Tema 10- Representación Jerárquica: Árboles Binarios

Germán Moltó
Escuela Técnica Superior de Ingeniería Informática
Universidad Politécnica de Valencia

Tema 10- Representación Jerárquica: Árboles Binarios

Índice general:

Objetivos

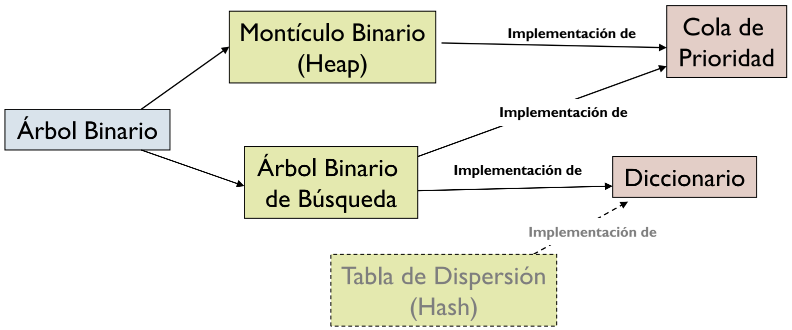
- Conocer la estructura de Árbol Binario y sus principales propiedades, operaciones y recorridos.
- Introducir el Árbol Binario de Búsqueda (ABB) como una de las EDAs más potentes orientadas a la búsqueda.
- Conocer las operaciones que soporta un ABB y cómo su coste varía en función de lo *Equilibrado* que esté el Árbol.

Bibliografía

- Libro de M.A.Weiss, "Estructuras de Datos en Java" (Adisson-Wesley, 2000).
  - Apartados 1 – 3 del capítulo 18 para las definiciones vistas en teoría.
  - Apartado 5 del capítulo 6 para la introducción de las estructuras de Árbol y Árbol Binario.
- Libro de R.Wiener y L.J. Pinson, "Fundamentals of OOP and Data Structures in Java" (Cambridge University Press, 2000).
  - Apartados 3, 4 y 9 del Capítulo 15

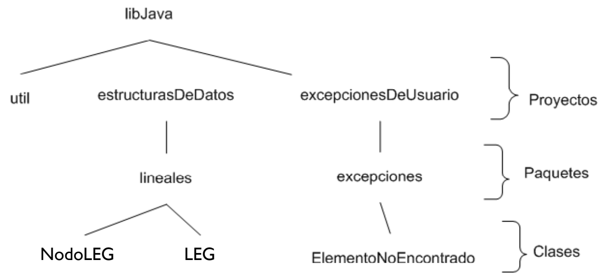
Justificación del Tema

- Estudiaremos diferentes implementaciones de los modelos de Cola de Prioridad y Diccionario.



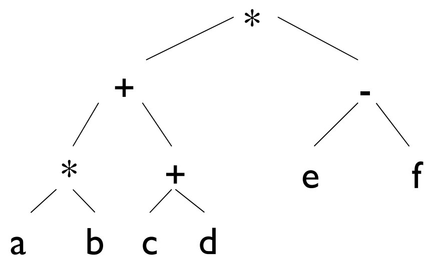
Introducción: Estructuras Jerárquicas (I)

- A menudo, los datos de una colección presentan una relación jerárquica:
  - Imposible representar linealmente con LEG o un array.
  - Ejemplo 1: Estructura de un directorio de librerías (al estilo del laboratorio de prácticas):



Introducción: Estructuras Jerárquicas (II)

- Ejemplo 2: Representación de una expresión aritmética totalmente parentizada en notación infija.
  - (((a*b)+(c+d))*(e-f))

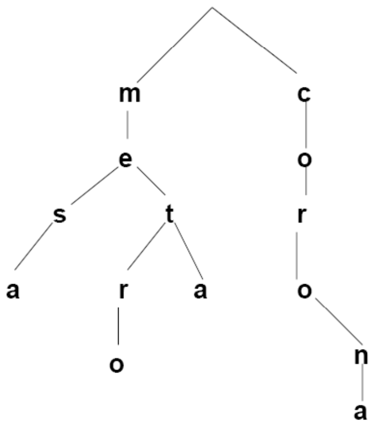


Introducción: Estructuras Jerárquicas (III)

- La estructura en Árbol:
  - Relaciones jerárquicas entre los datos de una colección.
  - Búsqueda en **tiempos sublineales**, lo que no se puede conseguir usando una representación lineal.
- Ejemplo: Representar la Colección de Palabras {*mes, mesa, meta, metro, coro y corona*}.
  - Alternativa 1: Usando un array de cadenas:
    - String [] vCadenas = new String[6];
    - ¿Cuál es el coste de buscar una palabra dada en esa colección?
    - ¿Existen instancias significativas?
  - Alternativa 2: Usando una lista enlazada de palabras.
    - ¿Cuál es el coste?
    - ¿Existen instancias significativas?

Introducción: Estructuras Jerárquicas (IV)

- Alternativa 3: Empleamos un árbol para representar la Colección de Datos

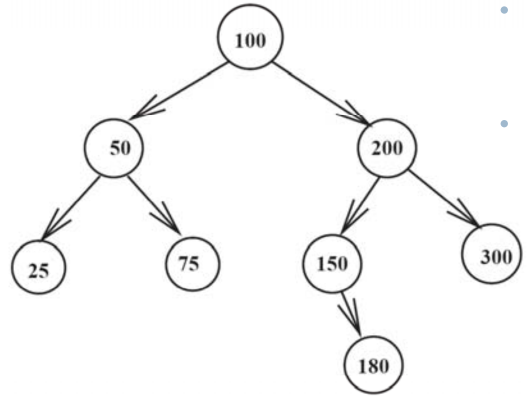


¿Cuál es el coste de buscar una palabra con esta representación jerárquica?

¿Es mejor el coste lineal con el número de palabras o proporcional a la longitud de la palabra?

Introducción: Estructuras Jerárquicas (V)

- Ejemplo de búsqueda eficiente en estructura jerárquica basada en un árbol.

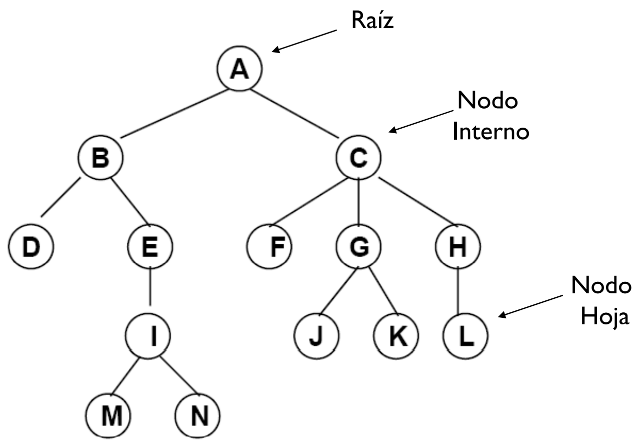


- Esta estructura jerárquica es un Árbol Binario de Búsqueda.
- Permite la búsqueda guiada entre elementos que implementan la interfaz Comparable<T>.

Introducción a los Árboles: Conceptos Generales

- Un **Árbol** es una estructura jerárquica que se define por:
  - Un conjunto de **Nodos** (uno de los cuales es distinguido como la **Raíz** del Árbol).
  - Un conjunto de **Aristas** de manera que:
    - Cualquier nodo H, a excepción de la raíz, está conectado por medio de una Arista a **un único** nodo P.
    - Se dice entonces que P es el **Padre** de H y H es un **Hijo** de P.
    - Un Nodo puede tener múltiples Hijos.
- Si un nodo tiene algún hijo es un nodo **Interno**, sino es una **Hoja**.

Árbol de Ejemplo



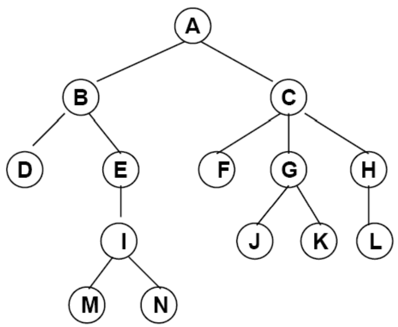
Conceptos: Camino, Profundidad, Nivel y Altura

- Desde la Raíz a cada nodo hay un único **Camino** de longitud igual a su número de Aristas.
  - Ejemplo: El Camino desde la Raíz hasta el nodo Hoja etiquetado como K es A – C – G – K (longitud 3).
- La **Profundidad** de un nodo es la longitud del camino que va desde la raíz hasta él.
  - Ejemplo: Profundidad del nodo Raíz: 0 (profundidad de G: 2)
- Todos los nodos a la misma Profundidad comparten **Nivel**.
  - Ejemplo: Los nodos I, J, K, L están a nivel 3
- La **Altura** de un Nodo es la longitud del Camino que va desde dicho Nodo hasta la hoja más profunda bajo él.
  - La altura de un Árbol corresponde a la altura de su nodo Raíz.
  - Ejemplo: Altura del Árbol : 4

Conceptos: Genealogía

- Los nodos con el mismo Padre son **Hermanos**
  - Ejemplo: Los nodos D y E son Hermanos (el Padre es B).
- Si hay camino desde un nodo u hasta un nodo v, se dice que:
  - u es un **Ascendiente** de v
  - v es un **Descendiente** de u
  - Todo nodo es ascendiente/descendiente de sí mismo.
  - Si u ≠v entonces se denomina ascendiente/descendiente **Propio**
  - Ejemplo: Ascendientes propios de N: I,E,B,A
  - Ejemplo: Descendientes propios de C: F,G,H,J,K,L
- El **Tamaño** de un Nodo es su número de descendientes.
- El Tamaño de un Árbol es el tamaño de su Nodo Raíz.
  - Ejemplo: Tamaño del Nodo C: 7
  - Ejemplo: Tamaño del Árbol: 14

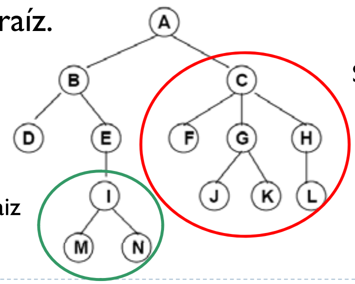
Prueba de Conceptos de Árbol



1. ¿Hermanos de D?:
   - E
2. ¿Tamaño de E?:
   - 4
3. ¿Ascendientes de G?:
   - G, C y A
4. ¿Profundidad de K?:
   - 3
5. ¿En que Nivel está E?:
   - 2
6. ¿Altura de B?:
   - 3
7. ¿Descendientes propios de G?:
   - J y K

Definición Recursiva de un Árbol

- Un Árbol es:
  - Un Árbol vacio, o
  - Un Nodo Raíz y cero o más sub-Árboles no vacios donde cada una de sus raíces está conectada por medio de una Arista con la Raíz del Árbol.
- Cada Nodo de un Árbol define un subárbol que es aquel que le tiene por raíz.

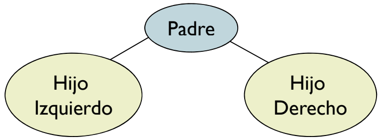


Subárbol cuya raiz es el nodo C

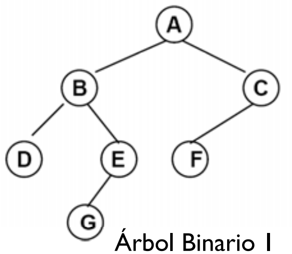
Subárbol cuya raiz es el nodo I

Árboles Binarios

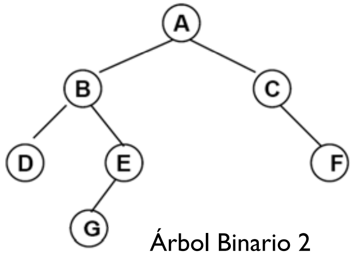
- Un **Árbol Binario** es un Árbol en el que ningún Nodo puede tener más de 2 Hijos



¿Cual es la diferencia entre los dos Árboles Binarios?



Árbol Binario 1



Árbol Binario 2

Árboles Binarios: Propiedades

- En un Árbol Binario de Altura H:
  1. El número máximo de Nodos del Nivel i es 2^i (0<= i <= H)
  2. El número máximo de Nodos es:

$$\sum_{i=0}^{H} 2^i = 2^{H+1} - 1$$

  3. El número máximo de Hojas es:

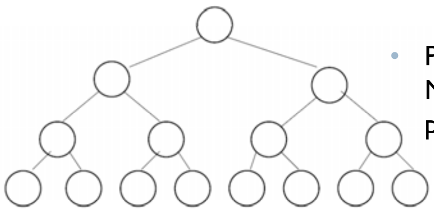
$$(2^{H+1} - 1) - (\sum_{i=0}^{H-1} 2^i) = 2^H$$

  4. El número máximo de Nodos Internos es:

$$(2^{H+1} - 1) - (2^H) = 2^H - 1$$

Árbol Binario Lleno

- Un Árbol Binario de Altura H es **Lleno** si tiene todos sus Niveles completos.
  - En cada Nivel i, 0 <= i <= H, tiene 2^i Nodos.
- Ejemplo de Árbol Binario Lleno, con Altura 3:



- Para un Árbol Binario Lleno de Tamaño N y Altura H, se cumplen las propiedades:

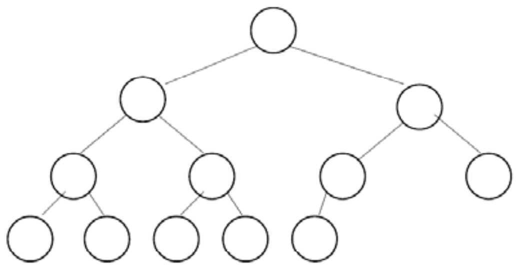
$$H = \lfloor \log_2 N \rfloor$$

$$N = 2^{H+1} - 1$$

- Árbol de Ejemplo:
  H = floor(log_2(15)) → 3

Árbol Binario Completo

- Un Árbol Binario es **Completo** si tiene todos sus Niveles completos a excepción, quizás del último que, entonces, debe de tener situados todos sus Nodos, las Hojas del Árbol, tan a la izquierda como sea posible.
- Ejemplo de Árbol Binario Completo:



- El Nivel 3 no está completo pero todas sus hojas están lo más a la izquierda posible.

Árbol Binario Completo: Propiedades

- Todo Árbol Binario Lleno es Completo, aunque no es cierta la afirmación inversa.
- Por lo tanto, se cumplen las siguientes propiedades:
  - Altura de un Árbol Binario Completo:

$$H \leq \lfloor \log_2 N \rfloor$$

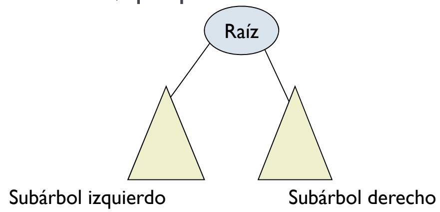
  - Número de Nodos de un Árbol Binario Completo de altura H:

$$2^H \leq N \leq 2^{H+1} - 1$$

  - La relación entre Altura (H) y Número de Nodos (N) de un AB Completo (o Lleno) determina si está Equilibrado:
    - Sus N datos se distribuyen en el árbol tal que la longitud de su camino más largo (i.e. Altura) está acotada por floor(log_2(N)).

Árbol Binario: Definición Recursiva

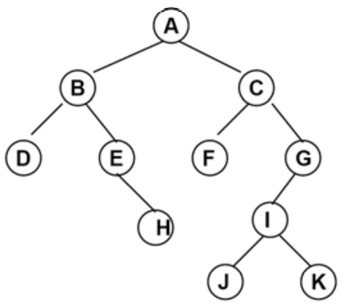
- Un Árbol Binario es:
  - Un Árbol Binario vacío o
  - Un Nodo Raíz y dos sub-Árboles Binarios, uno Izquierdo y otro Derecho, que pueden ser vacíos.



- La definición recursiva permite diseñar las operaciones que trabajan con un Árbol Binario fácilmente de manera recursiva, trabajando sobre su Nodo Raíz.

Árbol Binario: Recorridos

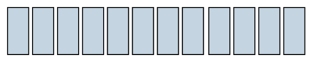
- Dado que un Nodo se identifica con el Árbol del que es Raíz, el recorrido en profundidad puede seguir el siguiente orden de visita:
  - Pre-Orden: Nodo → SubÁrbol Izquierdo → SubÁrbol Derecho
  - In-Orden: SubÁrbol Izquierdo → Nodo → SubÁrbol Derecho
  - Post-Orden: SubÁrbol Izquierdo → SubÁrbol Derecho → Nodo



- Recorridos en profundidad desde la raíz del Árbol.
  - Pre-Orden: A-B-D-E-H-C-F-G-I-J-K
  - In-Orden: D-B-E-H-A-F-C-J-I-K-G
  - Post-Orden: D-H-E-B-F-J-K-I-G-C-A
- Recorrido en Anchura (por Niveles) desde la raíz del Árbol:
  - A-B-C-D-E-F-G-H-I-J-K

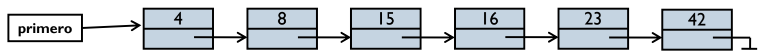
Árbol Binario de Búsqueda. Motivación: Búsqueda Dinámica (I)

- Organizar una colección de N elementos:
  - Mediante un Vector:



¿Cuál es el coste de realizar una búsqueda con estas representaciones?

  - Mediante una Lista Enlazada:



- El coste de buscar en una EDA lineal es, en el peor de los casos proporcional a la talla de la colección: O(N)
  - Si el array está ordenado, empleando una búsqueda binaria, se puede llegar a obtener un coste, en el peor de los casos de O(log2(N))
  - Si conocemos el punto exacto de la estructura en el que se encuentra el dato, se accede en tiempo constante: Θ(1)

Árbol Binario de Búsqueda. Motivación: Búsqueda Dinámica (II)

- Organizar la colección de datos para permitir un acceso eficiente implica que cada inserción en la colección debe mantener esa organización.
  - Por ejemplo, mantenemos un vector ordenado para hacer eficiente el acceso a los elementos (coste logarítmico).
  - Queremos insertar el elemento 7

1	2	5	8	9

  - Se debe buscar la posición de inserción adecuada para mantener la propiedad de ordenación (Búsqueda Binaria: $O(\log_2(N))$)

1	2	5	8	7	9

¿Cuál es el coste de realizar la inserción en el array una vez encontrada la posición?

Árbol Binario de Búsqueda. Motivación: Búsqueda Dinámica (III)

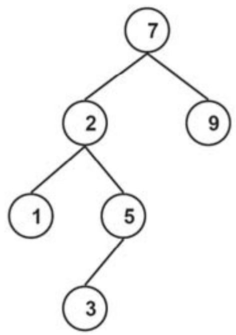
- Modelo de *Búsqueda Dinámica* :
  - Las operaciones insertar(x) y borrar(x), únicamente difieren de buscar(x) en la resolución de la búsqueda.
  - Queremos que su coste en el peor de los casos sea, como máximo **logarítmico** con el tamaño de la colección.
- Emplearemos el Árbol Binario de Búsqueda para organizar jerárquicamente los datos:
  - Usando cierta propiedad de ordenación.
  - Bajo ciertas condiciones de equilibrio permitirá implementar las operaciones de búsqueda con coste logarítmicos con el número de elementos, en el peor de los casos.

Árbol Binario de Búsqueda: Introducción (I)

- Un Árbol Binario de Búsqueda (ABB) permite representar los datos de una colección de forma jerarquizada y **ordenados** según cierto criterio.
  - Los Datos almacenados deberán implementar la interfaz **Comparable<T>**
- Un ABB es un Árbol Binario con la siguiente propiedad de orden:
  - Todos los datos de su subárbol izquierdo son menores (o iguales) que el ocupa su raíz.
  - Todos los datos de su subárbol derecho son mayores que el ocupa su raíz.
  - Los subárboles izquierdo y derecho también son ABB.

Árbol Binario de Búsqueda: Introducción (II)

- Ejemplo de Árbol Binario de Búsqueda (ABB):

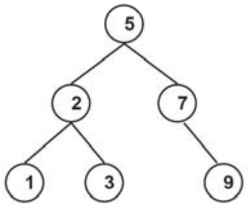


¡Verifica que realmente se trata de un ABB!

- En principio, la definición no contempla la existencia de datos repetidos, pero se puede modificar ligeramente para que sí lo soporte.

Árbol Binario Equilibrado

- Un Árbol Binario está **Equilibrado** o es **Perfectamente Balanceado** si la diferencia de alturas de los hijos derecho e izquierdo es, como máximo de 1.
- El ABB de la transparencia anterior NO es equilibrado.
  - Altura del hijo izquierdo de la raíz: 2
  - Altura del hijo derecho de la raíz: 0
- Ejemplo de ABB Equilibrado con los mismos datos:

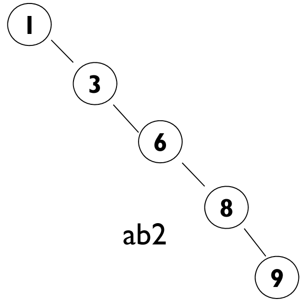


De manera intuitiva, ¿Cómo piensas que se ha conseguido equilibrar el árbol?

Propiedades de un Árbol Binario de Búsqueda

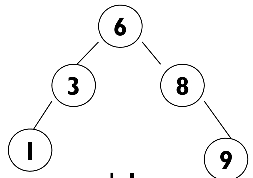
- Un ABB con N nodos tiene una altura entre $\log_2(N)$ y N-1

Ejemplo:



ab2

altura = 4



ab1

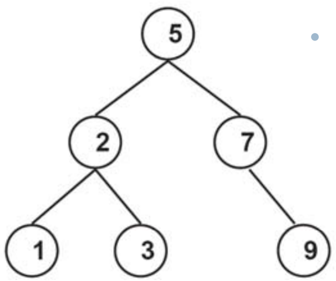
altura = 2

$\log_2(5)$ = 1.6094

Propiedades de un Árbol Binario de Búsqueda

- Si se listan sus Datos según un recorrido en In-Orden resulta una secuencia ordenada.

Ejemplo:



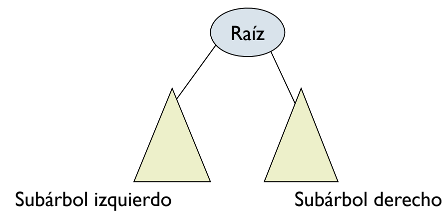
- El recorrido en In-Orden recorre:

  Subárbol Izquierdo – Nodo – Subárbol Derecho

- Recorrido en In-Orden:
  - 1 – 2 – 3 – 5 – 7 – 9

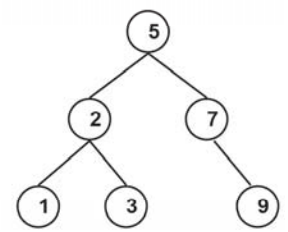
Propiedades de un Árbol Binario de Búsqueda

- El mínimo dato se encuentra en el nodo situado más a la izquierda.
  - Si existiera algún elemento más pequeño que él estaría situado en su subárbol izquierdo (por definición de ABB).
- El dato máximo se encuentra en el nodo situado más a la derecha.
  - Si existiera algún elemento más grande que él estaría situado en su subárbol derecho (por definición de ABB).



Búsqueda en un ABB

- La búsqueda compara el objeto *x* buscado con la raíz.
  - Si x es **menor** que la raíz, la búsqueda prosigue de manera recursiva por el subárbol **izquierdo**.
  - Si x es **mayor** que la raíz, la búsqueda prosigue de manera recursiva por el subárbol **derecho**.
- La búsqueda termina cuando se encuentra el elemento o cuando ya no quedan más nodos por visitar.



- La búsqueda del Integer(3) recorre los siguientes nodos del árbol:
  5 - 2 - 3
- La búsqueda del Integer(8) recorre los siguientes nodos del árbol:
  5 – 7 - 9

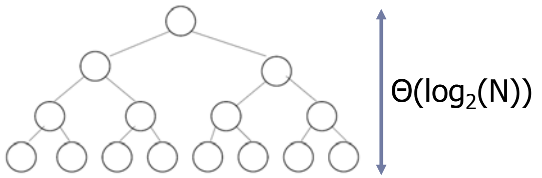
Búsqueda en un ABB: Complejidad Temporal

¿Hay instancias significativas?

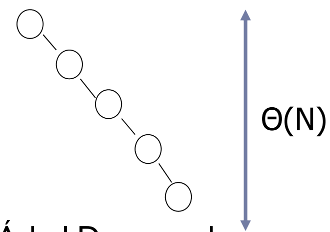
- Instancias significativas:
  - Caso Mejor: El elemento buscado está en la raíz del ABB.
    - Se realiza 1 comparación → Coste Constante
  - Caso Peor: El elemento está en cualquier hoja del nivel más alto o el elemento NO está en el ABB (y provoca el recorrido de la rama más larga).
    - Se realiza un número de comparaciones proporcional al número de niveles del ABB.
- En el peor de los casos, la **Complejidad Temporal** de la búsqueda en un ABB es **proporcional a la altura del árbol**.

Sobre la Altura de un ABB

- La Altura de un ABB varía entre Θ(log2(N)) y Θ(N).
  - Si el ABB es equilibrado, la altura es Θ(log2(N)) y, por lo tanto, el coste de la búsqueda en un ABB equilibrado es Θ(log2(N)) en el peor de los casos.
  - Si el ABB es degenerado, la altura es Θ(N) y, por lo tanto, el coste de la búsqueda en un ABB degenerado es Θ(N) en el peor de los casos.



Árbol Equilibrado



Árbol Degenerado

Búsqueda en un ABB: eMC

- Esfuerzo de Comparación: Número de comparaciones requerido para buscar un dato concreto en el Árbol.
  - En general, es *1 + nivelDelDato*
- **Esfuerzo Medio de Comparación** (eMC): Número promedio de comparaciones necesarias para encontrar cualquiera de los datos de un ABB concreto.

$$eMC = \frac{\sum_{nivel=0}^{H} (n^o Nodosnivel)(1 + nivel)}{N}$$

- El calculo del eMC a posteriori requiere un coste Θ(N):
  - Es necesario recorrer todos los nodos para contar los que hay en cada nivel y calcular la fórmula.

¿Cuál es el coste del cálculo del eMC a partir de un árbol dado?

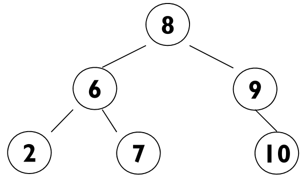
Estudio Experimental del Coste Medio de Búsqueda en un ABB (I)

- Interesa un eMC lo más bajo posible, para reducir el coste de búsqueda en ese ABB.
- El valor más bajo de eMC se obtiene para un ABB **perfectamente balanceado** (es el eMCOptimo).
- Dos formas de calcular el eMC:
  1. Cálculo explícito, empleando la fórmula vista.
  2. Cálculo acumulado, al realizar las inserciones en la clase ABB:
     - numTotalInserciones: Tamaño del árbol.
     - numTotalComparaciones: Número de comparaciones realizadas para la inserción de todos los elementos del ABB.
- Se obtiene el **Esfuerzo Medio de Comparación** (eMC):
  - eMC = numTotalComparaciones / numTotalInserciones

Estudio Experimental del Coste Medio de Búsqueda en un ABB (II)

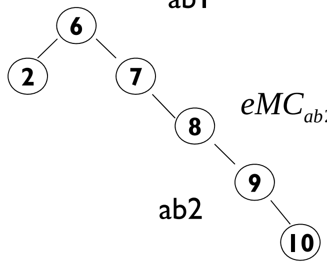
- Cálculo explícito del eMC

$$eMC = \frac{\sum_{nivel=0}^{H}(n^o Nodosnivel)(1+nivel)}{N}$$



ab1

$$eMC_{ab1} = \frac{1*1+2*2+3*3}{6} = \frac{14}{6} \approx 2.33$$



ab2

$$eMC_{ab2} = \frac{1*1+2*2+1*3+1*4+1*5}{6} = \frac{17}{6} \approx 2.83$$

Comparativa de Costes ABB frente Otros (I)

- Coste **Medio** (Asumimos ABB equilibrado).

Representación	Buscar	Insertar	Buscar Mínimo	Eliminar Mínimo
Lista Enlazada				
Lista Enlazada Ordenada				
Array Ordenado				
Árbol Binario de Búsqueda				

- N = Número de elementos de la colección.
- (*) Coste debido a mantener la contigüidad de los datos

Comparativa de Costes ABB frente Otros (I)

- Coste **Medio** (Asumimos ABB equilibrado).

Representación	Buscar	Insertar	Buscar Mínimo	Eliminar Mínimo
Lista Enlazada	$\Theta(N)$	$\Theta(1)$	$\Theta(N)$	$\Theta(N)$
Lista Enlazada Ordenada	$\Theta(N)$	$\Theta(N)$	$\Theta(1)$	$\Theta(1)$
Array Ordenado	$\Theta(\log_2(N))$	$\Theta(N)$*	$\Theta(1)$	$\Theta(N)$*
Árbol Binario de Búsqueda	$\Theta(\log_2(N))$	$\Theta(\log_2(N))$	$\Theta(\log_2(N))$	$\Theta(\log_2(N))$

- (*) Coste debido a mantener la contigüidad de los datos

Comparativa de Costes ABB frente Otros (II)

- Coste **Peor** (Asumimos ABB degenerado en una rama).

Representación	Buscar	Insertar	Buscar Mínimo	Eliminar Mínimo
Lista Enlazada	O(N)	O(1)	O(N)	O(N)
Lista Enlazada Ordenada	O(N)	O(N)	O(1)	O(1)
Array Ordenado	$O(\log_2(N))$	O(N)*	O(1)	O(N)*
Árbol Binario de Búsqueda	O(N)	O(N)	O(N)	O(N)

- (*) Coste debido a mantener la contigüidad de los datos