COMPUTING THE MODE IN JAVA

We'll now consider a more complex problem. Suppose we have an array of test scores, all integers between 0 and 100, and we want to find the *mode* — that is, the integer occurring most often. We'll see two techniques for computing this.

In the more obvious technique, found in Figure 19.2, we'll go through each number in the array and count how many times that number appears in the array. As we go along, we track which number has appeared most often. Once we complete counting each number in the array, we'll have found the mode.

Figure 19.2: The `computeMode` method.

```
public int computeMode(int[] nums) {
    int maxValue = -1;
    int maxCount = 0;
    for(int i = 0; i < nums.length; i++) {
        // count number of times nums[i] is in array
        int count = 0;
        for(int j = 0; j < nums.length; j++) {
            if(nums[j] == nums[i]) {
                count++;
            }
        }

        // remember the highest count we have seen
        if(count > maxCount) {
            maxValue = nums[i];
            maxCount = count;
        }
    }
    return maxValue;
}
```

An alternative technique is to create a second array, which I'll call `tally`, which initially has 0 throughout. We then go through the array of test scores, and with each score *k*, we increment entry *k* of `tally`. By the end of this process, `tally[i]` will hold the number of times `i` appears among the test scores. Our last step is to see which entry in `tally` is largest; the index of this entry is the mode.

Figure 19.3: The `tallyMode` method.

```
public int tallyMode(int[] nums) {
    // create array of tallies, all initialized to zero
    int[] tally = new int[101];
    for(int i = 0; i < tally.length; i++) {
        tally[i] = 0;
    }

    // for each array entry, increment corresponding tally box
    for(int i = 0; i < nums.length; i++) {
        int value = nums[i];
        tally[value]++;
    }

    // now find the index of the largest tally - this is the mode
    int maxIndex = 0;
    for(int i = 1; i < tally.length; i++) {
        if(tally[i] > tally[maxIndex]) {
            maxIndex = i;
        }
    }
    return maxIndex;
}
```

This second technique can often be significantly faster: It looks at each test score only once, whereas the first technique will look at each test score as many times as there are test scores. That is, if our parameter array contains n test scores, the first technique will look into the array n^2 times, whereas the second technique will look into the array only $n$times.

Source : http://www.toves.org/books/java/ch19-array/index.html