Pseudocódigo Krustkal

```
void krustkal (vector<Conection> conexiones) {

    //ordenar conexiones por costo
    conexiones.sorto(costo)

    int costoTotal = 0
    int cont = 0 // contador de edges
    CycleDetector cd (vector<int> nodos) // Clase para detectar ciclos

    //Para cada conexión revisar si se pueden conectar los nodos
    for (int i=0; i<conexiones.size(); i++) {
      int n1 = conexiones[i][0]
      int n2 = conexiones[i][1]
      int costo = conexiones[i][2]

      // Si tienen diferentes padres, conectar, ya que no se hace ciclo
      if (cd.getParent(n1) != cd.getParent(n2)) {

          IF (count == V-1) breaks
          else {
              cd.connect(n1, n2)
              costoTotal += costo
              cont++;
              T.push_back(conexiones[i]) //Guardar en mst
          }
      }
      print (conexiones, costoTotal)
}

class CycleDetector {                    ↑ Mayor rango = más hijos
private
    map<int, int> parent
    map<int, int> ranks // Cada nodo tiene rango=1 al inicio

public
    int getParent(n) {// Buscar padre de forma recursiva
      if (parent[n] == n) return n
      else getParent(parent[n])
    }
    void connect(n1, n2) {
        p1 = getParent(n1), p2 = getParent(n2)
        if (ranks[p1] < ranks[p2] parent[p1] = p2, ranks[p2]++

        else if (ranks[p1] > ranks[p2] parent[p2] = p1, ranks[p1]++

        else parent[p2] = parent[p1], ranks[p1]++
```