Laboratorio 8

Fernando González

18 de octubre de 2018

Problema 1

1. Teniendo una lista de adyacencia representando un grafo dirigido, cuanto me toma computar las *out-degrees* y las *in-degrees* de cada vértice.

Teniendo una lista de adyacencia dado un grafo dirigido, los $out-degrees$ de un vértice se pueden computar de manera que encontremos el largo de la lista adyacente, tomando en cuenta la suma de todos los tamaños de las listas adyacentes. Por lo tanto, nos tomaría $O(Len[v])$, y si queremos computar el tiempo de encontrar todos los $out-degrees$ de un vértice entonces es: $O(V+E)$.

Por otro lado, para computar el tiempo de los $in-degrees$ de un vértice debemos de encontrar el numero de veces que un vértice aparece en todas las listas de adyacencia. Entonces, lo que nos tomaría computar los $in-degrees$ de un vértice es: $O(VE)$.

2. Describa si utilizaría BFS o DFS para resolver los problemas a continuación:

-. Encontrar mas nodos en una red de BlockChain: DFS ya que si lo utilizamos, estaremos optimizando de cierta manera el recorrido de los nodos conocidos o desconocidos. Por lo que si usamos DFS solo sera necesario recorrer un nodo conocido, y luego prosigue a otro nuevo nodo.

-. Desarrollar un crawler para un motor de búsqueda: BFS, ya que es mas eficiente la búsqueda si es por variedad de temas o de paginas con relación a un tag.

-. Encontrar la salida a un laberinto: DFS, ya que así podremos hallar el camino correcto mas rápido, y sin necesidad de ir probando todos.

-. Sistema de GPS para encontrar caminos de A a B: DFS, la misma idea del laberinto, ya que si se usa de esta manera, se podrá ver cual es la ruta mas eficiente de A a B.

-. Detectar un ciclo dentro de un grafo: DFS, esto es ya que al accesar a los nodos a los que apunta el primer nodo, se puede encontrar al mismo sin necesidad de considerar otros.

Problema 2

Cuál es el running time de BFS si se representa al grafo como una matriz? Cuál es el running time si se representa como una lista?

En una matriz, debemos tomar en cuenta que cuando agarramos un vértice x el cual se encuentra en el nivel 0 de la matriz, todos los vértices adyacentes a el serán de nivel 1. Entonces, luego debemos demarcar todos los vértices adyacentes de todos los vértices del nivel 1, los cuales no tiene nivel para acceder al nivel 2. Por lo tanto, cada vértice pertenece a un nivel en especifico. Y cuando un elemento se encuentra en un nivel, verificamos por sus vértices adyacentes, lo cual hace que el computo sea de $O(|V|)$. Así, como los niveles representan —V— elementos dentro de la ejecución del algoritmo, el running time se convierte en $O(|V|*|V|)$, lo cual simplificado es: $O(|V|^2)$.

Por otro lado, en una lista, a diferencia de una matriz, para indicar que nodos son adyacentes a un vértice ya no es necesario el tiempo $O(|V|)$, ya que en una lista esto ya esta disponible para nosotros, por lo que el tiempo es proporcional a los vértices adyacentes, que en la suma de todos los vértices —V— es —E—. Por lo tanto, el running time del BFS en una lista es de $O(|V| + |E|)$.

Problema 3

Explique como un vértice u de un grafo dirigido puede terminar dentro del *depth-firts-tree* que contenga solo u. Aunque u tenga aristas saliendo y entrenado en el grafo G.

Un vértice u de un grafo dirigido puede terminar dentro del *depth-firts-tree* si de primero se recorre todos los vértices a los que apunta ese vértice v y luego, se recorre el vértice. Dado esto, al momento de que se revisa v, no se es necesario analizarlo del todo porque todo lo que apunta a ese vértice ya fue recorrido, y es así como un vértice u de un grafo dirigido puede terminar dentro del *depth-firts-tree*.