Estructuras de datos

Mergesort
Y algoritmos recursivos

a > 0

```
int funcion(int a){
    if(a==0){
        return 1;
    }
    else{
        return 1+funcion(a-1);
    }
}
```

condición de parada

llamado recursivo

funcion(5);

1 + funcion(4)

1 + 1 + funcion(3)

1 + 1 + 1 + funcion(2)

1 + 1 + 1 + 1 + funcion(1)

1 + 1 + 1 + 1 + 1 + funcion(0)

1 + 1 + 1 + 1 + 1 + 1

$n! = n \times (n-1) \times (n-2) \times \cdots \times 2 \times 1$

$5! = 5 \times \underbrace{4 \times 3 \times 2 \times 1}_{4!}$ $\quad 5! = 5 \times 4!$

$4! = 4 \times 3 \times 2 \times 1$ $\quad 4! = 4 \times 3!$

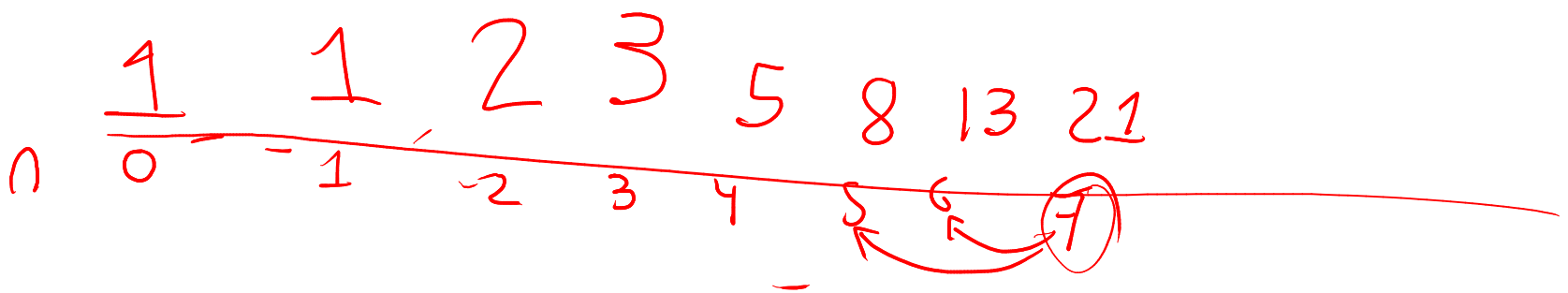
$fac(1) = 1$

$fac(n) = n \times fac(n-1)$

$fac(9) = 9 \times fac(8)$

$n \geq 1$

$$F_{fac}(n) = \begin{cases} n \times F_{fac}(n-1) & n > 1 \\ 1 & n = 1 \end{cases}$$



1) c función recursión

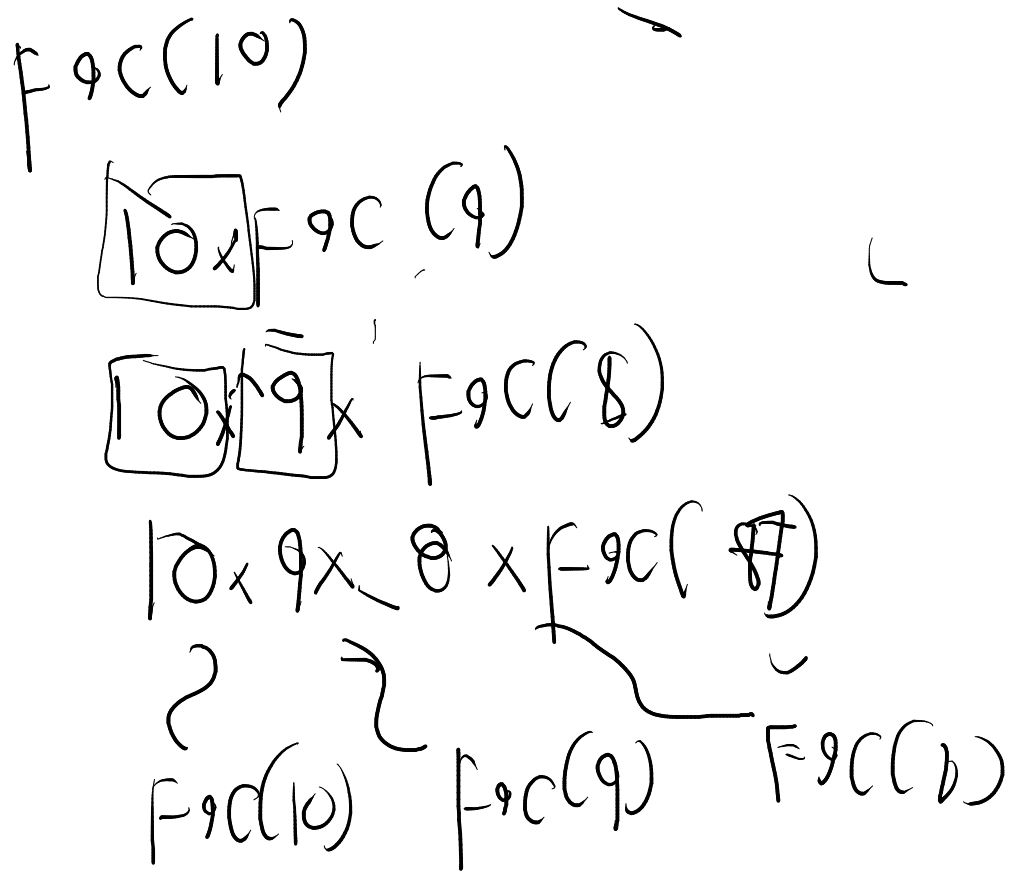
$$Fib(n) \begin{cases} Fib(n-1) + Fib(n-2) \\ 1 \qquad n == 0 \ \vee \ n == 1 \end{cases}$$

Genere una función recursiva que reciba un valor n
 entero mayor que 0,
y retorne la lista de factoriales desde 1 hasta ese n!.

Por ejemplo listaFactoriales(5)

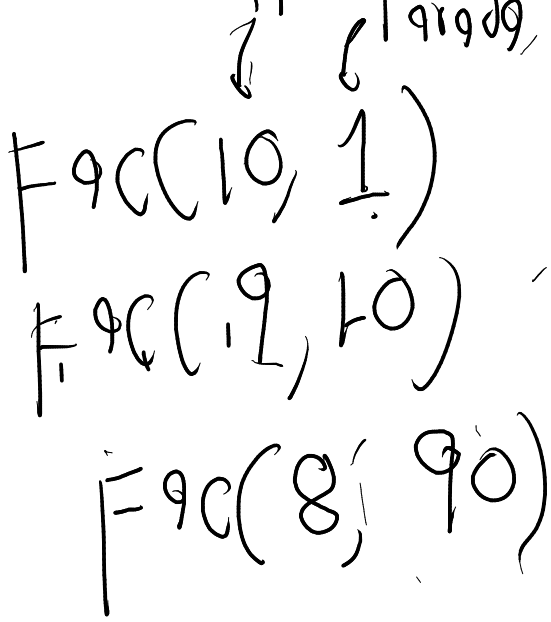
(1 2 6 24 120)

```
def factorial(n):
    if n==1:
        return 1
    else:
        return n*factorial(n-1)
```



Recursion cola

```
def factorial(n,res):
    if n==1:
        return res
    else:
        return factorial(n-1,res*n)
```

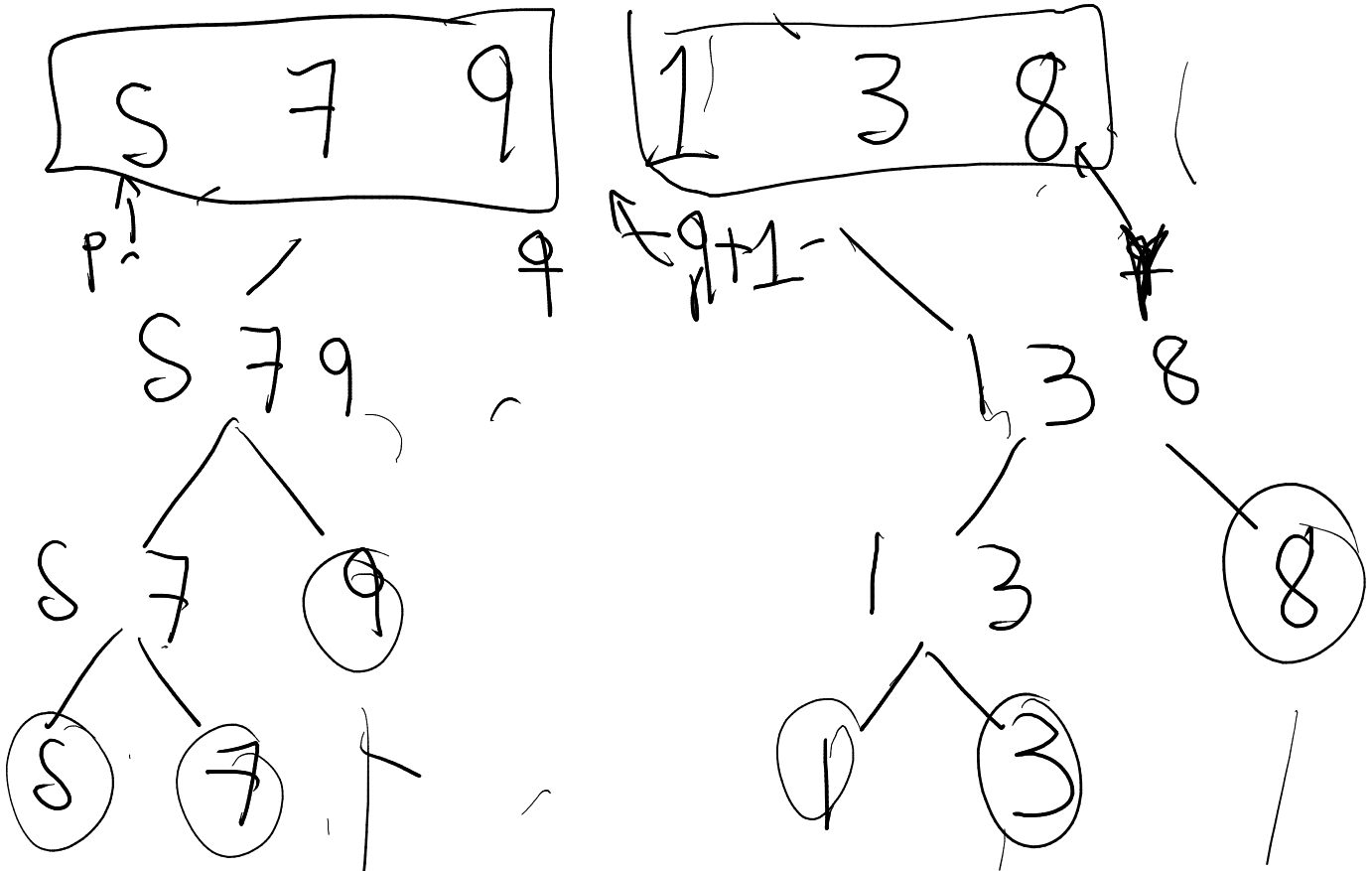


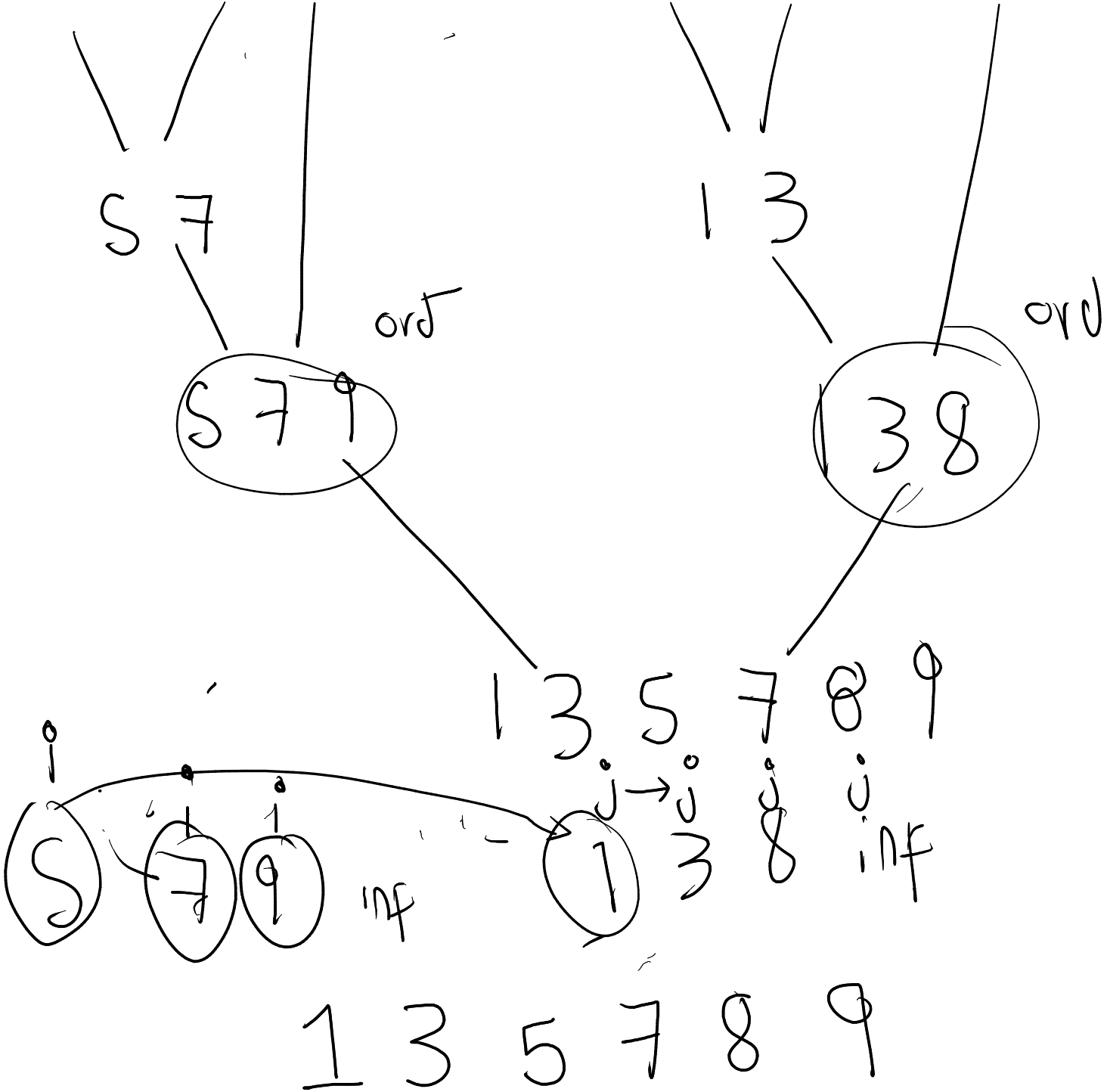
Fac(7, 10x9x8)

Fac(6, 10x9x8x7)

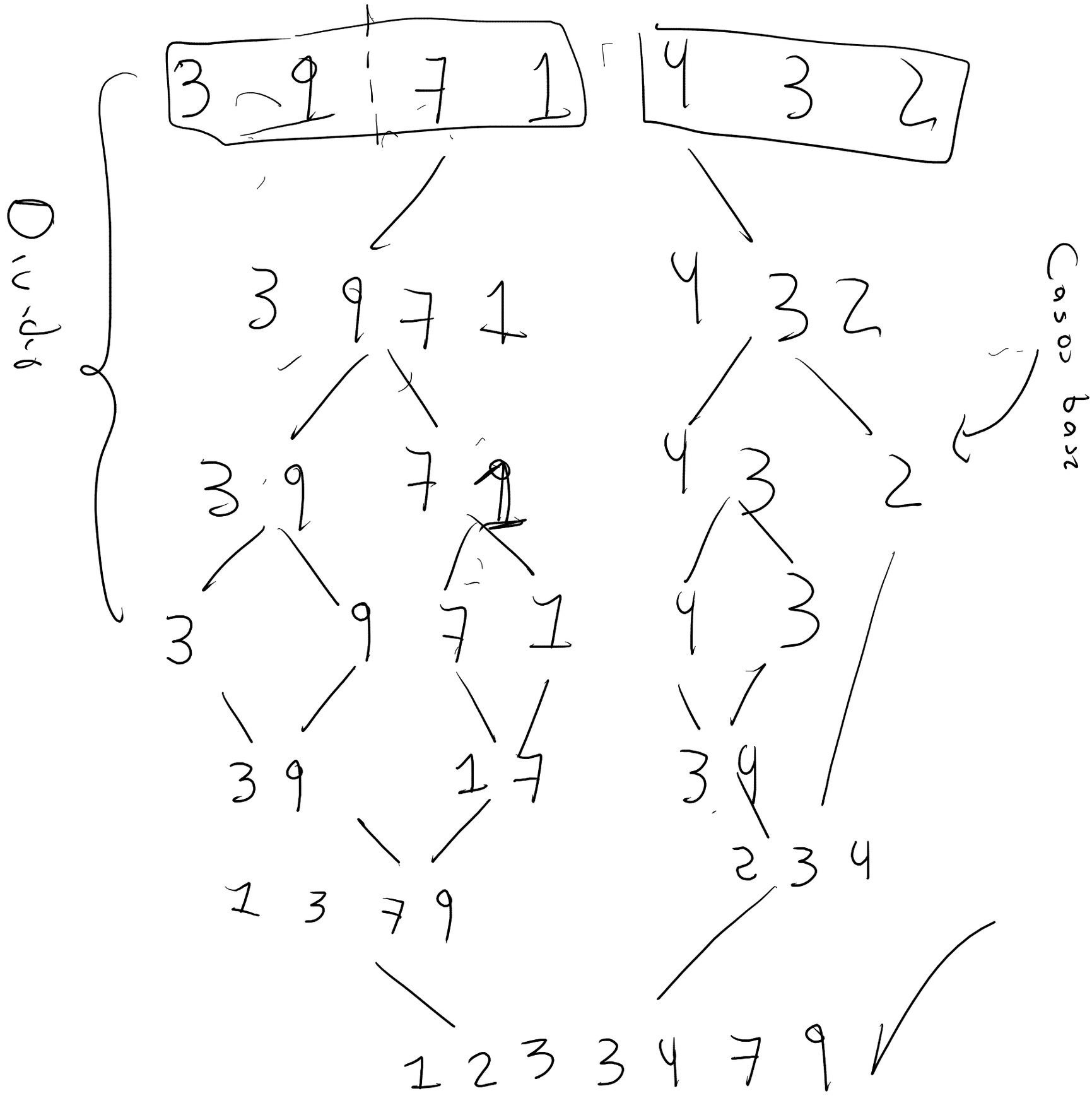
Mergesort: Ordenamiento

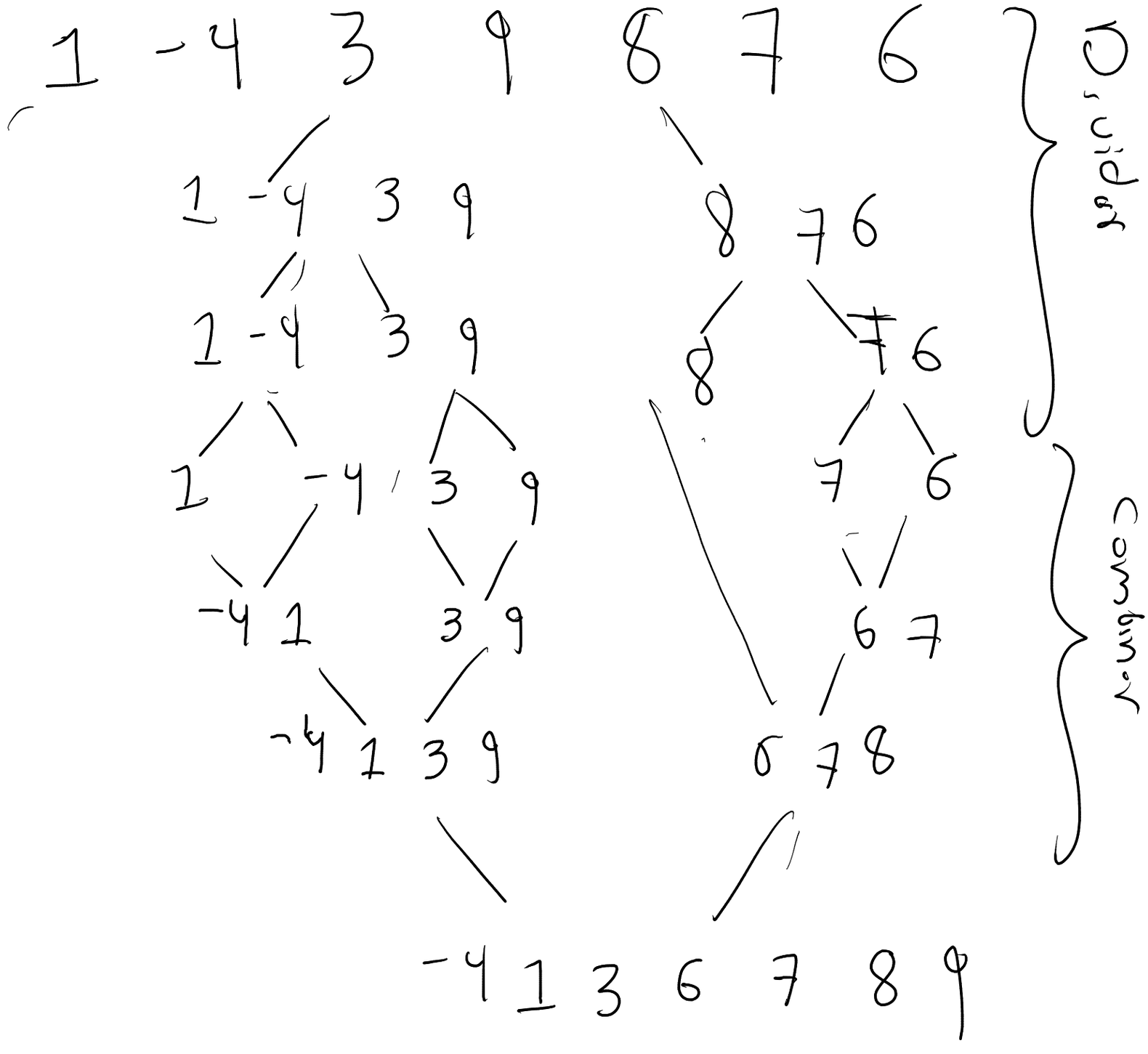
- Divida la lista a la mitad
- Genera dos arreglos de tamaño n/2
- Divida hasta que llegue a arreglos de tamaño 1, que son arreglos ordenados (trivial)
- Combine los arreglos (ganancia es que ambos estan ordenados)




5 7
1 3
ord
ord
5 7 9
1 3 8
1 3 5 7 8 9
i
j → i
j
i
5
7
9
inf
1
3
8
inf
1 3 5 7 8 9

1 3 7 9 12 15 20


3 9 7 1
4 3 2
Caso base
1 2 3 3 4 7 9


1 -4 3 9 8 7 6
1 -4 3 9
8 7 6
1 -4 3 9
8 7 6
1 -4 3 9
7 6
-4 1 3 9
6 7
-4 1 3 9
6 7 8
-4 1 3 6 7 8 9
Divide
Conquer

Análisis de algoritmos recursivos

Ejemplo, pensemos en este algoritmo para calcular la serie de Fibunnaci para un número (n) dado

Recuerda:

$$f(n)=f(n-1)+f(n-2), f(0)=1, f(1)=1$$

fibunnaci(n)

Si n = 0 retorne 1

Sino si n = 1 retorne 2

Sino fibunnaci(n -1) + fibunnaci(n - 2)

Análisis de algoritmos recursivos

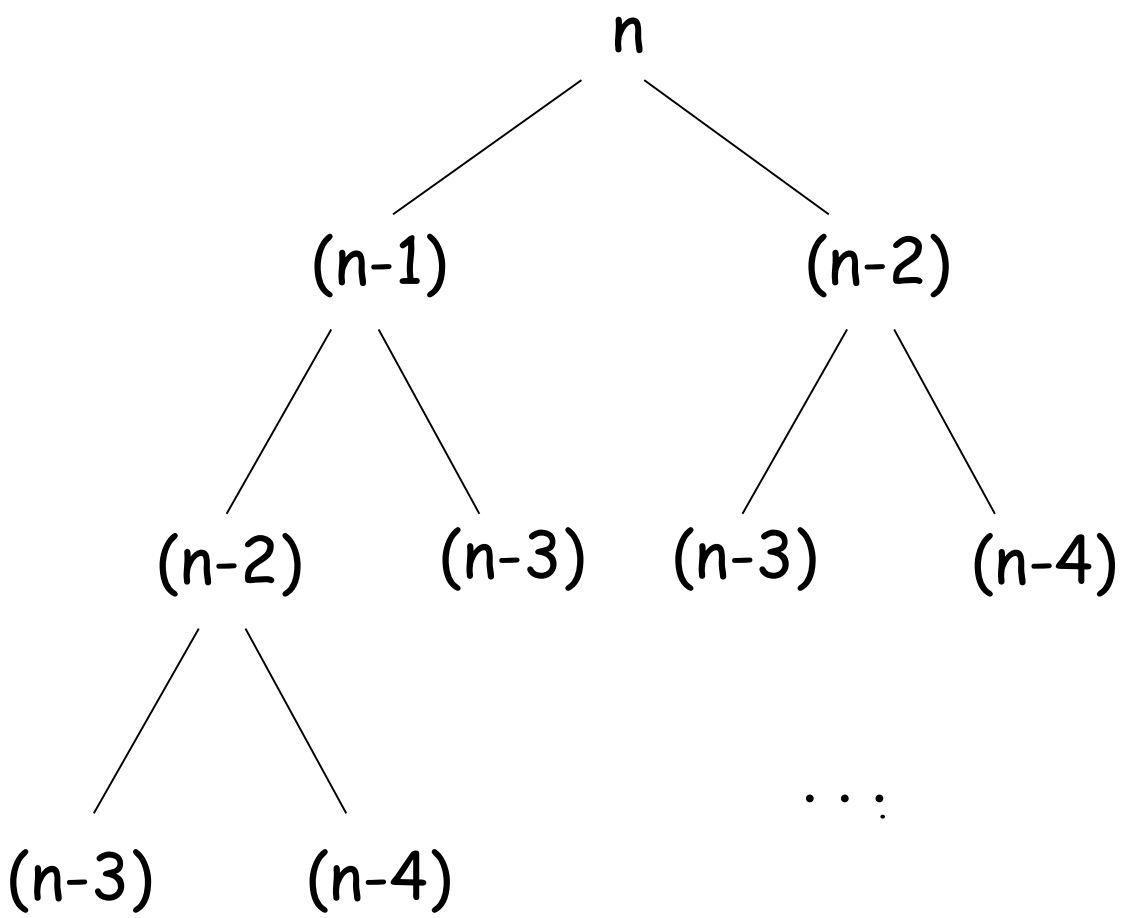
Si

Recuerda:

$$f(n)=f(n-1)+f(n-2), f(0)=1, f(1)=1$$

```
fibunnaci(n)
    Si n = 0 retorne 1
    Sino si n = 1 retorne 2
        Sino fibunnaci(n -1) + fibunnaci(n - 2)
```

Análisis de algoritmos recursivos



package punto1;

tk
glade
pyqt