DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

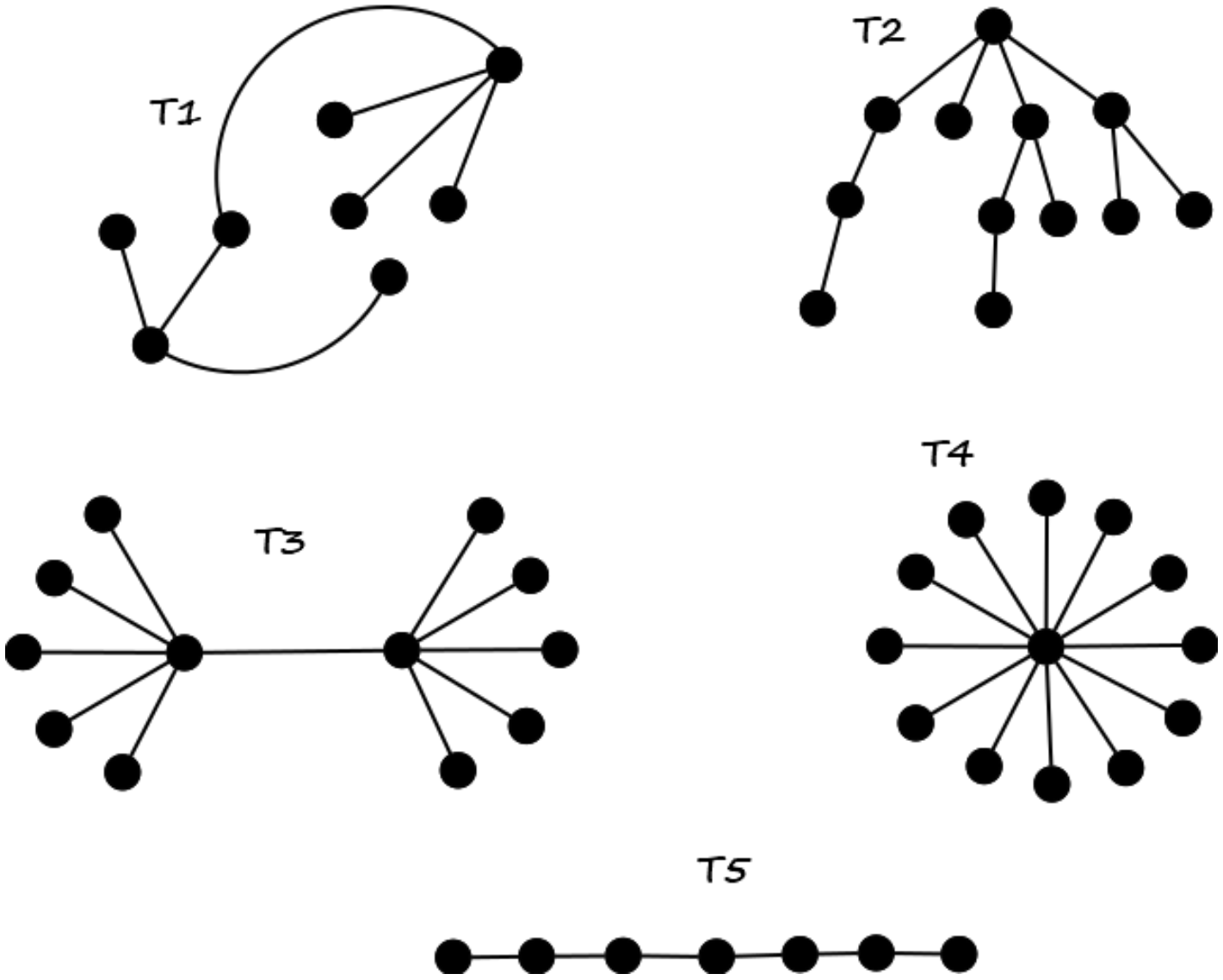
Algoritmos y Estructuras de Datos III

Primer cuatrimestre 2021 *(dictado a distancia)*

Árboles

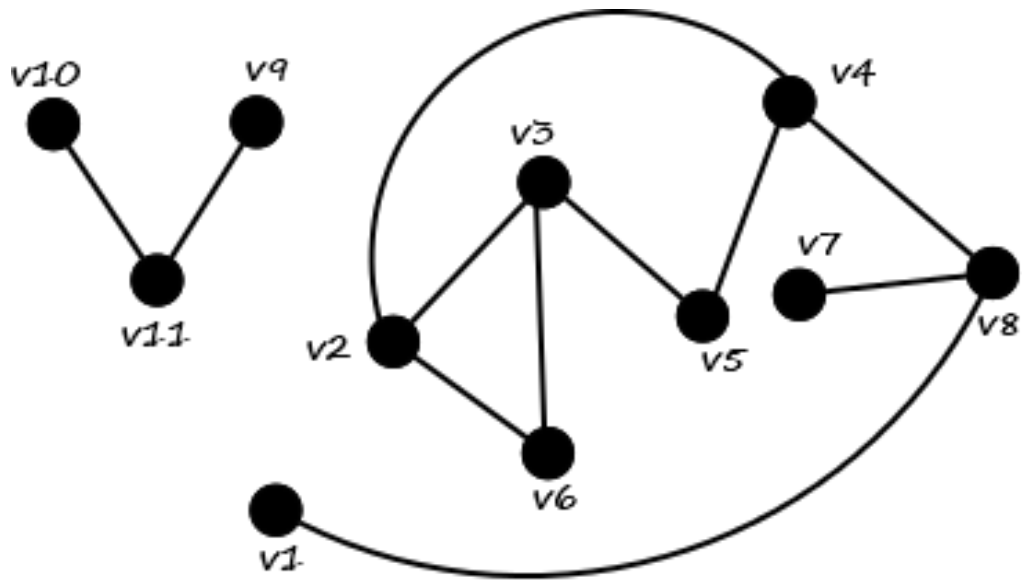
Árboles

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



Árboles

Una arista de G es *puente* si $G - e$ tiene más componentes conexas que G.



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

Árboles

Teorema: 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 exáctamente 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).

Lema 1: (Ejercicio 5 Práctica 2) 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.

Árboles

Una *hoja* es un nodo de grado 1.

Lema 3: Todo árbol no trivial tiene al menos dos hojas.

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

Un *bosque* es un grafo sin circuitos simples.

Corolario 1: Sea G un bosque con c componentes conexas. Entonces $m = n - c$.

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

Árboles

Teorema: 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 $m = n - 1$.

Árboles enraizados

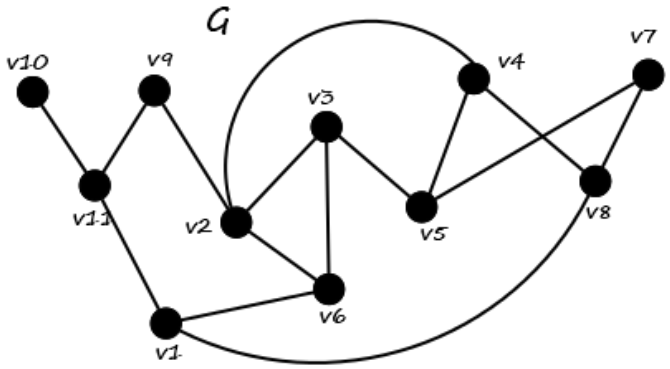
- Un *árbol enraizado* es un árbol que tiene un vértice distinguido que llamamos *raíz*.
- Explícitamente queda definido un árbol dirigido.
- El *nivel* de un vértice es la distancia de la raíz a ese vértice.
- La *altura* h de un árbol enraizado es el máximo nivel de sus vértices.
- Un árbol se dice *m-ario* si todos sus vértices, salvo las hojas y la raíz tienen grado a lo sumo $m+1$ y la raíz (exáctamente) a lo sumo m.
- Un árbol se dice *balanceado* si todas sus hojas están a nivel h o $h-1$.

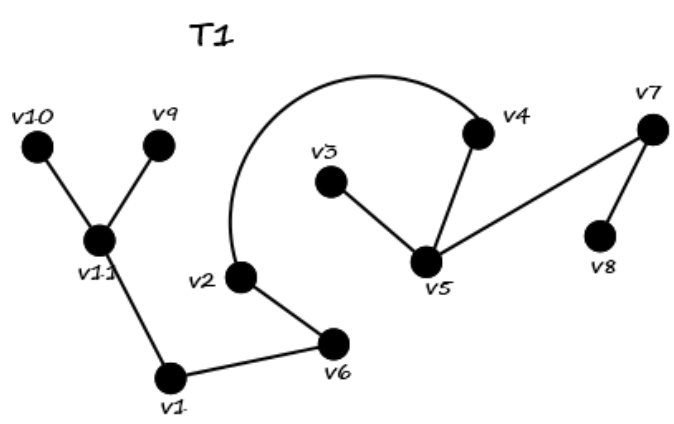
Árboles enraizados

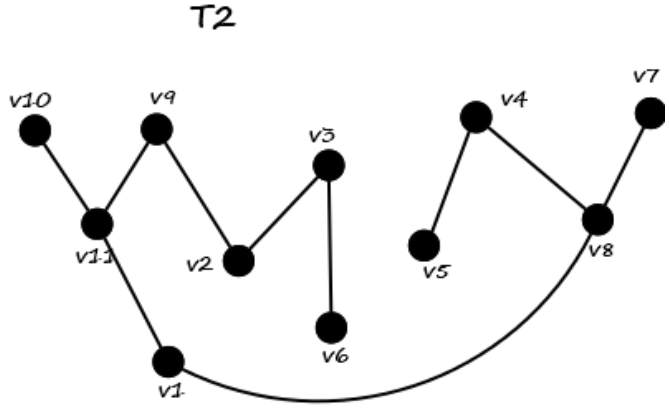
- Los vértices *internos* de un árbol son aquellos que no son ni hojas ni la raíz.
- Decimos que dos vértices adyacentes tienen *relación padre-hijo*, siendo el padre el vértice de menor nivel.

Árboles generadores

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







T_1 y T_2 son árboles generadores de G

Árboles generadores

Teorema:

- Todo grafo conexo tiene (al menos) un árbol generador.
- G conexo. G tiene un único árbol generador $\iff$ G es árbol.
- 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.

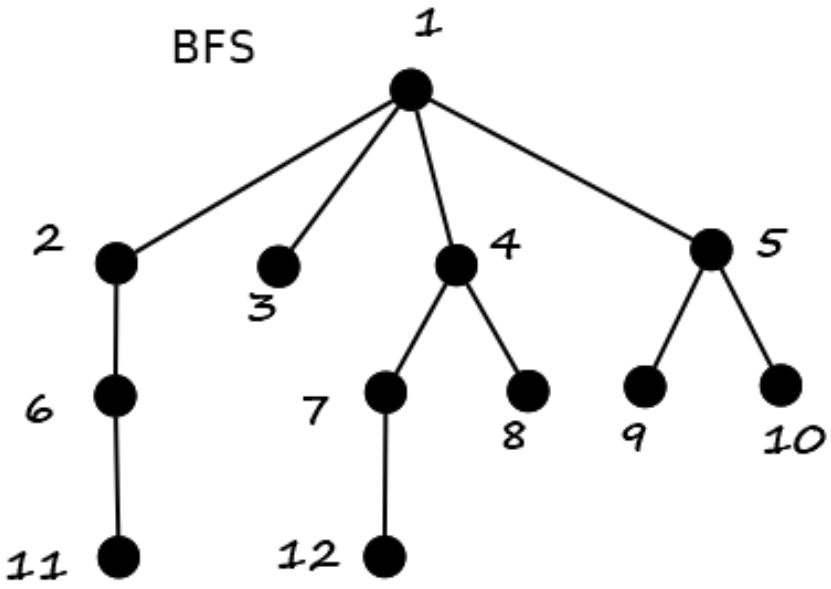
Recorrido de árboles o grafos

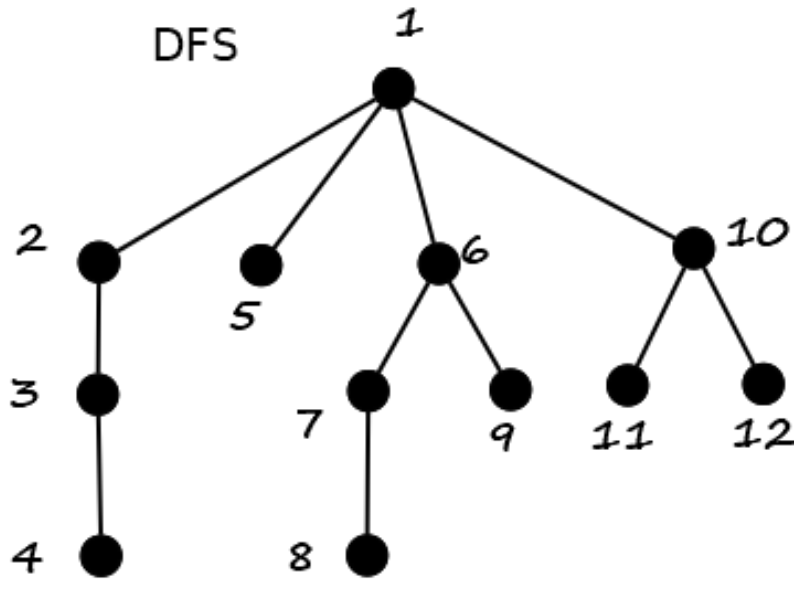
- En muchas situaciones y algoritmos, dado un árbol (o grafo), queremos recorrer sus vértices exáctamente una vez.

- Para hacerlo de una forma ordenada y sistemática, podemos seguir dos órdenes:
  - *a lo ancho* (*Breadth-First Search - BFS*): se comienza por el nivel 0 (la raź) y se visita cada vértice en un nivel antes de pasar al siguiente nivel.

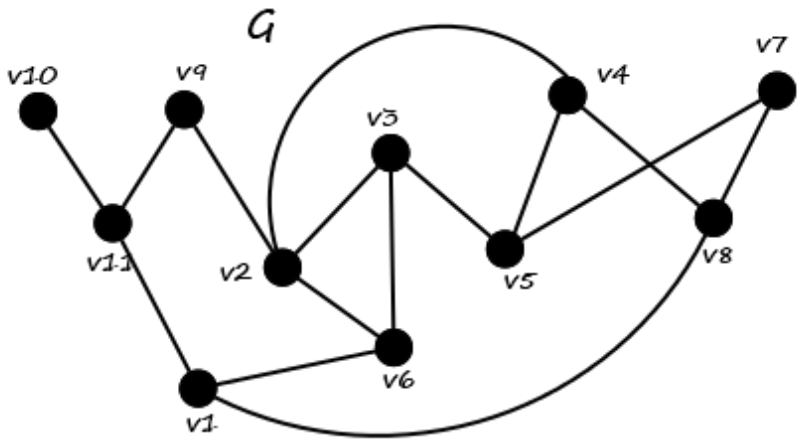
  - *en profundidad* (*Depth-First Search - DFS*): se comienza por la raíz y se explora cada rama lo más profundo posible antes de retroceder.

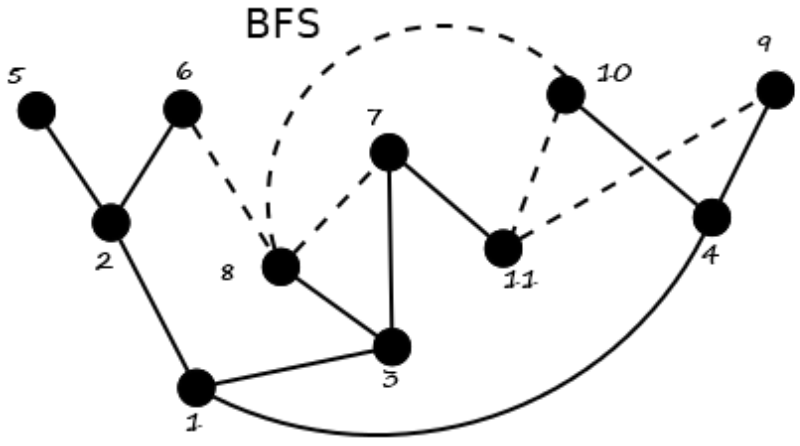
Recorrido de árboles o grafos

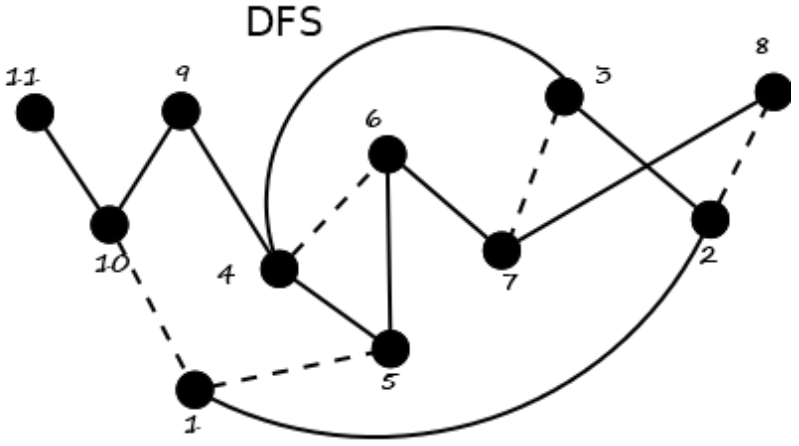




Recorrido de árboles o grafos







Recorrido de árboles o grafos

```
recorrer(G)
    salida: pred[i] = padre de v_i - orden[i] = numero asignado a v_i
    next ← 1
    r ← elegir un vertice como raiz
    pred[r] ← 0                                        (marcar vertice r)
    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
            pred[j] ← i                                (marcar vertice j)
            next ← next + 1
            orden[j] ← next
            LISTA ← LISTA ∪ {j}
        sino
            LISTA ← LISTA \ {i}
        fin si
    fin mientras
    retornar pred y orden
```

Recorrido de árboles o grafos

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

- **BFS**: *LISTA* implementada como cola.
- **DFS**: *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:

- 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 vértices que están a distancia menor que k.

Árbol generador mínimo

- Volvamos al problema se conectar n puntos.
- Supongamos además que cada posible enlace que se puede construir tiene un costo asociado (puede ser costo de contrucción o 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.

Árbol generador mínimo

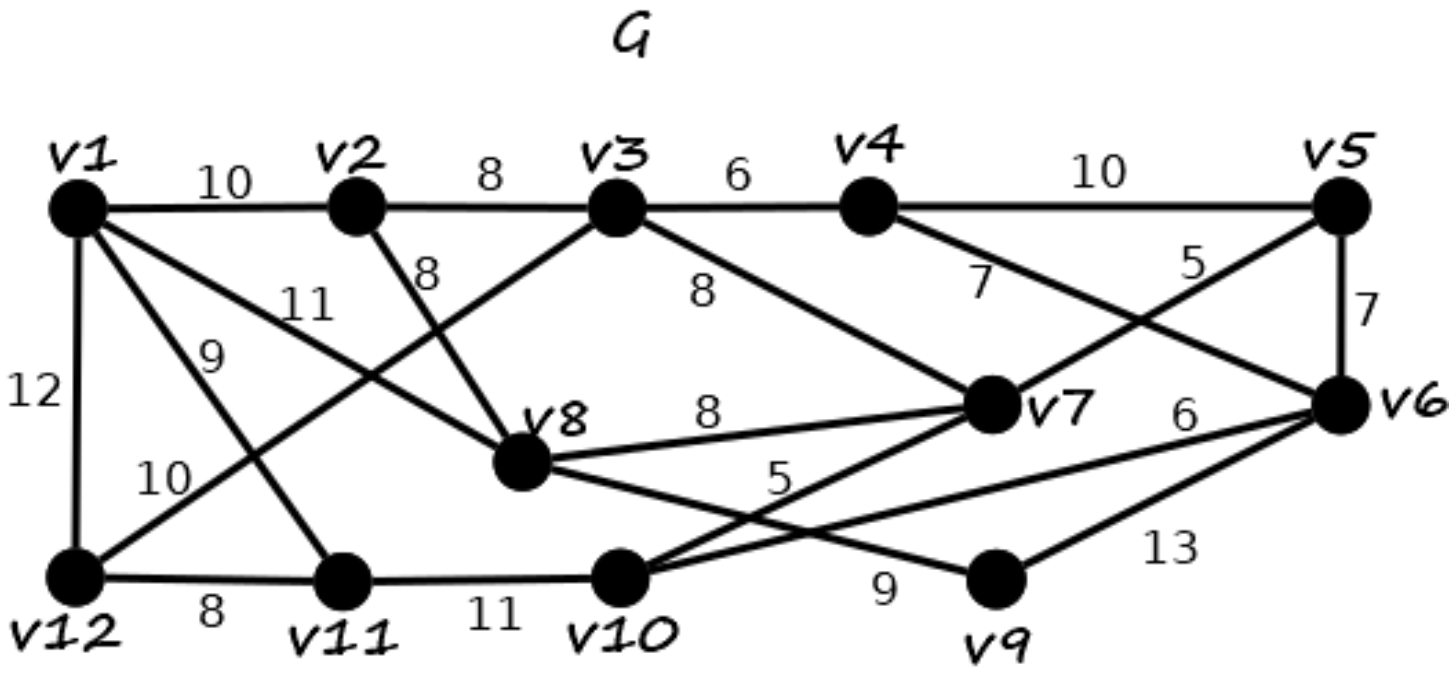
- Sea $T = (V, X)$ un árbol y $l : X \to \mathbb{R}$ una función que asigna costos a las aristas de T. Se define el *costo* de T como $l(T) = \sum_{e \in T} l(e)$.

- Dado un grafo $G = (V, X)$ un *árbol generador mínimo* de G, T, es un árbol generador de G de mínimo costo, es decir

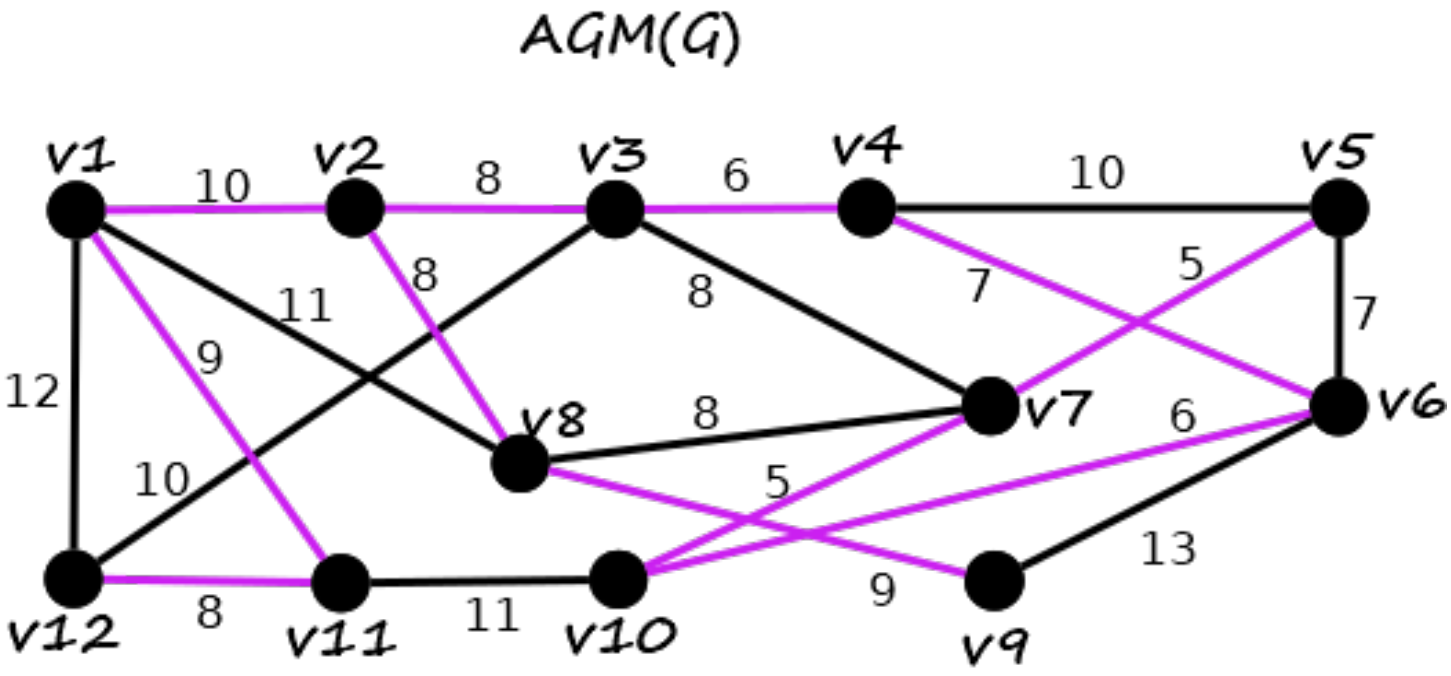
$$l(T) \leq l(T') \; \forall T' \text{ árbol generador de } G.$$

- Dado un grafo pesado en las aristas, $G = (V, X)$, el problema de *árbol generador mínimo* consiste en encontrar un AGM de G.

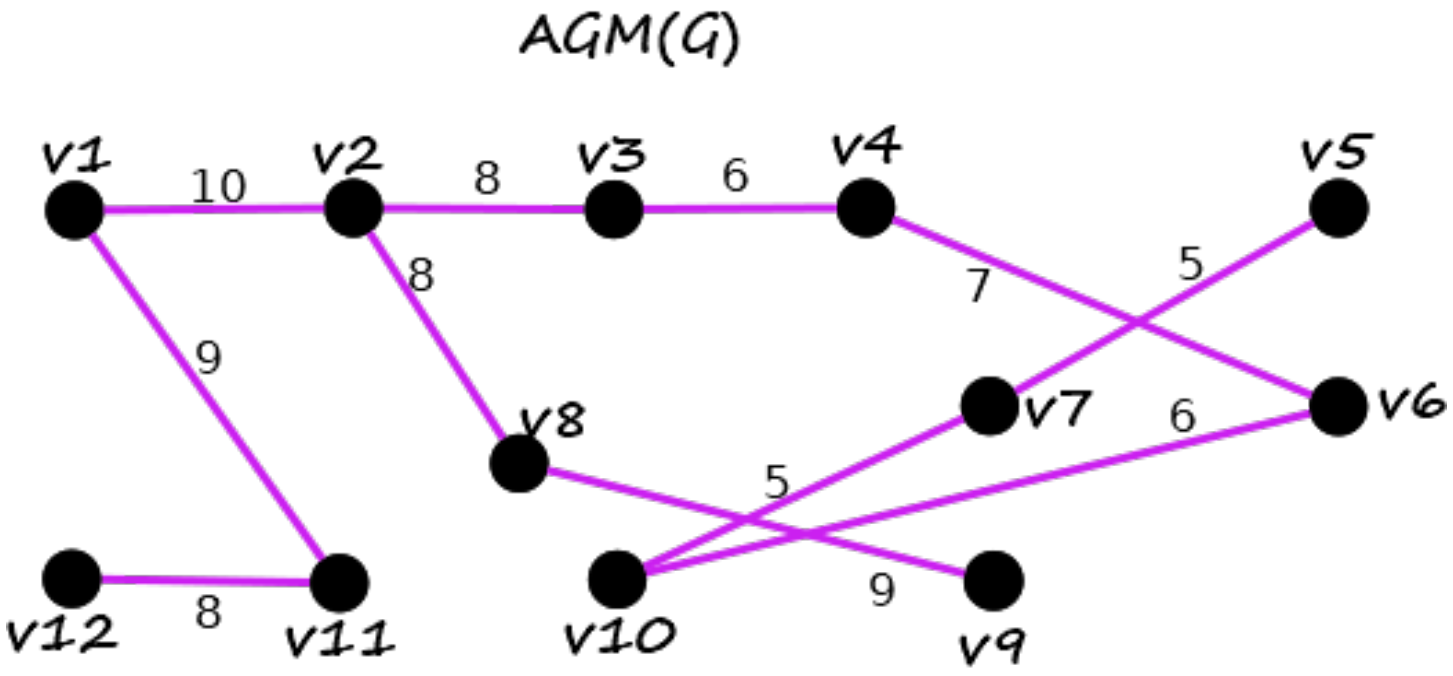
Árbol generador mínimo - ejemplo



Árbol generador mínimo - ejemplo



Árbol generador mínimo - ejemplo

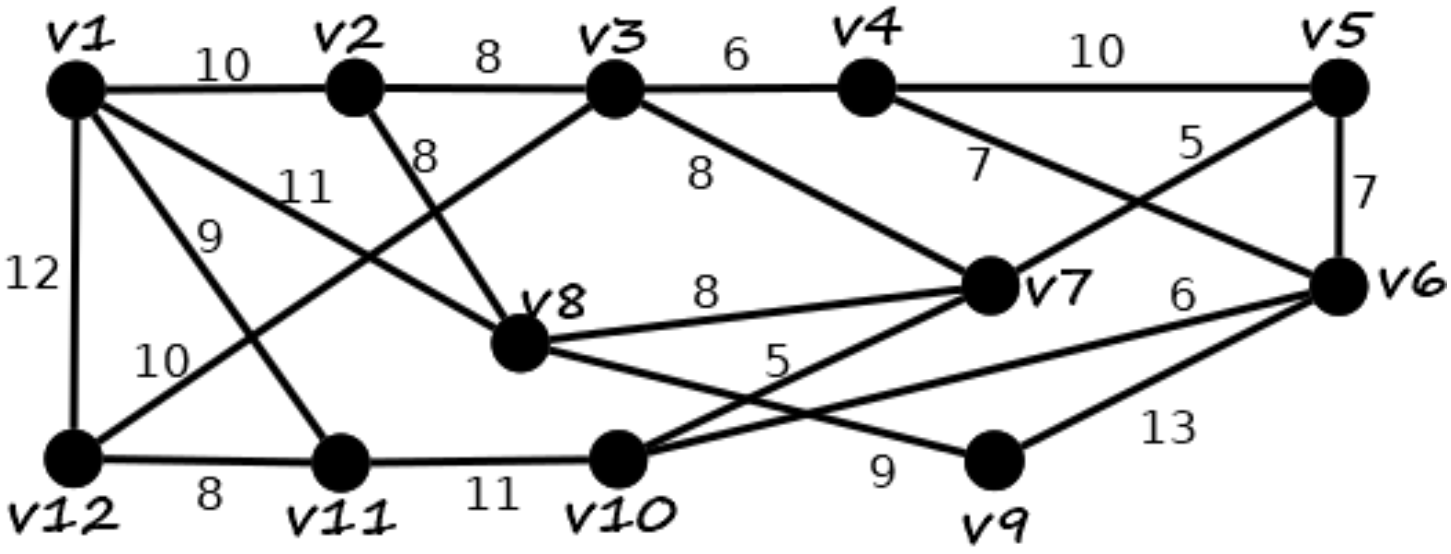


Árbol generador mínimo - Algoritmo de Prim (1957)

- 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 conjunto.
- 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 arista es agregada a X_T y el extremo a V_T.
- Cuando $V_T = V$ el algoritmo termina y las aritas de X_T definen un AGM de G.
- Esta es una elección golosa: de un conjunto de aristas candidatas, elige la *mejor* arista (menor costo).

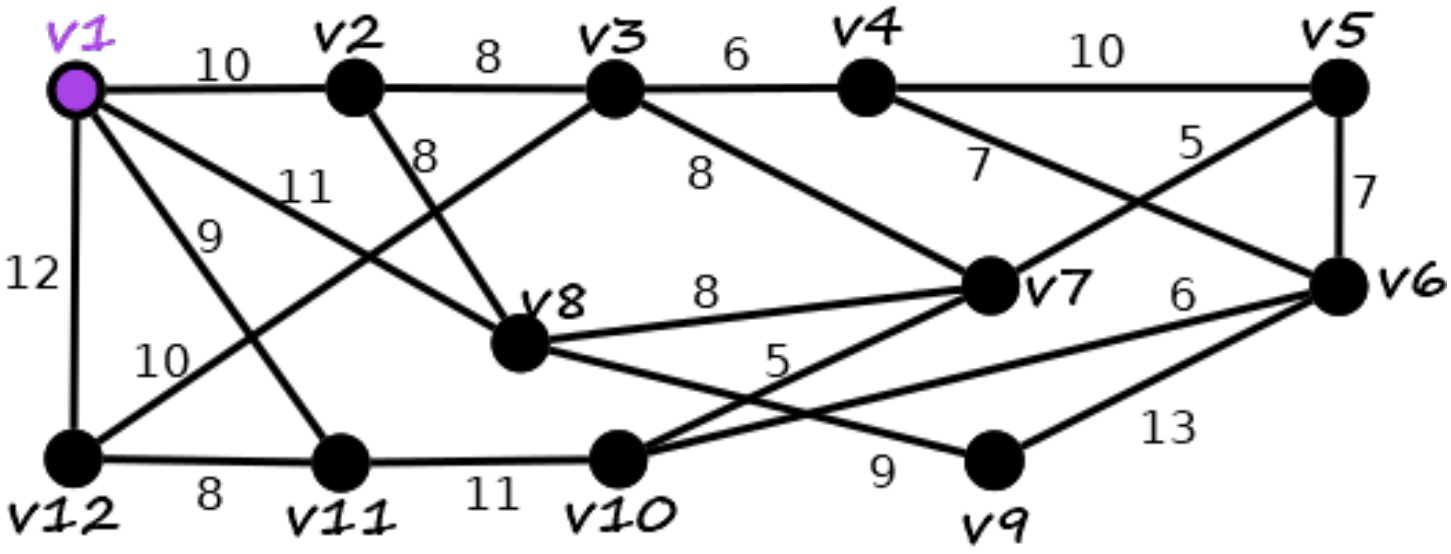
Árbol generador mínimo - Algoritmo de Prim

Iteración k: Subgrafo de un AGM conexo con k aristas



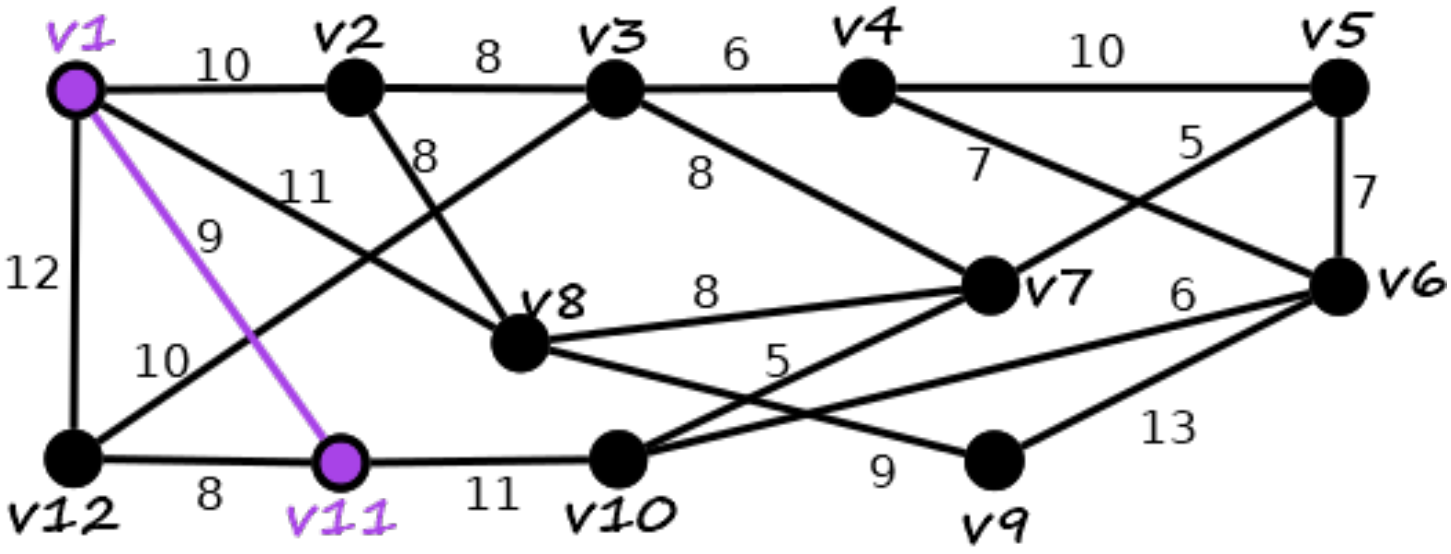
Árbol generador mínimo - Algoritmo de Prim

Iteración k: Subgrafo de un AGM conexo con k aristas



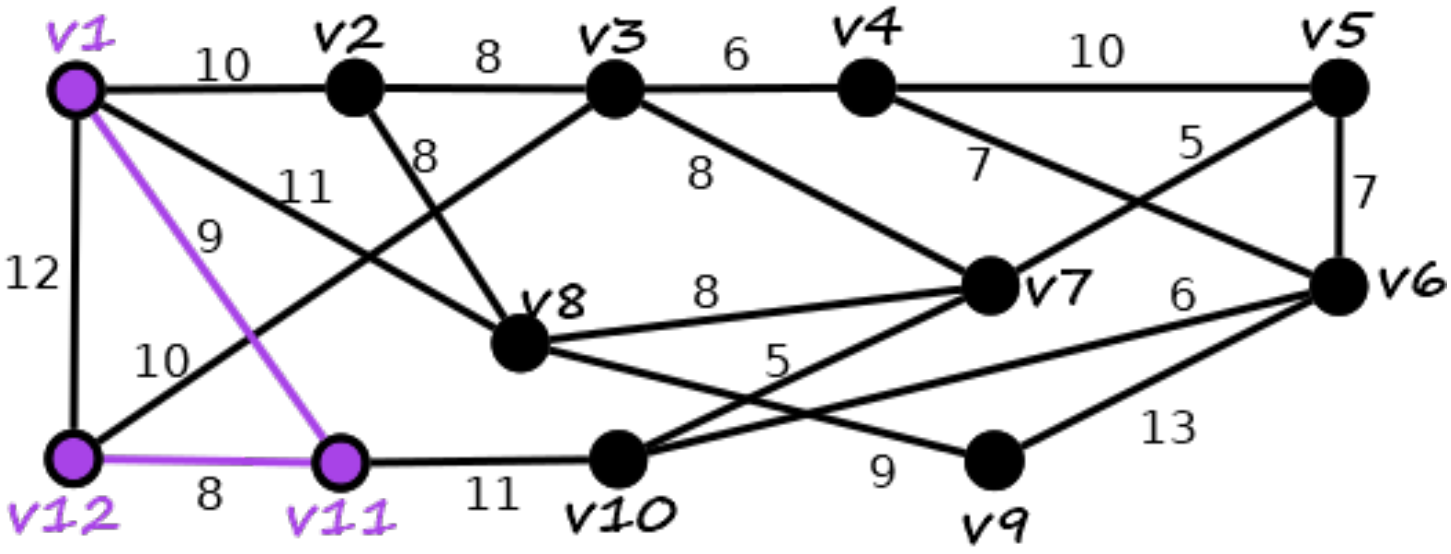
Árbol generador mínimo - Algoritmo de Prim

Iteración k: Subgrafo de un AGM conexo con k aristas



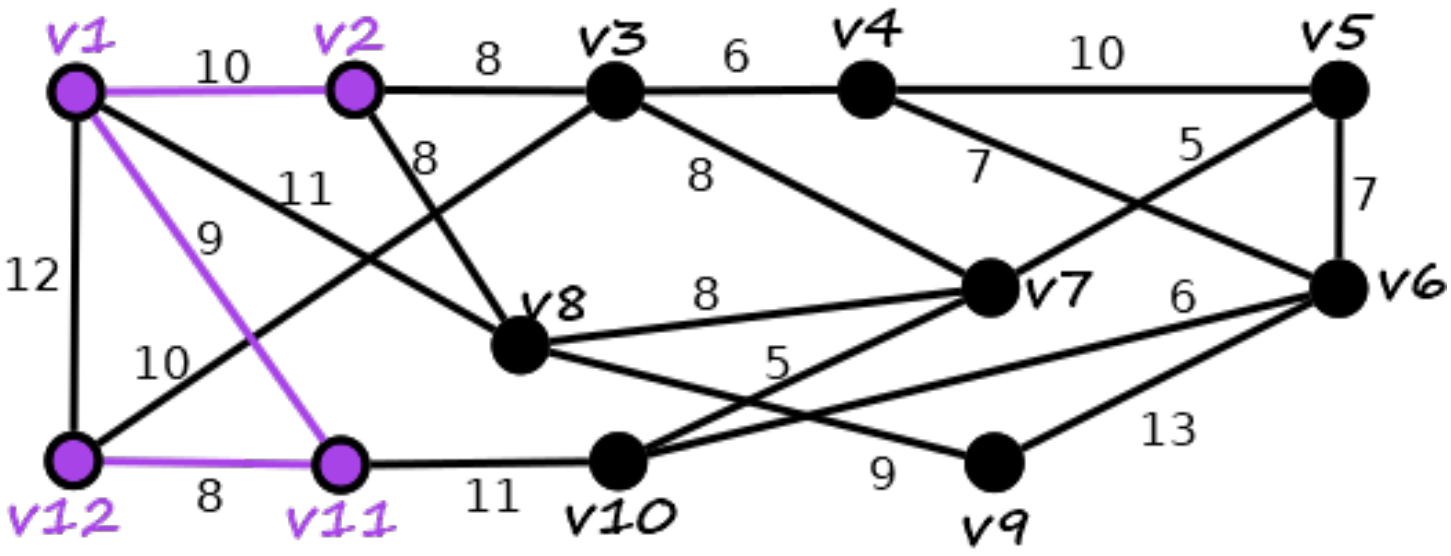
Árbol generador mínimo - Algoritmo de Prim

Iteración k: Subgrafo de un AGM conexo con k aristas



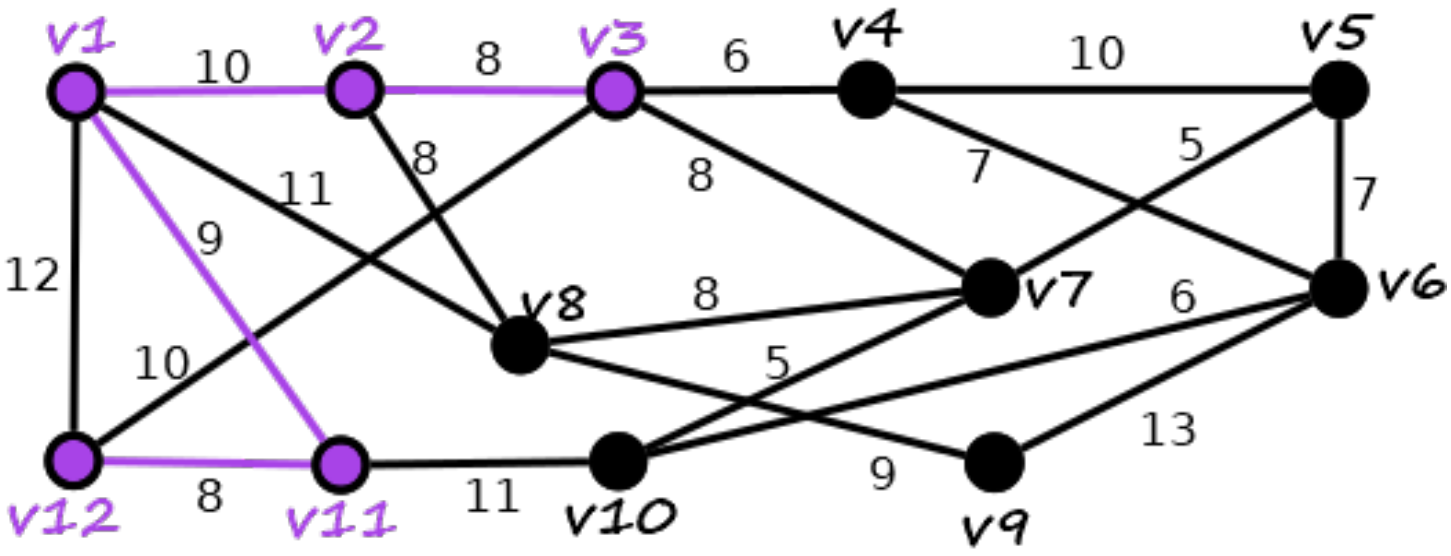
Árbol generador mínimo - Algoritmo de Prim

Iteración k: Subgrafo de un AGM conexo con k aristas



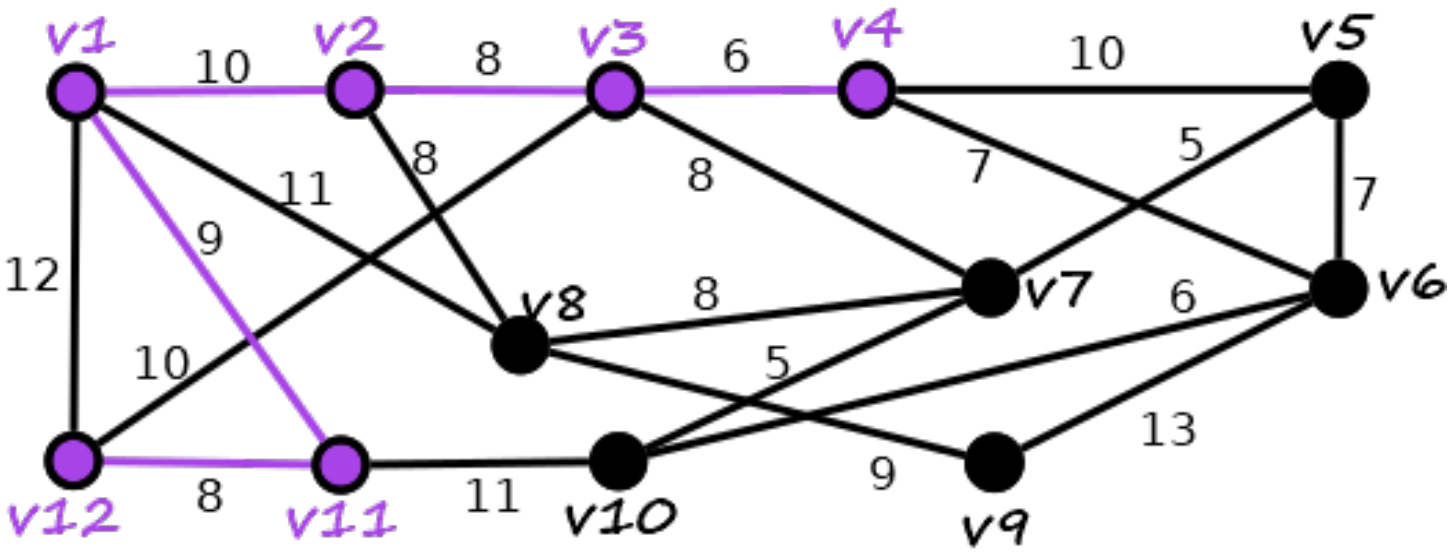
Árbol generador mínimo - Algoritmo de Prim

Iteración k: Subgrafo de un AGM conexo con k aristas



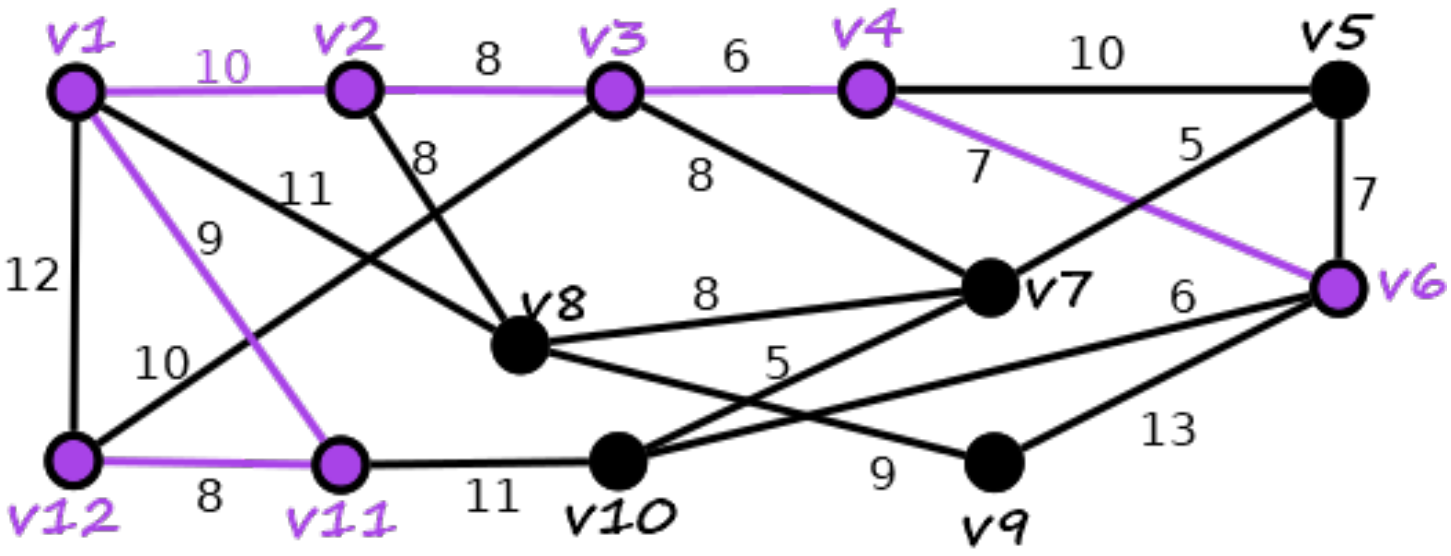
Árbol generador mínimo - Algoritmo de Prim

Iteración k: Subgrafo de un AGM conexo con k aristas



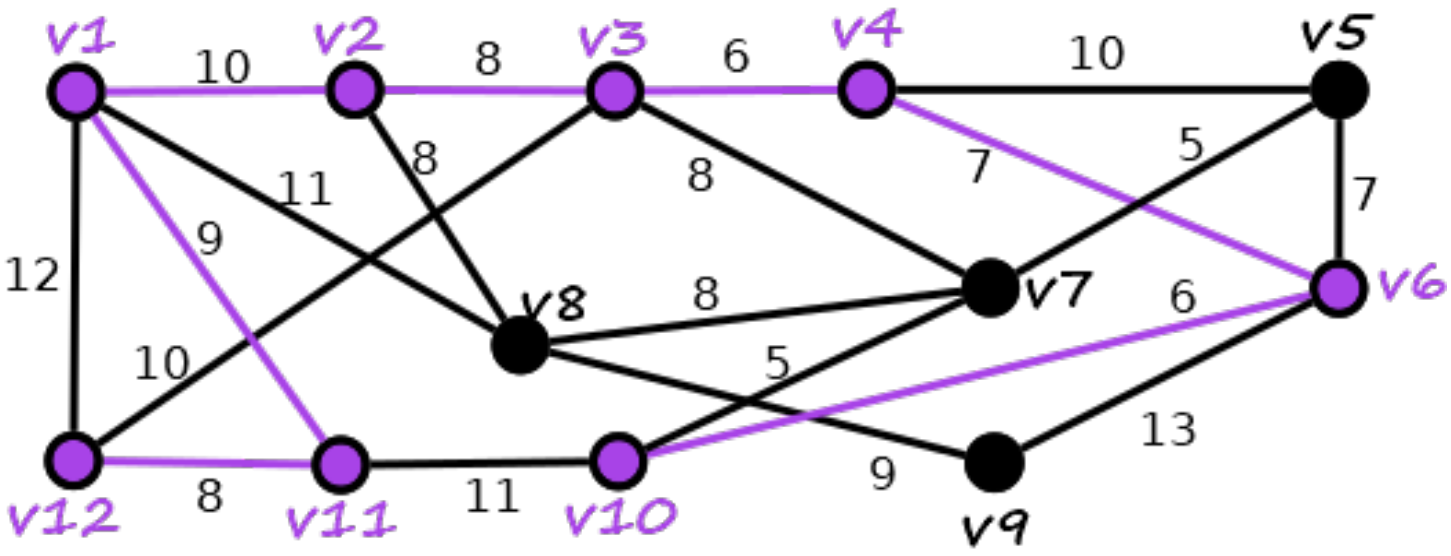
Árbol generador mínimo - Algoritmo de Prim

Iteración k: Subgrafo de un AGM conexo con k aristas



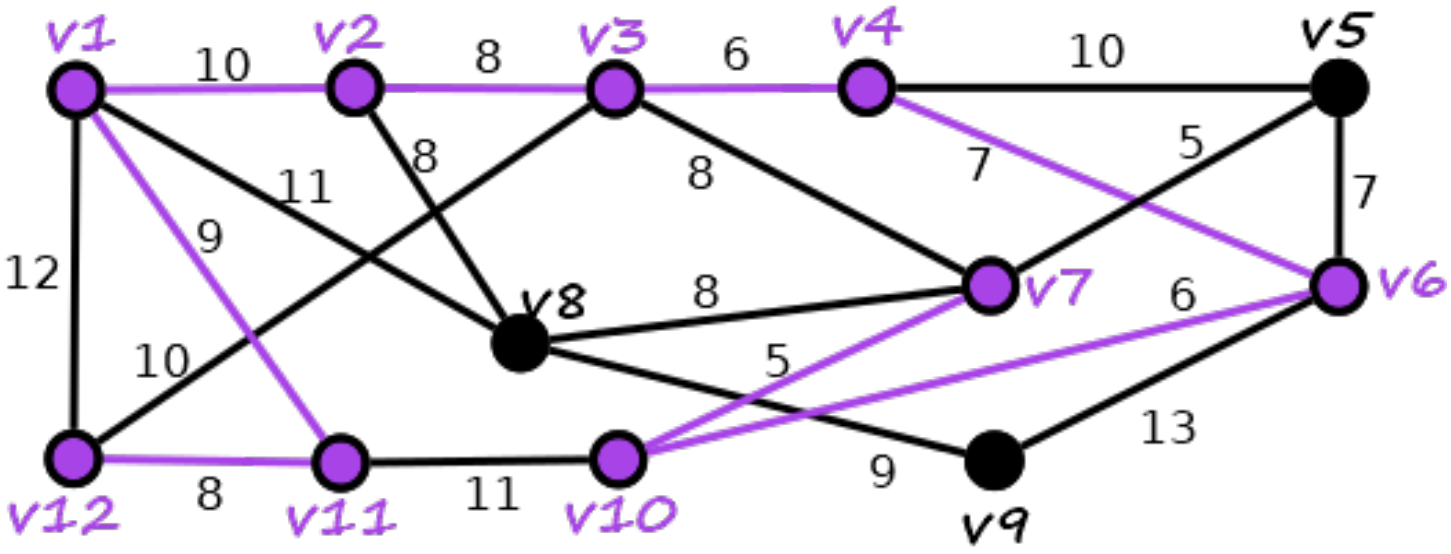
Árbol generador mínimo - Algoritmo de Prim

Iteración k: Subgrafo de un AGM conexo con k aristas



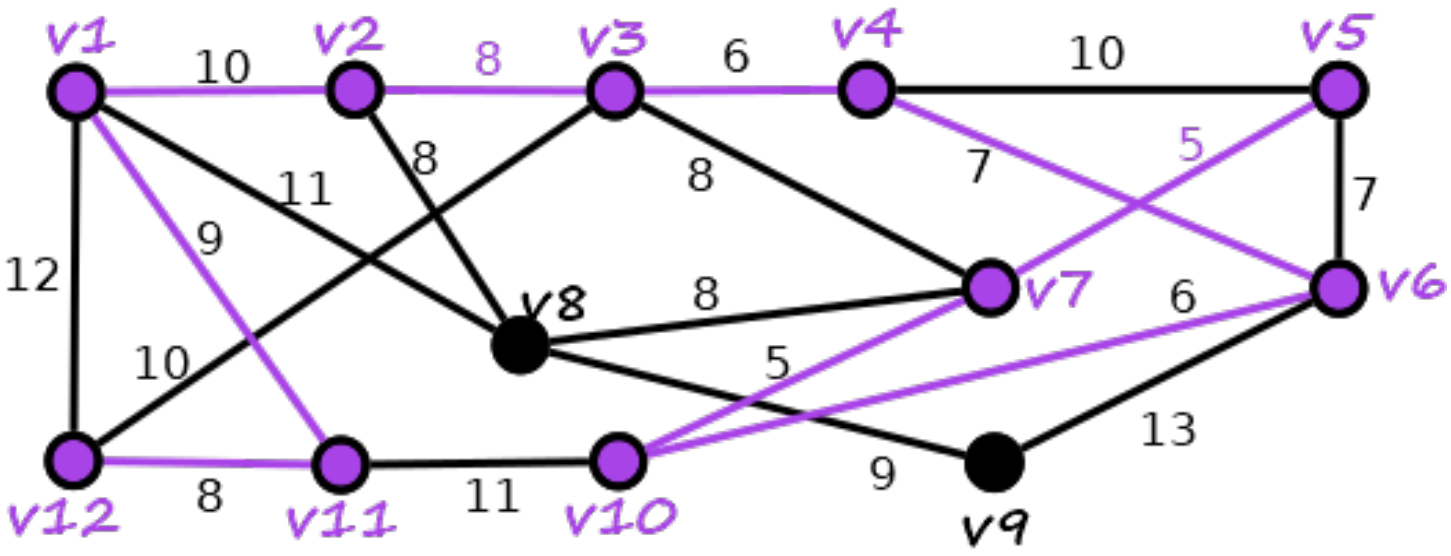
Árbol generador mínimo - Algoritmo de Prim

Iteración k: Subgrafo de un AGM conexo con k aristas



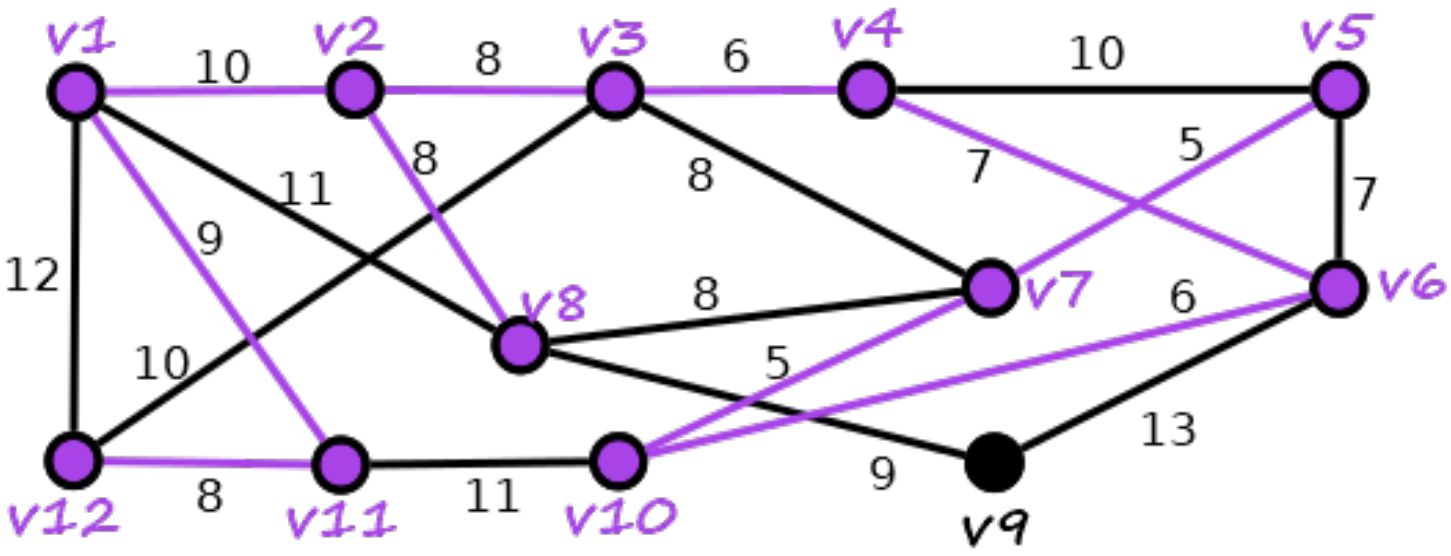
Árbol generador mínimo - Algoritmo de Prim

Iteración k: Subgrafo de un AGM conexo con k aristas



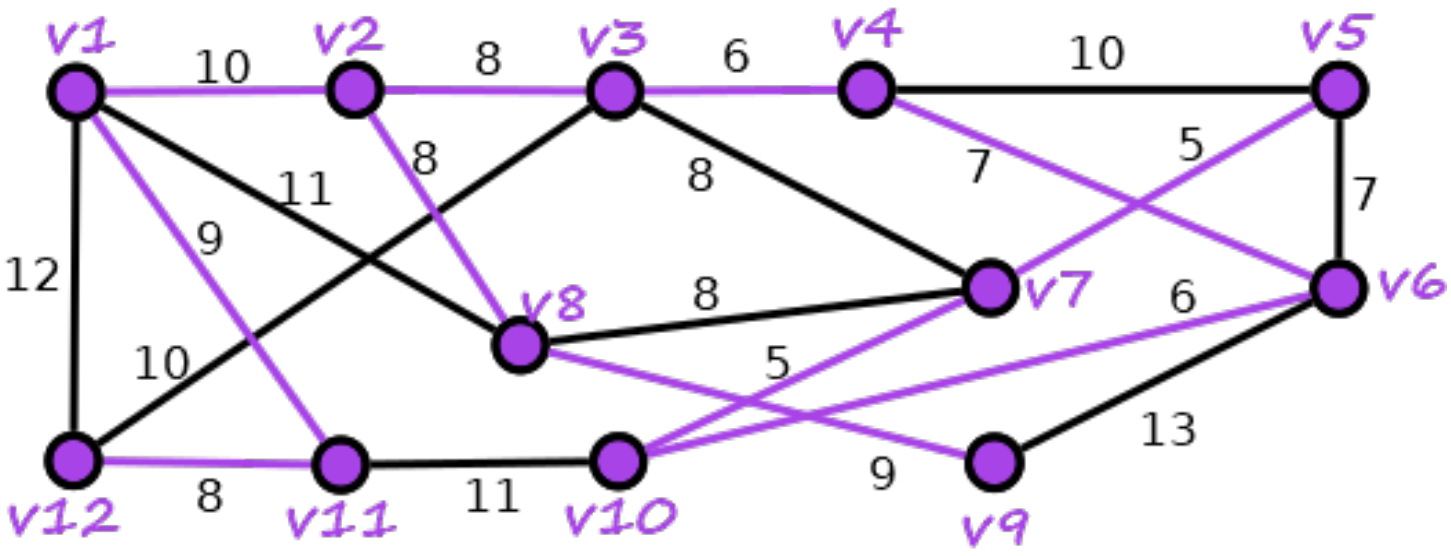
Árbol generador mínimo - Algoritmo de Prim

Iteración k: Subgrafo de un AGM conexo con k aristas



Árbol generador mínimo - Algoritmo de Prim

Iteración k: Subgrafo de un AGM conexo con k aristas



Árbol generador mínimo - Algoritmo de Prim

Prim(G)

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

$V_T \leftarrow \{u\}$ *(cualquier vertice)*
$X_T \leftarrow \emptyset$
$i \leftarrow 1$
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 $T = (V_T, X_T)$

Árbol generador mínimo - Algoritmo de Prim

Proposición: 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.

Para organizar la demostración, usaremos el siguiente lema.

Lema: Sea $T = (V, X_T)$ un árbol generador de $G = (V, X)$. Si $e \in X \setminus X_T$ y $f \in X_T$ una arista del ciclo de $T + e$, entonces $T' = T + e - f = (V, X_T \cup \{e\} \setminus \{f\})$ es árbol generador de G.

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

Árbol generador mínimo - Algoritmo de Kruskal (1956)

- Ordena las aristas del grafo de forma creciente según su peso.
- En cada paso elege la siguiente arista que no forme circuito con las aristas ya elegidas.
- El algoritmo para cuando selecciona $n - 1$ aristas.
- También es un algoritmo goloso.

Árbol generador mínimo - Algoritmo de Kruskal

Kruskal(G)

entrada: $G = (V, X)$ de n vertices y $l : X \to \mathbb{R}$

salida: $T =$ un AGM de G

$X_T \leftarrow \emptyset$

$i \leftarrow 1$

mientras $i \leq n - 1$ **hacer**

$e \leftarrow \arg\min\{l(e), e$ no forma circuito con las aristas de $X_T\}$

$X_T \leftarrow X_T \cup \{e\}$

$i \leftarrow i + 1$

fin mientras

retornar $T = (V, X_T)$

Árbol generador mínimo - Algoritmo de Kruskal

- Al comenzar el algoritmo, cuando todavía no se seleccionó arista alguna, cada vértice del grafo forma una componente conexa distinta

- En la primera iteración, los dos vértices extremo de la arista seleccionada van a pasar a formar una única componente conexa del nuevo bosque.

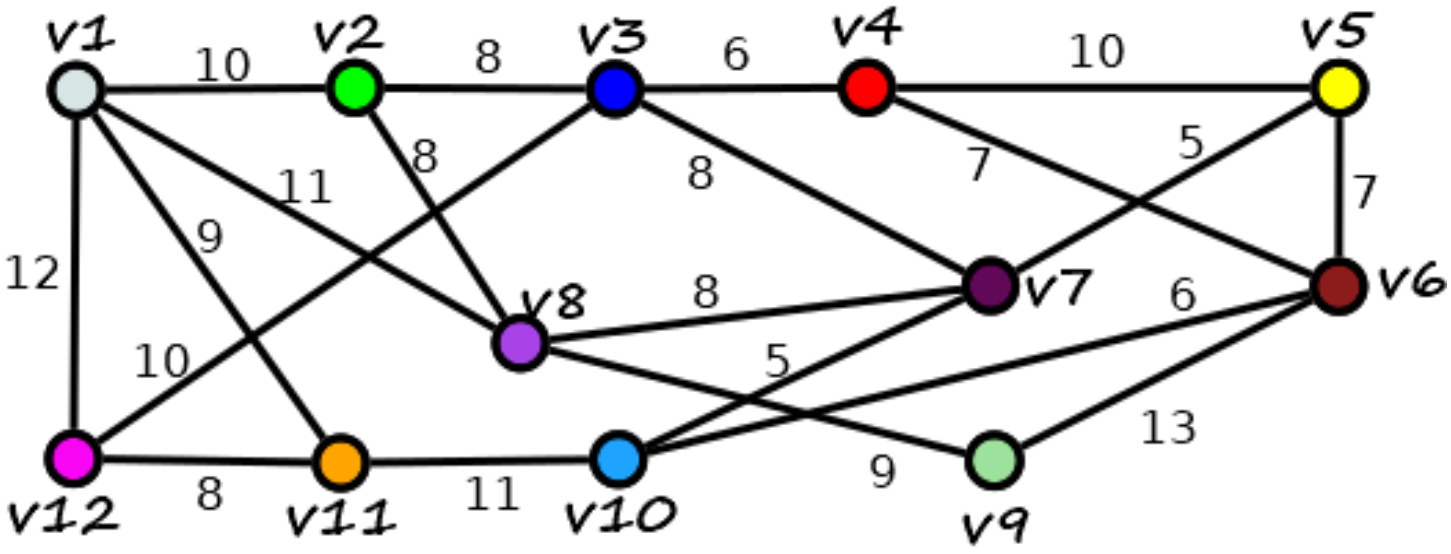
- Así, en cada iteración, si se elige la arista (u, w), se unen las componentes conexas de u y la de w.

- 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.

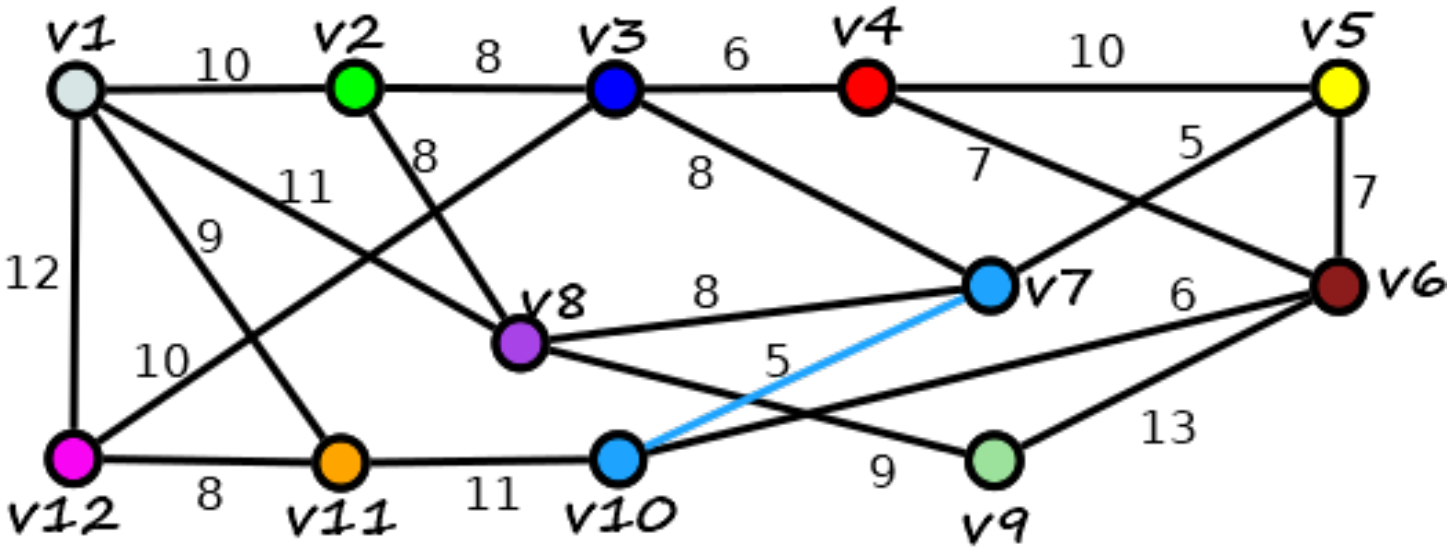
Árbol generador mínimo - Algoritmo de Kruskal

Iteración k: Subgrafo de un AGM con k aristas sin ciclos



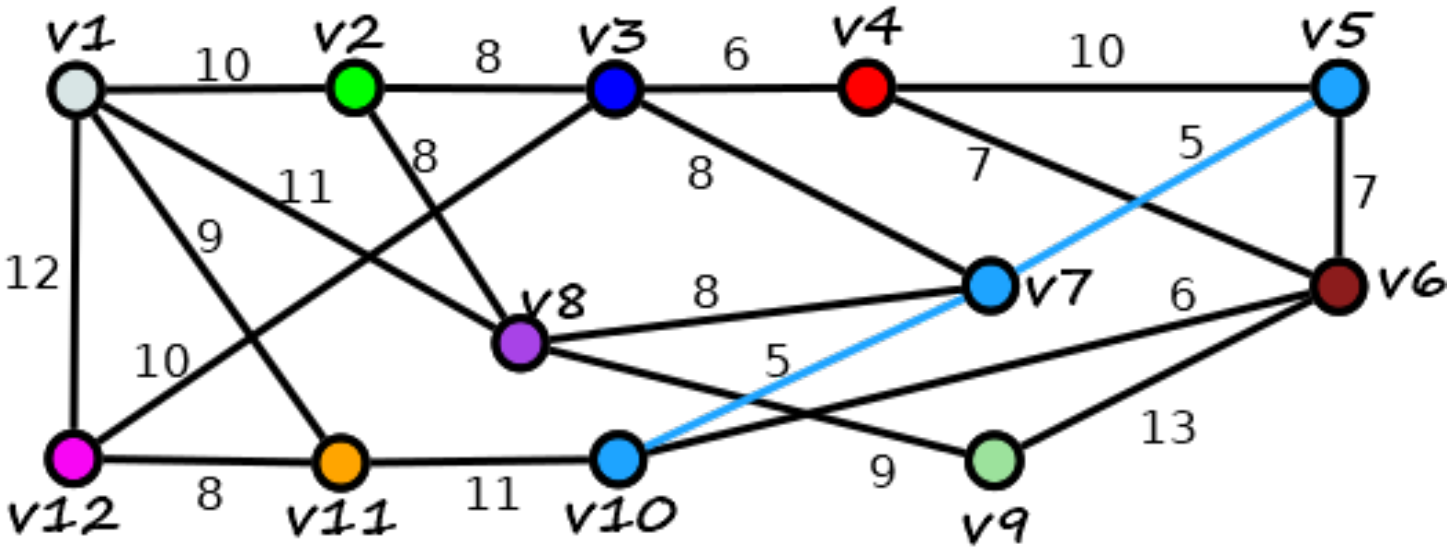
Árbol generador mínimo - Algoritmo de Kruskal

Iteración k: Subgrafo de un AGM con k aristas sin ciclos



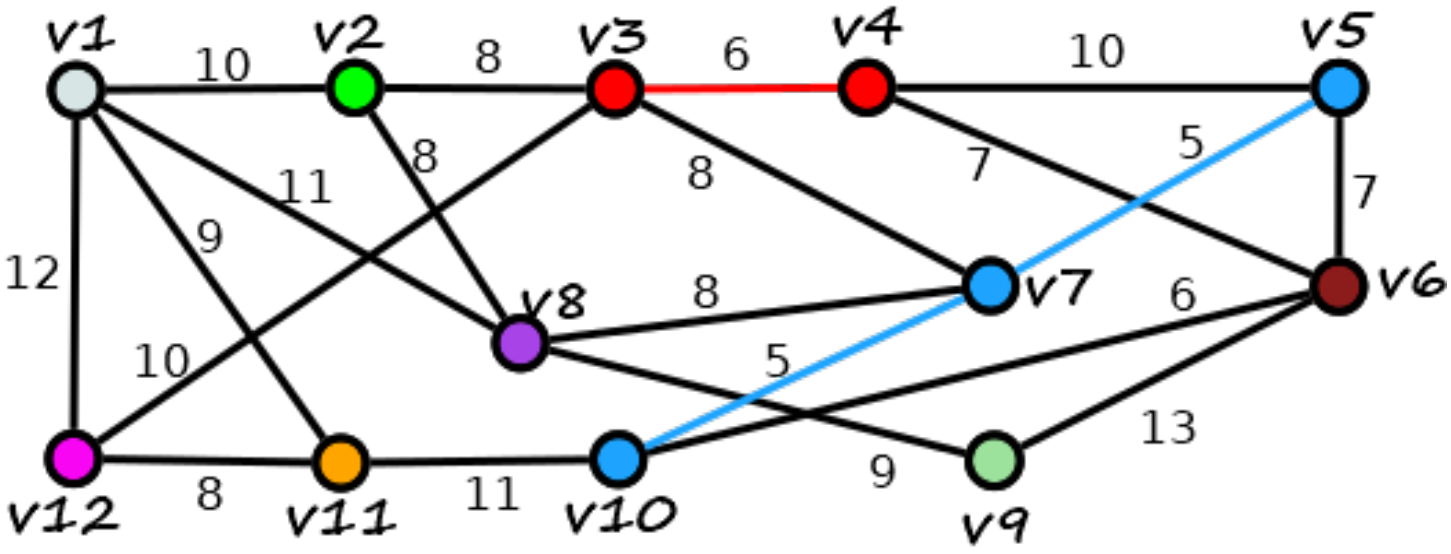
Árbol generador mínimo - Algoritmo de Kruskal

Iteración k: Subgrafo de un AGM con k aristas sin ciclos



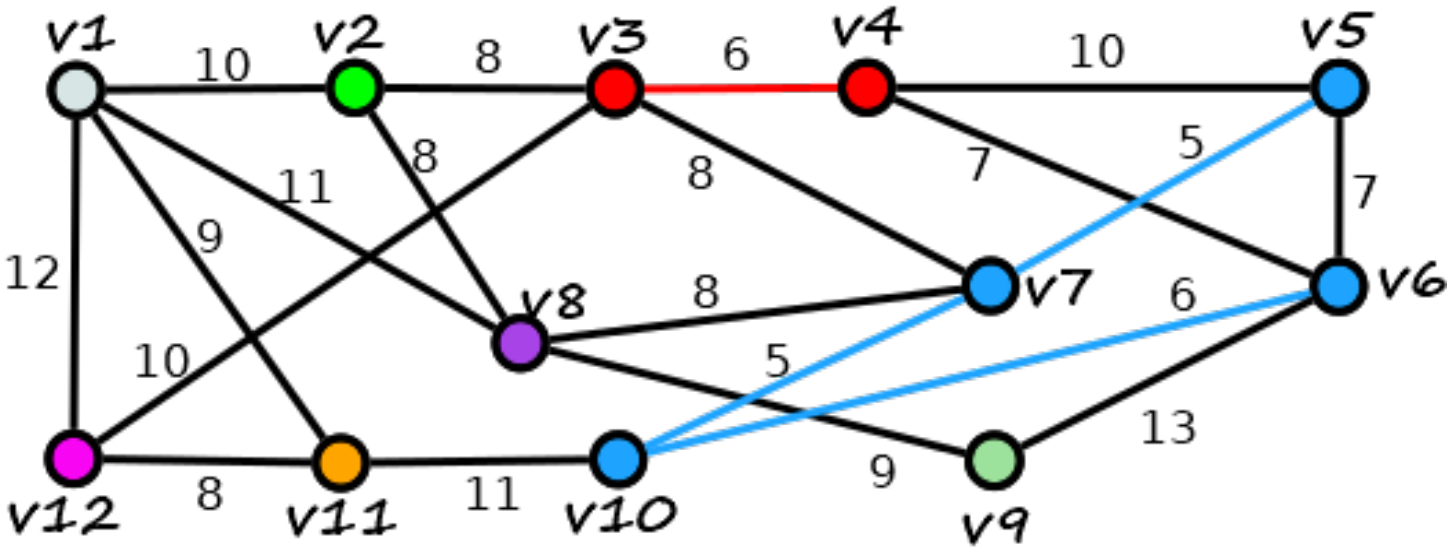
Árbol generador mínimo - Algoritmo de Kruskal

Iteración k: Subgrafo de un AGM con k aristas sin ciclos



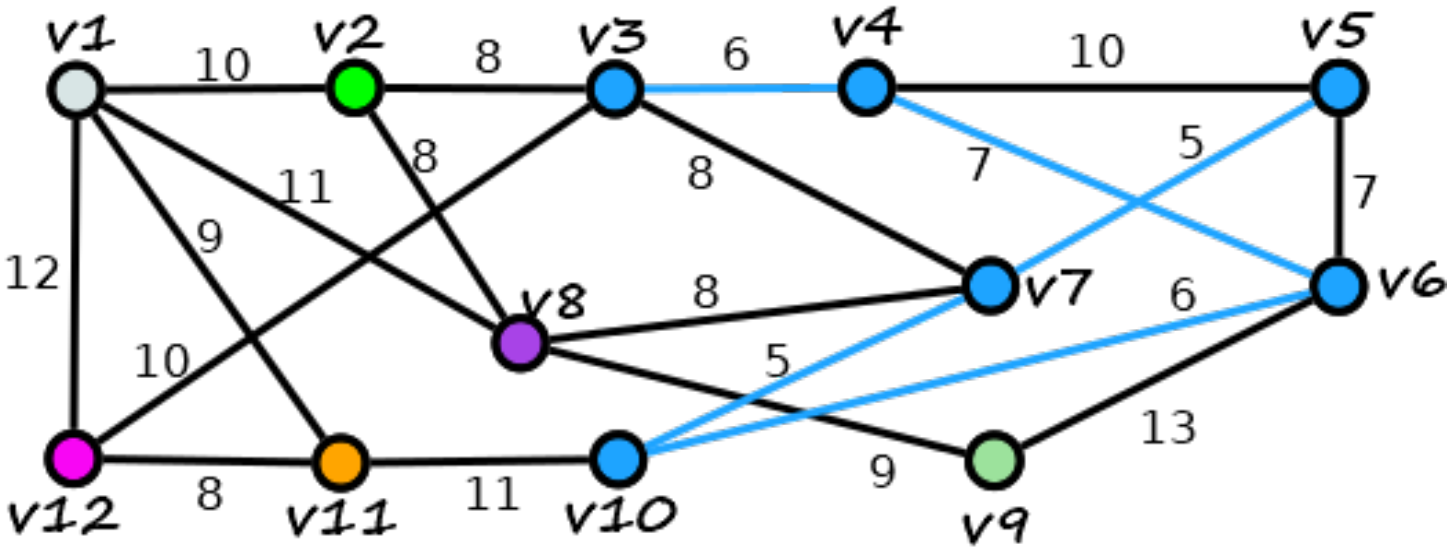
Árbol generador mínimo - Algoritmo de Kruskal

Iteración k: Subgrafo de un AGM con k aristas sin ciclos



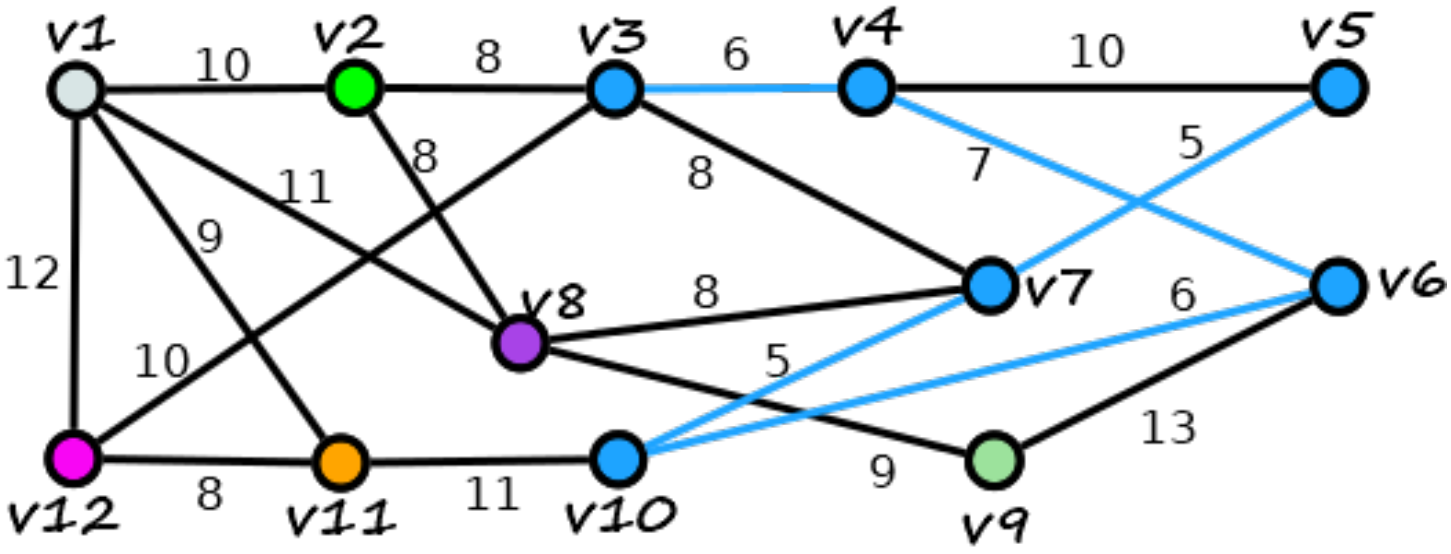
Árbol generador mínimo - Algoritmo de Kruskal

Iteración k: Subgrafo de un AGM con k aristas sin ciclos



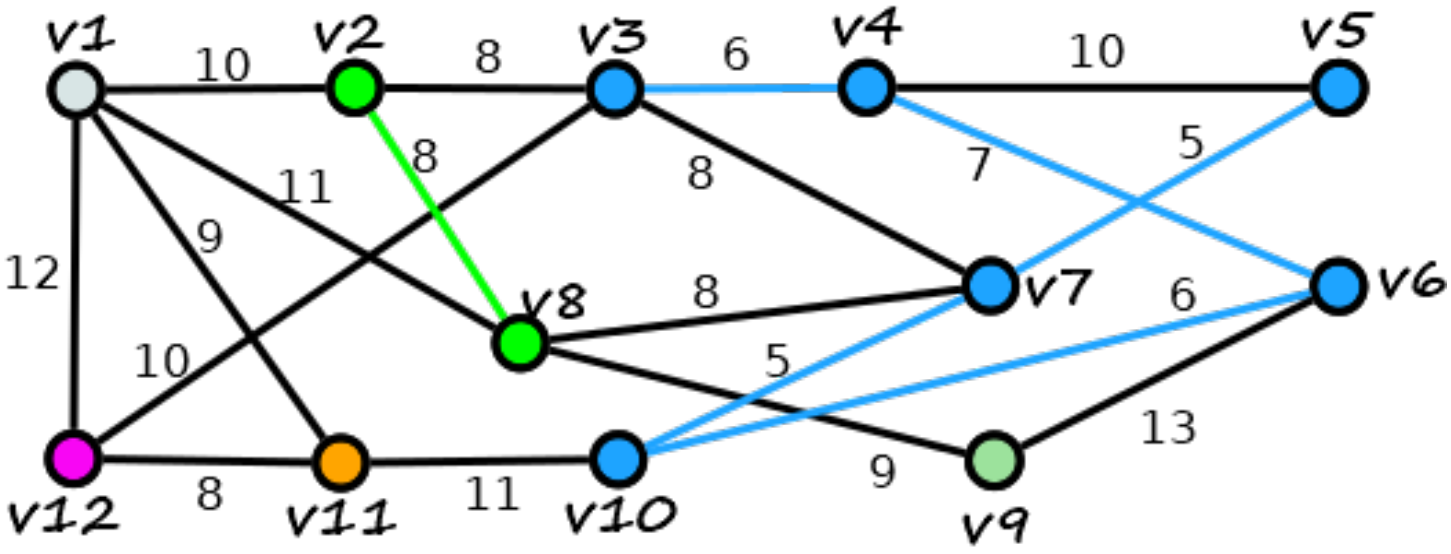
Árbol generador mínimo - Algoritmo de Kruskal

Iteración k: Subgrafo de un AGM con k aristas sin ciclos



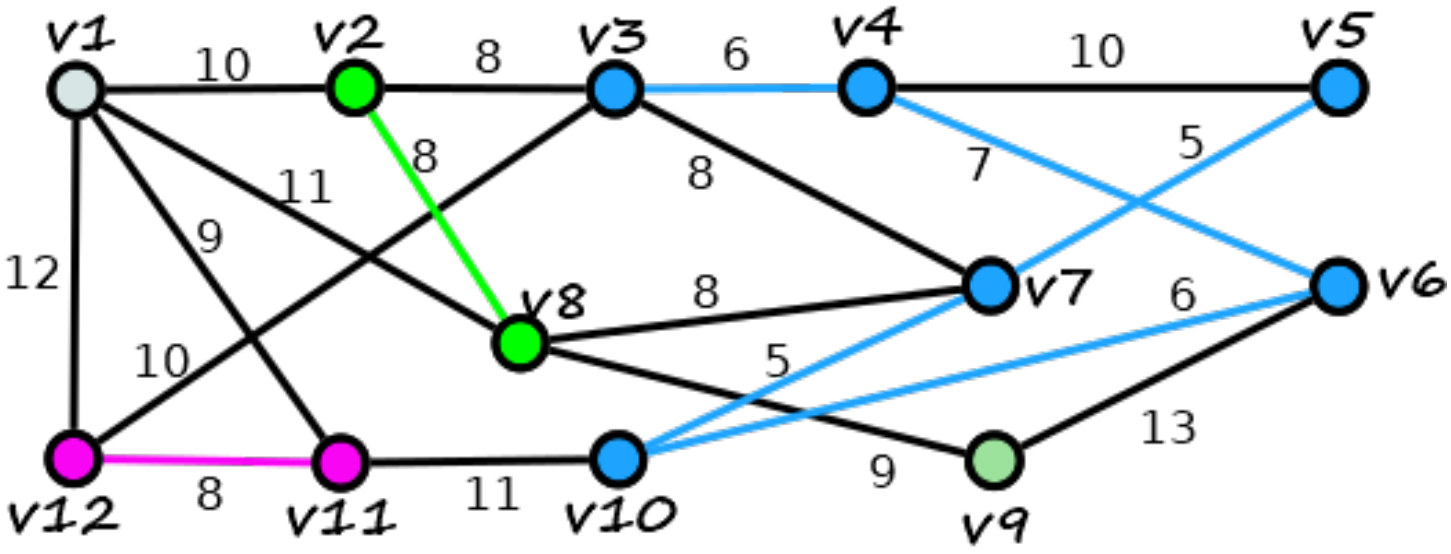
Árbol generador mínimo - Algoritmo de Kruskal

Iteración k: Subgrafo de un AGM con k aristas sin ciclos



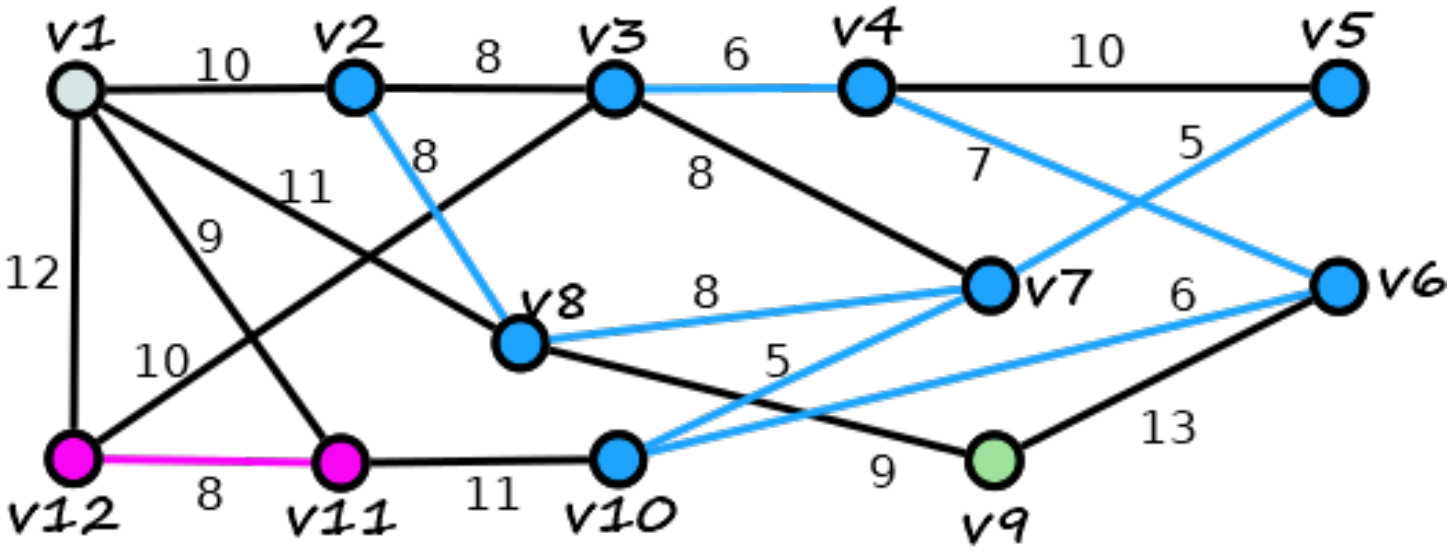
Árbol generador mínimo - Algoritmo de Kruskal

Iteración k: Subgrafo de un AGM con k aristas sin ciclos



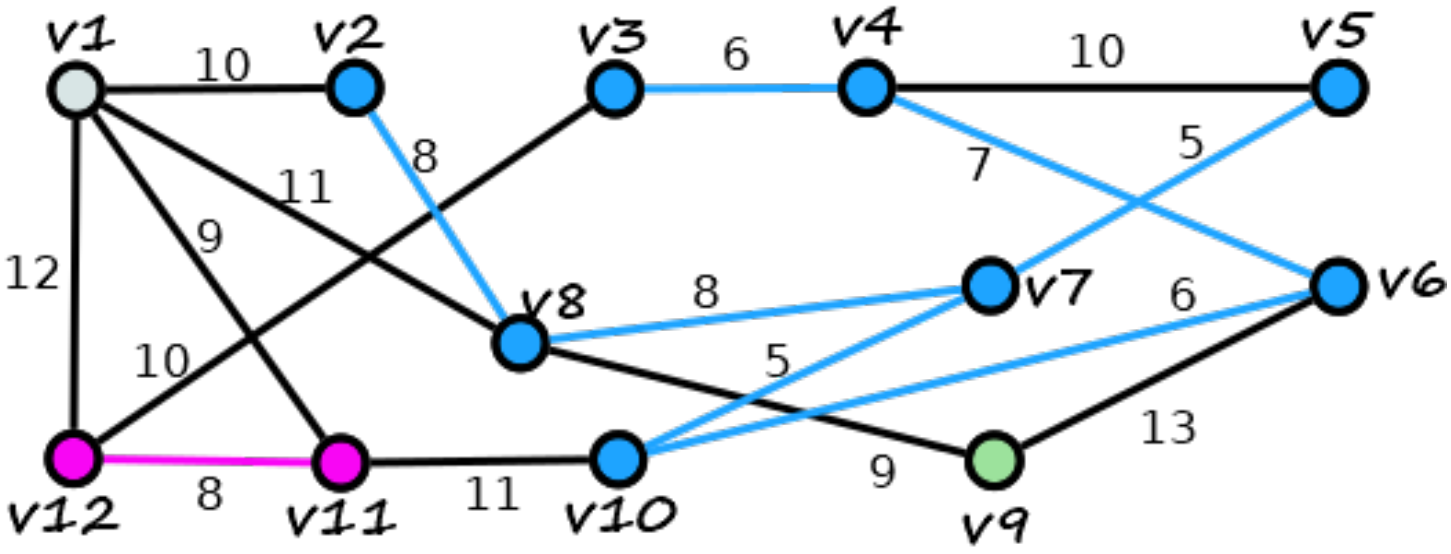
Árbol generador mínimo - Algoritmo de Kruskal

Iteración k: Subgrafo de un AGM con k aristas sin ciclos



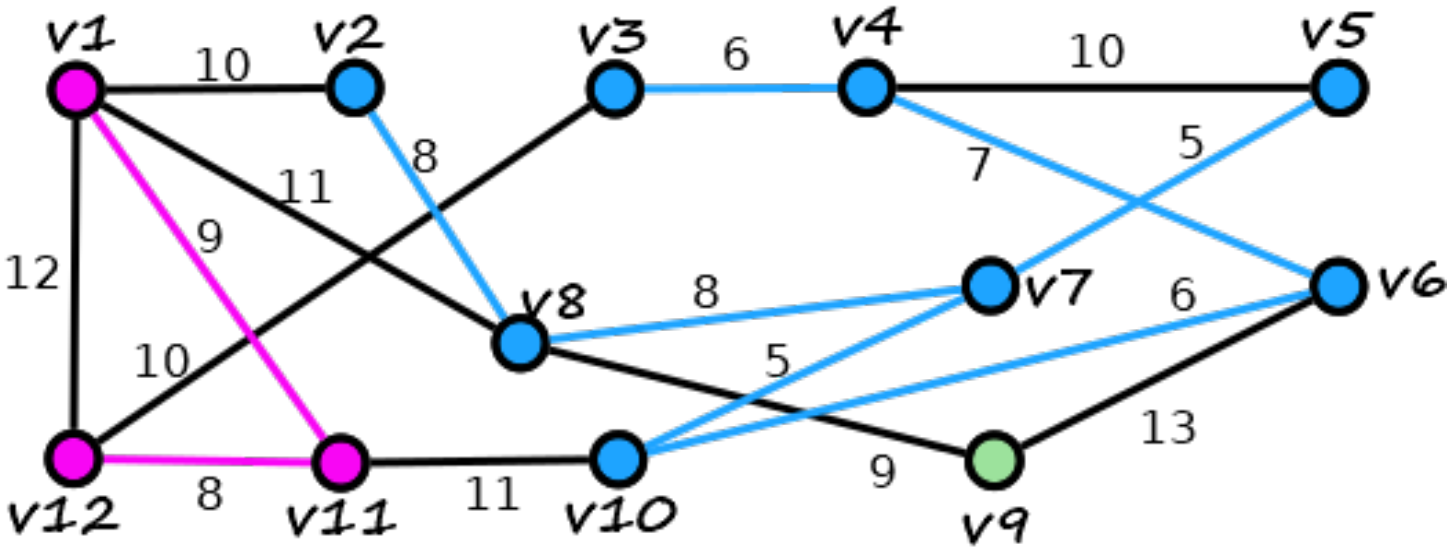
Árbol generador mínimo - Algoritmo de Kruskal

Iteración k: Subgrafo de un AGM con k aristas sin ciclos



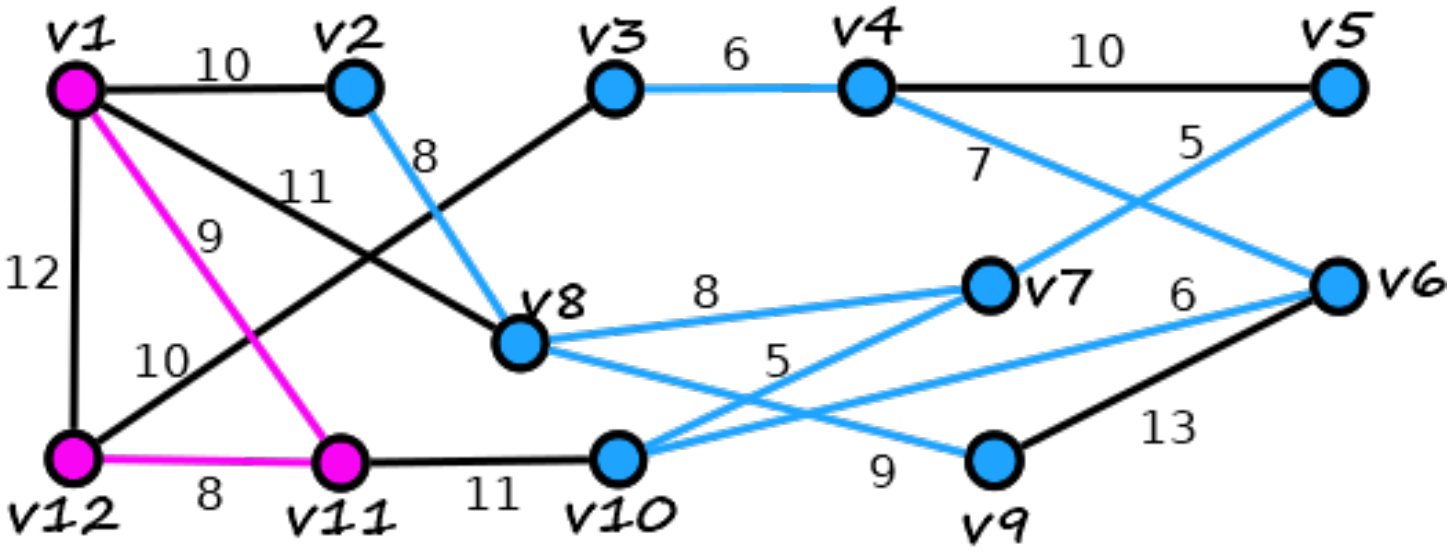
Árbol generador mínimo - Algoritmo de Kruskal

Iteración k: Subgrafo de un AGM con k aristas sin ciclos



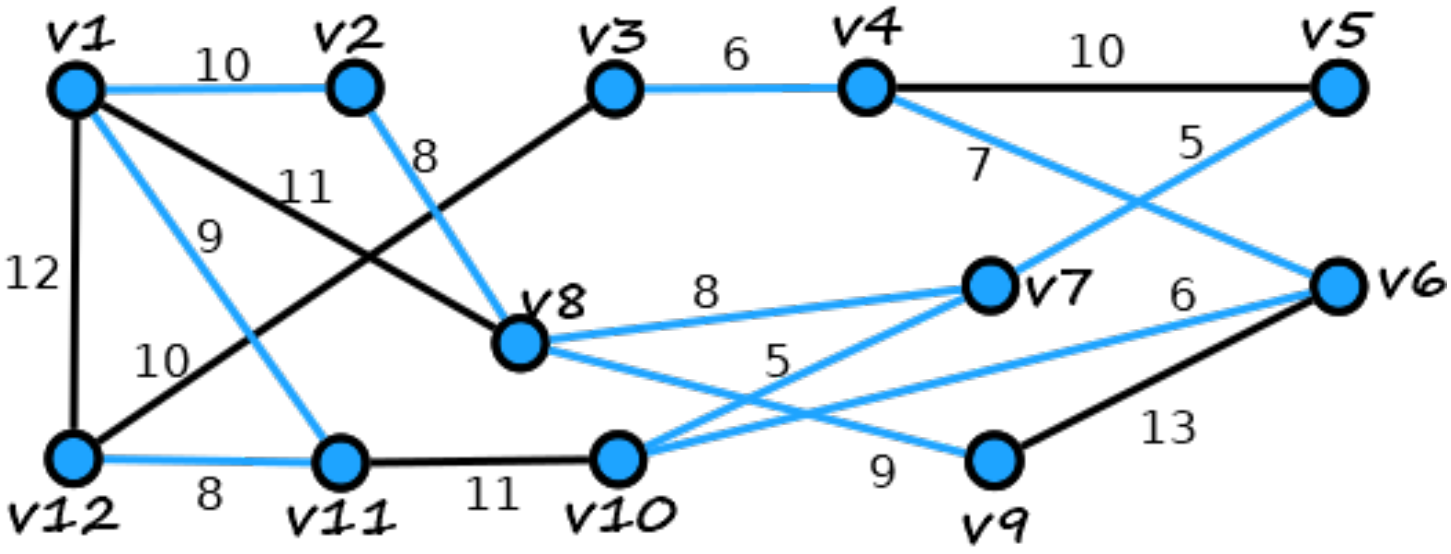
Árbol generador mínimo - Algoritmo de Kruskal

Iteración k: Subgrafo de un AGM con k aristas sin ciclos



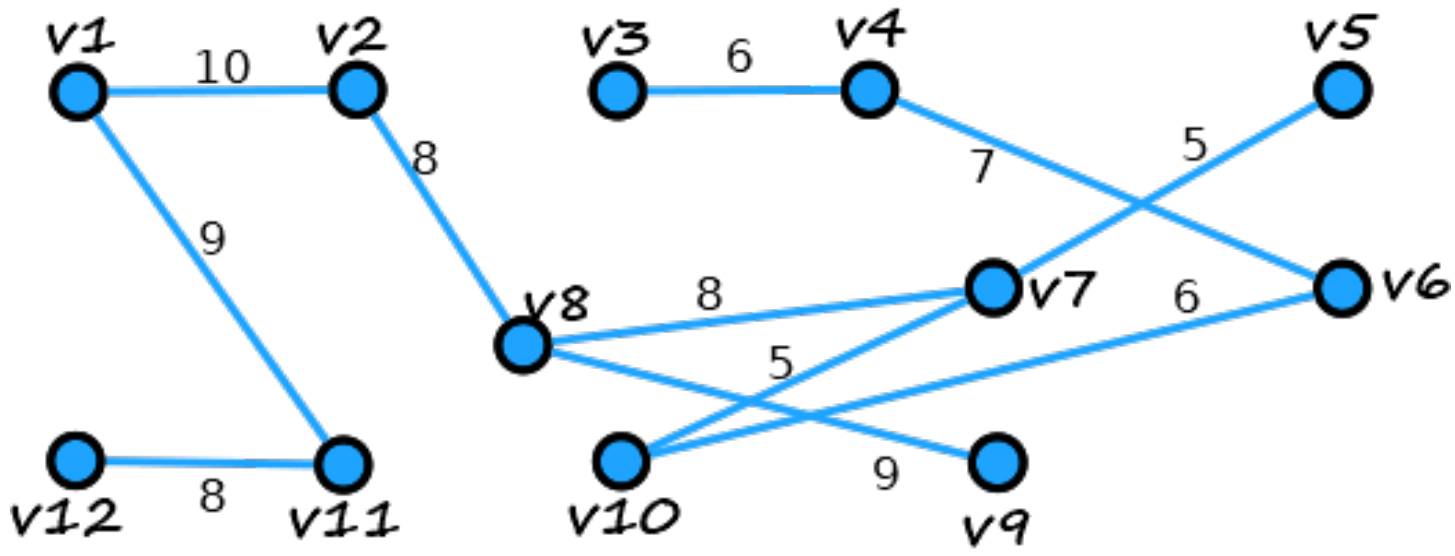
Árbol generador mínimo - Algoritmo de Kruskal

Iteración k: Subgrafo de un AGM con k aristas sin ciclos



Árbol generador mínimo - Algoritmo de Kruskal

Iteración k: Subgrafo de un AGM con k aristas sin ciclos



Árbol generador mínimo - Algoritmo de Kruskal

Kruskal(G)

entrada: $G = (V, X)$ de n vertices y $l : X \to \mathbb{R}$

salida: $T =$ un AGM de G

$X_T \leftarrow \emptyset$

$Cand \leftarrow X$

ordenar$(Cand)$

$i \leftarrow 1$

mientras $i \leq n - 1$ **hacer**

  $(u, w) \leftarrow$ mín$(Cand)$

  $Cand \leftarrow Cand \setminus \{(u, w)\}$

  si u y w no pertenecen a la misma comp conexa **hacer**

    $X_T \leftarrow X_T \cup \{(u, w)\}$

    $i \leftarrow i + 1$

  fin si

fin mientras

retornar $T = (V, X_T)$