Estructuras de Datos

Grado en Informática
Segundo Curso, segundo cuatrimestre
Escuela Politécnica Superior de Córdoba
Universidad de Córdoba
Curso académico 2017-2018

Práctica 4. Árbol abarcador de coste mínimo de un grafo no dirigido

- **Objetivo**
  - Generar el árbol abarcador de coste mínimo de un grafo no dirigido.

- **Primera parte**
  - Implementación de las siguientes clases:
    - Clase ***Grafo*** usando una **matriz de adyacencia**.
    - Clase ***Vértice***
    - Clase ***Lado*** *(opcional)*
    - **Observación**
      - Se deben utilizar las especificaciones propuestas en las clases de Teoría.

- **Segunda parte**
  - Codifica dos ficheros denominados ***funcionesAuxiliares.cpp*** y ***funcionesAuxiliares.hpp*** que permitan la implementación de las siguientes funciones, al menos:
    - **Cargar vértices desde un fichero**
      - Se suministran un fichero de ejemplo denominado **vertices.txt** que tiene las coordenadas cartesianas de N puntos del plano.
      - Al cargar un grafo desde un fichero, se tienen que calcular y almacenar las distancias entre todos los vértices: lados del grafo.
    - **Aplicar los algoritmos para obtener el árbol abarcador de coste mínimo**
      - Función que permita aplicar el **algoritmo de Prim.**
      - Función que permita aplicar el **algoritmo Kruskal.**
      - Ambas funciones recibirán recibirán un grafo no dirigido y devolverán otro grafo que que represente el árbol abarcador de coste mínimo.
    - **Importante**:
      - Se deberá codificar cualquier otra función auxiliar que se considere que sea necesaria.

- **Tercera parte**
  - Codifica un **programa principal** que tenga, al menos, el siguiente menú de opciones:
    1. Cargar un vértices desde un fichero.
    2. Mostrar el grafo por pantalla.
    3. Árbol abarcador de coste mínimo
       a) Aplicar el algoritmo de **Prim**
       b) Aplicar el algoritmo de **Kruskal**
       c) Mostrar el árbol abarcador de coste mínimo (si ha sido generado)
       d) Mostrar la longitud total del árbol abarcador de coste mínimo (si ha sido generado)

e) Etc.
4. Etc.
0. Terminar

- **Observación**
  - Se debe crear un fichero ***makefile***.

- **Desarrollo de la práctica número 4**
  - Duración de la práctica 3: tres sesiones de dos horas cada una.
  - **Plazo máximo de entrega**
    - **9:00 horas del 30 de mayo de 2018**
  - Se deberá subir un fichero comprimido denominado "**practica-4-usuario.zip**", donde "usuario" es el ***login*** de cada estudiante.
  - El fichero comprimido contendrá
    - makefile
    - Doxyfile
    - ficheros hpp
    - ficheros cpp
    - ficheros txt de ejemplo
  - **Observación**:
    - Se debe usar el espacio de nombres de la asignatura: **ed**

- **Evaluación**
  - **Importante**
    - La evaluación de la práctica se deberá hacer los días **30 y 31 de mayo y 1 de junio**
      - Se habilitará una *wiki* para que cada estudiante elija fecha y hora para la corrección de su práctica.
      - La corrección se realizará en el despacho del profesor.
  - La calificación de la práctica se basará
    - en la calidad y completitud del trabajo realizado.
    - y en la **defensa presencial de cada estudiante**.
  - **Se valorará**
    - La correcta implementación de las **clases**
    - La correcta codificación de las pre y post-**condiciones** mediante asertos.
    - La correcta codificación de las **funciones auxiliares** del programa principal.
    - El correcto funcionamiento del programa principal y su ampliación con nuevas opciones.
    - La documentación del código con doxygen.
    - La claridad del código.
    - El uso de macros de pantalla para mejorar la visualización de la información
    - **Y sobre todo**
      - *Un profundo conocimiento de la práctica codificada.*