Reflexión Individual

Actividad 4.3

Santiago Álvarez Valdivia

A01640172

Programación de estructuras de datos y algoritmos fundamentales

Tecnológico de Monterrey

Campus Guadalajara

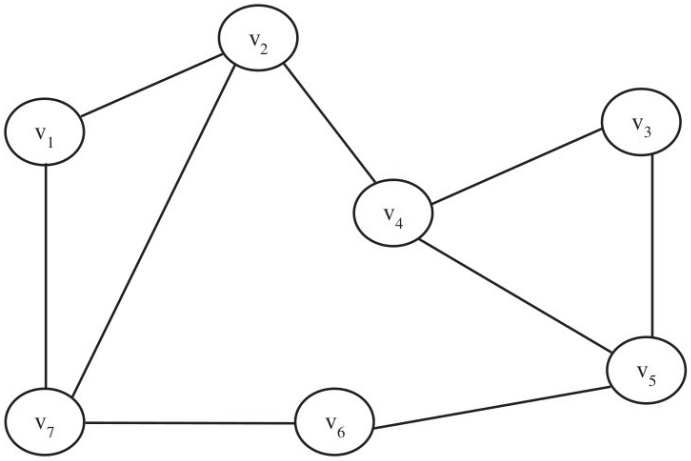
Escuela de Ingeniería y Ciencias

Profesor: Eduardo Arturo Rodríguez Tello

26 de noviembre de 2021

Grafos

Se le llama grafos a una estructura de datos no lineal, en la cual cada elemento se relaciona con otros elementos. Están formadas por nodos, llamados generalmente vértices, y por arcos, conocidos como aristas. Los nodos almacenan datos, mientras que las aristas indican conexiones o caminos, relativos a los datos almacenados.



El grado de un vértice (representado con v) es el total de aristas que tienen como extremo a v. Cuando el grado de un vértice es 0, significa que este está aislado del resto de nodos.

También cabe destacar que un grafo puede ser dirigido o no dirigido. El grafo de la imagen es uno no dirigido, donde el arco v_1 - v_7 puede ser recorrido tanto de v_1 a v_7 como de v_7 a v_1. Esto lo diferencia de un grafo dirigido, donde cada arista tiene un sentido determinado, que es especificado por que las aristas tienen forma de flecha.

Para representar un grafo computacionalmente, existen varias opciones, principalmente la matriz y la lista de adyacencia. La primera consiste en una matriz donde las filas y las columnas representan a los nodos, y se emplean valores de 0 o de 1 para indicar si una relación existe, o si no lo hace. La lista de adyacencia, por su parte, consiste en un arreglo de tamaño n, que contiene otros arreglos. Cada elemento *i* del arreglo principal, es un subarreglo con los vértices con los que se relaciona *i*.

Para resolver la presente actividad integradora, empleamos un grafo dirigido, que representamos por medio de una lista de adyacencia.

Descripción (a grandes rasgos) del programa

La situación problema consiste en tomar un conjunto de datos (direcciones IP y registros de incidencia entre direcciones) y obtener el número de relaciones en las que participa cada dirección IP.

En el programa, se emplean las direcciones como nodos, y las incidencias como relaciones. Como cada incidencia si tiene una IP de origen y una de destino bien definida, usamos un grafo definido como modelo. Por su parte, se emplea una lista de adyacencia para

representar a las aristas, ya que una matriz necesitaría demasiado espacio en memoria debido a la cantidad de nodos.

Este programa tiene el siguiente proceso en su operación:

1. Leer la primera parte de los datos proporcionados (que contiene únicamente a las direcciones IP involucradas, almacenándolas en un vector de C++
2. Ordenar las direcciones IP, para que vayan de menor a mayor (Empleando el algoritmo de ordenamiento QuickSort)
3. Leer la segunda parte del archivo de datos (que contiene las aristas). Para cada elemento, se hace lo siguiente:
    1. Se obtiene la IP origen y la IP destino
    2. Se busca la IP origen en el vector ya definido, para obtener su índice
    3. En la lista de adyacencia, se añade la IP destino, empleando el índice obtenido en el paso anterior
4. Una vez leídos todos los datos, se crea un árbol Heap que contenga las direcciones IP, y el grado de cada una.
5. Se toman los elementos del árbol, uno a uno, así obteniendo las direcciones IP de mayor a menor grado.

El uso de grafos fue indispensable para la creación del programa, ya que esta estructura de datos nos permite establecer (y contabilizar) de manera eficiente las relaciones entre nodos, en este caso entre las direcciones IP.

Como nota adicional, se puede mencionar la complejidad de algunos de los componentes empleados en el programa, donde n es el número de nodos, y m el número de relaciones.

- Búsqueda en el vector $O(n \log n)$
- Constructor del grafo (lectura y organización de datos) $O(n + m\, n \log n)$, incluyendo a la búsqueda de datos.
- Obtención del grado de cada IP $O(n)$

Referencias

[1] Buemo, S. G., Botello, F. O., & Cerpas, J. L. G. (2007). *Estructura de datos orientada a objetos*. Pearson Educación.