Instituto Tecnológico de Costa Rica
Ingeniería en Computación
Sede Interuniversitaria de Alajuela
Curso: Estructuras de Datos

Profesor: Adriana Céspedes

Tarea Programada 3
Tema: Coloreo de grafos y rutas cortas

Estudiantes:

David Umaña Blanco - 2016175133
Emmanuel Alfaro Brenes - 2016215572
Isaac Mena López - 2016130651

I Semestre 2017

Índice

1. Introducción

En esta tarea programada se desarrollará un proyecto utilizando el lenguaje de programación C++ y el paradigma orientado a objetos, para crear un programa que solucione los problemas del coloreo de mapas y el del mejor camino entre ciudades. Ambos problemas son aplicados usando TDA (Tipo de Dato Abstracto) de grafo no dirigido.

2. Enunciado del problema

El problema del coloreo de mapas con k colores consiste en, colorear un mapa con *k* colores de forma que sus regiones adyacentes no tengan el mismo color. Estas regiones podrían ser países, comunidades autónomas, provincias, municipios, etc. Los vértices adyacentes deben recibir colores distintos. (WikiRa, 2011).

El problema del mejor camino entre ciudades o el camino mas corto consiste en encontrar un camino entre dos vértices (o nodos) de tal manera que la suma de los pesos de las aristas que lo constituyen es mínima. Un ejemplo de esto es encontrar el camino más rápido para ir de una ciudad a otra en un mapa. En este caso, los vértices representarían las ciudades y las aristas las carreteras que las unen, cuya ponderación viene dada por el tiempo que se emplea en atravesarlas. Este problema se aplica en la teoría de grafos. (Wikipedia, 2017).

El programa implementado genera las soluciones a ambos problemas, muestra el grafo implementado

en una matriz y lee los archivos que contienen los vértices y las aristas, en archivos de formato tipo csv (del inglés comma-separated values) donde cada valor se encuentra separado por comas. La solución se modela para trabajar con grafo no dirigido, pero se implementó para grafo dirigido y no dirigido. Cada nodo del grafo contiene un ID (string) que lo identifica.

El programa muestra ambas soluciones, el coloreo de mapas y con la implementación del algoritmo de Floyd-Warshall, se muestra una matriz de costos, una matriz con los caminos más cortos y las listas de los vértices que hay que visitar en la ruta.

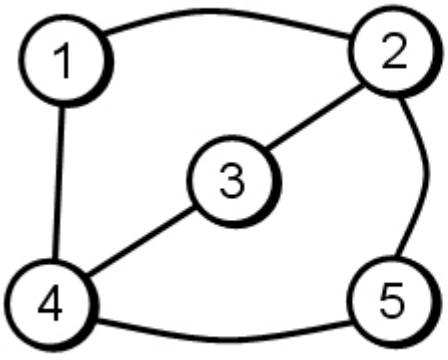
3. Definición de los modelos

Los modelos utilizados son una matriz que representa el grafo del mapa, dos matrices que representan la matriz de costos y la matriz de caminos para el algoritmo de Floyd-Warshall. Un arreglo donde se muestra el coloreo de los vértices.

4. Operadores básicos del modelo

Operadores para Grafo: Crear, Destruir, una matriz, posición del Nodo, Vértices, Coloreo, Matriz Costos, Matriz Caminos, Algoritmo Floyd-Warshall.

5. Diagrama de las Estructuras de Datos implementadas



M	1	2	3	4	5
1	0	1	0	1	0
2	1	0	1	0	1
3	0	1	0	1	0
4	1	0	1	0	1
5	0	1	0	1	0

Figura 1: Representación de un grafo con matriz. Fuente: Wikipedia.

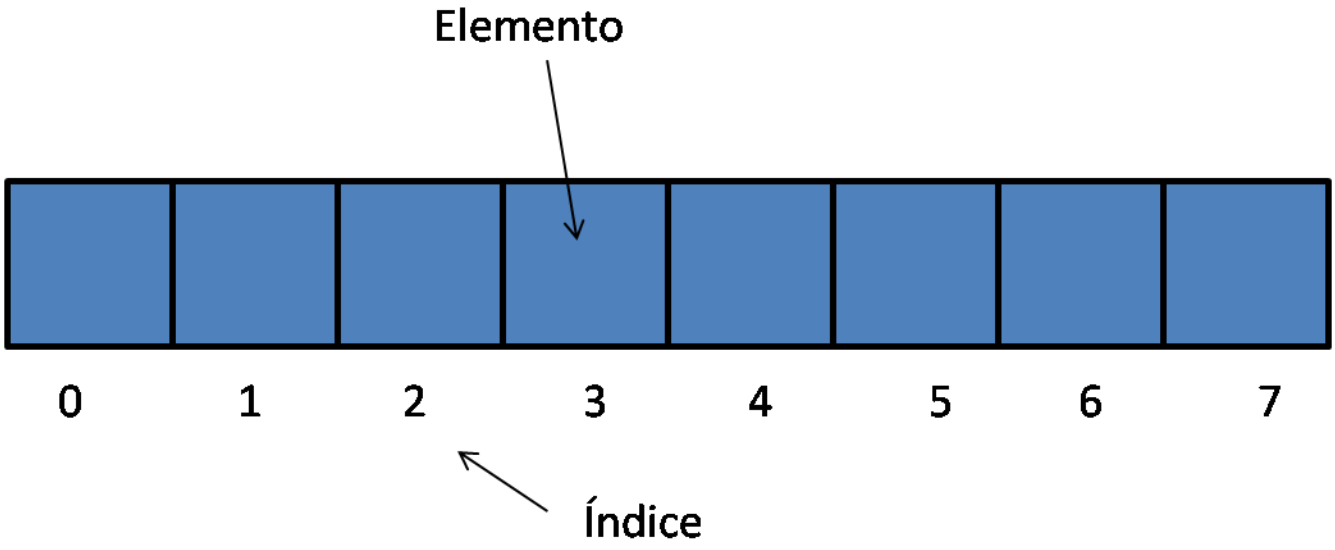


Figura 2: Representación de un arreglo. Fuente: http://programacionjand.blogspot.com/

6. Descripción de algoritmos

6.1 Colorear:

1. Definir color 1, colorear el primer nodo y pasar al siguiente.
2. Si ningún vecino tiene el ultimo color puesto (o alguno menor) se pone y pasa al siguiente nodo.
3. Si no se usa un color nuevo.

6.2 Floyd-Warshall:

1. Inicializar la matriz de caminos
2. Se fija la fila y columna k
3. Se iteran filas y columnas: en cada casilla se valida que en los sumandos de la casilla actual no hay ningún elemento absorbente y que no se está en la diagonal.
4. Se valida si se puede mejorar la casilla mediante la suma de las intersecciones

6.3 Mostrar rutas:

1. Se iteran filas y columnas, pasando por lo elementos en medio, hasta que el nodo destino es el siguiente en visitar.

6.4 Leer de datos archivo:

1. Para recuperar los datos de lo vértices primero se itera el archivo para saber la cantidad de vértices con los que se va a a trabajar. Luego se itera de nuevo recuperando los datos.
2. Se analiza línea por línea hasta que se lee una línea nula

7. Implementación

Para representar el grafo se genera una matriz, su tamaño varía dependiendo de la cantidad de vértices, que contenga el archivo que lea el programa. La entrada se representa por medio de dos archivos diferentes en formato csv, donde se encuentran, los ID de cada Nodo, los vértices y las aristas de los nodos.

En el coloreo de mapas, se usa un arreglo donde muestra el ID representado con un entero y su respectivo color, el resultado se muestra en un arreglo. En el problema del camino mas corto, se muestra la matriz de adyacencia del grafo, y como resultado muestra la matriz de costos y los caminos que debe tomar por cada inserción.

8. Requerimientos de hardware

La tarea fue desarrollada en computadoras portátiles:

Toshiba L455, 3GB RAM, Core 2 Duo, Ubuntu 16.04 LTS 64 bits.

Acer R3-471T, 8GB RAM, Core i7, Ubuntu 15.10 64bits.

Dell Inspiron M731R, 8GB RAM, AMD A10 Elite Quad-Core, Ubuntu 15.04 64 bits.

9. Requerimientos de software

Se utilizó Geany como IDE, gcc 5.4.0 20160609 como compilador, doxygen para la creación de la documentación del código, y Valgrind como detector de fugas de memoria.

10. Manual de usuario

¿Cómo se utiliza el programa?

Inicialmente debe rellenar los datos que se requieren en los archivos externos con la sintaxis respectiva (no dejar lineas en blanco después del final de los datos en los archivos):

1. Para Vértices : id,nombre
2. Para Artistas: id1, id2, peso

Luego sólo es necesario correr el programa, en seguida se le mostrarán dos opciones que le permiten ver las dos soluciones del grafo: para el coloreo o para la rutas cortas. Elija la que desea.

11. Datos de prueba y resultados

<table>
<tr><th>vertices .csv</th><th>aristas. csv</th><th>Coloreo</th><th>Matriz Caminos</th><th>Matriz Costos</th></tr>
<tr>
<td>1,A
2,B
3,C
4,D
5,E
6,F</td>
<td>1,2,3
1,3,5
1,4,1
2,1,3
2,5,9
3,1,5
3,4,7
3,5,7
3,6,1
4,1,1
4,3,7
4,6,4
5,2,9
5,3,7
6,3,1
6,4,4</td>
<td>1,2,2,3,1,1</td>
<td>
<table>
<tr><th>id</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th></tr>
<tr><td>1</td><td>1</td><td>2</td><td>3</td><td>4</td><td>2</td><td>4</td></tr>
<tr><td>2</td><td>1</td><td>2</td><td>1</td><td>1</td><td>5</td><td>4</td></tr>
<tr><td>3</td><td>1</td><td>1</td><td>3</td><td>6</td><td>5</td><td>6</td></tr>
<tr><td>4</td><td>1</td><td>1</td><td>6</td><td>4</td><td>6</td><td>6</td></tr>
<tr><td>5</td><td>2</td><td>2</td><td>3</td><td>6</td><td>5</td><td>3</td></tr>
<tr><td>6</td><td>4</td><td>4</td><td>3</td><td>4</td><td>3</td><td>6</td></tr>
</table>
</td>
<td>
<table>
<tr><th>id</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th></tr>
<tr><td>1</td><td>0</td><td>3</td><td>5</td><td>1</td><td>12</td><td>5</td></tr>
<tr><td>2</td><td>3</td><td>0</td><td>8</td><td>4</td><td>9</td><td>8</td></tr>
<tr><td>3</td><td>5</td><td>8</td><td>0</td><td>5</td><td>7</td><td>1</td></tr>
<tr><td>4</td><td>1</td><td>4</td><td>5</td><td>0</td><td>12</td><td>4</td></tr>
<tr><td>5</td><td>2</td><td>9</td><td>7</td><td>12</td><td>0</td><td>8</td></tr>
<tr><td>6</td><td>5</td><td>8</td><td>1</td><td>4</td><td>8</td><td>0</td></tr>
</table>
</td>
</tr>
<tr>
<td>1,A
2,B
3,C
4,D
5,E</td>
<td>1,2,3
2,3,2
3,4,1
4,5,5
5,1,7
2,4,2
2,5,1</td>
<td>1,2,1,3,4</td>
<td>
<table>
<tr><th>id</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th></tr>
<tr><td>1</td><td>1</td><td>2</td><td>2</td><td>2</td><td>2</td></tr>
<tr><td>2</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td></tr>
<tr><td>3</td><td>2</td><td>2</td><td>3</td><td>4</td><td>2</td></tr>
<tr><td>4</td><td>2</td><td>2</td><td>3</td><td>4</td><td>2</td></tr>
<tr><td>5</td><td>2</td><td>2</td><td>2</td><td>2</td><td>5</td></tr>
</table>
</td>
<td>
<table>
<tr><th>id</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th></tr>
<tr><td>1</td><td>0</td><td>3</td><td>5</td><td>5</td><td>4</td></tr>
<tr><td>2</td><td>3</td><td>0</td><td>2</td><td>2</td><td>1</td></tr>
<tr><td>3</td><td>5</td><td>2</td><td>0</td><td>1</td><td>3</td></tr>
<tr><td>4</td><td>5</td><td>2</td><td>1</td><td>0</td><td>3</td></tr>
<tr><td>5</td><td>4</td><td>1</td><td>3</td><td>3</td><td>0</td></tr>
</table>
</td>
</tr>
</table>

12. Problemas y limitaciones

No hubo ningún problema ni limitación.

13. Conclusiones y resultados

En el desarrollo de este programa se aprendió cómo implementar un grafo por medio de una matriz de adyacencia, cómo entender y resolver los problemas del coloreo de mapas y el del camino más corto, usando el algoritmo de Floyd-Warshall para encontrar la mejor ruta. Además se apreció la utilidad que se puede dar a la teoría de grafos en la solución de problemas cotidianos.

14. Bibliografía

Bhojasia, M. (s.f.). C++ Program to Implement Floyd-Warshall Algorithm. Recuperado de: http://www.sanfoundry.com/cpp-program-implement-floyd-warshall-algorithm/

Fitzhardinge J. et al (2016). Valgrind Recuperado de: http://valgrind.org/

Tröger E. et. al (2012). Geany. Recuperado de: https://www.geany.org/

van Heesch. D. (2016). Doxygen. Recuperado de: http://www.stack.nl/~dimitri/doxygen/