Árbol generador: algoritmos

S. Bianchi P. Fekete F. Domingo

[1] Deptartamento de Matemática
Escuela de Ciencias Exactas y Naturales
UNR

29 de septiembre de 2021

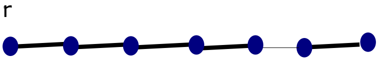
INTORODUCCIÓN

Sea $G=(V,E)$ un grafo conexo.

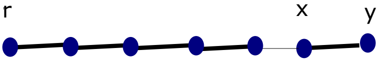
Sabemos existe $T=(V,E')$ árbol generador de G, es decir $E'\subset E$.

Cómo encontrar a T?

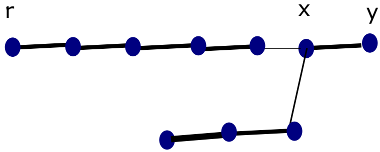
Sea $r\in V$. Construimos en G un camino simple P más largo posible desde r.
Ejemplo



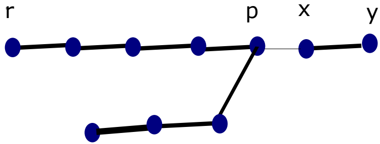
Si el camino incluye a todos los vértices en G, entonces es un árbol generador posible.
Sino, sean x e y los últimos vértices en el camino:



Retrocedemos al vértice x y construimos un segundo camino simple lo más largo posible desde x que no incluya vértices ya visitados:

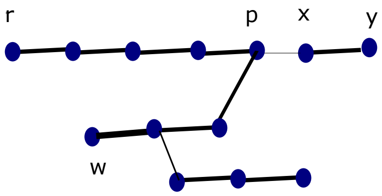


Si no existe tal camino, retrocedemos hacia el vértice p, padre de x en el camino y buscamos un camino simple lo más largo posible que incluya vértices no visitados.



Si todos los vértices de G están contemplados, se trata del árbol buscado.

Si aún el subgrafo no es generador, y w es último vértice alcanzado por el camino, retrocedemos hacia el padre de w y construimos un tercer camino simple lo más largo posible que no incluya vértices ya visitados:



Así continuamos el proceso hasta encontrar un árbol generador.

Esta técnica se llama **búsqueda en profundidad**.

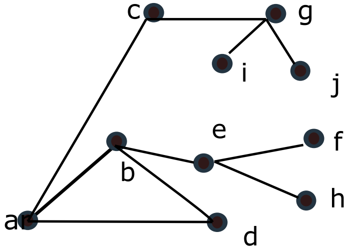
Sea $G = (V,E)$ grafo no dirigido y conexo, sin lazos y $|V| = n$.

Llamamos $v_1, v_2, \ldots, v_n$ a los vértices en V.

Algoritmo de búsqueda en profundidad

1. $v := v_1$, $V(T) = \{v\}$ y $E(T) = \emptyset$.
2. $i = min\{j : 2 \leq j \leq n,\ \{v, v_j\} \in E\ v_j \notin V(T)\}$.
    - Si no existe i, ir al Paso 3.
    - Si existe i, $V(T) := V(T) \cup \{v_i\}$, $E(T) := E(T) \cup \{v, v_i\}$, $v := v_i$ ir al Paso 2.
3.
    - Si $v = v_1$, entonces T es árbol generador.
    - Si $v \neq v_1$, sea u el padre de v en el árbol T, $v := u$ ir al Paso 2.

Veamos un ejemplo:

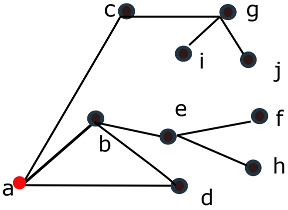


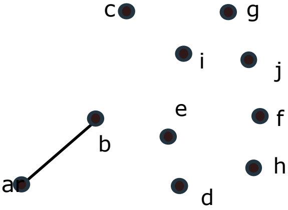
Aquí los vértices están ordenados alfabéticamente:

$$a, b, c, d, e, f, g, h, i, j.$$

Paso 1 Inicialización $v := a$ $V(T) = \{v\}$ y $E(T) = \emptyset$.

Paso 2 b es el primer vértice vecino de a que no ha sido visitado todavía.

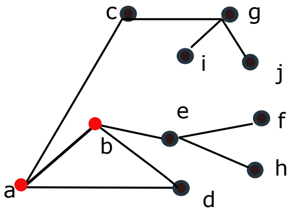


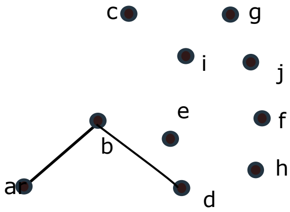


Agregamos $\{a, b\}$ a $E(T)$:

Asignamos $v := b$ y vamos al Paso 2.

Paso 2 $v = b$, el primer vértice vecino de b que no ha sido visitado todavía es d.

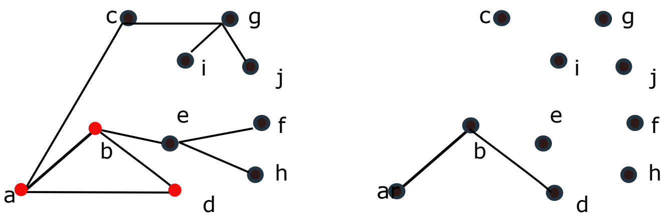




Agregamos $\{b,d\}$ a $E(T)$:

Asignamos $v := d$ y vamos al Paso 2.

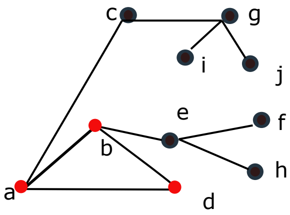
Paso 2 $v = d$, no existe vértice vecino de d que no haya sido visitado.

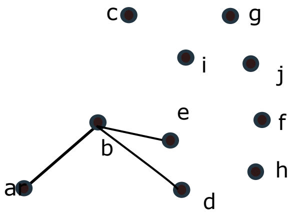


Paso 3 $v \neq a$, entonces retrocedemos al padre de d en el árbol T que es b.

Asignamos $v := b$ vamos al Paso 2.

Paso 2 $v = b$, existe vértice vecino de b que no ha sido visitado (e).



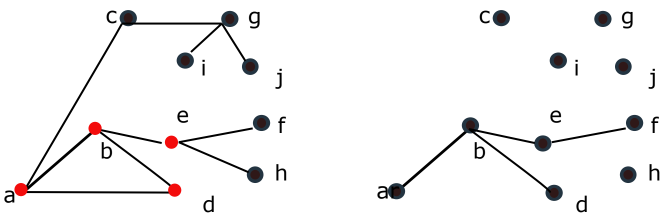


Agregamos $\{b, e\}$ a $E(T)$:

Asignamos $v := e$ y vamos al Paso 2.

Algoritmo de búsqueda en profundidad

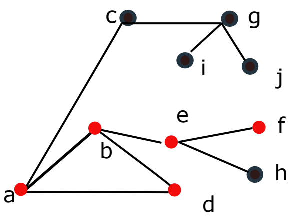
Paso 2 $v = e$, el primer vértice vecino de e que no ha sido visitado todavía es f.

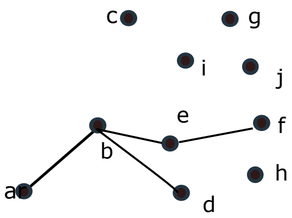


Agregamos $\{e,f\}$ a $E(T)$:

Asignamos $v := f$ y vamos al Paso 2.

Paso 2 $v = f$, no existe vértice vecino de f que no haya sido visitado.

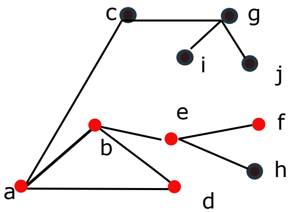


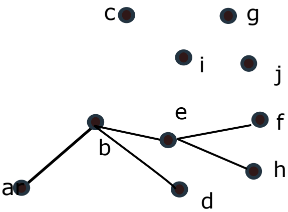


Paso 3 $v \neq a$, entonces retrocedemos al padre de f en el árbol T que es e.

Asignamos $v := e$ vamos al Paso 2.

Paso 2 $v = e$, existe vértice vecino de e que no ha sido visitado (h).

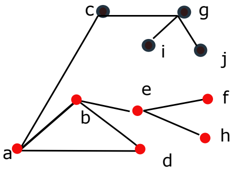


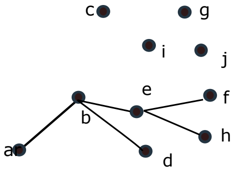


Agregamos $\{e, h\}$ a $E(T)$:

Asignamos $v := h$ y vamos al Paso 2.

Veamos:

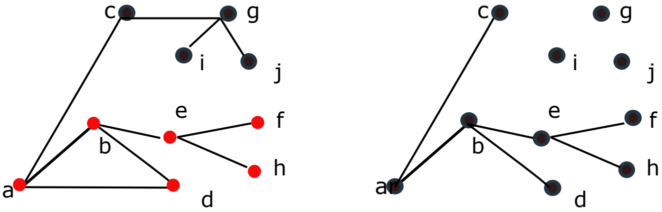




Paso 2 $v = h$, no existe vértice vecino de h que no haya sido visitado.
Paso 3 $v \neq a$, entonces retrocedemos al padre de h en el árbol T que es e.
Asignamos $v := e$ vamos al Paso 2.
Paso 2 $v = e$, no existe vértice vecino de e que no haya sido visitado .
Paso 3 $v \neq a$, entonces retrocedemos al padre de e en el árbol T que es b.
Asignamos $v := b$ vamos al Paso 2.
Paso 2 $v = b$, no existe vértice vecino de b que no haya sido visitado .
Paso 3 $v \neq a$, entonces retrocedemos al padre de b en el árbol T que es a.
Asignamos $v := a$ vamos al Paso 2.

ALGORITMO DE BÚSQUEDA EN PROFUNDIDAD

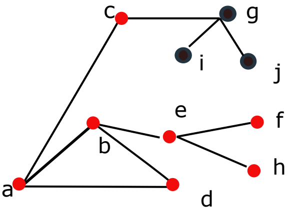
Paso 2 $v = a$, el primer vértice vecino de a que no ha sido visitado todavía es c.

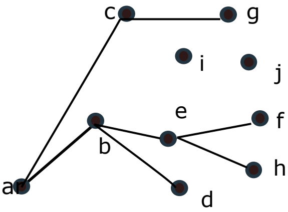


Agregamos $\{a, c\}$ a $E(T)$:

Asignamos $v := c$ y vamos al Paso 2.

Paso 2 $v = c$, el primer vértice vecino de c que no ha sido visitado todavía es g.

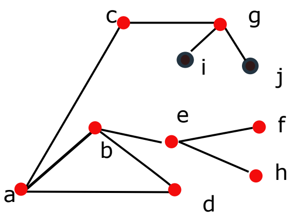


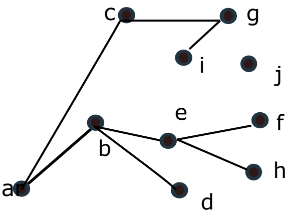


Agregamos $\{c, g\}$ a $E(T)$:

Asignamos $v := g$ y vamos al Paso 2.

Paso 2 $v = g$, el primer vértice vecino de g que no ha sido visitado todavía es i.

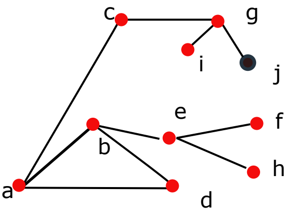


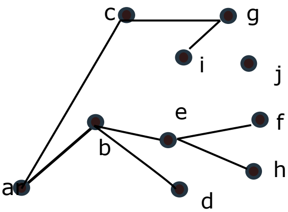


Agregamos $\{g, i\}$ a $E(T)$:

Asignamos $v := i$ y vamos al Paso 2.

Paso 2 $v = i$, no existe vértice vecino de i que no haya sido visitado.

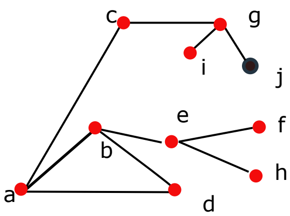


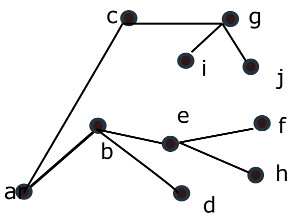


Paso 3 $v \neq a$, entonces retrocedemos al padre de i en el árbol T que es g.

Asignamos $v := g$ vamos al Paso 2.

Paso 2 $v = g$, existe vértice vecino de g que no ha sido visitado (j).

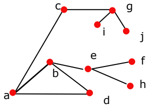


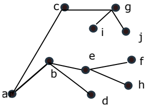


Agregamos $\{g, j\}$ a $E(T)$:

Asignamos $v := j$ y vamos al Paso 2.

Tenemos:





Paso 2 $v = j$, no existe vértice vecino de j que no haya sido visitado.

Paso 3 $v \neq a$, entonces retrocedemos al padre de j en el árbol T que es g.

Asignamos $v := g$ vamos al Paso 2.

Paso 2 $v = g$, no existe vértice vecino de g que no haya sido visitado .

Paso 3 $v \neq a$, entonces retrocedemos al padre de g en el árbol T que es c.

Asignamos $v := c$ vamos al Paso 2.

Paso 2 $v = c$, no existe vértice vecino de c que no haya sido visitado .

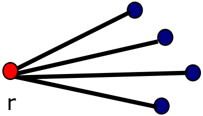
Paso 3 $v \neq a$, entonces retrocedemos al padre de c en el árbol T que es a.

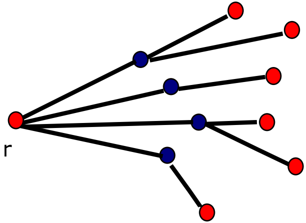
Asignamos $v := a$ vamos al Paso 2.

Paso 2 $v = a$, no existe vértice vecino de a que no haya sido visitado .

Paso 3 $v = a$, entonces T es un árbol generador.

Dado $G = (V, E)$ conexo, no dirigido y sin lazos, otro método de búsqueda de árbol generador, consiste en elegir un vértice raíz y recorrer sus adyacentes:





Desde cada hijo, recorremos vértices adyacentes no visitados.
Continuando con este proceso, no visitamos un vértice dos veces, de modo que no generamos un ciclo, como V es finito el proceso termina.
Esta técnica se la conoce como **búsqueda en ancho**

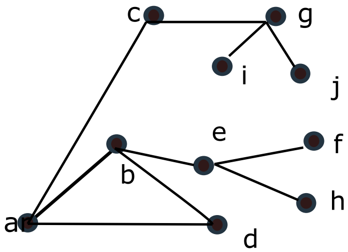
Sea $G=(V,E)$ grafo no dirigido y conexo, sin lazos y $|V|=n$.
Llamamos $v_1,v_2,\ldots,v_n$ a los vértices en V.

En este algoritmo será de utilidad una estructura de datos llamada *cola*.
Una cola es una lista ordenada en la que los elementos se insertan en el extremo final de la lista y se eliminan por el extremo inicial.
Sea Q una cola vacía.

Algoritmo de búsqueda en ancho

1. insertar v_1 en Q, $V(T)=\{v_1\}$ y $E(T)=\emptyset$.
2. eliminar los vértices al comienzo de Q. Al eliminar v en Q:
   - si la arista $\{v,v_i\}\in E$ y v_i no ha sido visitado para $i=2,\ldots,n$ entonces $E(T):=E(T)\cup\{v,v_i\}$. Ir al Paso 3.
   - caso contrario todos los vértices que se eliminan no generan nuevas aristas y el árbol T es un árbol generador.
3. insertar los vértices adyacentes a cada v del Paso 2 según el orden en el que fueron visitados. Ir al Paso 2.

Vamos a hacer un ejemplo con el mismo grafo que utilizamos anteriormente:

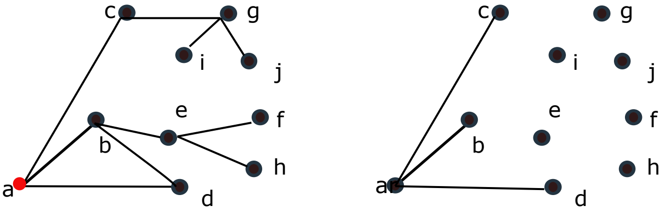


Recordemos que los vértices están ordenados alfabéticamente:

$$a, b, c, d, e, f, g, h, i, j.$$

Paso 1 Inicialización: insertamos a en Q, $V(T) = \{v\}$ y $E(T) = \emptyset$.

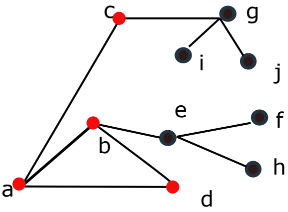
Paso 2 Eliminar a de Q, visitamos los vértices adyacentes: b, c, d.

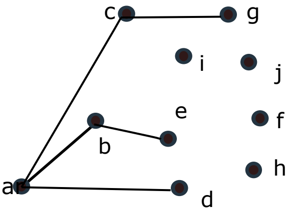


Como no fueron visitados previamente, entonces agregamos $\{a,b\}, \{a,c\}, \{a,d\}$ a $E(T)$.

Paso 3 Insertar b, c, d en Q, en ese orden e ir al Paso 2.

Paso 2 Eliminar b, c, d de Q, visitamos los vértices adyacentes no visitados: e, g.



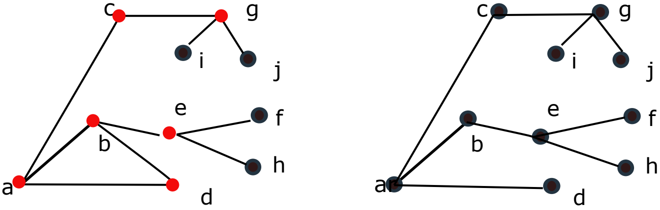


Agregamos $\{b, e\}, \{c, g\}$ a $E(T)$:

Paso 3 Insertar e, g en Q, en ese orden e ir al Paso 2.

ALGORITMO DE BÚSQUEDA EN ANCHO

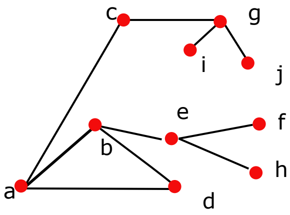
Paso 2 Eliminar e, g de Q, visitamos los vértices adyacentes no visitados: f, h, i, j.

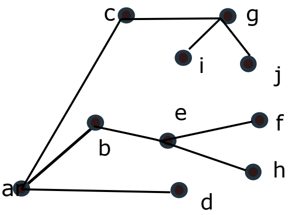


Agregamos $\{e,f\}, \{e,h\}, \{g,i\}, \{g,j\}$ a $E(T)$:

Paso 3 Insertar f, h, i, j en Q, en ese orden e ir al Paso 2.

Paso 2 Eliminar f, h, i, j de Q, no encontramos vértices no visitados





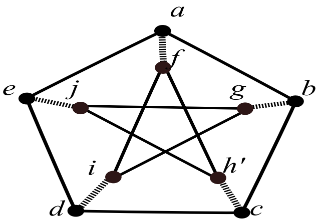
El árbol T es el árbol generador buscado.

GRAFO DE PETERSEN

Determinar el árbol generador que se consigue utilizando:

A) el algoritmo en profundidad,

B) el algoritmo en ancho.

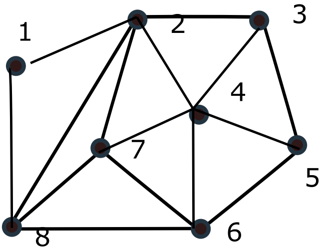


EJERCICIO 1

Determinar el árbol generador que se consigue utilizando:

A) el algoritmo en profundidad,

B) el algoritmo en ancho.

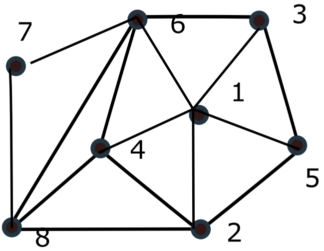


EJERCICIO 2

Determinar el árbol generador que se consigue utilizando:

A) el algoritmo en profundidad,

B) el algoritmo en ancho.



Responder las siguientes preguntas

Dado $G = (V, E)$ conexo:

1. Porqué puede asegurar que tanto el algoritmo en profundidad como el algoritmo en ancho terminan?
2. Porqué puede asegurar que tanto el algoritmo en profundidad como el algoritmo en ancho alcanzan a todos los vértices del grafo G?
3. Porqué puede asegurar que tanto el algoritmo en profundidad como el algoritmo en ancho no generan ciclos en el grafo G?