*Prim* (Grafo *G*)

*// Inicializamos todos los nodos del grafo. La distancia la ponemos a infinito y el padre de cada nodo a NULL*

*// Encolamos, en una cola de prioridad donde la prioridad es la distancia, todas las parejas <nodo,distancia> del grafo*

**por cada** *u* en *V[G]* **hacer**

distancia[*u*] = INFINITO

padre[*u*] = NULL

Añadir(cola,<*u*,distancia[*u*]>)

distancia[*u*]=0

**mientras** !esta\_vacia(cola) **hacer**

*// OJO: Se entiende por mayor prioridad aquel nodo cuya distancia[u] es menor.*

*u* = extraer\_minimo(cola) //devuelve el minimo y lo elimina de la cola.

**por cada** **v** adyacente a '**u'** **hacer**

**si** ((*v* ∈ cola) **&&** (distancia[*v*] > peso(u, v)) **entonces**

padre[*v*] = *u*

distancia[*v*] = peso(u, v)

Actualizar(cola,<*v*,distancia[*v*]>)

**DIJKSTRA** (Grafo *G*, nodo\_fuente *s*)

**para** *u* ∈ *V[G]* **hacer**

distancia[*u*] = INFINITO

padre[*u*] = NULL

distancia[*s*] = 0

adicionar (cola, (s,distancia[*s*]))

**mientras que** cola no es vacía **hacer**

*u* = extraer\_minimo(cola)

**para** todos *v* ∈ adyacencia[*u*] **hacer**

**si** distancia[*v*] > distancia[*u*] + peso (u, v) **hacer**

distancia[*v*] = distancia[*u*] + peso (u, v)

padre[*v*] = *u*

adicionar(cola,(v,distancia[*v*]))

1 **function** Kruskal(*G*)

2 **for each** vertex *v* in *G* do

3 Define an elementary cluster *C*(*v*) ← {*v*}.

4 Initialize a priority queue *Q* to contain all edges in *G*, using the weights as keys.

5 Define a tree *T* ← Ø //*T* will ultimately contain the edges of the MST

6 // n es el número total de vértices

7 **while** *T* has fewer than *n*-1 edges **do**

8 // edge u, v is the minimum weighted route from/to v

9 (*u*,*v*) ← *Q*.removeMin()

10 // previene ciclos en T. suma u, v solo si T no contiene una arista que una u y v.

11 // Nótese que el cluster contiene más de un vértice si una arista une un par de

12 // vértices que han sido añadidos al árbol.

13 Let *C*(*v*) be the cluster containing *v*, and let *C*(*u*) be the cluster containing *u*.

14 **if** *C*(*v*) ≠ *C*(*u*) **then**

15 Add edge (*v*,*u*) to *T*.

16 Merge *C*(*v*) and *C*(*u*) into one cluster, that is, union *C*(*v*) and *C*(*u*).

17 **return** tree *T*