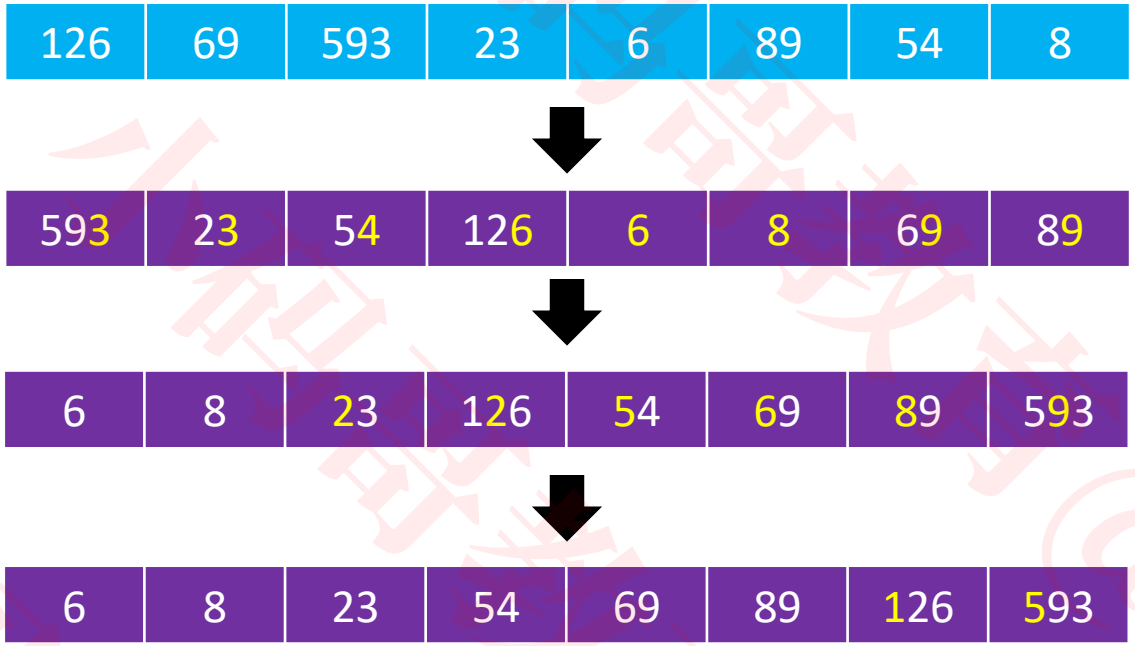
基数排序（Radix Sort）

- 基数排序非常适合用于整数排序（尤其是非负整数），因此本课程只演示对非负整数进行基数排序
- 执行流程：依次对个位数、十位数、百位数、千位数、万位数...进行排序（从低位到高位）



- 个位数、十位数、百位数的取值范围都是固定的0~9，可以使用计数排序对它们进行排序

- 思考：如果先对高位排序，再对低位排序，是否可行？

小码哥教育 SEEMYGO

基数排序 – 实现

```
int max = array[0]; // 最大值
for (int i = 1; i < array.length; i++) {
    if (array[i] > max) {
        max = array[i];
    }
}

int output[] = new int[array.length];
int counts[] = new int[10];
for (int divider = 1; divider <= max; divider *= 10) {
    // 对每一位进行计数排序
    countingSort(divider, output, counts);
}
```

小码哥教育 SEEMYGO

基数排序 – 实现

```
private void countingSort(int divider, int[] output, int[] counts) {
    for (int i = 0; i < counts.length; i++) {
        counts[i] = 0;
    }
    for (int i = 0; i < array.length; i++) {
        counts[array[i] / divider % 10]++;
    }
    for (int i = 1; i < counts.length; i++) {
        counts[i] += counts[i - 1];
    }
    for (int i = array.length - 1; i >= 0; i--) {
        output[--counts[array[i] / divider % 10]] = array[i];
    }
    for (int i = 0; i < array.length; i++) {
        array[i] = output[i];
    }
}
```

- 最好、最坏、平均时间复杂度：$O(d * (n + k))$，d 是最大值的位数，k 是进制。属于稳定排序
- 空间复杂度：$O(n + k)$，k 是进制

基数排序 – 另一种思路

126	69	593	23	6	89	54	8

0	1	2	3	4	5	6	7	8	9
			593	54		126		8	69
			23			6			89

593	23	54	126	6	8	69	89

0	1	2	3	4	5	6	7	8	9
6		23			54	69		89	593
8		126							

6	8	23	126	54	69	89	593

0	1	2	3	4	5	6	7	8	9
6	126				593				
8									
23									
54									
69									
89									

6	8	23	54	69	89	126	593

基数排序 – 另一种思路的实现

```java
int max = array[0]; // 最大值
for (int i = 1; i < array.length; i++) {
    if (array[i] > max) {
        max = array[i];
    }
}
```

```java
// 桶数组
int[][] buckets = new int[10][array.length];
// 每个桶的元素数量
int[] bucketSizes = new int[buckets.length];
for (int divider = 1; divider <= max; divider *= 10) {
    for (int i = 0; i < array.length; i++) {
        int no = array[i] / divider % 10;
        buckets[no][bucketSizes[no]++] = array[i];
    }
    int index = 0;
    for (int i = 0; i < buckets.length; i++) {
        for (int j = 0; j < bucketSizes[i]; j++) {
            array[index++] = buckets[i][j];
        }
        bucketSizes[i] = 0;
    }
}
```

- 空间复杂度是 O(kn + k)，时间复杂度是 O(dn)
- d 是最大值的位数，k 是进制