Práctica 4: Recorridos y Árboles

Compilado: 6 de mayo de 2024

Aquellos ejercicios marcados con ⋆ forman el conjunto **mínimo** que recomendamos resolver para cubrir todo el temario que será evaluado en los exámenes parciales.

Recorridos en profundidad

1. ⋆Sea T un árbol generador de un grafo (conexo) G con raíz r, y sean V y W los vértices que están a distancia par e impar de r, respectivamente.

   a) Observar que si existe una arista $vw \in E(G) \setminus E(T)$ tal que $v, w \in V$ o $v, w \in W$, entonces el único ciclo de $T \cup \{vw\}$ tiene longitud impar.

   b) Observar también que si toda arista de $E(G) \setminus E(T)$ une un vértice de V con otro de W, entonces (V, W) es una bipartición de G y, por lo tanto, G es bipartito.

   c) A partir de las observaciones anteriores, diseñar un algoritmo lineal para determinar si un grafo conexo G es bipartito. En caso afirmativo, el algoritmo debe retornar una bipartición de G. En caso negativo, el algoritmo debe retornar un ciclo impar de G. **Explicitar cómo es la implementación del algoritmo**; no es necesario incluir el código.

   d) Generalizar el algoritmo del inciso anterior a grafos no necesariamente conexos observando que un grafo G es bipartito si y solo si sus componentes conexas son bipartitas.

2. ⋆Una arista de un grafo G es *puente* si su remoción aumenta la cantidad de componentes conexas de G. Sea T un árbol DFS de un grafo conexo G.

   a) Demostrar que vw es un puente de G si y solo si vw no pertenece a ningún ciclo de G.

   b) Demostrar que si $vw \in E(G) \setminus E(T)$, entonces v es un ancestro de w en T o viceversa.

   c) Sea $vw \in E(G)$ una arista tal que el nivel de v en T es menor o igual al nivel de w en T. Demostrar que vw es puente si y solo si v es el padre de w en T y ninguna arista de $G \setminus \{vw\}$ une a un descendiente de w (o a w) con un ancestro de v (o con v).

   d) Dar un algoritmo lineal basado en DFS para encontrar todas las aristas puente de G. **Ayuda:** el algoritmo puede hacer un uso inteligente de un único DFS. Conceptualmente, y a los efectos de este ejercicio, puede convenir separar el algoritmo en dos fases. La primera fase aplica DFS para calcular el mínimo nivel que se puede alcanzar desde cada vértice usando back edges que estén en su subárbol. La segunda fase recorre todas las aristas (sin DFS) para chequear la condición.

3. Una orientación de un grafo G es un grafo orientado D cuyo grafo subyacente es G. (En otras palabras, D es una orientación de G cuando D se obtiene dando una orientación a cada arista de G). Para todo árbol DFS T de un grafo conexo G se define $D(T)$ como la orientación de G tal que $v \to w$ es una arista de $D(T)$ cuando: v es el padre de w en T o w es un ancestro no padre de v en T (Figura 1).

   a) Observar que $D(T)$ está bien definido por el Ejercicio 2*b*).

   b) Demostrar que las siguientes afirmaciones son equivalentes:

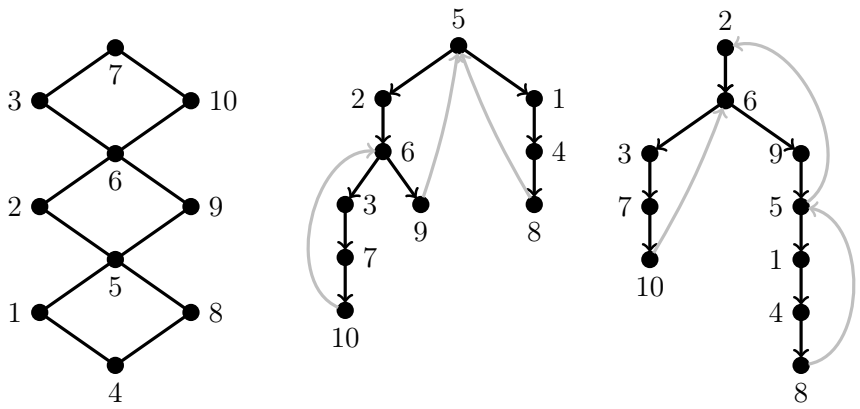

Figura 1. *En el centro y la derecha se ven dos árboles DFS T_1 y T_2 del grafo G de la izquierda marcados en negro, junto con las aristas grises que completan $D(T_1)$ y $D(T_2)$.*

I) G admite una orientación que es fuertemente conexa.

II) G no tiene puentes.

III) para todo árbol DFS de T ocurre que $D(T)$ es fuertemente conexo.

IV) existe un árbol DFS de T tal que $D(T)$ es fuertemente conexo.

Ayuda: para II) ⇒ III) observar que alcanza con mostrar que la raíz de $D(T)$ es alcanzable desde cualquier vértice v. Demuestre este hecho haciendo inducción en el nivel de v, aprovechando los resultados del Ejercicio 2.

c) Dar un algoritmo lineal para encontrar una orientación fuertemente conexa de un grafo G cuando dicha orientación exista.

4. La intendencia de una ciudad quiere orientar la mayor cantidad de calles posibles a fin de evitar accidentes. Actualmente, todas las calles son bidireccionales y unen un par de esquinas. Modelar el problema de decidir qué calles se deben orientar y en qué sentido a fin de minimizar la cantidad de calles bidireccionales que quedan. Proponer un algoritmo de tiempo $O(n+m)$ para resolver el problema.

Recorridos en anchura

5. ⋆Un árbol generador T de un grafo G es *v-geodésico* si la distancia entre v y w en T es igual a la distancia entre v y w en G para todo $w \in V(G)$. Demostrar que todo árbol BFS de G enraizado en v es v-geodésico. Dar un contraejemplo para la vuelta, i.e., mostrar un árbol generador v-geodésico de un grafo G que no pueda ser obtenido cuando BFS se ejecuta en G desde v.

6. Diseñar un algoritmo de tiempo $O(n+m)$ que, dado un grafo conexo G con pesos en sus aristas y un vértice v, determine el árbol de menor peso de entre todos los árboles v-geodésicos de G. **Justificar** que el algoritmo propuesto es correcto. **Ayuda:** pensar cuáles aristas pueden pertenecer a un árbol v-geodésico cualquiera, para elegir las que minimicen el peso total.

7. Queremos diseñar un algoritmo que, dado un digrafo G y dos vértices s y t, encuentre el recorrido de longitud par de s a t que use la menor cantidad de aristas.

a) Sea H el digrafo bipartito que tiene dos vértices v^0, v^1 por cada vértice $v \in V(G)$, donde v^0 es adyacente a w^1 en H si y solo si v y w son adyacentes en G. (Notar que $\{v^i \mid v \in V(G)\}$ es un conjunto independiente para $i \in \{0, 1\}$.) Demostrar que $v_1, \ldots, v_k$ es un recorrido de G si y sólo si $v_1^1, v_2^0, \ldots, v_k^{k \bmod 2}$ es un recorrido de H.

b) Sea $G^{=2}$ el digrafo que tiene los mismos vértices de G tal que v es ayacente a w en $G^{=2}$ si y solo si existe $z \in G$ tal que $v \to z \to w$ es un camino de G. Demostrar que G tiene un recorrido de longitud $2k$ si y solo si $G^{=2}$ tiene un recorrido de longitud k.

c) Diseñar dos algoritmos basados en las propiedades anteriores para resolver el problema de encontrar el recorrido de longitud par de s a t que use la menor cantidad de aristas.

d) Justifique cuál de los dos algoritmos es mejor, considerando: la complejidad temporal y espacial, la dificultad de la implementación y la posibilidad de modificar el algoritmo para encontrar recooridos de longitud impar.

8. ⋆Se tiene una grilla con $m \times n$ posiciones, cada una de las cuales tiene un número entero en $[0, k)$, para un $k \in \mathbb{N}$ dado. Dado un valor objetivo $w \in \mathbb{N}$ y una posición inicial (x_1, y_1), que tiene un valor inicial v_1, queremos determinar la mínima cantidad de movimientos horizontales y verticales que transformen v_1 en w, teniendo en cuenta que el i-ésimo movimiento transforma a v_i por $v_{i+1} = (v_i + z) \bmod k$, donde z es el valor que se encuentra en la casilla de destino del movimiento. Por ejemplo, para la siguiente grilla y $k = 10$, se puede transformar $v_1 = 1$ en $w = 0$ con tres movimientos $1 \to 6 \to 4 \to 9$, aunque la solución óptima es vía el camino $1 \to 3 \to 6$.

1	3	6
6	7	4
4	9	-3

Modelar este problema como un problema de grafos que se resuelva usando BFS en $O(kmn)$ tiempo.

9. Proponer un algoritmo de complejidad temporal $O(ds)$ que, dados dos naturales $d > 0$ y $s > 0$, o bien encuentre el mínimo n divisible por d y cuyos dígitos sumen exactamente s o bien reporte que dicho n no existe. **Ayuda:** recuerde que n se puede decomponer en dígitos $n_1 \ldots n_k$. Para cada $i = 1, \ldots, k$, el número con digitos $n_1 \ldots n_i$ tiene un resto d_i módulo d y una sumatoria s_i. Sabiendo que k debe ser mínimo, piense cómo cambian d_i y s_i al agregar el dígito n_{i+1}.

Árbol generador mínimo, camino minimax y maximin

10. ⋆Se define la distancia entre dos secuencias de naturales $X = x_1, \ldots, x_k$ e $Y = y_1, \ldots, y_k$ como $d(X, Y) = \sum_{i=1}^{k} |x_i - y_i|$. Dado un conjunto de secuencias $X_1, \ldots, X_n$, cada una de tamaño k, su grafo asociado G tiene un vértice v_i por cada $1 \leq i \leq n$ y una arista $v_i v_j$ de peso $d(X_i, X_j)$ para cada $1 \leq i < j \leq n$. Proponer un algoritmo de complejidad $O(kn^2)$ que dado un conjunto de secuencias encuentre el árbol generador mínimo de su grafo asociado.

11. ⋆Una empresa de comunicaciones modela su red usando un grafo G donde cada arista tiene una capacidad positiva que representa su *ancho de banda*. El *ancho de banda* de la red es el máximo k tal que G_k es conexo, donde G_k es el subgrafo generador de G que se obtiene de eliminar las aristas de peso menor a k (Figura 2).

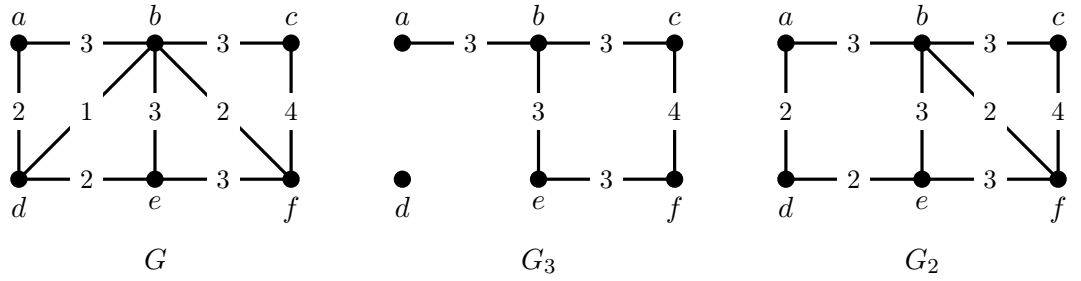

Figura 2. *El grafo G tiene ancho de banda 2 porque G_2 es conexo y G_3 no. Por otra parte, el ancho de banda del camino c, b, d es 1 mientras que el ancho de banda del camino c, b, e, d es 2. En general,* $\text{bwd}(c, d) = 2$ *mientras que* $\text{bwd}(a, e) = \text{bwd}(b, f) = 3$.

a) Proponer un algoritmo eficiente para determinar el ancho de banda de una red dada.

La empresa está dispuesta a hacer una inversión que consiste en actualizar algunos enlaces (aristas) a un ancho de banda que, para la tecnología existente, es virtualmente infinito. Antes de decidir la inversión, quieren determinar cuál es el ancho de banda que se podría obtener si se reemplazan i aristas para todo $0 \leq i < n$.

b) Proponer un algoritmo que dado G determine el vector $a_0, \ldots, a_{n-1}$ tal que a_i es el ancho de banda máximo que se puede obtener si se reemplazan i aristas de G.

12. Dado un grafo G con capacidades en sus aristas, el *ancho de banda* $\text{bwd}_G(C)$ de un camino C es la mínimo de entre las capacidad de las aristas del camino (Figura 2). El *ancho de banda* $\text{bwd}_G(v, w)$ entre dos vértices v y w es el máximo entre los anchos de banda de los caminos que unen a v y w (Figura 2). Un árbol generador T de G es *maximin* cuando $\text{bwd}_T(v, w) = \text{bwd}_G(v, w)$ para todo $v, w \in V(G)$. Demostrar que T es un árbol maximin de G si y solo si T es un árbol generador máximo de G. Concluir que todo grafo conexo G tiene un árbol maximin que puede ser computado con cualquier algoritmo para computar árboles generadores máximos. **Ayuda:** para la ida, tomar el AGM T' que tenga más aristas en común con T y suponer, para obtener una contradicción, que T' tiene una arista e' que no está en T. Luego, buscar una arista e en T que no este en T' tal que $(T' - e') + e$ sea también AGM para obtener la contradicción. Para la vuelta, tomar v y w en el AGM T' y considerar la arista xy de peso mínimo en el único camino de T' que los une. Luego, mostrar que xy tiene un peso al menos tan grande como cualquier otra arista que une las componentes conexas de $T' \setminus \{xy\}$ que contienen a v y w.

13. ⋆El algoritmo de Kruskal (resp. Prim) con orden de selección es una variante del algoritmo de Kruskal (resp. Prim) donde a cada arista e se le asigna una prioridad $q(e)$ además de su peso $p(e)$. Luego, si en alguna iteración del algoritmo de Kruskal (resp. Prim) hay más de una arista posible para ser agregada, entre esas opciones se elige alguna de mínima prioridad.

a) Demostrar que para todo árbol generador mínimo T de G, si las prioridades de asignación están definidas por la función

$$q_T(e) = \begin{cases} 0 & \text{si } e \in T \\ 1 & \text{si } e \notin T \end{cases}$$

entonces se obtiene T como resultado del algoritmo de Kruskal (resp. Prim) con orden de selección ejecutado sobre G (resp. cualquiera sea el vértice inicial en el caso de Prim).

b) Usando el inciso anterior, demostrar que si los pesos de G son todos distintos, entonces G tiene un único árbol generador mínimo.

14. Sea $q\colon V(G) \to \mathbb{Z}$ una función inyectiva para un grafo G. Demostrar que G tiene un único árbol generador mínimo si y solo si el algoritmo de Kruskal con prioridad q retorna el mismo árbol que el algoritmo de Kruskal con prioridad $-q$.

Ejercicios integradores

15. Se tiene una matriz cuadrada M de $n \times n$ con valores en $\{0, 1, 2\}$. Se quiere determinar una forma de conectar todas las posiciones de M con valor 1 a través de caminos que se muevan en sentido horizontal o vertical y no pasen por posiciones de M con valor 2. El objetivo es minimizar la longitud de todos los caminos en conjunto. Proponer un algoritmo con complejidad temporal $O(kn^2)$ para resolver este problema, donde k es la cantidad de posiciones de M con valor 1. Por ejemplo, en la siguiente matriz la longitud de los caminos en conjunto es 13.

0	1	1	1	0
0	2	2	0	0
0	0	1	2	1
0	0	0	0	0
1	0	1	0	0

16. Un grafo G es un *cactus* cuando cada una de sus aristas pertenece a un único ciclo.

a) Sea T un árbol DFS de un grafo G y sea $T(v, w)$ el unico camino entre v y w en T para todo $v, w \in V(G)$. Demostrar que G es un cactus si y solo si para toda arista $vw \in E(G) \setminus E(T)$ ocurre que $T(v, w) + vw$ es el único ciclo que contiene a las aristas en $T(v, w)$.

b) Demostrar que los grafos cactus tienen $O(n)$ aristas.

c) Diseñar un algoritmo de tiempo $O(n)$ para determinar si un grafo es un cactus. En caso afirmativo, el algoritmo debe retornar todos los ciclos del grafo. En caso negativo, el algortimo debe retornar dos ciclos que compartan una arista.

d) Diseñar un algoritmo de tiempo $O(n)$ para encontrar un árbol generador mínimo de un grafo cactus. **Justificar** que el algoritmo es correcto utilizando resultados conocidos.

e) Proponer una formula para contar la cantidad de árboles generadores mínimos de un grafo cactus que pueda ser computada en $O(n)$ operaciones de suma y multiplicación. **Demostrar** que la fórmula es correcta.

17. Dado un digrafo D, un ordenamiento $v_1, \ldots, v_n$ de $V(D)$ es un *orden topológico* de D cuando para toda arista $v_i \to v_j$ de D ocurre que $i < j$ (Figura 3). En la guía pasada vimos que D admite un orden topológico si y solo si D es acíclico. En este ejercicio vemos cómo determinar si D es acíclico y obtener un orden topológico usando DFS.

Sea v un vértice que alcanza todos los otros vértices de un digrafo D y sea T un árbol generador que se obtiene al ejecutar DFS desde v. Más aun, supongamos que los vértices hermanos de T

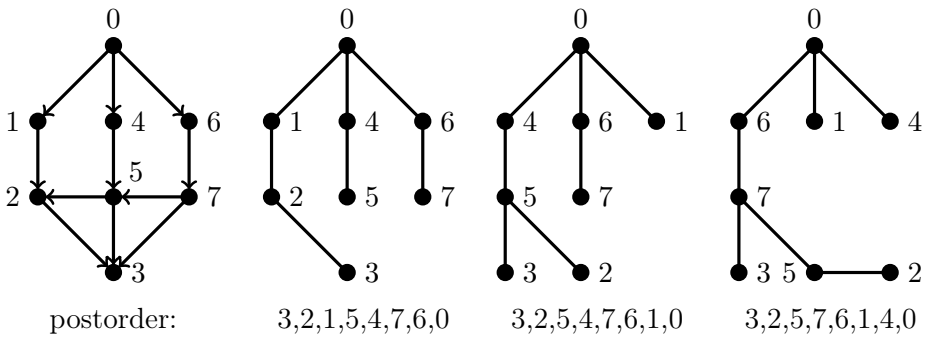

Figura 3. *Árboles DFS de un digrafo acíclico D y sus correspondientes post-órdenes, cada una de las cuales es el reverso de un orden topológico de D*

están ordenados de forma tal que u aparece antes que su hermano w cuando u fue descubierto antes que w por el algoritmo DFS (por lo tanto el vecindario de u fue procesado antes que el de w). Finalmente, sea S la secuencia que se obtiene al revisar T en sentido postorder (Figura 3; recordar que para todo árbol con raíz r y **secuencia** de subárboles $T_1, \ldots, T_k$ se tiene postorder(r) = postorder(T_1) + ... + postorder(T_k) + r).

a) Demostrar que D es acíclico si y solo si el reverso de S es un orden topológico de D.

b) Describir el algoritmo resultante para determinar si D es acíclico y obtener el orden topológico correspondiente.

c) Modificar el algortimo anterior para evitar la suposición de que existe un vértice que alcanza a todos los otros vértices.

18. Dado un grafo G queremos responder una serie de consultas de la siguiente forma: dados dos vértices v y w, determinar si existe un único camino entre v y w. Diseñar un algoritmo que dado un grafo G lo procese para generar una estructura de datos que permita responder cada consulta en $O(1)$ tiempo. El mejor algoritmo que conocemos toma tiempo $O(n + m)$.

19. Sea F un bosque generador de un grafo G pesado con una función $c\colon E(G) \to \mathbb{R}$. Decimos que una arista vw es *segura* si v y w pertenecen a distintos árboles de F. La arista vw es *candidata para el árbol T de F que contiene a v* cuando vw es segura y $c(vw) \leq c(xy)$ para toda arista segura xy tal que $x \in T$. Considere el siguiente *(meta-)*algoritmo que computa un árbol generador mínimo de (G, c):

```
1: Sea F = (V(G), ∅) un bosque generador de G.
2: Para i = 1, ..., n − 1:
3:     Agregar a F una arista candidata de algún árbol T.
```

a) Demostrar que el algoritmo anterior retorna un árbol generador mínimo T de G. **Ayuda:** hacer inducción en i y demostrar en cada paso que el bosque F_i es un subgrafo de un árbol generador mínimo de G.

b) Mostrar que tanto Prim como Kruskal son casos particulares de este algoritmo en las que la arista candidata se determina usando una política específica. Concluir que la demostración anterior prueba la corrección de Prim y Kruskal en forma conjunta.

Sin pérdida de generalidad, se puede suponer que todas las aristas de G tienen un peso diferente. En efecto, alcanza con extender $c(\cdot)$ a una función de pesos que $c'(\cdot)$ tal que $c'(v) = (c(v), v)$, donde v es el identificador del vértice (i.e., un numero en $[1, n] \cap \mathbb{N}$). Bajo la hipótesis de que todos los pesos son distintos, el algoritmo que consiste en insertar todas las aristas candidatas posibles a F en cada iteración computa un árbol generador mínimo por el inciso *a)*. Este algoritmo fue propuesto por Borůvka en 1926, mucho antes de que Prim y Krukal propusieran los suyos.

c) Describir una implementación simple del algoritmo de Borůvka que requiera $O(m \log n)$ tiempo cuando un grafo G con n vértices y m aristas es dado.