MA4701-1 - Optimización Combinatorial
Profesor: José Soto
Auxiliares: Nicolás Sanhueza - Christian von Borries

Auxiliar N°1

22 de agosto de 2013

Un **grafo simple** $G = (V(G), E(G))$ es un par donde $V(G)$ es un conjunto finito y $E(G) \subseteq \mathcal{P}(V(G))$ es tal que para todo $e \in E(G)$, $|e| = 2$. $V(G)$ son los **vértices** y $E(G)$ las **aristas** del grafo G. Si $e = \{x, y\}$ es una arista, la denotamos xy o yx. En este caso, decimos que los vértices x e y son **adyacentes** y que la arista e es **incidente** a x e y. El **grado** de un vértice v es el número de aristas que son incidentes a él, y se denota $d(v)$. Si H y G son grafos y $V(H) \subseteq V(G)$ y $E(H) \subseteq E(G)$, diremos que H es **subgrafo** de G, y lo denotaremos $H \subseteq G$. Sean $F \subseteq E(G)$ y $U \subseteq V(G)$. Decimos que F **genera** U si para todo vértice $u \in U$, existe una arista $e \in F$ incidente a U. Decimos que un grafo G' genera U si $E(G')$ genera U. Un subgrafo $G' \subseteq G$ es un **subgrafo generador** si G' genera $V(G)$. Un **camino** P es un grafo no vacío tal que $V(P) = \{x_0, x_1, \ldots, x_k\}$ y $x_i \neq x_j$ si $i \neq j$; y $E(P) = \{x_0x_1, x_1x_2, \ldots, x_{k-1}x_k\}$. Dado un grafo G y u, v vértices en G, un (u, v)-camino es un subgrafo de G que es un camino y $x_0 = u$ y $x_k = v$. Diremos que un grafo G es **conexo** si para todo u, $v \in V(G)$, existe un (u, v)-camino en G. Un grafo no conexo se dirá **disconexo**. Un **ciclo** C es un grafo no vacío tal que $V(C) = \{x_0, x_1, \ldots, x_k\}$, $k \geq 3$ y $x_i \neq x_j$ si $i \neq j$; y $E(C) = \{x_0x_1, x_1x_2, \ldots, x_{k-1}x_k, x_kx_0\}$. Un grafo se dirá **acíclico** o **bosque** si no tiene a un ciclo como subgrafo. Diremos que un grafo es un **árbol** si es conexo y acíclico.

P1. Sea G un grafo tal que su grado mínimo es mayor que 2. Demuestre que contiene un ciclo.

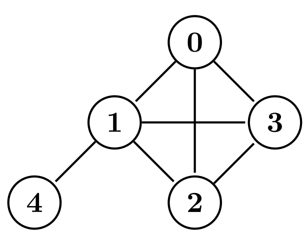
P2. Sea T un grafo. Muestre que son equivalentes,

1. T es un árbol.
2. Todo par de vértices en T están unidos por un único camino en T.
3. T es conexo y para toda $e \in E(G)$, $T - e := (V(T), E(T) \setminus \{e\})$ es disconexo.
4. T es acíclico y para toda $e \notin E(G)$, $T + e := (V(T), E(T) \cup \{e\})$ contiene un ciclo.

P3. Muestre que todo grafo conexo tiene un árbol generador.

P4. Sea G conexo con n vértices. Muestre que G es un árbol si y sólo si tiene $n - 1$ aristas.

Una **lista enlazada** consiste en una secuencia de «nodos», en los que se guardan campos de datos arbitrarios y uno o dos punteros al nodo anterior o posterior. Por otro lado, un **arreglo** (o *array*) es una zona de almacenamiento continuo, que contiene una serie de elementos del mismo tipo, permitiendo acceso a los elementos en un orden arbitrario.



$$A_G = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

$$\begin{array}{ccl} 0 & : & 1 \to 2 \to 3 \\ 1 & : & 0 \to 3 \to 2 \to 4 \\ 2 & : & 1 \to 0 \to 3 \\ 3 & : & 1 \to 2 \to 0 \\ 4 & : & 1 \end{array}$$

Figura 1: Una representación dibujística de un grafo dado por $V(G) = \{0, 1, 2, 3, 4\}$ y $E(G) = \{\{0, 1\}, \{0, 2\}, \{0, 3\}, \{1, 2\}, \{1, 3\}, \{1, 4\}, \{2, 3\}\}$. A un lado su matriz de adyacencia, y una posible lista de adyacencia.

Dado un grafo G en n vértices, buscamos una forma para representarlo en un computador. Suponemos sin pérdida de generalidad que $V(G) = \{0, 1, \ldots, n-1\}$, así que cada vértice tiene una asociación única a un entero. Una **matriz de adyacencia** es un arreglo de enteros A en dos dimensiones, de tamaño $n \times n$ tal que $A[i, j] = 1$ si y sólo si $ij \in E(G)$, $A[i, j] = 0$ en otro caso. Una **lista de adyacencia** es un arreglo de tamaño n de listas enlazadas de enteros, tal que $L[i]$ es una lista de enteros que contiene todos los vértices adyacentes a i en G.

P5. Implemente en seudocódigo, en una estructura de datos adecuada, los algoritmos de búsqueda en amplitud (BFS, por las siglas de *breadth first search*) y búsqueda en profundidad (DFS, por las siglas de *depth first search*) para encontrar árboles cobertores en grafos conexos. Explique cómo usar alguno de estos algoritmos para decidir si un grafo es o no conexo.