

# Articulation Points

Andrés Valencia Oliveros<sup>1,2</sup>

*Facultad de Ingeniería, Diseño e Innovación  
Institución Universitaria Politécnico Gran Colombiano  
Bogotá, Colombia*

---

## Resumen

...

*Keywords:* articulation point, cut vertex

---

## 1 Introducción

...

## 2 Teoría de grafos

En matemáticas y en ciencias de la computación, la teoría de grafos estudia las propiedades de los grafos. Un grafo  $G(V, E)$  es una colección de puntos, llamados vértices o nodos  $V = \{v_1, v_2, \dots\}$ , y segmentos de línea que conectan esos puntos, llamados aristas o arcos (en inglés *edges*)  $E = \{e_1, e_2, \dots\}$ ; cada arista  $e$  tiene dos *puntos finales*, que son vértices. Se escribe  $u \overset{e}{-} v$ , y significa que la arista  $e$  incide sobre los vértices  $u$  y  $v$ ; en este caso se puede decir que  $e$  conecta los vértices  $u$  y  $v$ , o que los vértices  $u$  y  $v$  son *adyacentes* [1].

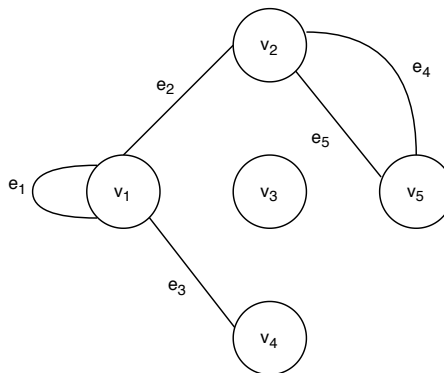


Fig. 1. Ejemplo de un grafo. [1]

---

<sup>1</sup> GitHub: [anvalenciao](#)

<sup>2</sup> Email: [anvalenciao@poligran.edu.co](mailto:anvalenciao@poligran.edu.co)

## 2.1 Grafo conexo

Un grafo  $G$  es conexo, si por cada dos vértices  $u$  y  $v$ , hay un camino (finito) que comienza en  $u$  y termina en  $v$  [1]. Para verificar si un grafo  $G$  es conexo, se puede aplicar un [algoritmo determinista](#) habitual, búsqueda en anchura en inglés *Breadth First Search* (BFS) o búsqueda en profundidad en inglés *Depth First Search* (DFS).

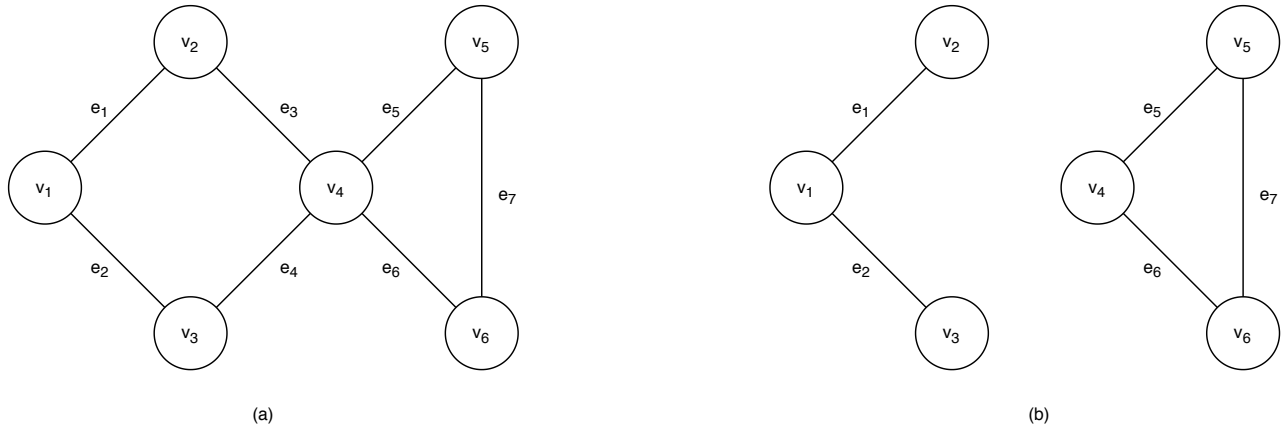


Fig. 2. Tipos de grafos. (a) Conexo. (b) Disconexo.

## 2.2 Grafo dirigido o digrafo

Un digrafo o grafo dirigido  $G(V, E)$  se define de manera similar a un grafo, excepto que el par de [puntos finales](#)  $(u, v)$  de cada arista ahora está ordenado. Se escribe  $u \xrightarrow{e} v$ , dónde  $u$  es el vértice inicial de  $e$ ; y  $v$  es el vértice final de  $e$ . Se dice que la arista  $e$  está dirigida de  $u$  a  $v$  [1].



Fig. 3. Tipos de grafos. (a) No dirigido. (b) Dirigido o digrafo.

## 3 Puntos de articulación

Un vértice  $v$  es un punto de articulación (o vértice de corte), si al eliminar el vértice  $v$  del grafo aumenta el número de componentes conectados. Es decir, genera algunos vértices inalcanzables para otros, se desconecta el grafo.

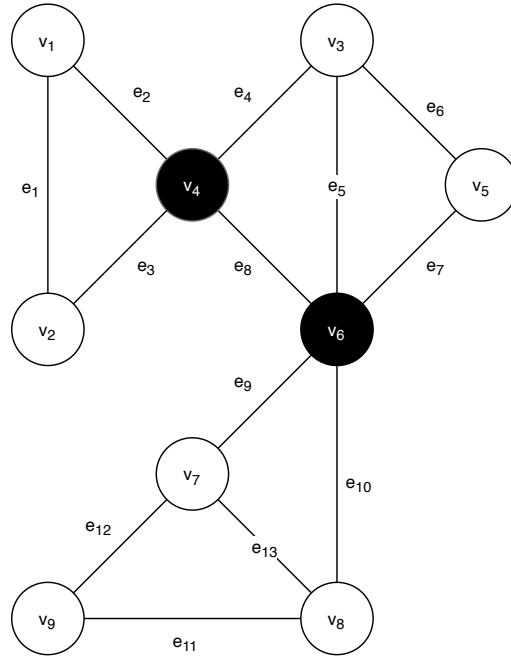


Fig. 4. Ejemplo de grafo con dos puntos de articulación  $v_4$  y  $v_6$ .

## 4 Puentes

Una arista se llama puente si al eliminarla del grafo (manteniendo los vértices) aumenta el número de componentes conectados.

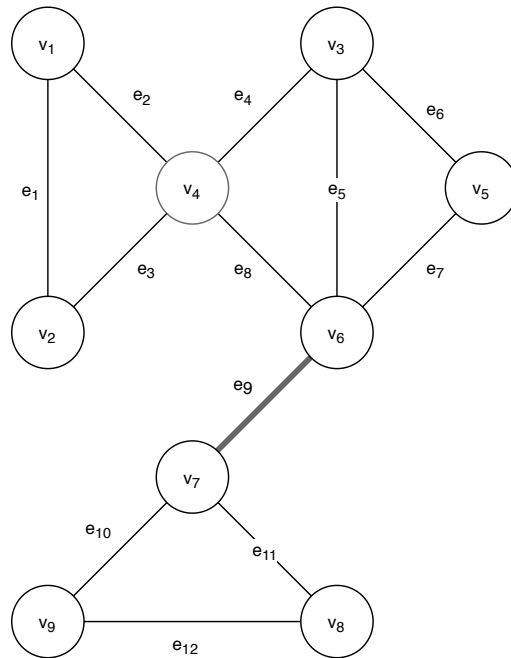


Fig. 5. Ejemplo de grafo con una arista puente  $e_9$ .

## 5 Algoritmos

lorem ipsum dolor sit amet. /home/andres/Documentos/ArticulationPoints/document/Algorithms/code

### 5.1 Algoritmo de Tarjan

El algoritmo de Tarjan para encontrar puntos de articulación

#### 5.1.1 Pseudocódigo

Algoritmo 1: sample code

```
1 // https://medium.com/@ziyoshams/graphs-in-javascript-cc0ed170b156
2 class Graph {
3   constructor(vertices) {
4     this.V = vertices;
5     this.graph = new Object();
6     this.Time = 0;
7   }
8
9   addEdge(u, v) {
10    if (this.graph[u] === undefined) {
11      this.graph[u] = new Array();
12    }
13
14    if (this.graph[v] === undefined) {
15      this.graph[v] = new Array();
16    }
17
18    this.graph[u].push(v);
19    this.graph[v].push(u);
20  }
21
22  APUtil(u, visited, ap, parent, low, disc) {
23    let children = 0;
24    //visited[u] = true;
25
26    //disc[u] = this.Time;
27    //low[u] = this.Time;
28    //this.Time += 1;
29    //console.log(this.graph[2]);
30
31    //for (let v of this.graph[u]) {
32    for (const v of this.graph[u]) {
33      console.log(v);
34    }
35  }
36
37  AP() {
38    var visited = new Array(this.V),
39        disc = new Array(this.V),
40        low = new Array(this.V),
41        parent = new Array(this.V),
42        ap = new Array(this.V);
43  }
44 }
45
46 // g1 = {0: [1], 1: [0, 2], 2: [1, 3], 3: [2]};
47 g1 = new Graph(4);
48 g1.addEdge(0, 1);
49 g1.addEdge(1, 2);
50 g1.addEdge(2, 3);
51
52 /*g1 = new Graph(5)
53 g1.addEdge(1, 0);
54 g1.addEdge(0, 2);
55 g1.addEdge(2, 1);
56 g1.addEdge(0, 3);
57 g1.addEdge(3, 4);*/
58 g1.AP();
59 console.log(g1);
```

### 5.1.2 Complejidad

#### Glosario de términos

**adyacentes** Si una arista conecta dos vértices, se dice que son adyacentes. [1](#)

**algoritmo determinista** Su comportamiento se puede predecir completamente a partir de la entrada, el algoritmo realiza los mismos cálculos y ofrece los mismos resultados[\[2\]](#). [2](#)

**BFS** *Breadth First Search*. [2](#)

**DFS** *Depth First Search*. [2](#)

**puntos finales** Dos vértices conectados por una arista. [1](#), [2](#)

#### Referencias

[1] S. Even, *Graph algorithms*. Cambridge University Press, 2011.

[2] P. E. Black, “deterministic algorithm,” 2009.