NOMBRE: Jose Maria de la Torre

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2133 —Estructuras de Datos y Algoritmos

Informe 4

1 ¿Qué metodo usaste para representar el grafo? ¿Por qué?

Para representar los grafos existen dos formas tipicas. Una es una matriz de adyacencia donde la matriz tiene todos las conexiones entre los distintos vertices (tamaño V^2). La otra opción es la lista de adyacencia que fue la que utilice yo para mi trabajo. Esta consiste en un arreglo (de tamaño la cantidad de vértices en el grafo) que contiene una lista ligada con los vértices adyacentes al vértice correspondiente. Luego en la posición "i" del arreglo habrá una lista ligada con todos los "pesos" de los vértices adyacentes (conectados) al vértice "i".

Utilice esta forma de representar el grafo, ya que al investigar (ítem 1 bibliografia), descubrí que para el algoritmo Prim, es mas eficiente listas de adyacencias ($\mathcal{O}(E \log V)$) por sobre la matriz de adyacencia ($\mathcal{O}(V^2)$).

Además, para grafos poco densos, según el libro guía del curso "Introduction to Algorithms" para representar grafos poco densos (nuestro caso), es mejor utilizar listas de adyacencias. Esto además, confirma lo dicho anteriormente con respecto a las complejidades para Prim para las distintas formas de representar el grafo.

2 ¿Tu algoritmo es codicioso? si no, ¿Pertenece a alguna familia de los algoritmos que hemos estudiado? Justifica

Un algoritmo codicioso si "sigue una heurística consistente en elegir la opción óptima en cada paso local con la esperanza de llegar a una solución general óptima"[1]

El algoritmo si es codicioso. Es codiciosos ya que cada vez que inserta una nueva arista al grafo solución, busca por el la arista con mayor flujo posible y menor costo. Elegir la arista con mayor flujo es la elección codiciosa que realiza mi algoritmo.

[1] https://es.wikipedia.org/wiki/Algoritmo_voraz

3 ¿Cual es la complejidad de tu algoritmo?

La complejidad del algoritmo (Prim) esta directamente esta relacionado con la estructura utilizada para representar el grafo y por el método utilizado para ordenar las aristas. Para mi algoritmo, lo que utilice (como fue dicho anteriormente) es una lista de adyacencia con un max heap.

El algoritmo primero inicializa el min heap, esto tarda $\mathcal{O}(V)$ ya que solamente hay que iterar una vez por sobre todos los vértices. Luego para extraer un vértice del árbol esto tarda $\mathcal{O}(\log V)$ (la altura del heap) y como hay que buscar todos los vértices del grafo, el tiempo en buscar y sacar todos los vértices es $\mathcal{O}(V \log V)$. En cada iteración sobre un vértice nuevo, hay que iterar sobre la lista de adyacencia, lo cual tarda $\mathcal{O}(E)$. Dentro de esta iteración, hay que actualizar el heap (ya que se elimino el nodo raíz) y esto tarda $\mathcal{O}(\log V)$, entonces en total esta parte del algoritmo tarda $\mathcal{O}(E \log V)$.

Luego, la complejidad final sera $\mathcal{O}(E \log V + V \log V)$ lo cual es equivalente a $\mathcal{O}(E \log V)$

4 Bibliografia

- http://www.geeksforgeeks.org/greedy-algorithms-set-5-prims-mst-for-adjacency-list-representation/
- Introduction to Algorithms, 3rd Edition (MIT Press) Thomas H. Cormen
- https://en.wikipedia.org/wiki/Prim