UNIVERSIDAD CENTRAL DE VENEZUELA

FACULTAD DE CIENCIAS

ESCUELA DE COMPUTACIÓN

Informe Proyecto #2

Algoritmos y Estructuras de Datos

Grupo docente:
Francisco Moreno
Tony Briceño
Bryan Silva

Integrantes:
Joseph Acevedo
Diego León

Caracas, marzo de 2025

Proyecto #2 - Gestión Arcana

El programa solicitado se compone de diversas clases y funciones auxiliares que se encargan del funcionamiento del mismo, la función principal “main” nada más llama a la acción “Entrada”, esto debido a que los datos a procesar se encuentran en un archivo llamado “spellList.in”, junto con un archivo llamado “underInvestigation.in” que contiene una lista con los nombres de los magos buscados por la ley.

Para resolver la problemática mencionada en el enunciado del proyecto se utilizaron técnicas derivadas del campo de la teoría de grafos, esto con la finalidad de poder representar adecuadamente cada hechizo dentro del programa.

Se diseñó la clase “Hechizo” para representar un grafo creado a partir de cada instancia de hechizos que aparecen en el archivo de entrada “spellList.in”, esta cuenta con los atributos “mágicos” del hechizo referente a los vértices que posee dicho grafo junto con el nombre del mago que realizó el hechizo y el nombre del mismo. La clase cuenta con su respectivo constructor y destructor, además de una función para obtener un vértice del arreglo que los almacena.

Seguidamente se diseñó la clase “Vértice” cuyos atributos se componen principalmente de un carácter de tipo “runa” que hace referencia al tipo de vértice, entendiendo que este puede representar ya sea una runa elemental, un punto de confluencia, una runa catalítica, runa de estabilidad, o un punto de soporte energético. Además de esto, cada vértice guarda una lista de aristas de manera tal que se puedan determinar los vértices adyacentes a este.

Por último, se creó la clase “Arista” la cual almacena un número real asociado al peso de cada arista o conexión, seguido de un entero que hace referencia al índice del vértice adyacente almacenado en el arreglo de vértices.

Estas tres clases en conjunto forman la estructura del grafo, a su vez el programa también se vale de las clases “Nodo” y “Lista” las cuales permiten modelar el comportamiento de una lista, y de la clase “Sospechoso” la cual posee como atributos el nombre de un mago y un entero que guarda el número de hechizos ilegales que realizó.

En lo que respecta a la funcionalidad general del programa, sólo se usaron las librerías “iostream” y “fstream”, en la función “main” se llama a la acción “Entrada” la cual se encarga principalmente de leer el contenido del archivo “spellList.in”, almacenar su información como un objeto de la clase “Hechizo” y procesarlo, sin embargo para completar la lógica de esta acción dentro del programa se realizan llamados a las siguientes funciones:

procesarHechizo: Esta acción se encarga de nombrar el hechizo que se está recibiendo actualmente y de determinar si este es ilegal o no en base a los criterios que presenta el enunciado, para ello esta acción se vale de las siguientes funciones:

- **confluenciaValida**: Chequea si el hechizo posee un solo punto de confluencia, seguidamente verifica que el punto de confluencia solo tenga puntos de soporte energético adyacentes a él, si no cumple cualquiera de estas condiciones retorna falso según lo indicado por los artículos 1 y 2.
- **modificarApellido**: Modifica el apellido del mago para añadirlo al nombre del hechizo, si este termina en vocal, se elimina la vocal y se añade “ium”, en caso contrario se añade “um” al final del apellido.
- **excesoRunasElementales**: Verifica principalmente que el hechizo no tenga más de 3 runas elementales, durante esta validación se chequea si el hechizo posee tanto

runa catalítica y de estabilidad. Si posee más de 3 runas elementales la función retorna falso y además queda establecido que el hechizo es arcano.

- **runasCataliticasValidas**: Verifica que una runa catalítica no sea adyacente a una runa elemental, si esto ocurre la función retorna falso.
- **encontrarCicloMasLargo**: Esta acción emplea un algoritmo de DFS (Depth First Search) para encontrar el ciclo simple más largo por número de aristas en un hechizo.
- **encontrarCaminoMasPesado**: La función encuentra el camino más pesado por ponderación de las aristas entre el punto de confluencia y una runa catalítica, runa elemental o de estabilidad. En caso de que la longitud del ciclo simple más largo por número de aristas sea mayor o igual que el camino más pesado por ponderación, se añadirá al final del nombre del hechizo la palabra “maximus”, si la longitud del ciclo simple más largo resulta ser menor se añade al final del nombre la palabra “modicum”. En el caso particular de que el hechizo sea arcano y no posee runa de estabilidad y runa catalítica se añadirá al final la palabra “Arcante”
- **asignarIndice**: Devuelve un entero que sirve de índice para los elementos de un arreglo que almacena secuencias de caracteres referentes al principio del nombre de cada hechizo, la función recibe un carácter y dependiendo de su tipo se le asignará al principio del nombre de cada hechizo el prefijo correspondiente según lo establecido en el artículo 8.

enviarDatos: La función genera un archivo de salida llamado “processedSpells.out” el cual contiene una lista con todos los hechizos legales e ilegales junto con el nombre del mago que los realizó

actualizarListaInvestigacion: La función actualiza el archivo de entrada “underInvestigation.in” con los nombres de los magos que realizaron al menos 3hechizos ilegales si estos no se encontraban previamente en el archivo.

magosBajoInvestigacion: Esta acción recibe los datos del archivo “underInestigation.in”.

Conociendo estas funcionalidades y objetivos, el flujo del programa tiene la siguiente estructura:

1. Lectura de Datos:

   La acción “Entrada” carga los datos de “spellList.in” (número de hechizos, seguido de N grupos según la cantidad de hechizos donde se recibe el nombre del mago, número de vértices, una cadena de caracteres donde cada elemento es un tipo de vértice, el número de aristas, y luego los índices de los vértices que conecta cada arista junto con su ponderación). Por último esta recibe un archivo llamado “underInvestigation.in” el cual contiene el nombre de los magos buscados por la ley.

2. Procesamiento de Hechizos:

   Directamente en la función “Entrada” se procesa cada hechizo al momento de su lectura antes de pasar al siguiente, se verifica que el hechizo no incumpla la normativa planteada por la LUGIS. Tras procesarlo, el hechizo se añade a una lista de hechizos legales o ilegales según corresponda.

3. Generar archivos de salida:

   Se genera un archivo llamado “processedSpells.out” en el que se escribe el contenido de las listas de hechizos mencionadas anteriormente. Por último se actualiza el archivo de entrada “underInvestigation.in”.