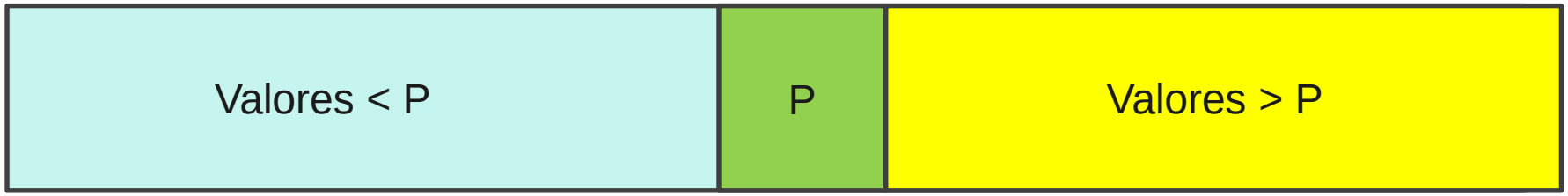
UNLaM

Métodos de ordenamiento no elementales

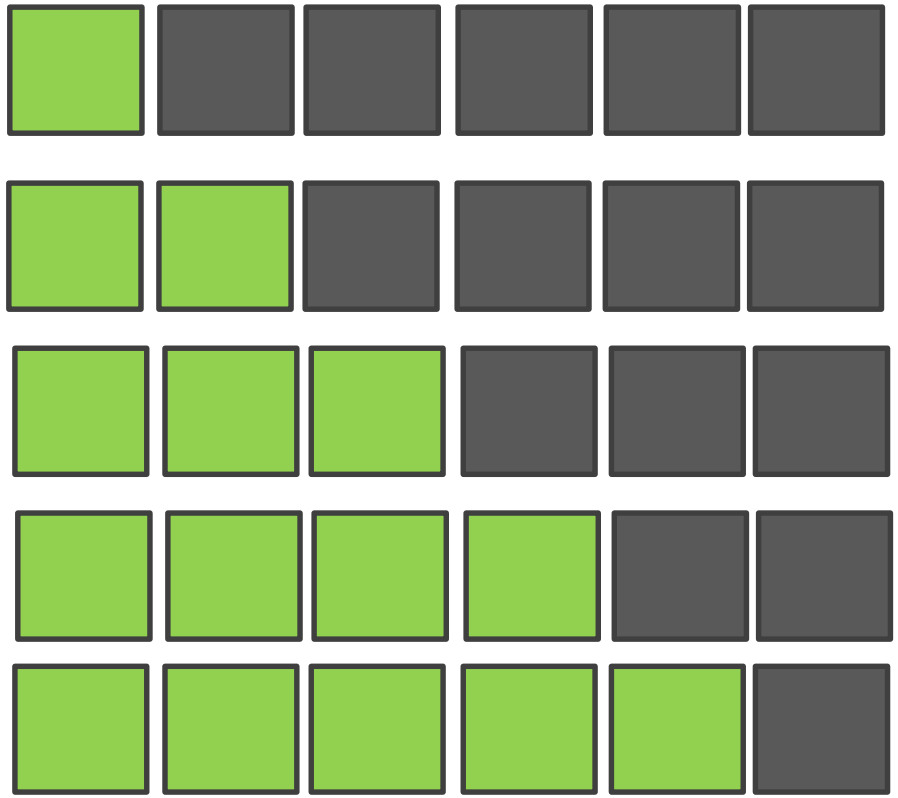
Quicksort



Elección del pivote

- **Usar el primer o último elemento del vector:** esta solución funciona bien cuando la entrada es completamente aleatoria, pero será muy deficiente si la entrada está ordenada (O(n2))
- **Pivote aleatorio:** es más seguro que la alternativa anterior, pero tiene los inconvenientes de la generación de los números aleatorios, ya que si estos son de buena calidad es una operación costosa.
- **Mediana de 3:** se trata de calcular la mediana de 3 elementos del vector. De esta forma la división de los elementos sería más balanceado. T[i] (primer elemento)T[j] (último elemento)T[(i+j) div 2] (elemento del medio)

Usar el primer o último elemento del vector

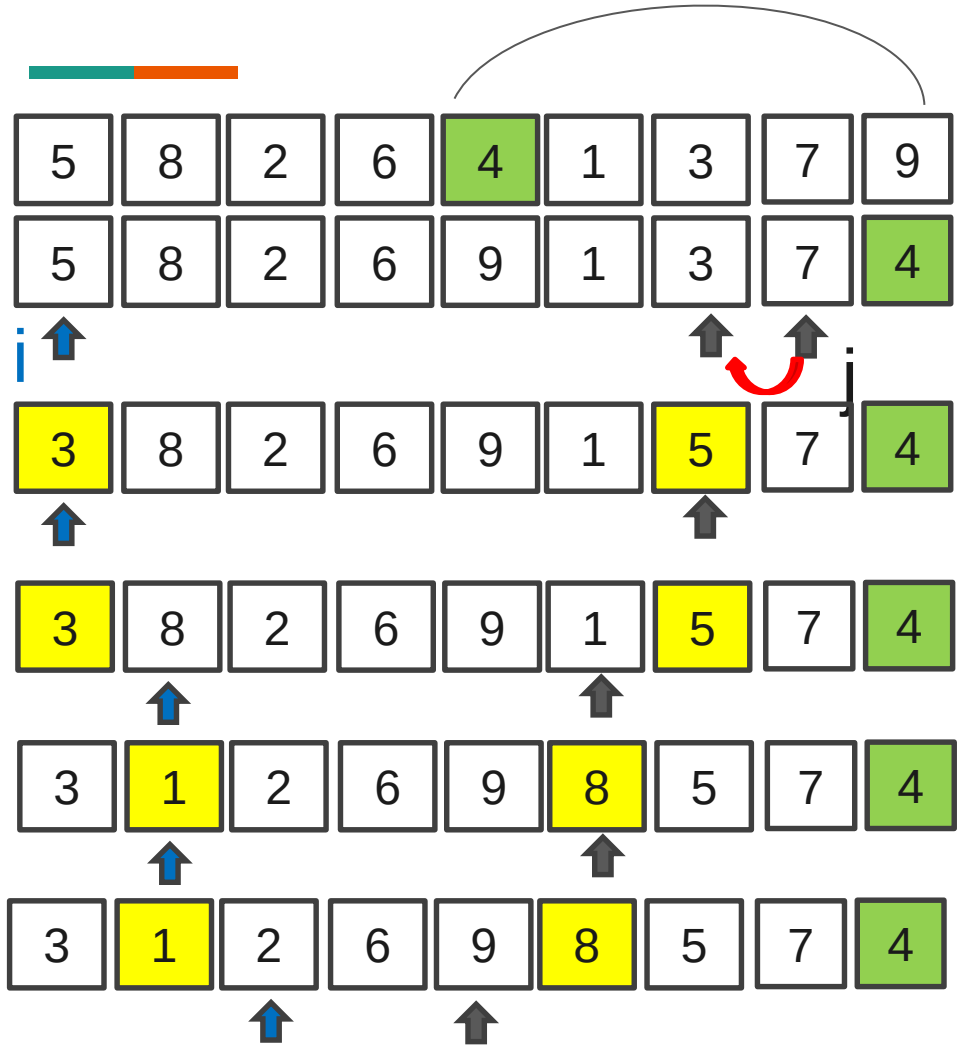


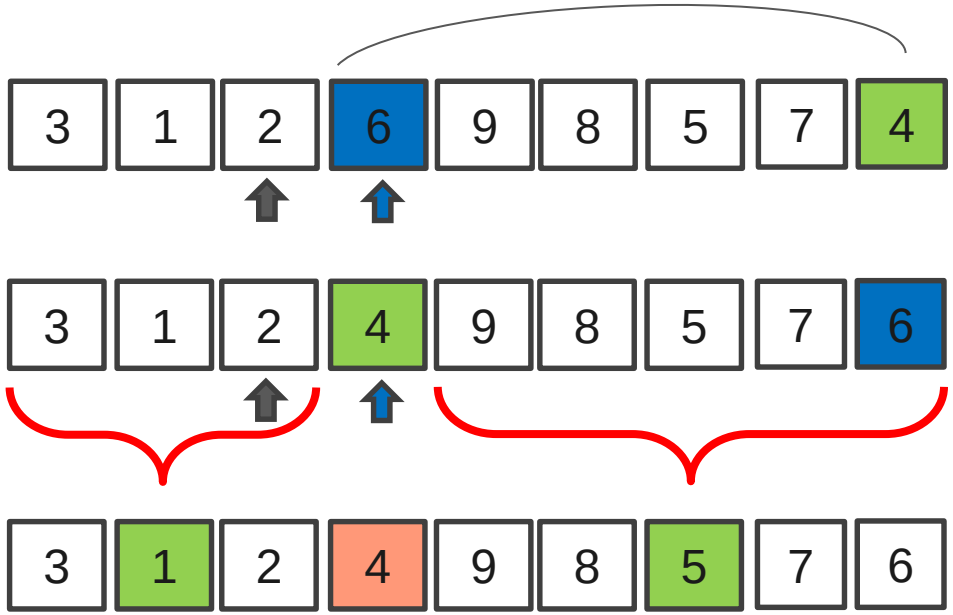
La entrada está ordenada

La complejidad computacional es O(n2)

Por iteración ordeno un solo elemento

Ejemplo

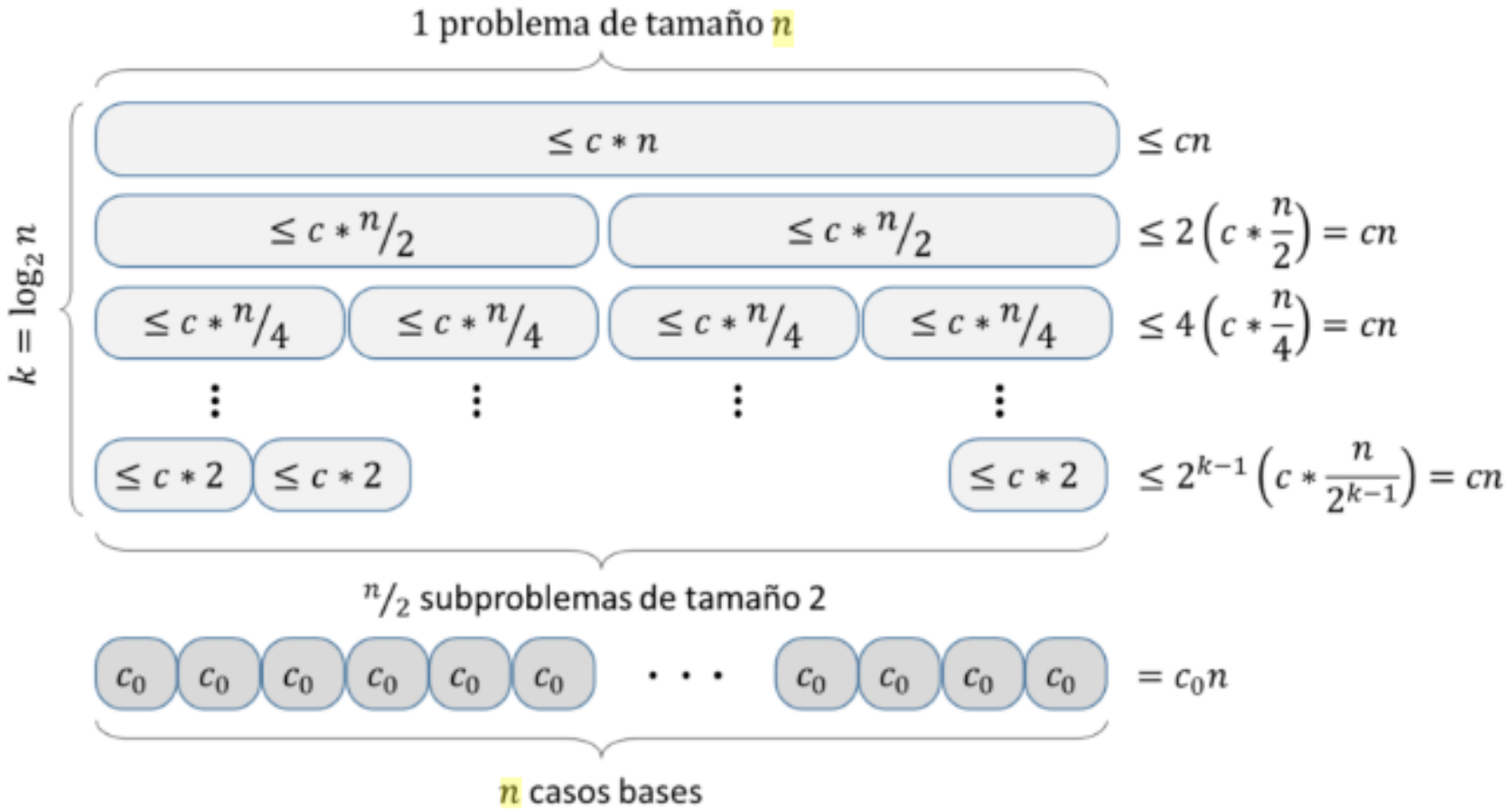

5 8 2 6 4 1 3 7 9
5 8 2 6 9 1 3 7 4
i
j
3 8 2 6 9 1 5 7 4
3 8 2 6 9 1 5 7 4
3 1 2 6 9 8 5 7 4
3 1 2 6 9 8 5 7 4


3 1 2 6 9 8 5 7 4
3 1 2 4 9 8 5 7 6
3 1 2 4 9 8 5 7 6

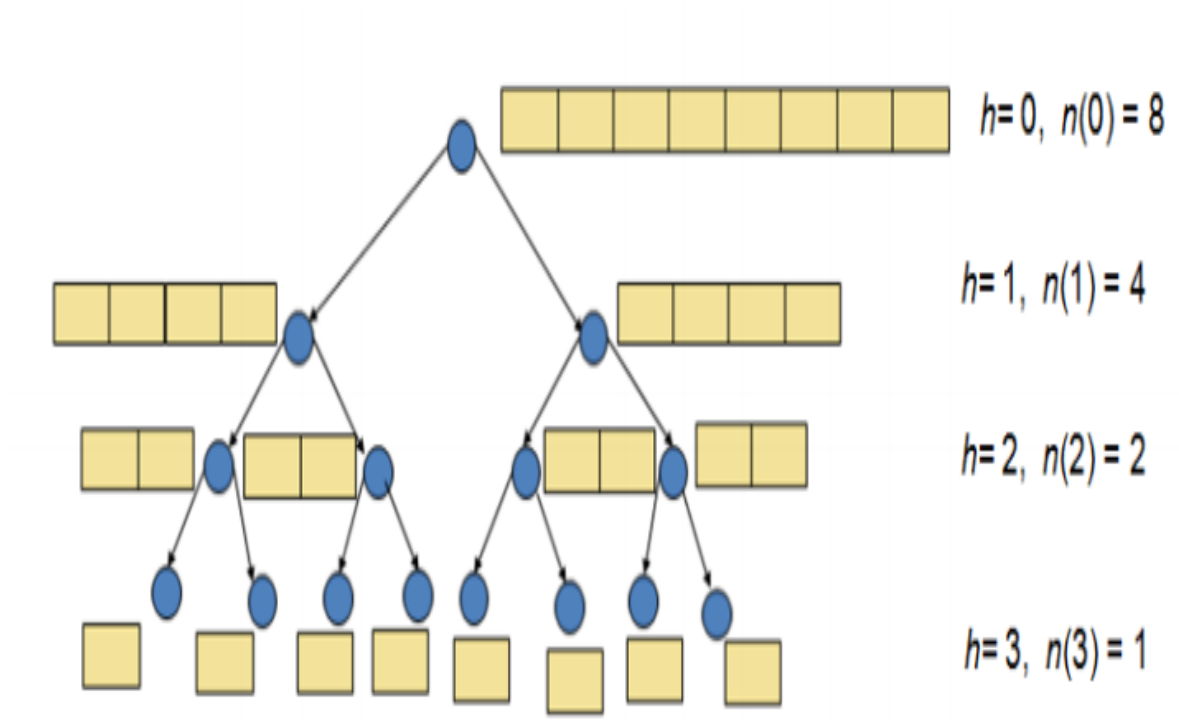
ALGORITMO RECURSIVO

```
pivote= a[(izq + der)/2] (elección del pivote)
do{
      Intercambiar el pivote con el último elemento (der), i=izq, j=der-1
      mientras (a[i]<pivote) i+ + //busco un elemento >= pivote
      mientras(a[j]>pivote) j - - //busco un elemento <= pivote
      si i< j
      Entonces     aux= a[i]
                   a[i]=a[j]
                   a[j]=aux
      i+ +
      j - -
}(mientras i<=j)
   si izq<i-1  entonces quicksort(a, izq, i-1)
   si i+1<der entonces quicksort(a, i+1, der)
Fin ALGORITMO RECURSIVO
```

Complejidad



Complejidad de la altura de un árbol binario



$2^{h+1}-1=1$

$2^{h+1}-1=n$

$2^{h+1}=n+1$

$h+1=\log_2(n+1)$

Ejemplo Fusión