Problema 2: Weird Journey

Rodrigo García Gómez RODRIGO.GARCIA@ESTUDIANTES.MATCOM.UH.CU

Tutor(es):
Alfredo Somoza

Resumen

Se pide, dado un grafo, encontrar la cantidad de formas de caminar por los nodos, pasando dos veces por todas sus aristas menos dos, y una vez por las dos restanes. El primer enfoque es una "fuerza bruta" en la que se simulan todos los recorridos de $(2m)$-2 pasos (considerando un paso como avanzar de una ciudad adyacente a otra por una carretera) posibles a realizar, partiendo desde cualquiera de las ciudades. Para todos estos recorridos, se toman las carreteras restantes y se eliminan los caminos inválidos o repetidos. El enfoque óptimo del problema consiste en, para todo par de carreteras, fijarlas como las seleccionadas para pasar una sola vez. Todas las carreteras restantes son duplicadas. Para esa selección de carreteras de una sola pasada, existirá un camino válido si y sólo si existe una cadena de euler en el nuevo grafo computado, o lo que es igual, si y sólo si el número de vértices (ciudades) con grado impar es 0 o 2.

1. Texto del problema

Little boy Igor wants to become a traveller. At first, he decided to visit all the cities of his motherland — Uzhlyandia.

It is widely known that Uzhlyandia has n cities connected with m bidirectional roads. Also, there are no two roads in the country that connect the same pair of cities, but roads starting and ending in the same city can exist. Igor wants to plan his journey beforehand. Boy thinks a path is good if the path goes over $m - 2$ roads twice, and over the other 2 exactly once. The good path can start and finish in any city of Uzhlyandia.

Now he wants to know how many different good paths are in Uzhlyandia. Two paths are considered different if the sets of roads the paths goes over exactly once differ. Help Igor — calculate the number of good paths.

2. El problema

Dado un grafo G de n vértices y m aristas, se quiere encontrar la cantidad de caminos válidos posibles sobre el grafo, partiendo desde cualquier vértice. Un camino es válido si pasa exacamente dos veces sobre $m-2$ aristas, y exactamente una vez por las dos restantes. El orden en que se recorre un camino es irrelevante para diferenciarlo de otro, dos caminos válidos serán distintos si las aristas que se recorren una sola vez en cada uno son distintas. Se tiene como entrada un entero n que representa la cantidad de vértices, un entero m que representa la cantidad de aristas y una lista de tamaño m con dos enteros que representan la existencia de una arista entre el vértice representado por el primero y el representado por el segundo. Se espera la salida como un entero único, representando la cantidad de caminos válidos.

3. Algoritmo DFS

En cada uno de los siguientes algoritmos se realiza primero una ejecución de `dfs` para com-

probar si G es conexo. En caso de no serlo se retorna de forma instantánea el valor 0.

3.1 Descripción

El simple `dfs` implementado toma a un vértice (inicialmente el vértice 0) y lo marca como visitado. Luego, cada nodo conectado a este, si no estaba ya marcado como visitado, es visitado recursivamente por el algoritmo.

3.2 Demostración

Demostrando que al final de la ejecución de `dfs`, si algún nodo quedó como no visitado, entonces el grafo no es conexo (1) y que si el grafo no es conexo, entonces algún nodo quedará como no visitado (2).
(1) Suponiendo que existe un vértice v que está conectado por algún camino al vértice inicial u del `dfs` y no fue visitado por el algoritmo. Entonces existiría al menos una arista $< v_1, v_2 >$ (sin pérdida de generalidad se dice que v_1 es más cercana a v y v_2 es más cercana a u) que pertenece a uno de los caminos que conectan a v con u tal que v_1 fue visitado y v_2 nunca fue visitado (note que v_2 puede ser v). Pero esto no es posible, ya que cuando el algoritmo procesa v_1, al estar v_2 conectado a v_1 y no haber sido este visitado, entonces lo visitaría.
(2) Como el grafo no es conexo, existirá al menos un vértice v que no esté conectado por ningún camino al vértice inicial u. Desde ninguno de los vértices v_1 alcanzables por el `dfs` iniciado en u, existirá una arista que lo enlace con alguno de los vértices v_2 conectados por algún camino con v, pues de lo contrario existiría un camino que uniera a u y v, formado a partir de enlazar al camino hecho por el dfs hasta v_1, con la arista $< v_1, v_2 >$, con el camino de v_2 a v.

La complejidad temporal de `dfs` es $O(n+m)$
Demostración: El algoritmo visitará una vez a cada nodo, pues cada vez que un nodo es visitado, se marca su posición con un 1 en la lista visit y no se vuelve a visitar ningún nodo una vez marcado. Además visitará exactamente dos veces a cada arista, ya sea para avanzar hacia un nodo, o para comprobar si este ya estaba visitado. La arista $< v_1, v_2 >$ será visitada a partir de la búsqueda de vecinos de v_1 y de v_2, las cuales se realizan sólo una vez por cada uno.

4. Idea Fuerza Bruta

Consiste en una simulación recursiva de todos los caminos posibles de $(m * 2)$–2 pasos o menos. Primeramente se forma el grafo como una matriz de adyacencia, para acceder de forma instantánea a un valor entero que se le asociará a cada arista. Este valor iniciará siendo 2 y representará la cantidad restante de veces que es posible caminar por esa arista. Luego se crea una lista vacía llamada `rests` en la que posteriormente se almacenarán todos los caminos que el algoritmo considere como válidos. Note que un camino válido estará representado sólo por dos tuplas que representan una pareja de aristas, Estas son las dos aristas por las que se caminará una sola vez (para ser caminos válidos distintos sólo importan estas dos aristas restantes, el orden por el que se camina es irrelevante).
Finalmente se hace un llamado recursivo de la función `calculate_from_vertex` por cada nodo v de G. Esta función simulará todas las posibles formas de caminar por G a partir de v, pasando a lo sumo dos veces por cada arista y finalizando cuando el contador (que comienza en 0 y se incrementa en 1 con cada paso) llega a $(2 * m)$–2. Por cada nodo u adyacente a v, si la arista que los conecta tiene valor mayor que 0, se le sustrae 1 a la misma y se llama recursivamente a la función a partir de u, con el contador incrementado en 1. Luego de esto la arista $< v, u >$ se deja con el mismo valor que tenía anteriormente. En el caso base (contador = (2*m)–2) se verifica cuáles fueron las aristas que quedaron con valor 1 y se almacenan en una variable `rest`. Si el tamaño de `rest` es igual a 2, se agrega `rest` a la lista `rests` (`rest` puede tener tamaño 0 si en lugar de quedar dos aristas con valor 1, queda una sola con valor 2). Luego de ejecutada la función todas las veces necesarias, basta con eliminar de `rests` a los

caminos duplicados y retornar como solución su tamaño.
La correctitud de este algoritmo se sustenta en la revisión de todos los caminos de tamaño (2*m) – 2 existentes que no pasan más de dos veces por la misma arista, ya que se revisa la partida desde todos los nodos y con cada iteración, todos los posibles movimientos a partir de estos hasta que se caminó la distancia requerida; además se tiene cuidado de no caminar más de dos veces por cada arista, evitando avanzar por las que tienen valor 0.

La función recursiva se invocará n veces (una por cada nodo) y esta a su vez se invocará a si misma, a lo sumo m veces. Este proceso de recursión se repetirá a lo sumo (2*m) – 2 veces, y por cada una de estas ramificaciones, el caso base que pertenece a $O(n^2)$ podrá ser ejecutado. por principio de la multiplicación el resultado pertenece a $O(n * m^(2 * m) * n^2)$ o lo que es lo mismo, $O(n^3 * m^(2 * m))$.

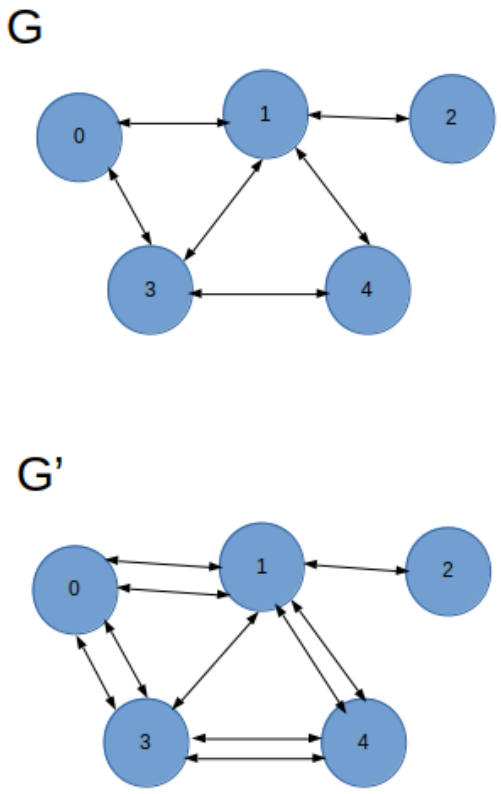

Figura 1: Ejemplo de construcción de G' a partir de G. Se seleccionaron como aristas de una sola pasada $< 1, 2 >$ y $< 1, 3 >$.

5. Primera solución encontrada

A partir de la solución anterior, fue evidente que la mejor forma de enfocar el problema no es simulando recorridos por la ciudad, sino, decidiendo, para cada par de caminos de la ciudad, si existe al menos un camino válido que los tenga a ellos dos como los caminos de una sola pasada (ya que todos los caminos que cumplan con esto contarán como un único incremento de la solución). Para lograr esto, se tomará cada pareja de aristas a_1 y a_2, y se computará un nuevo grafo (pseudografo) G' a partir de G, en el que se duplicarán todas las aristas, exceptuando a a_1 y a_2. En la figura 1 se ilustra un ejemplo de construcción de G'

Teorema 1 En G existirá un buen camino que pase sólo una vez por a_1 y a_2, si y solo si existe en G' una cadena de Euler

Demostración Demostrando que:
Existe Cadena de Euler en G' $\Rightarrow$ Existe buen camino en G que pasa solo una vez por a_1 y a_2.
Dado que existe una Cadena de Euler en G, entonces hay al menos un camino c' que pasa exactamente una vez por cada arista. Formaremos un camino c en G. Partimos por la primera arista que forma a c' y pasamos por cada una de estas en el mismo orden en que se presentan. Por cada arista que une a clauquier par de vértices u' y v' de c', añadiremos a c la arista (única) que une a u y v en G, siendo u',v' y u ,v los vértices de G' y G que representan a las mismas ciudades respectivamente (Estos existen porque G' fue armado a partir de G sin remover o añadir vértices, sólo añadiendo aristas). Para cada arista a del recién formado c hay dos opciones. Si a es a_1 o a_2, entonces aparece una única vez en c, ya que en G' sólo había una arista que unía a los vertices que representaban a las ciudades unidas por a_1 o a_2 y c' pasa por todas las aristas de G' exactamente una vez.

Si a no es a_1 ni a_2, entonces aparece en c exactamente dos veces, pues en G' hay exactamente dos aristas que unen a los vertices que representan a las ciudades unidas por a y c' pasa una vez por cada una de ellas. Por tanto c será un camino válido de G que pasa solo una vez por a_1 y a_2.

Demostrando que:
Existe buen camino en G que pasa solo una vez por a_1 y $a_2 \Rightarrow$ Existe Cadena de Euler en G'
El razonamiento es el mismo de la demostración anterior, pero esta vez se armará el camino c' a partir de c. Cada vez que c pasa por una arista, se agrega a c' una de las aristas de c que unen a las ciudades representadas por dicha arista que no se haya agregado previamente. Las aristas a_1 y a_2 están una sola vez en c y por tanto sus homólogas a_1' y a_2' estarán una sola vez en c'. Sea una arista a_0 diferente de a_1 y a_2, estará una sola vez en c, y por tanto, sus dos homólogas a_{01} y a_{02} serán agregadas ambas a c'. Por tanto c' conformará una cadena de Euler.

Demostración:

Demostrado el teorema anterior, solo resta analizar para cada grafo computado a partir de las parejas de aristas seleccionadas si existe una cadena de Euler.

Teorema 2 (demostrado en conferencia): Sea G un multigrafo, existe una cadena de Euler si y solo si la cantidad de vértices con grado impar es 0 o 2.

Teorema 3: Sea G un pseudografo, existe una cadena de Euler si y solo si la cantidad de vértices con grado impar es 0 o 2.

Demostración Demostrando que:
Existe Cadena de Euler =¿ La cantidad de vértices de grado impar es 0 o 2.
Se forma un nuevo grafo G' quitándole a G todos los lazos que tenga. Como G es un pseudografo, G' será un multigrafo. Al camino que forma la cadena de Euler existente en G se le quita las aristas que pasan por los lazos y se forma un camino c a partir de estas (el camino no se rompe por quitare lazos. Si antes el camino era $... < v_0, v_1 >, < v_1, v_1 >, < v_1, v_2 > \ldots$ el nuevo camino será $...(v0, v1), (v1, v2)\ldots)$. Dicho camino c pasará por todas las aristas de G' y por tanto en G' existirá cadena de Euler, lo que implica que su cantidad de vértices de grado impar es 0 o 2. Ahora, al volver a colocarle los lazos a G' para formar a G, la pariedad del grado de los vértices afectados no cambia pues a cada vértice se le suma 2 a su grado (si antes era par, par más 2 es par. Si antes era impar, impar más 2 es impar) y por tanto seguirán existiendo 0 o 2 vértices de grado impar.

Demostrando que:
La cantidad de vértices de grado impar es 0 o 2. $\Rightarrow$ Existe Cadena de Euler
Se forma un nuevo grafo G' quitándole a G todos los lazos que tenga. Como G es un pseudografo, G' será un multigrafo y como quitar lazos no afecta la pariedad de los vértices, la cantidad de vértices de grado impar sigue siendo 0 o 2. Por tanto en G' existe al menos una cadena de Euler, esta será c. Ahora se le agrega nuevamente los lazos a G' obteniendo G. Si se toma c y, a la primera aparición de cada vértice v al que se le puso un lazo, se le agrega la arista $< v, v >$ inmediatamente después (tengo $...< v_1, v >, < v, v_2 > \ldots$, se convierte en $... < v_1, v >, < v, v >, < v, v_2 > \ldots$) entonces el resultado será un camino que pasa exactamente una vez por cada arista del grafo G, lo que implica que existe una Cadena de Euler.

Por tanto, para comprobar si existen cadenas de Euler en los grafos computados basta con contar la cantidad de vértices con grado impar. Sin embargo, debido a las características de este grafo, todos los vértices que no forman parte de las dos aristas seleccionadas tendrán grado par. Entonces para resolver

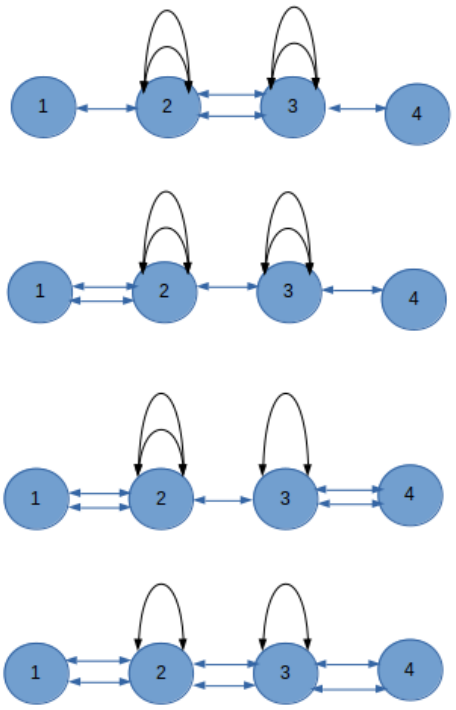

Figura 2: Ejemplos de casos de los cuatro vértices que forman parte de la pareja de aristas seleccionada. Los vértices de cada ejemplo, de arriba hacia abajo son: (1, 2, 3, 4), (2, 3, 3, 4), (2, 3, 3, 3) y (2, 2, 3, 3)

el problema basta con contar la pariedad de estos cuatro vértices seleccionados y esto se divide en los siguientes casos:

1. Los cuatro vértices son diferentes, entonces los cuatro tendrán grado impar y no habrá cadena de Euler.

2. Exactamente dos de estos vértices son los mismos, entonces dicho vértice tendrá grado par y los dos restantes grado impar. Existe cadena de Euler.

3. Tres de estos vértices son el mismo, entonces tanto este vértice como el restante tendrán grado impar. Existe cadena de Euler

4. Dos vértices son los mismos y los dos restantes son los mismos entre ellos, pero diferentes a los primeros, entonces no hay vértices de grado impar.

(Note que los cuatro vértices no pueden ser el mismo, pues las dos aristas seleccionadas serían la misma)

En la figura 2 se ilustran ejemplos de los 4 casos.

Para comprobar estos casos, se selecciona cada par de aristas existentes, se toman sus cuatro vértices y se crea un `set` con ellos. Si el tamaño del set es 2 o 3, se suma 1 al resultado pues existirá una Cadena de Euler, en caso contrario se suma 0.

En cuanto a la complejidad temporal se tiene 4 ciclos for lineales, una ejecución de dfs con costo $(N + M)$ y un doble ciclo de costo (N^2), por tanto el costo pertenece a $O(N^2)$

6. Algoritmo óptimo

Es posible optimizar la solución anterior agregándole un poco de combinatoria. Los casos a separar el problema también se pueden interpretar como:

1. Dos aristas no lazos, no adyacentes. Cuatro vértices impares, no existe Cadena de Euler.

2. Exactamente una de las aristas es un lazo. Dos vértices impares, existe Cadena de Euler.

3. Dos aristas no lazos, adyacentes. Dos vértices de grado impar, existe cadena de Euler.

4. Ambas aristas son lazos. Ningún vértice con grado impar. Existe Cadena de Euler.

Para calcular la solución no será necesario computar explícitamente todos los grafos (uno para cada pareja de vértices), lo que se hará será contar de cuantas maneras es posible darse estos casos si G' fuera computado a partir de cada pareja de aristas a_1 y a_2.

Por tanto basta con calcular el número de parejas de aristas no lazos adyacentes y sumarle la cantidad de lazos, multiplicada por m-1

(todas las aristas restantes). De esta forma se de cuantas maneras se puede escoger a a_1 y a_2 de manera que se cumpla 2, 3, o 4.
Para calcular las parejas de aristas adyacentes se suma la cantidad de parejas en las aristas entrantes de cada vértice, o lo que es igual, la cantidad de combinaciones de tamaño 2 de sus grados (los grados de los vértices sin contar a los lazos se guardan en la lista `cnt`). A esto se le suma la cantidad de lazos, multiplicados por m-1.
Ahora, como se incluyó dos veces los pares de dos aristas lazos, es necesario restarle la cantidad de combinaciones de tamaño dos de la cantidad de lazos.
Para hayar el número de combinaciones es necesario calcular los factoriales de los grados. Para no repetir esta operación cada vez que se quiere hallar una combinación, se halla primero todos los factoriales hasta el mayor número necesario y se almacena en una lista para acceder a ellos de forma instantánea.
La complejidad temporal de este algoritmo es menor que la anterior. Se eliminó el ciclo de n^2 dejando al algoritmo con un costo $O(N + M)$ provocado por la ejecución de el algoritmo dfs.