

OTRAS ESTRUCTURAS DE DATOS

CONTENIDO

- TABLAS HASH
- ARREGLOS ASOCIATIVOS
- COLAS DE PRIORIDAD

1

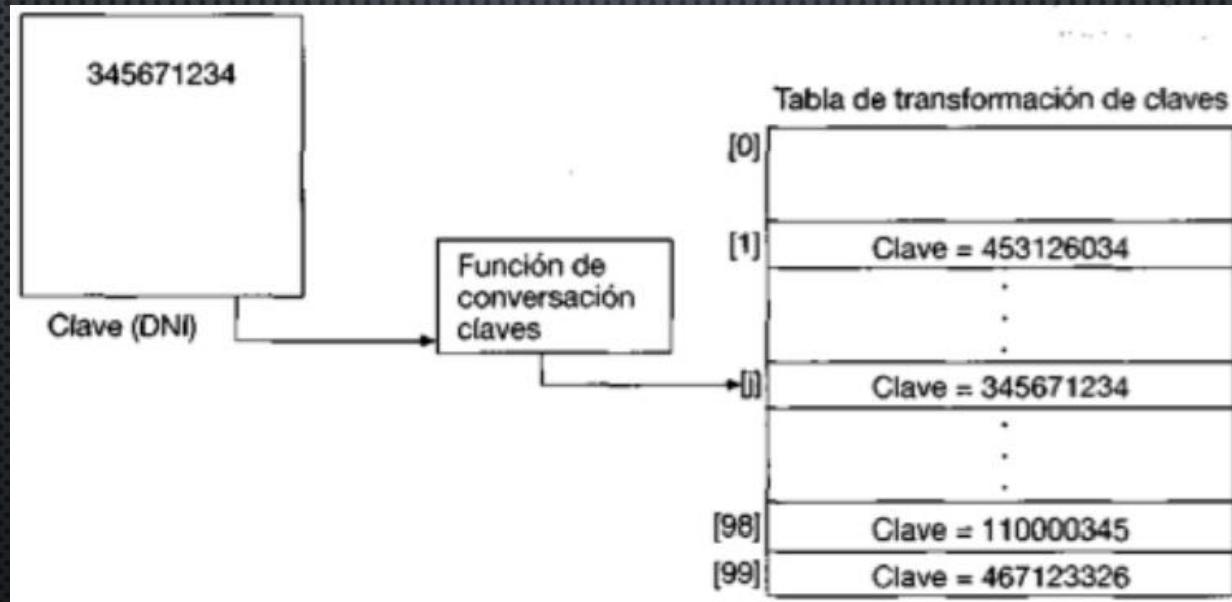
TABLAS HASH

CONCEPTO

TABLAS HASH

- ESTRUCTURA DONDE SE ALMACENAN PARES LLAVE, VALOR.
- SON UNA SOLUCIÓN AL PROBLEMA DE LA BÚSQUEDA YA QUE PERMITE ACCEDER DIRECTAMENTE A LA UBICACIÓN DONDE SE ENCUENTRE NUESTRO DATO, SIN TENER QUE REVISAR LAS OTRAS POSICIONES.
- SE DICE QUE LA UBICACIÓN ES COMPUTADA EN BASE AL VALOR DE LA LLAVE.

TABLAS HASH



TABLAS HASH

- EXISTEN CASOS DONDE ES SENCILLO ENCONTRAR UNA FUNCIÓN DE CONVERSIÓN.
- EXISTEN OTROS CASOS DONDE NO ES POSIBLE ESTABLECER UNA RELACIÓN TAN DIRECTA

TABLAS HASH

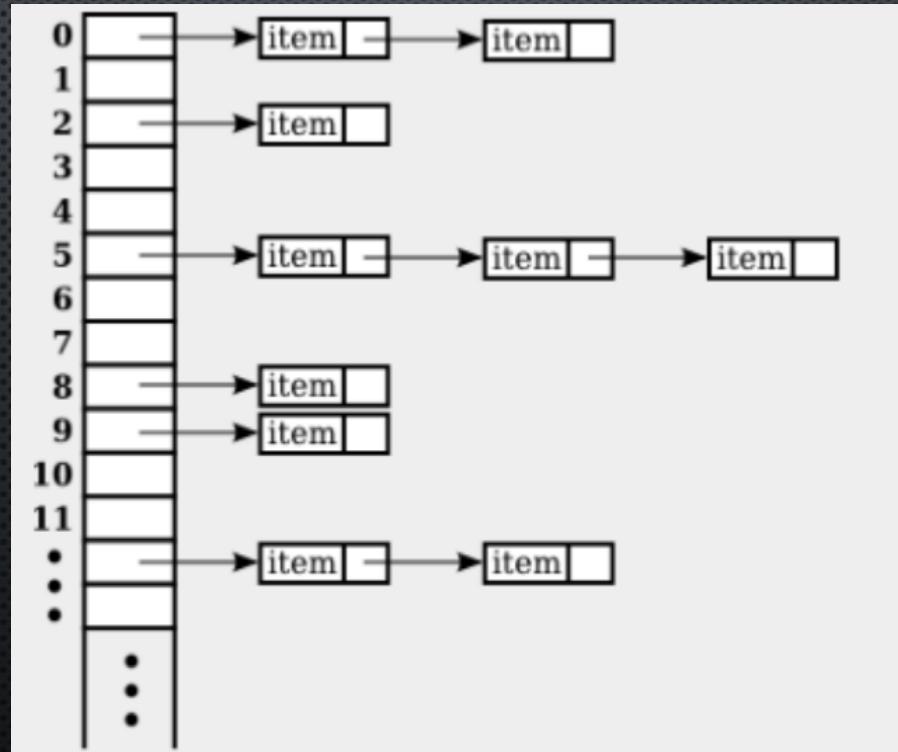
- ALMACENAN LA INFORMACIÓN EN ARREGLOS.
- EL ÍNDICE DEL ARREGLO NO ES IGUAL A LA LLAVE, PERO SE PUEDE CALCULAR A PARTIR DE ÉSTA. SI FUERAN IGUALES HABRÍAN CASOS DONDE EL ARREGLO SERÁ DE TAMAÑO INFINITO PARA PODER ALOJAR TODAS LAS DISTINTAS CLAVES.

TABLAS HASH

- CUANDO DOS O MÁS CLAVES TIENE EL MISMO CÓDIGO HASH DECIMOS QUE EXISTE UNA **COLISIÓN**.
 - Es parte del funcionamiento normal de una tabla hash, no es un error.
- La solución más común para el manejo de colisiones es que cada posición del arreglo a su vez sea una lista enlazada.



TABLAS HASH



2

ARREGLOS ASOCIATIVOS

MAPS & DICTIONARIES

ARREGLOS

- PODEMOS ACCEDER A CUALQUIER ELEMENTO POR MEDIO DE OPERACIONES “OBTENCIÓN” (GET) Y “ASIGNACIÓN” (PUT)
- ESTAS OPERACIONES SE MANEJAN A TRAVÉS DE UN NÚMERO ENTERO “I” QUE ESTARÁ EN UN RANGO DE 0 A N
 - ARREGLO[0]
 - ARREGLO[1]
 - ...
 - ARREGLO[N-1]

ARREGLOS ASOCIATIVOS

- ABSTRACCIÓN DEL CONCEPTO TRADICIONAL DE ARREGLOS
- TAMBIÉN CUENTA CON OPERACIONES “GET” Y “PUT”
- ESTAS OPERACIONES NO SE MANEJAN A TRAVÉS DE NÚMEROS ENTEROS,
SE MANEJAN POR MEDIO DE UN OBJETO DE TIPO T

DICTIONARY

```
var dictionary = new Dictionary<string, Student>();
dictionary.Add("EST1174908", new Student
{
    Id = 1174908,
    Name = "Pablo",
    LastName = "Godoy",
    Faculty = "Ingeniería"
});

dictionary.Add("EST1100508", new Student
{
    Id = 1100508,
    Name = "Diana",
    LastName = "Gutiérrez",
    Faculty = "Ingeniería"
});

dictionary.Add("EST12121212", new Student
{
    Id = 12121212,
    Name = "Test",
    LastName = "Prueba",
    Faculty = "Arquitectura"
});
```

Immediate Window

```
dictionary
Count = 3
[0]: {[EST1174908, 1174908 | Pablo Godoy | Ingeniería]}
[1]: {[EST1100508, 1100508 | Diana Gutiérrez | Ingeniería]}
[2]: {[EST12121212, 12121212 | Test Prueba | Arquitectura]}
```

3

COLAS DE PRIORIDAD

ARREGLO DESORDENADO, ARREGLO
ORDENADO Y MONTÍCULO (HEAP)

COLAS DE PRIORIDAD

- EXISTEN CASOS DONDE SE ORDENAN LOS ELEMENTOS SEGÚN CIERTA PRIORIDAD:
 - PROCESOS EN UN SO: LOS PROCESOS DE SISTEMA TIENEN MAYOR PRIORIDAD DE EJECUCIÓN QUE LOS PROCESOS DE USUARIO
 - ENRUTAMIENTO DE PAQUETES EN REDES: LOS PAQUETES DE PROTOCOLO “VOICE OVER IP” TIENEN PRIORIDAD SOBRE LOS DE DATOS REGULARES
 - COLAS DE BANCO: LOS ANCIANOS Y MUJERES EMBARAZADAS TIENEN PRIORIDAD

COLAS DE PRIORIDAD

- SON UNA GENERALIZACIÓN DE PILAS Y COLAS.
- EN LUGAR DE INSERTAR Y ELIMINAR ELEMENTOS EN UN ORDEN PREDETERMINADO, A CADA ELEMENTO SE LE ASIGNA UNA PRIORIDAD REPRESENTADA POR UN ENTERO.
- EL INGRESO SE REALIZARÁ EN BASE A LA PRIORIDAD
- LA ELIMINACIÓN SIEMPRE CORRESPONDERÁ AL ELEMENTO CON UNA MAYOR PRIORIDAD (EL VALOR MÁS PEQUEÑO)

COLAS DE PRIORIDAD

- ÉSTAS TIENEN UN TAMAÑO MÁXIMO PREDETERMINADO, COMO MEDIDA DE SEGURIDAD PARA EVITAR SOBRECARGAS EN EL SISTEMA EN QUE SE ESTÉ USANDO LA COLA (POR EJEMPLO LOS ATAQUES DOS)
- SE PUEDEN IMPLEMENTAR DE VARIAS FORMAS:
 - ▶ ARREGLO DESORDENADO
 - ▶ ARREGLO ORDENADO
 - ▶ HEAP

COLAS DE PRIORIDAD

CUENTAN CON LAS SIGUIENTES OPERACIONES:

- **VACÍO – *IsEmpty*:** SABER SI LA COLA ESTÁ VACÍA O NO
- **AÑADIR(PRIORIDAD) – *Add*:** AÑADE UN NUEVO ELEMENTO PARA EL CUAL SE ESPECIFICA LA PRIORIDAD
- **REMOVER – *Remove*:** REMUEVE EL ELEMENTO CON LA MÁS ALTA PRIORIDAD
- **CONSULTA – *Peek*:** VER EL ELEMENTO CON LA MÁS ALTA PRIORIDAD

ARREGLO DESORDENADO

- SE DEBE TENER UN ARREGLO DEL TAMAÑO LÍMITE DEFINIDO PREVIAMENTE, EL CUAL SE MANTENDRÁ DESORDENADO EN TODO MOMENTO.
- SE DEBERÁ TENER UNA VARIABLE QUE APUNTE A LA ÚLTIMA POSICIÓN DISPONIBLE EN EL ARREGLO (N).

ARREGLO DESORDENADO

■ LA INSERCIÓN CONSISTE EN AGREGAR EL NUEVO ELEMENTO EN LA POSICIÓN n Y AUMENTAR EN 1 EL VALOR DE n . ES UNA OPERACIÓN DE TIEMPO CONSTANTE $O(1)$.

■ LA ELIMINACIÓN CONSISTE EN UBICAR EL ELEMENTO MÁS PEQUEÑO DEL LISTADO CON UNA BÚSQUEDA SECUENCIAL, INTERCAMBIARLO POR EL ELEMENTO EN $n-1$ Y DECREMENTAR EN 1 A n . TIENE UN COSTO LINEAL DEBIDO A LA BÚSQUEDA $O(n)$.

ARREGLO ORDENADO

- SE DEBE TENER UN ARREGLO DEL TAMAÑO LÍMITE DEFINIDO PREVIAMENTE, EL CUAL MANTENDREMOS ORDENADO EN TODO MOMENTO.
- SE DEBEN TENER DOS VARIABLES, UNA QUE APUNTE A LA PRIMERA POSICIÓN DISPONIBLE EN EL ARREGLO Y LA OTRA A LA ÚLTIMA.
- LA INSERCIÓN CONSISTE EN AGREGAR EL NUEVO ELEMENTO EN LA POSICIÓN CORRESPONDIENTE.

ARREGLO ORDENADO - INSERCIÓN

- SE UBICA LA POSICIÓN PARA INSERTAR EL ELEMENTO POR MEDIO DE UNA BÚSQUEDA BINARIA. $O(\log n)$
- SE CORREN LOS ELEMENTOS POSTERIORES PARA DEJAR UN ESPACIO PARA EL NUEVO ELEMENTO. $O(n)$
- SE INSERTA EL ELEMENTO $O(1)$.

ARREGLO ORDENADO - ELIMINACIÓN

- CONSISTE EN TOMAR (ELIMINAR) EL ELEMENTO INDICADO POR EL PRIMER APUNTADOR
- INCREMENTAR EL APUNTADOR EN 1 $O(1)$.

COMPARACIÓN DE MÉTODOS

■ AL COMPARAR LOS DOS MÉTODOS VISTOS NOS DAMOS CUENTA QUE CADA UNO TIENE UN PUNTO EN CONTRA Y OTRO A FAVOR, POR LO QUE SURGE EL HEAP COMO UNA ESTRUCTURA QUE MANTIENE EL BALANCE EN LAS DOS OPERACIONES BÁSICAS.

Método	Inserción	Eliminación del mínimo
Arreglo ordenado	$O(n)$	$O(1)$
Arreglo desordenado	$O(1)$	$O(n)$
Heap	$O(\log n)$	$O(\log n)$

HEAP

- TAMBIÉN LLAMADO “MONTÍCULO”, ES UNA ESTRUCTURA DE DATOS BASADA EN UN ÁRBOL BINARIO BALANCEADO QUE PERMITE LA IMPLEMENTACIÓN DE COLAS DE PRIORIDAD.
- SE DICE QUE EL ELEMENTO MÁS PRIORITARIO (VALOR MÁS PEQUEÑO) SIEMPRE SE ENCUENTRA EN LA RAÍZ
- DEBE DE CUMPLIR CON DOS CONDICIONES BÁSICAS:
 - ▷ ORDEN INVARIANTE DE HEAP
 - ▷ FORMA INVARIANTE

ORDEN INVARIANTE HEAP

- EL ORDEN INVARIANTE GARANTIZA QUE EL NODO CON MAYOR PRIORIDAD SE ENCUENTRA EN LA RAÍZ.
- EXISTEN 2 ALTERNATIVAS DE ANALIZAR ESTA CONDICIÓN:
 - ALTERNATIVA 1: LA LLAVE DE UN NODO EN EL ÁRBOL ES MENOR O IGUAL QUE TODAS LAS LLAVES DE SUS HIJOS.
 - ALTERNATIVA 2: LA LLAVE DE UN NODO EN EL ÁRBOL ES MAYOR O IGUAL A SU PADRE, EXCEPTO LA RAÍZ.

ORDEN INVARIANTE HEAP

- LAS DOS OPCIONES SON EQUIVALENTES, SON DOS FORMAS DE VER LA MISMA SITUACIÓN.
- AMBOS CASOS IMPLICAN QUE EL MÍNIMO ELEMENTO DEL HEAP SE ENCUENTRA EN LA RAÍZ.
- SEGÚN EL PROBLEMA QUE ESTÉ ANALIZANDO, ASÍ ELEGIREMOS UN PUNTO DE VISTA MÁS ADECUADO PARA LA IMPLEMENTACIÓN.

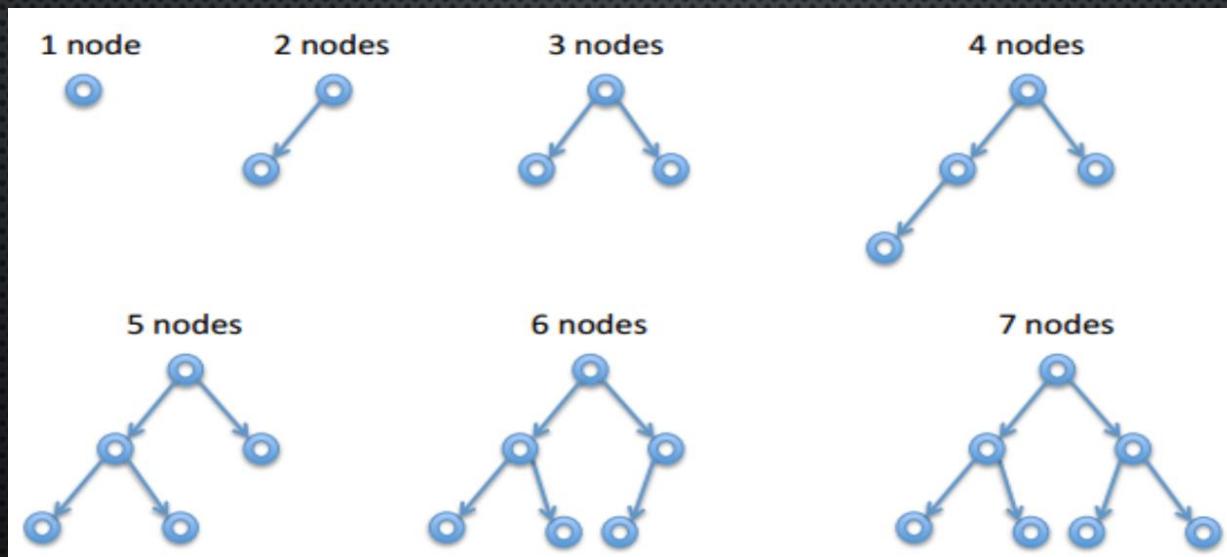
FORMA INVARIANTE

- ES LA OTRA RESTRICCIÓN QUE CUMPLE EL HEAP.
- LA FORMA DE UN HEAP ESTÁ COMPLETAMENTE DETERMINADA POR LA CANTIDAD DE ELEMENTOS QUE CONTIENE.
- ESTA RESTRICCIÓN INDICA QUE EL ÁRBOL SE LLENA NIVEL POR NIVEL DE IZQUIERDA A DERECHA.

INSERCIÓN

■ DEBEN CUMPLIRSE LAS RESTRICCIONES:

- ORDEN INVARIANTE
- FORMA INVARIANTE

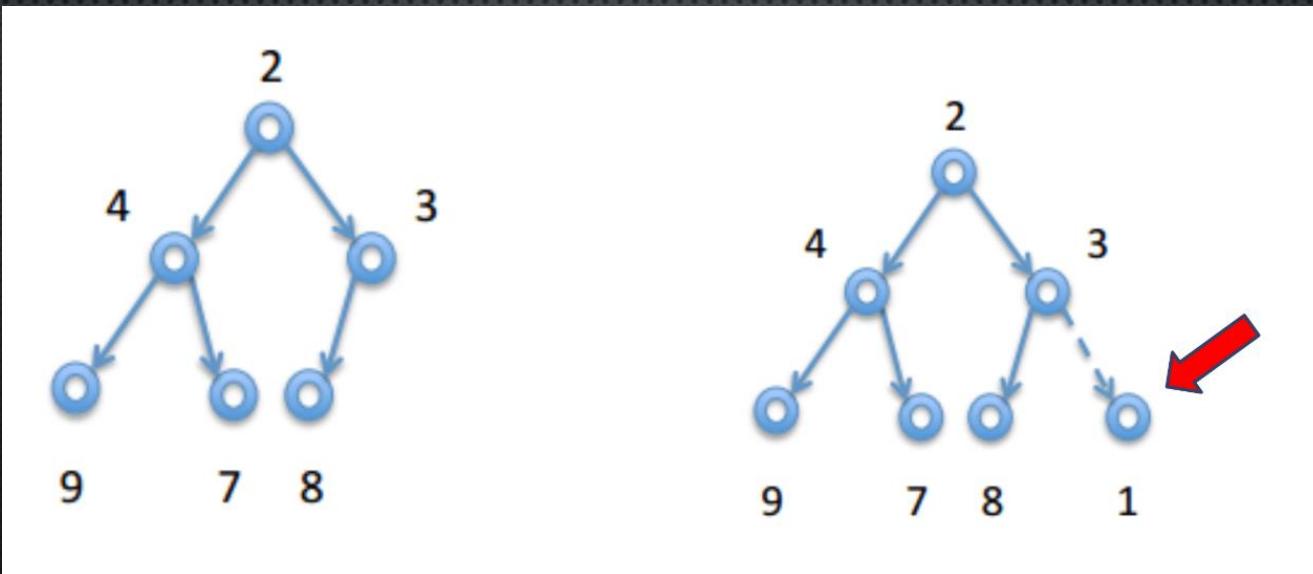


INSERCIÓN

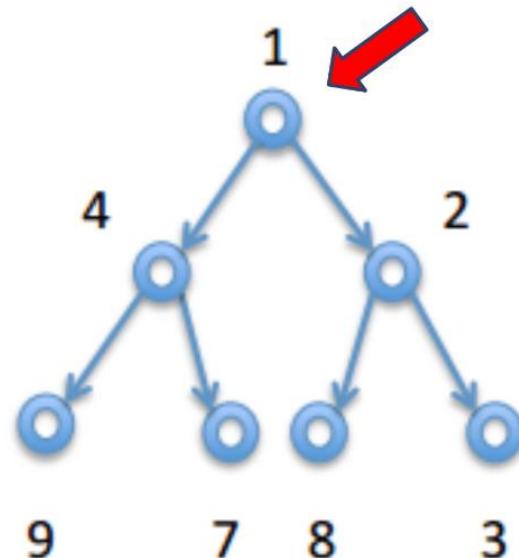
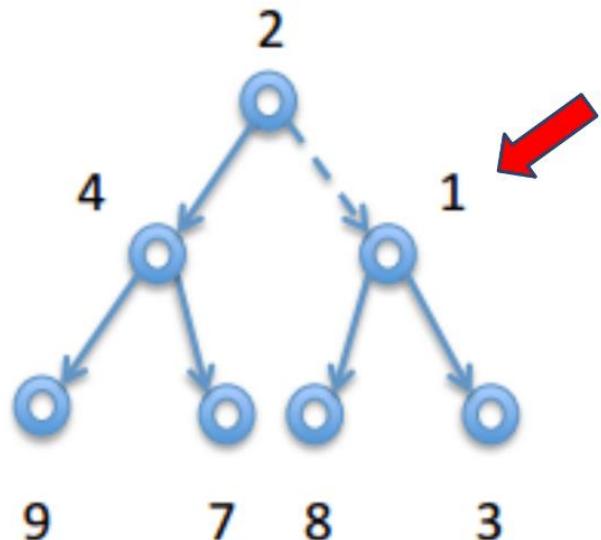
- CONSTA DE LOS SIGUIENTES PASOS
 - INSERTAR EL ELEMENTO EN LA POSICIÓN QUE CORRESPONDA (SEGÚN FORMA INVARIANTE)
 - SI AL INSERTAR EL ELEMENTO, ÉSTE VIOLA LA CONDICIÓN DE ORDEN INVARIANTE SE DEBE SUSTITUIR CON EL NODO PADRE.
 - SE REPETIR EL PASO ANTERIOR HASTA QUE SE CUMPLA LA CONDICIÓN DE ORDEN INVARIANTE O SE LLEGUE A LA RAÍZ

EJEMPLO

INSERTAR EL ELEMENTO “1” EN EL SIGUIENTE NODO



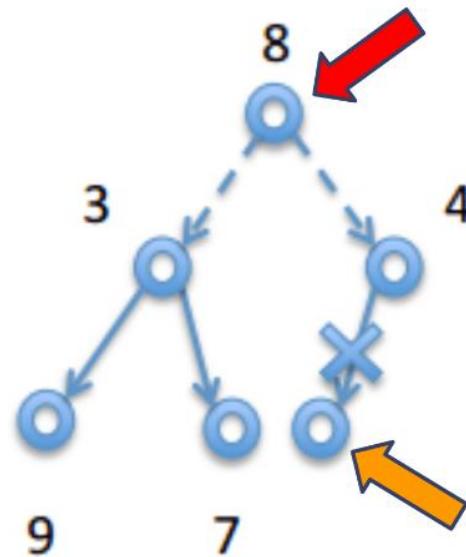
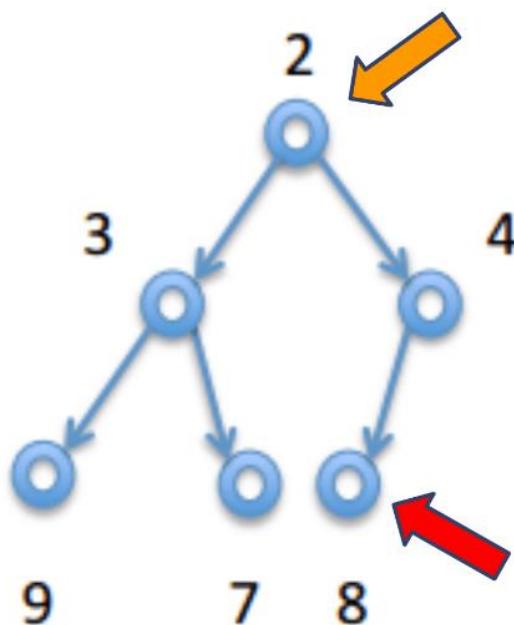
EJEMPLO



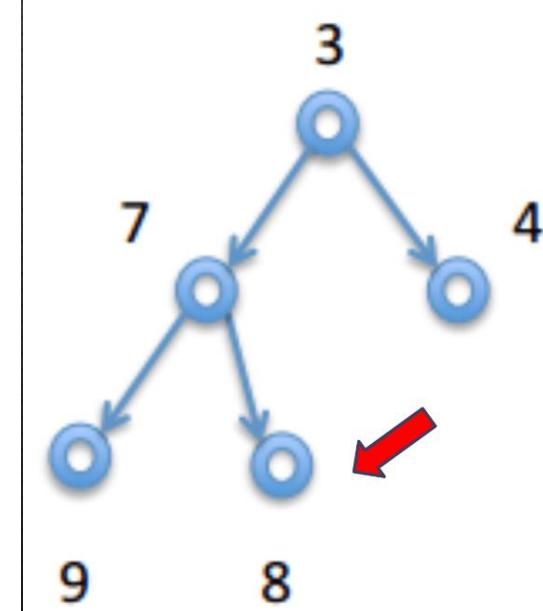
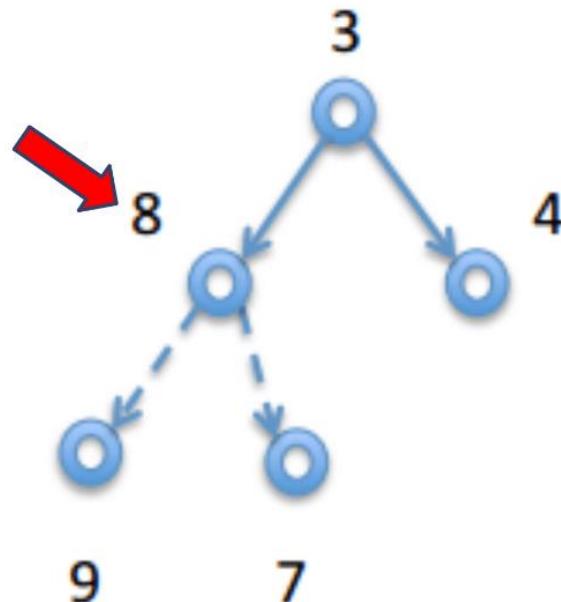
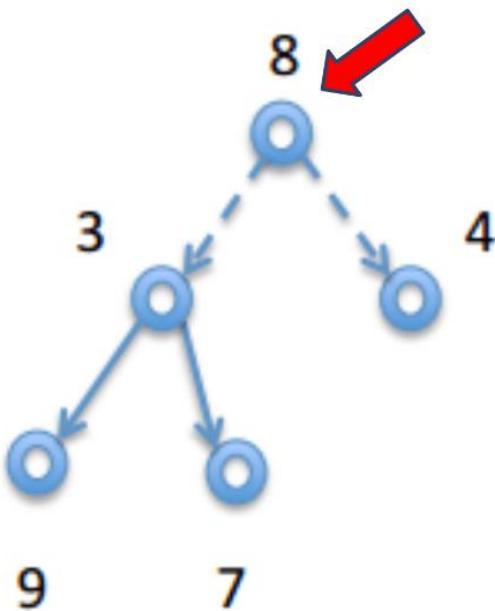
ELIMINACIÓN

- ELIMINAR LA RAÍZ.
- INTERCAMBIAR EL ÚLTIMO NODO INSERTADO CON LA RAÍZ. (SE VIOLARÁ LA CONDICIÓN DE ORDEN INVARIANTE)
- REALIZAR INTERCAMBIOS EVALUANDO LOS NODOS HIJOS E INTERCAMBIANDO POR EL NODO DE MAYOR PRIORIDAD.
- REPETIR EL PASO 3 HASTA QUE SE DEJE DE VIOLAR LA CONDICIÓN DE ORDEN INVARIANTE O ALCANCEMOS UNA HOJA

ELIMINACIÓN



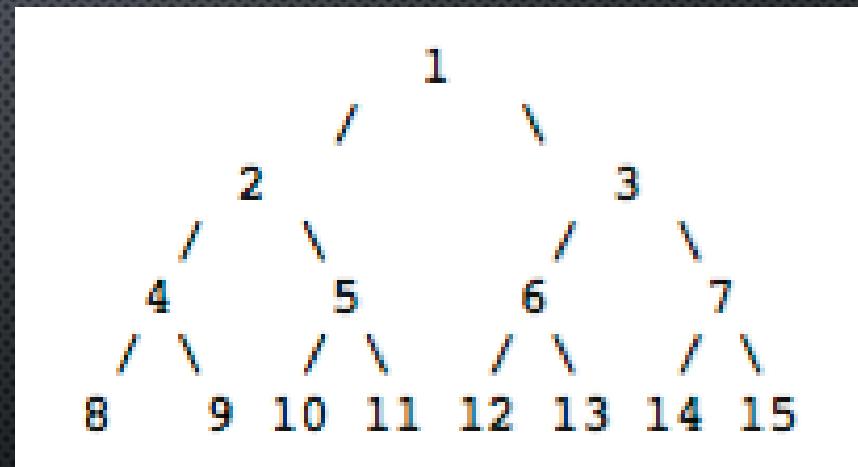
ELIMINACIÓN



HEAP (EN UN ARREGLO)

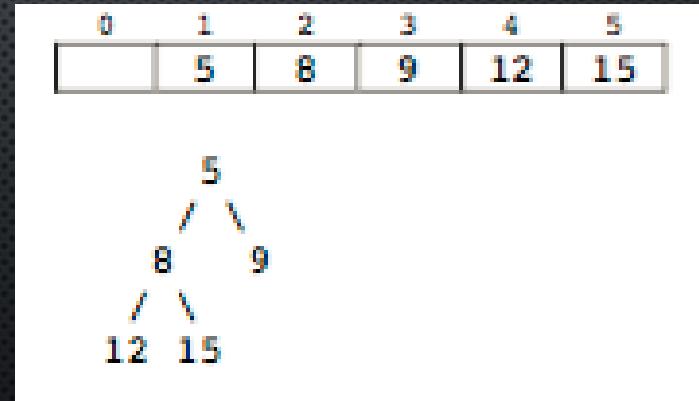
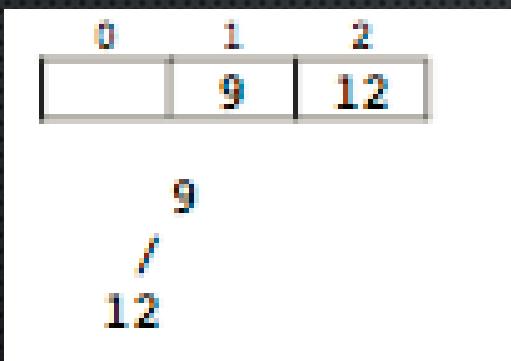
- SE APROVECHA QUE PARA CADA NODO “I” SE CUMPLE:

- EL HIJO IZQUIERDO ES $2*I$
- EL HIJO DERECHO ES $2*I + 1$
- EL PADRE ES $I \text{ DIV } 2$
- LA RAÍZ SIEMPRE SERÁ 1
- LA POSICIÓN 0 SIEMPRE ESTARÁ VACÍA



HEAP (EN UN ARREGLO)

■ ALGUNOS EJEMPLOS DE DICHA REPRESENTACIÓN:



HEAP SORT

- SI SE AGREGAN TODOS LOS ELEMENTOS DE UN LISTADO A UN HEAP (PASO 1) Y LUEGO SE REMUEVEN (PASO 2), SE OBTIENE UN LISTADO ORDENADO.
- LAS OPERACIONES DE INSERCIÓN Y BORRADO SON $O(\log n)$, SI ESTAS SE REPITEN N VECES, OBTENEMOS UN ORDENAMIENTO DE RENDIMIENTO $O(n \log n)$, SIMILAR AL DEL MERGE SORT.
- EL PRINCIPIO PARA TRABAJAR ESTE ORDENAMIENTO ES SIMILAR AL DEL ORDENAMIENTO POR INSERCIÓN.