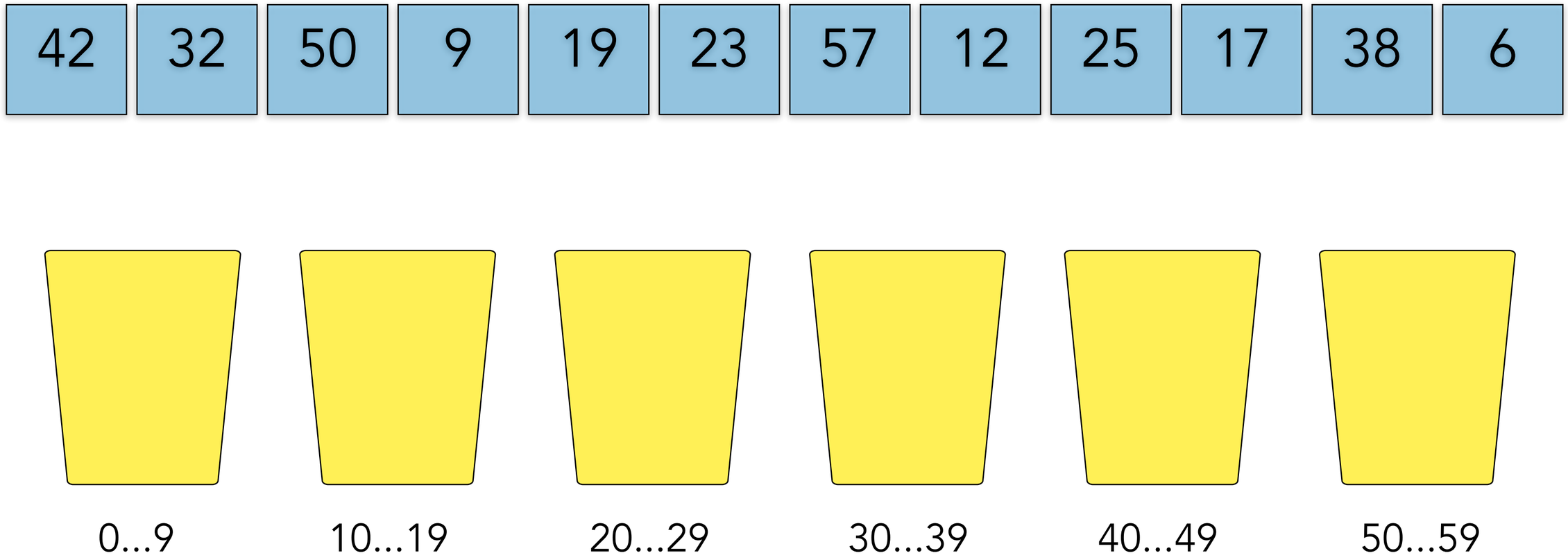
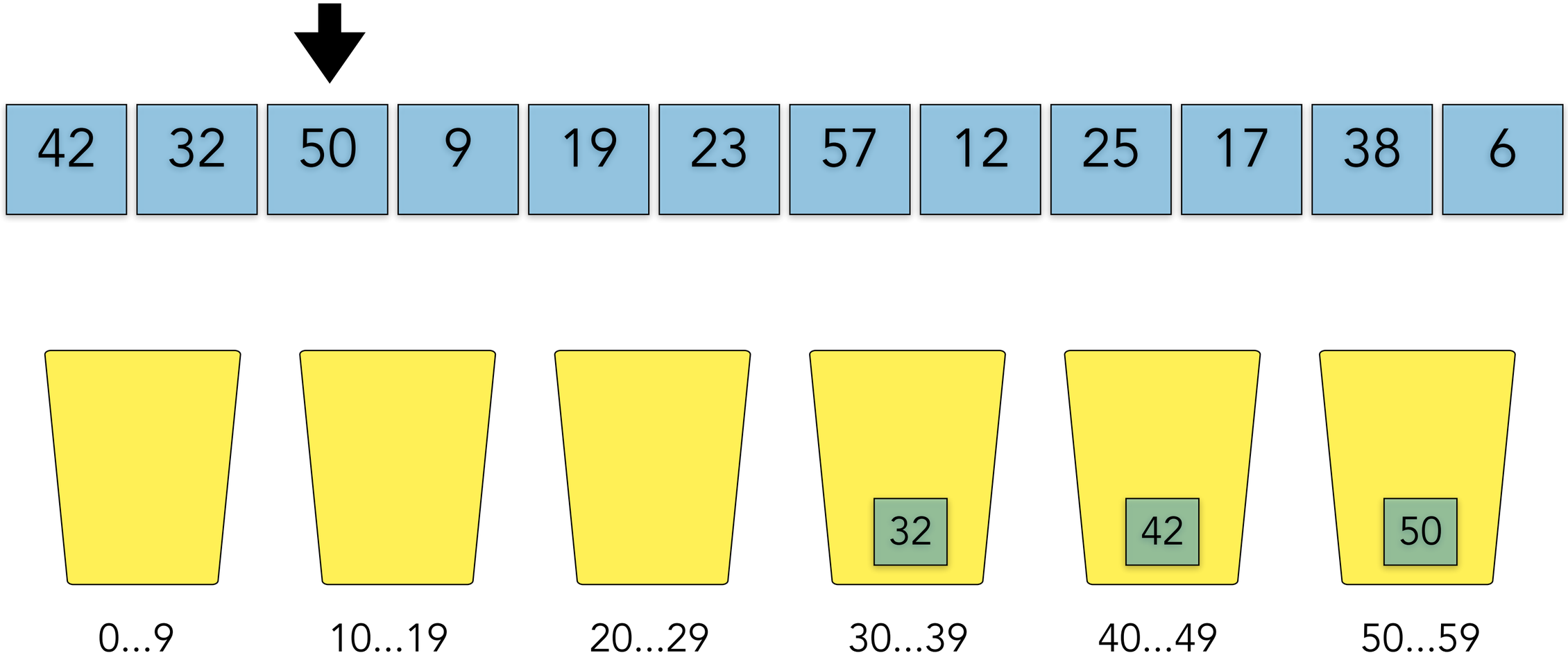
5. Sortieren und Suchen

5.6 Der Bucketsort

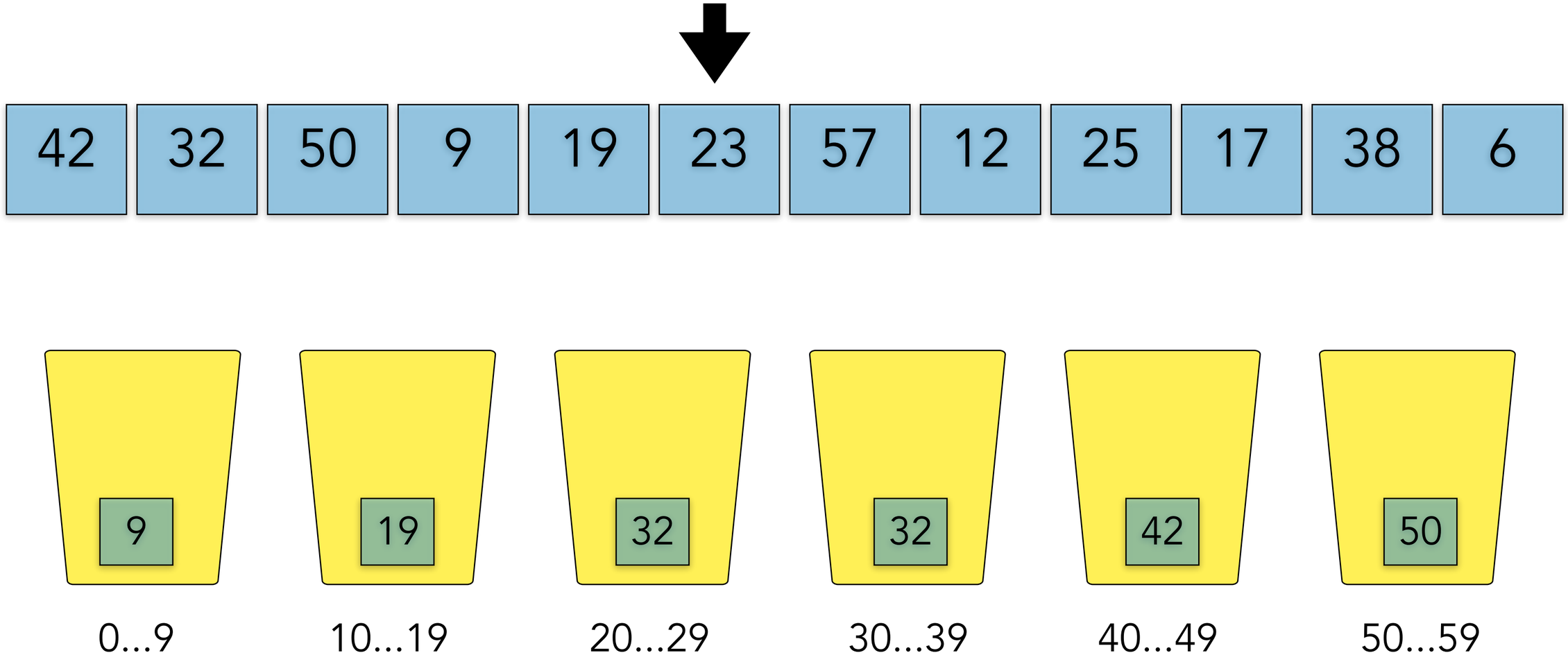
5.6 Bucketsort



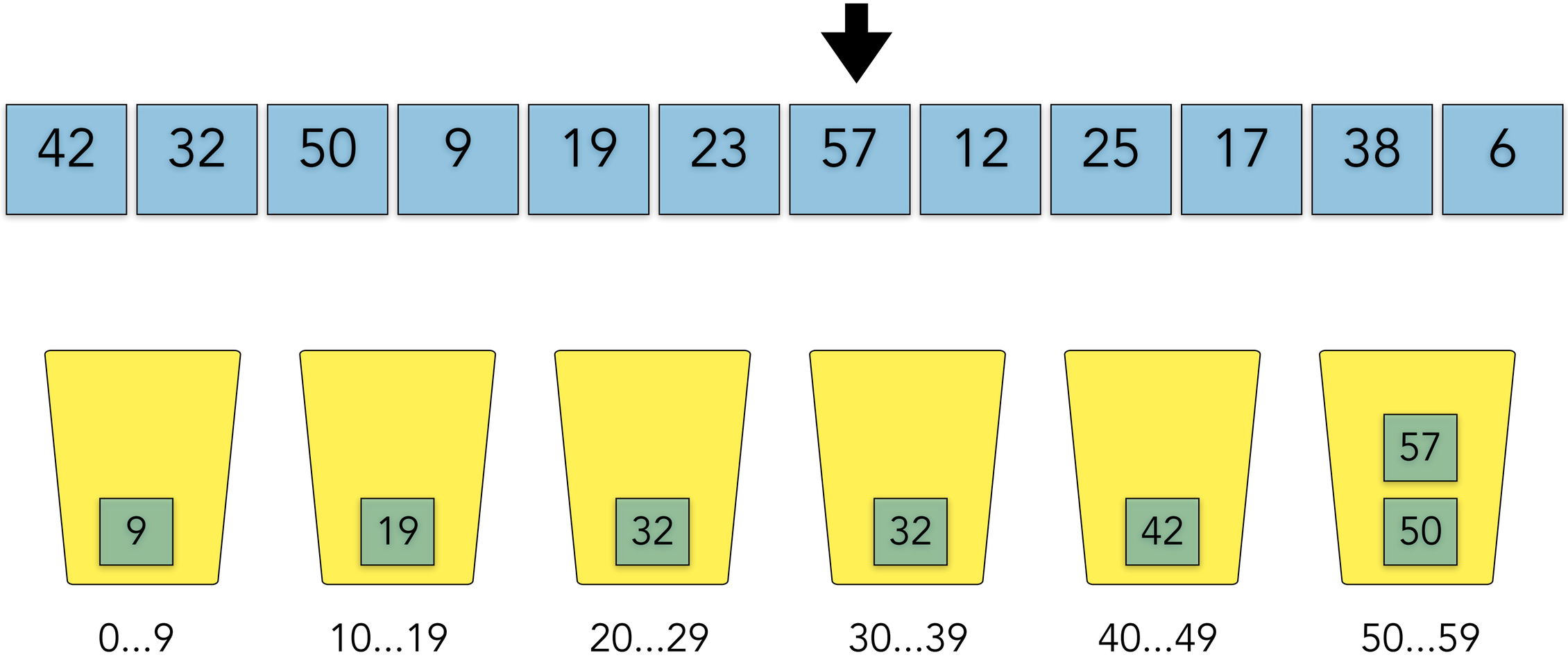
5.6 Bucketsort



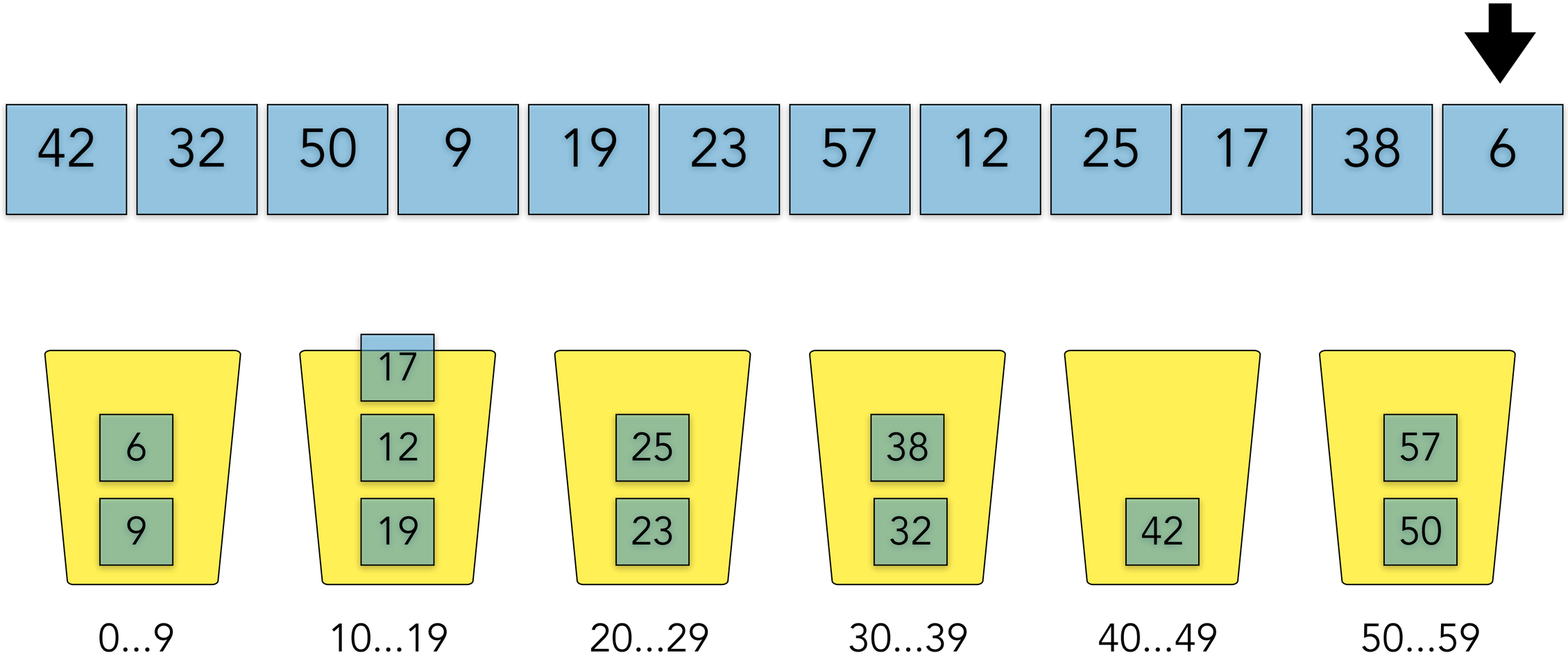
5.6 Bucketsort



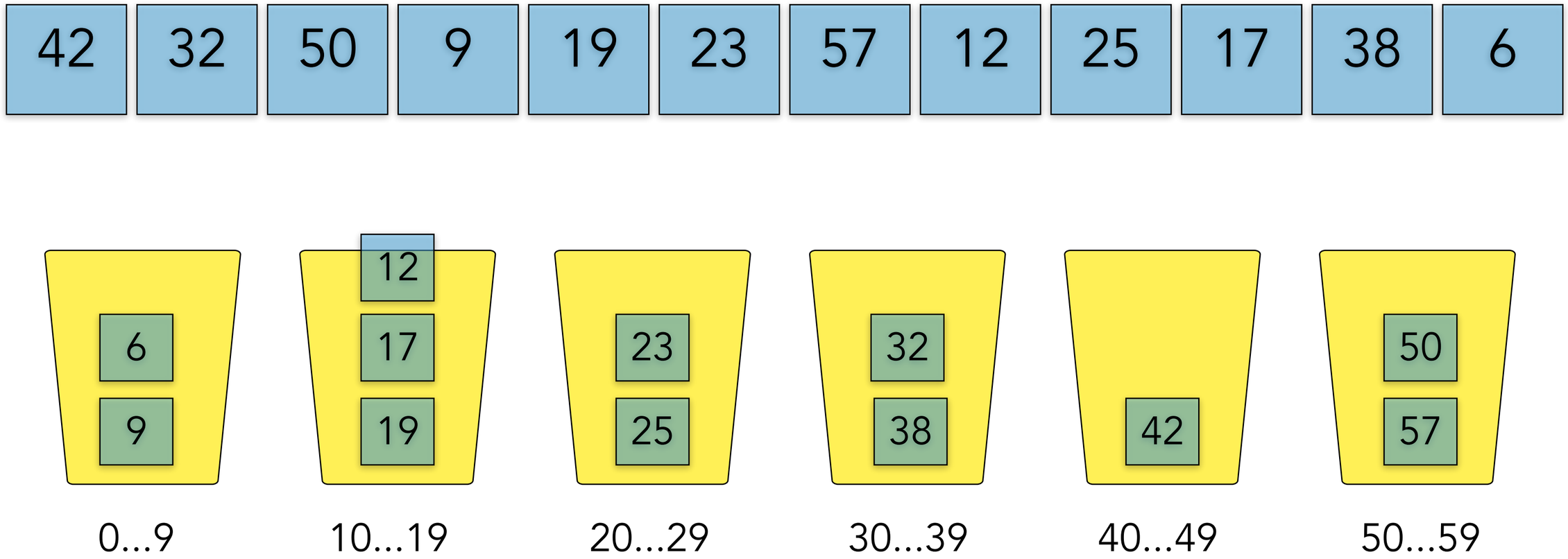
5.6 Bucketsort



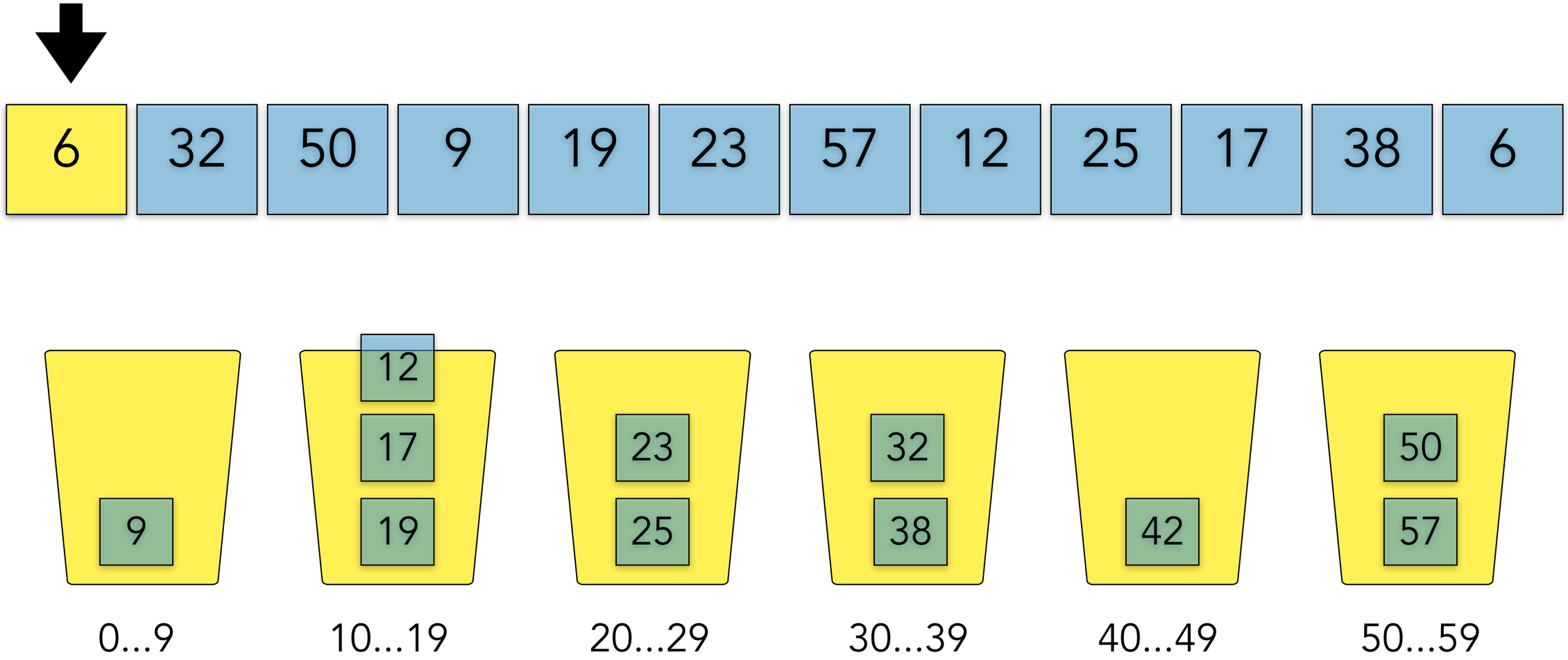
5.6 Bucketsort



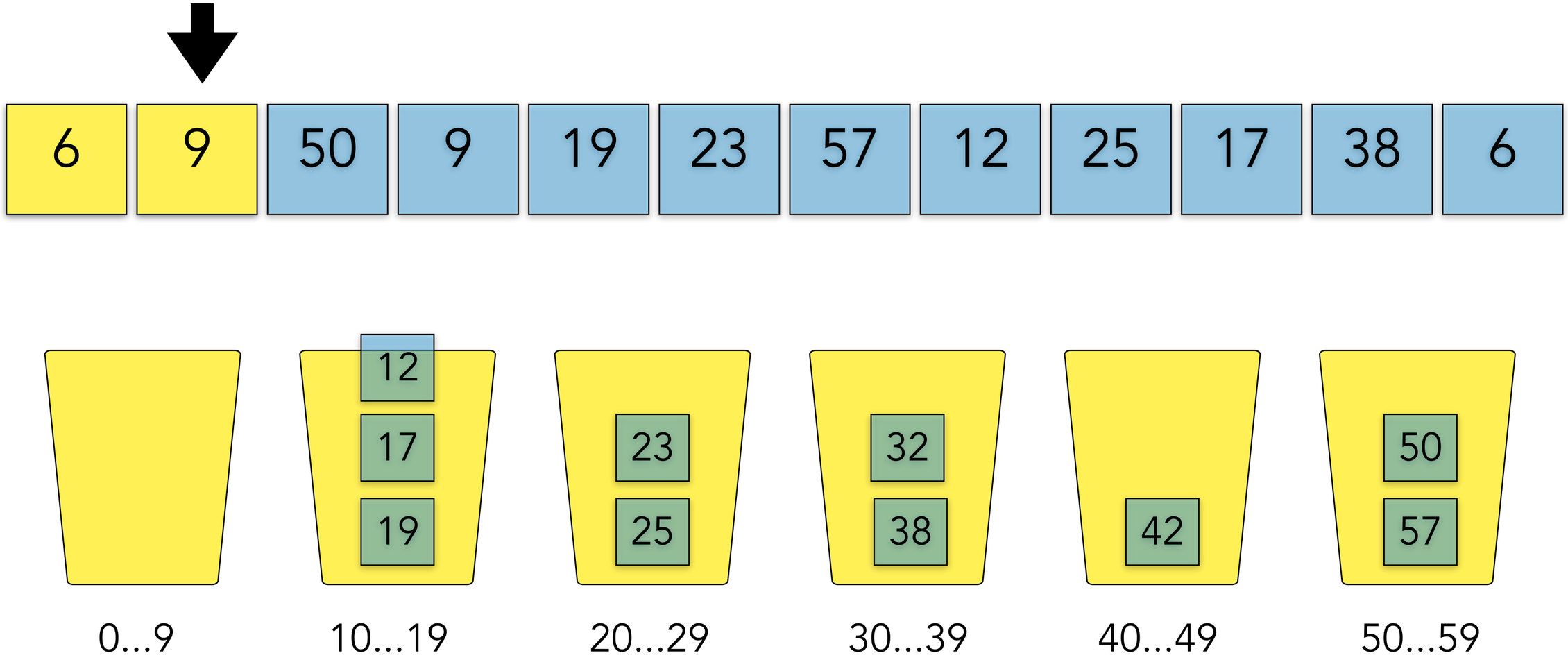
5.6 Bucketsort



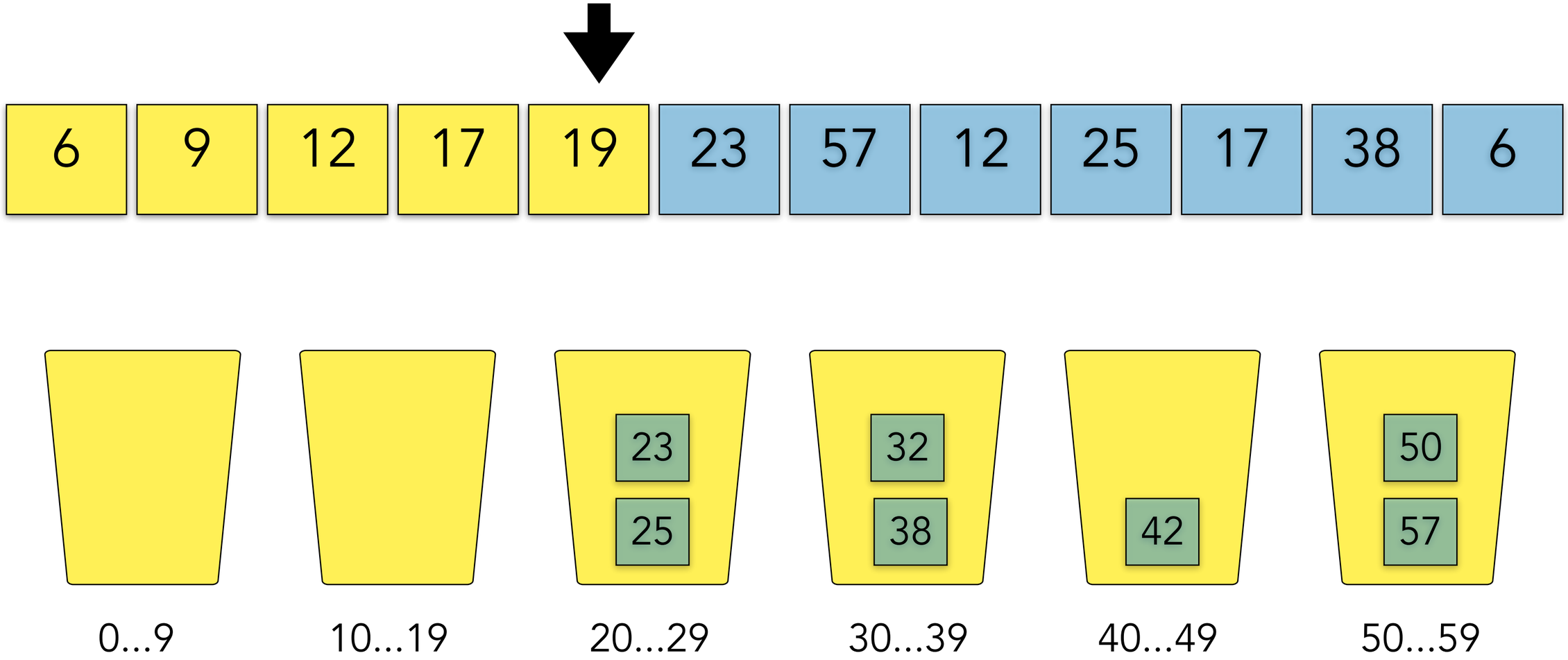
5.6 Bucketsort



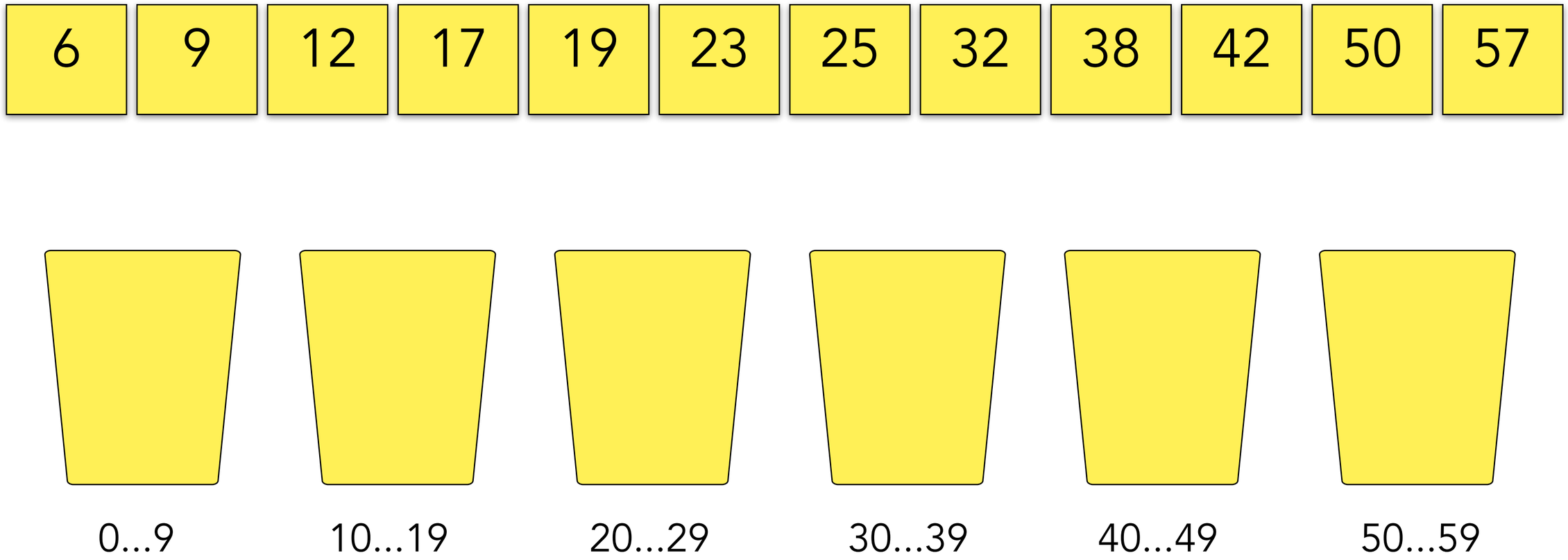
5.6 Bucketsort



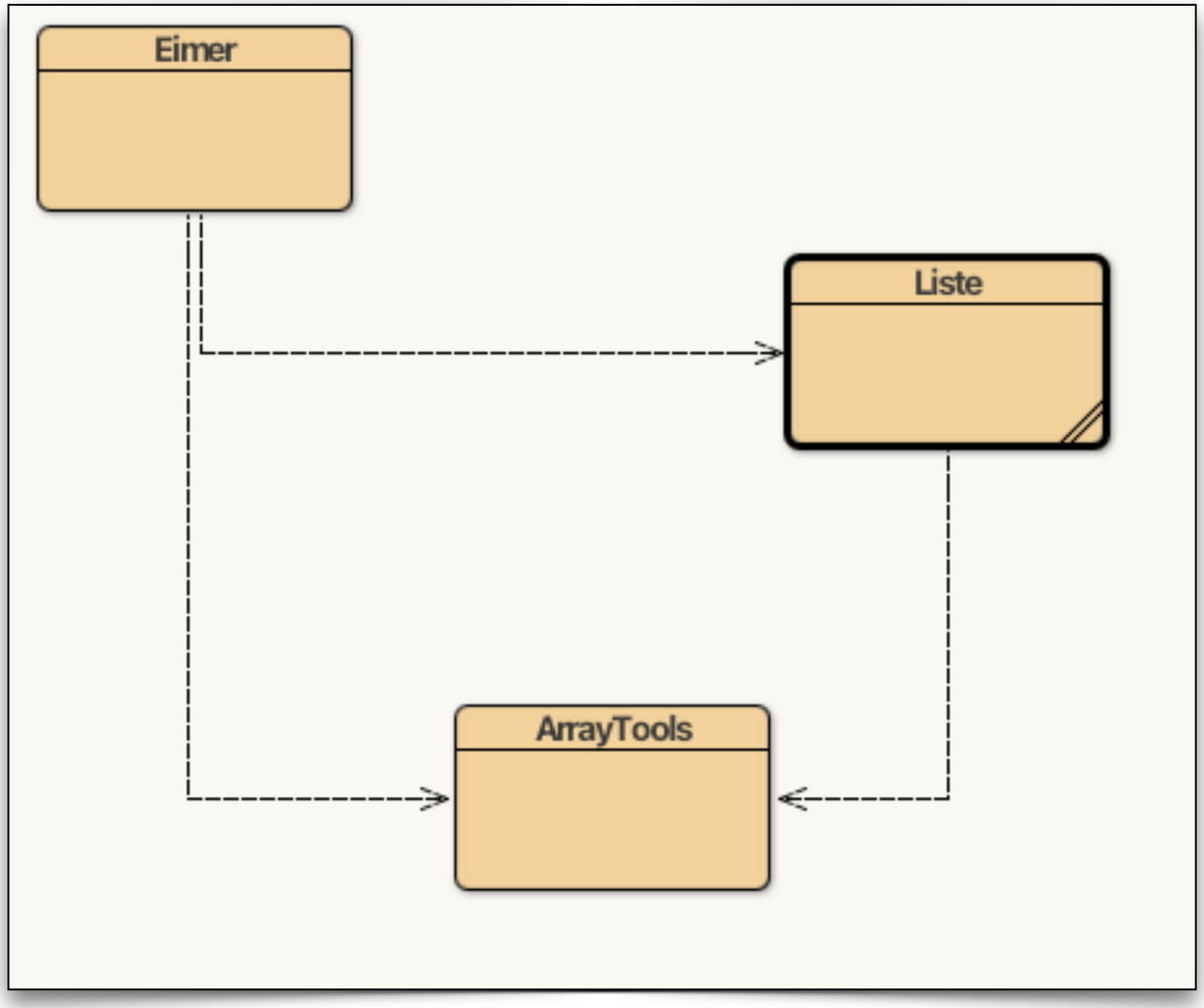
5.6 Bucketsort



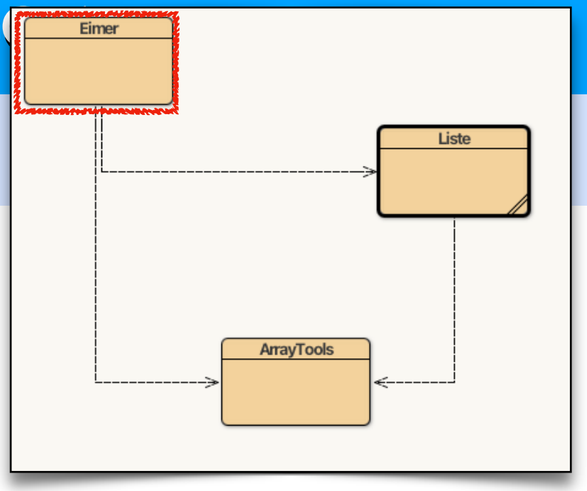
5.6 Bucketsort



5.6 Bucketsort

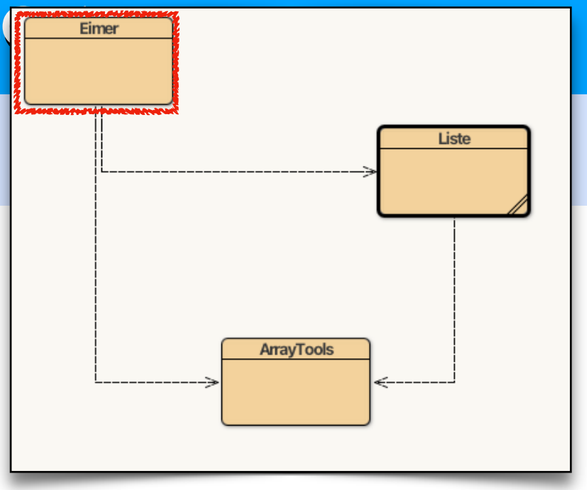


5.6 Bucketsort




```
1  public class Eimer
2  {
3      private final int MAX = 100;
4      private final int MAXWERT = 80;
5
6      private int[] zahlen;
7
```

5.6 Bucketsort



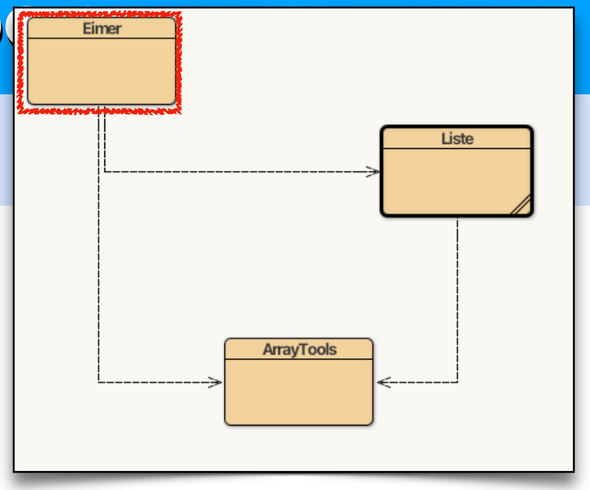

```
1  public class Eimer
2  {
3      private final int MAX = 100;
4      private final int MAXWERT = 80;
5
6      private int[] zahlen;
7
```

Größe des zu erzeugenden Arrays

größte enthaltene Zahl in diesem Array

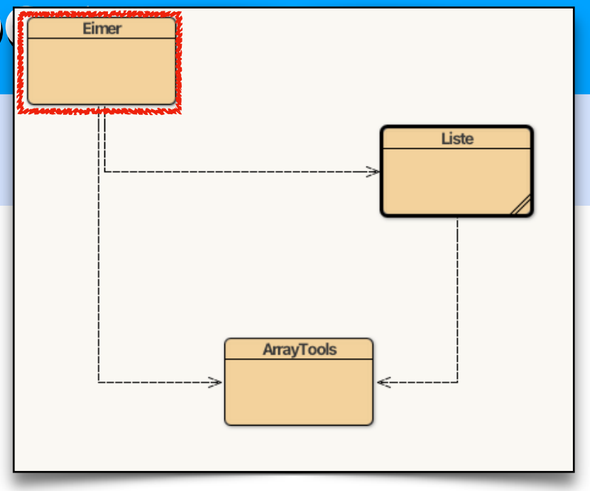
das zu sortierende Array

5.6 Bucketsort




```
8   public Eimer()
9   {
10      zahlen = ArrayTools.erzeugeArray(MAX);
11      ArrayTools.fuelleArrayMitZufallszahlen(zahlen, MAX, 1, MAXWERT);
12
13      System.out.println("Anzeigen des unsortierten Arrays");
14      ArrayTools.zeigeArray(zahlen, MAX);
15
16      bucketSortMitListe();
17
18      System.out.println("\nAnzeigen des sortierten Arrays");
19      ArrayTools.zeigeArray(zahlen, MAX);
20  }
```

5.6 Bucketsort



das Array wird erzeugt

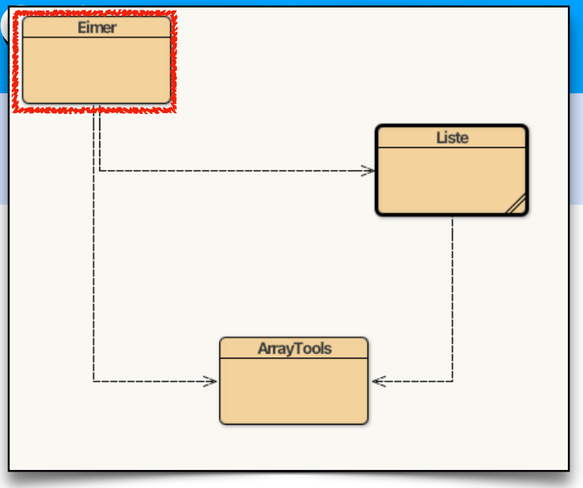
es wird komplett...

```
8   public Eimer()
9   {
10      zahlen = ArrayTools.erzeugeArray(MAX);
11      ArrayTools.fuelleArrayMitZufallszahlen(zahlen, MAX, 1, MAXWERT);
12
13      System.out.println("Anzeigen des unsortierten Arrays");
14      ArrayTools.zeigeArray(zahlen, MAX);
15
16      bucketSortMitListe();
17
18      System.out.println("\nAnzeigen des sortierten Arrays");
19      ArrayTools.zeigeArray(zahlen, MAX);
20  }
```

...mit Zufallszahlen im Bereich [1...MAXWERT] gefüllt

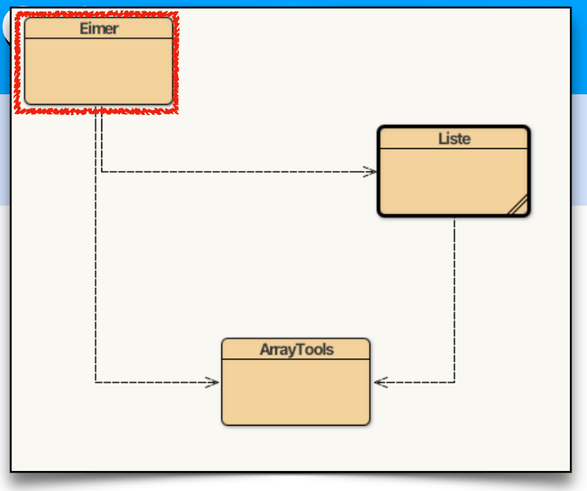
Hier wird das Array zur Kontrolle angezeigt.

5.6 Bucketsort




```
8   public Eimer()
9   {
10      zahlen = ArrayTools.erzeugeArray(MAX);
11      ArrayTools.fuelleArrayMitZufallszahlen(zahlen, MAX, 1, MAXWERT);
12
13      System.out.println("Anzeigen des unsortierten Arrays");
14      ArrayTools.zeigeArray(zahlen, MAX);
15
16      bucketSortMitListe();
17
18      System.out.println("\nAnzeigen des sortierten Arrays");
19      ArrayTools.zeigeArray(zahlen, MAX);
20  }
```

5.6 Bucketsort

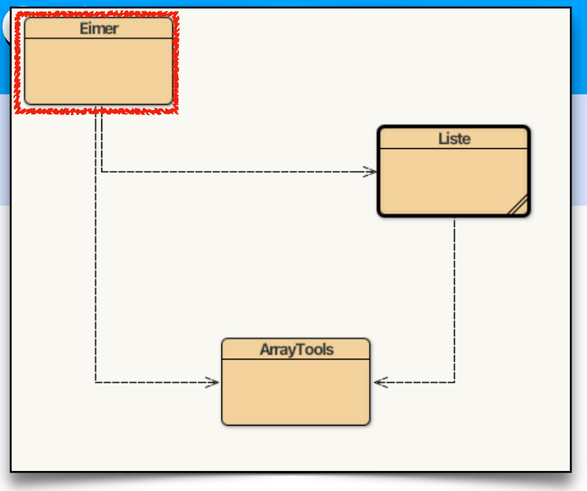



```
8   public Eimer()
9   {
10      zahlen = ArrayTools.erzeugeArray(MAX);
11      ArrayTools.fuelleArrayMitZufallszahlen(zahlen, MAX, 1, MAXWERT);
12
13      System.out.println("Anzeigen des unsortierten Arrays");
14      ArrayTools.zeigeArray(zahlen, MAX);
15
16      bucketSortMitListe();
17
18      System.out.println("\nAnzeigen des sortierten Arrays");
19      ArrayTools.zeigeArray(zahlen, MAX);
20  }
```

Aufruf der Sortier-Methode!

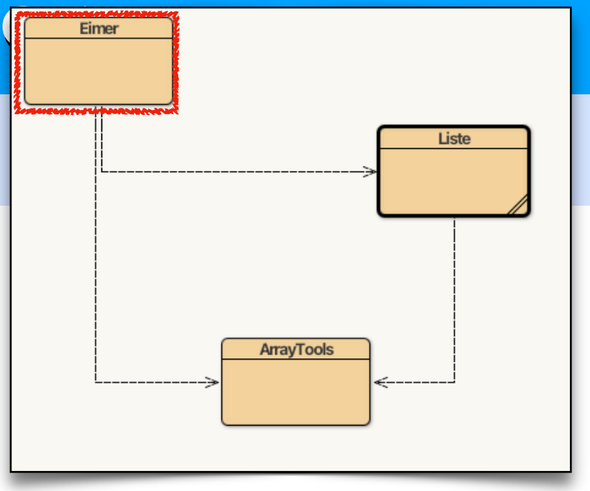
Hier wird das sortierte Array angezeigt.

5.6 Bucketsort




```
23      public void bucketSortMitListe()
24      {
25          final int BUCKETS = (MAXWERT / 10) + 1;
26
27          Liste[] bucket = new Liste[BUCKETS];
28          for (int i = 0; i < BUCKETS; i++)
29              bucket[i] = new Liste();
30
```

5.6 Bucketsort



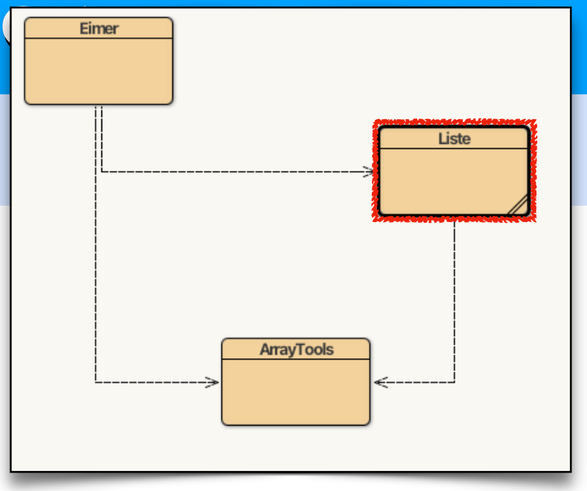

```
23      public void bucketSortMitListe()
24      {
25          final int BUCKETS = (MAXWERT / 10) + 1;
26
27          Liste[] bucket = new Liste[BUCKETS];
28          for (int i = 0; i < BUCKETS; i++)
29              bucket[i] = new Liste();
30
```

Anzahl der Buckets festlegen, als lokale Konstante!

Ein Array von Buckets anlegen

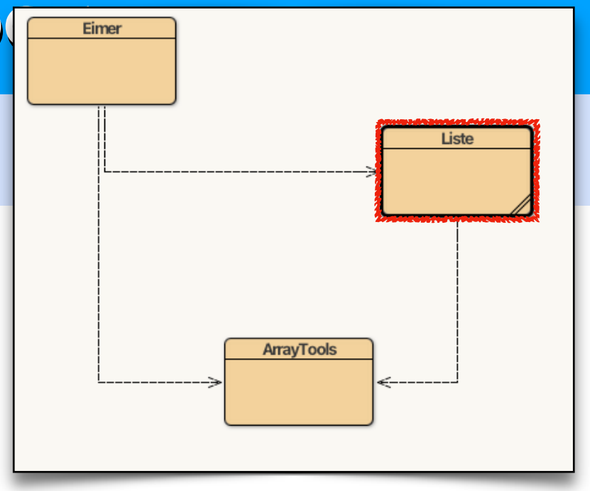
Jeder Bucket ist ein Objekt der Klasse **Liste**

5.6 Bucketsort




```
1  public class Liste
2  {
3      int[] element;
4      int   anzahl;
5
6      public Liste()
7      {
8          element = new int[100];
9          anzahl  = 0;
10     }
```

5.6 Bucketsort



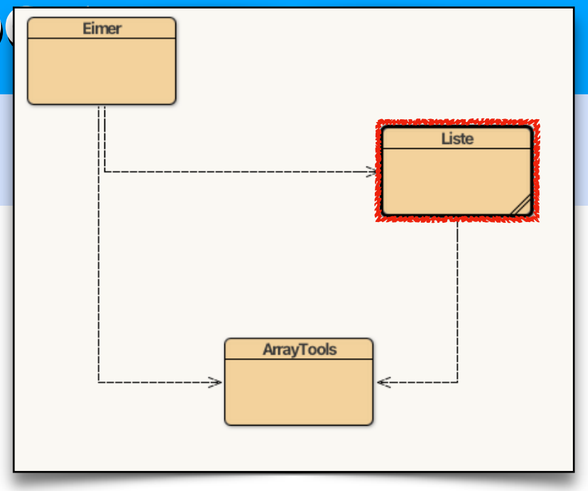

```
1  public class Liste
2  {
3      int[] element;
4      int   anzahl;
5
6      public Liste()
7      {
8          element = new int[100];
9          anzahl  = 0;
10     }
```

Das interne Array der Klasse Liste

Die Zahl der jeweils enthaltenen Elemente

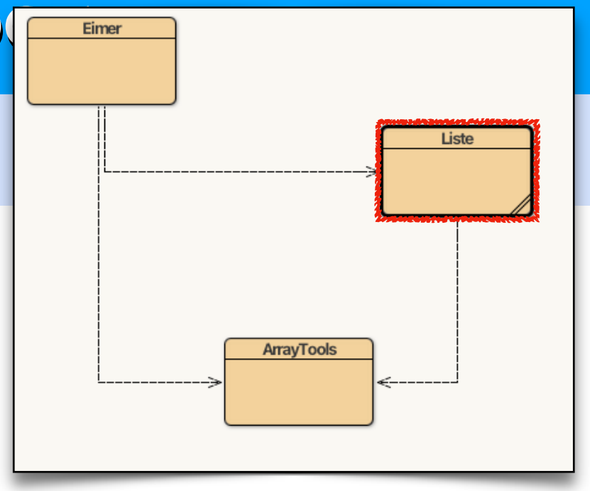
Die Zahl der jeweils enthaltenen Elemente

5.6 Bucketsort




```
12      public void fuegeEin(int neu)
13      {
14          if (anzahl >= 100) return;
15
16          int pos = 0;
17          while (pos < anzahl && element[pos] < neu)
18              pos++;
19
20          for (int i = anzahl; i > pos; i--)
21              element[i] = element[i - 1];
22
23          element[pos] = neu;
24          anzahl++;
25      }
```

5.6 Bucketsort




```
12  public void fuegeEin(int neu)
13  {
14      if (anzahl >= 100) return;
15
16      int pos = 0;
17      while (pos < anzahl && element[pos] < neu)
18          pos++;
19
20      for (int i = anzahl; i > pos; i--)
21          element[i] = element[i - 1];
22
23      element[pos] = neu;
24      anzahl++;
25  }
```

Alle Plätze belegt?

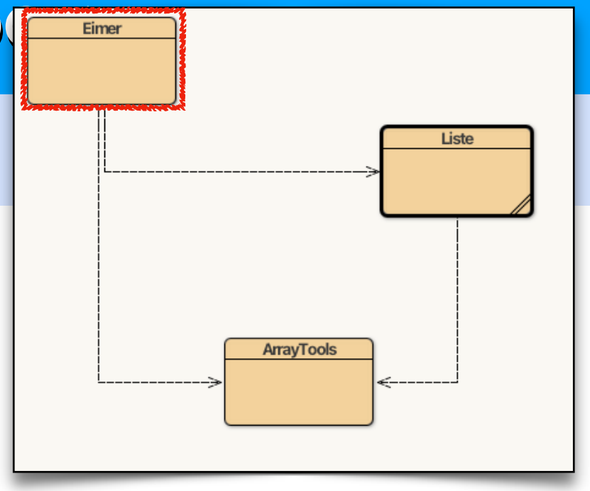
Suche nach der Einfügeposition.

Alle Elemente rechts der Einfügeposition rücken eine Position nach rechts auf.

Neues Element einfügen.

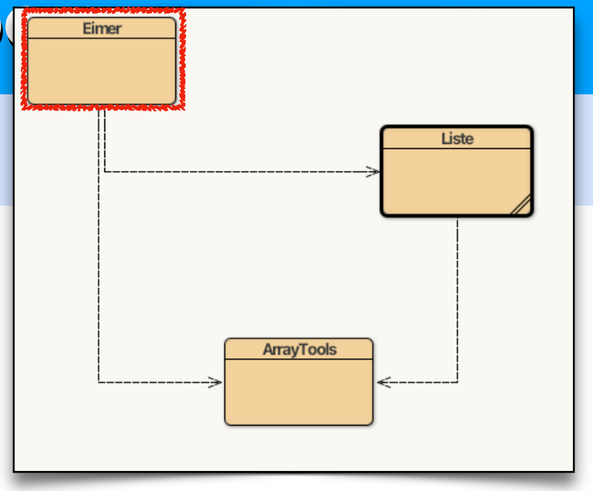
Die Anzahl der vorhandenen Elemente wird aktualisiert.

5.6 Bucketsort




```
31      for (int i = 0; i < MAX; i++)
32      {
33          int z = zahlen[i];
34          int b = z / 10;
35          bucket[b].fuegeEin(z);
36      }
```

5.6 Bucketsort



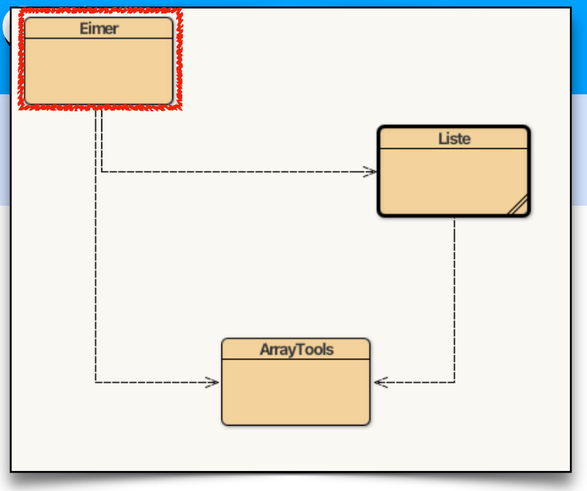

```
31      for (int i = 0; i < MAX; i++)
32      {
33          int z = zahlen[i];
34          int b = z / 10;
35          bucket[b].fuegeEin(z);
36      }
```

Aktuelles Element.

Index des zuständigen Buckets berechnen.

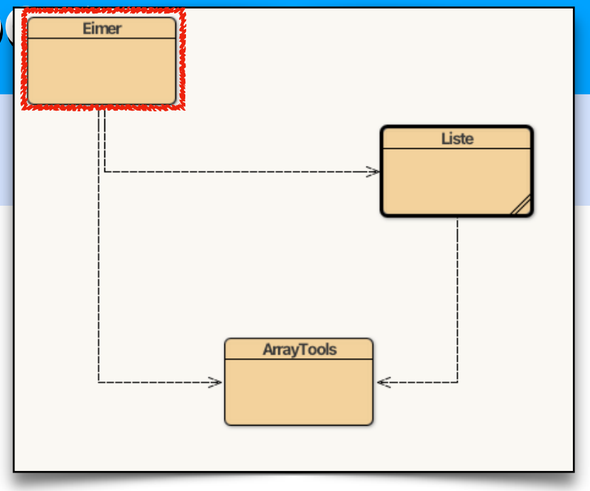
Arrayelement in den passenden Bucket sortiert einfügen.

5.6 Bucketsort




```
53      int index = 0;
54      for (int b = 0; b < BUCKETS; b++)
55      {
56          for (int i = 0; i < bucket[b].anzahl; i++)
57          {
58              zahlen[index] = bucket[b].element[i];
59              index++;
60          }
61      }
```

5.6 Bucketsort




```
53      int index = 0;
54      for (int b = 0; b < BUCKETS; b++)
55      {
56          for (int i = 0; i < bucket[b].anzahl; i++)
57          {
58              zahlen[index] = bucket[b].element[i];
59              index++;
60          }
61      }
```

Alle Buckets durchgehen.

Jede Zahl aus dem Bucket extrahieren.

Extrahierte Zahl in das Hauptarray zurück schreiben.

Nächste Position im Hauptarray festlegen.