1.4 Arrays

This lecture. Store and manipulate huge quantities of data.

Array. Indexed sequence of values of the same type.

Examples.
- 52 playing cards in a deck.
- 5 thousand undergrads at Princeton.
- 1 million characters in a book.
- 10 million audio samples in an MP3 file.
- 4 billion nucleotides in a DNA strand.
- 73 billion Google queries per year.
- 50 trillion cells in the human body.
- $6.02 \times 10^{23}$ particles in a mole.

Goal. 10 variables of the same type.

```java
// tedious and error-prone
double a0, a1, a2, a3, a4, a5, a6, a7, a8, a9;
a0 = 0.0;
a1 = 0.0;
a2 = 0.0;
a3 = 0.0;
a4 = 0.0;
a5 = 0.0;
a6 = 0.0;
a7 = 0.0;
a8 = 0.0;
a9 = 0.0;
...
da4 = 3.0;
da4 = 8.0;
...
double x = a4 + a8;
```
Many Variables of the Same Type

Goal. 10 variables of the same type.

// easy alternative
double[] a = new double[10];
...
a[4] = 3.0;
...
a[8] = 8.0;
...
double x = a[4] + a[8];

Many Variables of the Same Type

Goal. 1 million variables of the same type.

// scales to handle large arrays
double[] a = new double[1000000];
...
a[123456] = 3.0;
...
a[987654] = 8.0;
...
double x = a[123456] + a[987654];

Arrays in Java

Java has special language support for arrays.
- To make an array: declare, create, and initialize it.
- To access element i of array named a, use a[i].
- Array indices start at 0.

int N = 10;  // size of array
double[] a;  // declare the array
a = new double[N];  // create the array
for (int i = 0; i < N; i++)  // initialize the array
   a[i] = 0.0;  // all to 0.0

Compact alternative.
- Declare, create, and initialize in one statement.
- Default initialization: all numbers automatically set to zero.

int N = 10;  // size of array
double[] a = new double[N];  // declare, create, init

Vector Dot Product

Dot product. Given two vectors x[] and y[] of length N, their dot product is the sum of the products of their corresponding components.

double[] x = { 0.3, 0.6, 0.1 };
double[] y = { 0.5, 0.1, 0.4 };
int N = x.length;
double sum = 0.0;
for (int i = 0; i < N; i++) {
   sum = sum + x[i]*y[i];
}

<table>
<thead>
<tr>
<th>i</th>
<th>x[i]</th>
<th>y[i]</th>
<th>x[i]*y[i]</th>
<th>sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>.30</td>
<td>.50</td>
<td>.15</td>
<td>.15</td>
</tr>
<tr>
<td>1</td>
<td>.60</td>
<td>.10</td>
<td>.06</td>
<td>.21</td>
</tr>
<tr>
<td>2</td>
<td>.10</td>
<td>.40</td>
<td>.04</td>
<td>.25</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.25</td>
</tr>
</tbody>
</table>
Array-Processing Examples

<table>
<thead>
<tr>
<th>create an array with random values</th>
<th>double[] a = new double[N]; for (int i = 0; i &lt; N; i++) a[i] = Math.random();</th>
</tr>
</thead>
<tbody>
<tr>
<td>print the array values, one per line</td>
<td>for (int i = 0; i &lt; N; i++) System.out.println(a[i]);</td>
</tr>
<tr>
<td>find the maximum of the array values</td>
<td>double max = Double.NEGATIVE_INFINITY; for (int i = 0; i &lt; N; i++) if (a[i] &gt; max) max = a[i];</td>
</tr>
<tr>
<td>compute the average of the array values</td>
<td>double sum = 0.0; for (int i = 0; i &lt; N; i++) sum += a[i]; double average = sum / N;</td>
</tr>
<tr>
<td>copy to another array</td>
<td>double[] b = new double[N]; for (int i = 0; i &lt; N; i++) b[i] = a[i];</td>
</tr>
<tr>
<td>reverse the elements within an array</td>
<td>for (int i = 0; i &lt; N/2; i++) { double temp = b[i]; b[i] = b[N-1-i]; b[N-i-1] = temp; }</td>
</tr>
</tbody>
</table>

Shuffling a Deck

Ex. Print a random card.

String[] rank = {
   "2", "3", "4", "5", "6", "7", "8", "9",
   "10", "Jack", "Queen", "King", "Ace"
};
String[] suit = {
   "Clubs", "Diamonds", "Hearts", "Spades"
};
int i = (int) (Math.random() * 13); // between 0 and 12
int j = (int) (Math.random() * 4); // between 0 and 3
System.out.println(rank[i] + " of " + suit[j]);

Setting Array Values at Compile Time

Ex. Create a deck of playing cards and print them out.

String[] deck = new String[52];
for (int i = 0; i < 13; i++)
   for (int j = 0; j < 4; j++)
      deck[4*i + j] = rank[i] + " of " + suit[j];
for (int i = 0; i < 52; i++)
   System.out.println(deck[i]);

Q. In what order does it output them?

A. two of clubs
   two of diamonds
   two of hearts
   two of spades
   three of clubs
   three of clubs
  ...
B. two of clubs
   three of clubs
   four of clubs
   five of clubs
   six of clubs
   ...

Setting Array Values at Run Time
Goal. Given an array, rearrange its elements in random order.

Shuffling algorithm.
- In iteration $i$, pick random card from deck[$i$] through deck[N-1], with each card equally likely.
- Exchange it with deck[$i$].

```java
int N = deck.length;
for (int i = 0; i < N; i++) {
    int r = i + (int) (Math.random() * (N-i));
    String t = deck[r];
    deck[r] = deck[i];
    deck[i] = t;
}
```

Avoid "hardwired" constants

Build the deck

Shuffle

Print shuffled deck

Coupon Collector
Coupon Collector Problem

**Coupon collector problem.** Given \( N \) different card types, how many do you have to collect before you have (at least) one of each type?

\[
\text{assuming each possibility is equally likely for each card that you collect}
\]

Simulation algorithm. Repeatedly choose an integer \( i \) between 0 and \( N-1 \). Stop when we have at least one card of every type.

**Q.** How to check if we’ve seen a card of type \( i \)?

**A.** Maintain a boolean array so that \( \text{found}[i] \) is true if we’ve already collected a card of type \( i \).

```
public class CouponCollector {
    public static void main(String[] args) {
        int N = Integer.parseInt(args[0]);
        int cardcnt = 0;    // number of cards collected
        int valcnt = 0;     // number of distinct cards

        // do simulation
        boolean[] found = new boolean[N];
        while (valcnt < N) {
            int val = (int) (Math.random() * N);
            cardcnt++;
            if (!found[val]) {
                valcnt++;
                found[val] = true;
            }
        }
        // all N distinct cards found
        System.out.println(cardcnt);
    }
}
```

**Coupon Collector:  Mathematical Context**

**Coupon collector problem.** Given \( N \) different possible cards, how many do you have to collect before you have (at least) one of each type?

**Fact.** About \( N \left(1 + 1/2 + 1/3 + \ldots + 1/N\right) \approx N \ln N \).

\[
\text{see ORF 245 or COS 340}
\]

**Ex.** \( N = 30 \) baseball teams. Expect to wait \( \approx 120 \) years before all teams win a World Series.

**Challenge.** Debugging with arrays requires tracing many variables.
Q. Given a sequence from nature, does it have same characteristics as a random sequence?

A. No easy answer - many tests have been developed.

**Coupon collector test.** Compare number of elements that need to be examined before all values are found against the corresponding answer for a random sequence.

---

**Two-Dimensional Arrays**

**Two-dimensional arrays.**
- Table of data for each experiment and outcome.
- Table of grades for each student and assignments.
- Table of grayscale values for each pixel in a 2D image.

**Mathematical abstraction.** Matrix.

**Java abstraction.** 2D array.

---

**Two-Dimensional Arrays in Java**

**Array access.** Use \( a[i][j] \) to access element in row \( i \) and column \( j \).

**Zero-based indexing.** Row and column indices start at 0.

```java
int M = 10;
int N = 3;
double[][] a = new double[M][N];
for (int i = 0; i < M; i++) {
    for (int j = 0; j < N; j++) {
        a[i][j] = 0.0;
    }
}
```

---

**Reference:** Botstein & Brown group
Initialize 2D array by listing values.

```java
double[][] p = {
    { .02, .92, .02, .02, .02 },
    { .02, .02, .32, .32, .32 },
    { .02, .02, .02, .92, .02 },
    { .92, .02, .02, .02, .02 },
    { .47, .02, .47, .02, .02 },
};
```

**Matrix Addition**

 Görund two N-by-N matrices $a$ and $b$, define $c$ to be the N-by-N matrix where $c[i][j]$ is the sum $a[i][j] + b[i][j]$.

```java
double[][] c = new double[N][N];
for (int i = 0; i < N; i++)
    for (int j = 0; j < N; j++)
        c[i][j] = a[i][j] + b[i][j];
```

**Matrix Multiplication**

 Görund two N-by-N matrices $a$ and $b$, define $c$ to be the N-by-N matrix where $c[i][j]$ is the dot product of the $i^{th}$ row of $a[][]$ and the $j^{th}$ column of $b[][]$.

```java
double[][] c = new double[N][N];
for (int i = 0; i < N; i++)
    for (int j = 0; j < N; j++)
        for (int k = 0; k < N; k++)
            c[i][j] += a[i][k] * b[k][j];
```

**Array Challenge 2**

Q. How many scalar multiplications multiply two N-by-N matrices?

A. N  
B. $N^2$  
C. $N^3$  
D. $N^4$
Summary

Arrays.
- Organized way to store huge quantities of data.
- Almost as easy to use as primitive types.
- Can directly access an element given its index.

Ahead. Reading in large quantities of data from a file into an array.