Departamento de Cs. e Ingeniería de la Computación
Universidad Nacional del Sur

Algoritmos y Complejidad

Trabajo Práctico 7
Grafos I

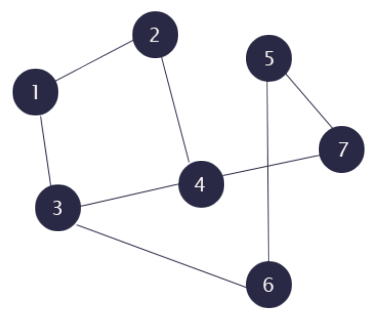
Primer cuatrimestre de 2017

1. *BFS: Recorrido por niveles*

   a) Escribir pseudocódigo del recorrido *BFS* (determinista) para las representaciones de grafo como listas de adyacencia y como matriz de adyacencia.

   b) Realizar el análisis de tiempo de ejecución para cada una de las variantes.

   c) Mostrar como quedarían los arreglos `nivel` y `padre` luego de un recorrido *BFS* con origen en el nodo 1 sobre el siguiente grafo



2. Para el algoritmo de búsqueda en profundidad sobre grafos no dirigidos:

   a) Realizar el análisis del tiempo de ejecución suponiendo que el grafo está representado por una lista de adyacencia.

   b) ¿Qué sucedería con el tiempo de ejecución si el grafo es representado por medio de una matriz de adyacencia?

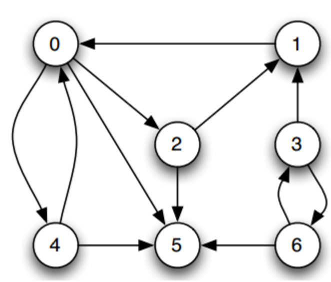
   c) Indicar cómo modificaría el algoritmo para encontrar los componentes conexos de un grafo.

3. Para el grafo que se muestra a continuación

   a) Mostrar la foresta obtenida luego de realizar un recorrido *DFS* visitando los nodos en orden creciente de numeración.

   b) Agregar la numeración de preorden (descubrimiento) y postorden (finalización) a cada nodo.

   c) Clasificar los arcos del grafo por categoría: de foresta, hacia atrás, hacia adelante, cruzado.



4. Dada la siguiente numeración de preorden y postorden de un recorrido *DFS*

Nodo	u	v	w	x	y	z
preorden	1	2	9	4	3	10
postorden	8	7	12	5	6	11

   Determinar de qué categoría pueden ser los arcos (v, y),(w, y),(z, z),(x, v),(u, x).

5. Probar que en un recorrido en profundidad de grafos no dirigidos nunca se producen arcos hacia adelante o arcos que cruzan. Dicho de otra forma, todo arco del grafo es un arco de la foresta o un arco hacia atrás.

6. Mostrar un ejemplo de un grafo dirigido G en donde la foresta de su recorrido en profundidad contenga un árbol con un único nodo u, aunque u tenga arcos incidentes y salientes en G.

7. Un grafo no dirigido $G = \langle N, A\rangle$ se dice *bipartito* si N puede particionarse en dos conjuntos N_1, N_2 tales que si $(u, v) \in A$ entonces vale que $u \in N_1$ y $v \in N_2$, o que $u \in N_2$ y $v \in N_1$. Usando el algoritmo de búsqueda por niveles, encontrar un algoritmo eficiente para determinar si un grafo no dirigido es bipartito.

8. Escribir un algoritmo que determine si un grafo no dirigido $G = \langle N, A\rangle$ contiene un ciclo, cuyo orden del tiempo de ejecución sea del $O(n)$, independiente de la cantidad de arcos del grafo.

9. Extender el algoritmo anterior para un grafo dirigido. ¿Cuál es el orden del tiempo de ejecución?

10. *Orden topológico*

    a) Definir el concepto de orden topológico de un grafo dirigido acíclico.

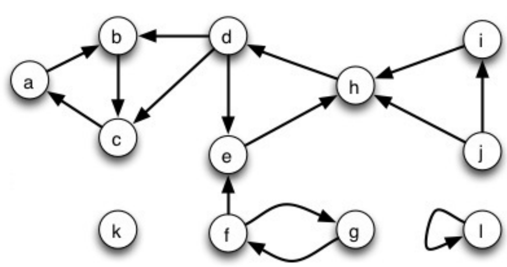
    b) Demostrar que en un grafo dirigido acíclico, para todo arco (u, v) se cumple que $f[v] < f[u]$.

    c) A partir del inciso anterior, demostrar que la lista de nodos ordenados por orden decreciente de f resulta en un orden topológico.

    d) Mostrar pseudocódigo de un algoritmo que resuelva el problema de encontrar un orden topológico en un tiempo de $O(n + a)$.

11. *Componentes fuertemente conexas*

    Dado el siguiente grafo dirigido



    a) Realizar un recorrido *DFS* y ordenar los nodos por tiempo de finalización descendiente.

    b) Mostrar la foresta resultante de realizar un recorrido *DFS* sobre el grafo traspuesto visitando los nodos en el orden del inciso anterior.

    c) A partir de la foresta obtenida en el inciso anterior. Determinar las componentes fuertemente conexas del grafo y construir el grafo de componentes.