


UNIVERSIDAD DE CARABOBO
Facultad Experimental de Ciencias y Tecnología
Departamento de Computación
Unidad Académica de Algoritmos, Programación y Lenguajes
CAO604: Lenguajes de Programación

Tarea #3: Programación Funcional

Semestre 1-2022
Valor: 35 % del Componente Práctico

Planteamiento

Considere la siguiente definición del tipo `Digrafo` (grafo dirigido):

```
module Digrafo (Digrafo(G),
                vertices,
                arcos,
                nVertices,
                nArcos,
                sucesores,
                antecesores,
                gradoSal,
                gradoEnt,
                depthFirstSearch,
                topologicalSort)
where
    data Digrafo v = G [v] (v -> [v])

    vertices :: Digrafo v -> [v]

    arcos :: Digrafo v -> [(v, v)]

    nVertices :: Digrafo v -> Int

    nArcos :: Digrafo v -> Int

    sucesores :: Eq v => Digrafo v -> v -> [v]

    antecesores :: Eq v => Digrafo v -> v -> [v]

    gradoSal :: Eq v => Digrafo v -> v -> Int

    gradoEnt :: Eq v => Digrafo v -> v -> Int

    depthFirstSearch :: Eq v => Digrafo v -> v -> [v]

    topologicalSort :: Eq v => Digrafo v -> [v]
```

Según esta definición, un digrafo de tipo `v` es un tipo de datos compuesto por la lista de vértices y una función que asigna a cada vértice la lista de sus sucesores.

Un ejemplo de digrafo, de acuerdo con esta definición, es el siguiente:

```
grafo1 = (G [1..4] suc) where
    suc 1 = [2,3]
    suc 2 = [4]
    suc 3 = [4]
    suc 4 = []
```

Adicionalmente, la descripción de las operaciones es la siguiente:

- `(vertices G)` es la lista de los vértices de `G` (**1 punto**)
- `(arcos G)` es la lista de los arcos de `G` (**2 puntos**)
- `(nVertices G)` es el número de vértices de `G` (**1 punto**)
- `(nArcos G)` es el número de arcos de `G` (**1 punto**)
- `(sucesores G x)` es la lista de sucesores directos del vértice `x` en `G` (**1 punto**)
- `(antecesores G x)` es la lista de antecesores directos del vértice `x` en `G` (**2 puntos**)
- `(gradoSal G x)` es el número de sucesores directos del vértice `x` en `G` (**1 punto**)
- `(gradoEnt G x)` es el número de antecesores directos del vértice `x` en `G` (**2 puntos**)
- `(depthFirstSearch G x)` es la lista de todos los vértices alcanzables desde `x` en `G`, aplicando búsqueda en profundidad (**4 puntos**)
- `(topologicalSort G)` es la lista que corresponde con el ordenamiento topológico de `G` (**5 puntos**)

Implemente el TDA `Digrafo`, bajo el modelo de programación funcional, utilizando el lenguaje de programación Haskell. Para ello, considere todas las operaciones antes mencionadas.

Observaciones

- La fecha tope de entrega de la asignación será el día jueves 27 de octubre de 2022, hasta las 11:59PM. Los archivos deben ser enviados por correo electrónico (`aammorales@gmail.com`). **No se recibirán asignaciones después de la fecha y hora especificadas en este ítem**.
- La asignación debe cumplir con un conjunto mínimo de funcionalidad a implementar, por lo que **sólo se considerará entregada si se obtiene en ella una calificación mayor o igual a cinco (05) puntos**.
- La asignación debe ser desarrollada en equipos de máximo dos (02) estudiantes.
- Los programas que entreguen deben estar debidamente documentados, y presentados con su nombre y su número de cédula de identidad.
- **El incumplimiento de las observaciones descritas anteriormente, generará puntos menos sobre la nota total de la asignación.**