Teórica 4: Árboles

En esta clase vamos a estudiar un tipo de grafos particular: los árboles. Estos grafos aparecen reiteradamente en la práctica y tienen propiedades que los hacen más dóciles que los grafos en general.

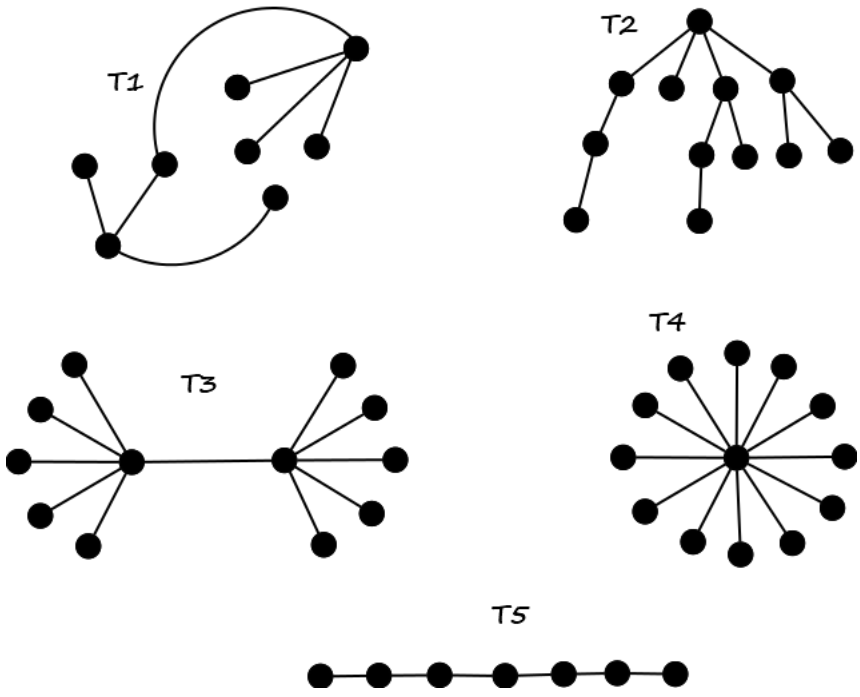
Como estructura abstracta, los árboles modelan jerarquía sobre los objetos, como los árboles genealógicos o estructuras de bases de datos. Además, hay muchos problemas *difíciles* sobre grafos que son *fáciles* cuando se trata de árboles. Por otro lado, hay muchas variantes de estructuras de datos eficientes basadas en árboles.

Además veremos un problema de optimización que involucra árboles generadores que aparece frecuentemente en diseño de redes de diferentes naturalezas.

1. Definición y propiedades

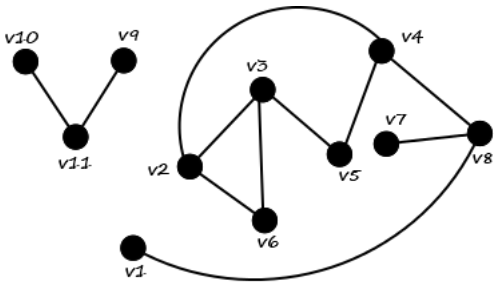
Definición 1. Un ***árbol*** es un grafo conexo sin circuitos simples.

Ejemplo 1.



Definición 2. Un ***puente*** es una arista de un grafo G si $G - e$ tiene más componentes conexas que G.

Ejemplo 2.



Puentes: (v_1, v_8), (v_4, v_8), (v_7, v_8), (v_9, v_{11}), (v_{10}, v_{11})

Lo primero que vamos a hacer es dar diferentes caraterizaciones de los árboles. Por un lado, esto nos permitirá conocer mejor a esta estructura, y por otro nos facilitará algunas demostraciones.

Teorema 1. *Dado un grafo $G = (V, X)$ son equivalentes:*

1. *G es un árbol.*
2. *G es un grafo sin circuitos simples y e una arista tal que $e \notin X$. $G + e = (V, X \cup \{e\})$ tiene exáctamente un circuito simple, y ese circuito contiene a e.*
3. *Existe exactamente un camino simple entre todo par de vértices.*
4. *G es conexo, pero si se quita cualquier arista a G queda un grafo no conexo (toda arista es puente).*

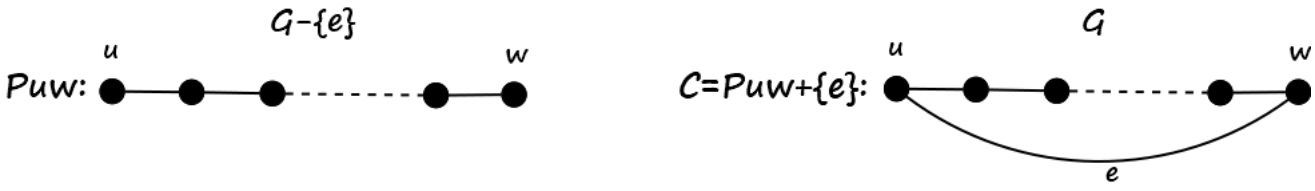
Para que la demostración nos quede más ordenada, primero vamos a enunciar dos lemas.

Lema 1 (Ejercicio 3.7 Práctica 3)**.** *La unión de dos caminos simples distintos entre dos vértices contiene un circuito simple.*

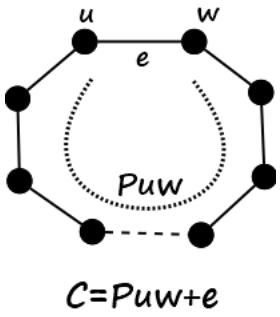
Lema 2. *Sea $G = (V, X)$ un grafo conexo y $e \in X$. $G - e = (V, X \setminus \{e\})$ es conexo $\iff$ e pertenece a un circuito simple de G.*

O equivalente: Una arista $e \in X$ es puente $\iff$ e no pertenece a un circuito simple de G.

Demostración Lema 2:
$\Longrightarrow$) $e = (u, w) \in X$ una arista cualquiera. Por hipótesis $G - e$ es conexo. Entonces existe un camino simple P_{uw} entre u y w en $G - e$ (que no usa a e). Luego, $C = P_{uw} + e$ es un circuito simple en G que contiene a e.



$\Longleftarrow$) Sea C un circuito simple de G que contiene a $e = (u, w)$. Podemos partir a C en la arista (u, w) y un camino simple entre u y w, P_{uw}. Como G es conexo, hay camino entre todo par de vértices. Si esos caminos no usan a e, siguen estando en $G - e$. Si un camino Q de G usa a e, en $G - e$ hay un camino alternativo que es cambiando a e por P_{uw} en Q. Entonces sigue habiendo camino entre todo par de vértices en $G - e$, es decir $G - e$ es conexo.



■

Demostración Teorema 1: Vamos a demostrar que $1 \implies 2$, $2 \implies 3$, $3 \implies 4$ y $4 \implies 1$.

$1 \implies 2$) G es árbol, es decir, conexo y sin circuitos. Sea $e = (u, w) \notin X$. Como G es conexo, existe un camino simple P_{uw} en G entre u y w. Entonces $P_{uw} + e$ es un circuito simple de $G + e$.

Ahora nos falta ver que no se puede haber generado más de un circuito simple. Vamos a hacerlo por el absurdo. Todo circuito simple de $G + e$ debe contener a e, ya que de lo contrario ese circuito estaría en G y G no tiene circuitos por ser árbol. Entonces supongamos que se generaron en $G + e$ dos circuitos simples que contienen a e, C y C'. Podemos partir estos circuitos en $C = P_{uw} + (u, w)$ y $C' = P'_{uw} + (u, w)$. P_{uw} y $P'_{u,w}$ son dos caminos simples distintos entre u y w en G. Por Lema 1, la unión de estos dos caminos contiene un circuito simple. Esto es absurso ya que G es árbol (no contiene circuitos).

$2 \implies 3$) Primero veamos que G es conexo, es decir que hay camino entre todo par de vértices, u y w. Si $(u, w) \in X$, el camino es la arista (u, w). Si u y w no son adyacentes G, por hipótesis $G + (u, w)$ contiene exactamente un cirtcuito simple C y (u, w) pertenece a ese circuito. Podemos partir ese circuito en $C = P_{u,w} + (u, w)$. $P_{u,w}$ es un camino simple entre u y w que está en G. Como u y w son dos vértices cualquiera, para todo par de vértices existe un camino entre ellos en G.

Falta ver que es único. Por el absursdo, supongamos que hay dos caminos distintos P y Q entre un par de vértices v y w. Por Lema 1, $P \cup Q$ tiene un circuito. Esto contradice 2).

$3 \implies 4$) Por 3) G es conexo. Por contradicción, supongamos que existe $e = (u, w) \in X$ tal que $G - e$ es conexo. Por Lema 2, e pertenece a un circuito simple C de G. Entonces podemos partir a $C = P_{uw} + (u, w)$. P_{uw} y (u, w) son dos caminos simples distintos entre u y w, contradiciendo 3).

$4 \implies 1$) Por 4) G es conexo. Por contradicción, supongamos que G tiene un circuito C. Sea $e = (u, w)$ una arista cualquiera de C. Por Lema 2, $G - e$ es conexo, contradiciendo 4). Luego G es conexo y sin circuitos, es decir, G es árbol.

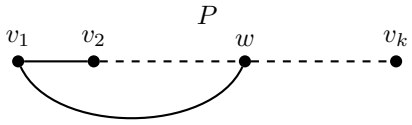
■

Definición 3. Una **hoja** es un vértice de grado 1.

Lema 3. *Todo árbol no trivial (de al menos dos vértcies) tiene al menos dos hojas.*

Demostración: Sea $P : v_1 \dots v_k$ un camino simple maximal (no extendible por sus extremos) en el árbol T. Esto es que todos los adyacentes de v_1 y todos los de v_k pertenecen al camino.

Veamos que v_1 y v_k son hojas. Por el absurdo supongamos que no, que $d(v_1) > 1$. Entonces, v_1 tiene otro adyacente w además de v_2, $w \neq v_2$, $w \in P$. Luego, $P_{v_1 w} + (w, v_1)$ es un circuito, contradiciendo que T es árbol. Por lo tanto, $d(v_1) = 1$. Lo mismo para v_k.



■

Lema 4. *Sea $G = (V, X)$ árbol. Entonces $m = n - 1$.*

Demostración: Haremos inducción en la cantidad de vértices.

Caso base: Para el árbol trivial, se cumple porque $n = 1$ y $m = 0$.

Paso inductivo: Para demostrar el paso inductivo, consideremos $T = (V, X)$ un árbol T con k vértices ($k > 1$).

Nuestra hipótesis inductiva es: ***Todo árbol T' de k' vértices ($k' < k$) tiene $k' - 1$ aristas***.

Por Lema 2, T tiene al menos una hoja u. Definimos $T' = (V \setminus \{u\}, X \setminus \{(u, v) \in X, \forall v \in V\}) = T - u$. T' es conexo (porque $d(u) = 1$) y no tiene circuitos, ya que T no los tiene. Entonces, T' es un árbol de $k - 1$ vértices. Por hipótesis inductiva, T' tiene $k-2$ aristas. Como $d(v) = 1$, T tiene una arista más que T'. Luego, T tiene $k - 1$ aristas.

■

Definición 4. Un ***bosque*** es un grafo sin circuitos simples.

Corolario 1. *Sea $G = (V, X)$ un bosque con c componentes conexas. Entonces $m = n - c$.*

Demostración: Para $i = 1, \dots, c$, sea n_i la cantidad de vértices de la componente i y m_i la cantidad de aristas. Como cada componente conexa de G es un árbol, podemos aplicar el lema anterior a cada una de ellas. Entonces, $m_i = n_i - 1$ para $i = 1, \dots, c$. Sumando, obtenemos que $m = \sum_{i=1}^{c} m_i = \sum_{i=1}^{c} (n_i - 1) = n - c$. ■

Corolario 2. *Sea $G = (V, X)$ con c componentes conexas. Entonces $m \geq n - c$.*

Demostración: Si G tiene circuitos, removerlos sacando una arista por vez hasta que grafo resultante $\hat{G}$ sea sin circuitos. Por Corolario 1 $\hat{m} = n - \hat{c} = n - c$. Como $m \geq \hat{m}$, $m \geq n - c$. ■

Teorema 2. *Dado un grafo G son equivalentes:*

1. *G es un árbol.*
2. *G es un grafo sin circuitos simples y $m = n - 1$.*
3. *G es conexo y $m = n - 1$.*

Demostración:

$1 \implies 2)$ Como G es árbol, G no tiene circuitos y $m = n - 1$ por Lema 4.

$2 \implies 3)$ Sea c la cantidad de componentes conexas de G. Por Corolario 1, $n - 1 = m = n - c$. Luego $c = 1$, es decir, G es conexo.

$3 \implies 1)$ Por contradicción, supongamos que G tiene un circuito simple. Sea $e = (v, w)$ una arista del circuito. Entonces, por Lema 1, $G - e$ es conexo y $m(G - e) = n - 2$, contradiciendo el Corolario 2. Luego, G no puede tener circuitos simples. G es árbol.

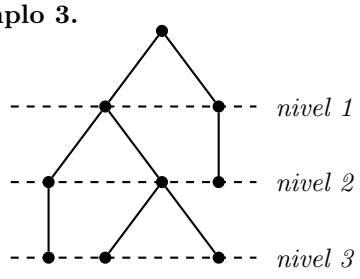
■

2. Árboles enraizados

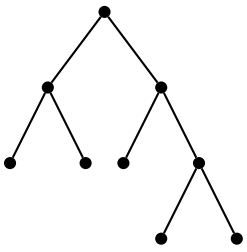
Definición 5.

- Un ***árbol enraizado*** es un árbol que tiene un vértice distinguido que llamamos ***raíz***. Explícitamente queda definido un árbol dirigido, considerando caminos orientados desde la raíz al resto de los vértices.
- Los vértices ***internos*** de un árbol son aquellos que no son ni hojas ni la raíz.
- El ***nivel*** de un vértice de un árbol con raíz es la distancia de la raíz a ese vértice.
- Decimos que dos vértices adyacentes tienen ***relación padre-hijo***, siendo el padre el vértice de menor nivel.
- La ***altura*** h de un árbol con raíz es la distancia desde la raíz al vértice más lejano.
- Un árbol se dice (exactamente) ***m-ario*** si todos sus vértices, salvo las hojas y la raíz tienen grado (exactamente) a lo sumo $m + 1$ y la raíz (exactamente) a lo sumo m.
- Un árbol se dice ***balanceado*** si todas sus hojas están a nivel h o $h - 1$.
- Un árbol se dice ***balanceado completo*** si todas sus hojas están a nivel h.

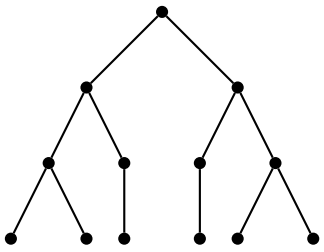
Ejemplo 3.



altura 3
2-ario
no exáctamente 2-ario



exáctamente 2-ario
balanceado
no balanceado completo



2-ario
balanceado completo

Ejemplo 4. *¿Cuántos vértices tiene en total un árbol exáctamente m-ario que tiene i vértices internos?*

Todo vértice, menos la raíz es hijo de alguien, hay m que son hijos de la raíz y el resto tiene que ser hijo de algún vértice interno:

$$n = 1 \text{ (raíz) } + m \text{ (hijos de la raíz) } + i * m \text{ (cada vértice interno tiene } m \text{ hijos) } = m(i+1)+1$$

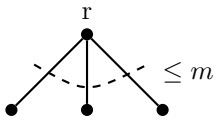
Teorema 3.

1. *Un árbol m-ario de altura h tiene a lo sumo m^h hojas.*
2. *Un árbol m-ario con l hojas tiene $h \geq \lceil \log_m l \rceil$.*
3. *Si T es un árbol exáctamente m-ario balanceado completo entonces $h = \lceil \log_m l \rceil$.*

Demostración:

1. Haremos inducción en la altura del árbol. Para el árbol trivial se cumple, ya que no tiene hojas. Éste prodría ser nuestro caso base, pero me parece más claro que sea $h = 1$.

   Caso base: $h = 1$

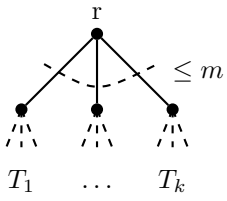
   

   # de hojas $\leq m = m^1$

   Paso inductivo: Para demostrar el paso inductivo, consideremos $T = (V, X)$ un árbol m-ario de altura $h > 1$.

   Nuestra hipótesis inductiva es: ***Todo árbol m-ario de altura $h' < h$, tiene a lo sumo $m^{h'}$ hojas.***

   Sea r la raíz de T y $d(r) = k \leq m$ (porque T es m-ario). Sean $T_1, \ldots, T_k$ las k componentes conexas de $T - r$.

   Las hojas de T son hojas de los T_i si $h_i \geq 1$, o raíces de los T_i con $h_i = 0$. Luego

   $$\#\text{ de hojas de } T = \sum_{\substack{i=1 \\ h_i \geq 1}}^{k} \#\text{ de hojas de } T_i + \sum_{\substack{i=1 \\ h_i = 0}}^{k} 1$$

   Cada T_i, $i = 1, \ldots, k$, es árbol m-ario de altura $h_i < h$. Por lo tanto, cada T_i cumple las hipótesis de la H.I.

Aplicando H.I. a cada T_i, # de hojas de $T_i \leq m^{h_i} \leq m^{h-1}$.

$$\#\text{ de hojas de } T \leq \sum_{i=1}^{k} m^{h_i} \leq \sum_{i=1}^{k} m^{h-1} = k * m^{h-1} \leq m * m^{h-1} = m^h.$$

2. T árbol m-ario con l hojas. Por punto anterior:

$$l \leq m^h \implies \log_m l \leq h \implies \text{ (por ser h entero) } \lceil \log_m l \rceil \leq h.$$

3. Si T es un árbol exáctamente m-ario balanceado completo, los $T_1, \ldots, T_m$ del paso inductivo del punto 1 son también árboles exáctamente m-arios balanceados completos (por lo que cumplen las hipótesis de la hipótesis inductiva) y los pasos son todas igualdades. Entonces,

$$l = m^h \implies \log_m l = h \implies \lceil \log_m l \rceil = h.$$

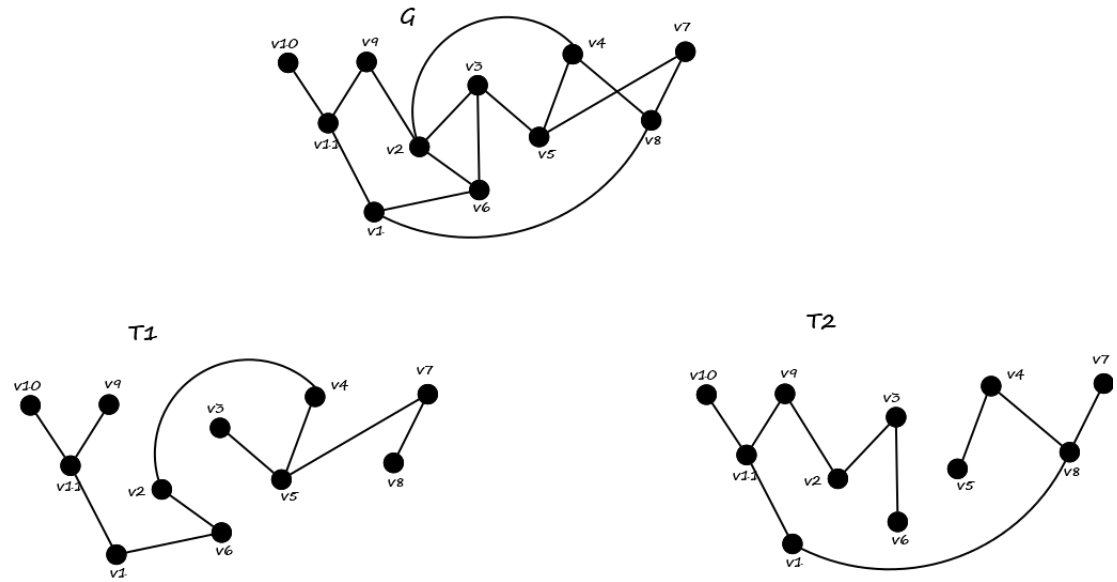
■

3. Árboles generadores

En muchas aplicaciones se quiere conectar n puntos (ciudades, centrales, servidores) mediante una red. Hay ciertas alternativas de enlaces posibles que se pueden construir, representadas mediante un grafo. Lo primero que tenemos que requerir es que este grafo sea conexo, de lo contrario no vamos a poder conectar todas las ciudades. Si queremos diseñar la red con la menor cantidad de enlaces posibles, vamos a necesitar elegir $n-1$ conexiones de las posibles, formando un árbol. Este subgrafo se llama árbol generador. Dependiendo de la aplicación, nos puede interesar árboles generadores con ciertas características, como por ejemplo que minimice la máxima distancia entre pares de vértices.

Definición 6. Un ***árbol generador*** (AG) de un grafo G es un subgrafo generador (que tiene el mismo conjunto de vértices) de G que es árbol.

Ejemplo 5.



T_1 y T_2 son árboles generadores de G

Teorema 4.

1. *Todo grafo conexo tiene (al menos) un árbol generador.*
2. *G conexo. G tiene un único árbol generador $\iff$ G es árbol.*
3. *Sea $T = (V, X_T)$ un AG de $G = (V, X)$ y $e \in X \setminus X_T$. Entonces $T + e - f = (V, X_T \cup \{e\} \setminus \{f\})$, con f una arista del único circuito de $G + e$, es árbol generador de G.*

Demostración.

1. Dado un grafo $G = (V, X)$ conexo, mediante el siguiente procedimiento vamos a construir un AG de G.

   a) Definimos $T = (V, X_T)$ con $X_T = X$.
   b) Mientras T tenga algún circuito:
      1) Sea $e \in X_T$ una arista de un cirtcuito de T.
      2) Definimos el nuevo T como $T = T - e$.
   c) Fin mientras.

   Como todas las aristas removidas no son puente, al finalizar este procedimiento T es un subgrafo generador conexo (por Lema 2) de G y todas sus aristas son puente. Por Teorema 1, T es árbol.

2. Ejercicio de la práctica.

3. Por hipótesis T es AG de G. Entonces T es conexo y tiene $n - 1$ aristas, donde $n = |V|$.

   Por Teorema 1, sabemos que $T + e$ tiene un único circuito, C, y e pertenece a ese circuito. Sea f una arista de C. Por Lema 2, si se quita una arista de un circuito el grafo sigue siendo conexo, es decir, $T+e-f$ es conexo.

   Entonces, $T + e - f$ es subgrafo generador de G, conexo y con $n - 1$ aristas, lo que implica que $T + e - f$ es árbol generador de G.

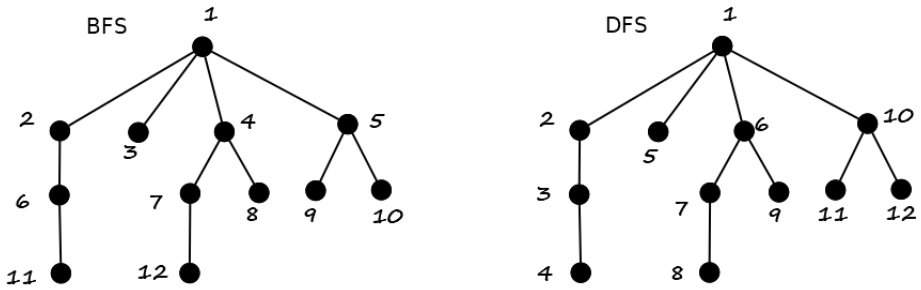
■

4. Recorrido de árboles o grafos

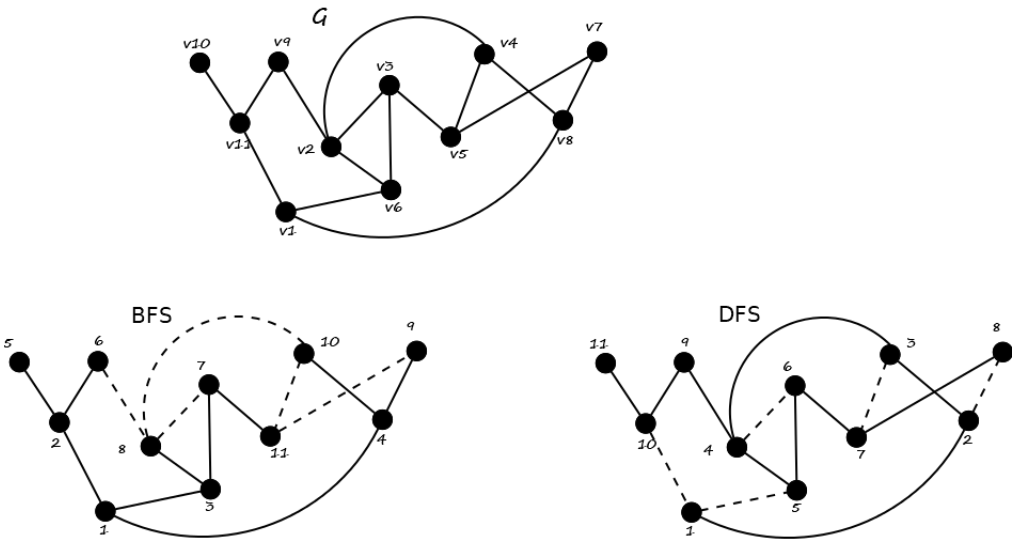
En muchas situaciones y algoritmos, dado un árbol (o grafo), queremos recorrer sus vértices exáctamente una vez. Para hacerlo de una forma ordenada y sistemática, podemos seguir un orden ***a lo ancho*** (Breadth-First Search **BFS**) o en ***profundidad*** (Depth-First Search **DFS**).

En **BFS** se comienza por el nivel 0 (la raíz) y se visita cada vértice en un nivel antes de pasar al siguiente. En **DFS** se comienza por la raíz y se explora cada rama lo más profundo posible antes de retroceder.

Ejemplo 6.


BFS
1
2
3
4
5
6
7
8
9
10
11
12
DFS
1
2
5
6
10
3
7
9
11
12
4
8

Ejemplo 7.


G
v10
v9
v3
v4
v7
v11
v2
v5
v8
v6
v1
BFS
5
6
10
9
7
2
8
11
4
3
1
DFS
11
9
3
8
6
10
4
7
2
5
1

```
recorrer(G)
    entrada: G = (V, X) de n vértices
    salida: pred[i] = padre de v_i - orden[i] = número asignado a v_i

    next ← 1
    r ← elegir un vértice como raíz
    (marcar vértice r)
    pred[r] ← 0
    orden[r] ← next
    LISTA ← {r}
    mientras LISTA ≠ ∅ hacer
        elegir un nodo i de LISTA
        si existe un arco (i, j) tal que j ∉ LISTA entonces
            (marcar vértice j)
            pred[j] ← i
            next ← next + 1
            orden[j] ← next
            LISTA ← LISTA ∪ {j}
        sino
            LISTA ← LISTA \ {i}
        fin si
    fin mientras
    retornar pred y orden
```

BFS y **DFS** difieren en el elegir:

- **BFS** (Breadth-First Search): $LISTA$ implementada como cola.
- **DFS** (Depth-First Search): $LISTA$ implementada como pila.

Cuando recorremos los vértices de un grafo G, los valores de *pred* implícitamente definen un AG de G. Estos dos procedimientos son la base de varios algoritmos, como por ejemplo para encontrar todas las componentes conexas de un grafo, los puntos de corte de un grafo conexo, determinar si un grafo tiene ciclos, en el problema de flujo máximo, camino mínimo de un grafo no pesado, encontrar los vértices que están a distancia menor que k de un vértice dado.

5. Árbol generador mínimo

Volvamos al problema de conectar n puntos. Supongamos además que cada posible enlace que se puede construir tiene un costo asociado (puede ser costo de contrucción, de mantenimiento, de utilización). Entonces, ahora nos interesa conectar los n puntos a costo mínimo. Esto se modela como un problema de optimización combinatoria, donde buscamos un árbol generador mínimo del grafo. Cuando el grafo tiene un costo asociado a las aritas o a los vértices, lo llamamos ***grafo pesado***.

Este problema fue planteado por Otakar Boruvka en 1926, en el contexto del diseño de la red eléctrica en el área rural de Moravia Meridional (República Checa). Boruvka desarrolló un método para encontrar una solución tan

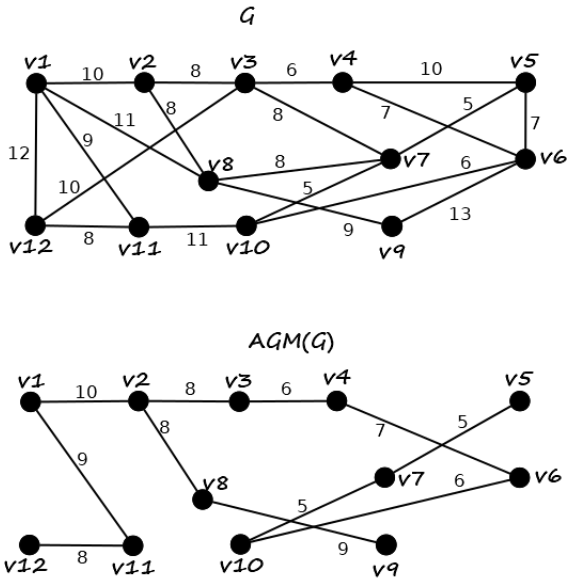
económica como fuera posible y lo publicó en dos artículos.

Formalmente:

Definición 7.

- Sea $T = (V, X)$ un árbol y $l : X \to \mathbb{R}$ una función que asigna costos a las aristas de T. Se define el ***costo*** de T como $l(T) = \sum_{e \in T} l(e)$.
- Dado un grafo $G = (V, X)$ un ***árbol generador mínimo*** de G, $AGM(G) = T$, es un árbol generador de G de mínimo costo, es decir
$$l(T) \leq l(T') \ \forall T' \text{ árbol generador de } G.$$
- Dado un grafo pesado en las aristas, $G = (V, X)$, el problema de ***árbol generador mínimo*** consiste en encontrar un AGM de G.

Ejemplo 8.



El problema de encontrar un árbol generador mínimo de un grafo está bien resuelto computacionalmente, es decir se conocen algoritmos polinomiales que lo resuelven. Los algoritmos más conocidos para este problema son el algoritmo de Prim (1957) y el de Kruskal (1956). Al igual que el método de Boruvka, son todos algoritmos golosos.

Algoritmo de Prim

El algoritmo de Prim construye incrementalmente dos conjuntos, uno de vértices V_T (inicializado con un vértice cualquiera) y otro de aristas X_T que comienza vacío. En cada iteración se agrega un elemento a cada uno de estos conjuntos. Cuando $V_T = V$ el algoritmo termina y las aritas de X_T definen un AGM de G.

En cada paso, se selecciona la arista de menor costo entre las que tienen un extremo en V_T y el otro en $V \setminus V_T$. Esta es una elección golosa: de un conjunto de aristas candidatas, elige la ***mejor*** arista (en este caso la de menor costo). Esta arista es agregada a X_T y el extremo a V_T.

Prim(G)
entrada: $G = (V, X)$ de n vértices y $l : X \to \mathbb{R}$
salida: $T =$ un AGM de G

$V_T \leftarrow \{u\}$ *(cualquier vértice)*
$X_T \leftarrow \emptyset$
$i \leftarrow 1$
mientras $i \leq n-1$ **hacer**
$e \leftarrow \arg\min\{l(e), e = (u, w), u \in V_T, w \in V \setminus V_T\}$
$X_T \leftarrow X_T \cup \{e\}$
$V_T \leftarrow V_T \cup \{w\}$
$i \leftarrow i + 1$
fin mientras
retornar $T = (V_T, X_T)$

Veamos que el algoritmo es correcto, es decir que retorna un AGM de G. Para ésto primero vamos a demostrar parte del invariante del ciclo. Notaremos por $T_k = (V_{Tk}, X_{Tk})$ al grafo que el algoritmo de Prim construyó al finalizar la iteración k, para $0 \leq k \leq n-1$. $T_0 = (V_0, X_0)$ se refiere a la inicialización antes de entrar a la primera iteración.

Proposición 1. *Dado $G = (V, X)$ un grafo conexo. $T_k = (V_{Tk}, X_{Tk})$, $0 \leq k \leq n-1$, es árbol y subgrafo de un árbol generador mínimo de G.*

Demostración: Haremos inducción en las iteraciones del ciclo, k.

Caso base: Antes de ingresar al ciclo, $k = 0$, $T_0 = (\{u\}, \emptyset)$ es árbol y subgrafo de todo AGM de G.

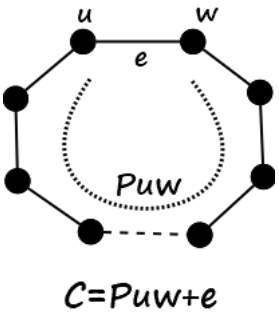
Paso inductivo: Para demostrar el paso inductivo, consideremos T_k, $k \geq 1$.

La hipótesis inductiva es: ***$T_{k'}$, $k' < k$, es árbol y subgrafo de algún $T = (V, X_T)$, T AGM de G.***

Llamemos w al vértice agregado en la iteración k y e a la arista. Es decir, $T_k = (V_k, X_k)$, $V_k = V_{k-1} \cup \{w\}$ y $X_k = X_{k-1} \cup \{(u, w) = e\}$, $u \in V_{k-1}$, $w \notin V_{k-1}$.
Por HI sabemos que T_{k-1} es árbol. Como T_k tiene un vértice y una arista más que T_{k-1} y es conexo, entonces T_k es árbol.

También sabemos, por HI, que existe $T = (V, X_T)$ AGM de G tal que T_{k-1} es subgrafo de T.

- Si $e \in X_T$, es decir si T contiene a e, entonces T_{k+1} también es subgrafo de T.
- Si $e \notin X_T$, entonces $T + e$ tiene un circuito simple, C, que contiene a e (por Teorema 1). Este circuito está formado por el único camino entre u y w que tiene T, P_{uw} más la arista e, $C = (u, w) + P_{uw}$.



Sea f una primera arista de P_{uw} que tiene un extremo en V_{k-1} y el otro no en V_{k-1} (existe porque $u \in V_{k-1}$ y $w \notin V_{k-1}$).

Definimos $T' = T + e - f = (V, X_T \cup \{e\} \setminus \{f\})$.

- T' es árbol generador de G (por Teorema 4).
- T_k es subgrafo de T'.
- T' es AGM de G: Como f es una arista elegible al comienzo de la iteración k, pero el algoritmo eligió a e, seguro se cumple que $l(f) \geq l(e)$. Entonces:

$$l(T') = l(T) + l(e) - l(f) \leq l(T).$$

Con esto demostramos que T_k es árbol y subgrafo de un AGM de G.

■

Teorema 5. *El algoritmo de Prim es correcto, es decir dado un grafo G conexo determina un árbol generador mínimo de G.*

Demostración: Al finalizar la iteración $n-1$, por Proposción 1, T_{n-1} es árbol y subgrafo de algún T AGM de G.

Además, T_{n-1} es subgrafo generador de G, ya que en cada iteración el algoritmo agrega un vértice distinto a V_T, y entonces $V_{n-1} = V$.

Entonces T_{n-1} es T, AGM de G. ■

Algoritmo de Kruskal

El segundo algoritmo que veremos fue desarrollado por J. B. Kruskal (1956). La idea de este algoritmo es ordenar las aristas del grafo de forma creciente según su peso y en cada paso elegir la siguiente arista que no forme circuito con las aristas ya elegidas. El algoritmo para cuando selecciona $n-1$ aristas (siendo n la cantidad de vértices del grafo). También es un algoritmo goloso.

```
Kruskal(G)
    entrada: G = (V, X) de n vértices y l : X → ℝ
    salida: T = un AGM de G

    X_T ← ∅
    i ← 1
    mientras i ≤ n − 1 hacer
        e ← arg mín{l(e), e no forma circuito con las aristas de X_T}
        X_T ← X_T ∪ {e}
        i ← i + 1
    fin mientras
    retornar T = (V, X_T)
```

Al comenzar el algoritmo, cuando todavía no se seleccionó arista alguna, cada vértice del grafo forma una componente conexa distinta (es un bosque de vértices aislados). En la primera iteración, los dos vértices extremo de la arista seleccionada van a pasar a formar una única componente conexa del nuevo bosque. Así, en cada iteración, si se elige la arista (u, w), se unen las componentes conexas de u y la de w. En cada iteración el bosque obtenido tiene una componente conexa menos que el anterior. El algoritmo termina cuando el bosque pasa a ser un árbol, es decir, conexo.

Con esta idea podemos reescribir el algoritmo de Kruskal:

```
Kruskal(G)
    entrada: G = (V, X) de n vértices y l : X → ℝ
    salida: T = un AGM de G

    X_T ← ∅
    Cand ← X
    ordenar(Cand)
    i ← 1
    mientras i ≤ n − 1 hacer
        (u, w) ← mín(Cand)
        Cand ← Cand \ {(u, w)}
        si u y w no pertenecen a la misma componente conexa de T = (V, X_T) hacer
            X_T ← X_T ∪ {(u, w)}
            i ← i + 1
        fin si
    fin mientras
    retornar T = (V, X_T)
```

La demostración de correctitud de Kruskal es muy similar a la de Prim. En cada iteración el grafo que arma el algoritmo es un bosque, ya que no contiene circuitos. Al finalizar la última iteración, como se incorporaron $n - 1$ aristas, pasa a ser árbol (sin circuitos y con $m - 1$ aristas). Ésto junto a la siguiente proposición demuestra la

correctitud de Kruskal.

Demostración. La demostración es muy similar a la de la Proposición 1.

Si $e = (w, v)$ es la arista incorporada en k-ésimo lugar, la única diferencia significativa es la definición de la arista f en la demostración del paso inductivo cuando $e \notin T_{k-1}$.

Llamamos G_u^{k-1} a la componente conexa de u antes de agregar la arista e. La arista f en la definición de $T' = T+e-f$ debe ser una arista que pertenzca al circuito $C = Pwu + e$ y que tenga un extremo en G_u^{k-1} y el otro extremo fuera de G_u^{k-1}. Esta arista seguro existe, porque u pertenece a esa componente y w no pertenece. Esta arista f es candidata a ser elegida por el algoritmo en lugar de e, por lo que $l(f) \geq l(e)$.

■

El algoritmo de Prim va armando subárboles de un AGM de G. En cambio, el algoritmo de Kruskal va armando bosques que son subgrafos de un AGM, pero no necesariamente conexos.

Las complejidades de estos algoritmos dependen fuertemente de las estructuras de datos utilizadas en su implementación. En general, para grafos esparsos (con pocas aristas) el algoritmo de Kruskal es más rápido que el de Prim.