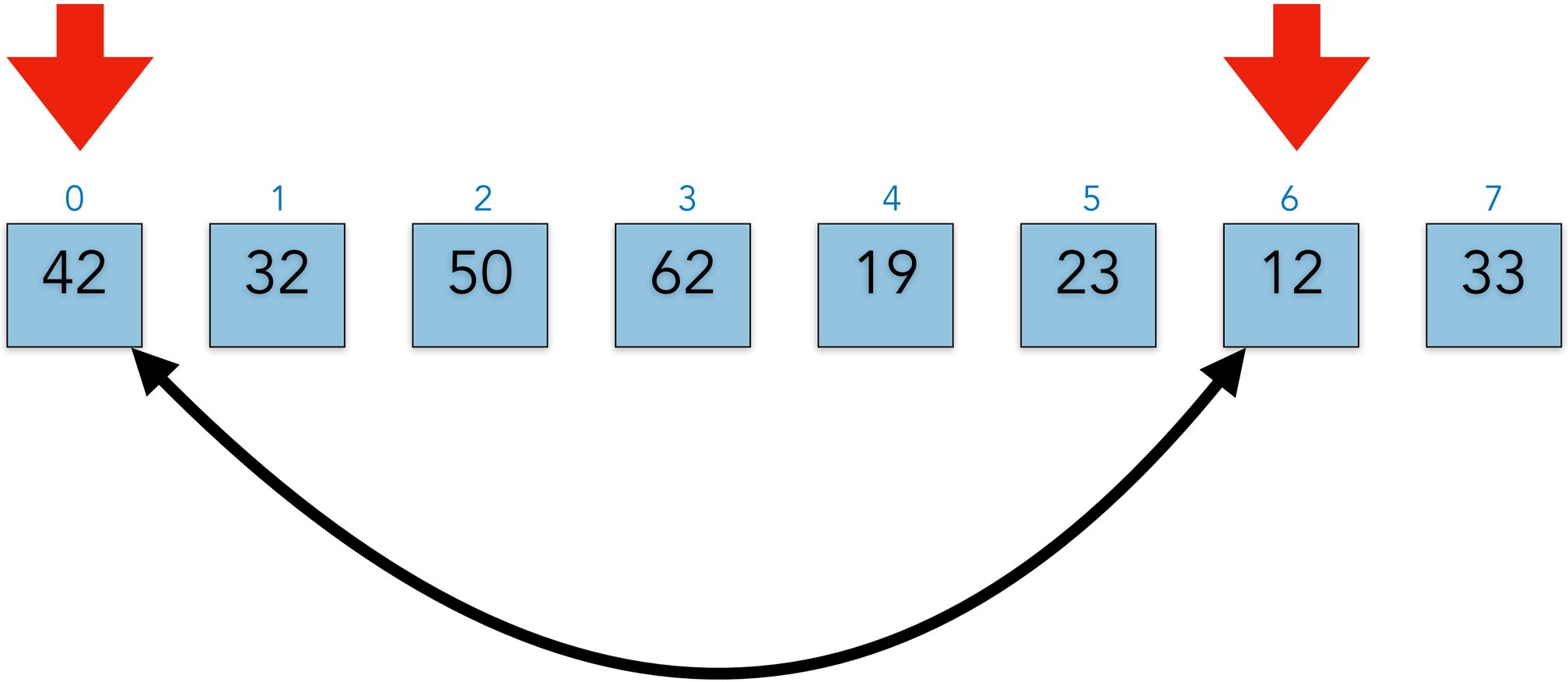
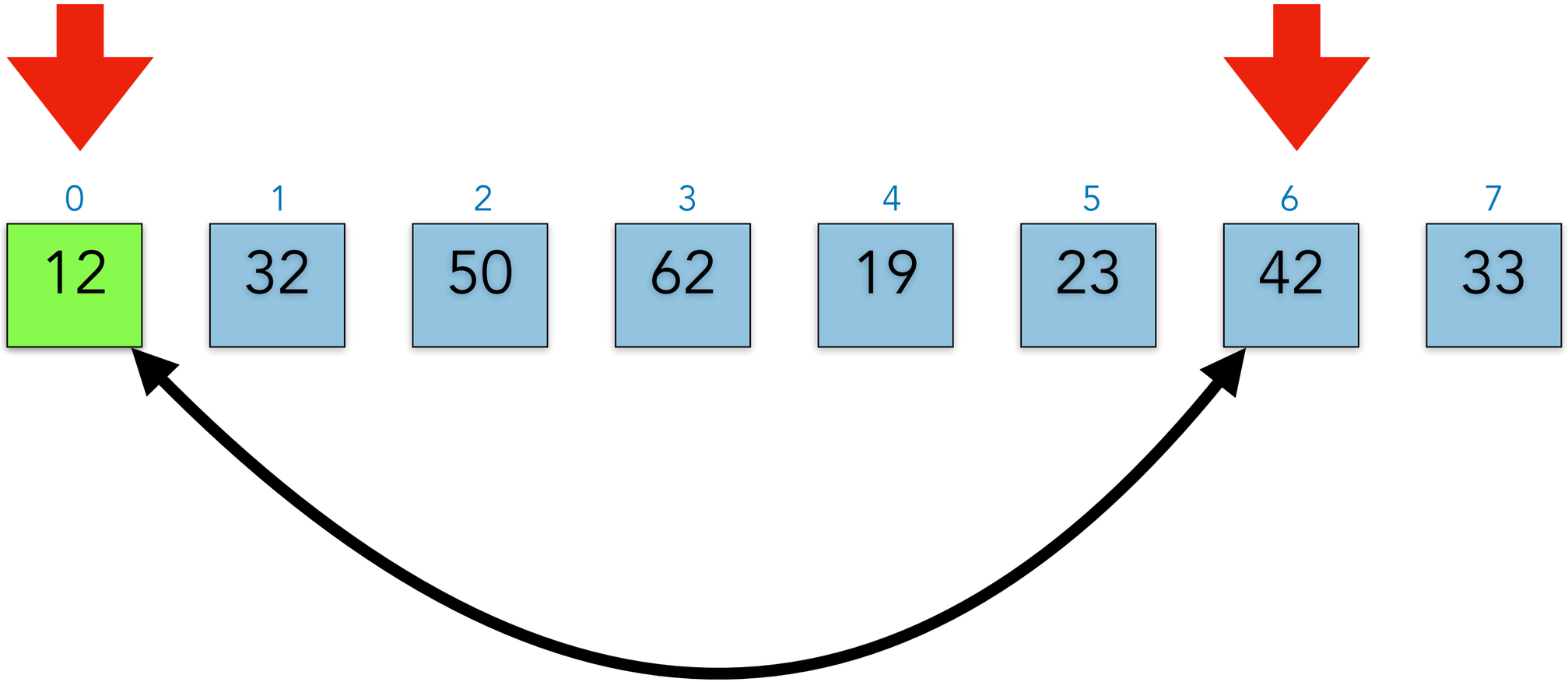
5. Sortieren und Suchen

5.3 Der Selectionsort

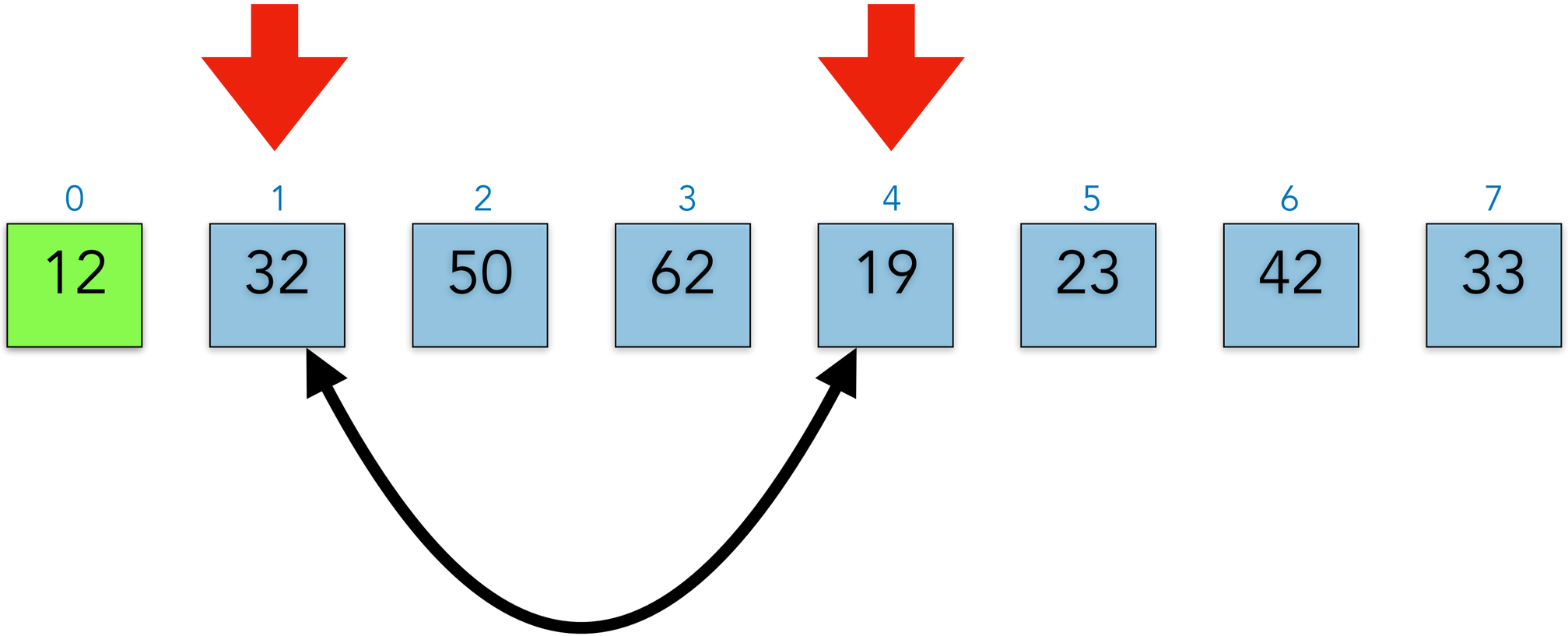
5.3 Selectionsort



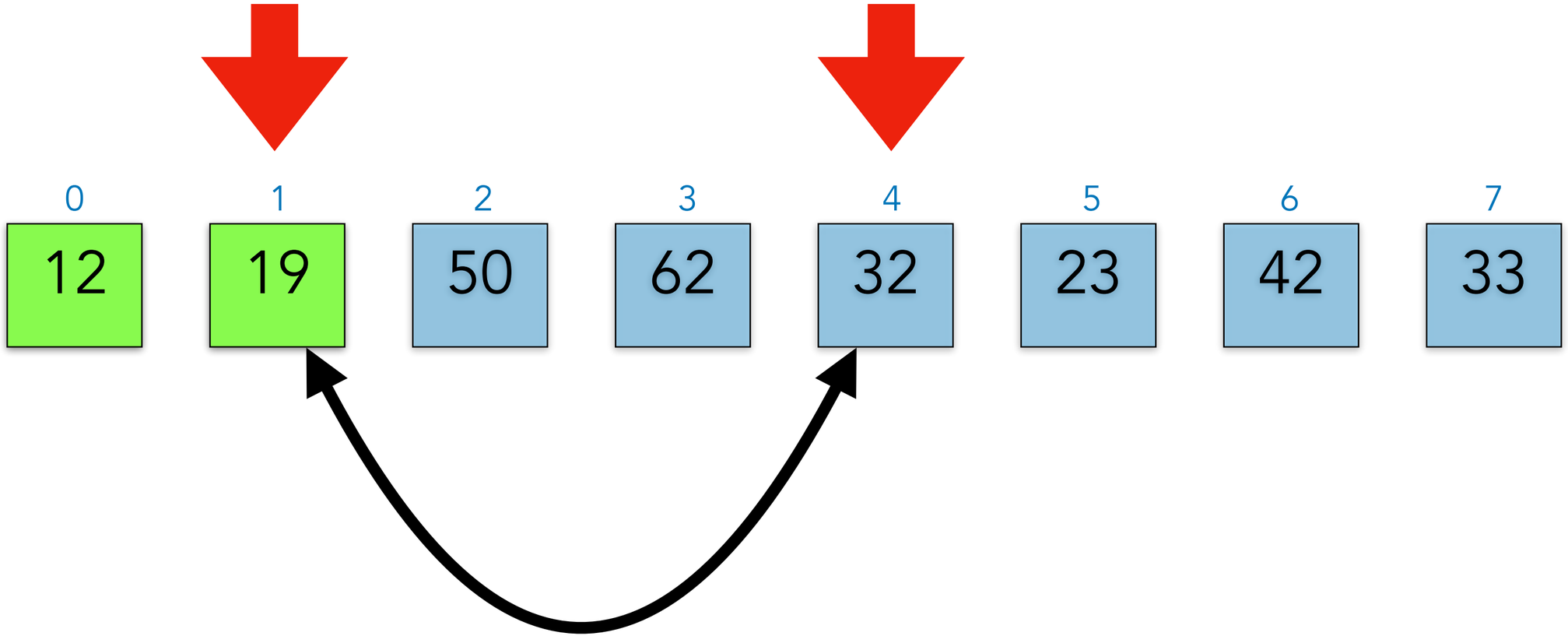
5.3 Selectionsort



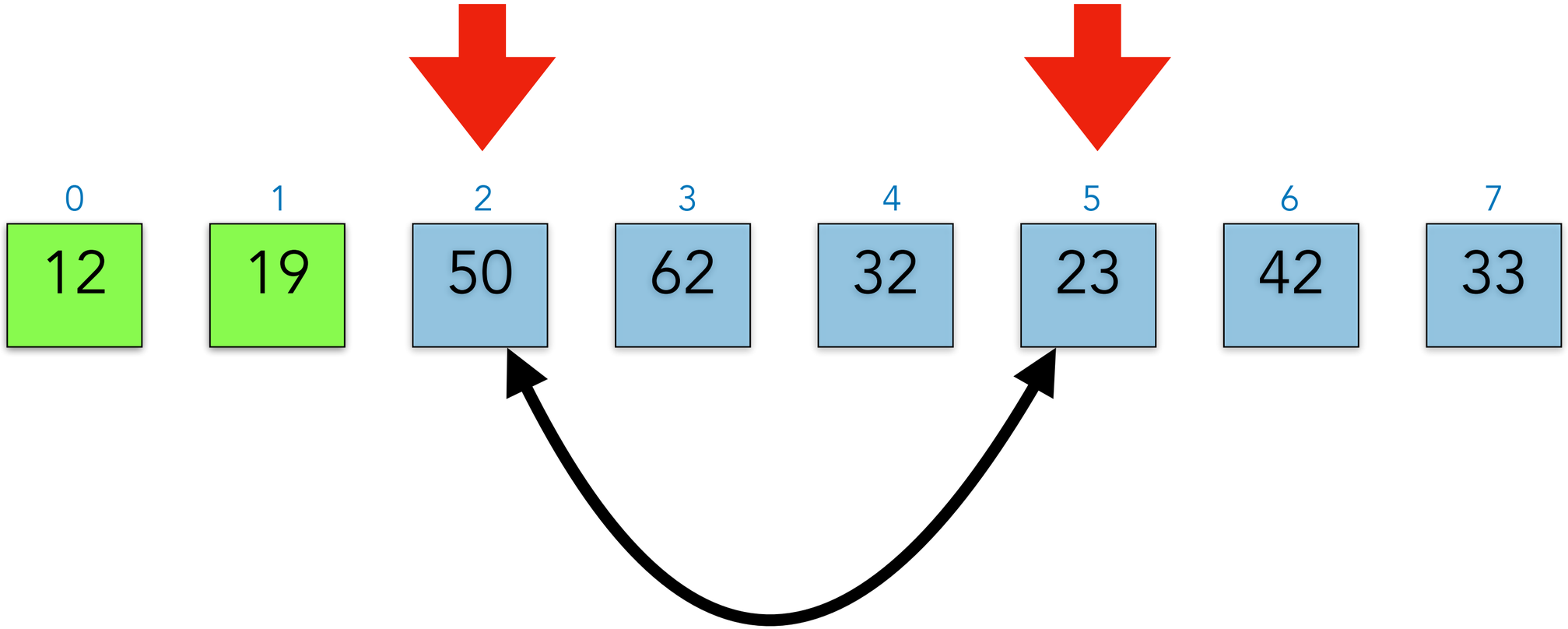
5.3 Selectionsort



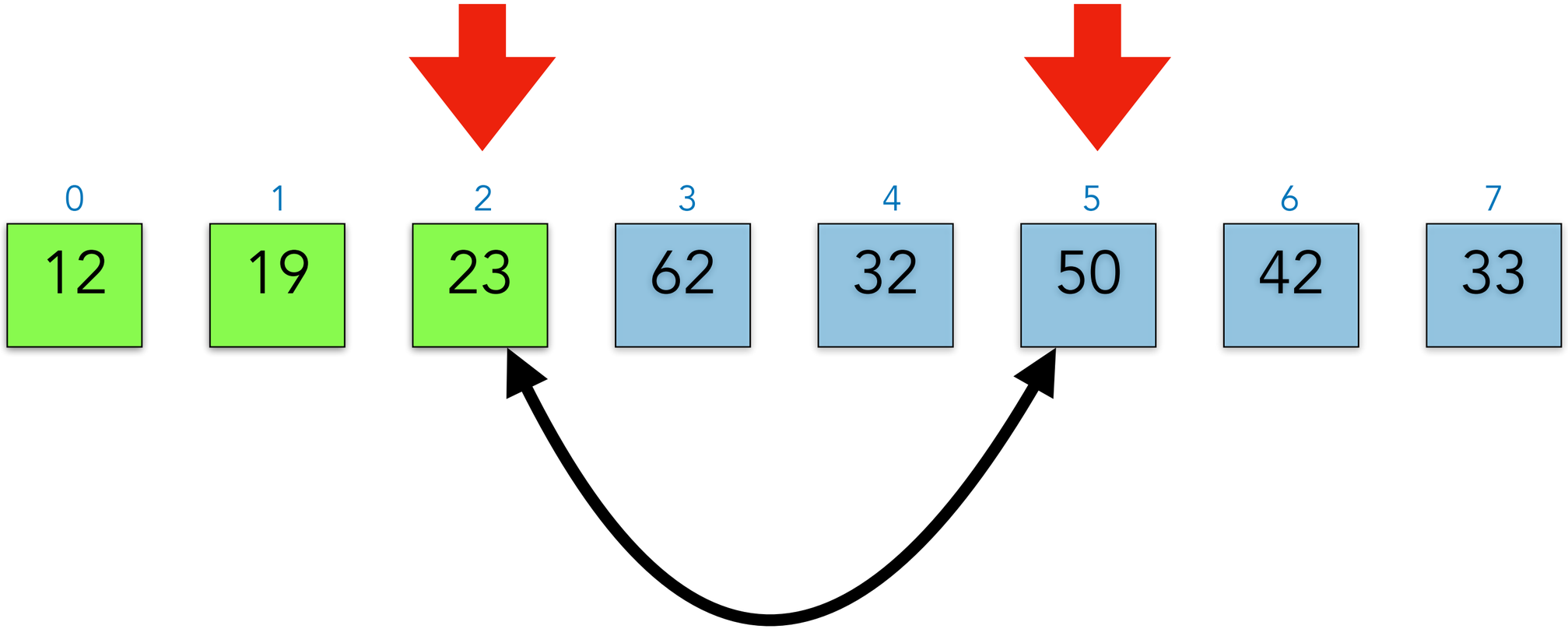
5.3 Selectionsort



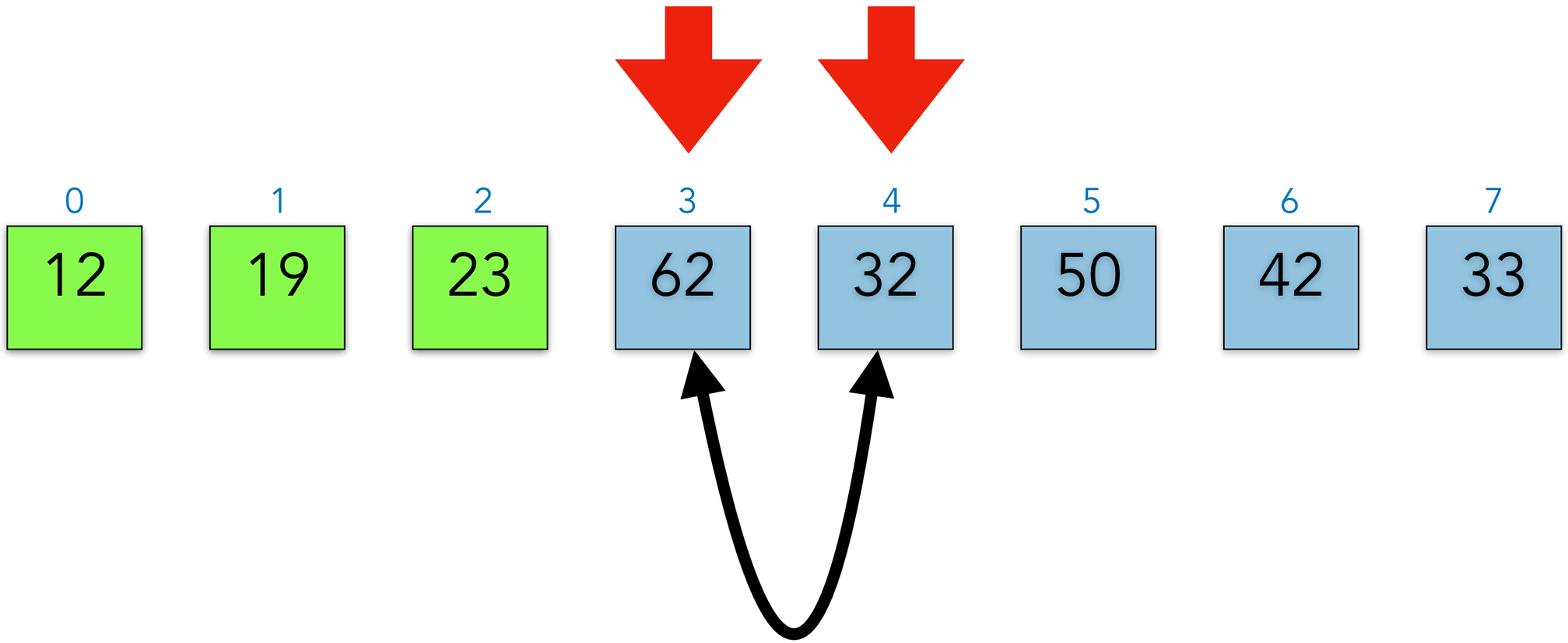
5.3 Selectionsort



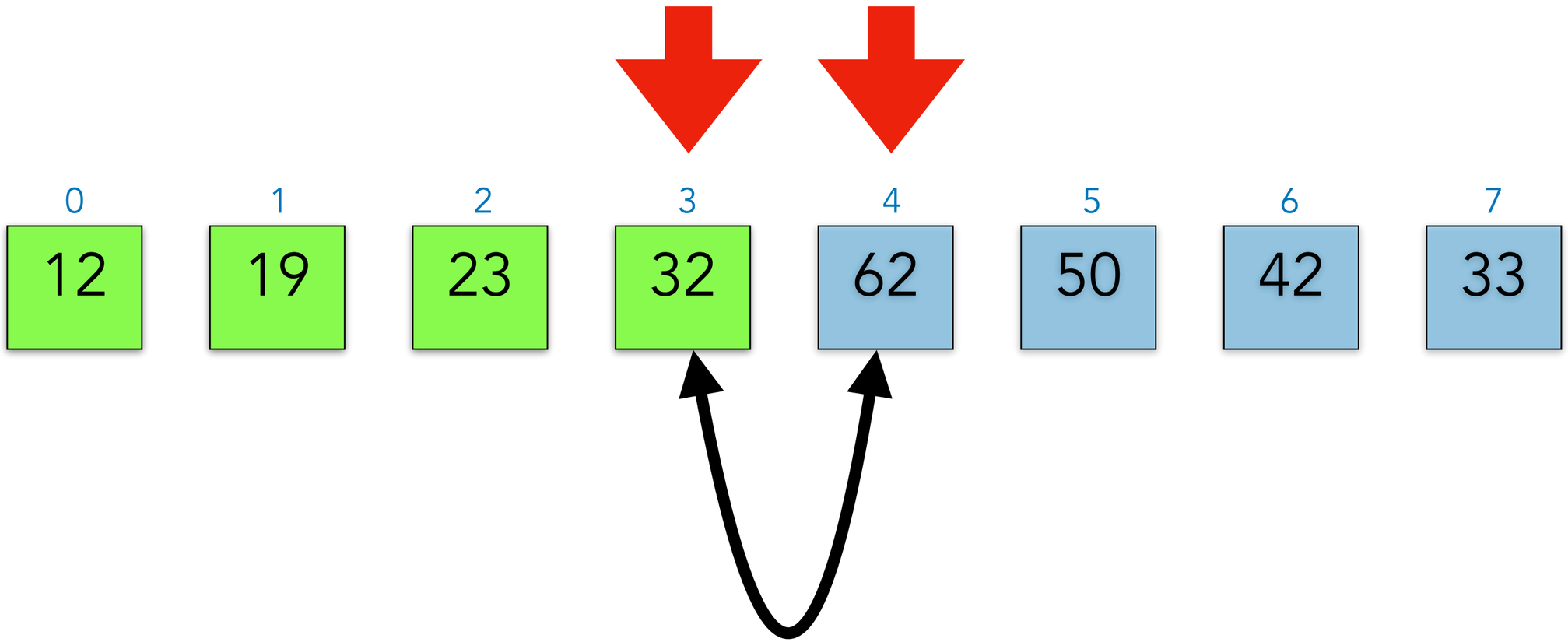
5.3 Selectionsort



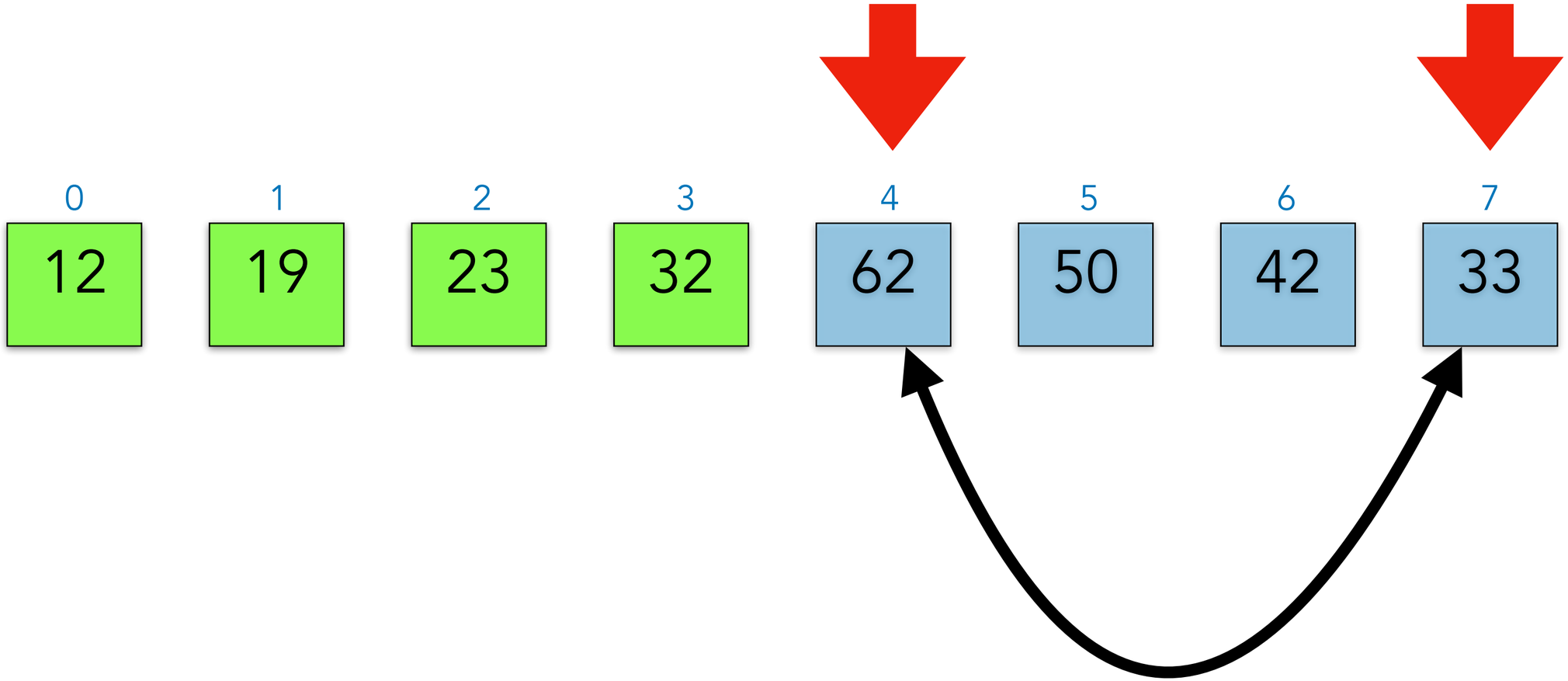
5.3 Selectionsort



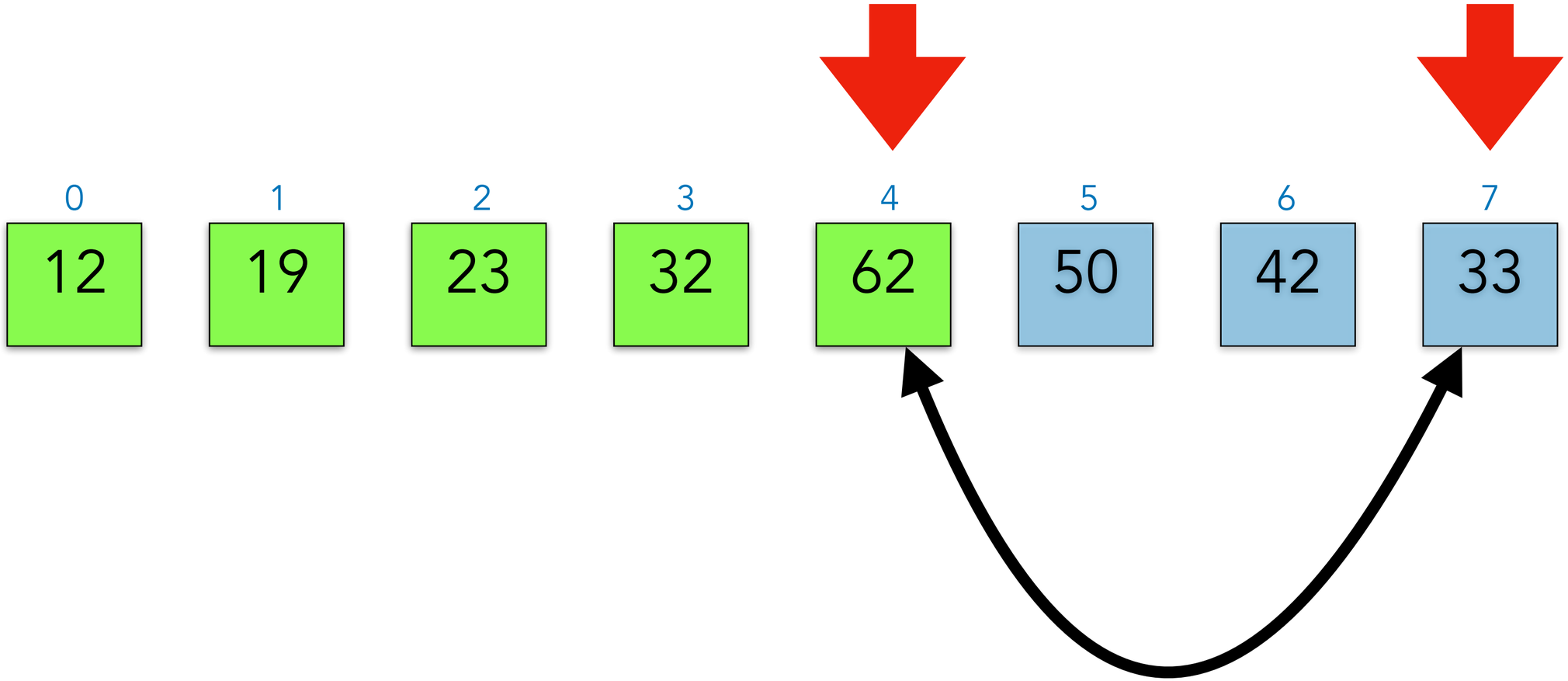
5.3 Selectionsort



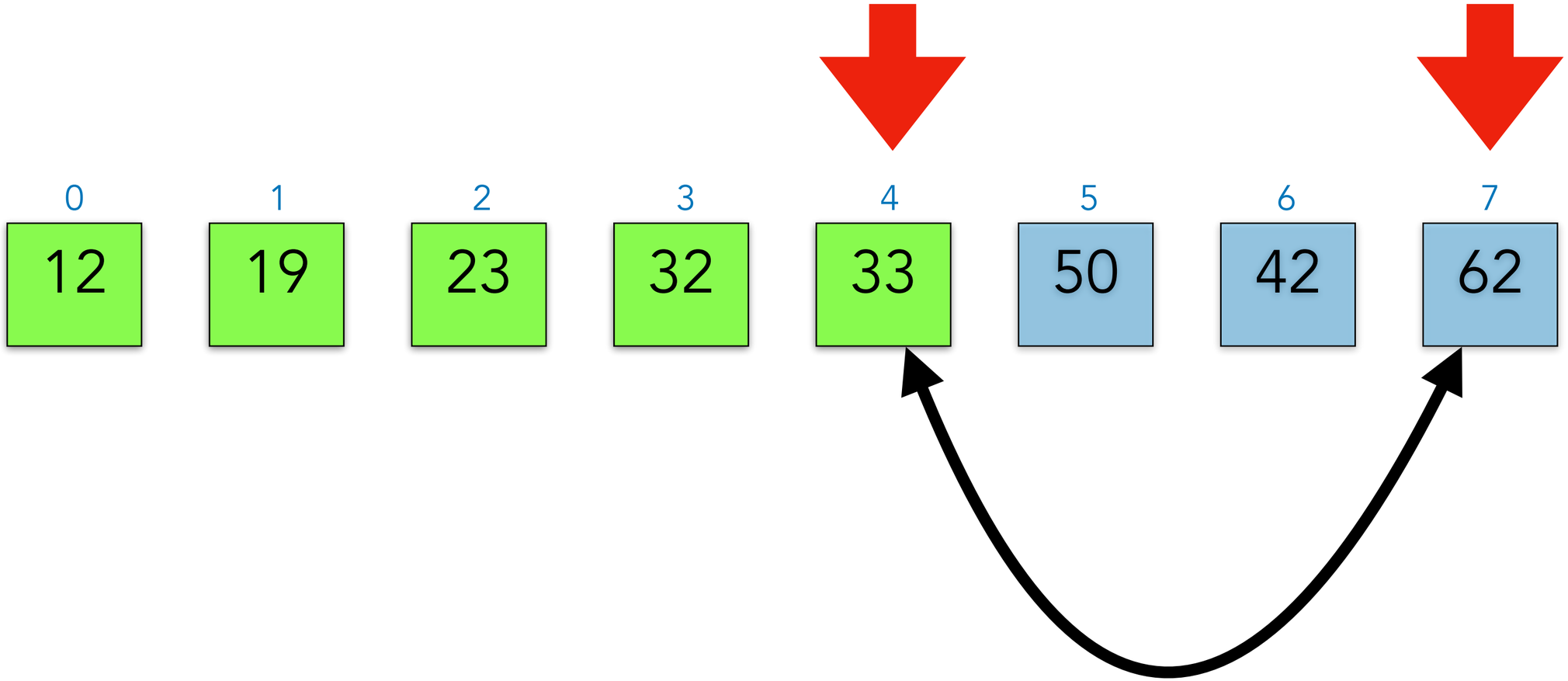
5.3 Selectionsort



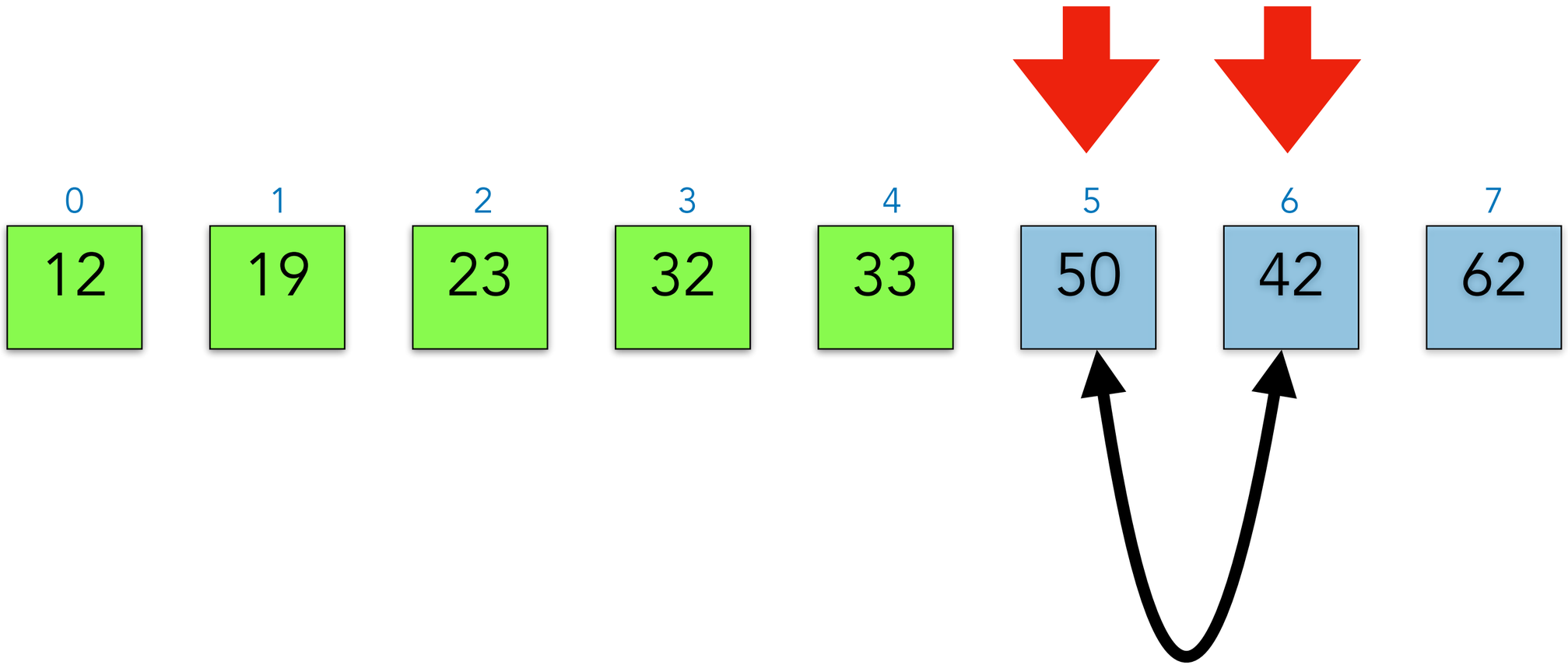
5.3 Selectionsort



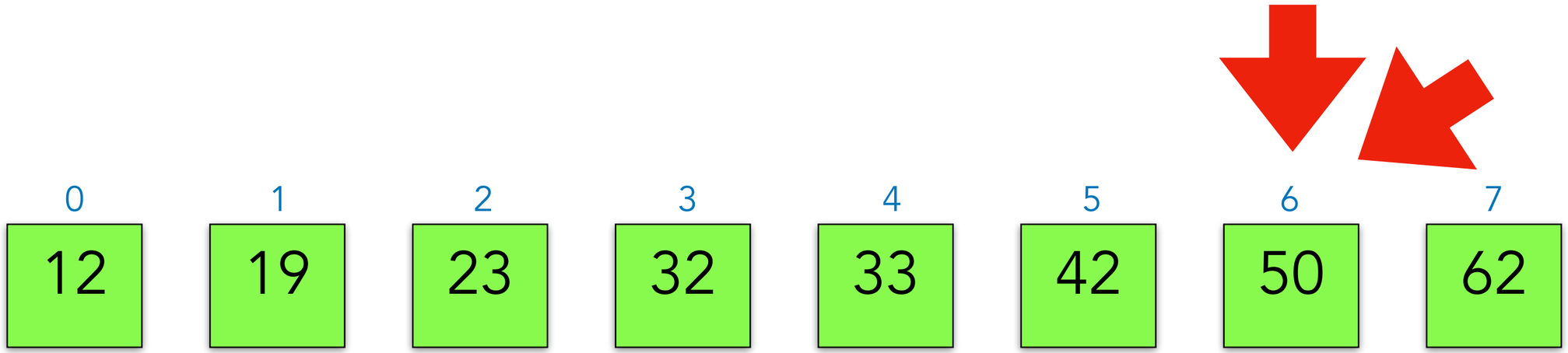
5.3 Selectionsort



5.3 Selectionsort



5.3 Selectionsort



5.3 Selectionsort

```
public void selectionSort()
{
    for (int i=0; i < zahlen.length-1; i++)
        tausche(i,getMiniPos(i));
}
```

5.3 Selectionsort

```
private int getMiniPos(int startIndex)
{
    int miniPos = startIndex;

    for (int i = startIndex+1; i < zahlen.length; i++)
    {
        if (zahlen[i] < zahlen[miniPos])
        {
            miniPos = i;
        }
    }
    return miniPos;
}
```

```
private void tausche(int i, int j)
{
    int temp = zahlen[i];
    zahlen[i] = zahlen[j];
    zahlen[j] = temp;
}
```

```
public void selectionSort()
{
    for (int i=0; i < zahlen.length-1; i++)
        tausche(i,getMiniPos(i));
}
```

5.3 Selectionsort

```
private int getMiniPos(int startIndex)
{
    int miniPos = startIndex;

    for (int i = startIndex+1; i < zahlen.length; i++)
    {
        if (zahlen[i] < zahlen[miniPos])
        {
            miniPos = i;
        }
    }
    return miniPos;
}
```

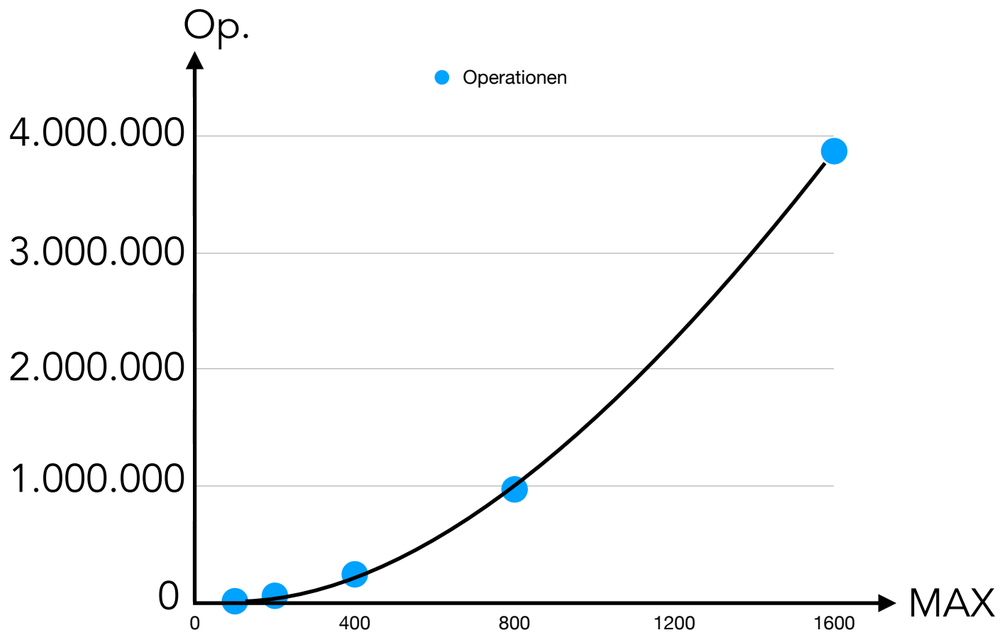
`startIndex` legt fest, ab welchem Index die Suche nach der kleinsten Zahl beginnen soll.

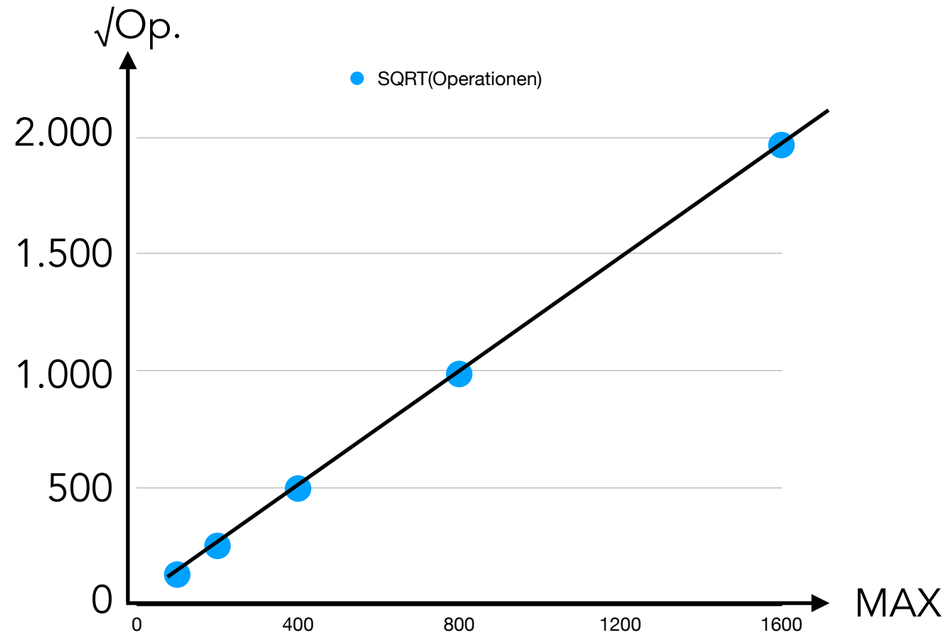
```
private void tausche(int i, int j)
{
    int temp = zahlen[i];
    zahlen[i] = zahlen[j];
    zahlen[j] = temp;
}

public void selectionSort()
{
    for (int i=0; i < zahlen.length-1; i++)
        tausche(i,getMiniPos(i));
}
```

5.3 Selectionsort

Selectionsort

```
 100 zahlen     16.415 Operationen
 200 zahlen     62.987 Operationen
 400 zahlen    246.677 Operationen
 800 zahlen    974.291 Operationen
1600 zahlen  3.871.203 Operationen
```





5.3 Selectionsort

Bubblesort: 28 Operationen

4 + 3 + 2 + 1 Vergleiche = 10 Operationen

3 + 2 + 1 + 0 Tauschvorgänge = 18 Operationen

Selectionsort: 16 Operationen

4 + 3 + 2 + 1 Vergleiche = 10 Operationen

1 + 0 + 1 + 0 Tauschvorgänge = 6 Operationen

5.3 Selectionsort

Bubblesort: 10 Operationen

4 + 3 + 2 + 1 Vergleiche = 10 Operationen
0 + 0 + 0 + 0 Tauschvorgänge = 0 Operationen

Selectionsort: 10 Operationen

4 + 3 + 2 + 1 Vergleiche = 10 Operationen
0 + 0 + 0 + 0 Tauschvorgänge = 0 Operationen

Bubblesort: 40 Operationen

4 + 3 + 2 + 1 Vergleiche = 10 Operationen
4 + 3 + 2 + 1 Tauschvorgänge = 30 Operationen

Selectionsort: 14 Operationen

4 + 3 + 2 + 1 Vergleiche = 10 Operationen
1 + 1 + 1 + 1 Tauschvorgänge = 4 Operationen

5.3 Selectionsort

Fazit 1:
Bei nahezu geordneten Zahlenfolgen sind beide Algorithmus ungefähr gleich schnell.

1	2	3	4	5

Bubblesort: 10 Operationen

4 + 3 + 2 + 1 Vergleiche = 10 Operationen
0 + 0 + 0 + 0 Tauschvorgänge = 0 Operationen

Selectionsort: 10 Operationen

4 + 3 + 2 + 1 Vergleiche = 10 Operationen
0 + 0 + 0 + 0 Tauschvorgänge = 0 Operationen

5	4	3	2	1

Bubblesort: 40 Operationen

4 + 3 + 2 + 1 Vergleiche = 10 Operationen
4 + 3 + 2 + 1 Tauschvorgänge = 30 Operationen

Selectionsort: 14 Operationen

4 + 3 + 2 + 1 Vergleiche = 10 Operationen
1 + 1 + 1 + 1 Tauschvorgänge = 4 Operationen

5.3 Selectionsort

Fazit 1:
Bei nahezu geordneten Zahlenfolgen sind beide Algorithmus ungefähr gleich schnell.

Bubblesort: 10 Operationen

4 + 3 + 2 + 1 Vergleiche = 10 Operationen
0 + 0 + 0 + 0 Tauschvorgänge = 0 Operationen

Selectionsort: 10 Operationen

4 + 3 + 2 + 1 Vergleiche = 10 Operationen
0 + 0 + 0 + 0 Tauschvorgänge = 0 Operationen

Fazit 2:
Je ungeordneter die Zahlenfolge ist, desto größer wird der Vorteil des Selectionsorts gegenüber dem Bubblesort.

Bubblesort: 40 Operationen

4 + 3 + 2 + 1 Vergleiche = 10 Operationen
4 + 3 + 2 + 1 Tauschvorgänge = 30 Operationen

Selectionsort: 14 Operationen

4 + 3 + 2 + 1 Vergleiche = 10 Operationen
1 + 1 + 1 + 1 Tauschvorgänge = 4 Operationen

5.3 Selectionsort

Zusammenfassung:
Je ungeordneter die Zahlenfolge ist, desto größer wird der Vorteil des Selectionsorts gegenüber dem Bubblesort: Beide benötigen zwar gleich viele Vergleiche, aber Bubblesort muss bei stark ungeordneten Folgen sehr viel mehr Vertauschungen durchführen, während Selectionsort pro Durchgang höchstens einmal tauscht.

Bub
4 +
0 +

onsort: 10 Operationen
+ 1 Vergleiche = 10 Operationen
+ 0 Tauschvorgänge = 0 Operationen

Bubblesort: 40 Operationen

4 + 3 + 2 + 1 Vergleiche = 10 Operationen
4 + 3 + 2 + 1 Tauschvorgänge = 30 Operationen

Selectionsort: 14 Operationen

4 + 3 + 2 + 1 Vergleiche = 10 Operationen
1 + 1 + 1 + 1 Tauschvorgänge = 4 Operationen