Programación 3
Año 2024

Práctico 3

Los ejercicios con una sección del libro asignada son de guía, para realizar durante su lectura, mientras que los ejercicios *entre asteriscos* son de evaluación.

Ejercicio 1 (Sección 3.1 - Definiciones básicas y aplicaciones)**.** Una empresa de servicios de Internet debe diseñar una red de antenas para proporcionar conexión 5G a toda una ciudad. En este sentido, la empresa determina que se deben instalar n antenas para cubrirla. La tarea pendiente es determinar cómo conectar las antenas entre sí mediante cables, de modo que haya conexión entre todas las antenas con la menor cantidad de conexiones.

(a) ¿Cómo modelaría este problema mediante un grafo?
(b) Una persona empleada por la empresa sugiere una solución que conecta todas las antenas con n conexiones. ¿Es esta solución óptima? Justifique.
(c) Otra persona propone una solución con $n - 2$ conexiones entre si. ¿Es esta solución válida? Justifique.
(d) Demuestre que la solución óptima tiene exactamente $n - 1$ conexiones.

Ejercicio 2 (Sección 3.2 - Conectividad y recorrida de grafos)**.** Consideramos el grafo de la figura 2.1.

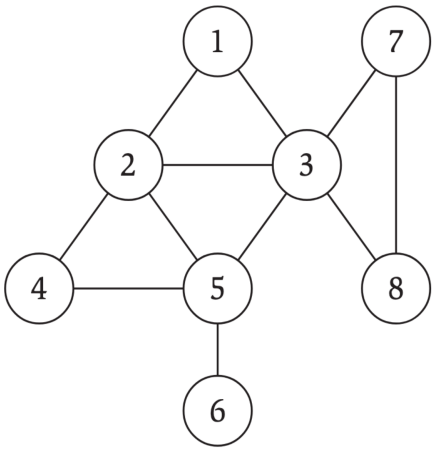


Figura 2.1: Grafo.

(a) Escriba 3 algoritmos distintos para recorrer un grafo.

(b) Utilice el algoritmo BFS para recorrer el grafo de la figura 2.1 empezando en el nodo 8 y muestre el árbol BFS resultante. ¿Este es el único árbol BFS posible para este grafo y este nodo inicial? Justifique.

(c) Utilice el algoritmo DFS para recorrer el grafo de la figura 2.1 empezando en el nodo 6 y muestre el árbol DFS resultante. ¿Este es el único árbol DFS posible para este grafo y este nodo inicial? Justifique.

(d) ¿Existe una recorrida DFS que dé como resultado el árbol construido en la parte (b)? Justifique.

(e) ¿Existe una recorrida BFS que dé como resultado el árbol construido en la parte (c)? Justifique.

Ejercicio 3 (Sección 3.3 - Implementando la recorrida de grafos mediante Colas y Pilas)**.**

(a) Implemente el algoritmo DFS de forma iterativa utilizando una pila.

(b) ¿Depende el árbol de recorrida DFS de su implementación? ¿No está el algoritmo sub-especificado? Justifique.

(c) ¿Qué modificaciones se le deben realizar al algoritmo DFS para que retorne el árbol de recorrida?

Ejercicio 4 (*Kleinberg & Tardos, Ex. 3.2*)**.** Dé un algoritmo para detectar si un grafo G contiene un ciclo. Si la respuesta es afirmativa, el algoritmo debe mostrar un ciclo como evidencia (no todos los ciclos que existan sino solo uno de ellos). Su algoritmo debe admitir una implementación cuyo tiempo de ejecución sea $O(m + n)$, donde n es la cantidad de vértices de G y m es la cantidad de aristas de G.

(a) Presente un algoritmo basado en DFS que resuelve el problema; reescriba cualquier algoritmo que utilice del libro de referencia.

(b) Demuestre la corrección de su algoritmo, esto es, que su algoritmo responde afirmativamente si y solo si G contiene un ciclo y, en dicho caso, que la evidencia mostrada es en efecto un ciclo de G.

Sugerencia:

- Demuestre que si su algoritmo responde afirmativamente, genera como salida una secuencia de vértices (la evidencia pedida) que es un ciclo de G.

- Demuestre que si G contiene un ciclo, su algoritmo responde afirmativamente.

(c) Demuestre que su algoritmo admite una implementación cuyo tiempo de ejecución es $O(m+n)$.
Sugerencia: Realice un análisis similar al de algoritmos para recorrida de grafos.

(d) Explique si es posible resolver el problema sin basarse en DFS.

Ejercicio 5 (*Kleinberg & Tardos, Ex. 3.9*)**.** Una red de comunicaciones consta de n equipos y m enlaces de conexión punto a punto entre ellos. Modelamos esta red como un grafo conexo $G = (V, E)$, donde V es el conjunto de equipos de la red y E contiene m aristas, cada una representando un enlace. En dicha red existe dos equipos, s y t, que están a más de $n/2$ enlaces de distancia. En términos del grafo G, el camino más corto entre los nodos s y t tiene un largo estrictamente mayor a $n/2$ (o sea, el largo es mayor o igual a $\lfloor \frac{n}{2} \rfloor + 1$).

(a) Demuestre que existe un equipo que si se retira de la red deja desconectados a s y t. En otras palabras, existe un nodo que si es eliminado de G deja a los nodos s y t en componentes conexas diferentes.
Sugerencia: Sea L_i, $0 \leq i < n$, el conjunto de nodos de G que están a distancia i de s. Muestre que existe al menos un valor de i, $1 \leq i \leq \left\lfloor \frac{n}{2} \right\rfloor$, tal que L_i contiene un solo nodo.

(b) Escriba un algoritmo que encuentra dicho nodo; debe admitir una implementación cuyo tiempo de ejecución sea $O(m+n)$.

(c) Muestre que su algoritmo admite una implementación con tiempo de ejecución que es $O(m+n)$.

Ejercicio 6 (*Kleinberg & Tardos, Ex. 3.10*)**.** En un determinado grafo cada vértice representa una persona vinculada a un escándalo del mundo del espectáculo y las aristas representan algún tipo de relación entre ellas. Un analista está interesado en investigar detenidamente la conexión entre pares de personas u, v que están conectadas por caminos cortos en este grafo. Está claro que la conexión entre algunos de estos pares puede ser puramente casual, por lo cual, además de estudiar el largo del camino más corto entre u, v, decide estudiar la *cantidad* de caminos de largo mínimo entre u, v.

Consideramos un grafo G y un par de vértices u, v de G.

(a) Escriba un algoritmo que calcula la cantidad de caminos de largo mínimo entre u y v (no los caminos en sí mismos, solo la cantidad). El algoritmo debe admitir una implementación cuyo tiempo de ejecución sea $O(m+n)$, donde n es la cantidad de vértices de G y m es la cantidad de aristas de G.

Sugerencia: Diseñe un algoritmo que para cada vértice w calcule la cantidad de caminos de largo mínimo, N_w, desde el vértice u hasta el vértice w.

(b) Demuestre la corrección de su algoritmo.

Sugerencia: Escriba una recurrencia para N_w, definiendo un paso base apropiado y expresando N_w, para $w \neq u$, en función de los valores N_t para vértices t que ocurren inmediatamente antes que w en un camino de largo mínimo desde u hasta w. Demuestre la corrección de esta recurrencia y muestre que su algoritmo la calcula correctamente.

(c) Demuestre que su algoritmo admite una implementación cuyo tiempo de ejecución es $O(m + n)$.

Ejercicio 7 (Sección 3.4 - Testeando bipartición: Una aplicación del algoritmo BFS)**.**

(a) Proponga un problema de la realidad que pueda ser modelado mediante un grafo G, y cuya solución sea a equivalente a determinar si el grafo es bipartito.

(b) Demuestre que si G tiene un ciclo con cantidad de aristas impar, entonces no es bipartito.

(c) Escriba un algoritmo basado en DFS que permita determinar si un grafo es bipartito.

Ejercicio 8 (*Kleinberg & Tardos, Ex. 3.4*)**.** Un coleccionista de mariposas regresa de una expedición con n especímenes que se sabe que son de dos especies diferentes. Como es muy difícil decidir directamente a qué especie pertenece una mariposa mirándola individualmente, el coleccionista analiza cada par de mariposas con el objetivo de juzgar si son de la misma especie o no. Si está convencido, establece un juicio y etiqueta el par como *iguales* o como *diferentes*; si tiene dudas no establece juicio y no etiqueta el par. Ahora quiere saber si el conjunto de m juicios que estableció es consistente, o sea si es posible asignar a cada mariposa una de dos especies, A o B, de manera que para todo par etiquetado como *iguales* se cumple que las mariposas fueron asignadas a la misma especie y para todo par etiquetado como *diferentes* se

cumple que las mariposas fueron asignadas a especies distintas.

(a) Si cada vez que estableció un juicio el par fue etiquetado como *iguales* ¿es posible que el conjunto de juicios sea inconsistente? Si no es posible explique por qué; si es posible escriba un algoritmo que resuelva el problema.

(b) Si cada vez que estableció un juicio el par fue etiquetado como *diferentes* ¿es posible que el conjunto de juicios sea inconsistente? Si no es posible explique por qué; si es posible escriba un algoritmo que resuelva el problema.

(c) Para el caso general (puede haber pares etiquetados como *iguales* y pares etiquetados como *diferentes*) escriba un algoritmo que resuelve el problema.

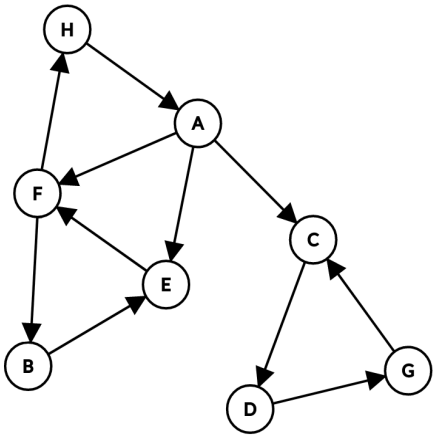
Sugerencia: Diseñe un algoritmo que asigne una de dos especies a cada mariposa.

Todos los algoritmos deben admitir una implementación cuyo tiempo de ejecución sea $O(m+n)$. Reescriba cualquier algoritmo que utilice el libro de referencia.

(d) Demuestre la corrección de sus algoritmos.

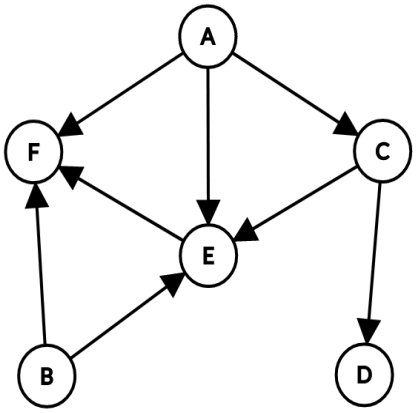
(e) Demuestre que sus algoritmos admiten una implementación cuyo tiempo de ejecución es $O(m+n)$.

Ejercicio 9 (Sección 3.5 - Conectividad en grafos dirigidos)**.** Consideramos el grafo G de la siguiente figura.



(a) Ejecute BFS desde el nodo A.
(b) Ejecute DFS desde el nodo B.
(c) A partir del resultado de las partes anteriores, ¿qué puede decir de las componentes fuertemente conexas de A y B? Justifique.
(d) Ejecute DFS desde el nodo A en G^{rev}.
(e) Encuentre todas las componentes fuertemente conexas de G. Justifique.

Ejercicio 10 (Sección 3.6 - Grafos dirigidos y acíclicos, y orden topológico)**.** Consideramos el grafo G de la siguiente figura.



(a) ¿Qué nodos no tienen aristas entrantes en el grafo?
(b) ¿Qué nodos no tienen aristas entrantes luego de eliminar los nodos de la parte anterior?
(c) Repita las partes (a) y (b) en G^{rev}.
(d) Usando las partes anteriores encuentre al menos cuatro ordenes topológicos distintos para G.
(e) ¿Estos son todos los órdenes topológicos posibles para G?
(f) Describa un algoritmo para encontrar un orden topológico y demuestre que existe una implementación que tiene tiempo de ejecución O(m+n).

Ejercicio 11 (*Kleinberg & Tardos, Ex. 3.3*)**.** Dado un grafo dirigido arbitrario G (no necesariamente un DAG), se quiere obtener alguno de los siguientes resultados:

- un orden topológico para G,

- un ciclo de G.

(a) Presente un algoritmo que resuelve el problema; reescriba cualquier algoritmo que utilice del libro de referencia.
Su algoritmo debe admitir una implementación cuyo tiempo de ejecución sea $O(m+n)$, donde n es la cantidad de vértices de G y m es la cantidad de aristas de G.
Sugerencia: Utilice las ideas que se presentan en la demostración del resultado (3.19) del libro de referencia.

(b) Demuestre la corrección de su algoritmo.

(c) Demuestre que su algoritmo admite una implementación con tiempo de ejecución que es $O(m+n)$.

Ejercicio 12 (*Kleinberg & Tardos, Ex. 3.12*)**.** A partir de una serie de entrevistas en una investigación histórica se recopilaron una serie de datos sobre un conjunto de n personas, $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$, que ya no están vivas. Los datos recopilados se refieren a pares de personas, (P_i, P_j), en cierto subconjunto de $\mathcal{P} \times \mathcal{P}$, y cada dato puede ser de alguno de los tipos:

1. P_i murió antes de que P_j naciera,
2. las vidas de P_i y P_j se solaparon en el tiempo.

Queremos determinar si nuestros datos son consistentes, es decir, si puede haber existido un conjunto de n personas tales que todos los datos recopilados se satisfacen, respetando además el orden natural de nacimiento y muerte para cada persona. Para ello necesitamos un algoritmo cuya salida sea alguno de los siguientes resultados:

- Una cronología de nacimientos y muertes de las n personas que verifique la consistencia de los datos recopilados.
Por ejemplo, si para $n = 3$ los datos recopilados indican que P_1 murió antes de que P_3 naciera y que las vidas de P_2 y P_1 se solaparon, la salida de nuestro algoritmo podría ser:

  nace P_2, nace P_1, muere P_1, nace P_3, muere P_2, muere P_3.

- Un mensaje indicando que tal cronología no existe y por lo tanto nuestros datos son inconsistentes.

(a) Presente un algoritmo que resuelve el problema. Describa la representación con que se recibe la entrada (en un formato natural para el problema descrito) y formalice un modelo para resolver el problema. Su algoritmo debe admitir una implementación cuyo tiempo de ejecución sea $O(m + n)$, donde m es la cantidad de datos que fueron recopilados.

(b) Demuestre la corrección de su algoritmo.

(c) Demuestre que su algoritmo admite una implementación con tiempo de ejecución que es $O(m + n)$.