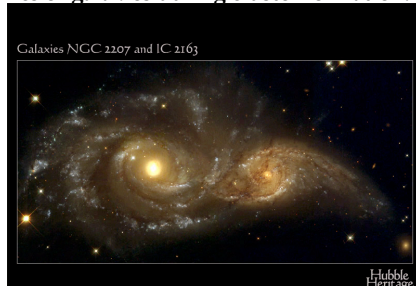


## N-Body Simulation

- Applications to astrophysics.
  - Orbits of solar system bodies.
  - Stellar dynamics at the galactic center.
  - Stellar dynamics in a globular cluster.
  - Stellar dynamics during the collision of two galaxies.
  - Formation of structure in the universe.
  - Dynamics of galaxies during cluster formation.



CompSci 100e

© Sedgewick & Wayne

1

2.1

## N-Body Simulation

- 1. Setup initial distribution of particles.
  - Need accurate data and model of mass distribution.
- 2. Compute forces between particles.
  - Direct sum:  $N^2$ .
  - Appel/ Barnes-Hut"  $N \log N$ .
- 3. Evolve particles using ODE solver.
  - Leapfrog method balances efficiency and accuracy.

$$F_i = \sum_{j \neq i} \frac{Gm_i m_j}{|r_i - r_j|^2 + \epsilon^2}$$

$\epsilon$  = softening parameter  
eliminates binary stars  
with  $r < \epsilon$   
hard binaries can be  
important  
source of energy

- 4. Display and analyze results.

$$\frac{dX_i}{dt} = V_i$$

$$m_i \frac{dV_i}{dt} = F_i$$

CompSci 100e

© Sedgewick & Wayne

2

2.2

## If Statement Examples

<i>absolute value</i>	<pre>if (x &lt; 0) x = -x;</pre>
<i>put x and y into sorted order</i>	<pre>if (x &gt; y) {     int t = x;     y = x;     x = t; }</pre>
<i>maximum of x and y</i>	<pre>if (x &gt; y) max = x; else     max = y;</pre>
<i>error check for division operation</i>	<pre>if (den == 0) System.out.println("Division by zero"); else         System.out.println("Quotient = " + num/den);</pre>
<i>error check for quadratic formula</i>	<pre>double discriminant = b*b - 4.0*c; if (discriminant &lt; 0.0) {     System.out.println("No real roots"); } else {     System.out.println((-b + Math.sqrt(discriminant))/2.0);     System.out.println((-b - Math.sqrt(discriminant))/2.0); }</pre>

CompSci 100e

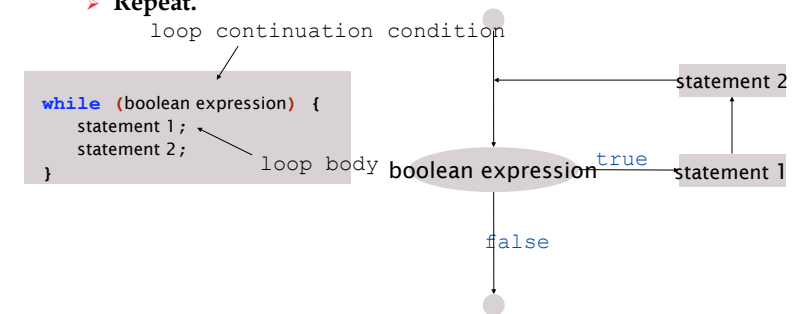
© Sedgewick & Wayne

3

2.3

## While Loop

- The while loop. A common repetition structure.
  - Check a boolean expression.
  - Execute a sequence of statements.
  - Repeat.



CompSci 100e

© Sedgewick & Wayne

2.4

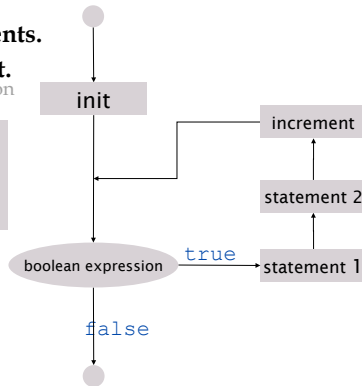
## For Loops

- The `for` loop. Another common repetition structure.

- Execute initialization statement.
- Check boolean expression.
- Execute sequence of statements.
- Execute increment statement.
- Repeat.

```
for (init; boolean expression; increment) {
    statement 1;
    statement 2;
}
```

Annotations: *loop continuation condition* points to the boolean expression; *body* points to the statements inside the curly braces.



## Loop Examples

<i>print powers of two</i>	<pre>int v = 1; for (int i = 0; i &lt;= N; i++) {     System.out.println(i + " " + v);     v = 2*v; }</pre>
<i>print largest power of two less than or equal to N</i>	<pre>int v = 1; while (v &lt;= N/2)     v = 2*v; System.out.println(v);</pre>
<i>compute a finite sum (1 + 2 + ... + N)</i>	<pre>int sum = 0; for (int i = 1; i &lt;= N; i++)     sum += i; System.out.println(sum);</pre>
<i>compute a finite product (N! = 1 × 2 × ... × N)</i>	<pre>int product = 1; for (int i = 1; i &lt;= N; i++)     product *= i; System.out.println(product);</pre>
<i>print a table of function values</i>	<pre>for (int i = 0; i &lt;= N; i++)     System.out.println(i + " " + 2*Math.PI*i/N);</pre>
<i>print the ruler function (see Program 1.2.1)</i>	<pre>String ruler = " "; for (int i = 1; i &lt;= N; i++)     ruler = ruler + i + ruler; System.out.println(ruler);</pre> <p>© Sedgewick &amp; Wayne</p>

## Problem 1

- Given  $n$ , calculate  $2^n$ 
  - What if you wanted to print all from  $2^0$  to  $2^n$ ?
  - What if you wanted to return the value?

## While Loops: Powers of Two

- Ex. Print first  $n$  powers of 2.
  - Increment  $i$  from 1 to  $n$ .
  - Double  $v$  each time.

```
int i = 0;
int v = 1;
while (i <= N) {
    System.out.println(v);
    i = i + 1;
    v = 2 * v;
}
```

$i$	$v$	$i \leq N$
0	1	true
1	2	true
2	4	true
3	8	true
4	16	true
5	32	true
6	64	true
7	128	false

Console:

```
1
2
4
8
16
32
64
```



$n = 6$

## Problem 2

- **Square Root:**
  - Given a real number  $c$  and some error tolerance  $\epsilon$
  - Estimate  $t$ , the square root of  $c$

## While Loops: Square Root

- **Q.** How might we implement `Math.sqrt()` ?
- **A.** To compute the square root of  $c$ :
  - Initialize  $t_0 = c$ .
  - Repeat until  $t_i = c/t_i$ , up to desired precision:  
set  $t_{i+1}$  to be the average of  $t_i$  and  $c/t_i$ .

$t_0$	=	2.0
$t_1$	=	$\frac{1}{2}(t_0 + \frac{2}{t_0}) = 1.5$
$t_2$	=	$\frac{1}{2}(t_1 + \frac{2}{t_1}) = 1.4166666666666665$
$t_3$	=	$\frac{1}{2}(t_2 + \frac{2}{t_2}) = 1.4142156862745097$
$t_4$	=	$\frac{1}{2}(t_3 + \frac{2}{t_3}) = 1.4142135623746899$
$t_5$	=	$\frac{1}{2}(t_4 + \frac{2}{t_4}) = 1.414213562373095$

computing the square root of 2

## Problem 3

- **Gambler's ruin.** Gambler starts with  $\$stake$  and places  $\$1$  fair bets until going broke or reaching  $\$goal$ .
  - What are the chances of winning?
  - How many bets will it take?

## Gambler's Ruin

- **One approach.** Monte Carlo simulation.
  - Flip digital coins and see what happens.
    - Pseudorandom number generation
    - `java.util.Random`
  - Repeat and compute statistics.

