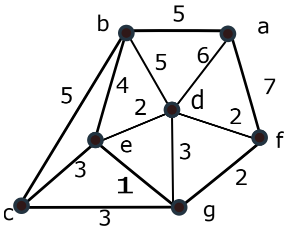
# Árbol generador óptimo: algoritmos de Prim y Kruskal

S. Bianchi     P. Fekete     F- Domingo

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

4 de octubre de 2021

Sea $G=(V,E)$ un grafo conexo que modeliza una red de comunicación entre distintas regiones.




Las regiones se representan mediante los vértices en el grafo y las conexiones entre regiones mediante aristas.

En este grafo, los números sobre cada arista representan los costos de conexión en cada caso.
El objetivo es mantener la conexión de la red al menor costo posible.

# Introducción

### Definición

Si $G=(V,E)$ es un grafo tal que a cada $e\in E$ se asigna $p(e)\in\mathbb{R}^+$, entonces se dice que $G$ es un grafo *ponderado*. El valor $p(e)$ se denomina *costo* o *peso* de $e\in E$.

### Definición

Si $G=(V,E)$ es un grafo ponderado y $T$ es un árbol generador de $G$, el costo o peso de $T$ es $\sum_{e\in E(T)} p(e)$.

**Problema** Dado $G=(V,E)$, grafo conexo no dirigido y sin lazos, es necesario obtener un árbol generador de $G$ de mínimo peso.
Vamos a ver dos algoritmos que resuelven el problema planteado:

**Algoritmo de Kruskal** (desarrollado por Joseph Kruskal 1928-2010)

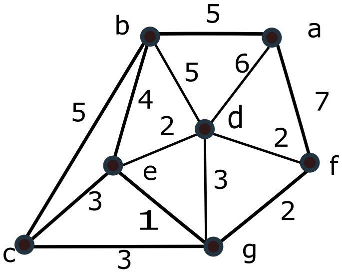
**Algoritmo de Prim** (desarrollado por Robert Prim 1921-)

Sea $G = (V,E)$ grafo no dirigido y conexo, sin lazos y $|V| = n$. Para cada arista $e \in E$ está definido el peso $p(e)$.

**Algoritmo de Kruskal**

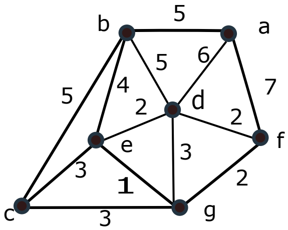
1. $i := 1$, sea $e_1 \in E$ tal que $p(e_1) = min\{p(e) : e \in E\}$. $E(T) = \{e_1\}$.
2. Para $1 \leq i \leq n-2$, sea $e_{i+1} \in E - E(T)$ tal que $p(e_{i+1}) = min\{p(e) : e \in E - E(T)\}$ y $e_{i+1}$ no forme un ciclo con las aristas en $E(T)$. $E(T) := E(T) \cup \{e_{i+1}\}$.
3. $i := i+1$
   - Si $i = n-1$, el subgrafo determinado por $E(T)$ es un árbol generador de $G$.
   - Si $i < n-1$, ir al Paso 2.

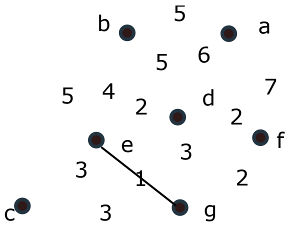
Veamos un ejemplo:




Los números sobre cada arco indican los costos.

Paso 1 Inicialización $i := 1$.
Elegimos la arista $\{e, g\}$.

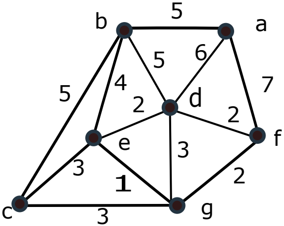


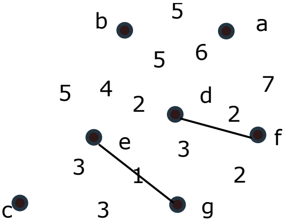




Agregamos $\{e, g\}$ a $E(T)$.

Vamos al Paso 2.

Paso 2 Elegimos una arista que no está en E(T) con el menor costo. Hay $3$ aristas posibles, elegimos $\{d,f\}$.



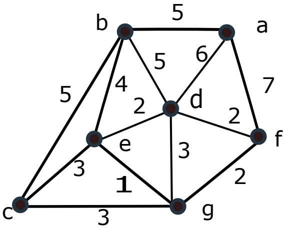



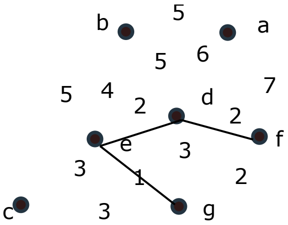

Agregamos $\{d,f\}$ a $E(T)$.

Paso 3 $i := 2$. Como $2 < 6$ vamos al Paso 2.

# ALGORITMO DE KRUSKAL

Paso 2 Seleccionamos $\{d,e\}$.

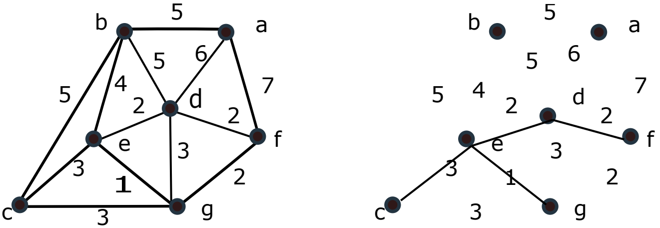



agregamos $\{d,e\}$ a $E(T)$.

Paso 3 $i := 3$. Como $3 < 6$ vamos al Paso 2.

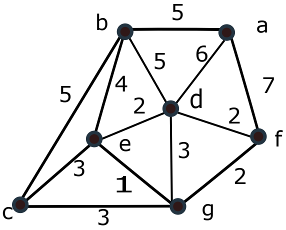
Paso 2 $\{f,g\}$ tiene el menor peso de las aristas que no están en $E(T)$, pero al agregarla a $E(T)$ forma un ciclo.
Entonces analizamos las aristas $\{c.e\},\{c,g\},\{d,g\}$ que tienen igual costo.
Descartamos $\{d,g\}$ porque produce un ciclo con las aristas en $E(T)$.
Seleccionamos $\{c,e\}$.

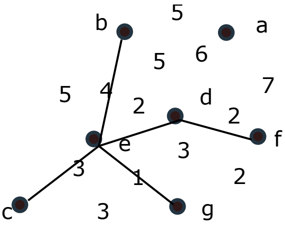



Agregamos $\{c,e\}$ a $E(T)$.

Paso 2 $i := 4$. Como $4 < 6$ vamos al Paso 2.

Paso 2 Seleccionamos $\{e, b\}$.

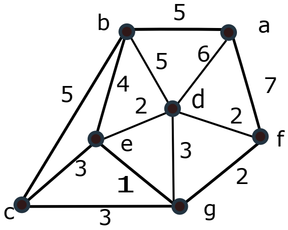


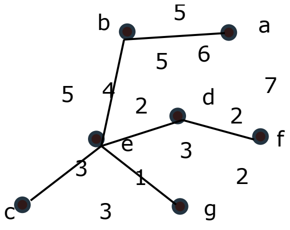




Agregamos $\{e, b\}$ a $E(T)$.

Asignamos $i := 5$ y como $5 < 6$ vamos al Paso 2.

Paso 2 seleccionamos $\{b, a\}$.




agregamos $\{b, a\}$ a $E(T)$.
Paso 3 $i := 6$. Como $6 = 7 - 1$ entonces las aristas en $E(T)$ corresponden al árbol generador de peso mínimo.

El algoritmo de Kruskal es un algoritmo *goloso*. En cada iteración selecciona la arista de menor peso que no crea ciclo.
Cómo nos aseguramos que efectivamente encuentra el árbol generador de peso mínimo?

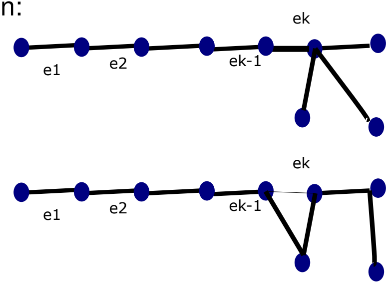
### TEOREMA

Sea $G = (V, E)$ un grafo no dirigido, conexo, ponderado y sin lazos. Cualquier árbol generador de $G$ obtenido mediante el algoritmo de Kruskal es óptimo.

**Demostración**

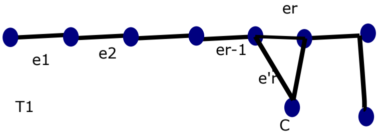
Sea $|V| = n$ y $T$ árbol generador obtenido mediante el algoritmo de Kruskal.
Sean $\{e_1, e_2, \ldots, e_{n-1}\}$ las aristas de $T$ enumeradas de acuerdo al orden en que fueron seleccionadas.

**Demostración**(cont.)
Dado $T'$ un árbol óptimo, sea $d(T') = k$ si $T'$ contiene a $\{e_1, e_2, \dots, e_{k-1}\}$ y no contiene a $e_k$.
Veamos una ilustración:

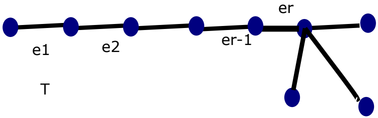



$T$ es el árbol de arriba y $T'$ es el de abajo.

**Demostración**(cont.)
Sea $T_1$ un árbol óptimo para el cual $d(T_1) = r$ sea máxima.
Si $r = n$, entonces $T = T_1$ queda probado el teorema.
Si $r \leq n-1$, al agregar la arista $e_r \in E(T)$ a $T_1$ obtenemos un ciclo $C$.
Veamos:

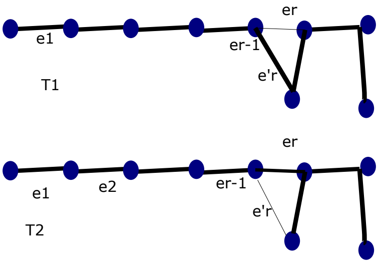



En $C$ existe una arista $e'_r$ que pertenece a $T_1$ y no pertenece a $T$.



**Demostración**(cont.)

Partimos de $T_1$, luego

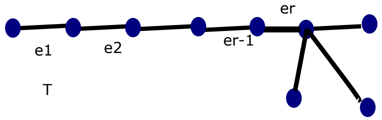



agregamos $e_r \in E(T_1)$ y eliminamos $e'_r$ y obtenemos el grafo $T_2$ que es conexo, tiene $n-1$ aristas es decir $T_2$ es un árbol.

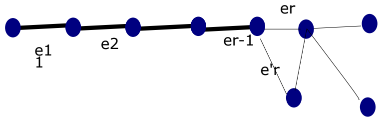
Entonces:

$$p(T_2) = p(T_1) + p(e_r) - p(e'_r)$$

**Demostración**(cont.) Recordemos que los arcos $e_1, e_2, \ldots, e_{r-1}, e_r$ fueron elegidos en el algoritmo de Kruskal con el criterio de minimizar el peso y no formar ciclos.




Sea $H$ el subgrafo de $G$ determinado por $e_1, e_2, \ldots, e_{r-1}$.




Es claro que agregar $e'_r$ a $H$ no forma ciclo. Por lo tanto $p(e_r) \leq p(e'_r)$.

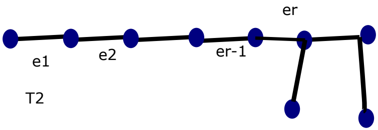
**Demostración**(cont.)

Entonces

$$p(T_2) = p(T_1) + p(e_r) - p(e'_r) \leq p(T_1).$$

Pero $T_1$ es óptimo, sigue que $p(T_2) = p(T_1)$. Por lo tanto $T_2$ es óptimo.

El árbol $T_2$ tiene $e_1, e_2, \ldots, e_{r-1}, e_r$ aristas en común con $T$.




Es decir $d(T_2) \geq r + 1 > r = d(T_1)$.

Esto contradice la elección de $T_1$.

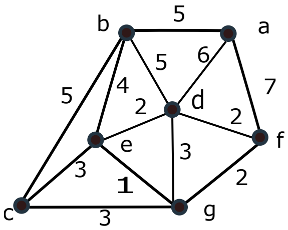
Resulta $T_1 = T$ y $T$ es árbol óptimo obtenido por el algoritmo de Kruskal.

Sea $G=(V,E)$ ponderado, no dirigido, conexo sin lazos.
Sea $n=|V|$. El algoritmo construye dos conjuntos de vértices de $V$ llamados $P$ y $N$.

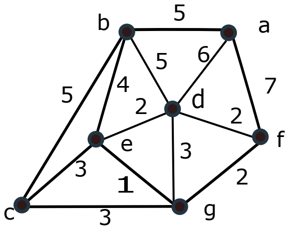
**Algoritmo de Prim**

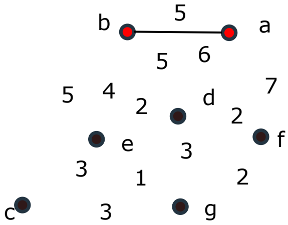
1. $i:=1$. Sea $v_1\in V$. Se definen $P:=\{v_1\}$, $N:=V-\{v_1\}$ y $E(T):=\emptyset$.
2. Para $1\leq i\leq n-1$, sean $P=\{v_1,v_2,\ldots,v_i\}$,$E(T)=\{e_1,e_2,\ldots,e_{i-1}\}$ y $N=V-P$. Agregar a $E(T)$ la arista de menor peso de $E$ que conecta un vértice de $P$ con un vértice de $N$. Sea $v_{i+1}\in N$ el vértice que cumple esta propiedad. $P\cup\{v_{i+1}\}$ y $N-\{v_{i+1}\}$.
3. $i:=i+1$
   - Si $i=n$, el subgrafo determinado por $E(T)$ es un árbol generador óptimo de $G$.
   - Si $i<n$, ir al Paso 2.

Vamos a utilizar el algoritmo de Prim en el mismo ejemplo anterior.



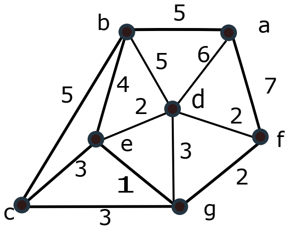
Paso 2 $P := \{a, b\}; N = \{c, d, e, f, g\}; E(T) = \{\{a, b\}\}$.

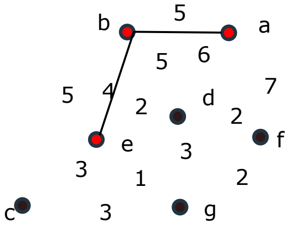



Paso 3 $i := 2$. Como $2 < 7$, entonces vamos al Paso 2.

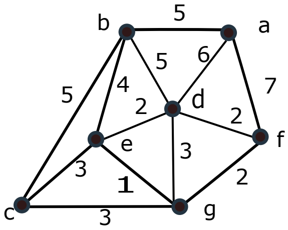
Paso 2 $P := \{a, b, e\}; N = \{c, d, f, g\}; E(T) = \{\{a, b\}, \{b, e\}\}$.

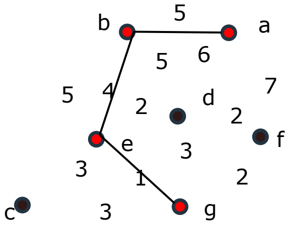



Paso 3 $i := 3$. Como $3 < 7$, entonces vamos al Paso 2.

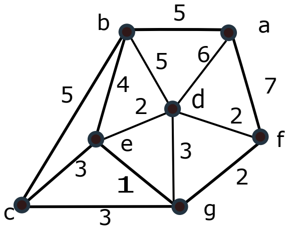
Paso 2 $P := \{a, b, e, g\}$; $N = \{c, d, f\}$; $E(T) = \{\{a, b\}, \{b, e\}, \{e, g\}\}$.

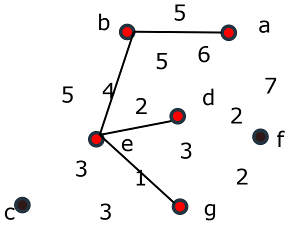



Paso 3 $i := 4$. Como $4 < 7$, entonces vamos al Paso 2.

Paso 2 $P := \{a,b,e,g,d\}$; $N = \{c,f\}$; $E(T) = \{\{a,b\},\{b,e\},\{e,g\},\{d,e\}\}$.

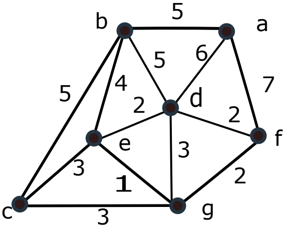


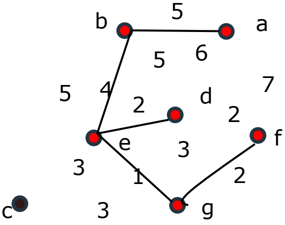




Paso 3 $i := 5$. Como $5 < 7$, entonces vamos al Paso 2.

# ALGORITMO DE PRIM

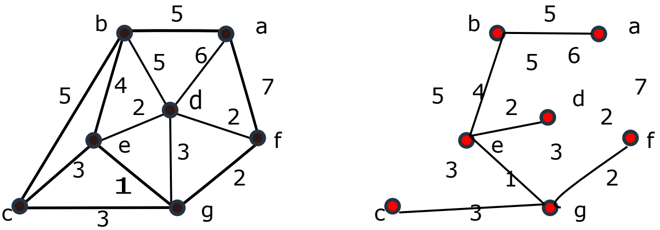
Paso 2 $P := \{a, b, e, g, d, f\}; N = \{c\};$
$E(T) = \{\{a, b\}, \{b, e\}, \{e, g\}, \{d, e\}, \{f, g\}\}.$




Paso 3 $i := 6$. Como $6 < 7$, entonces vamos al Paso 2.

Paso 2 $P := \{a, b, e, g, d, f, c\}; N = \emptyset;$
$E(T) = \{\{a, b\}, \{b, e\}, \{e, g\}, \{d, e\}, \{f, g\}, \{c, g\}\}.$




Paso 3 $i := 7$. $E(T)$ contiene a las aristas del árbol generador óptimo $T$.

Al igual que el algoritmo de Kruskal, el algoritmo de Prim es es un algoritmo goloso. En cada iteración selecciona la arista de menor peso que no crea ciclo. Nuevamente: cómo nos aseguramos que efectivamente encuentra el árbol generador de peso mínimo?

### Teorema

Sea $G = (V, E)$ un grafo no dirigido, conexo, ponderado y sin lazos. Cualquier árbol generador de $G$ obtenido mediante el algoritmo de Prim es óptimo.

**Demostración**

Ejercicio.