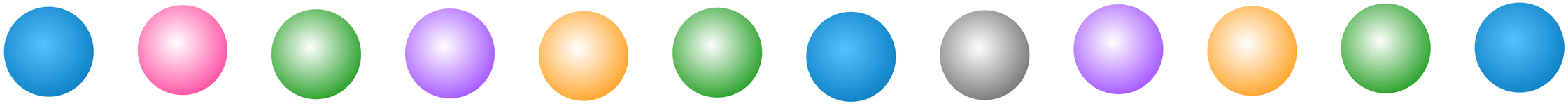
5. Sortieren und Suchen

5.2 Der Distributionsort

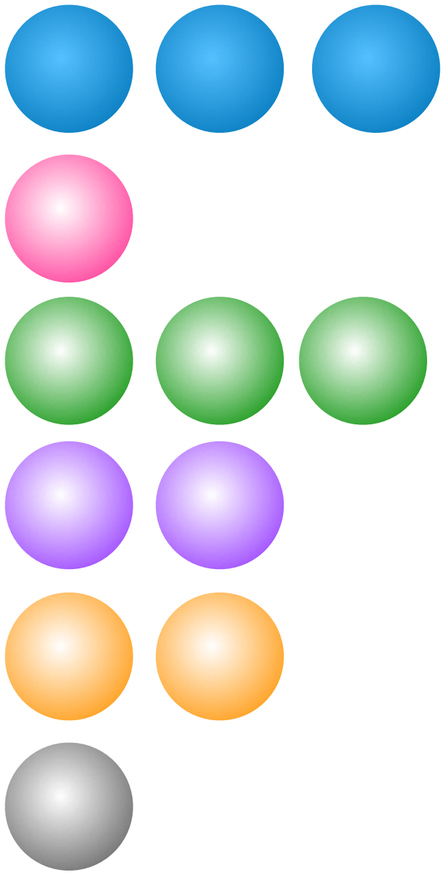
5.2 Distributionsort

Grundidee



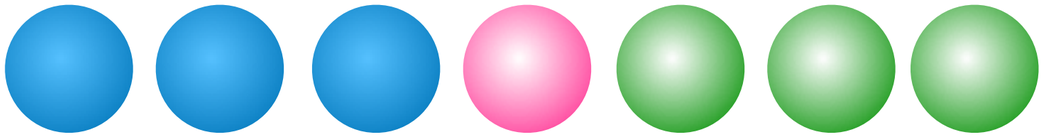
5.2 Distributionsort

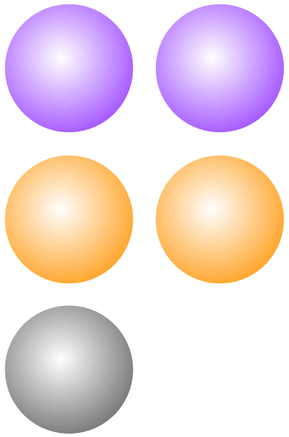
Grundidee



5.2 Distributionsort

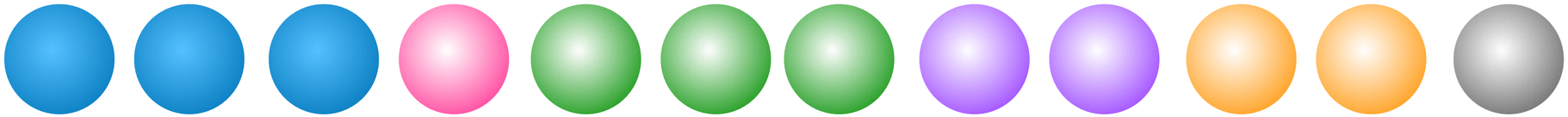
Grundidee





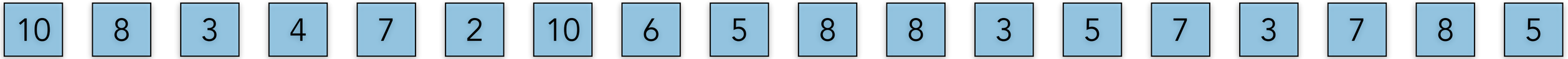
5.2 Distributionsort

Grundidee



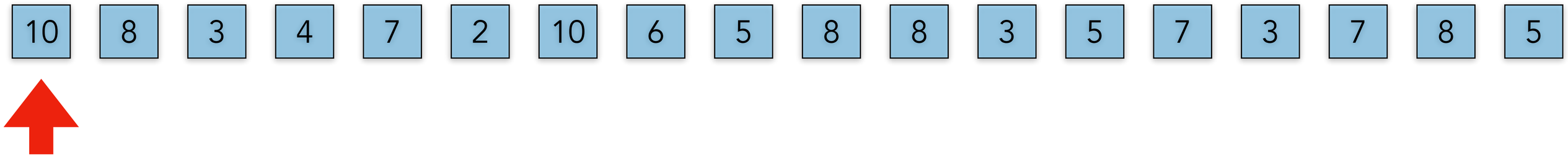
5.2 Distributionsort

Grundidee



5.2 Distributionsort

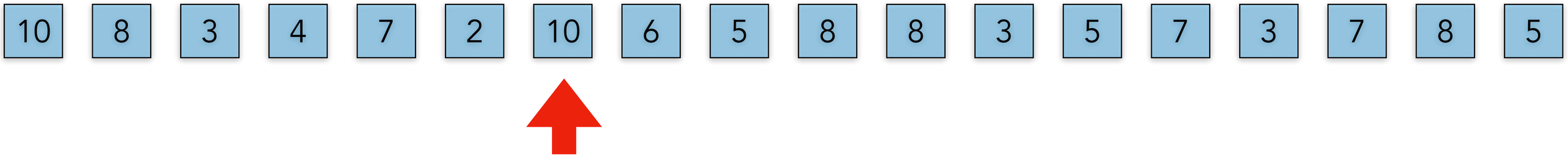
Grundidee



Zahl	1	2	3	4	5	6	7	8	9	10
Häufigkeit	0	0	0	0	0	0	0	0	0	1

5.2 Distributionsort

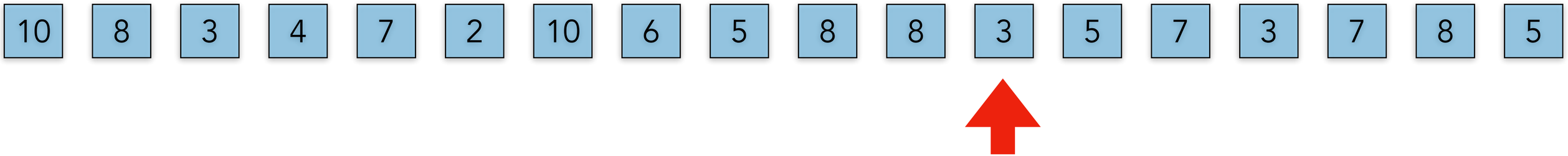
Grundidee



Zahl	1	2	3	4	5	6	7	8	9	10
Häufigkeit	0	1	1	1	0	0	1	1	0	2

5.2 Distributionsort

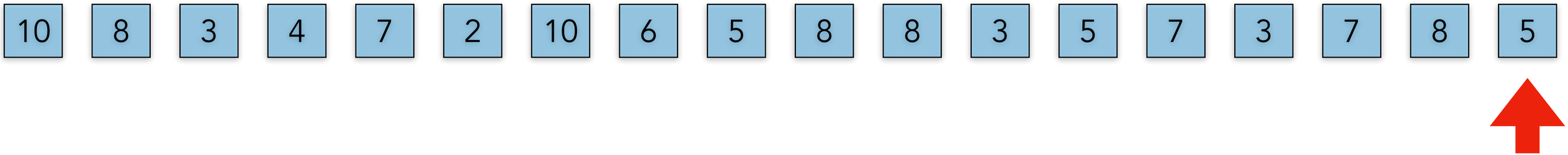
Grundidee



Zahl	1	2	3	4	5	6	7	8	9	10
Häufigkeit	0	1	2	1	1	1	1	3	0	2

5.2 Distributionsort

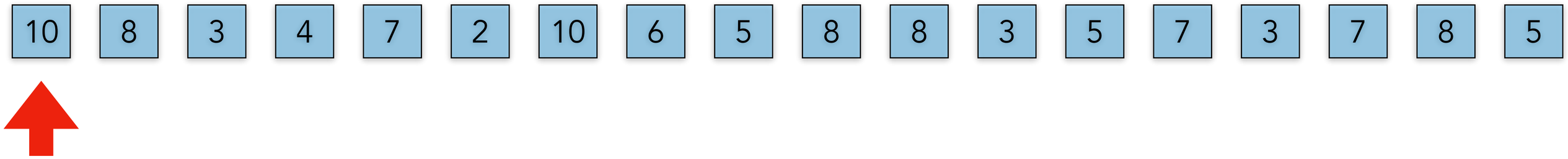
Grundidee



Zahl	1	2	3	4	5	6	7	8	9	10
Häufigkeit	0	1	3	1	2	1	3	4	0	2

5.2 Distributionsort

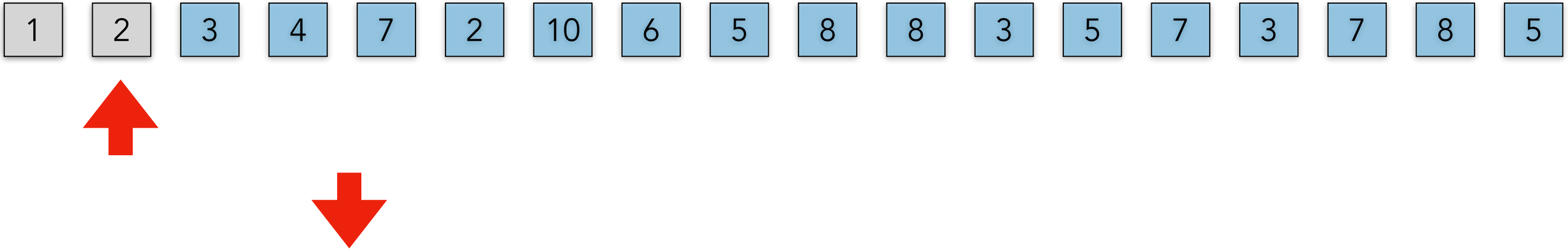
Grundidee



Zahl	1	2	3	4	5	6	7	8	9	10
Häufigkeit	0	1	3	1	2	1	3	4	0	2

5.2 Distributionsort

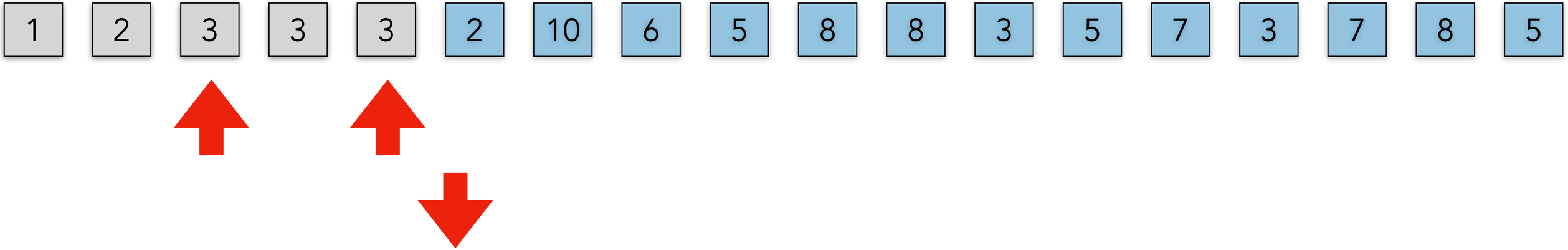
Grundidee



Zahl	1	2	3	4	5	6	7	8	9	10
Häufigkeit	0	1	3	1	2	1	3	4	0	2

5.2 Distributionsort

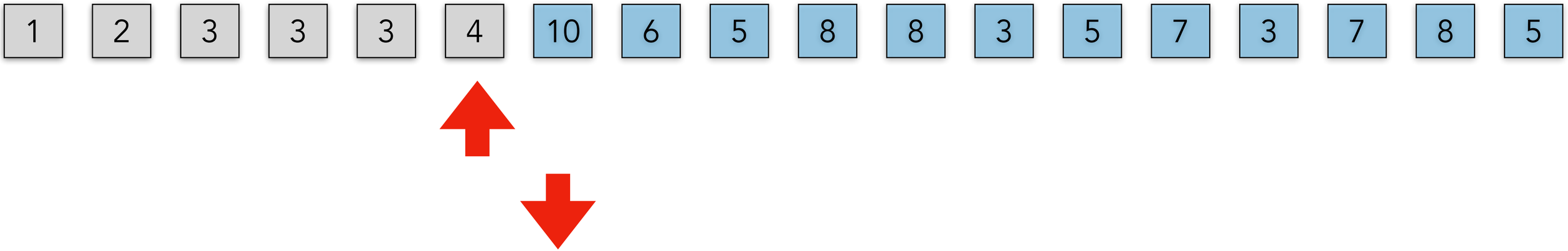
Grundidee



Zahl	1	2	3	4	5	6	7	8	9	10
Häufigkeit	0	1	3	1	2	1	3	4	0	2

5.2 Distributionsort

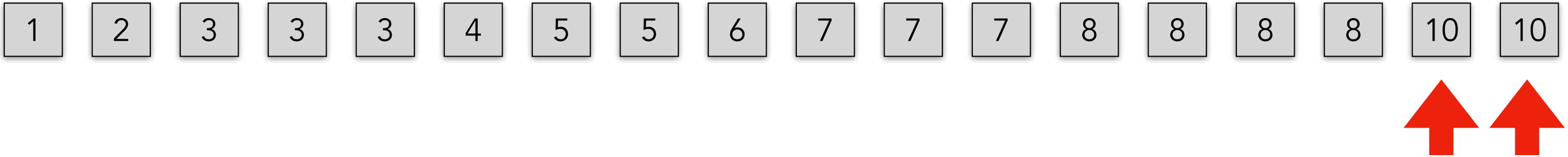
Grundidee



Zahl	1	2	3	4	5	6	7	8	9	10
Häufigkeit	0	1	3	1	2	1	3	4	0	2

5.2 Distributionsort

Grundidee



Zahl	1	2	3	4	5	6	7	8	9	10
Häufigkeit	0	1	3	1	2	1	3	4	0	2

5.2 Distributionsort

Java-Methode

```
public class NZahlen
{
    private static final int MAX = 1600;
    private static final int MAX_VAL = MAX/2;
    int[] zahlen, verteilung;

    public NZahlen()
    {
        zahlen = new int[MAX];
        verteilung = new int[MAX_VAL+1];
    }
```

Konstanten,
Instanzvariablen
Konstruktor

5.2 Distributionsort

Java-Methode

```
public void fuelleVerteilung()
{
    for (int i=0; i < MAX; i++)
    {
        verteilung[zahlen[i]]++;
    }
}
```

Verteilungs-Array füllen

Die Indices des Arrays **verteilung[ ]** sind Elemente des Arrays **zahlen[ ]**.

5.2 Distributionsort

Java-Methode

```
public void countingSort()
{
    int index = 0;
    fuelleVerteilung();

    for (int wert = 1; wert <= MAX_VAL; wert ++)
    {
        int anzahl = verteilung[wert];

        for (int i = 0; i < anzahl; i++)
        {
            zahlen[index] = wert;
            index++;
        }
    }
}
```

Verteilungs-Array

elementweise auslesen,

anzahl zuweisen.

5.2 Distributionsort

Java-Methode

```
public void countingSort()
{
    int index = 0;
    fuelleVerteilung();

    for (int wert = 1; wert <= MAX_VAL; wert ++)
    {
        int anzahl = verteilung[wert];

        for (int i = 0; i < anzahl; i++)
        {
            zahlen[index] = wert;
            index++;
        }
    }
}
```

Verteilungs-Array
elementweise auslesen,
anzahl zuweisen.

Zahlen-Array
Wert für Wert aufbauen

5.2 Distribut

Zeit-verhalten im Vergleich zu Bubble-sort

```
public void countingSortMitZaehlung()
{
    int index = 0;

    fuelleVerteilung();

    int operationen = MAX_VAL; // Array für Verteilung füllen
    operationen++;              // for-Schleife: Initialisierung

    for (int wert = 1; wert <= MAX_VAL; wert++)
    {
        operationen += 2;   // for-Schleife: Vergleich, Inkrement
        operationen += 1;   // die folgende Zuweisung

        int anzahl = verteilung[wert];

        operationen += 1;   // for-Schleife: Initialisierung

        // Fülle den Wert 'anzahl' mal in den Array
        for (int i = 0; i < anzahl; i++)
        operationen += 4;   // for-Schleife, zwei Zuweisungen

        {
            zahlen[index] = wert;
            index++;
        }
    }

    System.out.println("Für " + MAX + " Zahlen wurden " + operationen + " Operationen benötigt.");
}
```

5.2 Distributionsort

Zeitverhalten im Vergleich zu Bubblesort

MAX

```
 100 Zahlen     46623 Operationen.
 200 Zahlen    187872 Operationen.
 400 Zahlen    755862 Operationen.
 800 Zahlen   3036087 Operationen.
1600 Zahlen  12113724 Operationen.
```

Bubblesort

MAX

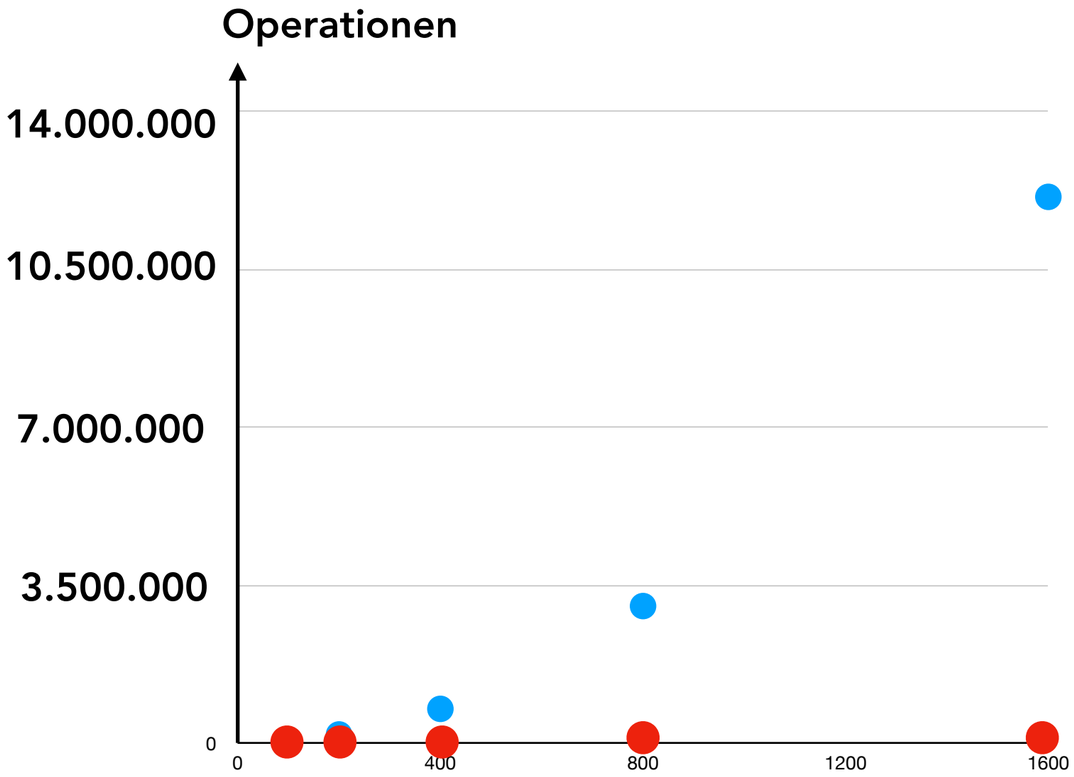
```
 100 Zahlen    651 Operationen.
 200 Zahlen   1301 Operationen.
 400 Zahlen   2601 Operationen.
 800 Zahlen   5201 Operationen.
1600 Zahlen  10401 Operationen.
```

Distributionsort

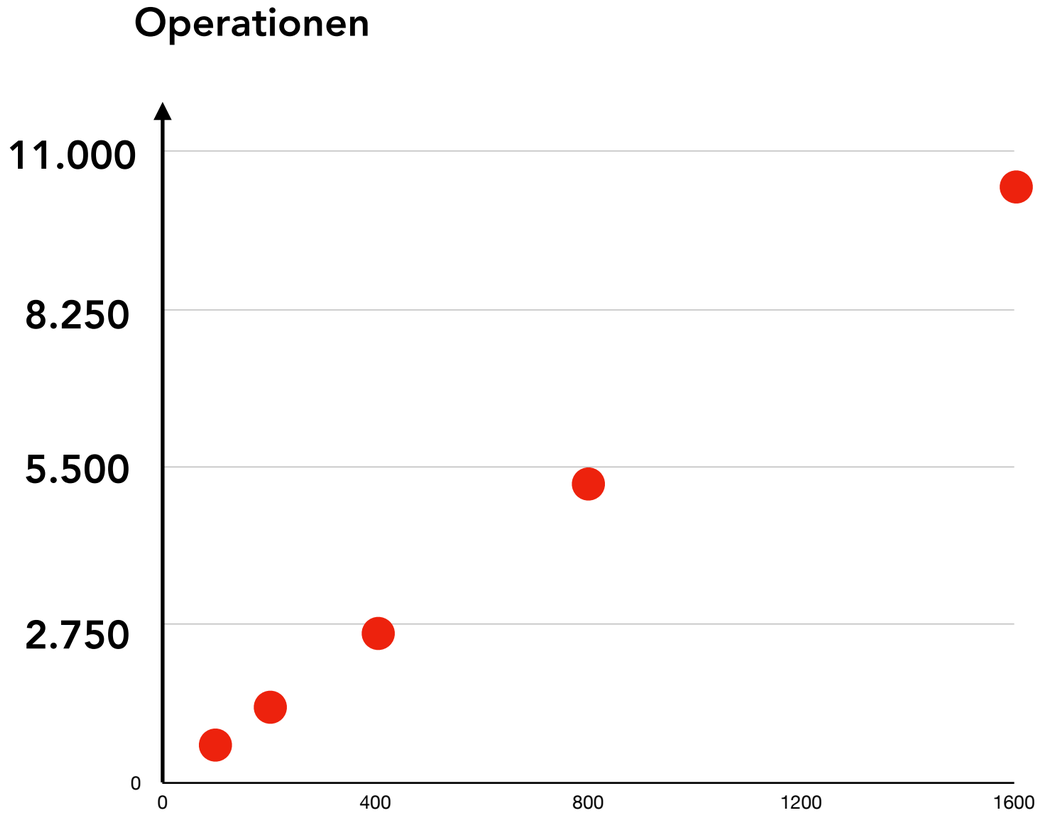
Wertebereich jeweils 1 ... MAX/2

5.2 Distributionsort

Zeitverhalten im Vergleich zu Bubblesort



Bubblesort



Distributionsort