More on Recursion

Study Sect 15.1, p. 415. Watch activity 15-2.1 on the CD.
In DrJava, write and test as many of the self-review exercises as you can (disregard those that deal with arrays).

No Labs the week of fall break (next week)

No office hours Friday, Monday, Tuesday, 8-12 October

More lunch dates scheduled on the CMS. You are invited.

A3 times

 mean 4.2 median 3.5

7..9 hours: 17
10..14 hours: 7
15..15 hours: 1

Thursday 7:30pm prelim:
A-K Olin 155 L-Z Olin 255

A game

while there is room
A draws — or |
B draws —— or |
A wants to get a solid closed curve.
B wants to stop A from getting a solid closed curve.
Who can win? What strategy to use?

A won the game to the right because there is a solid closed curve.

What does private mean?
Look it up! Index says p 155. 155-156 says: a component declared with modifier private in class C is accessible only in class C.

/** = “the female best-friend relation is mutual” — this person’s best friend thinks this person is their best friend. */
public boolean mutual() {
    return fbf!=null && this==fbf.fbf;
}

Can’t reference fbf in here

/** = if y is even then 2*y otherwise y*/
public static boolean d(int y) {
    if (y%2 == 0) {
        int k= 2 * y;
        return k;
    } else
        return y;
}

A: Never, since the argument is odd
B: Just before k=2*y is executed
C: Just the method body is executed
D: During step 1 of execution of the call

Consider the call d(5).
When is local variable k created (or drawn) during evaluation of the call?

A: Never, since the argument is odd
B: Just before k=2*y is executed
C: Just the method body is executed
D: During step 1 of execution of the call

What is the base case?
A: 0..1
B: 0..9
C: 0..99
D: 0..999
E: 0..9999

/** = non-negative n, with commas every 3 digits e.g. commafy(5341267) = “5,341,267” */
public static String commafy(int n) {
    What is the base case?
    A: 0..1
    B: 0..9
    C: 0..99
    D: 0..999
    E: 0..9999

    1: if (n < 1000)
    2: return “” + n;
    3: return commafy(n/1000) + “,” + to3(n%1000);
}

/** = if y is even then 2*y otherwise y*/
public static boolean d(int y) {
    if (y%2 == 0) {
        int k= 2 * y;
        return k;
    } else
        return y;
}

A: Never, since the argument is odd
B: Just before k=2*y is executed
C: Just the method body is executed
D: During step 1 of execution of the call

Consider the call d(5).
When is local variable k created (or drawn) during evaluation of the call?

A: Never, since the argument is odd
B: Just before k=2*y is executed
C: Just the method body is executed
D: During step 1 of execution of the call

What is the base case?
A: 0..1
B: 0..9
C: 0..99
D: 0..999
E: 0..9999

/** = non-negative n, with commas every 3 digits e.g. commafy(5341267) = “5,341,267” */
public static String commafy(int n) {
    What is the base case?
    A: 0..1
    B: 0..9
    C: 0..99
    D: 0..999
    E: 0..9999

    1: if (n < 1000)
    2: return “” + n;
    3: return commafy(n/1000) + “,” + to3(n%1000);
}

/** = p with at least 3 chars — 0’s prepended if necessary */
public static String to3(int p) {
    if (p < 10) return “” + p;
    if (p < 100) return “0” + p;
    return “” + p;
}
Recursive functions

Properties:

(1) \( b^c = b \cdot b^{c-1} \)
(2) For \( c \) even
\[ b^c = (b^2)^{c/2} \]

\( 3^3 \times 3^3 \times 3^3 \times 3^3 = (3^3)^4 = (3^3)^4 \)

Properties:

(1) \( b^c = b \cdot b^{c-1} \)
(2) For \( c \) even
\[ b^c = (b^2)^{c/2} \]

\( 3^3 \times 3^3 \times 3^3 \times 3^3 \times 3^3 = (3^3)^5 = (3^3)^5 \)

```java
public static int exp(double b, int c) {
    if (c == 0)
        return 1.0;
    if (c is odd)
        return b * exp(b, c–1);
    // c is even and > 0
    return exp(b*b, c / 2);
}
```

<table>
<thead>
<tr>
<th>c</th>
<th>number of calls</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>16</td>
<td>5</td>
</tr>
<tr>
<td>32</td>
<td>6</td>
</tr>
<tr>
<td>( 2^n )</td>
<td>( n + 1 )</td>
</tr>
</tbody>
</table>

32768 is \( 2^{15} \)
so \( b^{32768} \) needs only 16 calls!

Binary arithmetic

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Binary</th>
<th>Octal</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00</td>
<td>00</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>01</td>
<td>01</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>02</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
<td>03</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>04</td>
<td>16</td>
</tr>
<tr>
<td>5</td>
<td>101</td>
<td>05</td>
<td>32</td>
</tr>
<tr>
<td>6</td>
<td>110</td>
<td>06</td>
<td>64</td>
</tr>
<tr>
<td>7</td>
<td>111</td>
<td>07</td>
<td>128</td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>1001</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1010</td>
<td>12</td>
<td></td>
</tr>
</tbody>
</table>

Test c odd: Test last bit = 1

Subtract 1 when odd: Change last bit from 1 to 0.
Exponentiation algorithm processes binary rep. of the exponent.

Hilbert’s space-filling curve

As the size of each line gets smaller and smaller, in the limit, this algorithm fills every point in space. Lines never overlap.

All methods used in today’s lecture will be on course website.