Freie Universität Berlin

Sortieralgorithmen

Einführung

SS 2012

Prof. Dr. Margarita Esponda
Freie Universität Berlin

Freie Universität Berlin

Beispiel:

Array

3
5
7
11
17
19
23
29
31
34
37
57

Sortierte Menge

Eingabegröße:

n = Anzahl der sortierte Zahlen

Berechnungsschritt:

Vergleichsoperation

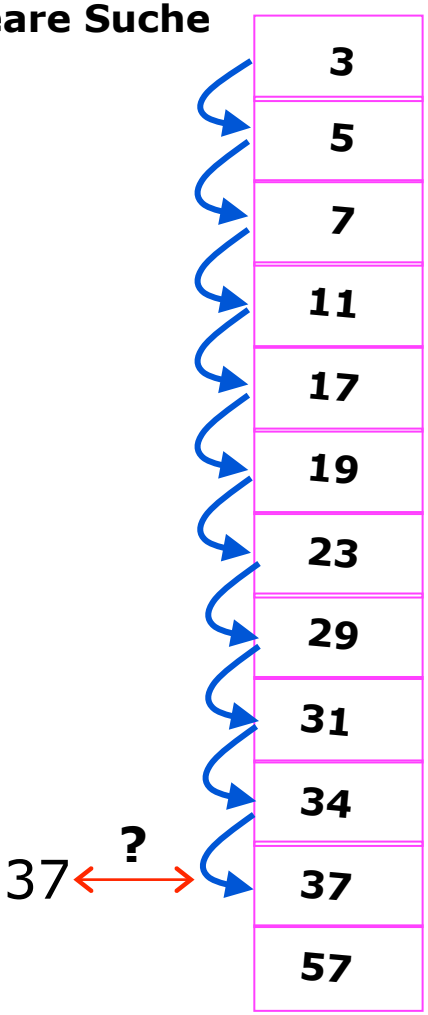
Komplexitätsanalyse:

T(n) = Anzahl der Berechnungsschritte, um eine Zahl zu finden.

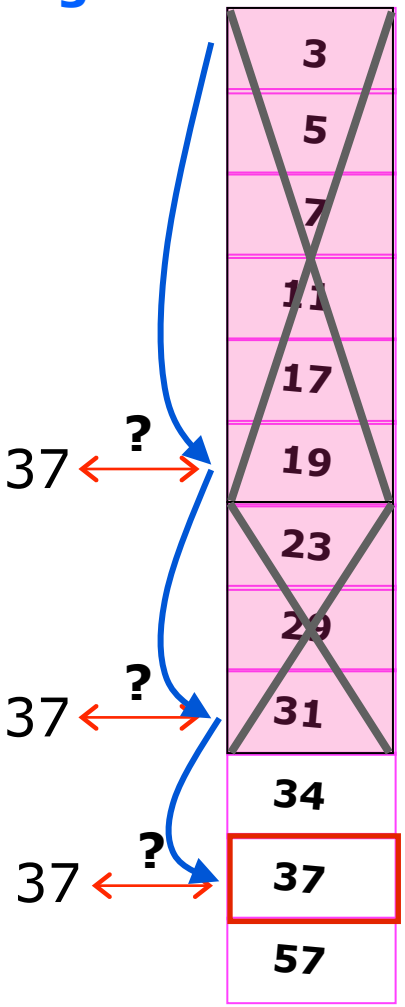
Freie Universität Berlin

Sortierte Menge

Lineare Suche



Im schlimmsten Fall **n** Schritte



nur **3** Schritte

Freie Universität Berlin

Maximale Schrittanzahl

$128 = 2^7$

$64 = 2^6$

$32 = 2^5$

$16 = 2^4$

$8 = 2^3$

$4 = 2^2$

$2 = 2^1$

$1 = 2^0$

$7 = \log_2 (128)$

Im schlimmsten Fall

$\log_2(n)$ Schritte

Lineare Suche

```
def linear_search(key, seq ):
    for i in seq:
        if i==key:
            return True
    return False
```

Freie Universität Berlin

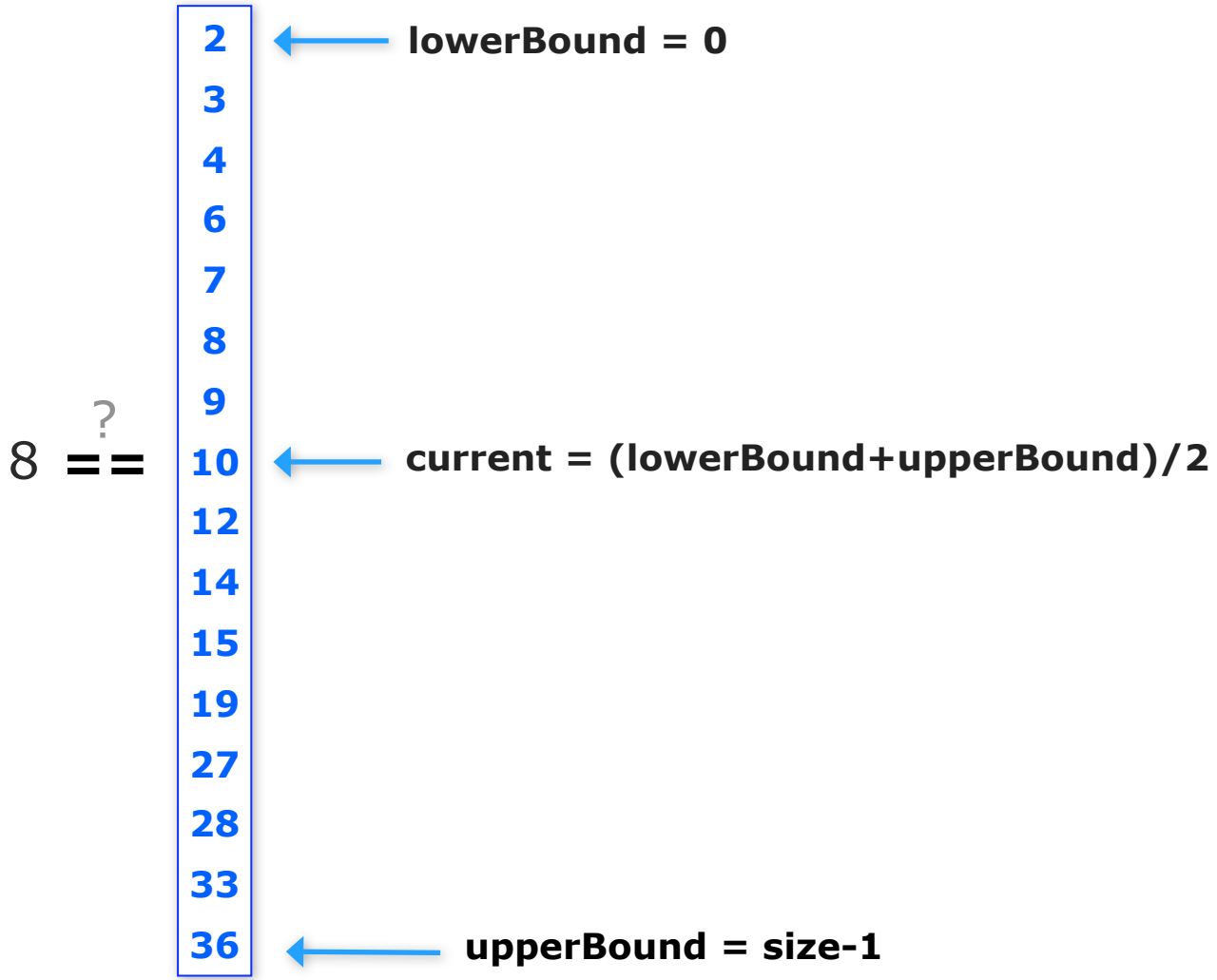
Binäre Suche

Rekursiv

```
def binary_search( key, seq, first, last ):
    if first<last:
        m = (first+last)//2
        if seq[m]==key:
            return True
        elif key<seq[m]:
            return binary_search(key,seq,first,m-1)
        else:
            return binary_search(key,seq,m+1,last)
    elif first==last:
        return seq[first]==key
    else:
        return False
```

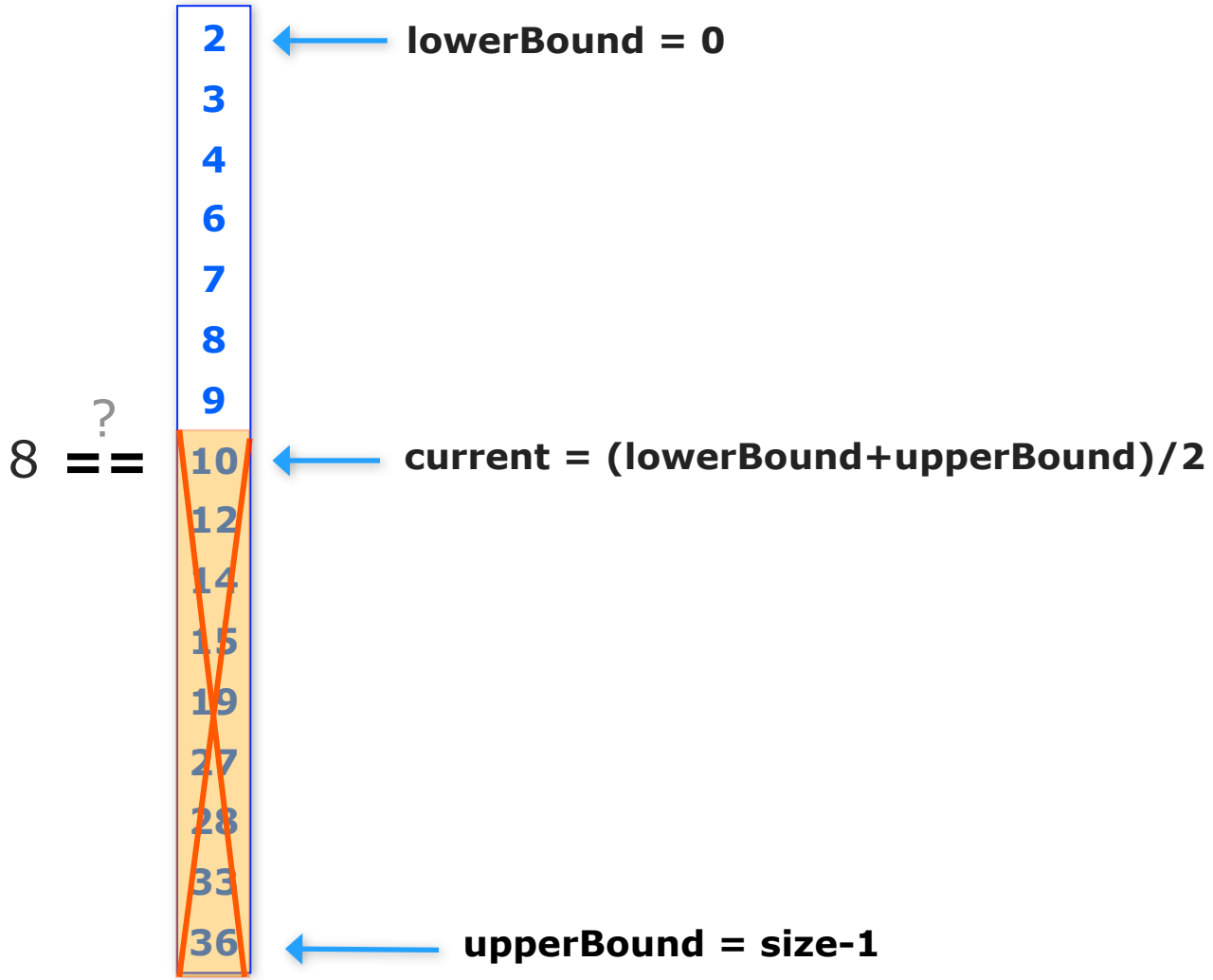
Freie Universität Berlin

Iterativ



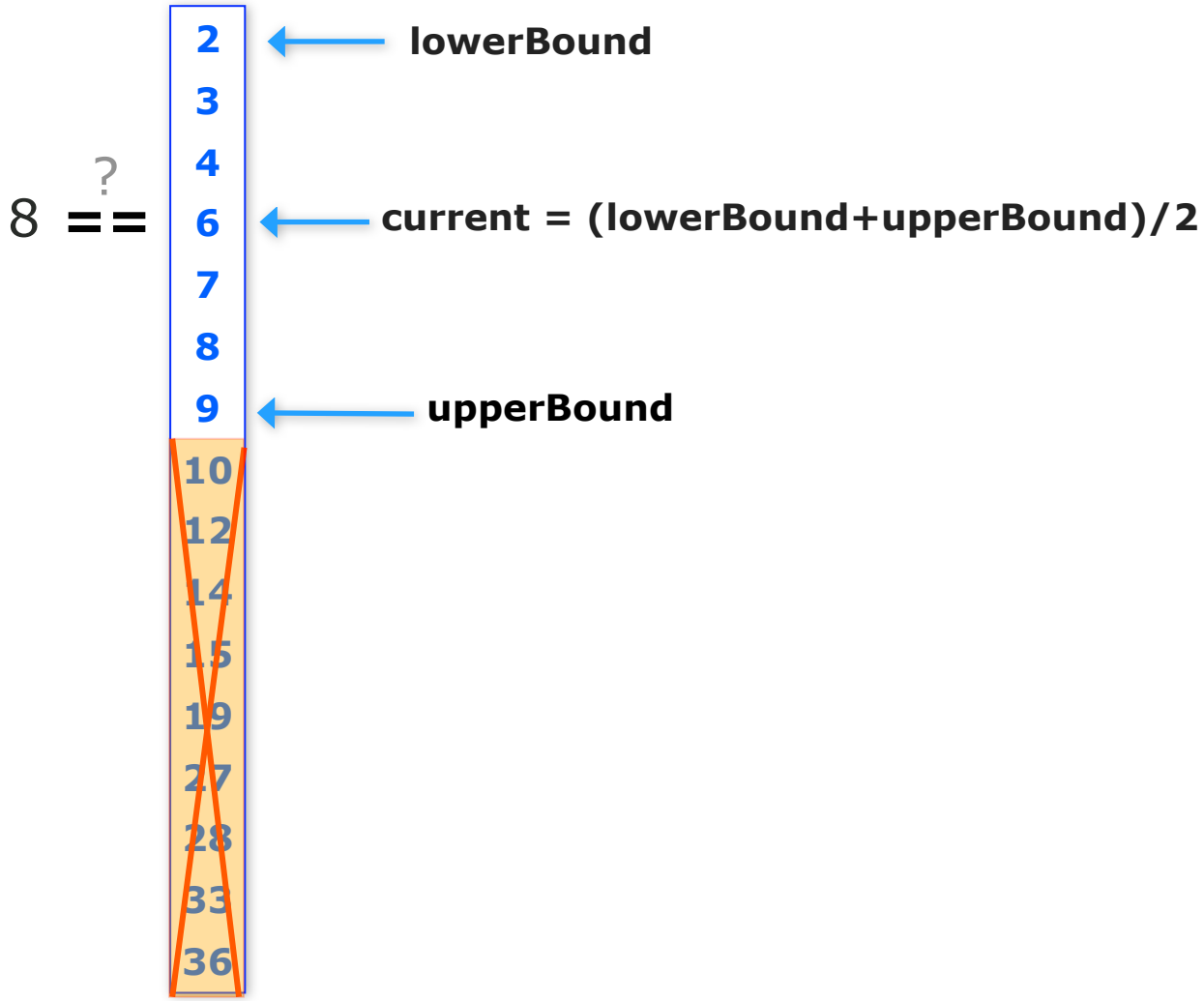
Freie Universität Berlin

Iterativ



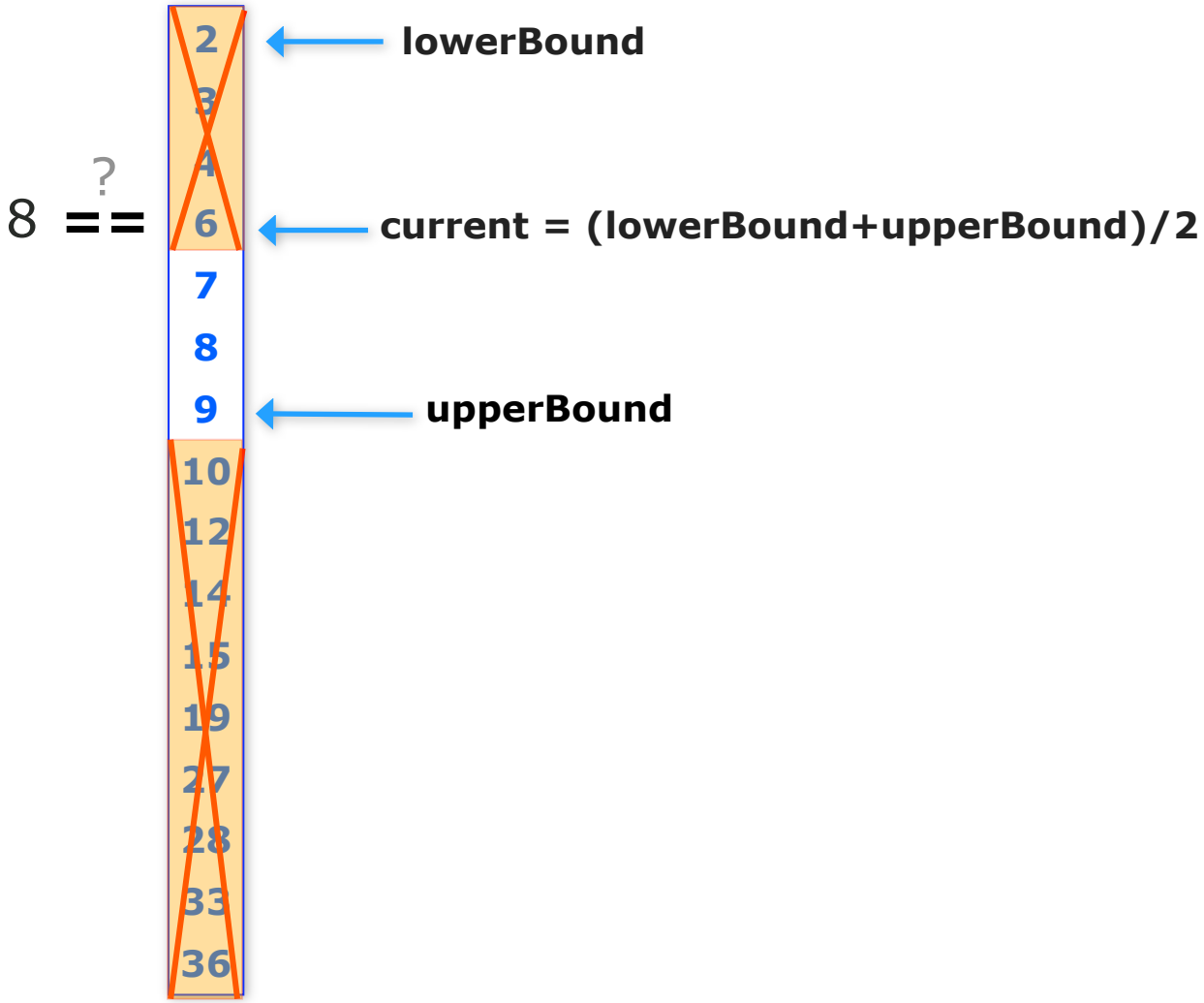
Freie Universität Berlin

Iterativ



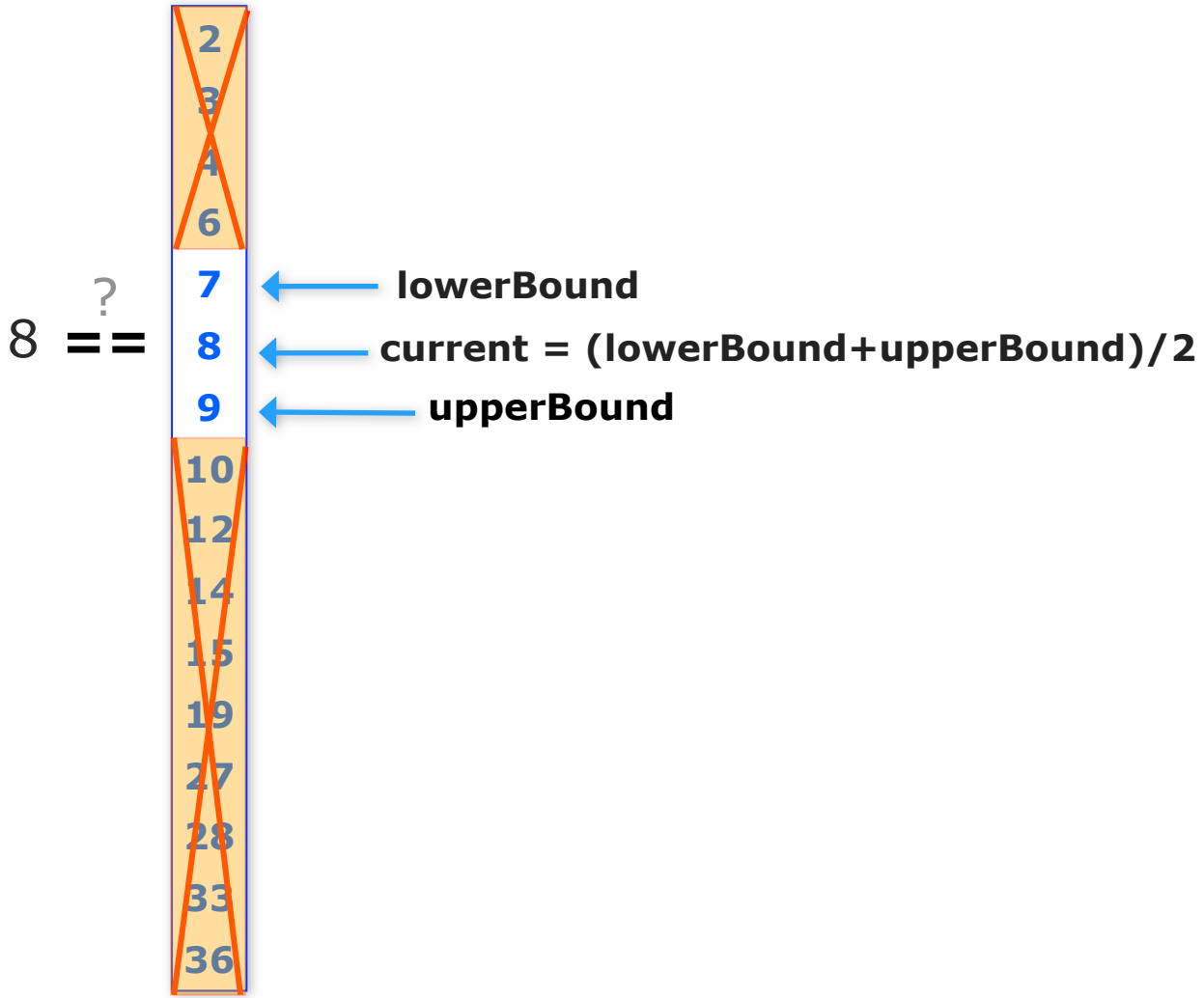
Freie Universität Berlin

Iterativ



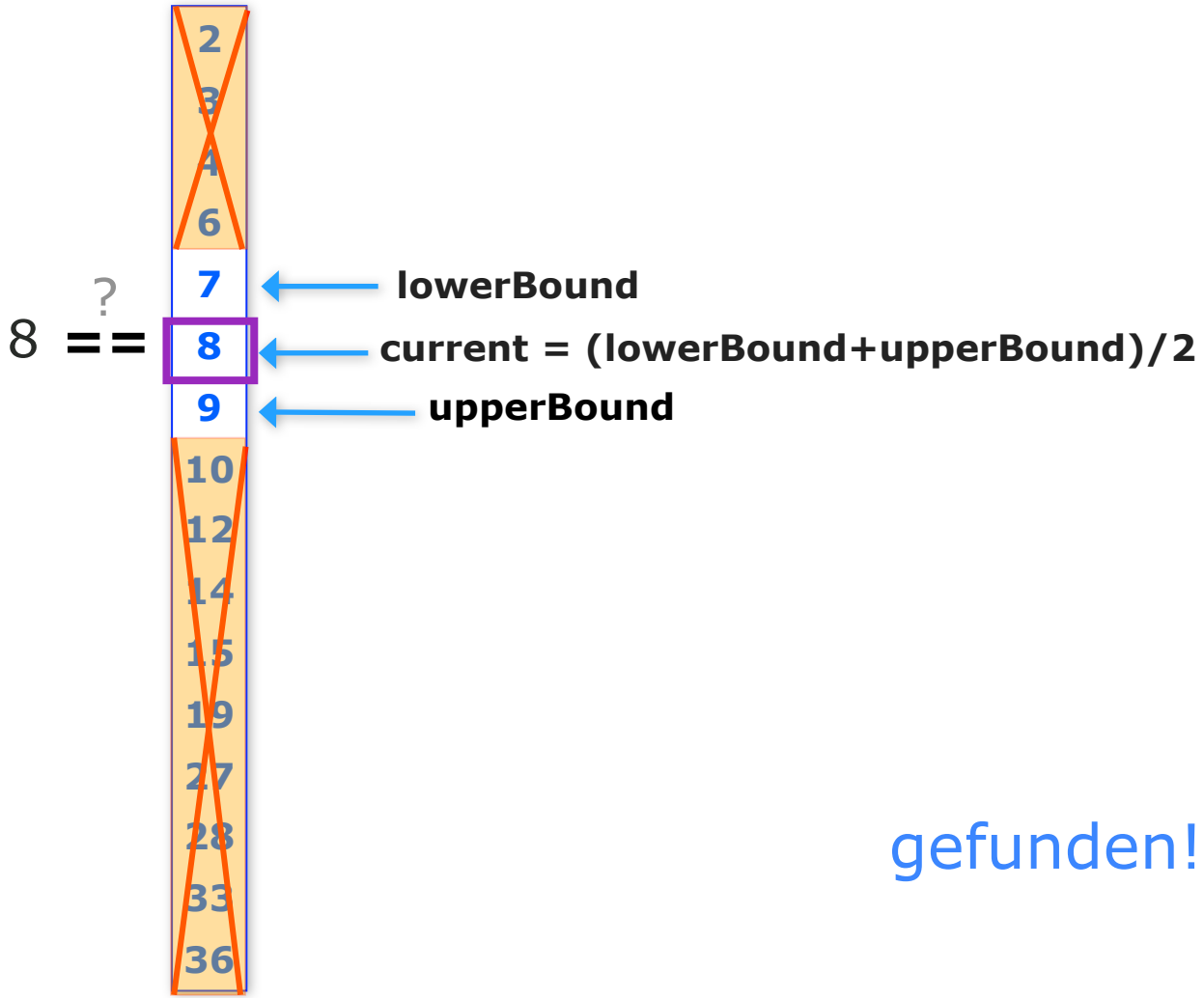
Freie Universität Berlin

Iterativ



Freie Universität Berlin

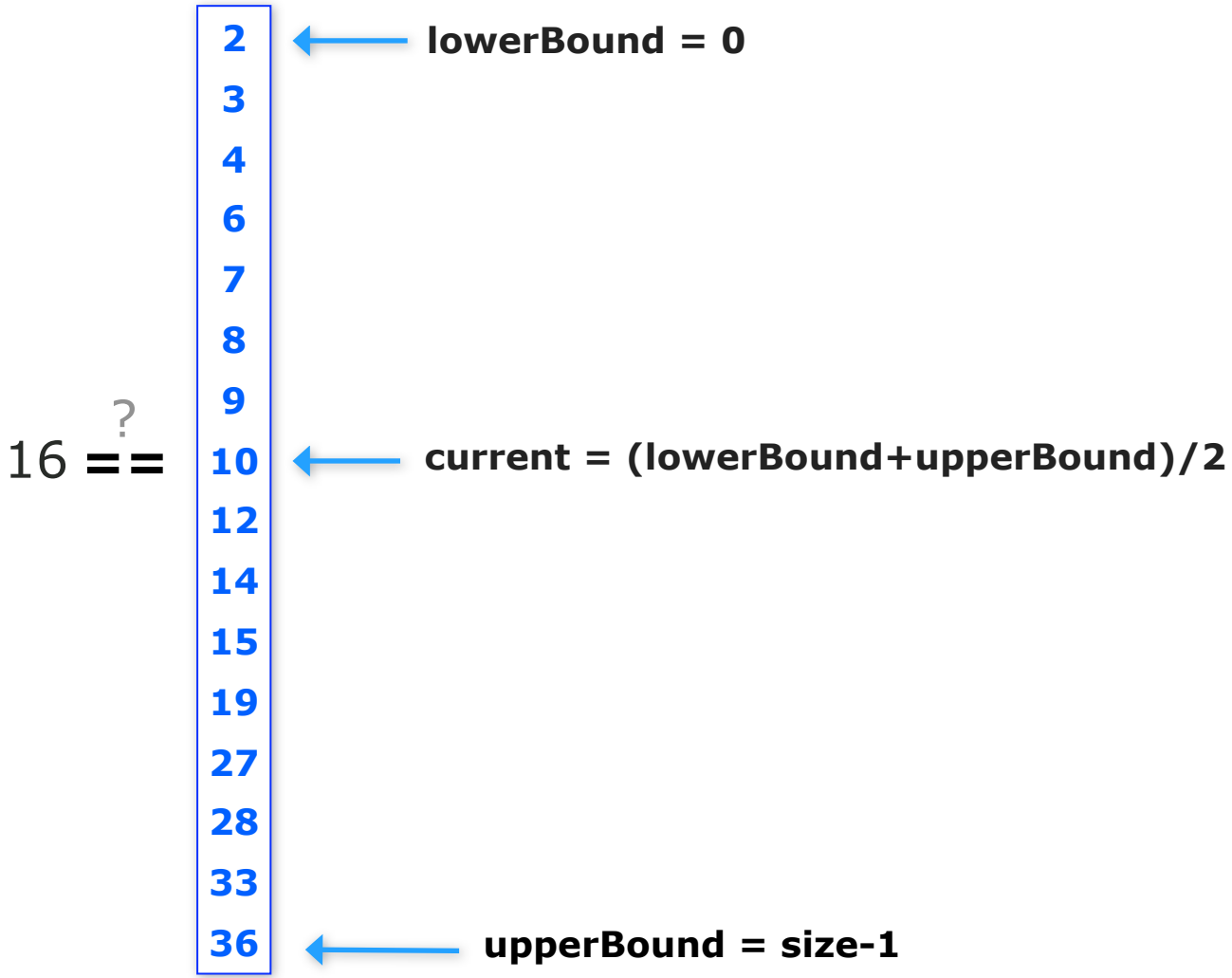
Iterativ



gefunden!

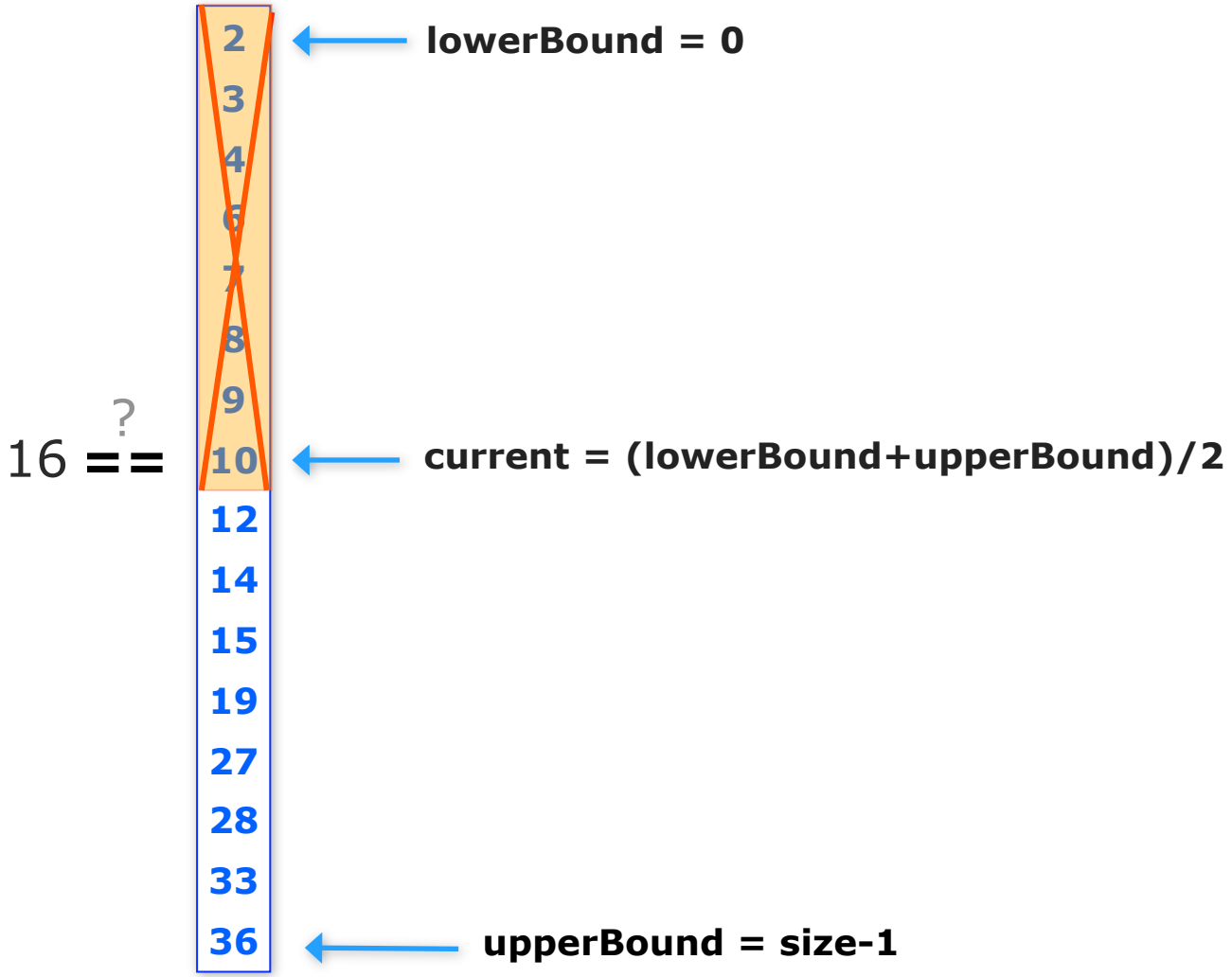
Freie Universität Berlin

Iterativ



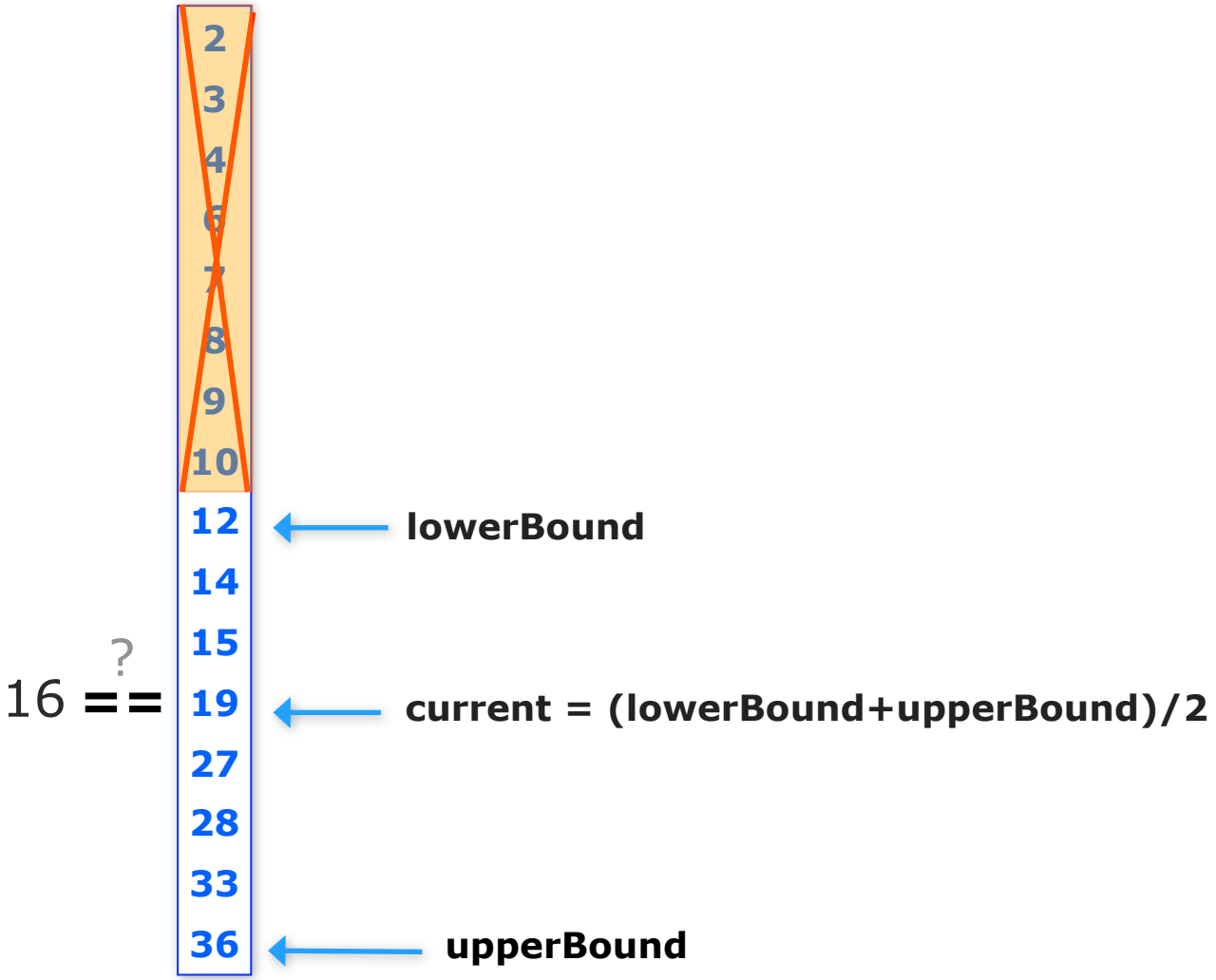
Freie Universität Berlin

Iterativ



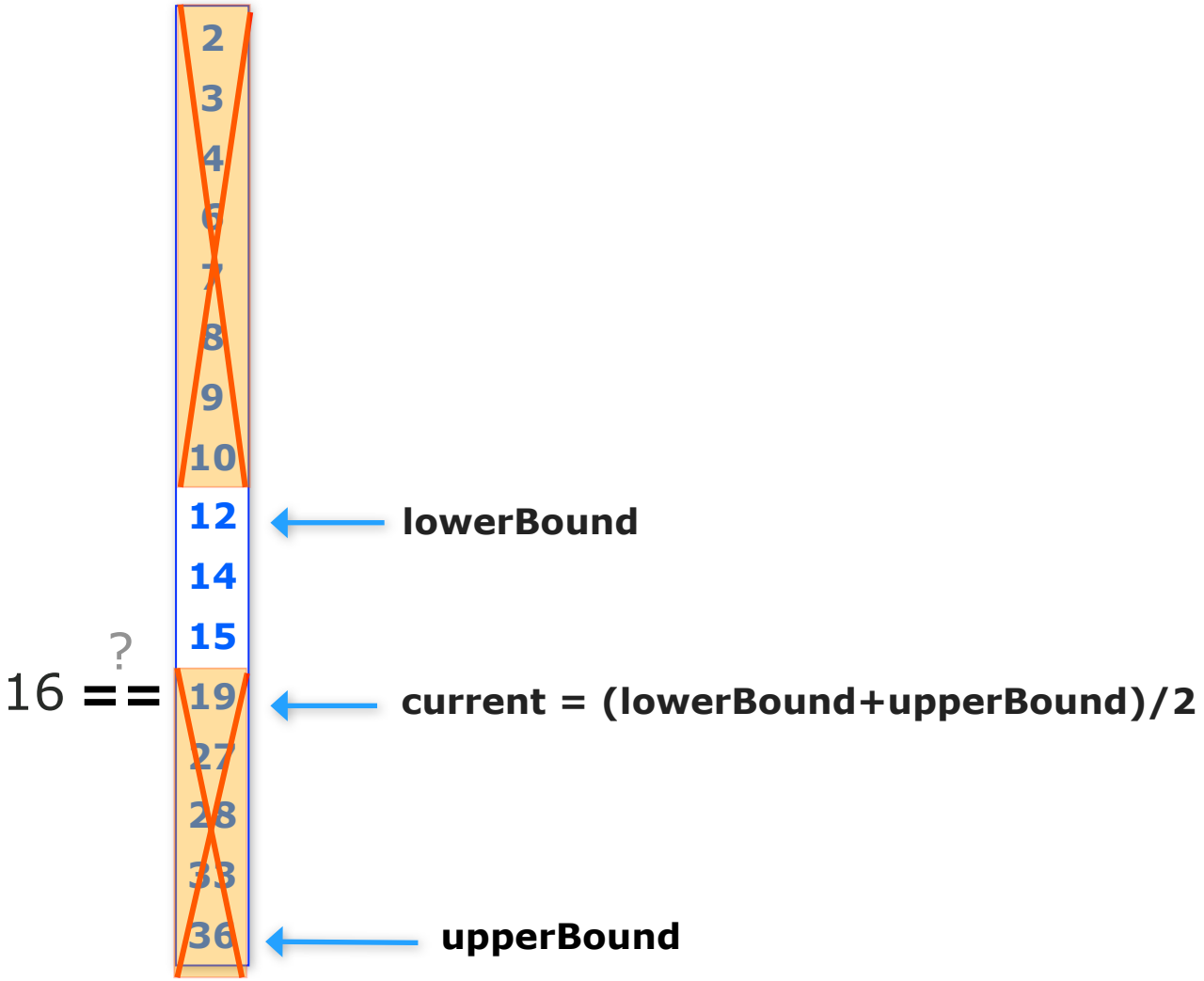
Freie Universität Berlin

Iterativ



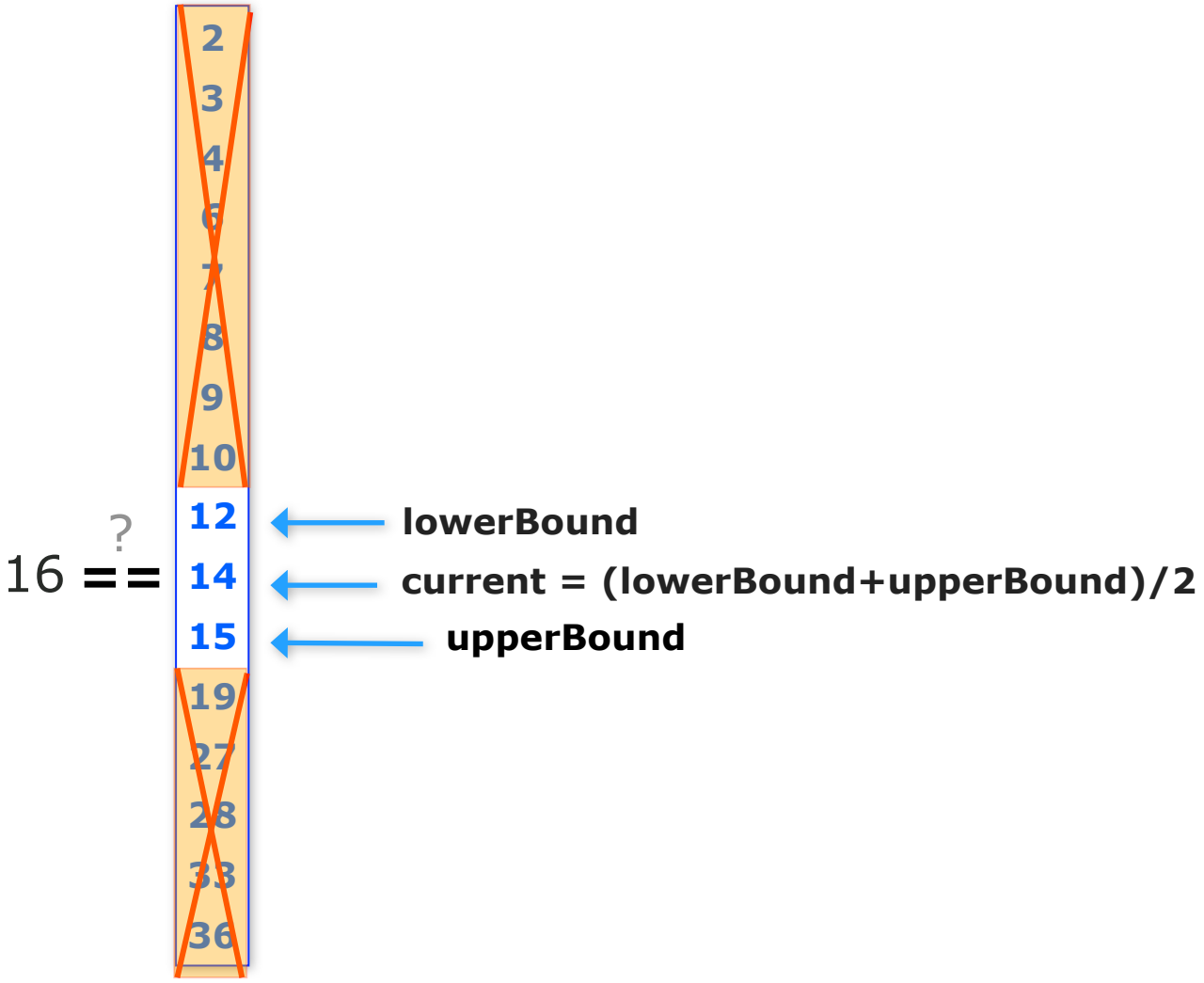
Freie Universität Berlin

Iterativ



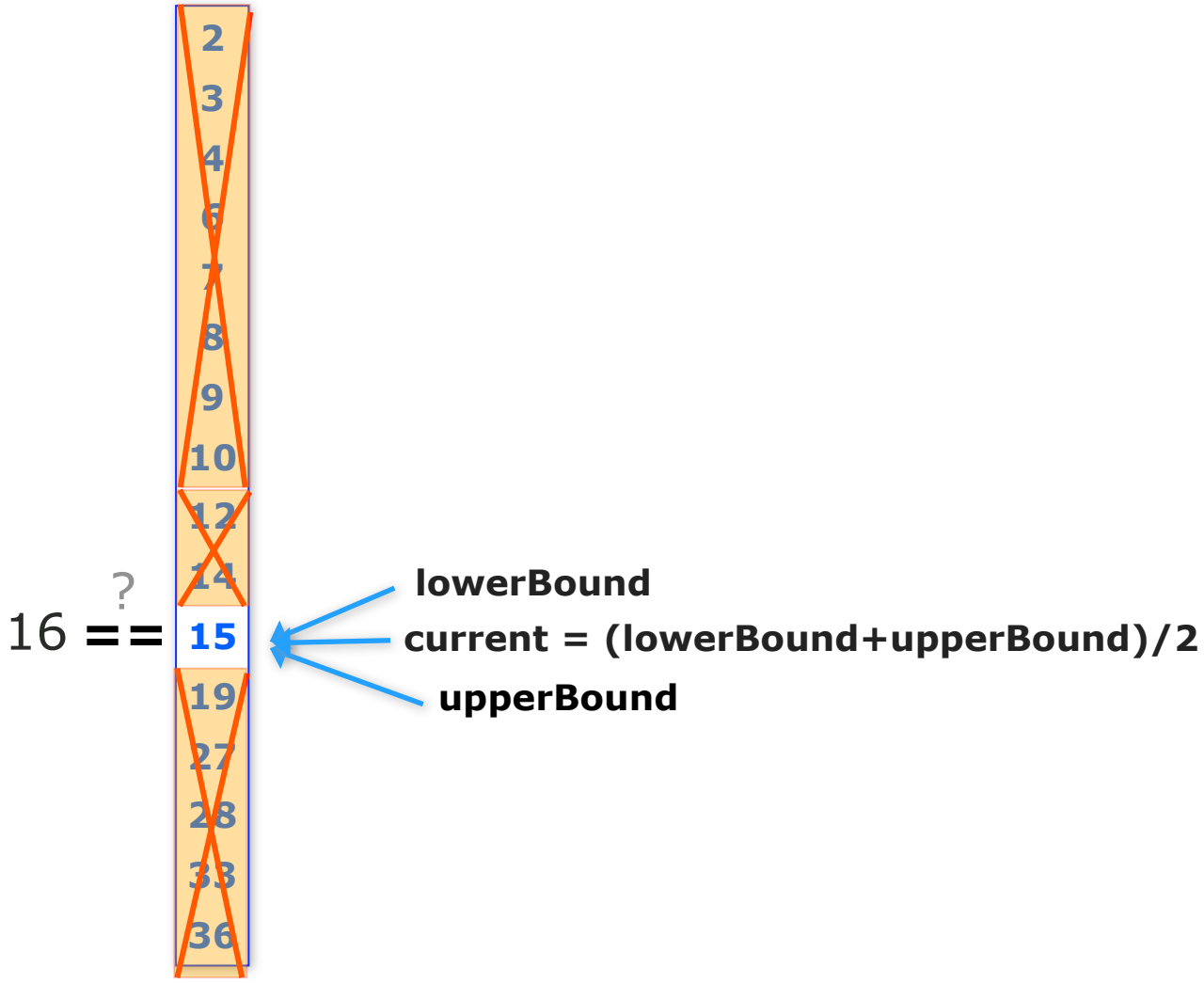
Freie Universität Berlin

Iterativ



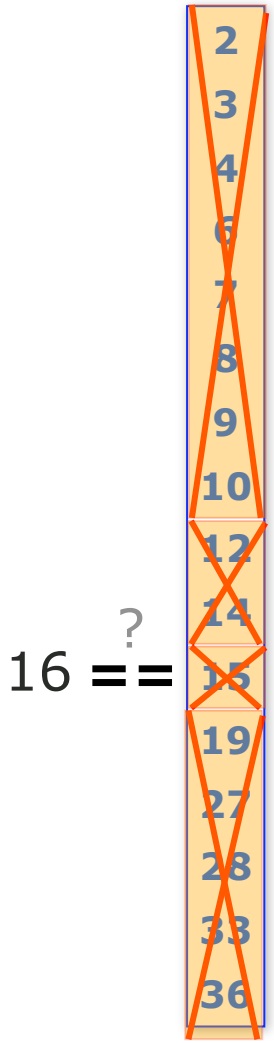
Freie Universität Berlin

Iterativ



Freie Universität Berlin

Iterativ



upperBound

lowerBound

Wenn **lowerBound>upperBound**

ist die Zahl **nicht** gefunden worden!

Freie Universität Berlin

Binärsuche in einem Feld

Iterativ

```
def binary_search (nums, key):

    lowerBound = 0
    upperBound = len(nums) - 1

    while  lowerBound <= upperBound:
            current = (lowerBound + upperBound)//2
            if   nums[current] == key:
                    return True
            else:
                    if  nums[current] < key:
                            lowerBound = current + 1
                    else:
                            upperBound = current - 1
    return False
```