13. ⋆Una empresa de comunicaciones modela su red usando un grafo G donde cada arista tiene una capacidad positiva que representa su *ancho de banda*. El *ancho de banda* de la red es el máximo k tal que G_k es conexo, donde G_k es el subgrafo generador de G que se obtiene de eliminar las aristas de peso menor a k (Figura 1).

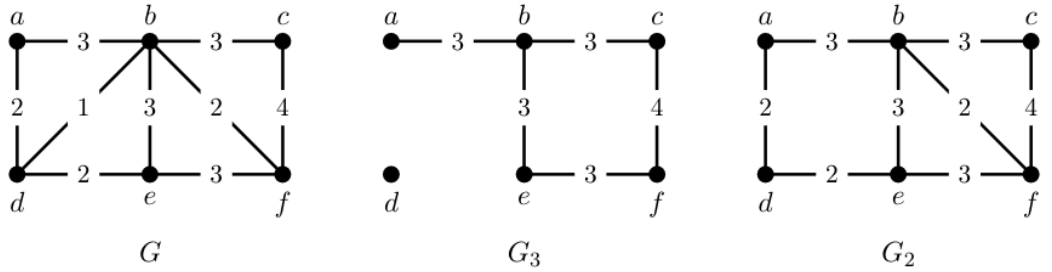

Figura 1. *El grafo G tiene ancho de banda 2 porque G_2 es conexo y G_3 no. Por otra parte, el ancho de banda del camino c, b, d es 1 mientras que el ancho de banda del camino c, b, e, d es 2. En general,* $\text{bwd}(c, d) = 2$ *mientras que* $\text{bwd}(a, e) = \text{bwd}(b, f) = 3$.

---

a) Proponer un algoritmo eficiente para determinar el ancho de banda de una red dada.

```
Algoritmo1(G):
  Para cada arista e ∈ E(G): //Doy vuelta los pesos para buscar el AGMáx, O(m)
    c(e) = -c(e)

  AGM <- Prim(G) //O(m+n log n)

  //Recupero los pesos originales, es AGM, hay n-1 aristas, n nodos, O(n)
  Para cada v en AGM:
    Para cada w en AGM[v]:
      c(v,w) = -c(v,w)

  minimo = inf
  Para cada v en AGM:
    Para cada w en AGM[v]:
      si c(v,w) < minimo:
        minimo := c(v,w)

  return minimo

La complejidad final es O(n+m+n log n) ∈ O(m + n log n)
```

La empresa está dispuesta a hacer una inversión que consiste en actualizar algunos enlaces (aristas) a un ancho de banda que, para la tecnología existente, es virtualmente infinito. Antes de decidir la inversión, quieren determinar cuál es el ancho de banda que se podría obtener si se reemplazan i aristas para todo $0 \leq i < n$.

b) Proponer un algoritmo que dado G determine el vector $a_0, \ldots, a_{n-1}$ tal que a_i es el ancho de banda máximo que se puede obtener si se reemplazan i aristas de G.

```
Algoritmo2(G):
  Para cada arista e ∈ E(G): //Doy vuelta los pesos para buscar el AGMáx, O(m)
    c(e) = -c(e)
```

```
AGM <- Prim(G) //O(m+n log n)

//Veo para toda arista del AGM, hay n-1 aristas, n nodos, O(n)
Para cada v en AGM:
  Para cada w en AGM[v]:
    c(v,w) = -c(v,w) //recupero el peso original

pesos_AGM = []

Para cada v en AGM:
  Para cada w en AGM[v]:
    pesos_AGM.agregar(c(v,w))

sol <- sort(pesos_AGM)

sol.agregar(inf)

return sol
```

Lo que habría que ver es lo siguiente, es bastante sencilla la inducción, pero como el enunciado no lo pide, no lo voy a probar. (`sol` es 0-indexed).

$$\forall i, 0 \leq i < n \underset{L}{\rightarrow} a_i = \begin{cases} \text{sol}[i] & \text{si } i < n-1 \\ \infty & \text{si } i = n-1 \end{cases}$$

Plantear:

- caso base $k = 0$
- caso inductivo $1 \leq k < n-1$
- caso $k = n-1$