MergeSort, parte 2

¿Cuál es la complejidad del MergeSort?

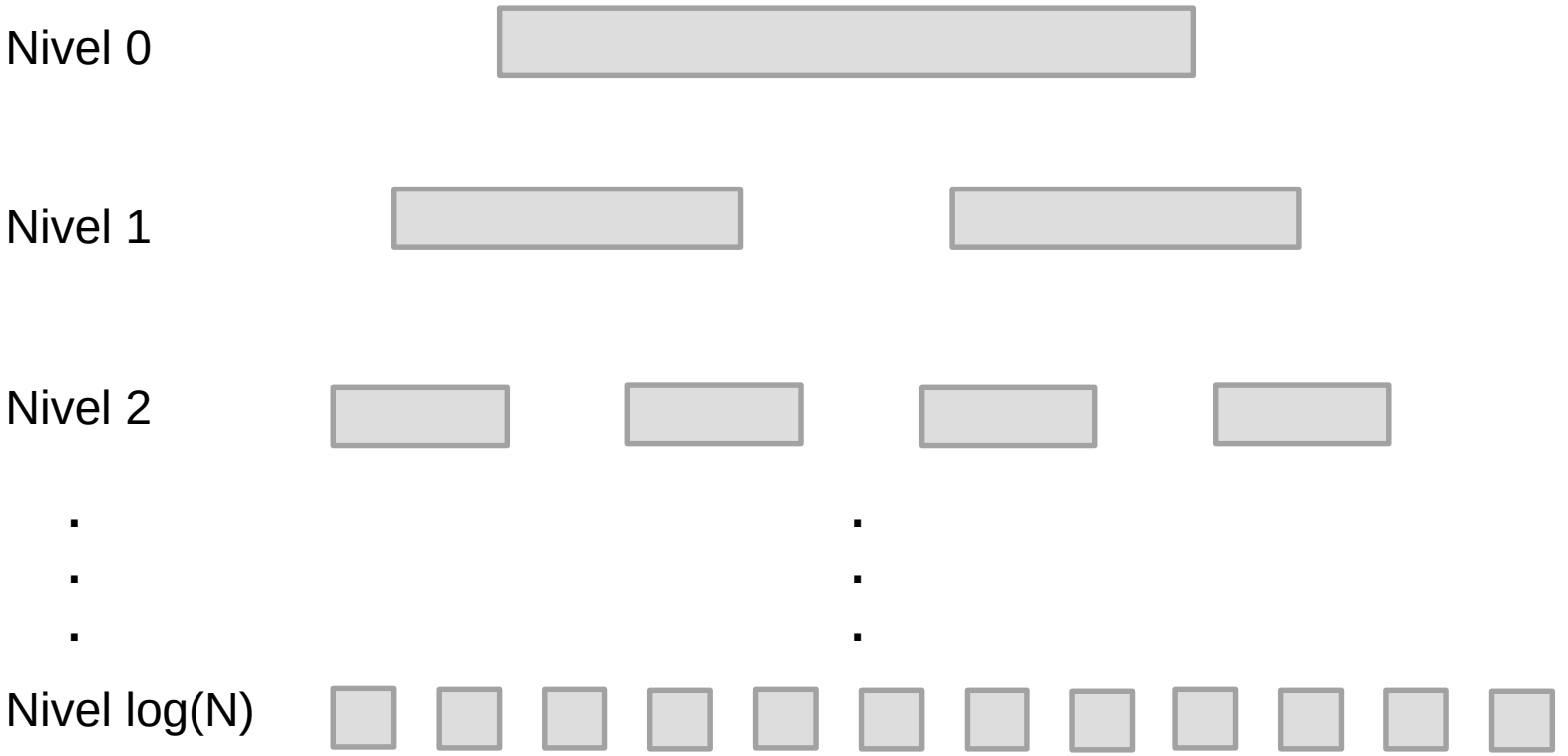
Preguntémonos primero: ¿cuál es la cantidad de operaciones de la función *merge* en un determinado llamado*?*

```
function merge(XL, XR, M){
   i,j,k = 0
   while k < M:
      if XL_i <= XR_j
         Z_k = XL_i
         i++
      else if XL_i > XR_j
         Z_k = XR_j
         j++
      else if i >= M/2
         Z_k = XR_j
         j++
      else
         Z_k = XL_i
         i++
      k++
   return Z
```

≈*3+7M*, siendo *M* el tamaño del arreglo resultante. Para simplificar consideremos ≤ 10*M* dado que M ≥ 1

Preguntémonos ahora: ¿cómo es el árbol de recursión resultante*?*

¿Cuál es la complejidad del MergeSort?



Hojas del árbol de recursión: casos base (arreglos de tamaño 1 ó 0)

En cada nivel j del árbol, ¿cuántos subproblemas hay y cuál es el tamaño de cada uno?

2^j y $\frac{N}{2^j}$

¿Cuál es la complejidad del MergeSort?

Número de operaciones en el nivel *j*, con *j = 0, 1, 2, … $log_2(N)$*

$\leq 2^j * 10 * \left(\frac{N}{2^j}\right) = 10N$

de subproblemas

Tamaño de los subproblemas

operaciones por subproblema

Y entonces el número total de operaciones del *mergeSort* es:

≤ *(log(N) + 1)*10N*, lo que significa *O(N.log(N))*