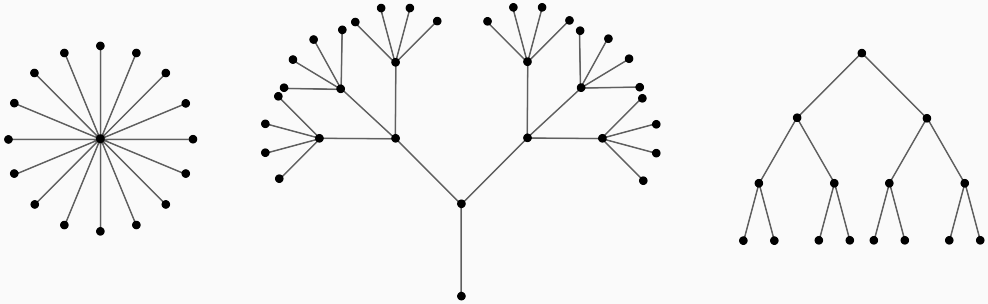
Matemática Discreta

Clase 24: Árboles generadores

Federico Olmedo y Alejandro Hevia

Departamento de Ciencias de la Computación
Universidad de Chile

Ejemplos de árboles



Caracterizaciones de árboles

Teorema

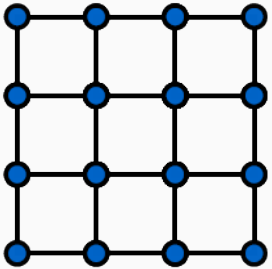
Sea $G = (V, E)$ un grafo no-dirigido. Las siguientes enunciados son equivalentes (y definen la noción de árbol):

1. G es conexo y no contiene ciclos simples;
2. G contiene un único camino simple entre cada par de nodos distintos;
3. G es conexo y $|E| = |V| - 1$;
4. G es conexo, pero quitar cualquiera de sus aristas lo desconecta;

Árboles generadores

Pavimentando caminos

Considere la siguiente red vial entre distintas ciudades, donde todos los caminos son de tierra.

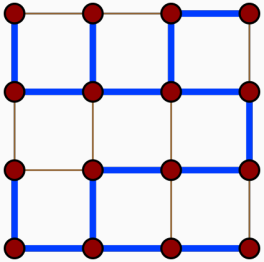


Asuma que se quiere pavimentar la mínima cantidad de caminos de tal manera que siempre se puede ir de una ciudad a otra transitando sólo caminos pavimentados.

Pregunta: ¿Cuál es la mínima cantidad de caminos que deben pavimentarse?

Pavimentando caminos

La respuesta viene dada por (el número de aristas de) un subgrafo que contenga a todos los vértices y que sea árbol.



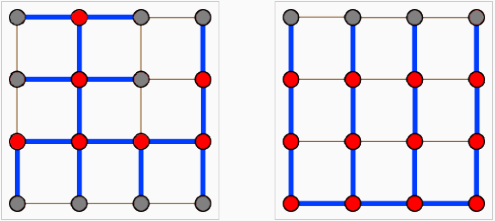
Esto se conoce como un árbol generador, árbol cobertor, o spanning tree del grafo.

Árboles generadores

Definición

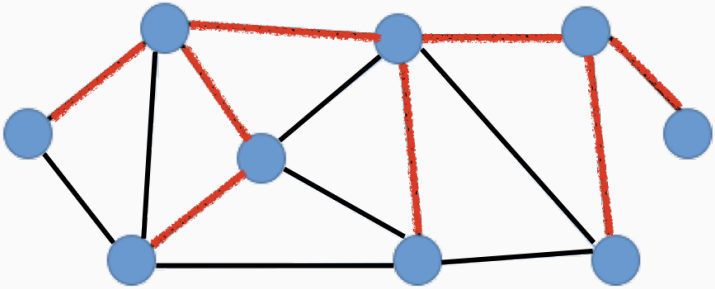
Un árbol generador de un grafo G es un subgrafo de G que contiene a todos sus nodos y que es un árbol.

Ejemplo: Dos árboles generadores distintos del mismo grafo.



Árboles generadores

Ejemplo:



Grafos que admiten árboles generadores

Pregunta: ¿Cómo podemos caracterizar los grafos que admites árboles generadores?

- Si un grafo admite un árbol generador, es trivialmente conexo.
- Y recíprocamente, ¿todo grafo conexo admite un árbol generador? Sí!

Teorema

Un grafo contiene un árbol generador si y sólo si es conexo.

Grafos que admiten árboles generadores

Teorema

Un grafo contiene un árbol generador si y sólo si es conexo.

Demostración

$\boxed{\Rightarrow}$ Trivial.

$\boxed{\Leftarrow}$ Sea G un grafo conexo. Si G es un árbol el resultado es inmediato. Si G no es un árbol, necesariamente tiene que tener un ciclo simple. Considero el subgrafo G' de G que se forma removiendo una arista del ciclo simple. Es fácil ver que el subgrafo G' es conexo (¿por qué?) y contiene todos los vértices de G. De nuevo, si G' es un árbol, el resultado es inmediato. Sino, vuelvo a repetir el proceso anterior (construyo un subgrafo G'' ...). Este proceso puede repetirse sólo un número finito de veces ya que en cada transformación elimino una arista y el número de aristas de G es finito. Por definición, el proceso se detiene cuando construye un árbol, que además es subgrafo de G y contiene todos sus vértices (estas dos propiedades son un "invariante" de la transformación que se lleva a cabo en cada paso). □

Construyendo árboles generadores

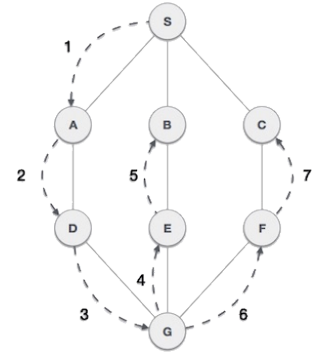
La prueba anterior provee un algoritmo para construir árboles generadores. Sin embargo dicho algoritmo no es muy eficiente ya que require identificar ciclos simples de un grafo.

Existen dos algoritmos más eficientes usados en la práctica:

- DFS: *depth first search* o búsqueda en profundidad
- BFS: *breath first search* o búsqueda a lo ancho

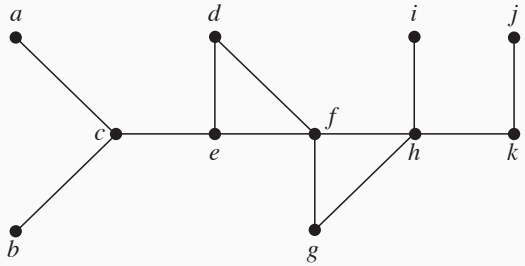
Búsqueda en profundidad (DFS)

Dado un grafo no-dirigido simple G, para encontrar un árbol generador el algoritmo de búsqueda en profundidad parte de un vértice y trata de construir el camino "lineal" más largo posible sin repetir vértices ya visitados antes de hacer un *backtracking* hacia los nodos anteriormente visitados (en orden inverso de visita).



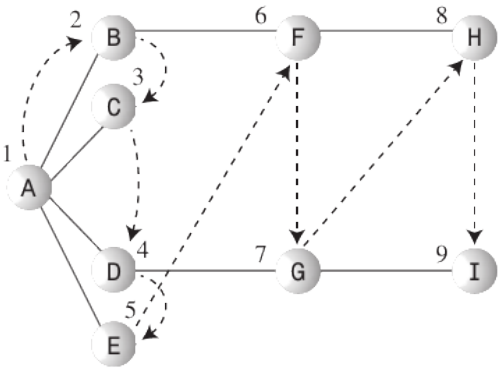
Búsqueda en profundidad (DFS)

Ejercicio: Aplique el algoritmo DFS para construir un árbol generador del siguiente grafo:



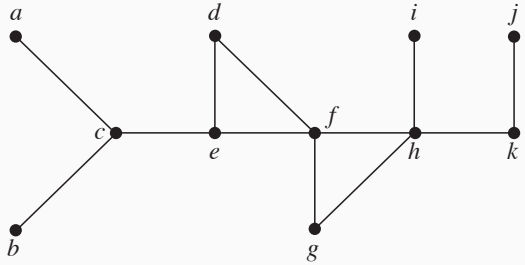
Búsqueda a lo ancho (BFS)

Dado un grafo no-dirigido simple G, para encontrar un árbol generador el algoritmo de búsqueda a lo ancho parte de un vértice y explora todos los nodos en un nivel, antes de pasar al siguiente nivel.



Búsqueda a lo ancho (BFS)

Ejercicio: Aplique el algoritmo BFS para construir un árbol generador del siguiente grafo:



Ejercicios

Ejercicio*: Describa los árboles generadores producidos por los algoritmos DFS y BFS en una rueda W_n, comenzado por el vértice central.

Ejercicio*: Describa los árboles generadores producidos por los algoritmos DFS y BFS en un grafo complete K_n.

Ejercicios

El siguiente ejercicio muestra que es posible encontrar una secuencia de árboles generadores que lleva de un árbol generador a cualquier otro árbol generador mediante la sucesiva inserción y eliminación de arcos.

Ejercicio: Sean T y T' dos árboles generadores distintos del mismo grafo simple G. Asuma que existe arco e en T que no pertenece a T'. Demuestre que existe un arco e' en T' tal que e' no está en T, y tal que T continúa siendo un árbol generador si le sacamos a e y le agregamos a e', y T' continúa siendo un árbol generador si le sacamos a e' y le agregamos a e.

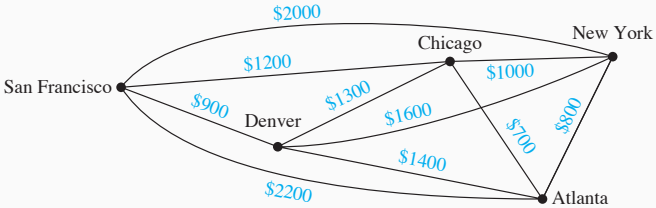
Árboles generadores de costo mínimo

Árboles generadores de costo mínimo

Si un grafo G ahora tiene costos asociados a cada arco, entonces un problema interesante es encontrar un árbol generador de peso mínimo.

Definición

Dado un grafo simple G, el árbol generador de peso mínimo es un árbol generador de G tal que la suma de los pesos de los arcos del árbol es mínima.



Hay dos algoritmos relevantes para encontrar un árbol de costo mínimo, el de **Prim** y el de **Kruskal**.

Algoritmo de Prim

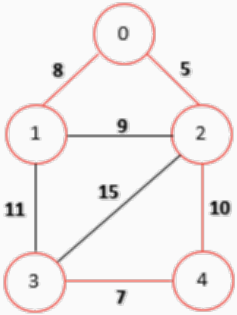
Algoritmo de Prim

Dado un grafo simple $G = (V, E)$, $n = |V|$, calcular el árbol de costo mínimo como sigue:

1. Escoger el arco de *menor peso* de G e incluirlo en el árbol.
2. Escoger el arco de menor peso de G incidente a un vértice existente del árbol, siempre que no genere un circuito simple.
3. Repetir el paso anterior hasta incluir $n - 1$ arcos en el árbol.

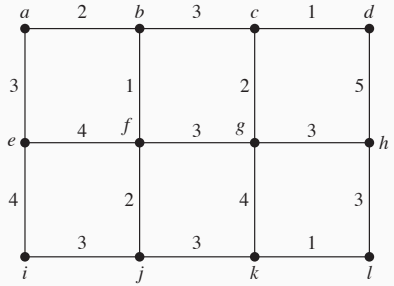
Algoritmo de Prim

Ejemplo:



Algoritmo de Prim

Ejercicio: Aplique el algoritmo de Prim al siguiente grafo.



Algoritmo de Kruskal

La idea del algoritmo de Kruskal es que no es necesario ir formando un árbol desde el comienzo.

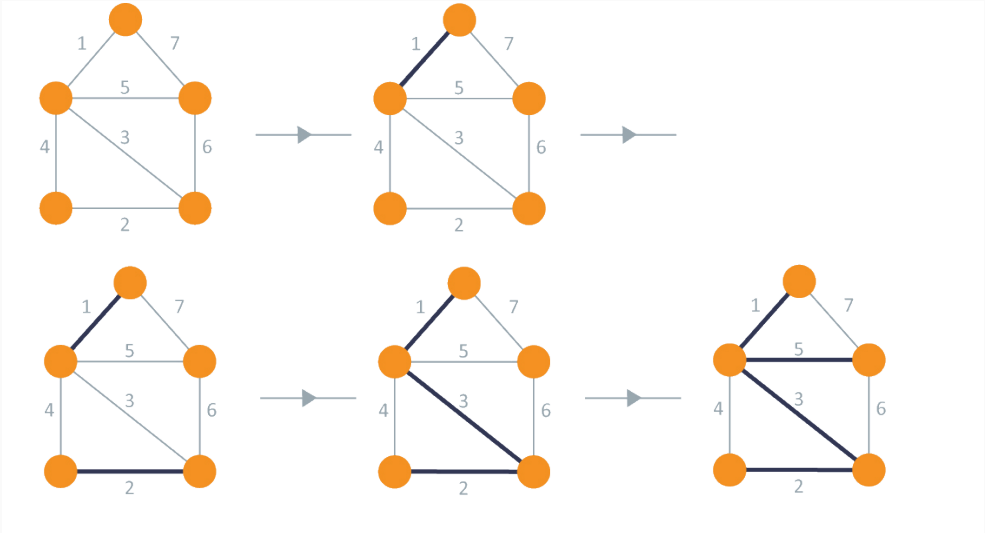
Algoritmo de Kruskal

Dado un grafo simple $G = (V, E)$, $n = |V|$, calcular el árbol de costo mínimo como sigue:

1. Escoger el arco de *menor peso* de G e incluirlo en el árbol.
2. Escoger el arco de menor peso de G (no necesariamente *incidente* a un vértice existente del árbol), siempre que no genere un circuito simple en algún árbol existente.
3. Repetir el paso anterior hasta incluir $n - 1$ arcos en el árbol.

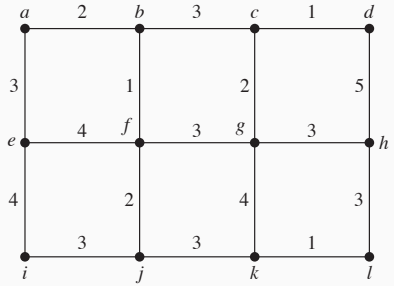
Algoritmo de Kruskal

Ejemplo:



Algoritmo de Kruskal

Ejemplo: Aplique el algoritmo de Kruskal al siguiente grafo.



Correctitud de Prim y Kruskal

Teorema (correctitud algoritmo Prim)

Dado un grafo simple $G = (V, E)$ de $n = |V|$ vértices y $e = |E|$ arcos, el algoritmo de Prim encuentra un árbol generador de peso mínimo.

Teorema (correctitud algoritmo Kruskal)

Dado un grafo simple $G = (V, E)$ de $n = |V|$ vértices y $e = |E|$ arcos, el algoritmo de Kruskal encuentra un árbol generador de peso mínimo.

Ejercicio: Demuestre ambos teoremas.

Costo: El algoritmo de Prim toma $\mathcal{O}(e \log(n))$ pasos y el de Kruskal toma $\mathcal{O}(e \log(e))$ pasos.

¿Cuál es mejor?