

Arboles de expansión

M. C Jorge García.

Imaginemos que tenemos un conjunto de ciudades, deseamos construir un sistema de carreteras tal que todas las ciudades estén conectadas entre ellas. Pero al mismo tiempo, deseamos que el costo de construir dicho sistema sea el mínimo posible.

Definición: (El conector mínimo)

Sea una red formada por un grafo conexo G y una función de costo $f : E(G) \rightarrow \mathbb{R}^+$. Deseamos encontrar un subgrafo de expansión T de G tal que el costo $\sum_{\forall e_i \in T} f(e_i)$ sea mínimo.

- Si $v_i v_j = e \notin E(G)$ podemos asignarle $f(e) = \infty$ y así trabajar con un grafo completo K_p donde $p = |V(G)|$.

- Claramente el subgrafo T de G que resuelve el conector mínimo debe ser un árbol.

Definición: (Árbol de expansión)

Sea G un grafo, si un subgrafo de expansión T de G es un árbol, entonces a T se le llama un **árbol de expansión** de G .

- Ahora sabemos que el problema del conector mínimo es el problema de encontrar el árbol de expansión T de G costo mínimo. Al árbol T le llamamos el **mínimo árbol de expansión**.

- 1 Supongamos que etiquetamos las aristas de $E(G)$ como e_1, e_2, \dots, e_q de tal manera que: $f(e_1) \leq f(e_2) \leq \dots \leq f(e_q)$.
- 2 Ahora empezamos con los $V(G)$ y las añadimos la arista e_1 .
- 3 Ahora añadimos la siguiente arista e_i tal que al añadirla a las aristas que ya llevamos no forme un ciclo. Si e_i formara un ciclo intentamos con e_{i+1} y así sucesivamente.
- 4 Continuamos este proceso hasta que tengamos las $q = p - 1$ aristas necesarias para un árbol.
- 5 Al árbol producido por este proceso le llamamos un **árbol económico**.

Teorema:

Sean G y f una red, y sea T un árbol económico de G . Entonces T es un mínimo árbol de expansión en G .

El algoritmo de Prim I

Entradas: Un grafo G una función de costo f y un vértice de inicio $r \in V(G)$.

Salidas: Un subgrafo $T \subseteq G$ que forma el mínimo árbol de expansión de G .

```
1:  $V(T) = \{r\}$  y  $E(T) = \emptyset$ 
2: while  $V(T) \neq V(G)$  do
3:    $\min = \infty$ 
4:   for all  $v \in V(T)$  do
5:     for all  $u \notin V(T)$  and  $u \in V(G)$  do
6:       if  $f(uv) < \min$  then
7:          $\min = f(uv)$ 
8:          $\text{vertice} = u$ 
9:          $\text{arista} = uv$ 
10:      end if
11:    end for
12:  end for
```

13: $V(T) = V(T) \cup \{ \text{vertice} \}$

14: $E(T) = E(T) \cup \{ \text{arista} \}$

15: **end while**

- Por convención en esta implementación $f(uv) = \infty$ si $uv \notin E(G)$
- Esta implementación del algoritmo de Prim toma $O(n^3)$ donde $n = |V(G)|$. Por lo tanto **no es óptima**
- Se puede mejorar la manera de buscar el vértice y la arista candidata.
- Si se guarda el grafo G con una lista de adyacencia y se guardan las posibles aristas en un *fibonacci heap*, se puede implementar de manera que $O(|E(G)| + |V| \lg |V|)$.

Ejemplo

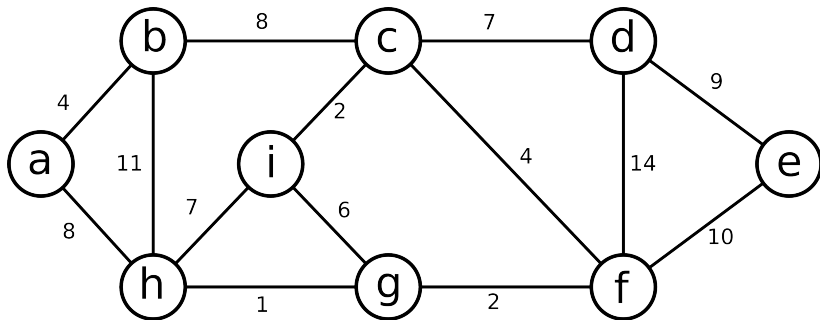


Figura: Calcular el árbol de expansión mínimo de G