Dos temas principales

Algoritmos de ordenación:

- los **algoritmos** son métodos, procedimientos —la especificación de una secuencia ordenada y finita de pasos a seguir— para resolver un problema, lograr un objetivo

Árboles de búsqueda:

- las **estructuras de datos** son formas de organizar los datos en la memoria principal del computador

  … de manera de facilitar el acceso a los datos, su actualización, etc.

En ambos casos, **demostraciones** de propiedades

Algoritmos de ordenación

Simples:

- **selection sort**
- **insertion sort**
- ordenan por la vía de comparar (e intercambiar) sólo elementos adyacentes

Basados en la estrategia algorítmica **dividir para reinar**:

- **merge sort**
- **quick sort**
- comparan (e intercambian) elementos no necesariamente adyacentes

Basado en una **estructura de datos**:

- **heap sort**
- un heap es una estructura de datos

Algoritmos de ordenación simples

selection sort:

- si los primeros *k* datos ya están ordenados,

  … entonces encuentra el menor de los *n* – *k* datos restantes

  … y lo intercambia con el dato en la posición *k*+1 (es *in place*)
- parte con *k* = 0

insertion sort:

- si los primeros *k* datos ya están ordenados entre ellos,

  … entonces toma el dato *r* en la posición *k*+1 y lo coloca ordenadamente en la posición que le corresponda entre los primeros *k*,

  … desplazando un puesto hacia la derecha (o hacia abajo) los datos ya ordenados mayores que *r* (es *in place*)
- parte con *k* = 1

Los tres pasos de la estrategia algorítmica **dividir para reinar**

1. Divide el problema en dos subproblemas del mismo tipo

2. Resuelve **recursivamente** cada subproblema

3. Encuentra la solución al problema original a partir de las soluciones a los dos subproblemas

Dada la naturaleza recursiva de la estrategia, antes de dividir el problema en dos subproblemas,

… se pregunta (paso **0**) si el tamaño del problema permite resolverlo directamente —sin recurrir a la recursión—

… en cuyo caso se lo resuelve directamente

Algoritmos de ordenación basados en dividir para reinar

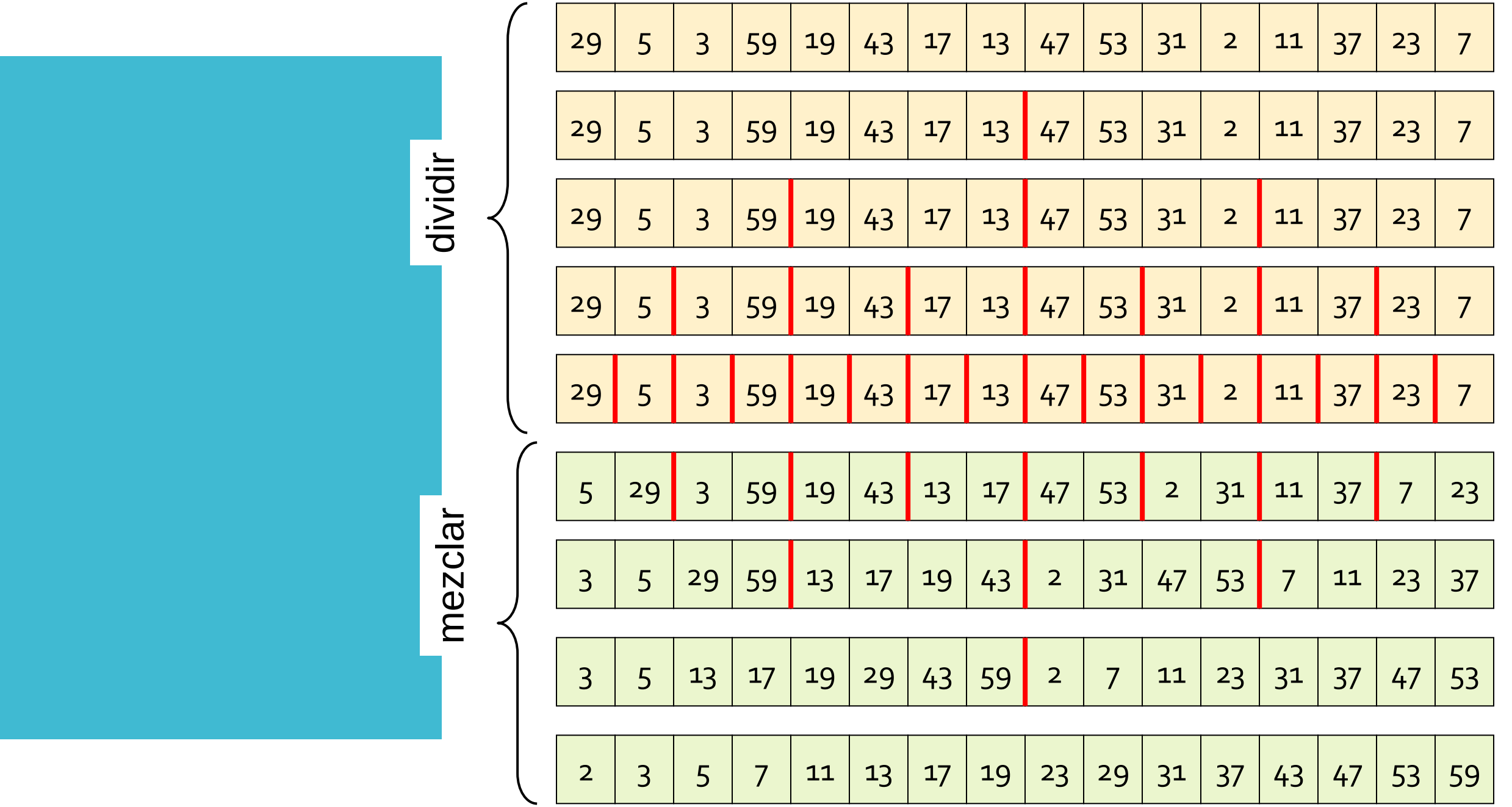
merge sort:

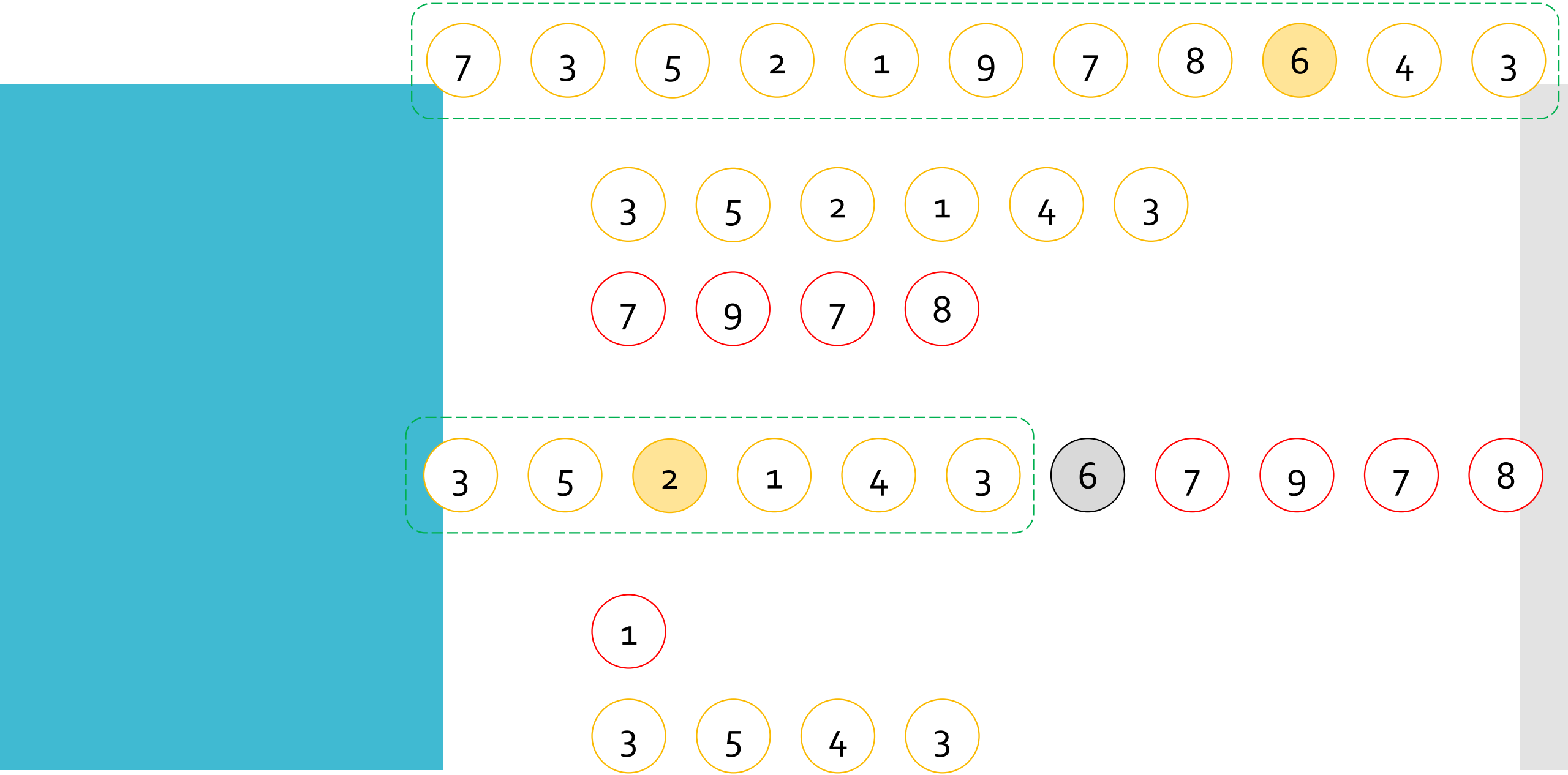
- el paso 1 consiste en dividir el problema trivialmente en (dos) mitades —la izquierda (o de arriba) y la derecha (o de abajo)
- el paso 3 consiste en mezclar las dos mitades —ya ordenadas en el paso 2— en un todo ordenado (no se hace *in place*)

quick sort:

- el paso 1 consiste en elegir un dato como *pivote* y luego redistribuir los demás datos,

  … de manera que los datos menores que el pivote queden a la izquierda del pivote (forman uno de los dos subproblemas) y los datos mayores que el pivote queden a la derecha (forman el otro subproblema)
- el paso 3 no es necesario, debido a la forma en que el problema fue dividido en el paso 1 (es *in place*)


dividir
29 5 3 59 19 43 17 13 47 53 31 2 11 37 23 7
29 5 3 59 19 43 17 13 47 53 31 2 11 37 23 7
29 5 3 59 19 43 17 13 47 53 31 2 11 37 23 7
29 5 3 59 19 43 17 13 47 53 31 2 11 37 23 7
29 5 3 59 19 43 17 13 47 53 31 2 11 37 23 7
mezclar
5 29 3 59 19 43 13 17 47 53 2 31 11 37 7 23
3 5 29 59 13 17 19 43 2 31 47 53 7 11 23 37
3 5 13 17 19 29 43 59 2 7 11 23 31 37 47 53
2 3 5 7 11 13 17 19 23 29 31 37 43 47 53 59


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

Árboles de búsqueda

Binario

- cumple la **propiedad de árbol binario de búsqueda** (ABB)

Binarios **balanceados**:

- **AVL**
- **rojo-negro**
- cumplen la propiedad de ABB

  … más una **propiedad adicional de balance**

No binarios **balanceados**:

- **2-3**
- **2-4** —una versión extendida de los 2-3
- cumplen una versión extendida de la propiedad de ABB

  … más una **propiedad adicional de balance**

Árbol binario de búsqueda

En un **árbol binario**, la información está almacenada en nodos, cada uno de los cuales tiene una conexión directa a cero, uno o dos otros nodos descendientes (hijos):

- si un nodo tiene nodos descendientes, éstos se llaman sus hijos izquierdo y/o derecho

… y es descendiente de un nodo (padre):

- el único nodo que no es descendiente de ningún otro nodo se llama la *raíz* del árbol

En un **árbol binario de búsqueda**, la información en los nodos —las *claves*— cumple la siguiente propiedad:

- la clave en el nodo padre es mayor que cualquier clave en los nodos descendientes por el lado izquierdo
- … y menor que cualquier clave en los nodos descendientes por el lado derecho