Journal **TechInnovation**
Volumen 1, Número 2, 2022
Universidad Estatal del Sur de Manabí
ISSN-e: 2953-6472

ARTÍCULO DE REVISIÓN

Importancia de árboles binarios en la programación

Importance of binary trees in programming

https://doi.org/10.47230/Journal.TechInnovation.v1.n2.2022.33-42

Recibido: 01-06-2022 **Aceptado:** 27-06-2022 **Publicado:** 31-07-2022

Karelis Cecibel Cedeño Zambrano[1]
https://orcid.org/0000-0001-9499-2327

Danna Deyaneyra Lucas Alay[2]
https://orcid.org/0000-0002-7332-7755

Luis Alexander Verá Polanco[3]
https://orcid.org/0000-0001-9653-3618

Jocelyn Mirley Vera Zambrano[4]
https://orcid.org/0000-0002-2617-3765

Vicente Fray Romero Castro[5]
https://orcid.org/0000-0001-5792-0105

1. Estudiante de la Carrera Tecnologías de la Información. Bachillerato. Estudiante de la Universidad Estatal del Sur de Manabí. Jipijapa, Manabí y Ecuador. cedeno-karelis7041@unesum.edu.ec
2. Estudiante de la Carrera Tecnologías de la Información. Bachillerato. Estudiante de la Universidad Estatal del Sur de Manabí. Jipijapa, Manabí y Ecuador. lucas-danna2967@unesum.edu.ec
3. Estudiante de la Carrera Tecnologías de la Información. Bachillerato. Estudiante de la Universidad Estatal del Sur de Manabí. Jipijapa, Manabí y Ecuador. vera-luis1491@unesum.edu.ec
4. Estudiante de la Carrera Tecnologías de la Información. Bachillerato. Estudiante de la Universidad Estatal del Sur de Manabí. Jipijapa, Manabí y Ecuador. vera-jocelyn6630@unesum.edu.ec
5. Ingeniero en Sistemas. Magister en Sistemas de Información Gerencial. Docente de la carrera Tecnologías de la Información en la Universidad Estatal del Sur de Manabí. Jipijapa. Manabí. Ecuador. vicente.romero@unesum.edu.ec

Volumen: 1
Número: 2
Año: 2022
Paginación: 33-42
URL: https://revistas.unesum.edu.ec/JTI/index.php/JTI/article/view/18
***Correspondencia autor:** cedeno-karelis7041@unesum.edu.ec

RESUMEN

El presente trabajo de investigación hace referencia a la importancia de los arboles binarios dentro de la programación, que es una estructura de datos es decir un grafo simple que permitirá generar un orden dentro de un árbol que contenga elementos de manera sencilla y ágil en el manejo de los tipos de árboles, así mismo aplicar las funciones en las operaciones básicas como son: insertar, eliminar, consultar, recorrer los elementos. El objetivo primordial es proporcionar información acerca de cómo radica la importancia de los árboles, sus características, propiedades, como construir mediante métodos, como almacenar estos datos en un árbol, como recorrer un árbol y cuál es el indicado para aplicarlo. Además, se analizará las dificultades que puede ocasionar la falta de conocimientos relacionado a este tema. En cuanto al desarrollo se manifestó ejemplos característicos, funciones de cómo implementarlas, estructura y representación de los mismos, para distinguir estas estructuras de datos de otras que se pueden presentar en la ejecución, para ello se utilizaron métodos de investigación científica predominando el método análisis-síntesis, junto al método histórico-lógico. Por consiguiente, los nodos que se aplican en las conexiones de cada punto por medio de aristas y vértices, en la vida cotidiana se emplean en las telecomunicaciones, en vista de que la representación hace énfasis a nodos interconectados como en una red. Definiendo así, que los arboles presentan ramificaciones que llevando un orden ya sea de forma ascendente o descendente permitirá almacenar cualquier tipo de dato que se desee.

Palabras clave: Binario, Nodos, Ramificaciones.

ABSTRACT

This research work refers to the importance of binary trees within programming, which is a data structure, that is, a simple graph that will allow generating an order within a tree that contains elements in a simple and agile way in handling. of the types of trees, likewise apply the functions in the basic operations such as: insert, delete, consult, go through the elements. The primary objective is to provide information about how the importance of trees lies, their characteristics, properties, how to build using methods, how to store this data in a tree, how to traverse a tree and which one is indicated to apply it. In addition, the difficulties that the lack of knowledge related to this topic can cause will be analyzed. Regarding the development, characteristic examples were manifested, functions of how to implement them, their structure and representation, to distinguish these data structures from others that can be presented in the execution, for this purpose scientific research methods were used, predominantly the analysis-method. synthesis, together with the historical-logical method. Therefore, the nodes that are applied in the connections of each point by means of edges and vertices, in daily life are used in telecommunications, since the representation emphasizes interconnected nodes as in a network. Defining in this way, that the trees present ramifications that carrying an order either ascending or descending will allow to store any type of data that is desired.

Keywords: Binary, Nodes, Ramifications.

Introducción

Los arboles binarios son grafos simples, es decir un conjunto de elementos (nodos) finitos están formados por una raíz, y pueden poseer varios subárboles que van hacia la derecha e izquierda; entre las aplicaciones en las que se emplean a los árboles se destacan la representación de estructuras en las cuales es posible tomar opciones desde distintos puntos, tomar varios nodos e inclusive, recorrer desde un nodo especifico, eliminar, consultar, e insertar más nodos al árbol según se lo requiera.

Entre los tipos de árboles se describen distintos, equivalentes, de búsquedad, completos, estrictos, de expresión, entre otros en las cuales se les atribuyen diferentes operaciones con sus determinadas funciones a emplear para su uso, además se conocen las generalidades que poseen los mismos como lo son sus niveles, altura, peso, orden grado que poseen los árboles para la ejecución de los mismo se emplean diferentes funciones de acorde a lo que se vaya a requerir.

En la actualidad los arboles binarios se emplean en diferentes ámbitos como en los sistemas de ordenamiento y almacenamiento de datos, ordenamiento jerárquico dentro de una entidad e incluso en las redes de telecomunicaciones, es decir se los visualiza más de lo normal en los procesos cotidianos, entre otros.

En esta investigación se pretende que los lectores adquieran conocimientos básicos de la implementación correcta en los lenguajes de programación más comunes los arboles binarios, haciendo un estudio previo de la estructura, sus representaciones, ventajas e inconvenientes, como se implementa los recorridos hasta el orden de como recorrerlos en cualquier lenguaje.

Teniendo, así como objetivo principal, brindar a los lectores una investigación relacionada con la importancia de los árboles, tipos, estructura dentro de las plataformas en las que se pueden programar, permiten estructurar los datos de manera correcta y ascendente o descendente, pre o post orden haciendo uso de estos, se realiza una búsquedad apropiada, almacenando los nodos en los subárboles.

El impacto que tendrá esta investigación en los usuarios es aceptable por lo que es una estructura de datos muy eficaz, eficiente y además muy utilizada en la actualidad en los lenguajes de programación, que permitirá ingresar, consultar, eliminar, recorrer, obtener datos de forma ordenada o como la deseen y así mismo hacer uso de las características, ventajas y desventajas de su usos e importancia que tienen los mismos.

Desarrollo

¿Qué es un Árbol binario?

Es una estructura de datos no lineales, muy similares a las listas doblemente enlazadas debido a que poseen dos punteros (nodos) en doble sentido; se caracteriza por tener un vértice principal del que se desprenden otras ramificaciones de tal manera se simula a un árbol. (RSS feed, s.f.)

En pocas palabras un arbol es un grafo simple que recorre un camino, utilizando un vector, poseen ramificaciones, constituyendo un orden dentro de los elementos que posee un arbol binario, son muy consecuentes en el uso de aplicaciones en la vida cotidiana.

Se realiza un mapeo uno a uno entre los árboles binarios, el cual en particular es usado en List para representar árboles; cada nodo N ordenado corresponde a un nodo N' el hijo de la izquierda de N' es el nodo correspondiente al primer hijo de N, y el hijo derecho de N' es el nodo correspondiente al siguiente hermano de N, es decir, el próximo nodo en orden entre los hijos de los padres de N. (Wikipedia®, 2019)

Estructura y representación de un árbol binario

La estructura de un árbol se la define de la siguiente manera una estructura recursiva teniendo en cuenta que cada nodo del árbol, junto con todos sus descendientes, y manteniendo la ordenación original constituye también un árbol o subárbol del árbol principal. (P, 2017)

El movimiento a través de árboles, salvo que implementemos punteros al nodo padre, será siempre partiendo del nodo raíz hacia un nodo hoja. Cada vez que se llegue a un nuevo nodo por cualquiera de los nodos se apunta para avanzar al siguiente nodo. (Estructuras de datosEstru, 2018)

```
struct Arbol {
        int dato;
        struct Arbol *rama1;
        struct Arbol *rama2;
        };
```

(Anónimo, estructurasite.wordpress.com, s.f.)

En la gráfica que se presenta se detalla algunas generalidades más comunes que posee un árbol binario haciendo énfasis a la estructura de datos.

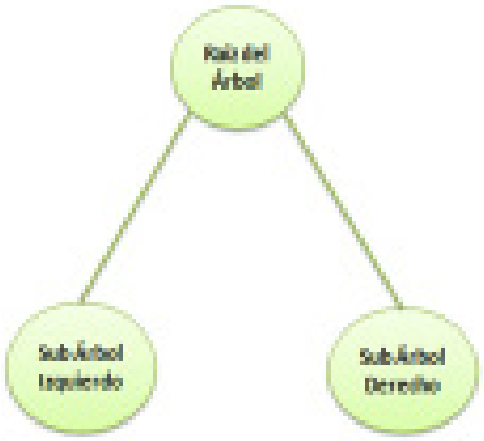

Ilustración 1: Estructura; Autor RSS feed

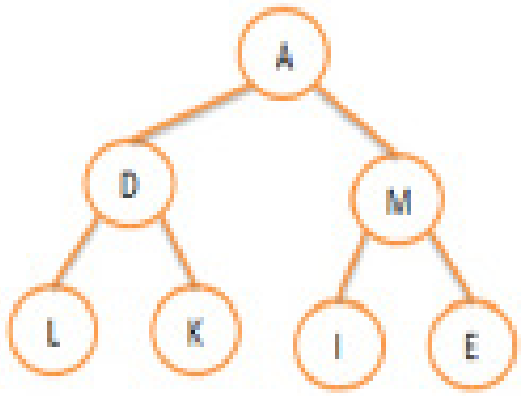

Ilustración 2: Representación de los arboles; Autor RSS feed

Generalidades de la estructura de los Árboles

Entre las generalidades de los árboles se estructura se presentan.

- **Nodos:** Los nodos son aquellos elementos que contiene un árbol; entre los tipos de nodos son:
    1. Nodo Raíz: Es el primer nodo de un Árbol.
    2. Nodo Padre: Son los nodos que tienen al menos un hijo.
    3. Nodo Hijo: Son todos los nodos que tiene un padre.
    4. Nodo Hermano: son aquellos nodos que comparte a un mismo padre.
    5. Nodo Hoja: Son los nodos que no tienen hijos.
    6. Nodo Rama: Tienen por lo general un hijo, y no son la raíz.

- **Nivel:** Se los denomina a cada generación que ocurre dentro del árbol, es decir cuando crece.
- **Altura:** Es el número máximo de niveles de un Árbol.
- **Peso:** Es el número de nodos que posee.
- **Orden:** Es el número máximo de hijos que puede tener un Nodo.
- **Grado:** Se refiere al número mayor de hijos que tiene alguno de los nodos. (oscarblancarte, 2014)

Tipos de Árboles

La clasificación de los arboles binarios pueden ser cuatros:

- Distinto: Son distintos cuando sus estructuras son diferentes.
- Similares: Cuando sus estructuras son idénticas, pero la información de sus nodos es diferente.
- Equivalentes: Son similares y los nodos contienen la misma información.
- Completos: Son aquellos árboles en los que todos sus nodos excepto los del ultimo nivel, tiene dos hijos; el subárbol izquierdo y el subárbol derecho. (EcuRed, 2018)
- Búsquedad: Son un tipo especial de árbol binario permite insertar elementos, facilitando así la búsqueda de un nodo. (RSS feed, s.f.)
- Estricto: Cada nodo puede contener 0 o 2 hijos.
- Lleno: En cada nodo la altura del subárbol izquierdo es igual a la del derecho, y ambos subárboles son árboles llenos. (Vaca Rodríguez, 2011)
- Árboles de expresión: Se utilizan para almacenar en la memoria de una computadora expresiones lógicas, aritméticas o algebraicas. (Matematicas, 2017)

Características

Entre las características que permiten la colección de datos se presentan diferentes propiedades o características:

- Poseen estructuras de datos no lineales debido a que cada elemento puede seguir varios elementos.
- Es una estructura jerárquica, la cual posee una raíz principal, en una colección de nodos (elementos).
- Se aplica en la representación de árboles genealógicos (es decir para la solución de bucles y ordenar elementos).
- Son dinámicos, debido a que su tamaño, forma y contenido pueden variar durante su ejecución. (Echeverry, 2015)
- El unico nodo que se distigue es el nodo raiz.
- El metodo mas usado para realizar las operaciones es por medio de un cursor, es decir se marca un nodo que servira como referencia.
- Se indica un nodo concreto por donde se desea comenzar. (Vaca Rodríguez, 2011)

Ventajas

Entre las ventajas que poseen el utilizar arboles se describen las siguientes:

- El número de accesos al árbol es menor que en una lista enlazada
- Las búsquedas son más eficientes cuando el árbol está equilibrado
- Permite recorrer todos los datos, en el orden que se desee o se requiera
- Permite el ordenamiento de manera continua. (Mosqueira, Guijarro, & Cabrero , 2017)

Desventajas

Sus desventajas o inconvenientes son pocas, pero de gran importancia.

- Los nodos que se van insertando deben tener un orden aleatorio y ser equilibrado.
- Si los datos no son de un mismo tipo no ingresaran
- Se debe seguir la estructura.
- El orden dentro de la programación suele tener un cierto grado de dificultad. (Mosqueira, Guijarro, & Cabrero)

Operaciones de los Árboles

Entre las operaciones básicas que se pueden describir las siguientes:

- Buscar un elemento.
- listar los hijos.
- Enumerar todos los elementos.
- Eliminar un subárbol.
- Moverse o recorrer el árbol de manera completa.
- Añadir o insertar un elemento al árbol
- Encontrar la raíz de cualquier nodo (Garcia)

Funciones de los Árboles

Para implementarlas en programación esta estructura de datos se debe analizar correctamente las funciones entre las que se destacan:

1. La función info(p) regresa el contenido de nd.
2. La función left(p) regresa un apuntador al hijo izquierdo de nd.
3. La función right(p) regresa un apuntador al hijo derecho de nd.
4. La función father(p) regresa un apuntador al padre de nd.
5. La función brother(p) regresa un apuntador al hermano de nd.
6. La función isLeft(p) regresa true si nd es un hijo izquierdo de algún otro nodo en el árbol, y false en caso contrario.
7. La función isRight(p) regresa true si nd es un hijo derecho de algún otro nodo en el árbol, y false en caso contrario. (Ruíz)
8. La función RecorrerArbol, aplicando recursividad, será tan sencilla como invocar de nuevo a la función RecorrerArbol para cada una de las ramas; es decir:

```
void RecorrerArbol(Arbol a)

{

if(a == NULL) return;

RecorrerArbol(a->rama[0]);

RecorrerArbol(a->rama[1]);

RecorrerArbol(a->rama[2]);

}
```

(Guerrero, 2019)

En la construcción de un árbol binario son útiles las operaciones:

- makeTree(x) crea un nuevo árbol que consta de un nodo único con un campo de información x, y regresa un apuntador a este nodo.
- setLeft(p,x) crea un nuevo hijo izquierdo de node(p) con el campo de información x.
- setRight(p, x) crea un nuevo hijo derecho de node(p) con el campo de información x. (Quevedo, 2014)

Formas de recorrer un árbol binario

Se presentan diversas formas de recorrer un árbol con sus elementos; siendo los consecuentes:

- Recorrido en Preorden: Este recorrido consiste analizar el nodo raíz, para poder recorrer el subárbol izquierdo para posterior el subárbol derecho y realizar el preorden.
- Recorrido en Inoorden: Consiste en examinar el nodo raíz, recorriendo el subárbol izquierdo y luego el derecho, conservando el recorrido.
- Recorrido en Postorden: Para cada subárbol se debe conservar el recorrido en Postorden, es decir, primero se visita la parte izquierda, luego la parte derecha y por último la raíz. (RSS feed, s.f.)
- Recorrido por niveles: Se recorren ordenados de menor a mayor nivel, a igualdad de nivel se recorren de izquierda a derecha.
- Recorrido no recursivo: Se introduce en la raíz y se agrega un bucle en el que se extrae un nodo, se recorre su elemento y se insertan sus hijos. (Anónimo, estructurasite.wordpress.com, s.f.)

Propiedades de los Árboles

Entre las propiedades se aplican operaciones para emplearlos en cualquier plataforma de programación, las cuales son:

- El recorrido inorden, recorre los elementos en orden de menor a mayor.
- El elemento mínimo es el primer nodo sin hijo izquierdo en un descenso por hijos izquierdos desde la raiz.
- El elemento máximo, nodo sin hijo derecho en un descenso por hijos derechos desde la raíz.
- Para buscar un elemento se parte de la raíz y se escoge el subárbol izquierdo si el valor buscado es menor que el del nodo o el subárbol derecho si es mayor.
- Para insertar, se busca en el árbol y se inserta como nodo hoja en el punto donde debería encontrarse.
- Para borrar, si tiene dos hijos se intercambia con el máximo de su subárbol izquierdo y se borra el máximo. (Vaca Rodríguez, 2011)

Importancia de los Árboles

Los árboles son modelos de gran utilidad pues su representación normalmente es una estructura jerárquica y este es una de sus principales aplicaciones; además se usan mucho en las modelaciones de búsquedas o problemas que dependan de la representación de una búsqueda en un espacio representable como un grafo. (Anónimo, s.f.)

Los árboles no son más que grafos, pues están compuestos de nodos y aristas que los unen; recorriendo un camino en el que el primer y último nodo son el mismo, y un Circuito simple es un circuito en que cada nodo aparece sólo una vez. Se dice que el grafo es Conectado si siempre existe un camino entre dos nodos del grafo. (Di Mare, 2006)

Almacenar los datos en un árbol

Los arboles pueden ser creados en diferentes lenguajes de programación ya sean con registros o referencias; por lo general son construidos a partir de la estructura de nodos y punteros, en donde se almacenarán datos, Si un nodo tiene menos de dos hijos, algunos de los punteros de los hijos pueden ser definidos como nulos para indicar que no dispone de dicho nodo, pueden ser almacenados como una estructura de datos implícita en vectores, y si el árbol es un árbol binario completo, este método no desaprovecha el espacio en memoria. (Wikipedia®, 2019)

Ejemplo de la inserción de un árbol binario

Basándose en la estructura de un árbol binario se plantea el siguiente ejemplo de cómo insertar un árbol en un lenguaje de programación.

```
procedure Insertar
  (Arbol : in out T_Arbol;
   Valor : in      Integer)
is
   Actual   : T_Nodo_Ptr := Arbol.Raiz;
   Anterior : T_Nodo_Ptr := null;
   Nuevo    : T_Nodo_Ptr := null;
begin

   --  Buscamos la posición donde hay que insertar el nuevo valor. Para ello
hacemos un
   --  bucle mientras no lleguemos al final de una rama y mientras el valor del
nodo en el
   --  que estamos no coincida con el valor que queremos insertar:
   while Actual /= null and then Actual.Info /= Valor loop
      Anterior := Actual;
      if Valor < Actual.Info then
         Actual := Actual.Izquierda;
      else
         Actual := Actual.Derecha;
      end if;
   end loop;
```

Materiales y métodos

Los materiales que llevaron a cabo en este trabajo de investigación fueron: diversas fuentes bibliográficas lo que conlleva a una investigación completa.

En cuanto al desarrollo de esta indagación se utilizaron los métodos de la investigación científica tales como:

Histórico- Lógico: Lo histórico se refiere al estudio del objeto en su trayectoria real a través de su historia, con sus condicionamientos sociales, económicos y políticos en los diferentes periodos. Lo lógico interpreta lo histórico e infiere conclusiones; Es decir este método fue empleado en la construcción de todo el proceso del trabajo investigativo, como para conocer que son los árboles, en que radica su importancia dentro de los lenguajes de programación.

- **Análisis-Síntesis:** El análisis es un procedimiento lógico que posibilita descomponer mentalmente un todo en sus partes y cualidades, en sus múltiples relaciones, propiedades y componentes; mientras que síntesis es la operación inversa, que establece mentalmente la unión o combinación de las partes previamente analizadas y posibilita descubrir relaciones y características generales entre los elementos de la realidad; utilizando así este método para profundizar y sintetizar la importancia de los arboles binarios dentro de los lenguajes de programación; asi mismo para la argumentación de los conceptos básicos de lo anteriormente mencionado. Posterior a ellos se emplearon los métodos teóricos lograron sintetizar que es necesario realizar una investigación profunda para conocer cuáles son las anomalías que puede presentarse al plantear un tema científico. Así mismo, se emplearon los

métodos bibliográficos y referenciales para revisar y constatar la información real sobre la investigación presente.

Resultados

Dentro de la información recabada en diferentes sitios de la web, se conoció no solamente que la aplicación de los árboles se realiza en programación, sino que a la vez en diferentes ámbitos siendo muy relevantes, como lo es el ordenamiento de datos información dentro de una empresa, además conocer las operaciones y funciones para insertar, eliminar, buscar, recorrer un elemento o dato dentro de un árbol binario.

El impacto social que se obtuvo en diferentes lectores, fue conmovedor debido a que mostraron iniciativa por conocer sobre la temática planteada, como su importancia, funciones, estructura para en los lenguajes de programación, siendo una estructura de datos muy empleada.

Conclusiones

Al conocer la importancia de los árboles y sus tipos; además como implementar los mismos en las plataformas de programación estas estructuras de datos son muy utilizadas en la actualidad, para ordenar datos en diferentes estructuras, que contengan información.

La investigación planteada resultó eficiente debido a que se pretende que los lectores adquirieran un mayor conocimiento de cual importante son los arboles binarios, implementar y utilizar sus operaciones, funciones, conociendo su estructura, sus ventajas y desventajas y empleándolos de manera correcta, para ordenar los elementos que puede contener un árbol.

Referencias

Anónimo. (s.f.). ecured.cu. Obtenido de Árbol (Grafo): https://www.ecured.cu/%C3%81rbol_(Grafo)

Anónimo. (s.f.). estructurasite.wordpress.com. Obtenido de Árbol: https://estructurasite.wordpress.com/arbol/

Di Mare, A. (15 de Febrero de 2006). Una clase C++ completa para conocer y usar árboles. Obtenido de Una clase C++ completa para conocer y usar árboles: http://www.di-mare.com/adolfo/p/TreeTdef.htm

Echeverry, A. (2015). monografias.com. Obtenido de Estructuración de datos y aplicación en Ingeniería de Sistemas: https://www.monografias.com/trabajos102/estructuracion-datos/estructuracion-datos.shtml

EcuRed. (11 de Sepriembre de 2018). ecured.cu. Obtenido de Árbol binario: https://www.ecured.cu/%C3%81rbol_binario

Estructuras de datosEstru. (2018). conclase.net. Obtenido de Árboles: http://www.conclase.net/c/edd/?cap=006#inicio

Garcia, G. (s.f.). garciagregorio.webcindario.com. Obtenido de TIPOS DE ARBOLES: https://garciagregorio.webcindario.com/ed/arboles_clasificacion.pdf

Guerrero, D. (09 de Septiembre de 2019). monografias.com. Obtenido de Arboles Binario: https://www.monografias.com/trabajos92/arboles-binario/arboles-binario.shtml

Matematicas. (23 de noviembre de 2017). medium.com. Obtenido de TEORIA DE ARBOLES BINARIOS: https://medium.com/@matematicasdiscretaslibro/cap%C3%ADtulo-12-teoria-de-arboles-binarios-f731baf470c0

Mosqueira, E., Guijarro, B., & Cabrero , M. (noviembre de 2017). quegrande.org. Obtenido de Árboles Binarios de Búsqueda: http://quegrande.org/apuntes/EI/1/EDI/teoria/08-09/arboles_binarios_de_busqueda.pdf

oscarblancarte. (22 de Agosto de 2014). oscarblancarteblog.com. Obtenido de Estructura de datos – Árboles: https://www.oscarblancarteblog.com/2014/08/22/estructura-de-datos-arboles/

P, S. (2017). uniovi.es. Obtenido de ÁRBOLES. ÁRBOLES BINARIOS.: http://www6.uniovi.es/usr/cesar/Uned/EDA/Apuntes/TAD_apUM_04.pdf

Quevedo, E. (16 de Mayo de 2014). iuma.ulpgc.es. Obtenido de Árbol Binario: http://www.iuma.ulpgc.es/users/jmiranda/docencia/programacion/Tema7_ne.pdf

RSS feed. (s.f.). wordpress.com. Obtenido de Árboles Binarios: https://hhmosquera.wordpress.com/arbolesbinarios/

Ruíz. (s.f.). utm.mx. Obtenido de ÁRBOLES.: http://www.utm.mx/~rruiz/cursos/ED/material/ABB.pdf

Vaca Rodríguez, C. (12 de diciembre de 2011). infor.uva.es. Obtenido de Estructuras de Datos y Algoritmos: https://www.infor.uva.es/~cvaca/asigs/doceda/tema4.pdf

Wikipedia®. (14 de noviembre de 2019). es.wikipedia.org. Obtenido de Árbol binario: https://es.wikipedia.org/wiki/%C3%81rbol_binario

Cómo citar: Cedeño Zambrano, K. C., Lucas Alay, D. D., Verá Polanco, L. A., Vera Zambrano, J. M., & Romero Castro, V. F. (2022). Importancia de árboles binarios en la programación. Journal TechInnovation, 1(2), 33–42. https://doi.org/10.47230/Journal.TechInnovation.v1.n2.2022.33-42