Grafos

- Recorridos DFS y BFS
  - Componentes conexas
  - Puntos de articulación
  - Puentes
- Camino de Euler
- Camino Hamiltoniano

Recorrido DFS

```
DFS(v) {
    marca v "visitado";
    para cada vértice w adyacente a v      //Recorro la lista de adyacentes
    si w está marcado "no visitado"
        DFS(w);

}
```

Recorrido BFS

```
BFS(v) {
    queue cola;
    cola.push(v) ;
    mientras( !cola.empty() ) {
     t = cola.front (); cola.pop();
     para cada w adyacente a t {
          si (w está marcado "no visitado") {
              cola.push(w) ;
              marcar w "visitado";
          }
     }
}
```

Componentes conexas

```
CC() {
    numCC = 0;
    setear todos los vértices como “no vistados”;
    para cada vértice v {
        si v está marcado ”no visitado” {
            imprimir ++numCC;
            dfs(v)
        }
    }
}
```

Punto de articulación

Sea G=(V,E) un grafo conectado no dirigido.

Un nodo a se dice **punto de articulación** de G, si existen nodos v y w tales que v, w y a son diferentes y todo camino entre v y w contiene a a.

Al quitar a el grafo se divide en dos o más partes.

Biconectividad

El grafo G es **biconectado** si para toda tupla de nodos v, w y a existe un camino de v a w que no contenga a a.

De otra forma

En un grafo no dirigido conexo pueden existir vértices que si se eliminan:

- "desconectan" el grafo
- lo dividen en componentes conexas

- Estos vértices clave son **puntos de articulación**
- Si un grafo no tiene punto de articulación es **biconexo**

Conectividad de un grafo

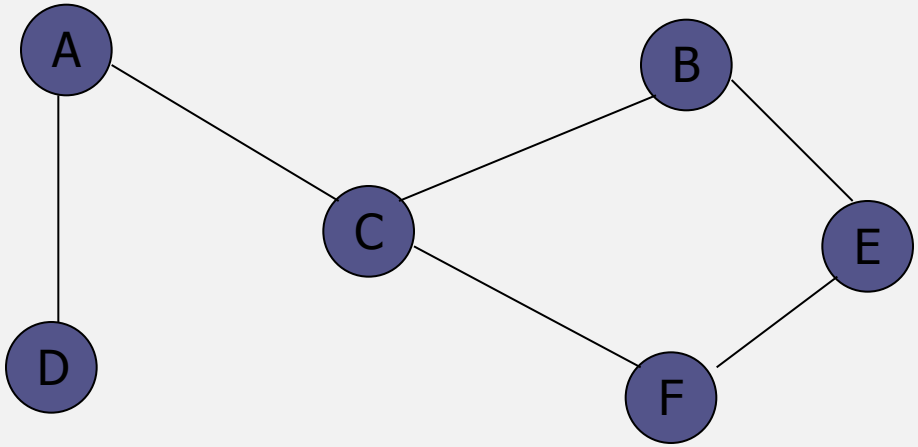
Es el número de nodos que se necesitan eliminar para dejar un grafo "desconectado"

Conectividad de un grafo

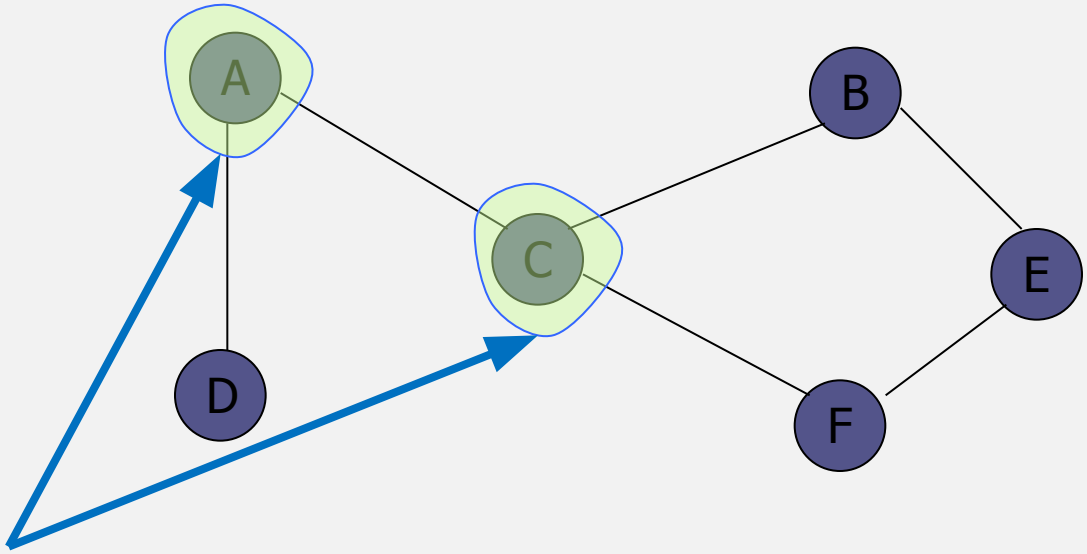
- Si el grafo representa un sistema de comunicaciones:

Que dicho grafo sea biconexo nos asegura el correcto funcionamiento de la red aunque falle el equipo de uno de los nodos.

Ejemplo



Ejemplo



Puntos de Articulacion

¿Cómo hallar los puntos de articulación?

¿Cómo hallar los puntos de articulación?

Una forma simple es remover cada vértice de a uno y chequear si se desconecta el grafo.

- Para cada vértice ***v*** hacer:
  - Remover ***v*** del grafo
  - Chequear si el grafo permanece conectado (usando BFS o DFS)
  - Agregar ***v*** al grafo

¿Cómo hallar los puntos de articulación?

Una forma simple es remover cada vértice de a uno y chequear si se desconecta el grafo.

- Para cada vértice ***v*** hacer:
  - Remover ***v*** del grafo
  - Chequear si el grafo permanece conectado (usando BFS o DFS)
  - Agregar ***v*** al grafo

El orden de ejecución es $O(|V|*(|V|+|E|))$

¿Cómo hallar los puntos de articulación?

- Se pueden determinar los puntos de articulación mediante un **solo recorrido DFS** del grafo

- Consideraciones a tener en cuenta:

¿Cómo hallar los puntos de articulación?

- Lema

Sea G=(V,E) un grafo no dirigido conexo y sea S=(V,T) un árbol de expansión en profundidad para G.

El vértice a es un punto de articulación de G si y sólo si:

1. a es la raíz y tiene más de un hijo o
2. a no es raíz y existe al menos un hijo s de a tal que no hay una arista posterior entre algún descendiente de s (incluyendo s) y un ancestro propio de a

Demostración del punto 1.

El vértice a es un punto de articulación de G si y sólo si:

1. a es la raíz y tiene más de un hijo

Demostración:

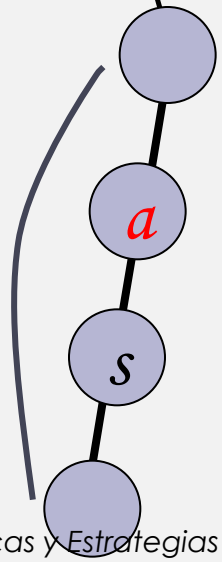
El *si* es claro

Para el *sólo si* sean a_1 y a_2 dos descendientes de a. Si a no fuera un punto de articulación existiría un camino entre a_1 y a_2 en el momento de visitar a_1, con lo que el segundo sería descendiente del primero

Demostración del punto 2.

El vértice a es un punto de articulación de G si y sólo si:

2. a no es raíz y existe al menos un hijo s de a tal que no hay una arista posterior entre algún descendiente de s (incluyendo s) y un ancestro propio de a



No se da esta situación

Demostración del punto 2.

Demostración:

Está claro que si a es punto de articulación se debe cumplir que para algún hijo no exista la arista mencionada.

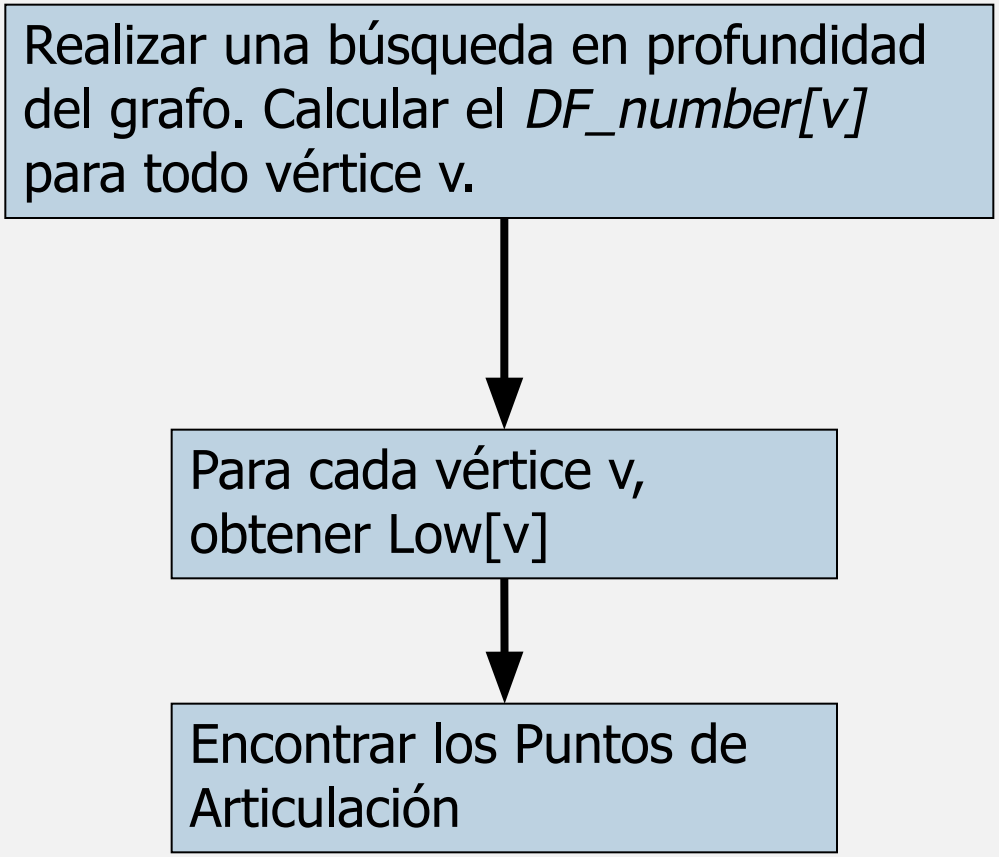
Para el *sólo si* suponemos que no se da la situación anterior, entonces si removemos a del grafo no existe ningún camino posible de s a ningún ancestro de a.

Detección del punto de articulación

De acuerdo a esta última propiedad, para saber si un nodo a es un punto de articulación es suficiente con verificar **si se puede acceder a un ancestro propio desde un descendiente de** a por un arco fuera del árbol.

Teniendo en cuenta la numeración generada por un recorrido DFS, se puede calcular para cada nodo el **ancestro más viejo (Low)** que se alcanza a través de arcos hacia atrás.

Pasos


Realizar una búsqueda en profundidad del grafo. Calcular el *DF_number[v]* para todo vértice v.
Para cada vértice v, obtener Low[v]
Encontrar los Puntos de Articulación

Pasos

Realizar una búsqueda en profundidad del grafo. Calcular el *DF_number[v]* para todo vértice v.

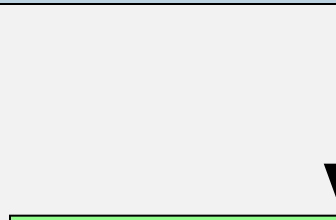
Para cada vértice v, obtener Low[v]

Encontrar los Puntos de Articulación

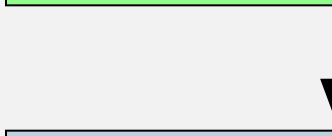
DF_number[v] es el número asignado al vértice al recorrer el grafo en la búsqueda en profundidad. Ordena los vértices en preorden en el árbol abarcador en profundidad.

Pasos

Realizar una búsqueda en profundidad del grafo. Calcular el *DF_number[v]* para todo vértice v.



Para cada vértice v, obtener Low[v]



Encontrar los Puntos de Articulación

Low[v] Se calcula Low[v] para todos los vértices v, visitándolos en un recorrido de orden posterior. Se toma Low[v] como el mínimo de:

- DF_number[v]
- DF_number[z] para cualquier vértice z para el cual haya una arista de retroceso (v,z) y
- Low[y] para cualquier hijo y de v.

Pasos

Realizar una búsqueda en profundidad del grafo. Calcular el *DF_number[v]* para todo vértice v.

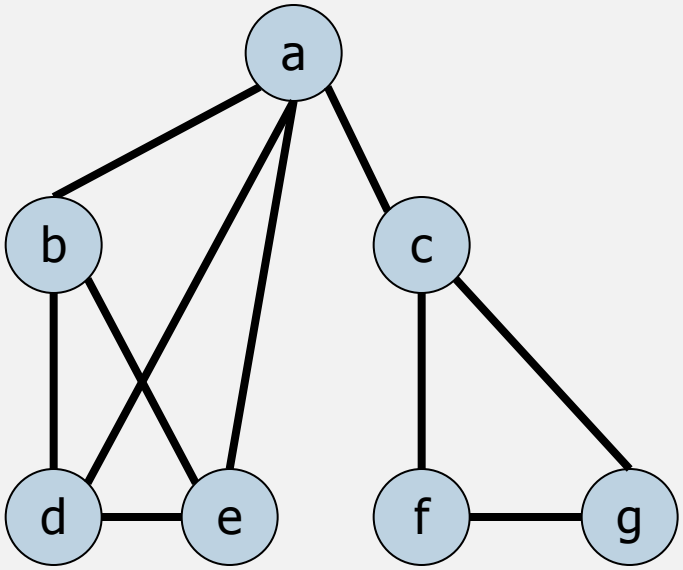
Para cada vértice v, obtener Low[v]

Encontrar los Puntos de Articulación

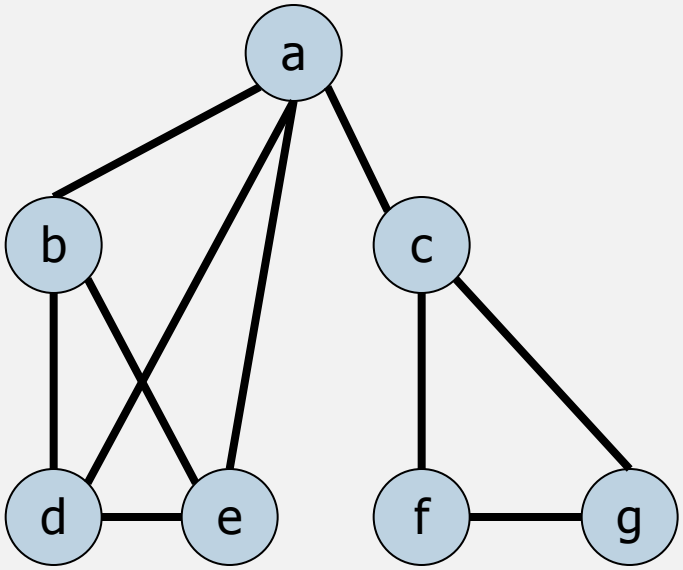
a) La raíz es un punto de articulación si, y sólo si, tiene dos o más hijos.

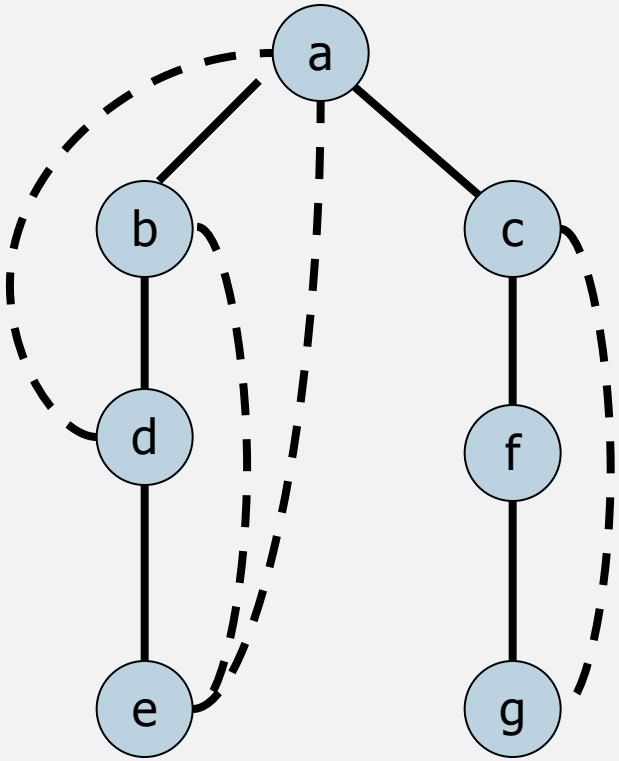
b) Un vértice v distinto de la raíz es un punto de articulación si y sólo si, hay un hijo w de v tal que Low[w] ≥DF_number[v]

Ejemplo

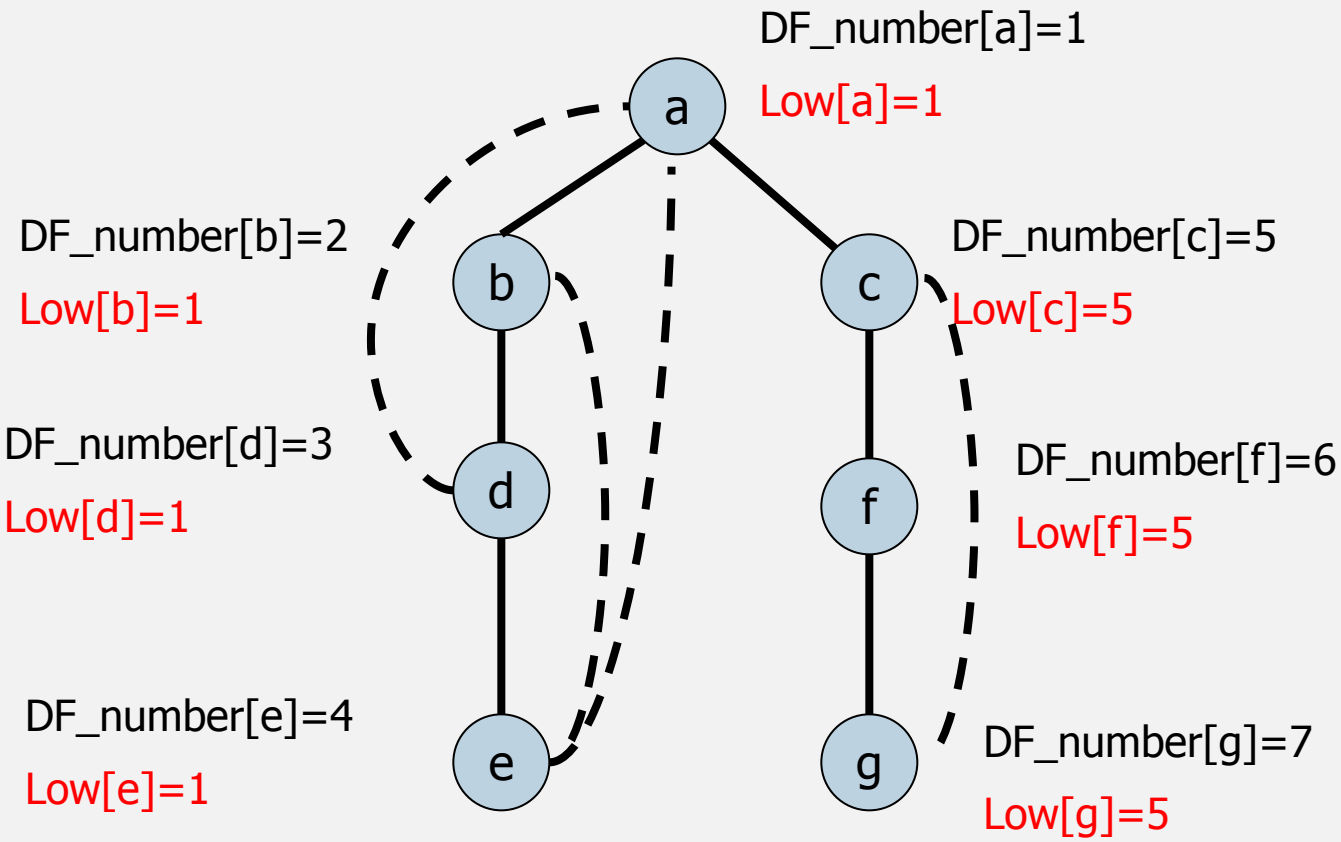


Ejemplo





Ejemplo


DF_number[a]=1
Low[a]=1
a
DF_number[b]=2
Low[b]=1
b
DF_number[c]=5
Low[c]=5
c
DF_number[d]=3
Low[d]=1
d
DF_number[f]=6
Low[f]=5
f
DF_number[e]=4
Low[e]=1
e
g
DF_number[g]=7
Low[g]=5

Código para encontrar los puntos de articulación

```
BUSCAR_PA(v) {
    marca v "visitado";
    DFNUMBER[v] ← COUNT;
    COUNT ← COUNT + 1;
    LOW[v] ← DFNUMBER[v];
    for cada vértice w en L[v] do
        if w está marcado "no visitado"{
            agrega (v,w) to T;
            FATHER[w] ← v;
            BUSCAR_PA(w);
            if LOW[w] ≥ DFNUMBER[v]
                // se encontró un punto de articulación v ;
            LOW[v] ← MIN(LOW[v], LOW[w])
        }
        else if w no es FATHER[v] then
            LOW[v] ← MIN(LOW[v], DFNUMBER[w])
}
```

Código para encontrar los puntos de articulación

```
PUNTOS_ARTICULACION {
    desmarcar todos los vértices;
    COUNT ← 1;
    for cada vértice v do
        if v está marcado "no visitado"{
            dfsRoot ← v; hijosRoot ← 0;
            BUSCAR_PA(v);
            if hijosRoot > 1
                dfsRoot es un punto de articulación;
        }
}
```

Código para encontrar los puntos de articulación

```
BUSCAR_PA(v) {
    marca v "visitado";
    DFNUMBER[v] ← COUNT;
    COUNT ← COUNT + 1;
    LOW[v] ← DFNUMBER[v];
    for cada vértice w en L[v] do
        if w está marcado "no visitado"{
            agrega (v,w) to T;
            FATHER[w] ← v;
            if v =dfsRoot
                hijosRoot ++;
            BUSCAR_PA(w);
            if LOW[w] ≥ DFNUMBER[v]
                // se encontró un punto de articulación v ;
            LOW[v] ← MIN(LOW[v], LOW[w])
        }
        else if w no es FATHER[v] then
            LOW[v] ← MIN(LOW[v], DFNUMBER[w])
}
```

Puentes

Sea G=(V,E) un grafo conectado no dirigido.

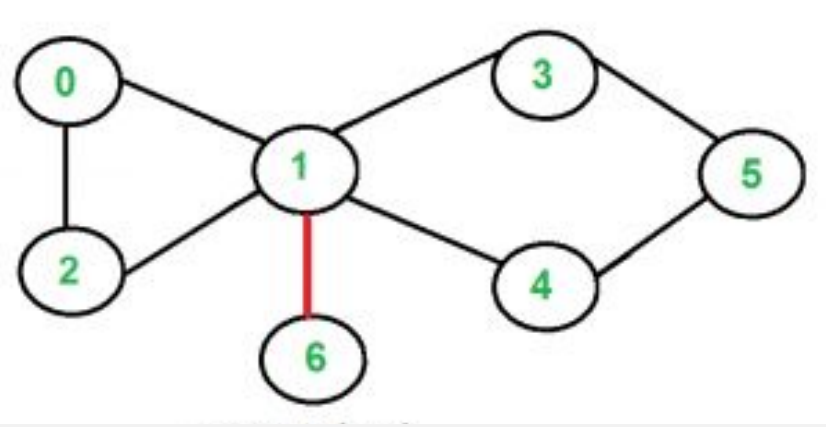
Una arista es un **puente** si y sólo si al remover dicha arista se desconecta el grafo.

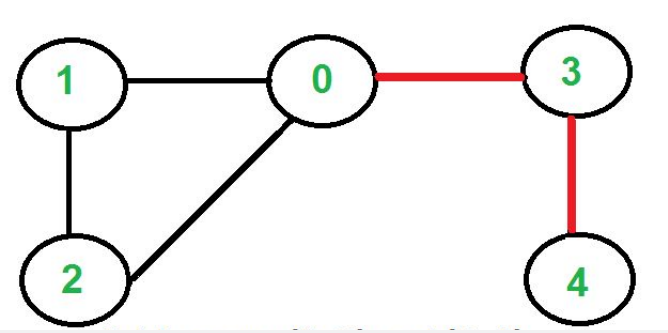
Si el grafo es no conexo, un puente es una arista que al removerla incrementa el número de componentes conexas.

Puentes

Al igual que los puntos de articulación, los puentes representan vulnerabilidades en la red.

Por ejemplo, si el grafo modeliza una red de computadoras, un punto de articulación representa una computadora crítica, mientras que un puente representa una conexión crítica.





¿Cómo hallar los puentes?

¿Cómo hallar los puentes?

Una forma simple es remover cada arista y chequear si se desconecta el grafo.

- Para cada arista (*u*,*v*) hacer:
  - Remover (*u*,*v*) del grafo
  - Chequear si el grafo permanece conectado (usando BFS o DFS)
  - Agregar (*u*,*v*) al grafo

¿Cómo hallar los puentes?

Una forma simple es remover cada arista y chequear si se desconecta el grafo.

- Para cada arista (*u*,*v*) hacer:
  - Remover (*u*,*v*) del grafo
  - Chequear si el grafo permanece conectado (usando BFS o DFS)
  - Agregar (*u*,*v*) al grafo

El orden de ejecución es $O(|E|*(|V|+|E|))$

¿Cómo hallar los puentes?

La idea es similar al algoritmo para hallar los puntos de articulación basado en el DFS

- Se realiza el recorrido DFS
- En el árbol DFS, una arista (***u***,***v***) es un puente si no existe ninguna otra alternativa de alcanzar a ***u*** o un ancestro de ***u*** desde el subárbol con raíz ***v***.
- El valor ***Low[v]*** representa el vértice visitado más temprano que es alcanzable desde el subárbol de ***v***.
- La condición para que una arista (u,v) sea puente es ***Low[v] > DF_number[u]***

Código para encontrar los puentes

```
SEARCHB(v) {
    marca v "visitado";
    DFNUMBER[v] ← COUNT;
    COUNT ← COUNT + 1;
    LOW[v] ← DFNUMBER[v];
    for cada vértice w en L[v] do
        if w está marcado "no visitado"{
            agrega (v,w) to T;
            FATHER[w] ← v;
            SEARCHB(w);
            if LOW[w] ≥ DFNUMBER[v]
                // se encontró un punto de articulación v ;
             if LOW[w] > DFNUMBER[v]
                                // se encontró un puente (v,w);
            LOW[v] ← MIN(LOW[v], LOW[w])
        }
        else if w no es FATHER[v] then
            LOW[v] ← MIN(LOW[v], DFNUMBER[w])
}
```

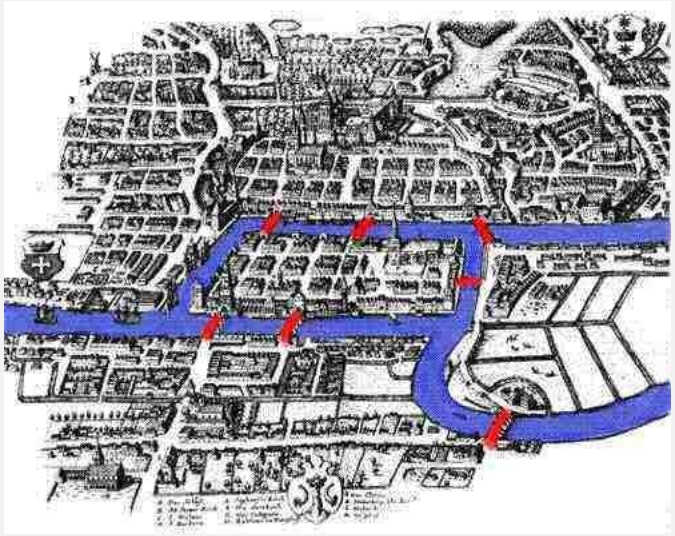
Recorridos Eulerianos y Hamiltonianos

Problemas de optimización de recorridos:

- Recorridos que pasan por todas las aristas del grafo una sola vez: caminos y circuitos eurelianos
- Recorridos que contienen todos los vértices del grafo una sola vez: caminos y ciclos hamiltonianos

Recorridos Eulerianos

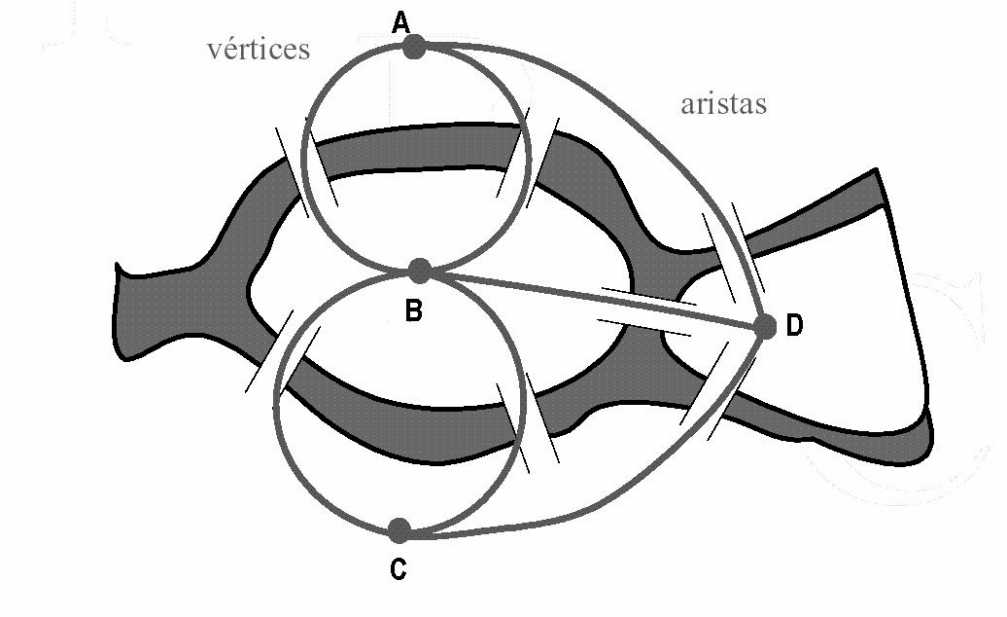
- Ciudad de Könisberg, en XVIII:

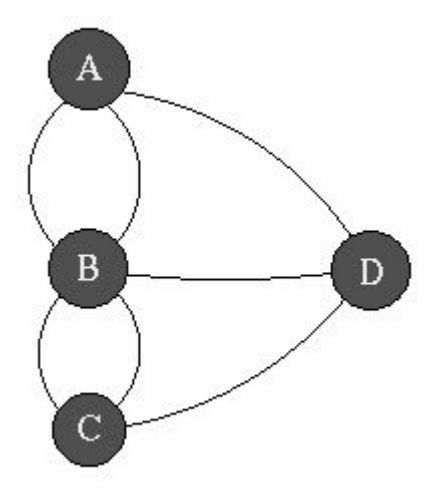


- Pregunta: ¿sería posible dar un paseo pasando por cada uno de los siete puentes, sin repetir ninguno, comenzando y terminando en el mismo punto?

Recorridos Eulerianos

- Representación propuesta por Leonard Euler en 1736:





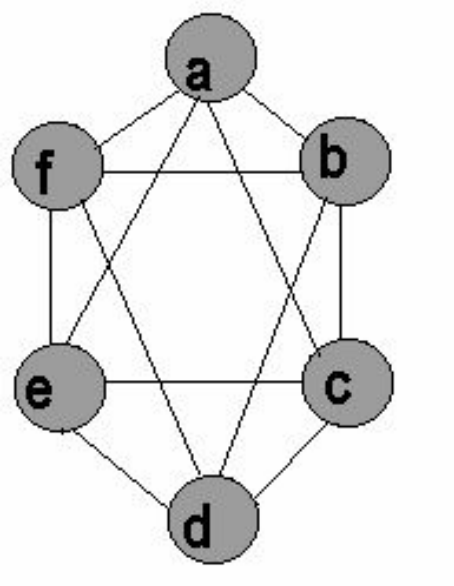
- ¿Existe un circuito que pase por todas las aristas una sola vez?

Recorridos Eulerianos

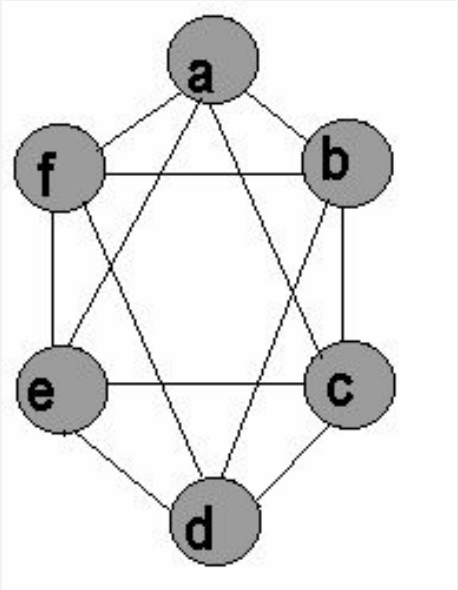
- A estos circuitos se les llama **circuitos eulerianos**, y a los grafos que los contienen **grafos eulerianos**.

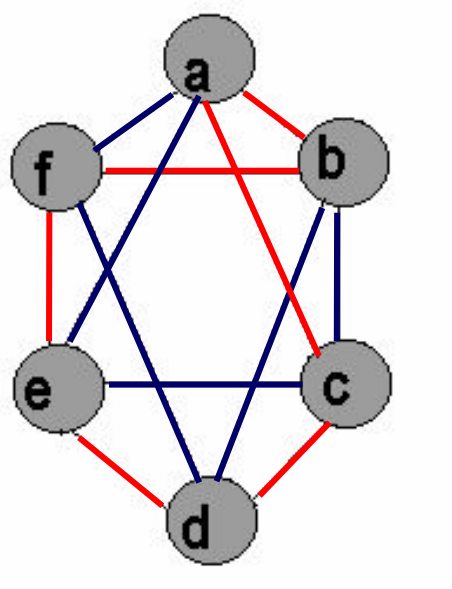
- **Grafo euleriano**: admite un recorrido que pasa por todas las aristas una sola vez, empezando y terminando en el mismo vértice. Los vértices sí se pueden repetir

Recorridos Eulerianos Ejemplos



Recorridos Eulerianos Ejemplos

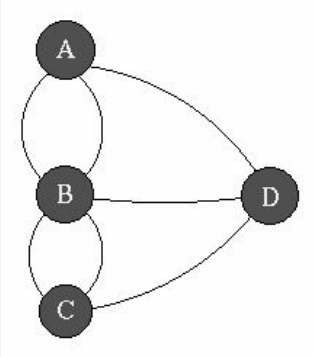




Circuito euleariano: **a,b,c,d,b,f,d,e,a,c,e,f,a**

Recorridos Eulerianos

- ¿Cómo saber si un grafo es euleriano?
- **Teorema de Euler**: Un grafo conexo es euleriano ↔no tiene vértices de grado impar
- Ejemplo:

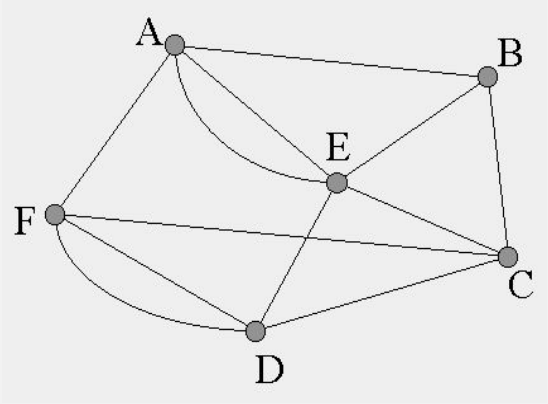


A tiene grado 3 → el grafo de los puentes no es euleriano.

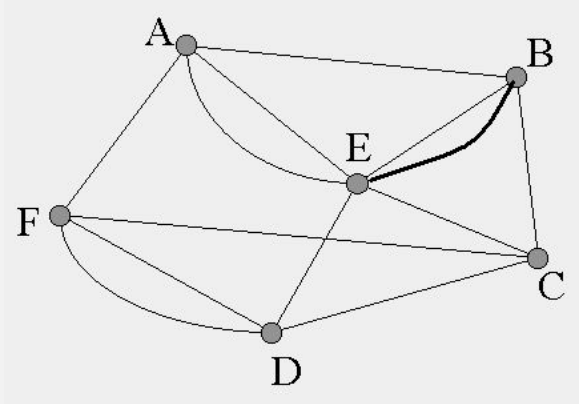
Recorridos Eulerianos

- Si el grafo tiene sólo dos vértices de grado impar tiene un **camino euleriano**.

Se puede convertir en euleriano agregándole una arista:



Semi-euleriano

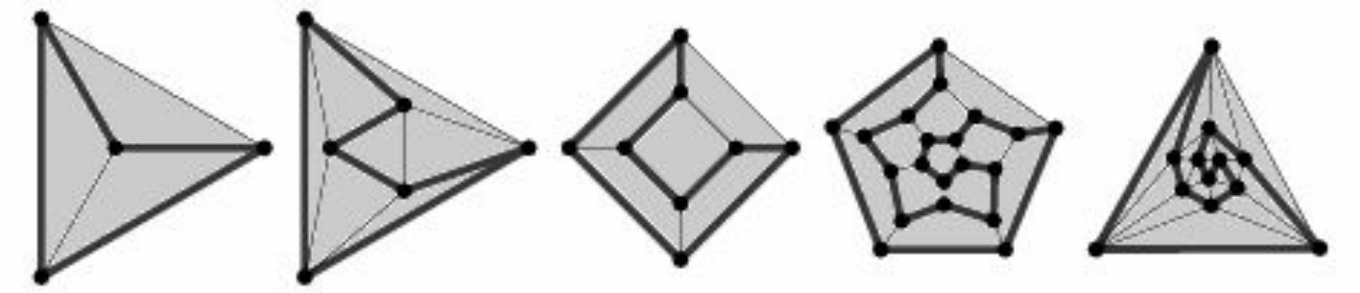


Euleriano

Recorridos Hamiltonianos

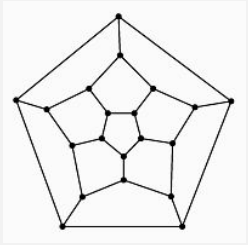
- Un grafo se dice **hamiltoniano** si existe un ciclo que recorre todos sus vértices. Al ciclo se le llama **ciclo hamiltoniano**

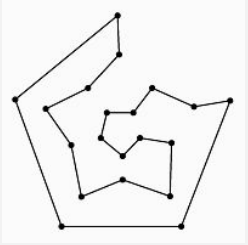
- Ejemplos:



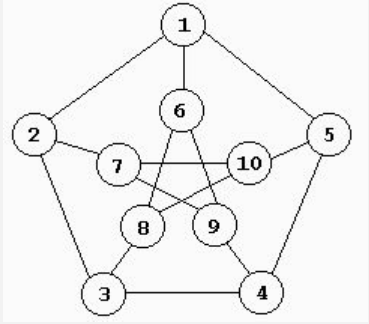
Recorridos Hamiltonianos

- No existe un método sencillo para saber si un grafo es no hamiltoniano → problema muy complejo (es uno de los mayores problemas sin resolver de la teoría de grafos)
- Ejemplo: Este grafo es hamiltoniano





- ...pero este no ¡es difícil de probar!



Recorridos Hamiltonianos

- Existen varias condiciones suficientes para establecer que un grafo es un grafo hamiltoniano:

1. Si un grafo G es conexo con:

   $|V| \geq 3$ y si el grado(v) $\geq |V|/2$ para cada vértice v de G, entonces G es hamiltoniano (Teorema de Dirac)

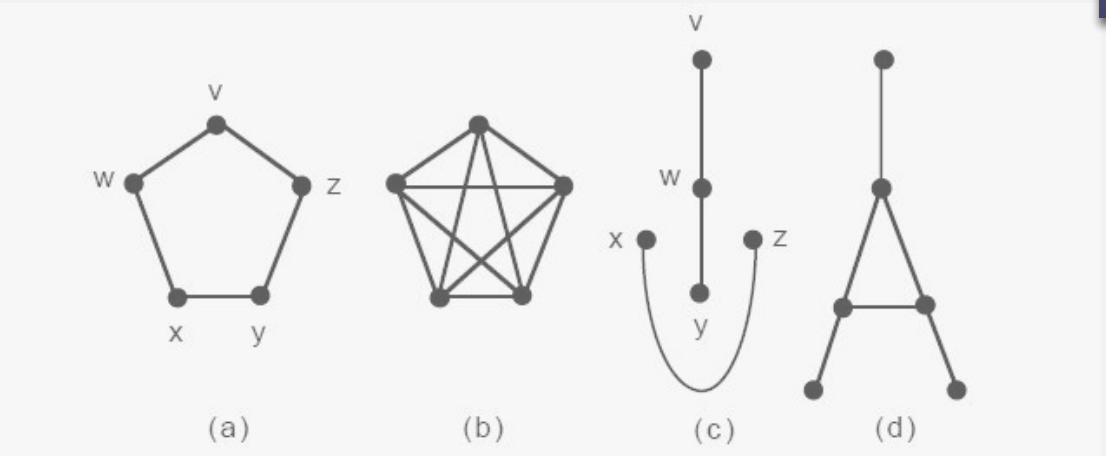
2. Un grafo que tiene al menos:

   $(1/2)(|V|-1)(|V|-2) + 2$ aristas es hamiltoniano

3. Si un grafo G es conexo con:

   $|V| \geq 3$ y si el grado(v) + grado (w) $\geq |V|$ para cada par de vértices v y w que no están conectados por una arista, entonces G es hamiltoniano (Teorema de Ore)

Recorridos Hamiltonianos:
Ejemplos



(a) Tiene un circuito hamiltoniano: v-w-x-y-z

(b) Se agregan más aristas, sigue siendo hamiltoniano. Todo grafo completo K$|V|$ con $|V| \geq 3$ es hamiltoniano.

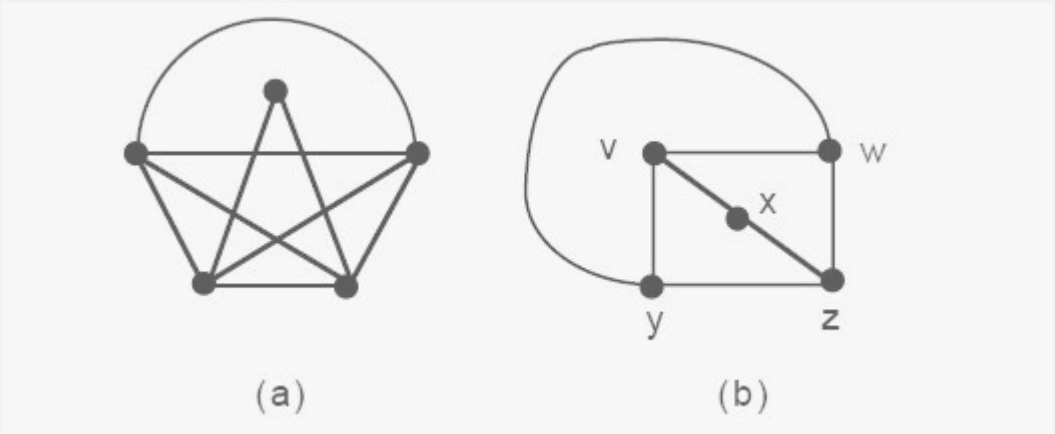
El grafo K5 tiene grado(v)=4 para cada v y $|V|=5$, por lo que satisface la condición 1.-

(c) Tiene un camino hamiltoniano, pero no un circuito

(d) No tiene camino hamiltoniano

Un grafo hamiltoniano debe tener al menos $|V|$ aristas. Ésta es una condición necesaria pero no suficiente, como se observa en (d)

Recorridos Hamiltonianos:
Ejemplos



(a) El grafo tiene |V|=5, lo que da (1/2) (|V|-1) (|V|-2) + 2 = 8. Satisface la condición 2

(b) El grafo tiene |V|=5, lo que da (1/2) (|V|-1) (|V|-2) + 2 = 8, pero tiene sólo 7 aristas. No satisface la condición 2. Si no tuviese el vértice en el medio, cumpliría la condición 2.
Este grafo cumple la condición 3: Hay tres pares de vértices distintos que no están conectados por ninguna arista. Verificamos que se cumpla:

Para <v,z> grado(v)+grado(z)= 3 + 3 = 6≥5
Para <w,x> grado(w)+grado(x)= 3 + 2 = 5≥5
Para <x,y> grado(x)+grado(y)= 2 + 3 = 5≥5

Fin de la primera parte