4.4. Árboles generadores

Definición. Sea G un grafo simple. Un *árbol generador* de G es un subgrafo T de G tal que:

- T contiene todos los vértices de G;
- T es un árbol.

Teorema. Sea G un grafo simple. Entonces G tiene un árbol generador si y sólo si G es conexo.

Lema. Sea $G = (V, E)$ un grafo simple, $U \subseteq V$ y (U, T) subgrafo árbol de G. Si $e = uv$ conecta $u \in U$ y $v \notin U$, entonces $(U \cup \{v\}, T \cup \{uv\})$ es un árbol.

Algoritmo 1 Exploración

Entrada: un grafo simple $G = (V, E)$ y una raíz $r \in V$.
Salida: un árbol generador de la componente conexa de r.
Partir con $U \leftarrow \{r\}$ y $T \leftarrow \emptyset$.
while existe $uv \in E$ con $u \in U$ y $v \notin U$ **do**
    Elegir $uv \in E$ con $u \in U$ y $v \notin U$.
    $U \leftarrow U \cup \{v\}$, $T \leftarrow T \cup \{uv\}$.
end while
return (U, T).

Lema. Sea $G = (V, E)$ un grafo simple y sea $H = (U, F)$ un subgrafo conexo de G. Si no existe una arista $uv \in E$ con $u \in U$ y $v \notin U$, entonces U es el conjunto de vértices de una componente conexa de G.

Teorema. Sea $G = (V, E)$ un grafo simple y sea $r \in V$. El algoritmo de exploración genérico devuelve un árbol generador de la componente conexa de r. En particular, si G es conexo, devuelve un árbol generador de G.

Definición (Tiempo de llegada)**.** En una ejecución del algoritmo de exploración, el *tiempo de llegada* de un vértice x es el instante en que x se agrega al conjunto U. Denotamos este tiempo por $\tau(x)$. La raíz tiene tiempo de llegada 0 y, cada vez que se agrega un vértice nuevo, se le asigna el siguiente tiempo de llegada.

Definición (Búsqueda en anchura (BFS))**.** La *búsqueda en anchura*, o *BFS*, es el algoritmo de exploración con la siguiente regla de elección. En cada paso escogemos una arista $uv \in E$ con $u \in U$ y $v \notin U$ tal que u tenga el menor tiempo de llegada posible:

$$\tau(u) = \text{mín}\{\tau(x) : x \in U \text{ y existe } y \notin U \text{ tal que } xy \in E\}.$$

Luego agregamos la arista uv y asignamos a v el siguiente tiempo de llegada. Si hay varias aristas que cumplen la regla, cualquiera de ellas puede elegirse.

Definición. Sea $G = (V, E)$ un grafo simple y sean $u, v \in V$. La *distancia* entre u y v, denotada por $d_G(u, v)$, es el largo mínimo de un camino desde u hasta v en G, medido por su número de aristas.

Proposición (BFS computa distancias mínimas)**.** Sea G un grafo simple conexo y sea $r \in V(G)$. El nivel de v en el árbol de búsqueda es exactamente $d_G(r, v)$.

Definición (Búsqueda en profundidad (DFS))**.** La *búsqueda en profundidad*, o *DFS*, es el algoritmo de exploración con la siguiente regla de elección. En cada paso escogemos una arista $uv \in E$ con $u \in U$ y $v \notin U$ tal que u tenga el mayor tiempo de llegada posible:

$$\tau(u) = \text{máx}\{\tau(x) : x \in U \text{ y existe } y \notin U \text{ tal que } xy \in E\}.$$

Luego agregamos la arista uv y asignamos a v el siguiente tiempo de llegada. Si hay varias aristas que cumplen la regla, cualquiera de ellas puede elegirse.

Definición. Un *grafo con pesos* es un grafo $G = (V, E)$ junto con una función $w\colon E \to \mathbb{R}$. Si T es un subgrafo de G, su *peso* es $w(T) = \sum_{e \in E(T)} w(e)$.

Definición. Sea G un grafo simple con pesos w. Un *árbol generador de costo mínimo (MST)* de G con pesos w es un subgrafo (U, T) tal que

- (U, T) es un árbol generador,
- minimiza $\displaystyle\sum_{e \in T} w(e)$ entre todos los árboles generadores de G.

Teorema. Sea G un grafo simple, conexo y con pesos. Suponga que todos los pesos de las aristas son distintos. Entonces G tiene un único árbol generador de costo mínimo.

Lema (Regla azul)**.** Sea $G = (V, E)$ un grafo simple, conexo y con pesos w distintos, $U \subseteq V$. Si $e = uv$ es una arista de menor costo entre las aristas con $u \in U$ y $v \notin U$, entonces el MST la contiene.

Algoritmo 2 Prim

Entrada: un grafo simple y conexo $G = (V, E)$ con pesos distintos y una raíz $r \in V$.
Salida: un árbol generador de costo mínimo de G.
Partir con $U \leftarrow \{r\}$ y $T \leftarrow \emptyset$.
while $U \neq V$ **do**
    Elegir $uv \in E$ con $u \in U$ y $v \notin U$ de menor costo.
    $U \leftarrow U \cup \{v\}$, $T \leftarrow T \cup \{uv\}$.
end while
return (U, T).

Teorema. Sea $G = (V, E)$ un grafo simple y conexo con pesos w distintos. El algoritmo de Prim devuelve un árbol generador de costo mínimo.

Lema (Regla roja)**.** Sea G un grafo simple, conexo y con pesos w distintos, y C las aristas de un ciclo en G. Si $e = uv$ es una arista de C de mayor costo, entonces no pertenece al MST.

Algoritmo 3 Kruskal

Entrada: un grafo simple y conexo $G = (V, E)$ con pesos w distintos.
Salida: un árbol generador de costo mínimo de G.
Ordenar las aristas por peso $w(e_1) < w(e_2) < \cdots < w(e_m)$.
Partir con $T \leftarrow \emptyset$.
for $i = 1, \ldots, m$ **do**
    if $(V, T \cup \{e_i\})$ es acíclico **then**
        $T \leftarrow T \cup \{e_i\}$.
    end if
end for
return (V, T).

Teorema. Sea $G = (V, E)$ un grafo simple y conexo con pesos w distintos. El algoritmo de Kruskal devuelve un árbol generador de costo mínimo.

Observación. En un grafo simple y conexo con pesos que pueden repetirse, Prim y Kruskal siguen encontrando un MST al romper los empates arbitrariamente.

Ejercicios

1. *a*) Sea G un grafo conexo y sea T un árbol generador de G.
    1) Demuestre que si e es una arista de G que no pertenece a T, entonces $T + e$ contiene exactamente un ciclo.
    2) Demuestre que si e es una arista de T, entonces $T - e$ tiene exactamente dos componentes conexas.

b) Sea G un grafo simple conexo. Diremos que una arista e de G es un *puente* si $G - e$ no es conexo. Demuestre que una arista e pertenece a todo árbol generador de G si y sólo si e es un puente. Concluya que G tiene un único árbol generador si y sólo si G es un árbol.

c) Un equipo quiere instalar túneles privados entre servicios de una aplicación distribuida. Los servicios son A, B, C, D, E, F y los posibles túneles, junto con su costo mensual, son

túnel	AB	AC	BC	BD	CD	CE	DE	DF	EF	BF
costo	4	2	5	1	3	7	2	6	4	8

Se quiere que todos los servicios puedan comunicarse entre sí, directamente o a través de otros servicios, minimizando el costo total.

1) Modele el problema como un problema de árboles generadores de costo mínimo.
2) Determine el costo mínimo total.

d) Hay n viviendas que se quieren conectar al suministro de agua. Entre las viviendas i y j se puede construir una cañería de costo c_{ij}. Además, la vivienda i puede conectarse directamente a la matriz principal de agua con costo $C_i \geq 0$. Se quiere que todas las viviendas tengan acceso al agua al menor costo total posible. ¿Cómo puede resolver este problema usando algoritmos que ya conoce?

2. Alicia se desplaza por una ciudad que tiene lugares de interés etiquetados por $1, 2, \ldots, n$. Alicia sabe que en cada trayecto ij hay un nivel de ruido r_{ij}. Suponga que $n \geq 2$, que estos niveles son positivos, que los trayectos se pueden recorrer en ambos sentidos y que forman un grafo simple y conexo. Alicia quiere seleccionar algunos de ellos para formar una red que conecte todos los lugares de interés.

a) Considere una ciudad con seis lugares y únicamente los siguientes trayectos, con sus respectivos niveles de ruido:

trayecto	12	13	15	23	24	25	34	35	46	56
ruido	1	4	9	5	8	2	6	3	7	10

Determine una red que conecte los seis lugares y minimice la suma de los niveles de ruido de sus trayectos. ¿Cuál es esa suma mínima?

b) Sea G un grafo simple, conexo y con pesos, y sea T un MST de G. Escribamos $G = (V, E)$ y denotemos sus pesos por w. Para dos vértices $u, v \in V(G)$ distintos, sea $P_T(u, v)$ el único camino simple de u a v en T. Demuestre que $P_T(u, v)$ minimiza el peso de la arista más pesada del camino.

Más precisamente, demuestre que para todo camino P de u a v en G se cumple

$$\max_{e \in P_T(u,v)} w(e) \leq \max_{e \in P} w(e).$$

c) ¿Cómo puede Alicia construir una red que conecte todos los lugares y minimice el máximo de los niveles de ruido de los trayectos elegidos? Describa y justifique un algoritmo usando la parte (b). En la instancia de la parte (a), indique una red óptima para este nuevo objetivo y el máximo nivel de ruido que alcanza.

3. Sea $G = (V, E)$ un grafo simple y conexo con pesos distintos, y sea $n = |V|$. El *algoritmo de Borůvka* comienza con $F = \varnothing$. Mientras (V, F) no sea conexo, realiza una ronda de la siguiente forma:

   - Al comienzo de la ronda, calcula los conjuntos de vértices $C_1, \ldots, C_k$ de las componentes conexas de (V, F).
   - Para cada C_i, elige la arista de menor peso con un extremo en C_i y el otro fuera de C_i.
   - Agrega simultáneamente a F el conjunto de aristas elegidas; si una arista fue elegida por dos componentes, la agrega una sola vez.

   a) Demuestre que, después de cada ronda, todas las aristas de F pertenecen al único MST de G. Concluya que (V, F) es un bosque.

   b) Demuestre que una ronda que comienza con $k > 1$ componentes termina con a lo más $k/2$ componentes.

   c) Concluya que el algoritmo termina en a lo más $\lceil \log_2 n \rceil$ rondas y devuelve el MST de G.