ÁRBOLES

Juan Pablo Silvestre (99127), Tomas Mancera (98649), Sebastián López (97500), Alejandro Salazar (97502), Juan Manuel Padilla (97196)

UNIVERSIDAD SAN BUENAVENTURA DE CALI

**UNIVERSIDAD DE SAN BUENAVENTURA CALI**

**FACULTAD DE INGENIERÍAS - PROGRAMA DE ING. SISTEMAS**

**ÁRBOLES**

**Introducción**

un grafo conexo que no contiene ciclos se llama árbol. Los árboles comenzaron a emplearse en 1857, cuando el matemático inglés Arthur Cayley los utilizó para contar cierto tipo de componentes químicos. Desde ese momento, los árboles se han empleado para resolver problemas en una gran variedad de disciplinas, como veremos en los ejemplos de este capítulo, Los árboles son particularmente útiles en informática, pues son empleados en un amplio espectro de algoritmos, Por ejemplo, los árboles se usan para construir algoritmos eficientes que localizan elementos en una lista. También pueden utilizarse en algoritmos, como los códigos de Huffman, que construyen códigos compresores eficientes, ahorrando costes en la transmisión de datos. y en su posterior almacenamiento, Los árboles se pueden emplear para estudiar juegos como las damas y el ajedrez y pueden ayudar a determinar estrategias ganadoras para cada uno de estos juegos. Los árboles pueden utilizarse para modelar procedimientos que se llevan a cabo mediante una secuencia de decisiones. Construir estos modelos puede ayudar a determinar la complejidad computacional de los algoritmos basados en una secuencia de instrucciones como los algoritmos de ordenación.

Los procedimientos para construir árboles que contengan todos los vértices de un grafo, incluyendo la búsqueda en profundidad y la búsqueda con anchura, pueden usarse para explorar sistemáticamente los vértices de un grafo. Recorrer los vértices de un grafo haciendo una búsqueda en profundidad, también conocida como de vuelta atrás”, es la base de la búsqueda sistemática de soluciones en una amplia variedad de problemas, tales como determinar el modo de colocar ocho reinas en un tablero de ajedrez de modo que dos cualesquiera no se amenacen.

Podemos modelar muchos problemas asignando pesos a las aristas de un árbol. Por ejemplo, utilizando árboles ponderados podemos desarrollar algoritmos para construir redes que contengan el conjunto menos costoso de líneas de teléfono que conectan distintos nodos de la red.

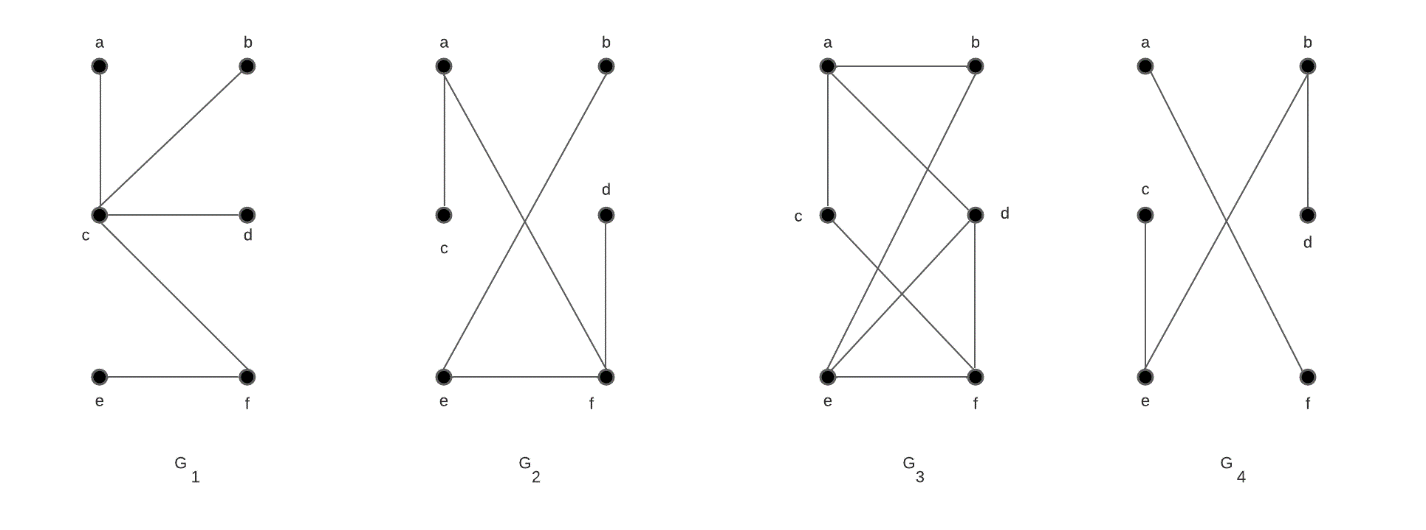
**Definición 1**

Un árbol es un grafo no dirigido, conexo y sin ciclos.

Puesto que un árbol no puede contener ciclos, es un grafo acíclico, tampoco puede tener bucles o aristas múltiples. Por tanto, un árbol es necesariamente un grafo simple.

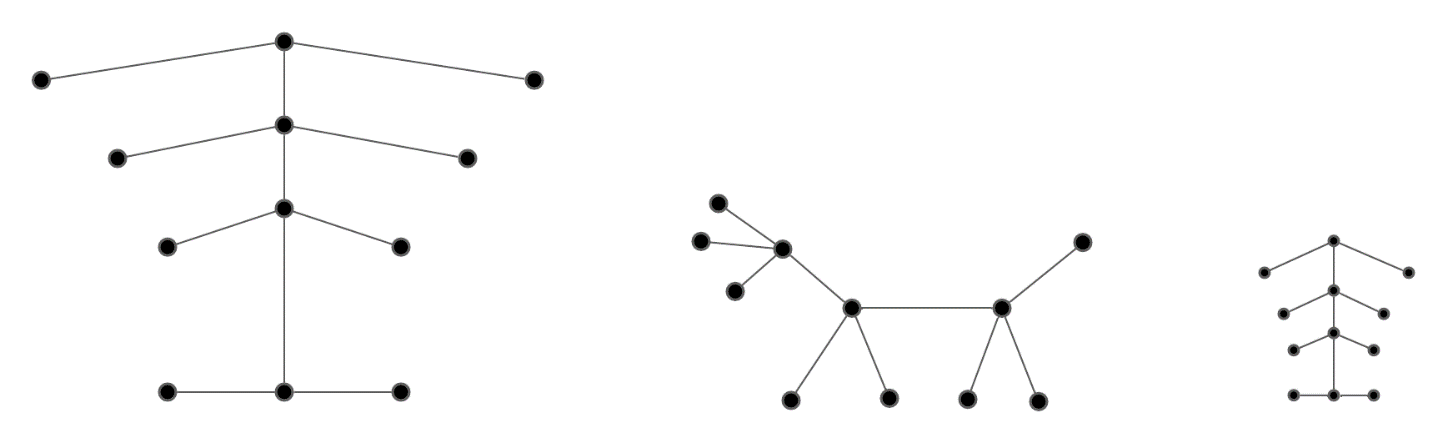
**Ejemplo 1**

¿Cuáles de los grafos de la figura son árboles?



Solución: G1 y G2 son árboles, puesto que ambos son grafos conexos y acíclicos. G3 no es un árbol, porque e, b, a, d, e es un ciclo del grafo. Finalmente, G4 no es un árbol, porque no es conexo.

Todo grafo conexo y acíclico es un árbol. ¿Qué podemos decir de los grafos acíclicos, pero no necesariamente conexos? Estos grafos se llaman bosques y tienen la propiedad de que cada una de sus componentes conexas es un árbol.



En muchas ocasiones, los árboles se definen como grafos no dirigidos con la propiedad de cada pareja de vértices está conectada por un único camino. El siguiente teorema prueba que está definición alternativa es equivalente a la nuestra.

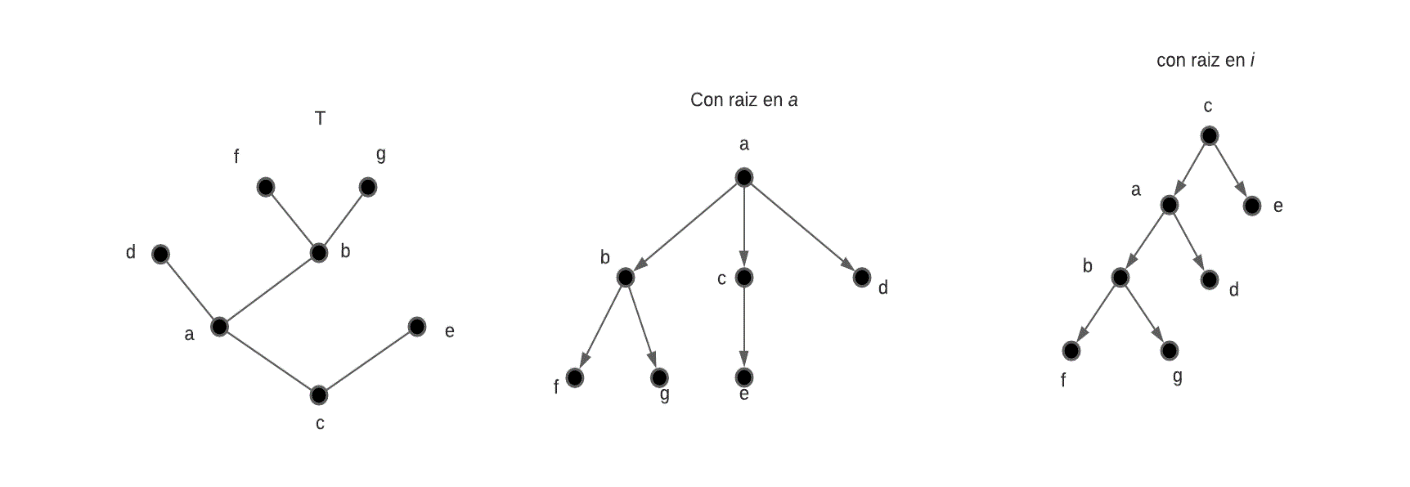
**Teorema 1**

Un grafo no dirigido es un árbol si, y solo si, hay un único camino entre cada pareja de vértices.

**Definición 2**

Un árbol con raíz es un árbol en el que uno de sus vértices ha sido designado como la raíz y todas están orientadas de modo que se alejan de la raíz. (Cada nodo tiene un solo padre, excepto el nodo raíz que no tiene padre).

Los árboles con raíz también se pueden definir recursivamente. Podemos transformar un árbol en un árbol con raíz sin más que elegir un vértice como raíz. Nótese que las distintas elecciones de la raíz producen diferentes árboles con raíz. Por ejemplo, la Figura 4 muestra los árboles con raíz. formados al designar como raíces los vértices a y c, respectivamente, en el árbol T. Normalmente, dibujaremos un árbol con raíz situando la raíz en la parte superior del grafo. Las flechas que indican el sentido de las aristas pueden omitirse en un árbol con raíz, puesto que la elección de la raíz determina las direcciones de las aristas.



La terminología de los árboles tiene orígenes botánicos y genealógicos. Supongamos que T es un árbol con raíz. Si v es un vértice de T distinto de la raíz, el padre de v es el único vértice u tal que hay una arista dirigida de u a v (el lector debería demostrar que tal vértice es único). Cuando u es el padre de v, se dice que v es hijo de u. Los vértices con el mismo padre se llaman hermanos. Los antecesores de un vértice diferente de la raíz son todos los vértices que aparecen en el camino desde la raíz hasta ese vértice, excluyendo a este último e incluyendo a la raíz.

Esto es, su padre, el padre de su padre y así hasta alcanzar la raíz. Los descendientes de un vértice v son aquellos vértices para los que v es un antecesor. Un vértice de un árbol se llama hoja si no tiene hijos.

Los vértices que tiene hijo se llaman vértices internos. La raíz es un vértice interno sino es el único vértice del grafo. En este caso, será una hoja.

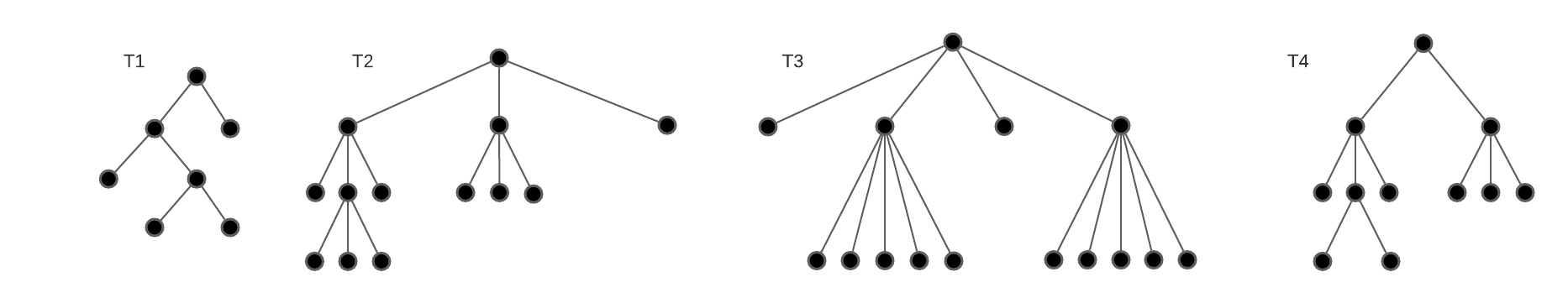
Si *a* es un vértice de un árbol, el subárbol con raíz en *a* es el subgrafo del árbol que contiene el vértice *a*, a todos sus descendientes y a todas las aristas incidentes en dichos descendientes.

**Definición 3**

Un árbol con raíz se llama árbol m-ario si todos los vértices internos tienen, a lo sumo, m hijos. El árbol se llama árbol m-ario completo si todo vértice interno tiene exactamente m hijos, un árbol m-ario con m = 2 se llama árbol binario.

**Ejemplo 3**

Los árboles con raíz de la figura, ¿Son árboles m-arios completos para algún entero positivo m?



**Solución:** T1, es un árbol binario completo, pues cada uno de sus vértices internos tiene dos hijos.

T2 es un árbol ternario completo, pues sus vértices internos tienen tres hijos. En T3, cada vértice interno tiene cinco hijos, de modo que T3, es un árbol 5-ario completo. T4, no es un árbol completo para ningún m, pues algunos nodos internos tienen dos hijos, mientras que otros tienen tres.

**Un árbol ordenado con raíz** es un árbol con raíz en el que los hijos de cada vértice interno están ordenados. Los árboles ordenados con raíz se dibujan de modo que los hijos de cada nodo interno se colocan ordenados de izquierda a derecha. Nótese que una representación de un árbol con raíz en el modo convencional determina un orden para sus aristas. Utilizaremos estas ordenaciones de las aristas en los dibujos sin mencionar explícitamente que estamos considerando un árbol ordenado con raíz.

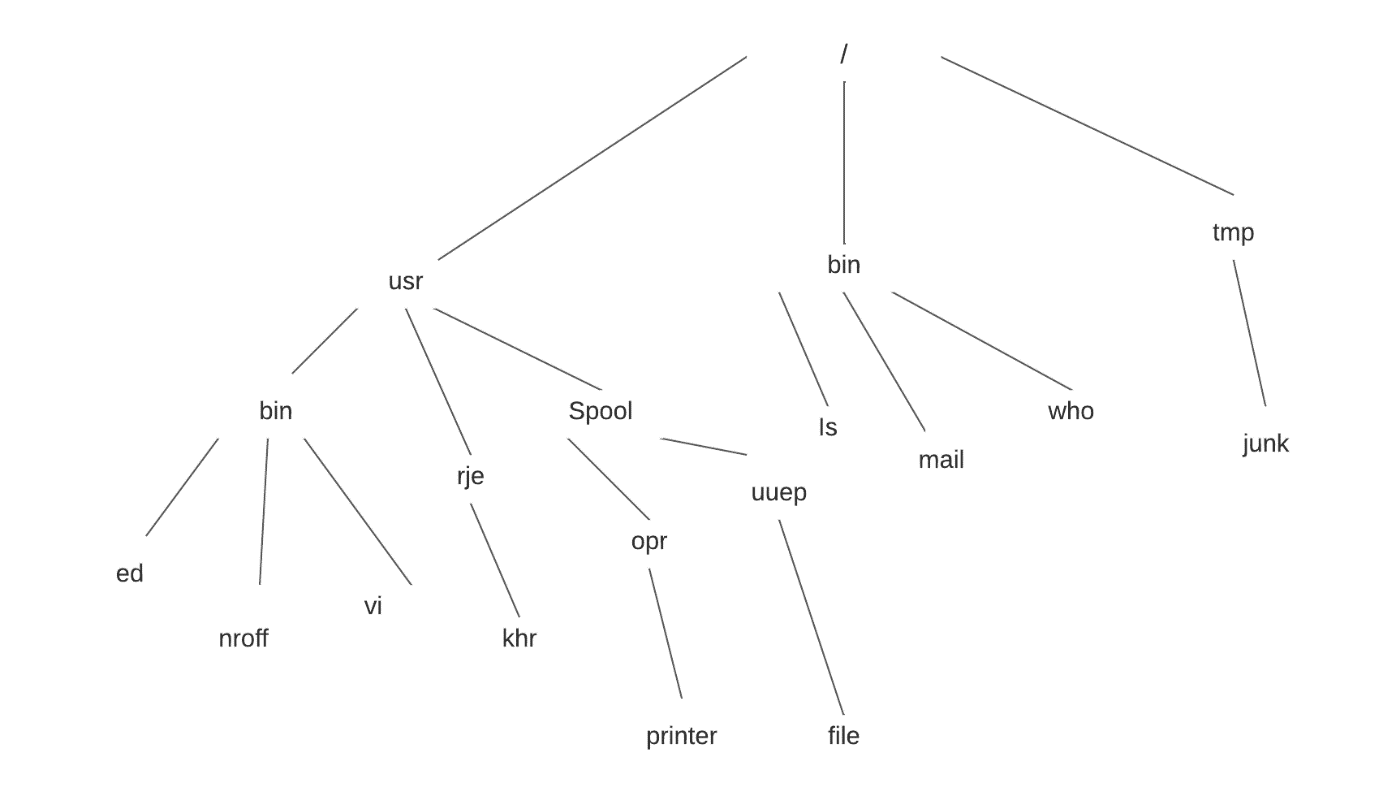
En un árbol binario ordenado (normalmente, llamado simplemente árbol binario), si cada vértice interno tiene dos hijos, el primer hijo se llama hijo izquierdo y el segundo se llama hijo derecho. El árbol con raíz en el hijo izquierdo de un vértice se llama subárbol izquierdo de este vértice y el árbol con raíz en el hijo derecho subárbol derecho.

**ÁRBOLES COMO MODELOS**

Los árboles se utilizan como modelos en ciencias tan diversas como la informática, la química, la geología, la botánica y la psicología. A continuación, describiremos varios modelos basados en árboles.

**Sistema de ficheros de ordenadores**

Sistemas de ficheros de ordenadores En la memoria de un ordenador, los ficheros se organizan en directorios. Un directorio puede contener tanto ficheros como subdirectorios. La raíz de un directorio contiene el sistema de ficheros completo. Así, un sistema de ficheros puede representarse mediante un árbol con raíz, donde la raíz representa el directorio raíz, los vértices internos representan los subdirectorios y las hojas representan los ficheros ordinarios o subdirectorios vacíos. Uno de tales sistemas de ficheros se muestra en la siguiente figura. En este sistema, el fichero khr está en el directorio rje. (Nótese que los enlaces con ficheros pueden dar lugar a circuitos si hay ficheros con más de una ruta de acceso).



**PROPIEDADES DE LOS ÁRBOLES**

Con frecuencia se necesitan resultados que relacionen los números de vértices de aristas en diferentes tipos de árboles.

**TEOREMA 2:** Un árbol de vértices, tienearistas.

Demostración: Se demostrará usando principio de inducción. Nótese que para cualquier árbol podemos elegir una raíz y considerar el árbol con raíz resultante.

Paso de inducción: La hipótesis de inducción afirma que todo árbol devértices tiene aristas, donde es un entero positivo. Supongamos que un árbol tiene vértices y es una hoja de (que debe existir puesto que el árbol es finito) y sea el padre de . Si eliminamos tanto el vértice como la arista que conecta a y se obtiene un árbol tiene aristas. De ahí se deduce que tiene aristas, ya que tiene una más que , la arista que conecta a y . Esto completa el paso de inducción.

El número de vértices de árbol -ario completo con un número prefijado de vértices internos está determinado, como muestra el teorema siguiente. Como el Teorema 2, denotaremos por al número de vértices de un árbol.

**TEOREMA 3:** Un árbol -ario completo con vértices internos tiene vértices.

**TEOREMA 4:** Un árbol -ario completo con

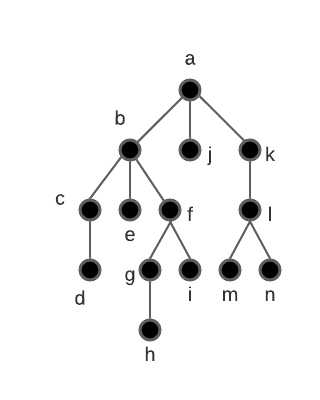
1. vértices tiene vértices internos y hojas.
2. vértices internos tiene vértices y hojas.
3. hojas tiene vértices e vértices internos.

**EJEMPLO:** Supongamos que alguien comienza una cadena de cartas. A cada persona que recibe una de esas cartas se le pide que la envíe a otras cuatro. Algunas personas lo hacen, pero otras no envían ninguna carta. ¿Cuántas personas han leído la carta, incluyendo a la primera persona, si nadie recibe más de una carta y si la cadena finaliza después de 100 personas que han visto la carta no hayan enviado ninguna? ¿Cuántas personas enviaron la carta?

**Solución:** La cadena de cartas se puede representar mediante un árbol 4-ario. Los vértices internos se corresponden con aquellas personas que enviaron la carta y las hojas con aquellas otras no la enviaron. Puesto que 100 personas no enviaron la carta, el número de hojas de este árbol con raíz es . De ahí, por el tercer apartado del Teorema 4, se tiene que el número de personas que han visto la carta es . También se tiene que el número de vértices internos es , así que 33 personas enviaron la carta.

Con frecuencia es preferible utilizar árboles con raíz que son “equilibrados” en el sentido de que los subárboles de cada vértice contienen caminos de aproximadamente la misma longitud. Posteriormente, se presentarán definiciones que clarificarán este concepto. El **nivel** de un vértice en un árbol con raíz es la longitud del único camino desde la raíz a dicho vértice. El nivel de la raíz, por definición, es cero. La **altura** de un árbol con raíz es la longitud del camino más largo desde la raíz a cualquier vértice.

**Ejemplo:** Calcular el nivel de vértice en el árbol con raíz de la siguiente la siguiente figura. ¿Cuál es la altura de este árbol?



**Solución:** La raíz está en el nivel 0. Los vértices y están en el nivel 1. Los vértices y están en el nivel 2. Los vértices y están en el nivel 3. Finalmente, el vértice está en el nivel 4. Puesto que el mayor nivel es el 4, el árbol tiene altura 4.

Un árbol con raíz -ario de altura está **equilibrado** o **balanceado** si todas sus hojas están en los niveles o

**Ejemplo:** ¿Cuáles de los árboles de la siguiente figura están equilibrados?

|  |  |
| --- | --- |
|  |  |
|  |  |

|  |
| --- |
|  |
|  |

**Solución:** está equilibrado, pues todas sus hojas están en los niveles 3 y 4. Sin embargo, no es un árbol equilibrado, pues sus hojas están en los niveles 2, 3 y 4. Finalmente, está equilibrado, pues todas sus hojas están en el nivel 3.

**TEOREMA 5:** Un árbol -ario de altura tiene, a lo sumo, hojas.

**COROLARIO 1:** Si un árbol -ario de altura tienehojas, entonces (Estamos utilizando la función *parte entera por exceso.* Recordamos es el menor entero mayor o igual que .

**APLICACIONES DE LOS ÁRBOLES**

**Introducción**

Discutiremos tres problemas que pueden estudiarse utilizando árboles. El primer problema es: ¿Cómo se pueden almacenar elementos en una lista de manera que todo elemento pueda ser localizado fácilmente? El segundo problema es: ¿Qué serie de decisiones deberíamos tomar para encontrar un objeto con una determinada propiedad en una colección de objetos de un cierto tipo? El tercer problema es: ¿Cómo se puede codificar de manera eficiente un conjunto de caracteres mediante una cadena de bits?

**Árboles Binarios de Búsqueda**

Los árboles binarios de búsqueda son una estructura de datos comúnmente utilizada en la informática para almacenar y buscar elementos. En matemáticas discretas, los árboles binarios de búsqueda se pueden entender en términos de teoría de conjuntos y relaciones de orden.

Un árbol binario de búsqueda es una estructura de datos que consta de nodos organizados en una estructura jerárquica. Cada nodo tiene un valor asociado y puede tener hasta dos hijos: un hijo izquierdo y un hijo derecho. Los valores de los hijos izquierdo y derecho son menores y mayores que el valor del nodo padre, respectivamente.

Podemos pensar en los nodos de un árbol binario de búsqueda como elementos de un conjunto ordenado. El nodo raíz del árbol representa el elemento más grande del conjunto, mientras que los nodos hoja representan los elementos más pequeños. Cada nodo intermedio representa un elemento que se encuentra entre el valor del nodo padre y los valores de sus hijos.

Para buscar un elemento en un árbol binario de búsqueda, comenzamos en el nodo raíz y comparamos el valor del nodo con el valor buscado. Si el valor del nodo es igual al valor buscado, hemos encontrado el elemento. Si el valor del nodo es mayor que el valor buscado, buscamos en el hijo izquierdo del nodo. Si el valor del nodo es menor que el valor buscado, buscamos en el hijo derecho del nodo. Repetimos este proceso hasta que encontremos el elemento buscado o lleguemos a un nodo hoja, lo que significa que el elemento no está presente en el árbol.

En conclusión, los árboles binarios de búsqueda son una herramienta importante en la informática para almacenar y buscar elementos de manera eficiente. En matemáticas discretas, se pueden entender en términos de conjuntos ordenados y relaciones de orden, lo que nos permite aplicar conceptos de teoría de conjuntos y relaciones de orden para entender mejor su funcionamiento.

**Ejemplo 1**

Construye un árbol binario de búsqueda para las palabras matemáticas, paleontología, geografía, zoología, meteorología, geología, psicología y cosmología (utilizando el orden alfabético).

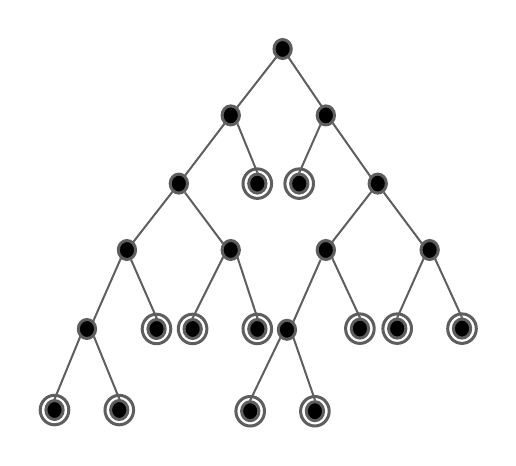
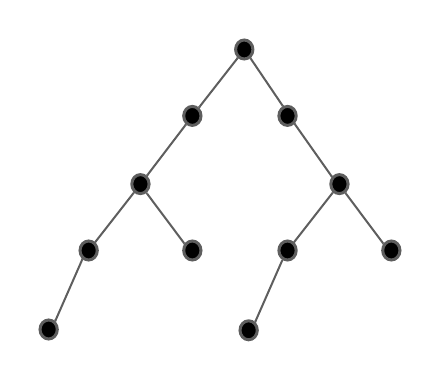
Solución: La Figura 1 muestra los pasos realizados en la construcción del árbol binario de búsqueda. La palabra matemáticas es la clave de la raíz. Puesto que paleontología es posterior a matemáticas (en el orden alfabético), añadimos el hijo derecho de la raíz con la clave paleontología. Puesto que geografía es anterior a matemáticas, añadimos el hijo izquierdo de la raíz con la clave geografía. Seguidamente, añadimos el hijo derecho del vértice etiquetado con paleontología y le asignamos la clave zoología, puesto que en el orden alfabético zoología es posterior a matemáticas y a paleontología. Análogamente, añadimos el hijo izquierdo del vértice de clave paleontología y le asignamos a este nuevo vértice la clave me meteorología. Añadimos el hijo derecho al vértice con clave zoología y le asignamos la clave psicología. Por último, incorporamos el hijo izquierdo del vértice con clave geografía y le asignamos la clave cosmología.

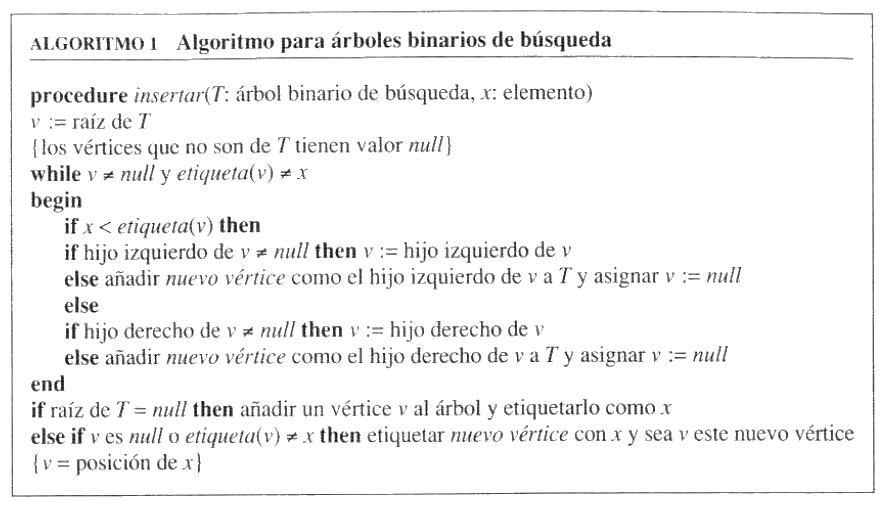
Para localizar un elemento intentamos añadirlo a un árbol de búsqueda binario. Lo localizamos si está presente en el árbol. El Algoritmo 1 proporciona un seudocódigo para localizar un elemento en un árbol binario de búsqueda y añadir un nuevo vértice con ese elemento como clave si éste no se encontrara en el árbol. El Algoritmo 1 localiza a x si es la clave de un vértice. Cuando x no es una clave, se añade un nuevo vértice con clave x. En el seudocódigo, v to el vértice que tiene a x como su clave y etiqueta(v) representa la clave del vértice v.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |

Vamos a determinar la complejidad computacional de este procedimiento. Supongamos que tenemos un árbol binario de búsqueda T par una lista de n elementos. Podemos construir, a partir de T, un árbol binario completo U sin más que añadir vértices no etiquetados cuando sea necesario de modo que cada vértice con clave tenga dos hijos. Este proceso se ilustra en la siguiente figura. Una vez hecho esto, podemos localizar fácilmente un elemento o añadir uno nuevo mediante una clave sin tener que añadir un vértice.

El mayor número de comparaciones necesarias para añadir un nuevo elemento es la longitud del camino más largo en U desde to raíz hasta una hoja. Los vértices internos de U son los vértices de T. De ahí se deduce que U tiene n vértices internos. Podemos utilizar la parte (2) del Teorema 4 de la sección 9.1 para concluir que U tiene n+ 1 hojas. Utilizando el Corolario I de la sección 9.1 obtenemos que la altura de U es mayor o igual que h = [log (n + 1)]. En consecuencia, es necesario realizar al menos [log (n + 1)] comparaciones para añadir un elemento. Nótese quo si U está equilibrado, su altura es [log (n + 1)] (por el Corolario I de la sección 9.1). Así, si un árbol binario de búsqueda está equilibrado, localizar o insertar un elemento no requiere más de [log (n + 1)] comparaciones. Un árbol binario de búsqueda puede dejar de estar equilibrado si se insertan nuevos elementos. Puesto que los árboles de búsqueda binarios y equilibrados proporcionan complejidad óptima en el peor caso para la búsqueda binaria, los algoritmos han sido diseñados de manera que el árbol de búsqueda se reequilibra a medida que se insertan elementos.





**ÁRBOLES DE DECISIÓN**

Los árboles con raíz pueden utilizarse para modelar problemas en los que una serie de decisiones llevan a una solución. Por ejemplo, un árbol binario de búsqueda puede emplearse para localizar elementos basándose en una serie de comparaciones, donde cada una de ellas nos dice si hemos localizado o no el elemento o si deberíamos ir hacia el subárbol derecho o hacia el subárbol izquierdo. Un árbol con raíz en el que cada vértice interno corresponde a una decisión, con un subárbol en dichos vértices para cada posible resultado de la decisión, se llama árbol de decisión. Las posibles soluciones del problema corresponden a los caminos desde la raíz hasta las hojas. El en siguiente ejemplo se ilustra una aplicación de los árboles de decisión.

**Ejemplo:** Supongamos que tenemos siete monedas. todas de igual peso, y una moneda falsa que pesa menos que las restantes. ¿Cuántas pesadas, utilizando una balanza, son necesarias para determinar cuál de las ocho monedas es la moneda falsa? Da un algoritmo para encontrar esa moneda.

**Solución**: Hay tres posibilidades para cada pesada en la balanza. Los dos platillos pueden tener igual peso. el primer platillo puede pesar más o puede pesar más el segundo platillo. Por tanto, el árbol de decisión, para la secuencia de pesadas es ternario. Hay, al menos, ocho hojas en el árbol de decisión, puesto que hay ocho posibles resultados (ya que cada una de las ocho monedas puede ser la falsa), y cada posible resultado debe estar representado por, al menos, una hoja. El mayor número de pesadas necesario para encontrar la moneda falsa es la altura del árbol de decisión. Por el Corolario 1 se sabe que la altura del árbol es al menos . De ahí, son necesarias al menos dos pesadas.

Se puede determinar qué moneda es la falsa utilizando dos pesadas. En la siguiente figura aparece el árbol de decisión que ilustra cómo hacerlo.

|  |
| --- |
| **COROLARIO 1:** Si un árbol -ario de altura tienehojas, entonces (Estamos utilizando la función *parte entera por exceso.* Recordamos es el menor entero mayor o igual que . |

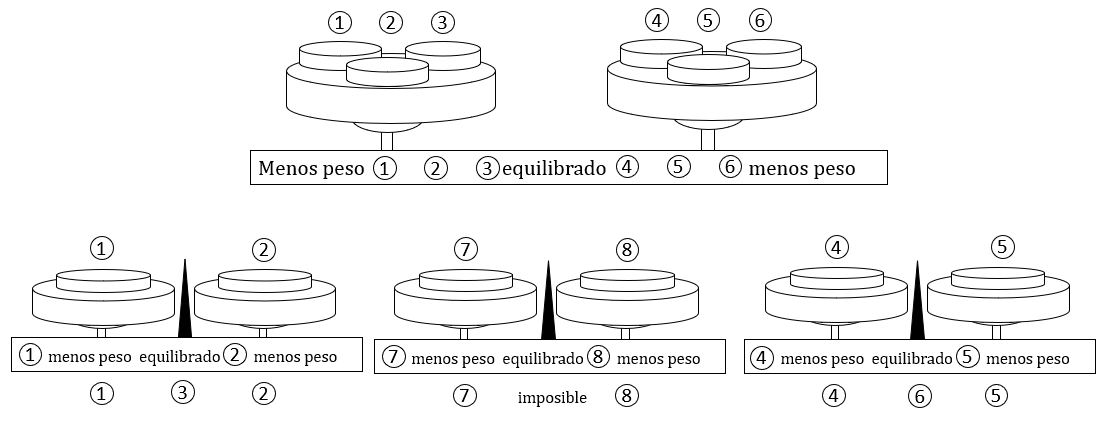


Figura: Un árbol de decisión para localizar una moneda falsa. La moneda falsa se señala en negrita debajo de cada una de las pesadas finales.

**LA COMPLEJIDAD DE LOS ALGORITMOS DE ORDENACIÓN**

Con el tiempo, se han desarrollado muchos y variados algoritmos de ordenación. Para decidir si un algoritmo de ordenación es eficiente o no, debemos calcular su complejidad. Utilizando los árboles de decisión como modelos, podemos calcular una cota inferior de la complejidad de los algoritmos de ordenación para el peor caso.

Nótese que dados elementos, hay formas posibles de ordenarlos, puesto que cada una de las permutaciones podría ser la ordenación correcta. Los algoritmos de ordenación con mayor frecuencia, están basados eh comparaciones binarias, esto es, la comparación de dos elementos en cada paso. El resultado de dicha comparación reduce el conjunto de órdenes posibles. Así, un algoritmo de comparación basado en comparaciones binarias se puede representar mediante un árbol de decisión en el que cada vértice interno representa una comparación de dos elementos. Cada hoja representa una de las permutaciones de los elementos.

**Ejemplo:**  en la siguiente figura se muestra un árbol de decisión que ordena los elementos de la lista

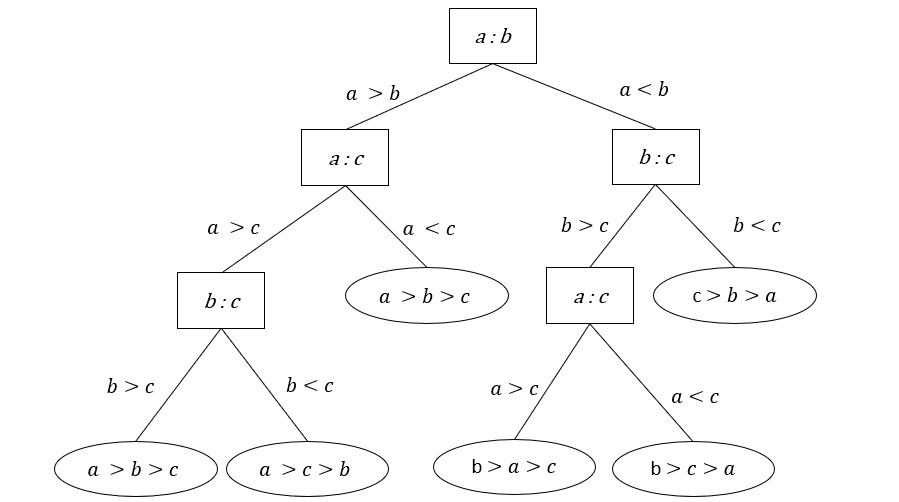


Figura: Un árbol de decisión para ordenar tres elementos distintos.

La complejidad de un algoritmo de ordenación basado en comparaciones binarias se mide en términos del número de comparaciones realizadas, El mayor número de comparaciones necesaria para ordenar una lista cualquiera de elementos proporciona el peor caso en la ejecución del algoritmo. El número máximo de tales comparaciones coincide con la longitud del camino más largo en el árbol de decisión que representa al procedimiento de ordenación. En otras palabras, e] mayor número de comparaciones necesarias para ordenar cualquier lista es precisamente la altura del árbol de decisión. Puesto que la altura de un árbol binario con hojas es al menos (utilizando el Corolario 1), se necesitan al menos comparaciones, como se establece en el Teorema 1.

**TEOREMA 1:** Un algoritmo de ordenación basado en comparaciones binarias requiere al menos comparaciones.

Podemos utilizar el Teorema 1 para dar una estimación en notación Ω del número de comparaciones realizadas por un algoritmo de ordenación basado en comparaciones binarias. Basta con observar que es , una de las funciones de referencia más comunes a la hora de estimar la complejidad computacional de algoritmos.

**TEOREMA 2:** El promedio del número de comparaciones realizadas por un algoritmo que ordena datos haciendo comparaciones binarias es .

**CÓDIGOS INSTANTÁNEOS**

En teoría de la información, los códigos instantáneos son códigos prefijos que se utilizan para codificar secuencias de símbolos de longitud variable, de tal manera que no haya ninguna secuencia que sea prefijo de otra secuencia codificada.

En otras palabras, los códigos instantáneos aseguran que cada símbolo de la secuencia se codifica con una cantidad mínima de bits, sin la necesidad de utilizar un carácter especial para indicar el final de cada símbolo. Esto permite una compresión de datos más eficiente, ya que se pueden representar secuencias de símbolos más largas con menos bits que si se utilizaran códigos fijos de longitud constante.

Los códigos instantáneos se pueden representar gráficamente mediante un árbol, donde cada símbolo se asocia con un camino desde la raíz hasta una hoja. La longitud del camino que lleva a cada símbolo determina su longitud de codificación, y se garantiza que no hay ningún camino que sea prefijo de otro camino. De esta manera, se puede obtener una codificación sin ambigüedades y eficiente en términos de espacio de almacenamiento.

|  |  |
| --- | --- |
|  |  |

Figura: El árbol binario de código instantáneo.

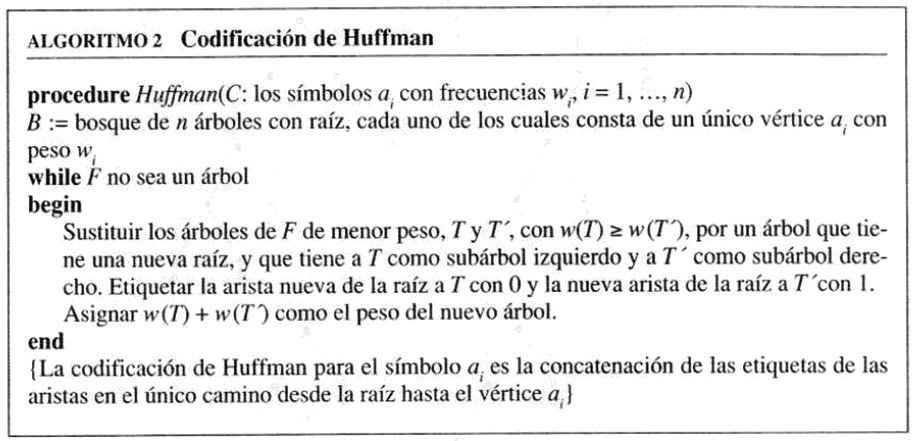
**CÓDIGOS DE HUFFMAN**

Los códigos de Huffman son un tipo de códigos de longitud variable utilizados en compresión de datos. Fueron inventados por David A. Huffman en 1952 y son muy utilizados en la compresión de archivos de texto y otros tipos de datos.

El objetivo de los códigos de Huffman es asignar códigos más cortos a los símbolos que aparecen con mayor frecuencia en un conjunto de datos y códigos más largos a los símbolos menos frecuentes. De esta manera, se puede lograr una compresión más eficiente, ya que los símbolos más frecuentes tendrán códigos más cortos y, por lo tanto, se representarán con menos bits.

El proceso para generar los códigos de Huffman implica construir un árbol de codificación basado en la frecuencia de aparición de cada símbolo en el conjunto de datos. Los símbolos se colocan en las hojas del árbol y se fusionan gradualmente hacia arriba, generando códigos de longitud variable a medida que se sube por el árbol. Los códigos se asignan a cada símbolo de forma tal que los códigos asignados a cada símbolo sean únicos y no haya códigos que sean prefijos de otros códigos.

Una vez que se ha construido el árbol de codificación, se pueden utilizar los códigos de Huffman para comprimir los datos. Cada símbolo en el conjunto de datos se reemplaza por su correspondiente código de Huffman antes de ser almacenado o transmitido. Como los símbolos más frecuentes tienen códigos más cortos, esto reduce la cantidad de bits necesarios para representar los datos, lo que resulta en una mayor eficiencia de almacenamiento o transmisión.



**Ejemplo:** Utiliza la codificación de Huffman para cifrar los siguientes símbolos con las frecuencias dadas: A:0.08, B:0.10, C:0.12, D:0:15, E:0.20, F:0.35. ¿Cuál es el promedio de bits utilizado para codificar un carácter?

**Solución:** En la siguiente figura se muestran los pasos realizados para codificar estos símbolos. Las codificaciones resultantes son las siguientes: la de A es 111 la de B es 110, la de C es 011, la de D es 010, la de E es 10y la de F es 00. El número promedio de bits utilizado para codificar un símbolo es:

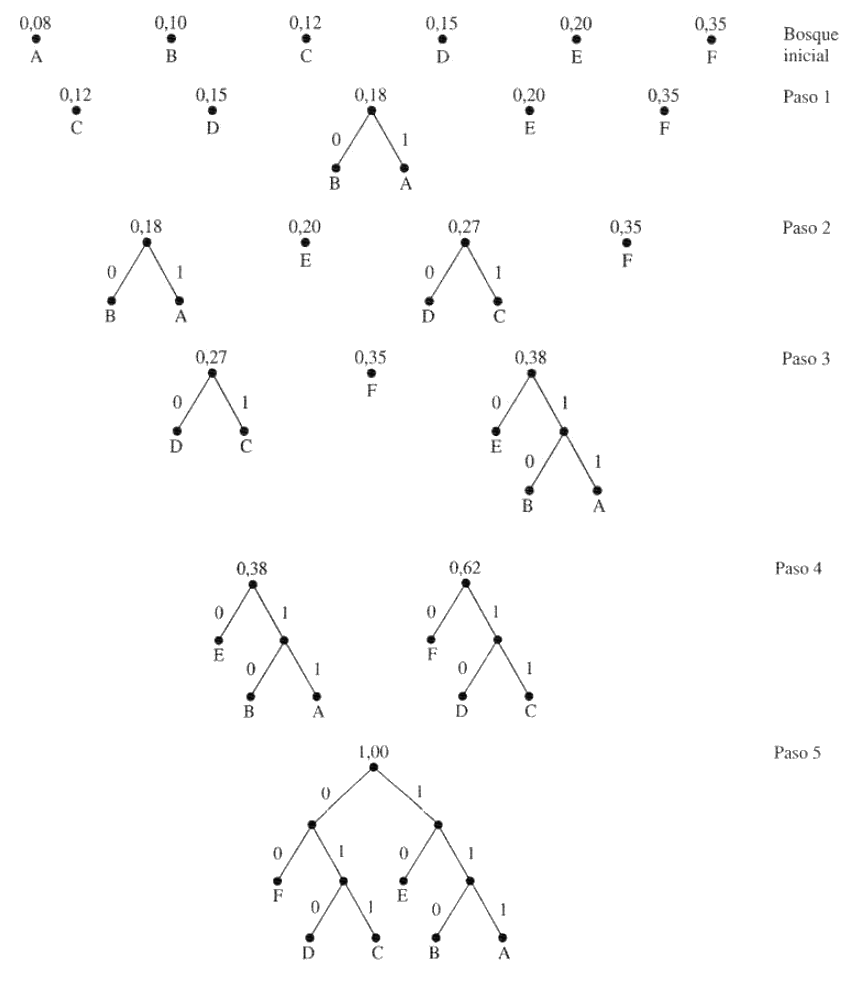


Figura: Codificación de Huffman de los símbolos del anterior ejemplo.

**ARBOLES DE JUEGOS**

En teoría de juegos, los árboles de juegos son estructuras de datos que se utilizan para representar gráficamente un juego, con el fin de analizar las diferentes opciones y estrategias disponibles para los jugadores.

Un árbol de juegos se construye mediante la representación de cada estado posible del juego en un nodo del árbol. A partir de cada nodo, se dibujan ramas que representan las diferentes jugadas posibles que se pueden realizar desde ese estado. Cada rama está etiquetada con la jugada que se realiza y con el resultado que se obtiene al realizar esa jugada.

Los árboles de juegos se utilizan para analizar diferentes estrategias y tomar decisiones informadas. Por ejemplo, un jugador puede explorar el árbol de juegos para determinar la mejor jugada posible en cada estado del juego. También se pueden utilizar técnicas como la estrategia de minimax y la poda alfa-beta para buscar el camino óptimo en el árbol de juegos.

En general, los árboles de juegos son útiles para representar y analizar juegos de dos o más jugadores, en los que cada jugador toma decisiones que afectan el resultado final del juego. Algunos ejemplos de juegos que se pueden representar mediante árboles de juegos son el ajedrez, el póker y el tic-tac-toe.

**Tres en raya.** El árbol de juego para el tres en raya es extremadamente largo y no puede dibujarse aquí, aunque cualquier ordenador puede construirlo fácilmente. Mostramos una parte del juego en la Figura G. Nótese que, si eliminamos las posiciones simétricas equivalentes, sólo necesitamos las tres posiciones iniciales de la Figura G. También mostramos un subárbol de este árbol de juego que lleva a la posición terminal de la Figura H, donde un jugador que puede ganar realiza un movimiento ganador.

|  |
| --- |
|  |
| Figura G. |

|  |
| --- |
|  |
| Figura H. |

**Definición 1:** El valor de un vértice en un árbol de juego se define recursivamente como:

1. El valor de una hoja es el pago del primer jugador cuando el juego finaliza en la posición representada por esa hoja.
2. El valor de un vértice interno en un nivel par es el máximo de los valores de sus hijos y el valor de un vértice interno de nivel impar es el mínimo de los valores de sus hijos.

**TEOREMA 3:** El valor de un vértice de un árbol de juego nos da la puntuación obtenida por el primer jugador si ambos jugadores siguen la estrategia de minimax y el juego comienza desde la posición representada por dicho vértice.

**MINIMAX**

La estrategia de minimax es un algoritmo utilizado en teoría de juegos para encontrar la mejor jugada en juegos de suma cero con información perfecta, donde dos jugadores compiten por obtener el mejor resultado posible en un juego determinado.

En la estrategia de minimax, se representa el juego mediante un árbol de búsqueda, donde cada nodo representa un estado del juego y las ramas que salen del nodo corresponden a las posibles jugadas que puede realizar el jugador en ese estado. En cada nodo del árbol, se calcula el valor minimax, que representa la mejor respuesta que el jugador contrario puede dar a la jugada actual.

El jugador maximizador trata de maximizar el valor minimax, es decir, elegir la jugada que maximiza su resultado, mientras que el jugador minimizador trata de minimizar el valor minimax, es decir, elegir la jugada que minimiza el resultado del jugador maximizador.

De esta manera, la estrategia de minimax se basa en la suposición de que el oponente también está tratando de maximizar sus ganancias, por lo que el jugador debe considerar las posibles respuestas del oponente para tomar la mejor decisión posible en cada turno.

En resumen, la estrategia de minimax en árboles de juegos es un enfoque para encontrar la mejor jugada en juegos de dos jugadores, donde el jugador maximizador trata de maximizar.

BIBLIOGRAFÍA

* *MATEMATICA DISCRETA Y SUS APLICACIONES (5a ED.)*. (2004, June 9). Casadellibro. <https://www.casadellibro.com.co/libro-matematica-discreta-y-sus-aplicaciones-5-ed/9788448140731/957996>