Arboles Generales

Es una colección de nodos que puede estar vacía o formada por un nodo distinguido(raíz), y un conjunto de subárboles

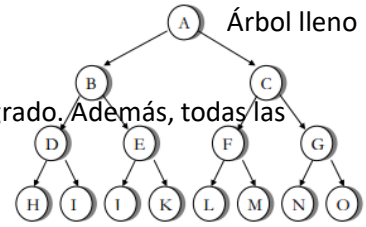
Camino: Es la secuencia de nodos desde n (mientras sea el padre de n*) hasta n*

Profundidad: Es la longitud desde la raíz hasta n. La raíz tiene profundidad 0

Altura: Es la longitud desde la raíz o el nodo dado hasta la hoja más lejana. Las hojas tienen altura 0

Grado: Es el grado del nodo con mayor cantidad de hijos

Árbol lleno

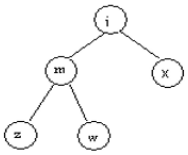


Árbol lleno: Dado un cierto grado, todos los nodos internos deben tener ese grado. Además, todas las hojas deben estar al mismo nivel.

Fórmula para calcular un árbol lleno: $(k^{h+1} - 1) / (k-1)$

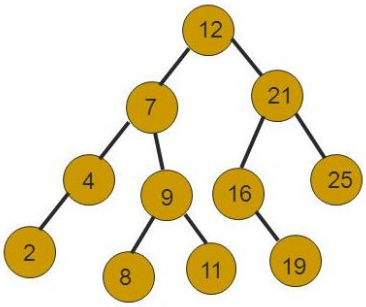
Árbol completo: Dado un árbol lleno hasta h-1 pero el ultimo nivel se va completando de izquierda a derecha.

Fórmula: $(k^{h} + k - 2) / (k-1)$ y $(k^{h+1} - 1)$



Recorridos:

Preorden: Se procesa primero la raíz y luego los hijos



Inorden: Se procesa el primer hijo, luego la raíz y después los demás hijos

Recorrido en Preorden

12, 7, 4, 2, 9, 8, 11, 21, 16, 19, 25

Recorrido en Inorden

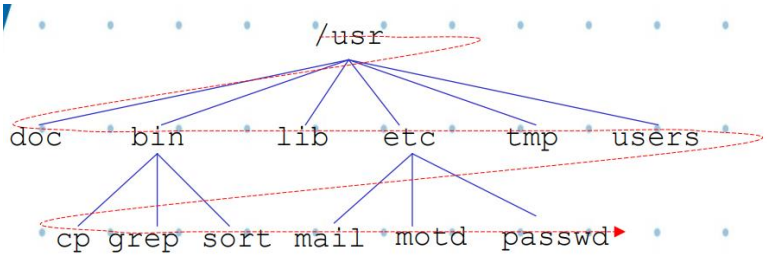
2, 4, 7, 8, 9, 11, 12, 16, 19, 21, 25

Postorden: Se procesan primero los hijos y luego la raíz

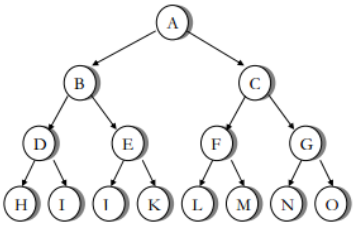
Recorrido en Postorden

2, 4, 8, 11, 9, 7, 19, 16, 25, 21, 12

Por niveles: Se procesan los nodos teniendo en cuenta sus niveles, primero la raíz, después sus hijos, nietos, etc.



Arboles Binarios



Lo mismo que el general nomás que en binario

Árbol lleno: $(2^{h+1}-1)$

Sea T un árbol binario completo de altura h, la cantidad de nodos N varía entre (2^{h}) y $(2^{h+1}– 1)$

Heap Binaria

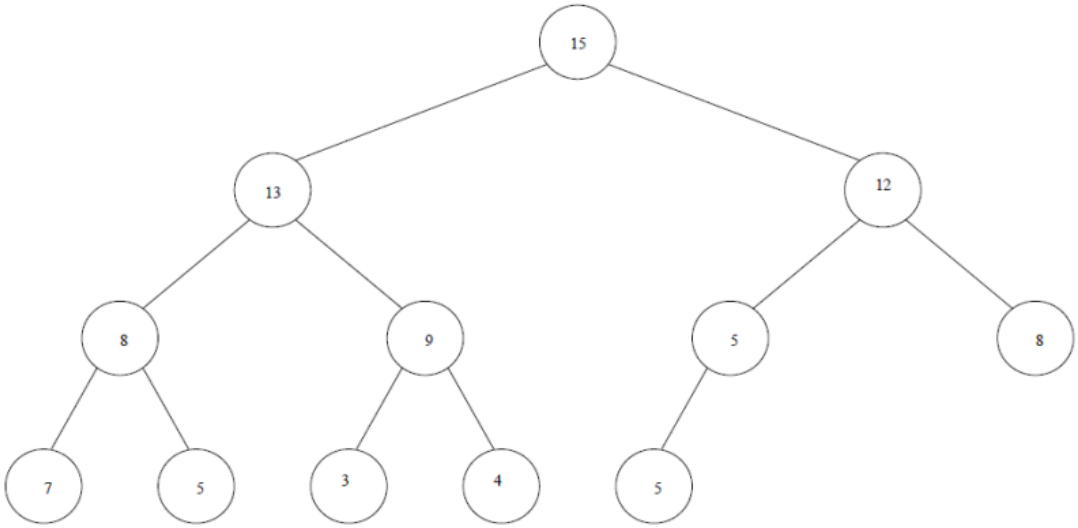
Heap Binaria: Es una cola de prioridad de una estructura de datos. Permite implementar colas de prioridad que no usan punteros y permite implementar operaciones con O(log N) operaciones en el peor caso.

Una heap es un árbol binario completo.

MinHeap:El elemento mínimo se almacena en la raíz. El dato almacenado en cada nodo es menor o igual al de sus hijos.

MaxHeap:El elemento máximo se almacena en la raíz. El dato almacenado en cada nodo es mayor o igual al de sus hijos.

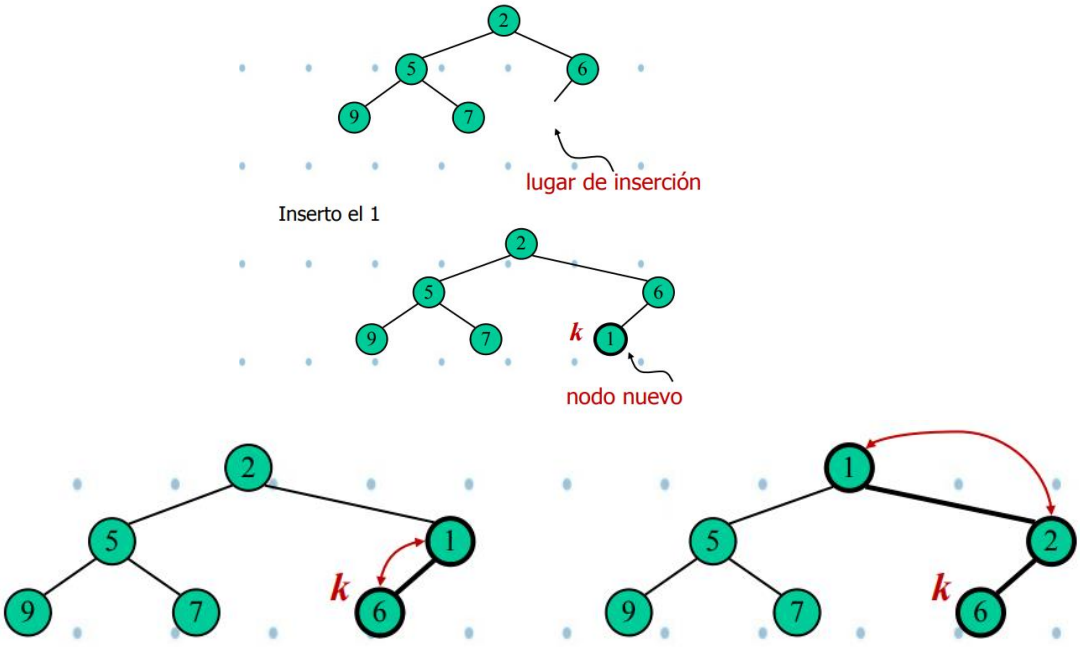
Ejemplo de MaxHeap:



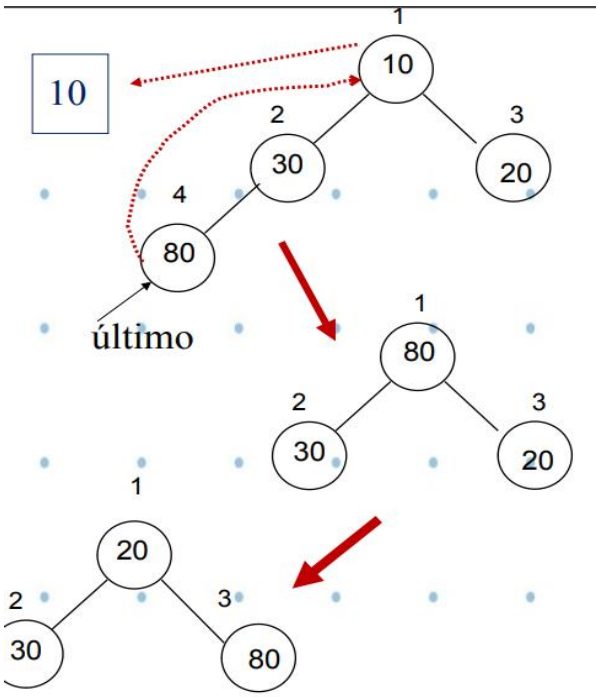
15	13	12	8	9	5	8	7	5	3	4	5
1	2	3	4	5	6	7	8	9	10	11	12

Una heap : Contiene un arreglo con los datos y un valor que indique el número de elementos almacenados.

Insert: El dato se inserta como último ítem de la heap y se debe hacer un filtrado hacia arriba para restaurar el orden. El filtrado termina cuando la clave k alcanza la raíz o nodo cuyo padre tiene una clave menor. El algoritmo recorre la altura de la heap que es O(log n) intercambios.



Delet: Se guarda el dato original de la raíz. Elimino el último elemento y lo almaceno en la raíz. Se filtra hacia abajo para restaurar el orden. Este se termina cuando se encuentra el lugar correcto donde insertarlo.



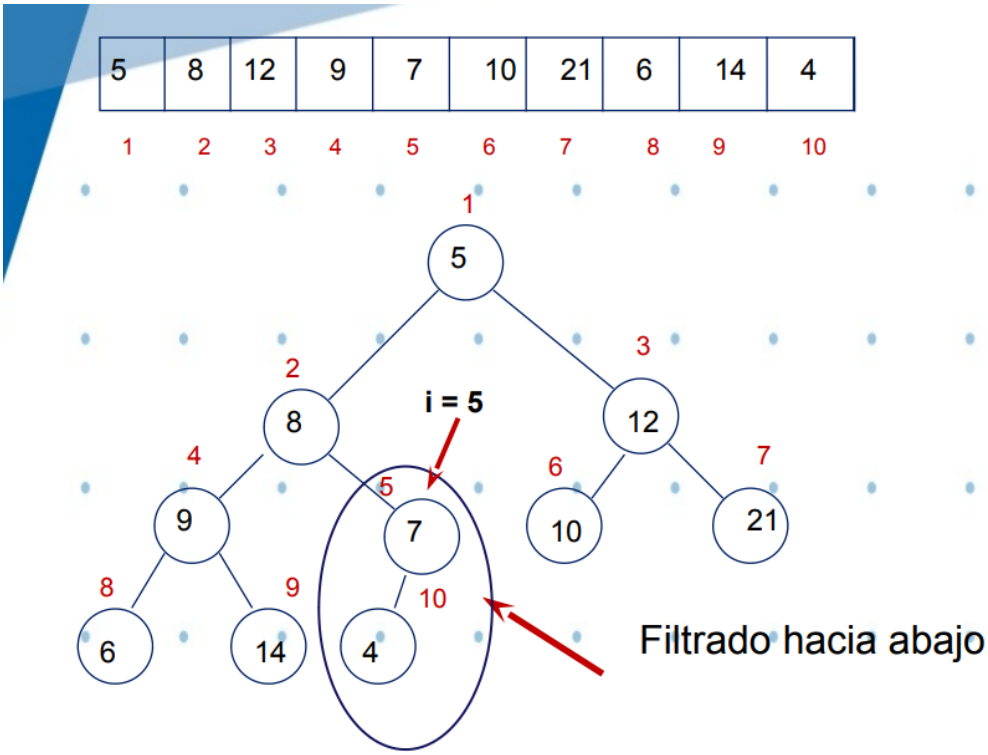
BuildHeap: Es para construir una heap con una lista de n elementos.

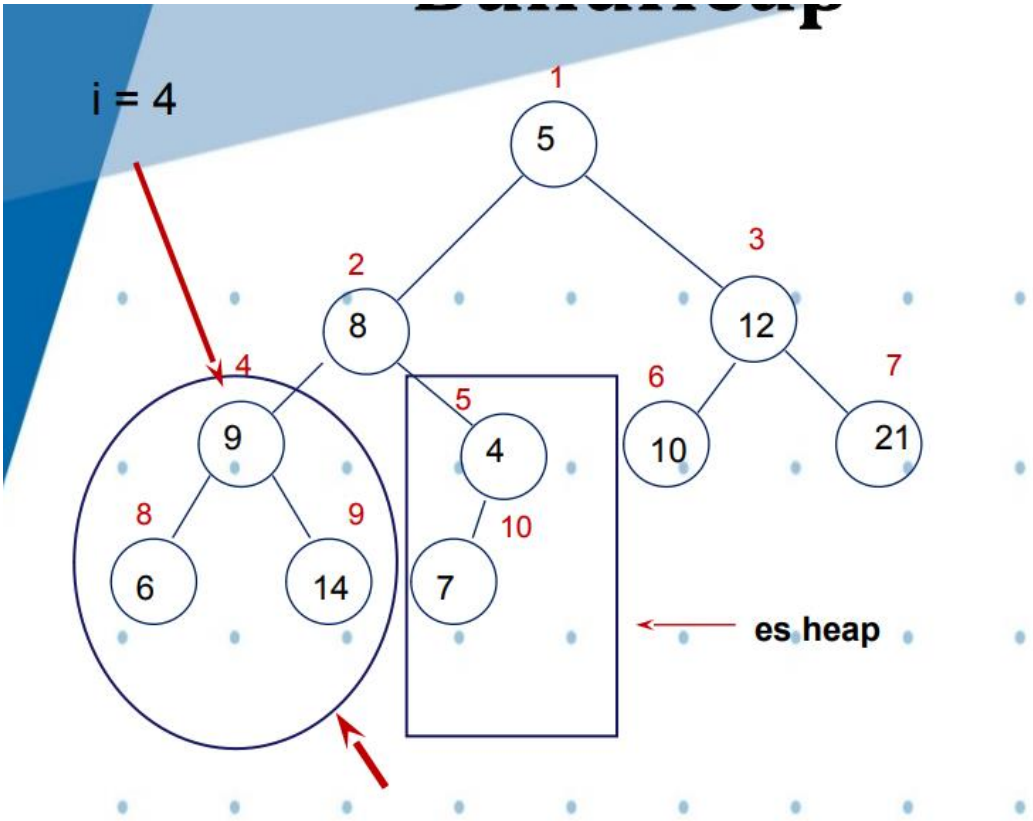
Se pueden insertar los elementos de a uno (n log n)(cantidad de operaciones)

Se puede usar un algoritmo de orden lineal, proporcional a los n elementos

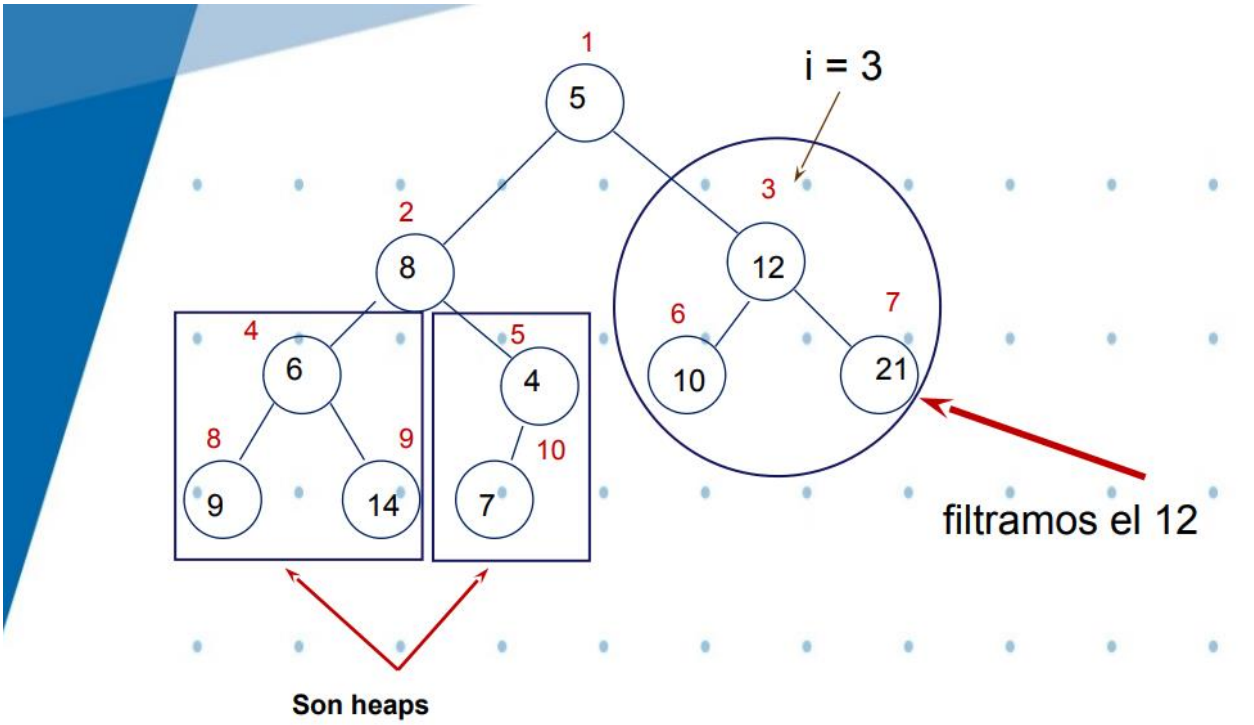
Para filtrar se elige el menor de los hijos, comparando el menor de los hijos con el padre. Se empieza filtrando el elemento que está en la posicion (tamaño/2), el resultado de dicha operación da la cantidad de nodos y la cantidad de hojas.

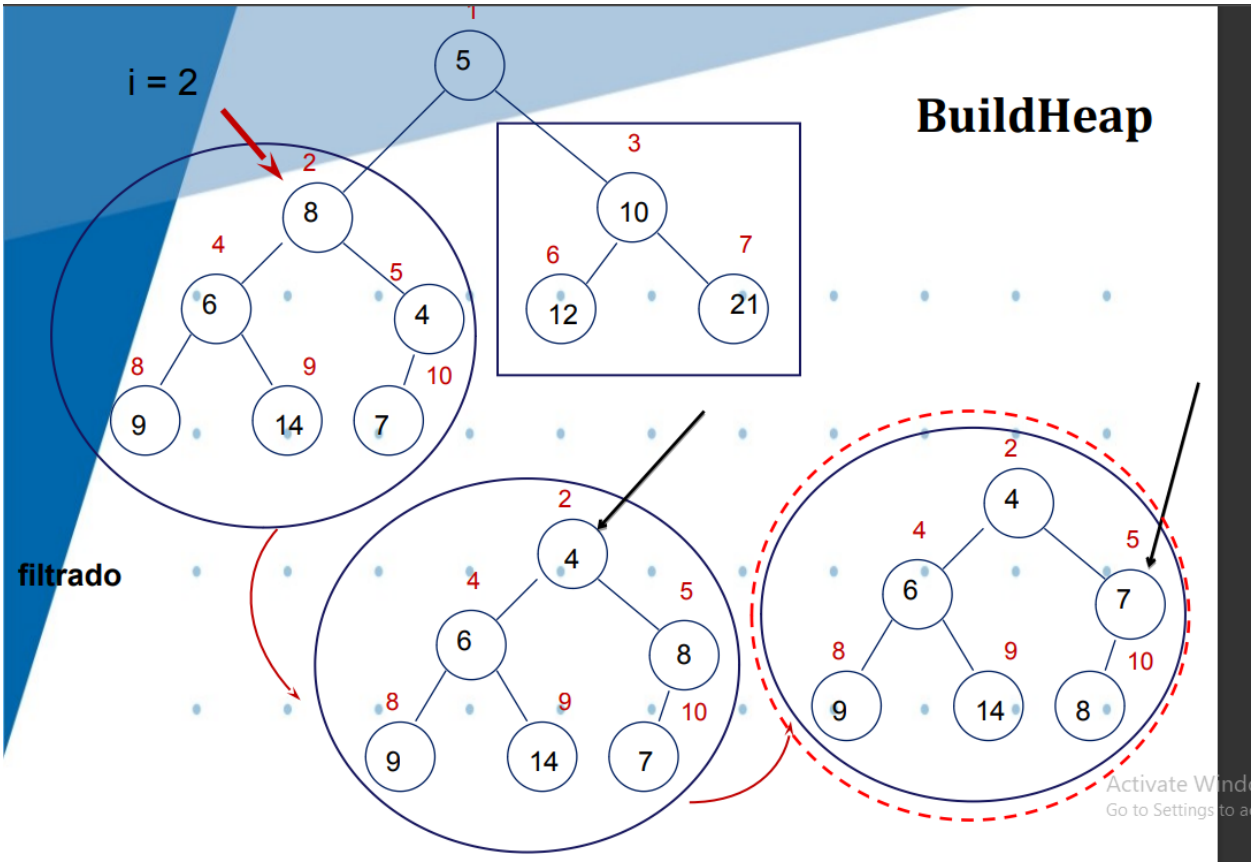
Es como una MinHeap pero tiene todos los elementos desordenados y hay que ordenarlos desde el resultado de tamaño/2, restandole -1 en cada operación.

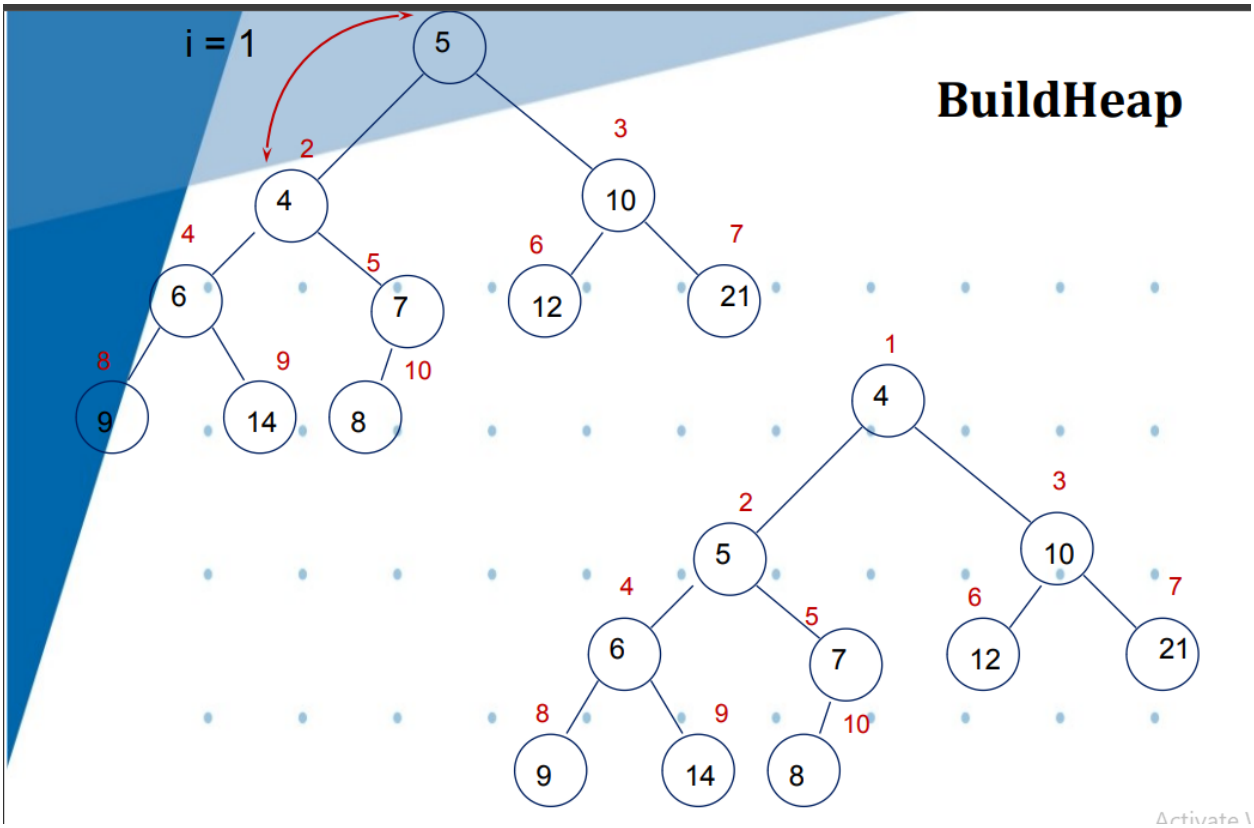




filtramos el 9


1
5
i = 3
2
8
3
12
6
7
4
5
6
10
21
8
9
10
9
14
7
filtramos el 12
Son heaps


1
5
i = 2
BuildHeap
2
8
3
10
4
6
5
4
6
7
8
9
10
9
14
7
12
21
filtrado
2
4
4
6
5
8
6
8
9
10
9
14
7
2
4
4
6
5
7
8
9
10
9
14
8



BuildHeap

Teorema:

En un árbol binario lleno de altura h que contiene $2^{h+1} - 1$ nodos, la suma de las alturas de los nodos es: $2^{h+1} - 1 - (h + 1)$

Demostración:

Un árbol tiene 2^i nodos de altura $h - i$

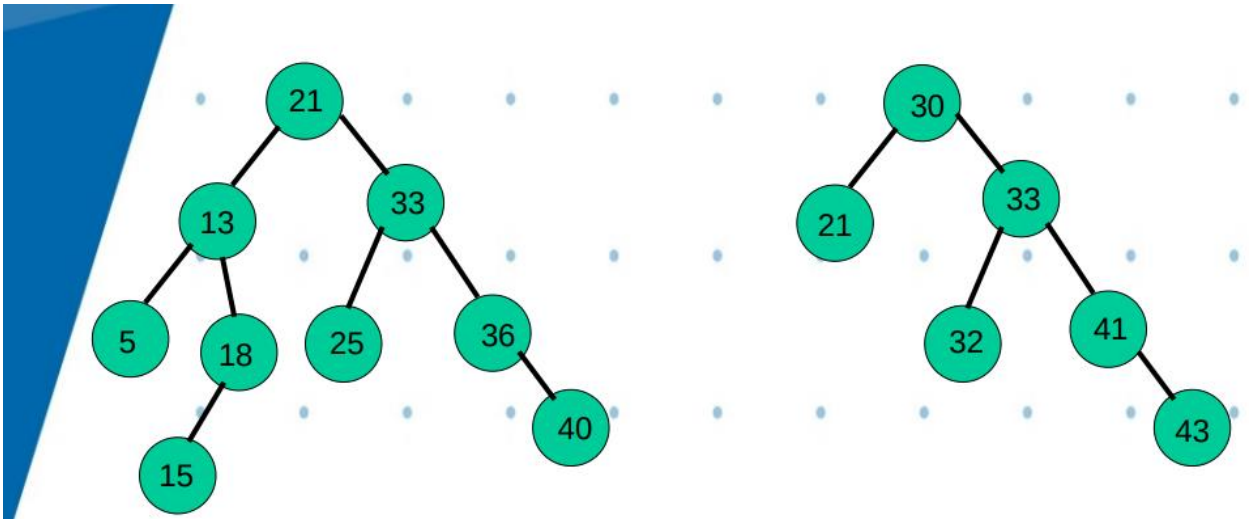
$$S = \sum_{i=0}^{h} 2^i (h-i)$$

$$S = h + 2\,(h-1) + 4\,(h-2) + 8\,(h-3) + \ldots\ldots 2^{h-1}\,(1)$$

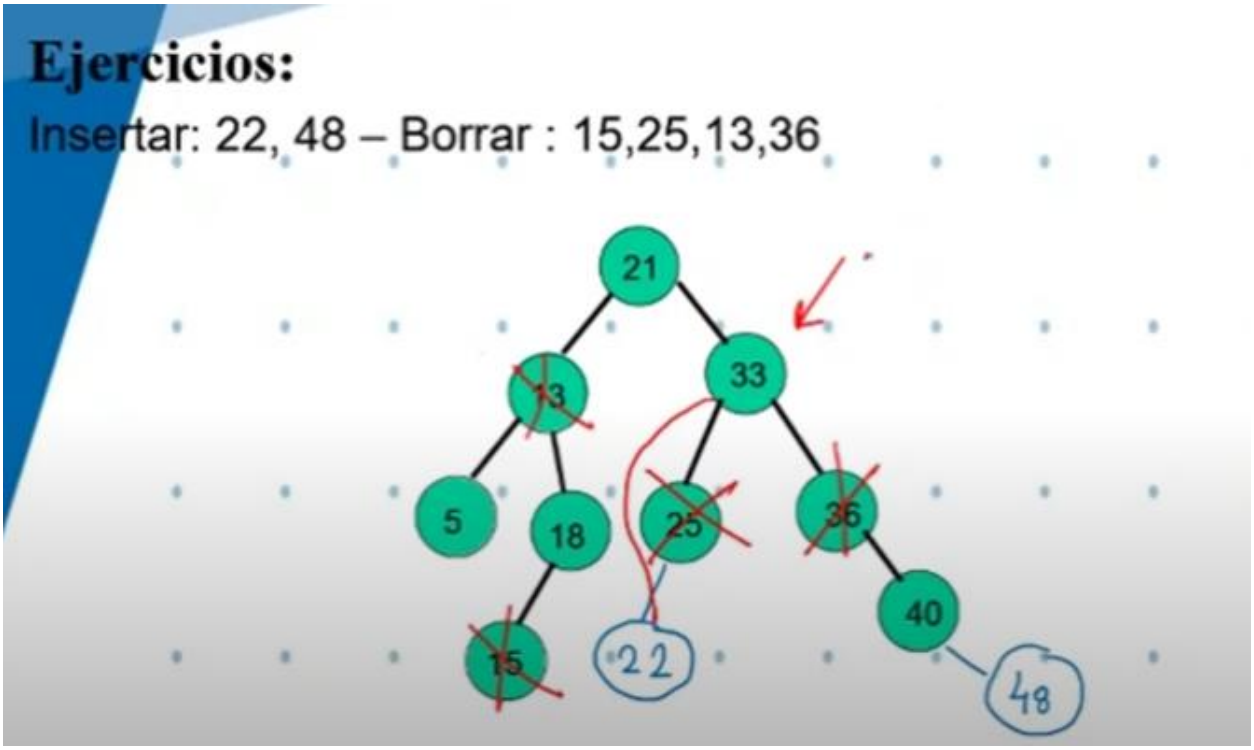
Árboles Binarios de búsqueda

Estructuralmente es un árbol binario.

La propiedad del orden es el siguiente: para cada nodo N del árbol, se tiene que cumplir que todos los nodos ubicados en el subárbol izquierdo contienen claves menores que la clave del nodo N y los nodos del subárbol derecho contiene claves mayores que la clave del nodo N



Insertar y eliminar elementos



Formula: $h \approx \log_2 n$

Árbol AVL

Es un árbol binario de búsqueda balanceado.

Un árbol está balanceado si la altura entre el subárbol izquierdo y el subárbol derecho es menor o igual a 1

Características: el balanceo garantiza que la altura del árbol es O(log n)

Al insertar elementos se hace de la misma manera que un ABB, se debe mantener el balanceo y en caso de que se desbalancee. En cada nodo se guarda información de la altura

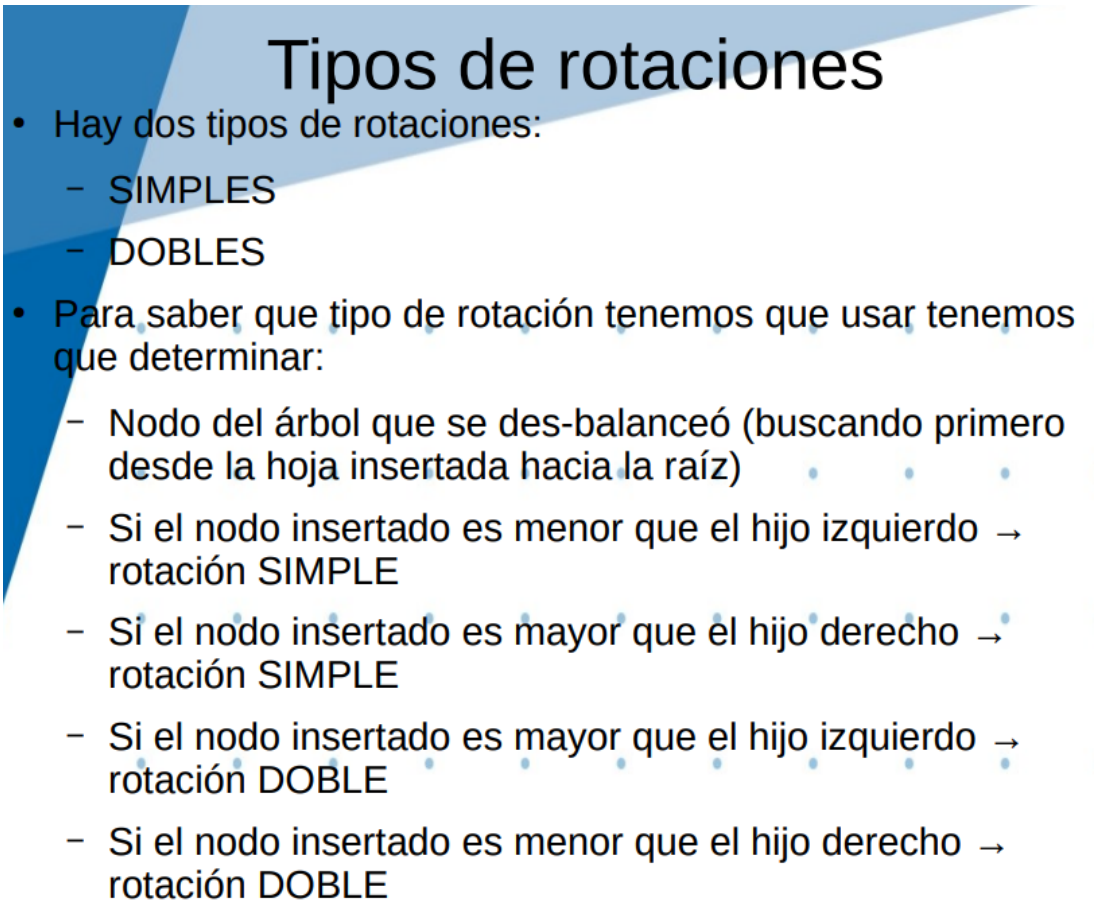
Para la inserción, Se controla y restaura el balanceo de esta forma:

Se recorre el camino de inserción en orden inverso, se controla el balanceo de cada nodo.

Si hay desbalanceo, se modifica el árbol mediante la rotación.

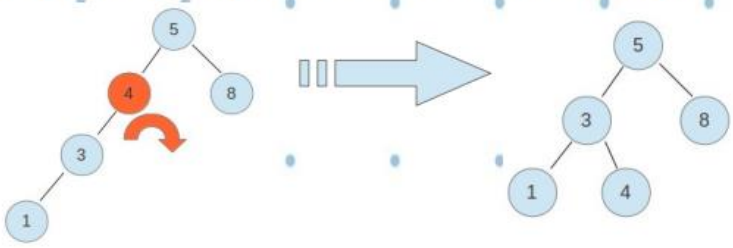
Se termina cuando todo el árbol queda balanceado

Tipos de rotaciones

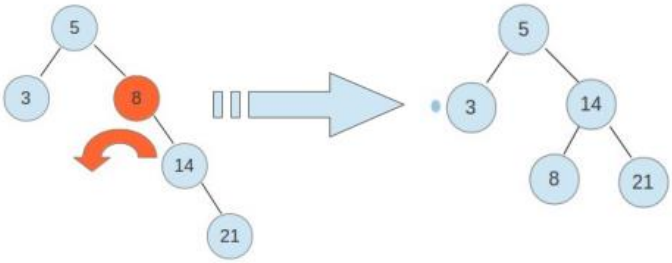


Rotaciones SIMPLES

- Se dan cuando:
  - El nodo insertado, por ejemplo el 1, es menor que el hijo izquierdo

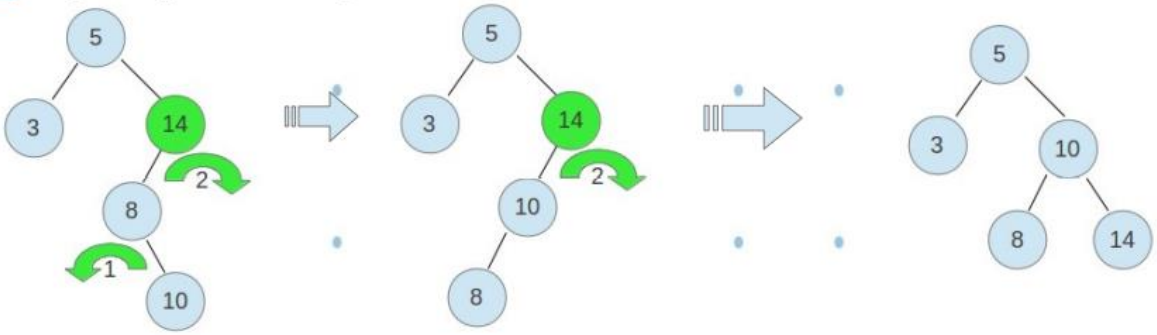


  - El nodo insertado, por ejemplo el 21, es mayor que el hijo derecho



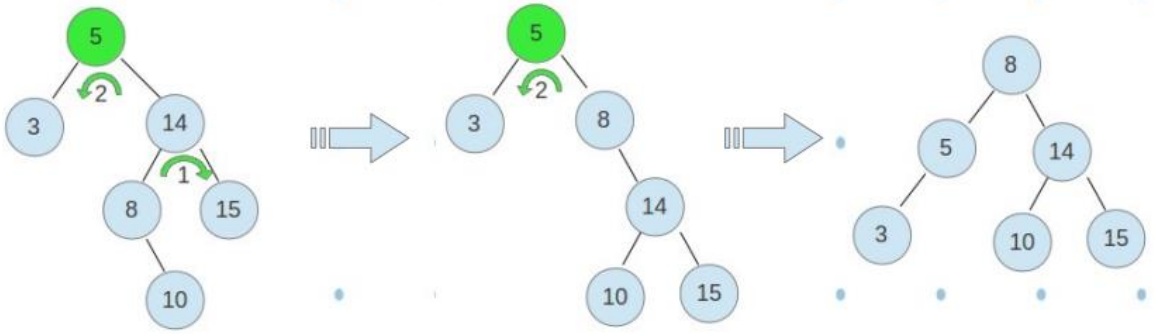
Rotaciones DOBLES

- Se dan cuando:
  - El nodo insertado es mayor que el hijo izquierdo
  - El nodo insertado es menor que el hijo derecho
- Requieren dos rotaciones simples, por ejemplo, luego de insertar el 10 tenemos:



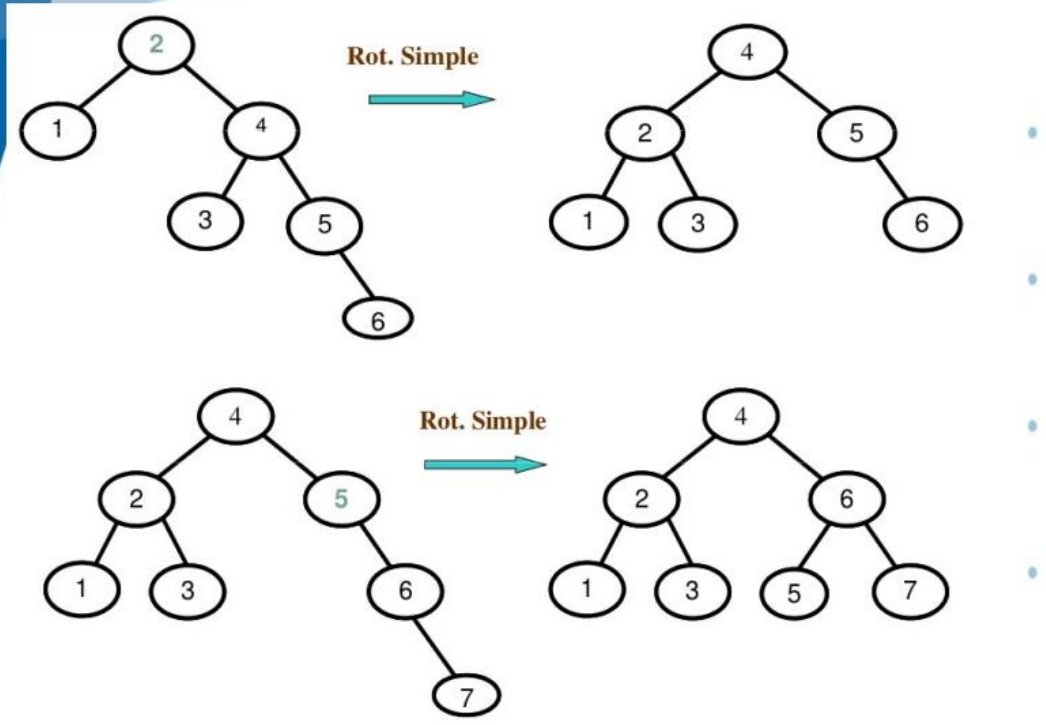
Rotaciones DOBLES

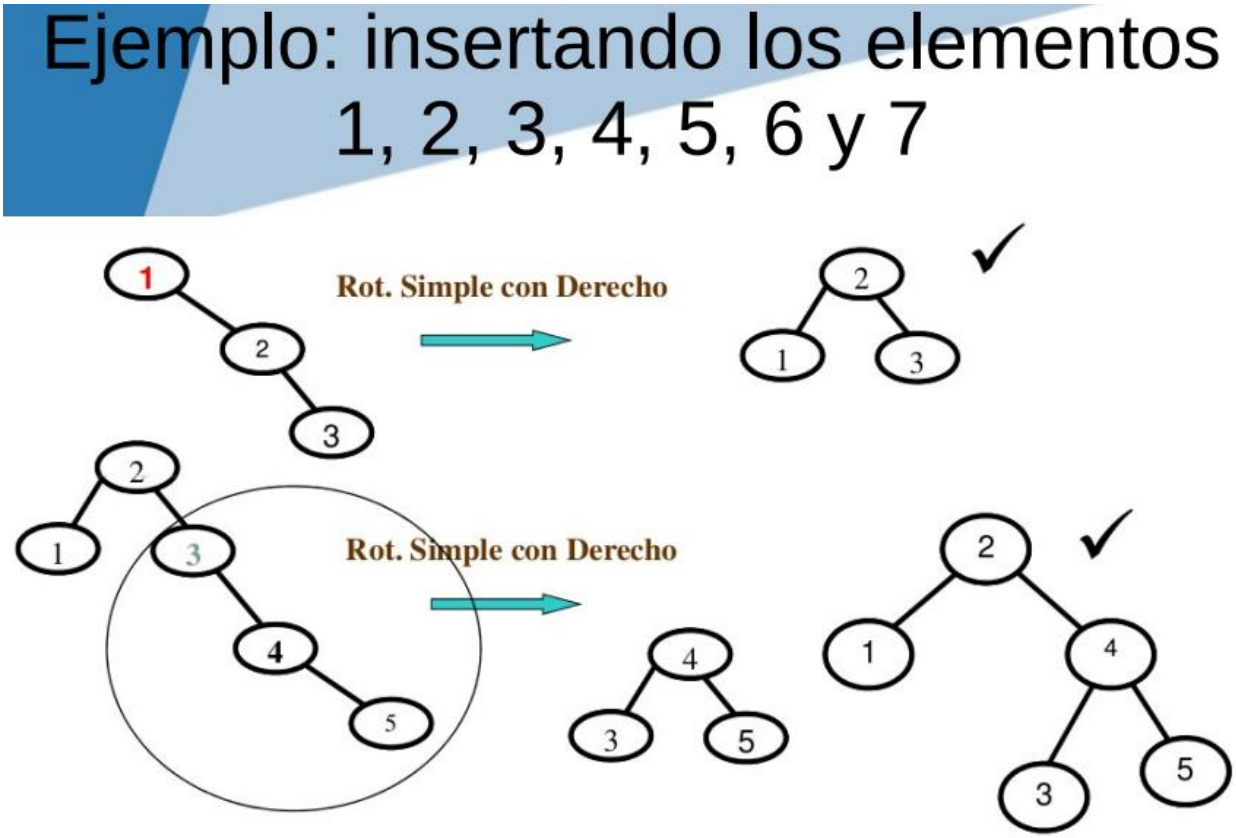
- Otro ejemplo: Al insertar el nodo 10, el árbol se desbalancea en el nodo 5



Inserción de elementos y balanceo

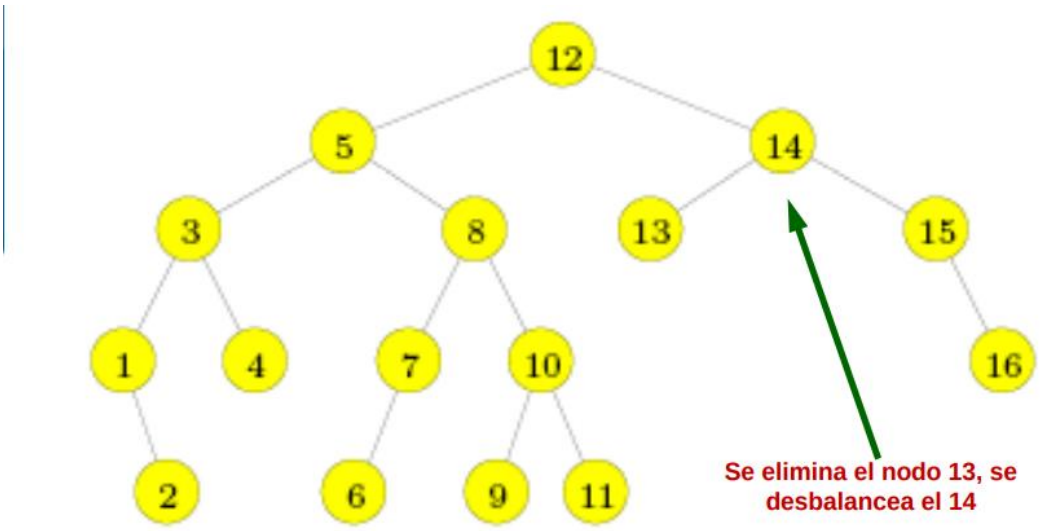
Ejemplo: insertando los elementos 1, 2, 3, 4, 5, 6 y 7

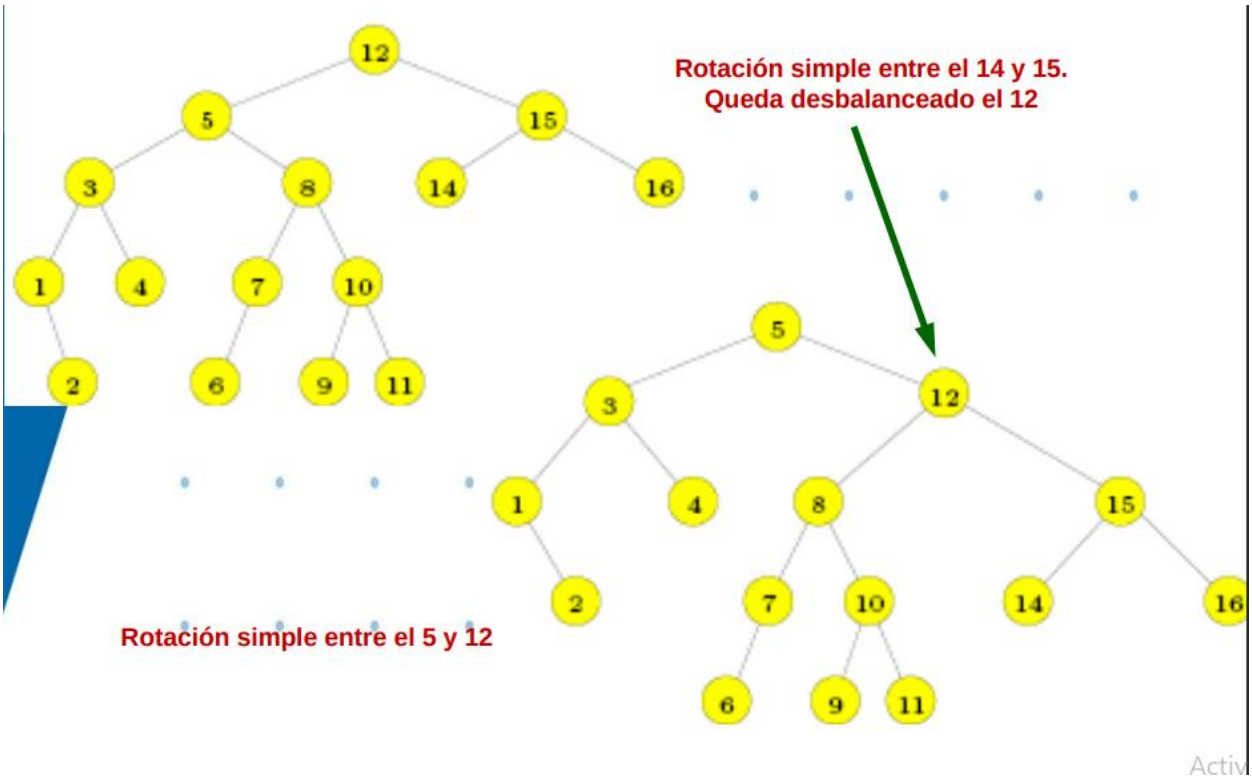




Eliminación y balanceo

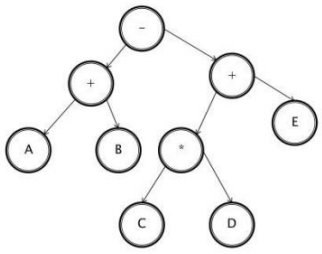
La eliminación de un nodo es similar al borrado en un árbol binario de búsqueda. Luego de realizar el borrado se debe actualizar la altura de todos los nodos, desde el nodo en cuestión hasta la raíz.





Algunos Ejercicios

Ejercicio 5
Dado el siguiente árbol de expresión:



Indique las expresiones aritméticas resultantes de realizar una impresión en pre-orden, in-orden y post-orden.

Pre-Orden: - + A B + * C D E

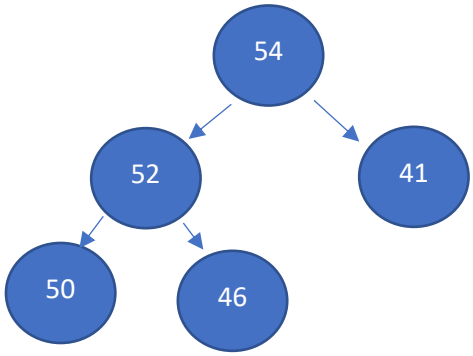
In-orden: A + B – C * D + E

Post-Orden: A B + C D * E + -

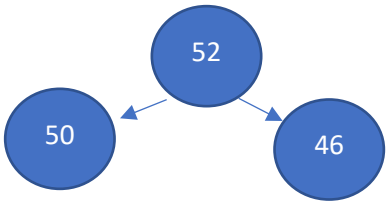
Ejercicio 6

Muestre las transformaciones que sufre una Max HEAP (inicialmente vacía) al realizar las siguientes operaciones:

a. Insertar 50, 52, 41, 54, 46
b. Eliminar tres elementos
c. Insertar 45, 48, 55, 43
d. Eliminar tres elementos.



Elimino 50, 54, 41



Inserto 45, 48, 55, 43