Heapsort

Estructura de Datos

Heapsort

- Este algoritmo consiste en almacenar todos los elementos del vector a ordenar en un montículo (heap), y luego extraer el nodo que queda como nodo raíz del montículo (cima) en sucesivas iteraciones obteniendo el conjunto ordenado.

- Basa su funcionamiento en una propiedad de los montículos, por la cual, la cima contiene siempre el menor elemento (o el mayor, según se haya definido el montículo) de todos los almacenados en él.

- El algoritmo, después de cada extracción, recoloca en el nodo raíz o cima, la última hoja por la derecha del último nivel.

- Lo cual destruye la propiedad heap del árbol.

- Pero, a continuación realiza un proceso de "descenso" del número insertado de forma que se elige a cada movimiento el mayor de sus dos hijos, con el que se intercambia. Este intercambio, realizado sucesivamente "hunde" el nodo en el árbol restaurando la propiedad montículo del árbol y dejando paso a la siguiente extracción del nodo raíz.

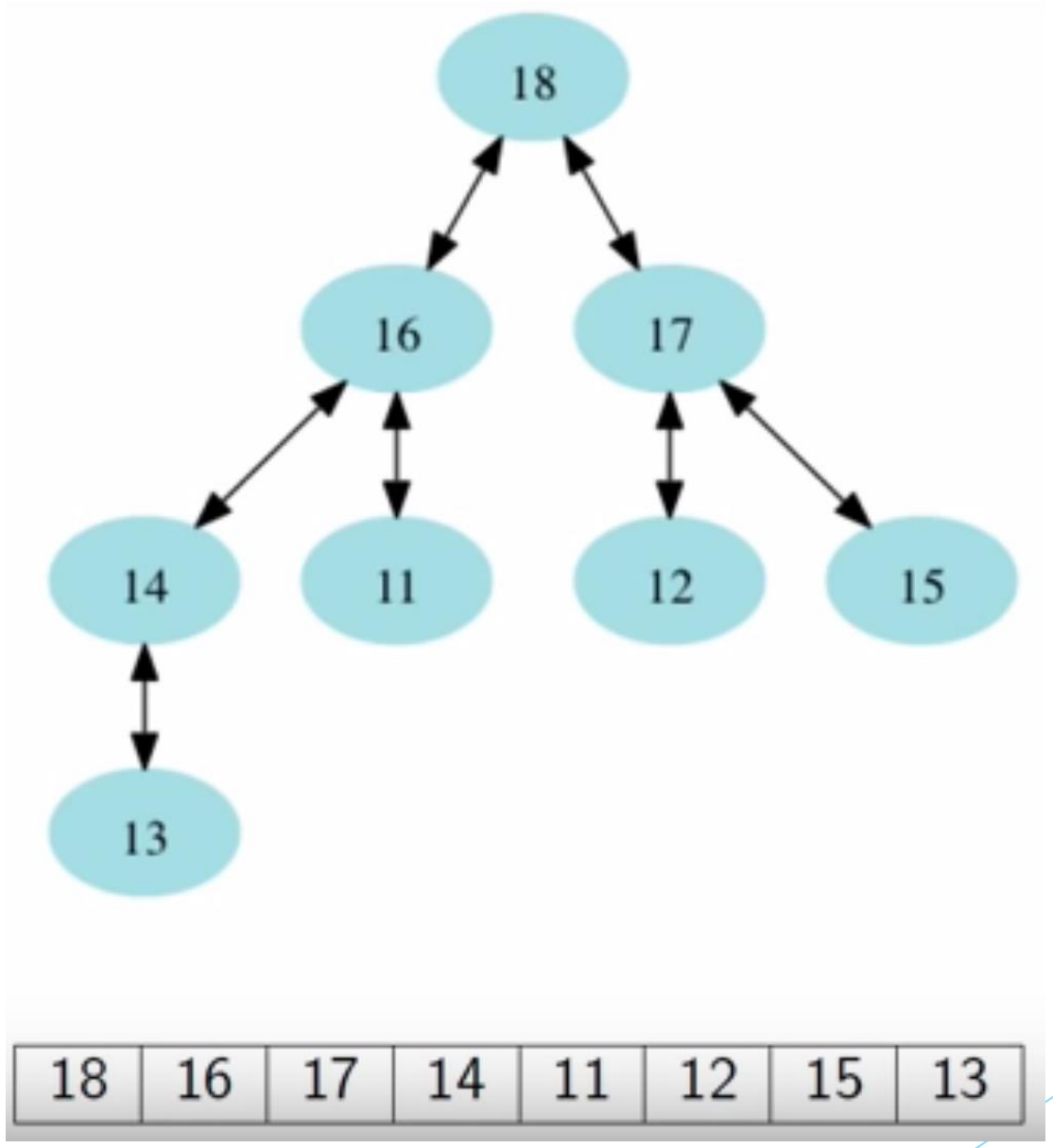
Ventajas y Desventajas Heapsort

Ventajas	Desventajas
• La principal ventaja es que éste método funciona más efectivamente con datos desordenados. • Su desempeño es en promedio tan bueno como el Quicksort y se comporta mejor que éste último en el peor de los casos • No utiliza memoria adicional	• No es estable, ya que se comporta de manera ineficáz con datos del mismo valor. • Método más

Algoritmo Lógico

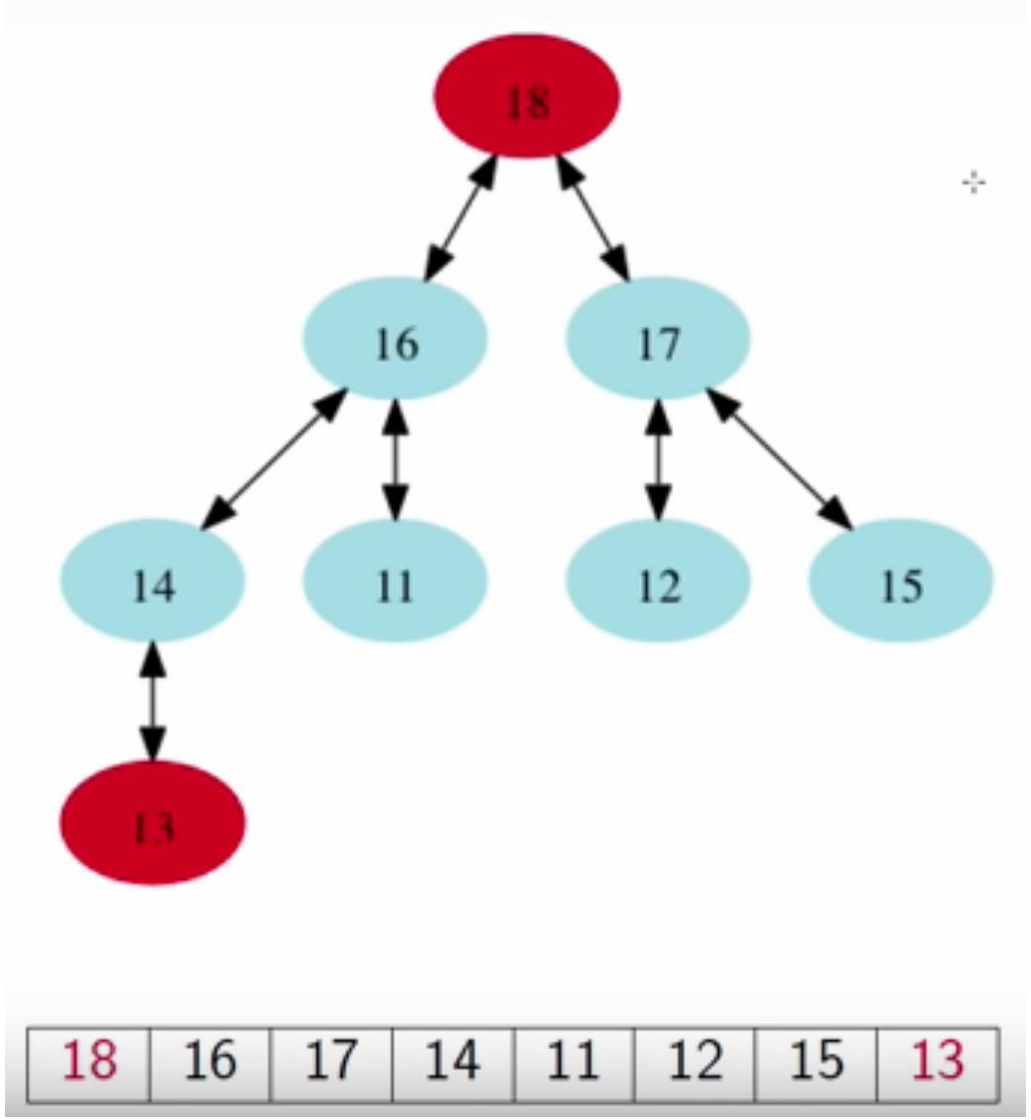
- Se construye el heap inicial a partir del arreglo original
- Se intercambia la raíz con el último elemento del montículo.
- El último elemento queda ordenado.
- El último elemento se saca del heap, no del arreglo.
- Se restaura el heap haciendo que el primer elemento baje a la posición que le corresponde, si sus hijos son menores.
- La raíz vuelve a ser el mayor heap.
- Se repite el paso 2 hasta que quede un solo elemento en el heap.

Ejemplo

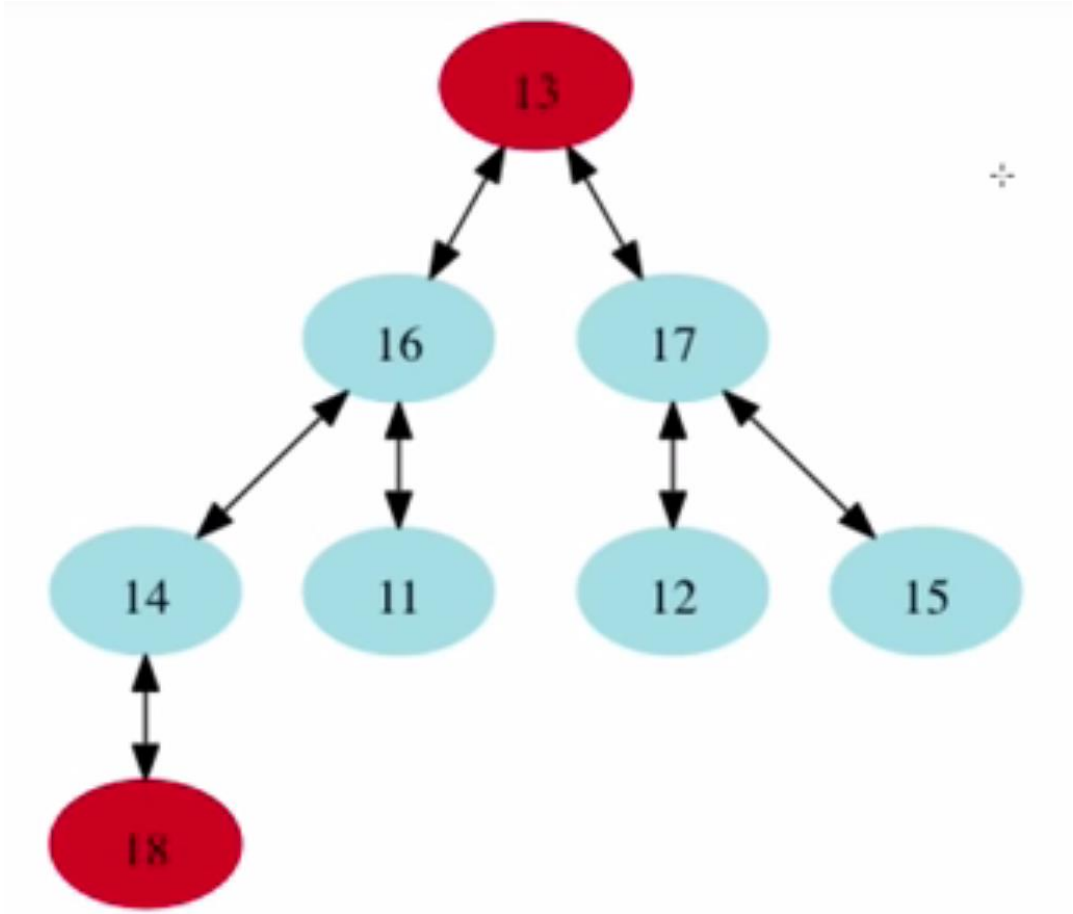


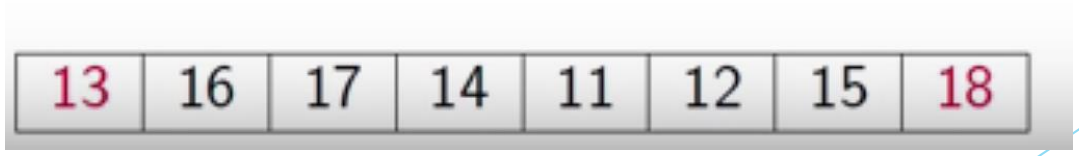
Ejemplo

Borrar 18

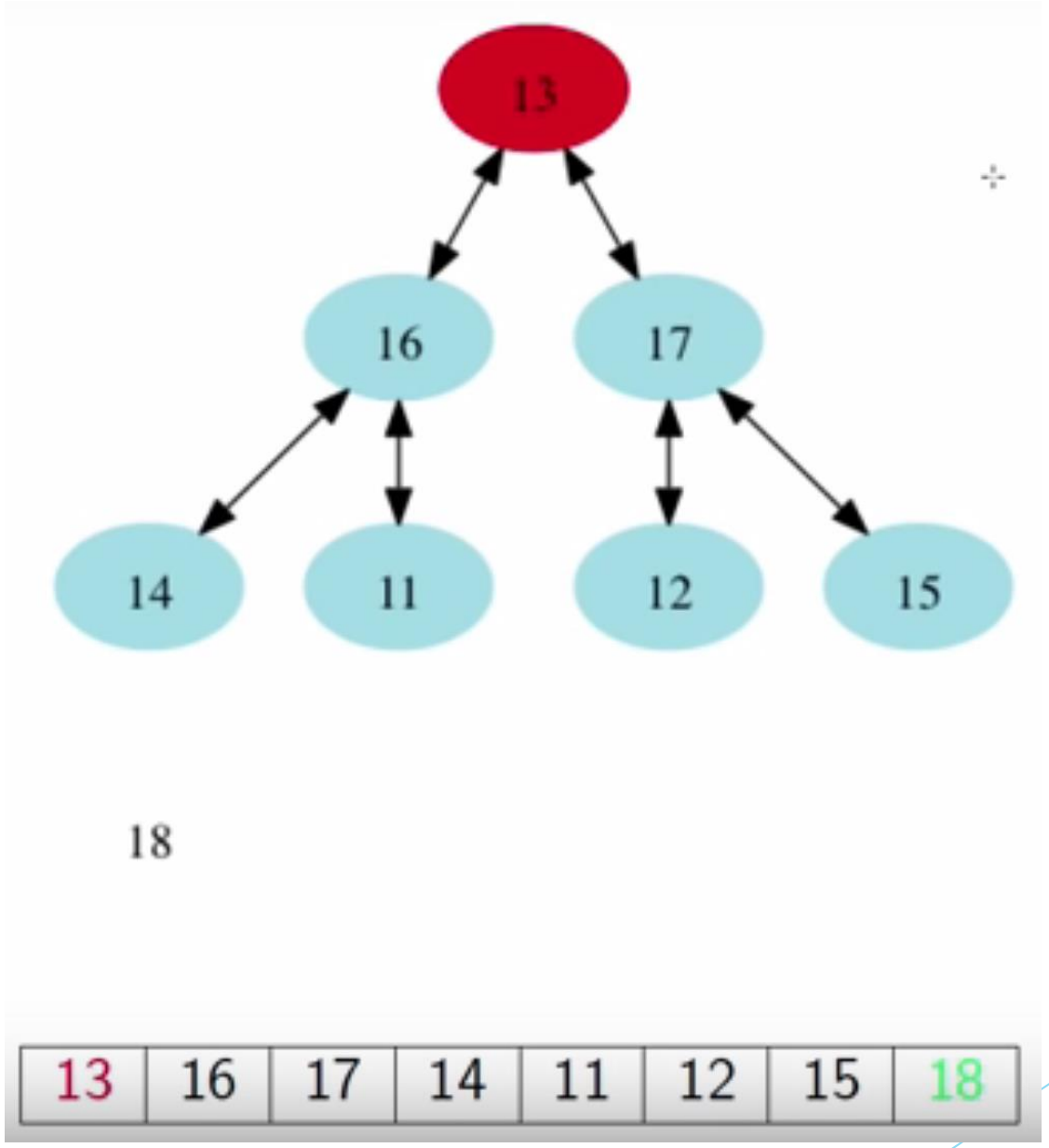


Ejemplo

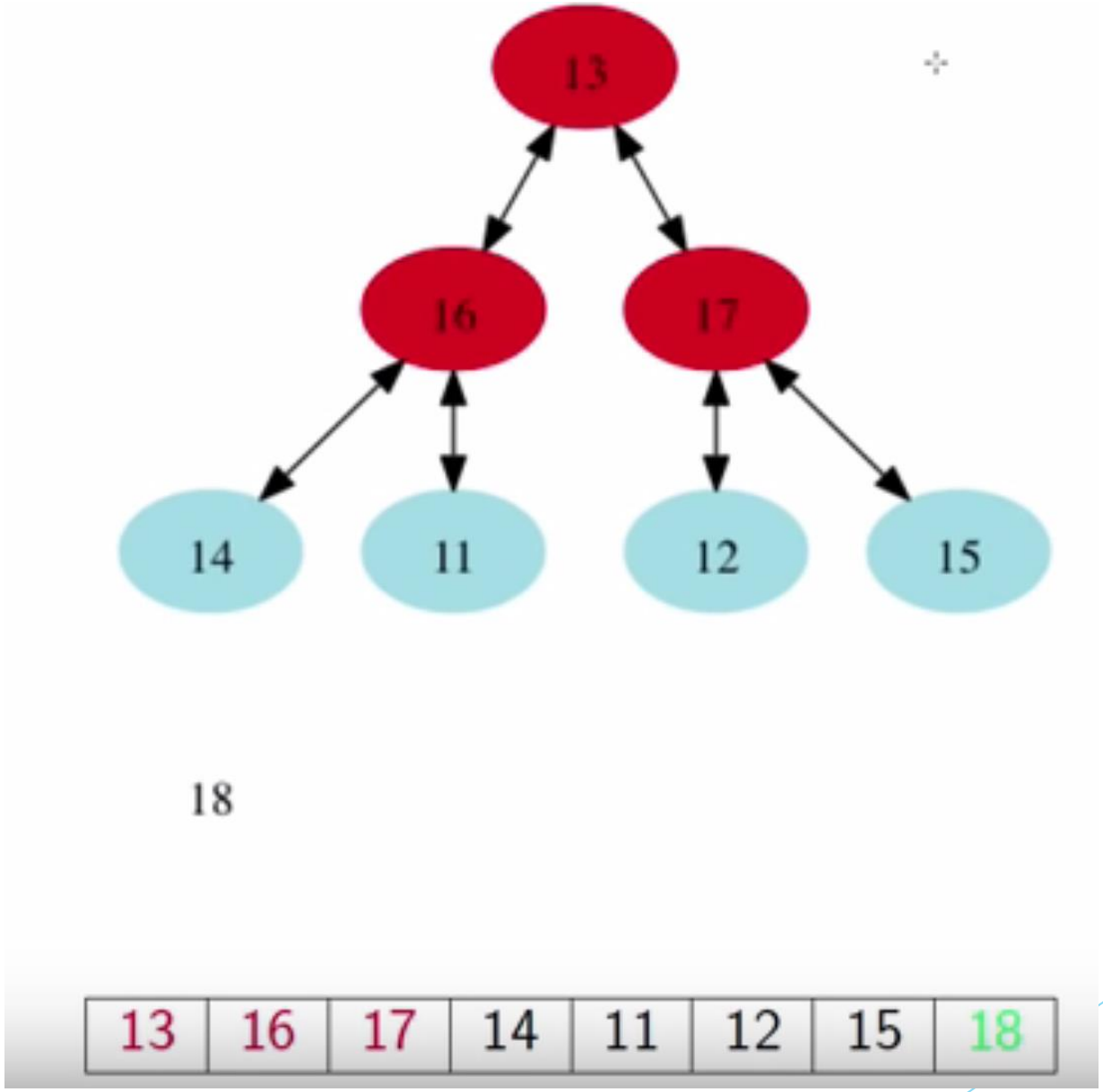




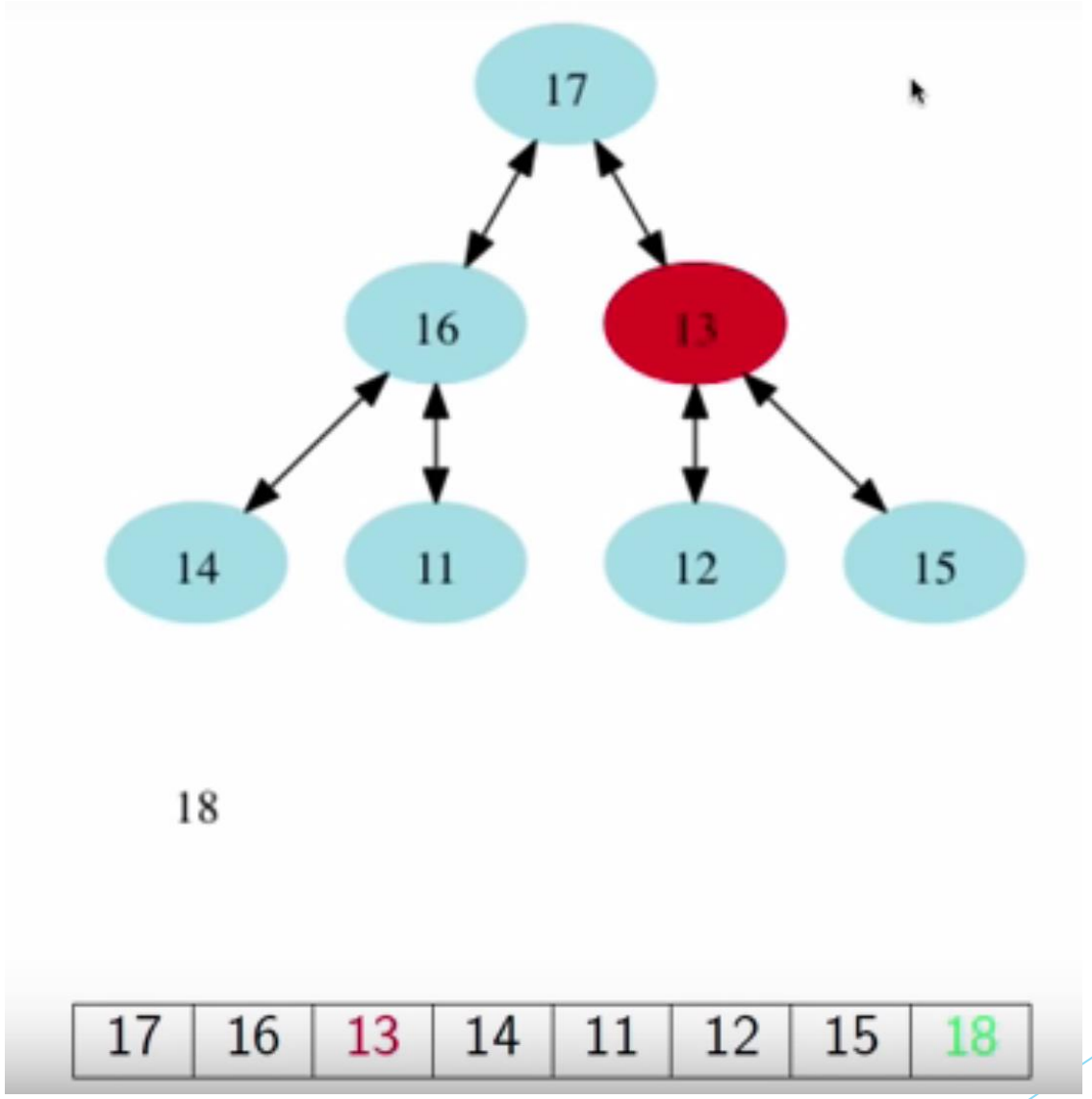
Ejemplo



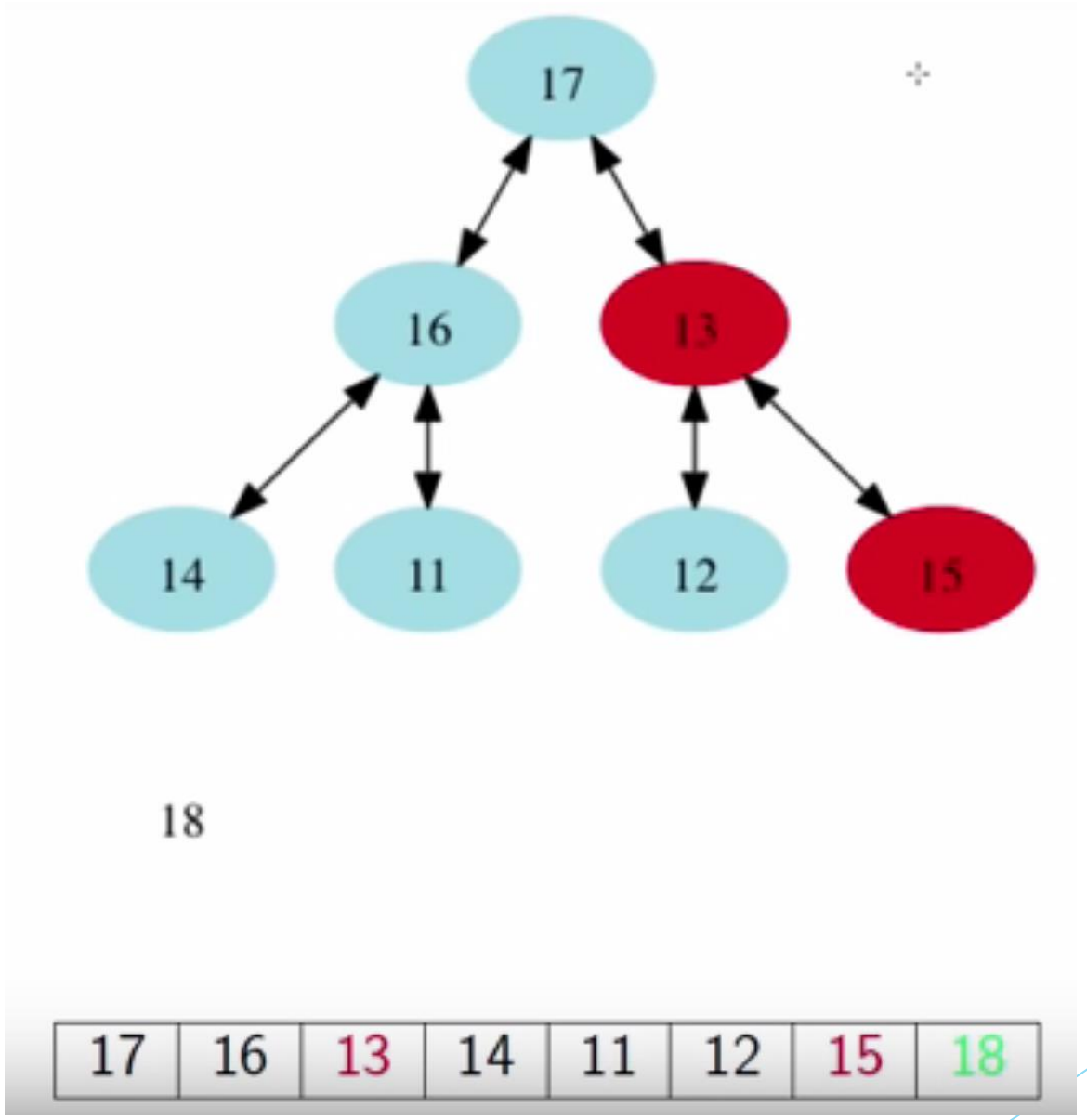
Ejemplo



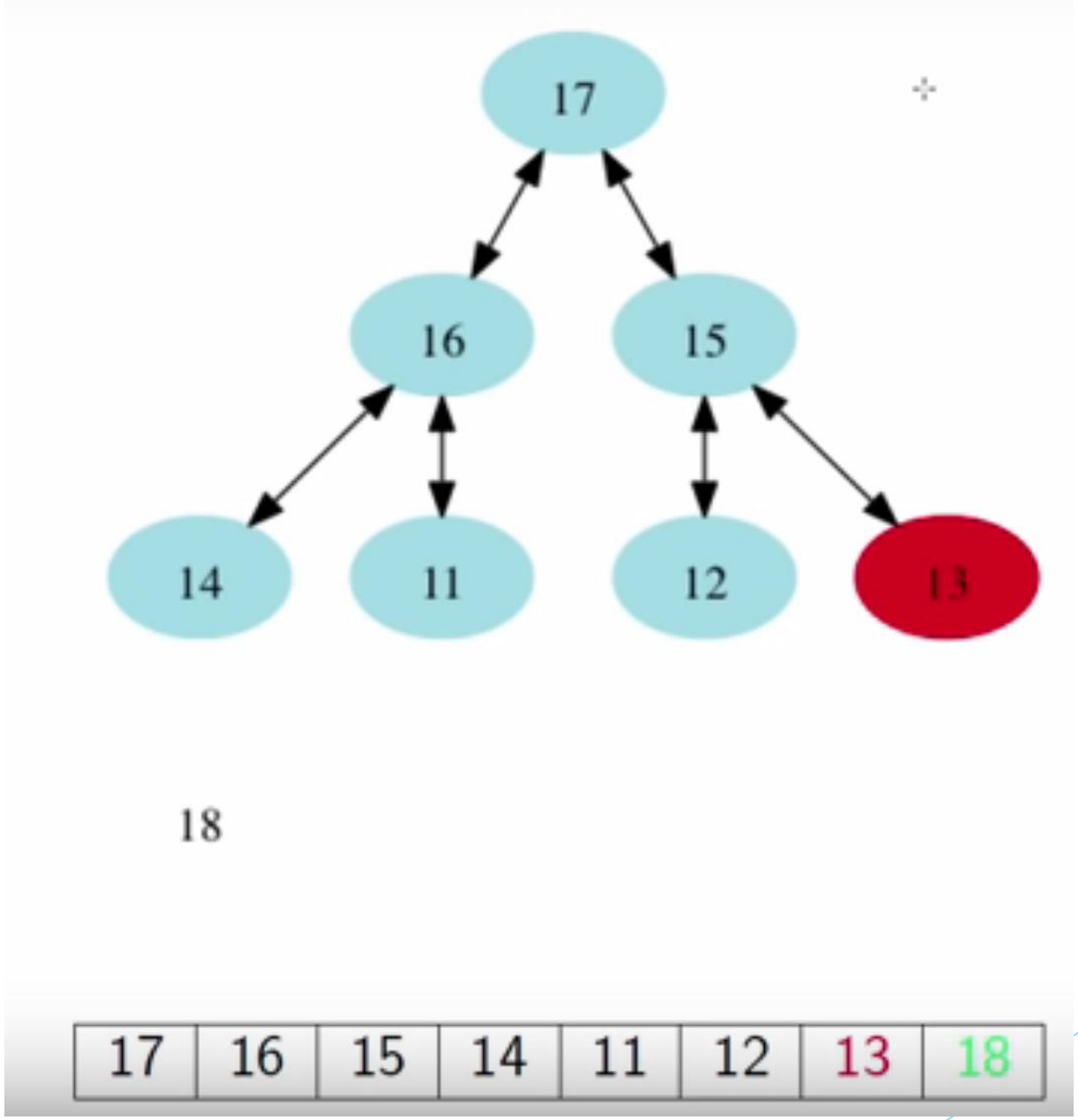
Ejemplo



Ejemplo

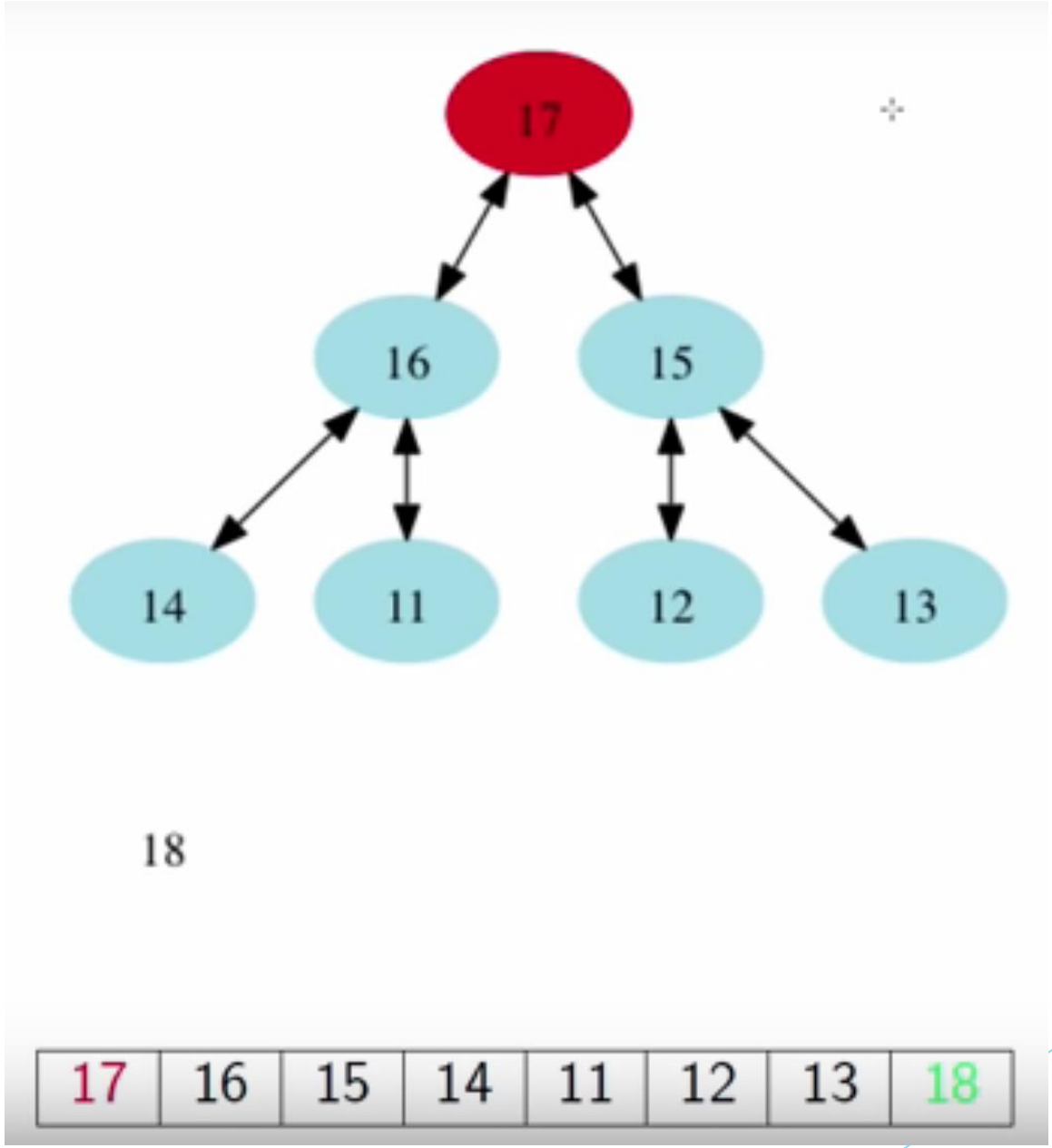


Ejemplo

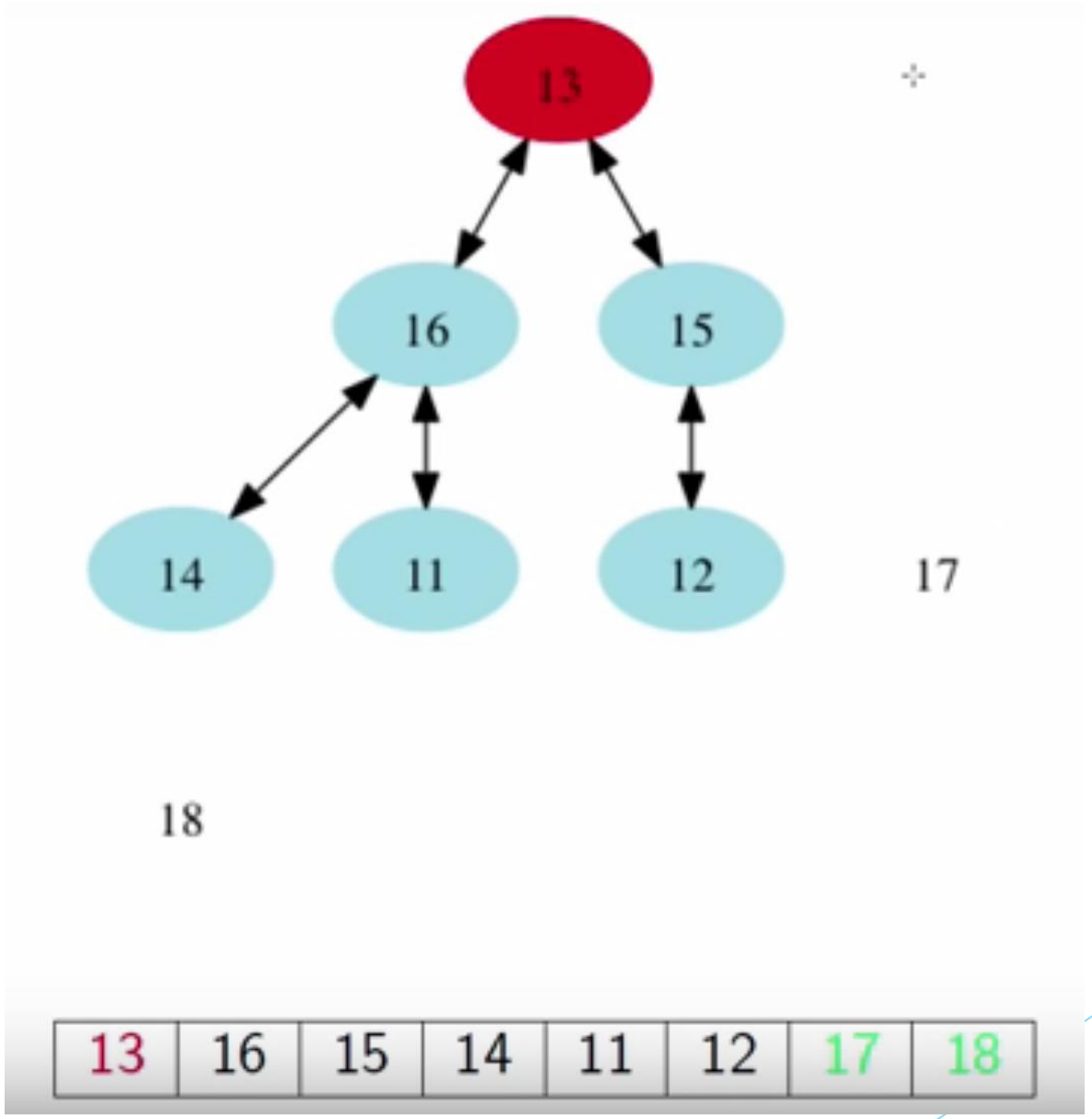


Ejemplo

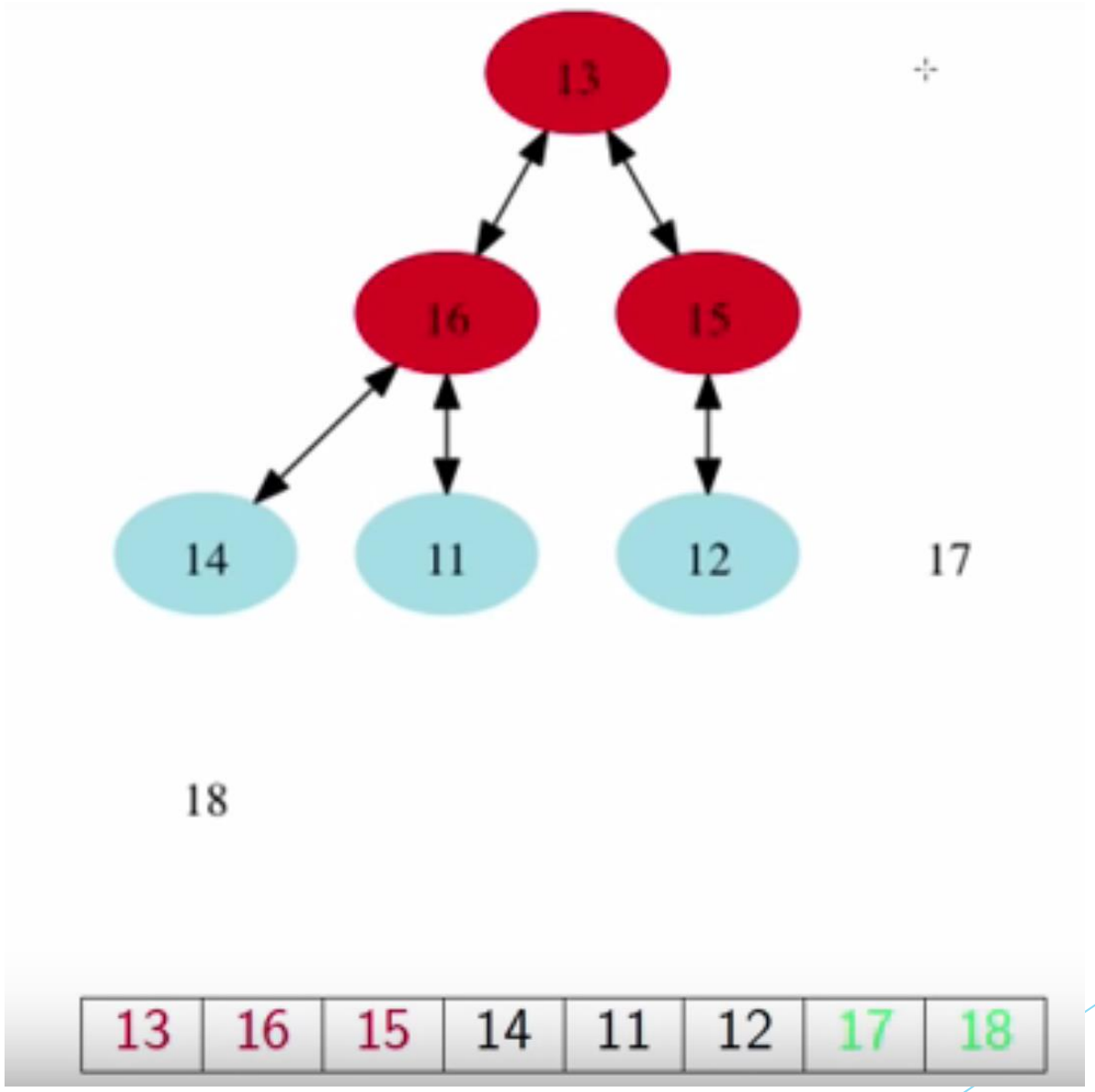
Borrar 17



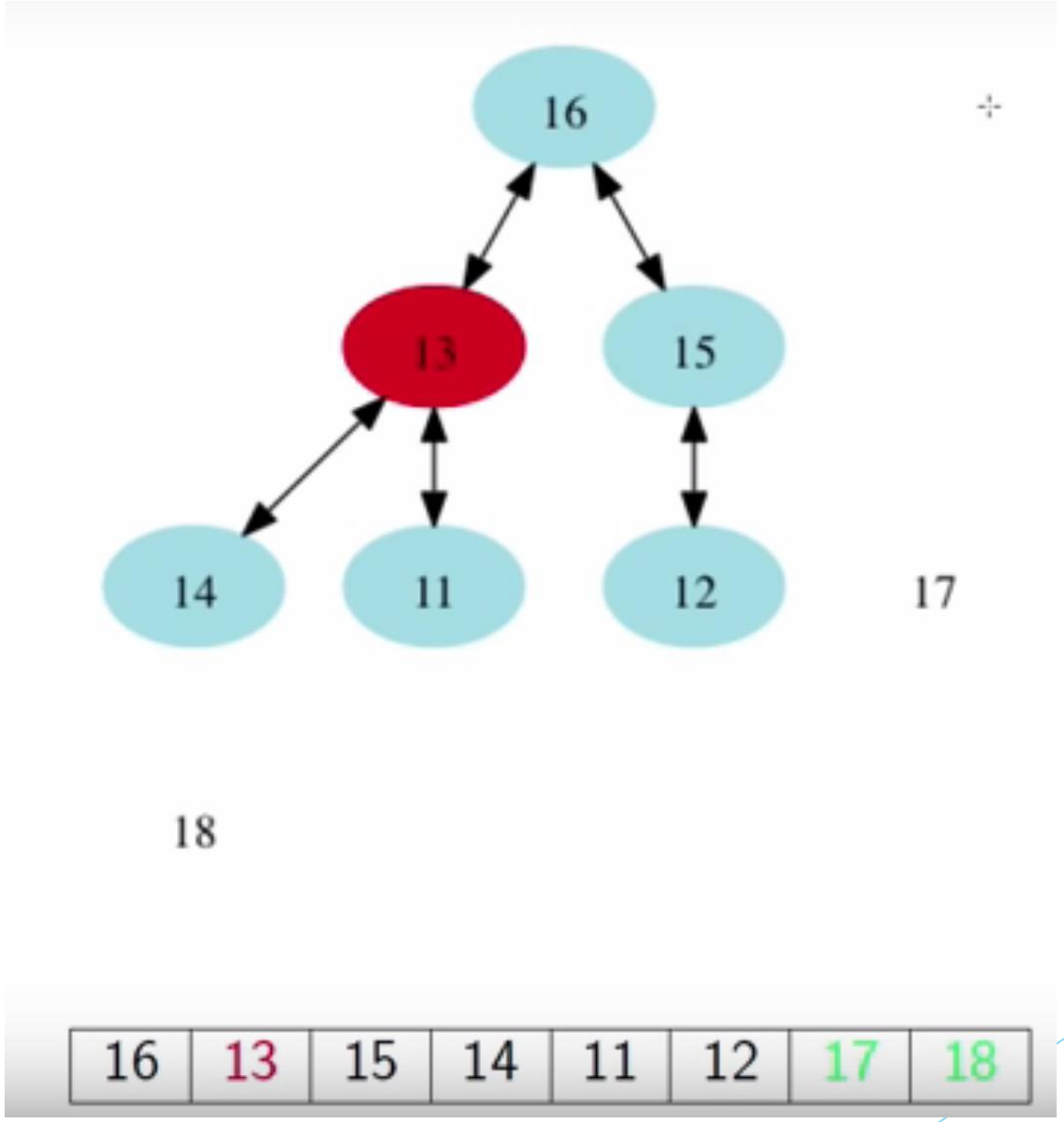
Ejemplo



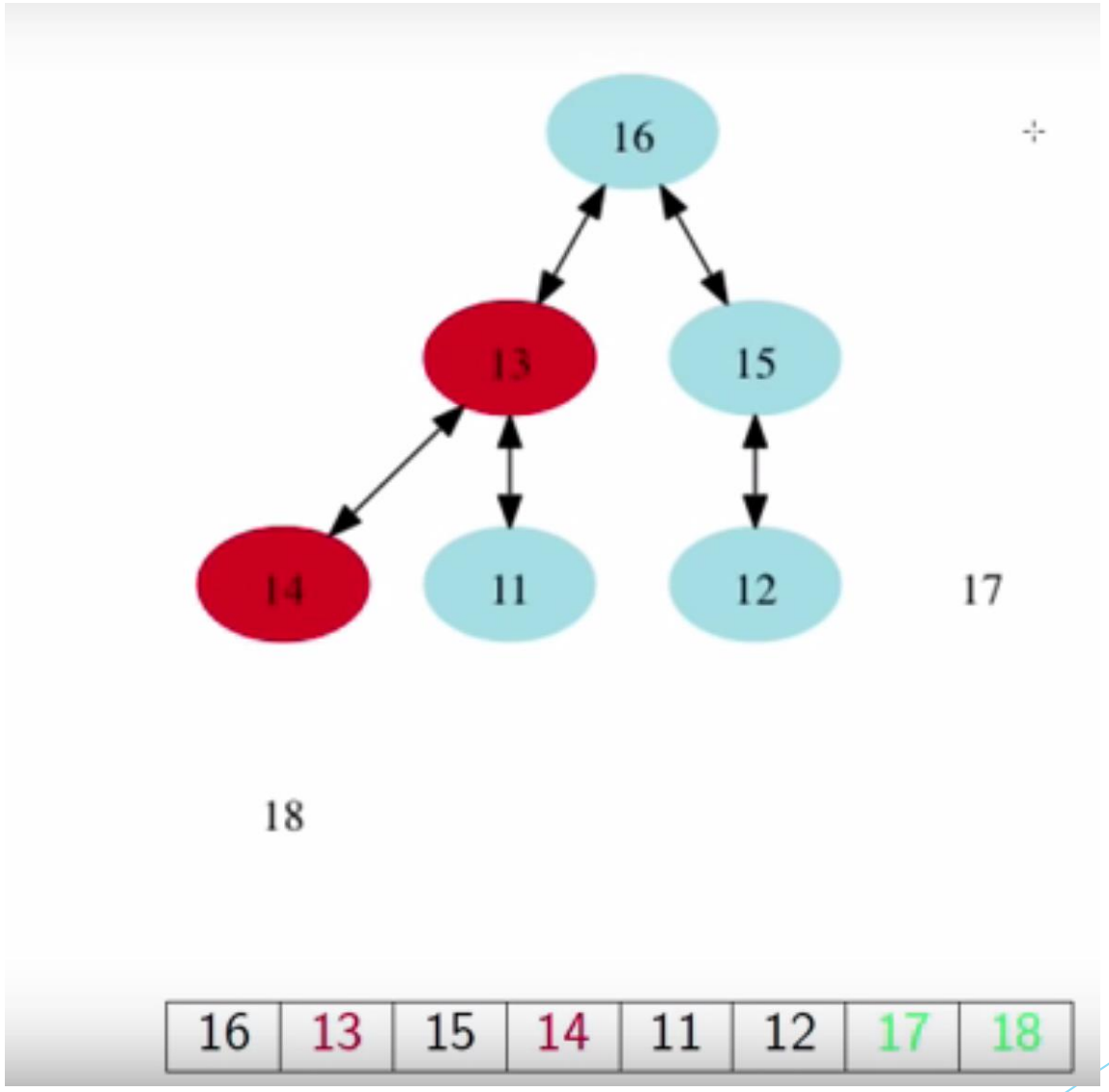
Ejemplo



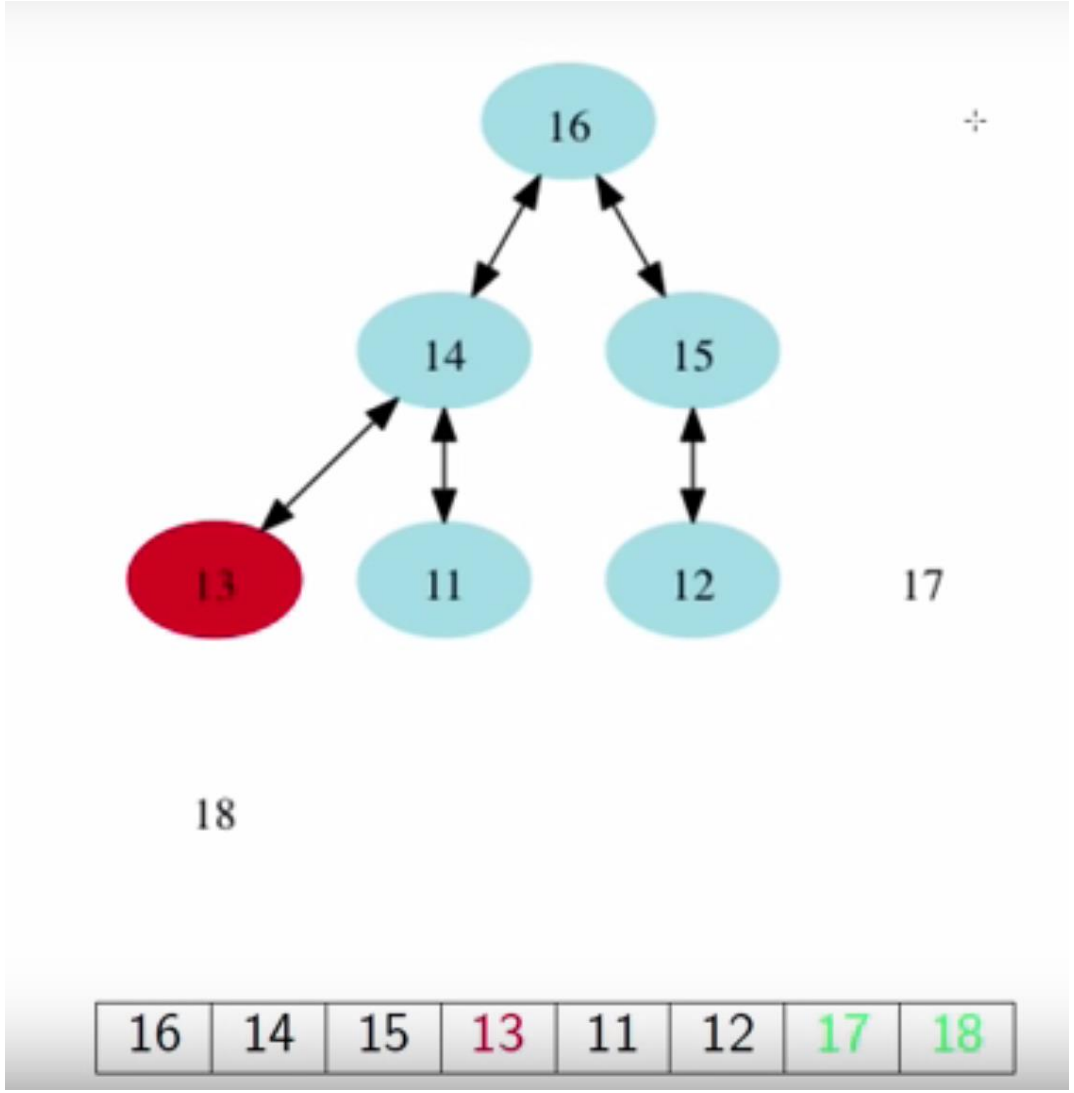
Ejemplo



Ejemplo

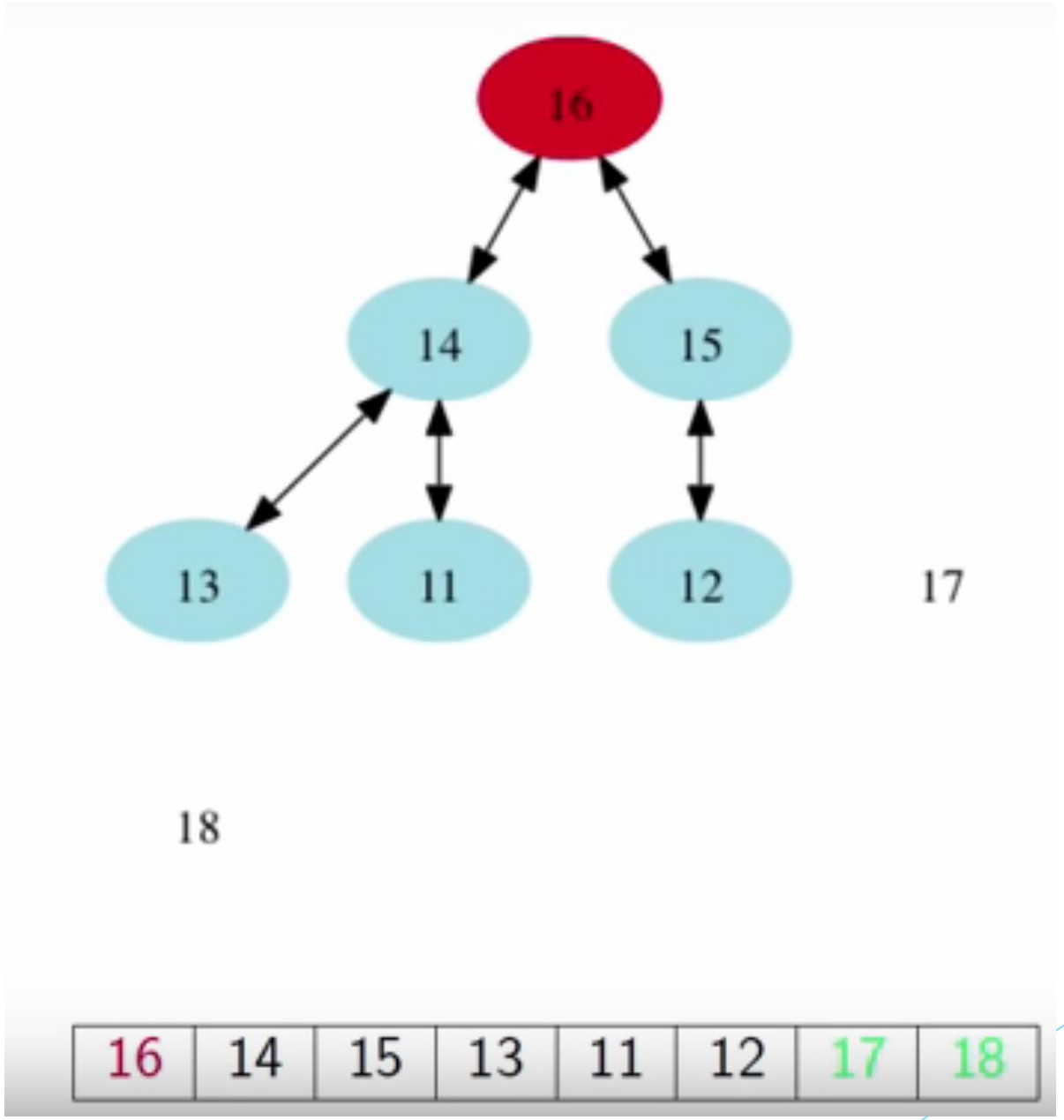


Ejemplo

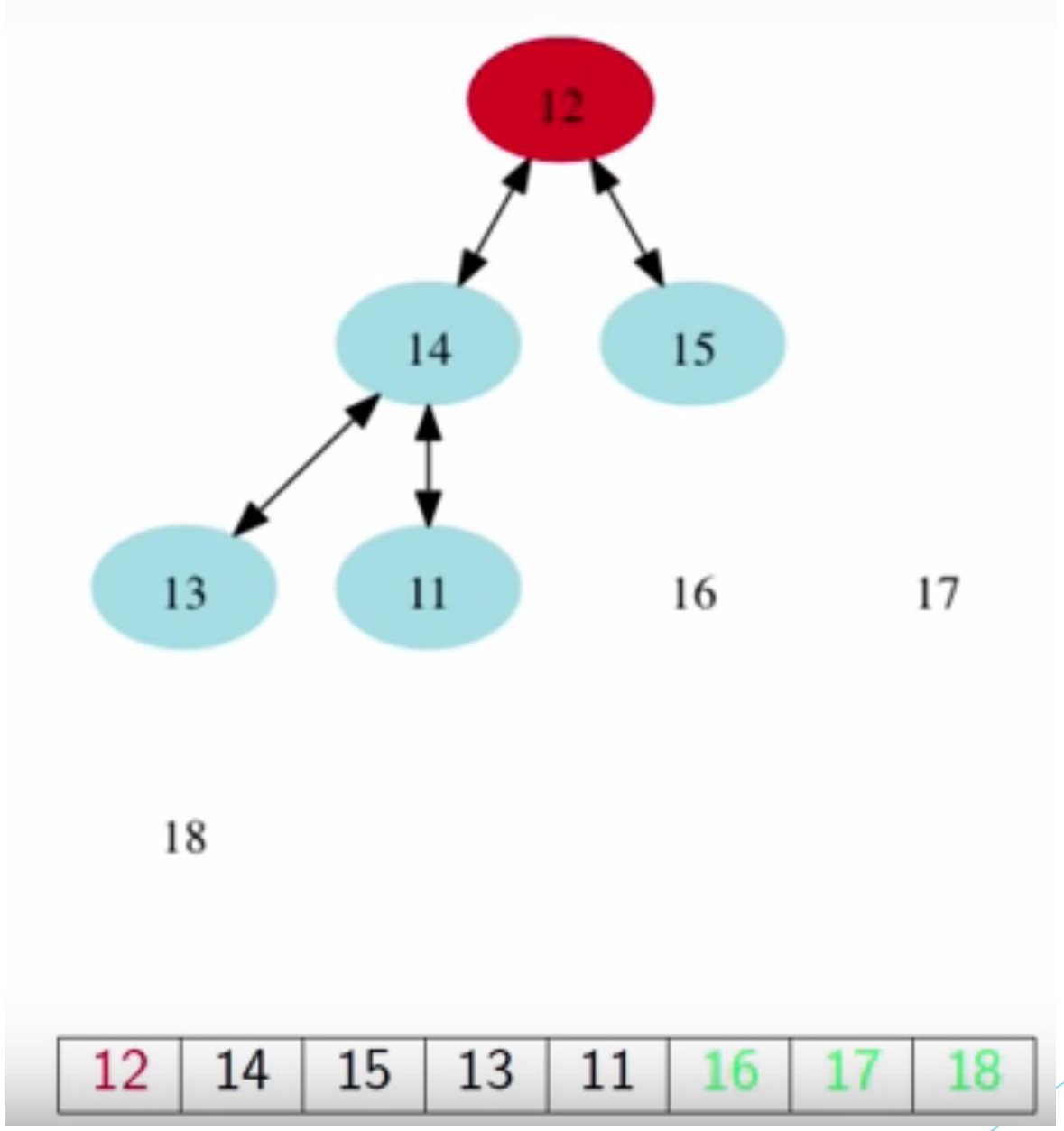


Ejemplo

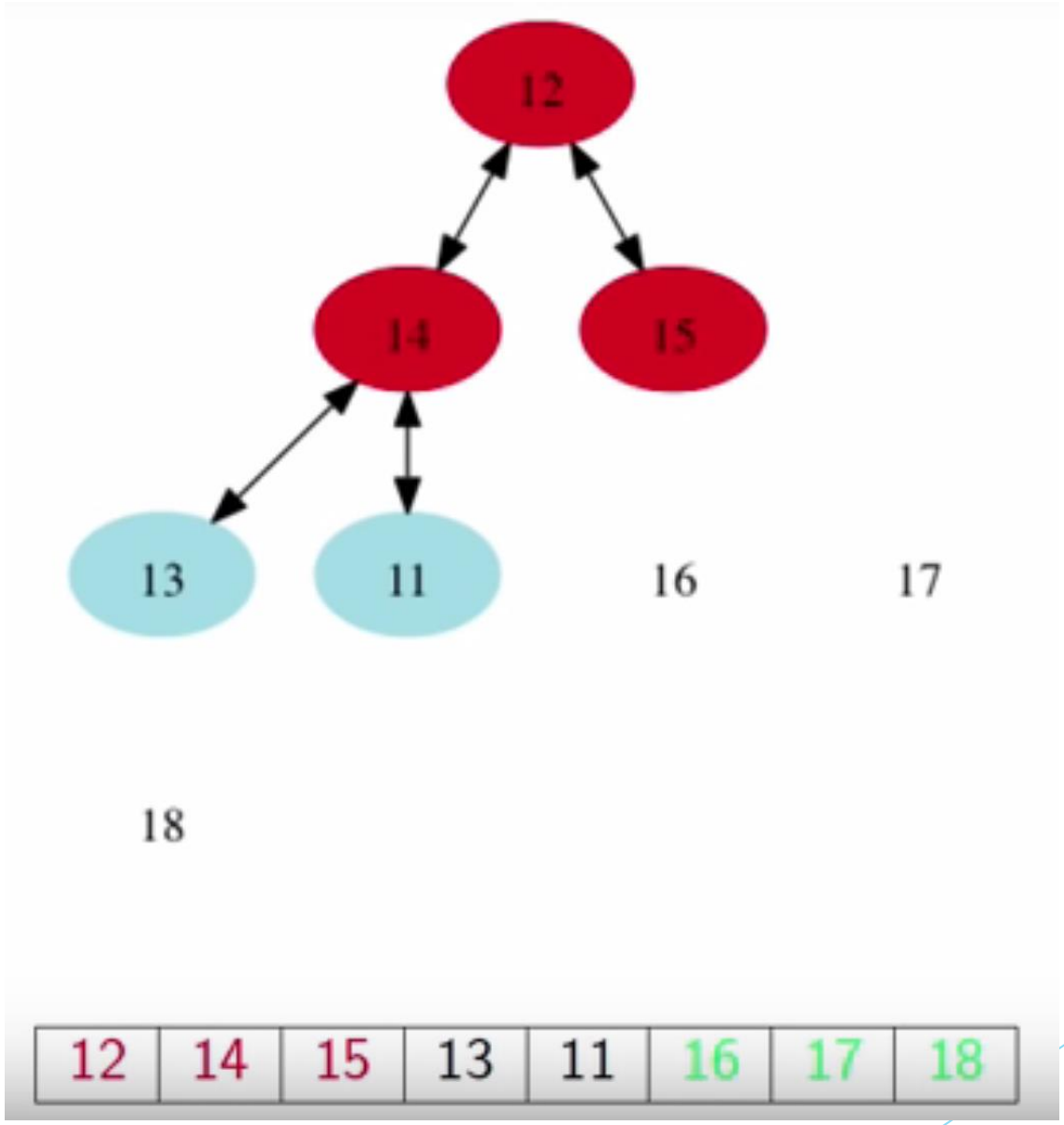
Borrar 16



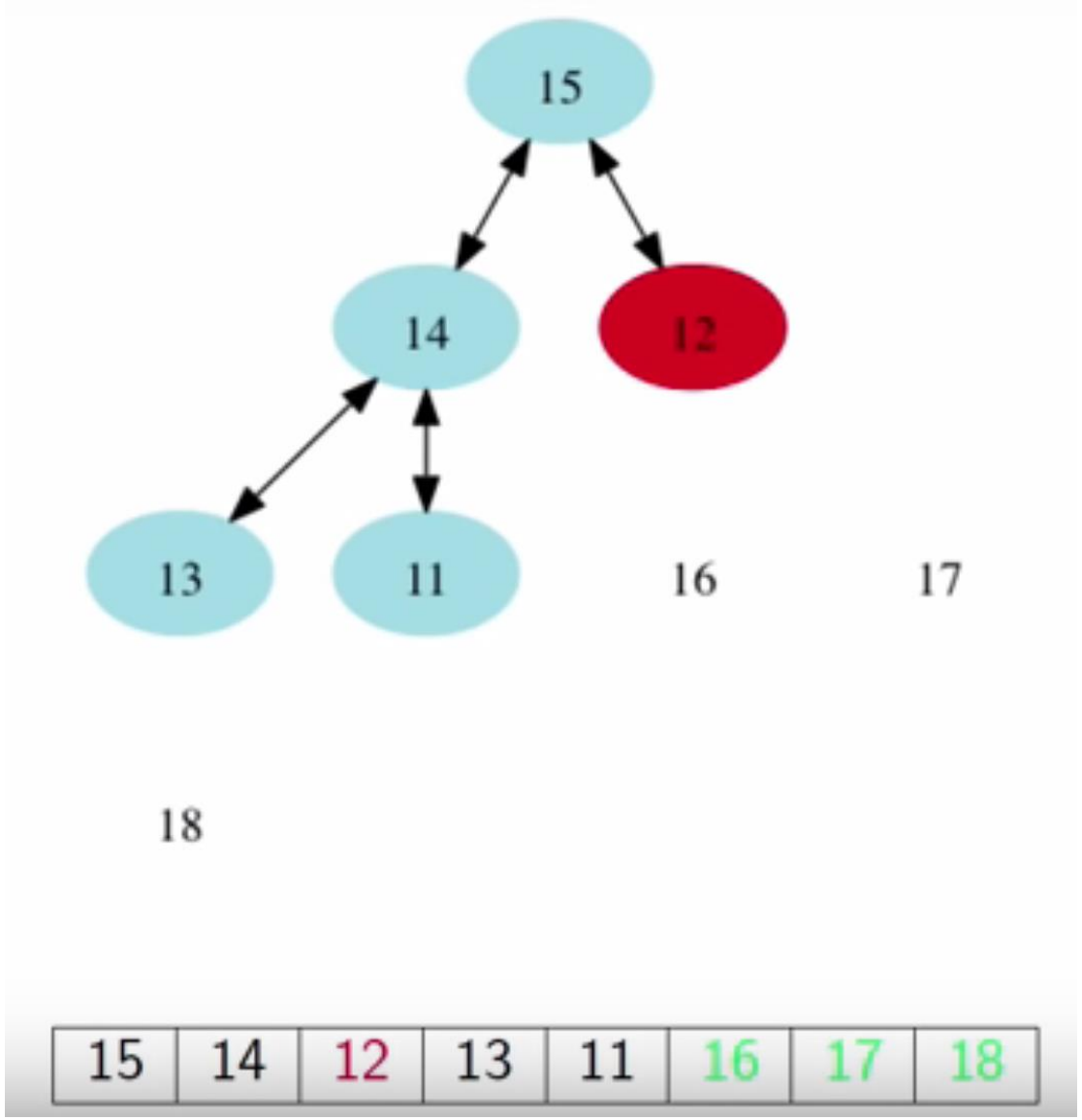
Ejemplo



Ejemplo

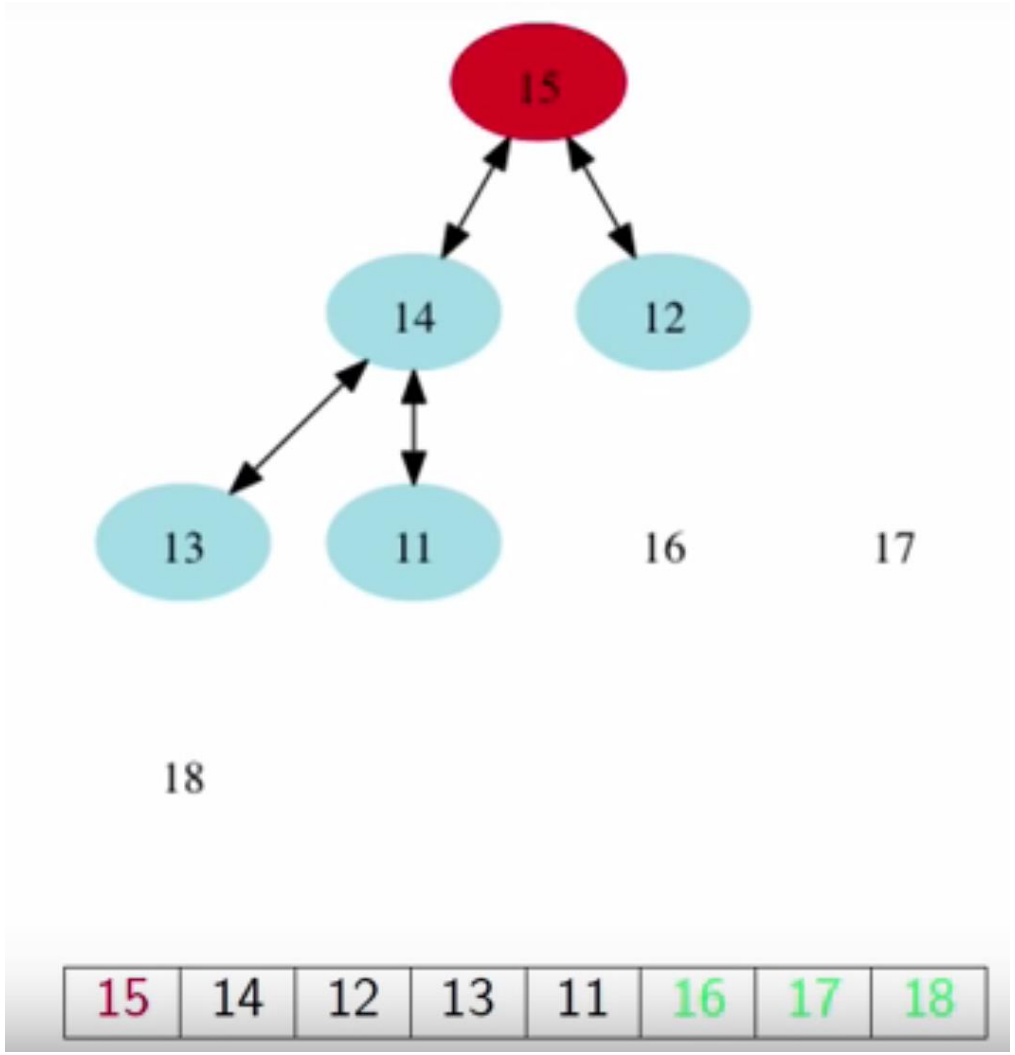


Ejemplo

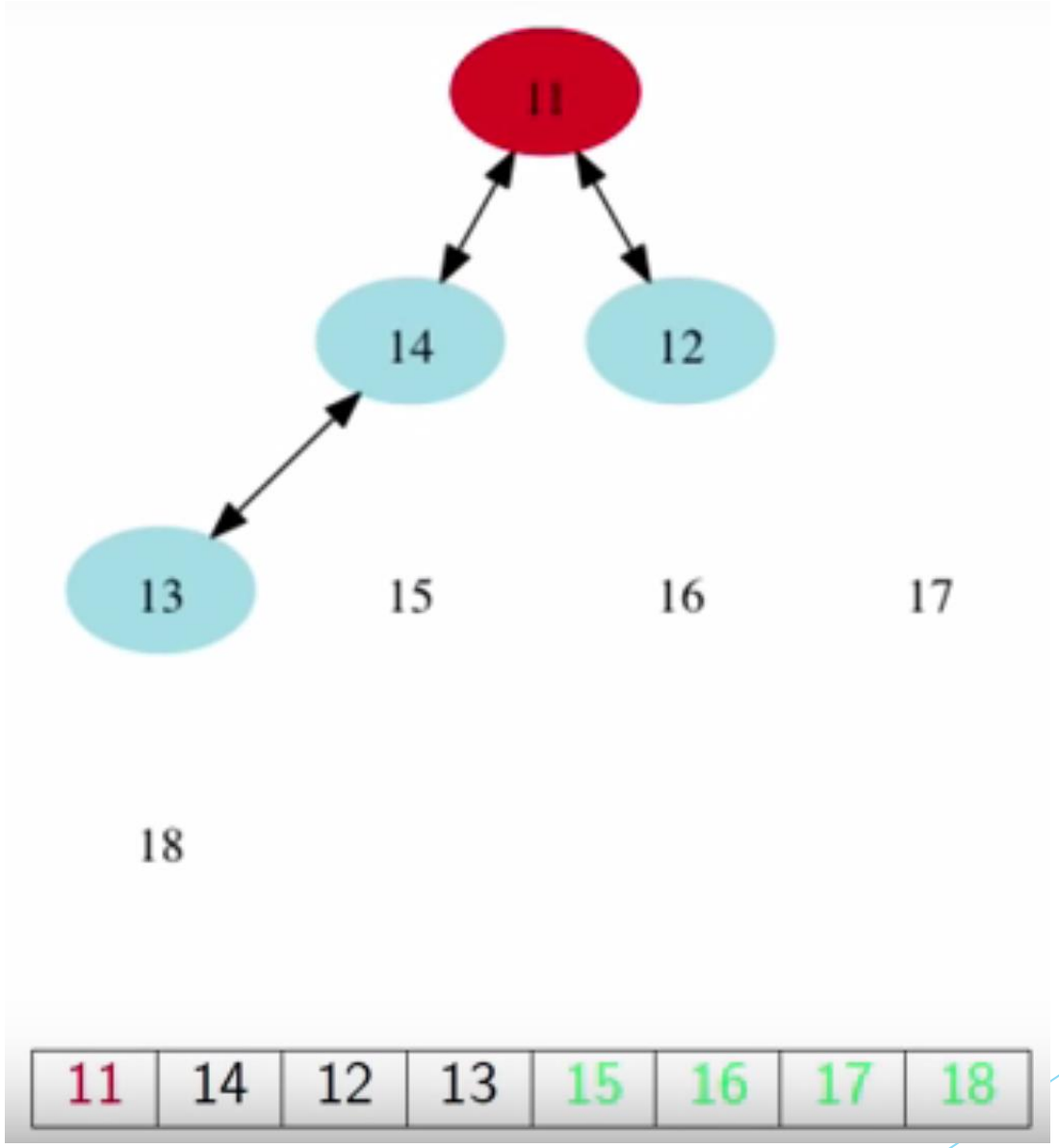


Ejemplo

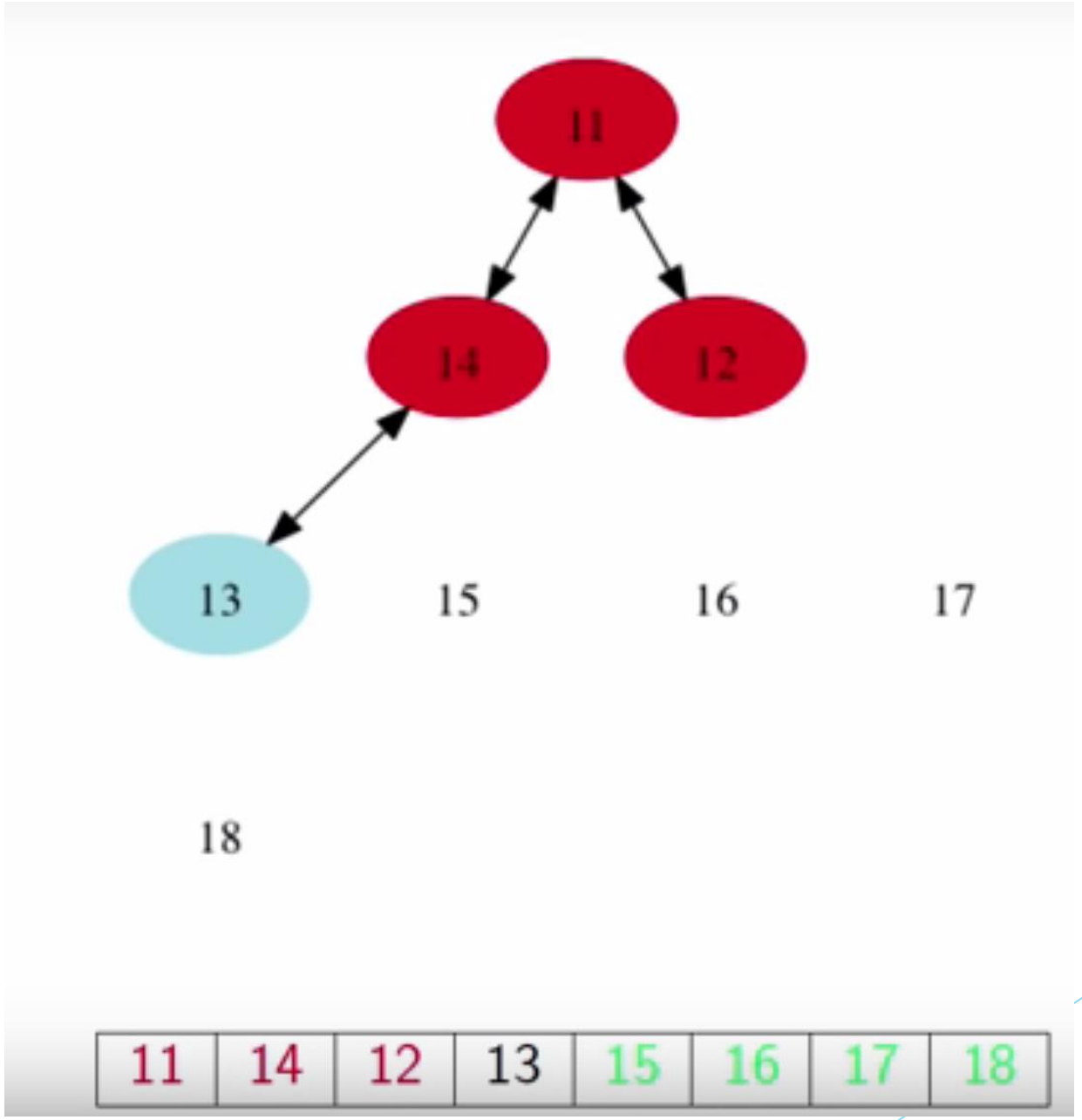
Borrar 15



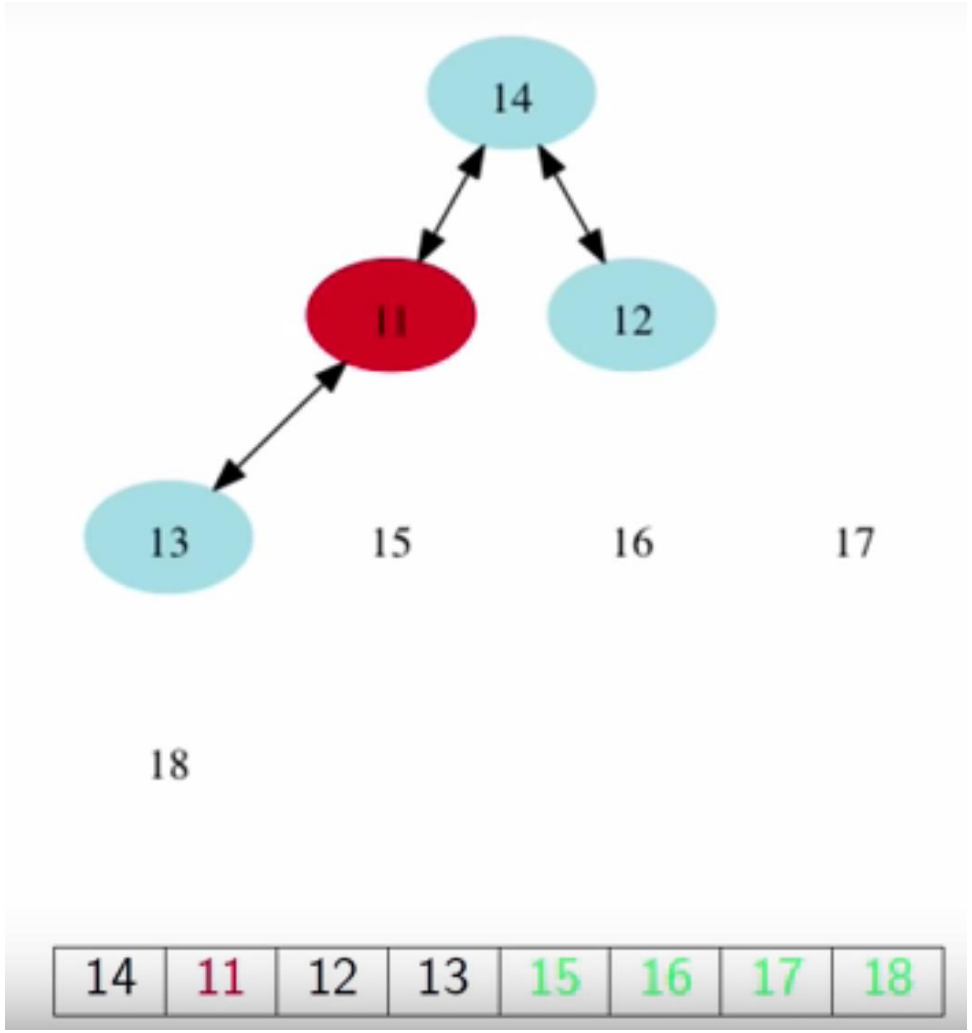
Ejemplo



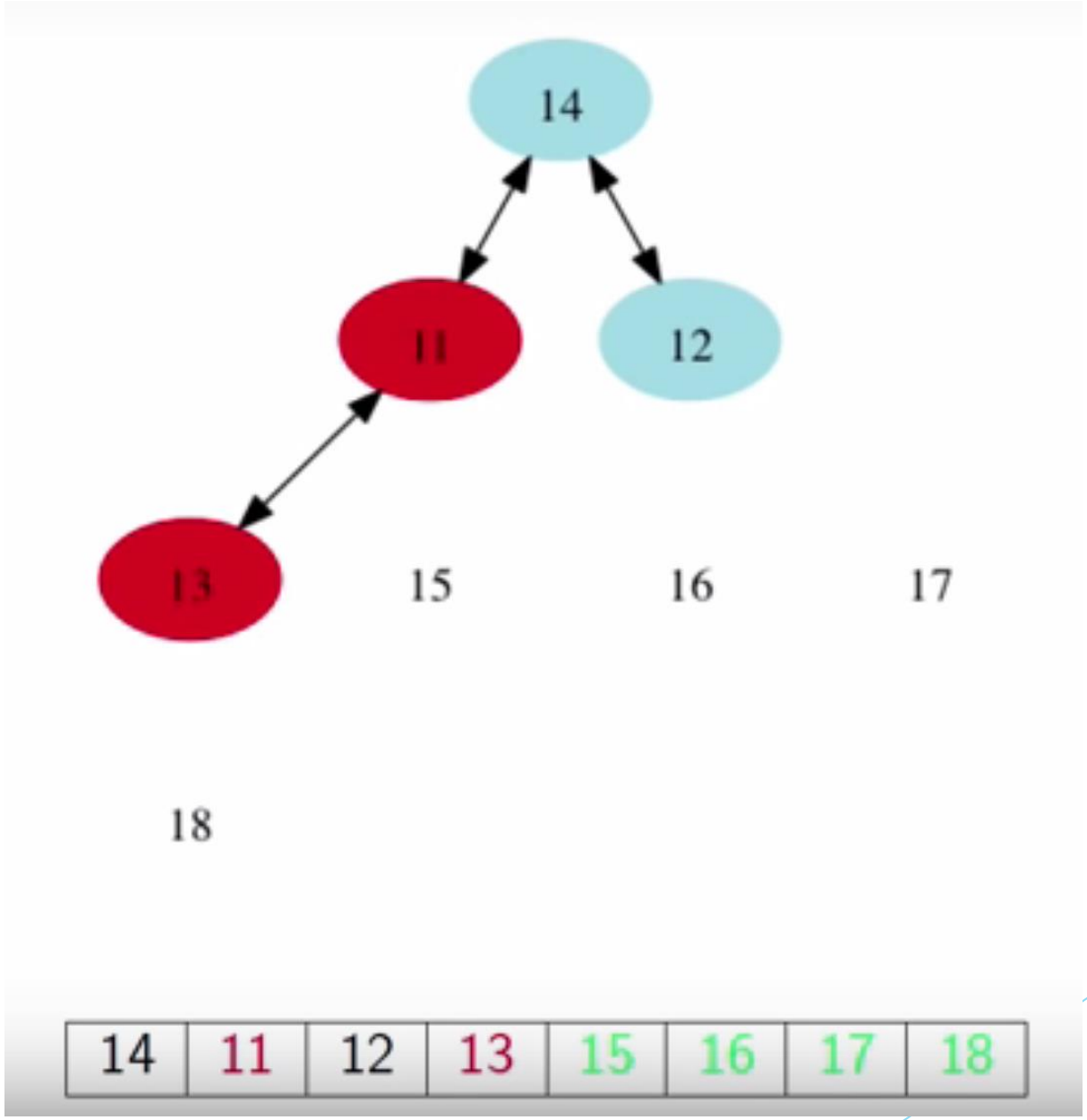
Ejemplo



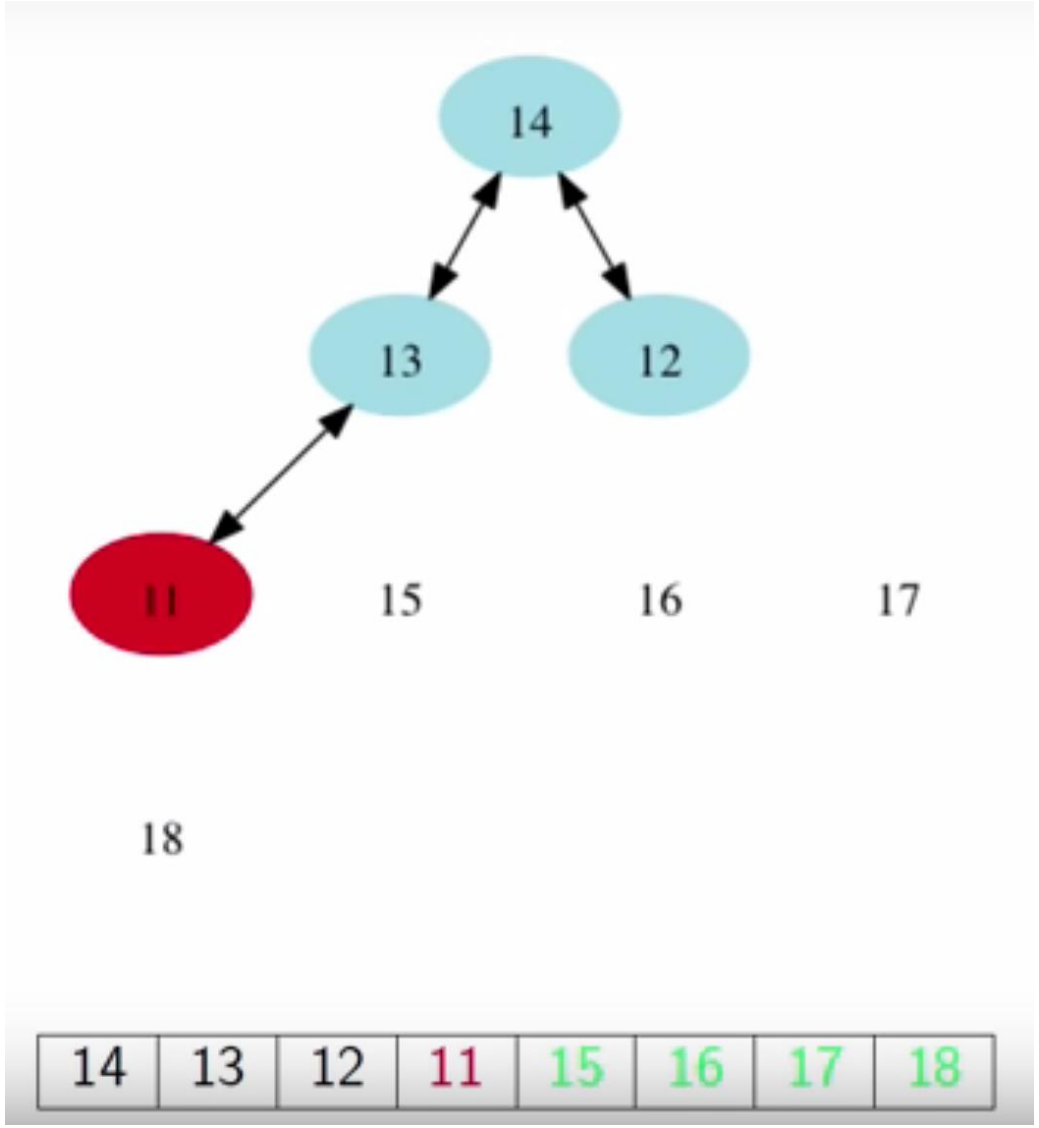
Ejemplo



Ejemplo

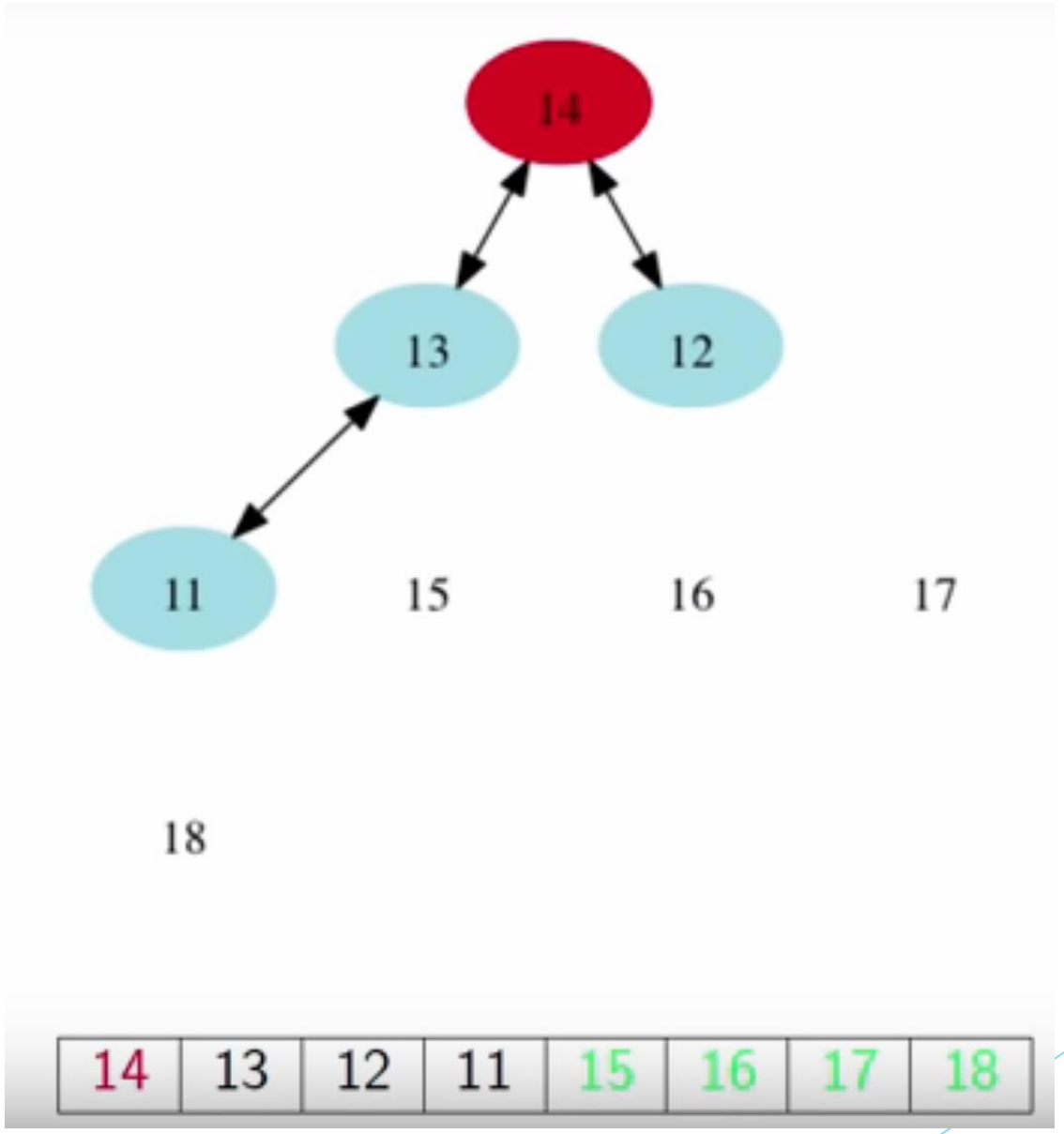


Ejemplo

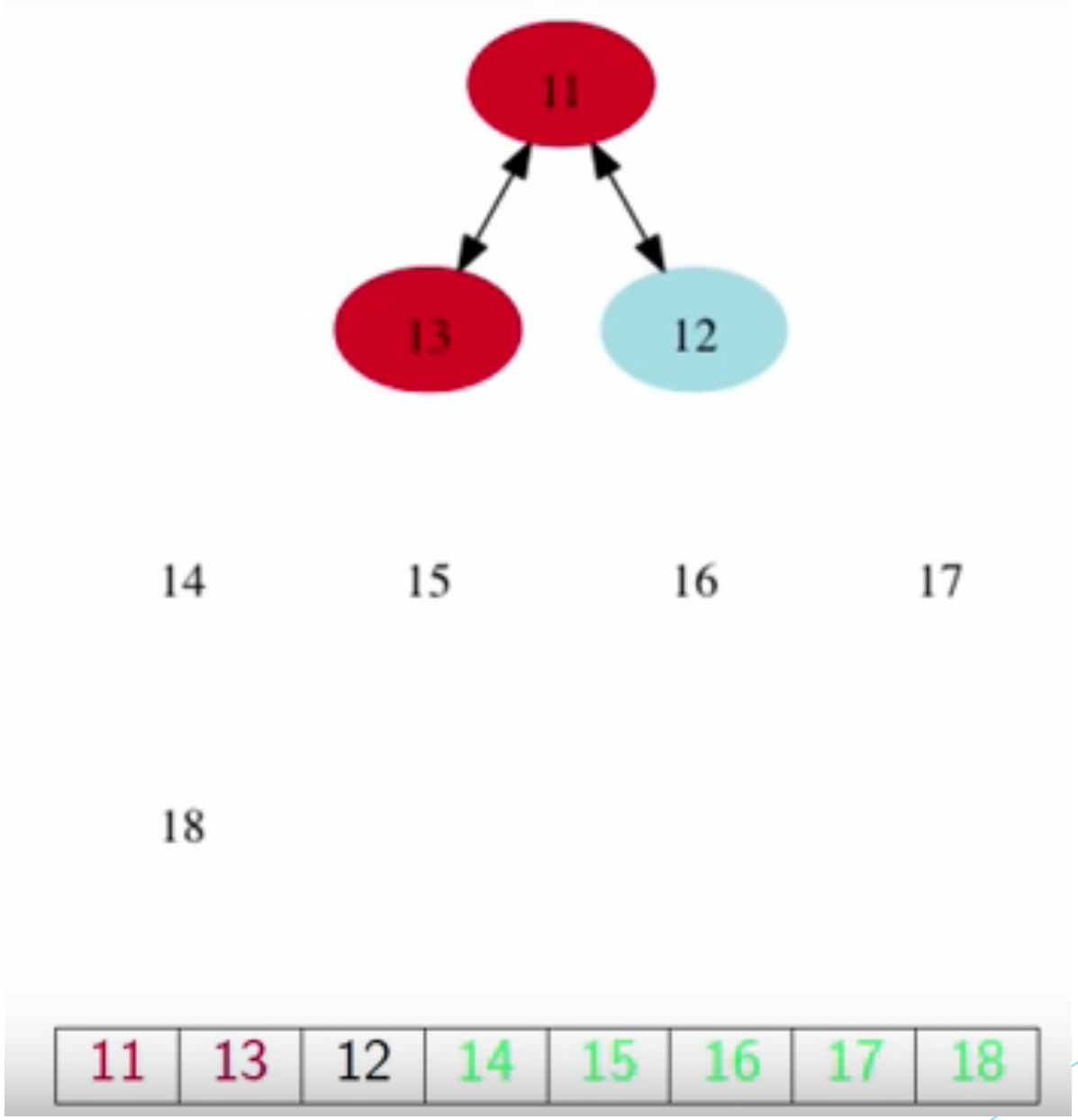


Ejemplo

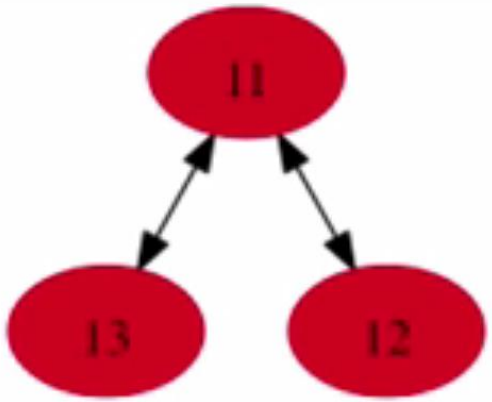
Borrar 14



Ejemplo



Ejemplo

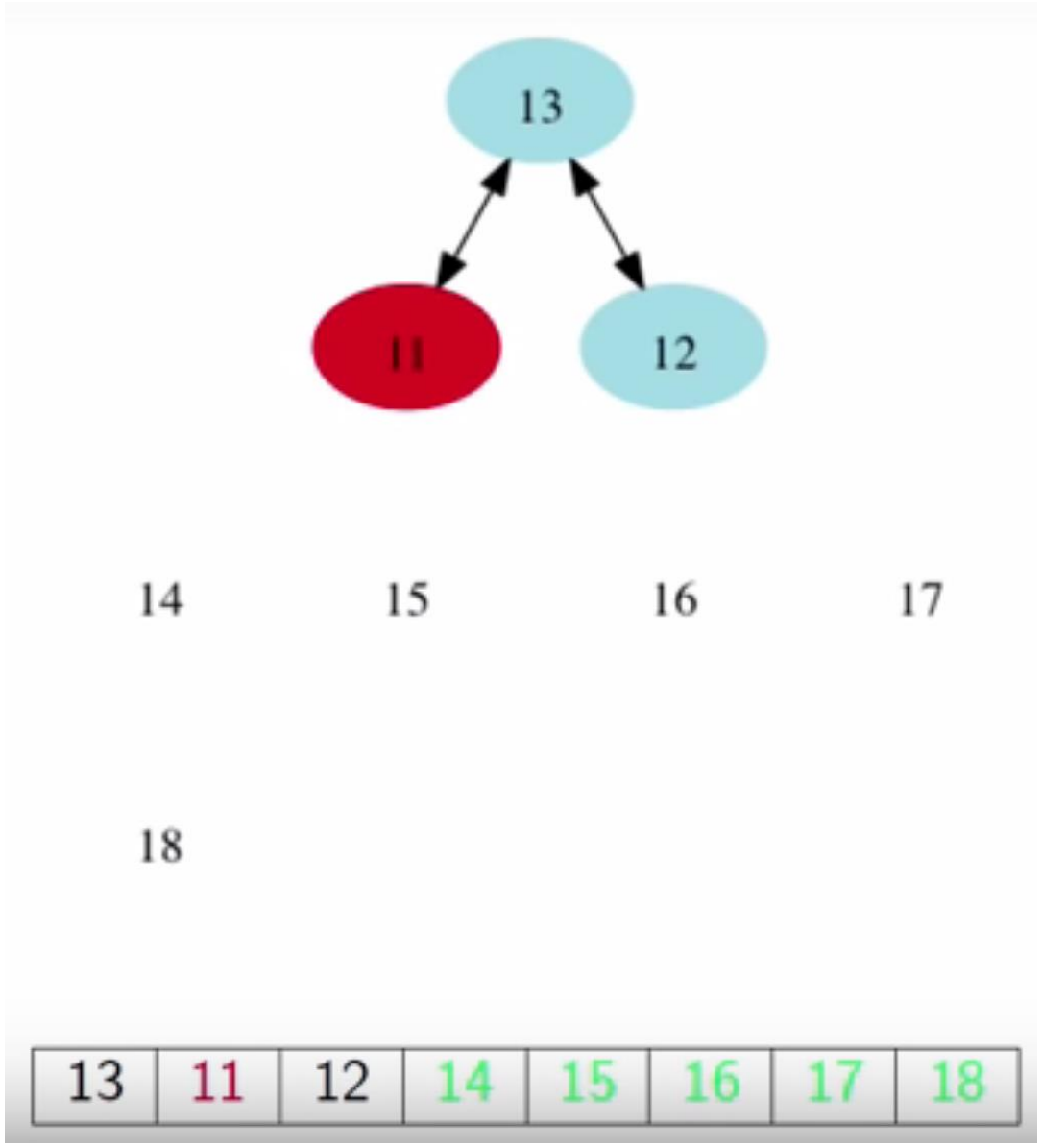


14 15 16 17

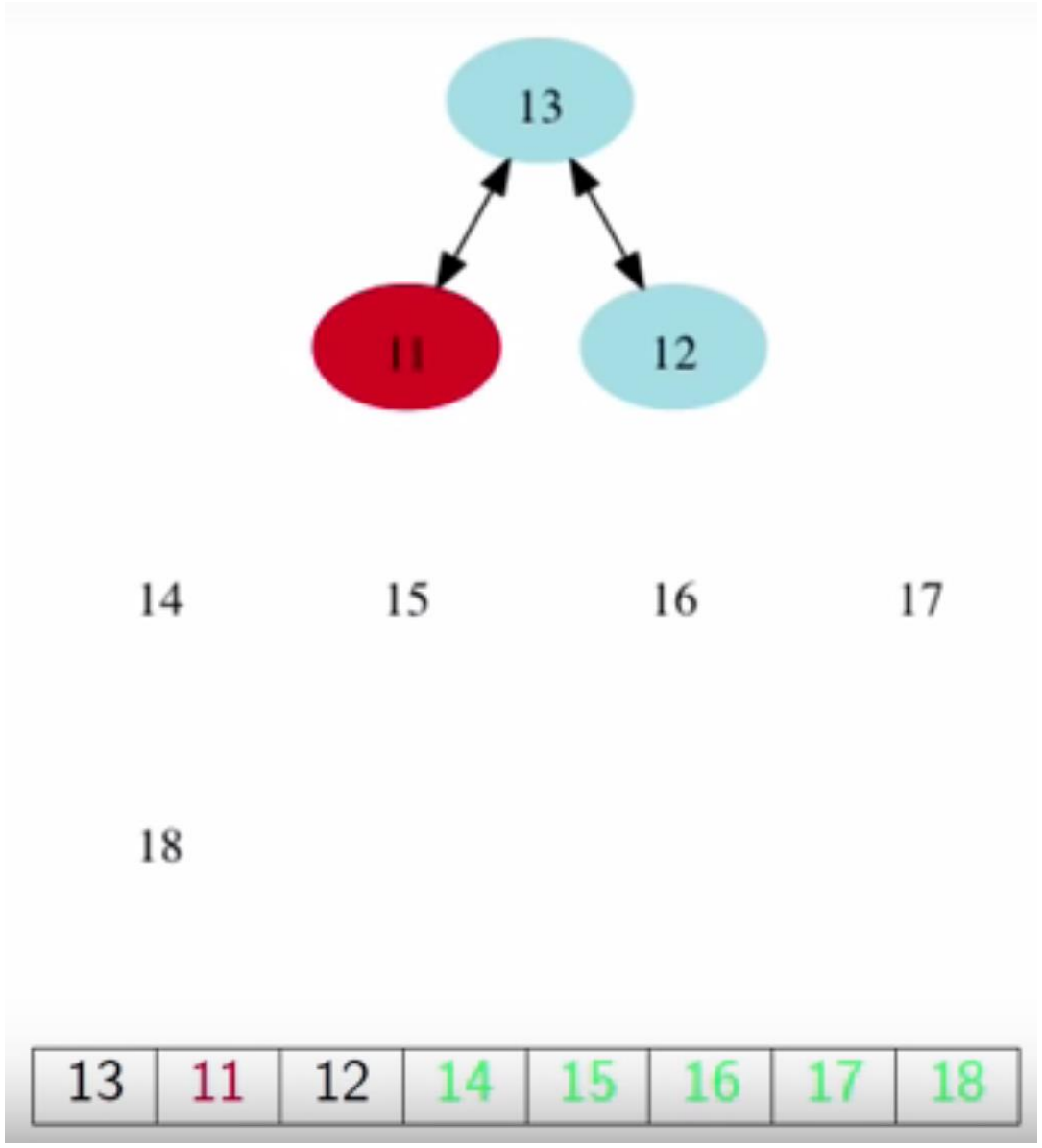
18

11	13	12	14	15	16	17	18

Ejemplo

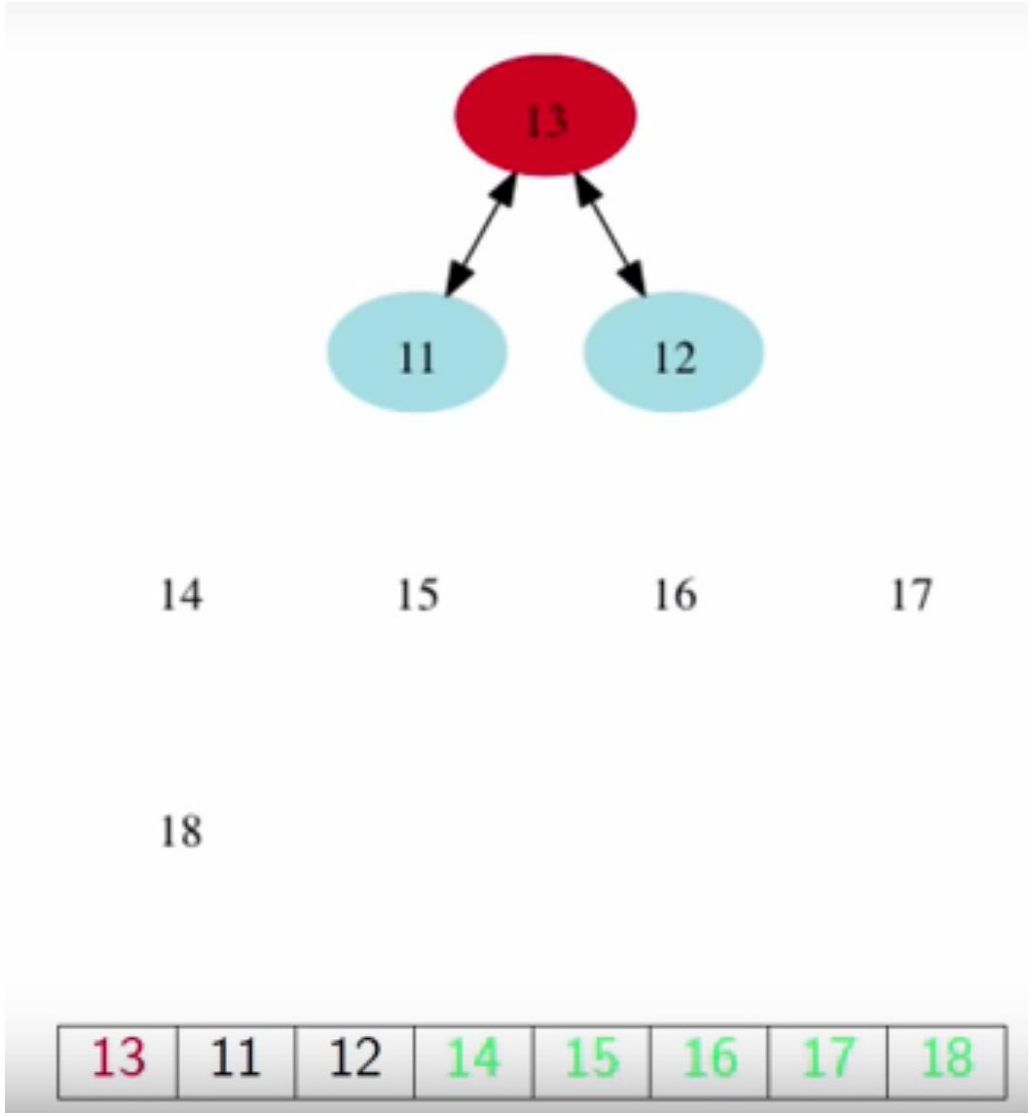


Ejemplo

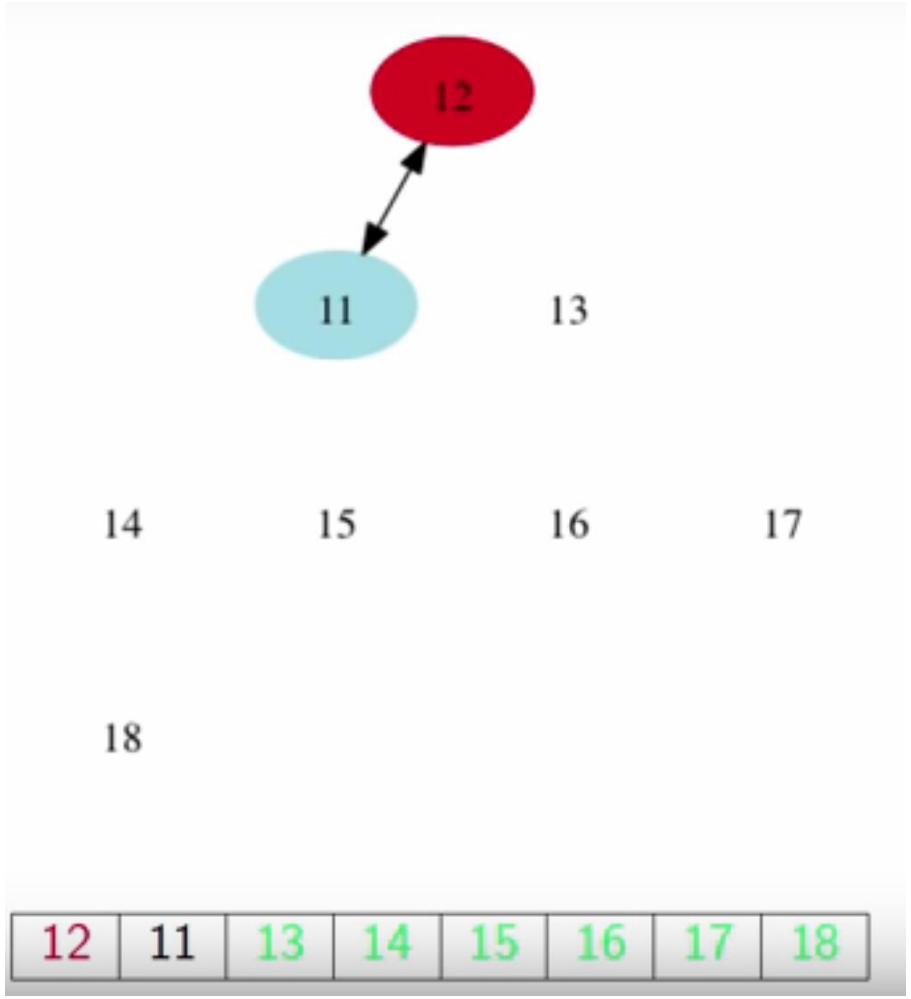


Ejemplo

Borrar 13

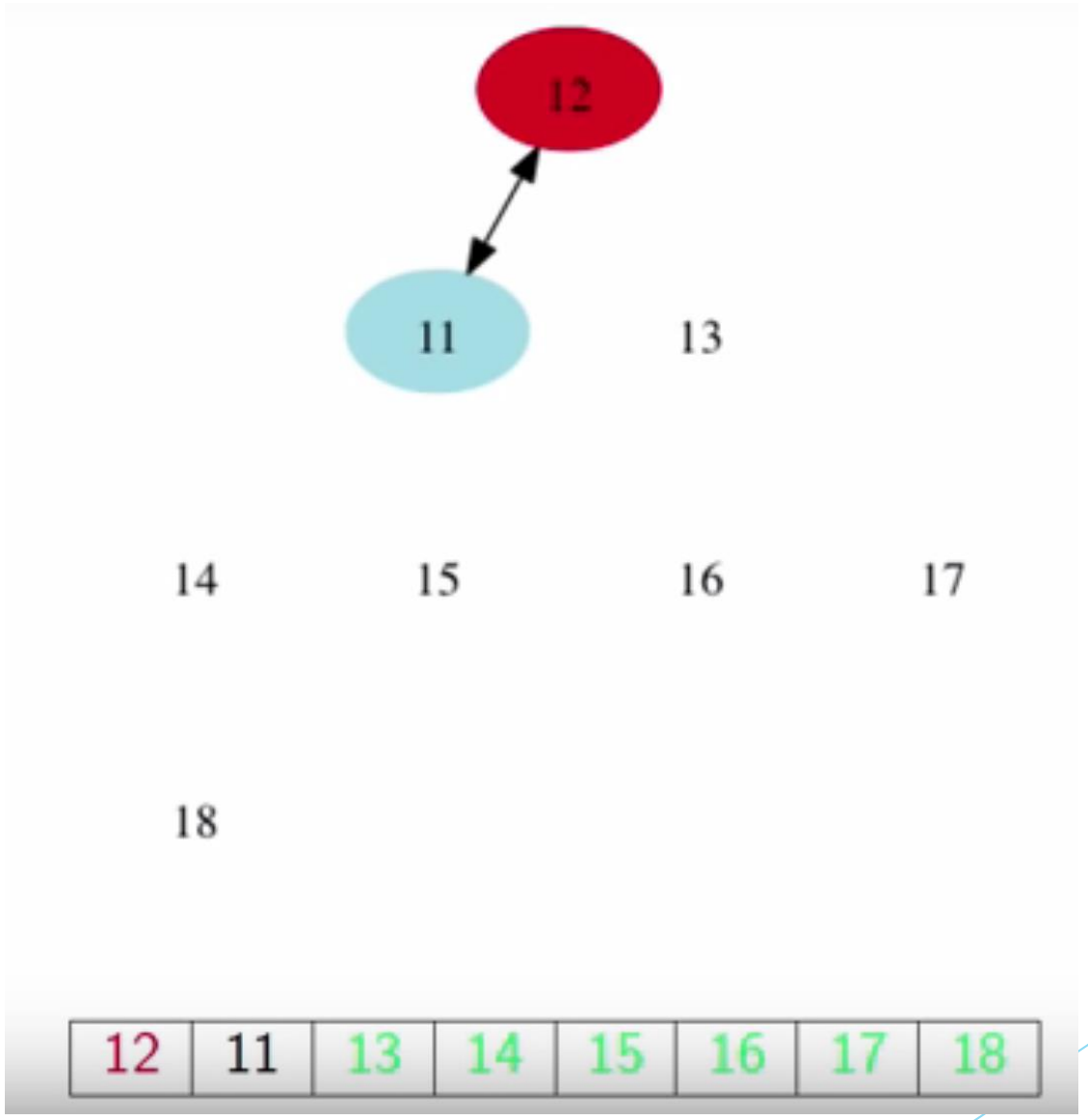


Ejemplo

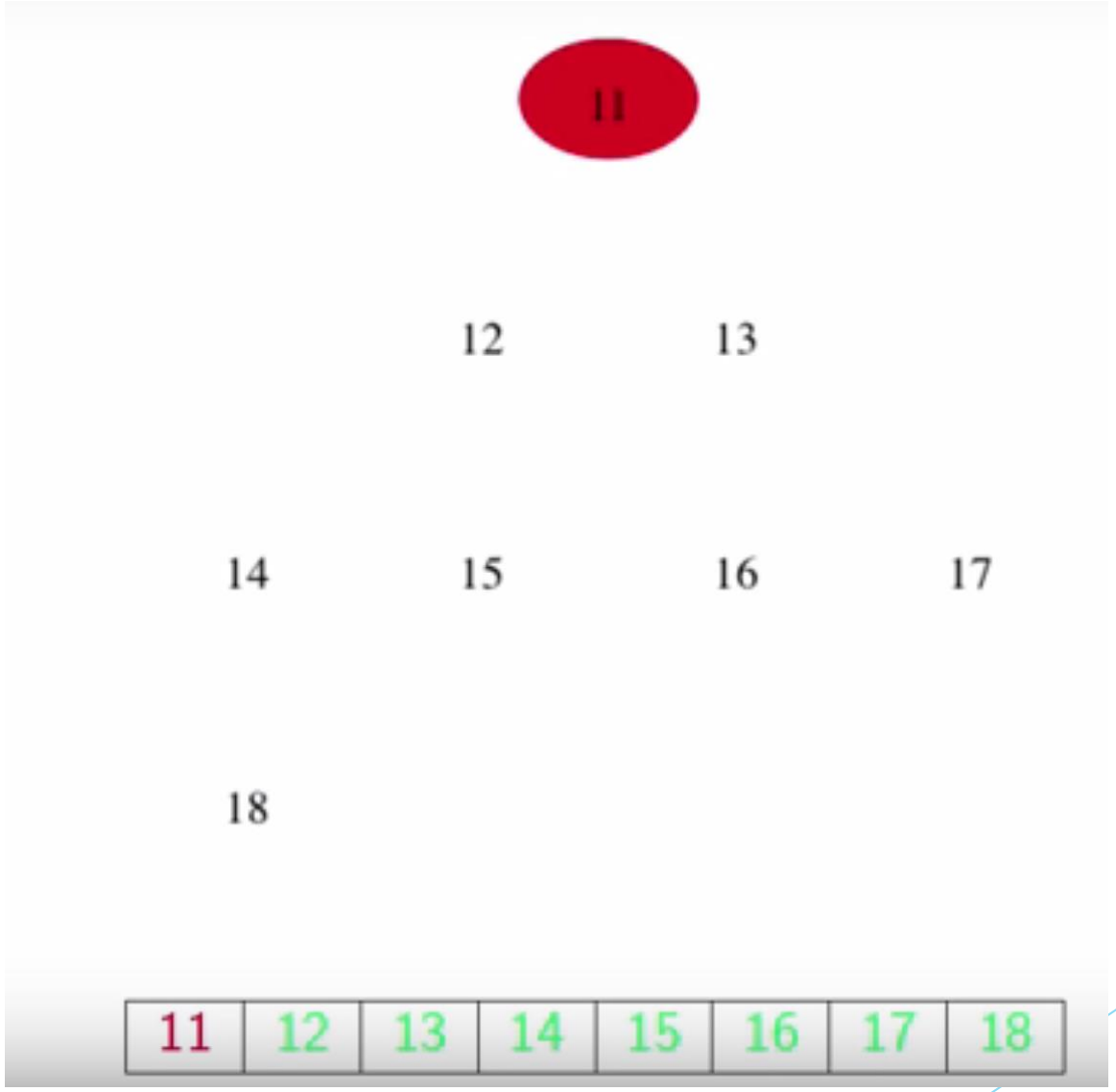


Ejemplo

Borrar 12

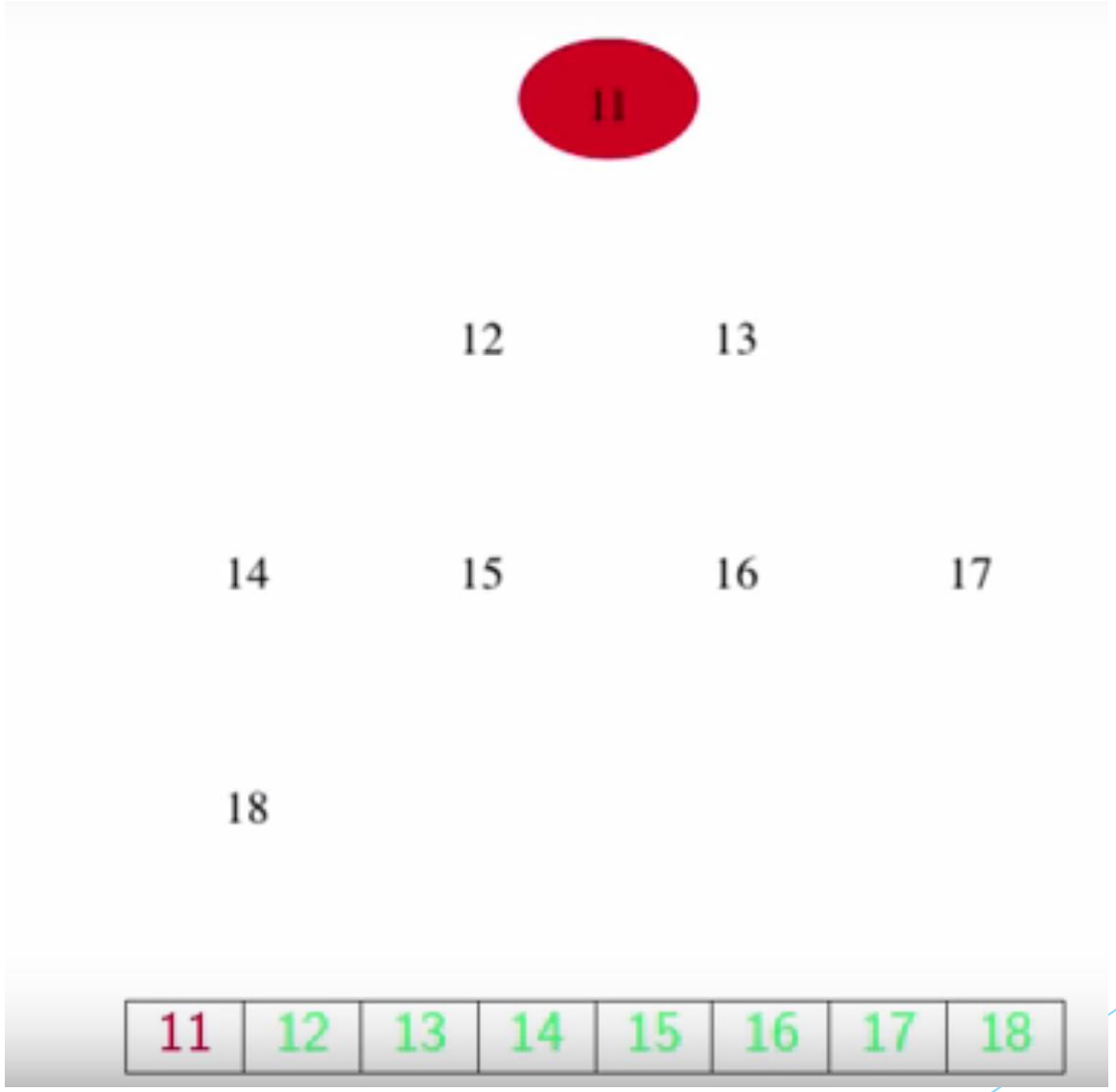


Ejemplo



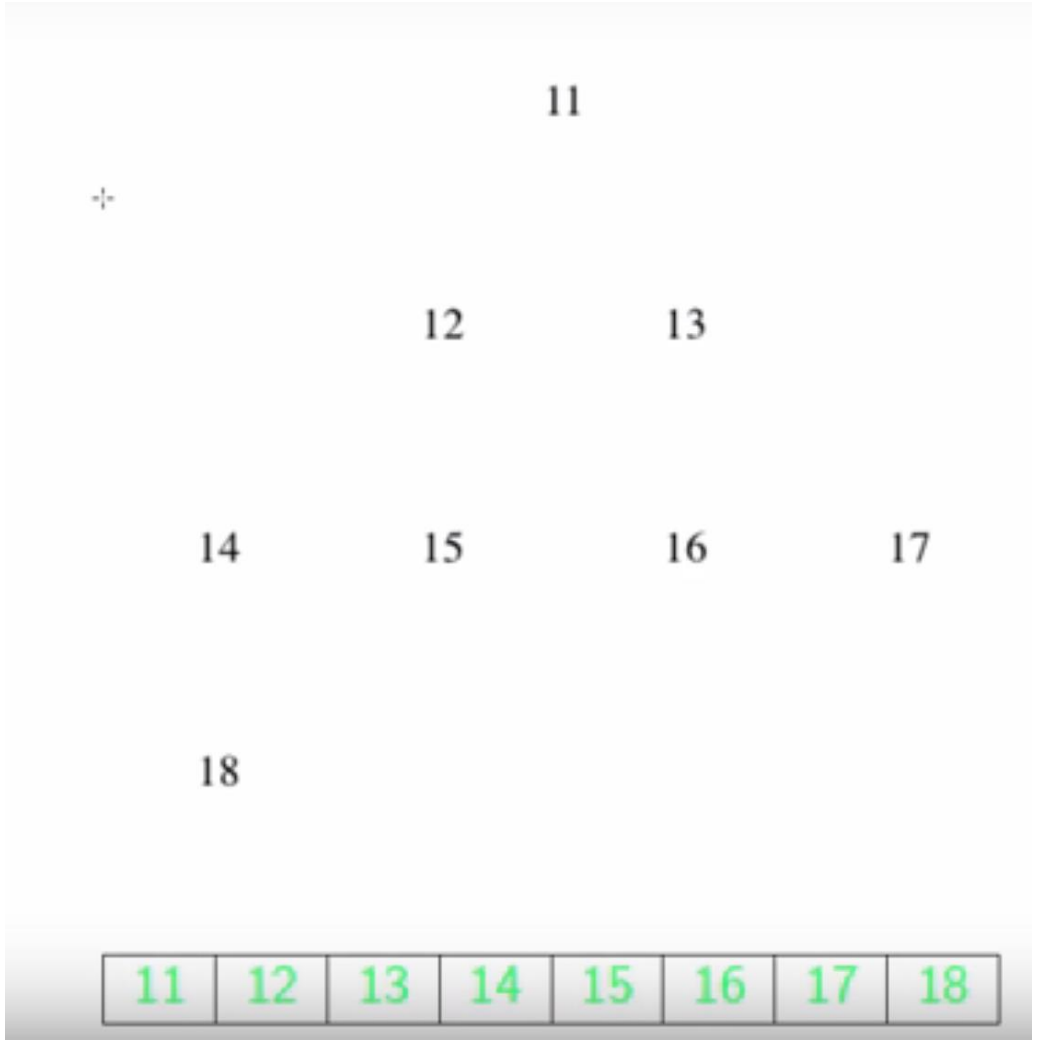
Ejemplo

Borrar 11



Ejemplo

Ordenado



GRACIAS