Mergesort

Divide and Conquer algorithms

Divide and Conquer

- Divide: Partir el problema en subproblemas que sean mas faciles de solucionar.

- Conquer: Resolver los subproblemas de manera recursiva. Si son suficientemente pequeños resolver de manera directa.

- Combine: Unir las soluciones a los subproblemas para resolver el problema principal.

Intuición Mergesort

- Dividir el arreglo en subarreglos.
- Ordenar los subarreglos recursivamente.
- “Mergear” los dos subarreglos para producir la respuesta ordenada.

Algoritmo

```
MERGE(A, p, q, r)
 1  n_1 = q − p + 1
 2  n_2 = r − q
 3  let L[1..n_1 + 1] and R[1..n_2 + 1] be new arrays
 4  for i = 1 to n_1
 5      L[i] = A[p + i − 1]
 6  for j = 1 to n_2
 7      R[j] = A[q + j]
 8  L[n_1 + 1] = ∞
 9  R[n_2 + 1] = ∞
10  i = 1
11  j = 1
12  for k = p to r
13      if L[i] ≤ R[j]
14          A[k] = L[i]
15          i = i + 1
16      else A[k] = R[j]
17          j = j + 1
```

Algoritmo

Invariant: Al inicio de cada iteración, el subarreglo A[p…k-1] contiene los k-p elementos mas pequeños de L y R. Ademas L[i] y R[j] son los elementos mas pequeños de su arreglo y no han sido copiados.

Algoritmo

```
MERGE-SORT(A, p, r)
1   if p < r
2       q = ⌊(p + r)/2⌋
3       MERGE-SORT(A, p, q)
4       MERGE-SORT(A, q + 1, r)
5       MERGE(A, p, q, r)
```

Complejidad

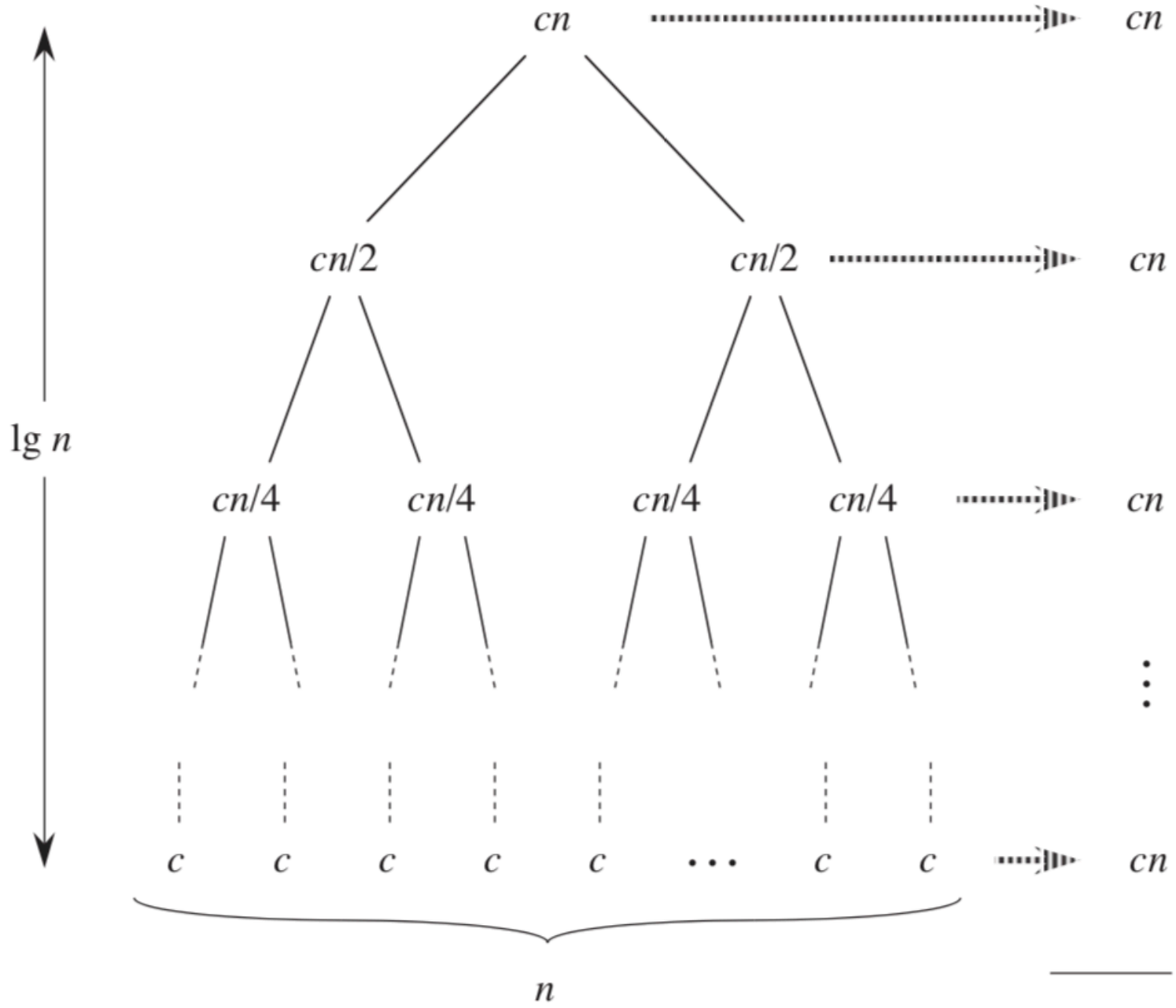
- Divide: **O(1)** Es una simple partición de arreglo por lo que toma tiempo constante.

- Conquer: **2T(n/2)** Definimos el running time como T(n) para 2 subproblema de tamaño n/2

- Combine: **O(n)** El algoritmo de combinación compara todos los elementos de los subarreglos en tiempo de n.

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 , \\ 2T(n/2) + \Theta(n) & \text{if } n > 1 . \end{cases}$$

Complejidad



(d)