ESTRUCTURA DE DATOS Y ALGORITMOS 2
CUARTA ENTREGA DE EJERCICIOS

Se aceptan entregas por Aulas hasta el 23/10/2016 a las 23:55

Para la realización de este trabajo la cátedra provee un proyecto de Visual Studio con las funciones a implementar, algunos datos de prueba y el framework visto en clase.

IMPORTANTE:

1. La definición de cada función incluye siempre indicar sus **pre** y **postcondiciones**.
2. Las pruebas provistas por la cátedra constituyen meros ejemplos y no son para nada exhaustivas. Se espera que los alumnos desarrollen pruebas propias para su código.
3. Debe entregarse solamente una carpeta comprimida conteniendo el proyecto “Obligatorio”, sin el Framework.

?1. TAD GRAFO.

1. Implemente la especificación de un grafo **dirigido** con aristas simples (solo se permite una arista entre un mismo origen y destino) dado por el archivo `Grafo.h`
2. Implementar la operación de Sistema que crea la implementación realizada en el punto anterior:

    - `Puntero<Grafo<V,A>> CrearGrafo(Puntero<FuncionHash<V>>, Comparador<V>)`

?2. Funciones Solicitadas.

Implementar las siguientes operaciones en la clase Sistema. Estas operaciones ya se encuentran definidas en la especificación en el archivo `Sistema.h`

- **HayCamino**.

  Que recibiendo un puntero a un grafo, retorna true si hay camino entre un vértice origen y uno destino.
- **EsConexo.**

  Que recibiendo un puntero a un grafo, retorna un elemento del enum “*TipoConexo*”, el cual indica si e l grafo es fuertemente conexo, débilmente conexo o si es no conexo.
- **OrdenTopologico**

  Que recibiendo un puntero a un grafo, retorna un Iterador de vértices que permite recorrerlos en un orden topológico.
- **ArbolCubrimientoMinimo**

Que recibiendo un puntero a un grafo, retorna un Iterador con las aristas que constituyen el árbol de cubrimiento mínimo. Las aristas se identificarán como Tupla<V, V>(origen, destino). Para implementar esta operación deberá tomarse el grafo como no dirigido.

- **ComponentesConexas**

Que recibiendo un puntero a un grafo, retorna un Iterador con las componentes conexas del grafo. Una componente conexa se representará como un Iterador de lo vértices que pertenecen a dicha componente.

?3. EJERCICIO GRAFO.

Considere un sistema para el registro y el análisis de los distintos medios de transporte entre las ciudades del país. De las ciudades, solo interesa conocer su nombre y de las conexiones entre las ciudades, se conoce: Ciudad origen, ciudad destino, tipo de transporte (avión u ómnibus), costo, tiempo y número de paradas intermedias. Dado un origen y un destino, solo puede haber una conexión, y además dicha conexión va desde origen hacia destino y no al revés. La cantidad máxima de ciudades está determinada por la constante MAX_CIUDADES.

Implementar las siguientes operaciones declaradas en la clase Sistema

- **AltaCiudad**.
  Recibe el nombre de la ciudad y da de alta la ciudad en el Sistema.
  Retorna ERROR si ya existe una ciudad con ese nombre.
- **AltaConexion**
  Recibe los datos de la conexión y la da de alta en el sistema.
  Retorna ERROR si origen o destino no pertenecen al sistema.
  Retorna ERROR si costo o tiempo valen 0.
- **CaminoMasBarato**

Dados el nombre de una ciudad origen y una destino retorna el camino de menor costo. Si existe más de un camino de menor costo, deberá retornar aquel de menor tiempo. A igual costo y tiempo, deberá retornar el que pase por menos ciudades.

Retorna ERROR si origen o destino no pertenecen al sistema.
Retorna ERROR si no hay camino entre origen y destino.

- **CaminoMenorTiempo**

Dados el nombre de una ciudad origen y una destino retorna el camino de menor tiempo. Si existe más de un camino de menor tiempo, deberá retornar aquel de menor costo. A igual costo y tiempo, deberá retornar el que pase por menos ciudades.

Retorna ERROR si origen o destino no pertenecen al sistema.
Retorna ERROR si no hay camino entre origen y destino.

- **CaminoMenosCiudades**

Dados el nombre de una ciudad origen y un destino retorna el camino que pasa por menos cantidad de ciudades. Si existe más de un camino de menor cantidad de ciudades, deberá retornar aquel de menor cantidad de paradas intermedias.

Retorna ERROR si origen o destino no pertenecen al sistema.

Retorna ERROR si no hay camino entre origen y destino.

- **CaminoMenosTrayectosOmnibus**

Dados el nombre de una ciudad origen y una destino retorna el camino que utiliza menor cantidad de medios de transporte ómnibus. A igual cantidad de medios de transporte, debe retornar el de menor tiempo.

Retorna ERROR si origen o destino no pertenecen al sistema.

Retorna ERROR si no hay camino entre origen y destino.

- **CaminoMenosTrayectosAvion**

Dados el nombre de una ciudad origen y una destino retorna el camino que utiliza menor cantidad de medios de transporte avión. A igual cantidad de medios de transporte, debe retornar el de menor tiempo.

Retorna ERROR si origen o destino no pertenecen al sistema.

Retorna ERROR si no hay camino entre origen y destino.

- **CaminoMenosParadasIntermedias**

Dados el nombre de una ciudad origen y una destino retorna el camino de menor cantidad de paradas intermedias. A igual cantidad de paradas, debe retornar el de menor costo.

Retorna ERROR si origen o destino no pertenecen al sistema.

Retorna ERROR si no hay camino entre origen y destino.

- **CaminoMasBaratoOmnibus**

Dados el nombre de una ciudad origen y una destino retorna el camino de menor costo. Si existe más de un camino de menor costo, deberá retornar aquel que use menos ómnibus como medio de transporte. A igual costo y medios de transporte, debe retornar el de menor tiempo.

Retorna ERROR si origen o destino no pertenecen al sistema.

Retorna ERROR si no hay camino entre origen y destino.