Tecnológico de Monterrey

Act 4.3 - Actividad Integral de Conceptos Básicos y Algoritmos Fundamentales (Evidencia Competencia)

Juan Pablo Zambrano Barajas A01636420

31 de Enero del 2023

Programación de estructuras de datos y algoritmos fundamentales

TC1031

Dr. Eduardo Arturo Rodríguez Tello

Los grafos son una estructura de datos no lineal que consiste de nodos y aristas, estas aristas pueden ser dirigidas y tener un peso, tienen bastantes usos en la vida real, pues sirven para representar relaciones entre objetos o entidades, por ejemplo se usan para representar redes, ya puede ser como los caminos de una ciudad, el camino de la electricidad o también en informática, un ejemplo muy popular es facebook, donde cada persona es un nodo con su respectiva información y sus amigos se conectan por aristas. Es por esto que en esta actividad los grafos son la mejor estructura de datos, pues nuestra bitácora contiene una IP origen, una destino y un peso entre ambos. Los grafos tienen muchas similitudes con los árboles, pues todos los árboles son grafos pero no todos los grafos son árboles, pues por ejemplo un grafo puede ser desconectado. Para representar grafos podemos usar una matriz de adyacencia o una lista de adyacencia, ambos tienen sus ventajas y desventajas, pero considero que la matriz tiene más desventajas que una lista, por ejemplo el borrar una arista en una matriz tiene un tiempo O(1), mientras que una lista es O(n), pero el espacio es más eficiente una lista la ventaja de la matriz, pues en una lista no tiene que ser simétrica, es decir solo necesita O(n) de espacio, al contrario que una matriz pues tiene que ser cuadrada (O(n^2)), por lo que puede quedar mucho espacio vacío. Las operaciones más típicas de un grafo son el crear/borrar un grafo con sus respectivos nodo/arista, buscar un nodo e imprimir el grafo.

Para imprimir un grafo vamos por cada uno de los índices del vector e imprimimos la lista enlazada presente, para crearlo tenemos que tener los nodos y las aristas y hacer un resize() para no desperdiciar memoria, luego declarar una lista enlazada de pares (vértice, peso), llenarla y cargarla en cada uno de los índices del vector, para recorrerlo existen métodos como el Breadth First search que lo recorre en amplitud y el Deep First Search que lo recorre en profundidad, también podemos buscar el camino más corto desde un nodo a otro para esto existen varios algoritmos, pero el que usamos fue el de Dijkstra usando Heaps, este es muy eficiente pues tiene un tiempo de ejecución promedio de O(ElogV), donde E son las aristas y V los vértices o nodos, también existe otra variante que usa el algoritmo de Prim, pero este tiene una complejidad de O(v^2), por lo que es una mejor opción usar heaps.

Referencias:

GeeksforGeeks. (2023a, enero 11). *Difference between Singly linked list and Doubly linked list*. https://www.geeksforgeeks.org/difference-between-singly-linked-list-and-doubly-linked-list/

GeeksforGeeks. (2023b, enero 17). *QuickSort*. https://www.geeksforgeeks.org/quick-sort/

GeeksforGeeks. (2023c, enero 20). *Graph Data Structure And Algorithms*. https://www.geeksforgeeks.org/graph-data-structure-and-algorithms/

GeeksforGeeks. (2023d, enero 27). *Dijkstra s Shortest Path Algorithm Greedy Algo 7*. https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/