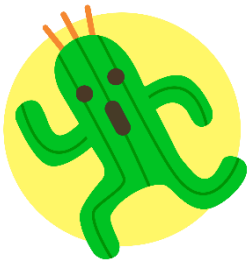
Grafos sin espinas

(4) Árboles



V 0.4 2023_11_12

Aprende sin espinas
con @carlos_cactus

Sócrates se equivocaba. El conocimiento no es lo único que crece al compartirse: La alegría también.

A la inspiración del bucle_infinito,
al Cibergrupo y al tHash_A, por su amistad,
y sobre todo, a quienes dicen "pero quiero"
cuando sienten "no puedo".

¡Un saludo sin espinas!
@carlos_cactus :D

Y si quieres saber más:
¡Encuéntrame en Telegram como @carlos_cactus o habla con Espinito, el bot Sin Espinas, en @GestionSinEspinasBot.

Únete a la comunidad de Telegram Sin Espinas y no te pierdas nada!

Deja de preocuparte por aprobar y ¡Aprende sin Espinas!

→ Esto está en perpetua mejora. Falta:

- **Algoritmo de Prim**
- **Complejidad O(n) de Prim**

ÍNDICE

1 CONCEPTOS BÁSICOS 4

1.1 Caracterización de árboles. Definiciones 4
1.1.1 Concepto de árbol 4
1.1.2 BOSQUE 5
1.1.3 GRAFO CONEXO EN EL CONTEXTO DE ÁRBOLES 5
1.1.4 MEDIDA DE UN BOSQUE 5
1.1.5 CONCEPTO DE HOJA 6
1.1.6 VÉRTICE INTERNO Y RAÍZ 6
1.1.7 CÁLCULO DEL NÚMERO DE HOJAS DE UN ÁRBOL 7

2 ÁRBOLES GENERADORES 9

2.1 Propiedades de árboles generadores 9

2.2 Construcción de árbol generador 9

2.3 Árboles generadores minimales 11

2.4 Algoritmo de Kruskal 12
2.4.1 Definición del algoritmo de Kruskal 12
2.4.2 Ejemplo de ejecución de algoritmo de Kruskal 13
2.4.3 Complejidad del algoritmo de Kruskal 14

2.5 Algoritmo de Prim 14
2.5.1 Definición del algoritmo de Prim 14
2.5.2 Ejemplo de ejecución de algoritmo de Prim 15
2.5.3 Complejidad del algoritmo de Prim 16

3 ÁRBOLES CON RAÍZ 17

3.1 Características de los árboles con raíz 17
3.1.1 Conceto de predecesor y sucesor 17
3.1.2 Conceptos de vértice padre, vértice hijo y vértices hermanos 17
3.1.3 Digrafo ASIMÉTRICO 17
3.1.4 Raíz de un digrafo 18
3.1.5 Definición de árbol con raíz 18
3.1.6 Notación de árbol para operaciones aritméticas 19

3.2 Árboles m-arios 20
3.2.1 Nivel de un vértice 20
3.2.2 Altura o profundidad h(T) de un árbol T 20
3.2.3 Árbol equilibrado 21
3.2.4 Árbol m-ario 22
3.2.5 Árbol binario 23
3.2.6 Árbol m-ario lleno (full) 23
3.2.7 ÁRBOL m-ARIO COMPLETO 24
3.2.8 ÁRBOL M-ARIO PERFECTO 24
3.2.9 Relación entre vértices internos y hojas en árboles m-arios COMPLETOS 25
3.2.10 Relación entre la altura h y el máximo número de hojas 25
3.2.11 Relación entre la altura mínima de un árbol m-ario y su número de hojas t 25
3.2.12 Relación entre la altura mínima de un árbol m-ario COMPLETO y EQUILIBRADO y su número de hojas t 25

3.3 Exploración de árboles binarios con raíz: PREORDEN, INORDEN y POSTORDEN 26

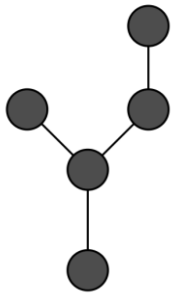
3.4 Complejidad de la búsqueda 29

3.5 EJERCICIOS 30

1 CONCEPTOS BÁSICOS

1.1 CARACTERIZACIÓN DE ÁRBOLES. DEFINICIONES

1.1.1 Concepto de árbol



Un árbol es un grafo conexo sin ciclos.

Se define el grafo T (*tree*) de orden n y medida m.

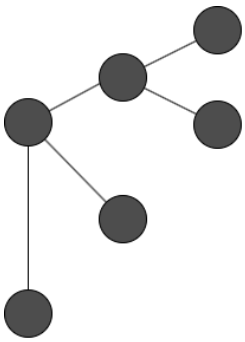
T es un árbol si satisface 2 propiedades:

1) Existe EXACTAMENTE UN SOLO CAMINO entre cualquier par de vértices.

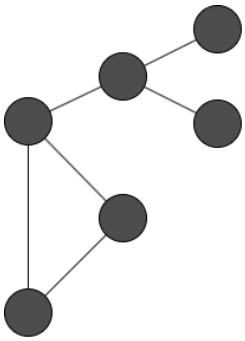
   Lo cual, es consecuencia de 2 hechos:
   - Es ACÍCLICO: existe COMO MÁXIMO UN camino entre cualquier par de vértices.
   - Es CONEXO: existe COMO MÍNIMO UN camino entre cualquier par de vértices.

2) Cumple que m = n – 1

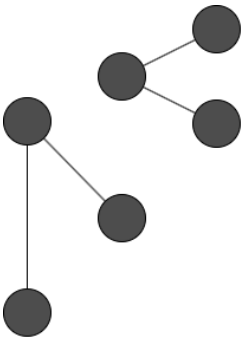
EJEMPLOS DE ÁRBOLES Y NO ÁRBOLES



El grafo A es un árbol, ya que es conexo y acíclico.



El grafo B NO ES UN ÁRBOL ya que no es acíclico.



El grafo C NO ES UN ÁRBOL ya que no es conexo. Se trata de un bosque.

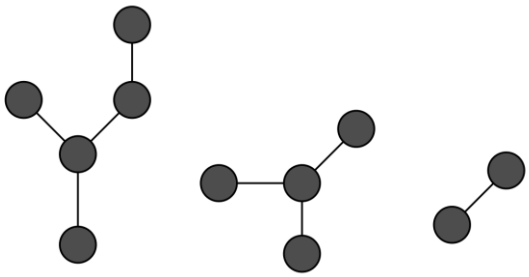
1.1.2 BOSQUE

Un bosque es la unión disjunta de árboles.
Se puede ver también d otras maneras:

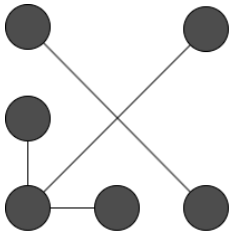
- Un conjunto no conexo de árboles.
- Grafo acíclico no conexo.

EJEMPLOS DE BOSQUE

El grafo A es un bosque con 3 componentes conexas, cada una de las cuales es, a su vez, un árbol.



El grafo B contiene 2 componentes conexas que, pese a mostrarse superpuestas, no están conectadas entre ellas.



1.1.3 GRAFO CONEXO EN EL CONTEXTO DE ÁRBOLES

Grafo compatible con árbol generador
En economía, es relevante la obtención de árboles generadores minimales de grafos ponderados.

1.1.4 MEDIDA DE UN BOSQUE

Un bosque de orden n formado por k árboles tiene medida n – k.
Se demuestra fácilmente:

$$|A| = \sum_{i=1}^{k} m_{Ti} = \sum_{i=1}^{k} (n_{Ti} - 1) = \left(\sum_{i=1}^{k} n_{Ti} \right) - k = n - k$$

Donde:

|A|	Medida del bosque
m_{Ti}	Vértices totales del árbol i
n_{Ti}	Aristas totales del árbol i

EJEMPLO DE MEDIDA DE UN BOSQUE:

Un bosque tiene 30 vértices y 24 aristas. Se desea conocer cuántas componentes conexas lo integran.
Se aprovecha la relación para la medida de un bosque:

$$|A| = \sum_{i=1}^{k} m_{Ti} = \sum_{i=1}^{k} (n_{Ti} - 1) = \left(\sum_{i=1}^{k} n_{Ti} \right) - k = n - k$$

En este caso:

$$24 = 30 - k$$

De lo cual, el bosque resulta necesariamente de la unión de 6 componentes conexas.

1.1.5 CONCEPTO DE HOJA

Una hoja de un árbol es un vértice v de un árbol cuyo grado de salida es $g^+(v) = 0$

Todo árbol de orden mínimo 2 contiene 2 hojas como mínimo.

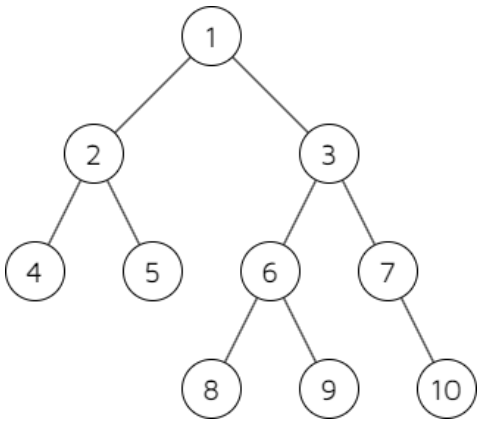
Una hoja también se denomina vértice TERMINAL, en contraposición a los vértices INTERNOS (no terminales).

1.1.6 VÉRTICE INTERNO Y RAÍZ

Un vértice es interno si tiene algún hijo.

Es decir, los vértices internos son aquellos que no son hojas.

Si un vértice interno no tiene padre, además de ser padre, se trata de la raíz del árbol.



EJEMPLO DE HOJA

Los vértices 4, 5, 8, 9 y 10 son hojas, pues no tienen hijos, o sea, son TERMINALES.

EJEMPLO DE VÉRTICE INTERNO

Los vértices 1, 2, 3, 6 y 7 son vértices internos, o sea, NO SON TERMINALES, puesto que tienen hijos (ya sean hojas o no).

EJEMPLO DE RAÍZ
Además de vértice interno, el vértice 1 es RAÍZ del árbol.
TODA RAÍZ ES VÉRTICE INTERNO, ya que no es terminal.

1.1.7 CÁLCULO DEL NÚMERO DE HOJAS DE UN ÁRBOL

EJEMPLO

Se desea calcular el número de hojas de un árbol T = (V,A) que tiene:

- 5 vértices de grado 3.
- 4 vértices de grado 2.
- El resto de los vértices son de grado 1 (hojas).

En primer lugar, se observa que el árbol T tiene orden n:

$$n = 5 + 4 + x = 9 + x \text{ vértices}$$

En segundo lugar, se cumple, para todo grafo simple no dirigido la fórmula de los grados de Euler, que relaciona la suma de los grados de los vértices g(v) con la medida |A|, que es el número de aristas:

$$\sum_{v \in V}^{k} g(v) = 2 \cdot |A|$$

En este caso:

$$5 \cdot 3 + 4 \cdot 2 + x \cdot 1 = 2 \cdot |A| = 2 \cdot m$$

Además, todo árbol satisface m = n – 1:

$$23 + x = 2 \cdot m = 2 \cdot (n - 1)$$

Entonces:

$$23 + x = 2n - 2$$

En este caso, para el orden n = 9 + x observado arriba:

$$23 + x = 2(9 + x) - 2$$

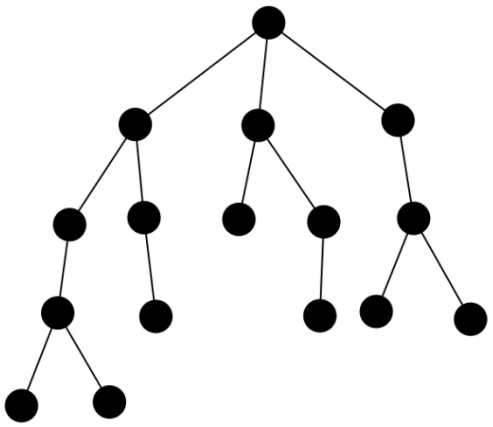
Es decir:

$$23 + x = 16 + 2x$$

Se alcanza x = 7. Es decir:

- Hay 7 hojas.
- El orden es n = 9 + 7 = 16 vértices.
- Hay m = n – 1 = 15 aristas

Y el árbol representado puede ser:



EJEMPLO

Se desea conocer la secuencia de grados de un árbol T sabiendo que es de orden 9 y contiene 3 vértices de grado 3.

- En primer lugar, se define el árbol T = (V,E) cuyo conjunto de vértices es:

$$V = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9\}$$

- Se considera su secuencia de grados:

$$x_1, x_2, x_3, x_4, x_5 x_5, x_6, x_7, x_8, x_9$$

De forma que para cada vértice de V su grado es:

$$g(v_i) = x_i \; con \; v_i \in V$$

Se sabe que:

1) Se encuentran 3 vértices de grado 3 en T:

# vértices	Grado g(v)
3	3

La secuencia se va concretando:

$$3,3,3, x_4, x_5 x_5, x_6, x_7, x_8, x_9$$

2) Además, todo árbol no nulo ha de tener, como mínimo, 2 hojas, es decir, vértices de grado 1:

# vértices	Grado g(v)
3	3
≥2	1

O sea:

$$3,3,3,1,1, x_5, x_6, x_7, x_8, x_9$$

3) La fórmula de los grados, para un árbol de n vértices con m = n – 1 aristas es:

$$\sum_{v \in V}^{k} g(v) = 2 \cdot |A| = 3 \cdot 3 + 2 \cdot 1 + x_6 + x_7 + x_8 + x_9 = 2 \cdot (9 - 1)$$

O sea:

$$11 + x_6 + x_7 + x_8 + x_9 = 16$$

Es decir:

$$x_6 + x_7 + x_8 + x_9 = 5$$

4) Por último, un árbol es CONEXO, de modo que NO HAY VÉRTICES DE GRADO 0. O sea:

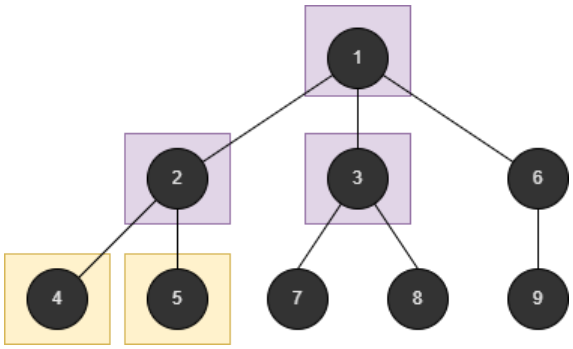
$$T \; es \; conexo \rightarrow \nexists g(v_i) = 0 \, , \forall v_i \in V$$

Es decir que $x_6, x_7, x_8, x_9 \geq 1$

5) De lo anterior, basta con ver que si $x_6 = 2$ y $x_7 = x_8 = x_9 = 1$ se satisface la secuencia gráfica.

O sea:

v_i	$g(v_i)$
1	3
2	3
3	3
4	1
5	1
6	x_6
7	x_7
8	x_8
9	x_9

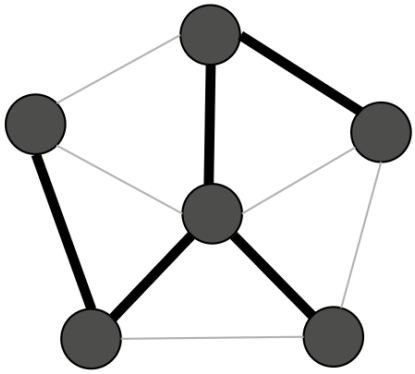


Y la secuencia gráfica es:

$$3,3,3,2,1,1,1,1,1$$

2 ÁRBOLES GENERADORES

2.1 PROPIEDADES DE ÁRBOLES GENERADORES

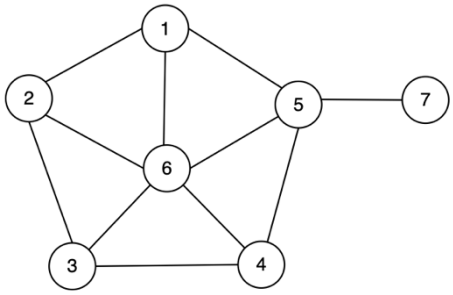


1. Un árbol generador es un subgrafo generador con estructura de árbol.
   Se denominan árboles de expansión (spanning tree).
2. Un subrafo G' = (V',E') es grafo generador de otro grafo G si se encuentra en V' todos los vértices de V.
3. Todo grafo conexo contiene necesariamente algún árbol generador (es decir, un subgrafo generador ACÍCLICO).
4. Todo grafo conexo de orden n contiene algún árbol generador de orden n y medida n – 1.
5. Todo grafo no conexo contiene, en cada una de sus componentes conexas, algún árbol generador.
   La reunión de los árboles generadores de un grafo no conexo es un bosque generador.
6. La eliminación de CUALQUIER arista de un árbol T, que es generador de G, provoca que DEJE DE SER CONEXO.
7. Si T es generador de G, y se cumple que T = G, entonces la ADICIÓN de CUALQUIER arista provoca a APARICIÓN de CICLOS.

2.2 CONSTRUCCIÓN DE ÁRBOL GENERADOR

Se puede obtener árboles generadores mediante la sucesiva supresión de aristas en grafos hasta la erradicación de sus ciclos.
Se puede emplear el algoritmo BFS.

EJEMPLO DE CONSTRUCCIÓN DE UN ÁRBOL GENERADOR:
Vía BFS, se obtiene:

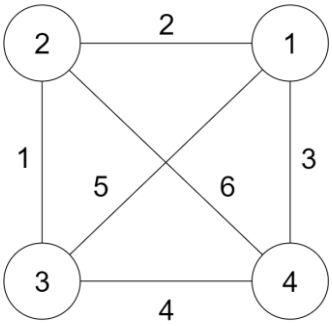


Iteración	Q	Vértices añadidos	Vértices eliminados	Conjunto V de T
0	1	1	-	{1}
1	12	2	-	{1,2}
2	125	5	-	{1,2,5}
3	1256	6	-	{1,2,5,6}
4	256	-	1	{1,2,5,6}
5	2563	3	-	{1,2,5,6,3}
6	563	-	2	{1,2,5,6,3}
7	5634	4	-	{1,2,5,6,3,4}
8	56347	7	-	{1,2,5,6,3,4,7}
9	6347	-	5	{1,2,5,6,3,4,7}
10	347	-	6	{1,2,5,6,3,4,7}
11	47	-	3	{1,2,5,6,3,4,7}
12	7	-	4	{1,2,5,6,3,4,7}
13	-	-	7	{1,2,5,6,3,4,7}

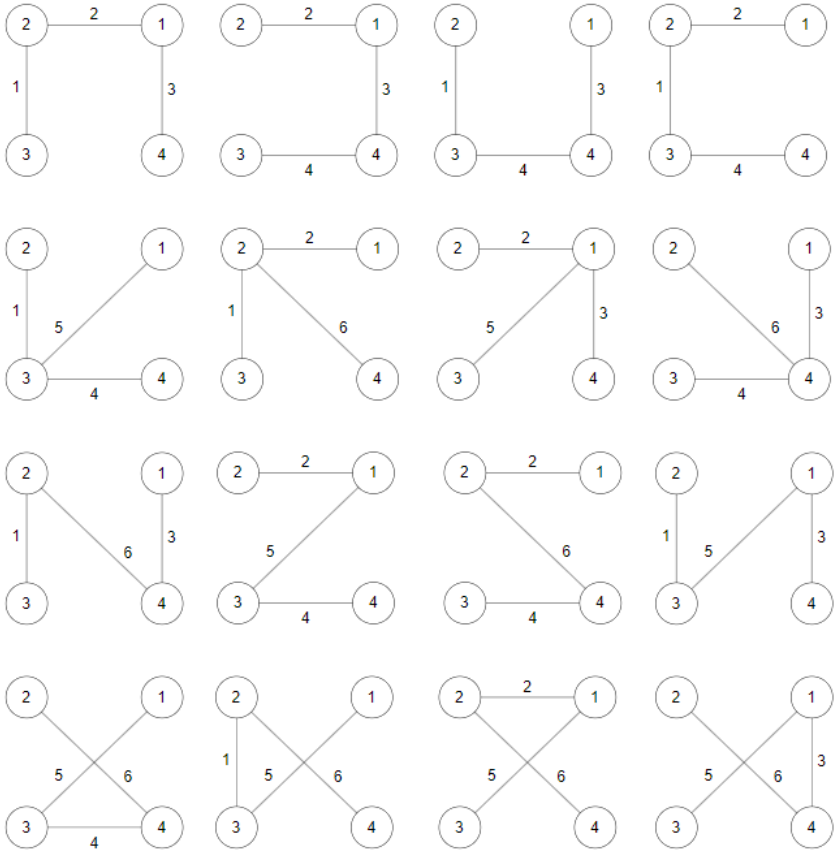
2.3 ÁRBOLES GENERADORES MINIMALES

Se consideran un grafo ponderado G = {V,A,W} y un árbol T generador de G.
Se define el PESO DEL ÁRBOL T como $w(T) = \sum_{e \in A} w(e)$.
Se define un ÁRBOL GENERADOR MINIMAL de G como un árbol T generador de G que tiene PESO w(T) MÍNIMO.

Para un grafo G NO CONEXO, se considera un BOSQUE GENERADOR MINIMAL con peso mínimo de entre todos los bosques generadores de G.
Ejemplo:



Dado el grafo ponderado G, se observan 16 árboles generadores:



De entre los cuales, hay uno de PESO MÍNIMO (presentado el primero).

2.4 ALGORITMO DE KRUSKAL

2.4.1 Definición del algoritmo de Kruskal

¡Este simulador está MUY BIEN!

- A partir de un grafo G = (V,A) de orden n, permite obtener un árbol generador minimal T = (V,A') que conecta TODOS los vértices de G mediante las n – 1 aristas de MENOR PESO de forma acíclica. Es decir, el árbol T resultante:
    - Es conexo.
    - Involucra TODOS los vértices de G.
    - Es ACÍCLICO.
    - Tiene PESO MÍNIMO.
- Opera de acuerdo con el principio de voracidad, es decir, tomando la decisión de menor coste en el contexto local en que se encuentra la ejecución en cada momento.
- El árbol minimal obtenido se representa como una lista de aristas, que empieza vacía y se va incrementando en una arista en cada iteración.
- En cada iteración, de entre las aristas de G, se escoge para incluir en T la arista DE MENOR PESO que no forme ciclos con las aristas ya presentes en T.
- Por tanto, NO HACE FALTA QUE LA ARISTA INCORPORADA SEA ADYACENTE A LAS YA INCORPORADAS.
- Se finaliza cuando se han elegido n – 1 aristas.

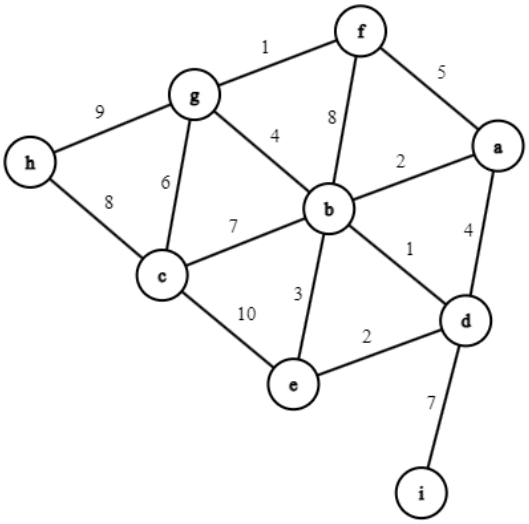
Se implementa mediante:

- Un grafo ponderado conexo G = {V,A,W} en forma de LISTA DE ADYACENCIAS.
- Una LISTA DE ARISTAS del grafo G ORDENADAS POR PESO.
- Una ESTRUCTURA EFICIENTE para PREVENIR la formación de CICLOS, o sea, comprobar que algún vértice de la arista evaluada en cada momento no se encuentre ya como extremo de alguna arista previamente incorporada al árbol T.
    - Si ambos extremos de la arista evaluada en un momento de la ejecución ya pertenecen al conjunto d vértices incorporados, la arista se descarta.
    - Si la unión entre el conjunto de vértices incorporados y la arista (representada por los 2 vértices que conecta) es DISJUNTA (no hay elemento sen común), la arista se puede incorporar.
- Un árbol T resultante.

Su ejecución manual se puede resumir en:

1. Se ORDENAN las aristas A en orden creciente según su PESO en una lista.
2. Se selecciona de la lista la arista DE MENOR PESO DISPONIBLE (no incluida en A').
3. Si la arista seleccionada:
    - FORMA CICLOS con alguna arista ya incluida en A' Se descarta.
    - NO FORMA CICLOS con ninguna arista ya incluida en A' Se incorpora a A'.
4. Se repite hasta haber incorporado a A' n – 1 aristas, es decir, los n vértices, sin formar ningún ciclo.
5. Los criterios con que se ordenan aristas de igual peso dan lugar a árboles minimales distintos. Por tanto, LA SOLUCIÓN NO ES ÚNICA.

2.4.2 Ejemplo de ejecución de algoritmo de Kruskal



1. Se ordenan las aristas del grafo G según su peso:

Arista	d,b	g,f	a,b	d,e	e,b	b,g	d,a	a,f	c,g	b,c	d,i	f,b	c,h	g,h	c,e
Peso	1	1	2	2	3	4	4	5	6	7	7	8	8	9	10

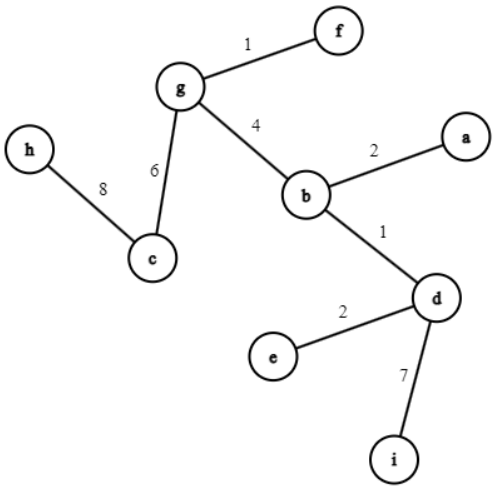
2. Se declara el árbol minimal generador T = (V,A') como una lista de aristas A' vacía:

A' = {Ø}

3. Se recorre la lista en orden de modo que se incorporan las n – 1 = 9 – 1 = 8 aristas de menor peso que no forman ciclo con las ya incorporada spreviamente (seleccionadas denotada por * y aristas descartadas en **negrita**):

Arista	(d,b)*	(g,f)*	(a,b)*	(d,e)*	**e,b**	(b,g)*	**d,a**	**a,f**	(c,g)*	**b,c**	(d,i)*	**f,b**	(c,h)*	**g,h**	**c,e**
Peso	1	1	2	2	3	4	4	5	6	7	7	8	8	9	10

Paso a paso es:



it 0 A' = {Ø}
it 1 A' = {(d,b)}
it 2 A' = {(d,b),(g,f)}
it 3 A' = {(d,b),(g,f),(a,b)}
it 4 A' = {(d,b),(g,f),(a,b),(d,e)}
it 5 Se descarta (e,b) porque forma ciclo.
it 6 A' = {(d,b),(g,f),(a,b),(d,e),(b,g)}
it 7 Se descarta (d,a) porque forma ciclo.
it 8 Se descarta (a,f) porque forma ciclo.
it 9 A' = {(d,b),(g,f),(a,b),(d,e),(b,g),(c,g)}
it 10 Se descarta (b,c) porque forma ciclo.
it 11 A' = {(d,b),(g,f),(a,b),(d,e),(b,g),(c,g),(d,i)}
it 12 Se descarta (f,b) porque forma ciclo.
it 13 A' = {(d,b),(g,f),(a,b),(d,e),(b,g),(c,g),(d,i),(c,h)}

La lista ha alcanzado n – 1 aristas. La ejecución se detiene.

- Nótese que las iteraciones 14 y 15 no llegan a realizarse:

it 14 Se descarta (g,h) porque forma ciclo.
it 15 Se descarta (c,e) porque forma ciclo.

Se verifica que TODOS los vértices de G están en T y que hay n – 1 = 8 aristas en T y el árbol minimal T es el representado arriba, cuyo coste total es de 8 + 6 + 1 + 4 + 2 + 1 + 2 + 7 = 31

2.4.3 Complejidad del algoritmo de Kruskal

Se distinguen 2 tipos de operaciones:

- La ORDENACIÓN de una lista de m aristas tiene complejidad O(m · log(m)) mediante algoritmos de clasificación como el Quicksort.

- El BUCLE principal, cuya complejidad global es de O(n·log(m)):
    - Se ejecuta n – 1 veces.
    - Realiza 2 operaciones, cuya complejidad es O(log(m)):
        - 2 búsquedas (en el conjunto de origen A y en el de destino A') para verificar el carácter acíclico de la posible incorporación.
        - 1 unión de conjuntos disyuntos (en caso de añadir la arista).

Por tanto, la complejidad de ejecución se encuentra en $\max(O(m \cdot \log(m)), O(n \cdot \log(m))$

Pero todo árbol cumple que m = n – 1, de manera que la complejidad real del algoritmo es siempre:

$$O(m \cdot \log(m)$$

2.5 Algoritmo de Prim

2.5.1 Definición del algoritmo de Prim

Opera de forma similar al algoritmo de Kruskal, con la diferencia de que el algoritmo de Prim EXIGE QUE LA ARISTA INCORPORADA SEA ADYACENTE a alguna de las ya incorporadas en el árbol generador minimal.
Por tanto, permite la expansión de un solo árbol CONEXO en todo momento de la ejecución.

Es decir, la arista incorporada:
- No debe estar previamente incorporada al árbol minimal.
- Su inclusión no debe formar CICLOS con las aristas ya incorporadas.
- Debe ser la de MENOR peso accesible.
- Debe ser ADYACENTE a alguna de las ya incorporadas.

La adyacencia de las incorporaciones garantiza el carácter CONEXO del árbol minimal.
Finaliza, como el algoritmo de Kruskal, cuando se han añadido al árbol n – 1 aristas.

En caso de encontrar múltiples aristas que satisfagan las condiciones de incorporación, se recurre al criterio de ordenación que presente la lista de aristas.

Se implementa mediante:
- Un grafo ponderado conexo G = {V,A,W} en forma de LISTA DE ADYACENCIAS.
- Una lista de vértices recorridos en el ordene n que se han visitado.
- Una tabla de pesos que refleja el peso de la arista de menos peso que conecta un vértice con otro, cuyas etiquetas se van actualizando en cada iteración, de forma similar a cómo corre en el algoritmo de Dijkstra.

La ejecución manual se puede formular como:

1. Se elige, como vértice inicial, uno cualquiera, al cual se le asigna el pivote de ejecución.
2. Se examinan las adyacencias del vértice pivote:
   a. Se actualiza la etiqueta de la de menor peso que no forme ciclos con el recorrido ya incorporado a T. Esa etiqueta ya es definitiva.
   b. Se incorpora ese vértice al recorrido (U = 1 en la tabla).
   c. Se desplaza el pivote al vértice recientemente incorporado al árbol mínimo.
3. Se repite el paso 2 hasta que el árbol T resultante contiene n – 1 aristas.

2.5.2 Ejemplo de ejecución de algoritmo de Prim

Se tiene el grafo:

4. Se ordenan las aristas del grafo G según su peso:
5. Se declara el árbol minimal generador T = (V,A') como una lista de aristas A' vacía:

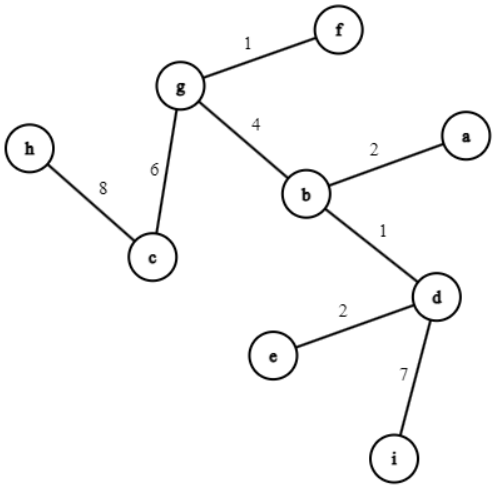
Arista	d,b	g,f	a,b	d,e	e,b	b,g	d,a	a,f	c,g	b,c	d,i	f,b	c,h	g,h	c,e
Peso	1	1	2	2	3	4	4	5	6	7	7	8	8	9	10

A' = {Ø}

6. Se recorre la lista en orden de modo que se incorporan las n – 1 = 9 – 1 = 8 aristas de menor peso que no forman ciclo con las ya incorporada spreviamente (seleccionadas denotada por * y aristas descartadas en **negrita**):

Arista	(d,b)*	(g,f)*	(a,b)*	(d,e)*	**e,b**	(b,g)*	**d,a**	**a,f**	(c,g)*	**b,c**	(d,i)*	**f,b**	(c,h)*	**g,h**	**c,e**
Peso	1	1	2	2	3	4	4	5	6	7	7	8	8	9	10

Paso a paso es:



it 0	A' = {Ø}
it 1	A' = {(d,b)}
it 2	A' = {(d,b),(g,f)}
it 3	A' = {(d,b),(g,f),(a,b)}
it 4	A' = {(d,b),(g,f),(a,b),(d,e)}
it 5	Se descarta (e,b) porque forma ciclo.
it 6	A' = {(d,b),(g,f),(a,b),(d,e),(b,g)}
it 7	Se descarta (d,a) porque forma ciclo.
it 8	Se descarta (a,f) porque forma ciclo.
it 9	A' = {(d,b),(g,f),(a,b),(d,e),(b,g),(c,g)}
it 10	Se descarta (b,c) porque forma ciclo.
it 11	A' = {(d,b),(g,f),(a,b),(d,e),(b,g),(c,g),(d,i)}
it 12	Se descarta (f,b) porque forma ciclo.
it 13	A' = {(d,b),(g,f),(a,b),(d,e),(b,g),(c,g),(d,i),(c,h)}

La lista ha alcanzado n – 1 aristas. La ejecución se detiene.

- Nótese que las iteraciones 14 y 15 no llegan a realizarse:

it 14 Se descarta (g,h) porque forma ciclo.
it 15 Se descarta (c,e) porque forma ciclo.

Se verifica que TODOS los vértices de G están en T y que hay n – 1 = 8 aristas en T y el árbol minimal T es el representado arriba, cuyo coste total es de 8 + 6 + 1 + 4 + 2 + 1 + 2 + 7 = 31

2.5.3 Complejidad del algoritmo de Prim

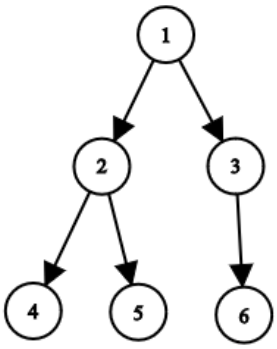
3 ÁRBOLES CON RAÍZ

3.1 CARACTERÍSTICAS DE LOS ÁRBOLES CON RAÍZ

3.1.1 Conceto de predecesor y sucesor

En un grafo dirigido G = (V,A), todo arco (u,v) permite distinguir:

- Vértices PREDECESORES Situados entre el origen del arco y la raíz.
- Vértices SUCESORES Situados entre el destino del arco y las hojas.



El vértice 1 es predecesor de todos los demás.

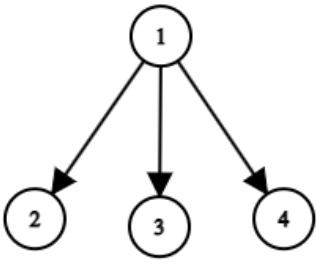
Los vértices 2, 3, 4, 5 y 6 son sucesores del vértice 1.

El vértice 2 es predecesor de 4 y 5.

El vértice 4 es sucesor del vértice 2.

3.1.2 Conceptos de vértice padre, vértice hijo y vértices hermanos

En un grafo dirigido G = (V,A), si hay un arco (u,v), se distinguen:



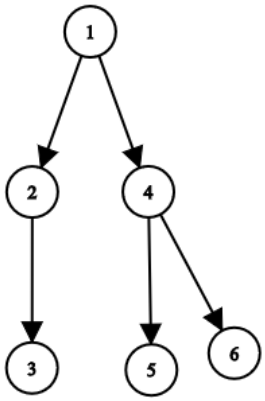
- El vértice u como PADRE (superior)
- El vértice v como HIJO. (inferior)

En el digrafo G, el vértice 1 es PADRE de 2, 3 y 4, que, a su vez, son HIJOS de 1.

Asimismo, 2, 3 y 4 son vértices hermanos entre ellos, ya que parten del mismo padre.

Por tanto, todo padre da lugar a algún hijo y todo hijo parte de un padre.

3.1.3 Digrafo ASIMÉTRICO

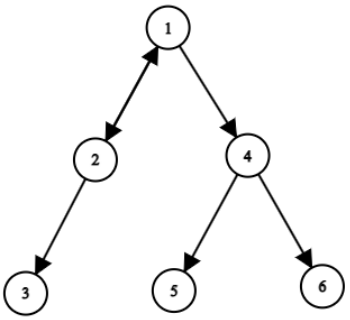


Un digrafo es ASIMÉTRICO si no tiene ciclos de longitud 2, de modo que en todo momento el recorrido ESTÁ ORIENTADO en un único sentido.

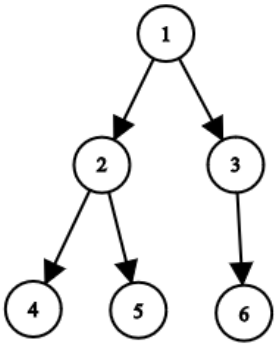
Esto se cumple en G = (V,A) si:

$$\forall (u,v) \in A \rightarrow \nexists (v,u) \in A$$

En cambio, el siguiente digrafo no es asimétrico, pues presenta los arcos (1,2) y (2,1):



3.1.4 Raíz de un digrafo



Un vértice r es RAÍZ de un digrafo G = (V,a) si existe un camino DIRIGIDO desde r y hasta cada otro vértice v del digrafo. Es decir:

$$\exists\, u - v\ \forall v\ \in V$$

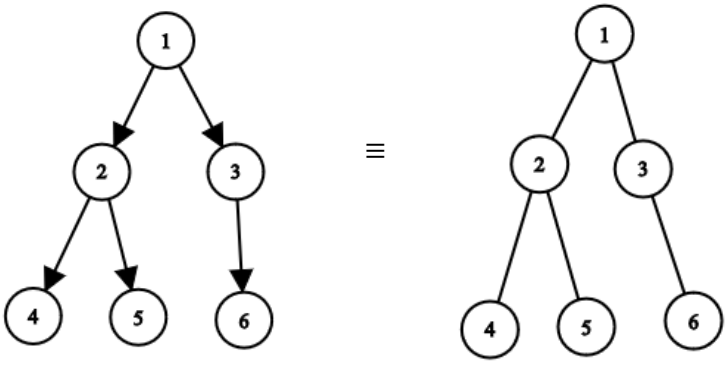
En el siguiente digrafo G, el vértice 1 es Raíz de G.

3.1.5 Definición de árbol con raíz

Se define como ÁRBOL CON RAÍZ un digrafo T = (V,A) que cumple:

1. T es digrafo asimétrico (carente de ciclos de longitud 2).
2. El mismo grafo T NO DIRIGIDO, denominado grafo subyacente, también debe ser árbol.
3. Hay al menos un camino DIRIGIDO r → v desde la raíz r hasta CADA OTRO vértice v.
4. Su raíz r cumple $g^-(r) = 0$, es decir, NINGÚN ARCO incide sobre r.
5. Todos sus vértices DISTINTOS DE LA RAÍZ r cumplen $g^-(v) = 1$, lo cual conlleva que, DESDE CADA VÉRTICE, EL CAMINO HASTA LA RAÍZ ES ÚNICO.
6. El grafo subyacente (igual que el digrafo T pero NO orientado), tiene como opuesto $\bar{T}$ un grafo CONEXO con un vértice r que satisface también las propiedades 4 y 5.

En el contexto de los ÁRBOLES, ya que se representan de forma jerárquica, la raíz se observa convencionalmente en LO MÁS ALTO, de modo que es habitual omitir el carácter dirigido de los arcos, que se representan como en lugar de flechas.



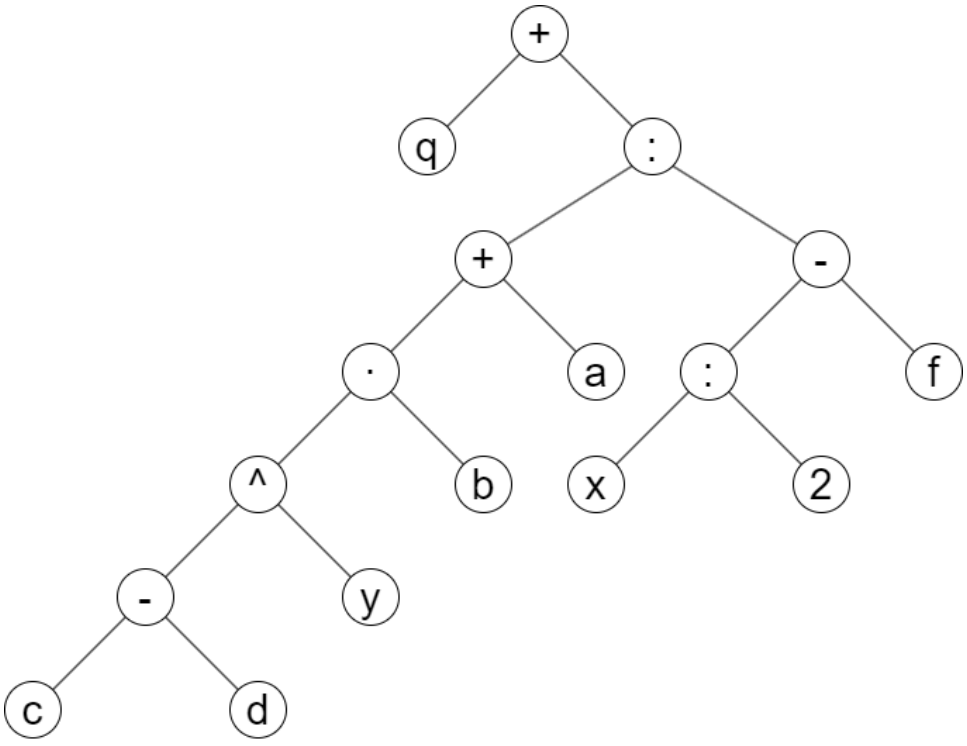
3.1.6 Notación de árbol para operaciones aritméticas

Los árboles con raíz ofrecen, entre otras muchas aplicaciones, la posibilidad de representar relaciones algebraicas que corresponden con las relaciones entre operadores y operandos.

Ejemplo:

Se desea representar la operación $\frac{a+b\cdot(c-d)^y}{\frac{x}{2}-f}+q$ como árbol.

Basta con seguir la precedencia de operaciones para obtener el árbol con raíz mostrado



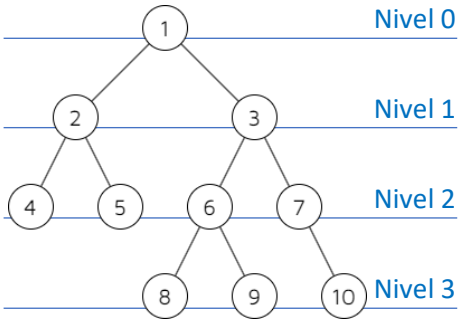
3.2 ÁRBOLES M-ARIOS

3.2.1 Nivel de un vértice

El nivel de un vértice v de un árbol T es la longitud (número de aristas) del camino único r – v

Se inicia en el nivel 0, ya que la longitud del camino r – r para ir de la raíz r hasta la raíz r es 0.

Por ejemplo, este árbol tiene 4 niveles, del 0 al 3



Nivel 0: vértice 1
Nivel 1: vértices 2 y 3
Nivel 2: vértices 4, 5, 6 y 7
Nivel 3: vértices 8, 9 y 10

3.2.2 Altura o profundidad h(T) de un árbol T

La altura o profundidad de un árbol es el máximo de entre los niveles de sus vértices:

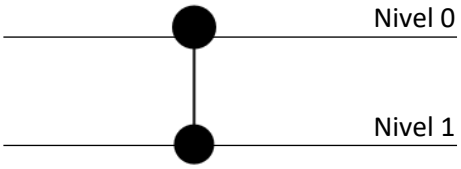
$$h(T) = \max(nivel(v) | \, v \in V)$$

La altura SE INICIA EN 0, ya que se relaciona con los niveles del grafo.

Nótese que un árbol trivial de orden n tiene altura n – 1:

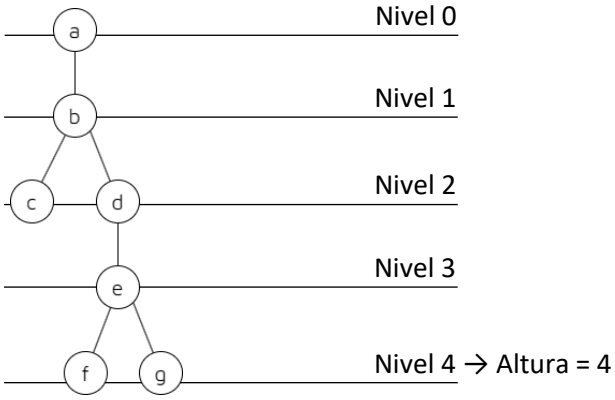
→ Altura = 0

Para un árbol formado por un padre y un hijo:



→ Altura = 1

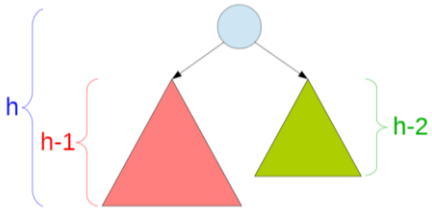
Este es un árbol de 5 niveles (del 0 al 4), cuya PROFUNDIDAD o ALTURA es 4:



3.2.3 Árbol equilibrado

Un árbol equilibrado es aquel que solo presenta hojas en el último nivel y en el penúltimo.

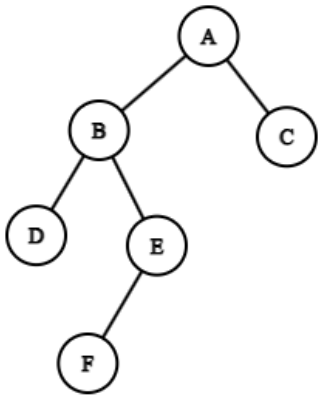
Un árbol binario es equilibrado si para cualquiera de sus vértices se cumple que su subestructura por la izquierda presenta la misma altura que su subestructura por la derecha, o bien, esas alturas difieren, como máximo, en 1 solo nivel.



Formalmente, la condición de equilibrio em altura se da si los subárboles inducidos por vértices hermanos (que comparten padre) no difieren en una altura de más de 1 unidad.

Esta propiedad es especialmente relevante en el ámbito de los árboles AVL (o sea de Adelson-Velskii y Landis), puesto que permite el diseño de estructuras de datos capaces de reestructurarse mediante ciertas transformaciones denominadas rotaciones para dar lugar a árboles binarios equilibrados, lo cual permite explorarlas de forma mucho más eficiente.

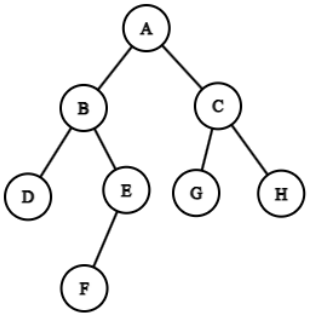
EJEMPLO



Como se observan hojas en los niveles 3 (vértice F), 2 (vértice D) y 1 (vértice C), el árbol es no equilibrado.

Desde el vértice raíz A hay una componente inducida por la derecha (hacia C) Cuya altura es de 1, mientras que la componente inducida por la izquierda (hacia B) tiene altura 3.

EJEMPLO



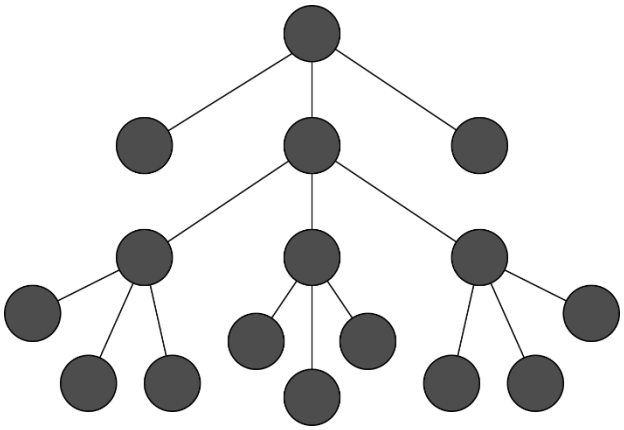
En cambio, este sí es un árbol equilibrado puesto que presenta hojas exclusivamente en los últimos dos niveles.

3.2.4 Árbol m-ario

Se denomina árbol m-ario a un árbol cuyos vértices tienen, como máximo, un grado de salida comprendido entre 0 y m.
Nótese que m se usa para representar el factor de ramificación y no la medida.
Es decir, el parámetro m se define por el máximo de entre los grados de salida de los vértices.

O sea: $0 \leq g^{+}(v) \leq m$

EJEMPLO DE ÁRBOL TERNARIO (m = 3)



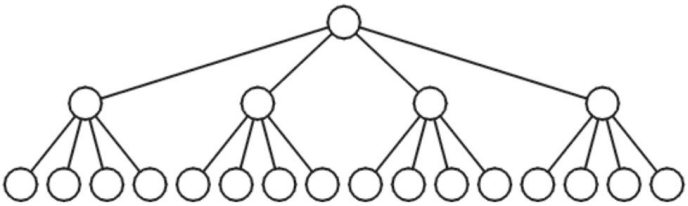
El grafo es 3-ario porque todos sus vértices internos son, como máximo de grado 3.

Además, es LLENO, ya que TODOS sus vértices internos son de grado m = 3.

No es PERFECTO, ya que no todas las hojas están al mismo nivel.

Es EQUILIBRADO, ya que solo hay hojas en los 2 niveles más alejados de la raíz.

EJEMPLO DE ÁRBOL 4-ARIO (m = 4)



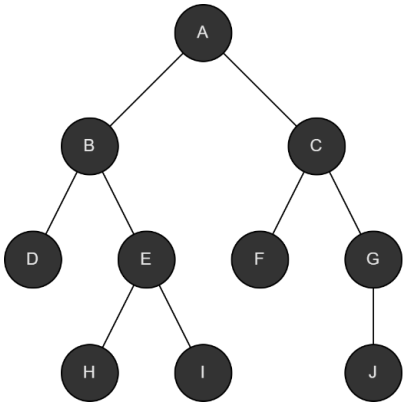
El grafo es 4-ario ya que todos sus vértices internos son, como máximo, de grado m = 4.

El grafo NO ESTÁ LLENO ya que NO TODOS los vértices internos son de grado m.

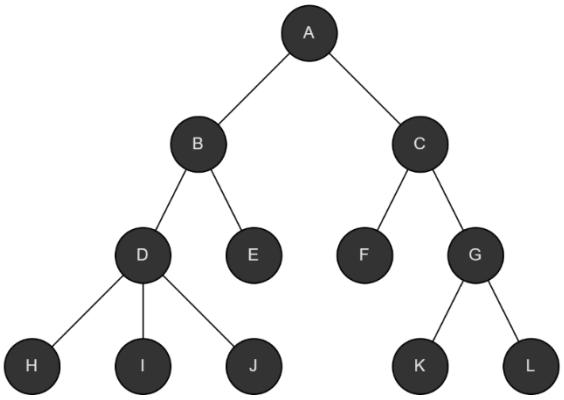
3.2.5 Árbol binario

Se trata de un árbol m-ario con m = 2. Es decir, un árbol en que cada vértice interno tiene, como MÁXIMO, 2 hojas.

EJEMPLO



Este árbol es binario (m-ario con m = 2) ya que no hay vértice interno que tenga más de 2 hojas.



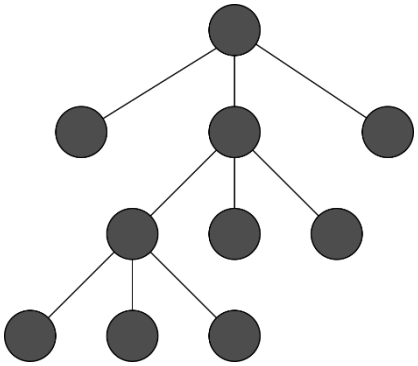
Este árbol NO ES BINARIO ya que el vértice interno D tiene más de 2 hojas.

3.2.6 Árbol m-ario lleno (full)

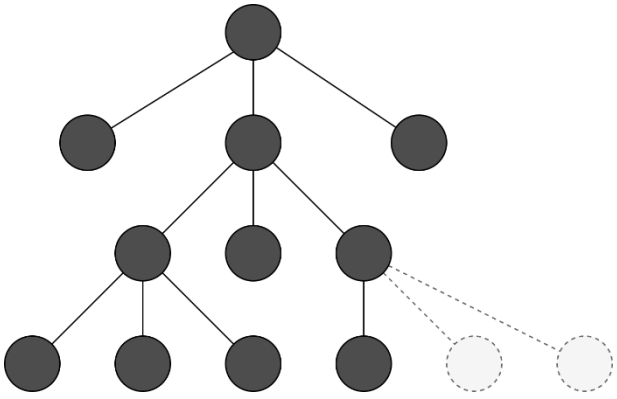
Un árbol m-ario es LLENO si TODO VÉRTICE INTERNO tiene EXACTAMENTE m hijos.

O sea, TODO vértice de un árbol m-ario LLENO pertenece exclusivamente a 1 de 2 categorías:

- O bien es VÉRTICE INTERNO con grado de salida $g^+(v) = m$.
- O bien es HOJA con grado de salida $g^+(v) = 0$.



Este grafo 3-ario es LLENO, ya que todo padre tiene m hijos.



En cambio, este grafo 3-ario es NO LLENO, ya que hay algún padre que ni tiene grado m = 3.

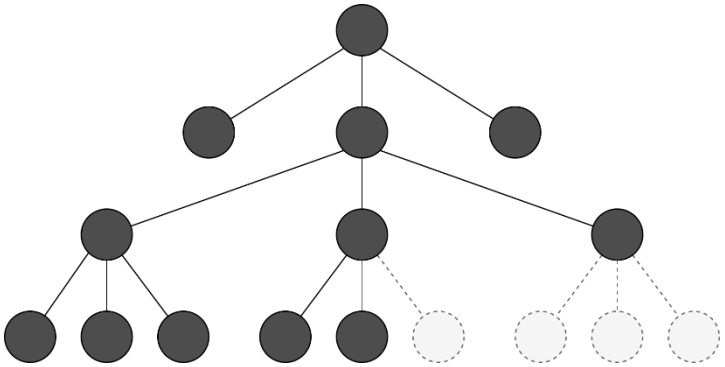
Aunque no alcanza a ser lleno, SÍ ES COMPLETO, ya que las hojas que le faltan para ser lleno:

1) Solo faltan en el último nivel.
2) Solo faltan por la derecha.

3.2.7 ÁRBOL m-ARIO COMPLETO

Un árbol es completo es un árbol lleno al que le faltan hojas de cierta forma en particular:
1) SOLO faltan hojas en el último nivel
2) SOLO faltan específicamente por la derecha.

Por así decir, tiene todas las ramas para ser m-ario lleno, pero "faltan algunas hojas" por la derecha: es "casi lleno".

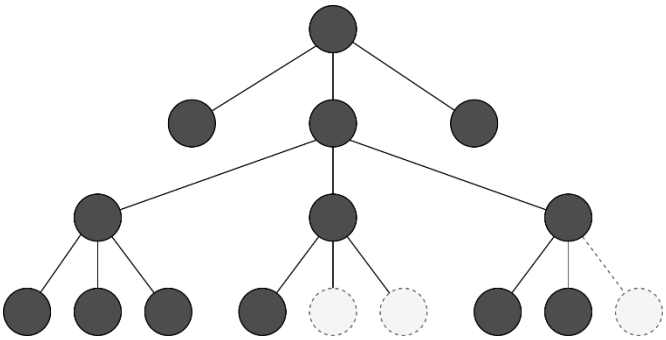


EJEMPLO DE ÁRBOL 3-ARIO COMPLETO

Este árbol 3-ario es completo ya que cumple ambos criterios:

1) Solo faltan hojas en el último nivel.
2) Solo faltan por la derecha.

Sucede además que NO ES PERFECTO, porque tiene hojas fuera del último nivel.



EJEMPLO DE ÁRBOL 3-ARIO NO COMPLETO

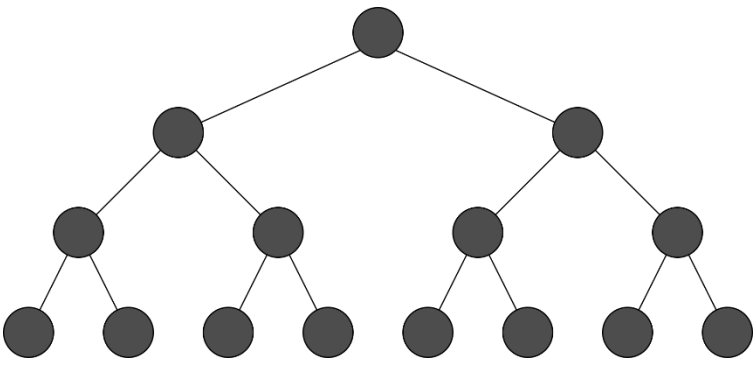
Este grafo cumple solo 1 de los 2 requisitos para ser completo:
Faltan hojas SOLO en el último nivel para ser lleno, pero NO faltan SOLO por la derecha.

Como las hojas que lo diferencian de un grafo 3-ario lleno NO se concentran a la derecha

3.2.8 ÁRBOL M-ARIO PERFECTO

Un árbol m-ario es PERFECTO si cumple 2 requisitos:

- Todos los vértices internos son de grado m, es decir, está LLENO.
- Todas las hojas están en el mismo nivel.



EJEMPLO DE ÁRBOL BINARIO PERFECTO

Un árbol m-ario perfecto es un árbol m-ario lleno que SOLO tiene hojas en el último nivel.

3.2.9 Relación entre vértices internos y hojas en árboles m-arios COMPLETOS

Un árbol m-ario COMPLETO T = (V,A) con t hojas e i vértices internos cumple:

$$t = i(m-1) + 1$$

Nótese que el orden n debe cumplir:

$$n = t + i$$

Ya que todo vértice o bien es interno o bien es hoja.

Entonces, aislando n, se tiene:

$$n = t + i = mi + 1$$

O escrito de otra manera:

$$n - 1 = mi$$

Además, considerando:

$$\begin{cases} n - 1 = mi \\ n = t + i \end{cases}$$

Aislando n e igualando, se alcanza, para todo árbol COMPLETO m-ario:

$$mi + 1 = t + i$$

De lo cual, aislando i:

$$i = \frac{t-1}{m-1}$$

3.2.10 Relación entre la altura h y el máximo número de hojas

Para un árbol m-ario de altura h se cumple que el número MÁXIMO de hojas t es:

$$t \leq m^h$$

3.2.11 Relación entre la altura mínima de un árbol m-ario y su número de hojas t

De lo anterior, se desprende que la altura h de todo árbol m-ario con m ≥ 2 y t hojas, satisface:

$$h \geq \lceil \log_m t \rceil$$

Es decir, la altura mínima para un árbol m-ario corresponde al entero siguiente del logaritmo del número de hojas t en base m.

3.2.12 Relación entre la altura mínima de un árbol m-ario COMPLETO y EQUILIBRADO y su número de hojas t

De lo anterior, para un árbol COMPLETO y EQUILIBRADO, se desprende que la altura h cumple:

$$h = \lceil \log_m t \rceil$$

3.3 Exploración de árboles binarios con raíz: PREORDEN, INORDEN Y POSTORDEN

Se identifican 3 algoritmos principales para la exploración de árboles binarios:

- Exploración en PREORDEN (o en profundidad)
- Exploración en INORDEN
- Exploración en POSTORDEN

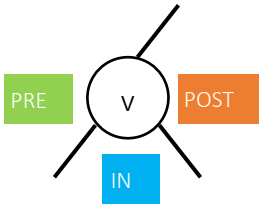
Topológicamente, el trayecto de exploración es el mismo en los 3 casos.
Se distinguen según el momento del recorrido en el cual los vértices ingresan en la secuencia de vértices explorados:

- Exploración en PREORDEN — El vértice se incluye la PRIMERA vez que se recorre.
- Exploración en INORDEN — El vértice se incluye la SEGUNDA vez que se recorre.
- Exploración en POSTORDEN — El vértice se incluye la TERCERA vez que se recorre.

Estos algoritmos ACCEDEN a cada vértice 1 sola vez, aunque el pivote que guía su ejecución manual pase más de una vez por ellos.

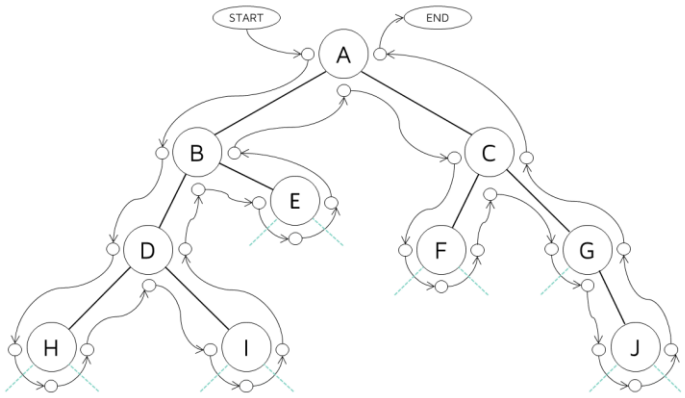
Los 3 algoritmos tienen complejidad lineal de acuerdo con el orden n del árbol, es decir, O(n).
Se considera una raíz r cuyos hijos son v_1 y v_2, de los cuales se desprenden los subárboles T_1 y T_2 respectivamente.

Para aplicarlo, basta considerar 3 posiciones en torno a cualquier vértice v, tenga o no tenga hijos, de acuerdo con el recorrido direccional a través de un grafo:

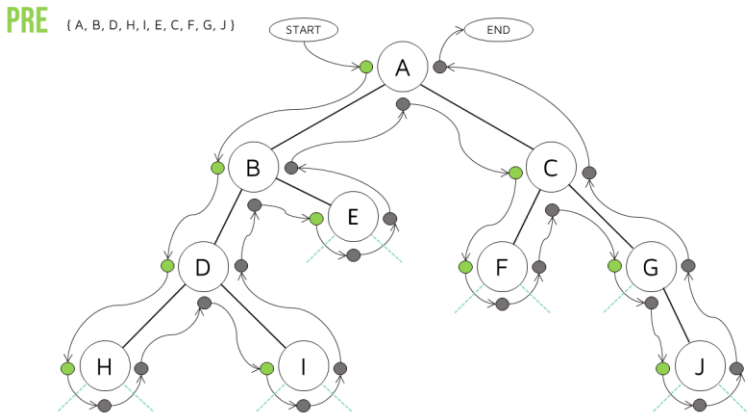


- Posición PRE
- Posición IN
- Posición POST

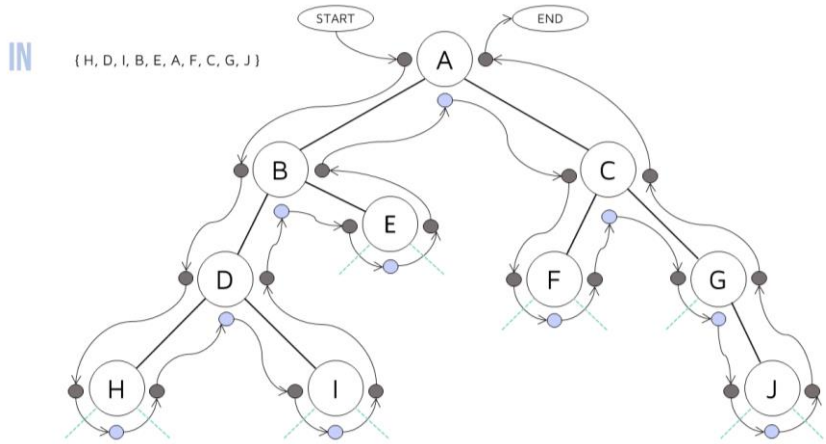
Y considerar el recorrido direccional de lectura, que es, por convenio:



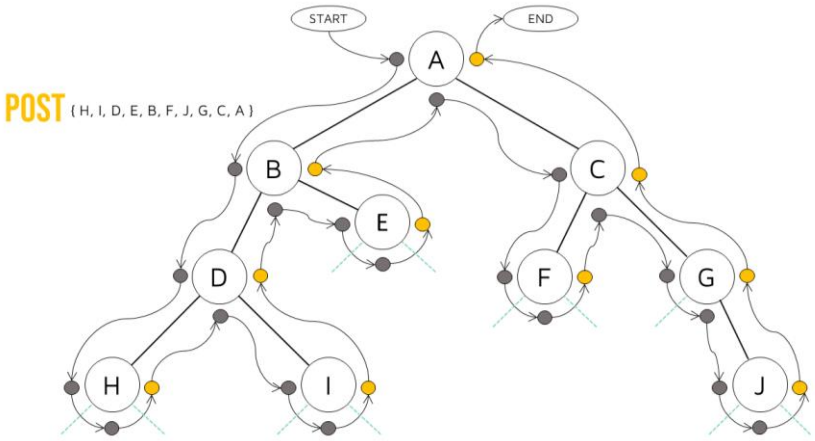
Un vértice ingresa en la lista de vértices explorados en PREORDEN en el momento en que es alcanzado por el recorrido en la posición PRE:



Un vértice ingresa en la lista de vértices explorados en INORDEN en el momento en que es alcanzado por el recorrido en la posición IN:

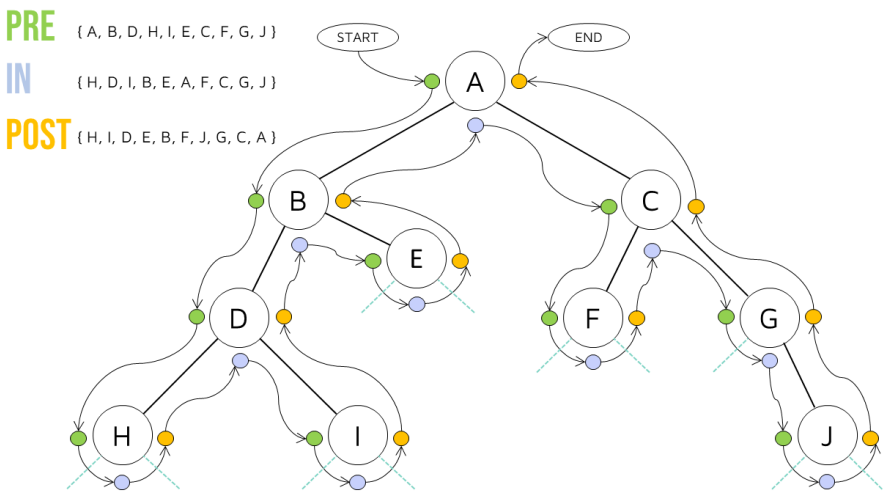


Un vértice ingresa en la lista de vértices explorados en POSTORDEN en el momento en que es alcanzado por el recorrido en la posición POST:



EJEMPLO

ALGORITMO PREORDEN-INORDEN-POSTORDEN



Otros algoritmos de exploración de árboles binarios son el recorrido por niveles y la ordenación de mayor a menor:

1. Recorrido de árbol binario por niveles:

Se visitan todos los vértices de un nivel antes de visitar el primer vértice del nivel siguiente.
O sea:

- Primero la raíz (nivel 0)
- Luego sus hijos (nivel 1)
- Luego sus nietos (nivel 2)
- …

2. Árbol binario ORDENADO: Máximo y mínimo de un árbol binario

La estructura binaria del árbol permite su ordenación de acuerdo con un solo criterio:
Todo vértice a la izquierda de una raíz es MENOR que ella.

Los conceptos de ORDEN en un árbol binario y los recorridos en PREORDEN, INORDEN y POSORDEN permiten definir:

- Máximo del árbol binario último elemento de INORDEN y de PREORDEN
  Se trata del hijo de más a la IZQUIERDA del último nivel.

- Mínimo del árbol binario primer elemento de INORDEN y de POSORDEN
  Se trata del hijo de más a la DERECHA del último nivel.

3.4 Complejidad de la búsqueda

El orden en un árbol permite caracterizar la búsqueda en su estructura mediante un comportamiento recursivo:

- Si el valor buscado es MENOR que el del vértice visitado, se sigue a la IZQUIERDA.
- Si el valor buscado es MAYOR que el del vértice visitado, se sigue a la DERECHA.

La complejidad de esta búsqueda es proporcional a la altura h y, en promedio es del orden:

$$O(\log(h))$$

En el peor de los casos, se tendrá un árbol como lista, o sea:

$$O(n)$$

3.5 EJERCICIOS

1) ¿cuál es el número máximo de hojas que tiene un árbol 11-ario de altura 7?

Para un árbol m-ario de altura h se cumple que el número MÁXIMO de hojas t es:

$$t \leq m^h$$

En este caso:

$$t \leq 11^7 = 19487171\ hojas$$

2) ¿Cuál es la altura mínima de un árbol 14-ario con 462 hojas?

Se tiene, para todo árbol m-ario:

$$t \leq m^h$$

De lo cual:

$$h \geq \lceil \log_m t \rceil$$

O sea, hay que tomar el SIGUIENTE ENTERO al resultado de $\log_m t$
En este caso:

$$h \geq \lceil \log_{14} 462 \rceil = \lceil 2{,}3 \rceil = 3$$

3) ¿Cuál es el número de vértices internos de un árbol completo 7-ario de 151 hojas?

Un árbol m-ario COMPLETO con t hojas e i vértices internos cumple:

$$t = i(m-1) + 1$$

O sea:

$$i = \frac{t-1}{m-1}$$

En este caso:

$$i = \frac{t-1}{m-1} = \frac{151-1}{7-1} = \frac{150}{6} = 25\ vértices\ internos$$

4) ¿Cuántas hojas tiene un árbol completo 7-ario con 187 vértices internos?

Sabiendo que el árbol es completo, se cumple:

$$t = i(m-1) + 1 = 187 \cdot (7-1) + 1 = 1123\ hojas$$

5) Sea T un árbol de orden 47 con un mínimo de 39 vértices de grado 1 y un mínimo de 6 vértices de grado 3. ¿Cuánto vale la suma de los grados de los otros dos vértices?

Se sabe: $n = 47$

Se considera la suma total de grados S:

$$S = \sum_{v \in V}^{k} g(v) = 39 \cdot 1 + 6 \cdot 3 + x$$

Donde x es la suma parcial de incidencias desconocida de los otros vértices, aparte de las 39 hojas y los 6 de grado 3.

Pero todo árbol cumple:

$$n = m + 1$$

Entonces:

$$m = n - 1 = 47 - 1 = 46$$

Y todo grafo, sea o no árbol, satisface el teorema de Euler:

$$\sum_{v \in V}^{k} g(v) = 2 \cdot m$$

En este caso:

$$S = \sum_{v \in V}^{k} g(v) = 2 \cdot 46 = 92$$

De lo cual:

$$S = 92 = 39 \cdot 1 + 6 \cdot 3 + x$$

O sea:

$$x = 92 - 39 - 18 = 35$$

6) ¿Se puede construir un árbol m-ario completo y equilibrado con 423 hojas en caso de que el árbol sea binario? ¿Y ternario? ¿Y cuaternario?

Para un árbol m-ario, se considera su orden n, su número de hojas t, su número de vértices internos i y su altura h.

Se cumple, para todo árbol m-ario COMPLETO:

$$n = t + i$$
$$n = mi + 1$$

De lo cual:

$$i = \frac{t-1}{m-1}$$

Además, si el árbol m-ario completo también es EQUILIBRADO:

$$h = \lceil \log_m t \rceil$$

En este caso, se tiene t = 423 hojas

Con m = 2, se observa:

$$i = \frac{t-1}{m-1} = \frac{423-1}{2-1} = 422\ vértices\ internos$$

Por tanto, el orden del árbol es:

$$n = 422 + 423 = 845$$

Esos 845 vértices del árbol completo y equilibrado están distribuidos en los niveles de una altura de 0 hasta h, que cumple:

$$h = \lceil \log_m t \rceil = \lceil \log_2 423 \rceil = 9$$

Si el árbol fuera PERFECTO, la distribución de los 845 vértices en esos 9 niveles sería:

Nivel	0	1	2	3	4	5	6	7	8	9
vértices	1 (raíz)	2	4	8	16	32	64	128	256	512

Pero el árbol no es perfecto, de modo que se tiene:

1 + 2 + 4 + 8 + 16 + 32 + 64 + 128 + 256 = 511 vértices ocupando los niveles superiores y, por tanto, solo quedan 845 – 511 = 334 para el último nivel.

Todos los vértices del último nivel son hojas. Por tanto, se tienen 334 hojas en el último nivel y 423 – 334 = 89 hojas en el nivel inmediatamente superior.