We develop recursive functions and look at execution of recursive functions.

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).

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

Wednesday 3:35 lab is less crowded; go there instead?

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).

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

Wednesday 3:35 lab is less crowded; go there instead?

3. Find x.

4. A game

i. while there is room

A draws or B draws or

ii. 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?

Board can be any size: m by n dots, with m > 0, n > 0

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

/** = non-negative n, with commas every 3 digits
 e.g. commafy(5341267) = “5,341,267” */

public static String commafy(int n) {

// n >= 1000

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) {

// c is even and > 0

return exp(b*b, c / 2);

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>2^a</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>00</td>
<td>00</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>01</td>
<td>01</td>
<td>01</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>02</td>
<td>10</td>
<td>02</td>
<td>4</td>
<td>100</td>
</tr>
<tr>
<td>03</td>
<td>11</td>
<td>03</td>
<td>8</td>
<td>1000</td>
</tr>
<tr>
<td>04</td>
<td>100</td>
<td>04</td>
<td>16</td>
<td>10000</td>
</tr>
<tr>
<td>05</td>
<td>101</td>
<td>05</td>
<td>32</td>
<td>100000</td>
</tr>
<tr>
<td>06</td>
<td>110</td>
<td>06</td>
<td>64</td>
<td>1000000</td>
</tr>
<tr>
<td>07</td>
<td>111</td>
<td>07</td>
<td>256</td>
<td>10000000000000000</td>
</tr>
<tr>
<td>08</td>
<td>1000</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>09</td>
<td>1001</td>
<td>11</td>
<td>Test c odd: Test last bit = 1</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1010</td>
<td>12</td>
<td>Divide c by 2: Delete the last bit</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Subtract 1 when odd: Change last bit from 1 to 0.</td>
<td></td>
</tr>
</tbody>
</table>

Exponentiation algorithm processes binary rep. of the exponent.

Hilbert’s space-filling curve

Hilbert(1): As the size of each line gets smaller and smaller, in the limit, this algorithm fills every point in space. Lines never overlap.

Hilbert(2):

Hilbert(n):

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

Exponentiation algorithm processes binary rep. of the exponent.