



UC Berkeley EECS
Lecturer SOE
Dan Garcia

CS10 : The Beauty & Joy of Computing

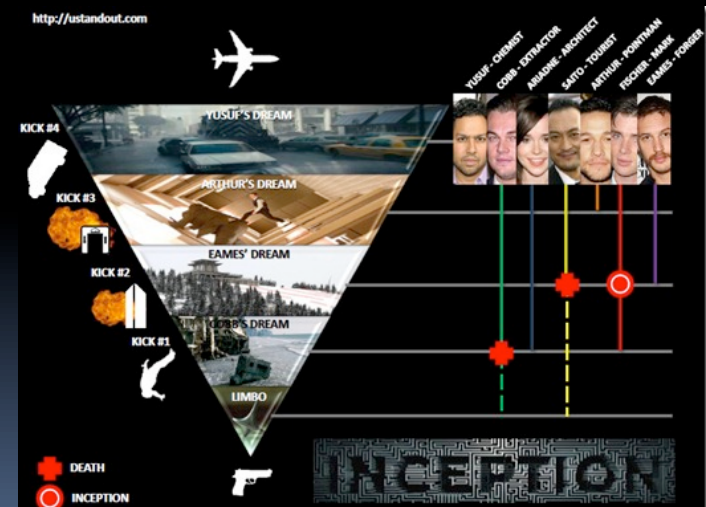
Lecture #9 : Recursion

2012-02-02



GO SEE INCEPTION!

The coolest movie last year highlights recursion, and it was up for best picture. If you haven't seen it yet, you should, because it will help you understand recursion!!



New Rule: Use scratch paper in lab!

The problems there are hard enough that you won't be able to keep it in your head!



Overview

■ Recursion

▣ Demo

- Vee example & analysis
- Downup

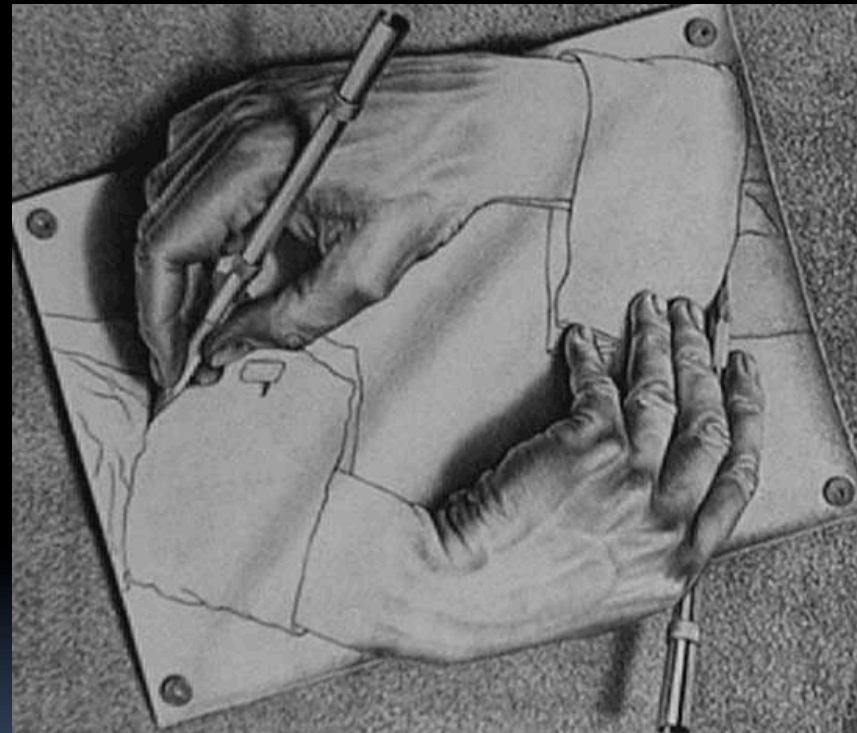
▣ You already know it

▣ Definition

▣ Trust the Recursion!

▣ Conclusion

M. C. Escher : *Drawing Hands*





"I understood Vee & Downup"

- a) Strongly disagree
- b) Disagree
- c) Neutral
- d) Agree
- e) Strongly agree



M. C. Escher : *Fish and Scales*





Definition


- **Recursion: (noun) See recursion.** 😊
- *An algorithmic technique where a function, in order to accomplish a task, calls itself with some part of the task*
- **Recursive solutions involve two major parts:**
 - **Base case(s)**, the problem is simple enough to be solved directly
 - **Recursive case(s)**. A recursive case has three components:
 - **Divide** the problem into one or more simpler or smaller parts
 - **Invoke** the function (recursively) on each part, and
 - **Combine** the solutions of the parts into a solution for the problem.
- **Depending on the problem, any of these may be trivial or complex.**





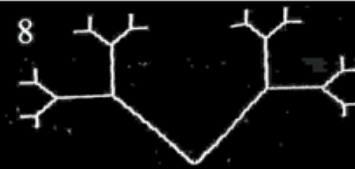
You already know it!

1




There is a little green house
And inside the little green house
There is a little brown house
And inside the little brown house
There is a little yellow house
And inside the little yellow house
There is a little white house
And inside the little white house
There is a little red heart
Warm and loving.


8




12



13



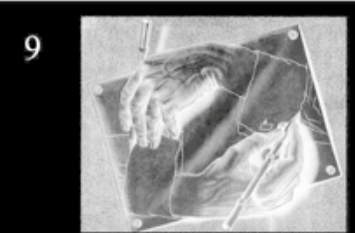
3




2

$$n! = n \cdot (n - 1)!$$

9



10




14

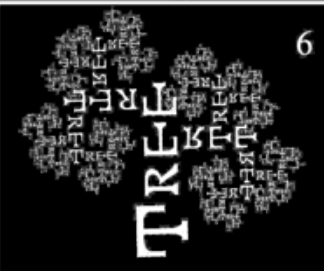
*Mother Goose Rhyme
Myself*

As I walked by myself
And talked to myself,
Myself said unto me:
"Look to thyself,
for nobody cares for thee."
I answered myself
And said to myself
In the selfsame repartee:
"Look to thyself,
Or not look to thyself,
The selfsame thing will be."

4




6




A KING IS A SON OF A KING

5



7



IF ALL WERE ONE

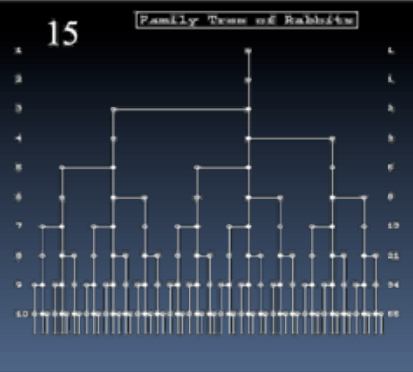
If all the seas were one sea,
What a great sea that would be!
And if all the trees were one tree,
What a great tree that would be!
And if all the men were one man,
What a great man he would be!
And if the great man took the great axe,
And cut down the great tree,
And let it fall into the great sea,
What a splash splash that would be!

11

55555
4444
333
22
1
22
333
4444
55555

15

Family Tree of Rabbits



(c) 2001, Task & Graphic Design: Dali Levy, Technion - Israel Institute of Technology

Contact: Levy.dalit@gmail.com





Trust the Recursion

- **When authoring recursive code:**
 - The base is usually easy: “when to stop?”
 - In the recursive step
 - How can we break the problem down into two:
 - A piece I can handle right now
 - The answer from a smaller piece of the problem
 - Assume your self-call does the right thing on a smaller piece of the problem
 - How to combine parts to get the overall answer?
- **Practice will make it easier to see idea**

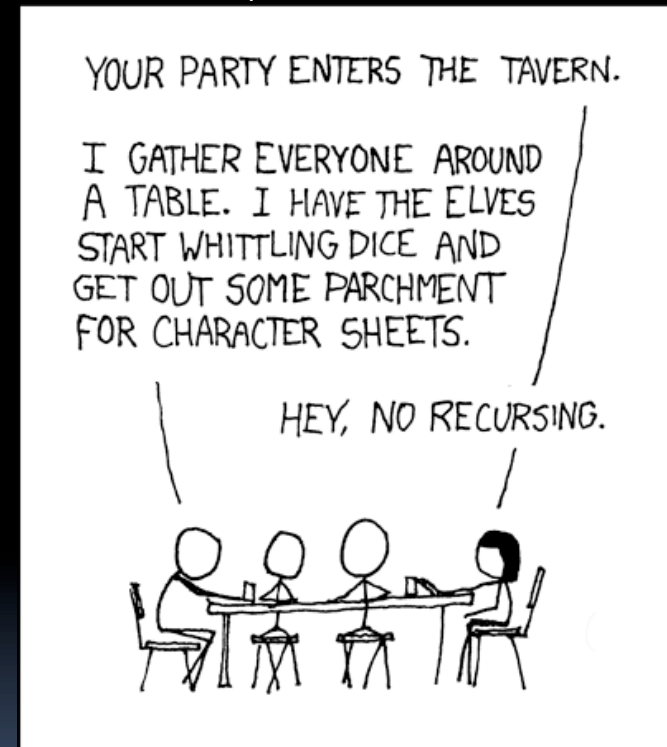




Sanity Check...

- Recursion is ■ Iteration (i.e., loops)
- Almost always, writing a recursive solution is ◆ than an iterative one
 - a) more powerful than, easier
 - b) just as powerful as, easier
 - c) more powerful than, harder
 - d) just as powerful as, harder

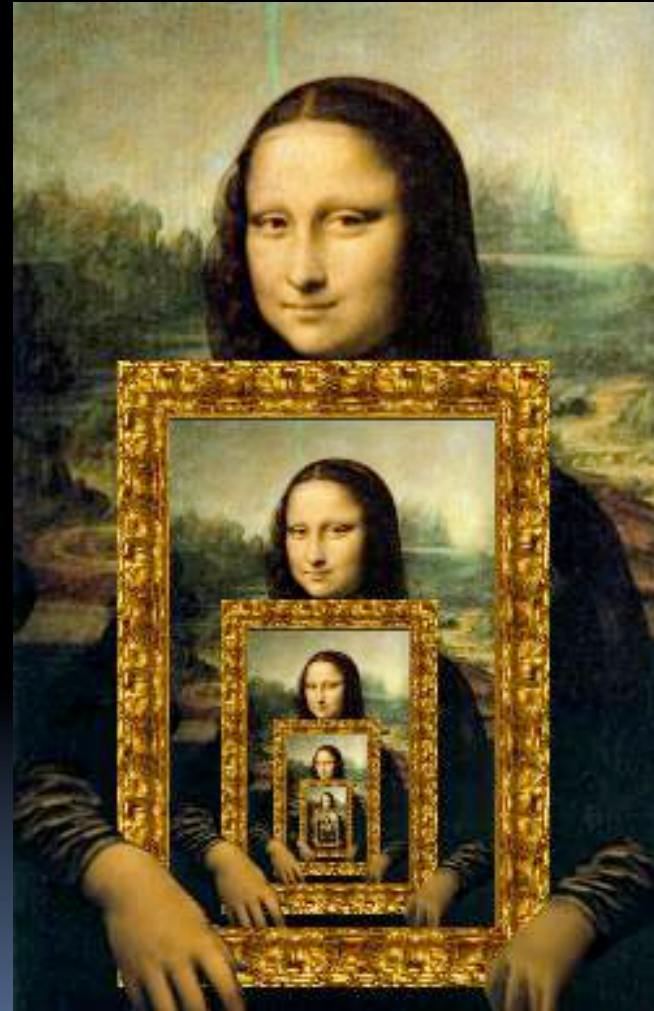
<http://xkcd.com/244/>





Summary

- Behind Abstraction, **Recursion is probably the 2nd biggest idea about programming in this course**
- It's tremendously useful when the problem is self-similar
- It's no more powerful than iteration, but **often leads to more concise & better code**



<http://www.dominiek.eu/blog/?m=200711>

Garcia, Spring 2012

