Traversal algorithms
Array traversal

- **traversal**: An examination of each element of an array.

- Traversal algorithms often takes the following form:
  ```java
  for (int i = 0; i < <array>.length; i++) {
    do something with <array>[i];
  }
  ```

- Examples:
  - printing out the elements
  - searching for a specific value
  - rearranging the elements
  - computing a value based on the elements
Example: Printing array elements

```java
int[] list = { 4, 1, 9, 7 };
for (int i = 0; i < list.length; i++) {
    System.out.println(i + " : " + list[i]);
}
```

Output:

```
0 : 4
1 : 1
2 : 9
3 : 7
```

- How could we change the code to print the following?
  4, 1, 9, 7
Example: Searching an array

```java
int[] list = { 4, 1, 2, 7, 6, 3, 2, 4, 0, 9 };
int largestEven = 0;
for (int i = 0; i < list.length; i++) {
    if (list[i] % 2 == 0 && list[i] > largestEven) {
        largestEven = list[i];
    }
}
System.out.println("Largest even: "+ largestEven);
```

Output:
Largest even: 6

- What assumptions does this code make?
Shifting elements in an array
Array insertion

How would you insert a number into an array of sorted integers? Assume that the largest number gets bumped off the array.

<table>
<thead>
<tr>
<th>index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>1</td>
<td>6</td>
<td>18</td>
<td>37</td>
<td>64</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>1</td>
<td>3</td>
<td>6</td>
<td>18</td>
<td>37</td>
</tr>
</tbody>
</table>
public static void insertInOrder(int[] array, int num) {
    int insertionIndex = findInsertionIndex(array, num);
    if (insertionIndex < array.length) {
        for (int i = array.length - 1; i >= insertionIndex + 1; i--) {
            array[i] = array[i-1];
        }
        array[insertionIndex] = num;
    }
}

public static int findInsertionIndex(int[] array, int num) {
    for (int i = 0; i < array.length; i++) {
        if (num < array[i]) {
            return i;
        }
    }
    return array.length;
}
Rotating elements left

<table>
<thead>
<tr>
<th>index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>3</td>
<td>8</td>
<td>9</td>
<td>7</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>8</td>
<td>9</td>
<td>7</td>
<td>5</td>
<td>3</td>
</tr>
</tbody>
</table>
Rotating elements left: Solution

```java
public static void rotateLeft(int[] array) {
    int first = array[0];
    for (int i = 0; i < array.length - 1; i++) {
        array[i] = array[i + 1];
    }
    array[array.length - 1] = first;
}
```

- What assumptions does this code make?
Exercise: Random elements

- Write a method named `printRandomNumbers` that accepts an array of integers as one parameter, and \( n \), how many numbers to print, as a second parameter. The method will print out \( n \) random elements *with repetition* from the array.

- Write a method named `printRandomNumbers` that accepts an array of integers as one parameter, and \( n \), how many numbers to print, as a second parameter. The method will print out \( n \) random elements *without repetition* from the array.
public static void printRandomNumbers(int[] numbers, int n) {
    Random rand = new Random();

    int numNumbers = numbers.length;
    for (int i = 1; i <= n; i++) {
        int index = rand.nextInt(numNumbers);
        System.out.println(numbers[index]);

        // shift elements to the left
        for (int j = index; j < numNumbers - 1; j++) {
            numbers[j] = numbers[j + 1];
        }

        numNumbers--;
    }
}

- What happens to the array after this method finishes?
  - How could you preserve the contents of the array?
String traversal

- **Strings are like arrays of chars.**

<table>
<thead>
<tr>
<th>index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>l</td>
<td>e</td>
<td>t</td>
<td>t</td>
<td>e</td>
<td>r</td>
<td>s</td>
</tr>
</tbody>
</table>

- We can write algorithms to traverse strings to compute information.

- What useful information might the following string have?

  "ADRARRADRRADGADARRRARARAADADDRDDRRDADAAD"
String traversal: Example

// string stores voters' votes
// (R)EPUBLICAN, (D)EMOCRAT, (A)LEX, (G)IORGIO
String votes = “ADRARRADRRADGADARRARARAADADDRDDRRDADAAD”;
int[] counts = new int[4]; // R -> 0, D -> 1, B -> 2, M -> 3
for (int i = 0; i < votes.length(); i++) {
    char c = votes.charAt(i);
    if (c == 'R') {
        counts[0]++;
    } else if (c == 'D') {
        counts[1]++;
    } else if (c == 'A') {
        counts[2]++;
    } else { // c == 'G'
        counts[3]++;
    }
}
System.out.println(Arrays.toString(counts));

Output:
[13, 12, 14, 1]
Example data: Class attendance

Consider the following dataset which represents attendance for three sections of five students:

```
111111010111111010011101101101110001110010100
0100011001010001010010010101010100101001001000
10010100101100010010010101010100100111000101
```

```
week1  week2  week3  week4  week5  week6  week7  week8  week9
11111   11010   11111   10100   11101   10110   11000   11100   10100
```

```
student1     student2     student3     student4     student5
1             1             0             1             0
```
Data transformations

- Sometimes we will use data in one form to compute new data in another form.
  - Often each *transformation* is stored into its own array.

- Transformations require a *mapping* between the original data and array indices.
Typical mappings

- **Tally**
  
  “If the input value is the integer $i$, do something with array index $i$.”

- **Based on the position in the data**
  
  “Store the $i$th value we read into index $i$.”

- **Explicit mappings**
  
  “Count occurrences of ’R’ into index 0 and occurrences of ’D’ into index 1.”
Exercise: Section attendance

- Write a program that reads the preceding section data and produces output such as the following:

  Section #1:
  Sections attended: [9, 6, 7, 4, 3]
  Student scores: [20, 20, 20, 16, 12]
  Student grades: [100.0, 100.0, 100.0, 80.0, 60.0]

  Section #2:
  Sections attended: [4, 6, 2, 2, 3]
  Student scores: [16, 20, 8, 8, 12]
  Student grades: [80.0, 100.0, 40.0, 40.0, 60.0]

  Section #3:
  Sections attended: [5, 4, 2, 5, 3]
  Student scores: [20, 16, 8, 20, 12]
  Student grades: [100.0, 80.0, 40.0, 100.0, 60.0]
Solution: Section attendance

// This program reads data representing which students attended
// which discussion sections and produces output of the students'
// section attendance and scores.
import java.util.*;

public class Sections {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int section = 1;    // used to count sections

        while(true) {
            System.out.println("Enter attendance (blank to quit): ");
            String line = input.nextLine();    // one section's data
            if("".equals(line)) {
                break;
            }
            processSection(section, line);
            section++;
        }
    }
}
public static void processSection(int sectionNum, String line) {
    System.out.println("Section #" + sectionNum + ":");

    int[] attended = new int[5];       // count sections attended
    for (int i = 0; i < line.length(); i++) {
        char c = line.charAt(i);
        if (c == '1') {                // student attended section
            attended[i % 5]++;
        }
    }
    System.out.println("Sections attended: " + Arrays.toString(attended));

    ...
}
Solution: Section attendance

// compute section score out of 20 points
int[] scores = new int[5];
for (int i = 0; i < scores.length; i++) {
    scores[i] = Math.min(4 * attended[i], 20);
}
System.out.println("Student scores: "+ Arrays.toString(scores));

// compute section grade out of 100%
double[] grades = new double[5];
for (int i = 0; i < scores.length; i++) {
    grades[i] = 100.0 * scores[i] / 20;
}
System.out.println("Student grades: "+ Arrays.toString(grades));
System.out.println();