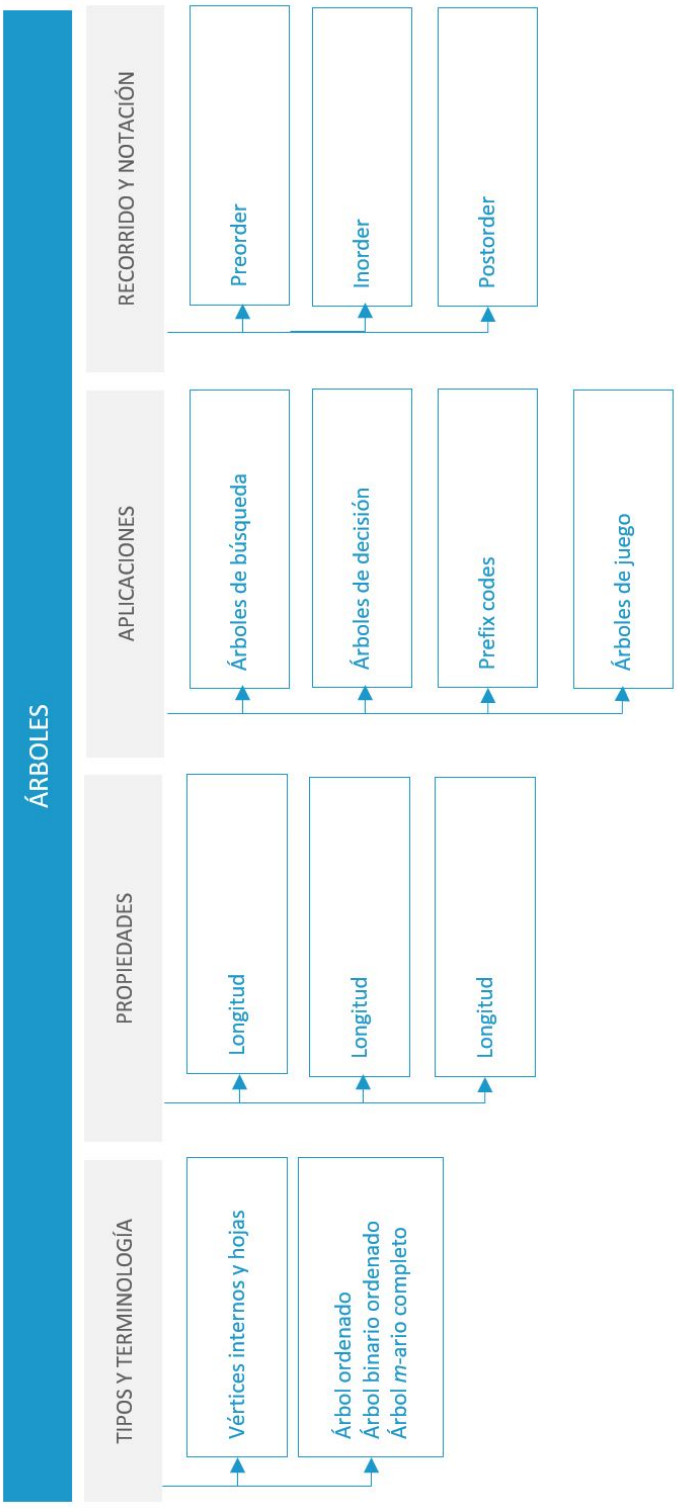
Tema 10

Álgebra y Matemática discreta

Árboles

Índice

Esquema


ÁRBOLES
TIPOS Y TERMINOLOGÍA
Vértices internos y hojas
Árbol ordenado
Árbol binario ordenado
Árbol m-ario completo
PROPIEDADES
Longitud
Longitud
Longitud
APLICACIONES
Árboles de búsqueda
Árboles de decisión
Prefix codes
Árboles de juego
RECORRIDO Y NOTACIÓN
Preorder
Inorder
Postorder

Ideas clave

10.1 Introducción y objetivos

En este último tema de la asignatura se aborda el concepto de Árbol, el cual se construye apoyado en la teoría de grafos, siendo un árbol un caso particular de grafo y sobre el que se definen una serie de propiedades, que permiten dar respuesta a numerosos problemas matemáticos, también en el ámbito de la ingeniería y la ingeniería informática.

En términos simples, un árbol es una estructura de datos que consiste en nodos conectados por aristas, que representan relaciones jerárquicas. Cada árbol tiene un nodo raíz, que es el nodo superior que no tiene padre, y el resto de los nodos se organizan en ramas o subárboles que se extienden desde la raíz.

En el ámbito de la ingeniería informática, los árboles se utilizan en una amplia variedad de aplicaciones, como en la optimización de algoritmos de búsqueda y ordenamiento, en la representación de estructuras de datos, en la construcción de compiladores y en la implementación de redes de computadoras, ya que, los árboles son estructuras de datos jerárquicas que permiten representar relaciones de orden entre elementos.

En la optimización de algoritmos de búsqueda y ordenamiento o en la construcción de compiladores, los árboles juegan un importante papel, al utilizarse para representar la estructura de un programa fuente de manera jerárquica, lo que facilita la generación de código objeto. También, tienen una gran utilidad en la implementación de redes de computadoras, ayudando a modelar la topología de la red y optimizar el enrutamiento de paquetes.

Un árbol a fin de cuentas, es el grafo conexo más simple y su aparición por primera vez, viene de la mano de GUSTAV ROBERT KIRCHHOFF (1824–1887) en sus trabajos sobre redes eléctricas. Más tarde, la teoría sobre árboles e incluso su denominación, la desarrolló ARTHUR CAYLEY (1821–1895), que en 1857 utilizó un tipo especial de grafo para enumerar diversos isómeros de los hidrocarburos saturados.

Con la llegada de los computadores, se dió un impulso a su utilización debido, como hemos visto, a las numerosas aplicaciones que tienen en el ámbito de la informática.

En concreto, en este capítulo, se pretende alcanzar los siguientes objetivos:

- Definir el concepto de Árbol .
- Propiedades de los árboles.
- Conocer las aplicaciones de los árboles.

Para alcanzar estos objetivos, se propone la siguiente subdivisión del contenido de este tema:

- Introducción y objetivos.
- Definiciones.
- Propiedades de los árboles.
- Aplicaciones.
- Recorrido de los árboles.

10.2 Definiciones

Comenzamos con la definición de árbol, para la que se da por supuesto haber estudiado el tema 9, en el que se apoyan muchas de las definiciones que se darán a continuación.

Definición 1

Sea $G = (V, E)$ un grafo (no digrafo). Se dice que G es un **árbol** si es conexo y no contiene ciclos.

Con frecuencia, se nombra un árbol por $T = (V, E)$.

Un árbol que conste de un solo vértice y sin lados es un árbol **degenerado**; un conjunto de árboles recibe el nombre de bosque, es decir, un grafo sin ciclos, si es conexo, es un árbol y si no lo es, sus componentes conexas serán árboles.

Cuando en un árbol se designa un vértice preferente del que parten las distintas ramas del árbol, a este se le llama **raíz**. Por tanto, en un árbol con raíz, podemos designar las direcciones de las aristas del árbol, siendo esta la que se aleja de la raíz. Así, un árbol junto con su raíz, genera un grafo dirigido que se conoce como árbol con raíz.

Definición 2

Un *árbol con raíz* es un árbol en el que uno de sus vértices ha sido designado como la *raíz* y rodas las aristas están orientadas de modo que se alejan de la raíz. Gráficamente, se suele dibujar un árbol con raíz, situando esta en la parte superior del dibujo.

Dado un árbol con raíz T y un vértice v distinto de la raíz, se define el **padre** de v, como

aquel vértice u tal que existe una arista dirigida de u a v. En esta situación, se dice que v es **hijo** de u. Los vértices que tienen el mismo padre u, se dice que son **hermanos**. Así, un padre u, puede tener distintos hijos $v_1, v_2, ..., v_n$, los cuales son hermanos entre si, pero cada vértice v, únicamente tiene un padre. Los vértices que conectan un vértice v con el vértice raíz, son los **antecesores** de v, sin contar el propio vértice pero si la raíz.

Los **descendientes** de un vértice son todos aquellos para los que v es un antecesor. Se llama *hoja de un árbol*, a aquel vértice que no tiene ningún hijo. Así mismo, se llama *vértice interno* a los que si tienen hijos. Dado un árbol T con un vértice a, se denomina **subárbol** de T con raíz en a, al que está formado por a como raíz e incluye todos los descendientes de a y las aristas que inciden en dichos descendientes.

Ejemplo 1.

En los árboles de la figura 1 indica para T_1 los hijos de c, los ascendentes de e y el padre de h; para el árbol T_2 indicar los ascendentes de j, el padre de c y los hermanos de i. Indica también cuales son vértices internos y cuales hojas.

Solución:

Para T_1, se tiene que los hijos de c son los vértices $g, h, i,$ y j. Los ascendentes de e son los vértices b y a y el padre de h es el vértice g.

Para el árbol T_2, se tiene que los ascendentes de j son los vértices i, d y a, el padre de c es a y los hermanos de i son los vértices g y h. Además, los vértices a, b, d e i, son vértices internos, mientras que el resto son hojas.

Un árbol se puede transformar en un árbol con raíz, simplemente designando uno de sus vértices como raíz. La elección de la raíz, dará lugar a árboles distintos en cada caso.

Definición 3

Un árbol con raíz se llama **árbol** $m-ario$ si todos los vértices internos tienen, a lo

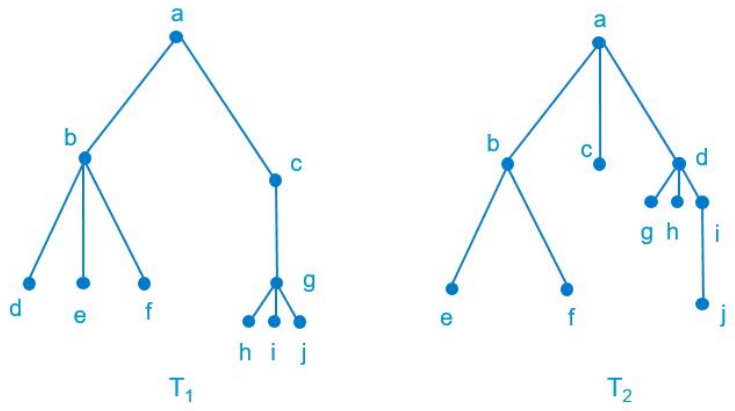


Figura 1: Árboles con raíz. Fuente: Elaboración propia

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**.

Se llama árbol ordenado a un árbol con raíz donde los hijos de cada vértice interno están ordenados, es decir, donde un recorrido *inorder*, como se verá mas adelante, devuelve los vértices ordenados. De forma particular, un árbol binario ordenado (normalmente llamado solo árbol binario) es un árbol ordenado con $m = 2$. Los árboles binarios ordenados son una estructura muy utilizada en programación.

10.3 Propiedades de los árboles

Teorema 1

Todo árbol con n vértices, tiene $n - 1$ arcos

Este teorema implica que dado un grafo G que cumple las condiciones necesarias para ser un árbol, es decir, ser conexo y no tener ciclos, implica siempre, que dicho grafo tendrá $n - 1$ arcos. Se puede demostrar fácilmente siguiendo el procedimiento de demostración por inducción visto en el tema 2 de la asignatura.

Teorema 2

Un árbol $m-ario$ completo con i vértices internos contiene exactamente $n = mi + 1$ vértices.

Atendiendo a este teorema, se puede obtener cualquiera de las tres variables que definen el tamaño de un árbol en función de una de ellas. Dichas variables son el número de vértices n, el número de vértices internos i y el número de hojas l.

Teorema 3

Un árbol $m-ario$ completo cumple que:

- Si tiene n vértices entonces tiene $i = (n-1)/m$ vértices internos y $l = [(m-1)n+1]/m$ hojas.
- Si tiene i vértices internos entonces tiene $n = m+1$ vértices y $l = (m-1)i+1$ hojas.
- Si tiene l hojas, entonces tiene $n = (ml-1)/(m-1)$ vértices y $i = (l-1)/8m-1)$ vértices internos.

Dado un árbol con raíz, se llama **longitud** del vértice u al vértice v al número de arcos entre ellos. Se llama **nivel** de un vértice v a la longitud desde la raíz a v. Se llama **altura** al máximo nivel de los vértices.

Un árbol *m-ario balanceado* es un árbol en el que la altura de dos subárboles difiere como mucho en una unidad y cada subárbol está balanceado. Es decir, los subárboles de un árbol balanceado contienen aproximada mente el mismo número de elementos, pero **no está completo**. Dicho de otra forma, un árbol con raíz *m-ario* de altura h está equilibrado o balanceado si tocias sus hojas están en los niveles h o $h-1$.

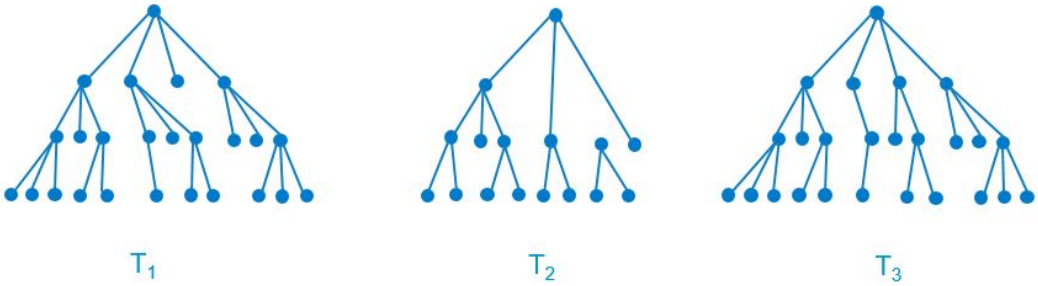


Figura 2: Árboles balanceados y no balanceados.

Teorema 4

Dado un árbol *m-ario* con altura h, existen en el como mucho m^h hojas.

Ejemplo 2.

Indicar la altura de las árboles T_1, T_2 y T_3 de la figura 2 e indicar si están equilibrados o no.

Solución:

El árbol T_1 tiene altura 4 y no está balanceado, ya que en la altura 2 existen una hoja. El arbol T_2, es en realidad un bosque, que que es un conjunto de dos árboles independientes. El árbol de mayor tamaño, tiene altura 3 y si está balanceado, al igual que el pequeño, que es en este caso de altura $h = 2$. Por último, el árbol T_3, tiene altura $h = 3$ y en este caso si está balanceado, pues todas las hojas se encuentran en los niveles $h = 4$ y $h - 1 = 3$.

10.4 Aplicaciones de los árboles

Tal como se señaló en la introducción, la teoría de árboles tiene numerosas aplicaciones, también en el ámbito de la ingeniería informática. Mostramos aqui cuatro ejemplos representativos de aplicaciones reales en las que se utiliza la teoría de árboles.

1. **Árboles binarios de búsqueda**. Una forma de ordenación especialmente útil para encontrar elementos eficientemente. En caso de tener que hacer inserciones resulta especialmente útil una variante llamada **árbol binario balanceado**.

2. **Árboles de decisión**. Es una herramienta de apoyo a la decisión donde la solución a un problema se modela como una serie de vértices internos corresponden a las decisiones y las hojas corresponden a la soluciones. Cada solución corresponde a un camino desde la raíz a una hoja.

3. **Prefix codes** (códigos instantáneos). Son una forma de codificación de longitud variable en la que el prefijo de un código no es parte del prefijo de otro código (Por esta razón posiblemente sería mejor llamarlos *«prefix-free code»*, pero la forma estándar en la literatura es *«prefix-code»*).
   Los *prefix codes* se usan frecuentemente el algoritmos de compresión y tienen la ventaja de que no hay ningún otro código con el mismo prefijo, con lo que se pueden decodificar códigos de longitud variable sin ambigüedad. Se puede usar un árbol binario para representar la decodificación de un prefix code.

4. **Árboles de juegos**. Los árboles se pueden utilizar para analizar las consecuencias de un movimiento en juegos como las damas, nim o ajedrez.

Para profundizar en la aplicación del árbol binario de búsqueda, puede visualizarse el siguiente video en el que se describe que es y cómo construir un árbol binario de búsqueda (ABB), y como se realizan las búsquedas, eliminaciones e inserciones, poniendo un ejemplo.

Accede al vídeo: Árbol Binario de Búsqueda

10.5 Recorridos de árboles

Para finalizar este tema, se desarrolla a continuación el concepto de recorrido de árboles, como una técnica utilizada para explorar y procesar los nodos de un árbol de manera ordenada y sistemática. Este método se utiliza en una amplia variedad de aplicaciones informáticas, como la búsqueda de elementos en estructuras de datos, la evaluación de expresiones matemáticas y la optimización de algoritmos.

Los procedimientos o algoritmos utilizados para recorrer todos y cada uno de los vértices de un árbol, están basados en la ordenación que tiene cada uno de los vértices o hijos. Pero antes de realizar un recorrido, es necesario etiquetar cada uno de los vértices. Para ello podemos proceder de forma recursiva como sigue:
En primer lugar , se etiqueta la raíz con el entero 0. Seguidamente, sus hijos tendrán de izquierda a derecha las etiquetas 1,2,...,k, para el nivel 1. Para cada vértice v den nivel con etiqueta A, se etiquetan a sus k_v hijos, también de izquierda a derecha, como $A.1, A.2, ..., A.k_v$. Este modo de etiquetar, se conoce como **sistema de etiquetado universal** o **canónico.**
En la figura 4 se puede ver un árbol con raíz ordenado en el que se han marcado las etiquetas siguiendo el sistema universal y cuyo orden lexicográfico es el siguiente:

$$0 < 101.1 < 1.2 < 1.3 < 2 < 3 < 3.1 < 3.1.1 < 3.1.2 < 3.1.3.1 < 3.1.2.2 < 3.1.2.3 < 3.1.3 < 3.2 < 4.4.1 < 5 < 5.1.1 < 5.2 < 5.3$$

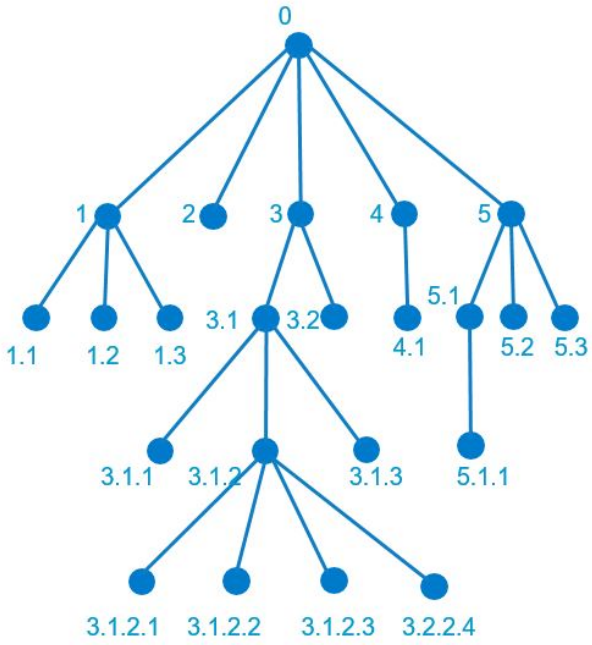


Figura 3: Árbol con etiquetado universal o canónico.Fuente: Elaboración propia

Recorridos

Se llama algoritmo de recorrido de un árbol a un procedimiento que nos permite visitar sistemáticamente cada vértice del árbol. Empezando por la raíz, cuando hay que interaccionar con un vértice con hijos, los recorremos siguiendo un algoritmo predefinido. Los tres algoritmos de recorrido más habituales son:

- **Preorder**: Cuando se visita primero la raíz, luego el vértice izquierdo y finalmente el vértice derecho.
- **Inorder**:Cuando se visita primero el vértice izquierdo, luego la raíz y finalmente el vértice derecho.
- **Postorder**: Cuando se visita primero el vértice izquierdo, luego el derecho y finalmente la raíz.

Cada uno de estos algoritmos de recorrido, se puede definir de forma recursvia.

Definición 4

Sea T un árbol ordenado con raíz r. Si T consta sólo del vértice r, entonces r es el recorrido en preorder de T. En otro caso, supongamos que $T_1, T_2, \cdots, T_n$, son los subárboles de r listados de izquierda a derecha en T. El *recorrido en preorder* comienza visitando r, continúa recorriendo T_1 en preorder, luego T_2 en preorder y así sucesivamente hasta recorrer T, en preorder.

En el recorrido preorder, el nodo raíz se visita primero, luego se visitan los nodos del subárbol izquierdo y finalmente se visitan los nodos del subárbol derecho.

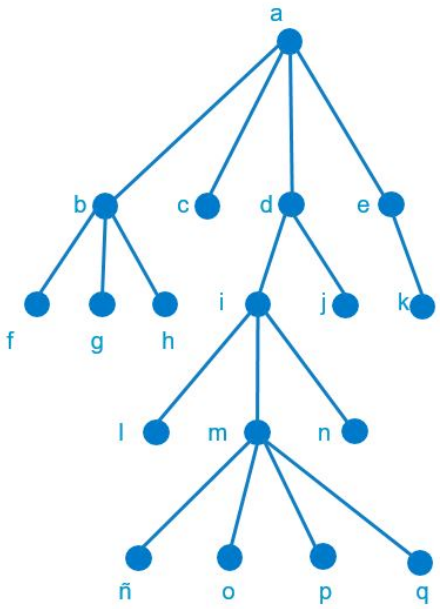


Figura 4: Árbol T.Fuente: Elaboración propia

Ejemplo 3.

¿En que orden se recorre el árbol T de la figura 4 siguiendo en preorder?

Solución:

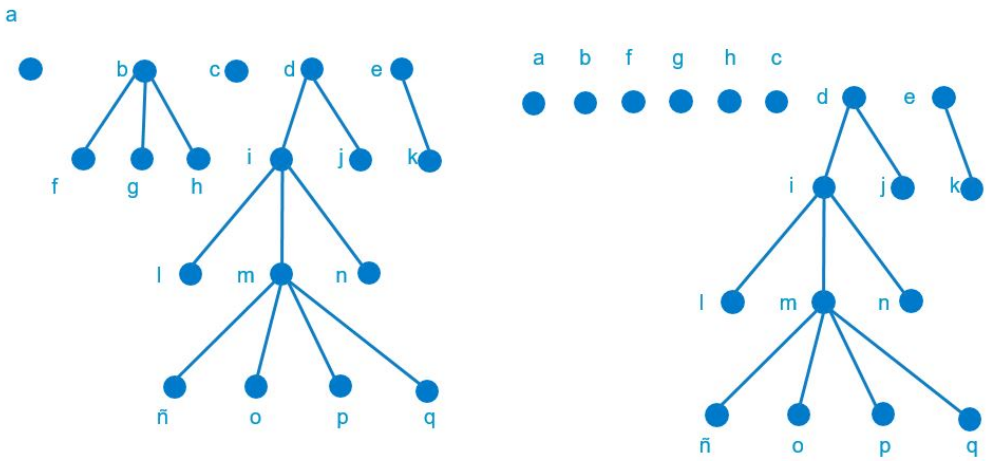
En primer lugar se lista la raíz a, seguida del subárbol de raíz b en preorder, después la lista de c, que es simplemente c. A continuación, en preorder la lista del subárbol d, que nos lleva a listar el subárbol de raíz i, en preorder, comenzando por l, el subárbol m en preorder ($\tilde{n}, o, p, q$), después n, j y el subárbol con raíz en

e que es k.

Por tanto, el recorrido en preorder del árbol de la figura 5 es:

a, b, f, g, h, c, d, i, l, m, ñ, o, p, q, n, j, e, k.

Los pasos seguidos en el recorrido preorder se muestran en la figura 6.



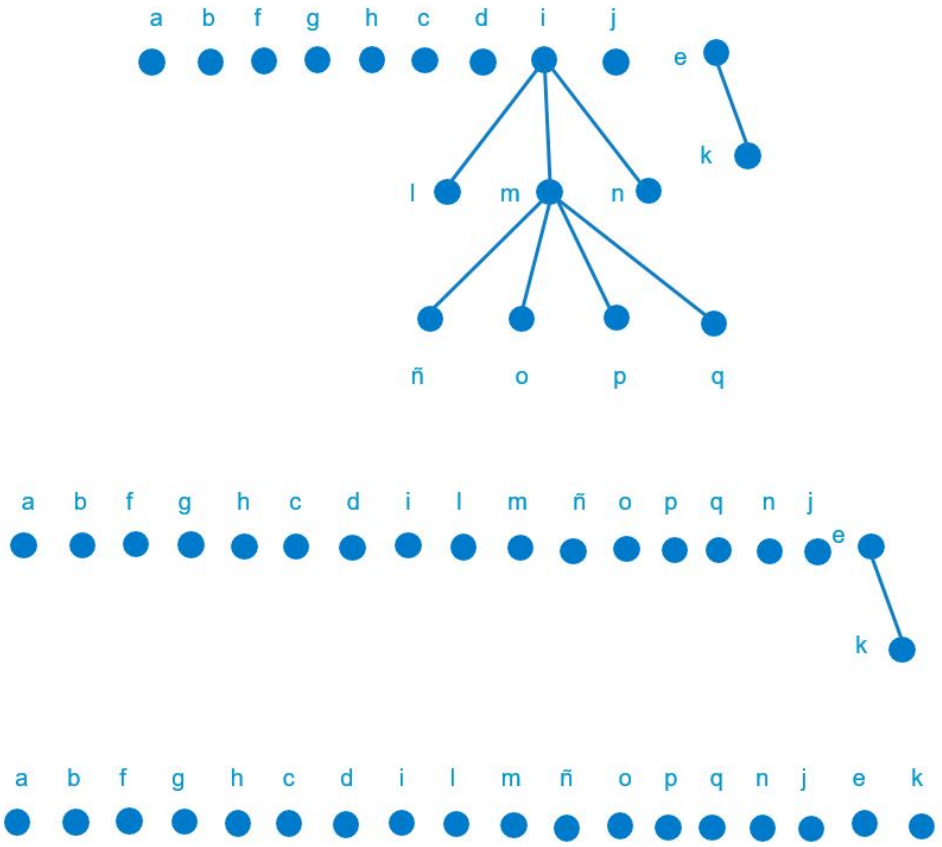


Figura 5: Pasos del recorrido preorder del árbol T.Fuente: Elaboración propia

Teorema 5

Sea T un árbol ordenado con raíz r. Si T consta sólo de r, entonces r es el recorrido en in orden de T. En otro caso, supongamos que $T_1, T_2 \cdots, T_n$ son los subárboles de r listados de izquierda a derecha en T. El *recorrido en inorder* comienza recorriendo T_1 en inorder y continúa visitando r, a continuación recorre T_2 en inorder, después T_3 y así hasta T_n en inorder.

En el recorrido inorder, los nodos del subárbol izquierdo se visitan primero, luego se visita el nodo raíz y finalmente se visitan los nodos del subárbol derecho.

Ejemplo 4.

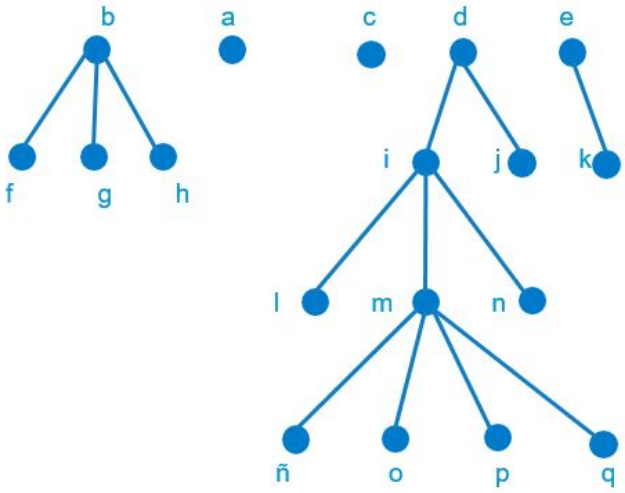
¿En que orden se recorre el árbol T de la figura 4 siguiendo en inorder?

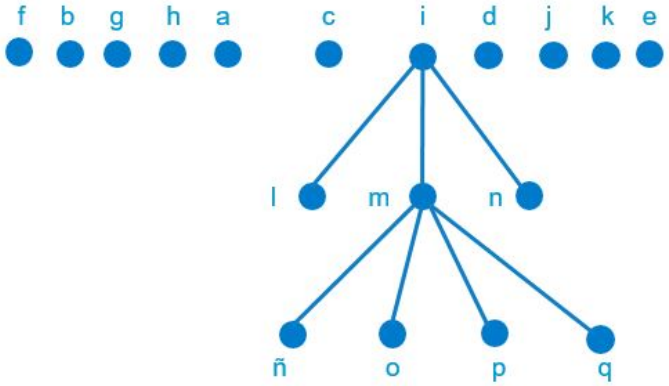
Solución:

El recorrido en inorder comienza con el recorrido del subárbol con raíz en b en inorder, luego la raíz a, posteriormente la lista de c, que es el mismo c, después en inorder y así sucesivamente, hasta llegar a la lista final:

f, b, g, h, a, c, l, i, ñ, m, o, p, q, n, d, j, k, e.

Se pueden ver los pasos recorridos en inorder en la figura 6





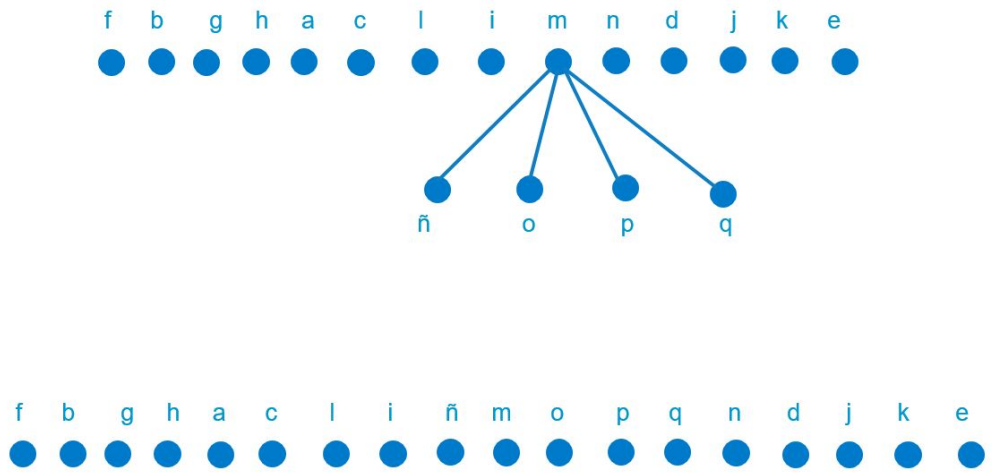


Figura 6: Pasos del recorrido inorder del árbol T.Fuente: Elaboración propia

Por último, definimos el tercero de los tres algoritmos de recorrido mas importantes, el recorrido en postorder:

Definición 5

Si el árbol T consta solo de la raíz r, entonces el recorrido postorder de T solo consiste en visitar la raíz r.

En caso contrario, si T tiene n subárboles $T_1, T_2, ..., T_n$, el recorrido postorder de T se realiza visitando primero los subárboles en postorder, comenzando por el subárbol izquierdo T_1, luego el subárbol T_2 y así sucesivamente hasta llegar al subárbol derecho T_n. Finalmente, se visita el nodo raíz r.

En otras palabras, el recorrido postorder de T se define recursivamente como sigue:

Realiza el recorrido postorder del subárbol T_1.

Realiza el recorrido postorder del subárbol T_2.

...

Realiza el recorrido postorder del subárbol T_n. $n+1$. Visita el nodo raíz r.

Cabe destacar que estas definiciones son específicas para árboles binarios, pero se pueden generalizar para árboles n-arios cambiando el orden en el que se visitan los hijos.

Los recorridos preorder e inorder son útiles en la implementación de algoritmos que procesan árboles y estructuras de datos como grafos y redes neuronales.

El recorrido postorder permite procesar los nodos hijos antes de procesar el nodo padre. También se utiliza en la eliminación de nodos de un árbol, ya que permite eliminar los nodos hijos antes de eliminar el nodo padre.

Ejemplo 5.

¿En que orden se recorre el árbol T de la figura 4 siguiendo en postorder?

Solución:

Los pasos para seguir el recorrido en postorder se pueden ver en la figura 7.

EL recorrido comienza por el subarbol b, seguido del c y posteriomente el d para finalizar con a. Cada uno de estos árboles, a su vez se recorren en postorder. El recorrido postorder del árbol es:

f, g, h, b, c, l, ñ, o, p, m, n, i, j, d, k, e, a.

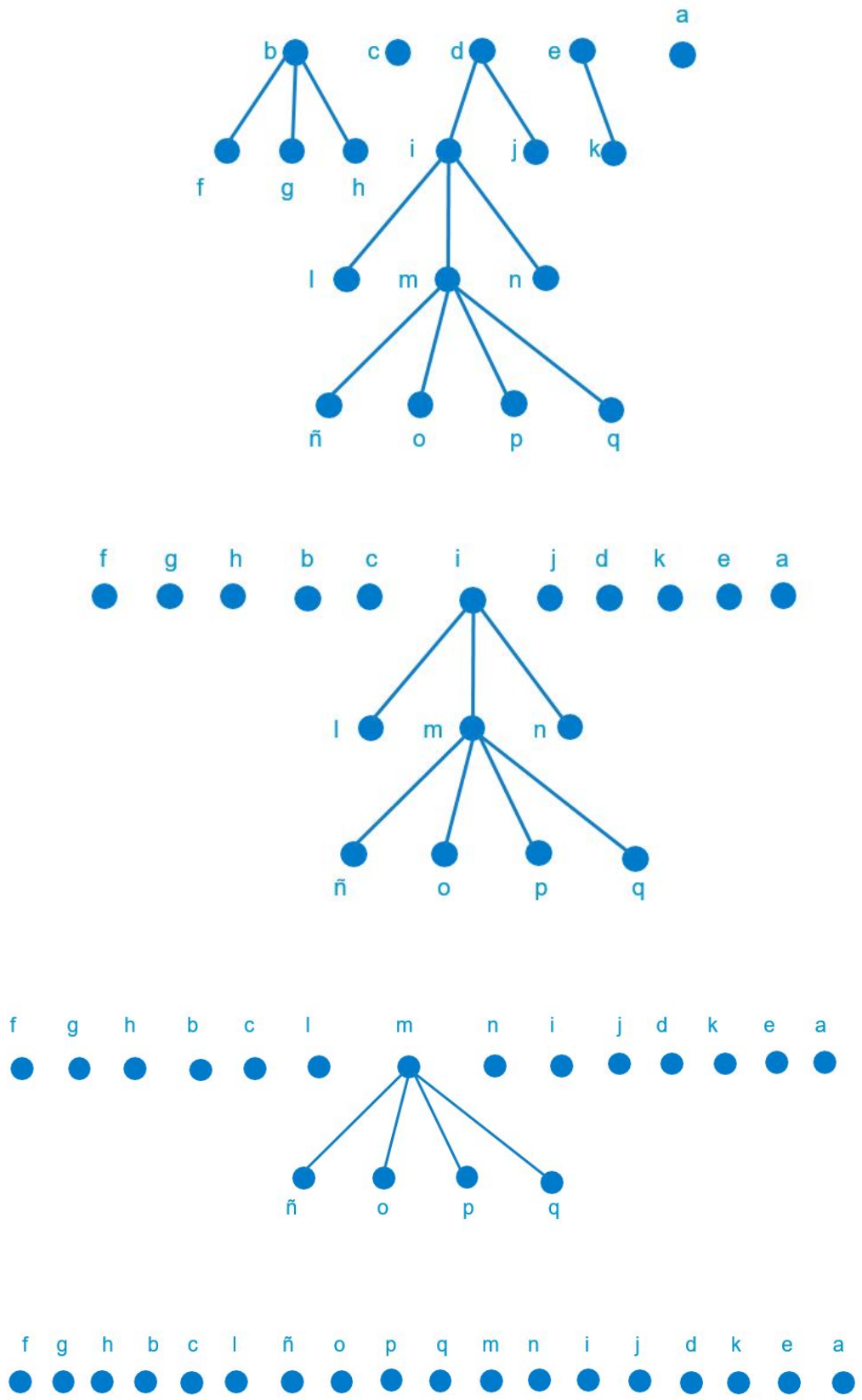


Figura 7: Pasos del recorrido postorder del árbol T. Fuente: Elaboración propia

En este vídeo se realizan los recorridos Preorder, Inorder y Postorder en un Árbol Binario de Búsqueda (ABB).

Accede al vídeo: Recorridos Preorder, Inorder y Postorder en un Árbol Binario de Búsqueda

Se puede usar un árbol con raíz para representar expresiones aritméticas donde los vértices internos representan las operaciones y las hojas representan las variables. Dado que varios árboles extraídos de diferentes expresiones aritméticas pueden dar lugar a la misma expresión aritmética, cuando se recorren inorder es necesario usar paréntesis siempre que encontramos un operador. A esta forma de recorrer incluyendo paréntesis se le llama **expresión infija** o **forma infija** de la expresión.

Análogamente se llama **forma prefija** al resultado de recorrer el árbol en preorder. A la forma prefija también se le llama *notación polaca*. La ventaja de la forma prefija respecto a la forma infija es que no es necesario incluir paréntesis durante el recorrido del árbol ya que la expresión obtenida no es ambigua.

Análogamente se llama **forma postfija** al resultado de recorrer el árbol en postorder. A la forma postfija también se le llama *notación polaca inversa*. En la notación polaca inversa tampoco es necesario incluir los paréntesis ya que el orden en que se ejecutan los operadores tampoco es ambiguo.

10.6 Referencias bibliográficas

Rosen K. (2012). Discrete.Mathematics.and ItsApplications. Mac Graw Hill.

Knuth, D. (2006). Art of Computer Programming. Addison-Wesley Professional..

Grimaldi, R.P., (2004). Discrete and combinatorial mathematics (5th ed). Pearson.

Valiente, G. (2007). Algorithms on Trees and Graphs. Springer.

10.7 Cuaderno de ejercicios

Ejercicio 1.

Identifica las hojas, vértices interiores y raíz de los árboles T y G de la siguiente figura:

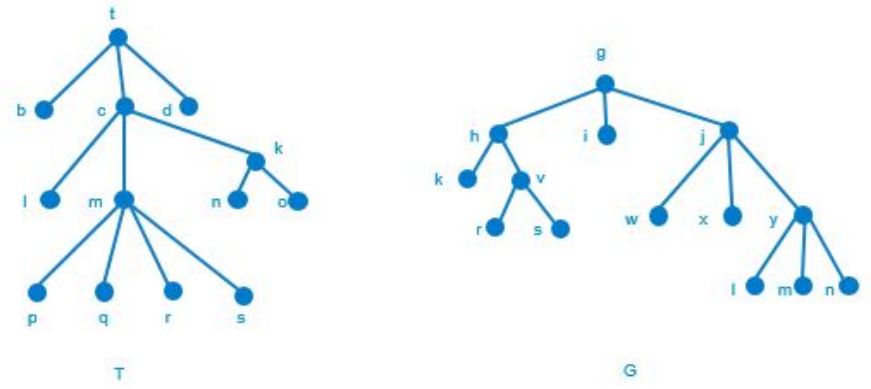


Figura 8: Ejercicio 1.

Ejercicio 2.

¿Cuáles de los siguientes árboles son 3-arios?, ¿cuáles son árboles completos 3-arios?

Figura 9: Ejercicio 2.

Ejercicio 3.

Etiqueta siguiendo el sistema de etiquetado universal el árbol de la figura siguiente:

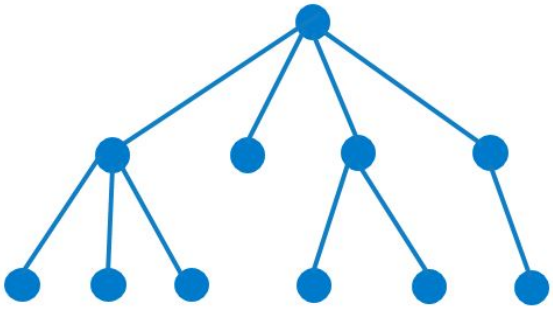


Figura 10: Ejercicio 3.

Ejercicio 4.

Escribe el recorrido preorder, postorder e inorder del árbol del ejercicio 3

Ejercicio 5.

Cambia la raíz del siguiente árbol para que esté balanceado.

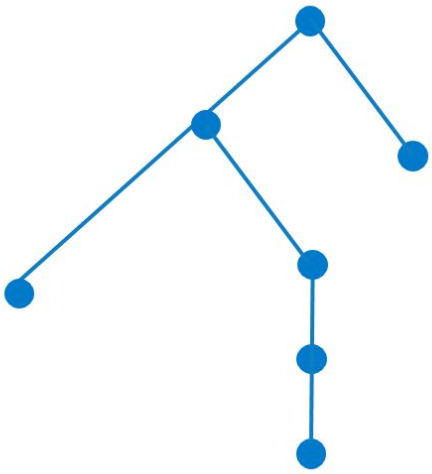


Figura 11: Ejercicio 5.

Ejercicio 6.

¿Cuál es la altura de los árboles del ejercicio 5 antes y después de balancearlos?

Ejercicio 7.

Sabemos que un árbol m-ario completo tiene trece hojas. ¿Qué valores puede tomar m? Dibuja los posibles arboles con estas características.

Ejercicio 8.

Recorre el siguiente árbol según el algoritmo preorder.

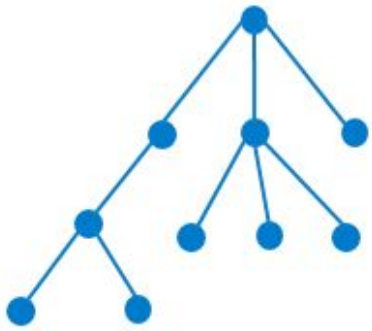


Figura 12: Ejercicio 8.

Ejercicio 9.

Recorre el árbol del ejercicio 8 en postorder

Ejercicio 10.

Construye el árbol asociado a la siguiente operación:

$$(a+b)c = a\frac{c}{a+b}$$

10.8 Soluciones cuaderno de ejercicios

Solución 1. Para el árbol T,

- Los vértices interiores son: c, m, k
- Las hojas son $b, d, l, n, o, p, q, r, s$
- La raíz es el vértice t

Para el árbol G,

- Los vértices interiores son: h, v, j, y
- Las hojas son $i, k, r, s, w, x, l, m, n$
- La raíz es el vértice g

Solución 2.

El segundo árbol es un árbol 3-ario completo puesto que todos los vértices que tienen hijos, tienen exactamente 3. El último árbol es un árbol 3-ario, ya que tiene exactamente 3 vértices.

Solución 3.

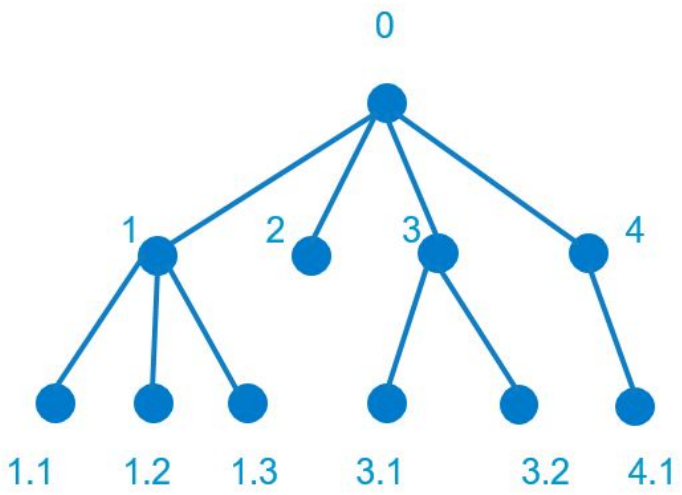


Figura 13: Solución ejercicio 3.

Solución 4. Siguiendo el sistema de etiquetado universal dado en la solución del ejercicio 3, los recorridos son:

- Preorder: 0, 1, 1.1, 1.2, 1.3, 2, 3, 3.1, 3.2, 4, 4.1
- Inorder: 1.1, 1, 1.2, 1.3, 0, 2, 3.1, 3, 3.2, 4.1, 4
- Postorder: 1.1, 1.2, 1.3, 1, 2, 3.1, 3.2, 3, 4.1, 4, 0

Solución 5.

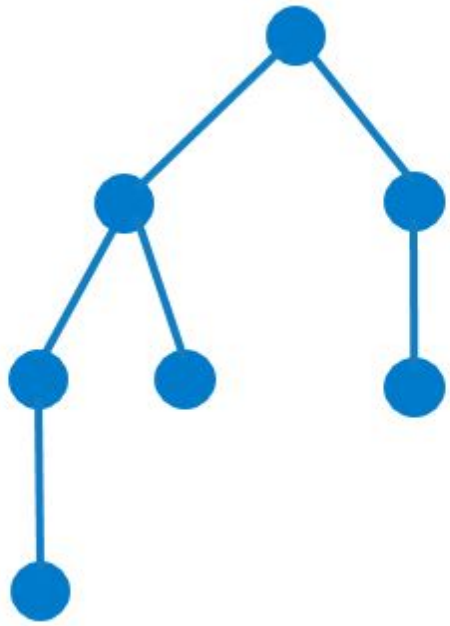


Figura 14: Solución ejercicio 5.

Solución 6. El árbol del ejercicio 5 es $h = 5$, pues es la longitud mas larga desde el vértice raíz hasta el vértice mas alejado.
En caso de la solución del ejercicio 5, la altura es $h = 4$

Solución 7. Sabemos que un árbol m-ario completo con l hojas tiene

$$i = (l-1)/(m-1)$$

vértices interiores. En nuestro caso, $i = 12/(m-1)$. Sabemos que i tiene que ser un valor entero, luego m solo puede tomar los valores 4 o 5. Los valores correspondientes de i son 4 y 3, respectivamente. Los dos árboles resultantes son:

Figura 15: Ejercicio 7.

Solución 8. Adoptando el etiquetado universal, el recorrido preorder es: 0, 1, 1.1, 1.1.1, 1.1.2, 2, 2.1, 2.2, 2.3, 3

Solución 9.

Adoptando el etiquetado universal, el recorrido postorder es: 1.1.1, 1.1.2, 1.1, 1, 2.1, 2.2, 2.3, 2, 3, 0

Solución 10.

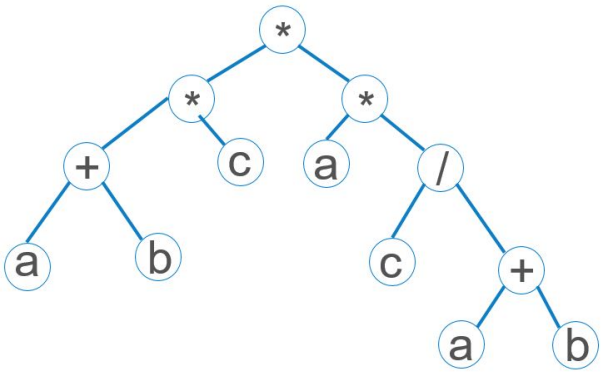


Figura 16: Solución ejercicio 10.

10.9 A fondo

A fondo

Árbol de decisión en valoración de inversiones

Un árbol de decisión en valoración de inversiones es una técnica de análisis de proyectos de inversión que se basa en la representación gráfica de decisiones de inversión secuenciales.

Martín López, S. (s.f.). Árbol de decisión en valoración de inversiones. Expansión.com http://www.expansion.com/diccionario-economico/arbol-de-decision-en-valoracion-de-inversiones.html

Minimax e Poda Alfa-Beta

El algoritmo minimax es un método de decisión para minimizar la pérdida y maximizar la ganancia esperada en juegos con adversario. Este vídeo de explica cómo el algoritmo crea el árbol de decisión.

Beckemkamp, E. (6 de diciembre de 2012). Minimax e Poda Alfa-Beta [vídeo]. YouTube. https://www.youtube.com/watch?v=ceU9sNFaSM8

10.10 Test

Test

1. Dado un árbol completo binario con i=512 vértices internos, ¿cuántos vértices tiene?

A. 512.

B. 1024.

C. 1025.

D. 2048.¿Qué es un grafo mixto?

2. Dado un árbol 4-ario completo con n=181 vértices, ¿cuántas hojas tiene?

A. 80.

B. 136.

C. 240.

D. 242.

3. Dado un árbol m-ario con l hojas, ¿cuántos vértices tiene?

A. n=li+1.

B. n=mi+1.

C. n=(ml−1)/(m−1).

D. n=(l−1)/(m−1)..

4. En un árbol balanceado se cumple que:

A. No hay hojas a distinto nivel.

B. Como mucho hay una diferencia de 1 en el nivel de las hojas.

C. Todas las hojas tienen dos hijos.

D. Todas las hojas tienen m hijos.

5. Dado un árbol m-ario con altura h, ¿cuántas hojas tiene como máximo?

A. mh.

B. mh.

C. hm.

D. m!

6. Los *prefix codes* pueden:

A. Tener ambigüedad en la codificación y decodificación.

B. Solo tener ambigüedad en la codificación.

C. Solo tener ambigüedad en la decodificación.

D. Codificarse y decodificarse sin ambigüedad

7. Para comprimir datos se suelen usar:

A. Árboles binarios de búsqueda.

B. Árboles binarios balanceados.

C. Árboles de decisión.

D. *Prefix codes.*

8. El subárbol derecho se visita por último en:

A. Recorrido postorder.

B. Recorrido inorder.

C. Recorrido preorder.

D. Recorrido preorder e inorder.

9. La notación polaca inversa corresponde con un recorrido:

A. Preorder.

B. Inorder.

C. Postorder.

D. Preorder o postorder.

10. ¿En qué notaciones no es necesario incluir paréntesis en una expresión para evitar ambigüedades?

A. Infija y postfija.

B. Prefija.

C. Prefija y postfija.

D. Siempre es necesario incluir paréntesis para evitar ambigüedades.