

# Recursividad

## CI-0112 Programación 1

Sivana Hamer - sivana.hamer@ucr.ac.cr  
Escuela de Ciencias de la Computación e Informática  
Universidad de Costa Rica  
Licencia: CC BY-NC-SA 4.0



# RECURSION

It recurs.

UNIVERSIDAD DE COSTA RICA

sivana.hamer@ucr.ac.cr

1

La recursividad define algo en términos de sí mismo

Al programar la recursividad es cuando una función se invoca a si mismo, directa o indirectamente

A screenshot of a Google search results page. The search bar at the top contains the query "recursion". Below the search bar, there are several search result snippets. One snippet from Wikipedia is highlighted with a red box and the text "Did you mean: recursion?". Other snippets mention "recursion in computer science" and "recursion in mathematics". At the bottom of the page, there is a link to "Did you mean: recursion?".

Función recursiva  
llamada *countdown*

↓  
public void countdown(int n) {

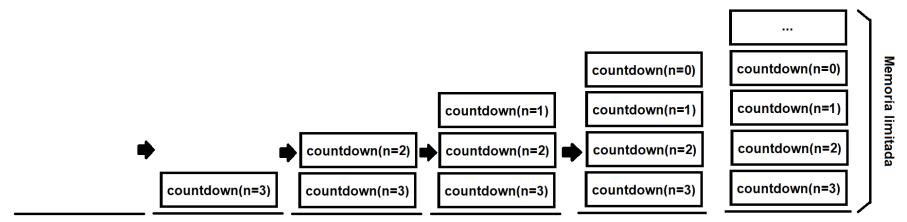
    countdown(n-1); ← Caso recursivo de  
} *countdown* directa

Pero, si corremos el programa sucede lo siguiente...

```
C:\Users\sivan\source\repos\programming-i>java Blastoff
Exception in thread "main" java.lang.StackOverflowError
at Blastoff.countdown(Blastoff.java:3)
```



Cada vez que se invoca una función se guarda en la pila de invocaciones



Ocupamos un caso base que detiene las llamadas recursivas

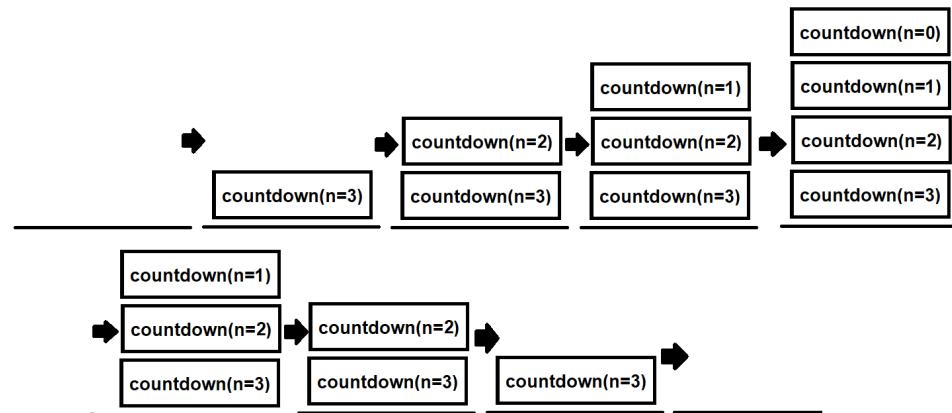
Función recursiva  
llamada `countdown`

```
↓
public void countdown(int n) {
    if (n == 0){
        System.out.println("Blastoff!");
    } else {
        countdown(n-1); ← Caso recursivo de
                           countdown
    }
}
```

Caso base para  
 $n$  igual a 0 de  
`countdown`

Caso recursivo de  
`countdown`

Con en caso base, se vacia la pila de invocaciones y pasar los resultados de una invocación



Leonardo de Pisa (Fibonacci) intento resolver el problema de cuantos conejos se pueden reproducir idealmente...



1

Veamos en más detalle la recursión  
al programar con un ejemplo:  
Fibonacci

Si empezamos a analizar la secuencia de Fibonacci podemos ver lo siguiente...

Resultado	Mes	Parejas
$f_0$	0	1
$f_1$	1	1
$f_2$	2	2
$f_3$	3	3
$f_4$	4	5
$f_5$	5	8
$f_6$	6	13
$f_7$	7	21
...	...	...

Buscando una relación entre los datos, podemos notar que se puede definir mediante la siguiente secuencia...

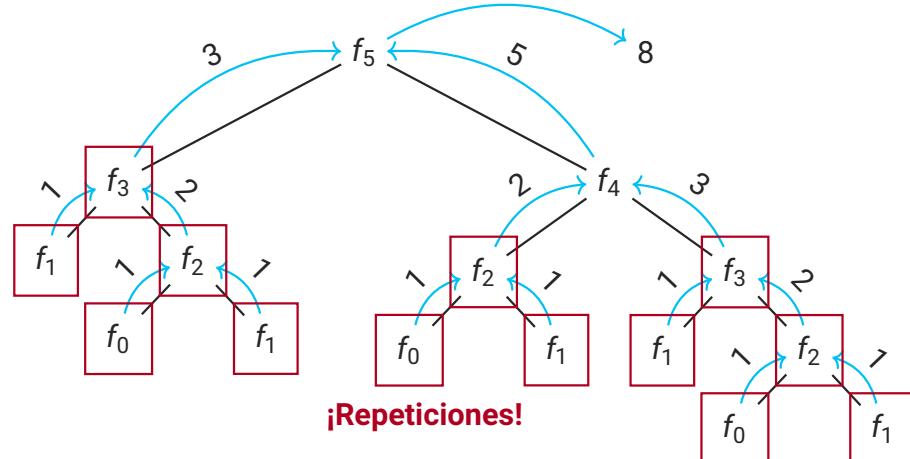
$$f_n = \begin{cases} 1 & 0 \leq n \leq 1 \\ f_{n-2} + f_{n-1} & n > 1 \end{cases}$$

Con la recursividad se puede resolver el problema pensando en los casos más básicos, usando su solución para los siguientes casos (Bottom-up recursivo)

## Ahora implementemos Fibonacci en código

```
public int fibonacci(int n) {  
    if (n == 0 || n == 1)  
        return 1;  
    return fibonacci(n-2) + fibonacci(n-1);  
}
```

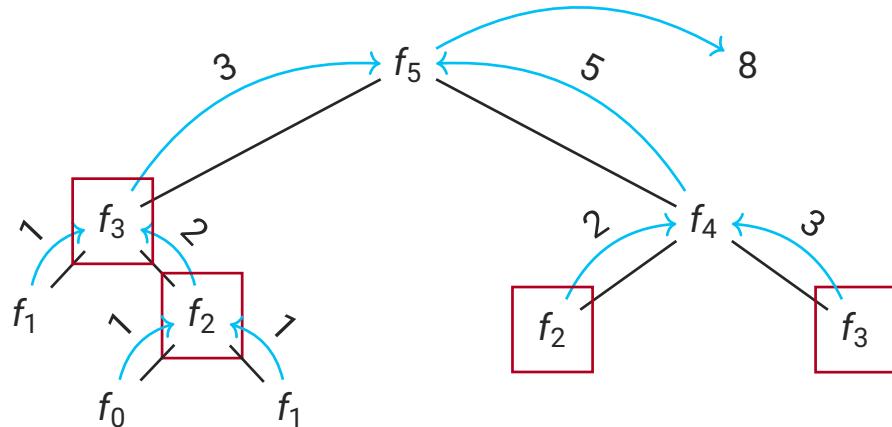
Las invocaciones que se hacen usando un bottom-up recursivo...



Podemos dividir el problema en subproblemas y resolver cada uno de ellos (top-down recursivo a.k.a. Memoization)

```
public int fibonacci(int n) {  
    return fibonacci(n, new int [n+1]);  
}  
public int fibonacci(int n, int [] memo){  
    if (n==0 || n ==1) return n;  
    if (memo[n] == 0 ){  
        memo[n] = fibonacci(n-1, memo) + fibonacci(n-2, memo);  
    }  
    return memo[n];  
}
```

Las invocaciones que se hacen usando memoization...



Podemos guardar en un caché los datos e ir iterando sobre los valores sin llamadas recursivas (Programación dinámica)

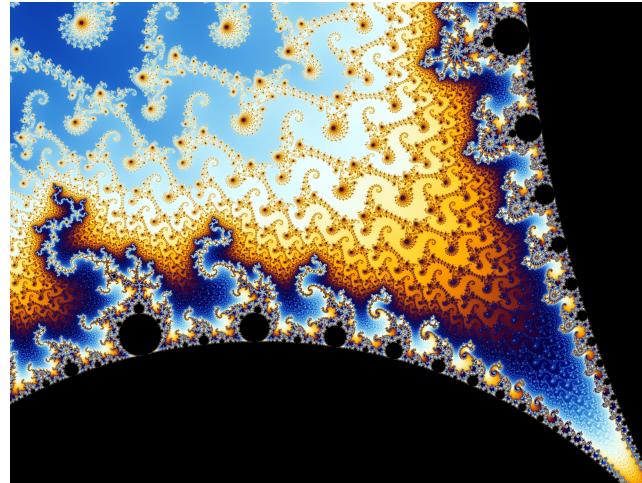
```
public int fibonacci(int n){  
    //Es iterativo pero el problema es recursivo  
    if (n==0 || n ==1) return n;  
    //Nota: No se ocupa realmente el arreglo  
    int [] memo = new int [n + 1];  
    memo[0] = 0;  
    memo[1] = 1;  
    for(int index = 2; index <= n; index++){  
        memo[index] = memo[index -1] + memo[index-2];  
    }  
    return memo[n];  
}
```

¿Cuáles son ejemplos de recursividad?

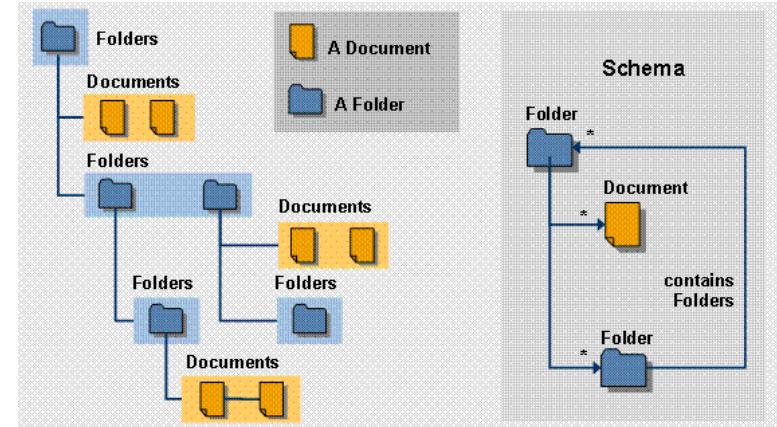
Las Matroyshkas pueden tener dentro de ellas una Matroyshka



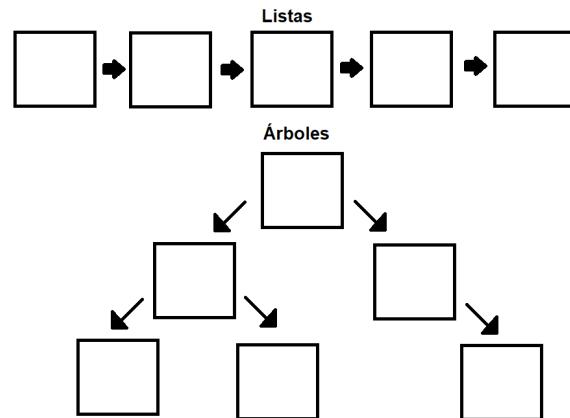
Un fractal es un patrón que se repite sin fin



Una carpeta de una computadora tiene archivos y otras carpetas



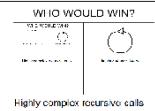
En estructuras de datos recursivas (listas y árboles)



¿Cuándo se debería utilizar la recursividad?

# WHO WOULD WIN?

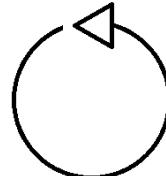
## WHO WOULD WIN?



Highly complex recursive calls

Highly complex recursive calls

Simple and basic loops



Simple and basic loops

## Referencias I

S. Hamer, "Recursividad," Material del curso CI-0202 Universidad de Costa Rica de Sivana Hamer, 2021.

A. B. Downey and C. Mayfield, *Think Java: How to Think Like a Computer Scientist*, second edition ed., 2020.

D. J. Eck, *Introduction to Programming Using Java*, eighth edition ed. Geneva (NY): Hobart and William Smith Colleges, Department of mathematics and computer science, 2020.

G. L. McDowell, *Cracking the coding interview: 189 programming questions and solutions*. CareerCup, 2019.

[Image]. [Online]. Available: [https://help.sap.com/saphelp\\_scm70/helpdata/ru/5d/97384162316532e10000000a1550b0/content.htm?no\\_cache=true](https://help.sap.com/saphelp_scm70/helpdata/ru/5d/97384162316532e10000000a1550b0/content.htm?no_cache=true)

[Image]. [Online]. Available: <https://www.dookinternational.com/blog/wp-content/uploads/2018/07/a2.jpeg>

## Referencias II

W. Beyer. Mandel zoom 09 satellite head and shoulder. [Image]. [Online]. Available: [https://commons.wikimedia.org/wiki/File:Mandel\\_zoom\\_09\\_satellite\\_head\\_and\\_shoulder.jpg](https://commons.wikimedia.org/wiki/File:Mandel_zoom_09_satellite_head_and_shoulder.jpg)

Recursion. [Image]. [Online]. Available: <https://algodaily.com/categories/recursion>

E. Grimson. 6. recursion and dictionaries. [Video]. [Online]. Available: <https://www.youtube.com/watch?v=WPSejyX1-4s>

A. Sweigart. Recursion for beginners: A beginner's guide to recursion. [Video]. [Online]. Available: <https://www.youtube.com/watch?v=AfBqVVKg4GE>

ImpulseTheFox. Recursion vs. loops. [Image]. [Online]. Available: [https://www.reddit.com/r/ProgrammerHumor/comments/78ubt2/recursion\\_vs\\_loops/](https://www.reddit.com/r/ProgrammerHumor/comments/78ubt2/recursion_vs_loops/)