```haskell
-----------------------------------------------
-- Estructuras de Datos.  2018/19
-- 2º Curso del Grado en Ingeniería [Informática | del Software | de Computadores].
-- Escuela Técnica Superior de Ingeniería en Informática. UMA
--
-- Examen 4 de febrero de 2019
--
-- ALUMNO/NAME:
-- GRADO/STUDIES:
-- NÚM. MÁQUINA/MACHINE NUMBER:
--
-- Weighted Graph implemented by using a dictionary from
-- sources to another dictionary from destinations to weights
-----------------------------------------------

module DataStructures.Graph.DictionaryWeightedGraph
  ( WeightedGraph
  , WeightedEdge(WE)
  , empty
  , isEmpty
  , mkWeightedGraphEdges
  , addVertex
  , addEdge
  , vertices
  , numVertices
  , edges
  , numEdges
  , successors
  ) where

import Data.List(nub, intercalate)

import qualified DataStructures.Dictionary.AVLDictionary as D

data WeightedEdge a w  = WE a w a deriving Show

instance (Eq a, Eq w) => Eq (WeightedEdge a w) where
  WE u w v == WE u' w' v' = (u==u' && v==v' || u==v' && v==u')
                            && w == w'

instance (Eq a, Ord w) => Ord (WeightedEdge a w) where
  compare (WE _ w _) (WE _ w' _) = compare w w'

data WeightedGraph a w  = WG (D.Dictionary a (D.Dictionary a w))

empty :: WeightedGraph a w
empty = WG D.empty

addVertex :: (Ord a) => WeightedGraph a w -> a -> WeightedGraph a w
addVertex (WG d) vertex = WG (D.insert vertex D.empty d)

addEdge :: (Ord a, Show a) => WeightedGraph a w -> a -> a -> w -> WeightedGraph a w
addEdge (WG d) src dst w
  | not (D.isDefinedAt(src) d) || not (D.isDefinedAt(dst) d) = error "vértice no
encontrado"
  | otherwise = WG insertDST
    where
      insertDST = D.insert dst (D.insert src w (v)) insertSRC
      insertSRC = D.insert src (D.insert dst w (u)) d
      Just u = D.valueOf src d
      Just v = D.valueOf dst d

edges :: (Eq a, Eq w) => WeightedGraph a w -> [WeightedEdge a w]
edges  (WG d) = nub [(WE u w v) | (u, vs) <- D.keysValues d, (v, w) <- D.keysValues
vs]

successors :: (Ord a, Show a) => WeightedGraph a w -> a -> [(a,w)]
successors (WG d) vertex
  | not (D.isDefinedAt vertex d) = error "vértice no encontrado"
  | otherwise = D.keysValues (u)
    where
      Just u = D.valueOf vertex d


-- NO EDITAR A PARTIR DE AQUÍ
-- DON'T EDIT ANYTHING BELOW THIS COMMENT

vertices :: WeightedGraph a w -> [a]
```

```haskell
vertices (WG d) = D.keys d

isEmpty :: WeightedGraph a w -> Bool
isEmpty (WG d) = D.isEmpty d

mkWeightedGraphEdges :: (Ord a, Show a) => [a] -> [WeightedEdge a w] ->
WeightedGraph a w
mkWeightedGraphEdges vs es = wg'
  where
    wg = foldl addVertex empty vs
    wg' = foldr (\(WE u w v) wg -> addEdge wg u v w) wg es

numVertices :: WeightedGraph a w -> Int
numVertices = length . vertices

numEdges :: (Eq a, Eq w) => WeightedGraph a w -> Int
numEdges = length . edges

instance (Eq a, Show a, Eq w, Show w) => Show (WeightedGraph a w) where
  show wg  = "DictionaryWeightedGraph("++vs++", "++as++")"
   where
    vs  = "("++ intercalate ", " (map show (vertices wg)) ++")"
    as  = "(" ++ intercalate ", " (map showEdge (edges wg)) ++ ")"
    showEdge (WE x w y)  = intercalate "-" [ show x, show w, show y ]

```