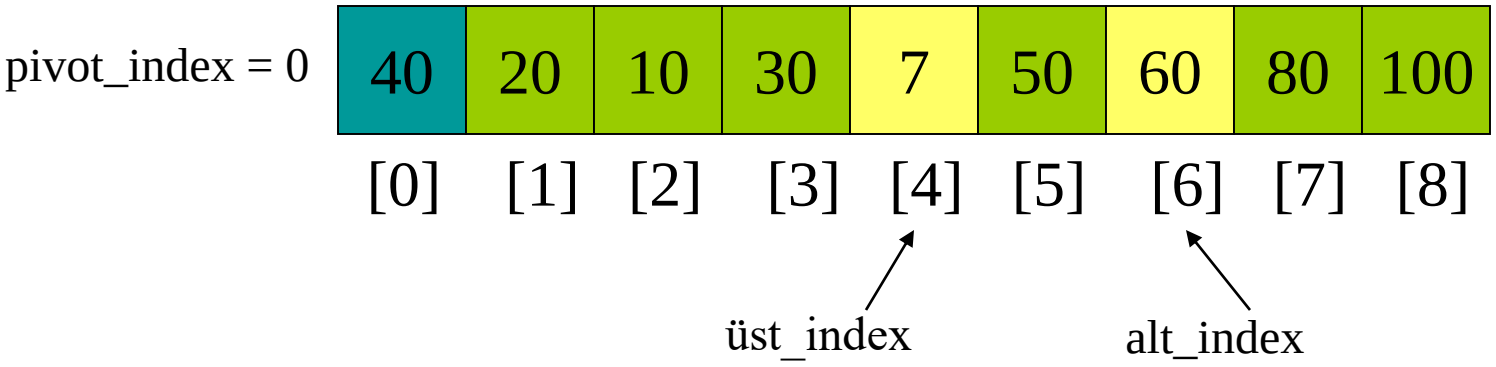
1. While data[üst_index] <= data[pivot]
   ++üst_index
2. While data[alt_index] > data[pivot]
   --alt_index
3. If üst_index < alt_index
   swap data[üst_index] and data[alt_index]
4. While alt_index > üst_index, go to 1.



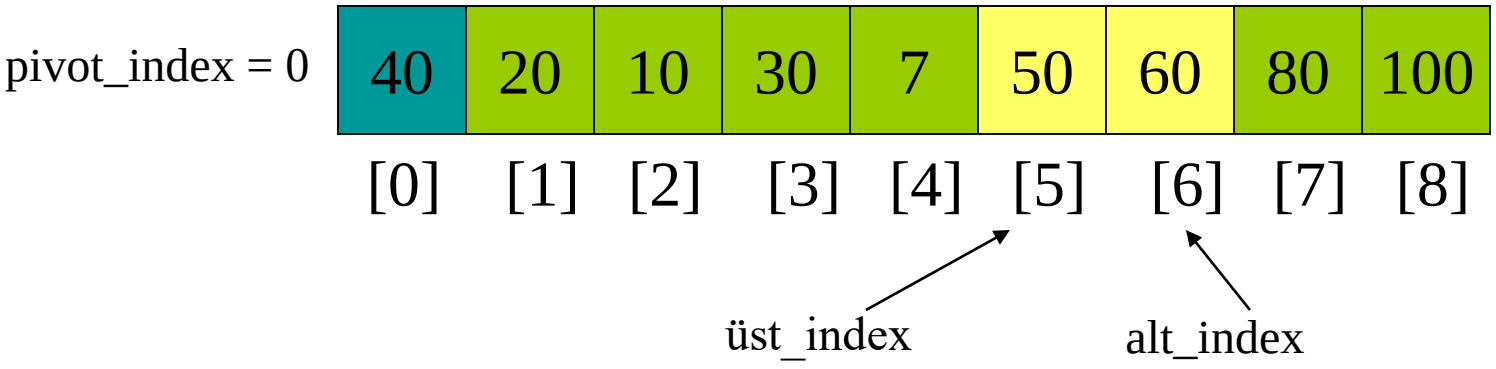
1. While data[üst_index] <= data[pivot]
   ++üst_index
2. While data[alt_index] > data[pivot]
   --alt_index
3. If üst_index < alt_index
   swap data[üst_index] and data[alt_index]
4. While alt_index > üst_index, go to 1.



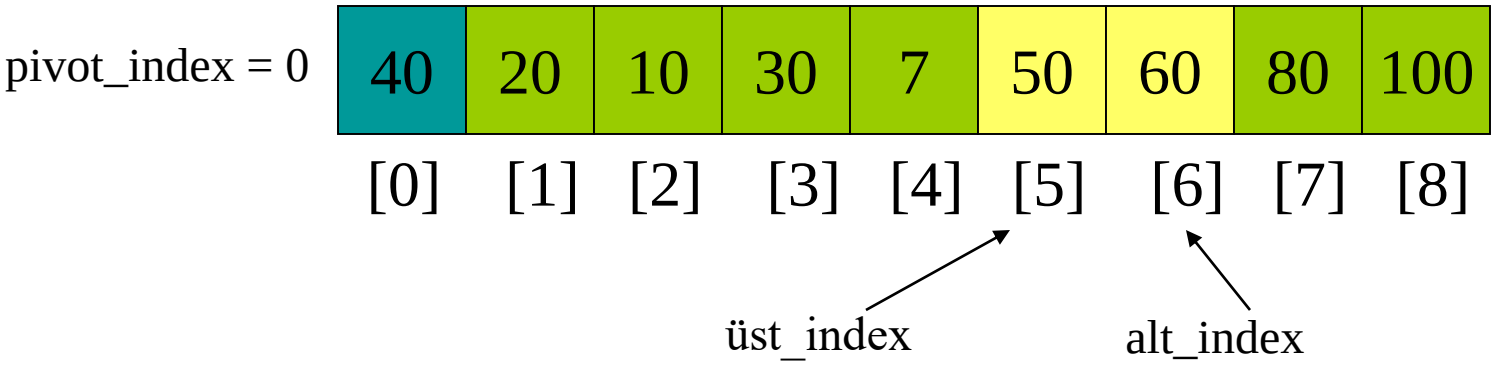
1. While data[üst_index] <= data[pivot]
   ++üst_index
2. While data[alt_index] > data[pivot]
   --alt_index
3. If üst_index < alt_index
   swap data[üst_index] and data[alt_index]
4. While alt_index > üst_index, go to 1.



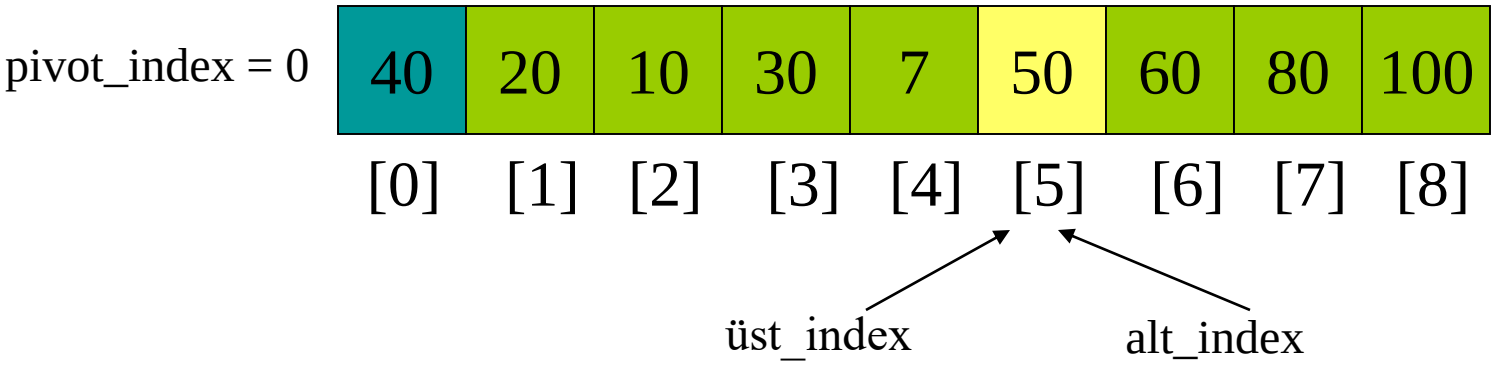
1. While data[üst_index] <= data[pivot]
   ++üst_index
2. While data[alt_index] > data[pivot]
   --alt_index
3. If üst_index < alt_index
   swap data[üst_index] and data[alt_index]
4. While alt_index > üst_index, go to 1.



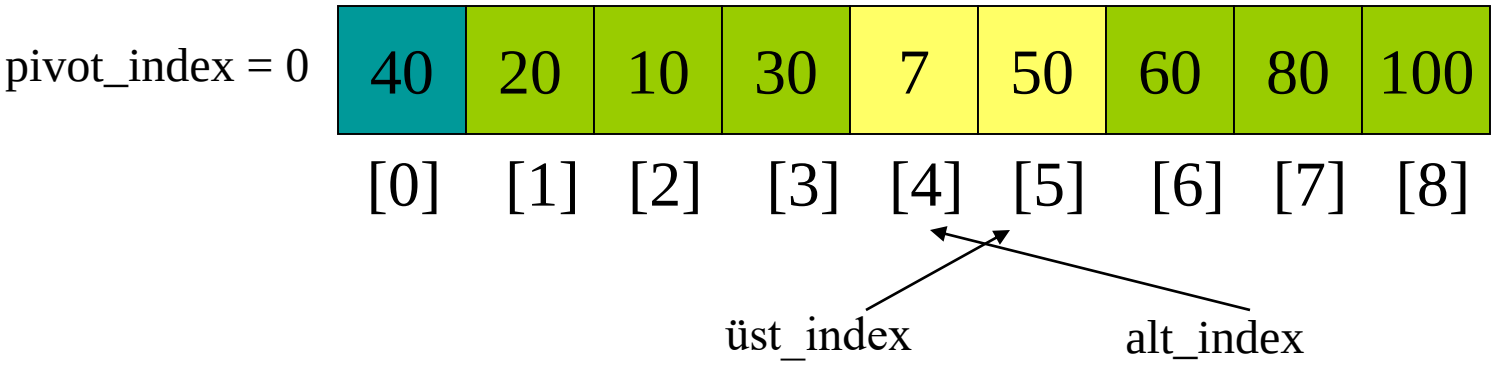
1. While data[üst_index] <= data[pivot]
   ++üst_index
2. While data[alt_index] > data[pivot]
   --alt_index
3. If üst_index < alt_index
   swap data[üst_index] and data[alt_index]
4. While alt_index > üst_index, go to 1.



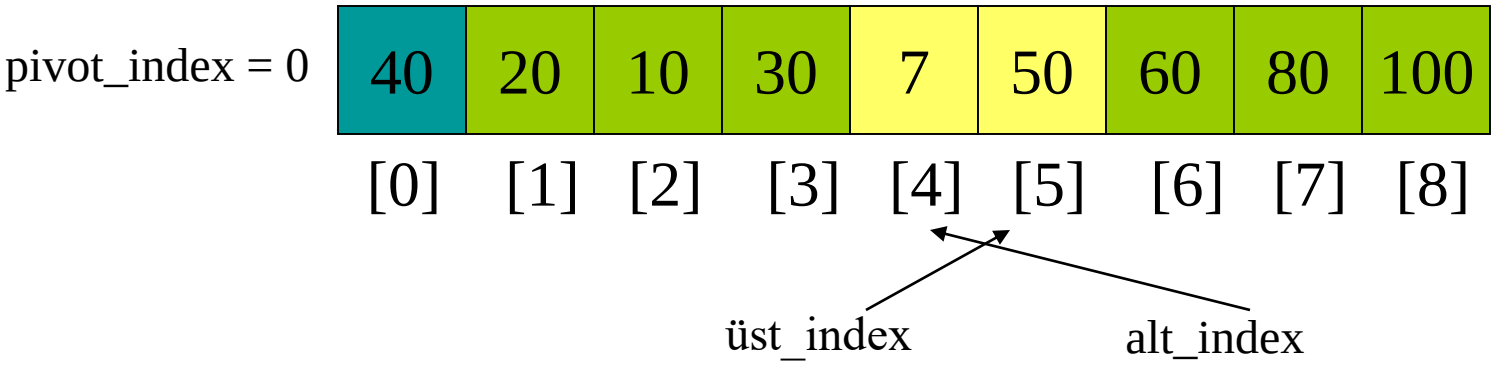
1. While data[üst_index] <= data[pivot]
   ++üst_index
2. While data[alt_index] > data[pivot]
   --alt_index
3. If üst_index < alt_index
   swap data[üst_index] and data[alt_index]
4. While alt_index > üst_index, go to 1.



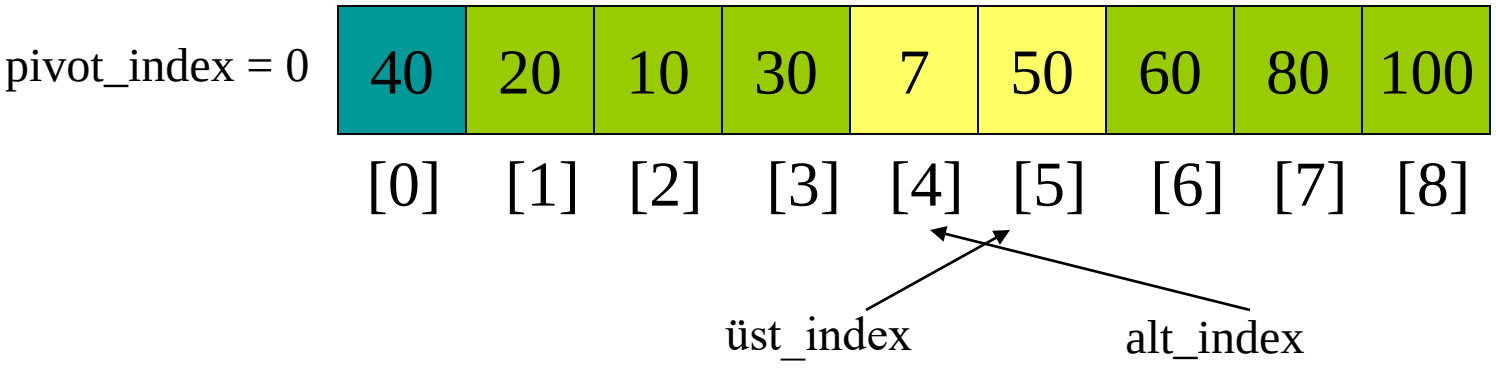
1. While data[üst_index] <= data[pivot]
   ++üst_index
2. While data[alt_index] > data[pivot]
   --alt_index
3. If üst_index < alt_index
   swap data[üst_index] and data[alt_index]
4. While alt_index > üst_index, go to 1.



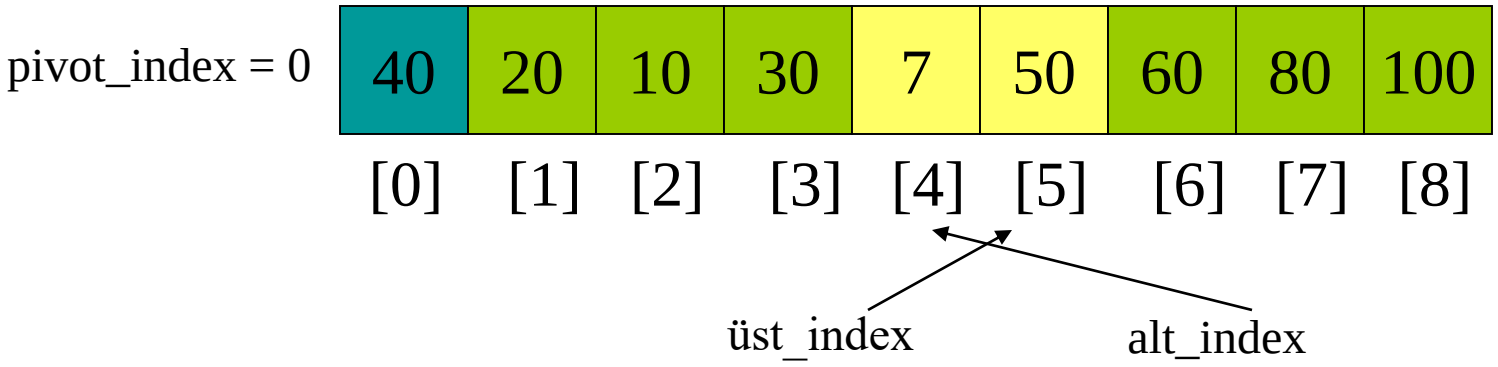
1. While data[üst_index] <= data[pivot]
   ++üst_index
2. While data[alt_index] > data[pivot]
   --alt_index
3. If üst_index < alt_index
   swap data[üst_index] and data[alt_index]
4. While alt_index > üst_index, go to 1.



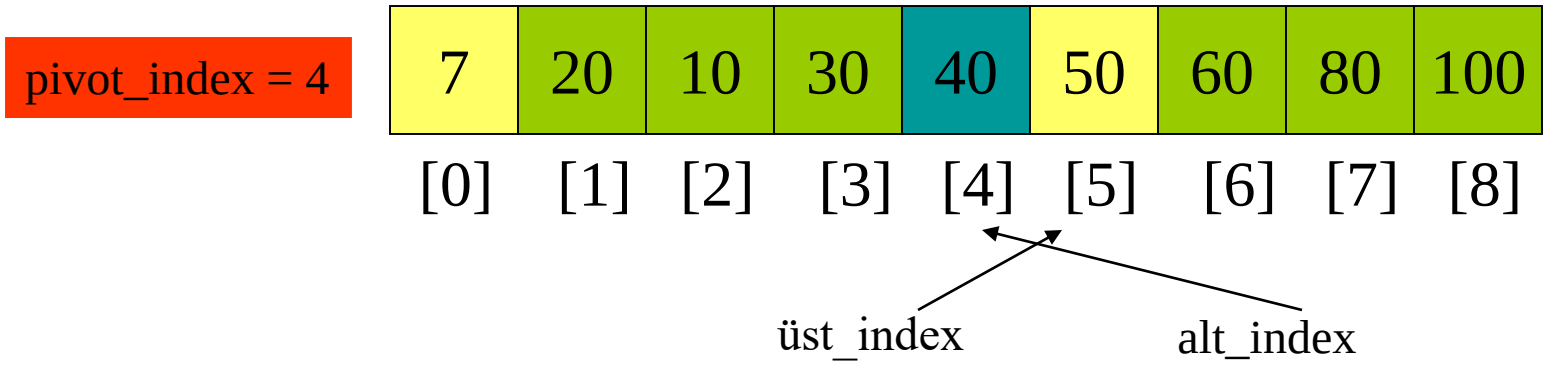
1. While data[üst_index] <= data[pivot]
   ++üst_index
2. While data[alt_index] > data[pivot]
   --alt_index
3. If üst_index < alt_index
   swap data[üst_index] and data[alt_index]
4. While alt_index > üst_index, go to 1.



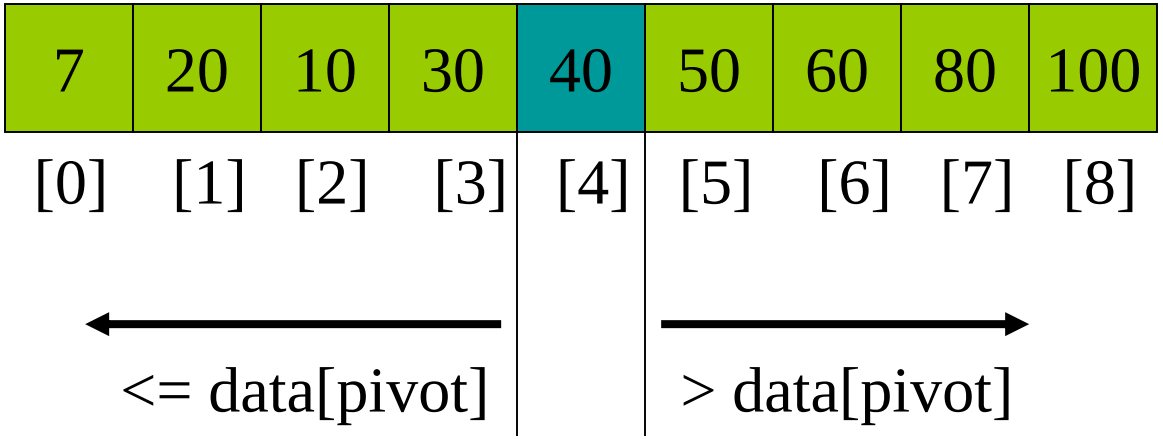
1. While data[üst_index] <= data[pivot]
   ++üst_index
2. While data[alt_index] > data[pivot]
   --alt_index
3. If üst_index < alt_index
   swap data[üst_index] and data[alt_index]
4. While alt_index > üst_index, go to 1.
5. Swap data[alt_index] and data[pivot_index]



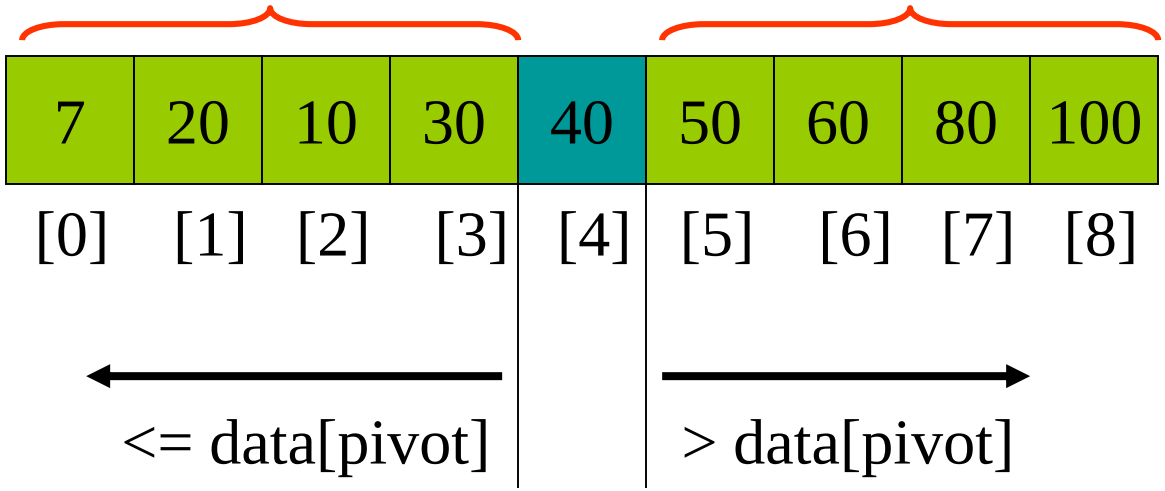
1. While data[üst_index] <= data[pivot]
   ++üst_index
2. While data[alt_index] > data[pivot]
   --alt_index
3. If üst_index < alt_index
   swap data[üst_index] and data[alt_index]
4. While alt_index > üst_index, go to 1.
5. Swap data[alt_index] and data[pivot_index]



Oluşturulan iki alt dizi

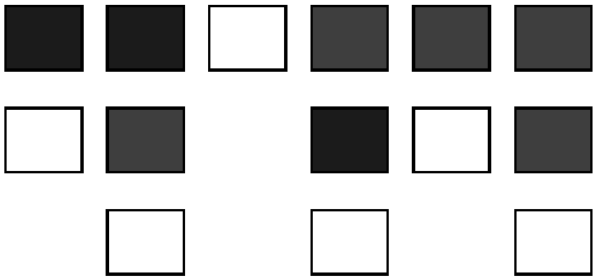


Recursion: Altdizilere Quicksort



Quicksort Algoritmasının Analizi

- Elemanlar düzensiz dağılmışsa,
- **“Best case” çalışma zamanı: $O(n \log_2 n)$**
  - Pivot eleman herzaman ortada
  - Her rekürsif çağrımda dizi iki eş parçaya ayrılmakta.



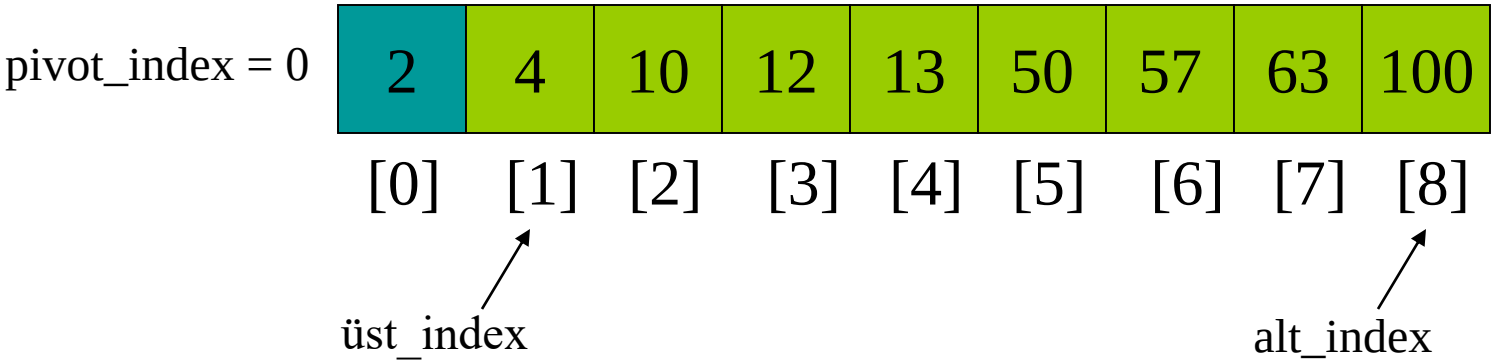
Quicksort Analysis

Worst case: $O(N^2)$

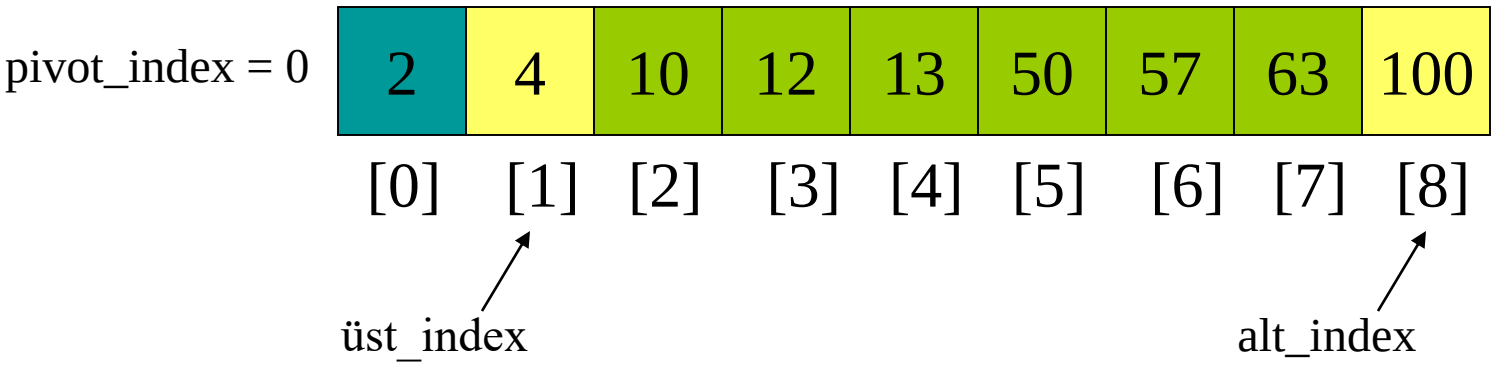
- Her rekürsif çağrımda pivot en küçük elemandır.
- Dizi zaten sıralı olması halidir.

Quicksort: Worst Case

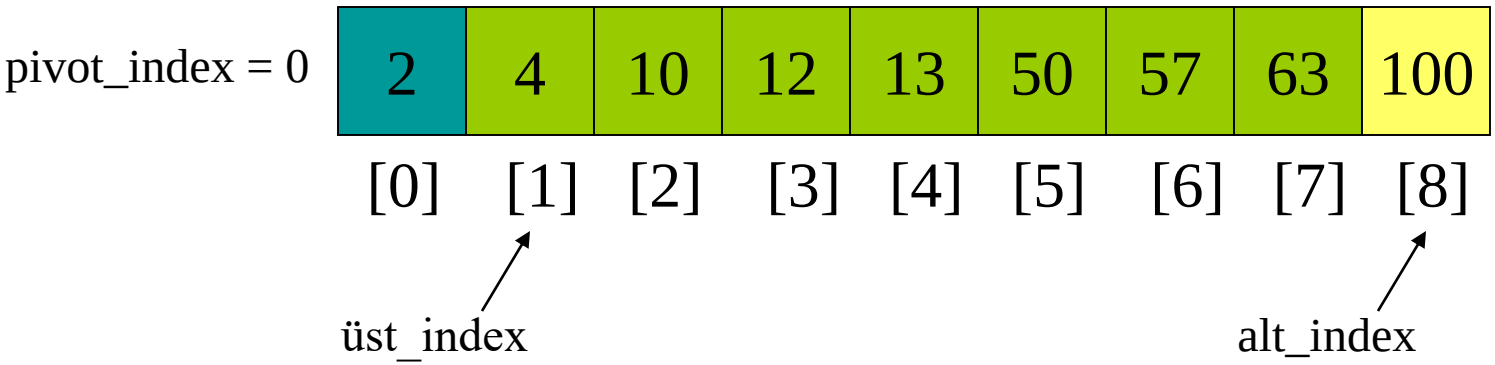
- İlk eleman pivot olarak seçilmiş olsun.
- Dizi zaten sıralı olsun



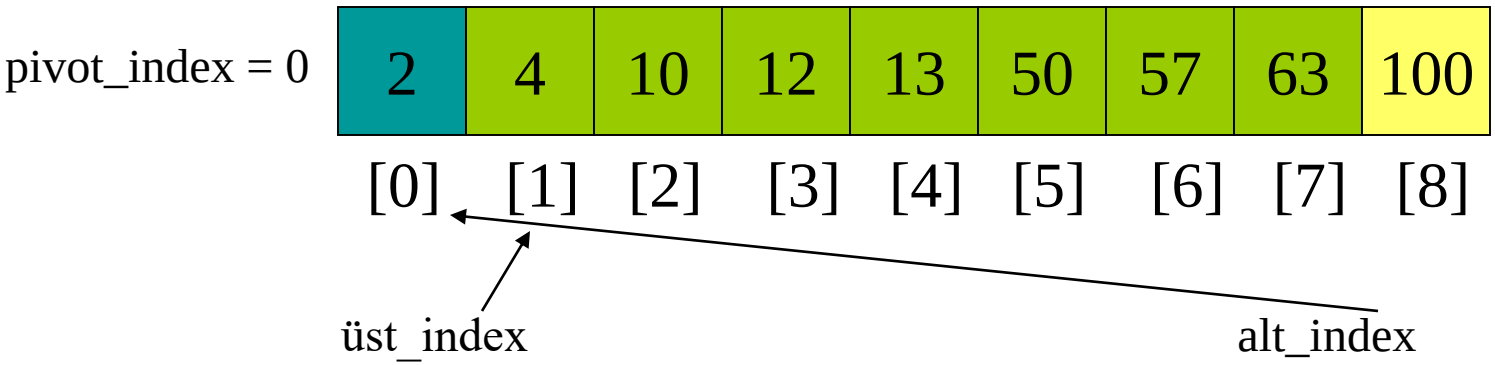
1. While data[üst_index] <= data[pivot]
   ++üst_index
2. While data[alt_index] > data[pivot]
   --alt_index
3. If üst_index < alt_index
   swap data[üst_index] and data[alt_index]
4. While alt_index > üst_index, go to 1.
5. Swap data[alt_index] and data[pivot_index]



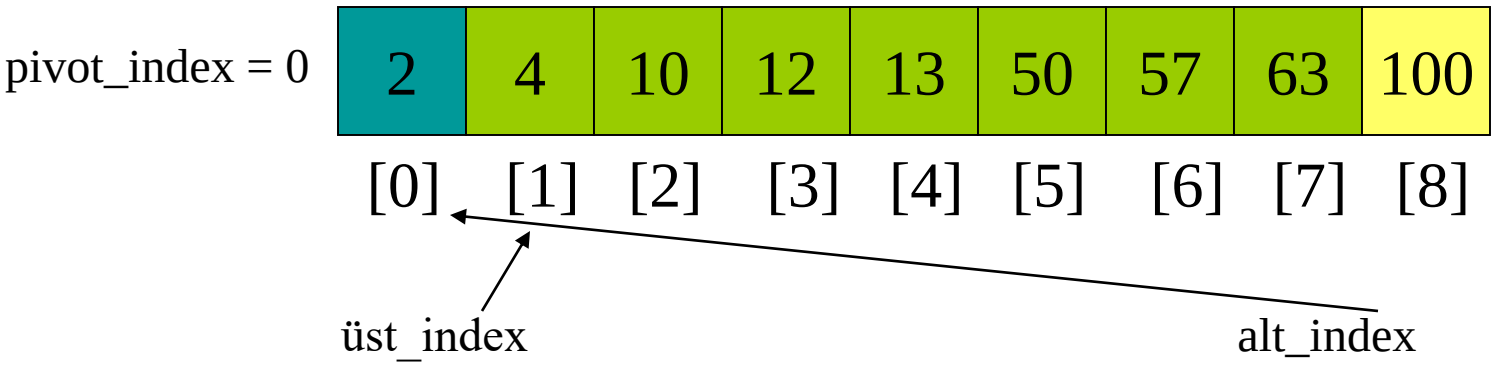
1. While data[üst_index] <= data[pivot]
   ++üst_index
2. While data[alt_index] > data[pivot]
   --alt_index
3. If üst_index < alt_index
   swap data[üst_index] and data[alt_index]
4. While alt_index > üst_index, go to 1.
5. Swap data[alt_index] and data[pivot_index]



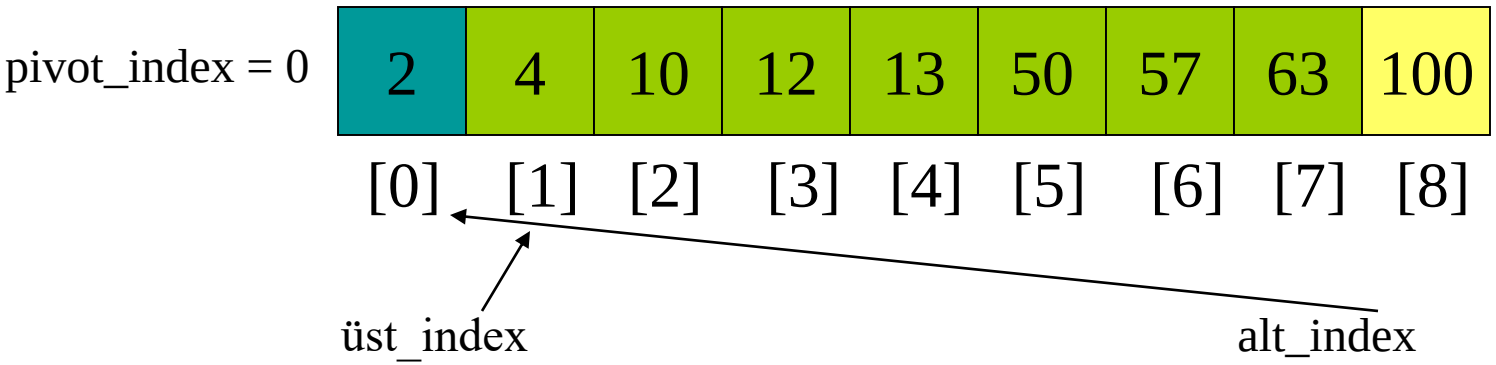
1. While data[üst_index] <= data[pivot]
   ++üst_index
2. While data[alt_index] > data[pivot]
   --alt_index
3. If üst_index < alt_index
   swap data[üst_index] and data[alt_index]
4. While alt_index > üst_index, go to 1.
5. Swap data[alt_index] and data[pivot_index]



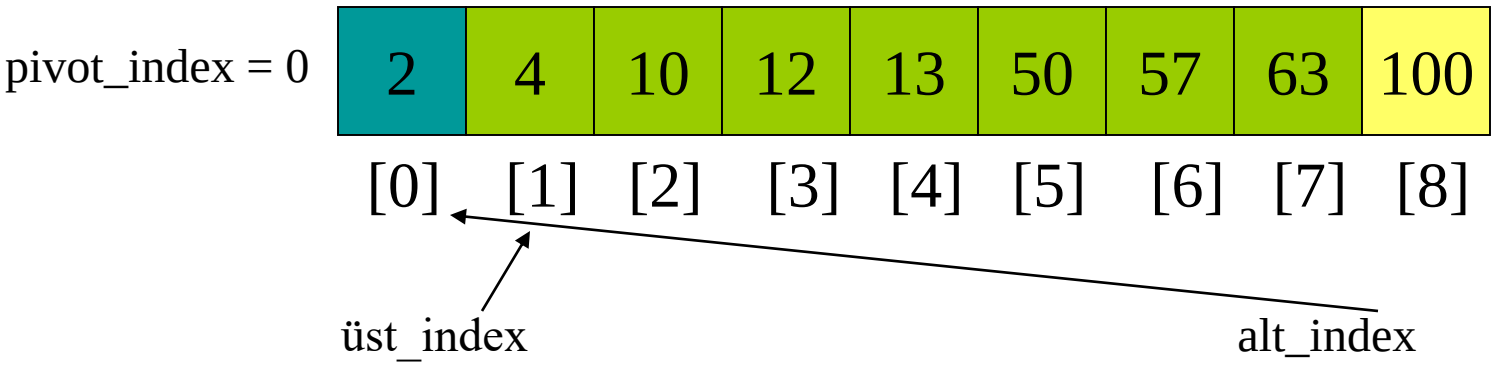
1. While data[üst_index] <= data[pivot]
   ++üst_index
2. While data[alt_index] > data[pivot]
   --alt_index
3. If üst_index < alt_index
   swap data[üst_index] and data[alt_index]
4. While alt_index > üst_index, go to 1.
5. Swap data[alt_index] and data[pivot_index]



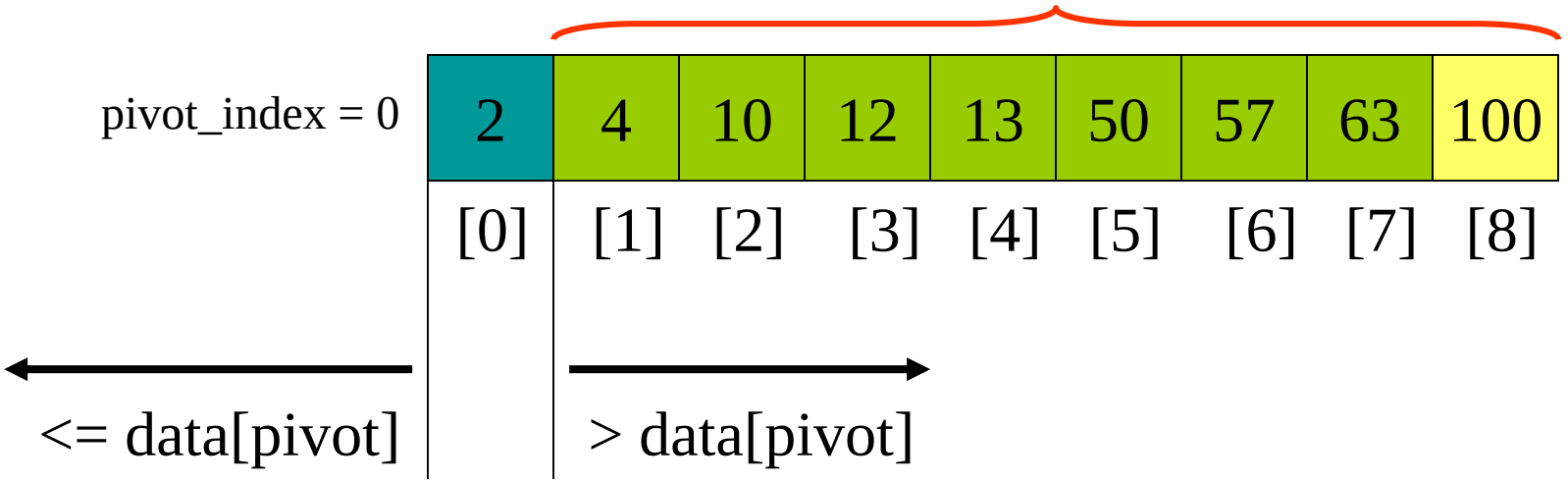
1. While data[üst_index] <= data[pivot]
   ++üst_index
2. While data[alt_index] > data[pivot]
   --alt_index
3. If üst_index < alt_index
   swap data[üst_index] and data[alt_index]
4. While alt_index > üst_index, go to 1.
5. Swap data[alt_index] and data[pivot_index]



1. While data[üst_index] <= data[pivot]
   ++üst_index
2. While data[alt_index] > data[pivot]
   --alt_index
3. If üst_index < alt_index
   swap data[üst_index] and data[alt_index]
4. While alt_index > üst_index, go to 1.
5. Swap data[alt_index] and data[pivot_index]



1. While data[üst_index] <= data[pivot]
   ++üst_index
2. While data[alt_index] > data[pivot]
   --alt_index
3. If üst_index < alt_index
   swap data[üst_index] and data[alt_index]
4. While alt_index > üst_index, go to 1.
5. Swap data[alt_index] and data[pivot_index]



$T(n) = T(k) + T(n - k) + \alpha n$

En kötü durumda dizinin bölünmesi 1 ve n-1 elemanlı biçiminde olacaktır.

$T(n) = T(1) + T(n - 1) + \alpha n$

$T(n) = T(n - 1) + T(1) + \alpha n$

$= [T(n - 2) + T(1) + \alpha(n - 1)] + T(1) + \alpha n$

$= T(n - 2) + 2T(1) + \alpha(n - 1 + n)$

$= [T(n - 3) + T(1) + \alpha(n - 2)] + 2T(1) + \alpha(n - 1 + n)$

$= T(n - 3) + 3T(1) + \alpha(n - 2 + n - 1 + n)$

$= [T(n - 4) + T(1) + \alpha (n - 3)] + 3T(1) + \alpha (n - 2 + n - 1 + n)$

$= T(n - 4) + 4T(1) + \alpha (n - 3 + n - 2 + n - 1 + n)$

$= T(n - i) + iT (1) + \alpha (n - i + 1 + ... + n - 2 + n - 1 + n)$

$= T(n - i) + i.T(1) + \alpha(\sum_{j=0}^{i-1}(n - j))$

n-i=1
i=n-1

$= T(1) + (n-1)T(1) + \alpha(\sum_{j=0}^{n-2}(n - j))$

$$\sum_{j=0}^{n-2} j = \sum_{j=1}^{n-2} j = (n-2)(n-1)/2$$

Karesel çalışma zamanı

Eniyi Durum Analizi

En iyi durumda dizi her defasında tam olarak ikiye bölünecektir.

$$T(n) = 2T(n/2) + \alpha n \ = 2(2T(n/4) + \alpha n/2) + \alpha n$$

$$T(n/2) \ = \ 2T(n/4) + \alpha n/2$$

$$T(n) = 2T(n/2) + \alpha n.)$$

$$= 2^2 T(n/4) + 2\alpha n$$

$$= 2^2(2T(n/8) + \alpha n/4) + 2\alpha n$$

$$= 2^3 T(n/8) + 3\alpha n$$

$$= 2^k T(n/2^k) + k\alpha n$$

$$n \ = \ 2^k$$

$$n/2^k \ < \ 1$$

$$k = \log n.$$

$$T(n) = nT(1) + \alpha n \log n, \text{ which is } O(n \log n).$$

Quicksort Algoritmasının Çalışma Zamanı Analizi

- **Best** case : **$O(n \log_2 n)$**
- **Worst** case : **$O(n^2)$**
- **Average** case : **$O(n \log_2 n)$**

Quicksort

QUICKSORT(A, p, r)

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

Algoritma rekürsif yapıda olduğundan bilgisayarın yığıtı yoğun bir biçimde kullanılmaktadır.