Competitive Programming

Lesson 4 – Basic Recursion

Written by Jason Sun

and Puneet Bhat

1

Recursion

A way to define a problem in terms of itself.

What is recursion?

Recursion is when a **function calls itself**. You would need this when, to perform the function's task, the function first needs to perform another function call.

It is a way to break a seemingly large problem into smaller bits and pieces that are to be solved in steps. In other words, it is all about **defining a problem in terms of itself**.

Simple Recursion Example

```
1    def sumOfFirstNums(n):
2        if n <= 1:
3            return 1
4        else:
5            return n + sumOfFirstNums(n - 1)
```

This is similar to PSA.

Check Your Understanding

```
1    def sumOfSquares(n):
2        total = 0
3        while n > 0:
4            total += n * n
5            n -= 1
6        return total
```

Does this function use recursion to complete its tasks?

2

Laws for Recursion

How do we make a working recursive function?
Well, there are a few rules!

1 – Start

A recursive algorithm must call itself, recursively.

2 – End

A recursive algorithm must have a base case.

3 – Progress

A recursive algorithm must change its state and move toward the base case.

Base Case?

What's a base case? Well, it's just something **you know for sure** given that scenario!

When a case is **not a base case**, that means you don't know what to do or return, and instead must **perform some recursion** to find the answer or do what is needed.

Check Your Understanding

What is the output of this code if **n = 2**?

```
1    def factorial(n):
2        return n * factorial(n-1)
3
4    n = int(input())
5    print(factorial(n))
```

You would get an error. Why?
Which recursive law was broken?

3

Applications of Recursion

Some common applications of recursion, and things to consider and remember.

Subproblems

Some tasks can be broken down, and solved by solving those smaller problems.

For example, what is the **n**th term of the fibonacci sequence?
Fibonacci sequence: 1, 1, 2, 3, 5, 8, 13, ...

Where each next term is the sum of the previous two terms, and **f(1) = f(2) = 1**.

$$f(\mathbf{n}) = f(\mathbf{n} - 1) + f(\mathbf{n} - 2)$$

```
Function fibo(n):
        If n <= 2:
                Return 1
        Else:
                Return fibo(n-1) + fibo(n-2)
```

What is fibo(7)?
Answer = 13

This is just an example. Recursion is not the fastest way to solve this problem!

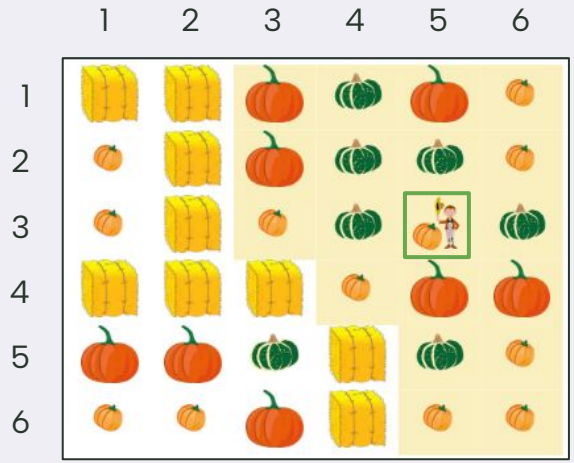
Try to think of a way to solve this problem in O(n) time!

Hint: Try to start calculating the sequence on paper and see how fast that is?

Graph Traversal

You can simulate travelling or exploring through recursion! Although this involves **graph theory**, which hasn't been learnt yet, but recursion can be used intuitively like so.

For example, you can use recursion to simulate **travelling** across a **grid**! The example below is floodfill (travelling to all reachable places from a start point).




```
Function explore(row, col):
        If the cell is not haybale:
                If haven't explored this cell:
                        Do anything we need to do for this cell
                        For each r, c in cells around:
                                explore(r, c)

explore(3, 5) # the starting point: row = 3, col = 5
```

Try To Solve!

Given any mathematical equation with addition, subtraction, multiplication, division, and brackets, find the simplified value.

How would you solve this normally, with just math?

Solution

We actually use recursion! Given, for example:

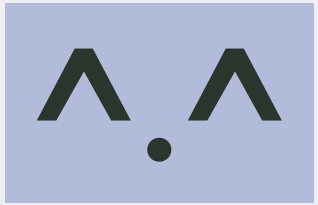
```
1   | (2 + 3) * (5 - 1) = ?
2.1 | (2 + 3) = 5
2.2 | (5 - 1) = 4

1   | (2 + 3) * (5 - 1) = ?
1   | 5 * 4 = ?
1   | 5 * 4 = 20
```

Visualization!

“5 Simple Steps for Solving Any Recursive Problem”
Reducible

https://www.youtube.com/watch?v=ngCos392W4w



Practice

If you want to truly improve, practice! Do the homework as well as some DMOJ questions outside of class. It can be a fun hobby!

Homework

Junior

1) Word Scrambler - https://dmoj.ca/problem/ics4p1
2) Rimuru's Number Game - https://dmoj.ca/problem/occ19s2
3) Tests or Test Cases? - https://dmoj.ca/problem/ccc96s3

Senior

1) CCC '24 J5 | Harvest Waterloo - https://dmoj.ca/problem/ccc24j5
2) Cheeky Checkers - https://dmoj.ca/problem/uacc1p3
3) Ctudor's Cute Orchids - https://dmoj.ca/problem/valentines18j5s2
4) CCC '05 J5 - Bananas - https://dmoj.ca/problem/ccc05j5

Thanks!

Do you have any questions?

https://github.com/CryoJS/DSA-Handbook