



Estructura De Datos

Trabajo Grupal

Tema**:** Exposición Final de Semestre.

Integrantes: Códigos:

Juana Delgado 6078

Aldas Wilmer 6035

Cando Lizbeth 5951

Pilataxi Claudio 6042

Romero Marcelo 6030

Semestre: Año Lectivo:

Tercero “A” Abril-Agosto 2016

Fecha de Envió: Fecha de Recepción:

21/07/2016 29/07/2016

Riobamba-Ecuador

ESCUELA SUPERIOR POLITECNICA DE CHIMBORAZO

Facultad de Informática y Electrónica  
Escuela de Ingeniería en Sistemas

# INTRODUCCIÓN

Principalmente la materia estructura de datos es una forma de programar de manera estructurada utilizando más el conocimiento de razonamiento abstracto, lógico, para de esta manera lograr metas esperadas que son la forma de programar mucho más sencillo.

Con el presente trabajo se pretende aclarar todos los conocimientos adquiridos durante todo el semestre en la materia de estructura de datos durante el período abril – agosto 2016.

Para lo cual se plantea un resumen de toda la materia de estructura de datos jerárquica, además se platean ejercicios para cada tema propuesto en el aula de clase. En esta unidad se ha visto más acerca de diagramas, arboles, grafos, que son temas mucho más elementales para la carrera de Ingeniería en Sistemas.

Cada uno de los ejercicios planteados demuestran los logros alcanzados en la materia de estructura de datos, y como trabajo final se ha desarrollado el presente resumen.

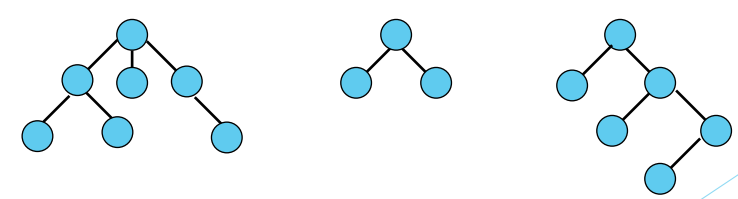
# OBJETIVOS

* Plantear los ejercicios definidos para cada tema
* Obtener un resumen de toda la materia de estructura jerárquica
* Demostrar los logros obtenidos durante el transcurso de este informe
* Facilitar la comprensión de la materia con cada uno de los ejercicios

# MARCO TEÓRICO

## ARBOLES

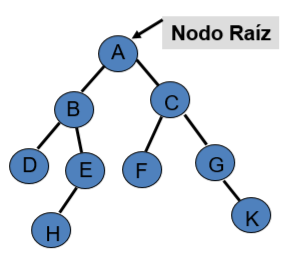
* Son Estructuras de Datos no lineales.
* Es una colección de nodos donde cada uno, además de almacenar información, guarda la dirección de sus sucesores.
* Es una estructura de datos jerárquica.
* La relación entre los elementos es de uno a muchos.



Terminología

Nodo: Cada elemento en un árbol.

Nodo Raíz: Primer elemento agregado al árbol.



Nodo Padre: Se le llama así al nodo predecesor de un elemento.

Nodo Hijo: Es el nodo sucesor de un elemento.

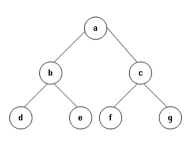
Hermanos: Nodos que tienen el mismo nodo padre.

Nodo Hoja: Aquel nodo que no tiene hijos.

Subárbol: Todos los nodos descendientes por la izquierda o derecha de un nodo.

## ARBOLES BINARIOS

* A los arboles ordenados de grado dos se les conoce como arboles binarios ya que cada nodo del árbol no tendrá más de dos descendientes directos.
* Las aplicaciones de los arboles binarios son muy variadas ya que se les puede utilizar para representar una estructura en la cual es posible tomar decisiones con dos opciones en distintos puntos.
* La representación gráfica de un árbol binario es la siguiente:



Existen cuatro tipos de árbol binario:

* A. B. Distinto.
* A. B. Similares.
* A. B. Equivalentes.
* A. B. Completos.

## ARBOLES BALANCEADOS

Un árbol AVL es un tipo especial de árbol binario ideado por los matemáticos rusos Adelson-Velskii y Landis.

Fue el primer árbol de búsqueda binario auto-balanceable que se ideó.

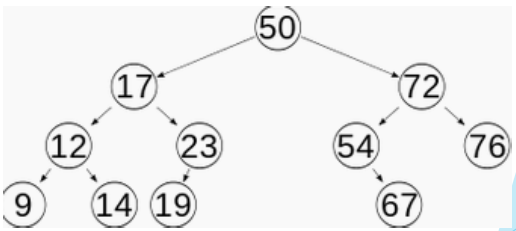
Los árboles AVL están siempre equilibrados de tal modo que para todos los nodos, la altura de la rama izquierda no difiere en más de una unidad de la altura de la rama derecha o viceversa.

Gracias a esta forma de equilibrio (o balanceo), la complejidad de una búsqueda en uno de estos árboles se mantiene siempre en orden de complejidad O(log n).

El factor de equilibrio puede ser almacenado directamente en cada nodo o ser computado a partir de las alturas de los subárboles.

Para conseguir esta propiedad de equilibrio, la inserción y el borrado de los nodos se han de realizar de una forma especial.

Si al realizar una operación de inserción o borrado se rompe la condición de equilibrio, hay que realizar una serie de rotaciones de los nodos.



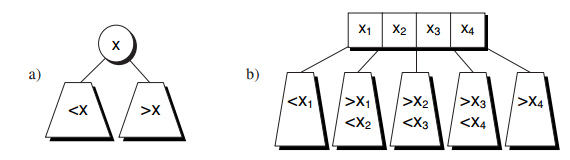
## ARBOLES B

### Definición:

Un árbol B de orden p es básicamente un árbol de búsqueda n-ario donde los nodos tienen p hijos como máximo, y en el cual se añade la condición de balanceo de que todas las hojas estén al mismo nivel. La definición formal de árbol B es la siguiente.

### Características:

* Los nodos internos son de la forma (p1, x1, p2, x2,..., xq−1, pq), siendo pi punteros a nodos hijos y xi claves, o pares (clave, valor) en caso de representar diccionarios.
* Si un nodo tiene q punteros a hijos, entonces tiene q − 1 claves



* Para todo nodo interno, excepto para la raíz, p/2 ≤ q ≤ p. Es decir, un nodo puede tener como mínimo p/2 hijos y como máximo p. El valor p/2 − 1 indicaría el mínimo número de claves en un nodo interno y se suele denotar por d.
* La raíz puede tener 0 hijos (si es el único nodo del árbol) o entre 2 y p.
* Los nodos hoja tienen la misma estructura, pero todos sus punteros son nulos.
* En todos los nodos se cumple: x1 < x2 <...< xq−1.
* Para todas las claves k apuntadas por un puntero pi de un nodo se cumple:
* Si i = 1 entonces k< k1.
* Si i = q entonces xi-1 <k.
* En otro caso, xi−1 <k<xi.
* Todos los nodos hoja están al mismo nivel en el árbol.

### Representación del tipo árbol B

Pasando ahora a la implementación de árboles B de cierto orden p, lo normal es reservar espacio de forma fija para p hijos y p − 1 elementos. La definición, suponiendo que el tipo almacenado es T.

### Inserción en un árbol B

El procedimiento de inserción en un árbol B debe asegurar la propiedad que impone que todas las hojas estén al mismo nivel. La nueva entrada debe insertarse siempre en un nodo hoja. Si hay sitio en la hoja que le corresponde, el elemento se puede insertar directamente.

### Eliminación en un árbol B

Igual que la inserción en un árbol B hace uso de la partición de un nodo en dos, la eliminación se caracteriza por el proceso de unión de dos nodos en uno nuevo, en caso de que el nodo se vacíe hasta menos de la mitad.

## GRAFOS

**Definición:** Es un conjunto de puntos y un conjunto de líneas, cada una de las cuales une un punto con otro. Los puntos se llaman nodos o vértices de un grafo y las líneas se llaman aristas o arcos.

**Aristas**

Son las líneas con las que se unen las aristas de un grafo y con la que se construyen también caminos.

**Aristas Adyacentes:** Se dice que dos aristas son adyacentes si coinciden en el mismo vértice.

**Aristas Paralelas:** Se dice que dos aristas son paralelas si vértice inicial y el final son el mismo.

**Aristas Cíclicas:** Arista que parte de un vértice para entrar en el mismo.

**Cruce:** Son dos aristas que cruzan en un punto.

**Vértices**

Son los puntos o nodos con los que está conformado un grafo.

Llamaremos grado de un vértice al número de aristas de las que es extremo. Se dice que un vértice es par o impar según lo sea su grado.

**Vértices Adyacentes:** si tenemos un par de vértices de un grafo (U, V) y si tenemos un arista que los une, entonces U y V son vértices adyacentes y se dice que U es el vértice inicial y V el vértice adyacente.

**Vértice Aislado:** Es un vértice de grado cero.

**Vértice Terminal:** Es un vértice de grado 1.

### Clasificación de los grafos

* **Dirigidos:** Cada arco está representado por un par ordenado de vértices, de forma que representan dos arcos diferentes.
* **No dirigidos:** En este el par de vértices que representa un arco no está ordenado

### Tipos de grafos

1. **Grafo regular:** Aquel con el mismo grado en todos los vértices.
2. **Grafo bipartito:** Es aquel con cuyos vértices pueden formarse dos conjuntos disjuntos de modo que no haya adyacencias entre vértices pertenecientes al mismo conjunto
3. **Grafo completo:** Aquel con una arista entre cada par de vértices
4. **Grafo nulo:** Se dice que un grafo es nulo cuando los vértices que lo componen no están conectados, esto es, que son vértices aislados.
5. **Grafos Isomorfos:** Dos grafos son isomorfos cuando existe una correspondencia biunívoca (uno a uno), entre sus vértices de tal forma que dos de estos quedan unidos por una arista en común.
6. **Grafos Platónicos:** Son los Grafos formados por los vértices y aristas de los cinco  sólidos regulares      (Sólidos Platónicos), a saber, el tetraedro, el cubo, el octaedro, el dodecaedro y el icosaedro
7. **Grafos conexos:** Un grafo se puede definir como conexo si cualquier vértice V pertenece al conjunto de vértices y es alcanzable por algún otro. Otra definición que dejaría esto más claro sería: “un grafo conexo es un grafo no dirigido de modo que para cualquier par de nodos existe al menos un camino que los une”.
8. **Grafos dirigidos (bigrafo):** A un grafo dirigido se le puede definir como un grafo que contiene aristas dirigidas, como en el siguiente caso.

### Recorridos de un grafo

**Recorrido en anchura:**    El recorrido en anchura supone recorrer el grafo, a partir de un nodo dado, en niveles, es decir, primero los que están a una distancia de un arco del nodo de salida, después los que están a dos arcos de distancia, y así sucesivamente hasta alcanzar todos los nodos a los que se pudiese llegar desde el nodo salida.

**Recorrido en profundidad:**el recorrido en profundidad trata de buscar los caminos que parten desde el nodo de salida hasta que ya no es posible avanzar más. Cuando ya no puede avanzarse más sobre el camino elegido, se vuelve atrás en busca de caminos alternativos, que no se estudiaron previamente

## RECORRIDO DE GRAFOS

Recorrer un grafo significa tratar de alcanzar todos los nodos que estén relacionados con uno que se llama de salida.

Existen dos técnicas para recorrer un grafo:

* Recorrido en anchura (BFS)
* Recorrido en profundidad (DFS)

**Recorrido en profuncidad (DFS)**

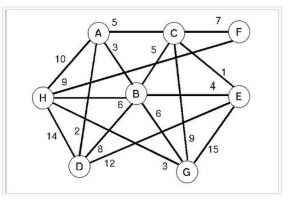
* Recorrido DFS (Depth First Search)
* Trata de buscar los caminos que parten desde el nodo de salida hasta que y no es posible avanzas más.
* Cuando ya no puede avanzar más sobre el camino elegido, se regresa en busca de caminos alternativos, que no se estudiaron previamente.
* La búsqueda en profundidad empieza por un vértice V del grafo G; V no visitado; así hasta que no haya más vértices no visitados.

**Recorrido en anchura (BFS)**

* Recorrido BFS (Beadth First Search)
* Supone recorrer el grafo, a partir de un nodo dado, en niveles, es decir, primero los que están a una distancia de un arco del nodo de salida.
* Después los que están a dos arcos de distancia y así sucesivamente hasta alcanzar todos los nodos a los que se pudiese llegar desde el nodo salida.
* Este método comienza visitando el vértice de partida A, luego visitar los adyacentes que no estuvieron ya visitados. Así sucesivamente con los adyacentes.
* Éste método utiliza una cola como estructura auxiliar en la que se mantienen los vértices que se vayan a procesar posteriormente.

## CAMINO MAS CÓRTO

* En la teoría de grafos, el problema del camino más corto es el problema que consiste en encontrar un camino entre dos vértices (o nodos) de tal manera que la suma de los pesos de las aristas que lo constituyen es mínima.
* Un ejemplo de esto es encontrar el camino más rápido para ir de una ciudad a otra en un mapa.
* En este caso, los vértices representarían las ciudades y las aristas las carreteras que las unen, cuya ponderación viene dada por el tiempo que se emplea en atravesarlas.



* El problema del camino más corto puede ser definido para grafos no dirigidos, dirigidos o mixtos.
* El problema es también conocido como el problema de los caminos más cortos entre dos nodos, para diferenciarlo de las siguientes generalizaciones:

+ El problema de los caminos más cortos desde un origen, en el cual se tiene que encontrar los caminos más cortos de un vértice origen v a todos los demás vértices del grafo.

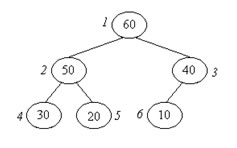
+ El problema de los caminos más cortos con un destino, en el cual se tiene que encontrar los caminos más cortos desde todos los vértices del grafo a un único vértice destino, esto puede ser reducido al problema anterior invirtiendo el orden.

+ El problema de los caminos más cortos entre todos los pares de vértices, el cual se tiene que encontrar los caminos más cortos entre cada par de vértices (v, v') en el grafo.

## HEAPS

### Definición

Un heap o montículo es una estructura de datos similar a un árbol binario de búsqueda pero ordenado de distinta forma por prioridades y además se representa siempre como un árbol binario completo representado como un arreglo. (García, 2014)



### Características

Un montículo es un árbol binario completo tal que puede:

* Estar vacío
* El valor de la prioridad en la raíz es mayor, (menor) o igual que la prioridad de cualquiera de sus hijos.
* Ambos subárboles son montículos o heap.

### Propiedades del Heap

Debe cumplir dos propiedades:

* Un árbol binario completamente lleno, con la posible excepción del nivel más bajo, el cual se rellena de izquierda a derecha. Estos árboles se denominan árboles binarios completos.
* Todo nodo debe ser mayor que todos sus descendientes. Por lo tanto, el máximo estará en la raíz y su búsqueda y eliminación se podrá realizar rápidamente.

### Otras características:

* Todos los heaps son árboles binarios. No son necesariamente ABBs.
* El árbol está completamente balanceado excepto el último nivel, que debe estar lleno de izquierda a derecha.
* Para un elemento del heap en la posición k, sus hijos deberán estar en las posiciones 2k y 2k+1 del heap.
* Un HEAP puede representarse en un arreglo.
* Toda lista ordenada es un heap.

### Operaciones con los heaps

Pasos para insertar en un Heap:

* Agregamos el nodo. (de izquierda a derecha) Comparamos son su padre.
* Si es mayor permutamos hasta llegar a la raíz Repetimos el paso 1 y 2 hasta llenar el nivel. Una vez llenado ese nivel pasamos al siguiente nivel.
* Eliminar Datos

Pasos para eliminar:

* Eliminamos la raíz del heap
* Una vez eliminada remplazamos la raíz con el último nodo del último nivel.
* Comparamos si los hijos de la nueva raíz son menores.
* Si son menores no se hace ninguna permutación Si son mayores (o uno de ellos) se hace permutación con el hijo mayor.
* Repetimos los pasos anteriores hasta no tener nodos para eliminar.

## HEAPSORT

## HASHING

A las **funciones resumen** también se les llama **funciones *hash*** o **funciones *digest*** [1](https://es.wikipedia.org/wiki/Funci%C3%B3n_hash#cite_note-1) [2](https://es.wikipedia.org/wiki/Funci%C3%B3n_hash#cite_note-2) [3](https://es.wikipedia.org/wiki/Funci%C3%B3n_hash#cite_note-3) . Una [función](https://es.wikipedia.org/wiki/Funci%C3%B3n_(matem%C3%A1tica)) *hash* H es una función computable mediante un [algoritmo](https://es.wikipedia.org/wiki/Algoritmo) tal que:

*H*: *U* → *M*

*x* → *h*(*x*)

Tiene como entrada un conjunto de elementos, que suelen ser [cadenas](https://es.wikipedia.org/wiki/Cadena_de_caracteres), y los convierte en un rango de salida finito, normalmente [cadenas](https://es.wikipedia.org/wiki/Cadena_de_caracteres) de longitud fija. Es decir, la función actúa como una proyección del conjunto U sobre el conjunto M.

Características de los hash:

En definitiva las funciones hash se encargan de representar de forma compacta un archivo o conjunto de datos que normalmente es de mayor tamaño que el hash independientemente del propósito de su uso.

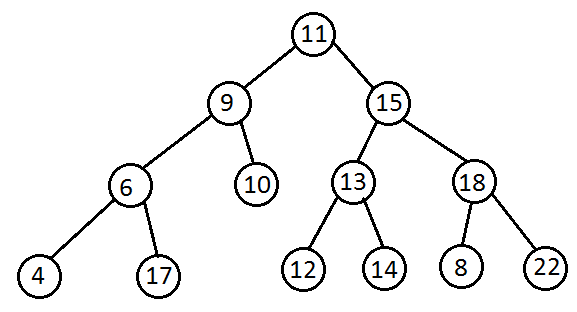
# EJERCICIOS DE LOS TEMAS PLANTEADOS EN ESTRUCTURAS JERÁRQUICAS

## ARBOLES

### Arboles binarios

#### Inserción

Insertar el nodo 26



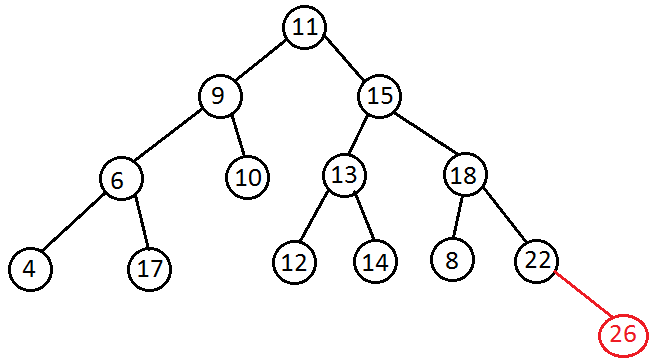
Comparar:

26>11 si derecha

26>15 si derecha

26>18 si derecha

26>22 si insertar el nodo



#### Eliminación

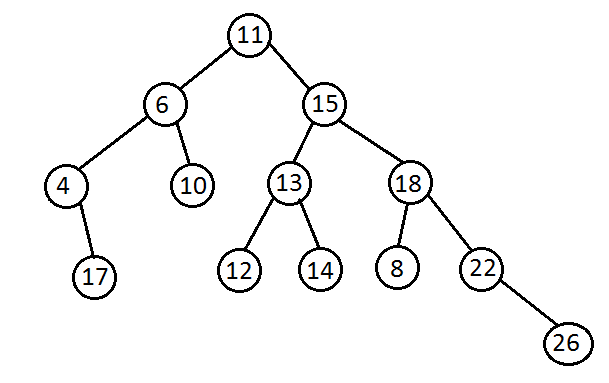
Eliminar el nodo 9

Comparar

9>11 no izquierda

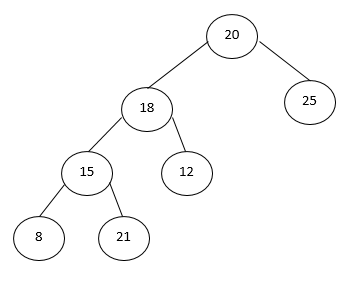
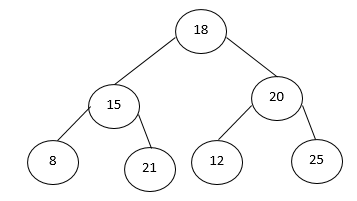
Eliminar el 9

Ordenar

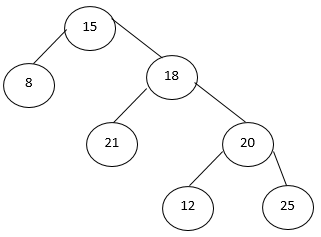
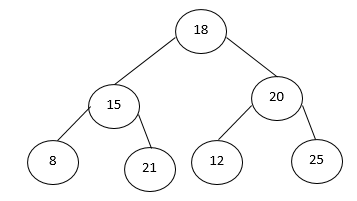


## Arboles Balanceados

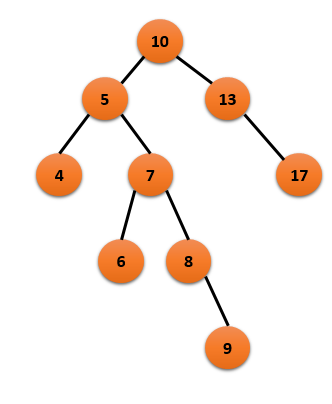
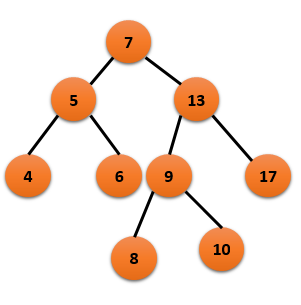
### Rotación a la izquierda



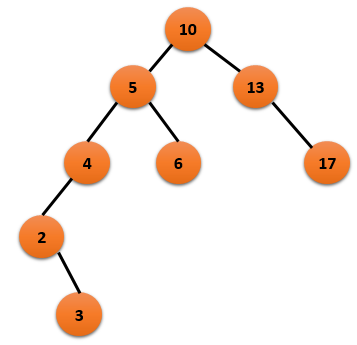
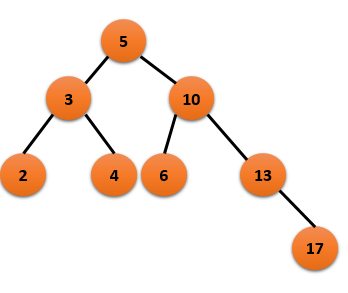
### Rotación a la derecha



### Rotación doble a la izquierda

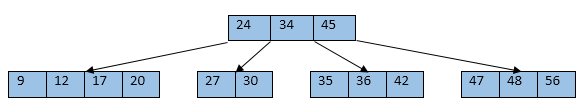
 

### Rotación doble a la derecha

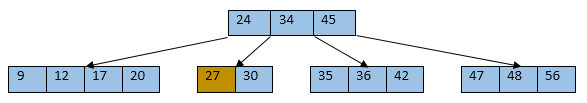
 

## Arboles B

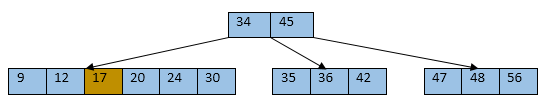
* **Eliminar el nodo 27**



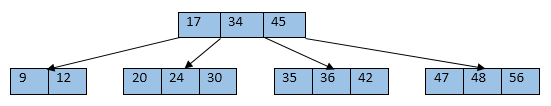
1. Buscamos el número para eliminarlo, en este caso el 27.



1. Cuando eliminamos el número 27, el subárbol queda solo con un nodo por lo cual debe de unirse al otro subárbol y realizar la unión, en este caso con la raíz a la que pertenecen.

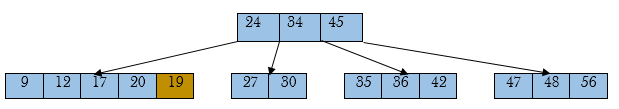


1. Escoger el número medio del subárbol y subirlo a la raíz, y así queda equilibrado el árbol B.

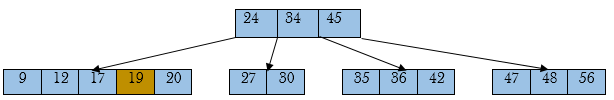


**Método Insertar**

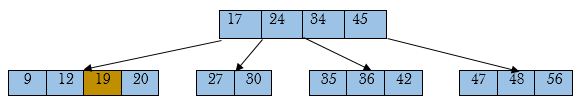
* **Insertar el nodo 19**



1. Ordenamos



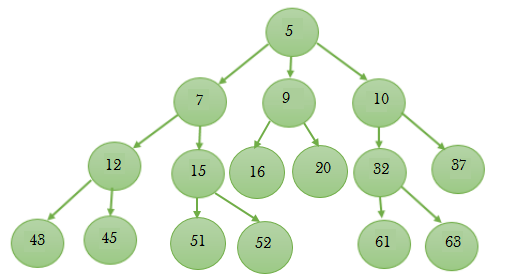
1. En un nodo solo hay como ingresar mínimo 2 elementos y máximo 4 elementos por tal motivo debemos de ordenar el árbol.



## GRAFOS

### Recorridos de grafos

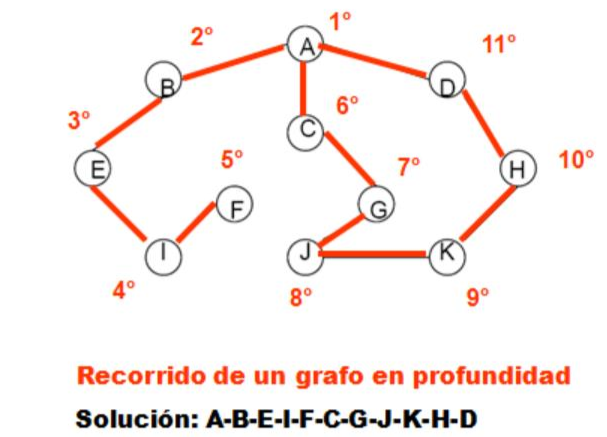
#### Amplitud



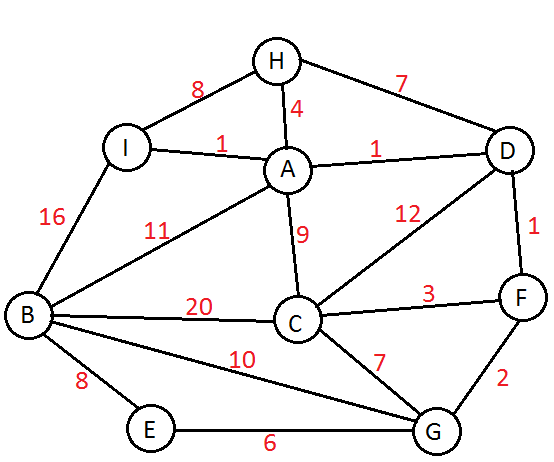
* **Recorrido por amplitud:**

5 – 7 – 9 – 10 – 12 – 15 – 16 – 20 – 32 – 27 – 43 – 45 51 – 52 – 61 - 63

#### Profundidad



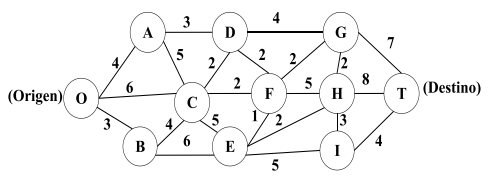
### Camino más corto

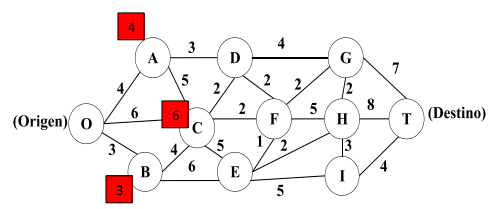
Se desea encontrar el camino más corto desde el nodo h hasta el nodo G

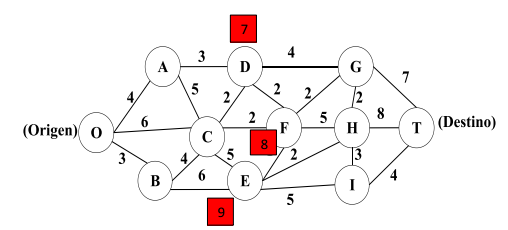
CAMINO MAS CORTO

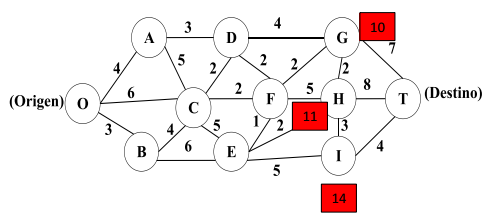
H-A-D-F-G

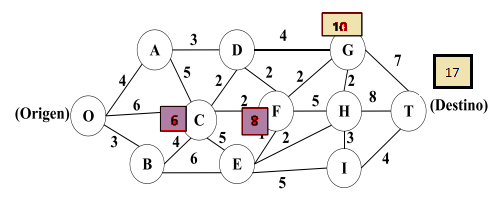
**Ejercicio del camino más cortó**











* **Camino más corto =** O + A + D + G + T
* **Resultado** = 4 + 3 + 4 + 7 = 17

## HEAPS

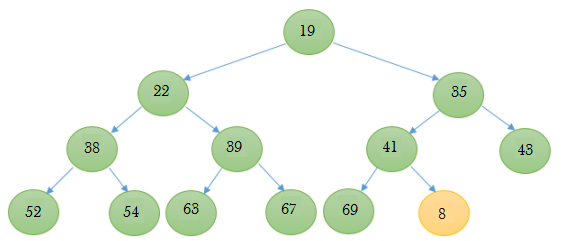
### Inserción

1. **Ejercicio 1**

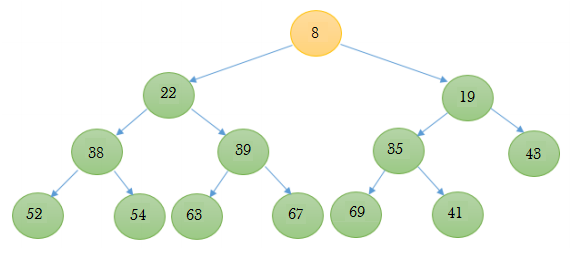
**Insertamos el nodo 8**

Insertamos el nodo 8 al final de la secuencia del árbol, ordenamos de menor a mayor.

En este caso el nodo 8 es el menor de todos los nodos y debe ser la raíz y lo ordenamos con la pregunta si es mayor o menor al nodo que lo contiene.

****

**Ordenamos**

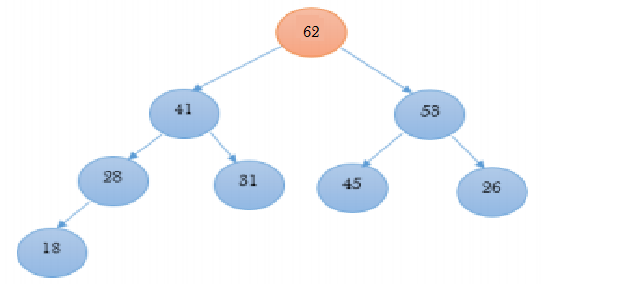
****

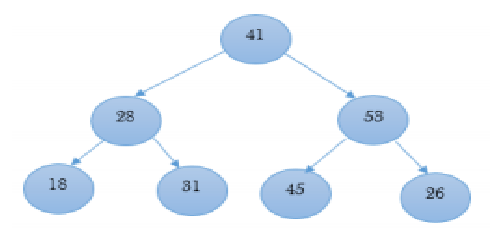
### Eliminación

1. **Ejercicio 1**

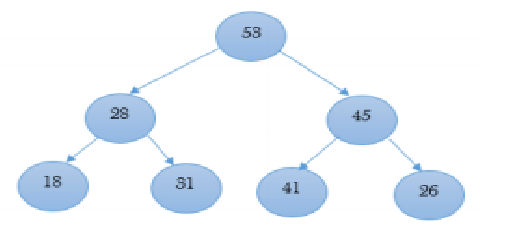
**Eliminamos el nodo 39**

En este caso eliminamos la raíz 39, por lo tanto esta raíz es de mayor a menor y empezamos a ordenarlo, para que el árbol quede de forma ordenada.

****

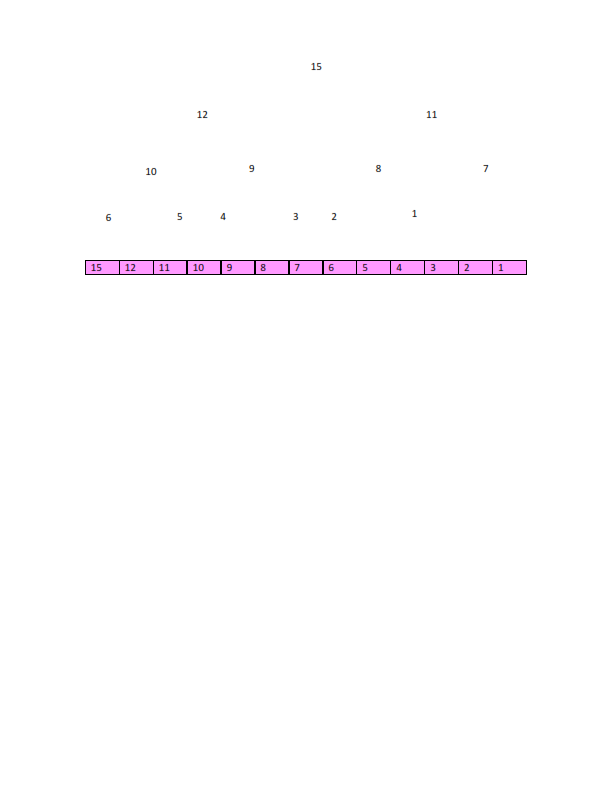
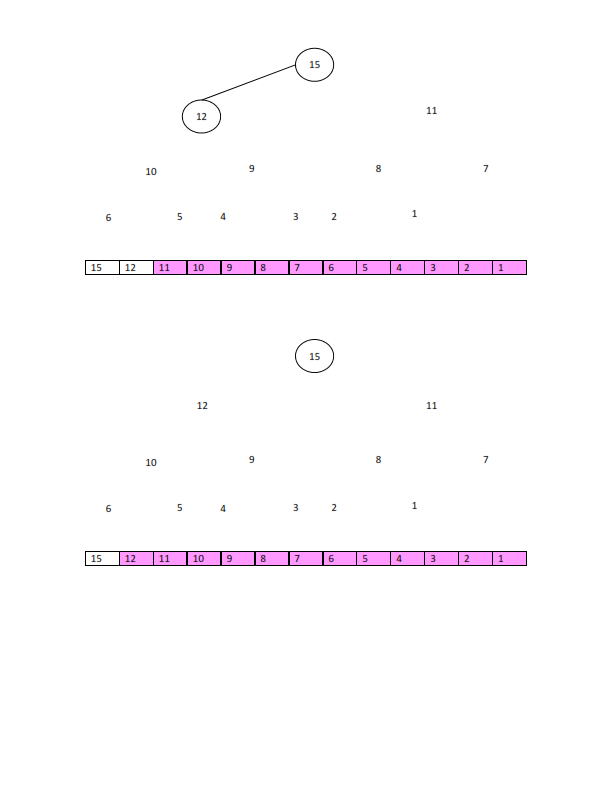
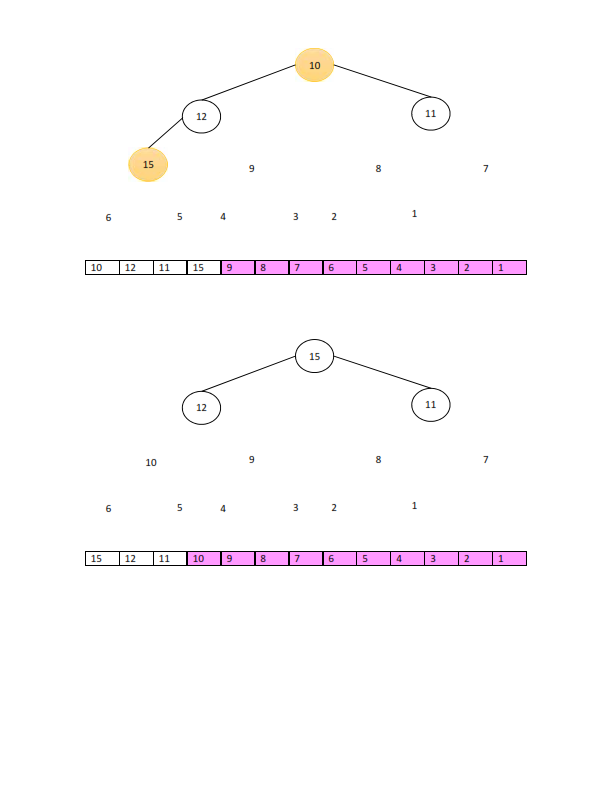
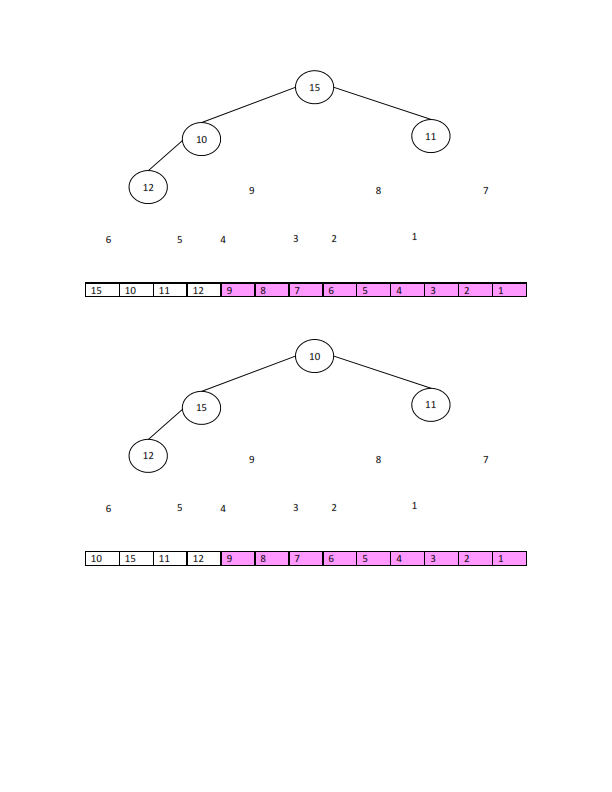
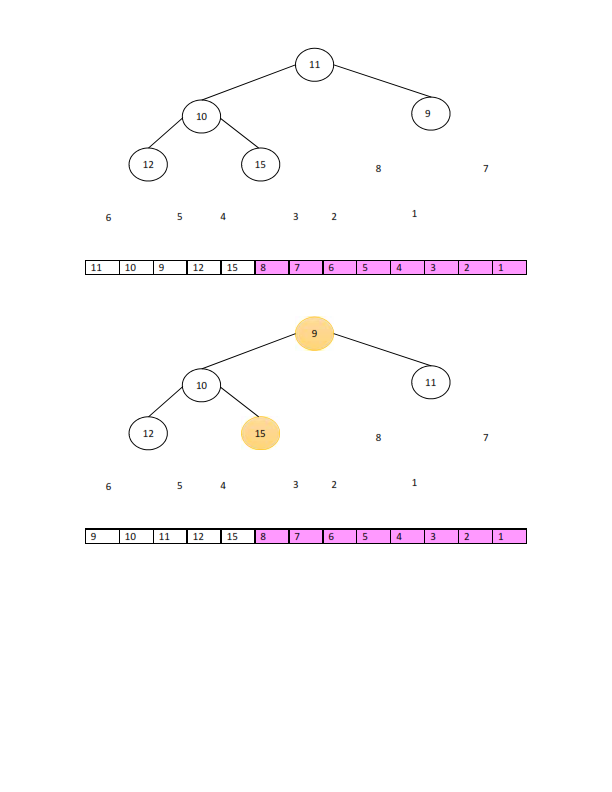
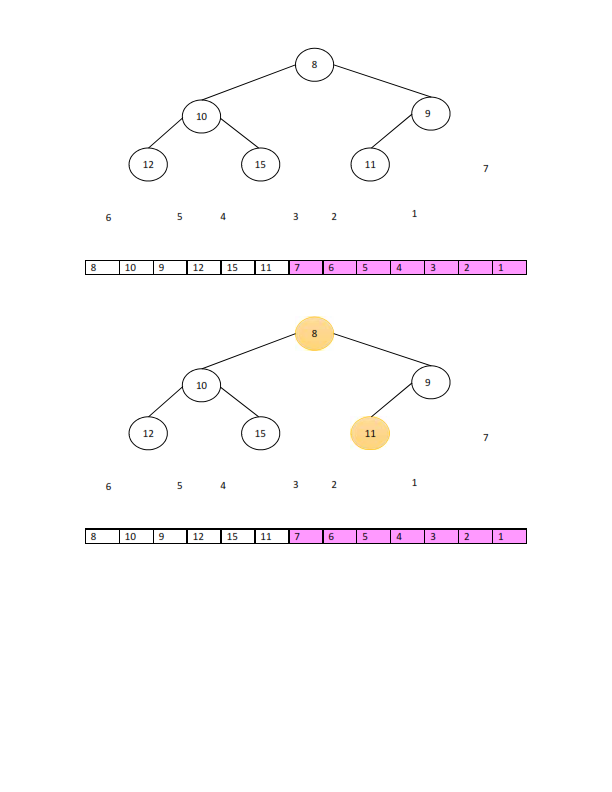
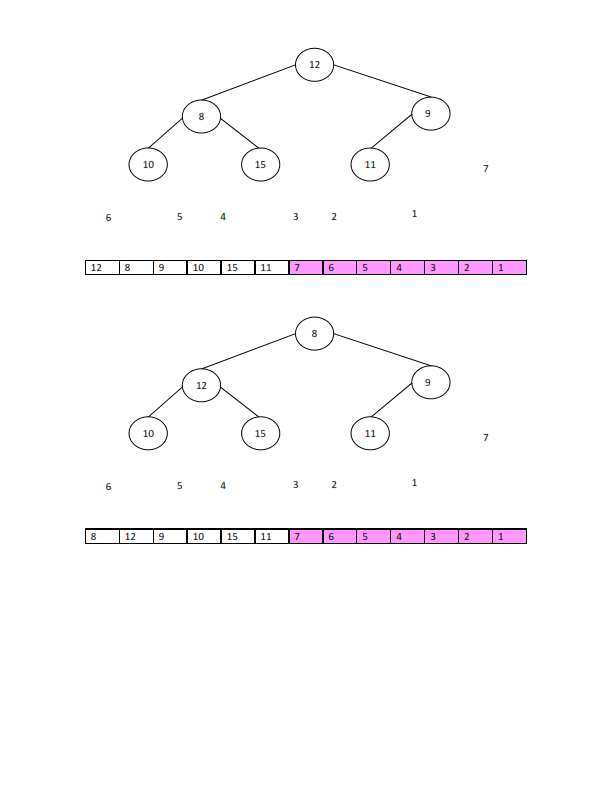
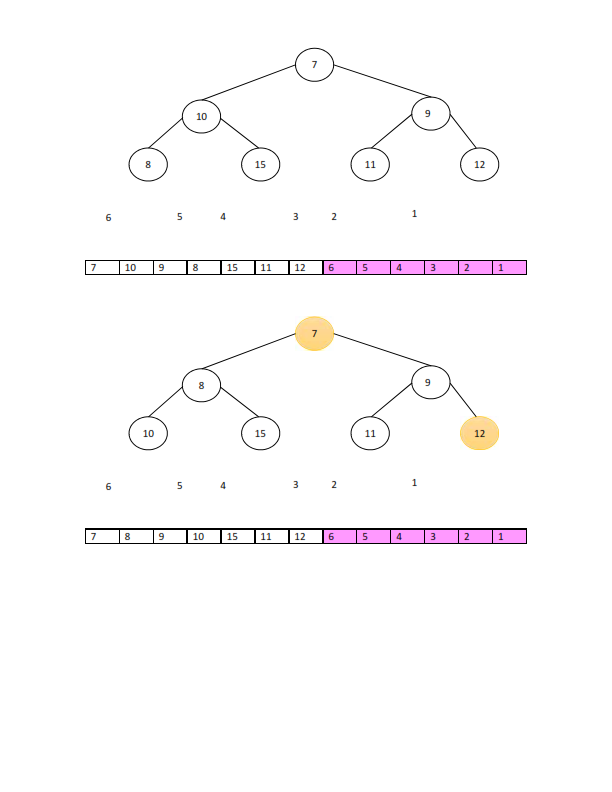
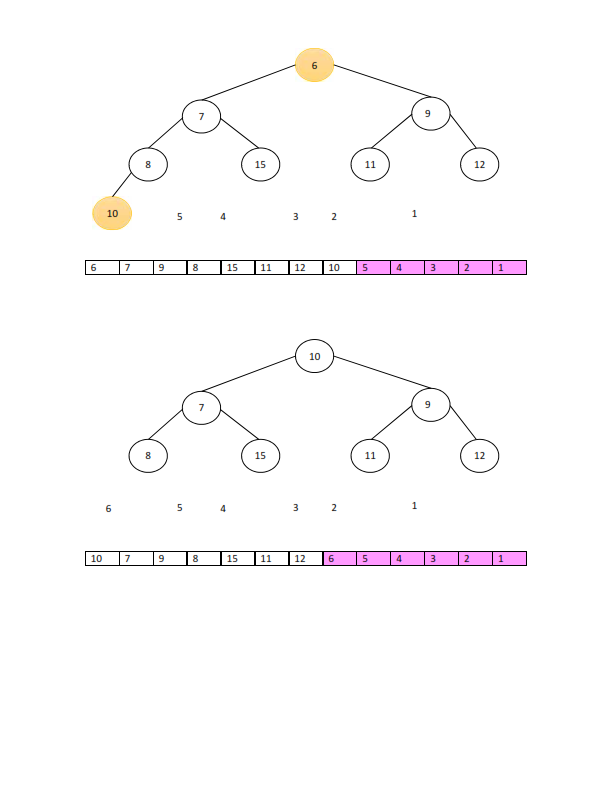
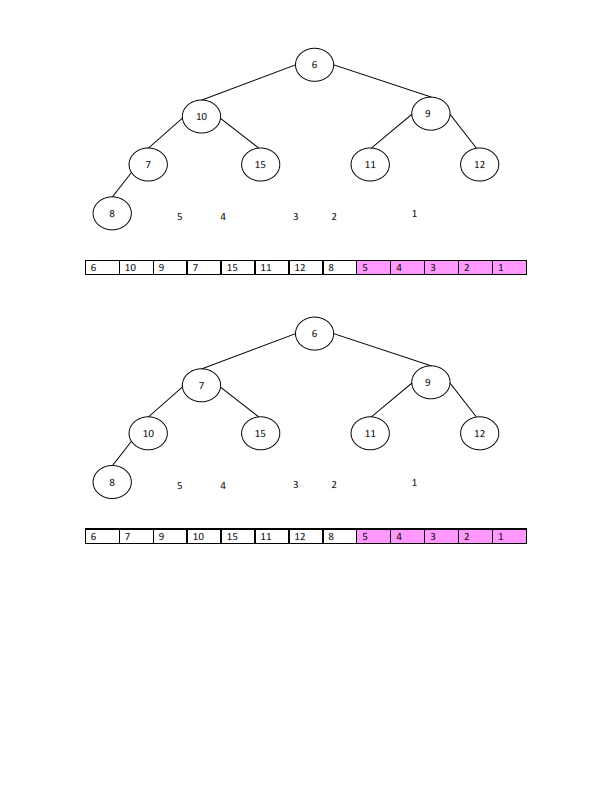
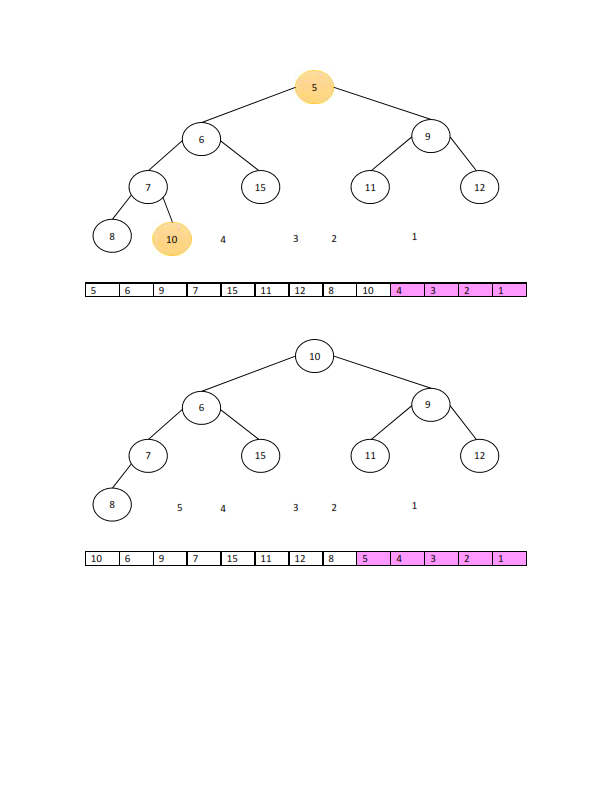
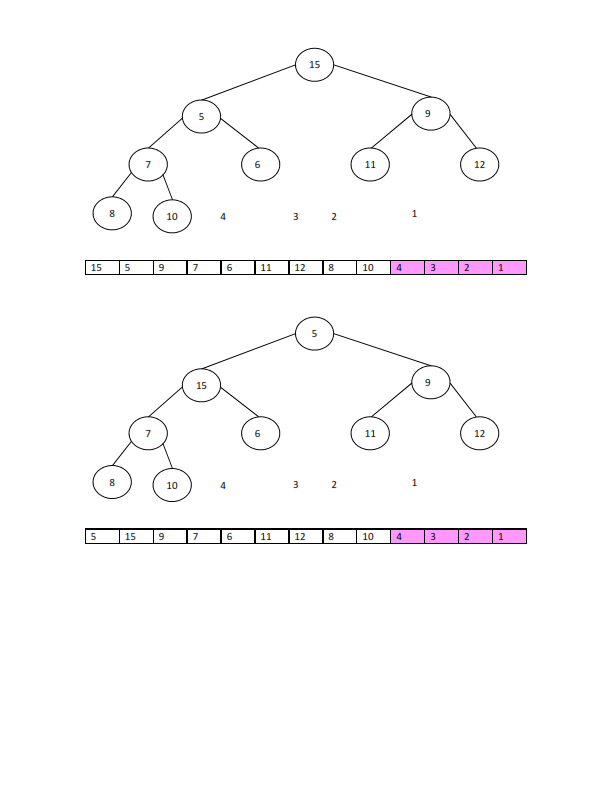
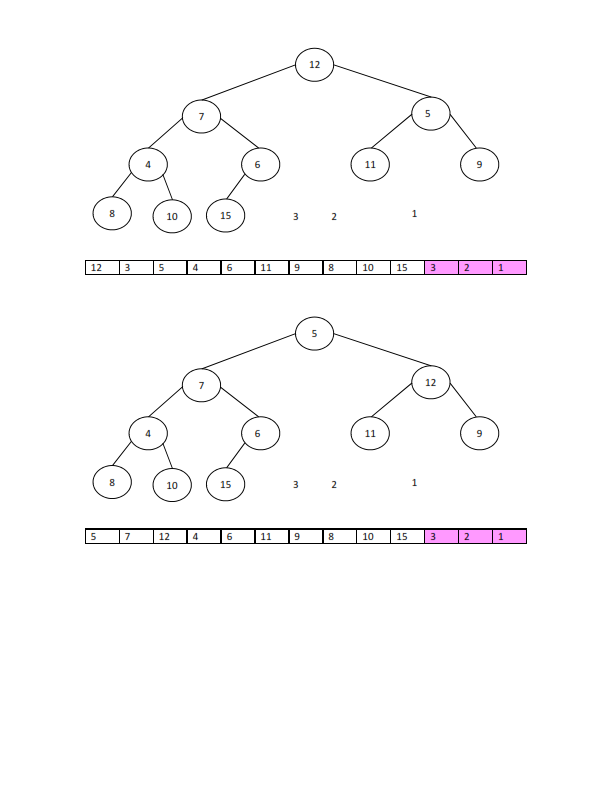
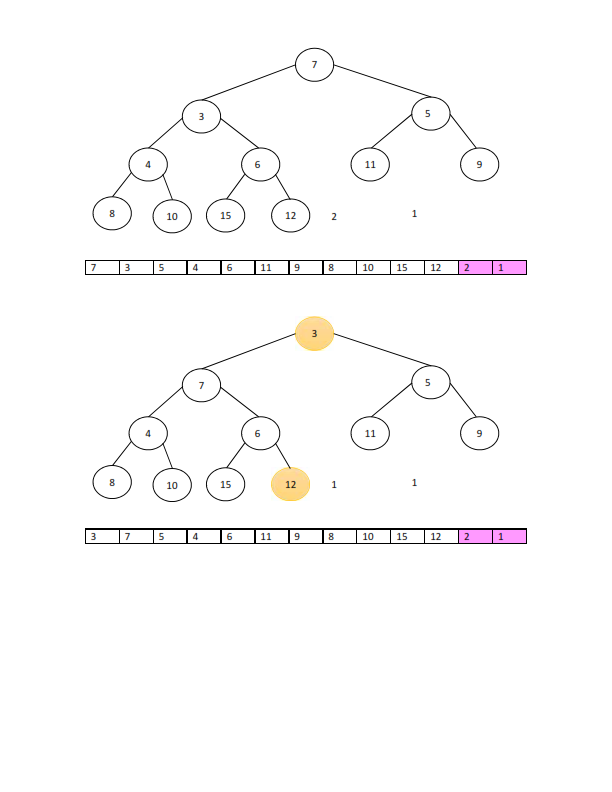
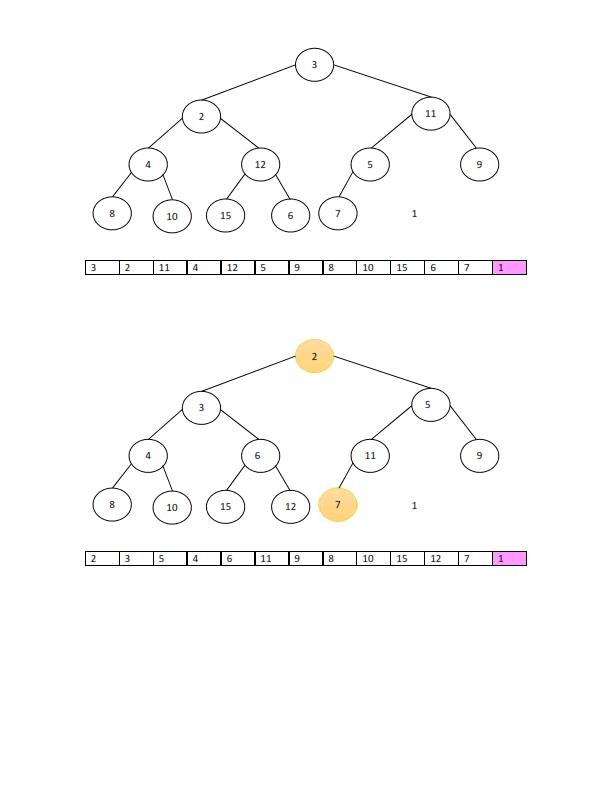
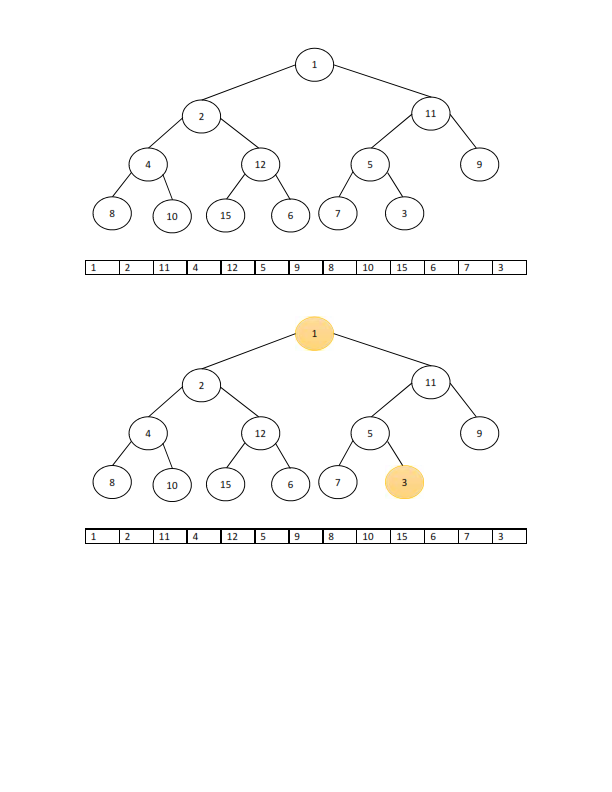
****

**Ordenamos**

****

## HEAPSORT

### Ordenamiento



## HASHING

### Inserción

#### Utilizando el método de exploración

Inserción.

Se tiene una tabla Hash en la cual tenemos los siguientes valores {G, H, J, Z, L, M, T, P}, insertar en la tabla ordenadamente sabiendo que el número es igual a su posición en el alfabeto

G = 7

H = 8

J = 10

Z = 26

L = 12

M = 13

T= 20

P = 16

G mod 7 = 0 H mod 7 = 1

C:\Users\MARCVICENT3\Downloads\Proyecto\1.png

J mod 7 = 3 Z mod 7 = 5

C:\Users\MARCVICENT3\Downloads\Proyecto\2.png

L mod 7 = 5 M mod 7 = 6

C:\Users\MARCVICENT3\Downloads\Proyecto\3 (1).png

T mod 7 = 6 P mod 7 = 2

C:\Users\MARCVICENT3\Downloads\Proyecto\4.png

#### Utilizando el método de encadenamiento

Metodo de encadenamiento enlazado.

Realizaremos el mismo ejercicio anterior pero ahora con el metodo de encadenamiento enlazado.

G = 7

H = 8

J = 10

Z = 26

L = 12

M = 13

T= 20

P = 16

G mod 7 = 0

C:\Users\MARCVICENT3\Downloads\Proyecto\1en (1).png

H mod 7 = 1

C:\Users\MARCVICENT3\Downloads\Proyecto\2en.png

J mod 7 = 3

C:\Users\MARCVICENT3\Downloads\Proyecto\3en.png

Z mod 7 = 5

C:\Users\MARCVICENT3\Downloads\Proyecto\4to.png

L mod 7 = 5

C:\Users\MARCVICENT3\Downloads\Proyecto\5to.png

M mod 7 = 6

C:\Users\MARCVICENT3\Downloads\Proyecto\6to.png

T mod 7 = 6

C:\Users\MARCVICENT3\Downloads\Proyecto\6to.png

T

P mod 7 = 2

C:\Users\MARCVICENT3\Downloads\Proyecto\6to.png

T

### Búsqueda

Del ejercicio anterior buscar la letra P

P mod 7 = 4, entonces empezamos a comparar con cada una de las llaves hasta llegar a la 5ta posición que es en la que se encuentra nuestra P.

C:\Users\MARCVICENT3\Downloads\Proyecto\4.png

P

4

Búsqueda en el método enlazado.

Para este ejercicio utilizaremos el ejercicio anterior del método enlazado para ver cómo funciona en uno de esos casos.

Vamos a buscar la letra T, para esto buscamos el modulo.

T mod 7 = 6.

Entonces empezamos a buscar en cada uno de las llaves hasta llegar al 6, como vemos que en este nodo se encuentra la letra M seguimos a la derecha hasta encontrar la letra T.

C:\Users\MARCVICENT3\Downloads\Proyecto\6to.png

T

6

# CONCLUSIONES

* Se ha logrado obtener toda la información necesaria para el desarrollo de todos los ejercicios, además ha sido de mucha ayuda para el desarrollo de un informe bien resumido, alcanzando una buena comprensión.
* Se demostró que el desarrollo de cada uno de los ejercicios nos ayudó a alcanzar un nivel más de comprensión y exactitud para el desarrollo de ejercicios de estructura jerárquica.
* Mediante la colaboración de todos los integrantes se ha logrado exponer la materia desarrollada, para que de esta manera lográramos despejar las dudas de algunos compañeros de clases.
* Se debe comprender también que el desarrollo de cada uno de los ejercicios es demasiado extenso por lo cual se pudo haber omitido varios pasos durante el desarrollo.

# RECOMENDACIONES

* Plantear adecuadamente los ejercicios propuestos para un mejor desarrollo
* Practicar más de lo necesario para obtener un mejor dominio para cada tema
* Generar un breve resumen para cada tema planteado
* Tener conocimientos previos a la materia
* Obtener más información fuera de clases
* Estudiar para los exámenes

# BIBLIOGRAFÍA

<https://www.uam.es/personal_pdi/ciencias/gallardo/capitulo8a.pdf>

<http://www.uaeh.edu.mx/docencia/P_Presentaciones/icbi/asignatura/Cap6ARBOLES.pdf>

<http://moodle2.unid.edu.mx/dts_cursos_mdl/lic/TI/MC/AM/11/Recorrido_arbol.pdf>

<https://www.cs.princeton.edu/~rs/AlgsDS07/10Hashing.pdf>