Fundamentos de análisis y diseño de algoritmos

Universidad del Valle

Facultad de Ingeniería

Escuela de Ingeniería de sistemas y computación

Septiembre 2017

Divide y vencerás

Introducción

Ejemplos

Cálculo de complejidad de algoritmos recursivos

Divide y vencerás

Introducción

Algoritmos en la computación

Dividir y conquistar

- Considera recursividad
- Dividir el problema en subproblemas. Dividir hasta problema trivial
- Conquistar los subproblemas (solucionarlos recursivamente)
- Combinar las soluciones de los subproblemas para crear la solución al problema original

Divide y vencerás

Ejemplos

Algoritmos en la computación

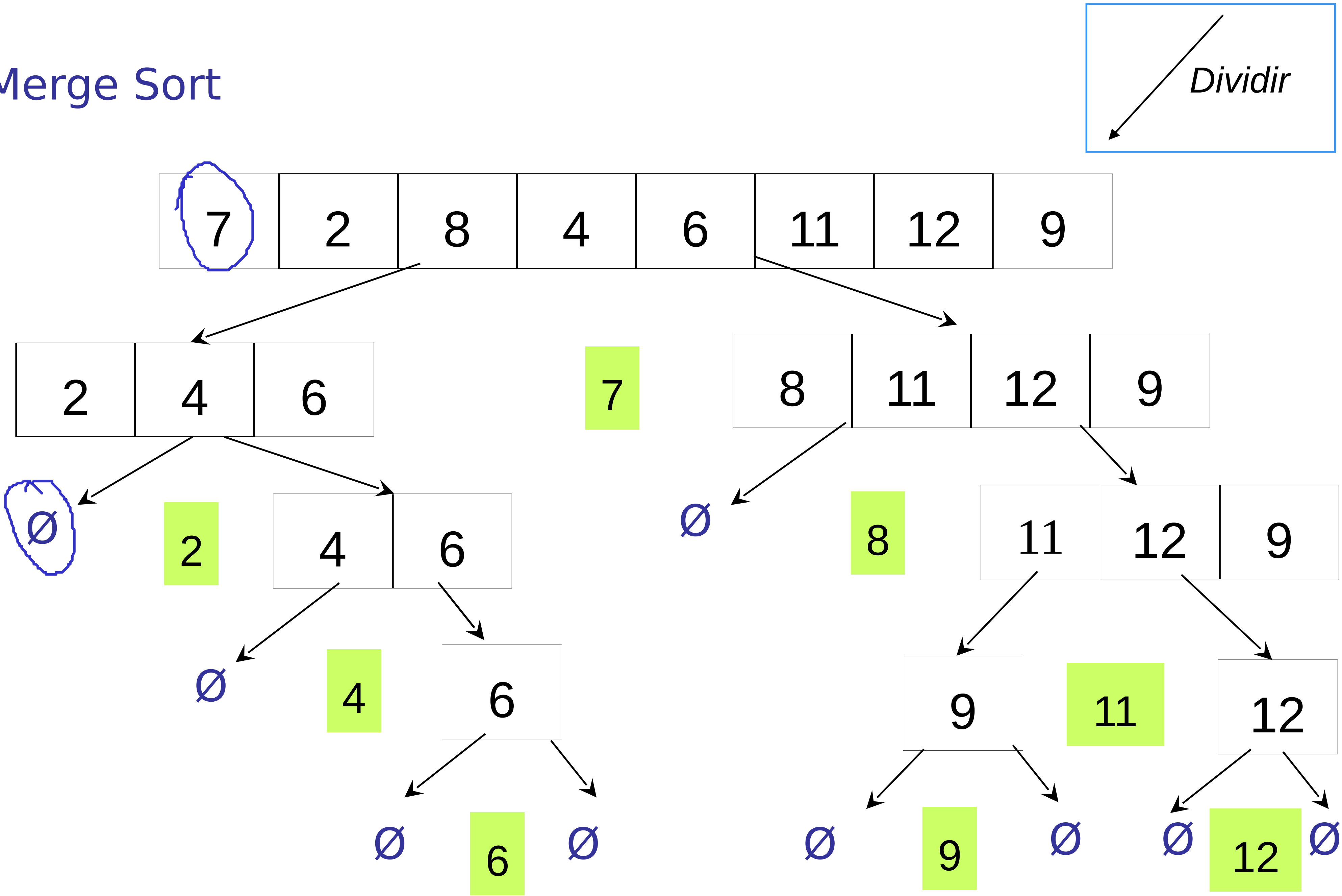
Algoritmo MergeSort

El caso trivial del algoritmo es ordenar una lista vacía (está ordenado por defecto)

- Elija un pivote (primer elemento de la lista)
- Genere dos listas: uno con los mayores y otros con los menores o iguales (sin incluir el pivote)
- Ordene estas dos listas, y aplique la estrategia hasta llegar al caso trivial
- Mezcle las listas de los menores o iguales con el pivote y la lista con los mayores

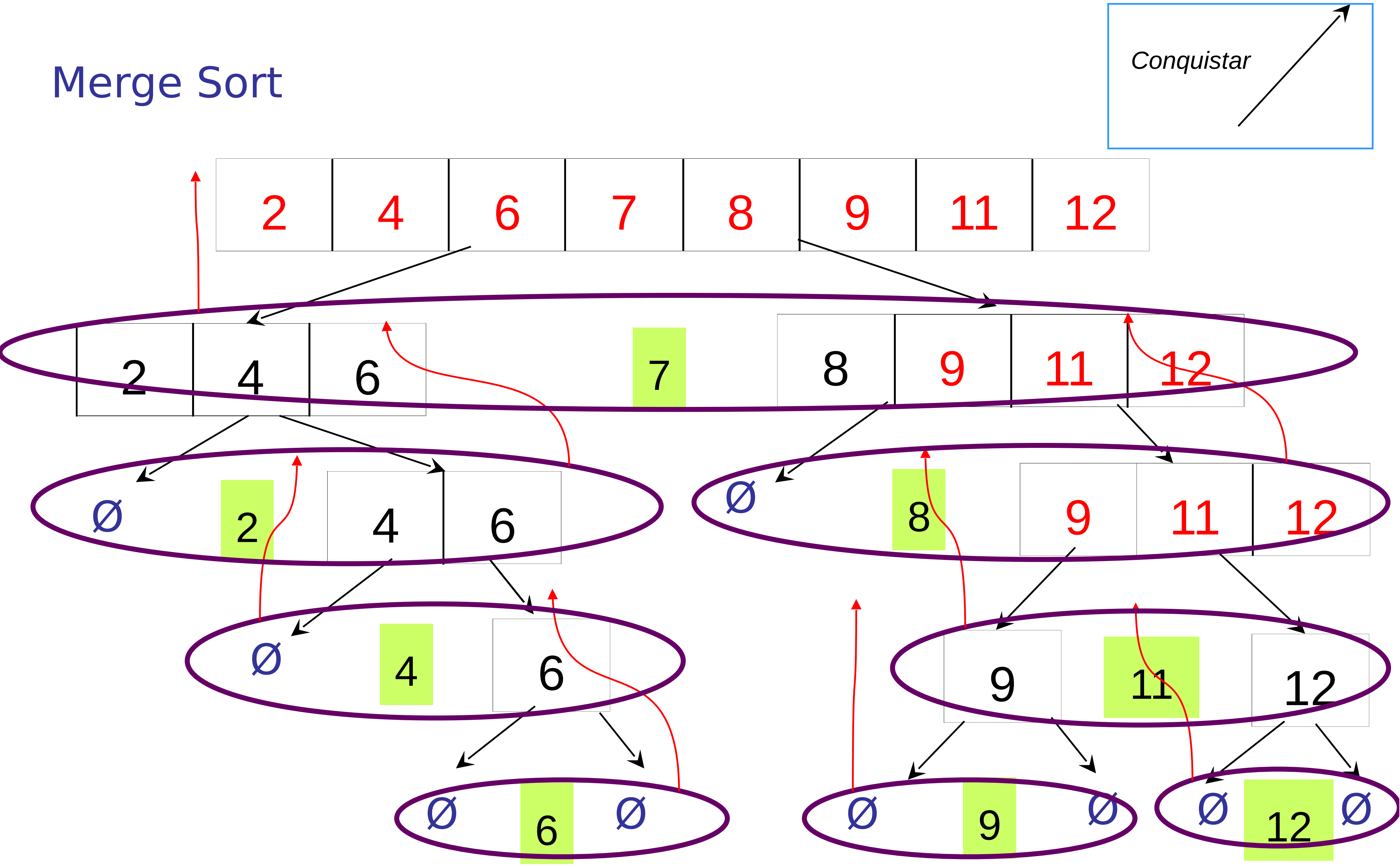
Algoritmos en la computación

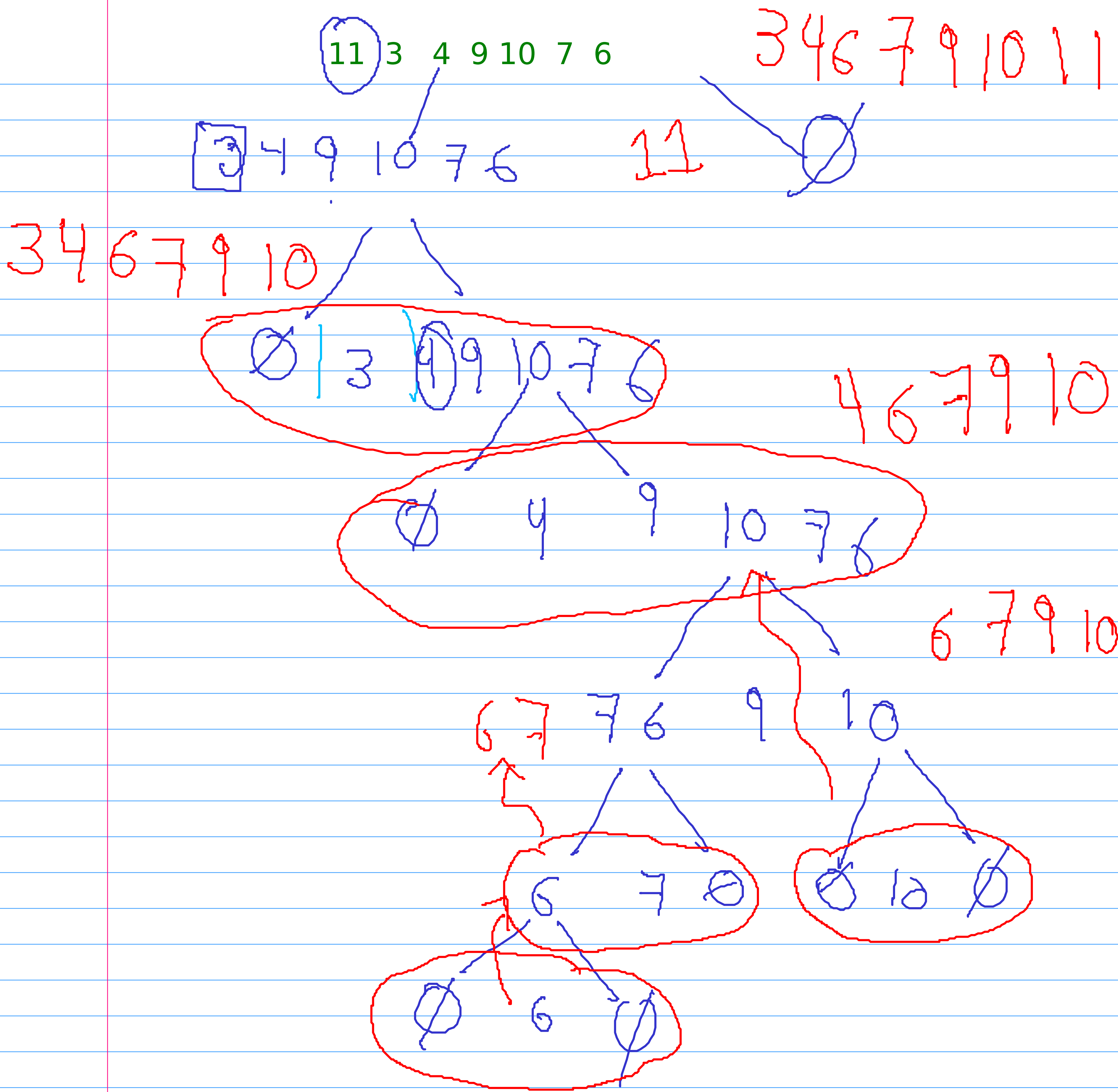
Merge Sort



Algoritmos en la computación

Merge Sort




11 3 4 9 10 7 6
3 4 6 7 9 10 11
3 4 9 10 7 6
11
3 4 6 7 9 10
Ø 3 Ø 9 10 7 6
4 6 7 9 10
Ø 4 9 10 7 6
6 7 9 10
6 7 7 6 9 10
6 7 Ø
Ø 10 Ø
Ø 6 Ø

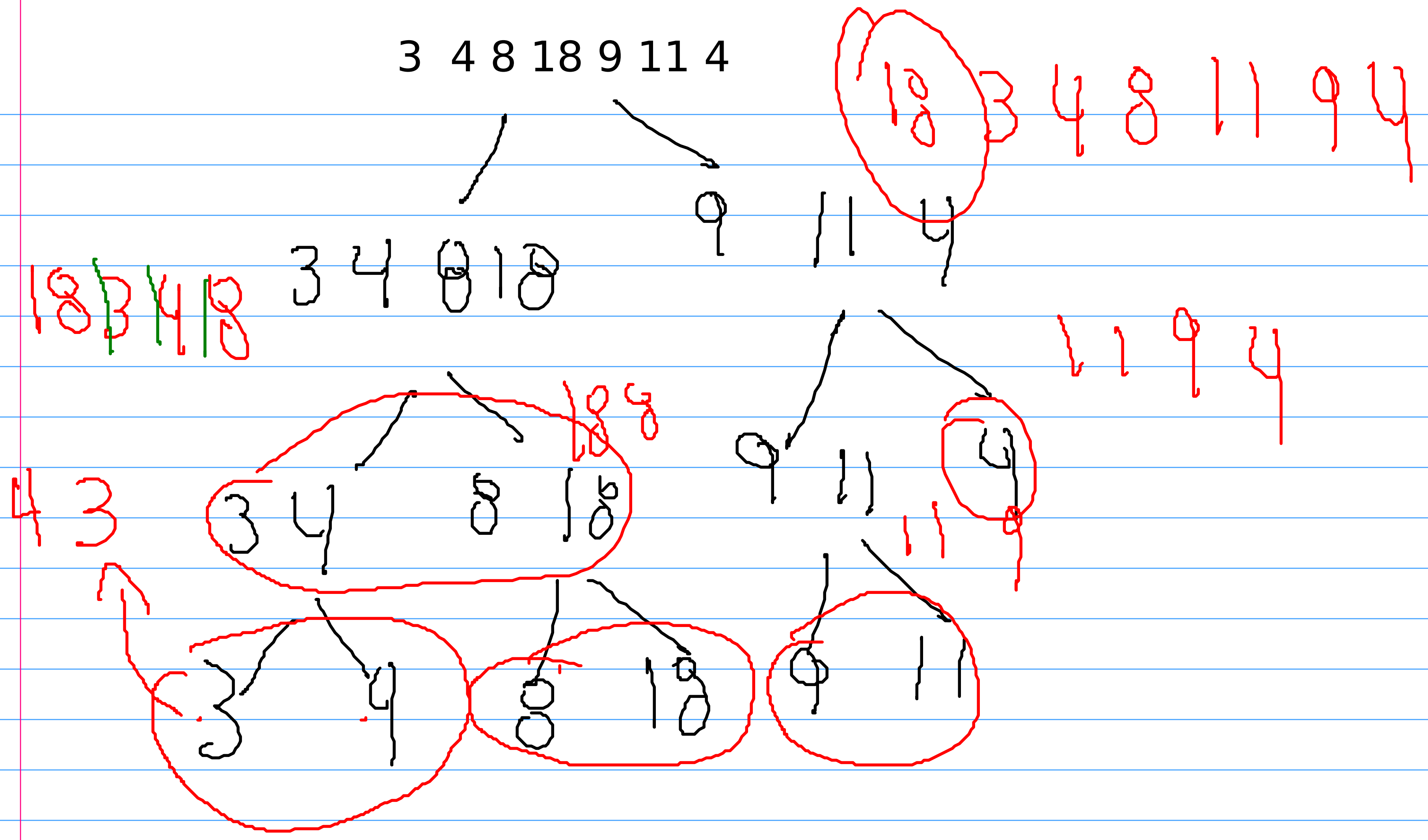
Algoritmos en la computación

Buscar el máximo de una lista

Dividir divida la lista a la mitad sucesivamente,

Conquistar Llegar al caso trivial de tener un elemento. Este será el mayor de la lista.

Combinar Combinar sucesivamente las listas, dejando como primer elemento el mayor. Así, al llegar al la lista completa el primer elemento será el mayor


3 4 8 18 9 11 4

Algoritmos en la computación

Búsqueda binaria

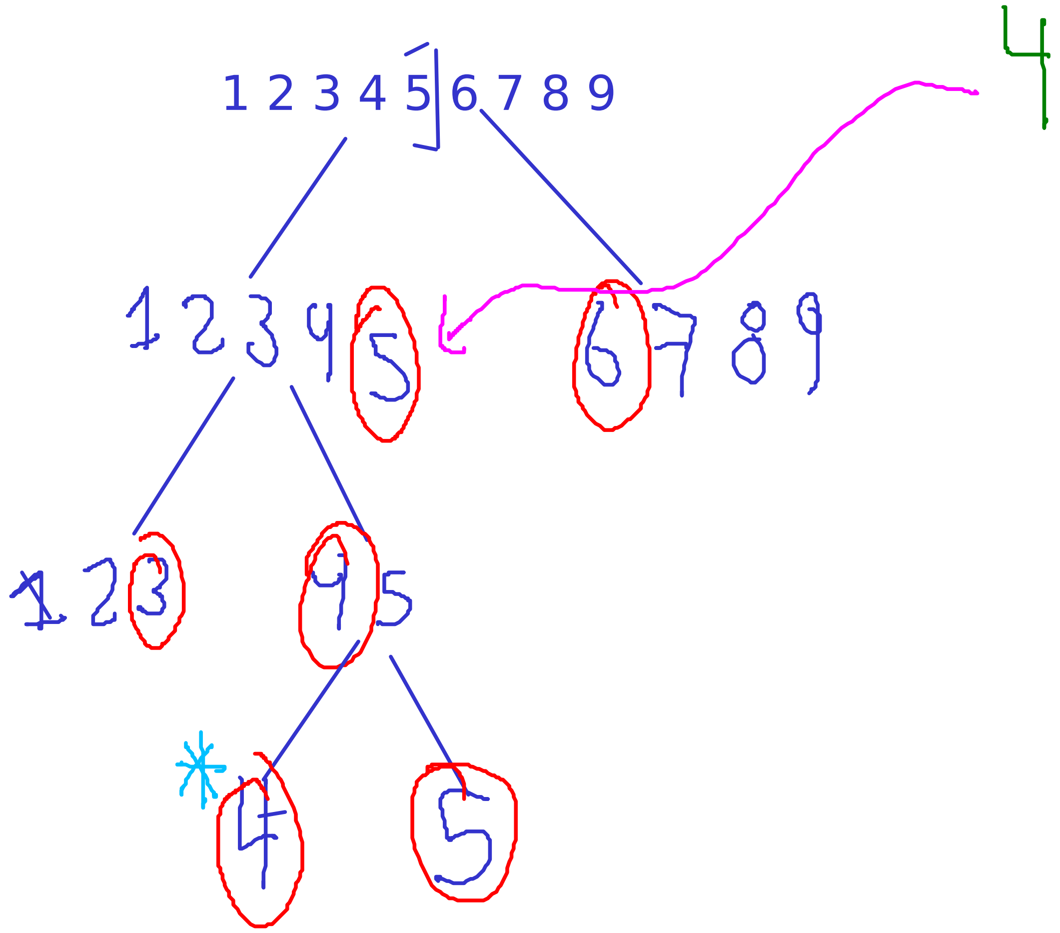
Suponga que la lista está ordenada y que busca un elemento x

Dividir divida la lista a la mitad,

Conquistar Examine el ultimo elemento de la primera lista y el primero de la segunda. Si el primero es menor o igual que x, repita dividir sobre la primera lista. En caso contrario hagalo sobre la segunda.

Combinar El espacio de búsqueda irá reduciéndose hasta encontrar el elemento.

¿Cual es el costo computacional?. ¿Si la lista está desordenada, vale la pena ordenar y aplicar este algoritmo?


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

Divide y vencerás

Cálculo de complejidad de algoritmos recursivos

Análisis de algoritmos recursivos

Un algoritmo recursivo tiene las siguientes partes:

1) Una condición de parada

2) Un llamado recursivo

El análisis de estos algoritmos lo realizaremos analizando

1) Su complejidad en un llamado

2) Cómo es el llamado recursivo y cómo cambia la entrada a medida que se realizan los llamados

3) Cómo es la forma de la entrada para llegar a la condición de parada

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)

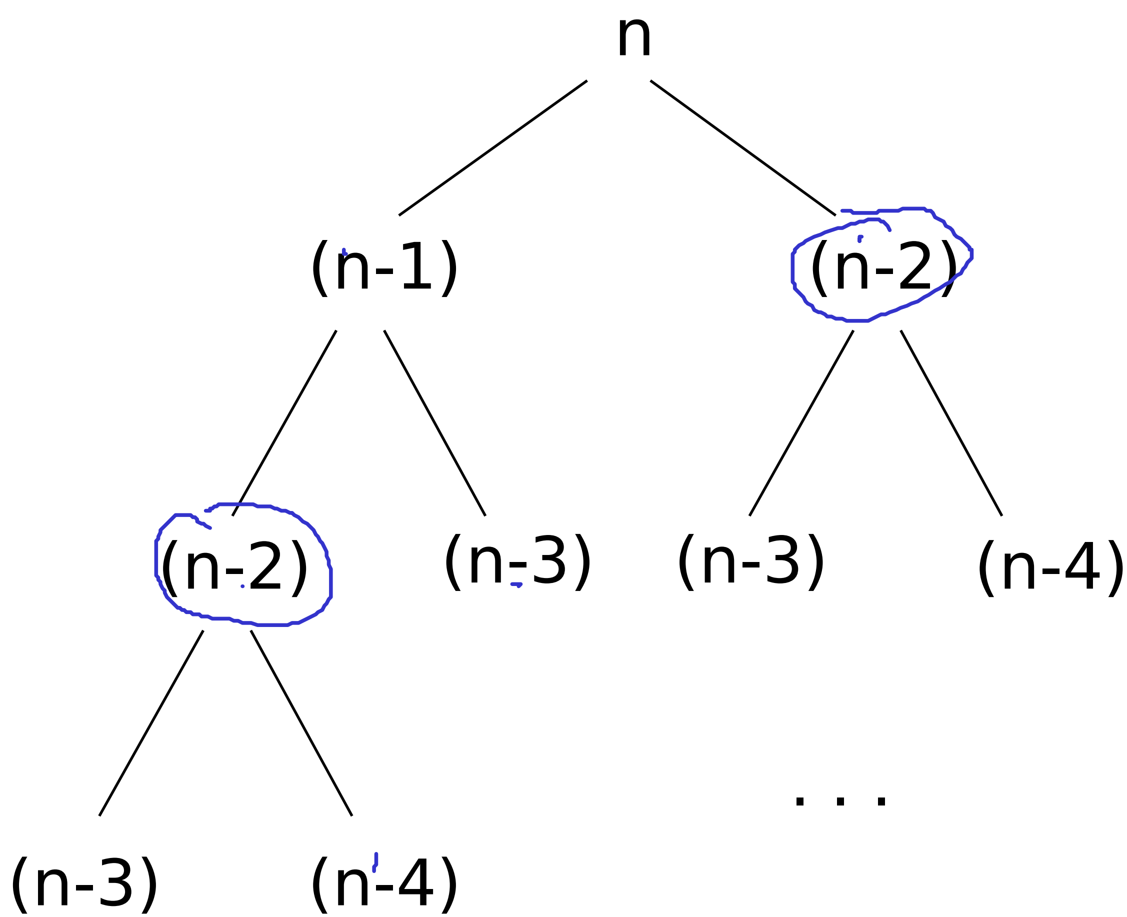
O(1)

Si n = 0 retorne 1

Sino si n = 1 retorne 2

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

Análisis de algoritmos recursivos



Análisis de algoritmos recursivos

Si observamos para cada llamado de n se realiza el llamado para n -1 y n -2, la parada se encuentra cuando n = 0 y n = 1 entonces, la complejidad de este algoritmo está dada por la relación

$$T(n)=T(n-1)+T(n-2)+O(1)$$

Se cuenta O(1) en cada llamado debido a la verificaciones que debe realizar, las cuales son ejecutadas en **tiempo constante**

Si observa en las condiciones de parada el tiempo también es constante, entonces:

$$T(0)=O(1), T(1)=1$$

$$r^2 - r - 1 \qquad \frac{1 \pm \sqrt{1-4(-1)(1)}}{2}$$

$$r = \frac{1 \pm \sqrt{5}}{2}$$

$$T^h(n) = a\left(\frac{1+\sqrt{5}}{2}\right)^n + b\left(\frac{1-\sqrt{5}}{2}\right)^n$$

$$T_p = c$$

$$c = c + c + d \qquad c = -d$$

$$T(n) = a\left(\frac{1+\sqrt{5}}{2}\right)^n + b\left(\frac{1-\sqrt{5}}{2}\right)^n - d$$

$$O(r^n)$$

Análisis de algoritmos recursivos

Siempre que se trabaje con algoritmos recursivos el cálculo de la complejidad va tener la forma

$$T(n) = T(x_1) + T(x_2) + \ldots T(x_n) + f(n)$$

Donde $T(x_i)$ indica cómo cambia la entrada en cada llamado recursivo y $f(n)$ representa la complejidad de los procesos realizados en un **sólo** paso.

Análisis de algoritmos recursivos

Tarea

Analizar el algoritmo Merge Sort para estas condiciones

Mejor caso

Particiones: $n/2$ y $n/2$ complejidad ordenar $O(1)$.

Caso promedio

Particiones: $n/8$, $7n/8$ complejidad ordenar $O(n)$

Peor caso

Particiones $n/2$ y $n/2$ complejidad ordenar $O(n)$

Mejor caso

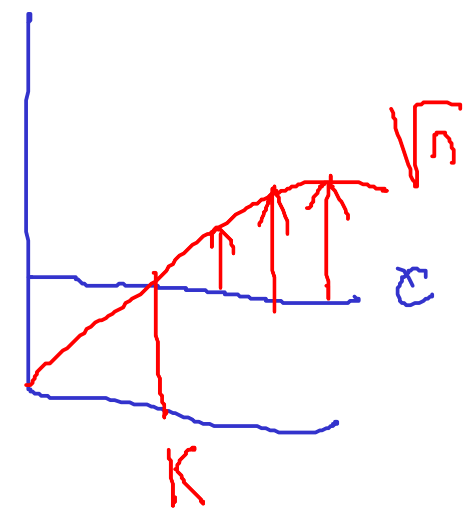
$$T(n) = 2T(n/2) + O(1)$$

$a = 2$ $\quad b = 2$ $\quad f(n) = O(1) = c$

$\log_2 2 = 1$

$f(n) = O(n^{1-\epsilon})$ $\qquad c = O(\sqrt{n})$

$$T(n) = \Theta(n)$$



$$T(n) = T(n/8) + T(7n/8) + O(n)$$

$$O(n\log(n))$$

Por caso

$$T(n) = 2T(n/2) + n \qquad n\log(n)$$

Análisis de algoritmos recursivos

Tarea

Análizar algoritmos que tienen el siguiente comportamiento.

Algoritmo 1

Particiones: n -2 y n – 4, complejidad cada paso O(n)

Algoritmo 2

Particiones: n/2 y n/3, complejidad cada paso O(log(n))

Análisis de algoritmos recursivos

Tarea

Analizar el siguiente algoritmo:

Conquistar

Algoritmo(n)

Si n = 0 retorne 1

Si n = 1 retorne 2

O(1)

Combinar

Sino Si ~~n es par~~ retorne n + f($\lfloor n/2 \rfloor$)

Dividir

$$T(n) = T(n/2) + O(1)$$

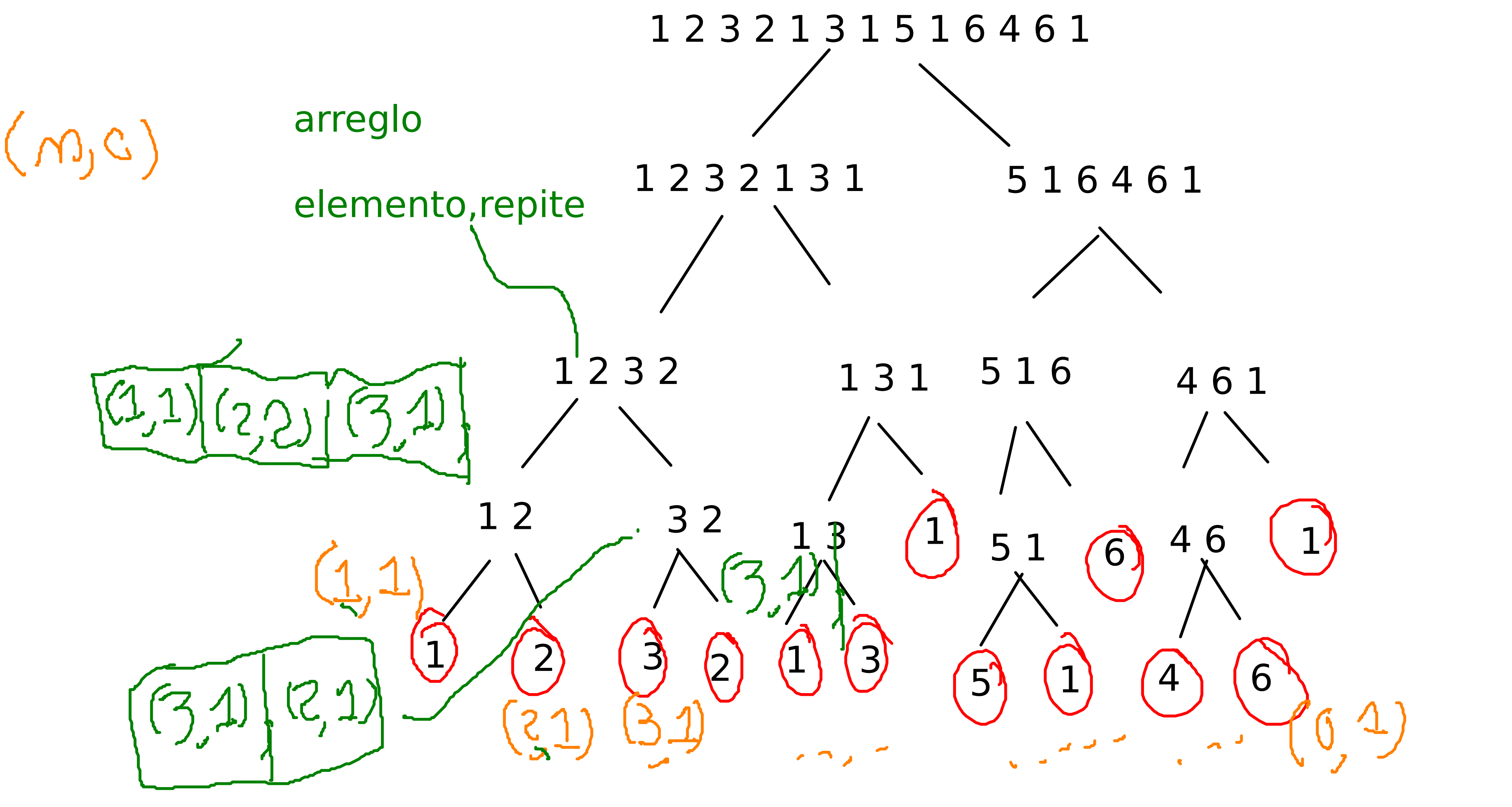
$$T(0) = O(1)$$

$$T(1) = O(1)$$

Encontrar la moda de un vector

$T(n) = 2T(n/2) + O(n)$

EJemplo



Referencias

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2009. Introduction to Algorithms, Third Edition (3rd ed.). The MIT Press. Chapter 4

Gracias

Próximo tema:

Estructuras de datos