Práctica 2: Introducción a la teoría algorítmica de grafos

Demostración de propiedades simples sobre grafos

1. Demostrar, usando inducción en la cantidad de aristas, que todo digrafo D satisface

$$\sum_{v \in V(D)} d_{in}(v) = \sum_{v \in V(D)} d_{out}(v) = |E(D)|.$$

2. Demostrar, usando la técnica de reducción al absurdo, que todo grafo tiene al menos dos vértices del mismo grado. **Ayuda:** prestar atención a la secuencia ordenada de los grados de los vértices.

3. Un *grafo orientado* (ver Figura 1) es un digrafo D tal que al menos uno de $v \to w$ y $w \to v$ no es una arista de D, para todo $v, w \in V(D)$. En otras palabras, un grafo orientado se obtiene a partir de un grafo no orientado dando una dirección a cada arista. Demostrar en forma constructiva que existe un único grafo orientado cuyos vértices tienen todos grados de salida distintos. **Ayuda:** aprovechar el ejercicio anterior y observar que el absurdo no se produce para un único grafo orientado.

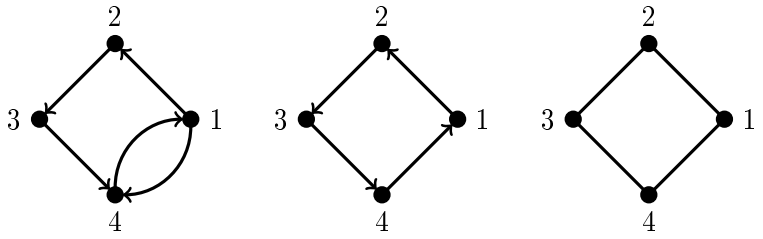

Figura 1. *(a) Un digrafo que no es grafo orientado porque tanto* $1 \to 4$ *como* $4 \to 1$ *son aristas; (b) un grafo orientado que se puede obtener dando orientaciones a las aristas del grafo (c).*

4. Demostrar, usando inducción en la cantidad de vértices, que todo grafo de n vértices que tiene más de $((n-1)(n-2))/2$ aristas es conexo. Opcionalmente, puede demostrar la misma propiedad usando otras técnicas de demostración.

5. Sean P y Q dos caminos de un grafo G que unen un vértice v con otro w. Demostrar en forma directa que G tiene un ciclo cuyas aristas pertenecen a P o Q. **Ayuda:** denotar $P = v_0, \ldots, v_p$ y $Q = w_0, \ldots, w_q$ con $v_0 = w_0 = v$ y $v_p = w_q = w$. Definir explícitamente cuáles son los subcaminos de P y Q cuya unión forman un ciclo.

6. Sea G un grafo conexo. Demostrar por el contrarrecíproco que todo par de caminos simples de longitud máxima de G tienen un vértice en común. **Ayuda:** suponer que hay dos caminos disjuntos en vértices de igual longitud y definir explícitamente un camino que sea más largo que ellos.

Representación de grafos y digrafos

En la práctica vamos a usar dos representaciones básicas para un (di)grafo G. Por un lado, el *conjunto de aristas* que es un par (V, E) donde $V = V(G)$ y $E = E(G)$. Por otra parte, el *conjunto de vecindarios* (o conjunto de *adyacencias*) que es un par (V, N) donde V es un conjunto de vértices y N es una función que para cada $v \in V(G)$ retorna el conjunto $N(v)$ de vértices adyacentes a v, i.e., su vecindario. (En el caso de digrafos, tenemos dos funciones, para vecindario de entrada y salida.) Sin pérdida de generalidad, vamos a suponer que los vértices son números entre 0 y $|V(G)| - 1$.
Las instancias de varios de los problemas computacionales que consideramos utilizan (di)grafos generales (i.e., sin propiedades particulares). A partir de ahora, vamos a suponer que estas instancias se representan con su conjunto de aristas. Las dos razones para esta suposición son: 1. los conjuntos de aristas no consumen espacio adicional con respecto al tamaño de un grafo; y 2. el conjunto de aristas no supone ninguna propiedad adicional sobre los datos. En los casos en donde las instancias sean grafos con alguna estructura especial (i.e., árboles), la representación de la instancia puede ser distinta (e.g., una función que dado un vértice no raíz retorna su único padre).

7. Discutir (brevemente) las ventajas y desventajas en cuanto a la complejidad temporal y espacial de las siguientes implementaciones de un conjunto de vecindarios para un grafo G, de acuerdo a las siguientes operaciones:

   Operaciones

   a) Inicializar la estructura a partir de un conjunto de aristas de G.
   b) Determinar si dos vértices v y w son adyacentes.
   c) Recorrer y/o procesar el vecindario $N(v)$ de un vértice v dado.
   d) Insertar un vértice v con su conjunto de vecinos $N(v)$.
   e) Insertar una arista vw.
   f) Remover un vértice v con todas sus adyacencias.
   g) Remover una arista vw.
   h) Mantener un orden de $N(v)$ de acuerdo a algun invariante que permita recorrer cada vecindario en un orden dado.

   Estructuras de datos

   a) $N(v)$ se representa con una secuencia (vector o lista) que en cada posición tiene el conjunto $N(v)$ implementado sobre una secuencia (lista o vector). Cada vértice es una estructura que tiene un índice para acceder en $O(1)$ a $N(v)$. Esta representación se conoce comúnmente como *lista de adyacencias*.
   b) ídem anterior, pero cada $w \in N(v)$ se almacena junto con un índice a la posición que ocupa v en $N(w)$. Esta representación también se conoce como *lista de adyacencias*, pero tiene información para implementar operaciones dinámicas.
   c) $N(v)$ se representa con un vector que en cada posición i tiene un vector booleano A_i con $|V(G)|$ posiciones tal que $A_i[j]$ es verdadero si y solo si i es adyacente a j. Esta representación se conoce comúnmente como *matriz de adyacencias*.
   d) $N(v)$ se representa con un vector que en cada posición tiene el conjunto $N(v)$ implementado con una tabla de hash. Esta representación es un mix entre las representaciones clásicas de matriz de adyacencias y lista de adyacencias.

8. Demostrar que las representaciones por listas de adyacencias de un grafo (ejercicio anterior) se pueden construir en $O(n+m)$ tiempo. ¿Qué ocurre si se usa una tabla de hash? ¿Y si se construye una matriz de adyacencias?

Algoritmos sobre grafos y digrafos (opcionales)

9. Recordar que el *vecindario* de un vértice v es el conjunto $N(v)$ que contiene a todos los vértices adyacentes a v. El *vecindario cerrado* es $N[v] = N(v) \cup \{v\}$. Dos vértices u y v son *gemelos* cuando $N(u) = N(v)$, mientras que son *mellizos* cuando $N[u] = N[v]$ (Figura 2).

   a) Observar que las relaciones de gemelos y mellizos son relaciones de equivalencia (i.e., son reflexivas, transitivas y simétricas).

   b) Probar que el siguiente algoritmo encuentra la partición de $V(G)$ en vértices mellizos. **Ayuda:** demostrar por invariante que, luego del paso i, u y w pertenecen al mismo conjunto de $\mathcal{P}_i$ si y sólo si $N[u] \cap \{v_1, \ldots, v_i\} = N[w] \cap \{v_1, \ldots, v_i\}$.

   1. Sea $\mathcal{P}_0 = \{V(G)\}$ ($\mathcal{P}$ es un conjunto de conjuntos)
   2. Sea $v_1, \ldots, v_n$ un ordenamiento cualquiera de $V(G)$.
   3. Para i desde 1 hasta n:
   4. Poner $\mathcal{P}_i := \{W \cap N[v_i] \mid W \in \mathcal{P}_{i-1}\} \cup \{W \setminus N[v_i] \mid W \in \mathcal{P}_{i-1}\}$.
   5. $\mathcal{P}_n$ es la partición buscada.

   c) Describir la implementación del algoritmo, especificando las estructuras de datos utilizadas. La mejor implementación que conocemos tiene complejidad temporal $O(n + m)$.

   d) ¿Qué debería modificarse para que el algoritmo encuentre la partición en vértices gemelos?

10. En este ejercicio diseñamos un algoritmo para encontrar ciclos en un digrafo. Decimos que un digrafo es *acíclico* cuando no tiene ciclos dirigidos. Recordar que un (di)grafo es *trivial* cuando tiene un sólo vértice.

    a) Demostrar con un argumento constructivo que si todos los vértices de un digrafo D tienen grado de salida mayor a 0, entonces D tiene un ciclo.

    b) Diseñar un algortimo que permita encontrar un ciclo en un digrafo D cuyos vértices tengan todos grado de salida mayor a 0.

    c) Explicar detalladamente (sin usar código) cómo se implementa el algoritmo del inciso anterior. El algoritmo resultante tiene que tener complejidad temporal $O(n + m)$.

    d) Demostrar que un digrafo D es acíclico si y solo si D es trivial o D tiene un vértice con $d_{out}(v) = 0$ tal que $D \setminus \{v\}$ es acíclico.

    e) A partir del inciso anterior, diseñar un algoritmo que permita determinar si un grafo D tiene ciclos. En caso negativo, el algoritmo debe retornar una lista $v_1, \ldots, v_n$ de vértices tales que $d_{out}(v_i) = 0$ en $D \setminus \{v_1, \ldots, v_{i-1}\}$ para todo i. En caso afirmativo, el algoritmo debe retornar un ciclo.

    f) Explicar detalladamente (sin usar código) cómo se implementa el algoritmo del inciso anterior. El algoritmo resultante tiene que tener complejidad temporal $O(n + m)$.

11. Un *triángulo* de un grafo G es una tripla $\{v, w, z\}$ que induce un subgrafo completo (de tamaño 3; ver Figura 2). Considerar los siguientes algoritmos para decidir si un grafo G de n vértices y $m > n$ aristas tiene un triángulo.

Algoritmo cúbico

1. Computar la matriz de adyacencias A de G.
2. Retornar verdadero si existen $v, w, z \in V(G)$ tales que $A_{vw}A_{wz}A_{vz} = 1$.

Algoritmo cuadratico

1. Computar las listas de adyacencias N de G.
2. Para cada $v \in V(G)$:
3. Marcar cada $w \in N(v)$.
4. Retornar verdadero si existe $wz \in E(G)$ tal que w y z están marcados.
5. Desmarcar cada $w \in N(v)$.

Algoritmo subcuadrático

1. Computar las listas de adyacencias N de G.
2. Para cada $v \in V(G)$ que tenga $d(v) \geq \sqrt{m}$:
3. Marcar cada $w \in N(v)$.
4. Retornar verdadero si existe $wz \in E(G)$ tal que w y z están marcados.
5. Desmarcar cada $w \in N(v)$.
6. Para cada $v \in V(G)$ que tenga $d(v) < \sqrt{m}$:
7. Marcar cada $w \in N(v)$.
8. Para cada $w \in N(v)$ que tenga $d(w) < \sqrt{m}$:
9. Retornar verdadero si existe $z \in N(w)$ que esté marcado.
10. Desmarcar cada $w \in N(v)$.

a) Argumentar por qué cada algoritmo es correcto.

b) Demostrar que el algoritmo cúbico requiere tiempo $\Theta(n^3)$, el algoritmo cuadrático requiere tiempo $\Theta(m^2)$ y el algoritmo subcuadrático requiere tiempo $O(m^{3/2})$. **Ayuda:** recordar que $O(\sum_{v \in V(G)} d(v)) = O(m)$; para el algoritmo subcuadrático, demostrar primero que todo grafo tiene $O(\sqrt{m})$ vértices con grado al menos $\sqrt{m}$.

c) Determinar un mejor y un peor caso para cada uno de los algoritmos.

Ejercicios integradores (opcionales)

12. Un grafo G es *threshold* si para cada par de vértices $u, v \in V(G)$ tales que $d(u) \leq d(v)$ ocurre que $N(u) \subseteq N(v)$ o $N[u] \subseteq N[v]$.

a) Demostrar que si G es un grafo threshold, entonces los vértices de grado k son todos mellizos entre sí, o todos gemelos entre sí, para todo $0 \leq k \leq n-1$.

b) Demostrar que si G es un grafo threshold, entonces tiene algún vértice de grado 0 o alguno de grado $n-1$.

c) Demostrar que si G es un grafo threshold, entonces $G \setminus \{v\}$ es un grafo threshold para todo $v \in V(G)$.

d) Decimos que un grafo H tiene una descomposición threshold si $V(H)$ admite un ordenamiento $v_1, \ldots, v_n$ tal que v_i es un vértice de grado 0 o de grado $i-1$ en el subgrafo de H inducido por $\{v_1, \ldots, v_i\}$. Usando b. y c., demostrar que G es un grafo threshold si y sólo si admite una descomposición threshold.

e) Proponga una estructura de datos para un grafo threshold G basado en los ítems anteriores. La estructura de datos debe ocupar $O(n)$ bits. Indique cómo se determina si dos vértices son adyacentes.

f) Diseñe un algoritmo que, dado un grafo G cualquiera, determine si G es o no threshold. En caso afirmativo, debe mostrar cómo se construye la representación del inciso anterior.

g) Sea $v_1, \ldots, v_n$ una descomposición threshold de G y $w_1, \ldots, w_n$ una descomposición threshold de H. Demuestre que G es isomorfo a H si y solo si $f(v_i) = w_i$ es un isomorfismo. **Ayuda:** intente hacer inducción en n.

13. Decimos que un digrafo (con *loops*) tiene forma de ρ cuando todos sus vértices tienen grado de salida igual a 1 (Figura 3).

a) Demostrar en forma constructiva que si un digrafo es conexo y tiene forma de ρ entonces tiene un único ciclo dirigido. Notar que si $v \to v$ es un *loop*, entonces v, v es un ciclo. Recordar que un digrafo es conexo cuando su grafo subyacente es conexo.

b) Diseñar un algoritmo para encontrar todos los ciclos de un digrafo con forma de ρ (no necesariamente conexo). **Ayuda:** notar que los vértices con grado de entrada 0 no estan en ciclos; luego, se pueden sacar iterativamente. Demostrar que todos los vértices tienen grado de entrada 1 en el grafo resultante y, por lo tanto, todos los vértices pertenecen a un ciclo.

Consideremos un período de tiempo circular $[0, T]$ tal que llegado el momento T se vuelve a contabilizar el tiempo 0 (e.g., un día, una semana, etc). Dentro del tiempo $[0, T]$ se encuentran definidas n actividades, la i-ésima de las cuales se desarrolla empezando en el instante s_i y terminando en el instante t_i (si $s_i > t_i$, entonces la actividad contiene el instante $0 = T$).

En el problema de selección de actividades periódicas se busca determinar la máxima cantidad de actividades que un agente puede realizar rutinariamente, suponiendo que el agente es capaz de realizar una única actividad en cada instante. Formalmente, el objetivo es determinar la máxima razón x/y para una secuencia circular de actividades $A_1, \ldots, A_x$ que se realiza en y períodos completos cuando A_{i+1} se inicia lo antes posible una vez terminado A_i, para todo $1 \leq i \leq x$ (con $A_{x+1} = A_1$). Considere la siguiente estrategia golosa para decidir qué actividad j conviene elegir si se elige la actividad i: tomar j como una actividad que no se solapa con i y cuyo tiempo de finalización es el primero desde t_i en un recorrido del tiempo en el sentido de las agujas del reloj.

Definir el digrafo de actividades D que tiene un vértice i por cada actividad y que tiene un arco (arista) $i \to j$ cuando j es la elección golosa que se toma si se elige i.

c) Observar que D es un digrafo con forma de ρ.

d) (Difícil) Demostrar que algún ciclo de D es una solución al problema de selección de actividades. **Ayuda:** demostrar por inducción en i ($i \leq x$) que existe una solución $A_1, \ldots, A_x$ donde A_{i+1} es la elección golosa para A_i, tomando $A_{x+1} = A_1$.

e) Usando los resultados anteriores, dar un algoritmo para resolver el problema de selección de actividades, suponiendo que las actividades se describen usando un conjunto de pares $[s_i, t_i]$ donde $0 \leq s_i, t_i \leq T$, $s_i \neq t_i$ y $T = 2n$.

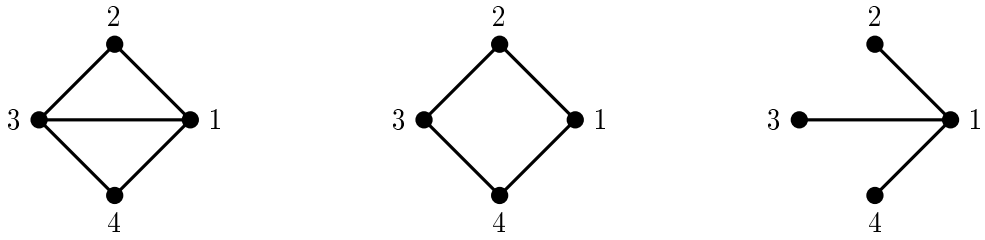

Figura 2. *El grafo diamante (izquierda), el grafo C_4 (centro) y el grafo garra (derecha). Los vértices 1 y 3 son mellizos en el diamante porque $N[1] = N[3] = \{1, 2, 3, 4\}$ mientras que 2 y 4 son gemelos porque $N(2) = N(4) = \{1, 3\}$. En la garra, $N(2) = N(3) = N(4) = \{1\}$ y, por lo tanto, 2, 3 y 4 son gemelos, mientras que en C_4 la partición en gemelos es $\{2, 4\}$ y $\{1, 3\}$. El diamante tiene 2 triángulos $\{1, 2, 3\}$ y $\{1, 3, 4\}$ mientras que C_4 y la garra no tienen triángulos. Notar que el diamante es threshold porque $N(2) = N(4)$ y $N[2] \subseteq N[1] = N[3]$; ciertamente $(2, 4, 1, 3)$ es una descomposición threshold del diamante. En cambio, C_4 no es threshold porque tanto $N(1)$ y $N(2)$ como $N[1]$ y $N[2]$ son incomparables por inclusión. Finalmente, la garra es threshold (por qué?).*

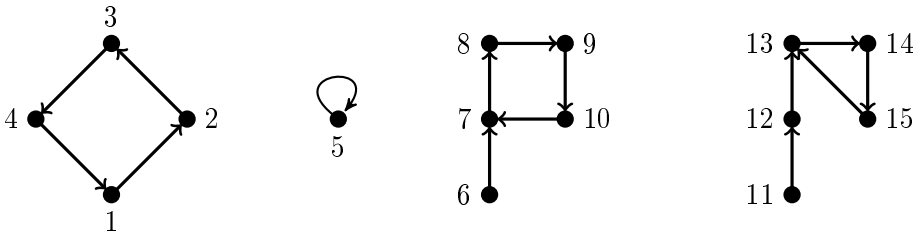

Figura 3. *Un digrafo disconexo con forma de ρ; cada componente conexa tiene forma de ρ.* ***Ayuda:*** *notar que si se sacan los vértices con grado de entrada 0 en forma iterativa, entonces cada componente es un ciclo dirigido.*