Métodos Algorítmicos en Resolución de Problemas I

Grado en Ingeniería Informática
Doble Grado en Ingeniería Informática y Matemáticas

Hoja de ejercicios 4 Curso 2024–2025

Ejercicios de grafos

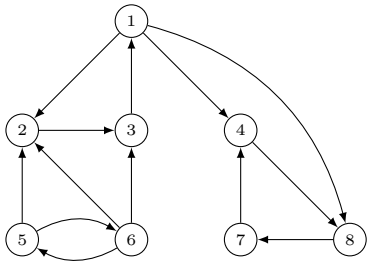
Ejercicio 1 Se llama *traspuesto* de un grafo (V, A) al grafo (V, A^T) donde $A^T = \{(v, u) \mid (u, v) \in A\}$. Diseñar algoritmos para calcular el traspuesto de un grafo y dar su coste en los casos en que el grafo venga representado por listas de adyacencia y por matriz de adyacencia.

Ejercicio 2 Se llama *cuadrado* de un grafo (V, A) al grafo (V, A^2) donde $A^2 = \{(u, w) \mid (u, v) \in A \wedge (v, w) \in A\}$. Diseñar algoritmos para calcular el cuadrado de un grafo, y dar su coste, en los casos en que el grafo venga representado por listas de adyacencia, y por matriz de adyacencia.

Ejercicio 3 En un curso con n estudiantes se conoce la relación $A = \{(x, y) \mid x \text{ ama a } y\}$, siendo $a =\mid A \mid$. Diseñar un algoritmo que etiquete a todos los estudiantes, bien como *amantes*, si solo aparecen en parte la izquierda de las tuplas de A, bien como *amados*, si solo aparecen en la derecha, o como *solitarios* si no aparecen, o bien que indique que tal asignación no es posible. Calcular su coste suponiendo que el grafo venga representado por listas de adyacencia y por matriz de adyacencia.

Ejercicio 4 Un vértice p de un grafo dirigido se llama *sumidero* si para todo otro vértice v existe la arista (v, p) pero no la arista (p, v). Escribir un algoritmo que detecte la presencia de un sumidero en un grafo G en tiempo $O(n)$, donde n es el número de vértices. El algoritmo debe aceptar un grafo representado por su matriz de adyacencia.

Ejercicio 5 Considera el siguiente grafo:



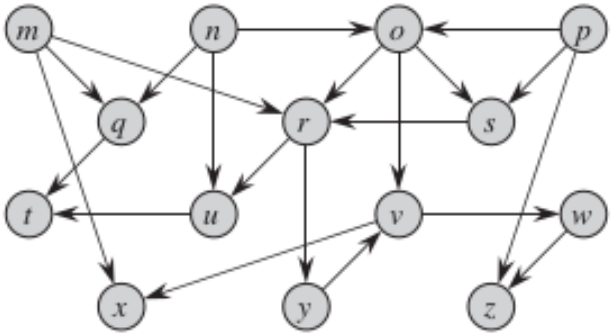
Suponiendo que los vértices adyacentes se visitan atendiendo a su orden numérico y que los vértices iniciales que sean necesarios también se eligen en función de su orden numérico,

1. muestra cómo progresa el recorrido en anchura y dibuja el árbol asociado al recorrido;
2. muestra cómo progresa el recorrido en profundidad, indicando los tiempos de descubrimiento y finalización de cada vértice, y dibuja el árbol asociado al recorrido. A partir de esta numeración, ¿es posible calcular una ordenación topológica de sus vértices?

Ejercicio 6 Escribir un algoritmo que se base en el recorrido en profundidad para determinar si un grafo no dirigido es acíclico.

Ejercicio 7 Se dice que un grafo no dirigido $G = (V, E)$ es k-coloreable si todos los vértices de G pueden ser coloreados usando k colores diferentes de manera que no haya dos vértices adyacentes que tengan el mismo color. Diseñar un algoritmo cuyo coste en tiempo esté en $O(|V| + |E|)$ que coloree un grafo con dos colores o determine que el grafo no es 2-coloreable.

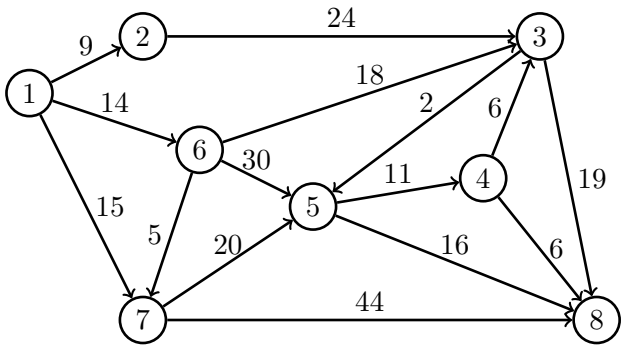
Ejercicio 8 Mostrar el resultado y los pasos intermedios del algoritmo de ordenación topológica para el grafo de la figura. Cada parte del algoritmo que necesite recorrer un conjunto de vértices lo hará siempre en orden alfabético.



Ejercicio 9 Dado un grafo dirigido y valorado $G = (V, A)$, la capacidad de flujo de un camino es el mínimo de los costes de sus aristas.

Dados dos vértices s y t, y un valor C, implementar un algoritmo que encuentre un camino desde s hasta t con una capacidad de flujo mayor o igual que C o que diga que tal camino no existe. La complejidad en tiempo de la solución propuesta debe estar en $O(|V| + |A|)$.

Ejercicio 10 Dado el siguiente grafo dirigido y valorado, ejecutar sobre él el algoritmo de Dijkstra a partir del vértice 1, e indicar en una tabla de tantas líneas como se necesiten: (a) la iteración en la que se incorpora cada vértice al conjunto de vértices seleccionados; (b) la distancia mínima calculada para cada vértice; (c) el camino mínimo calculado para el mismo; y (d) las distancias mínimas provisionales a los vértices que aún no han sido seleccionados.



Ejercicio 11 Modificar el algoritmo de Dijkstra para que además indique para cada vértice *cuántos* caminos mínimos hay a él desde el vértice origen. ¿Cambia el coste del algoritmo?

Ejercicio 12 Dado un grafo dirigido y valorado, junto con un vértice origen v_0, escribir un algoritmo que modifique lo menos posible el de Dijkstra de forma que devuelva los caminos de mínimo coste de v_0 a cualquier otro vértice, pero garantice además que, en caso de haber más de un camino de mínimo coste, devuelva el que tenga menor número de aristas.

Ejercicio 13 Durante la Segunda Guerra Mundial y con el fin de poder transportar pertrechos con comodidad, los japoneses construyeron puentes de dirección única que conectaban determinados pares de islas. De cada uno conocían su *limitación de peso*. Establecida su base de operaciones en la isla de Midway, se preguntaron cuál sería el *peso máximo* de los tanques que podrían trasladar desde Midway hasta cada isla destino, y cuál sería *la ruta de puentes* que deberían utilizar en cada caso.

1. Proporcionar un algoritmo que resuelva el problema de forma óptima.
2. La aviación americana destruyó el puente entre las islas A y B y los japoneses debían mandar una partida de peso P a la isla C. El Coronel, tras consultar brevemente la salida del algoritmo, tranquilizó a sus subordinados indicándoles que el ataque no afectaría a ese envío. ¿Cómo llegó el Coronel a esa conclusión?

3. Tras consultar también muy brevemente la citada salida, el General les indicó que en realidad la destrucción de ese puente no alteraría sus planes de suministros a *ninguna* de las islas del archipielago. ¿Cómo llegó el General a esa conclusión?

Ejercicio 14 En *Silvania* disponen de una red de caminos bidireccionales que conectan directa o indirectamente todas las ciudades que la conforman. Desgraciadamente, el ejercito de *Transilvania* es mucho más potente y podrá hacer caer una a una las ciudades de Silvania, destruyendo como consecuencia los caminos que unen en cada caso la ciudad conquistada con la parte aun no conquistada. Los habitantes de Silvania pueden disponer sus defensas de manera que la conquista tenga que realizarse en el orden que ellos decidan. Demuestra que podrán escoger un orden tal, que durante todo el proceso se mantengan totalmente conectadas (por caminos aún no destruídos) las ciudades que sigan formando parte de la Silvania *libre*. Diseña un algoritmo eficiente que fije el orden en que los silvanos irán dejando caer sus ciudades.

Ejercicio 15 Dado un grafo $G = (V, A)$ no dirigido, valorado y conexo, se tienen las siguientes versiones abstractas de los algoritmos de Kruskal y Prim, que devuelven un *árbol de recubrimiento mínimo*:

funcion Kruskal (V,A) **retorna** T
  $T = \{\}$; $n = |V|$; L = aristas_por_costes_crecientes(A);
  mientras $|T| < n - 1$ **hacer**
    a = L.min_coste(); L.borra_min();
    si no_hay_ciclos($T \cup \{a\}$) **entonces**
      $T = T \cup \{a\}$
retorna T

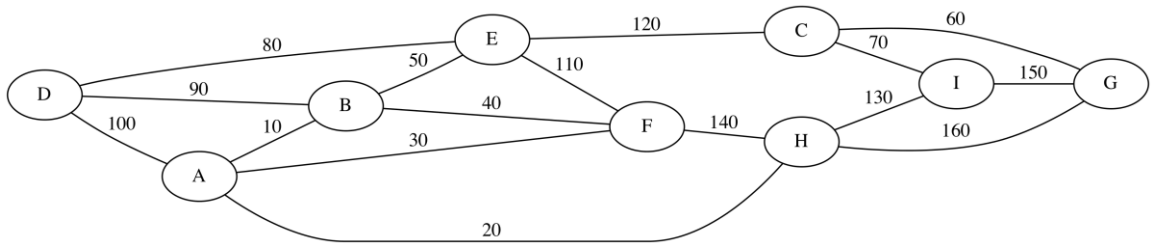
funcion Prim (V,A) **retorna** T
  $T = \{\}$; $n = |V|$; $V_T = \{1\}$; salen=$\{(1, i) \in A \mid 2 \leq i \leq n\}$;
  repetir $n - 1$ **veces**
    a = salen.min_coste(); j = vertice de a que no esta en V_T;
    salen = (salen$- \{(i, j) \in$ salen$\mid i \in V_T\}) \cup \{(j, k) \in A \mid k \notin V_T\}$
    $T = T \cup \{a\}$; $V_T = V_T \cup \{j\}$;
retorna T

Indicar cuál sería el comportamiento de cada uno de los algoritmos anteriores, en el caso de que el grafo suministrado no fuera conexo. Si necesitas precisar cómo está implementado el grafo, o cómo se implementan las funciones que aparecen en el seudocódigo, indica las hipótesis utilizadas. A continuación, indica las modificaciones necesarias en cada uno de ellos, para que los algoritmos devuelvan un resultado razonable, incluso cuando el grafo dado no sea conexo. ¿Cuál sería dicho resultado en cada caso?

Ejercicio 16 Decimos que un grafo dirigido es *redondo*, cuando contiene un ciclo en el que aparecen todos sus vértices, y ningún otro ciclo de longitud menor.

1. Dado un grafo dirigido y un vértice v cualquiera del mismo, aplicar una variante del algoritmo de Dijkstra, o de la exploración en anchura, para calcular la longitud mínima de un ciclo que contenga al vértice v. El algoritmo devolverá un valor imposible de no existir ninguno.
2. Dar un algoritmo que decida si un grafo dirigido es redondo y calcular su complejidad.

Ejercicio 17 Considera el siguiente grafo no dirigido y valorado:



1. Da el orden en el que las aristas se añaden al ARM (árbol de recubrimiento de coste mínimo) utilizando el algoritmo de Kruskal.
2. Da el orden en el que las aristas se añaden al ARM utilizando el algoritmo de Prim, considerando A como vértice inicial.
3. Supón que se añade la arista H—C al grafo. ¿Qué valoraciones de H—C harían que esta nueva arista perteneciera al ARM?

Ejercicio 18 Dado un grafo no dirigido, conexo, valorado y en el que no hay dos aristas que tengan el mismo coste, razonar con el mayor rigor posible que los algoritmos de Kruskal y Prim necesariamente calcularán el mismo árbol de recubrimiento de coste mínimo.

Ejercicio 19 Nos dan un grafo $G = \langle V, A\rangle$ no dirigido, valorado y conexo y un ARM (árbol de recubrimiento de coste mínimo) para él. También nos dan una arista nueva $(u, v) \notin A$, con un valor asociado c. Se pide un algoritmo de coste en $O(|V|)$ que modifique el ARM de manera que el resultado sea un ARM para $G' = \langle V, A \cup \{(u, v)\}\rangle$. Razona la corrección y el coste.

Ejercicio 20 Consideremos un grafo G no dirigido, conexo y valorado. ¿Cuáles de los siguientes métodos sirven para calcular un *árbol de recubrimiento de coste máximo* de tal grafo?

1. Aplicar una versión modificada del algoritmo de Kruskal en la que las aristas se eligen de mayor a menor coste.
2. Aplicar una versión modificada del algoritmo de Prim en la que la cola de prioridad usada es de máximos y en la que se cambian las comparaciones $\leq$ por comparaciones $\geq$.
3. Transformar el grafo G en $\overline{G}$ a base de invertir el signo de todos los valores y aplicar a continuación al algoritmo de Kruskal o de Prim estándar a $\overline{G}$ para calcular su árbol de recubrimiento de coste mínimo.

Ejercicio 21 En un videojuego se pueden recorrer diversos mundos en los que hay disponibles muchos tesoros. Los mundos están representados por un único grafo no dirigido G donde los N vértices son los escenarios del juego y las aristas los caminos que unen esos escenarios y que permiten a los jugadores ir de uno a otro. Un vector T guarda la información acerca de los tesoros: para cada escenario guarda o bien -1 si no hay tesoro o un valor positivo con el valor del tesoro situado en ese escenario.

En una sesión de juego cada jugador es situado en un escenario inicial, solo puede moverse siguiendo las aristas del grafo y puede coger a lo sumo un tesoro. Elena ha reunido un grupo de K jugadores y queda con ellos para jugar varias sesiones de juego. Al principio de cada sesión de juego conoce el escenario inicial de cada uno de sus jugadores, dado en un vector E, y desea saber el mayor tesoro que puede conseguir cada uno de ellos en esa sesión. Cada jugador juega sobre su propia copia del mundo, es decir, que los jugadores no se afectan entre sí.

Por ejemplo, supongamos que en el juego hay siete escenarios numerados de 1 a 7, con tesoros en los escenarios 1, 3 y 6 con valores 10, 20 y 30 respectivamente; y aristas $(1, 2)$, $(1, 3)$, $(4, 5)$ y $(5, 6)$. Y supongamos que Elena tiene 3 jugadores. En una sesión en la que los jugadores están situados inicialmente en los escenarios 1, 4 y 7, los mayores tesoros que pueden conseguir cada uno son respectivamente 20, 30 y 0. Sin embargo, en una sesión en la que los jugadores están situados inicialmente en los escenarios 1, 2 y 5, los mayores tesoros que pueden conseguir cada uno son respectivamente 20, 20 y 30.

Diseñar e implementar un algoritmo eficiente que permita a Elena conocer los mayores tesoros que pueden conseguir sus jugadores en cada sesión de juego.

Ejercicio 22 Petunia y Amapola viven en puntos distintos de su ciudad y tienen que desplazarse todos los días en coche desde su casa al trabajo. Como la gasolina es muy contaminante han decidido quedar en una intersección de la ciudad y compartir uno de sus coches a partir de ahí hasta llegar al trabajo, pues de dicha forma el gasto total de gasolina es menor. Supongamos que la ciudad viene representada por un grafo no dirigido conexo, donde las casas de Petunia y Amapola, el

trabajo y las intersecciones entre calles son vértices y cada arista que conecta intersecciones tiene asociado un coste que representa el gasto de gasolina para recorrer esa calle en coche. Por ejemplo si quedasen en la panadería y el menor coste de llegar desde casa de Petunia a la panadería fuese 5, desde la casa de Amapola hasta la panadería fuese 7 y desde la panadería hasta el trabajo fuese 4, el coste total de ambos viajes sería 16. Sin embargo, si realizasen por separado los viajes de menor gasto de gasolina con costes por ejemplo, 8 y 10 respectivamente, el gasto total 18 sería superior y contribuirían más a la contaminación de la ciudad. Ayuda a Petunia y Amapola a decidir en qué intersección de la ciudad han de quedar y cuánta gasolina gastarán para llegar al trabajo de forma que se minimice el consumo de gasolina realizado entre ambas. ¿Qué cambios tendrías que hacer en caso de que el grafo sea dirigido?