Articulation Points

Andrés Valencia Oliveros[1,2]

Facultad de Ingeniería, Diseño e Innovación
Institución Universitaria Politécnico Grancolombiano
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].

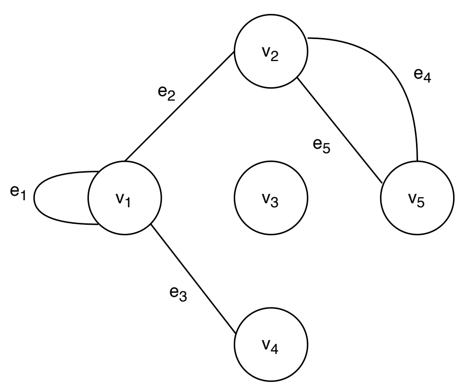

Fig. 1. Ejemplo de un grafo. [1]

[1] GitHub: anvalenciao

[2] Email: 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).

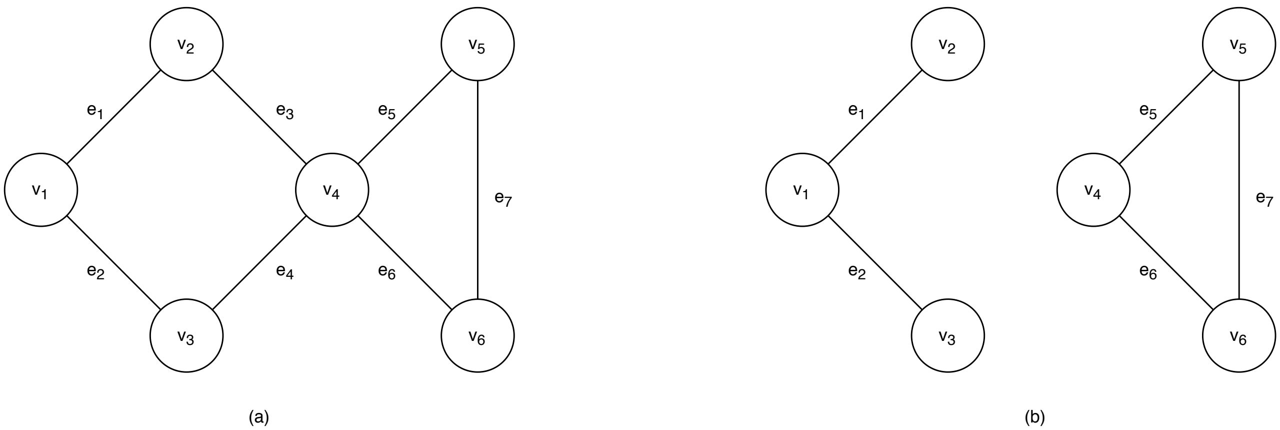

Fig. 2. Tipos de grafos. (a) Conexo. (b) Disconexo.

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].

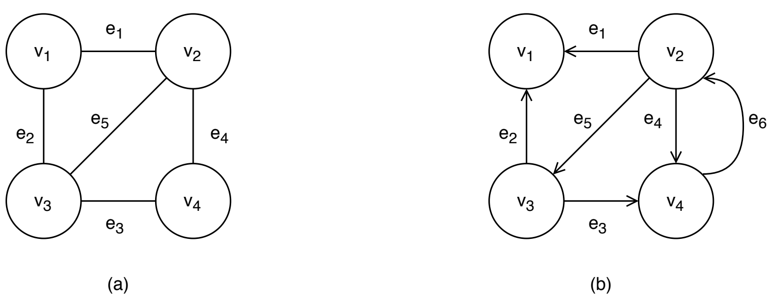

Fig. 3. Tipos de grafos. (a) No dirigido. (b) Dirigido o digrafo.

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.

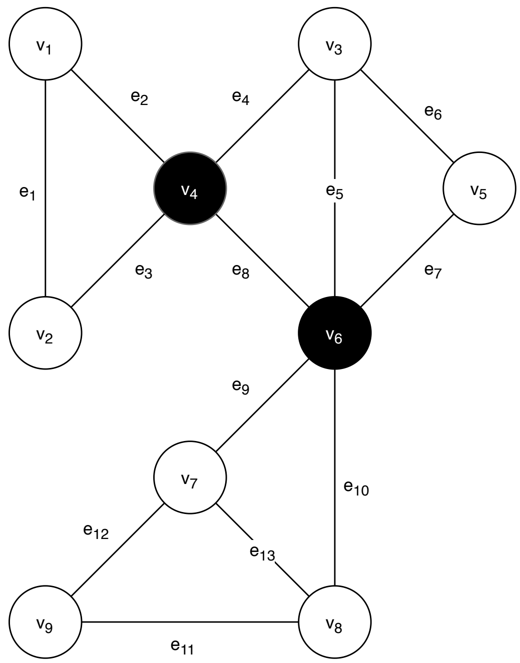

Fig. 4. Ejemplo de grafo con dos puntos de articulación v_4 y v_6.

4 Puentes

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

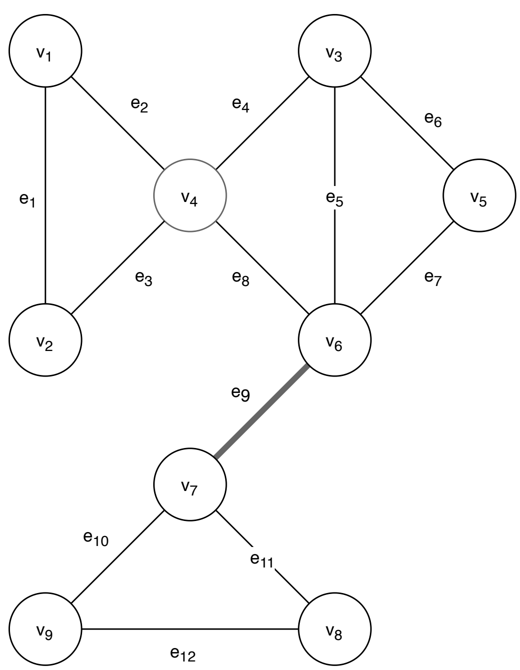

Fig. 5. Ejemplo de grafo con una arista puente e_9.

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

Algorithm 1 Linear time depth first search

function GETARTICULATIONPOINTS(i, d)
    $visited[i] \leftarrow true$
    $depth[i] \leftarrow d$
    $low[i] \leftarrow d$
    $childCount \leftarrow 0$
    $isArticulation \leftarrow false$
    for all ni **in** $adj[i]$ **do**
        if not $visited[ni]$ **then**
            $parent[ni] \leftarrow i$
            GetArticulationPoints($ni, d+1$)
            $childCount \leftarrow childCount + 1$
            if $low[ni] \geq depth[i]$ **then**
                $isArticulation \leftarrow true$
            end if
            $low[i] \leftarrow Min(low[i], low[ni])$
        else if $ni \neq parent[i]$ **then**
            $low[i] \leftarrow Min(low[i], depth[ni])$
        end if
    end for
    if ($parent[i] \neq null$ **and** $isArticulation$) **or** ($parent[i] = null$ **and** $childCount > 1$) **then**
        return Output i as articulation point
    end if
end function

Algoritmo 1: sample code

```
// Test: Ubuntu 19.04 / node v10.15.2
// Command: node /{path}/Tarjan.js
class Graph {
  constructor() {
    this.graph = new Object();
    this.time = 0;
  }

  addEdge(u, v) {
    if (this.graph[u] === undefined) {
      this.graph[u] = new Array();
    }

    if (this.graph[v] === undefined) {
      this.graph[v] = new Array();
    }

    this.graph[u].push(v);
    this.graph[v].push(u);
  }

  DFS(u = 0, visited = [], ap = [], parent = [], low = [], disc = []) {
    let children = 0;
    visited[u] = true;

    disc[u] = this.time;
    low[u] = this.time;
    this.time += 1;
```

```

    for (const v of this.graph[u]) {
      if (visited[v] == false) {
        parent[v] = u;
        children += 1;

        this.DFS(v, visited, ap, parent, low, disc);

        low[u] = Math.min(low[u], low[v]);
        if (parent[u] == -1 && children > 1) {
          ap[u] = true;
        }

        if (parent[u] != -1 && low[v] >= disc[u]) {
          ap[u] = true;
        }

      } else if (v != parent[u]) {
        low[u] = Math.min(low[u], disc[v]);
      }
    }
  }

  ArticulationPoint() {
    let V =  Object.keys(g1.graph).length; // Vertices
    let visited = new Array(V),
        disc =  new Array(V),
        low =  new Array(V),
        parent =  new Array(V),
        ap = new Array(V);

    for (let i = 0; i < V; i++)
    {
        parent[i] = -1;
        visited[i] = false;
        ap[i] = false;
    }

    for (let i = 0; i < V; i++) {
      if (visited[i] == false) {
        this.DFS(i, visited, ap, parent, low, disc);
      }
    }

    for (let i = 0; i < V; i++) {
      if (ap[i] == true) {
        console.log(i + " ");
      }
    }

  }
}

// g1 = {0: [1], 1: [0, 2], 2: [1, 3], 3: [2]};
g1 = new Graph();
g1.addEdge(0, 1);
g1.addEdge(1, 2);
g1.addEdge(2, 3);
g1.ArticulationPoint();
```

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.