Instituto Tecnológico de Costa Rica

Sede Interuniversitaria

Ingeniería en Computación

# Estructuras de Datos - III Proyecto Programado

Profesora: Samanta Ramijan Carmiol

Estudiantes: Natán Fernández de Castro - 2017105774
Pablo Venegas Sánchez - 2018083497

Documentación Externa

Rutas Ficticias – Programa de Rutas en Lenguaje C

II Semestre 2018

## 1. Tabla de Contenidos

## 2. Descripción del Problema

El problema consiste en implementar un programa de Rutas Ficticias de una serie televisiva o película, en nuestro caso se eligió la serie Juego de Tronos (Game Of Thrones), y mediante un mapa ficticio de la serie o película crear un grafo con pesos para distancia de cada punto de localidad en el mapa a otro. Al usuario se le da la posibilidad de cargar un archivo de tablas .csv que contiene una matriz de pesos con las conexiones de las localidades, recorrer el grafo en anchura (BFS), actualizar el mapa conectando o desconectando dos puntos y un reto de camino más corto según algoritmo Warshall.

## 3. Diseño del Programa

### 3.1 Decisiones de diseño

Se decidió implementar patrón de diseño Modelo Vista Controlador, en cual permite asignar responsabilidades en lo que respecta a eventos, peticiones y obtención de datos. Es decir, se divide la parte visual, datos (lógica) y obtención de esos datos. Esto permitiría mayor orden en la estructuración del diseño del programa, corrección de errores y control sobre estructuras.

Además se decide por medio de archivos de cabecera (.h) controlar las funciones privadas y públicas que cada archivo .c maneja. Estos permite controlar la información que se muestra sobre la solución e implementación.

Para cargar el archivo de los pesos del grafo se abre y se recorre en ejecución para generar una matriz de adyacencia con base a lo que va leyendo del archivo. Además se encuentra otro archivo .csv aparte con lo nombres de cada punto del grafo al cual se le aplica el mismo proceso que al archivo de pesos.

- **cargar_rutas:** Realiza la carga del archivo .csv, una vez en ejecución genera la matriz de adyacencia según captura el contenido del archivo.
- **bfs:** Esta función tiene como propósito realizar el recorrido en anchura de la matriz de adyacencia (BFS) y mostrar en pantalla el resultado del recorrido. Básicamente lo que hace es que a través de la implementación de una cola, va almacenando los nodos hermanos del nodo actual conforme va avanzando.
- **optimizar_rutas:** LLama a la función floyd() que realiza el algoritmo de ruta_mas_corta.
- **floyd:** Mediante el algoritmo Floyd-Warshall de ruta más corta en un grafo con peso recorre la matriz de adyacencia a partir de dos puntos del grafo dados, la salida de esta

función será una matriz de caminos, que se usará en la función buscar_recorrido_mas_corto para mostrar el resultado del camino final al usuario.

- **buscar_recorrido_mas_corto:** Se encarga de solicitarle al usuario los dos puntos que se usará en la búsqueda del camino más corto, lo que hace esta función es que mediante se vaya actualizando la matriz de caminos realice llamada a función floyd(), realiza validaciones de casos como caminos inaccesibles (infinito) o mantenerse en rango del tamaño de la matriz de adyacencia.
- **reto_random:** Función destinada a buscar la ruta más corta entre dos nodos aleatorios, en sí lo que hace es semejante a buscar_recorrido_mas_corto() pero con los dos nodos de partida y destinos utilizando función srand() de Lenguaje C.
- **actualizar_mapa:** Conecta o desconecta dos puntos en la matriz de adyacencia según el usuario requiera. Lo que realiza es capturar el peso deseado, si desea desconectar dos nodos el peso es 999 (infinito), y los dos nodos a conectar o desconectar. Así modifica la matriz de adyacencia con la información capturada.

**4. Análisis de Resultados**

Objetivos Alcanzados

- Resultados del trabajo casi satisfactorios, se lograron implementar la mayoría de requerimientos. Excepto por algunos casos de ruta más corta a la hora de mostrar el camino final, ya que omite la impresión de algunos nodos del camino óptimo resultante.
- Mejor comunicación y trabajo en equipo.
- No se pudo lograr implementar los puntos extra de interfaz para el programa.

**5. Manual de Usuario**

**cargar_archivo:** Cargar archivo tipo .CSV y genera matriz de adyacencia.
**recorrer_grafo**: Imprime en consola el recorrido en anchura del grafo BFS.
**actualizar_mapa**: Conectar-Desconectar dos nodos dados en matriz de adyacencia.
**ruta_mas_corta**: Busca camino más corto utilizando algoritmo de Floyd-Warshall.
**reto_random**: Busca camino más corto entre 2 nodos aleatorios con Floyd-Warshall.
**ayuda**: Imprimir en consola comandos disponibles.
**acerca_de**: Imprimir en consola parámetros del programa, curso y desarrolladores.

## 6. Lecciones Aprendidas

Este proyecto fue de provecho para comprender el funcionamiento y comportamiento de un grafo con peso. Los algoritmos utilizados como BFS y Floyd Warshall conceptualizan este comportamiento de los grafos y su aplicación al proyecto fue exitosa. Es una forma de abstraer una estructura de datos utilizada en aplicaciones cotidianas en un contexto simple hasta contextos complejos como redes o motores de búsqueda.