

Recursion:

I'mma break yo brain

CTEC 243

What is?

- A recursive method is a method that calls itself

Why?

- Recursive solutions can sometimes be more “elegant” than imperative ones
 - Easier to read
 - Easier to understand
 - Avoids side-effects
- Recursive solutions can often be slower than imperative ones
 - How StackOverflow.com got its name

Understanding Recursion should help you understand other programming concepts, even if you don't use it as often.

Solving Problems with Recursion

Recursion Terms

- Recursion Depth - how many times a method calls itself
- Base Case - a scenario in which the problem can be solved without recursion
- Recursive Case - a scenario you can use recursion to solve
- Direct Recursion - Recursion in which a method directly calls itself
- Indirect Recursion - Recursion in which a chain of two or more method calls returns to the method that originated the chain. For example, method A calls method B which in turn calls method A.

Three Questions

- Base-Case Question - is there a non-recursive way out?
- Smaller-Caller Question - does every recursive call move towards the Base-Case?
- General-Case Question - does it do what you want it to?

Factorials

$$n! = 1 * 2 * 3 * 4 * \dots * n$$

$$0! = 1$$

- How can we redefine this?
 - Hint: $5! = 4! * 5$
- In what cases do we not need to do any more calculations?
- How can we work towards that base case?

So yeah, let's code it

Calculate 4!

4! =



Calculate 4!

$$4! = 4 \times (4 - 1)!$$

$$4! = 4 \times 3!$$

$$4! = 4 \times \square$$



Calculate 4!

4!

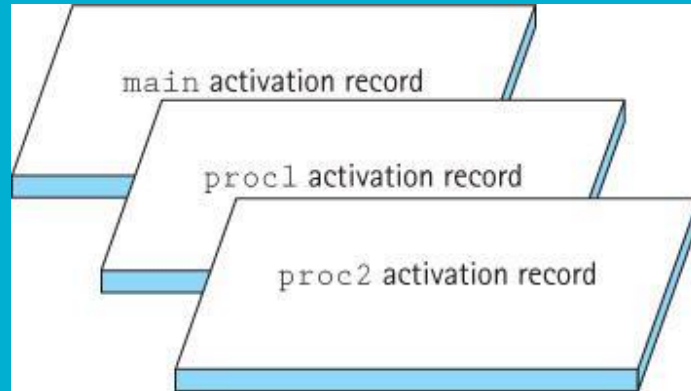
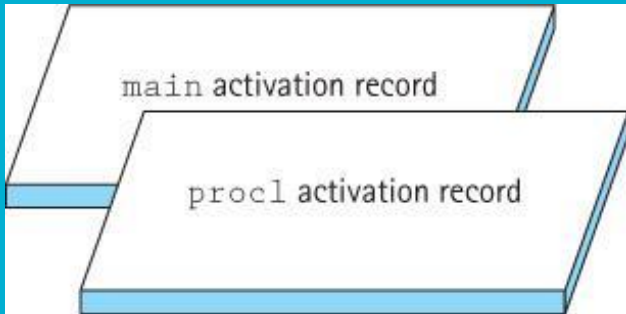
Calculate 3!

4!

3! =

4!

Dynamic Storage Allocation



On Tail Recursion

- A recursive function is tail recursive when recursive call is the last thing executed by the function
- This can help optimize memory
 - ...but not in Java
- Source/More Information - <https://www.geeksforgeeks.org/tail-recursion/>

The Accumulator

- An extra parameter you define and pass in your recursive methods
- “Accumulates” information in each call

Let's rewrite the Factorial sequence to use an accumulator.

Iterative Solutions

- If a recursive algorithm is Tail-Recursive, then it could be replaced with a while loop
- If a recursive algorithm is not Tail-Recursive, then you may still be able to create an iterative solution
- In many languages, Iterative solutions are more efficient than Recursive ones

Common Errors

- Forgetting a base case
- Not reducing the problem with each recursive call
- Writing the recursive call in such a way that the base case is never reached

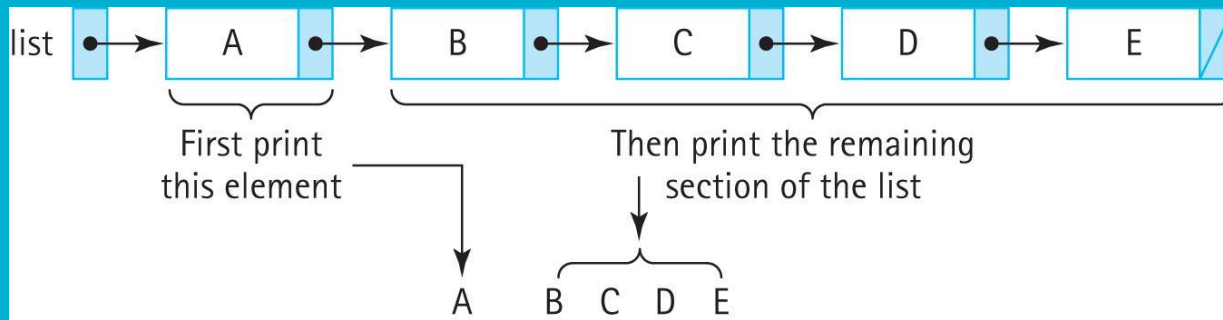
Linked Lists

Recursive Processing of Linked Lists



- A linked list is a recursive structure

Printing a Linked List Recursively



```
void recPrintList(LLNode<String> listRef)
{
    if (listRef != null)
    {
        System.out.println(listRef.getInfo());
        recPrintList(listRef.getLink());
    }
}
```

Recursion Examples

Summing a Range of Array Elements

- `rangeSum(arrayInQuestion, startingIndex, endingIndex)`

The Fibonacci Series

- Look, I'll admit this one is a little annoying
- The Fibonacci Series - a list of numbers in which the next number is made up of the sum of the last two numbers
 - 0,1,1,2,3,5,8,13, 21... and so on
- Let's break our brains on this for a bit
 - We want to find the number in the n th place of the sequence
 - Yes, this is a common programming problem you can easily look up online, but let's just think for a moment
 - I'll even put a hint on the next slide (no peeking).

Fibonacci Hints

- $\text{Fibonacci}(0) = 0$
- $\text{Fibonacci}(1) = 1$
- $\text{Fibonacci}(n) = \text{Fibonacci}(n-1) + \text{Fibonacci}(n-2)$ for $n \geq 2$

Let's write a Fibonacci function. Remember, we don't need to output the whole sequence, we just want a function that tells us what the n th number in the sequence is.

Recursive Binary Search

- This one will require a sorted array
 - If the value is in the middle, return the middle value
- If the middle value is greater than the sought value, search the lower half of the array
- If the middle value is smaller than the sought value, search the greater half of the array

Practice

Recursive Multiplication

Write a recursive function that accepts two arguments into the parameters x and y. The function should return the value of x times y.

Hint: multiplication can be performed as repeated addition.

$$4 * 2 = 2 + 2 + 2 + 2$$

Basset Hound Ears

Given n number of basset hounds, calculate how many droopy basset hound ears there should be. Use recursion to solve the problem.

All Star

Given a string, compute recursively a new string where all the adjacent chars are now separated by a “*”.

allStar(“hello”) -> “h*e*l*l*o”

allStar(“abc”) -> “a*b*c”

allStar(“ab”) -> “a*b”

