Repaso de Recursión

Repaso de Recursión

- Se dice que un objeto es recursivo cuando forma parte de sí mismo, es decir puede definirse en términos de sí mismo.

- En programación, la recursividad es la propiedad que tienen los Algoritmos de llamarse a sí mismos para resolver un problema.

Repaso de Recursión

- Ejemplos de definiciones recursivas:
  - Factorial de un número

    $0! = 1$

    Si $n > 0$, $n! = n * (n-1)!$

  - Potencia de un número

    $x^0 = 1$

    Si $y > 0$, $x^y = x * x^{y-1}$

  - Estructuras de datos

    Árboles

Repaso de Recursión

- Ejemplos de soluciones recursivas:
  - Buscar un elemento en un arreglo
  - Ordenar un arreglo de elementos
  - Recorrer un árbol

Repaso de Recursión

- Soluciones recursivas:
  - División sucesiva del problema original en problemas más pequeños del mismo tipo
  - Se van resolviendo estos problemas más sencillos
  - Con las soluciones de éstos se construyen las soluciones de los problemas más complejos

Repaso de Recursión

- Ejemplo:

  Programar un algoritmo recursivo que permita invertir un número

```
int invertir (int n)
 {
        if (n < 10)        //caso base
          System.out.print(n);
        else
          System.out.print(n mod 10); // el resto de la división entera
          invertir (n div 10);
 }
```

Repaso de Recursión

- Ejecución:

Entrada: 123

```
int invertir (int n)
{
  if (n < 10)        //caso base
   System.out.print(n);
  else
   System.out.print(n mod 10); // el resto de la división entera
  invertir (n div 10);
}
```

Salida:

Repaso de Recursión

- Ejecución:

Entrada: 123

```
int invertir (int n)
{
  if (n < 10)        //caso base
   System.out.print(n);
  else
   System.out.print(n mod 10); // el resto de la división entera
  invertir (n div 10);
}
```

invertir (123)
n = 123
print (3)
invertir (12)

Salida: 3

Repaso de Recursión

- Ejecución:

Entrada: 123

```
int invertir (int n)
{
  if (n < 10)        //caso base
   System.out.print(n);
  else
   System.out.print(n mod 10); // el resto de la división entera
  invertir (n div 10);
}
```

invertir (123)
n = 123
print (3)
invertir (12)

invertir (12)
n = 12
print (2)
invertir (1)

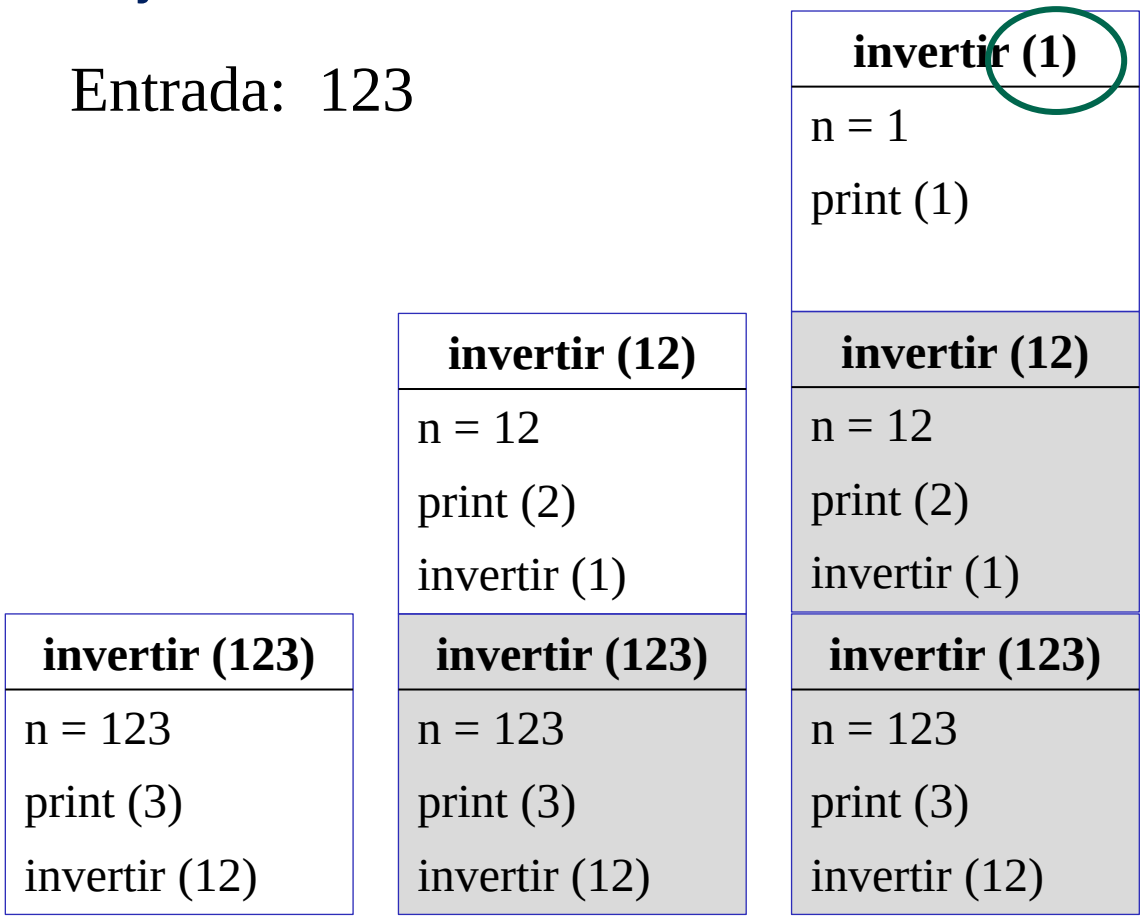
invertir (123)
n = 123
print (3)
invertir (12)

Salida: 3 2

Repaso de Recursión

- Ejecución:

Entrada: 123



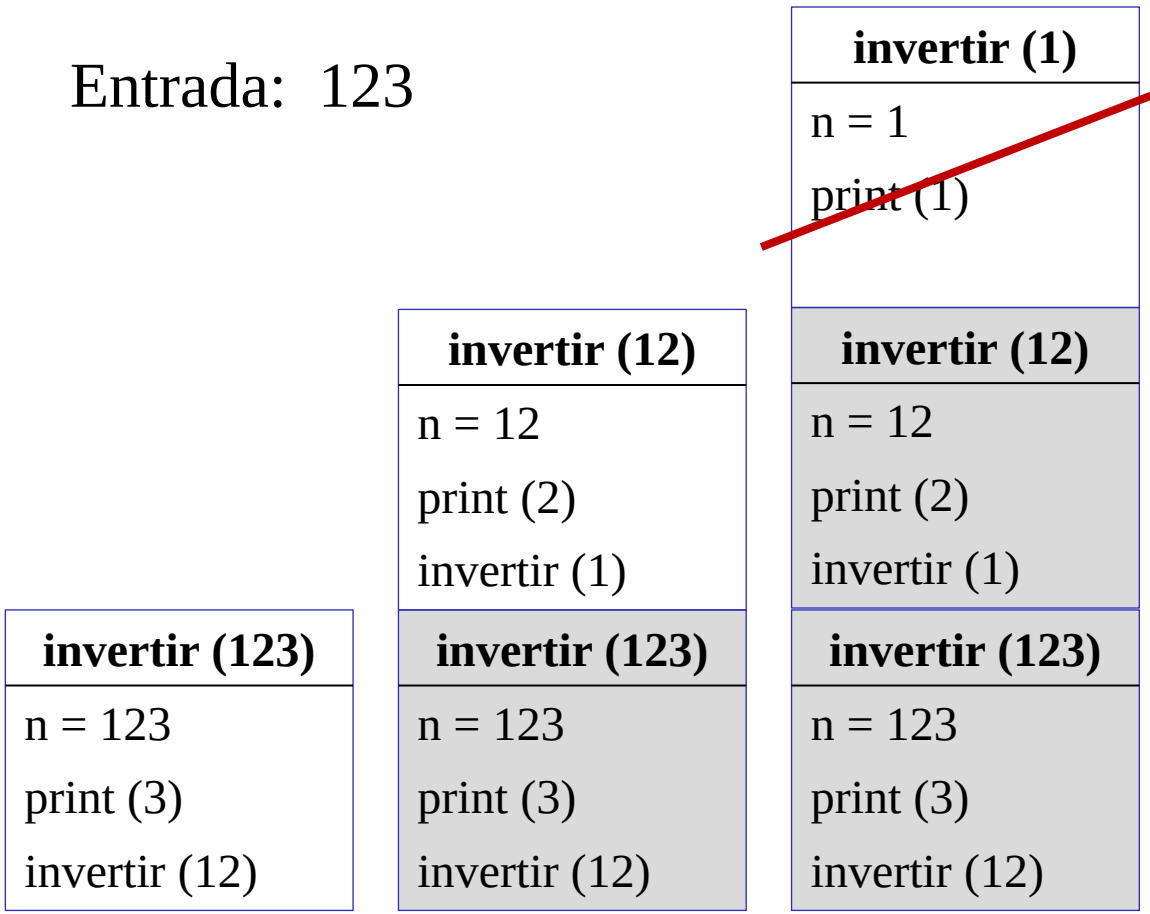

```
int invertir (int n)
{
  if (n < 10)          //caso base
   System.out.print(n);
  else
   System.out.print(n mod 10); // el resto de la división entera
  invertir (n div 10);
}
```

Salida: 3 2 1

Repaso de Recursión

- Ejecución:

Entrada: 123




```
int invertir (int n)
{
  if (n < 10)        //caso base
   System.out.print(n);
  else
   System.out.print(n mod 10); // el resto de la división entera
  invertir (n div 10);
}
```

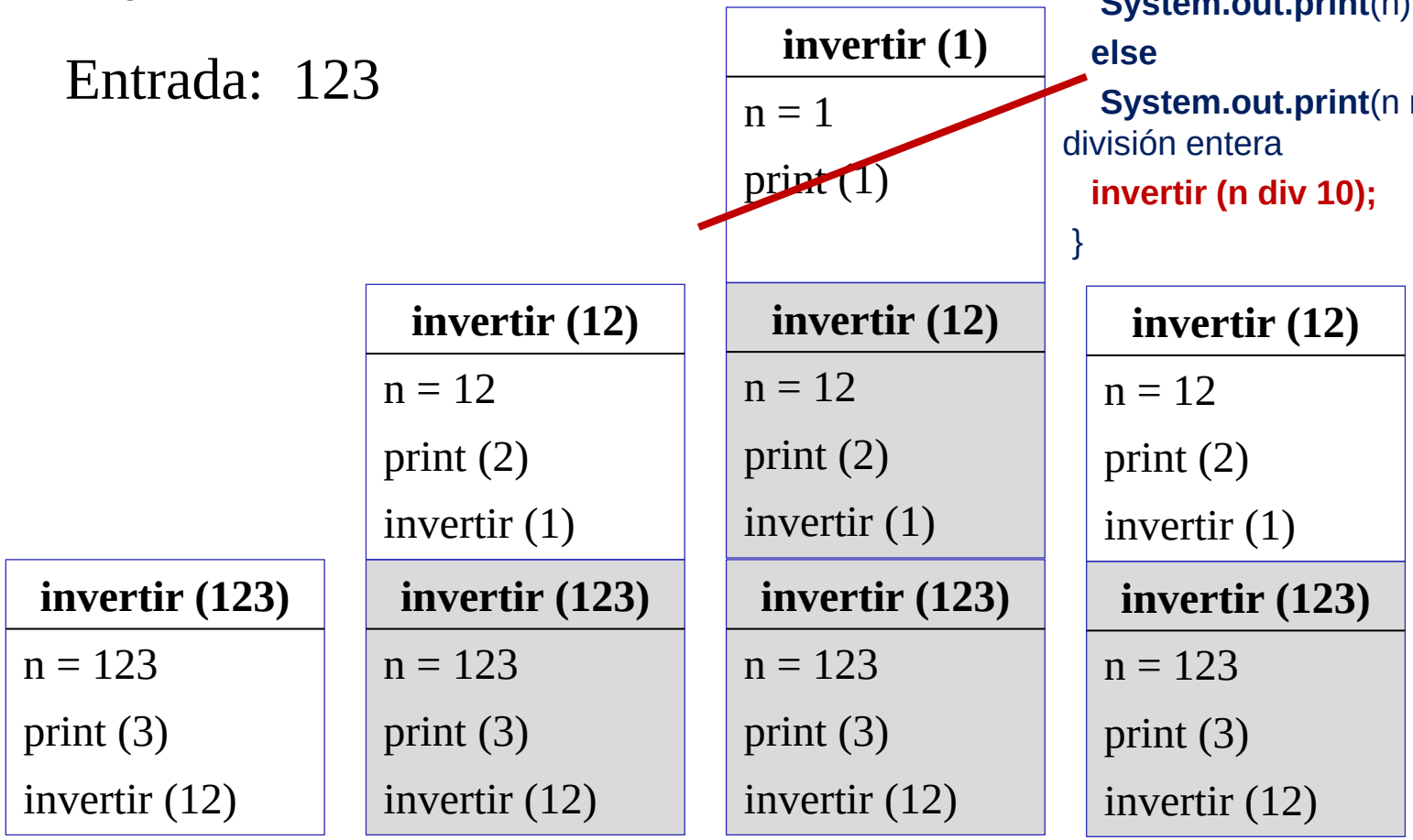
Salida: 3 2 1

Repaso de Recursión

- Ejecución:

Entrada: 123

```
int invertir (int n)
{
  if (n < 10)        //caso base
   System.out.print(n);
  else
   System.out.print(n mod 10); // el resto de la división entera
  invertir (n div 10);
}
```



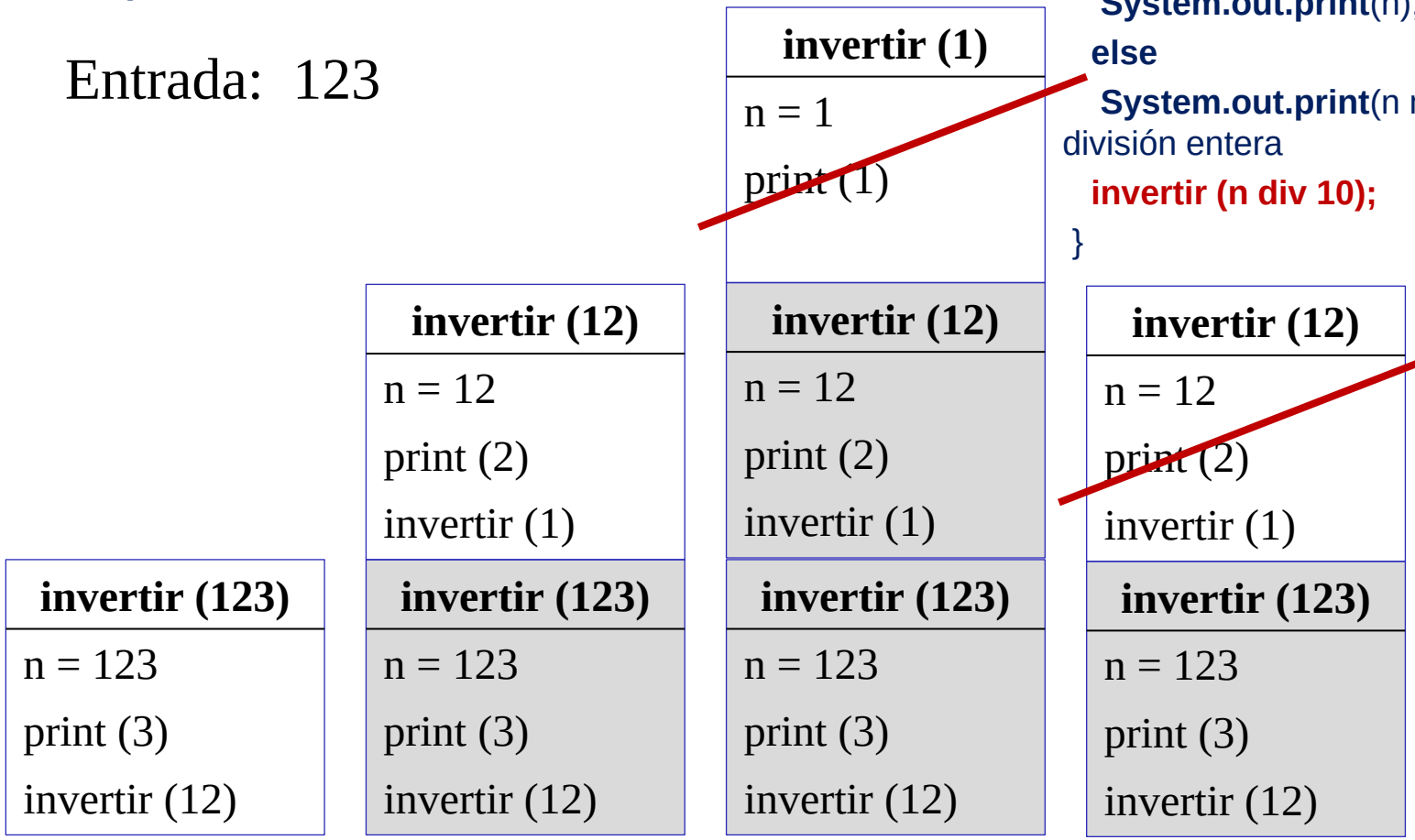
Salida: 3 2 1

Repaso de Recursión

- Ejecución:

Entrada: 123

```
int invertir (int n)
{
  if (n < 10)        //caso base
   System.out.print(n);
  else
   System.out.print(n mod 10); // el resto de la división entera
  invertir (n div 10);
}
```



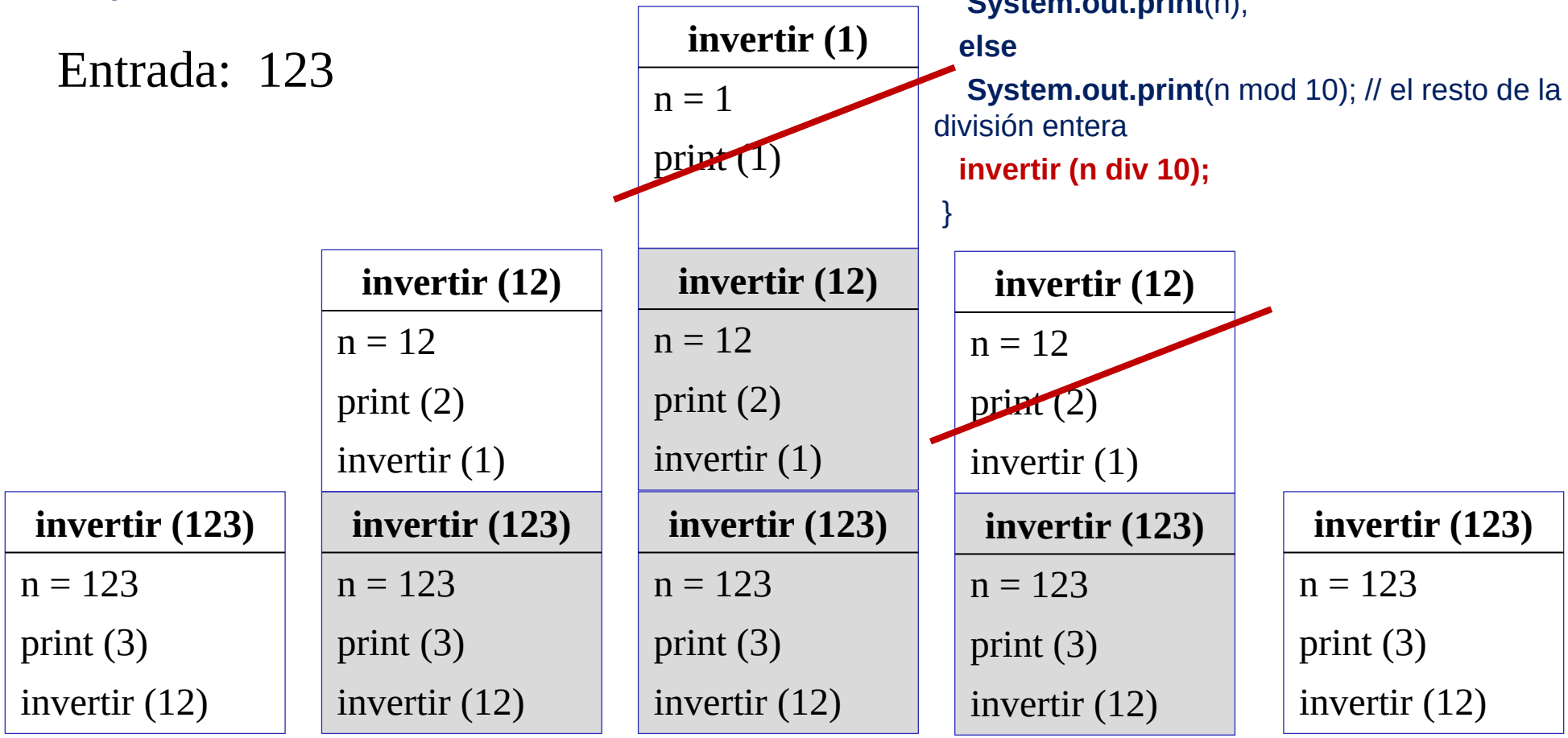
Salida: 3 2 1

Repaso de Recursión

- Ejecución:

Entrada: 123

```
int invertir (int n)
{
  if (n < 10)        //caso base
   System.out.print(n);
  else
   System.out.print(n mod 10); // el resto de la división entera
  invertir (n div 10);
}
```



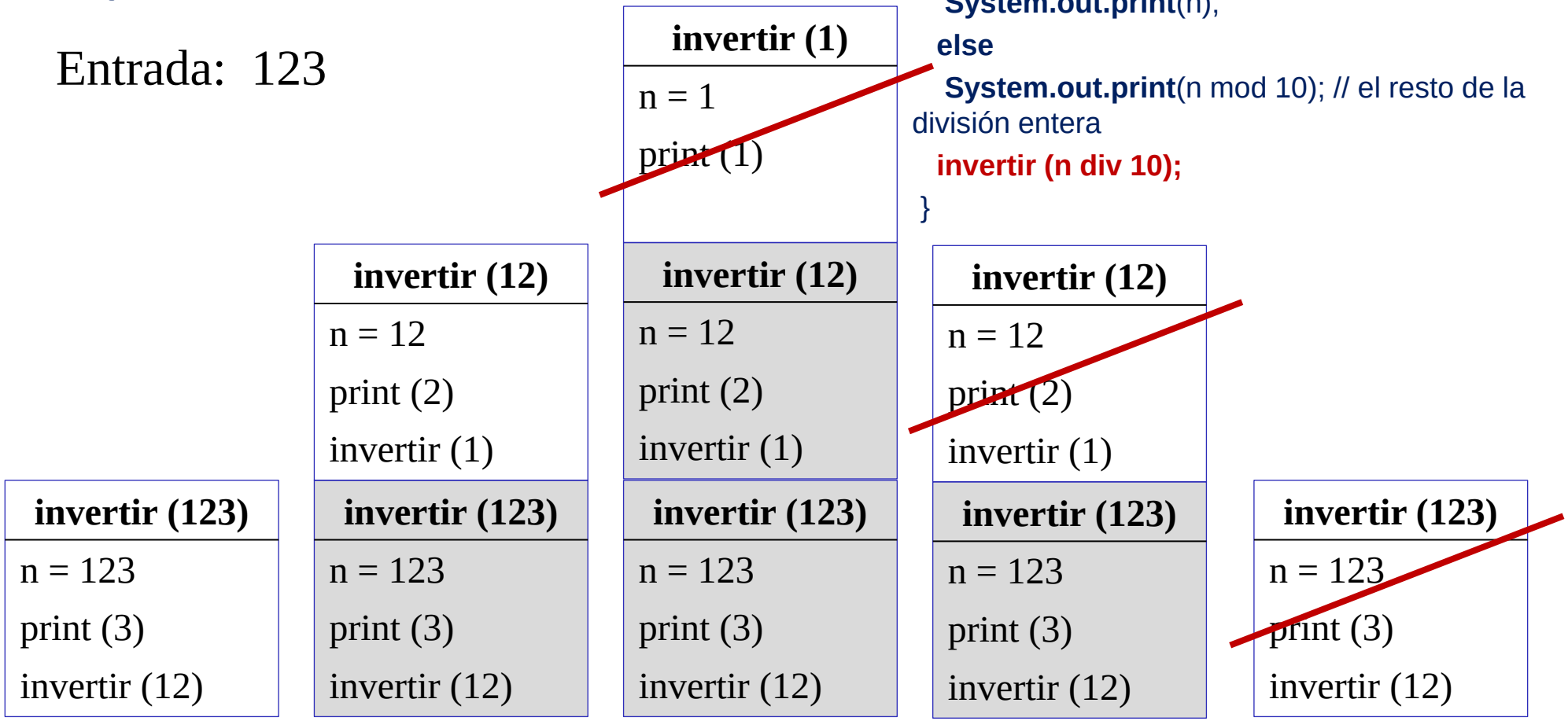
Salida: 3 2 1

Repaso de Recursión

- Ejecución:

Entrada: 123

```
int invertir (int n)
{
  if (n < 10)        //caso base
   System.out.print(n);
  else
   System.out.print(n mod 10); // el resto de la división entera
  invertir (n div 10);
}
```



Salida: 3 2 1

Repaso de Recursión

- Ejemplo 2: Algoritmo de ordenación ***MergeSort***

La estrategia del algoritmo consiste en dividir el vector en 2 partes (sub-vectores), ordenarlos y luego hacer un Merge de estos sub-vectores ya ordenados. Cada uno de esos sub-vectores se ordenan aplicando la misma estrategia, hasta tanto el vector contenga sólo un dato y en ese caso se lo devuelve (el sub-vector está ordenado).

Características ***recursivas*** del algoritmo

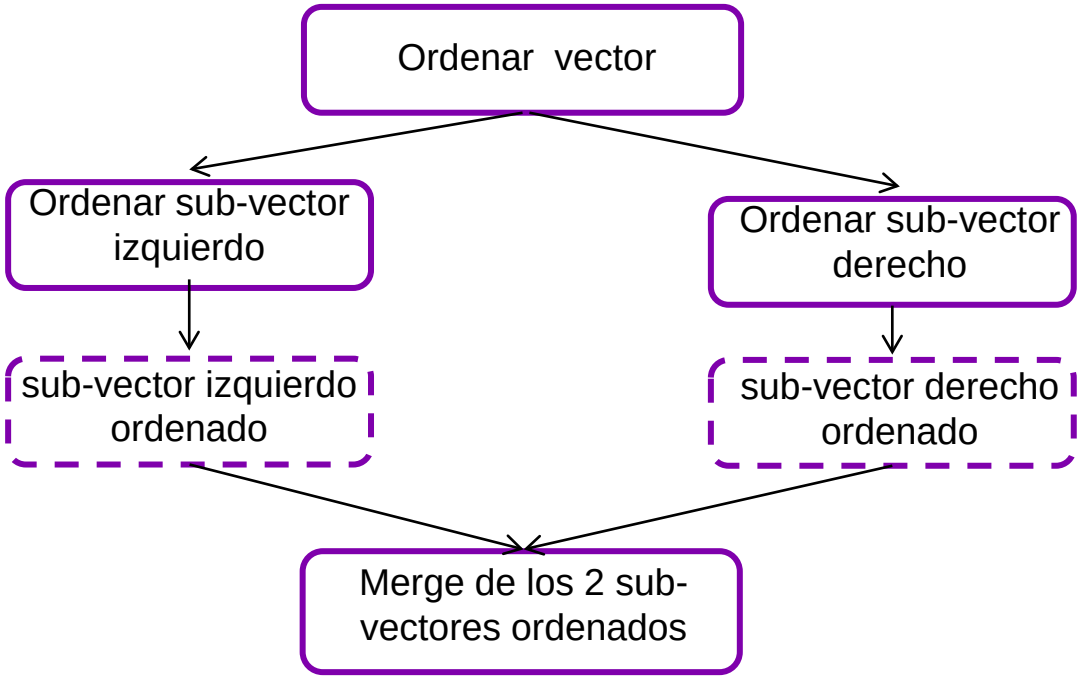
- ✓ Se resuelven 2 sub-problemas más pequeños
- ✓ Se combinan los resultados de cada solución
- ✓ Se cuenta con un caso base.

Repaso de Recursión

- Ejemplo 2: Algoritmo de ordenación ***MergeSort***

La estrategia del algoritmo consiste en dividir el vector en 2 partes (sub-vectores), ordenarlos y luego hacer un Merge de estos sub-vectores ya ordenados. Cada uno de esos sub-vectores se ordenan aplicando la misma estrategia, hasta tanto el vector contenga sólo un dato y en ese caso se lo devuelve (el sub-vector está ordenado).

Gráficamente:

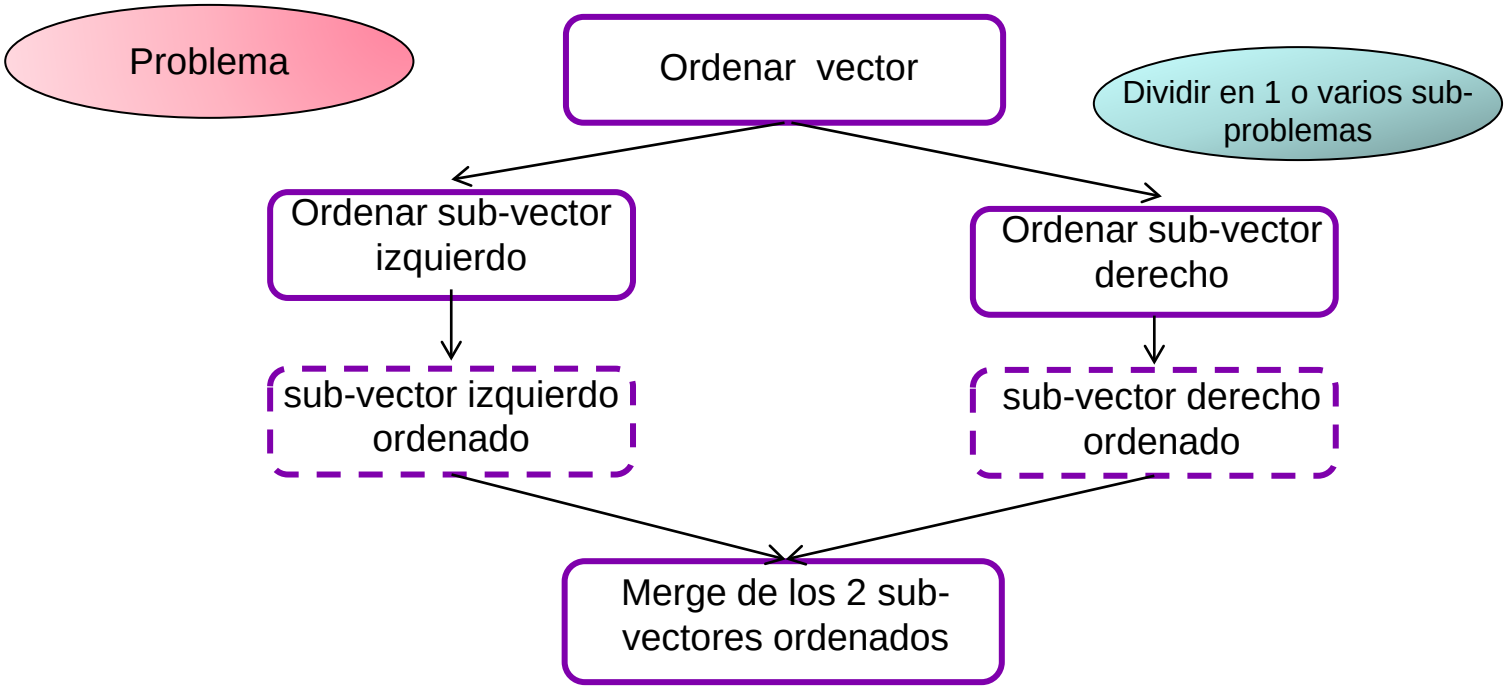


Repaso de Recursión

- Ejemplo 2: Algoritmo de ordenación ***MergeSort***

La estrategia del algoritmo consiste en dividir el vector en 2 partes (sub-vectores), ordenarlos y luego hacer un Merge de estos sub-vectores ya ordenados. Cada uno de esos sub-vectores se ordenan aplicando la misma estrategia, hasta tanto el vector contenga sólo un dato y en ese caso se lo devuelve (el sub-vector está ordenado).

Gráficamente:

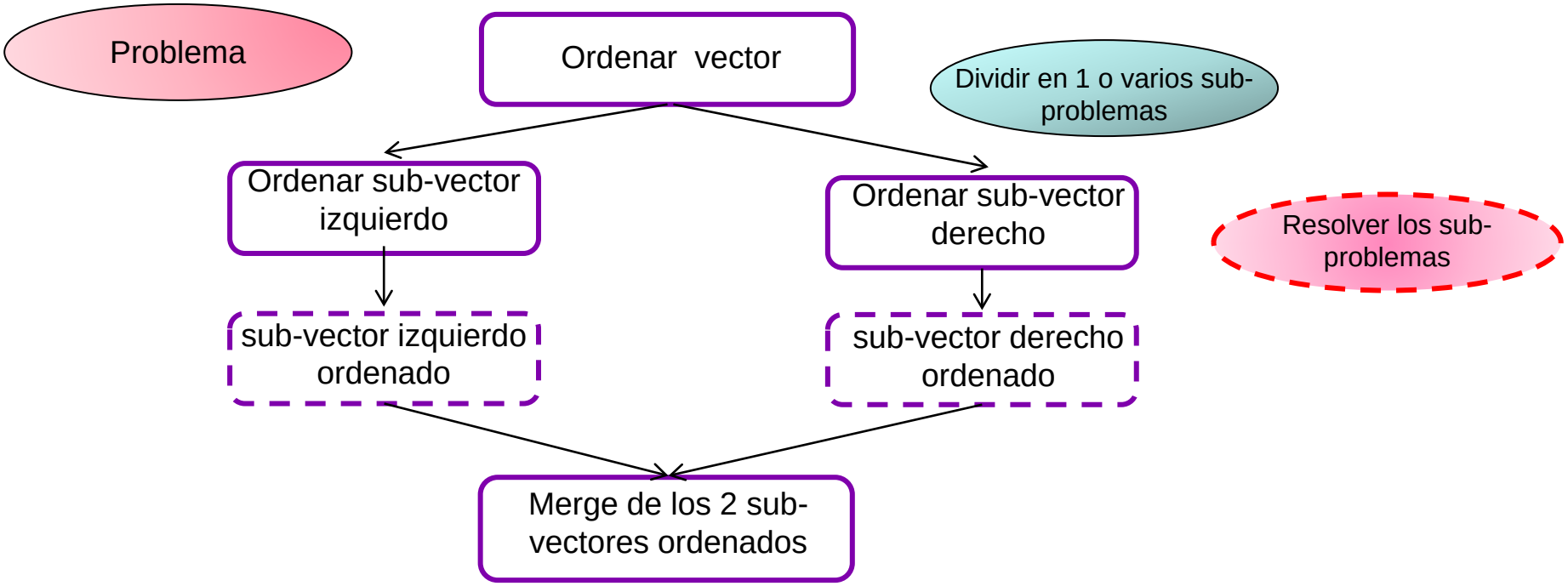


Repaso de Recursión

- Ejemplo 2: Algoritmo de ordenación ***MergeSort***

La estrategia del algoritmo consiste en dividir el vector en 2 partes (sub-vectores), ordenarlos y luego hacer un Merge de estos sub-vectores ya ordenados. Cada uno de esos sub-vectores se ordenan aplicando la misma estrategia, hasta tanto el vector contenga sólo un dato y en ese caso se lo devuelve (el sub-vector está ordenado).

Gráficamente:

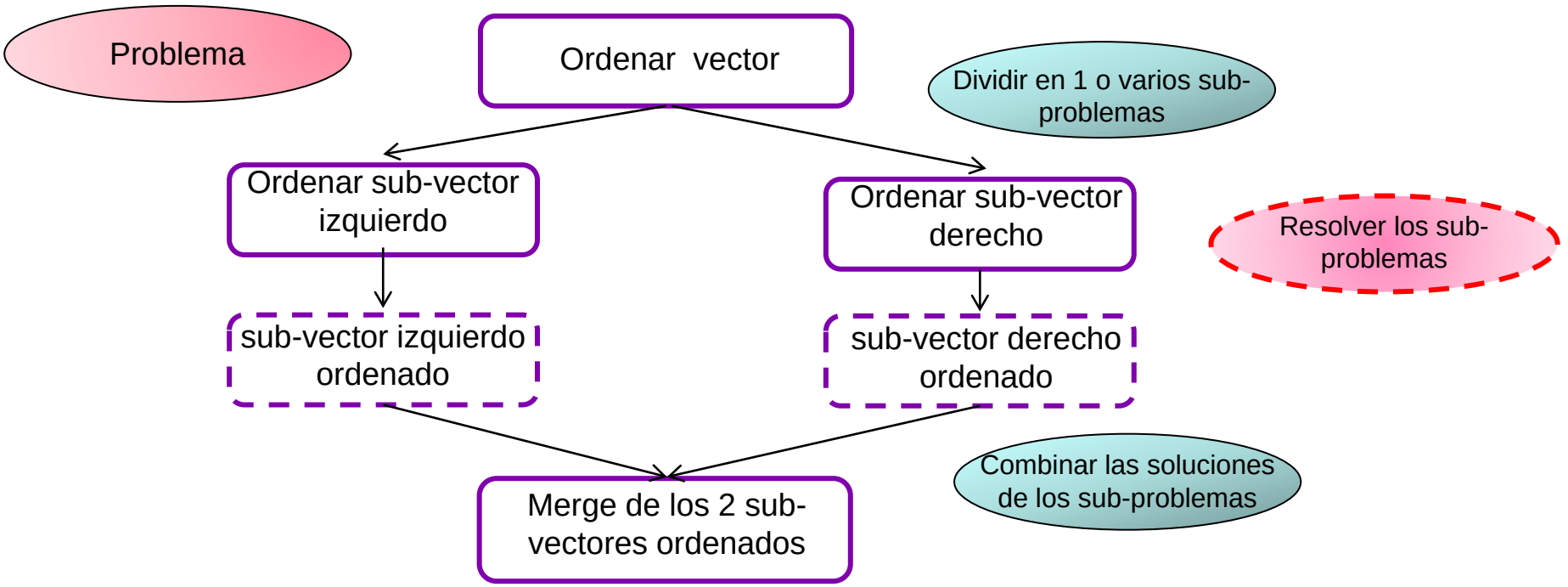


Repaso de Recursión

- Ejemplo 2: Algoritmo de ordenación ***MergeSort***

La estrategia del algoritmo consiste en dividir el vector en 2 partes (sub-vectores), ordenarlos y luego hacer un Merge de estos sub-vectores ya ordenados. Cada uno de esos sub-vectores se ordenan aplicando la misma estrategia, hasta tanto el vector contenga sólo un dato y en ese caso se lo devuelve (el sub-vector está ordenado).

Gráficamente:

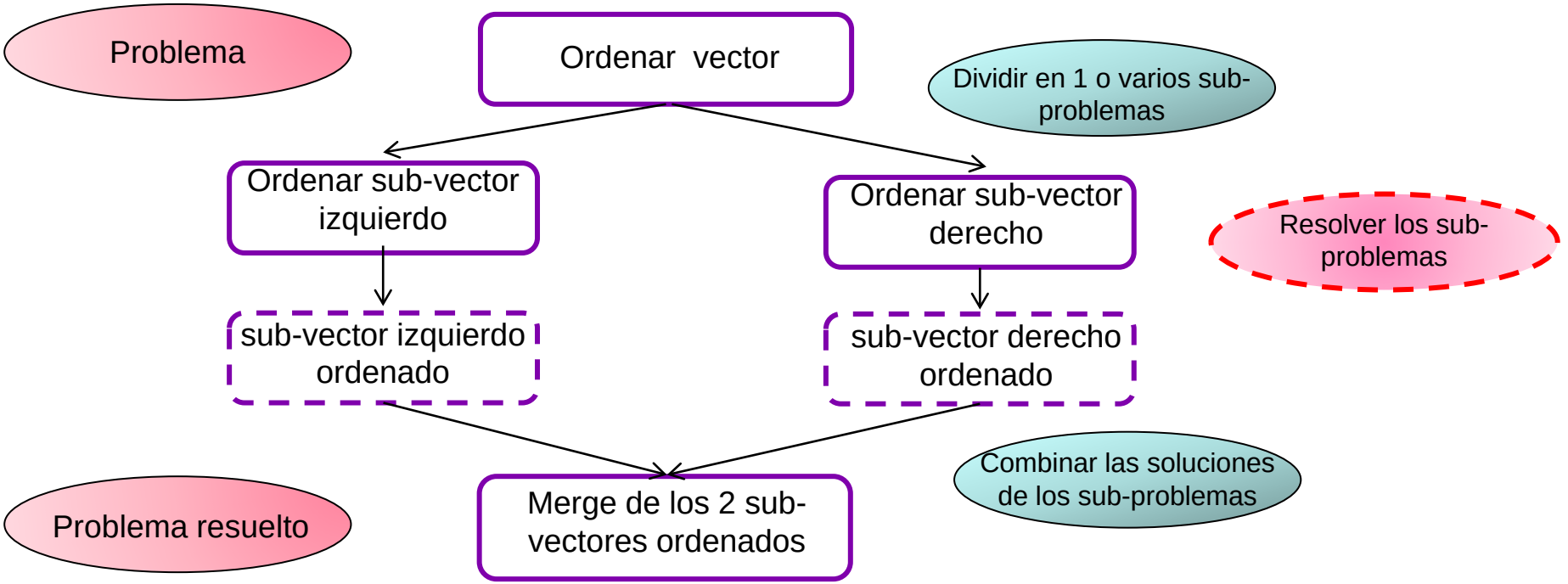


Repaso de Recursión

- Ejemplo 2: Algoritmo de ordenación ***MergeSort***

La estrategia del algoritmo consiste en dividir el vector en 2 partes (sub-vectores), ordenarlos y luego hacer un Merge de estos sub-vectores ya ordenados. Cada uno de esos sub-vectores se ordenan aplicando la misma estrategia, hasta tanto el vector contenga sólo un dato y en ese caso se lo devuelve (el sub-vector está ordenado).

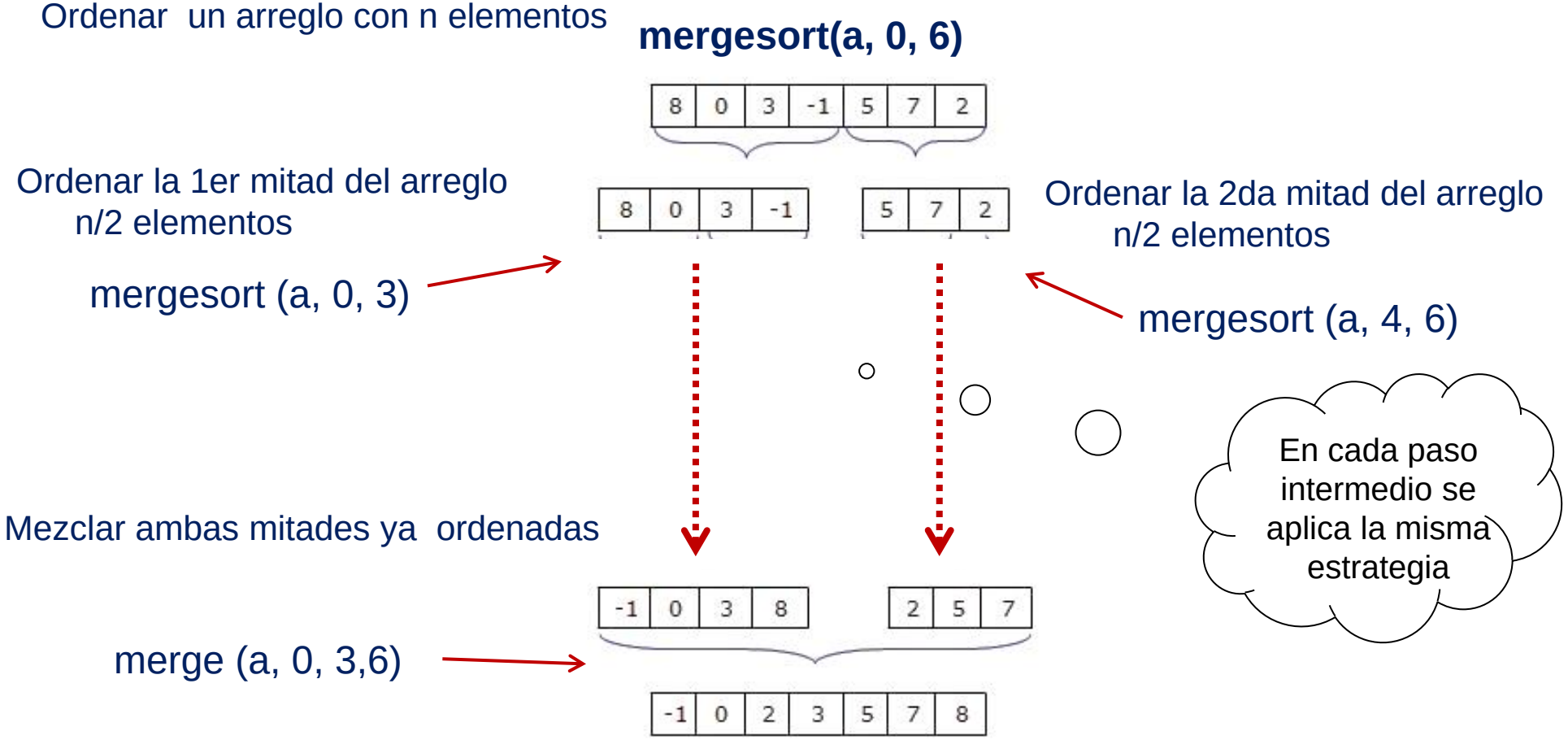
Gráficamente:



Repaso de Recursión

Ejemplo 2: Estrategia

Ordenar un arreglo con n elementos



Repaso de Recursión

Ejemplo 2:

```
    public static void mergesort (int a[],int izq, int der) {
(a)       if ( izq<der ) {
(b)                  int m = (izq+der)/2;
(c)                  mergesort (a,izq, m);
(d)                  mergesort (a,m+1, der);
(e)                  merge (a, izq, m, der);
                      }
          }
```

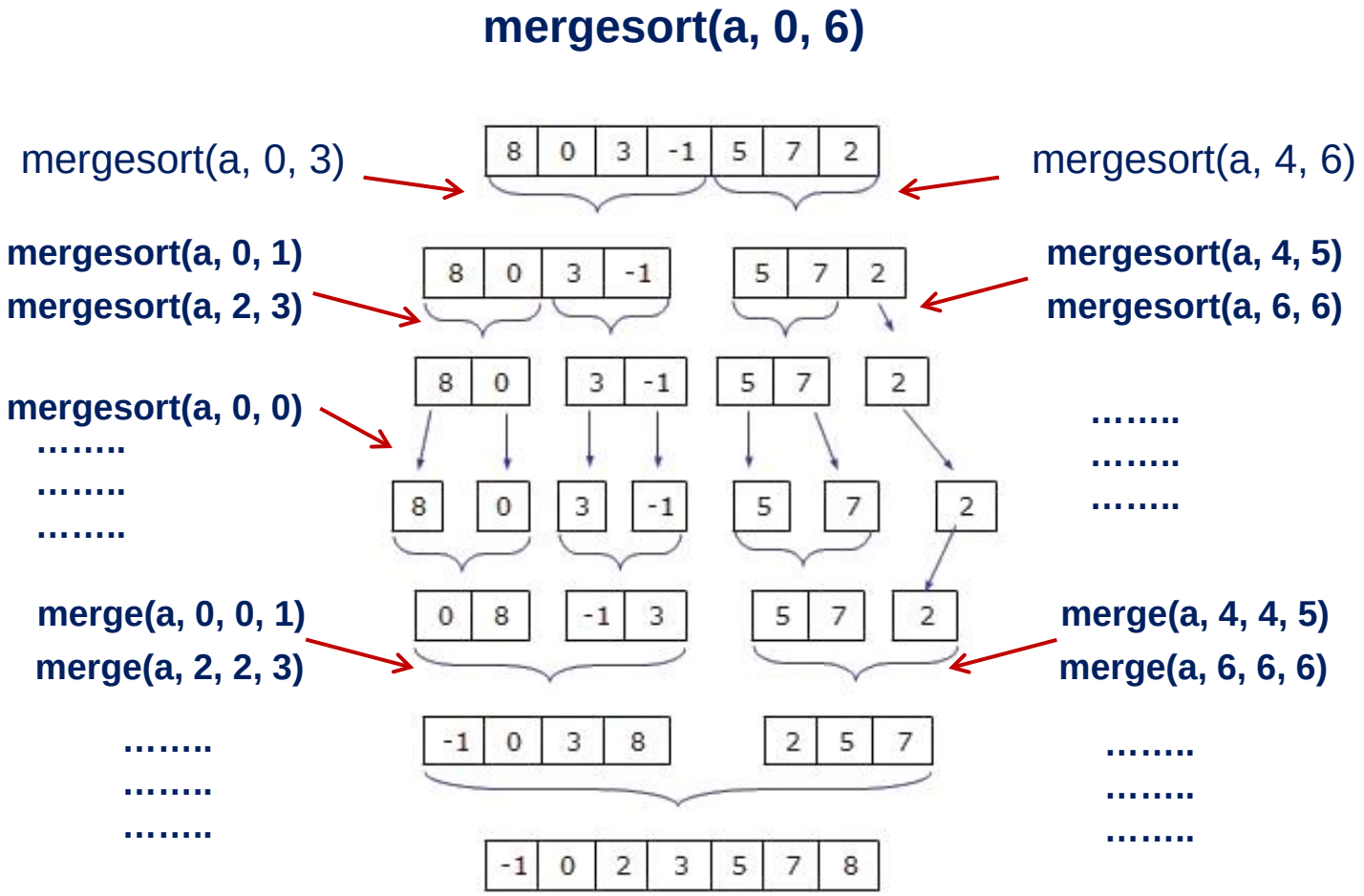
Repaso de Recursión

Ejemplo 2 (cont.) :

```
public static void merge ( int a[], int izq, int m, int der) {
   int i, j, k;
   int [] b = new int [a.length];      //array auxiliar
   for  ( i=izq; i<=der; i++ ) {       //copia ambas mitades en el array auxiliar
            b[i]=a[i]; }
   i=izq; j=m+1; k=izq;
   while (i<=m && j<=der) {   //copia el siguiente elemento más grande
            if (b[i]<=b[j])
                  a[k++]=b[i++];
            else
                  a[k++]=b[j++];
  }
   while (i<=m)              //copia los elementos que quedan de la
            a[k++]=b[i++];    //primera mitad (si los hay)
}
```

Repaso de Recursión

Ejemplo 2: Ejecución



Repaso de Recursión

Ejemplo 2: Ejecución

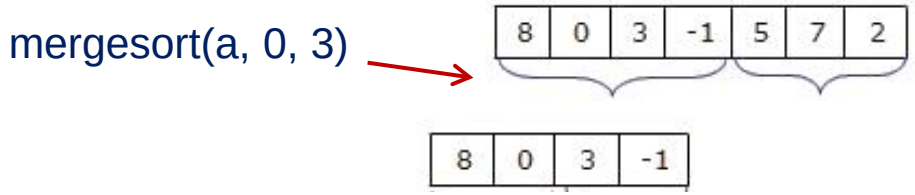
mergesort(a, 0, 6)

Repaso de Recursión

Ejemplo 2: Ejecución

mergesort(a, 0, 6)

mergesort(a, 0, 3)

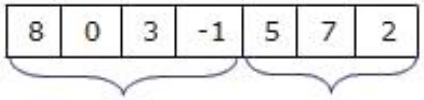


Repaso de Recursión

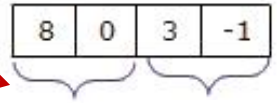
Ejemplo 2: Ejecución

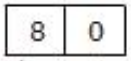
mergesort(a, 0, 6)

mergesort(a, 0, 3)



mergesort(a, 0, 1)



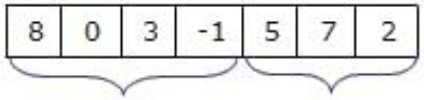


Repaso de Recursión

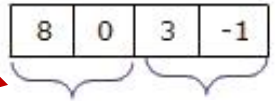
Ejemplo 2: Ejecución

mergesort(a, 0, 6)

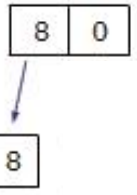
mergesort(a, 0, 3)



mergesort(a, 0, 1)

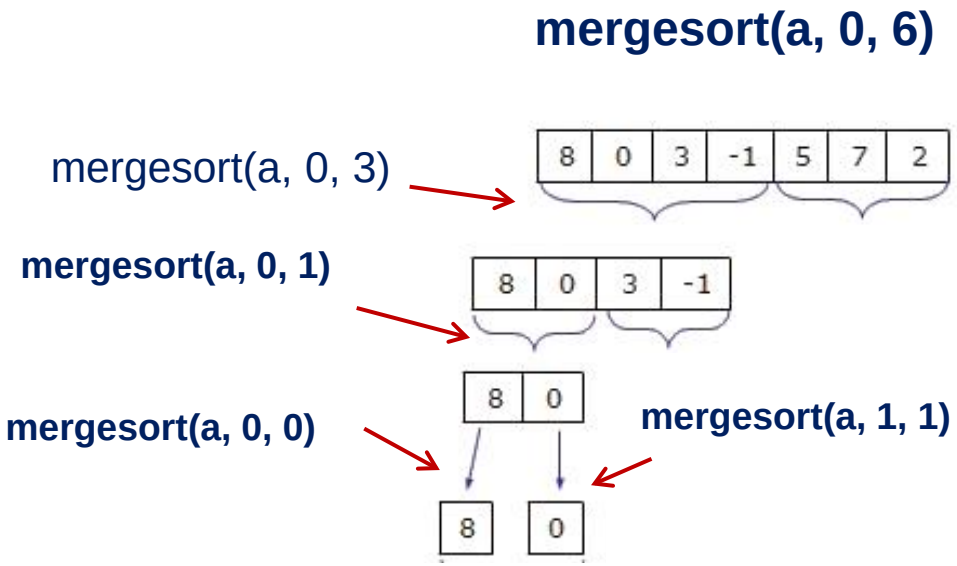


mergesort(a, 0, 0)



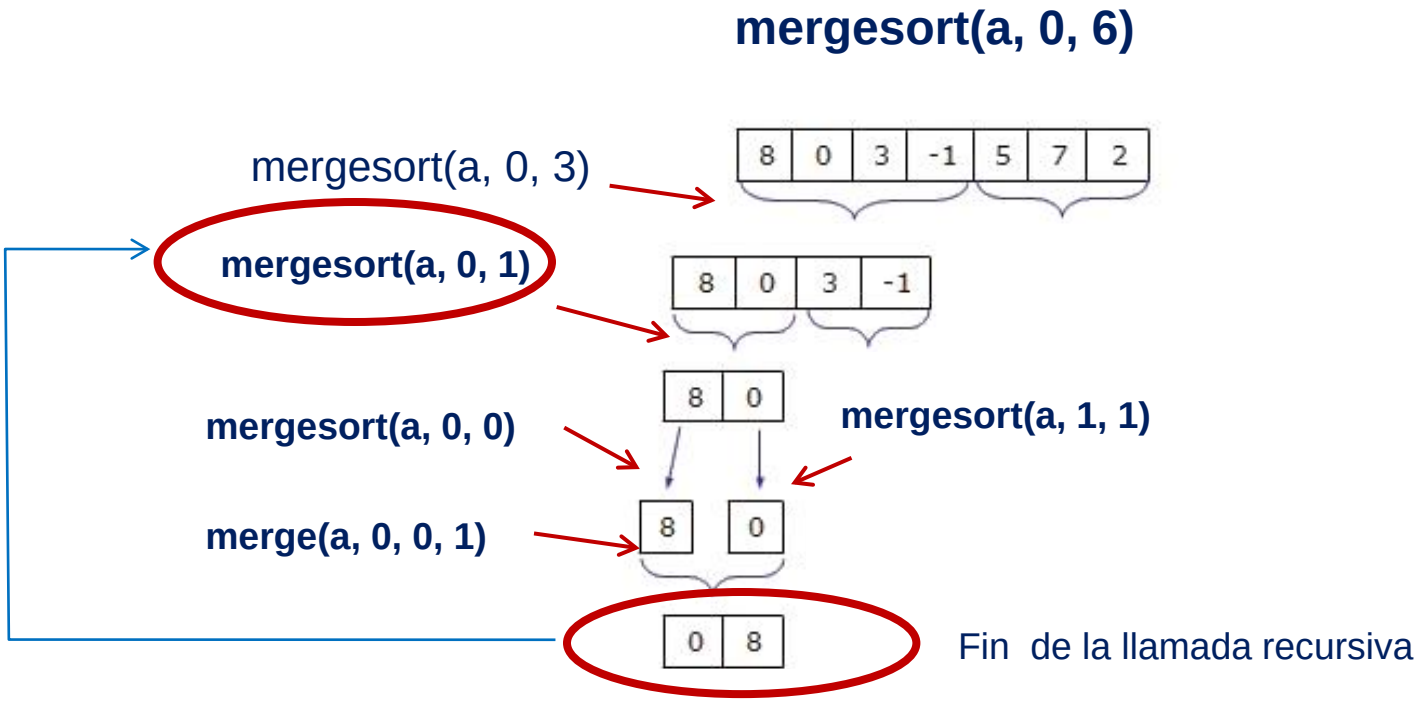
Repaso de Recursión

Ejemplo 2: Ejecución



Repaso de Recursión

Ejemplo 2: Ejecución



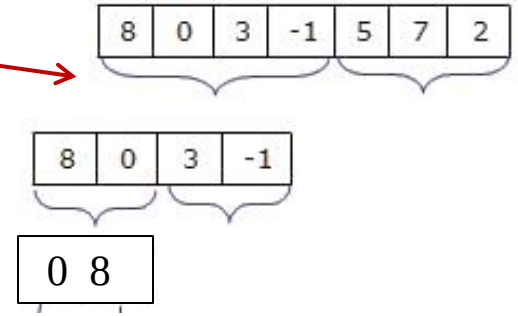
Repaso de Recursión

Ejemplo 2: Ejecución

mergesort(a, 0, 6)

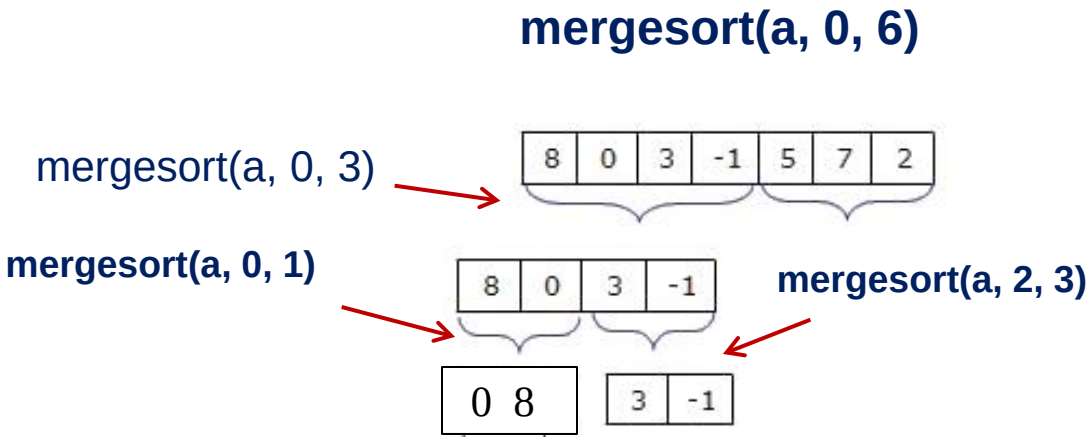
mergesort(a, 0, 3)

mergesort(a, 0, 1)



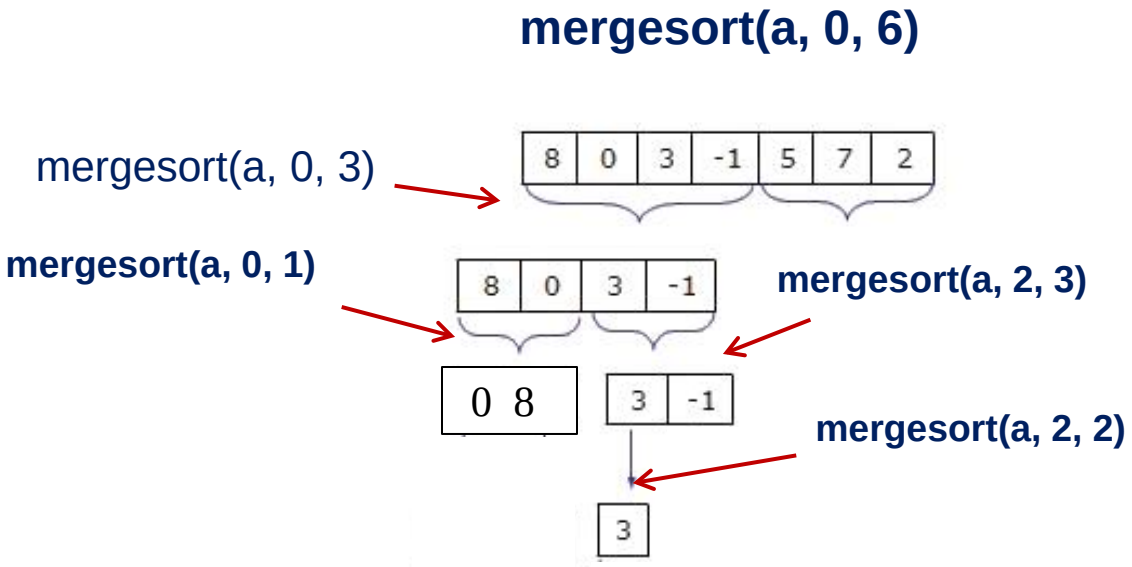
Repaso de Recursión

Ejemplo 2: Ejecución



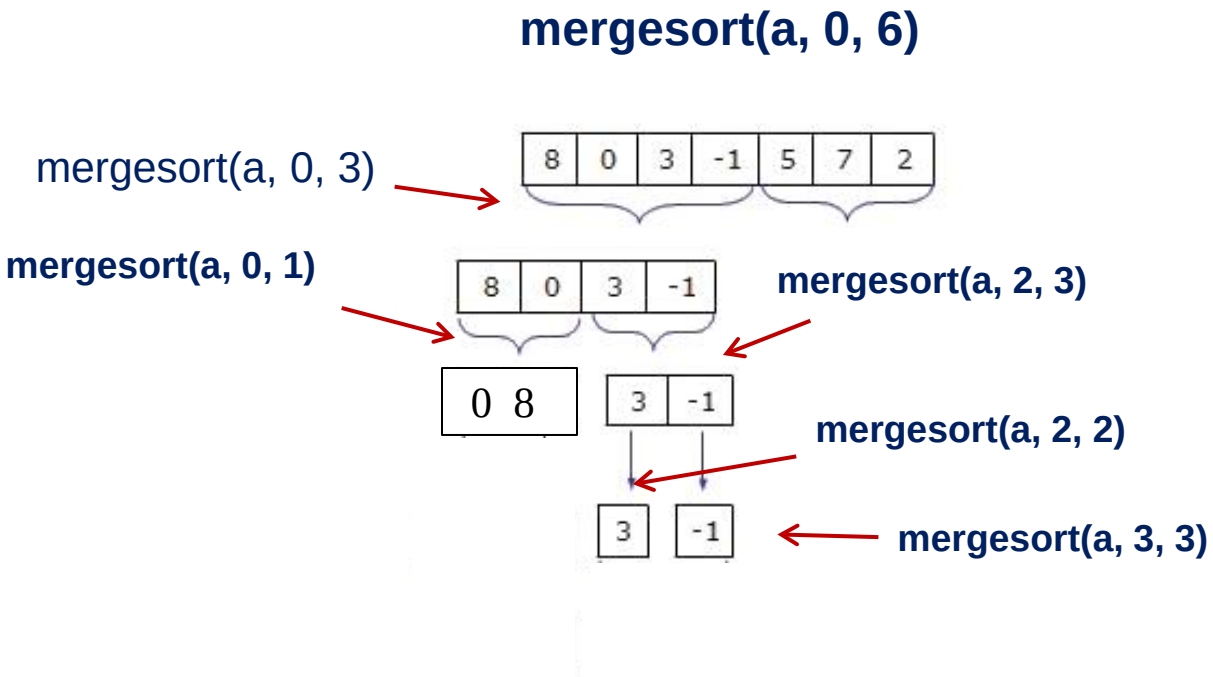
Repaso de Recursión

Ejemplo 2: Ejecución



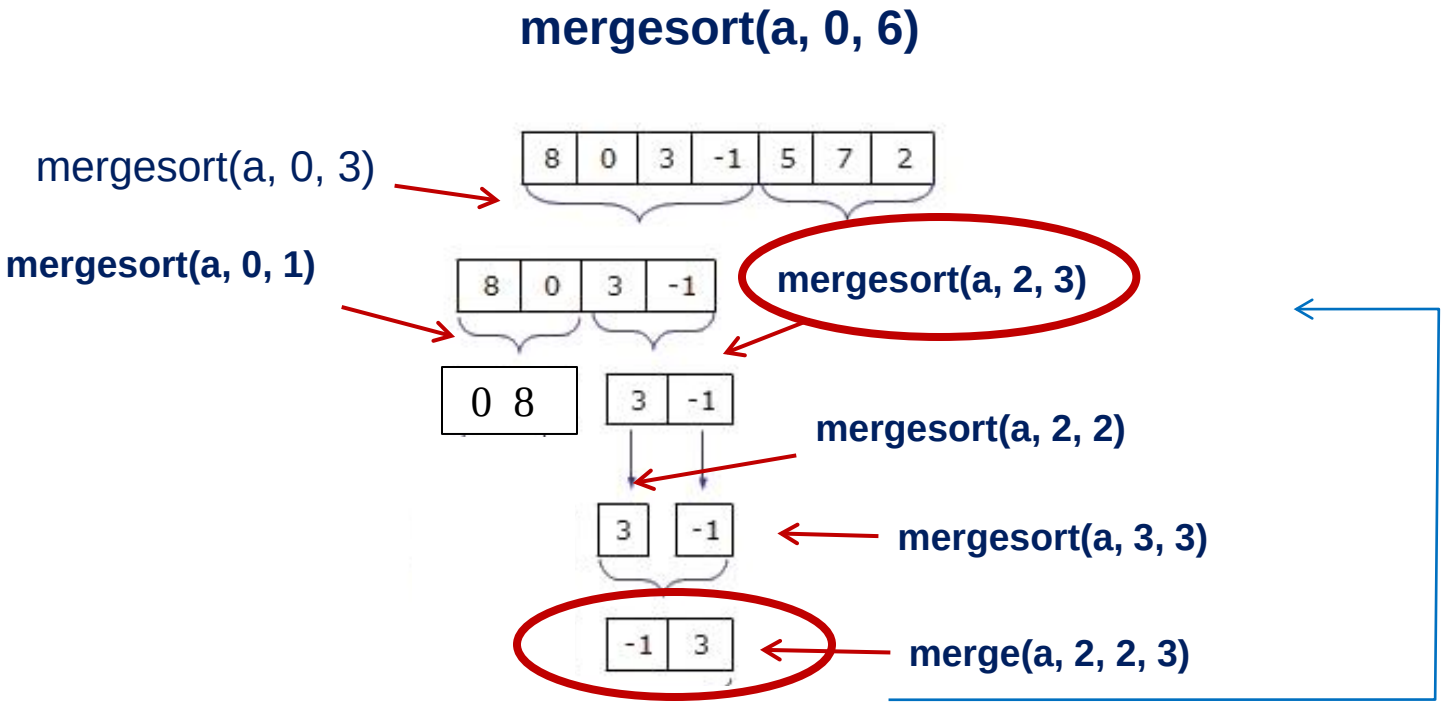
Repaso de Recursión

Ejemplo 2: Ejecución



Repaso de Recursión

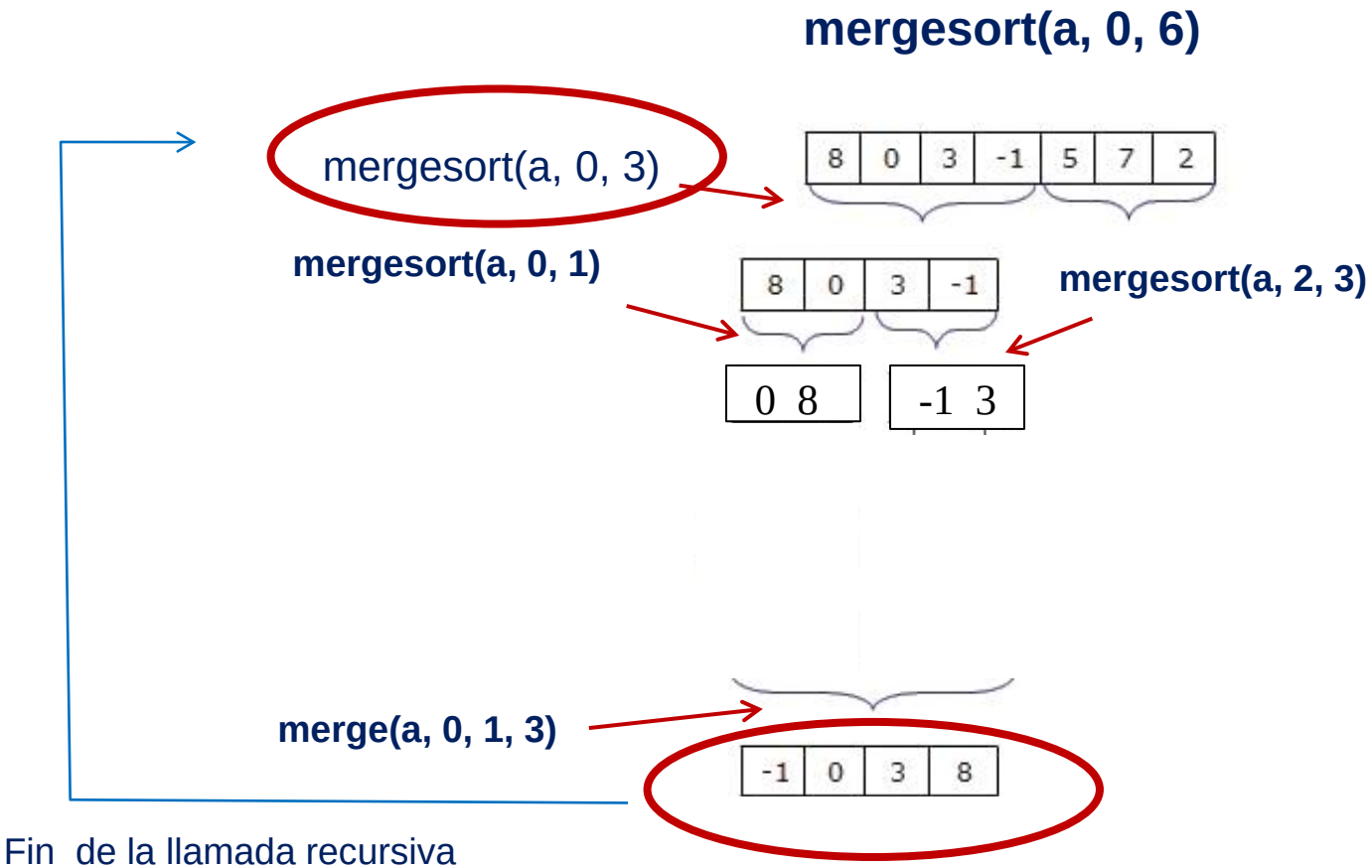
Ejemplo 2: Ejecución



Fin de la llamada recursiva

Repaso de Recursión

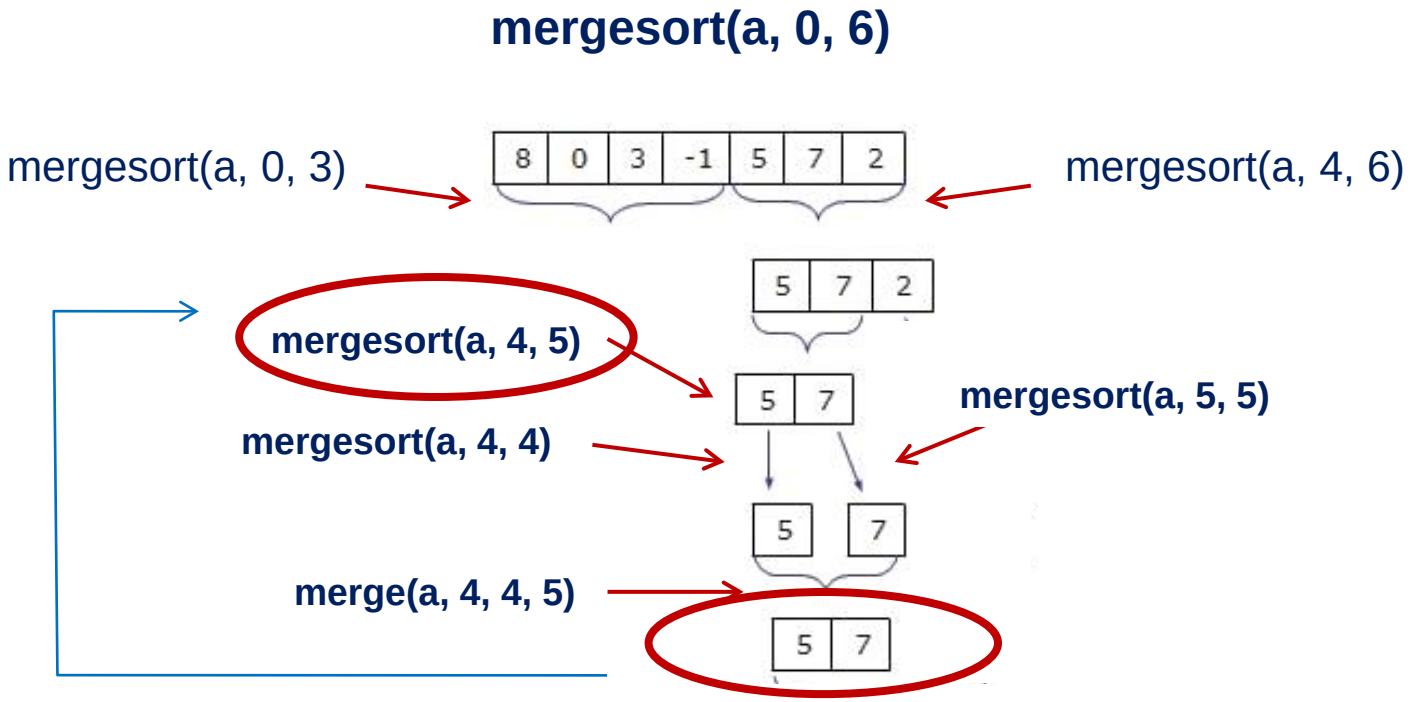
Ejemplo 2: Ejecución



Fin de la llamada recursiva

Repaso de Recursión

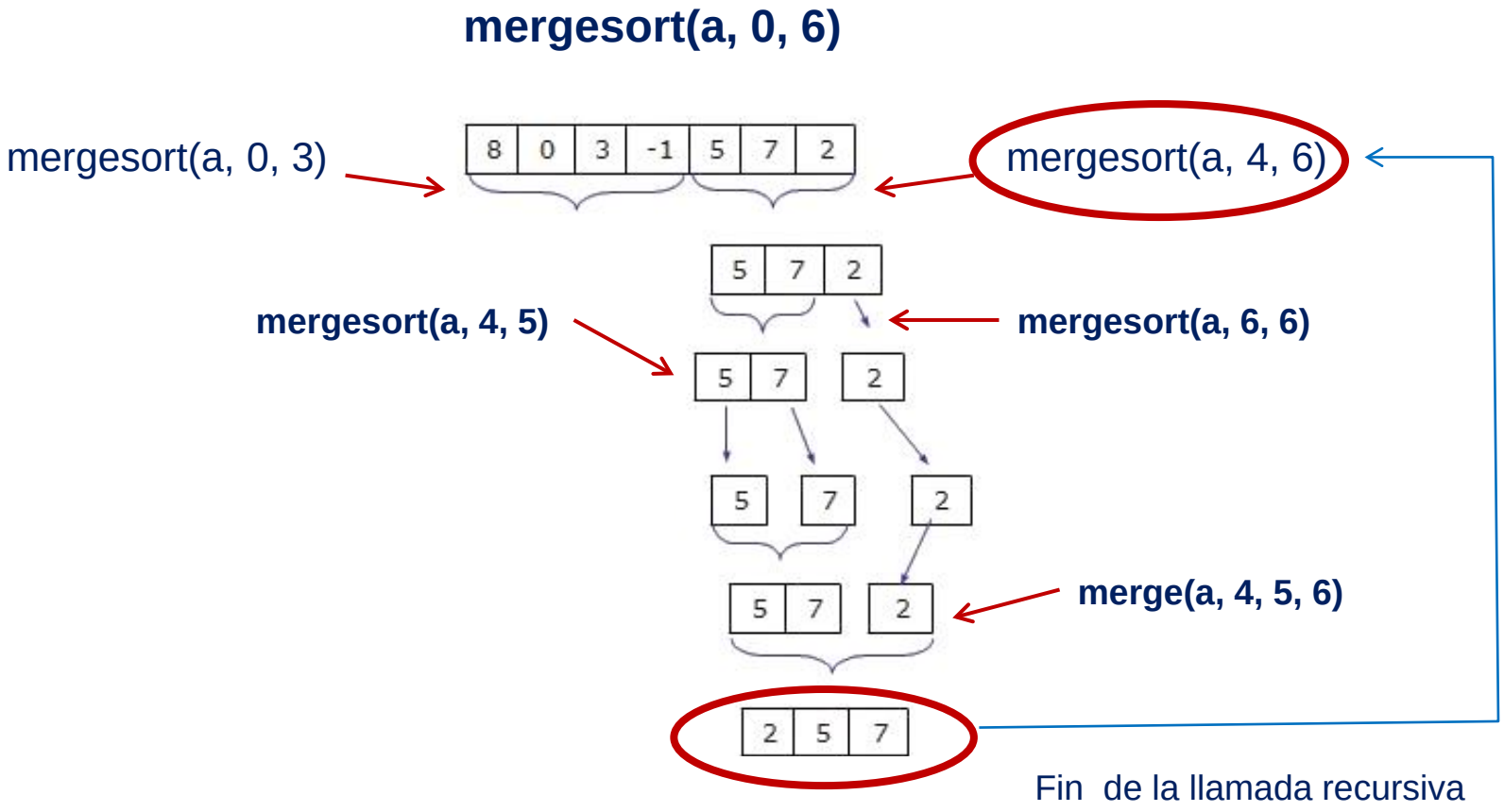
Ejemplo 2: Ejecución



Fin de la llamada recursiva

Repaso de Recursión

Ejemplo 2: Ejecución



Repaso de Recursión

Ejemplo 2: Ejecución