Heaps

Estructura de Datos

Introducción

- La estructura heap es frecuentemente usada para implementar *colas de prioridad*. En este tipo de colas, el elemento a ser eleminado (borrados) es aquél que tiene mayor (o menor) prioridad.

- En cualquier momento, un elemento con una prioridad arbitraria puede ser insertado en la cola. Una estructura de datos que soporta estas dos operaciones es la *cola de prioridad máxima (mínima)*.

- Se asume que un elemento es una estructura con una miembro dato *key* además de otros miembros datos.

- Por otra parte se sabe que cualquier estructura de datos que implemente una *cola de prioridad máxima* tiene que implementar las operaciones *insert* y *delete*.

Heaps (Montículos)

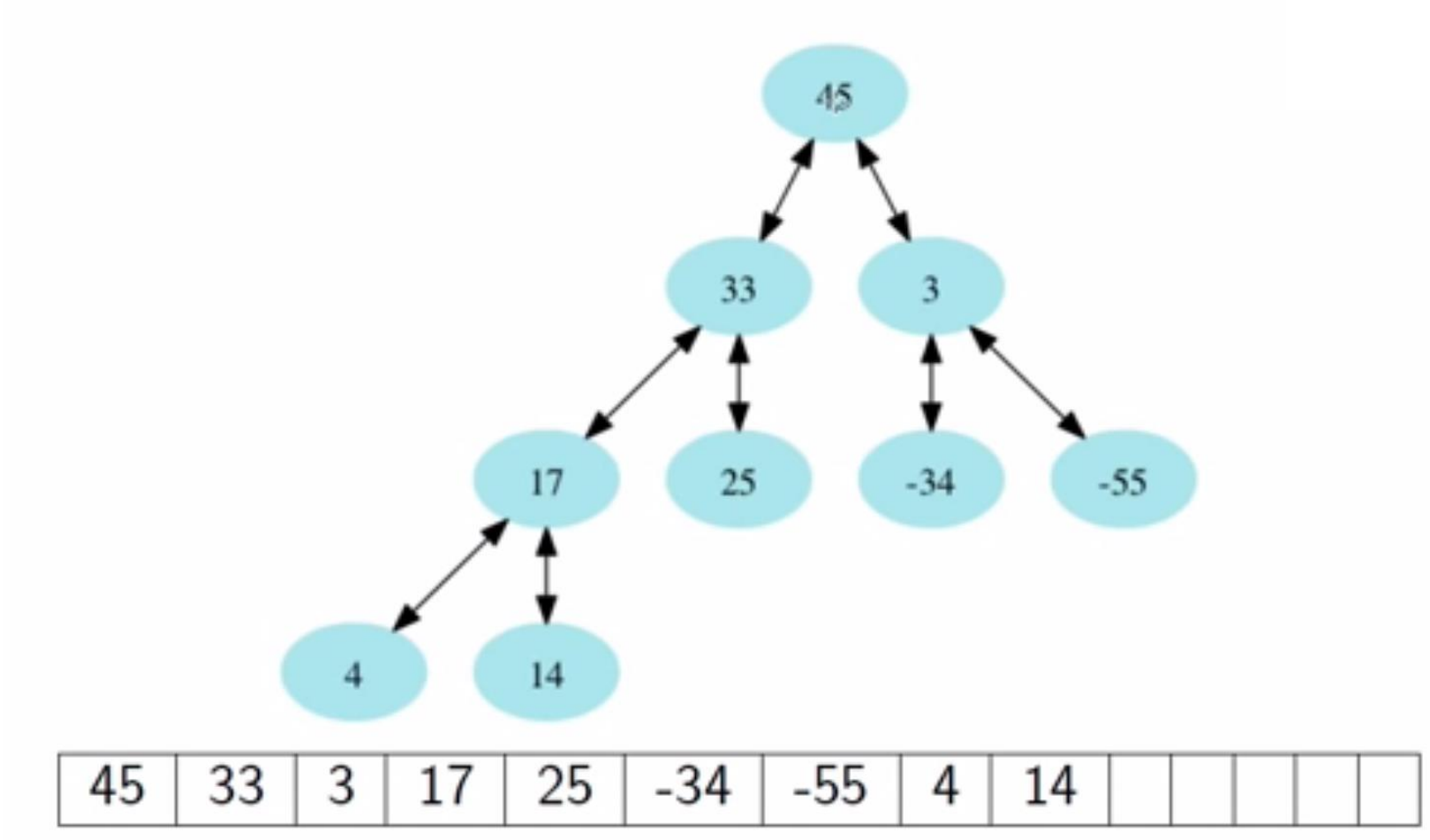
- En computación, un montículo (heap en inglés) es una estructura de datos del tipo árbol con información perteneciente a un conjunto ordenado.

- Los montículos máximos tienen la característica de que cada nodo padre tiene un valor mayor que el de cualquiera de sus nodos hijos, mientras que en los montículos mínimos, el valor del nodo padre es siempre menor al de sus nodos hijos.

- Un árbol cumple la condición de montículo si satisface dicha condición y además es un árbol binario casi completo.

- Un árbol binario es completo cuando todos los niveles están llenos, con la excepción del último, que se llena desde la izquierda hacia la derecha.

Heaps (Montículos)



Tipos

- Un max (min) tree es un árbol en el cual el valor de la llave de cada nodo no es menor (mayor) que la de los valores de las llaves de sus hijos (si tiene alguno).

- Un max heap es un árbol binario completo que es también un max tree.

- Por otra parte, un min heap es un árbol binario completo que es también un min tree.

- De la definición se sabe que la llave del root de un min tree es la menor llave del árbol, mientras que la del *root* de un *max tree* es la mayor. Las operaciones básicas de un heap son:
  - Creación de un heap vacío
  - Inserción de un nuevo elemento en la estructura heap.
  - Eliminación del elemento más grande del heap.

Categorías de un heap

Existen tres categorías de un heap: *max heap*, *min heap* y *min-max heap*.

Un heap tiene las siguientes tres propiedades:

- Es completo, esto es, las hojas de un árbol están en a lo máximo dos niveles adyacentes, y las hojas en el último nivel están en la posición *leftmost*.
- Cada nivel en un heap es llenado en orden de izquierda a derecha.
- Está parcialmente ordenado, esto es, un valor asignado, llamado *key* del elemento almacenado en cada nodo (llamado *parent*), es menor que (mayor que) o igual a las llaves almacenadas en los hijos de los nodos izquierdo y derecho.

Categorías de un heap

- Si la llave (*key)* de cada nodo es mayor que o igual a las llaves de sus hijos, entonces la estructura heap es llamada *max heap*.

- Si la llave (*key)* de cada nodo es menor que o igual a las llaves de sus hijos, entonces la estructura heap es llamada *min heap*.

- En una estructura *min-max heap*, un nivel satisface la propiedad *min heap*, y el siguiente nivel inferior satisface la propiedad *max heap*, alternadamente.

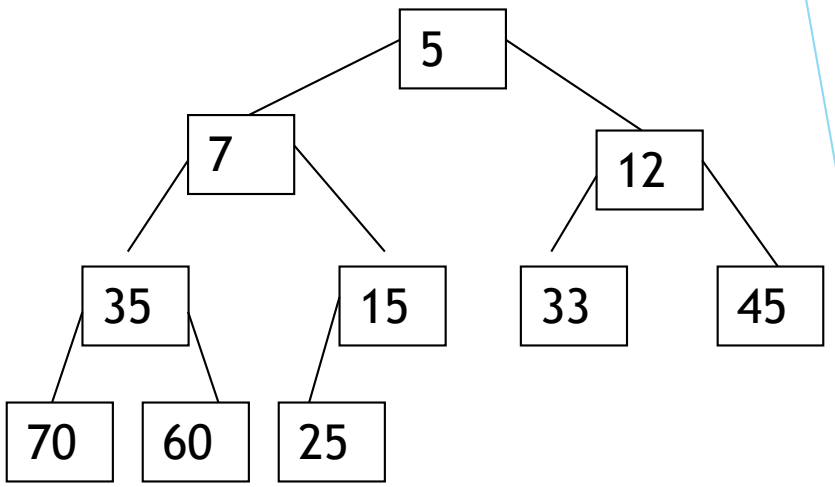
- Un *min-max heap* es útil para colas de prioridad de doble fin.

Categorías de un heap

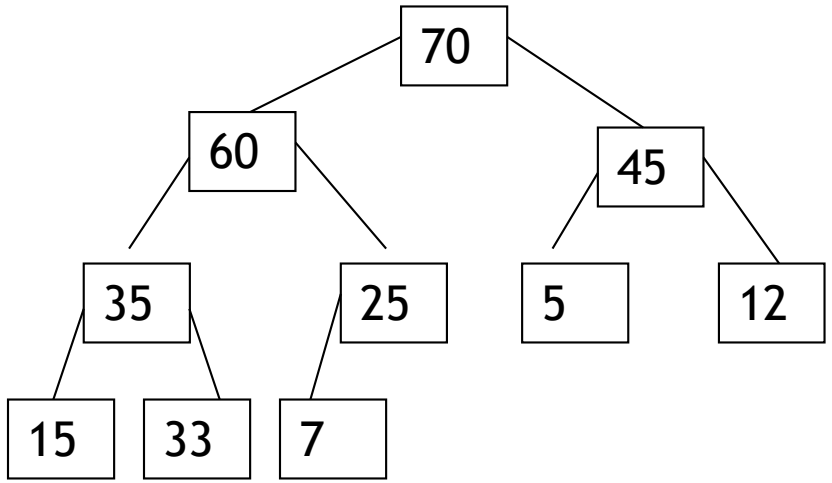
Ejemplo de los tres tipos de heap para el mismo conjunto de valores *key*:

A = {33,60,5,15,25,12,45,70,35,7}

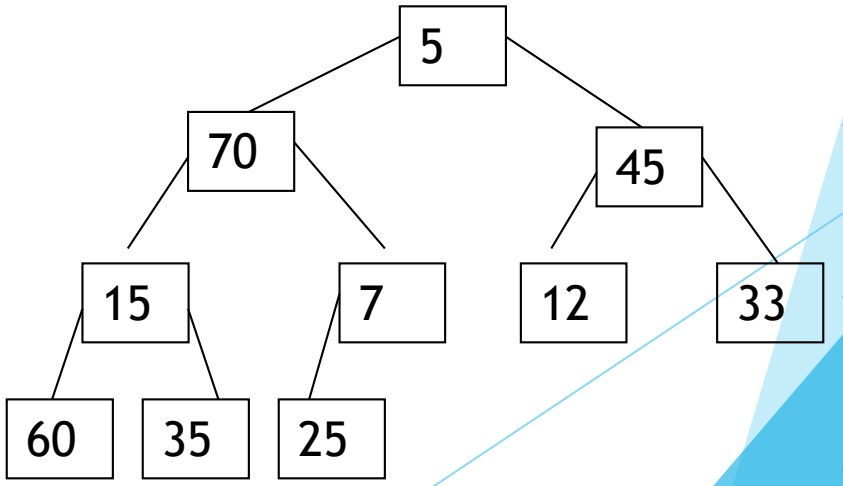
Min heap



Max heap



Min - Max heap



Niveles

Min

Max

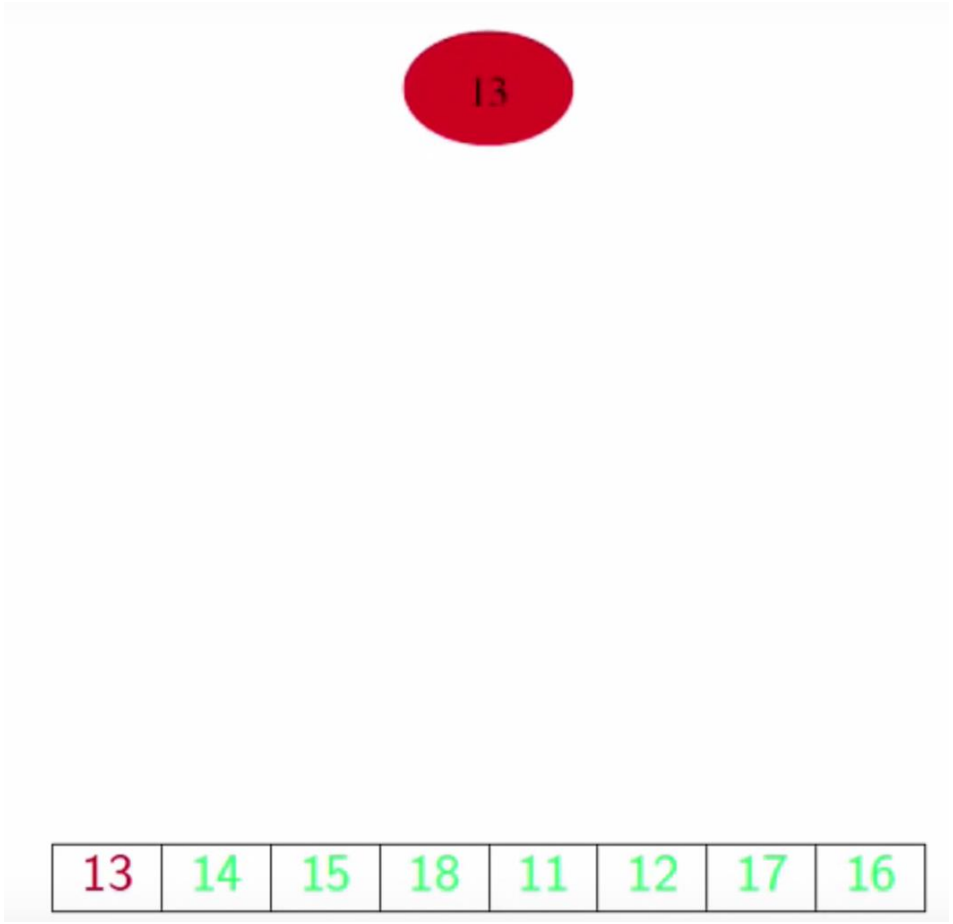
Min

Max

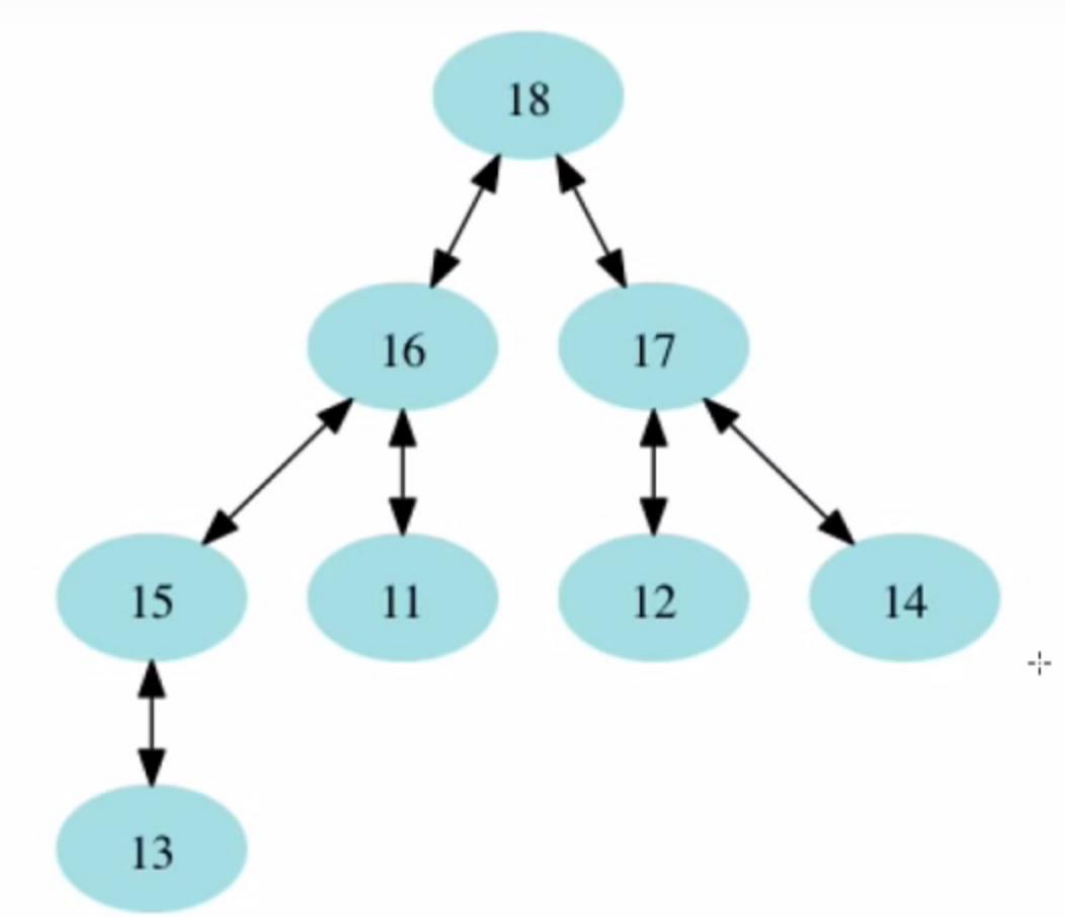
Heaps (Montículos)

- En un montículo de prioridad, el mayor elemento (o el menor, dependiendo de la relación de orden escogida) está siempre en el nodo raíz.
- Por esta razón, los montículos son útiles para implementar colas de prioridad.
- Una ventaja que poseen los montículos es que, por ser árboles completos, se pueden implementar usando arreglos (arrays), lo cual simplifica su codificación y libera al programador del uso de punteros.
- La eficiencia de las operaciones en los montículos es crucial en diversos algoritmos de recorrido

Operaciones – Crear (Opción 1)

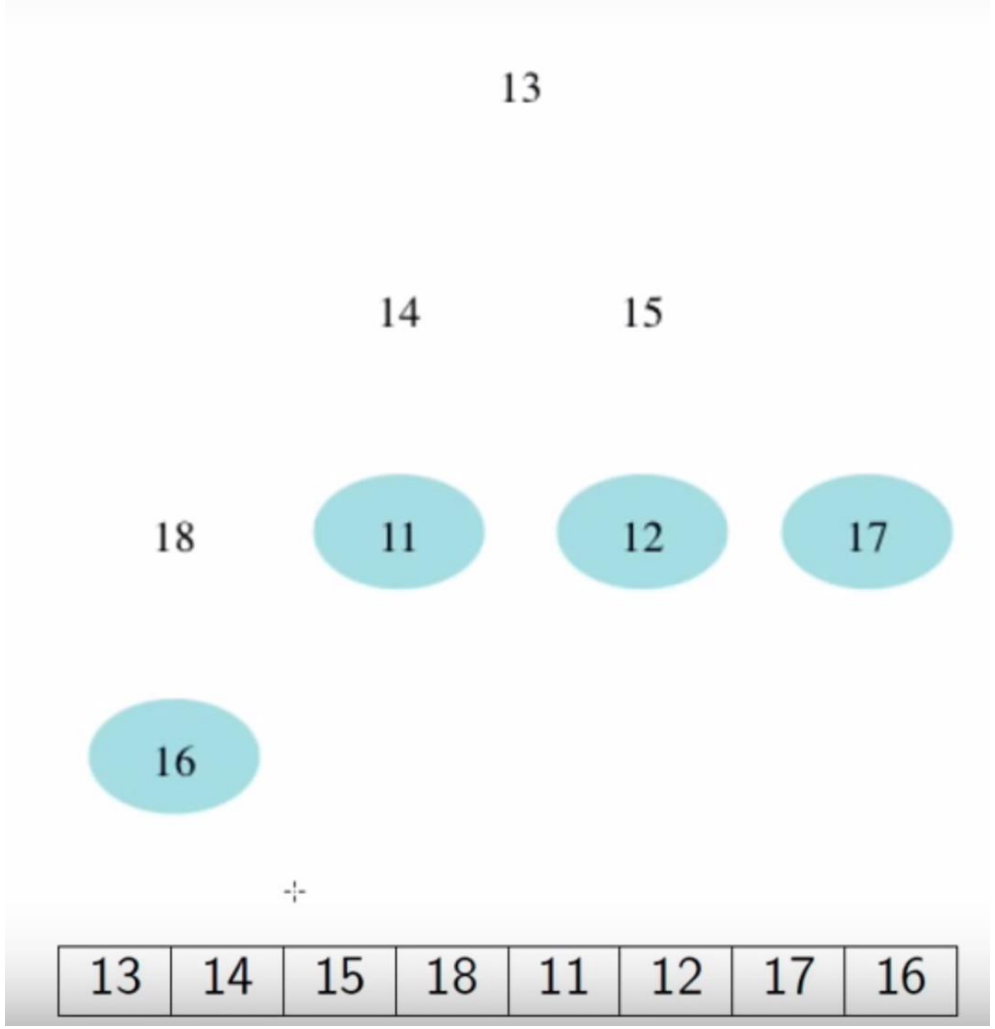


Operaciones – Crear (Opción 1)

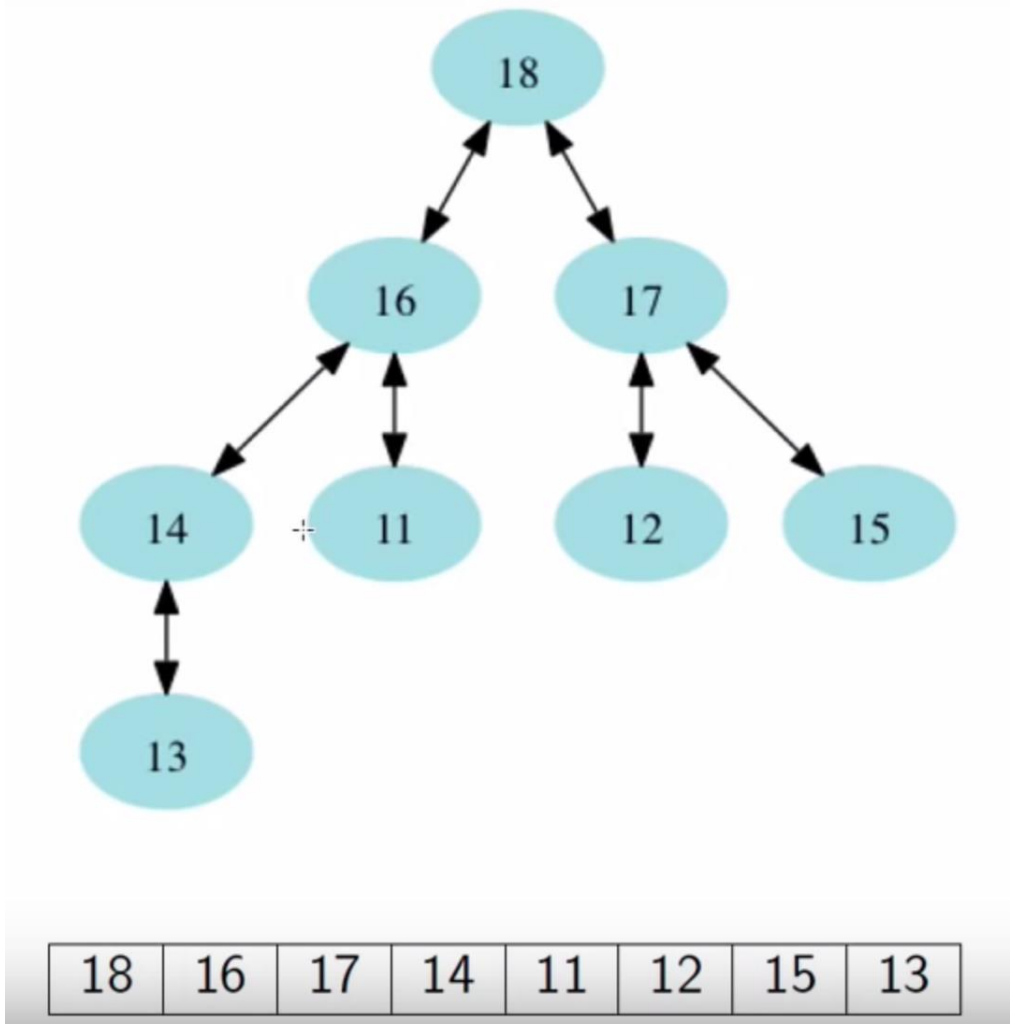


18	16	17	15	11	12	14	13

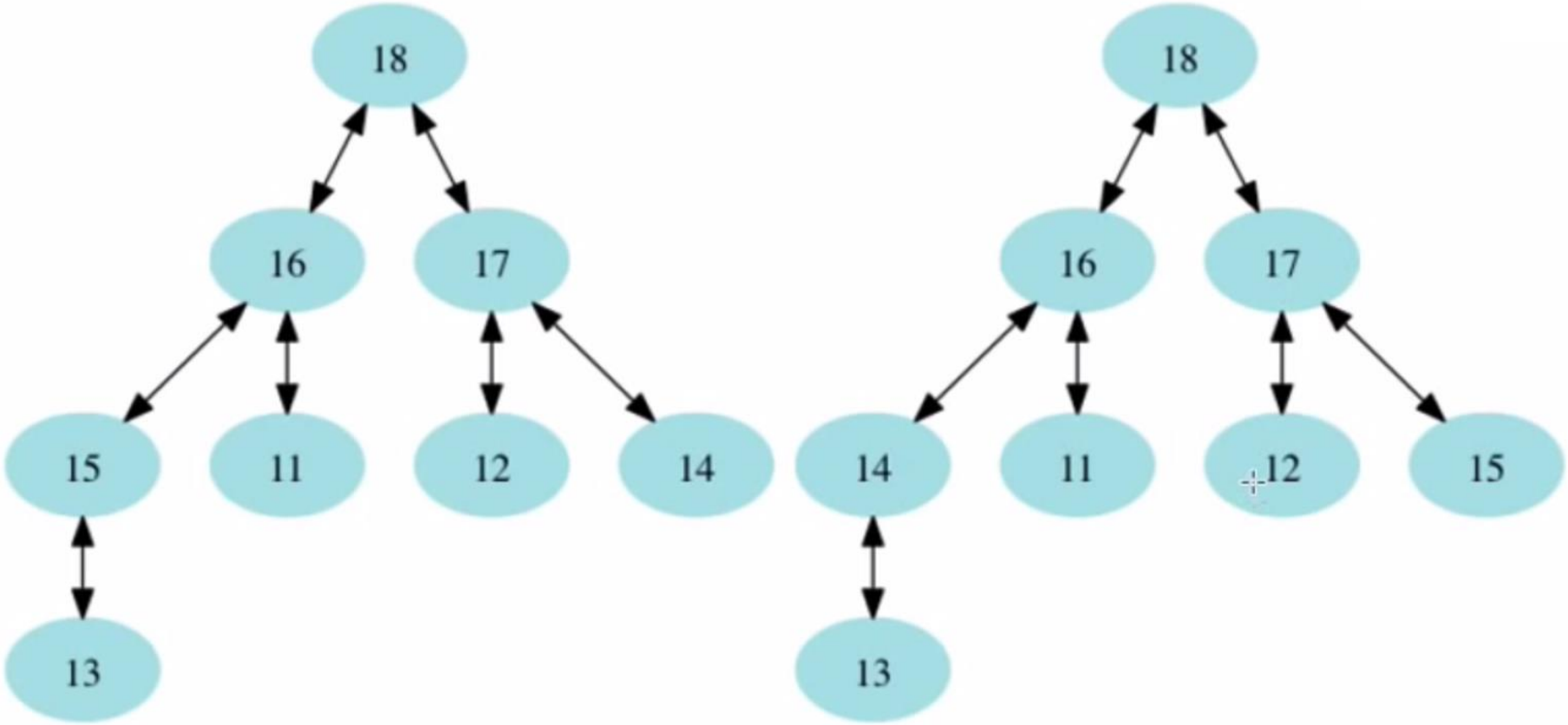
Operaciones – Crear (Opción 2)



Operaciones – Crear (Opción 2)



Comparación



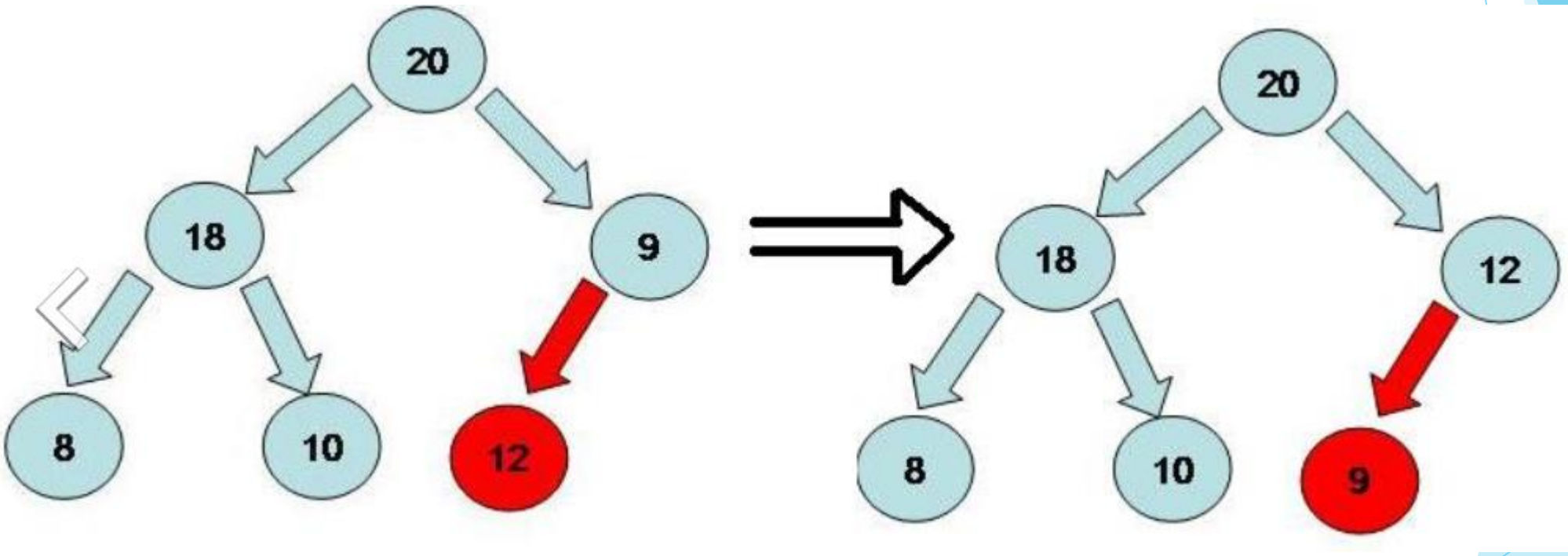
Operaciones - Insertar

- Esta operación parte de un elemento y lo inserta en un montículo aplicando su criterio de ordenación.

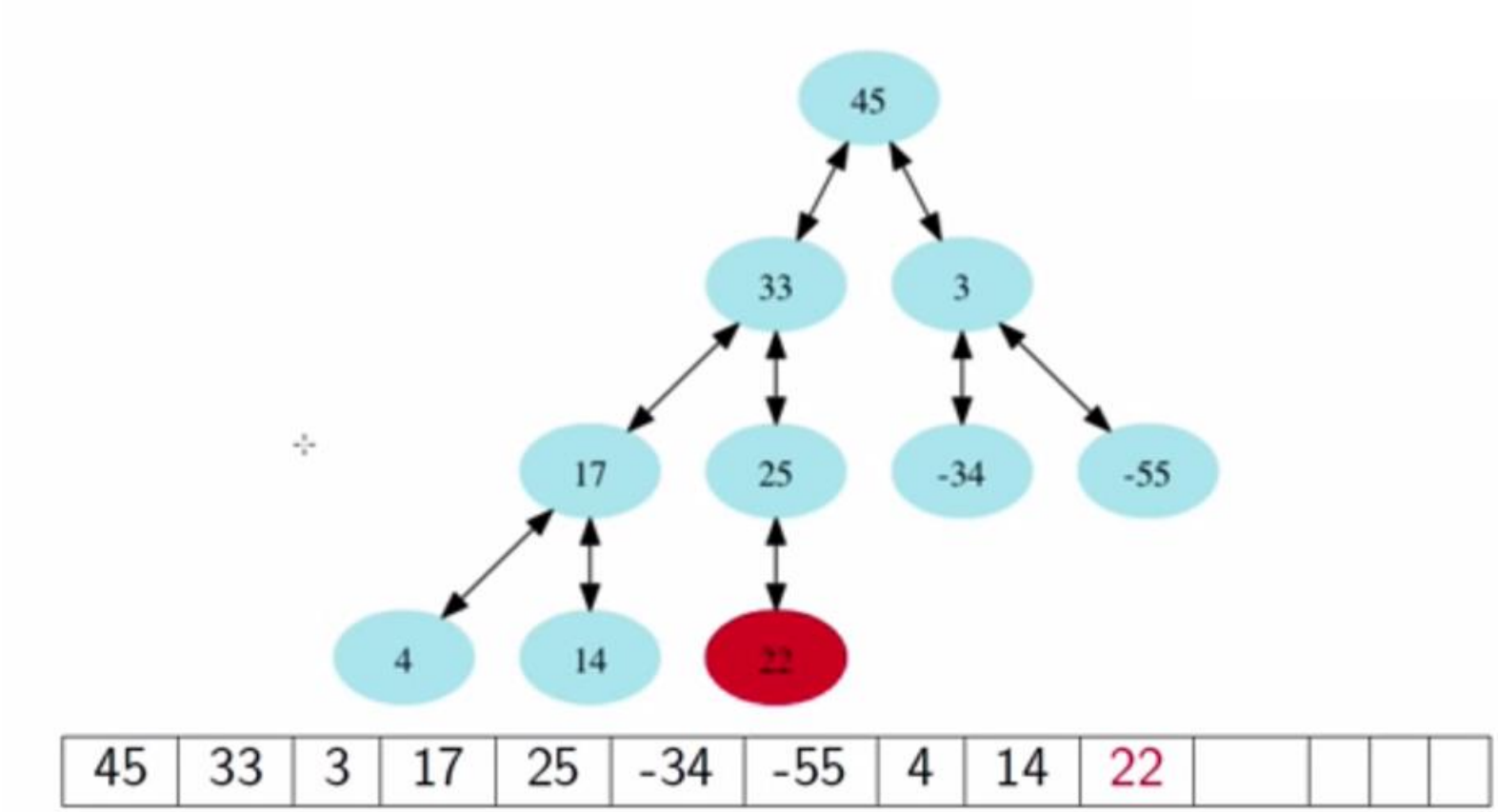
- Si se supone que el monticulo está estructurado de forma que la raíz es mayor que sus hijos, comparamos el elemento a insertar (incluido en la primera posición libre) con su padre.

- Si el hijo es menor que el padre, entonces el elemento es insertado correctamente, si ocurre lo contrario se sustituye el hijo por el padre.

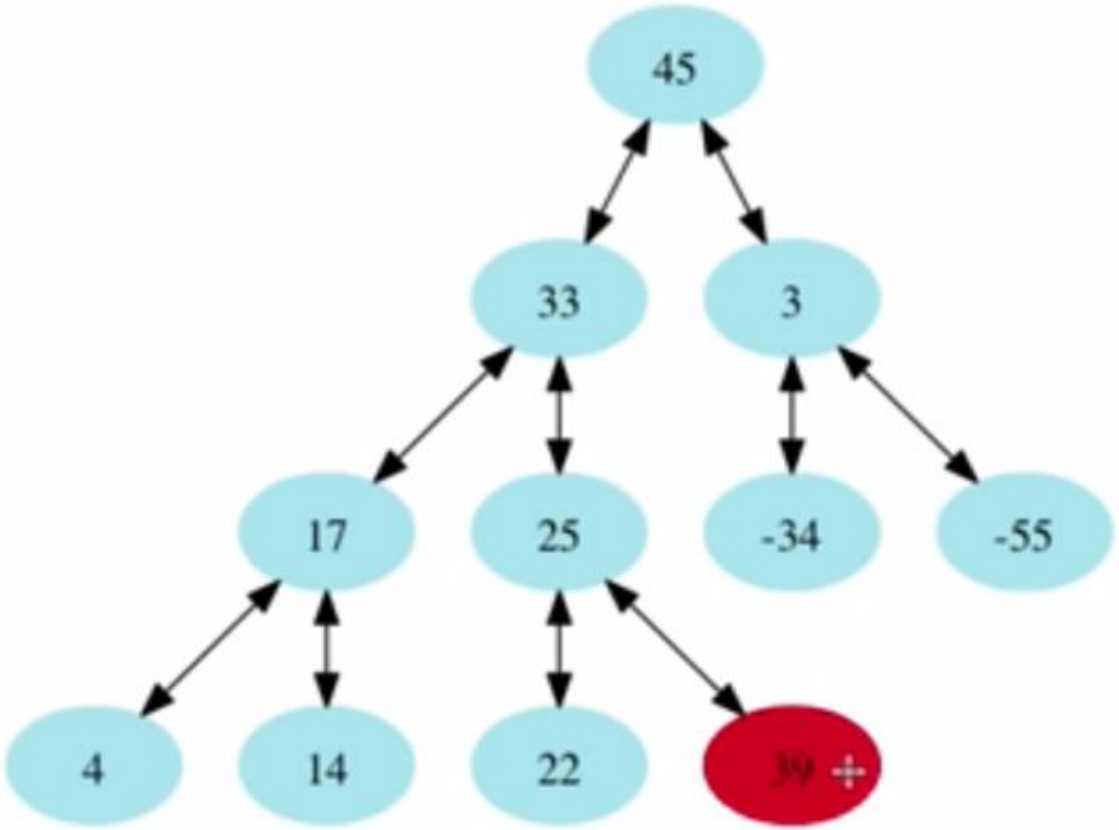
Operaciones - Insertar



Operaciones – Insertar (Ejemplo 1)

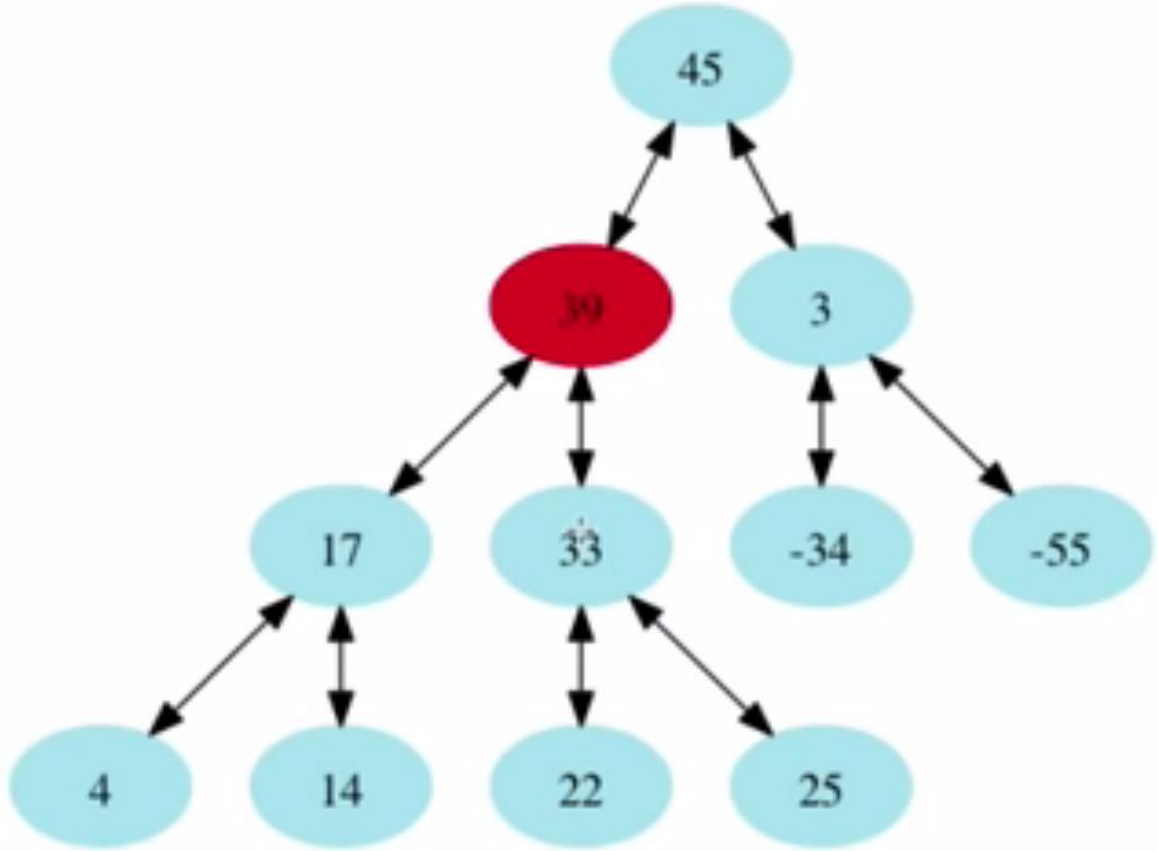


Operaciones – Insertar (Ejemplo 2)



45	33	3	17	25	-34	-55	4	14	22	39			

Operaciones – Insertar (Ejemplo 2)



45	39	3	17	33	-34	-55	4	14	22	25			

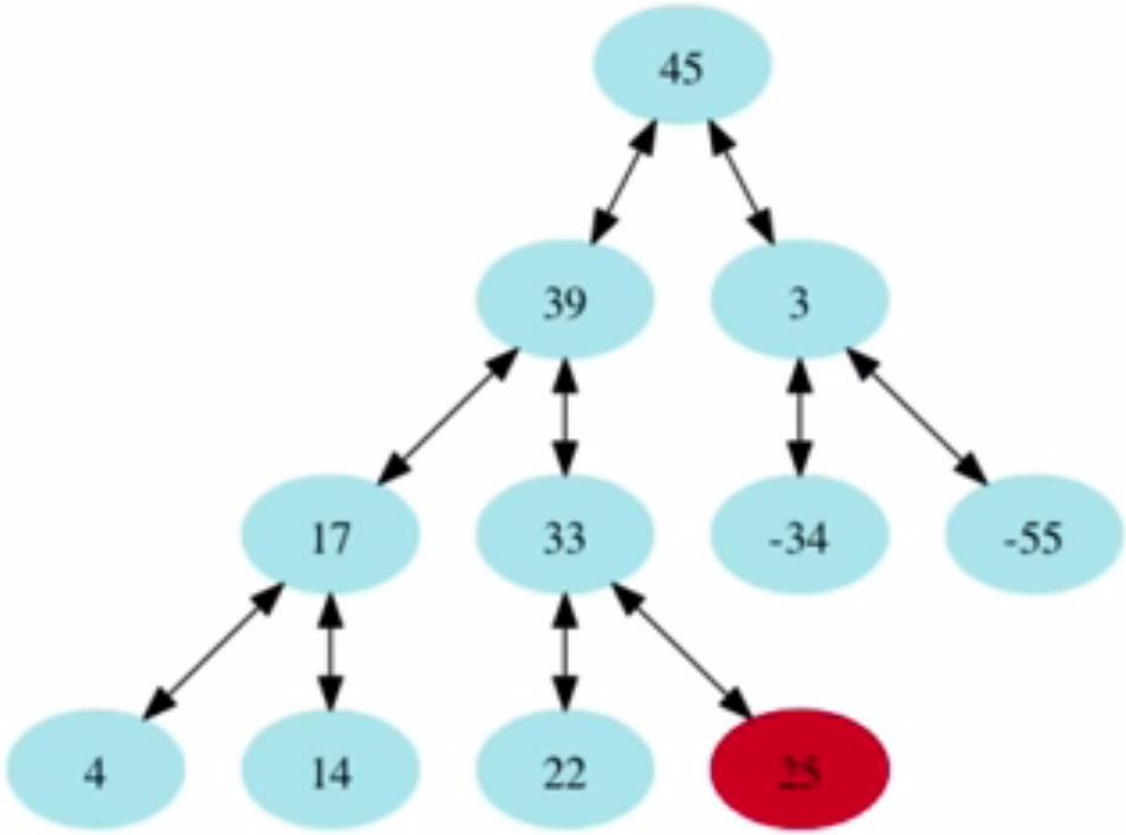
Operaciones - Eliminar

- En este caso se elimina el elemento máximo de un montículo. La forma más eficiente de realizarlo sería buscar el elemento a borrar, colocarlo en la raíz e intercambiarlo por el máximo valor de sus hijos satisfaciendo así la propiedad de montículos de máximos.

- Se puede observar que ya está colocado en la raíz al ser un montículo de máximos, los pasos a seguir son:

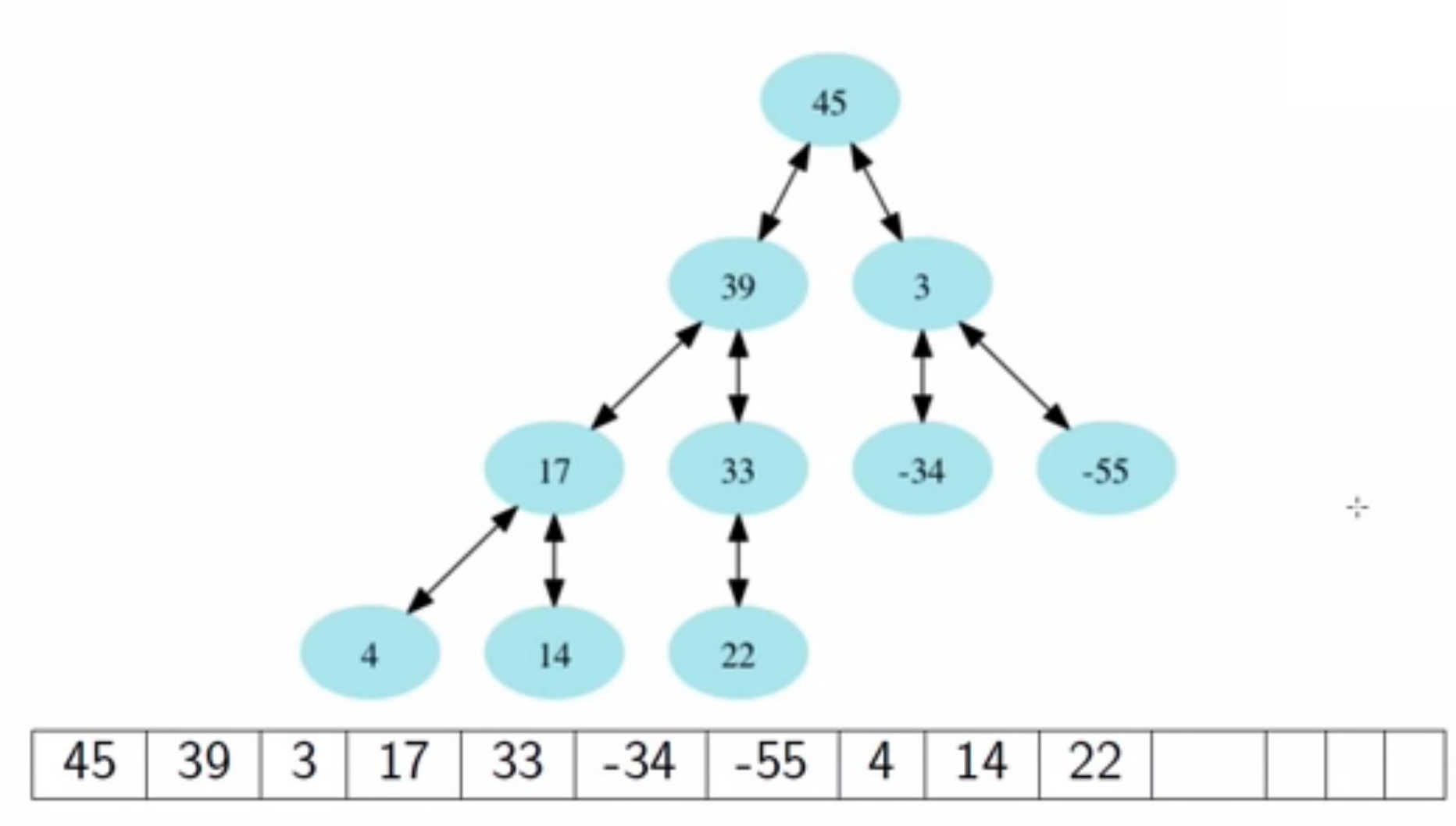
  - Eliminar el elemento máximo (colocado en la raíz).
  - Se sube el elemento que se debe eliminar, para cumplir la condición de montículo a la raíz, que ha quedado vacía.
  - Una vez hecho esto queda el último paso el cual es ver si la raíz tiene hijos mayores que ella si es así, aplicamos la condición y sustituimos el padre por el mayor de sus progenitores.

Operaciones – Eliminar (Ejemplo 1)

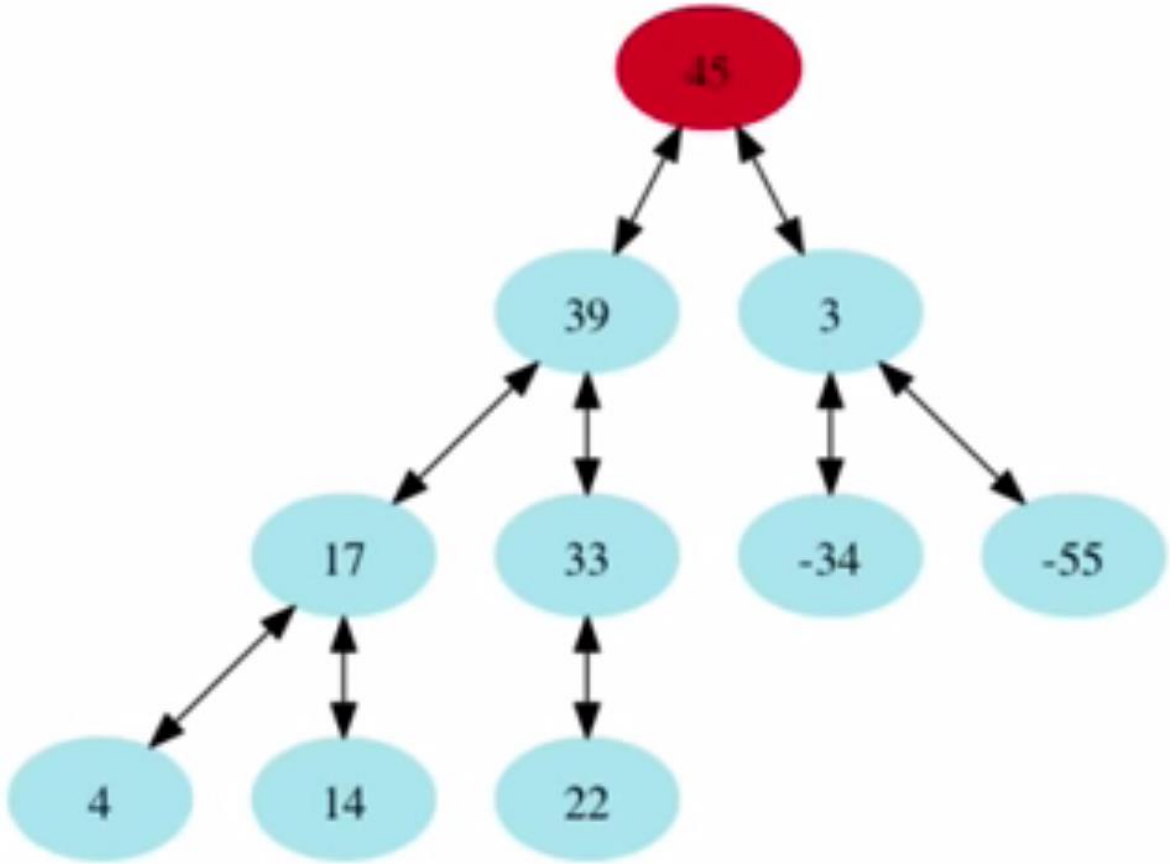


45	39	3	17	33	-34	-55	4	14	22	25			

Operaciones – Eliminar (Ejemplo 1)

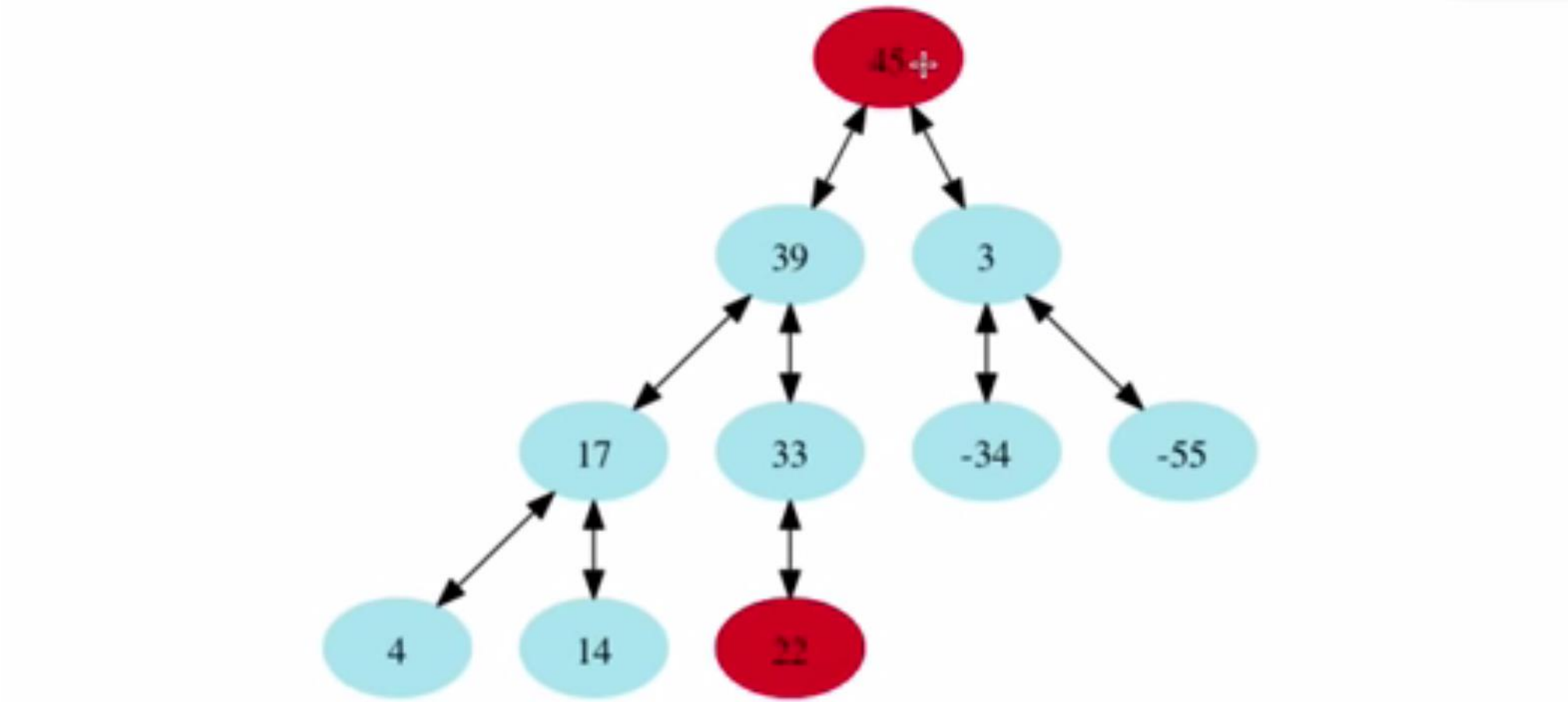


Operaciones – Eliminar (Ejemplo 2)



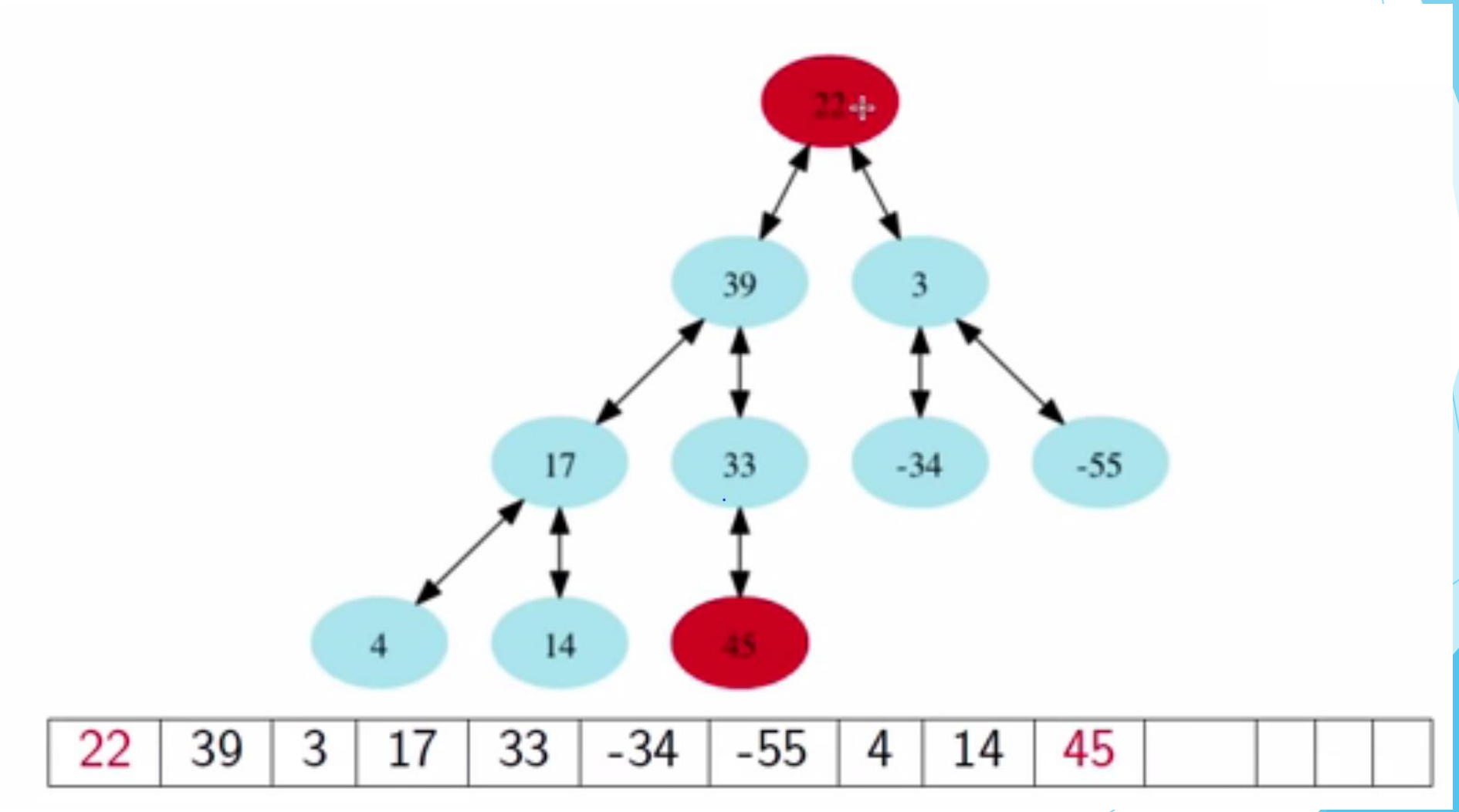
45	39	3	17	33	-34	-55	4	14	22				

Operaciones – Eliminar (Ejemplo 2)

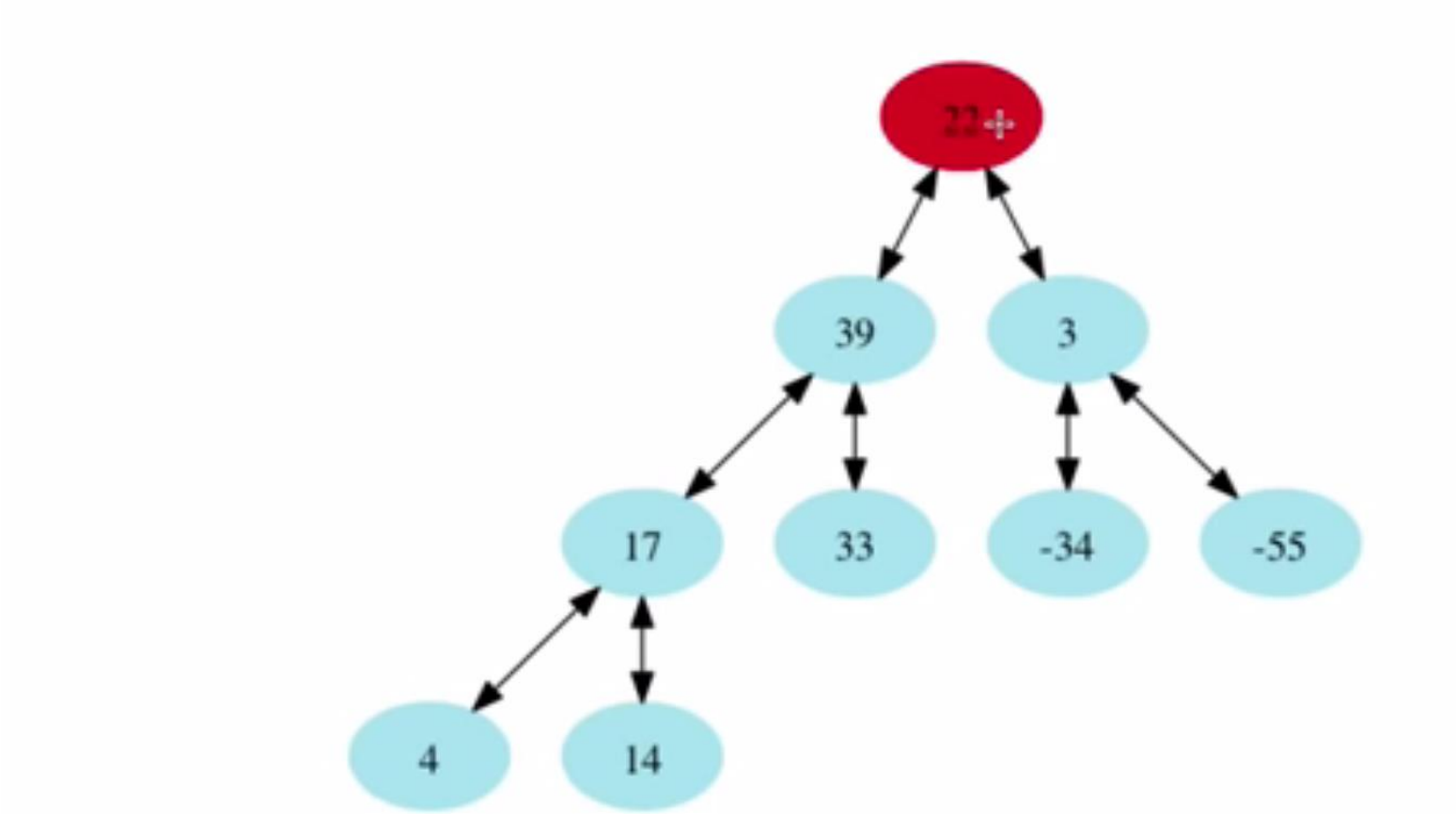


45	39	3	17	33	-34	-55	4	14	22				

Operaciones – Eliminar (Ejemplo 2)

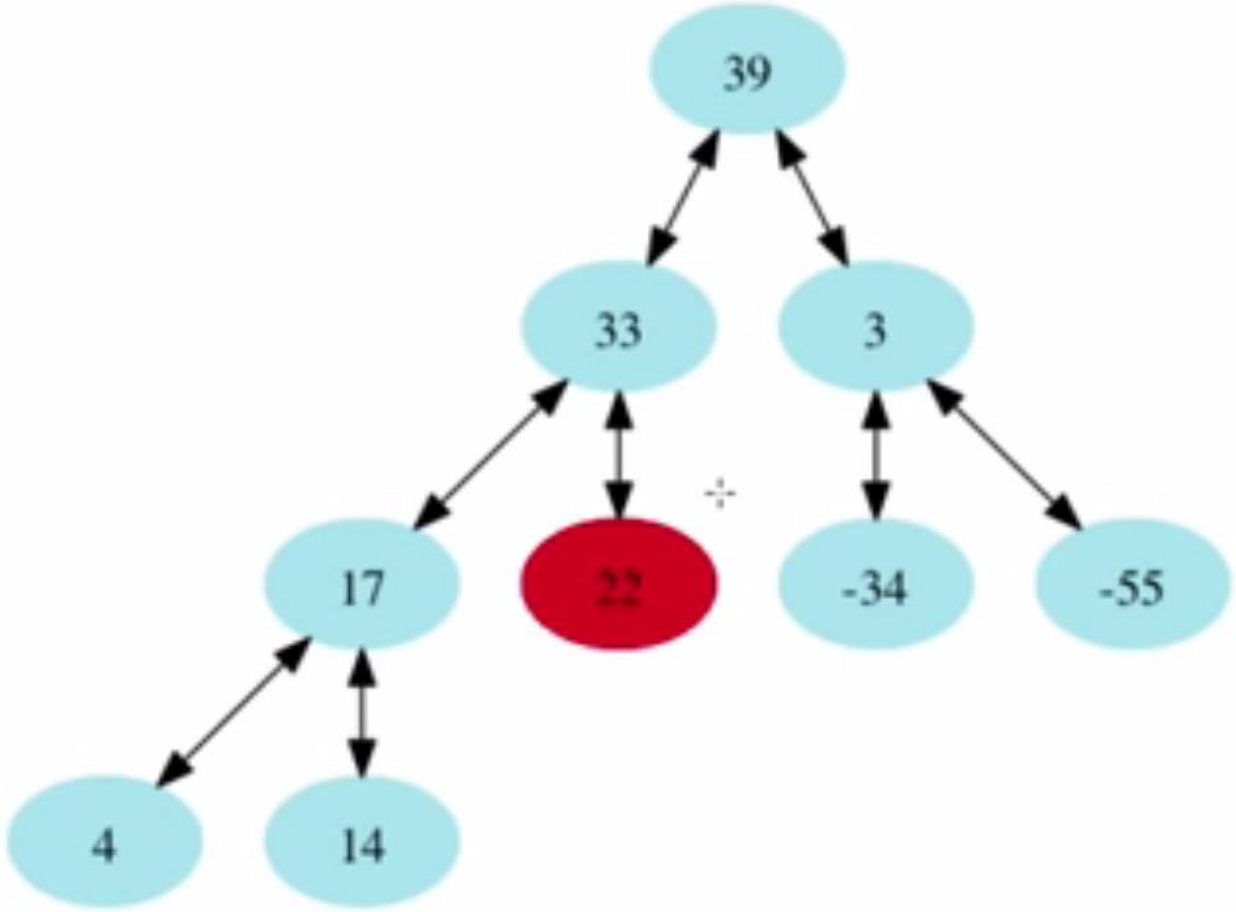


Operaciones – Eliminar (Ejemplo 2)



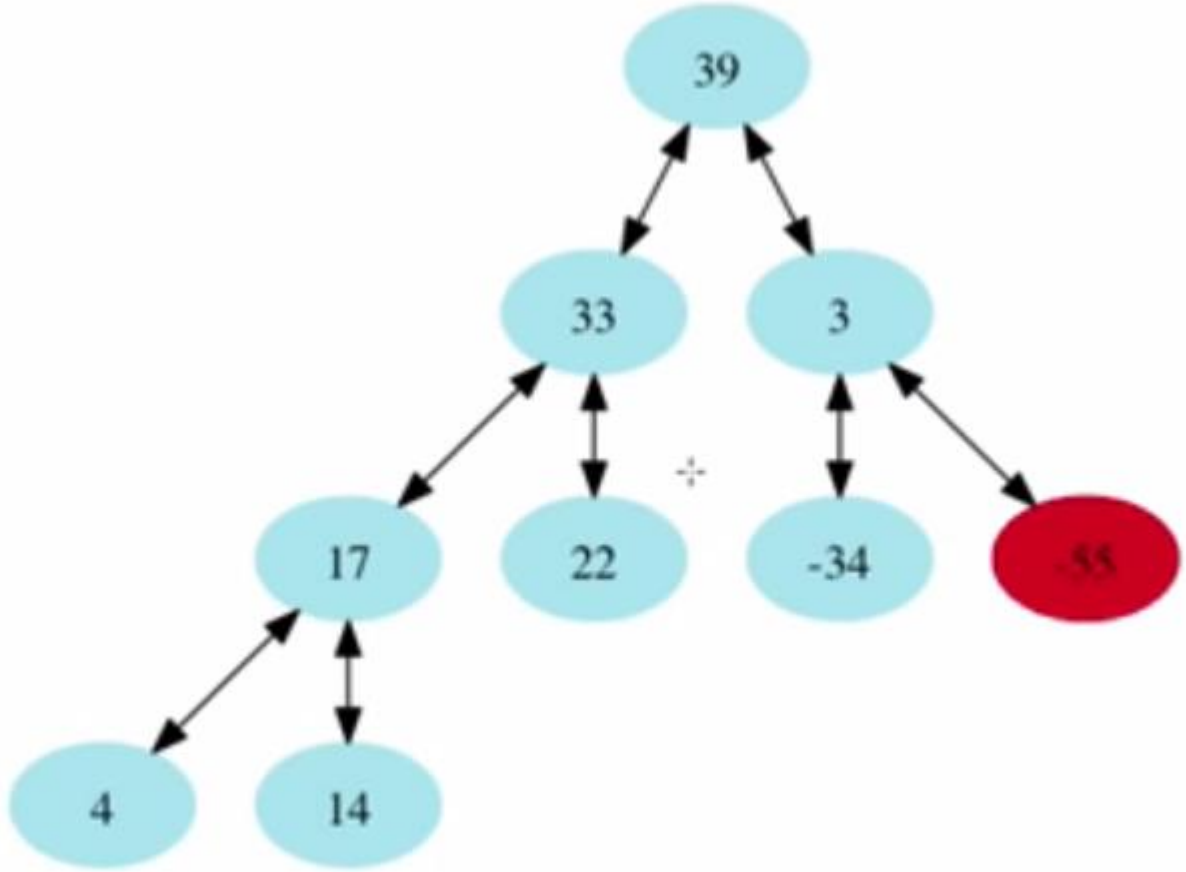
22	39	3	17	33	-34	-55	4	14					

Operaciones – Eliminar (Ejemplo 2)



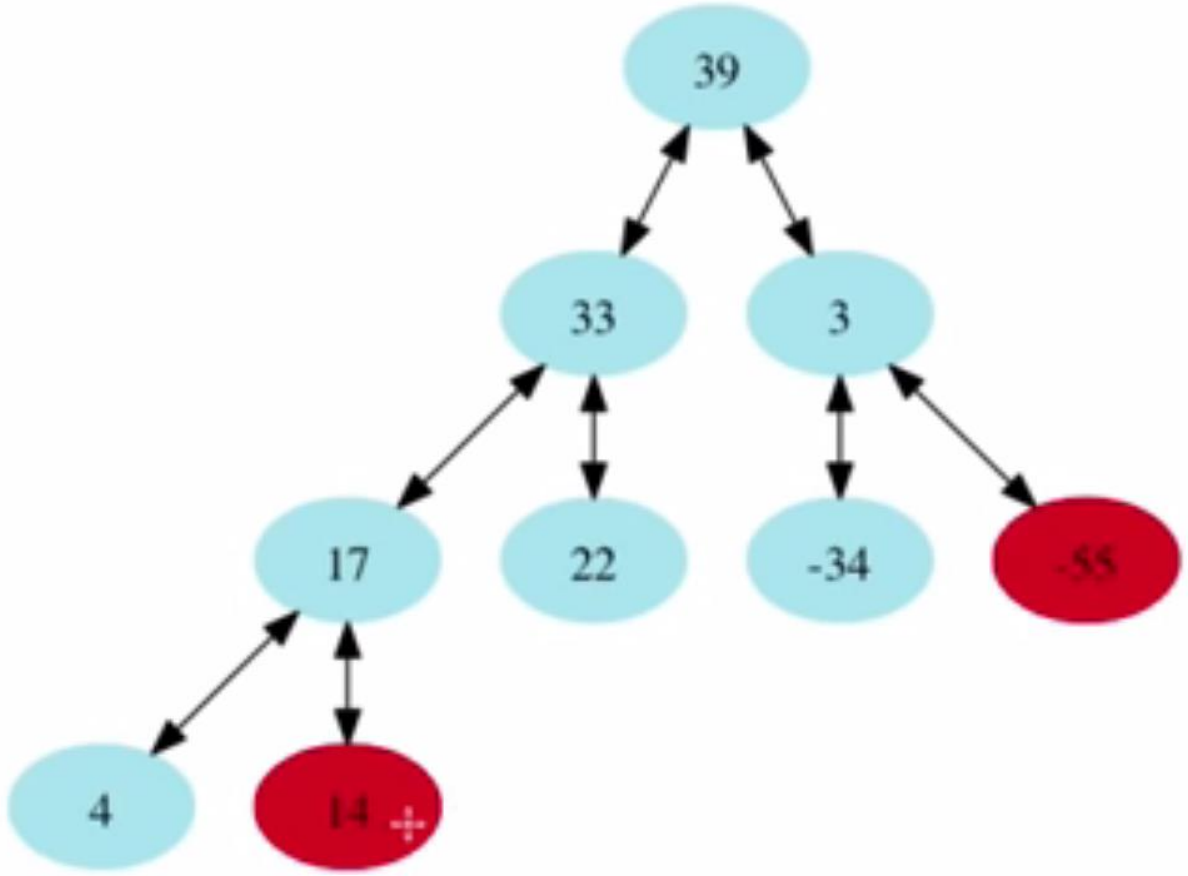
39	33	3	17	22	-34	-55	4	14					

Operaciones – Eliminar (Ejemplo 3)



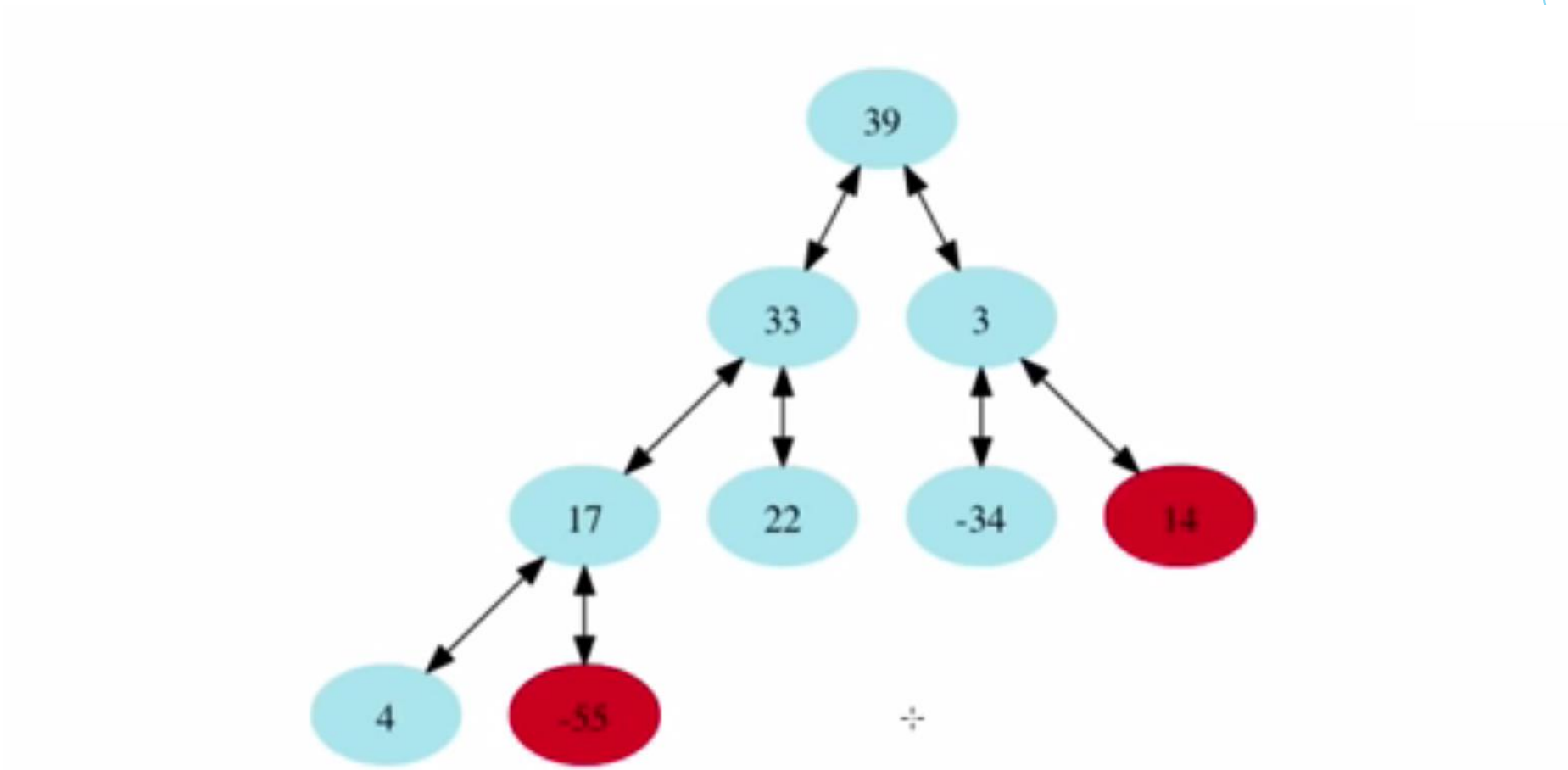
39	33	3	17	22	-34	-55	4	14					

Operaciones – Eliminar (Ejemplo 3)



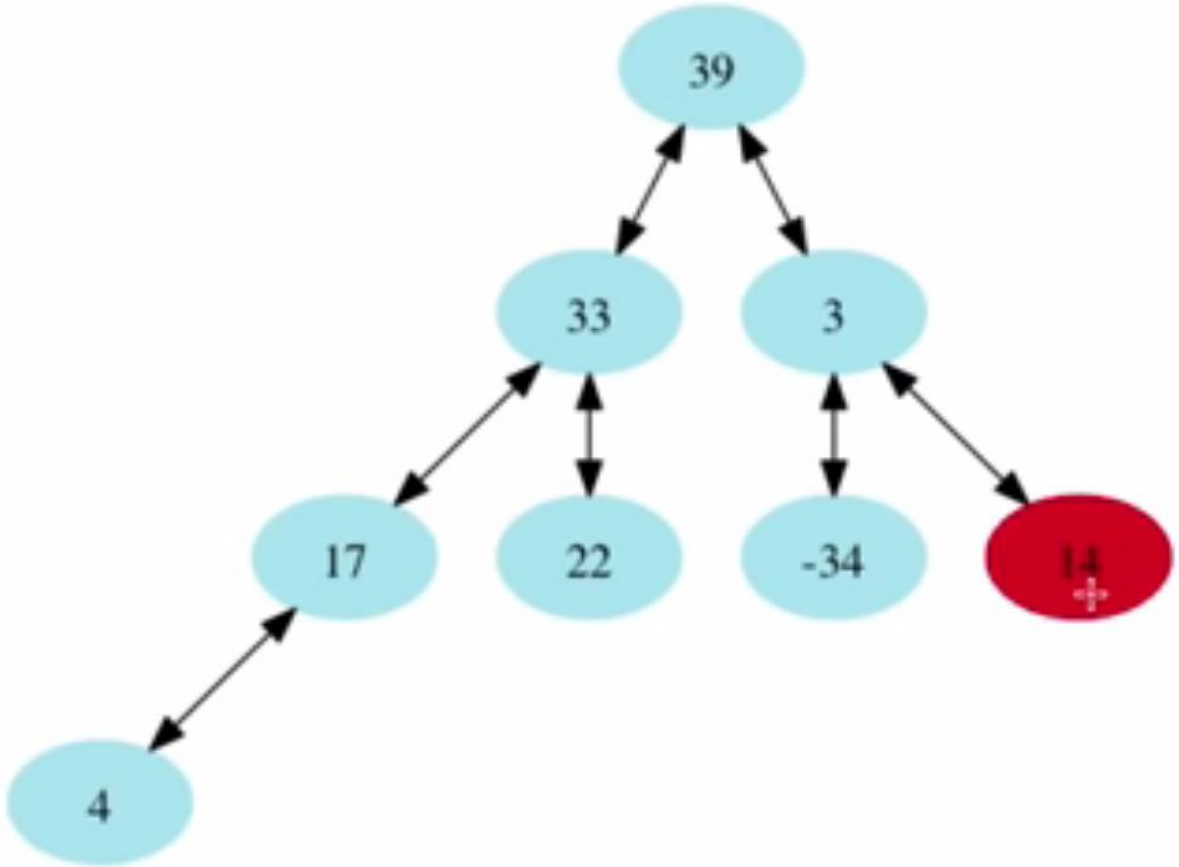
39	33	3	17	22	-34	-55	4	14					

Operaciones – Eliminar (Ejemplo 3)



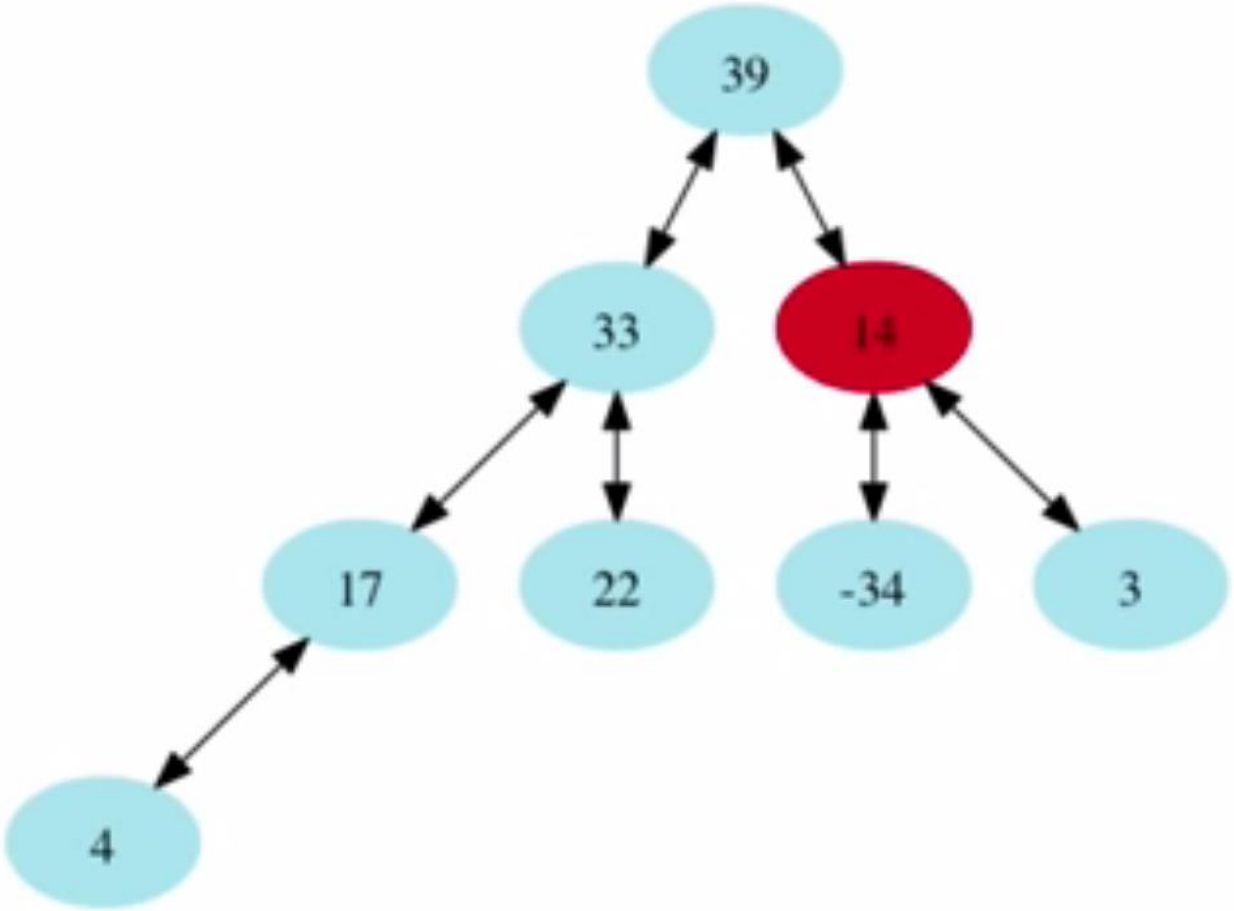
39	33	3	17	22	-34	14	4	-55					

Operaciones – Eliminar (Ejemplo 3)



39	33	3	17	22	-34	14	4						

Operaciones – Eliminar (Ejemplo 3)



39	33	14	17	22	-34	3	4						

GRACIAS