

Week 5 - Friday

COMP 2100

Last time

- What did we talk about last time?
- Recursion

Questions?

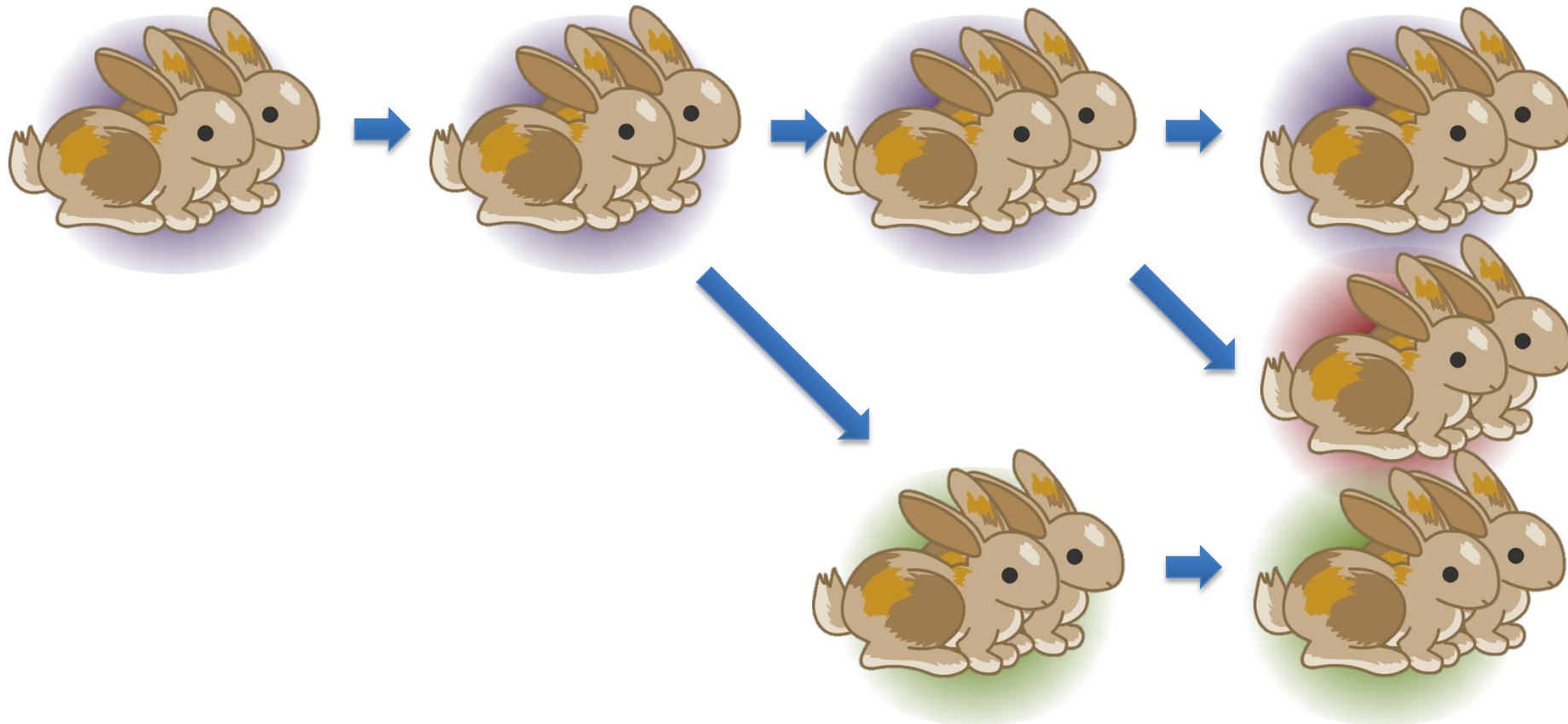
Project 2

Assignment 3

Issues of Efficiency

Fibonacci

- The sequence: 1 1 2 3 5 8 13 21 34 55...
- Studied by Leonardo of Pisa to model the growth of rabbit populations



Fibonacci problem

- Find the n^{th} term of the Fibonacci sequence
- Simple approach of summing two previous terms together
- Example: $n = 7$

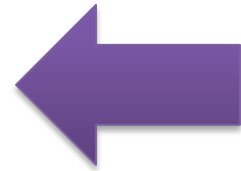
■	1	1	2	3	5	8	13
	1	2	3	4	5	6	7

Recursion for Fibonacci

- Base cases ($n = 1$ and $n = 2$):
 - Result = 1
- Recursive case ($n > 2$):
 - Result = fibonacci($n - 1$) + fibonacci($n - 2$)

Code for Fibonacci

```
public static int fib( int n ) {  
  
    if (n <= 2) {  
        return 1;  
    } else {  
        return fib(n - 1) + fib(n - 2) ;  
    }  
}
```



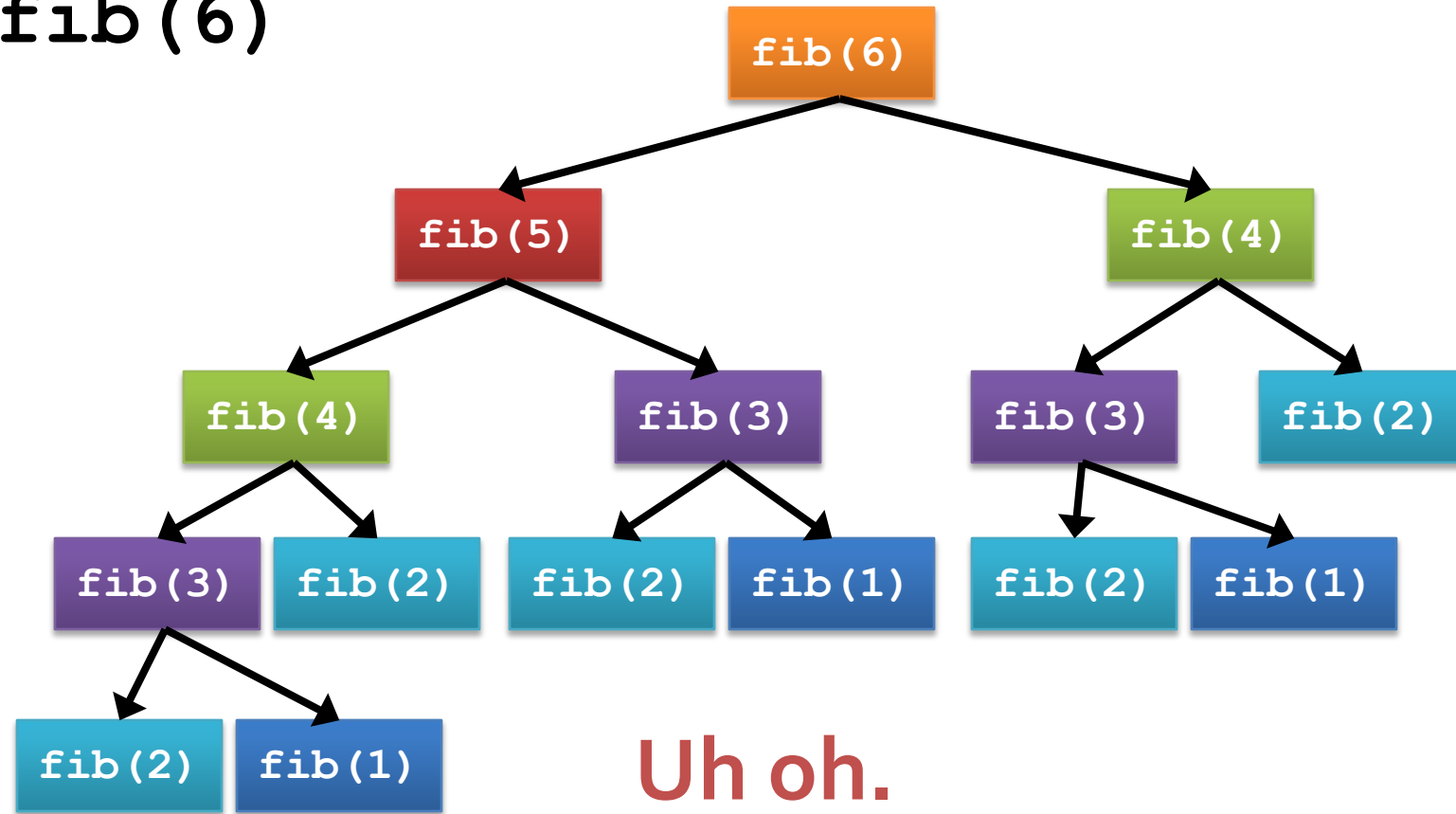
Base Case



Recursive
Case

What's the running time for `fib()`?

- Example: **fib**(6)



Uh oh.

Exponential time for `fib`

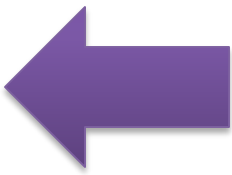
- For most cases, calling `fib()` makes calls two more calls to `fib()`, which each make two more calls to `fib()`, and so on...
- Many values are redundantly computed
- The final running time is $O(2^{n/2})$

Can we do better?


- The recursion is fine from a mathematical perspective
- We just need to avoid recomputing lower terms in the sequence
- We can use the idea of carrying along both the $(n - 1)$ term and the $(n - 2)$ term in each recursive step

Code for better Fibonacci

```
public static int fib2( int a, int b, int n ) {  
  
    if (n <= 2) {  
        return b;  
    } else {  
        return fib2(b, a + b, n - 1);  
    }  
}  
  
// proxy method  
int fib( int n ) {  
    return fib2(1, 1, n);  
}
```



Base Case



Recursive Case

Exponentiation

- We want to raise a number x to a power n , like so: x^n
- We allow x to be real, but n must be an integer greater than or equal to 0
- Example: $(4.5)^{13} = 310286355.9971923828125$

Recursion for exponentiation

- Base case ($n = 0$):
 - Result = 1
- Recursive case ($n > 0$):
 - Result = $x \cdot x^{(n-1)}$

Code for exponentiation

```
public static double power(double x, int n) {
```

```
    if (n == 0) {
```

```
        return 1;
```

```
    } else {
```

```
        return x * power(x, n - 1);
```

```
    }
```

```
}
```

 Base Case


Recursive
Case

Running time for power

- Each call reduces n by 1
- $n + 1$ total calls
- What's the running time?
 - $\Theta(n)$

Review

Week 1

- Programming model
- Java
 - OOP
 - Polymorphism
 - Interfaces
 - Exceptions
 - Generics
- Java Collections Framework

Week 2

- Big Oh Notation
 - Formal definition: $f(n)$ is $O(g(n))$ if and only if
 - $f(n) \leq c \cdot g(n)$ for all $n > N$
 - for **some** positive real numbers c and N
 - Worst-case, asymptotic, upper bound of running time
 - Ignore lower-order terms and constants
- Big Omega and Big Theta
- Abstract Data Types
- Array-backed list

Week 3

- Stacks
 - FILO data structure
 - Operations: push, pop, top, empty
 - Dynamic array implementation
- Queues
 - FIFO data structure
 - Operations: enqueue, dequeue, front, empty
 - Circular (dynamic) array implementation
- JCF implementations: **Deque<T>** interface
 - **ArrayDeque<T>**
 - **LinkedList<T>**

Week 4

- Linked lists
 - Performance issues
 - Single vs. double
 - Insert, delete, find times
- Linked list implementation of stacks
- Linked list implementation of queues

Sample Problems

Running time

- Let M and N be two integers, where M is no larger than N
- Use Big Θ notation to give a tight upper bound, in terms of N , on the time that it will take to
 - Add M and N (by hand, using the normal algorithm)
 - Multiply M and N (by hand, using the normal algorithm)
- Use Big Θ notation to give the same bounds but this time in terms of n , where n is the number of digits in N

What's the running time in Θ ?

```
int end = n;
int count = 0;
for (int i = 1; i <= n; ++i) {
    end /= 2;
    for (int j = 1; j <= end; ++j) {
        count++;
    }
}
```

What's the running time in Θ ?

```
int end = n;  
int count = 0;  
for (int i = 1; i <= n*n; i += 2) {  
    count++;  
}
```

What's the running time in Θ ?

```
int count = 0;
for (int i = 1; i <= n; ++i) {
    for (int j = 1; j <= n/j; ++j) {
        count++;
    }
}
```

Lighten

- If we increase the R, G, and B values of every pixel by 25%, the image will get lighter

- Let **Color** be the following:

```
public class Color {  
    public int red;  
    public int green;  
    public int blue;  
}
```

- Let **pixels** be a **Color**[][] array with **height** rows and **width** columns
- Write the code to lighten the image by 25% (by multiplying by 1.25)
- Don't forget to round the results before storing them back into each color component

Reverse a linked list

Assume the following:

```
public class List {  
    private static class Node {  
        public int data;  
        public Node next;  
    }  
  
    private Node head = null;  
    ...  
}
```

Write a method in **List** that reverses the linked list.

Code to reverse a linked list

```
public void reverse() {  
    if (head != null) {  
        Node reversed = head;  
        Node temp = head;  
        Node rest = head.next;  
        temp.next = null;  
        while (rest != null) {  
            temp = rest;  
            rest = rest.next;  
            temp.next = reversed;  
            reversed = temp;  
        }  
        head = reversed;  
    }  
}
```

Palindrome

- Write a method that takes a **CharBuffer** object
- The **CharBuffer** object has two methods:
 - **nextChar()** which extracts a char from the input stream
 - **hasNextChar()** which returns true as long as there is another char to extract
- The method should return **true** if the input stream is a palindrome (the same backwards as forwards) and **false** otherwise
- Use no **String** objects or arrays (other than the ones embedded in the stack)
- Hint: Use at least 3 JCF **Deque** (stack) objects

Ticket out the Door

Upcoming

Next time...

- Exam 1 on Monday!

Reminders

- Office hours from 1:45-4 p.m. canceled today
- Review for Exam 1
- Keep working on Project 2