Recursión

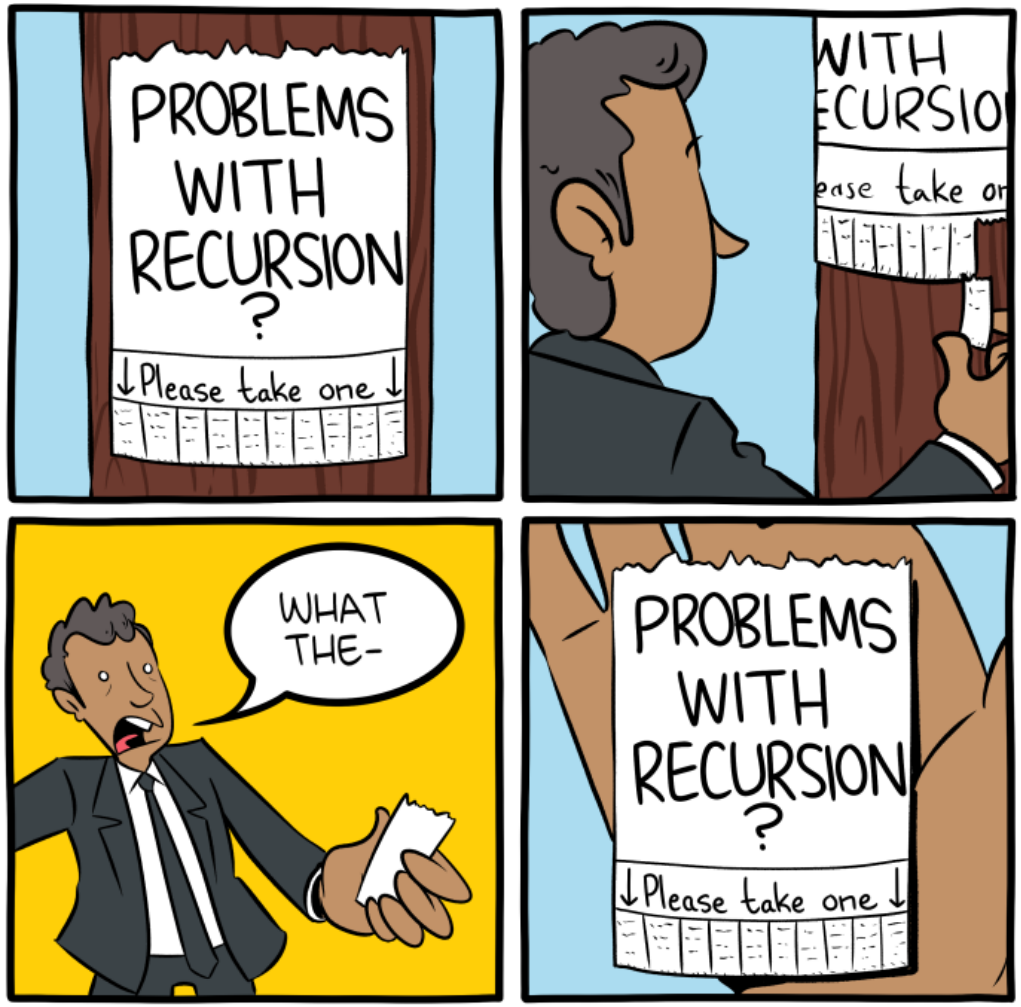
Programación de estructuras de datos y algoritmos fundamentales

Francisco J. Navarro B. BEng, MSc, PhD
fj.navarro.barron@tec.mx

Tecnológico de Monterrey

08-2022

Contenido


PROBLEMS WITH RECURSION ?
↓Please take one↓
WITH
ECURSIO
ease take or
WHAT THE-
PROBLEMS WITH RECURSION ?
↓Please take one↓
smbc-comics.com

Recursión

- Las funciones recursivas son aquellas definidas en términos de sí misma.
- Se llaman a sí mismas como parte del proceso para resolver un problema.
- Comúnmente cuando necesitamos computar un valor a través de **repetición** lo realizamos mediante **loops**. Por ejemplo:

```
unsigned int factorial(unsigned int n){
    int f = 1;
    for(int i = 2; i <= n; i++) f*=i;
    return f;
}
```

Recursión

- Una función recursiva logra la repetición al llamarse a sí misma reiteradas veces, calculando un valor en ese proceso.

$$n! = n \times (n-1) \times (n-2) \times \ldots \times 1$$
$$\text{para } n > 0$$

```
unsigned int factorial(unsigned int n){
    if(n <= 1) return 1;
    return n * factorial(n - 1);
}
```

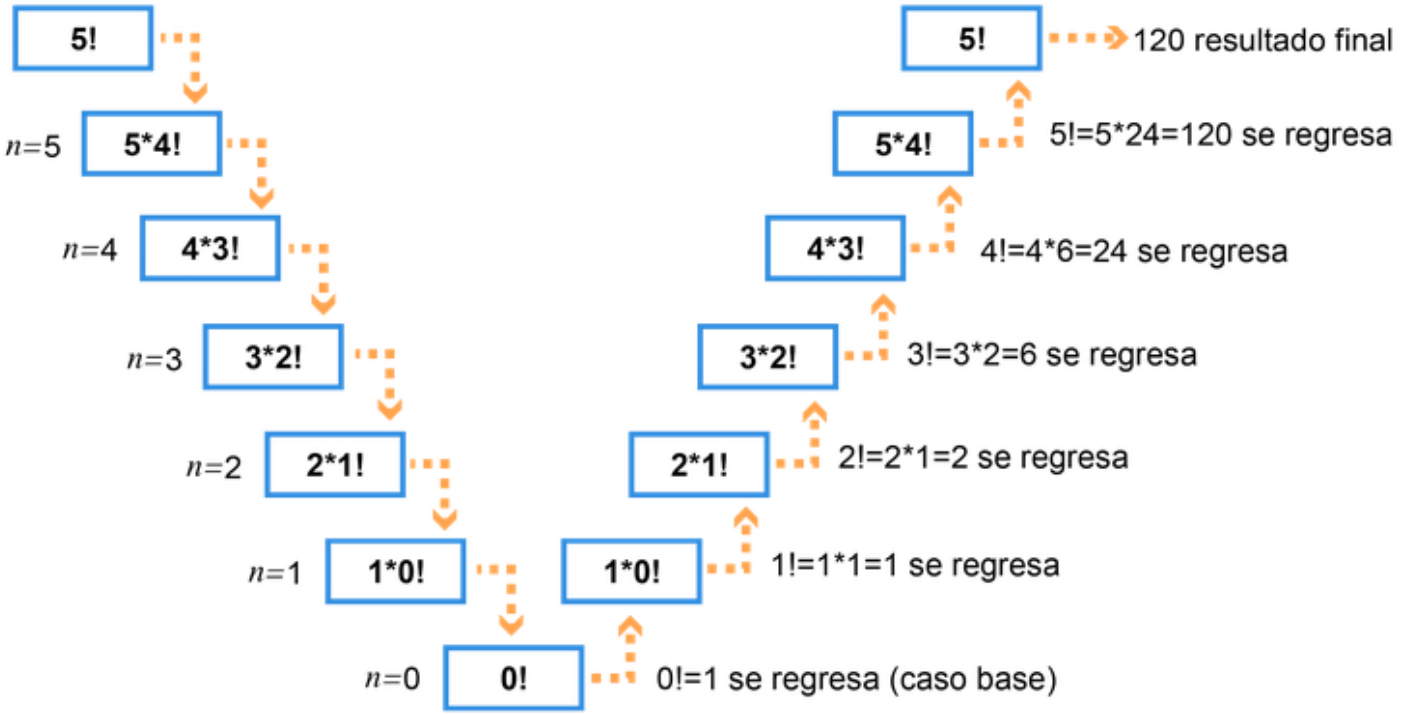
Partes de una función recursiva

$$n! = \begin{cases} 1, & n = 0 \\ n \times (n-1)!,) & n \geq 1 \end{cases} \quad (1)$$

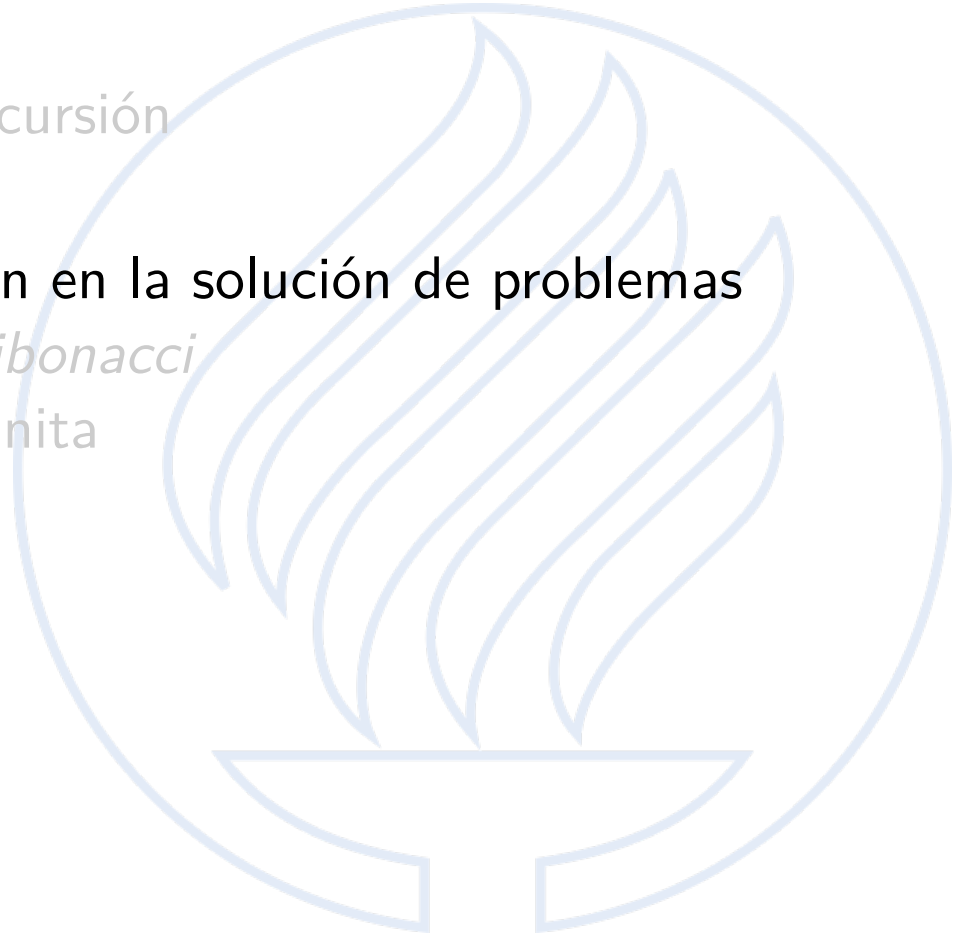
```
unsigned int factorial(unsigned int n){
    if(n <= 1) return 1;
    return n * factorial(n - 1);
}
```

- **Caso base**: La función no se llama a sí misma. Usualmente retorna un resultado predefinido.
- **Caso recursivo**: La función se invoca a sí misma devolviendo un resultado, y debe ser una llamada que lleve hacia el caso base.

Consideraciones para emplear la recursión



- Llamadas a función quedan en espera en el *stack* (pila).
- Variables locales de cada llamada recursiva son independientes.



La serie de *Fibonacci*

La serie de *Fibonacci* inicia de la siguiente manera:

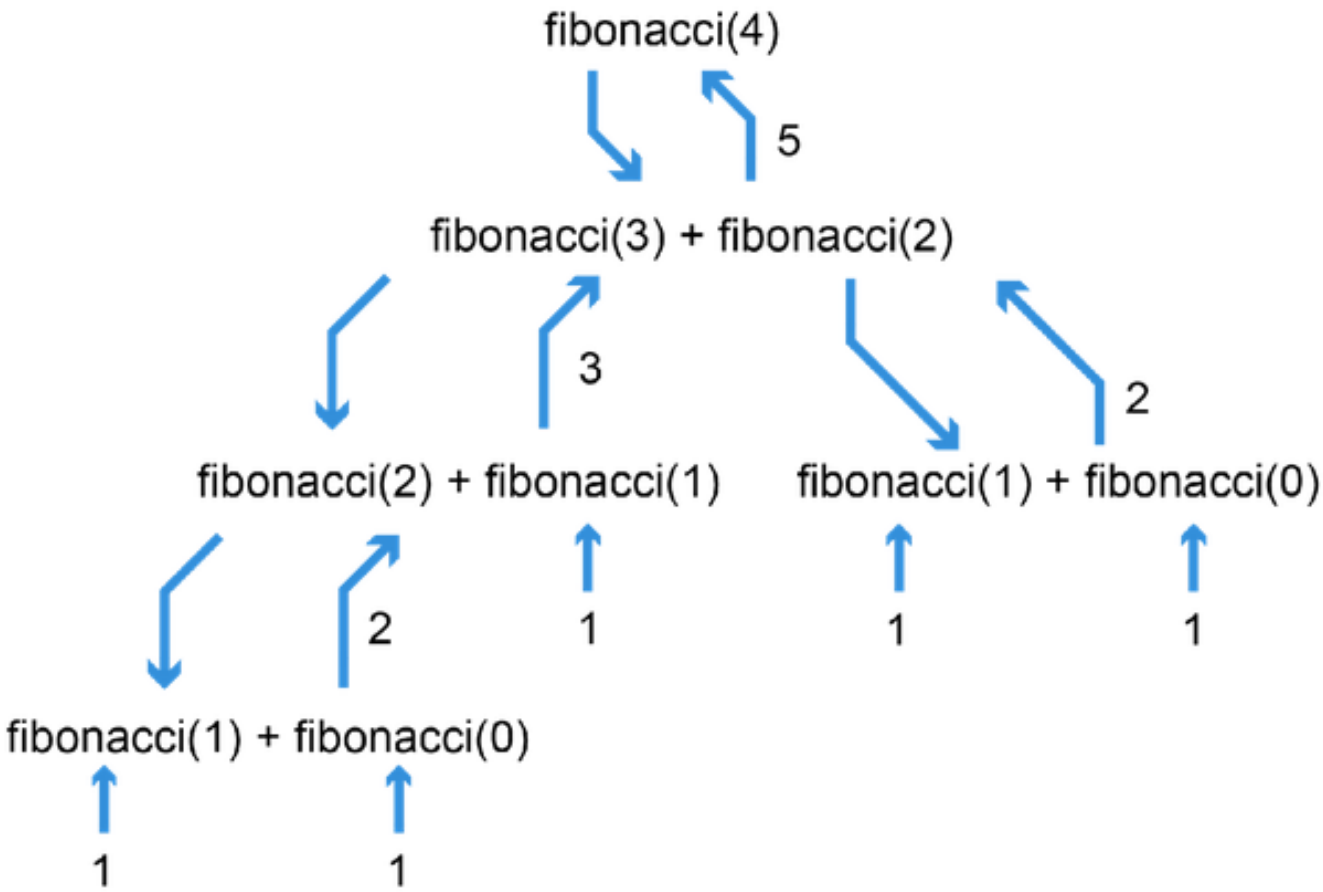
$$1, 1, 2, 3, 5, 8, 13, 21, 34, 55, \ldots \tag{2}$$

- **Caso base**: Los 2 (primeros) números.
- **Caso recursivo**: Siguientes n-ésimos términos

Codificación de *Fibonacci*

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

Recursión en *Fibonacci*



- Cálculo del 5to elemento: 9 llamadas a función.
- Cálculo del 6to elemento: x llamadas a función?

Recursión infinita

```
int bad(int n){
    if(n==0)
        return 0;
    else
        return bad(n / 3 + 1) + n - 1;
}
```

- Sin caso base ó aproximación a el → *stack overflow*
- Lógica circular 😵

Actividad 1 - Imprimiendo números mediante recursión

Diseña una función que reciba un entero positivo n y mediante recursión tome un solo dígito y lo despliegue hasta terminar de imprimir n.

1. Define el **caso base** mediante una función *printDigit(n)* para $0 \leq n < 10$
2. Usa el **caso recursivo** para $n \geq 10$

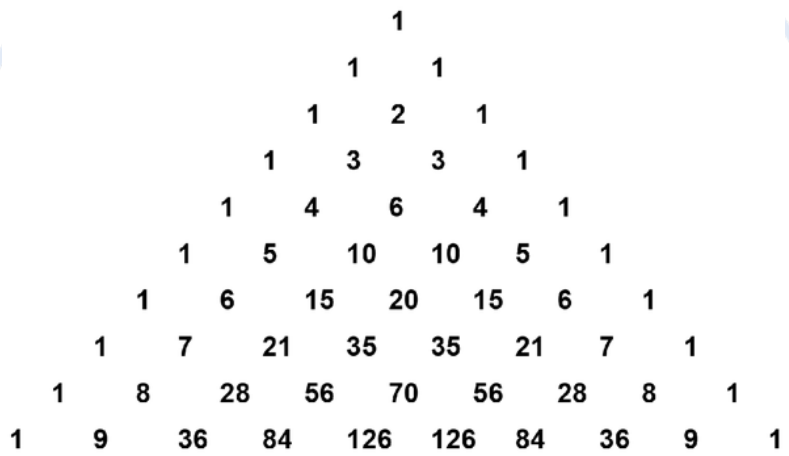
Actividad 2 - Cálculo de potencia

Diseña una función recursiva ***long potencia***(*int base, int n*) que reciba como parámetro dos enteros positivos. La función debe calcular un número (*base*) elevado a una potencia *n*.

$$f(x) = x^n$$

Actividad 3 - Triángulo de Pascal

Los coeficientes de un binomio elevado a una potencia son representados por el triángulo de *Pascal* en honor al matemático *Blaise Pascal*. Las primeras 10 filas de este triángulo son:



- El 1, en la "punta"superior, representa el coeficiente al elevar $(a+b)^0$, con resultado igual a 1.
- La siguiente fila representa los coeficientes al elevar $(a+b)^1$, es decir $a+b$.
- La tercer fila son los coeficientes de elevar $(a+b)^2$, los valores 1,2,1, son los coeficientes del polinomio resultante $a^2+2ab+b^2$ y así sucesivamente.

Actividad 3 - Triángulo de Pascal

Diseña una función recursiva ***int trianguloPascal(int r, int c)*** que calcule el número del triángulo de Pascal que se ubica en la fila *r* y columna *c*.

- Por ejemplo, al llamar la función con los valores 7 y 2 la función debe retornar 21.
- Prueba tu función implementando otra función ***void triangularesHastaFila****(int n)* la cual recibe como parámetro el número de fila hasta la cual se quieren imprimir los números del triángulo de Pascal.


Thanks!