Algoritmos y estructuras de datos

Conexidad en gráficas y digráficas

Francisco Javier Zaragoza Martínez

Universidad Autónoma Metropolitana Unidad Azcapotzalco
Departamento de Sistemas

Universidad Autónoma Metropolitana
Casa abierta al tiempo Azcapotzalco

Posgrado en Optimización

4 de junio de 2021

Citas

Steve Jobs

No puedes conectar los puntos mirando hacia adelante. Sólo puedes conectarlos mirando hacia atrás. Así que debes confiar en que los puntos se conectarán de alguna manera en tu futuro.

Tim Berners-Lee

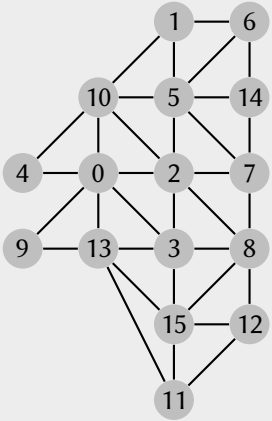
En una visión extrema, se puede ver al mundo sólo como conexiones, nada más... El cerebro no tiene saber hasta que se forman conexiones entre las neuronas. Todo lo que sabemos, todo lo que somos, viene de la forma en que nuestras neuronas se conectan.

Mark Zuckerberg

La conectividad no puede ser un privilegio de la gente en los países más ricos. Creemos que conectar a todos en el mundo es uno de los grandes retos de nuestra generación.

Gráficas

Caminos y conexidad



Camino Secuencia de vértices en la que cada dos consecutivos están unidos por una arista.

Camino simple Camino en el que no se repiten vértices.

Gráfica conexa Existen caminos entre cada dos vértices.

Componente conexa Si una gráfica no es conexa, entonces tiene varias componentes.

Circuito Camino cerrado (inicio = fin).

Ciclo Circuito en el que no se repiten vértices.

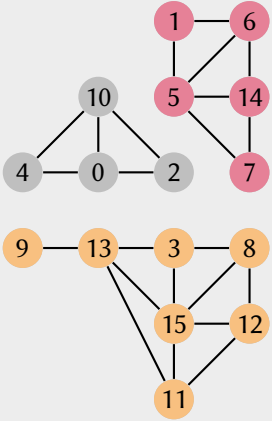
Bosque Gráfica que no tiene ciclos.

Árbol Bosque conexo.

Árbol abarcador Abarca todos los vértices.

Gráficas

Componentes conexas



Una gráfica $G = (V, E)$ que no es conexa consta de varias componentes conexas. Equivalentemente, V se puede partir en k subconjuntos $V_1, V_2, \ldots, V_k$ tales que cada subgráfica G_i inducida por V_i es conexa y no se puede agregar ningún vértice a V_i para que lo siga siendo.

Tanto la búsqueda en profundidad como la búsqueda en amplitud se pueden usar para encontrar las componentes conexas. Se necesita que la visita enumere el vértice que se acaba de visitar y que la visita no recursiva separe las listas de vértices. Esto se puede hacer con un arreglo inverso a `visto[u] = orden`, es decir `inv[orden-1] = u` (el negativo en la llamada no recursiva).

Gráficas

Implementación de componentes conexas

```
int componentesConexas(grafica g, int inv[]) {
  int orden = 0, k = 0;
  int *visto = (int *) calloc(g.n, sizeof(int));
  for (int u = 0; u < g.n; u++)
    if (visto[u] == 0) {
      k++;
      conexa(g, u, visto, inv, &orden);
      inv[visto[u]-1] = -inv[visto[u]-1];
    }
  free(visto);
  return k;
}
```

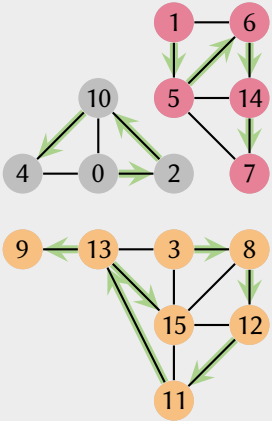
Gráficas

Implementación recursiva de una componente conexa

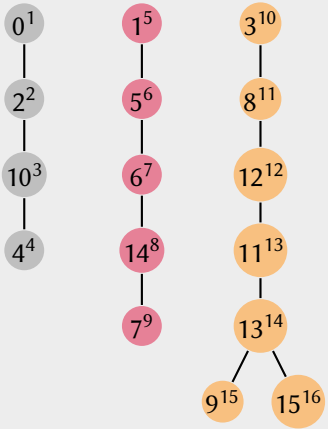
```
void conexa(grafica g, int u, int visto[], int inv[], int *orden)
{
  inv[*orden] = u;
  visto[u] = ++(*orden);
  for (int v = 0; v < g.n; v++)
    if (g.a[u][v] == 1 && visto[v] == 0)
      conexa(g, v, visto, inv, orden);
}
```

Gráficas

Ejemplo de componentes conexas

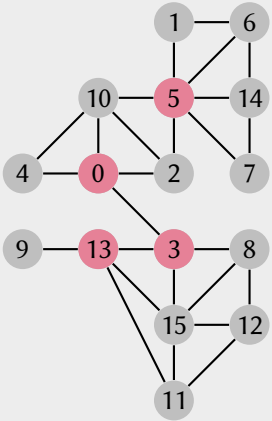


u	visto[u]	inv[u]
0	1	-0
1	5	2
2	2	10
3	10	4
4	4	-1
5	6	5
6	7	6
7	9	14
8	11	7
9	15	-3
10	3	8
11	13	12
12	12	11
13	14	13
14	8	9
15	16	15



Gráficas

Componentes biconexas

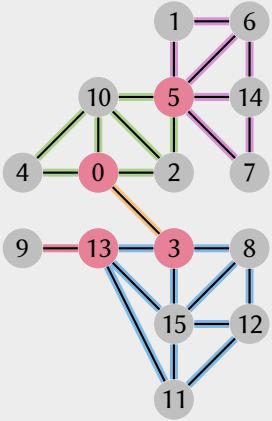


Una gráfica $G = (V, E)$ puede tener puntos de articulación, es decir, vértices que al borrarlos aumentan la cantidad de componentes conexas. A estos vértices también se les llama cuellos de botella o puntos únicos de fallo.

Una gráfica conexa que no tiene puntos de articulación se llama biconexa. Equivalentemente, en una gráfica biconexa cada pareja de vértices distintos está conectado por al menos dos caminos internamente disjuntos.

Gráficas

Componentes biconexas



Una gráfica conexa que no es biconexa consta de varias componentes biconexas. Equivalentemente, E se puede dividir en k subconjuntos $E_1, E_2, \ldots, E_k$ tales que cada subgráfica G_i inducida por E_i es biconexa y no se puede agregar ninguna arista a E_i para que lo siga siendo.

La búsqueda en profundidad recursiva se puede usar para encontrar las componentes biconexas de una gráfica.

1. Un vértice u no es de articulación si cada uno de sus sucesores v tiene un descendiente conectado a un punto más alto que u.
2. La raíz r es de articulación si tiene más de un sucesor.

La búsqueda devolverá el vértice más alto encontrado.

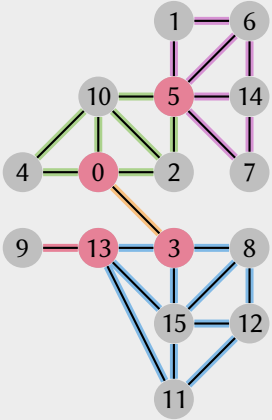
Gráficas

Implementación recursiva de componentes biconexas

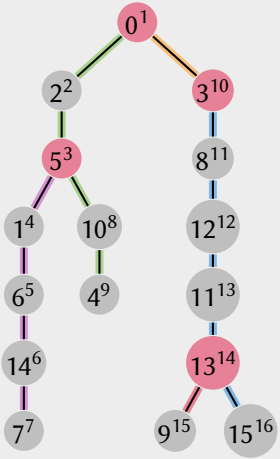
```
int biconexa(grafica g, int u, int visto[], int *orden) {
  visto[u] = ++(*orden);
  int alto = *orden;
  for (int v = 0; v < g.n; v++)
    if (g.a[u][v] == 1 && visto[v] == 0) {
      int temp = biconexa(g, v, visto, orden);
      if (temp < alto)
        alto = temp;
      if (temp >= visto[u])
        articulacion(u); // procesar articulacion
    } else if (g.a[u][v] == 1 && visto[v] < alto)
      alto = visto[v];
  return alto;
}
```

Gráficas

Ejemplo de componentes biconexas

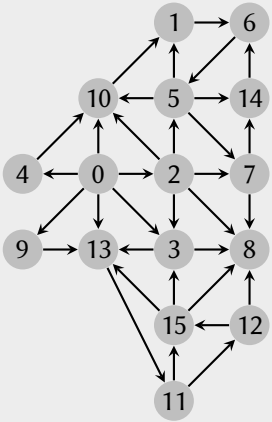


u	visto[u]	alto
0	1	1
1	4	3
2	2	1
3	10	1
4	9	1
5	3	1
6	5	3
7	7	3
8	11	10
9	15	14
10	8	1
11	13	10
12	12	10
13	14	10
14	6	3
15	16	10



Digráficas

Caminos y conexidad fuerte



Camino Secuencia de vértices en la que existe un arco de cada vértice al siguiente.

Camino simple Camino en el que no se repiten vértices.

Digráfica fuertemente conexa Existen caminos de cada vértice a todos los demás.

Componente fuertemente conexa Si una digráfica no es fuertemente conexa, entonces tiene varias componentes fuertemente conexas.

Circuito Camino cerrado (inicio = fin).

Ciclo Circuito en el que no se repiten vértices.

Digráfica acíclica Digráfica que no tiene ciclos.