Universidad de Costa Rica
Escuela de Ingeniería Eléctrica

IE-0217 Estructuras Abtractas de Datos y Algoritmos para Ingeniería

---

Propuesta de Proyecto 1

Librería de Procesamiento de Grafos en C++

Autores:
Diego Álvarez A.
Ernesto Céspedes M.
Hugo Zuñiga C.

Profesor:
Fransisco Siles C.

16 de septiembre de 2013

1. Objetivos

1.1. Objetivo General

- Implementar una librería de procesamiento de grafos para C++

1.2. Objetivos Especfícos

- Desarrollar las estructuras de datos: grafo no dirigido, grafo dirigido y árbol de expansión.
- Implementar algoritmos de busqueda de paths, de conectividad y deteccion de ciclos basado en los algoritmos básicos de Depth First Search y Breadth First Search.
- Implementar los algoritmos de: Kruskal y Prim's para arboles de expansion
- Implementar el algoritmo de Djikstra y el algoritmo de Bellman-Ford para búsqueda del camino más corto.

2. Conceptos

2.1. Grafos No Dirigidos

Un grafo es un conjunto de vertices o nodos los cuales se conectan a través de aristas o ejes.Este Conjunto de elementos permite representar relaciones binarias entre elementos de un conjunto y tiene su aplicación en problemas que involucran mapas, circuitos, redes de computadoras, etc. Se dice que un grafo está conectado si existe un path que conecta un vértice con todos los otros vértices del grafo. Una representación gráfica de un grafo se muestra en la figura [**?**] en la que se señalan algunas de la definiciones señaladas.

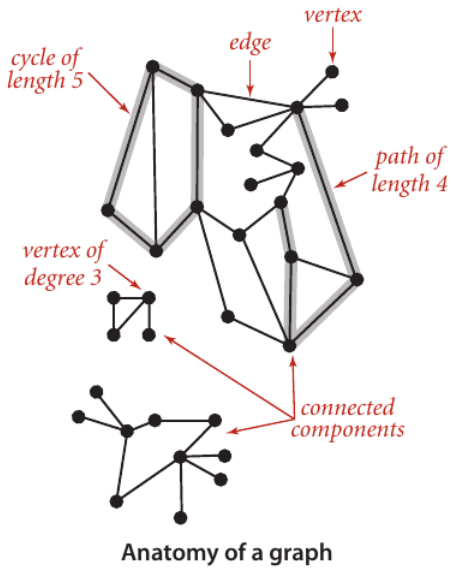

Figura 1: Gráfo de 250 nodos con 1273 aristas, y su MST.

Los algorímos simples de procesamiento de grafos no dirigidos se presentan a continuación:

2.1.1. Depth First Search

Este es un método recursivo, el cual permite evaluar todos los vértices y conexiones en un grafo. Es utilizado para problemas de conectividad, para saber si un dato es conexo, detectar ciclos dentro de un grafo. La idea del algorítmo es ir expandiendo todos y cada uno de los nodos que va localizando de forma recurrente, en un camino concreto. Cuando no encuentra más camino regresa.

2.1.2. Breadth-first search

Este es un método que se utiliza principalmente para la búsqueda de los paths más cortos. El Bread-first search va formando un arbol a medida que va recorriendo un grafo, empezando desde la

raiz y se exploran todos los nodos adyacentes, y luego a los nodos adyacentes de estos de manera consecutiva hasta que recorre todo el árbol

2.2. Gráfos Dirigidos

Los grafos dirigidos presentan la particularidad de que las conexiones son dirigidas, lo cual significa que cada una de las adyacencias se indica en una dirección específica. A partir de esta estructura de datos se pueden obtener una diversidad de algoritmos, de los cuales se planea implementar los siguientes:

Algoritmos	**Descripción**
Alcanzabilidad	Permite determinar los vértices que se encuentran conectados y son alcanzables desde un elemento específico.
Detección de ciclos	Permite determinar la presencia de ciclos en un grafo dirigido.
Topological Sort	Permite devolver una lista con los grafos ordenados, de manera que no haya ningún elemento antes de los que le tienen ejes dirigidos hacia el.
Conectividad Fuerte (Algoritmo de Kosaraju)	Este algoritmo permite saber determinar si existe una conectividad fuerte entre dos vértices, que existe un camino que lo conecta en ambos sentidos.

2.3. Grfos de Arístas con peso

Un grafo de aristas con peso, o edge weighted es un modelo de grafo que asocia pesos o costos con cada arista o edge. Estos son utilizados en muchas aplicaciones reales tales como para representar rutas de vuelos, el largo de cables o su costo en una red eléctrica. Minimizar costos tiene un natural interés en estas situaciones. En este proyecto se desarrollarán dentro de la librería, algoritmos para poder examinar grafos de aristas con peso indirectos, principalmente para un problema específico, el árbol de expansión mínimo, o minumum spaning tree (MST) de un grafo con pesos, él cual básicamente es un subgrafo sin ciclos que incluye todos los vértices, y cuya suma de pesos nos es mayor que el peso de cualquier otro árbol de expansión.

2.3.1. Algoritmo de Prim

Además se implementará el algoritmo de Prim. Este selecciona cualquier vértice y selecciona su edge menos pesado, agregándolo a un solo spanning tree después de recorrer todos los vértices. En otras palabras construye un árbol agregando aristas de manera iterativa hasta que se obtiene un árbol de expansión mínima. El algoritmo comienza con un solo vértice. Después, en cada iteración, agrega al árbol actual una arista de peso mínimo que no completa un ciclo. Este algoritmo puede ser

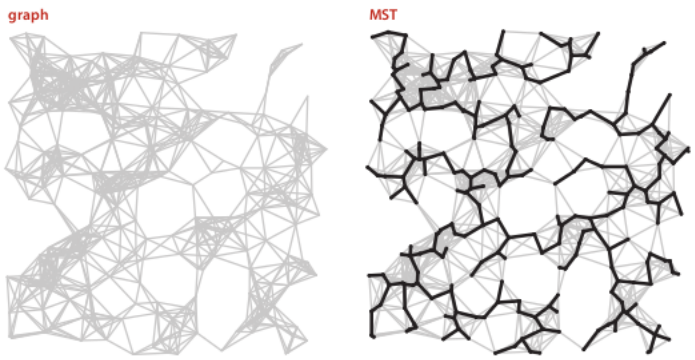

Figura 2: Gráfo de 250 nodos con 1273 aristas, y su MST.

utilizado en muchos campos como lo son las carreteras, vías férreas, aéreas o marítimas. También en redes eléctricas o de telefonía.

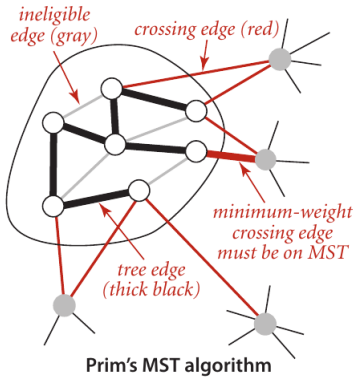

Figura 3: MST obtenido por el algoritmo de Prim.

Este algoritmo se profundizará analizando sus dos versiones, la versión perezosa o Lazy y la versión Eager. Cuya diferencia radica en la cantidad de espacio de memoria y tiempo para realizar el cálculo, ya que en la primera estos son proporcionales a la cantidad de edges, contrario a la segunda que utiliza memoria proporcional a los vértices, aunque el tiempo si es proporcional a los edges.

2.3.2. Algoritmo de Kruskal

Resuelve la misma clase de problema que el de Prim, salvo que en este no parte desde ningún nodo elegido al azar. Para resolver el mismo problema lo que hace es pasarle a la función una lista con las aristas ordenada de menor a mayor, y va tomando una para formar el MST. En un principio cada nodo está en un grupo distinto, al elegir un edge de la lista comprueba si no están los nodos conectados ya en el mismo grupo, de no estarlo fusiona ambos grupos y comprueba si ha encontrado ya la solución, para devolver el resultado.

3. Cronograma

Semana	**Actividades**
2-6 Setiembre	Definición del alcance del proyecto.
9-13 Setiembre	Creación de las estructuras de datos y algoritmos basicos.
16-20 Setiembre	16-20 Desarrollo de las estructuras y especificación para las pruebas.
23-27 Setiembre	23-27 Desarrollo de las pruebas y empaquetamiento de la librería.
1-3 octubre	Últmas revisiones, documentación y preparación de la presentación

Referencias

[1] Cormen, T., Leiseron, C. & Rivest, R. (2009) *Introduction to Algorithms: Third edition*. Massachusetts : MIT.

[2] Sedgewick, R. & Wayne, R. (2011) *Algorithms: Four edition*. Boston Massachusetts : Pearson.