

Algoritmo de Hierholzer

El algoritmo de Hierholzer encuentra un ciclo (dirigido) en un multigrafo (dirigido) euleriano. Por lo tanto, desde el principio supondremos que el grafo es euleriano, saltándonos la comprobación sobre el grado de los vértices.

Notación: Usaremos el símbolo \leftarrow para asignar valores a variables: $x \leftarrow y$ significará que asignamos el valor y a la variable x . Las funciones booleanas tomarán los valores *true* o *false*.

Algoritmo:

Sea G un grafo euleriano dirigido, con vértices $\{1, \dots, n\}$; y sea s un vértice de G . El algoritmo construye un ciclo euleriano con origen en s .

1. Estructuras de datos necesarias:

- Listas de incidencia A_1, A_2, \dots, A_n ; para cada arista e denotamos el vértice final por $\text{end}(e)$.
- Listas K y C para guardar las secuencias de aristas que forman un ciclo. Usaremos listas doblemente relacionadas, es decir, cada elemento en la lista está relacionado con su predecesor y su sucesor, de modo que estos se puedan encontrar fácilmente.
- Una aplicación booleana, que denotamos 'used', sobre el conjunto de vértices: $\text{used}(v)$ toma valor *true* si v estaba en K y valor *false* en otro caso; y una lista L conteniendo todos los vértices v para los cuales $\text{used}(v) = \text{true}$.
- Un puntero $e(v)$ que apunta, para cada vértices v , a una arista e con origen en v ($e(v)$ no está definido al inicio del algoritmo).
- Una aplicación booleana, que denotaremos 'new', en el conjunto de aristas: $\text{new}(e)$ toma el valor *true* si e no está todavía contenido en el ciclo cerrado.
- Variables u, v para los vértices y e para las aristas.

2. Procedimiento: TRACE($v, \text{new}; C$)

El siguiente procedimiento construye un ciclo C formado por aristas no usadas todavía, con origen en un vértice dado v .

- Si $A_v = \emptyset$, entonces **return**.
- (Sabemos que $A_v \neq \emptyset$) Buscar la primera arista e en A_v y borrar e de A_v .
- Si $\text{new}(e) = \text{false}$, ir a (1).
- (Sabemos que $\text{new}(e) = \text{true}$) Añadir e a C .
- Si $e(v)$ no está definido, asignar a $e(v)$ la posición en que e aparece en C .
- Asignar $\text{new}(e) \leftarrow \text{false}$ y $v \leftarrow \text{end}(e)$.
- Si $\text{used}(v) = \text{false}$, añadir v a la lista L y asignar $\text{used}(v) \leftarrow \text{true}$.
- Ir a (1).

Aquí **return** significa que el procedimiento ha acabado, por lo tanto se produce un 'ir hasta el final del procedimiento'.



3. Procedimiento EULER($G, s; K$).

- (1) $K \leftarrow \emptyset$, $used(v) \leftarrow false$ para todo vértice v , $new(e) \leftarrow true$ para toda arista e .
- (2) $Used(s) \leftarrow true$, añadir s a L .
- (3) TRACE($s, new; K$);
- (4) Si L es vacío, **return**.
- (5) Sea u el último elemento de L . Borrar u de L .
- (6) $C \leftarrow \emptyset$
- (7) TRACE($u, new; C$)
- (8) Insertar C enlazado a $e(u)$ en K .
- (9) Ir a (4).

En el paso (3), se ha construido un ciclo maximal K con origen en s , y todos los vértices que aparecen en K se han almacenado en L . En los pasos (5) a (8), comenzando en el último vértice u de L , se construye un recorrido C que consiste en todas las aristas que no se habían usado todavía (aquellas con $new(e) = true$), y se inserta este camino en K . Por supuesto, C puede ser vacío. Como G es conexo (ya que es euleriano) el algoritmo termina sólo si para cada vértice v de G se tiene $used(v) = true$, y no hay más ciclos posibles. Si G es un grafo dirigido, el algoritmo funciona sin la función 'new', en este caso, tenemos que borrar cada arista de la lista de incidencia después de que se ha usado.