Recursion: The Black Magic of Programming
Learning the Dark Art

- Theory
- Definition
- Constructing a Recursive Function Definition
- Practice
- Implementation and Data Structures
Theory
Definition

Recursion: See recursion

But I spelled ‘recursion’ correctly...
Recursive Function:

a function that is defined in terms of itself
Black Magic

You have seen such dark mathemagic before!
The Fibonacci Numbers
Fibonacci Numbers

How are the Fibonacci Numbers defined?

Fib(0) = 0
Fib(1) = 1
Fib(2) = 1
Fib(3) = 2

The Dark Sequence: 0, 1, 1, 2, 3, 5, 8, 13, 21...
Fibonacci Numbers

Can we generalize this progression?

Fib(n) = ?
Fibonacci Numbers

Can we generalize this progression?

\[ \text{Fib}(n) = \text{Fib}(n-1) + \text{Fib}(n-2) \]

However, I claim that this alone is a poorly cast spell!
Casting Recursive Spells

The novice witch or wizard is overly concerned with the recursive power of their spell and they forget:

- A strong base for their spell!
Casting Recursive Spells

A proper recursive spell is composed of:

- A series of one or more base cases
- A series of one or more recursive spells
Fibonacci Numbers

The Correct Fibonacci Spell

A strong spell base

Fib(0) = 0
Fib(1) = 1

The recursive spell

Fib(n) = Fib(n-1) + Fib(n-2)
Conjuring Practice

A crafty wizard has proposed the following challenge: “What is the sum of the first n integers greater than zero?”

Unfortunately, your wand does not have the ability to perform loop iterations or multiplication/division, only addition.

Assumption: You are not the powerful mathemagician Carl Frederich Gauss.
Conjuring Practice

We must divide this challenge up into smaller challenges that we can solve easily:

\[ \text{GaussSum}(n) = n + \text{GaussSum}(n-1) \]

Lower your wands! What have we forgotten?
Remember: A powerful recursive spell must have one or more base cases.

What is the GaussSum(n) base case?

GaussSum(0) = 0
public static int GaussSum(int n) {
    if (n == 0) {
        return 0;
    }
    return (n + GaussSum(n - 1));
}
Gauss' Solution

Rewrite the problem as follows:

\[(100+1)+(99+2)+(98+3)+\ldots\]

Note that each quantity sums to 101, which is \(n+1\)

Note that there are 50, or \(n/2\), quantities

In general, GaussSum\((n)\) = \((n+1)(n/2)\)
Stacks

A stack is a “last-in first-out” data structure similar to a stack of papers

i.e. The bottom sheet can only be removed after all sheets above it have been removed

Method calls use stacks with entries called activation records, rather than sheets
Stacks

Consider what happens when a call to MethodB is made from MethodA.

The information relevant to MethodA must be saved so that when we return from MethodB, we can resume execution.
Stacks

In order to gain insight into how recursion correctly arrives at the solution, we will trace through a sample execution of the GaussSum spell.
public static int GaussSum(int n) {
    if (n == 0) {
        return 0;
    }
    return (n + GaussSum(n-1));
}

In: n=Q

return (n + GaussSum(n-1));
In: n=0
return 0

In: n=1
return (1 + GaussSum(1-1))

In: n=1
return (1 + GaussSum(1-1))

In: n=1
return (1 + GaussSum(1-1))

In: n=2
return (2 + GaussSum(2-1))

In: n=2
return (2 + GaussSum(2-1))

In: n=2
return (2 + GaussSum(2-1))

In: n=2
return (2 + GaussSum(2-1))

In: n=3
return (3 + GaussSum(3-1))

In: n=3
return (3 + GaussSum(3-1))

In: n=3
return (3 + GaussSum(3-1))

In: n=3
return (3 + GaussSum(3-1))

In: n=3
return (3 + GaussSum(3-1))

In: n=3
return (3 + GaussSum(3-1))

In: n=3
return (3 + GaussSum(3-1))

In: n=3
return (3 + GaussSum(3-1))

In: n=3
return (3 + GaussSum(3-1))

In: n=3
return (3 + GaussSum(3-1))
When Spells Go Wrong
Infinite Recursion

When a recursive spell is cast without properly defined base cases, the recursion continues ad infinitum.
Infinite Recursion

```java
public static int Fib(int n) {
    fib = Fib(n-1) + Fib(n-2)
    if (n == 0)
        return 0;
    if (n == 1)
        return 1;
    return fib;
}
```

What is wrong?

Stack Overflow

Note: Space Turtles!
Efficiency
Efficiency

A skilled wizard or witch knows that their spell must manifest quickly to defeat their opponent.

As fate would have it, both the Fibonacci Sequence and the Gaussian Sum should NEVER be computed recursively!
Efficiency

The Fibonacci Sequence, in particular, generates too many recursive calls to be useful

Solution?

Iterative Approach

Always consider switching to an iterative solution when recursion is too costly
When To Use Recursion

When is recursion useful?

- Recursive solutions are often more concise
- Some problems have natural recursive solutions
- Iterative solution would be convoluted
A Proper Recursion Application
The Challenge

A crafty wizard has proposed the following challenge: “Given an array of sorted numbers, find the location of element X”

To check if element $I = X$, you tap I with your wand and yell “Prae Quam”!

Note: “Prae Quam” is Latin for “Comparison”
The Challenge

However, to defeat the wizard, you must GUARANTEE to use less than $N$ uses of your wand to perform a comparison (lest your magic be exhausted)

How do we win the challenge?
The Challenge

Key Observation: If \( X \) is greater than the middle element of the list, it is in the right half. Otherwise, it is in the left half.

Can anyone think of a recursive solution?
Binary Search

Solution:

1. Compare $X$ with the middle element of the list.
2. Recursively search the proper half of the list.
X = 63

Binary Search

Prae Quam!

DONE!
public int BinS(int[] a, int f, int l, int k) {
    if (f > l) return -1;
    else {
        int mid = (f + l) / 2;
        if (key == a[mid]) return mid;
        if (key < a[mid])
            return BinS(a, f, mid - 1, k);
        if (key > a[mid])
            return BinS(a, mid + 1, l, k);
    }
}

In O(log n) Time!
You have Mastered Recursive Spells

No Infinite Recursion!

Who has actually seen Fantasia?